ALEXANDER TECHNOLOGICAL EDUCATIONAL INSTITUTE OF THESSALONIKI

DEPARTMENT OF INFORMATION TECHNOLOGY

POSTGRADUATE STUDIES PROGRAM ON

WEB INTELLIGENCE

Content-Based Image Retrieval using Deep Learning

Master's Thesis

PANTELIS I. KAPLANOGLOU

Supervised by: Konstantinos I. Diamantaras
Professor, Department of Information Technology, ATEI of Thessaloniki

Thessaloniki, June 2017

This page is intentionally left blank.

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEB INTELLIGENCE

Ανάκτηση Εικόνων με Βάση το Περιεχόμενο με χρήση Μεθόδων Βαθιάς Μάθησης

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΠΑΝΤΕΛΗ Ι. ΚΑΠΛΑΝΟΓΛΟΥ

Επιβλέπων : Κωνσταντίνος Διαμαντάρας
Καθηγητής, Τμήμα Μηχανικών Πληροφορικής, Α.Τ.Ε.Ι. Θεσσαλονίκης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 20 Ιουνίου 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Κωνσταντίνος Διαμαντάρας Καθηγητής Α.Τ.Ε.Ι.Θ.	Μιχαήλ Σαλαμπάσης Καθηγητής Α.Τ.Ε.Ι.Θ.	Δημοσθένης Σταμάτης Καθηγητής Α.Τ.Ε.Ι.Θ.

Θεσσαλονίκη, Ιούνιος 2017

(Υπογραφή)

.....

Παντελής Ι. Καπλάνογλου

Μηχανικός Πληροφορικής Α.Τ.Ε.Ι.Θ.

Περίληψη

Τα Βαθιά Συνελικτικά Νευρωνικά Δίκτυα (CNNs) έχουν δημιουργήσει νέες προοπτικές για την Υπολογιστική Όραση και πρόσφατα έχουν εφαρμοστεί στην Ανάκτηση Εικόνας με Βάση το Περιεχόμενο (CBIR). Μολαταύτα η εφαρμοσιμότητα τους στον πραγματικό κόσμο για αναγνώριση εικόνας στην Ρομποτική και Ιατρική Απεικόνιση είναι ακόμα ένα ανοικτό θέμα προς διερεύνηση. Επιπρόσθετες βελτιώσεις στην ακρίβεια σε συνδυασμό με μείωση του υπολογιστικού κόστους είναι τα βασικά προβλήματα που πρέπει να αντιμετωπίσουν οι μελλοντικές προτάσεις. Η παρούσα Μεταπτυχιακή διπλωματική εργασία προτείνει μια νέα προσέγγιση εισάγοντας το BioCNN. Το Βιο-εμπνευσμένο Συνελικτικό Νευρωνικό Δίκτυο (Bio-inspired Convolutional Neural Network) είναι ένας καινοτόμο είδος αρχιτεκτονικής που μιμείται το ανθρώπινο οπτικό σύστημα, ξεκινώντας από τους φωτοϋποδοχείς και τους νευρώνες τους αμφιβληστροειδούς, διαμέσου του έξω πλευρικού γονατώδους πυρήνα (LGN) και καταλήγοντας στα κύτταρα V1 του πρωτεύοντος οπτικού φλοιού. Το δίκτυο εκπαιδεύεται για ταξινόμηση με κατάβαση δυναμικού, χρησιμοποιώντας μια νέα τεχνική μεταβλητού ρυθμού μάθησης που λέγεται Κατάβαση Δυναμικού Αυξομειούμενης Ώσης (Throttled Gradient Descent - TGD). Γενικεύοντας τις διάφορες προσεγγίσεις Σάκου Οπτικών Λέξεων (Bag of Visual Words - BoVW) , ο όρος Γλώσσα Οπτικών Χαρακτηριστικών (Visual Features Language - VFL) χρησιμοποιείται για να περιγράψει μια αναπαράσταση εικόνας φτιαγμένη από «οπτικές λέξεις», που εμπεριέχουν τα τοπικά χαρακτηριστικά της. Το σύνολο των περιγραφέων χαρακτηριστικών για τις περιοχές της εικόνας, συναθροίζεται από τις ενεργοποιήσεις ενός συνελικτικού επιπέδου του BioCNN. Κατόπιν συσταδοποιείται σε οπτικές λέξεις μέσω του Ριζωματικού k-Means (RkMeans), μιας παραλλαγής του Ιεραρχικού k-Means (HkMeans) με μεταβλητό παράγοντα διακλάδωσης. Η σύντηξη οπτικών χαρακτηριστικών υλοποιείται με συνάθροιση των διαφορετικών VFL σε μια ενοποιημένη αναπαράσταση της εικόνας, που ευρετηριοποιείται από μια μηχανή αναζήτησης κειμένου. Ο σκοπός αυτής της διπλωματικής είναι να εξερευνήσει το ερευνητικό πεδίο και να διεξάγει μια αρχική μελέτη χρησιμοποιώντας την αλληλουχία BioCNN-CBIR. Κατά την διάρκεια αυτής, η αρχιτεκτονική BioCNN έδειξε υποσχόμενα αποτελέσματα φτάνοντας σε παρόμοια επίδοση με νικήτριες αρχιτεκτονικές του ImageNet στο CIFAR10, χρησιμοποιώντας μια τάξη μεγέθους λιγότερες παραμέτρους δικτύου από αυτές.

Λέξεις Κλειδιά: Μηχανική Μάθηση, Ανάκτηση Πληροφορίας, Βαθιά Εκμάθηση, Συνελικτικά Νευρωνικά Δίκτυα, BioCNN, Γλώσσα Οπτικών Χαρακτηριστικών, Αναγνώριση Εικόνας

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

Deep Convolutional Neural Networks (CNNs) have created new perspectives for Computer Vision and have recently been applied for Content-Based Image Retrieval (CBIR). Nevertheless their applicability for real-world image recognition in Robotics and Medical Imaging is still a subject open to research. Additional improvements in accuracy combined with reduction in computational costs are key issues that future proposals need to address. This Master's thesis proposes a new approach, by introducing the BioCNN. The Bio-inspired Convolutional Neural Network is a novel kind of architecture that imitates the human visual system starting from the photoreceptors and the neurons in the retina, through the lateral geniculate nucleus (LGN) and ending with the V1 cells of the primary visual cortex. The network is trained for classification with gradient descent, using a new technique of variable learning rate called Throttled Gradient Descent (TGD). Making a generalization of the various Bag of Visual Words (BoVW) approaches, the term Visual Features Language (VFL) is used to describe an image representation made of visual words that encapsulate its local features. The set of feature descriptors for the image regions is assembled from the activations of a BioCNN convolutional layer. Then it is clustered into visual words using the Rooting k-Means (RkMeans) algorithm, a variation of Hierarchical k-Means (HkMeans) with variable branching factor. Visual feature fusion is implemented by assembling the different VFLs into a single textual image representation that is indexed by a text search engine. The aim of this thesis is to explore this research field and conduct an initial study using the BioCNN-CBIR pipeline. During this the BioCNN architecture showed promising results in CIFAR10, by achieving similar performance with ImageNet winning architectures using an order of magnitude less network parameters.

Keywords: Deep Learning; Content-Based Image Retrieval; Bio-Inspired Convolutional Neural Network; BioCNN; Visual Features Language; Image Recognition

I would like to dedicate this Master's thesis to my wife Stella, especially for her daily efforts, the patience and care for me. My postgraduate studies would not have become possible without any of these.

To my parents Ioannis and Kyriaki, for giving me Paedeia which shaped me in the person I have become.

To my godmother Gianna Michailidou D.D.S., for believing and supporting this effort and for her more valuable asset that she spends with me, time.

Last but not least, to my supervising professor Dr. Konstantinos I. Diamantaras for his valuable guidance, which handed me a set of keys for opening doors to the exciting world of Machine Learning.

Contents

1 Introduction 1

1.1 Image Recognition and Retrieval 1
1.2 Deep Learning 2
1.3 Biological Inspiration and the Vision for AI 3
1.4 Contributions of the Thesis 4
1.5 Outline of the thesis 5

2 Image Recognition with Deep Learning 7

2.1 Introduction 7
2.2 The Image Recognition Problem Setting 7
2.2.1 Digital Imaging and Digital Image Representation 7
2.2.2 Image Recognition and Image Retrieval 9
2.2.3 Properties and Quality Factors of an Image Recognition System 10
2.3 Deep Learning 12
2.3.1 Diversity of Deep Networks, Groups and Meta-Groups 12
2.3.2 A Short History of Deep Learning for Image Recognition 16
2.4 CNN for Image Classification 17
2.4.1 Convolution Operation on Color Images 17
2.4.2 Convolutional Layers 20
2.4.3 Pooling Layers 22
2.4.4 Activation Functions 25
2.4.5 Pre-training a CNN for Classification 27
2.5 State-Of-The-Art CNN architectures 35
2.5.1 Advances in Computer Vision through ILSVRC 35
2.5.2 The CNN architecture evolution through ILSVRC 36
2.6 Future Research Directions for CNNs 47
2.6.1 Evaluation of the Current State-of-the-Art 47
2.6.2 Open Research Subjects for Deep Convolutional Neural Networks 48
2.6.3 Alternative Directions 49

3 Describing Images in a Language of Visual Features 51

3.1 Introduction 51
3.2 Content-based Image Retrieval using a Visual Features Language 52
3.2.1 Image Retrieval based on Visual Features 52
3.2.2 Text Retrieval and the Bag-of-Words 53

3.2.3 Vector Space Model for Information Retrieval 54
3.2.4 Image Retrieval using Bag-of-Visual-Words 56
3.2.5 The Visual Features Language (VFL) 56
3.2.6 Handcrafted Feature Extractors 57
3.2.7 Creation of the Visual Vocabulary 59
3.3 Using Convolutional Neural Networks for CBIR 61
3.3.1 The Deep Neural Content-based Image Retrieval Pipeline 61
3.3.2 DNN Feature Extractors before ILSVRC2012 63
3.3.3 Review of CNN-based CBIR 63
3.3.4 Open Research Subjects for CNN-CBIR 65
3.4 Bio-inspired Models for Universal Visual Features 67
3.4.1 Intuition from ZFNet Visualization 67
3.4.2 Inspiration from the Human Visual System 68
3.4.3 Foundations of the Bio-inspired Convolutional Neural Network 71
3.4.4 Models Inspired by the Human Visual System 72

4 The Bio-inspired Convolutional Neural Network (BioCNN) 75

4.1 Introduction 75
4.2 Overview of the Bio-Inspired CNN 76
4.2.1 Inspiration 76
4.2.2 Motivation 76
4.2.3 Hypotheses 77
4.2.4 Architectural Overview 79
4.3 Artificial Retinal Neural Network (ARNN) 83
4.3.1 Color Space Transformation with the Photoreceptors Abstraction 83
4.3.2 Modeling the Outer Plexiform Layer 85
4.3.3 Modeling the Inner Plexiform Layer 92
4.4 Artificial Primary Visual Cortex (APVC) 103
4.4.1 Lateral Geniculate Nucleus 103
4.4.2 Gabor V1 Simple Cells 104
4.4.3 Characteristics of BioCNN stem output 107
4.4.4 Higher Visual Cortex Convolutional Layers 109

5 Creating a VFL Vocabulary from BioCNN Features 111

5.1 Introduction 111
5.2 Convolutional Layer Activations as Features 111
5.2.1 Semantics of Neural Activations 111
5.2.2 Assemblage of the Descriptor Set 113
5.3 High Dimensionality Clustering on Large Descriptor Sets 114

5.3.1 Hierarchical k-Means 114
5.3.2 Rooting k-Means 115
5.3.3 Visual Word Assignment 116
5.3.4 Extensibility into a Visual Features Language Grammar 117
5.4 Indexing the Image as a Visual Document 119
5.4.1 Composing a Visual Document 119
5.4.2 Using text for the Visual Document image representation 120
5.4.3 Indexing an Image Collection 121

6 BioCNN-CBIR Evaluation 123

6.1 Introduction 123
6.2 Metrics for evaluating CNN-CBIR models 124
6.2.1 The Complexity of Evaluating CNN-CBIR 124
6.2.2 Supervised Classification Metrics 125
6.2.3 Custom Metrics for Monitoring the Training Process 128
6.2.4 Metrics for the Creation of a Visual Features Language 130
6.2.5 Information Retrieval Metrics 131
6.3 Evaluation System for Experiments 133
6.3.1 Training, Validation and Test Sets 133
6.3.2 10-fold Cross Validation 133
6.3.3 Gradient Descent Learning 134
6.3.4 Minibatches, Sample Shuffling and Subfolds 138
6.3.5 Visual Features Language Setup 138
6.3.6 Image Retrieval Environment 139
6.4 Image Datasets 141
6.4.1 CIFAR10 and CIFAR100 dataset 141
6.4.2 Caltech 101 dataset 141
6.4.3 The Caltech101-2017 randomly chosen image collection 142
6.4.4 The HIQ-1 Handpicked Images Set 144
6.5 Experiments Setup 145
6.5.1 Classification Training Process 145
6.5.2 Nominal TGD Gradient Training Hyperparameters 145
6.5.3 AlexNet / ZFNet architecture reproduction 146
6.5.4 BioCNN Architectures for CIFAR10 147
6.5.5 BioCNN Architectures for Caltech101-2017 148
6.6 Image Classification Results 150
6.6.1 Classification Results on CIFAR10 150
6.6.2 Classification Results on Caltech101 153
6.7 VFL Creation and Image Retrieval 157

6.7.1 VFL Creation Experiments 157
6.7.2 Image Retrieval Experiments 160
6.8 Discussion 166
6.8.1 Best Performance on CIFAR10 Using a Fraction of Parameters 166
6.8.2 Confirming the Transferability of BioCNN Features. 166
6.8.3 Competitive Performance on Caltech101-2017 167
6.8.4 Feature Quality of Sub-Optimal Classifiers 168
6.8.5 Benefits from the Visual Features Language Concept 168
6.8.6 Understanding of the CNN-CBIR complexity 169

7 Software Engineering Details 171

7.1 Three Tier Software Architecture 171
7.2 Machine Learning R&D Environment 172
7.2.1 Accelerated Computing with CUDA 172
7.2.2 Google Tensorflow, Python 173
7.2.3 SciKit-Learn, MatPlotLib 175
7.2.4 Terrier IR Search Engine 176
7.2.5 TALOS Framework 177
7.2.6 Data Memory Tier 178

8 Conclusion 179

8.1 Summary 179
8.1.1 Introducing the BioCNN Deep Neural Network 179
8.1.2 Content-Based Image Retrieval with the Visual Features Language 180
8.1.3 Software Implementation of a Web 4.0 Intelligence Tier 181
8.2 Future Perspectives 182

9 Bibliography 183

1 Introduction

1.1 Image Recognition and Retrieval

The field of Computer Vision and its vast extend, presents great challenges for research, but shows in parallel great potential for real-world applications and innovations. In the domain of image recognition, research effort focuses on finding models to detect and identify objects that appear in digital still images or frames of video sequences. The classification of real-world objects given a picture of them is a common image recognition task that usually includes supervised learning of the models, during which multiple images are used as the representing samples of a given class. A more challenging task is to search inside a collection of images for occurrences of an object, given only the picture of the object or parts of its picture. A set of corresponding images will be retrieved and returned by the system, regardless of their labels and based on visual relevance.

That is directly equivalent to searching into a collection of text documents using a combination of keywords or an entire sentence, in order to retrieve relevant documents. This analogy can bring the image recognition problem into the domain of Information Retrieval (IR). Content-based image retrieval (CBIR) can be defined as an image recognition task, during which the detection of key visual elements inside a collection of pictures returns a set of visually relevant images. Existing methodologies for text-based retrieval can be utilized on visual information, provided that we can create a language to represent it. In a written language the characters representing vowels and consonants and the diacritical marks which indicate breathing and vocal shaping, can be all considered as values for features of the spoken language. In equivalence, the visual features that might refer to color, texture or shape can form visual words that describe a region of a picture, used in a visual document which describes the whole picture. A simplified breakdown of this problem would be: a) Extract the proper visual features from an image b) Create a language that will describe them c) Index the collection of images exactly like a collection of text documents.

1.2 Deep Learning

Neural networks have a long history starting back in the 1940s, with the cornerstone paper of McCulloch and Pitts and the works of Wiener, Neumann and Hebb. From this early stage and on, there had been golden periods followed by times of standstill [1]. The research progress of the 1980s and early 1990s has stumbled upon the difficulties of training complex network structures using gradient descent methods, partially caused by the limitations of the technology at the time. It was not until the last 10 years, that new revolutionary methods combined with advances in hardware design which made massive parallelism [2] broadly available, have provided the capability to train networks with an increased number layers. A third factor that can be mentioned is the transition of the web into the social era, also known as web 2.0 that brought motivation for commercial applications of Artificial Intelligence (AI).

The term Deep Neural Network (DNN) was coined to describe any network that has more than the three layers found in the reference architecture of a Multilayer Perceptron (MLP), which are input, hidden and output. Correspondingly, Deep Learning (DL) is the domain that researches the methodologies and techniques used in Deep Neural Networks, seeking solutions that will be based on them. Isolating through a taxonomy of proposals, the prominent DNN groups are Deep Belief Networks (DBNs), Deep Autoencoders (DAs), Deep Recurrent Neural Networks (DRNNs), Deep Convolutional Neural Networks (DCNNs or CNNs) and some meta-groups like Network-In-Network and the recently introduced Generative Adversarial Networks.

The potential of Deep Learning for solving complex problems, was shown in various image recognition tasks. In the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC2012), the top performing solution for the classification of images to 1000 categories, was a DCNN outperforming the second best by almost 10%. The AlexNet, as it is known from the name of the primary author of the corresponding paper [3], showed that Deep Neural Networks can be a powerful choice for image recognition tasks. It also shifted the focus of research to the original bio-inspired idea of receptive field mapping, incepted years ago as a solution to the T-C problem [4] and later elaborated as the first CNN networks [5]. The last years, key players in the computer industry like Microsoft and Google have demonstrated their interest, making significant advances that led to deeper and better performing networks. The results on image classification are impressive, surpassing the measured human performance on the natural image dataset that is used in ILSVRC.

Models which are pre-trained on this image collection have shown that they can achieve top performance on different image collections. Following the method of transfer learning, the learned parameters of the CNN layers are fine-tuned on new datasets which may

belong to entirely different domains of problems. There is an intuition that the functionality of the convolutional layers resembles that of a general purpose feature extractor, which can be used for CBIR. The visual features are extracted from images simply by recalling them through a trained deep CNN. The activation values of a convolutional layer can be considered as fine, local or global visual features. For each coordinate of the layer's output map, activation vectors are the local feature descriptors for the underlying mapped region of pixels. These can be used to create corresponding "visual words" in a language specific for the architecture and the learned state of the CNN.

This intra-domain approach involves both Deep Learning and Information Retrieval and creates new research subjects: a) Designing and training of CNNs as good feature extractors. b) Selection of neural network layers that have the proper features for better retrieval performance. c) Mitigation of the high dimensionality of neural layer activations for computationally effective representations and indexing of large image collections.

1.3 Biological Inspiration and the Vision for AI

The success of the CNNs and their bio-inspired origin creates an intuition that solutions to image recognition and image retrieval problems can be found by imitating biological models. The human visual system itself is a solution found in the natural world, so effective that helped us not only to have a complex visual perception of the world, but also to build some of our non-primitive emotions like aesthetic quality. Our perception includes a constant task of retrieval based on the properties of objects that pass through our field of vision. Initially our eyes sense the world as light stimuli captured by the retina, processed and transmitted to the visual cortex. From there several additional layers of neural processing create a set of features for the real-world scene that are finally stored in our visual memory.

A compendium of knowledge has already been accumulated by Visual Neuroscience with detailed studies on key points of interest. Its theories are supported by a large amount of experimental observations that have created corresponding models of Computational Neuroscience. Nevertheless, there are still various aspects of our visual system that little is known for them and are a subject of ongoing research. The approach for this Master's thesis, includes a study of the biological cells found in the human visual system, in order to gain simple but useful knowledge. This will be used to design DNNs and functions of artificial neurons, with the target of better visual feature extraction.

If we consider Artificial Intelligence (AI) as the ultimate problem, extracting features inside still images is a small fraction of the greater intelligent vision sub-problem. New models should be simple but also adequate enough to support this complex perception

mechanism. In human biology, the eyes have a starting functionality simple enough to be available at the very first minutes of birth, while in other mammals with more heightened vision like cats, it becomes available after some days. Human vision needs only a few months of adaptation to become complex enough to support human intelligence. After these thoughts, the need of an interdisciplinary approach to solve computer vision is considered, because useful information may exists outside of the context of Computer Science research. Experts in Human Sciences might be needed to reach the goal of creating intelligent machines.

1.4 Contributions of the Thesis

This Master's thesis belongs to the interdisciplinary field of Web Intelligence (WI) which combines knowledge primarily from Web Science and Artificial Intelligence and uses it for related research and development. WI also seeks to create the next generation of AI-empowered web innovations that can be considered as *web 4.0*. For this thesis Deep Learning (DL), Data Mining (DM) and Information Retrieval (IR) are combined to build a prototype of a CBIR system. It can be considered to contribute to WI by illustrating the connections between its various topics of interest and their combined power in a CBIR model that could be in the core of an intelligent image search engine.

To accomplish its targets it draws inspiration from nature using basic knowledge about biological visual systems, which is combined with state-of-the-art Deep Learning methodology. Following the intuition that open problems in the field of Computer Vision may have been optimally solved in nature through evolution, this thesis attempts to leverage observations and theories of Visual Neuroscience for creating a Deep Convolutional Neural Network (CNN). This approach shows the prospects of an interdisciplinary effort for solving related problems.

The thesis introduces a new kind of CNN network the *Bio-Inspired Convolutional Neural Network (BioCNN)* with the main characteristic of a hybrid parameter set, consisting of both pre-defined and learnable parameters. The first layers are modeled after the human retina and the primary visual cortex (V1) in order to act as fine and primitive feature extractors, while the deeper layers are trainable convolutional layers. According to the author's best knowledge, this thesis is the first to propose a hybrid CNN which at the same time engineers the complete stream of biological visual processing from photoreceptor cells to V1 cells into its design. In order to model peak sensitivities to specific colors, a *Gaussian Convolutional Layer* is introduced. Furthermore to train the CNN the method of *Throttled Gradient Descent* is used to regulate a variable learning rate by inspecting the validation results at the end of each epochs. Moreover classification experiments using BioCNN

architectures show a prospect of using an order of magnitude less trainable parameters to achieve competitive performance, by surpassing the winning architecture of ILSVRC2013 for the classification task on the CIFAR10 dataset. The main contribution of this thesis, could be considered as giving guidelines for creating other BioCNN architectures in future research efforts.

The *BioCNN-CBIR* model tries to address several open issues of using CNNs in CBIR for the purpose of including them in real-word mobile and autonomous applications for the Internet of Things (IoT). The quality of the extracted features in the current state-of-the-art CNNs is a subject of research that has just been begun. The applicability of transfer learning to non-natural image domains like medical imaging has been recently examined. Higher image resolution, lower computational cost, smaller memory footprint and no dependency upon a computational cloud are some functional requirements for real-world IoT applications. Rich and invariant features that may work regardless of the image dataset are qualitative requirements. If universally applicable visual features could be discovered, the need for fine-tuning the models on different image domains would be eliminated.

During this work a framework for the systematic study of Deep Learning CBIR (DL-CBIR) is organized under the concept of a *Visual Features Language (VFL)*, a text representation that will describe the contents of images in a format directly usable by text IR engines. A Visual Document (VDOC) can be an image representation in VFL and a Visual Booklet (VBOOK) an assembly of VDOCs, fusing features from different spatial resolutions and thus different visual information semantics. For creating the VFL a variation of the Hierarchical k-Means (HkMeans) clustering algorithm is used, that is named *Rooting k-Means (RkMeans)* and has a variable branching factor.

Furthermore the *TALOS* software framework was implemented for the needs of this thesis, which is built on the Tensorflow computational framework that facilitates hardware accelerated machine learning. It abstracts the nodes of a computational graph using objects and classes, simplifying the development and training of DNNs without requiring detailed knowledge of the node-based implementation and the parallel-distributed training process.

1.5 Outline of the thesis

This remaining chapters are organized as follows: In Chapter 2 there is a description of the image recognition problem, a reference to basic Deep Learning concepts and their applications for image recognition tasks. It includes a detailed study of Deep Convolutional Neural Networks as the primary tool for this work and a review of state-of-the-art CNN architectures identifying possible research subjects for further improvement.

Chapter 3 explains how the text retrieval methodology from the Information Retrieval domain is applicable to Content-Based Image Retrieval in addition to common feature extraction methods, which existed before Deep Learning. Also it introduces a new term in the image recognition context, the Visual Features Language (VFL) that is coined to disambiguate the various Bag-of-Words (BoW) models. Moreover there is a review for identifying existing solutions of CNN-based CBIR. Furthermore it describes the root concepts of a bio-inspired approach in modeling novel CNN networks, with an accompanying review of historic and concurrent with this thesis bio-inspired proposals.

Chapter 4 presents the novel Bio-inspired Convolutional Neural Network, or BioCNN, with an overview of the idea the concepts it introduces and a reference architecture. It includes the full details of the various neural network layers that correspond one-by-one to the early stages of the human visual pathway, namely cone and rod photoreceptors, the layers of the retina, the lateral geniculate nucleus (LGN) and the V1 visual cortex. The mathematical expressions in Chapter 4 are supplemented with graphs and corresponding visualizations of the convolutional layer internals.

Chapter 5 describes the methods from Data Mining and Information Retrieval that are used in the complete BioCNN-CBIR pipeline. It also describes the Rooting k-Means (RkMeans) variation of the Hierarchical k-Means clustering algorithm and its application inside the context of the VFL.

In Chapter 6 there is a reference to the metrics needed for the evaluation of experiments, before the detailed description of the environment and processes of the experiments. The setup of choices and hyperparameters used in specific experiments is presented before the image classification and image retrieval results. At the end of the chapter the experimental findings are discussed.

Chapter 7 contains details about hardware and software that has been utilized for the implementation needs of the CBIR pipeline, from the perspective of Software Engineering. It presents the machine learning development environment that has been organized for the implementation needs, which includes tools like the Google Tensorflow computation framework. The basic concepts of the TALOS framework that has been developed to hide the implementation complexity and speed up research on Deep Neural Networks, are described in this chapter.

Ending in Chapter 8, which summarizes the Master's thesis and concludes its findings, the future perspectives are discussed along with ideas for improvements.

2 Image Recognition with Deep Learning

2.1 Introduction

To understand the benefits of Deep Learning (DL) for Content-Based Image Retrieval (CBIR), we must first state the general problems of image recognition and explore the variety of Deep Learning (DL) methodologies, in order to identify the best suitable neural network architectures. Starting at section 2.2, this Chapter describes the process of sampling digital images of physical scenes at the lowest levels and mentions the various tasks that are related to image recognition. Additionally, the needed quality aspects of image recognition are presented as the targets for an ideal computer vision system. Section 2.3 contains a baseline reference of Deep Learning methodologies and section 2.4 a comprehensive study of the selected choice for this thesis, the Convolutional Neural Network (CNN). Section 2.5 makes a synopsis of the most important and state-of-the-art CNN architectures in a timeline of improvements, which have been achieved through the ILSVRC competition. Section 2.6 discusses issues in the current state-of-art, identifying potential needs for future research.

2.2 The Image Recognition Problem Setting

2.2.1 Digital Imaging and Digital Image Representation

Digital image sensors can be distinguished into two common types, Charged-Coupled Devices (CCD) that had been massively used in the past and the Complementary Metal-Oxide Semiconductors (CMOS) that are now widely used for high-definition imaging. The primary functionality of a CMOS is to capture light that hits its surface in various intensities and

convert them into an array of electrical signals [6]. At the end of its processing pipeline the system splits the wavelength of the incoming light into three values of the chromatic components red (R), green (G), blue (B) of the RGB color system [7]. The most usual solution for this is to filter the light before it hits the sensor through a color filter array (CFA), which is a mosaic of red, green and blue *subpixels*. Each colored subpixel is placed over a photodiode that measures only the respective color component. The RGB values for each pixel are calculated through *demosaicking*, a spatial interpolation that takes into account the distribution of the RGB colors in the filter. Typically a Bayer pattern CFA is used, that has a 2×2 pattern of one red two green and one blue squares [6].

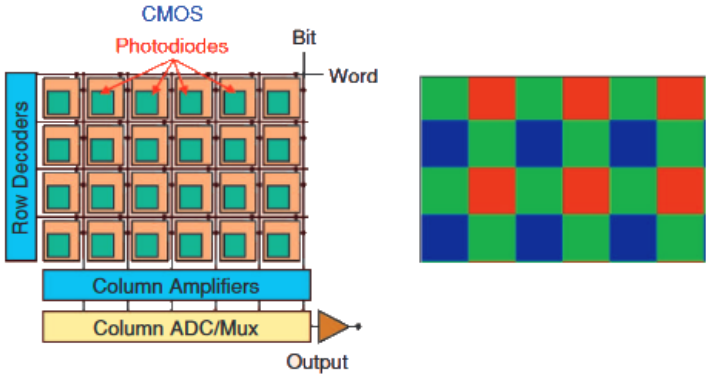

Figure 1: Hardware architecture of a CMOS sensor and a corresponding Bayer pattern CFA [6]

The resulting digital format is stored in memory and has the R, G and B values for each pixel. Depending on the color depth of the digital image format, different bits per pixel are used to store these values. The past decades the 8-bit capacity, that could support only 3 bits for R and G channels (8 intensities) and 2 bits for the B channel (4 intensities), has been replaced by the de-facto standard of 24-bit, known as *true color*. Having 8 bits per color each value can represent 256 intensities of red, green and blue and a total of 16.8 million colors. More accurate representations have already been proposed under the term *deep color*, having color depths of 30, 36, 48 and 64 bits [8], increasing the granularity of color representation. The input images used in color image recognition models are mainly 24-bit true color RGB images

2.2.2 Image Recognition and Image Retrieval

The function of image recognition systems can be summarized as deriving visual information from the raw pixel data of digital images, in order to establish knowledge on a depiction of the physical world. This depiction can be a natural photograph, an image resulting from a medical scan, a snapshot of the cosmos taken from a telescope and other images from various optical or radiographic instruments.

Image recognition includes various tasks such as, classifying pictures or scenes to classes or concepts, detecting single or multiple presences of objects into them and additionally determining the position of each occurrence [9]. During supervised learning of an image recognition model, prior knowledge is used so that the model can adjust its parameters, capturing internally the archetypes of visual features. An image collection or dataset has predefined categories with multiple image samples linked to each one of them. The supervised learning of a model for a classification task is using these relations as ground truth knowledge for visually similar images.

If we think that the discriminative power of the classifier is based on the visual similarities of images that belong to the same parent class, these visual features could be used as the basis for other tasks. A system could exploit relations between objects and their features, to retrieve a relevant image from a stored collection of images. An image query of an unknown to the system object or concept can retrieve images that relate visually, and potentially implement a belong-to, part-of, similar-to and other relationships.

The task of image retrieval serves this purpose. Given a collection of digital images and a query consisting of an image or parts of it, it retrieves a set of relevant images by measuring their visual similarity. When using only the content of the images, without any additional text annotation, the image retrieval task is defined as content-based image retrieval. The information system that implements the CBIR task is an image search engine, which operates on indexes of distributedly stored image data, a schema that is both scalable and efficient. Adding images to the index does not include retraining the model, thus extending the image collection in a CBIR system has insignificant computational costs.

The image retrieval can be considered a superset of image recognition tasks. If the features of an objects are found in an image and it is retrieved in the top rank, the retrieval task is additionally an object detection task. The proper design of the image retrieval model can take into account the position of the visual features inside the image. Image retrieval with regard to the position of the visual information can be similar to object localization.

2.2.3 Properties and Quality Factors of an Image Recognition System

Visual similarity can be based in various factors, and an effective image recognition system should encompass several aspects of the visual information that exists in raw image data. These must be handled by the system's capabilities, each one contributing to the overall quality of image recognition. The first capability is color distinction ignoring minor variations like exposure to bright light or lack of illumination. Detecting contrasting colors is a fundamental quality for fine-grained features like points or edge primitives. Texture is a visual property that discriminates objects of the same shape, for example animals that are identified from characteristic color patterns in their skin and motifs in their fur. Also the roughness of a 3D surface can be identified though the coarseness of texture in its 2D projection. When fine points are connected they form primitive lines or curves of specific orientation and thickness. Directionality, length and width are properties that in some cases indicate a difference between objects. On the opposite, features of the same object should remain invariant of geometric transformations like resizing, rotating and mirroring. Primitive shapes can be derived by the directionality and length of primitive lines. A simple example is when an object recognition derives a triangle from three lines, which by two share common endpoints. Complex shape detection is a quality requirement for a system that can combine primitive shapes. For example combining a semi-circle, followed by a rectangle, followed by a semi-circle creates the complex *stadium shape* [10].

Figure 2. Stadium shape as a composition of basic shapes of the same or different colors. Conceptual set arithmetic operations are used to express the relations between the shapes

Higher functionalities of an image recognition system would use the visual information that is extracted from the image data, in order to constitute knowledge that is useful inside a broader intelligent system. These include understanding of analogies and symmetries of objects, detection of light-sources and removal of shadows, ignoring background clutter and having tolerance to object occlusion. Point symmetries like rotation and reflection [11], refer to the capability of detecting the same object when is it simply rotated around a center or when it is mirrored. Identification of a regular disposition of objects, overlapping tiles of objects, geometric self-similarity, centered symmetries and all the possible combination of them would be qualities of an ideal image recognition system [11]. Light-source detection is needed for understanding a scene and is useful for practical applications in real-world

systems, when the detection of a light source activates an appropriate actuator [12]. The position of the light-source is the cause of shadows, whose removal might be beneficial for real world scenes [13].

The most challenging functionality of the system is the ability to recognize an object when it is partially occluded by another, keeping features near occlusion edges or rejecting them when they are part of background or clutter [14]. An equally difficult task is to identify the deformations in the transmitted light of an object that are caused by a transparent surface and detect both the lens and the object. All the aforementioned quality requirements are only the basis for the 3D comprehension of a scene by an intelligent vision system, which will possess higher qualities like depth perception and detection of 3D symmetries. This will empower applications of intelligent agents in the physical world, which must fundamentally detect and avoid obstacles in order to navigate in it. Inspiration from nature can be an approach to solve these difficult problems [15].

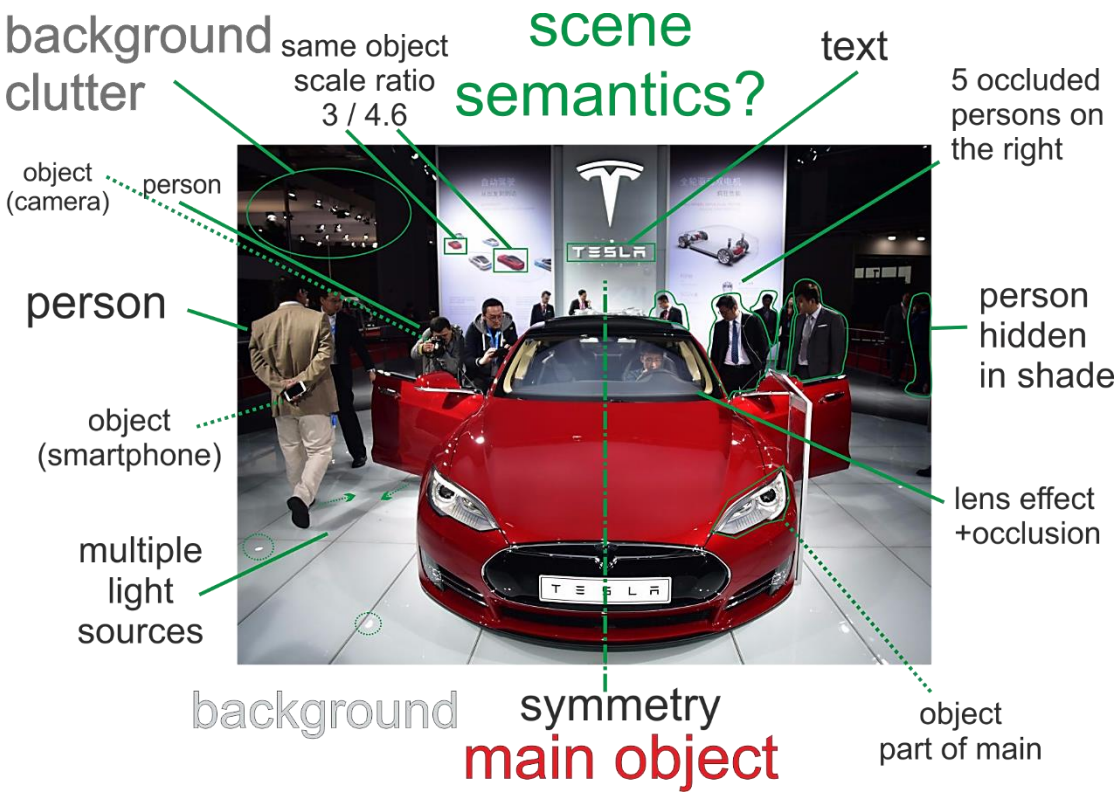

Figure 3: Scene comprehension qualities of an ideal computer vision system operating on a photograph of an exhibited Tesla Model S (photograph by Johannes Eisele/AFP/Getty Images [16])

2.3 Deep Learning

2.3.1 Diversity of Deep Networks, Groups and Meta-Groups

The depth of a multilayer Artificial Neural Network (ANN) architecture can be simply described as the number of layers which hold the model's parameters. The reference architecture of a fully-connected Multilayer Perceptron (MLP) neural network has an input, hidden and output layer. The parameters in MLP can only be found at the last two layers. A network can be characterized as deep when there are more than three layers in the architecture of the model. In the domain of image recognition a plethora of proposals and solutions that use Deep Neural Networks (DNN) already exists. An outline of some basic DNN types is potentially useful before examining their use in image recognition and CBIR tasks. The general concepts of these types can taxonomize networks to groups or meta-groups that may span over different categorizations.

The Deep Belief Networks (DBNs) are modeled with layers of stochastic neurons, each one expressing probabilities of correlations between incoming and outgoing values. These probabilistic weights express an association between input characteristics and the derived output features, thus each neuron unit can be considered a feature detector [17]. A DBN can be trained without supervision using an abstraction, according to which a DBN is considered a stack of Restricted Boltzmann Machines (RBMs) [18].

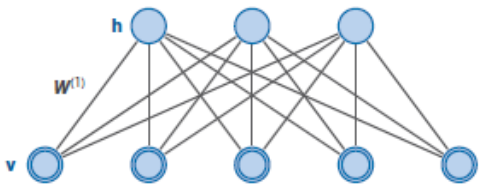

Figure 4: Restricted Boltzmann Machine (RBM) [19]

Each RBM is composed of two consecutive DBN layers and the target of an unsupervised learning process is the minimization of the total energy for each machine using Gibbs sampling [20] [21], a Monte Carlo Markov Chain (MCMC) technique [22]. A practical use of a pre-trained DBN is that the learned weights can be transferred to the same fully-connected MLP structure as good initial values for a supervised training process. This has been experimentally proven to be beneficial for gradient descent methods like back-propagation, helping to converge faster and to a better solution. An intuitive description for the internal functionality of the DBNs for image recognition, is that they can “fantasize” in their output response what was probably the given image. This allows a reconstruction of the input given the output activations, which can categorize DBNs as generative models [19].

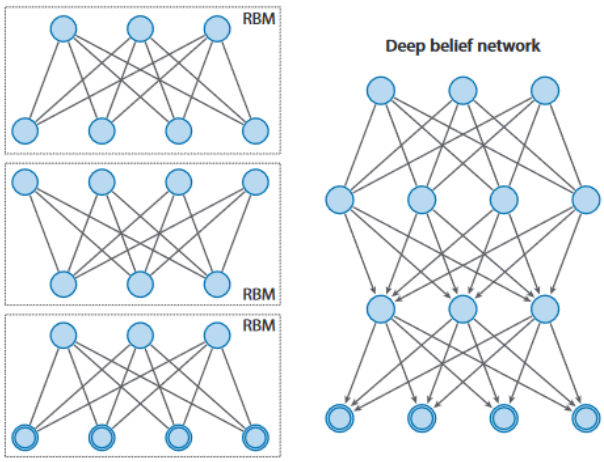

Figure 5: Stacked RBM abstraction of a DBN [19]

Another type of deep network, that has a long-standing background, is the Deep Autoencoder (DA) [23] [24], that is also trained using unsupervised layer-wise training [25]. Their functionality can be simply described as copying their input to their output but at the same time making the output more representative. They can reduce the dimensionality of data, remove noise and filter out the less distinctive features. Depending on the dimension of the code D_h in comparison to the input D_x the DA can be undercomplete when $D_h < D_x$ or overcomplete when $D_h > D_x$. An undercomplete DA will force the detection of salient features in the input data. Sparse Autoencoders is a type of DA that responds to unique statistical features of input data [26], encapsulating in their structure some latent variables that are useful to explain the data. Denoising Autoencoders (DAE) [27] receive corrupted input and try to reconstruct the original data, thus eliminating the corruption reason or noise.

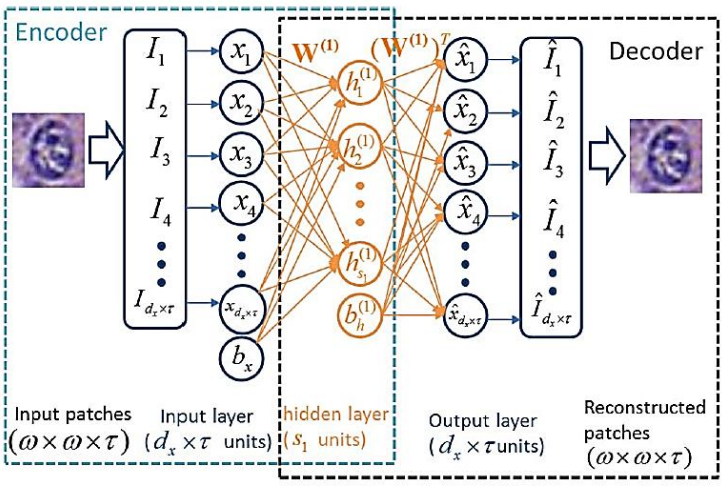

Figure 6: Autoencoder architecture used in [28]

A practical application of DA networks is extracting appropriate features for classification, coping at the same time with big image data, like in high-resolution medical imaging [28].

Autoencoders can be thought in image recognition as “focusing” on the underlying factors that create the visual features of the scene and discarding the unimportant image data.

The most widely applied type of DNN in the domain of image recognition is the Deep Convolutional Neural Network, DCNN or simply CNN [5] . It combines some key ideas like the use of local receptive fields, the use of replicated or shared weights, spatial sub-sampling and overlapping neurons on the same input, that were discussed by Rumelhart, Hinton and Williams in [4] and later implemented by LeCun et. al in [29]. Because of the high-dimensional nature of images, MLPs that work on pixels could not be practically implemented, due to the enormous count of synaptic weights resulting from the dense interconnections. Instead we can abstract a count of neurons that belong to the same type, share the same weights and are mapped sparsely around specific regions of the image. Their input is the pixels of a local region, which has a shape and size that is identical to the neurons’ receptive fields. Using the mathematical function of convolution [30] in subsequent horizontal and vertical steps on the image, we can create the synaptic sums for neurons of the same type, without the need for dense connections or keeping separate weights for each one of them.

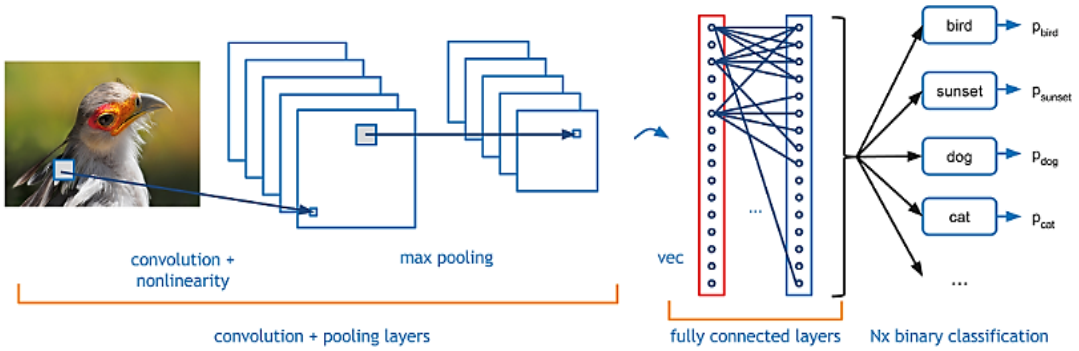

Figure 7: Deep Convolutional Neural Network (CNN) for classification (image from [31]).

The output of a convolutional layer of neurons can be calculated as the result of an activation function which receives the convolution result plus an additional bias. Training of a CNN can be done by minimizing an error function with gradient descent methods, exactly like training a fully connected MLP. The idea of local receptive fields moving on input [4] was inspired and coeval with by the work of Hubel and Wiesel on the cat’s visual system [32], but it was systematically elaborated as a neural network architecture in the early CNN implementations like LeNet-5 [5]. Artificial neurons in CNN layers are similar to biological neurons and can be intuitively be described as “sensing” visual features on an input array of RGB pixels. The neural networks in this thesis are categorized in the CNN group of deep neural networks.

The recently introduced deep learning architectural concept of Network-In-Network (NiN) [33], has inspired many state-of-the-art CNN architectures, yielding significant results.

Its concepts could be further expanded to other DNN networks and all networks that implement these can belong to the NiN meta-group. According to it, a micro neural network is placed between two convolutional layers to increase the representational power of the extracted features. The micro-architecture of multiple layers be considered a *convolutional module* inside the CNN architecture. An intermediate NiN layer can be expressed as an 1×1 convolutional layer, and has been directly incorporated as such in CNN implementations [34] [35]. Attempting an abstraction with object oriented terms, mini networks are "private" members of convolutional modules with "encapsulated" functionality.

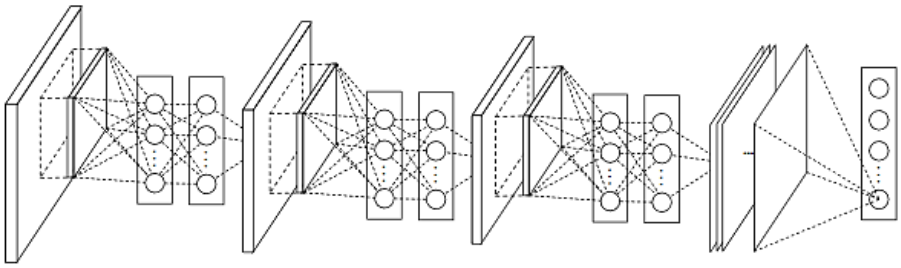

Figure 8. Network-in-Network Architecture with three modules each having one convolutional layer followed by a micro-network [33]

The most recent and interesting meta-group of deep networks with applications in Computer Vision, are the Generative Adversarial Networks (GANs) [36]. They are introducing a novel deep learning method that is called *Adversarial Learning.* There are two networks the generator and the discriminator that are trained in parallel on an image dataset, in an unsupervised manner. The generator starts from random noise and present outputs that should gradually follow the underlying distribution of the image set. The discriminator decides if the statistical features of the generated images match the real images and they are both trained accordingly. After some epochs of training the generated images look more and more real, because the generator has learned to "forge" specific features of the actual images. Implemented as CNNs the Deep Convolutional Generative Adversarial Networks (DCGANs) displayed interesting perspectives of use in object recognition, like capturing of scale, rotation and position of object features [37].

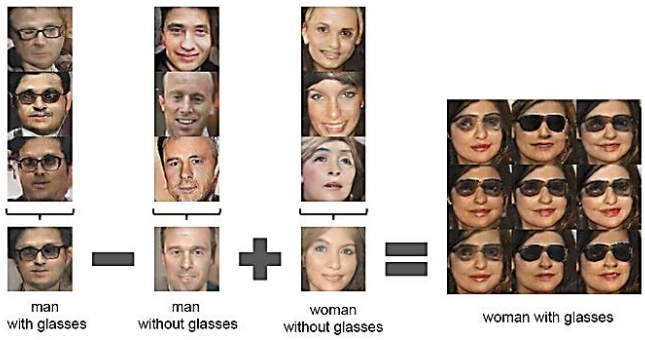

Figure 9: Demonstration of visual features captured by a DCGAN, implementing vector arithmetic operations [37]

2.3.2 A Short History of Deep Learning for Image Recognition

The deep learning research in the image recognition domain has yielded significant results in the past decade and is creating the dynamics for real-world applications of neural networks. The timeline of events starts in 2006 when it was demonstrated on the MNIST handwritten digits dataset, that using unsupervised DBN pre-training can help the supervised gradient descent training to converge faster and to a state-of-the-art classification performance [17]. In the same paper, training a DA for the purpose of dimensionality reduction outperformed PCA. In 2011 the Convolutional Auto-Encoder (CAE) generated the best results for any unsupervised method of training using unprocessed images of the CIFAR10 image dataset [38]. In order to draw safer conclusions on the effectiveness of deep learning models and compare them to other approaches a common benchmarking ground was needed.

This ground has proven to be the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) that is annually held [9]. It was in 2012 that the advantages of the CNN networks were shown in this competition. The winner of ILSVRC2012 for the task of image classification was team SuperVision from the University of Toronto which introduced a deep CNN, afterwards known as AlexNet [3]. It surpassed with a great gap the second best non-neural model of team ISI from the University of Tokyo. Since then, all ILSVRC winners in classification used CNN architectures. The models which are pre-trained during the classification task, have been proven additionally effective for the task of object detection [39] [40].

2.4 CNN for Image Classification

2.4.1 Convolution Operation on Color Images

Convolution is a mathematical operation between two functions that is used to model the shared weights and the local receptive fields of neurons in a layer of a CNN. For the functions $g(t)$ and $f(t)$ the convolution in the time domain is the integral [30] :

$$h(t) = (g * f)(t) = \int_{-\infty}^{\infty} g(t-x)f(x)dx \tag{1}$$

For machine learning f returns values from the input vector and the convolutional kernel g will be a window of size n that is moved over the input with a stride of s. The one-dimensional convolution at a specific position x of the 1D input vector is:

$$conv1D(x) = (g * f)(x) = \sum_{j=1}^{n} g(j)f(x-j+\frac{n}{2}) \tag{2}$$

The 1D convolution is simply the sum of the multiplications of their values. We can extend this to define a two-dimensional convolution, suitable for operating in pixel data. The corresponding convolutional kernel will have a 2D *receptive field map* (RFM) as the sliding window. The weights of this map are shared by neurons of the same type, which are mapped to a grid of local regions. For the kernel g we define two spatial hyperparameters, width a and height b. The 2D convolution centered on a pixel (x, y) for an input feature $p = p(x, y)$ is:

$$conv2D(\boldsymbol{p}) = (g * f)(\,p(x,y)\,) = \sum_{j=1}^{a}\sum_{i=1}^{b} g(i,j)f(p(x-j+\frac{a}{2}, y-i+\frac{b}{2})) \tag{3}$$

For $a = b = n$ the kernel is a square and the single RFM is a $n \times n$ matrix of weights that slides horizontally and vertically over the input image. With a stride $s = 1$ a convolution will be performed on all pixels of the image, resulting in a 2D matrix of one extracted feature:

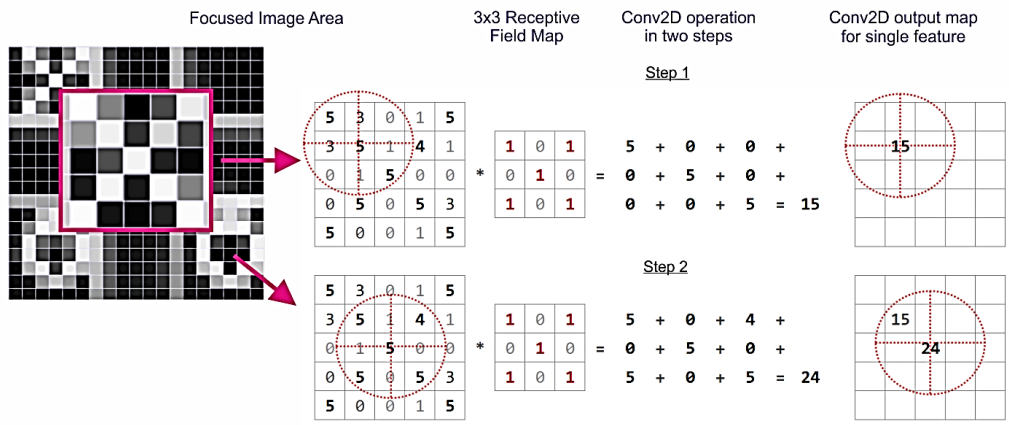

Figure 10: Convolution with stride 2 on a grayscale image, using a 3x3 RFM of an X-shaped feature.

Convolution in 2D blends paired values of g and f through multiplication to produce the total sum at a point of the 2D input matrix. This thesis introduces the following notation for the convolution spatial window and the common stride on both horizontal and vertical directions:

Convolution Window Notation	$[\boldsymbol{a} \times \boldsymbol{b} \sim \boldsymbol{s}]$

For example $[11 \times 11 \sim 4]$ is used to represent a squared-shape convolutional kernel map with a side of 11 pixels and a stride of 4 pixels in both horizontal and vertical dimensions.

Equation 3 is applicable in grayscale images that have only one feature per pixel, the intensity of light. To express a convolution on color images that have $m = 3$ RGB input features, there will be 3 receptive field maps in the kernel and the equation is modified as:

$$conv2D(\boldsymbol{p}) = (g * f)(p(x,y,k)) = \sum_{k=1}^{m}\sum_{j=1}^{a}\sum_{i=1}^{b} g(i,j,k) f(p(x-j+\frac{a}{2}, y-i+\frac{b}{2}, k)) \quad (4)$$

The index k selects an input feature channel from $\{R, G, B\}$ and its respective RFM, having $k = 1$ for red, $k = 2$ for blue and $k = 3$ for green. The function $p = p(x,y,k)$ returns the value of the color component k at a point (x,y) and $g(i,j,k)$ the weight for the color component at position (i,j) of the k^{th} RFM. For RGB color images there will be a matrix of weights $G_k = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1j} \\ \vdots & & \ddots & \vdots \\ w_{i1} & w_{i2} & \cdots & w_{ij} \end{bmatrix}$ for each input feature k and the function $g(i,j,k) = w_{k_{ij}}$.

The result of the two-dimensional convolution operation on all input feature channels will produce a 2D matrix for one extracted feature that convolves them:

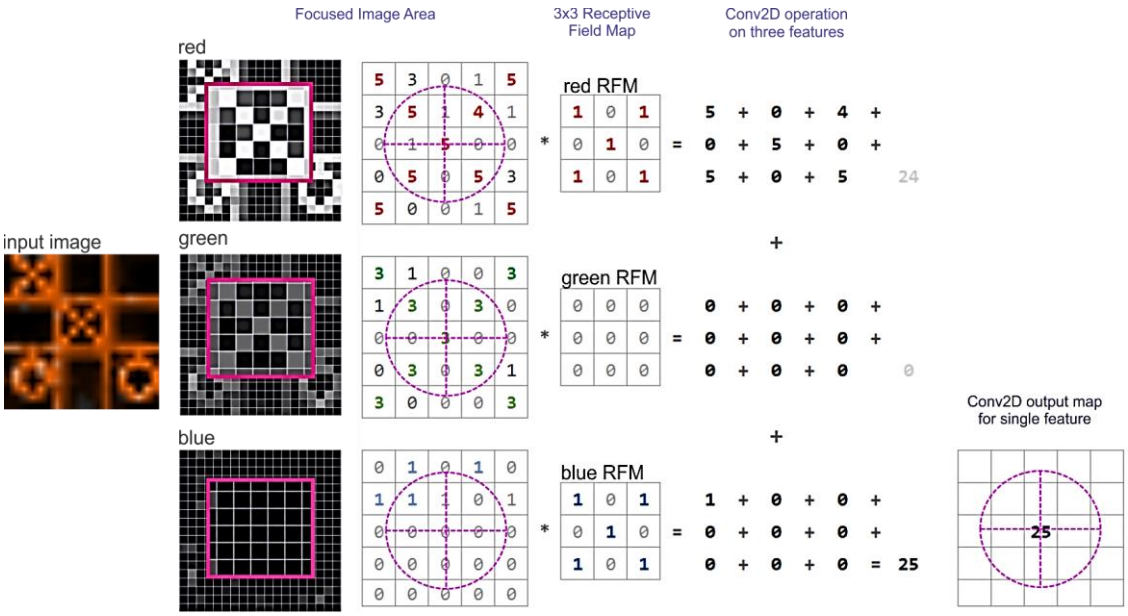

Figure 11: Convolution on an RGB image with orange shapes, using a 3x3 RFM of a magenta-colored X-shaped feature. For the range of RGB values [0, 5] the maximum result of convolution with this kernel can be calculated as 50. The current region of input produces a 50% level of activation.

The number of RFM maps for a single output feature is equal to the number of input features, introducing a third hyperparameter for the kernel its input depth $d_{input} = m$. The resulting output map of $conv2D(p)$ will be an array of the features values which are extracted from m convolved input features.

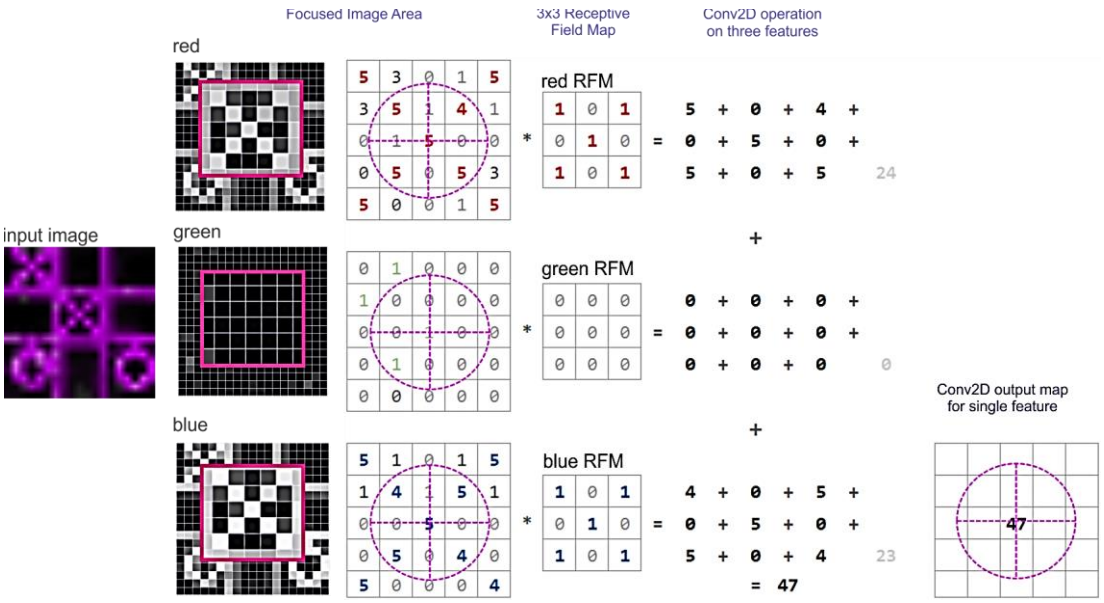

Figure 12: Convolution on an RGB image with magenta shapes, using a 3x3 RFM of a magenta-colored X-shaped feature. Comparing to the sample used in figure 11, this produces a 94% level of activation, 188% stronger, due to the blue RFM that has matching input features.

The convolutional kernel may represent columns of neurons each one activated on a different feature, which are the biological equivalent of hypercolumns that are formed by neurons in the mammals' primary visual cortex [41] [42]. For multiple output features the kernel g becomes a 4D tensor with a fourth hyperparameter, the depth of output features $d_{output} = h$ that will be described in this thesis with the term *output hyperdepth*. For a common terminology the $d_{input} = m$ will be named *input hyperdepth*. The generalized two-dimensional convolution operation that supports $m - to - h$ relations between input and output features using a 4D convolutional kernel can be formulated as:

$$\boldsymbol{U}_z = conv2D_z(\boldsymbol{p}) = (g_z * f)\big(p(x,y,k)\big) \quad : k = 1..m \quad , z = 1..h \tag{5}$$

The index z refers to a 3D slice of a kernel that has m receptive field maps for the z^{th} output feature. The results for each location of input will be placed at the 3D activation map U_z. The notation of the convolution is extended to include the input and output feature hyperdepths of the kernel, along with the spatial dimensions a, b and the stride s.

Convolution Notation for 4D Kernels	$[\, \boldsymbol{a} \times \boldsymbol{b} \sim \boldsymbol{s} \mid \boldsymbol{m} \rightarrow \boldsymbol{h} \,]$

Convolution with a 4D kernel blends m incoming features values with their weights in the respecting RFMs resulting in h values of output features, thus multiplexing them inside the $[\,a \times b\,]$ local region. The operation is performed repeatedly at (x, y) coordinates by incrementing x and y by a common stride s. When $s > 1$ a convolution on a square image of dimension w will result in a spatial downsampling reducing the resolution of the image representation to a new spatial dimension ${w_{output}}^2 < \left(w_{input}/s\right)^2$.

At the edges of the image the indexes of function p in equation 4 will be outside of the image boundaries. In order to circumvent this an appropriate padding of the image with extra pixels is done, typically with zero values for their RGB components. The function to calculate a valid integer square dimension w_{output} for the output of the convolution when using square-shaped kernels of size *n* and an appropriate zero padding p is:

$$w_{output} = \frac{\left(w_{input} - n + 2p\right)}{s} + 1\,, \qquad w_{output} \in \mathbb{N} \tag{6}$$

For example, a convolution $[\,32 \times 32 \sim 3 \mid m \to h]$ is done on an image with resolution 103×103. Without padding $p = 0$ and with $p = 2$ the result is a non-integer output size, hence not valid padding sizes. Using $p = 2$ it is calculated $w_{output} = \frac{(103-32+2\cdot 2)}{3} + 1 = \frac{75}{3} + 1 = 26$. The convolution operation will result in an output 3D tensor of activations $[\,26\,x\,26 \mid h\,]$. The overlapping areas of input between neurons at the various sampling point on the image, depends upon the choice of the convolution hyperparameters s and n.

2.4.2 Convolutional Layers

Practically convolution is a transformation of m input features to h output features, depicted as $m \to h$ in the notation, that can either keep the same spatial resolution or reduce it, depending on the choice of stride. For its application in the context of artificial neural networks, it needs to be formulated as a synaptic sum of the neuron inputs. Generally an artificial neuron is defined as a function

$$y = f(u(x)) = f(\boldsymbol{w}\boldsymbol{x} + b) \tag{7}$$

where x is the vector of input values, w the vector of the synaptic weights, b a single value of bias, $u = u(x)$ the synaptic summation function and f the activation function. To model the operation of multiple neurons of the same type in the same neural network layer, the convolution can be expressed as a dot product. We consider X an input 3D tensor of an image

region that is mapped to the neuron at the current position of the convolution stride. W_z is a 3D slice of the 4D convolution kernel, that has the synaptic weights for each input feature for a class of neurons z. All neurons in the class share the same weights but are mapped in different regions of the image. The synaptic sum of inputs and weights for each neuron in a convolutional layer is:

$$\boldsymbol{U_z} = u_z(p(x,y,k)) = conv2D(\boldsymbol{W_z} \cdot \boldsymbol{X}) + \boldsymbol{b_z} \quad : k = 1..\ m, \qquad z = 1..h \qquad (8)$$

The function p returns the value of the k input feature at coordinates(x, y), b_z is the 1D vector of biases for the m inputs and u_z is the 2D output map for feature z . The weight sharing characteristic between neurons of the same type is implemented by the common RFM maps. Moreover X and W have the same square dimensions n implementing the locality of the receptive field mapping in the region around (x, y). The sliding operation on the input with the stride s implements the sparsity in the connections between the convolutional layer and its input layer. The choice of dimension n along with s regulates overlapping of two or more neurons of the same class on the incoming data, like forming synapses on the same input neuron's axon. The overlapping scheme creates a redundancy of the forward propagated information, especially useful when a downsampling of the image is necessary.

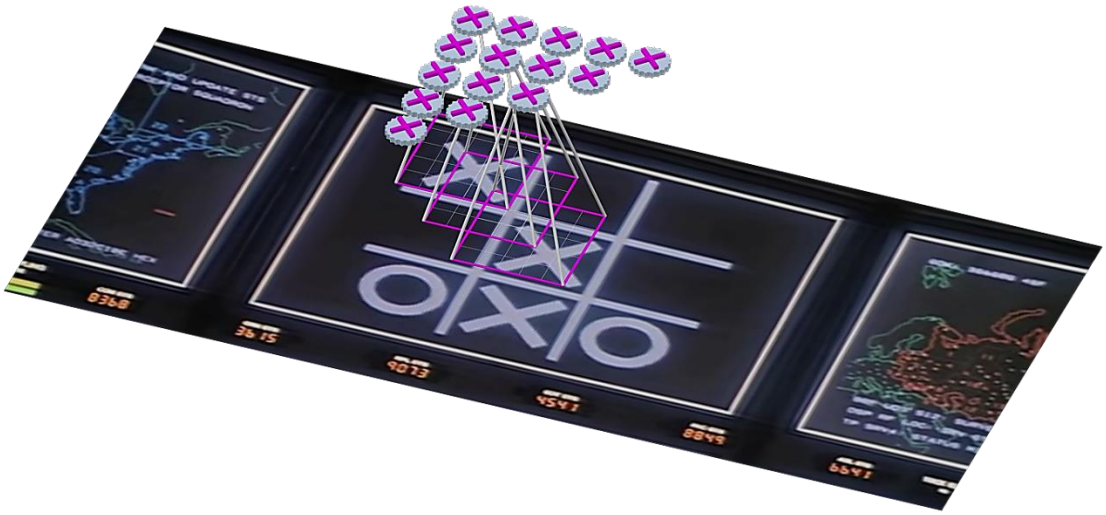

Figure 13: Spatial arrangement of overlapping neurons in a convolutional layer $[5 \times 5 \sim 2\,|3 \rightarrow h]$. All neurons for the magenta-colored X-shaped feature, share a common slice of weights of the 4D convolutional kernel. (Input sample is a frame from the 1983 MGM movie "War Games" depicting an AI trying to find a winner in Tic-Tac-Toe).

Different kind of features are extracted at a specific point by neurons of different types in the same layer that are stacked into a hypercolumn and spatially mapped in the same region of the image. Moreover the convolutional layer is a cube of neuron columns, which are arranged in a grid to cover the whole image area with potentially overlapping windows. The generalized synaptic sum function for *h* different types of neurons in a convolutional layer is:

$$\boldsymbol{U} = conv2D(\boldsymbol{G} \cdot \boldsymbol{X}) + \boldsymbol{B} \quad (9)$$

The two-dimensional convolution of the 3D input tensor X and the 4D convolutional kernel G is added to a 2D matrix of biases is B resulting in the 3D tensor U of the synaptic sums. Completing the definition of the neuron in a convolutional layer we denote the activation function f and the 3D output tensor Y that holds the activation maps for all neurons across the image:

$$\boldsymbol{Y} = f(\boldsymbol{U}) = f(conv2D(\boldsymbol{G} \cdot \boldsymbol{X}) + \boldsymbol{B}) \quad (10)$$

Figure 14: Hypercolumns of neurons in a convolutional layer $[5 \times 5 \sim 3 \mid 3 \rightarrow h\,]$ showing different types of neurons. There are 7 neurons mapped in the same spatial region of the input image. , which correspond to an output hyperdepth $h = 7$ of the convolutional layer.

2.4.3 Pooling Layers

The convolutional layers generate multiple feature activation maps in the resulting 3D output volumes. For increasing the depth of the network and the layers' hyperdepths in a computationally efficient manner, two needs must be satisfied. The image must be spatially downsized as the number of features increases from the initial three RGB values to the scale of hundreds or thousands. For example, a true color input image of resolution 255×255 processed by a convolutional layer $[\,5 \times 5 \sim 1 \mid 3 \rightarrow 6\,]$ the total count of weights in the kernel can be calculated as $5 * 5 * 3 * 6 = 450$ and the count of features in the 3D output $255 * 255 * 6 = 390150$. If using a double stride $[\,5 \times 5 \sim 2 \mid 3 \rightarrow 6\,]$ then the valid output is calculated according to formula 6 is $126 * 126 * 6 = 95256$, that is a 75.6% spatial reduction. The second need is to discard noise that would be propagated in deeper layers of the CNN, like image compression artifacts [43]. Also unimportant local variation needs to be discarded in order to create internally useful image representations.

For these reasons, convolutional layers are typically followed by layers that implement pooling functions [44] on regions of the feature activation maps, or pools of features. The pooling operation results into a tensor with the same hyperdepth of features and a spatially smaller image representation when $s > 1$. An *average pooling layer* (AVP) calculates the average over a local pool in the resulting activation map of a convolution.

$$avgpool(\boldsymbol{Y}) = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} p\left(x - i + \frac{m}{2}, y - j + \frac{n}{2}\right) : p \in Y \tag{11}$$

This resembles a convolution of Y with a kernel g having weights set to the value $\frac{1}{m \cdot n}$ where $m \cdot n$ the total count of neurons in the pool. There can be overlapping average pooling regions and the result will be a moving average calculation on the image pixels. Smoothing of the values may have benefits in reduction of noise, but also blur the fine features that exist in the higher resolution of the image.

For this reason local average pooling is rarely used and the *max pooling layer* (MAXP) is typically found, which has a different effect on image downsampling. The operation keeps the maximum activation value in pooled region and discards the others.

$$maxpool(\boldsymbol{Y}) = \max_{i,j}(p(x - i + \frac{m}{2}, y - j + \frac{n}{2})) : p \in Y \tag{12}$$

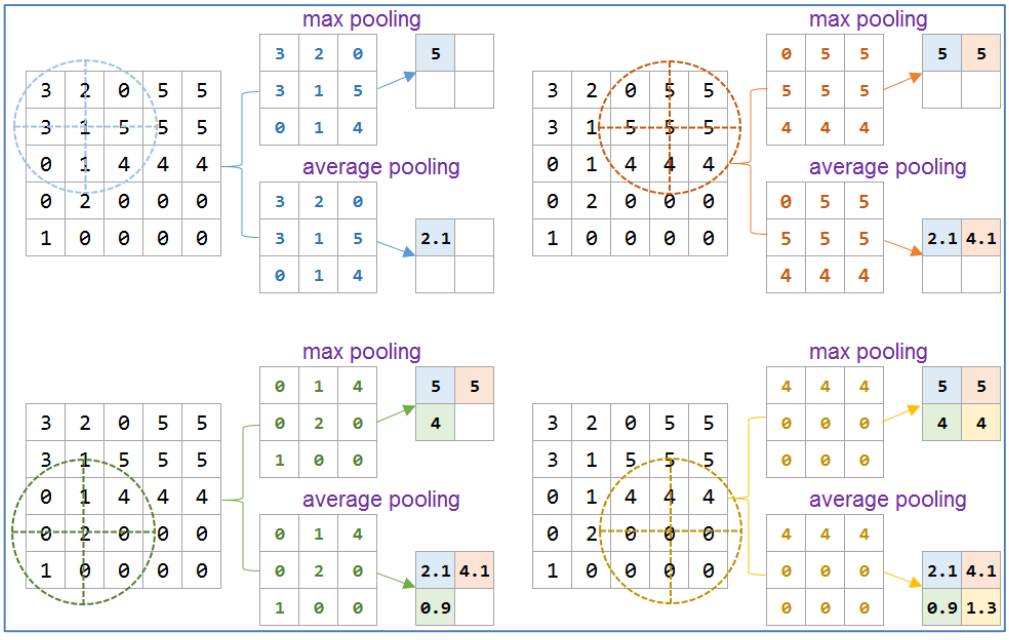

Figure 15: Max and average pooling layers [$3 \times 3 \sim 2$] over a single 5×5 activation map. The max value in the bottom left region of input (green crosshair) and the bottom right region (yellow crosshair) is 4, while the average for the same pools are 0.9 and 1.3. The max value is inhibited by the other local values in average pooling.

Keeping the max activations in overlapping pools may have this effect: The strongest feature inside a broader neighborhood will be the only one kept it in the pooled output. This is typically desired for the CNN functionality, but it can also have a negative side effect because fine details in lower resolutions or entire rows or columns of lower activations could be discarded. The latest type of pooling that has been successfully used in the CNN context is *global average pooling* (GLAVP). Proposed as part of the NiN baseline architecture, it is used after the last convolutional layer [33]. The GLAVP layer calculates the average of each feature activation map and these values are fed directly into the classification output layer. This can remove the need for fully connected layers in a CNN based classifier. It is considered by the authors as a structural regularizer of the CNN, transforming feature activations into confidence maps, by creating correspondences between features and classes.

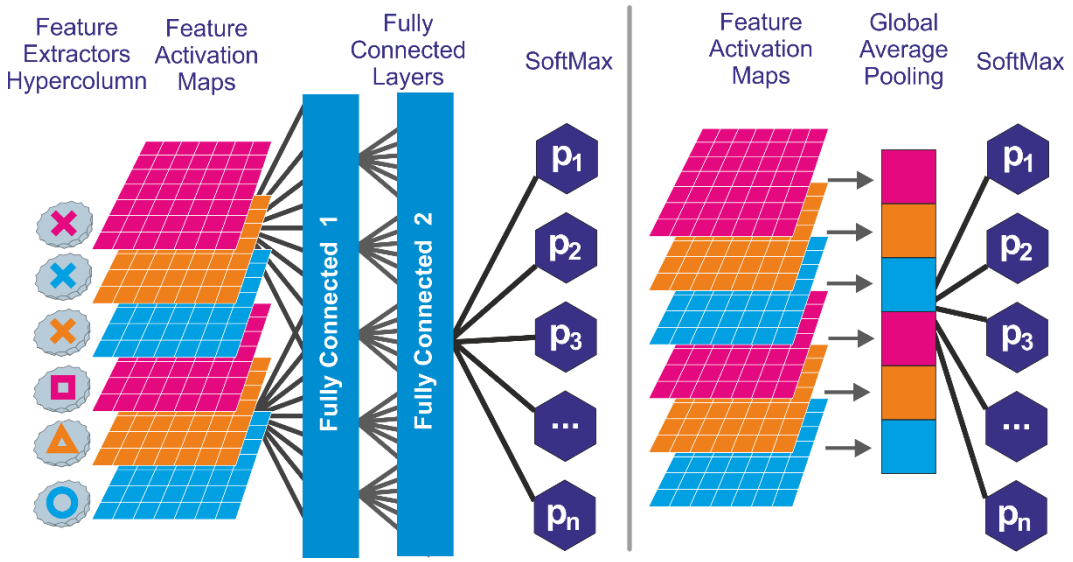

Figure 16: Global average pooling layer replacing the fully connected layers. The output layer implements a Softmax operation with $p_1, p_2, \ldots, p_n$ the predicted probabilities for each class.

Seeking an analogy in nature for pooling layers, it has been already studied by Neuroscience a property of biological neurons called *lateral inhibition*. The transmitted activations of feed-forward cells are pooled and an inhibitory signal is sent back to them, thus modifying the amount of input that reaches the receiver cells [45]. Neurons that have stronger responses tend to inhibit the response of neighboring neurons. We can think of max pooling as an implementation of lateral inhibition, where the max response fully inhibits its non-max neighborhood.

Properties of human visual perception like brightness induction have been linked in a recent study to lateral inhibition in the retina [45]. Various optical illusions that appear from differences in brightness levels were assumed to be caused by lateral inhibition.

Figure 17: Brightness induction illusions. Left brightness contrast and right brightness assimilation. The perceived as different intensities of gray, for the two center squares and for the vertical gray lines, have the same brightness level. Their background is causing lateral inhibition of neurons on illumination boundaries [45].

2.4.4 Activation Functions

The activation function of the convolutional neuron introduces a desired non-linearity in the network. This is preferable for deep networks because non-linearities introduced between successive layers can create a more expressive model [46]. The output should not be a mere linear combination of the input in order to have discriminative power encapsulated in the weights of the convolutional kernels. Usually the activation function used in a convolutional layer is a rectifier, the *Rectified Linear Unit* or ReLU [47] :

$$y = ReLU(u) = \max(u, 0) \quad , u \in \mathbb{R} \quad : y \in [0, +\infty) \tag{13}$$

A variant that restricts the output range is ReLU6:

$$y = min(\, ReLU(u), 6\,) = \min(\, \max(u, 0)\,, 6\,) \quad : u \in \mathbb{R} \quad , y \in [0,6] \tag{14}$$

The choice of a function without sign symmetry (anti-symmetric) that produces only positive output values, has been made to mitigate the *vanishing gradients problem* [48] that emerges during gradient descent training of deep neural networks. When using the sigmoid activation function, the error flow through back-propagation can decrease exponentially until it vanishes [49]. Having derivatives of positive values can help alleviate this [47] [50]. A smoother rectifier function with the same properties is Softplus [51].

$$y = softplus(u) = log(1 + e^{u}) \quad : u \in \mathbb{R} \quad , y \in [0, +\infty) \tag{15}$$

The functionality of rectifiers has a relation to observations of Neuroscience, since it can approximate the activations of cortical neurons, who mostly stay in an unsaturated state [47].

However all the basic rectifiers with non-negative values have a mean activation larger than zero. Used as input for the next layer, the neurons do not inhibit each other with different signs $+/-$ in the synaptic sums of the receiving neurons. This is biasing the next layer negatively during the learning process, an effect that is called *bias shift*. If during the training process neurons of the same layer start to coordinate their responses in order to follow some underlying correlation, the shift will increase [50]. For this reason symmetric

rectifiers have been proposed. The *Leaky Rectified Linear Unit* or LReLU [52] is an activation function with some variants that have been proposed since its introduction.

$$y = LReLU(u) = \begin{cases} ReLU(u) & ,u \geq 0 \\ -a \cdot ReLU(-u) & ,u < 0 \end{cases} \qquad (16)$$

$$0 < a \leq 1\,, \qquad u \in \mathbb{R}, \quad y \in (-\infty, +\infty)$$

In the LReLU α is a positive constant with a typical value of α = 0.01. A variant is the *Parametric Rectified Linear Unit* or PReLU proposed in [53] with the only difference that α becomes a set of learnable parameters in the network and their values are determined through gradient descent optimization. A second variant is the *Randomized Leaky Rectified Linear Unit* or RReLU [54] that has also a set of α values one for each layer, but randomly chose. During training phase these values are randomly picked from a uniform distribution and during recall an average of values $\bar{a}$ is used for deterministic result.

A more systematic approach in finding a solution of the bias shift problem while mitigating at the same time the vanishing gradient problem, is found in [50] where the *Exponential Linear Unit* or ELU is introduced.

$$y = ELU(u) = \begin{cases} ReLU(u) = max(u, 0) & ,u \geq 0 \\ a \cdot (e^u - 1) & ,u < 0 \end{cases} \qquad (17)$$

$$0 < a \leq 1\,, \qquad u \in \mathbb{R}, \quad y \in (-a, +\infty)$$

For the positive values of the synaptic sum u it is a simple ReLU and restricts the negative values using a small constant a. It has been proved experimentally that the correction of bias shift can benefit learning, both in speed of convergence and generalization capabilities of the model. The results in [50] report a state-of-the-art performance for classification on CIFAR100 dataset, using ELUs for convolutional layers.

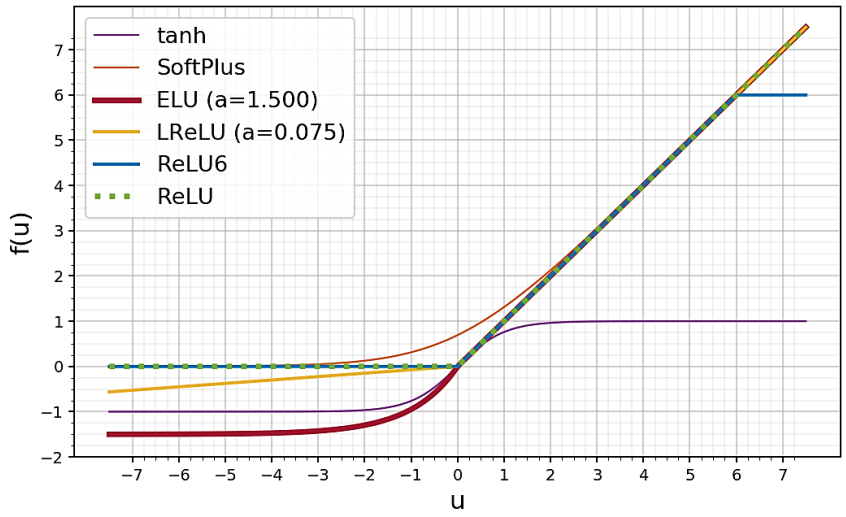

Figure 18: Comparison of different activation functions

2.4.5 Pre-training a CNN for Classification

2.4.5.1 CNN-based classifier and gradient descent training

The classification task is important in the context of CNN, since objects that are members of the same class share visual similarities. Presenting multiple sample images of a class during supervised training of a CNN, results in a set of learned parameters which transforms the convolution kernels into efficient visual feature extractors. A pre-trained CNN on the classification task, can be reused for other image recognition or image retrieval tasks. Following a *transfer learning* [55] approach, the learned parameters of the model are reused on a completely different dataset of images. The model needs to be fine-tuned through a complementary training process where new images are presented. Completely different image domains for pre-training and fine-tuning can be used like in [56], where a CNN that has been pretrained on natural images and general classes and was fine-tuned on medical images. The experimental study of transfer learning has created an intuition that the convolutional features learned on large image datasets are somehow generic.

For classification, a CNN architecture usually consists of several convolutional (CONV) and pooling (POOL) layers organized in convolutional modules, followed by a classifier that is placed after the last convolutional module, denoted here as CONV_{OUT}. The classifier after the CONV_{OUT} usually consists of a stack of fully connected (FC) layers and a Softmax (SMAX) [57] output layer. The 3D activation maps of the CONV_{OUT} layer are flattened into a vector so that each value is forward fed to all neurons of the FC layer. The Softmax function outputs n responses with the probability of each class [58]:

$$f(k) = z_k^* = \frac{e^{z_k}}{\sum_{k_1} e^{z_{k_1}}} \tag{18}$$

This function ensures that $\sum_k z_k^* = 1$. The output of a SMAX layer with n classes is

$$softmax_k(X) = P(C_k | X) = \frac{e^{\theta_k^T X - b(\theta_k) + \log(\mathrm{P(C_k)})}}{\sum_{j=1}^{n} \mathrm{e}^{\theta_j^T X - b\left(\theta_j\right) + \log(P(C_j))}} \tag{19}$$

where P the probabilities of classes $C_k = \{c_1, c_2, \ldots c_k\}$ given the input vector X. In the case of using a GLAVP layer after the CONV_{OUT} layer the classifier can simply be the Softmax layer. The probabilities $P(C_k \mid X)$ are directly linked with the visual features, because the X_{SMAX} input for the SMAX layer would be a vector of averages for each feature activation map.

The spatial dimensions and the hyperdepth of the CONV_{OUT} layer output is a basic decision in the design of a CNN, because the image representation must have been

sufficiently reduced so that the features h and the $d \times d$ dimensions of the activation map multiplied by the number of fully connected neurons would not overflow the memory requirements for the model. The number of weights for the first FC layer or the SMAX layer, when using square-shapes images and square kernels is $d^2 \cdot h \cdot n$, where d the dimension of the CONV_{OUT}, h the hyperdepth of its output features and n the number of fully connected neurons. The d^2 factor indicates the significance of spatial downsampling in the CNN architecture.

Since the convolution is differentiable [59] and by using the derivative of the activation function the network can be trained with gradient descent methods. By minimizing an error function, the error gradients are back-propagated from the SMAX layer to the FC, POOL and CONV layers, modifying the weights of the receptive field maps in convolutional kernels and their corresponding biases. Using the *multiclass categorical cross entropy* (CCE) as an error function to be minimizes, the gradient descent method implements variance minimization [60]. Supported by a theoretical basis and experimental results, it has been shown that it can help the model to convergence to a better local optimum than mean squared error [61]. For the classification of samples to n classes, the CCE error function for a given sample of a class is using the probabilities predicted by the model for each class $y^k_{PREDICTED}$ and the corresponding to the sample ground truth probabilities y^k_{ACTUAL}.

$$J_{CCE} = -\sum_{i=1}^{n}\sum_{k=1}^{c} y^{(k)}_{ACTUAL} \log(softmax_k\left(y^{(k)}_{PREDICTED}\right)) \tag{20}$$

For training CNN networks, some usual gradient descent methods include Backpropagation with Momentum [4], Nesterov Momentum [62], Stochastic Gradient Descent [63], AdaGrad [64], AdaDelta [65] and RMSProp [66]. A recent stochastic optimization method, that has minimized memory needs and combines features from AdaGrad and RMSProp is the Adam/AdaMax method, which shows promise of faster and better convergence [67]. Nevertheless the benefits of using different training methods with regard to the quality of the learned CNN features needs to be further examined.

The more recent architectural choices for CNN classifiers, tend to remove the fully connected layers in attempt to alleviate overfitting, which is the most important problem during training. The connections between the GLAVP layer and the Softmax output layer create a direct correspondence between convolutional features and classes, regularizing the entire network [33]. It eliminates the need of minimizing an additional error, thus preventing additional overfitting that is potentially introduced by the FC layers. Regularization is one of the categories of solutions against overfitting that are mentioned in the following paragraphs.

2.4.5.2 Overfitting and early stopping

The problems of vanishing gradients, bias shift and overfitting, occur during training of a deep convolutional neural network. Overfitting starts when the model passes a certain threshold of optimization on the training data, continuing to improve its accuracy on them but at the same time its generalization capability deteriorates. This can easily happen with image datasets due to the diversity of visual information that may exist and the millions of parameters of a deep CNN. To avoid overfitting the training usually includes an early stopping mechanism that can be achieved through validating a small subset of images at the end of each epoch, which are not presented as training samples inside the epoch. This is known as the validation subset of the ground truth image set, while the unknown images used to test the network accuracy are considered the unknown test image set.

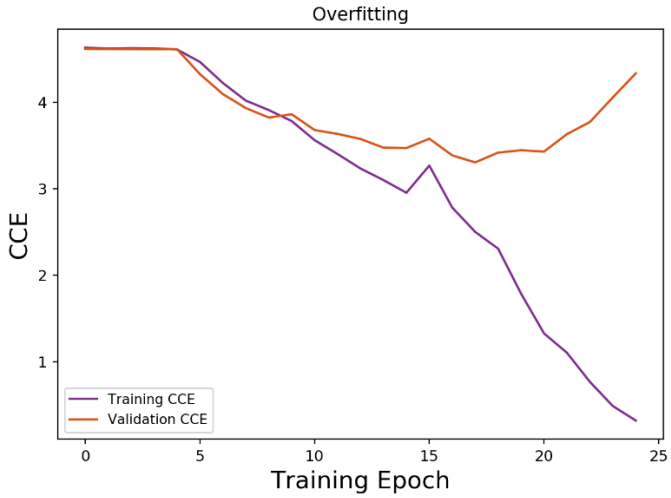

Figure 19: Overfitting during 25 epochs of training. The training and validation categorical cross entropy (CCE) error curves start to diverge after epoch 9.

2.4.5.3 Regularizers

A technique against overfitting is to regularize the network using weight decay [68], which enforces small weight values and avoids unequal participation of the input values. This is done by either penalizing the absolute value of the weights difference using the L_1 (Manhattan) norm $L_1 = \|x\|_1 = \sum_{i=1}^{n}|x_i|$ or the Euclidian distance of the weights using the L_2 (Euclidian) norm $L_2 = \|x\|_2 = \sqrt{\sum_{i=1}^{n}|x_i|^2}$ [69].

The main cause for overfitting was identified as the complex co-adaptations of neurons that may occur in the FC layers of a CNN. A proposed regularizer is the Dropout method [70], which randomly excludes a set of neurons from training by a predefined probability that is a layer-specific hyperparameter. A typical value for the dropout keep probability is 50%, which has proven helpful against the overfitting problem. Dropout is not

used during recall of images in a pre-trained model. A generalization of Dropout which is motivated by the same idea of dynamic sparsity between the intra-layer synapses, is DropConnect [71]. It deactivates synapses between neurons of two layers, by masking out the weights and biases. In a recent proposal DisturbLabel [72] imposes the regularization in the loss layer, usually the SMAX output layer. Its selects a random subset of training data during the training process and replaces their ground-truth labels with incorrect values. The fraction of the training data with distorted labels is determined by a noise rate a , which has a value either large $a = 0.9\ (90\%)$ or small $a = 0.1\ 10\%$ depending on the dataset. It has been already mentioned that the global average pooling layer of the NiN architecture, which provides average feature values as direct input to the SMAX loss layer, is also considered a network regularizer [33].

An interesting statement in [70] gives to the Dropout regularization as bio-inspired notion. Dropout has been successful into learning better parameters for a model, like the biological mechanism which enhances evolvability and avoids "evolutionary dead-ends", described by the theory of mixability [73].

2.4.5.4 Data Augmentation

Despite the helpful effect of regularizers, the root cause might be considered the countless combinations of visual information that may exist. The effort to mitigate overfitting can be focused on the image data. The proposal is to augment the dataset with transformations of the original image without distorting the objects inside it, leaving it still recognizable according to human perception. These label-preserving transformations may be implemented on the fly, without the need of extra physical or external memory. They may include cutting random crops from the original image, rotating and flipping the image. Additionally in [3] the RGB values of pixels are modified in each training epoch. Before the start of the training PCA [74] is performed over RGB vectors for every pixel in the image dataset and the principal components are discovered. This can be considered a learned color space [75] through PCA. In each training epoch and for each image, random values are drawn from a zero-centered Gaussian distribution. The new pixel vectors $p_{NEW_{x,y}}$ for the RGB values at coordinates (x, y) are calculated as:

$$p_{NEW_{x,y}} = p_{x,y} + [\ p_1, p_2, p_3] \cdot [\ v_1\lambda_1 + v_2\ \lambda_2 + v_3\ \lambda_3\ ]^T \tag{21}$$

The values v_1, v_2, v_3 are chosen randomly and are common for all pixels in the image, $p_1, p_2\ , p_3$ are the eigenvectors and $\lambda_1, \lambda_2, \lambda_3$ the respective eigenvalues. The data augmentation techniques may help a model acquire invariance qualities like illumination and rotation invariance that are desired for effective image recognition.

2.4.5.5 Data Preprocessing

Preprocessing of the image dataset is also used to increase accuracy, typically by "whitening" their RGB values and normalization. PCA whitening is one of the typical preprocessing techniques used. Given an eigendecomposition $\bar{X}\bar{X}^T = PDP^T$ where D is the diagonal matrix of eigenvalues, PCA whitening is the transformation of an image RGB values from X_{OLD} to X_{NEW}:

$$\boldsymbol{X}_{NEW} = \boldsymbol{W} \cdot \boldsymbol{X}_{OLD} = \sqrt{\boldsymbol{D}^{-1}}\, \boldsymbol{P}^{-1}\, \boldsymbol{X}_{OLD} \tag{22}$$

The part $W = \sqrt{D^{-1}}\, P^{-1}$ part can be considered as a *whitening filter* on the image dataset. Additionally ZCA whitening [68] transforms the image adding a small amount of ε to the eigenvalues matrix that regulates the whitening filter intensity.

$$X_{NEW} = \boldsymbol{W}\ \cdot\ \boldsymbol{X}_{OLD} = \sqrt{(\boldsymbol{D} + \varepsilon)^{-1}}\, \boldsymbol{P}^{-1}\, \boldsymbol{X}_{OLD} \tag{23}$$

Using higher values of ε may enhance edges between different colors and illumination intensities. Whitening comes with an increased computational cost for training, due to the eigendecomposition performed on a large amount of input data.

Normalization includes subtractive and divisive normalization [76]. In subtractive normalization, the mean of the respective R, G and B values of pixels in the surrounding $n \times n$ area are subtracted from the pixel vector $p_{x,y} = [c_{x,y,red,}\, c_{x,y,green}, c_{x,y,blue}]$. Using a square window of odd dimension n centered at (x, y), the subtractive normalization function for a color component $c = c(x, y, z) :\ z \in \{red, green, blue\}$ is:

$$c_{SUB}(x, y, z) = c(x, y, z) - conv2d_z\ (\ \boldsymbol{G}\ \cdot \boldsymbol{C}_{x,y}\ ) \tag{24}$$

Where G is a Gaussian kernel [77] with $\sum_{ij} g_{ij} = 1$ and $C_{x,y}$ the neighborhood of pixels around (x, y). Three two-dimensional convolution operations are performed for red, green and blue channels. Divisive normalization based on equation 24 is:

$$\begin{gathered} c_{DIV}(x, y, z) = \frac{c_{SUB}(x, y, z)}{\max(\boldsymbol{S_z}, \boldsymbol{M}_z, \theta)} \\ \boldsymbol{S}_z = \sqrt{conv2d_z(\boldsymbol{G} \cdot \boldsymbol{C}^2_{SUB_z})} \\ \boldsymbol{M}_z = \overline{\boldsymbol{S}_z} \end{gathered} \tag{25}$$

Where θ is small threshold to avoid division by zero, S_z standard deviation of the subtractive-normalized area around (x, y) for color component z that is weighted by the Gaussian kernel G. M_z is mean of S_z for the color component z.

2.4.5.6 In-Network Normalizations

Normalization functions can be placed inside the CNN structure and are useful to level the output of the convolution operation to a desired scale. The vanishing gradient problem occurs when the gradients shrink through time, and rectifier activation functions prevent negative input values to the next layers. The exact opposite is the *exploding gradients problem* [48], [78] that may occur when the gradients during training of a CNN grow out of bounds. Due to the lack of upper limit in the range of activation functions like ReLU, the input of the next layers may grow to inappropriately big values. Solutions to scale these values and restrict them into desired bounds may improve the accuracy of the model. The Local Response Normalization (LRN) layer that is placed after a CONV or POOL layer has helped to increase accuracy of the CNN classifier in [3]. The Local Contrast Normalization (LCN) layer which implements the combination of subtractive and divisive normalization [79] replaces LRN in [80]

Batch Normalization (BN) is a recently introduced normalization technique which works on the statistics of image mini-batches that are fed during training [81]. It is not a distinct layer but rather a function inside a convolutional layer. Its placement before [35] or after the activation function [82] with regard to potential gain in accuracy is a matter of debate. It addresses the problem of *internal covariate shift* and minimizes it by providing a whitened input to the next layer. This comes with an increased computational cost for the training process, which is reported in [82] as a 30% overhead.

The latest proposal at the time of writing is Batch Renormalization (BRN) [83] which is a solution to the ineffectiveness of BN when small mini-batches are used. The mean and variance that are used by BN are inaccurately calculated over a small sample count. BRN ensures that normalized activations depend only a single image instead of the whole mini-batch. The method shows increased accuracy over BN in large-scale image classification experiments using a very deep neural network architecture.

2.4.5.7 Weight Initializers

It has been experimentally determined that the initial values of weights in the convolutional kernels have a significant effect on the convergence of the model during training, impacting both the needed epochs and the final accuracy on the model. Large or small weights can have different impact on convergence, due to different magnitudes of the propagated weight updates [82]. Simply choosing random values in a certain interval is ineffective compared to choosing them randomly from a uniform distribution. Better methods have been proposed with the characteristic of keeping the variance of the input constant when it is multiplied by the layer weights. In Glorot and Xavier initializations [84], that are the

name and surname of paper's primary author, values are chosen from a uniform distribution but they are restricted in an interval which takes into account the layer's hyperdepths. The initialization range can be generally expressed as

$$[-\sqrt{\frac{3 \cdot a}{d_{INPUT} + (a-1) \cdot d_{OUTPUT}}}, \sqrt{\frac{3 \cdot a}{d_{INPUT} + (a-1) \cdot d_{OUTPUT}}}] \quad (26)$$

where d_{INPUT} and d_{OUTPUT} are the respective input and output hyperdepths for the layer. When $a = 1$ the limits of Xavier initialization $\sqrt{\frac{3}{d_{INPUT}}}$ are used and when a = 2 the Glorot initialization $\sqrt{\frac{6}{\cdot d_{INPUT} + d_{OUTPUT}}}$. An additional multiplicative factor β can be used for the formula so that the random weights are restricted in the interval:

$$[-\sqrt{\frac{3 \cdot a \cdot \beta}{d_{INPUT} + (a-1) \cdot d_{OUTPUT}}}, \sqrt{\frac{3 \cdot a \cdot \beta}{d_{INPUT} + (a-1) \cdot d_{OUTPUT}}}] \quad (27)$$

An improved initializer is proposed in [53] together with the introduction of the PReLU activation function. It is suitable for very deep networks and it is known as known as Kaiming or MSRA, from the primary author and the group of researchers from Microsoft Research Asia. The Random Walk initialization described in [85] takes into account a random walk of the log of norms for the back-propagated errors during training. The Orthonormal initializer [86] is the base for the recently proposed Layer-Sequential Unit-Variance orthogonal initialization (LSUV) [82]. The authors compare LSUV with other initialization schemes and evaluate its use with a variety of activation functions, reporting superior results.

2.4.5.8 Revisiting the existing training techniques.

All these techniques have been experimentally proven to significantly increase the accuracy of the CNN-based classifiers, but at the same time create extra needs for computational resources that might be proven restrictive for real-world applications. If preprocessing is required for an architecture to display good accuracy, the fine-tuning of the learned model on another dataset may also require this. Until now most models are trained on millions of image samples, mainly on the ImageNet dataset [9]. If transferred on similar sized or higher volume datasets, the fine-tuning may reveal that the architecture is not up-scalable, or requires a large number of fine-tuning epochs, that would be equal to training the model from scratch.

A generic approach independent from the statistics of visual information contained in an image dataset can theoretically create universally transferable features. Moreover all the

aforementioned methods have shown impressive benefits only when used with a large count of samples per class in the order of thousands. Using the same techniques to train the same architecture from an initial state on a dataset with a small number of samples per class, overfitting could deteriorate the accuracy of the model. This is experimentally confirmed by the results of this thesis and probably the reason for the poor performance stated in [80] when a state-of-the-art architecture was trained with 30 samples per class instead of 1000.

Ideally the benefits of preprocessing, regularization and normalization should be encapsulated in the architecture of the CNN, the functionality of its layers, their connectivity and the properties of the activation functions. More generally an optimum selection of the model architectural, functional and training hyperparameters should control overfitting and increase the classification performance.

Using in-layer normalizations that include divisions like BN and LCN, need a small threshold in the denominator to prevent divisions by zero. This may introduce numerical instability if a small precision error is accumulated on the last digit of the precision. Popular implementations are using 32bit single precision numbers and such problems have been already reported [87]. However such effects might be concealed from the results of the Softmax output layer and can only be identified in the convolutional layer activations. This makes them non-deterministic and the CNN inappropriate to be used as feature extractor in the CBIR context. This is a problem that was identified during experiments conducted for this thesis and the decision not to include LCN or Batch Normalization was taken.

2.5 State-Of-The-Art CNN architectures

2.5.1 Advances in Computer Vision through ILSVRC

The need of a common testing ground for the various image recognition models has created the idea of a competition on the same dataset that will have a large number of sample images. The PASCAL Visual Object Classes (VOC) challenge has been established in 2005 [88] and gathered up to 19737 images of 20 classes in the PASCAL VOC 2010 dataset. This competition has been superseded by the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) that has been active since 2010 [9]. As its classification benchmark it has assembled a large scale image dataset in the magnitude of a million images for 1000 categories. Held annually, the competition has raised significant attention to the domain of Computer Vision, with participations from key players in the Information Technology (IT) industry. At the time of writing, the last completed challenge was ILSVRC2016 and had 5 tasks: 1) Object classification and localization for 1000 categories 2) Object detection for 200 categories 3) Object detection for 30 categories 4) Scene classification for 365 categories 5) Scene parsing for 150 categories [89].

The past years significant CNN architectures have emerged as winners on ILSVRC, each one representing the state-of-the-art of that time. Even when not competing, novel architectures are benchmarked on the ImageNet dataset in order to display impressive or state-of-the-art results. Usually the solutions competing in ILSVRC deploy an ensemble of networks [90], using different learned states of the same model or completely different CNN architectures that are trained on the same images. They provide a polyphony of output responses for a given sample, which combined can more accurately predict the target class. Having measured the human classification performance on the ILSVRC2012 dataset with an estimated 5.1% top-5 classification error, the CNN that introduced the PReLU activation function was the first to surpass it [53].

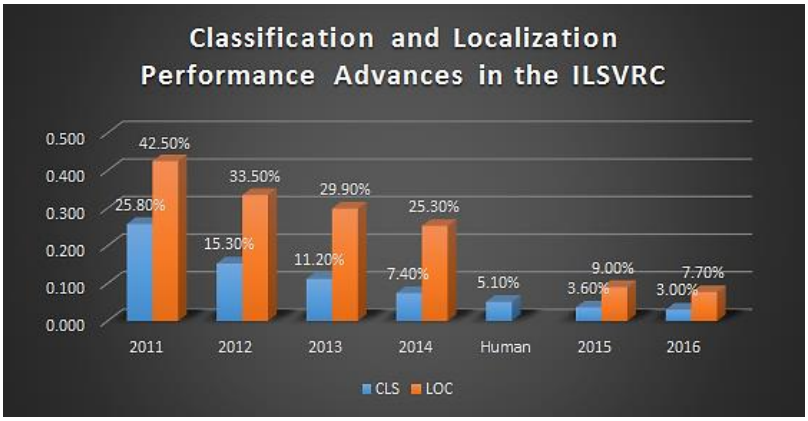

Figure 20: Top-5 classification and localization error through the years of the ILSVRC competition

2.5.2 The CNN architecture evolution through ILSVRC

2.5.2.1 AlexNet (Team Supervision)

The best model for classification in ILSVRC2012 was a CNN that is currently nicknamed as AlexNet, from the primary author of the corresponding paper Alex Krizhevsky [3]. The top-5 classification (CLS) error was 15.3% in both the competition results and the paper, surpassing the second best model that used Fisher Vectors by 9.8%. The same model had the best performance in localization (LOC) with an error of 33.5%. AlexNet was a breakthrough that showed the practical application of deep learning and the clear advantage of convolutional neural networks for image recognition problems.

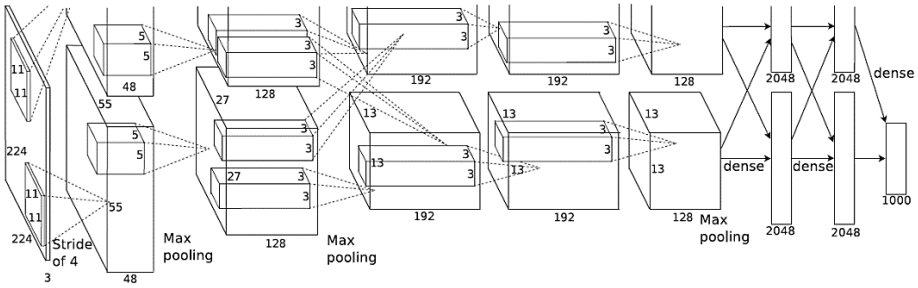

Figure 21: AlexNet architecture, segregated for training on two GPUs [3] .

The activation function used for all convolutional layers was ReLU. The architecture of AlexNet consists of the following modules, described with the notation of section 2.4.1:

Module	Convolutional Layer	Pooling / Normalization /
CONV1 Module	$[11 \times 11 \sim 4 \mid 3 \rightarrow 96]$	MAXPOOL$[3 \times 3 \sim 2] \rightarrow$ LRN
CONV2 Module	$[5 \times 5 \sim 1 \mid 96 \rightarrow 256]$	MAXPOOL$[3 \times 3 \sim 2] \rightarrow$ LRN
CONV3 Module	$[3 \times 3 \sim 1 \mid 256 \rightarrow 384]$	$\rightarrow$ LRN
CONV4 Module	$[3 \times 3 \sim 1 \mid 384 \rightarrow 384]$	$\rightarrow$ LRN
CONV5 Module	$[3 \times 3 \sim 1 \mid 384 \rightarrow 256]$	MAXPOOL$[3 \times 3 \sim 2] \rightarrow$ LRN
FC1 Module	$[12544 \rightarrow 4096]$	DROPOUT 50%
FC2 Module	$[4096 \rightarrow 4096]$	DROPOUT 50%
Softmax	$[4096 \rightarrow 1000]$	

Table 1: Convolutional modules of the AlexNet architecture.

For reducing overfitting it uses data augmentation with random crops and PCA-based random color value alterations. Additionally the Dropout regularizer with 50% probability is used in both FC layers. The AlexNet is considered a reference architecture for many implementations that use CNN models and the pretrained model has already been used for CBIR.

2.5.2.2 ZFNet (Team Clarifai - ZF)

From 2013 and on the ILSVRC training dataset remained the same so that comparisons can be made. The winner architecture for the CLS task is nicknamed ZFNet from the initials of the surnames Zeiler and Fergus, authors of the paper [80]. An ensemble of ZFNet networks presented by team Clarifai had the best test top-5 error of 11.2% in the competition results, an improvement of 4.1% compared to previous year. A different top-5 accuracy is reported in the paper with 14.8% top-5 error for an ensemble of 6 ZFNets, while the same metric for an ensemble of 7 AlexNets was only 0.5% lower.

The contribution of ZFNet was the visualization method that was used in its design. A depiction of the learned features helped the winners understand the inner functionality of the convolutional layers. Gaining insight on the AlexNet architecture, they found that the first 96 features extracted from RGB data in CONV1 module could be improved.

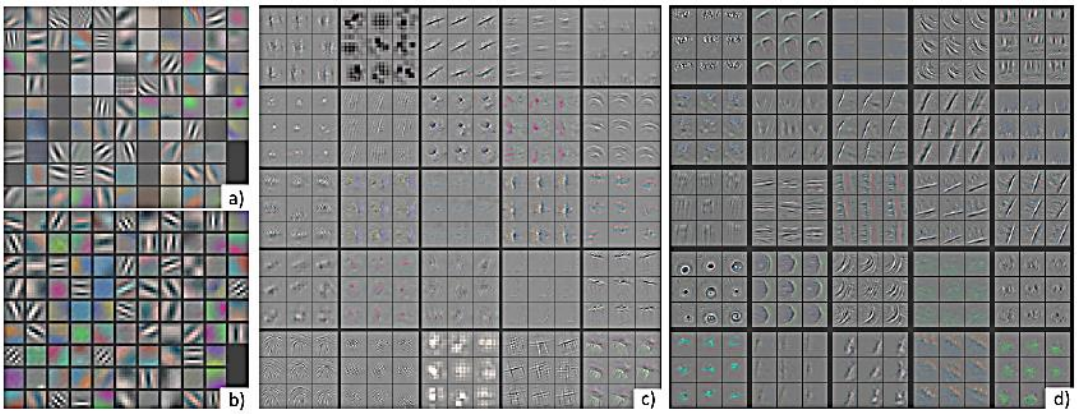

Figure 22: Visualization of convolutional kernel receptive field maps in [80] for (a) AlexNet CONV1 (b) ZFNet CONV1, (c) AlexNet CONV2 and (d) ZFNet CONV2.

To detect problems in the architecture it uses the deconvolutional layer [91], a technique that reconstructs the input of a convolutional module given its activation map. The features in activation maps are projected in the source pixels that originated them and by this technique the feature extractors that do not perform well, are visualized with black or faded receptive field maps. The architecture of ZFNet is almost identical with AlexNet, using ReLU activation functions and optionally LCN normalization instead of LRN.

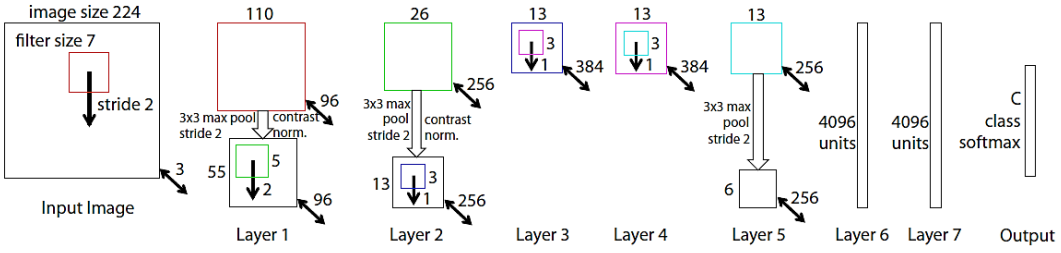

Figure 23: ZFNet architecture. The apparent differences with AlexNet are located in the first two layers with kernels $[7 \times 7 \sim 2|\ 3 \rightarrow 96]$ and $[5 \times 5 \sim 2|\ 96 \rightarrow 256]$ [80].

Module	Convolutional Layer	Pooling / Normalization / Regularization Layer
CONV1 Module	$[7 \times 7 \sim 2 \mid 3 \rightarrow 96]$	MAXPOOL$[3 \times 3 \sim 2] \rightarrow$ LCN
CONV2 Module	$[5 \times 5 \sim 2 \mid 96 \rightarrow 256]$	MAXPOOL$[3 \times 3 \sim 2] \rightarrow$ LCN
CONV3 Module	$[3 \times 3 \sim 1 \mid 256 \rightarrow 384]$	$\rightarrow$ LCN
CONV4 Module	$[3 \times 3 \sim 1 \mid 384 \rightarrow 384]$	$\rightarrow$ LCN
CONV5 Module	$[3 \times 3 \sim 1 \mid 384 \rightarrow 256]$	MAXPOOL$[3 \times 3 \sim 2] \rightarrow$ LCN
FC1 Module	$[12544 \rightarrow 4096]$	DROPOUT 50%
FC2 Module	$[4096 \rightarrow 4096]$	DROPOUT 50%
Softmax	$[4096 \rightarrow 1000]$	

Table 2: Convolutional modules of the ZFNet "narrow" architecture with maximum hyperdepth of 384. The ZFNet "wide" architecture has a hyperdepths of 512 instead of 256 and 1024 instead of 384.

The only architectural differences between AlexNet and the baseline ZFNet are the lower stride and smaller spatial size of CONV1 from $[11 \times 11 \sim 4]$ to $[7 \times 7 \sim 2]$ and the increased stride of CONV2 from $[5 \times 5 \sim 1]$ to $[5 \times 5 \sim 2]$ that achieve the same downsampling ratio. The loss in detail by having a large stride has been testified by the results of ZFNet. Since then, a stride larger than 2 on input pixels was avoided by the state-of-the-art CNNs for input resolutions of 227×227 and 299×299.

It is reported in [80] that an increase on the hyperdepth of convolutional layers from 256 to 512 and from 384 to 1024 can decrease of the top-5 error by an extra 0.5%. This was one of the first indications that increasing the *width* of a deep network, a term used to express to the output hyperdepths of all layers, improves performance.

In transfer-learning experiments, the pre-trained ZFNet was fine-tuned on Caltech101 and Caltech256 datasets. Using the weights learned on ImageNet it achieved 86.5% and 70.6% accuracy respectively. These were state-of-the art for this dataset at that time, thus showing the transferability of the features learned on a large scale image dataset. However re-training the same ZFNet architecture with only 30 samples per class, for each one of the 101 classes of the Caltech101 dataset resulted in only 46.5% accuracy [80].

2.5.2.3 OverFeat

The ILSVRC2013 winner for the LOC task was the model OverFeat with 29.9% top-5 localization error [92]. This model was the first to compete in all three tasks CLS, LOC and object detection (DET) of the 2013 challenge, taking in each one top places. It demonstrated the benefits of an increasing number of features that starts from 96 in the first layer, ending up to 1024 in the last layer. The macro-architecture of OverFeat can be classified together with AlexNet and ZFNet, it also uses ReLU non-linearities, but has a basic difference of not using normalization layers between the convolutional modules.

Module	**Convolutional Layer**	**Pooling / Normalization / Regularization Layer**
CONV1 Module	$[11 \times 11 \sim 4 \mid 3 \rightarrow 96]$	MAXPOOL$[2 \times 2 \sim 2]$
CONV2 Module	$[5 \times 5 \sim 1 \mid 96 \rightarrow 256]$	MAXPOOL$[2 \times 2 \sim 2]$
CONV3 Module	$[3 \times 3 \sim 1 \mid 256 \rightarrow 512]$	
CONV4 Module	$[3 \times 3 \sim 1 \mid 512 \rightarrow 1024]$	
CONV5 Module	$[3 \times 3 \sim 1 \mid 1024 \rightarrow 1024]$	MAXPOOL$[2 \times 2 \sim 2]$
FC1 Module	$[36864 \rightarrow 3096]$	DROPOUT 50%
FC2 Module	$[3096 \rightarrow 4096]$	DROPOUT 50%
Softmax	$[4096 \rightarrow 1000]$	

Table 3: Convolutional modules of the OverFeat architecture.

The apparent increasing factor of the features hyperdepth from the first to the last convolutional layer showed that this can benefit performance not only for classification but for other tasks. An interesting point is the output hyperdepth $h_{CONV5} = 1024$ that was the largest presented until that time. An additional characteristic is the use of non-overlapping pools in the MAXP layers of the OverFeat CNN.

2.5.2.4 *Inception version 1 (Team GoogleLeNet)*

The winners of the ILSVRC2014 CLS and LOC tasks showed that deeper CNN architectures, with more than 8 layers of the previous years, can yield better performance. The GoogleLeNet team from Google Inc. showed also the corporate interest for CNN application. They presented significant architectural proposals with the Inception architecture [34], using a 22 layer network that scored a CLS top-5 error of 6.7%, lowering the best performance of ZFNet by 4.5% and in comparison with AlexNet by 8.5%. Since the first ILSVRC2014 version the architecture has advanced at the time of writing to Inception version 4.

It introduces a new type of convolutional module the Inception module, which includes an implementation of the NiN as a 1×1 convolutional layer, the parallel operation of kernels with different spatial sizes of 1×1, 3×3, 5×5 and a feature concatenation as the output of the module. The 1×1 convolution is used for dimensionality reduction, producing a smaller hyperdepth of features before the computationally expensive convolutions with greater spatial sizes. There are two types of Inception modules type-A and type-B that are stacked onto each other to form a very deep network. In both types there is a combination of max pooling and convolutional layers with different kernel spatial sizes, which are operating in parallel on the same input.

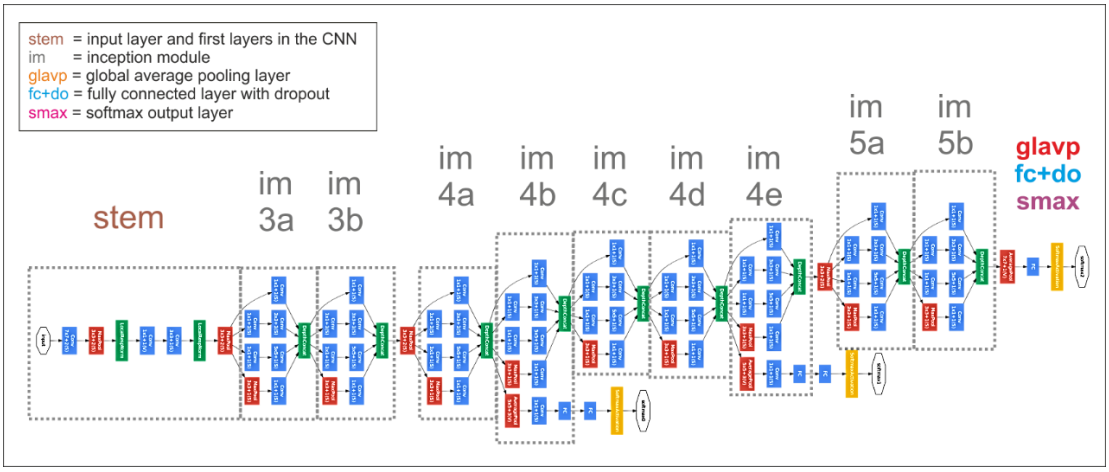

Figure 24. Inception version 1 (GoogleLeNet) macro-architecture [34]. The first part is the stem that starts from the input layer up to the first max pooling layer. Tree stacks (3, 4, 5) of Inception modules (im) are found before a global average pooling (glavp), a fully connected linear layer with dropout 40% (fc+do) and the Softmax output layer (smax).

As seen in the macro-architecture tree stacks of Inception modules 3, 4 and 5, have in-between two MAXP layers for spatial reduction. After the Inception 5b module there is a GLAVP as proposed in NiN and the average of each feature activation map is fed to the units of a FC linear layer that is trained with 40% dropout keep probability. The first part of the Inception macro-architecture can be consider as the *stem* into a very deep and branched hierarchy of convolutional layers. A definition for the CNN stem would be a stack of

convolutional and pooling layers used to extract the first features from pixel data and to reduce the spatial resolution of the image. In the Inception-v1 stem, the spatial size of the image is reduced from the input spatial size 224×224 to the internal representation size 28×28. Two more MAXP layers with stride 2 are used to halve the spatial resolution, before the Inception 4a and Inception 5a modules, resulting in a final representation size $[\, 7 \times 7 \mid 1024 \,]$ before the GLAVP layer.

Subsystems	Layers / Modules	Pooling / Normalization / Regularization Layer
Stem	1: $[7 \times 7 \sim 2 \mid 3 \rightarrow 64]$ 2: $[3 \times 3 \sim 1 \mid 64 \rightarrow 192]$	After both convolutional layers $\mathrm{MAXPOOL}[\, 3 \times 3 \sim 2]$
Inception 3a	$[\mathrm{U} \sim 1 \mid 192 \rightarrow 256]$	
Inception 3b	$[\mathrm{U} \sim 1 \mid 256 \rightarrow 480]$	$\mathrm{MAXPOOL}[3 \times 3 \sim 2]$
Inception 4a	$[\mathrm{U} \sim 1 \mid 480 \rightarrow 512]$	
Inception 4b	$[\mathrm{U} \sim 1 \mid 512 \rightarrow 512]$	
Inception 4c	$[\mathrm{U} \sim 1 \mid 512 \rightarrow 512]$	
Inception 4d	$[\mathrm{U} \sim 1 \mid 512 \rightarrow 528]$	
Inception 4e	$[\mathrm{U} \sim 1 \mid 528 \rightarrow 832]$	$\mathrm{MAXPOOL}[3 \times 3 \sim 2]$
Inception 5a	$[\mathrm{U} \sim 1 \mid 832 \rightarrow 832]$	
Inception 5b	$[\mathrm{U} \sim 1 \mid 832 \rightarrow 1024]$	
Global Average Pooling	$[\rightarrow 1024]$	
FC Linear	$[50176 \rightarrow 1024]$	DROPOUT 40%
Softmax	$[1024 \rightarrow 1000]$	

Table 4: Convolutional subsystems of the Inception version 1 architecture. The U sign indicates a concatenation between activation maps of convolutional kernels with different spatial sizes.

The Inception architecture incorporates the proposal of OverFeat, gradually increasing the hyperdepths of layers starting from 3 up to 1024. If we consider that Inception-v1 CNN is both deep and gradually increases its width, its *geometry* can be imagined as an inverse pyramid volume. The convolutional layers use the ReLU activation function and during training the images are augmented with the photometric distortion technique [34].

2.5.2.5 VGG19 (Team VGG)

The second best for the CLS task in ILSVRC2014 with a top-5 error of 7.3% , was at the same time the best model in the LOC task with a 25.3% top-5 error, introducing the concept of very deep convolutional networks using only convolutional kernels of 3×3 dimensions. The model of Visual Geometry Group (VGG) from the University of Oxford has a depth of 19 convolutional layers and a hyperdepth of 512 features at the last CONV layer [93]. It has two FC layers before the SMAX layer. There are convolutional subsystems which contain multiple layers of equal hyperdepths. In each subsystem a downsampling is performed using a MAXP layer with stride 2 and non-overlapping $[2 \times 2]$ pools. The classifier has the typical structure of two FC layers before the SMAX layer.

The convolution operation runs for every pixel or activation value of the input maps on either 3×3 receptive fields or 1×1 when they incorporate the NiN concept. According to the authors of [93], two stacked 3x3 convolutions with a stride 1 could cover an effective receptive field of 5×5 and three cover a 7×7 region. This stack can offer a reduction of the needed parameters in RFMs when compared to kernels of larger spatial sizes like $[11 \times 11]$ or $[7 \times 7]$ used respectively by AlexNet and ZFNet. Furthermore it can regularize large maps forcing them to be decomposed into successive smaller.

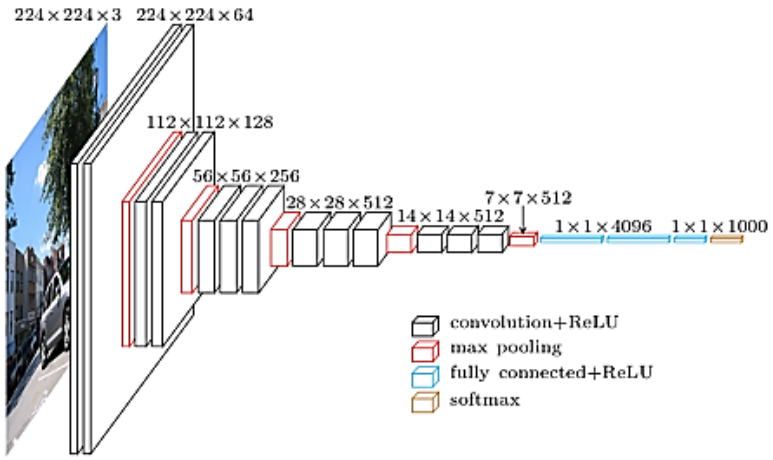

Figure 25. VGG16 macro-architecture (image from [94])

The type E of the VGG architecture contains 19 layers of learnable weights organized in 5 subsystems, hence the name VGG19. Each subsystem in a VGG architectures have a stack of 2, 3 or 4 CONV layers followed by a MAXP for spatial reduction. The macro-architecture details of VGG19 are presented in the following table:

Subsystems	Convolutional Layers	Pooling / Normalization / Regularization Layer
CONV1 Subsystem	1: [3 × 3 | 3 → 64] 1: [3 × 3 | 64 → 64]	After 2nd convolutional layer MAXPOOL[2 × 2 ~ 2]
CONV2 Subsystem	1: [3 × 3 | 64 → 128] 1: [3 × 3 | 128 → 128]	After 2nd convolutional layer MAXPOOL[2 × 2 ~ 2]
CONV3 Subsystem	1: [3 × 3 | 128 → 256] 3 ×: [3 × 3 | 256 → 256]	After 4th convolutional layer MAXPOOL[2 × 2 ~ 2]
CONV4 Subsystem	1: [3 × 3 | 256 → 512] 3 ×: [3 × 3 | 512 → 512]	After 4th convolutional layer MAXPOOL[2 × 2 ~ 2]
CONV5 Subsystem	4 ×: [3 × 3 | 512 → 512]	After 4th convolutional layer MAXPOOL[2 × 2 ~ 2]
FC1 Module	[25088 → 4096]	DROPOUT 50%
FC2 Module	[3096 → 4096]	DROPOUT 50%
SMAX	[4096 → 1000]	

Table 5: Convolutional subsystems of the VGG19 architecture.

An incorporation of 1×1 convolution is mentioned for VGG16 type C architecture in the original paper [93] and this can be considered another CNN in the NiN meta-group. Furthermore it uses L_2 regularization during training and the same data augmentation techniques with AlexNet. The hyperdepth of layers have an increasing ratio, until $h = 512$ where only $512 \rightarrow 512$ feature transformations are done in the last 5 / 7 layers, for VGG16 / VGG19 respectively. The geometry of VGG networks can be imagined as a reverse pyramid volume at the first layers and a cuboid at the last layers.

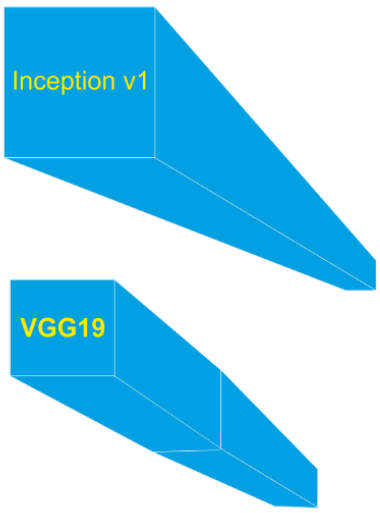

Figure 26: Conceptual CNN geometries of Inception-v1 and VGG19.

2.5.2.6 ResNet-152 (Team MSRA)

For a second consecutive year the introduction of a novel proposal determined the winner for classification and localization tasks. In ILSVRC2015 the team from Microsoft Research Asia introduced the Deep Residual Networks, or in short ResNets [35], in an ensemble that scored a CLS top-5 error of 3.57% significantly better than the measured human performance on the ILSVRC test dataset which is 5.1%. It also topped the LOC task with a localization error of 9.02% and the DET task with mean average precision 62.07%. Even though the network depth increases drastically up to 152 layers the amount of needed parameters are a fragment of these calculated for a corresponding VGG architecture.

Increasing the depth of the network to improve accuracy brings in focus the vanishing/exploding gradients problem. It was detected that there is a certain depth of network that reaches the peek accuracy and after that it starts to degrade, not because of overfitting, but by the excess of depth leads that leads to higher training errors. Following the thought that a deeper model should produce no higher training error than its shallower counterpart, the residual connections are introduced as the major functional change of ResNet. They are a type of *shortcut connections* [95] that feeding the values of layer L_n to the layer L_{n+1+a} where $a > 0$, thus skipping the next a layers.

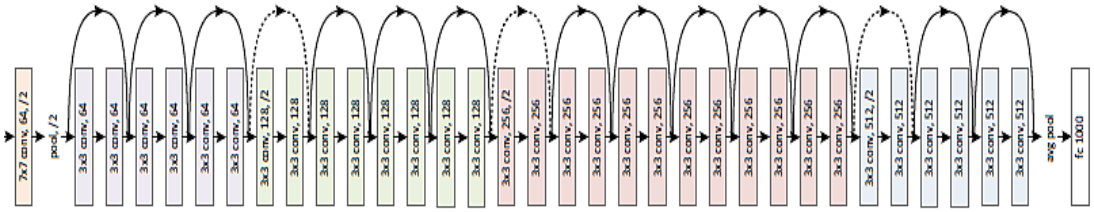

Figure 27: The ResNet deep residual convolutional neural network with 34 layers [35]. Arrows indicate the shortcut connections between layers that are used to implement the residual functionality.

In the residual convolutional module the shortcut connection bypasses two layers, but also performs *identity mapping* [96], by adding the input x to the convolution output before the activation function. The generalized function of the residual block is:

$$\begin{aligned} \boldsymbol{X} = \boldsymbol{Y}_n = f_n(\boldsymbol{X}_n) = relu(conv2d(\boldsymbol{G}_n \cdot \boldsymbol{X}_n) + \boldsymbol{B}_n) \\ resmod(\boldsymbol{X}) = relu\big(conv2d(\boldsymbol{G}_{n+2} \cdot \big(f_{n+1}(\boldsymbol{X})\big) + \boldsymbol{B}_{n+2}) + \boldsymbol{X}\big) \end{aligned} \tag{28}$$

The ResNet utilizes Batch Normalization [81] before each activation function and the function is be modified as:

$$resmod(\boldsymbol{X}) = relu\Big(bn\big(conv2d\big(\boldsymbol{G}_{n+2} \cdot \big(f_{n+1}(\boldsymbol{X})\big) + \boldsymbol{B}_{n+2}\big) + \boldsymbol{X}\big)\Big) \tag{29}$$

There are two types of residual modules the normal block and the bottleneck block that uses 1×1 convolutions. The bottleneck block reduces the dimensionality of the input features

with a $[1 \times 1 \,|256 \rightarrow 64]$ kernel in order to perform a computationally inexpensive $[3 \times 3 \,|64 \rightarrow 64]$ and then re-increase it with $[1 \times 1 \,|64 \rightarrow 256]$. Spatial downsampling is done by convolutions with a stride of 2.

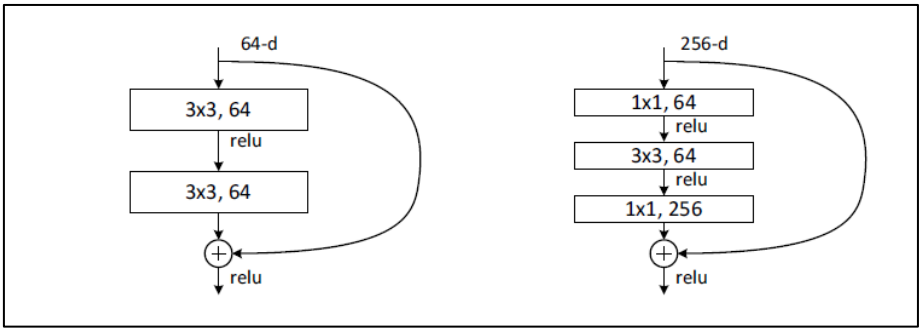

Figure 28: Two types of residual building blocks without batch normalization, the one in the right is the "bottleneck" block [35].

For training, it follows the same preprocessing and data augmentation with AlexNet, without using dropout a weight decay of 0.0001 and batch normalization before the activation function. The best architecture has an overall of 152 layers and uses the ReLU activation function. Recently it has been studied the use of ELU with ResNets [97]. A GLAVP is also found in ResNet after the last residual block, followed by a FC linear layer. The macro-architecture of ResNet-152 is:

Subsystems	**Layers**	**Pooling / Normalization / Regularization Layer**
Stem	$[7 \times 7 \sim 2 \mid 3 \rightarrow 64]$	$\text{MAXPOOL}[3 \times 3 \sim 2]$
ResConv 2	$3 \times \begin{cases} [1 \text{ x } 1 \mid \rightarrow 64] \\ [3 \text{ x } 3 \mid \rightarrow 64] \\ [1 \text{ x } 1 \mid \rightarrow 256] \end{cases}$	
ResConv 3	$8 \times \begin{cases} [1 \text{ x } 1 \mid \rightarrow 128] \\ [3 \text{ x } 3 \mid \rightarrow 128] \\ [1 \text{ x } 1 \mid \rightarrow 512] \end{cases}$	Downsampling with convolution stride ~ 2
ResConv 4	$36 \times \begin{cases} [1 \text{ x } 1 \mid \rightarrow 256] \\ [3 \text{ x } 3 \mid \rightarrow 256] \\ [1 \text{ x } 1 \mid \rightarrow 1024] \end{cases}$	Downsampling with convolution stride ~ 2
ResConv 5	$3 \times \begin{cases} [1 \text{ x } 1 \mid \rightarrow 512] \\ [3 \text{ x } 3 \mid \rightarrow 512] \\ [1 \text{ x } 1 \mid \rightarrow 2048] \end{cases}$	Downsampling with convolution stride ~ 2
Global Average Pooling	$[\rightarrow 2048]$	
Fully Connected	$[2048 \rightarrow 1000]$	
Softmax	$[1000 \rightarrow 1000]$	

Table 6: Convolutional subsystems of the ResNet-152 architecture.

2.5.2.7 Inception-ResNet version 2 and Inception version 4

The latest development on CNN architectures is a fusion of the Inception module [34] with the residual connections introduced by ResNet [98]. It has the best published performance on the ImageNet dataset at the time of writing, with an ensemble of networks that has a top-5 error of 3.08%. It is the forth revision on the original idea of GoogleLeNet and the complete architecture details are provided in the paper. It also evaluates the computational cost between architectures with residual Inception modules and those with "pure" Inception modules. The experiments make a direct comparison of the performance between Inception-ResNet-v2 and Inception-v4 showing an insignificant difference of 0.1%. Nevertheless the useful conclusion is drawn that an Inception-ResNet architecture can be trained faster than a regular Inception architecture.

The winner team of the ILSVRC2016 competition was the Trimps-Soushen from the Third Research Institute of the Ministry of Public Security (TRIMPS) of P.R. China. They achieved a CLS top-5 error of 3% and a LOC error or 7.7% setting new top accuracy for this task on the competition. Although no relative paper has been published on their methods, they name their submissions as Ensemble1 to Ensemble6 [89] and state in a short description that their approach is based on the image classification models like Inception, Inception-ResNet, ResNet and Wide Residual Network (WRN) [99], probably an ensemble of them. From the architectural point of view the current state-of-the-art architecture for classification on the ImageNet dataset can be considered the ensemble of 4 CNN networks mentioned in [98], that consists of one Inception-v4 and three Inception-ResNet-v2 CNNs.

2.6 Future Research Directions for CNNs

2.6.1 Evaluation of the Current State-of-the-Art

The improvement in classification and localization performance during the years of the ILSVRC is astonishing. In the years from 2012 to 2015 each winner made an important contribution that resulted in the design of superior CNN models. The latest results may seem that there is no further room for big improvements on the classification task. Nevertheless a question is whether the same performance can be achieved by computational inexpensive models? A second question that arises is whether the major architectural changes and functional improvements are all connected with an increase in the computational cost [100]. An extra computational cost is implied, when lowering the stride in ZFNet, increasing the network depth in VGG, increasing the hyperdepth of features in convolutional layers, parallel operating layers in the Inception Module and functional changes with the residual connections used for the deepest networks.

The success is experimentally proven but no formal system has been used to explain the CNN architecture superiority in image recognition tasks. Thus all models are considered ad-hoc and best architecture can be determined by experimental results or using useful insights like the visualization in [80]. In order to achieve the best accuracy, ensembles of networks are used. This might be desirable in a competition but it can be considered an inefficient scheme for real-world applications, because it multiplies the demand for computational resources and increases the amount of needed time for a class prediction. Additionally the state-of-the-art CNN solutions use non-neural methods to boost their performance like normalization and whitening of the input data before training or batch normalization inside their structure. This increase of the computational cost [82] may not be desirable for applications that seek to minimize it in order to use low-end hardware.

For achieving invariance qualities they extensively use techniques of dataset augmentation, thus instructing the model to identify an object in different positions or different colorizations. As a consequence substantially more time is needed for each training epoch that may also be needed during fine-tuning on a different image dataset. For an autonomous robotic agent learning through its operation in real-world conditions this time may be considered significant.

The complexity and increased depth of the best models like ResNet and Inception raises the memory demands. Low-end devices in the IoT have a very small amount of memory compared with these that can host a state-of-the-art architecture. Moreover their complexity has also an impact in the amount of time needed to train the model. The recall

time can be also undesirably long and for efficiency the model needs to be hosted in a computational cloud. The last two factors prohibit a near real-time and independent functionality of an image recognition system, due to the total round-trip time needed to transmit a high resolution photo, recall it in an ensemble of CNN networks and receive the output response.

The improvement due to architectural choices themselves can be a matter of debate. The connection of increased depth and increased accuracy was questioned by the emergence of the vanishing/exploding gradients problem in the very deep CNNs [35]. The residual connections that address this problem, showed minimum performance increase when used inside Inception architectures [98]. The primary author of the original GoogleLeNet-Inception paper suggests in [100] that the width and the depth of network should be balanced together in order to reach optimal performance for a fixed computational cost.

Two standing questions remain. Which is are the factors that combined contribute in the optimal CNN design? It must be studied whether the architectural choice, the learning process, the data themselves or the various performance increase techniques make the difference. The second question is how to determine which CNN architecture is a better visual features extractor, rather than the best capture of the dataset statistics that is beneficial for classification?

2.6.2 Open Research Subjects for Deep Convolutional Neural Networks

In order to address these questions each CNN architecture must be evaluated in all image recognition tasks and CBIR. An analogous performance in all tasks can be a clear indication of the quality of its design. Including a CBIR task in an annual large scale image recognition contest will provide a good testbed for research efforts and major improvement may emerge.

The significant problems that future research must overcome are overfitting, bias shift and the problems of exploding/vanishing gradients, which only recently have been formally addressed for CNNs [35] [50]. Further research is needed to fully control the effect of these problems. The demands to fit a CNN into smaller memory and operate faster, can drive research into finding models with less parameters that may perform similarly with larger models. Faster learning methods that may also convergence to better learning states, can be the subject of a research, since they have already been already connected with improved performance of a CNN model.

A fundamental problem of neural networks that is significantly enhanced in state-of-the-art CNN architectures is that they don’t have the ability to generalize when then

number of examples per class is small. This has been initially detected by the ZFNet experiments on the Caltech101 dataset, comparing its ImageNet-pretrained state with fine-tuning and an initial state trained from scratch using 30 images per class [80]. The same architecture had 86.5% state-of-the-art accuracy when fine-tuned on Caltech101, while only 46.5% when trained from scratch. That leads to the simple deduction that state-of-the-art CNN architectures can discriminate between hundreds of classes only if a large number of samples per class is available during training. On the contrary, the biological function of human visual perception needs only a small number of archetypical images of an object to sufficient distinguish it from others or relate it to visually similar objects.

2.6.3 Alternative Directions

Three interesting alternative directions exist for creating the next generation of CNN networks. The vast design space that is defined by the various CNN hyperparameters, could be searched in order to find the optimal models. Methods of *Design Space Exploration* (DSE) have been proposed that include Evolutionary Algorithms [101], Bayesian Optimization [102], Random Search [103], Response Surface Modelling [104] and Reinforcement Learning [105]. Generalizing this idea, machine learning could be used to discover the optimal machine learning models. This direction has a great potential for discovering new CNN models in an automated fashion.

The second category of research aims for less computationally expensive solutions. Model compression is a recently introduced methodology in the domain of CNNs, starting in [106] where the parameters of AlexNet were decreased by a factor of $9\times$ and the parameters of VGG parameters by a factor $13\times$, both without loss of accuracy. This has further improved in Deep Compression [107] with a reduction of $35\times$ for AlexNet and $49\times$ for VGG. It employs pruning of weights, quantization for smaller bits of the internal representation and compression using the lossless Huffman Coding [108].

A simple approach is to design the CNN with a minimalistic strategy. Less computationally expensive 1×1 convolutions, smaller hyperdepths in the early layers, downsampling at a later stage and small depth of the architectures are proposed by the SqueezeNet [109]. It introduces the *Fire* convolutional module with a $[1\times1\,|\rightarrow h_1]$ "squeeze" layer that feeds into two parallel "expand" layers, EXPAND1:$[1\times1\,|\rightarrow h_2]$ and a EXPAND2: $[1\times1\,|h_3]$. The hyperdepths for each fire module are $h=h_3=h_2$ and $h_1=h/4$ with as small number of $h\in\{64,128,192,256\}$. Also it uses a GLAVP layer, and dropout of 50% between consecutive convolutional layers. This results into $50\times$ reduction of memory requirements compared to AlexNet without loss in accuracy. Combining SqueezeNet

with model compression it can create a $510\times$ smaller CNN with the same performance as AlexNet.

The third category tries to create models that will function according to human perception, where various generalizations are made by the visual features that are detected at first sight of an object. One-shot learning [110] introduced the concept of training a model with as low as one sample per class for predicting unknown samples that belong to it. During epochs of training, the knowledge already obtained by the model for other classes is exploited to help it generalize other newly presented categories. It encapsulates this knowledge in a prior probability density function and given one training sample it generates the posterior probabilities. The initial Bayesian approach is ported into the Deep Learning domain with a generative DBN in [111]. There are already proposal towards the direction of porting one-shot learning into a CNN model, namely the Siamese Neural Networks in [112] and the Matching Networks [113]. One-shot learning could benefit CNN models by making them more scalable, not requiring fine-tuning with a fixed count of categories

3 *Describing Images in a Language of Visual Features*

3.1 Introduction

The early attempts to create image retrieval systems used textual metadata for images. This process involved human experts that created image annotations [114]. It was initially difficult to find an image inside a collection, using only pixel data for the search query and exploiting the visual information inside them, such as color, shape or texture. Currently there are many efforts for localizing an object that dominates a scene and detect it amongst others to retrieve relevant images. The content-based image retrieval (CBIR) task can be defined as the retrieval of images which uses only visual information and operates on the basis of visual relevance. That implies a method by which the relevant image is retrieved using its distinctive features, without explicit annotations about objects inside it or the category that it belongs. The definition can be extended for images of a concept, like love, beauty, speed, hot. A CBIR system should retrieve the concept's visual representations or snapshots of its real-world manifestations. This chapter explains the recently introduced methods of Deep Learning CBIR (DL-CBIR) and specifically the CNN-based CBIR. In section 3.2 the connection of image retrieval with text retrieval is drawn by the Bag-of-Visual-Words (BoVW) model that is generalized by this thesis with the Visual Features Language (VFL). The standard approaches and current state-of-the-art features for CBIR are mentioned along with the baseline methodology. Section 3.3 presents the CNN-based CBIR with an extensive review of existing bibliography, in order to identify issues for research. The chapter ends with section 3.4, where an alternative bio-inspired approach is considered for image recognition and CBIR. Its foundations are presented in connection with the human visual system, whose basic physiology is briefly described.

3.2 Content-based Image Retrieval using a Visual Features Language

3.2.1 Image Retrieval based on Visual Features

The term *Query By Image and video Content* (QBIC) was a synonym to Content-Based Image Retrieval (CBIR) in the early days, with systems like IBM's QBIC [115] and Photobook from MIT [116]. A massively used image search engine which evolved into CBIR is Google Image Search, the most popular image retrieval system in the web 2.0 era. Inspecting the evolution of this search engine, there are noticeable improvements, probably due to advances of its underlying image retrieval technology that is not open to public. CBIR was initially performed on image color statistics, like a global color histogram, region based dominant colors and multi-resolution histograms [117]. In addition to color, features for texture and shape has been proposed in an early stage [118]. A vector that contains feature values for a region of the image is the region's *feature descriptor* and the vector that represents features for the whole image a *global descriptor.*

Texture feature descriptors were initially modeled using the mathematical properties of the wavelet transform [119]. Global texture descriptors may consist of features like fractal dimension or circular Moran autocorrelation function, which both measure the surface roughness [120]. Additional methods measure the coarseness of the texture grain, use entropy to measure the lack or order or presence of symmetry, or use the statistics of brightness fluctuations within a region. The Tamura texture features are notable for their organization according to human visual perception with concepts like coarseness, contrast, directionality, line-likeness, regularity and roughness [121]. [117]. An effort to systematically organize visual features in a standard, created the MPEG-7 standard [120] coded as ISO/IEC 15938-1:2002 [122].

Other early approaches in *feature engineering* used formal methods that include shape similarity measures, image segmentation, corner point and interest point detection. Gabor features [123] are used in a filter bank [124] that is applied on an image to create a histogram of the filters' output for an image region. They can be applied in different scales and directions on the same region and typically the mean and standard deviation of the responses are used for image retrieval [120].

Features are considered invariant to specific label-preserving transformations of an image, if their values do not deviate when such transformations are applied to its pixels. The basic characteristic of a label-preserving transformation is that it does not distort the

visual information in such a way that could alter the human perception of the image. Changes in color contrast, rotation and scaling of an object are such transformations. Invariant features are very useful for CBIR and the Scale-Invariant Feature Transform (SIFT) method [125] that is presented in section 3.2.6 is the most common approach.

3.2.2 Text Retrieval and the Bag-of-Words

The task of information retrieval can be simply described as finding information in a collection of documents that satisfies the information needs of a query. Visual information in documents that is processed by the human sense of sight, can be split to two main categories, text and pictures [126]. Text information are represented as a sequence of characters which can be considered features of the natural language document. The various encodings like ASCII, UTF-8 or UCS-2 are a set of integer values for the digital representation of these character-features. In natural languages letters, syllabograms or pictograms create words, words create sentences and sentences fill text documents. A text query may contain words, sentences or entire documents and the text retrieval operation will result in a set documents which share similarities with the query.

In a natural language some words like prepositions or discourse markers can be found more frequently than others. The Zipf's Law [127] is based on the frequencies of all words inside a natural language that are sorted by descending order and describes the probability of occurrence in the language as inverse proportional of its position in the order. The graph of the *Zipfian* distribution provides an intuition that frequency can be used as a basis to build suitable retrieval methods for text.

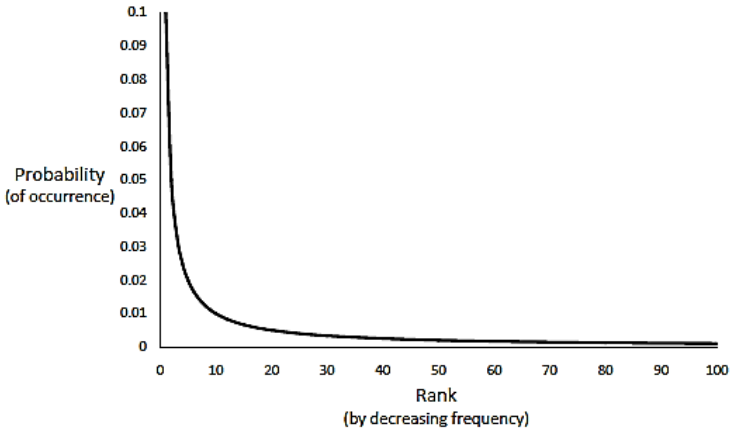

Figure 29: Zipf's Law [128]

We define as the smallest block of information that is meaningful in a context a term. Both the given query and the documents in the collection are lists of terms. If the terms are words in a natural language and their position in the list has no importance in the retrieval task, we can think the word list as a *Bag of Words* (BoW). Usually a simple histogram of word

occurrences is used by the BoW models to represent both the document and the query as unordered lists of terms. The BoW model is used in CBIR for the convenience of an orderless local visual features representation and not for creating global image histograms.

3.2.3 Vector Space Model for Information Retrieval

The BoW abstraction for text information can be used in retrieval models like the *Vector Space Model* (VSM), that is conveniently simple and sufficiently effective for using in both text and image retrieval [128]. In this model a document D can be represented by vector of weights for each word-term in the list. For a document with n terms:

$$D = [d_1, d_2, \dots, d_n] \tag{30}$$

The documents collection C can be a two dimensional matrix that holds term weights for each document. For m documents in the collection and n unique terms in it, the dense matrix is:

$$C = \begin{bmatrix} D_1 \\ D_2 \\ \dots \\ D_m \end{bmatrix} = \begin{bmatrix} d_{11}, d_{12}, \dots d_{1n} \\ d_{21}, d_{22}, \dots d_{2n} \\ \dots \\ d_{m1}, d_{m2}, \dots d_{mn} \end{bmatrix} \tag{31}$$

A query Q with t terms in the BoW is also a vector of its orderless term weights:

$$Q = [q_1, q_2, \dots, q_t] \tag{32}$$

The retrieval method returns a set of R relative documents using a function that calculates a similarity measure between query Q and the documents in the collection C. The cosine similarity is a typically used measure in the VSM, but alternative measures can be used like soft cosine similarity [129]. Cosine similarity between a query and each document in the collection is expressed as:

$$f(Q, C) = cosine(Q, D_i) = \frac{Q \cdot D_i}{\|Q\| \cdot \|D_i\|} = \frac{\sum_{j=1}^{t} q_j \cdot d_{ij}}{\sqrt{\sum_{j=1}^{t} {q_j}^2 \cdot \sum_{j=1}^{t} {d_{ij}}^2}} : D_i \in C \tag{33}$$

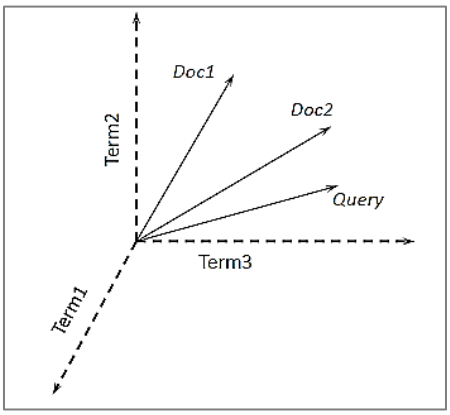

Figure 30: Vector Space Model (VSM) representation for 3 term-dimensions of documents. There is 1 query and 2 documents expressed by the 3D vectors [128]

The term weights should reflect the importance of each terms inside a document or query and for that the term frequency (TF) is calculated. It is the count of the occurrences f_x of a term x in a document D_i normalized by the sum of occurrences of t terms in the document, which are used as dimensions in the VSM.

$$w_{ix} = \frac{f_{ix}}{\sum_{j=1}^{t} f_{ij}} \tag{34}$$

It has been experimentally shown that using the logarithm of the count is more effective than a normalized count. The weights of a term x in a document $D_i \in C$ or a query Q_i can be expressed as the term frequency:

$$tf_{ix} = \begin{cases} 1 + \log_{10} f_{ix} & , f_{ix} > 0 \\ 0 & , f_{ix} = 0 \end{cases} \tag{35}$$

Recalling the Zipf's law, some terms are generally more common and can be found in most of documents. They should not bias the weights, by including irrelative documents in a retrieval operation due to the presence of common terms. The most common are discarded at an early stage and are not expressed as dimensions of a document in the VSM.

To build a more discriminative weight, the significance of the term in the entire collection should be taken into account. For this reason the calculation of the term weights is using the inverse document frequency (IDF):

$$idf_x = \log_{10} \frac{m}{df_x} \ : df_x \leq m \tag{36}$$

Where m the count of all documents in the collection and df_x the document frequency (DF) of the term, or simply the count of documents that contain the term x. The logarithms are used to "dampen" the effect of the inverse DF. The combination of the term frequency and the inverse document frequency is known as TF-IDF that is a common weighting scheme used in the VSM. The TF-IDF weight is calculated for each term in a document D_i as

$$w_{ix} = tfidf_{ix} = tf_{ix} \cdot idf_x = \frac{(1 + \log(f_{ix})) \cdot \log(\frac{m}{df_x})}{\sqrt{\sum_{x=1}^{t} \left((1 + \log(f_{ix})) \cdot \log(\frac{m}{df_x})\right)^2}} \tag{37}$$

The vectors Q and C for the cosine similarity metric will correspond to the $TFIDF_q$ weights for the query terms and the $TFIDF_d$ weights for the document terms:

$$f(Q, C) = cosine(Q, C) = \frac{TFIDF_q \cdot TFIDF_d}{\|TFIDF_q\| \cdot \|TFIDF_d\|} \tag{38}$$

3.2.4 Image Retrieval using Bag-of-Visual-Words

Content-based image retrieval can be considered analogous to text retrieval, using the abstraction that image documents containing visual features are equivalent to text documents containing words. In pattern recognition terms, a text document in the BoW contains words that are descriptors of "linguistic features" containing integer values of the text encoding system. The equivalent for an image is an assembly of orderless *visual words* [130] that each one is a descriptor of visual features for a region, which is a vector of real numbers or a vector quantized to integer values. These represent some aspect of visual information like color, texture, shape that exists in the image. This model is named *Bag of Visual Words* (BoVW) and can use the same methodologies with text retrieval, like the VSM and the TF-IDF weighting scheme.

The problem of CBIR can be now restated as: How to create a BoVW from raw image pixels? Starting from the lowest level, a CBIR system should extract visual features out of local pixel regions. The feature extraction method will use some encoding system in order to represent them, with real or the more manageable integer values. The encoded feature values are the visual words for each image patch. All visual words of an image compose the corresponding *"visual document"* that will be part of the image collection used for CBIR. This process rises additional questions such as: a) which are the best features and how to extract them, b) what will be the shape, size and where is the location of the image regions, c) how to handle the high dimensionality of visual descriptors?

Two terms that are analogous to the abstraction of the BoVW model are the Bag of Features (BoF) [131] and the Bag of Regions (BoR) [132] and a third approach is the Bag of Keypoints (BoK) [133]. The BoVW and BoF can be considered interchangeable terms that refer to an image representation of orderless local feature descriptors. We make a distinction that only BoF is related with a histogram-based image representation. The BoR model could be considered an extension on the BoVW logic, that uses regions of unequal sizes that are created through segmentation [132]. The BoK model is also a histogram of visual words that is used for classification. To disambiguate these approaches a single term is needed as the common ground of all methods that aim in the creation of visual words.

3.2.5 The Visual Features Language (VFL)

This thesis generalizes the various "bag-of" approaches introducing the new term *Visual Features Language* (VFL) to describe a model which organizes visual information contained in images using a set of visual words. A codebook in the BoVW approach is the

vocabulary in the VFL model, which is generated from pixel data of an image collection and contains all the visual words found in a subset of the language. Different languages may occur, either by using different methods on the same image dataset or by using the same method on different data. The *Visual Document (*VDOC*)* is an image representation that contains visual words of different image regions and it is the equivalent of a text document with words in some context. The regions in a VFL model can have different shapes and sizes. Also their proximity in the image can be either taken into account or ignored. The VSM can be used with visual documents and its sparse weight vectors are suitable for large-scale image retrieval using visual words.

According to the above definition, the BoVW/BoF can be considered a special case of a VFL that disregards location of visual words in an orderless representation. The BoF uses histograms of VFL words for image representations. The BoR is also a special case of a VFL model with variable-sized visual words that describe regions of unequal size and variable shape. An alternative histogram representation is the Vector of Locally Aggregated Descriptors (VLAD), which is more compact and better performing than BoF [134] and is derived from the visual words. Histogram-based global image descriptors are not visual documents but can be considered as the statistics of them, like word counts in text documents.

Future work can bring into surface other analogies between VFL image retrieval and natural language text retrieval. These may include, the semantics of a visual word in a context, the creation of similar visual words from root visual words, "grammar functions" on words, composite visual words and other interesting equivalences. Visual semantics might be different from the natural language semantics used for annotating the images. The conceptual *VFL grammar* will resemble the vector arithmetic done by the DCGAN in [37]. Furthermore it would not be a case of learned image-text correlations, like the multimodal neural language models in [135], where image features are learned together with their textual descriptions. Bringing the visual information into the natural language plain through the creation of a universally applicable VFL, may provide solutions for other image recognition tasks.

3.2.6 Handcrafted Feature Extractors

The creation of a VFL begins with determining the appropriate regions in order to extract local features. Regions can be defined at fixed grids, or areas around interest points on the image, which contain the most useful information. One of the most successful feature localization and extraction methods that uses interest points, is the Scale Invariant Feature Transform (SIFT) [125]. It attends the problem of extracting features that are invariant to geometric transformations like scale and rotation.

To detect an interest point SIFT uses the one-dimensional Gaussian function [136] that is convolved horizontally and vertically on different scales of an image, starting from the scale of its pixel resolution. To generate the values for the next level on a pyramid of different image scales, SIFT approximates the two-dimensional Laplace-of-Gaussian (LoG) function [137] by using the Difference-of-Gaussians (DoG) function [138]. To extract the SIFT points of interest, or SIFT key-points, the local minimum and maximum of a 3×3 area is determined by comparing the center to its surrounding neighbors. The peaks correspond to the dominant directions of local gradients [139].

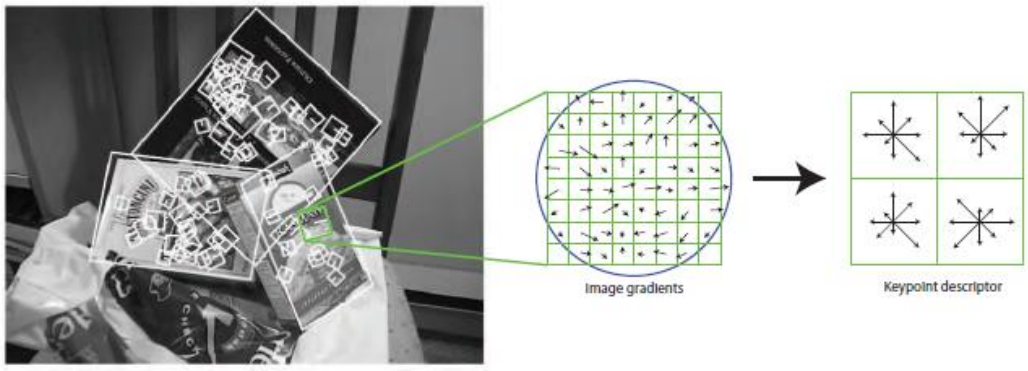

Figure 31: The SIFT region detection, feature extraction and feature descriptor [139]

The SIFT local features are extracted on 16×16 locations around the interest point. For each location there is a 4×4 matrix of histograms, having 8 gradient orientations bins, resulting in a feature descriptor of 128 dimensions for each sampled region [140]. As stated in the original paper, the authors are inspired by the complex cells [42] of the primary visual cortex in human vision and cells in the inferior temporal cortex (IT) [125]. This is an indication of bio-inspired computing beyond neural network approaches. There are several other approaches to detect points of interest [140] like the Harris Laplace corner detector [141] and other popular feature extractors [142] like the Speeded Up Robust Features (SURF) [143] and the Maximally Stable External Regions (MSER) [144]. Except SIFT other feature descriptors which can be classified as state-of-the-art are the HOG [145], DAISY [146] and the recently introduced Distinctive Efficient Robust Features (DERF) [142]. The authors list in the DERF paper mention various "handcrafted" local descriptor methods. Also various interest point detectors have been evaluated for their performance in [147].

Considering the applicability in real-world applications, the localization and feature extraction process should have both performance and computational efficiency. Handcrafted features are suitable for these needs and the combination of the Harris corner detector with SIFT features has already been used in robot applications [148].

3.2.7 Creation of the Visual Vocabulary

The SIFT descriptor of a region contains 128 features values that may be considered high-dimensional. In a collection of known images, a vast count of descriptors of high-dimensionality cannot be encoded using the possible combinations. For this reason, the feature vectors need to be quantized into a discrete space of visual words that will define the entries of the VFL vocabulary. This is equivalent to the BoVW codebook and contains all the valid visual words for a specific image collection. It is a dictionary that indexes the most common combinations of feature values, like the most common words of a natural language as the combination of syllables. The visual words are Voronoi cells of feature descriptors in a multi-dimensional feature space. The typical approach for creation of a VFL vocabulary is to perform clustering on the descriptor set. Each cluster center could correspond to a VFL visual word, or equally a code in the BoVW codebook.

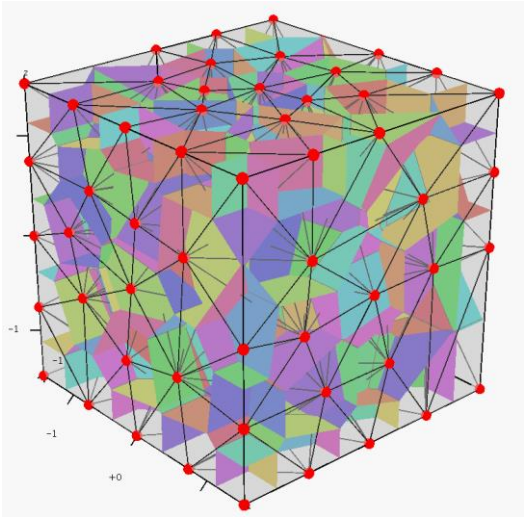

Figure 32: Three-dimensional Voronoi cells generated by TetGen [149].

Various clustering algorithms and methodologies from the field of Data Mining (DM) can be used like the popular k-Means [150] and its variations Hierarchical k-Means (HkMeans) [151] and Approximate K-Means (AkMeans) [152]. Applicable methods are density based clustering like DBScan [153], fuzzy clustering like Fuzzy C-Means (FCM), Support Vector Clustering [154], graph clustering like the Markov Cluster Process (MCL) [155] and neural clustering like SOM [156] . An experimental evaluation of different clustering methods for visual vocabulary generation is found in [157]. Other approaches of codebook creation include Soft assignment [158] and Hamming embedding [159], which allow very large vocabularies. Attempts to build vocabularies in the scale of million visual words include the Approximate Gaussian Mixture Model (AGMM) [160] and Fine Quantization [161] which is on a probabilistic approach. The diversity of visual information may create problems related to the *curse of dimensionality* [162], hence a dimensionality

reduction is usually needed before clustering. Typically amongst linear dimensionality reduction methods [163] based on matrix factorization, PCA is the most popular choice [164].

The k-Means clustering method can be inapplicable for a large count of descriptors since it has a linear complexity of $\mathcal{O}(Mk)$ for k clusters. The thought of using k-Means in a hierarchical manner, which could benefit in both computational efficiency and quality of the clusters, created the variation of Bisecting k-Means (BkMeans) [165]. Its extension the HkMeans clusters the descriptors into k initial clusters and then partitions the data points X according to their parent cluster into $X_1, X_2, \dots X_k$. Each partition is then recursively clustered to k sub-clusters, thus k becomes a constant branching factor in the resulting hierarchy of clusters [151]. Both the HKM and AKM are significantly less complex than simple k-Means with computational costs of $\mathcal{O}(b\,M \log k)$ and $\mathcal{O}(vn\,M \log k)$ respectively [166]. In this work a variant of the HKM will be used for computationally inexpensive clustering and additionally suitable for the characteristics of natural language words. Feature descriptors of unknown images that were not used for clustering are assigned clusters according to their distance from the clusters' centroids, thus quantizing the image.

Recently it has been found that generating a vocabulary by randomly selecting the visual words has similar performance to popular k-Means approaches [167], indicating that the creation of the VFL vocabulary is an open research subject. It is a very crucial point in the CBIR pipeline because the cluster centers must be expressive of the visual feature combinations clustered around them. The retrieval model will fail to generalize if the VFL has a significant amount of bias on the images that were used to generate it. The codebook generation is a complex problem by itself that needs detailed study and it is handled by this thesis in a simplified approach.

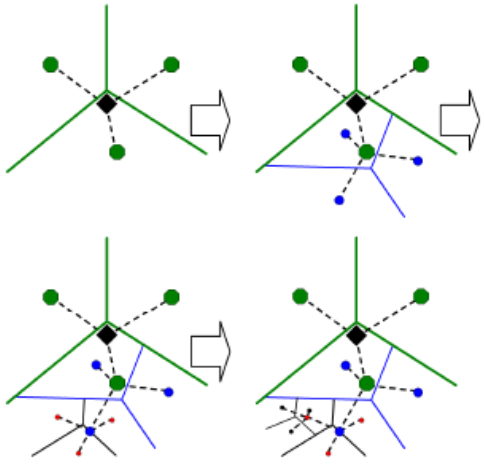

Figure 33: Hierarchical k-Means clustering with branching factor b=3. It runs k-Means clustering for k=3 in each level of the hierarchy depth 1 (top-left), depth 2 (top-right), depth 3 (bottom-left) and depth 4 (bottom-right).

3.3 Using Convolutional Neural Networks for CBIR

3.3.1 The Deep Neural Content-based Image Retrieval Pipeline

The promising results of deep neural networks in various image recognition problems and the success of the CNNs in the ILSVRC competition against non-neural approaches, has created a new kind of CBIR. The change in the usual BoVW process is that features are extracted by a CNN which has been previously trained on image data, thus replacing the handcrafted feature extractors like SIFT.

The activation vector for a specific point inside the 3D convolutional layer activation map is used as its feature descriptor, with high dimensionality depending on the kernel's hyperdepth. The regions of interest have typically a fixed square-shaped size and their center points are evenly arranged forming a grid on the image. This is simplified compared to interest point detection, but has proven efficient in experiments. At the time of writing, a formal reason for the performance of CNNs has not yet been provided. Moreover the quality of these neural features for CBIR is an open issue in this recently introduced sub-domain of research.

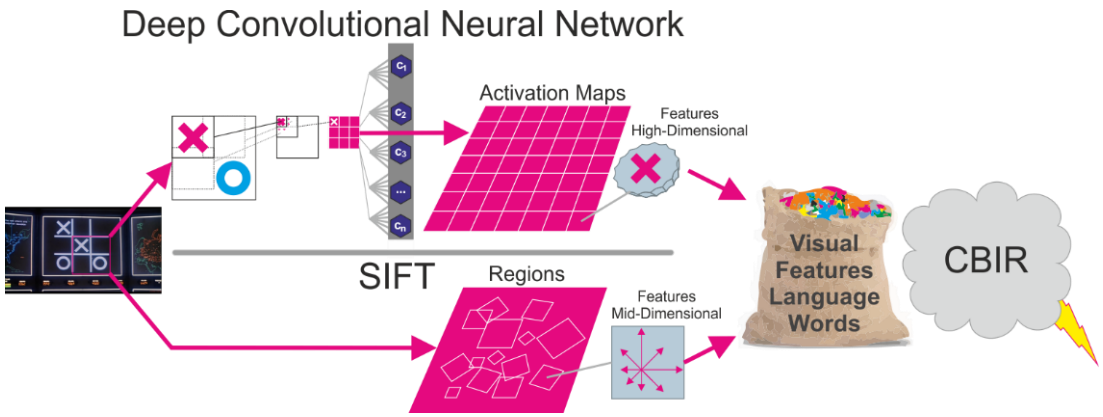

Figure 34: Comparison of CBIR with CNN-based features with classic SIFT feature extraction. They have in common the creation of a Visual Features Language (VFL) and the CBIR model.

The research done in *Deep Neural Content-based Image Retrieval* (DN-CBIR) attempts to discover DNN architectures that will be used as general purposed feature extractors for CBIR. The methods of the codebook creation, indexing and retrieval can remain the same in both CNN-based and SIFT-based image retrieval. Multiple tasks are combined in a pipeline of processes with various methods from the domains of Machine Learning, Data Mining and Information Retrieval. These can be described by the following workflow of steps:

1. Supervised training of a DNN which is typically a CNN. This is performed on an image collection with annotations like class labels. The learned parameters of the network will incorporate the feature extraction capabilities and its layers will act as feature extractors.
2. An image is recalled through the pre-trained DNN. The output activations of a selected layer are the feature descriptors for the image regions. Different layers correspond to different scales of the image resolution.
3. Dimensionality reduction is incorporated in the DNN structure or runs as an additional process using the neural activation vectors as data points.
4. The feature descriptors are quantized to create a VFL vocabulary that is specific for a) the trained state of the DNN architecture, b) the image dataset used for training and c) the selected neural network layer.
5. The image regions are described by visual words of the VFL vocabulary. Each word is a representation of a region's features. All the visual words of the image compose an image representation that can be considered a visual document.
6. The ordering of visual words in the document can be disregarded as in the BoVW or exploited by the information retrieval method.
7. Visual documents can be indexed by a search engine and retrieved by queries, using standard text IR methods.

The tasks of image classification, object detection and object localization can be ported as image retrieval tasks on VFL representations. Retrieving an image of the same class in the top rank can be equivalent to classification. The image areas can be segmented and indexed as different fields in the CBIR system. A fielded search with an image of an object can detect it and localize it inside the returned set of relevant images.

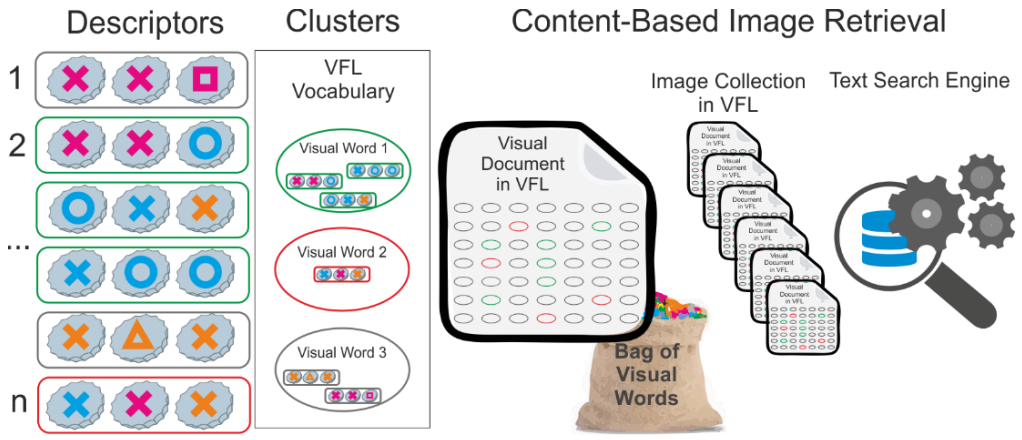

Figure 35: Content-Based Image Retrieval according to the VFL model.

3.3.2 DNN Feature Extractors before ILSVRC2012

Before the success of the CNN deep neural networks at the ImageNet challenge, the related approaches to image retrieval were DBNs (Deep Belief Networks) and Deep Autoencoders (DAs). In one of the first attempts to use deep neural network activations as descriptors, the DBN output was used to create a global image representations using binary codes, without adherence to the BoVW model [168]. In [169] the visual words are formed using a DBN and they are counted in a global co-occurrence vector in an approach that fall to the BoF paradigm. In [170], a PhD thesis concurrent with the ILSVRC2012, DBN features are extracted for classification and evaluated against SIFT, making proposals for the future use of CNN-based features. The use of DAs for CBIR on a large number of images was demonstrated in [171]. A pre-trained DBN was used to initialize and then fine-tune the Autoencoder feature extractor. The authors state their approach as Bag of Batches, which shares similarities with the BoF model. The image retrieval operation in [171] uses the method of Semantic Hashing [172].

3.3.3 Review of CNN-based CBIR

The evolution of CBIR has resulted in the current two branches: a) Retrieval with handcrafted feature extractors b) Retrieval with CNN learned feature extractors. The second branch has two categories with regard to feature extraction [166]. According to the *one-pass* CNN-based CBIR, features are extracted by recalling an image once in order to get the activation 3D tensors from convolutional modules or from fully connected layers. The dimension of the image patch descriptor equals to the hyperdepth of the convolutional kernel. The second category is *two-pass* CNN-based CBIR, where features on a single descriptor are assembled from multiple image patches that are directly pooled to the form a visual word. In both categories the convolutional kernels be considered to extract local or global features depending on the layer that they operate [173]. We will focus in the one-pass CNN-based CBIR approach that can provide a simple implementation of a Visual Features Language.

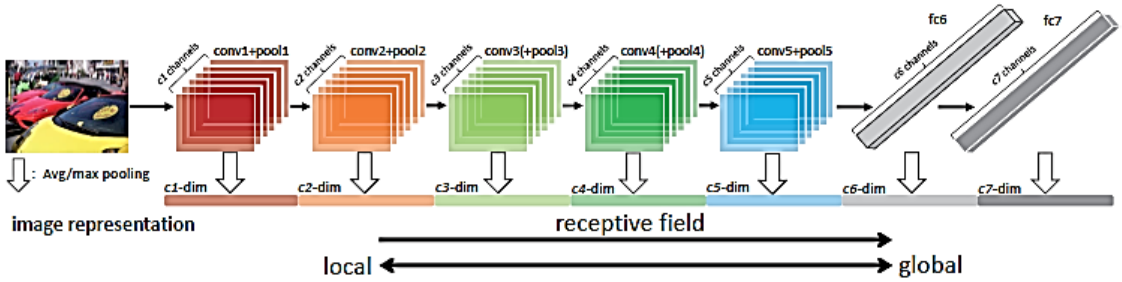

Figure 36: Scale of extracted features in relation to the depth of the layer in a CNN classifier [173].

The use of CNNs as feature extractors has become popular after the ILSVRC2012. Nevertheless it was not until recently that their use in CBIR pipelines was proposed and implemented. In DeCAF [174] the pre-trained AlexNet on the ILSVRC2012 dataset is considered a general purpose feature extractor for image recognition. The extracted features from various layers are evaluated for image retrieval. The idea of *feature fusion* considers extracting CNN activations from multiple layers and aggregating them into descriptors [166] [173]. In [175] features are aggregated for each pixel using a large contextual window of $184\ x\ 184$ pixel around it, according to the concept of *super pixels*. Each descriptor has features from three scales centered on the same pixel and encoded in multiple fields, starting from the highest to the lowest spatial resolutions. In [173] the feature fusion is made by combining activations from different layers that correspond to different image resolutions. Each activation-feature is assigned a different weight, following the score-level fusion scheme of [176].

Geometric feature combinations are performed by SPP-Nets which are custom ZFNet, AlexNet and OverFeat architectures, pre-trained on the ImageNet dataset for classification [177]. It uses Spatial Pyramid Matching (SPM) [178] with a spatial pooling layer after the last convolutional layer. It pools the activations into histograms with a fixed number of spatial bins, regardless of the image size. The proposed scheme was the third best in ILSVRC2014 for the CLS task and the second best for the DET task. Transferring the ImageNet learned-state of the SPP-ZFNet on the Caltech101 dataset and fine-tuning with the standard setup, achieved the current state-of-the art classification accuracy of 93.42% [177].

A CNN-based feature fusion proposal for image retrieval is the multi-scale orderless pooling (MOP-CNN) [179]. It combines features from various layers that are aggregated into VLAD global image descriptors. However it uses small codebooks of 100 clusters and global features by design. In [173] there is an extensive evaluation of features that are extracted from pre-trained AlexNet. The authors propose the use of larger than $224\ \times 224$ image resolutions to increase retrieval performance. They also discuss the use of average and max pooling on the activation values of features and experimentally testify that the CONV5 layer yields better performance than the FC layers.

The pre-trained VGG was used in [180] yielding state-of-art-results on image retrieval datasets like Oxford5k, Paris6k, Holidays, Sculp6K and UKB. The performance gain has extra computational cost due the use of the VGG and the distance metric method which is used for its queries. Its authors report a complexity of $\mathcal{O}(L^3)$ where L is the number of different size of patches used. A complete approach to CNN-based CBIR that illustrates competitiveness of the BoVW model is the Bag of Local Convolutional Features (BLCF)

[181]. It argues that its approach is less computational expensive than VLAD image representation, thus up-scalable for larger collections. The method is competitive to the VGG-based solution on the Oxford5k and Paris6k datasets. Moreover it uses and evaluates IR methodologies like re-ranking and query expansion.

In a recent work the pre-trained AlexNet and VGG networks are used to extract features that are post-processed using PCA-whitening and a method of learned whitening [182]. The two methods are used also for dimensionality reduction. The networks are fine-tuned without supervision and a vast vocabulary of 16 million visual words is created through Fine Quantization [182]. Their increased performance has higher computational costs, requiring the use of learned whitening and the VGG architecture.

3.3.4 Open Research Subjects for CNN-CBIR

At the time of writing all approaches in utilizing CNNs for CBIR have in common a supervised training process on the ImageNet dataset. Increasingly deep architecture yields better retrieval results as illustrated in [182], but in expense of computational resources.

The review of the current state-of-the-art has created an important question: How universal are the CNN features learned from ImageNet data? Despite the large collection of images, images of 1000 objects may not capture the diversity of visual information. The need for extra fine-tuning is present in all attempts to transfer the pre-trained model. Considering the Visual Features Language paradigm, there will be a different language for each different ImageNet CNN architecture. For a specific architecture that is transferred, a fine-tuning of the model in another dataset renders this language invalid, because the clustering process on the new data will generate a totally different VFL. This creates problems with the scalability of the CBIR system that needs to index new images, without rebuilding the model and with an insignificant computational cost. Unknown images may need additional fine-tuning and re-building of the VFL vocabulary that does not serve the aforementioned needs. In the case that no fine-tuning is made the descriptors of the new image may not be assigned proper clusters, resulting in an image representation that will fail to describe its visual features.

The SIFT methodology has a universal function and needs no fine-tuning, not depending on the domain of visual information and the given image dataset. A target for research would be to design and train a CNN architecture, so its layers would act as universally applicable neural feature extractors. Learned CNN parameters contain a certain amount of bias on the training data, indicated by the high overfitting during training. Eliminating this might be part of the solution for a *universal CNN feature extractor*.

The quality of learned features is not yet formally explained nor fully evaluated by experiments. More insight on the CNN internal functionality can lead to decisions that will improve it, something already seen with the design of the ZFNet.

Moreover, the diversity of different codebook generation methods has not been extensively studied with regard to the various types of CNN architectures, training sets of images, selections of convolutional layers and dimensionality reduction techniques. For image retrieval, the potential benefits of having a large VFL vocabulary is a question to be answered. An evaluation of the retrieval performance with different magnitudes of codebook sizes, could decide the best between thousands and millions of unique visual words.

Research focused in the IR domain could evaluate different weighting schemes and models for all the above combinations. Understanding the extreme complexity of the problems that the DN-CBIR research has to solve is the starting point. This thesis aims to unveil the combined aspects of the problem and contribute to the related research with the presentation of novel proposals.

3.4 Bio-inspired Models for Universal Visual Features

3.4.1 Intuition from ZFNet Visualization

An intuition for this thesis is that understanding the biological solutions for vision and importing them as part of a Deep Convolutional Neural Network, may bring additional insight into the design of its architecture. This is supported by the fact that many top performing models and techniques like the CNN itself and the SIFT are bio-inspired. Visualizing the learned feature extractors of AlexNet and ZFNet, an observation can be made with respect to models and patterns in nature. The first 96 features that were learned on trichromatic RGB pixel data, resemble the receptive fields of the directionally sensitive V1 simple cells. Even more interesting is the emergence of two blob detectors for red-like and green-like colors in ZFNet, where in AlexNet there was only one for blobs of red-like color. This indicates that some fine-grained visual features were previously discarded by AlexNet, due to the large stride.

Figure 37: Visualization of the 96 feature detectors of CONV1 layer. Left: AlexNet, Right: ZFNet. The red and green arrows indicate blob detectors of opponent colors in ZFNet [80]

This is a very interesting finding, because CNN as a bio-inspired deep neural network can evolve its kernels through an optimization process, forming receptive field patterns that are found in nature. Nevertheless there are two separate stages in the biological visual processing path, where blobs of light and oriented lines of light are detected. In AlexNet and ZFNet architectures they are both extracted on first convolutional layer which receives the image RGB input.

3.4.2 Inspiration from the Human Visual System

In nature the solution to visual recognition has been found through aeons of evolution and begins at the physiology of the eyes. The eye is an organ that captures photons of different wavelengths that are bounced from objects. Passing through the *pupil* and focused by the *lens*, the light of a scene is properly projected on the *retina* [183], a coating cell tissue in the inner back half of the *bulbus oculi* [184]. The biological retina has a shape of convex hemispherical surface. The human visual system performs feature extraction on the intensity and spectral properties of the captured light and the entire visual perception is built on these primitive features. The equivalent pipeline for Deep Neural Networks can start at the digital image sensor which convert light to digital images and performs feature extraction on RGB pixel values. The trichromatic nature of the RGB color system is built after the knowledge that there are three types of color-sensitive *photoreceptors* in the human retina [185]. The *cone* cells are sensitive to short (S), medium (M) and long (L) wavelengths, respectively to blue, green and red areas of the spectrum, defining the range of the human visible spectrum. Their responses have peaks on specific wavelengths and there are various Neuroscience models to describe the functions and *spectral sensitivities* of the cones. [186]

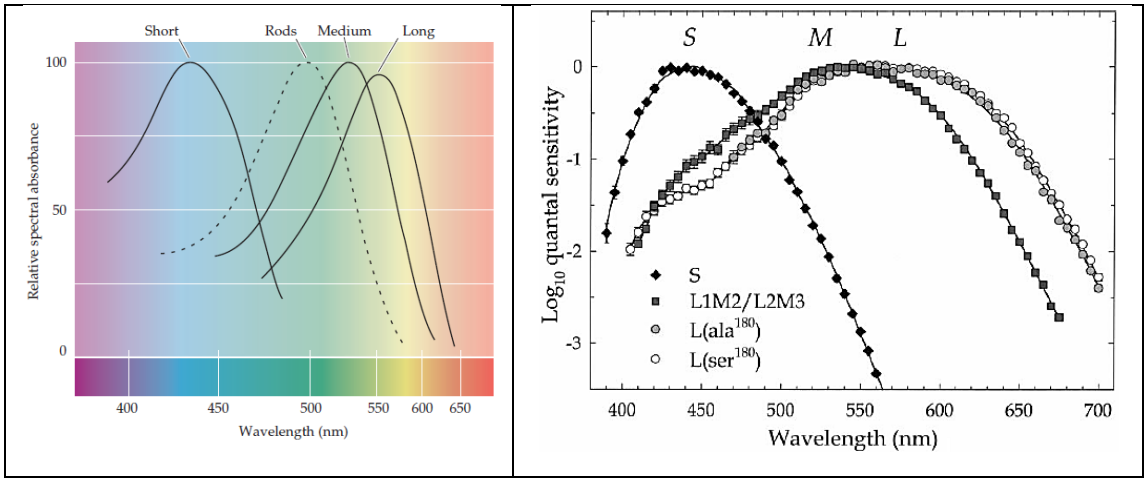

Figure 38: Sensitivities of S,M and L cones according to Dowling [187] that correspond to the blue, green and red colors of the spectrum. Left: Relative spectral absorbance [184] , Right: Log_{10} quantal sensitivity reported in [186].

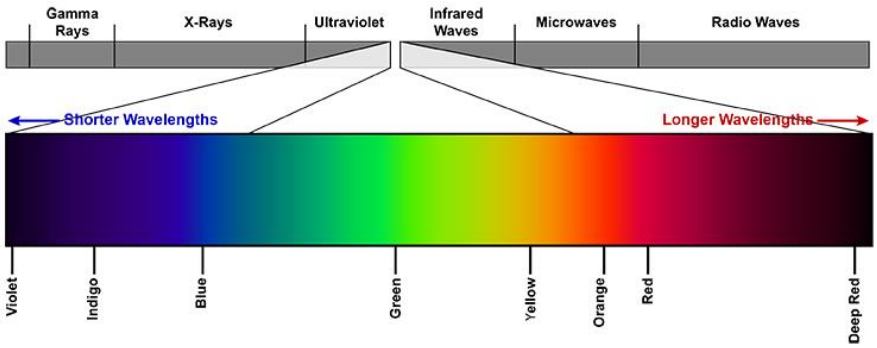

Figure 39: The visible spectrum and the locations of the seven rainbow color (image from [188])

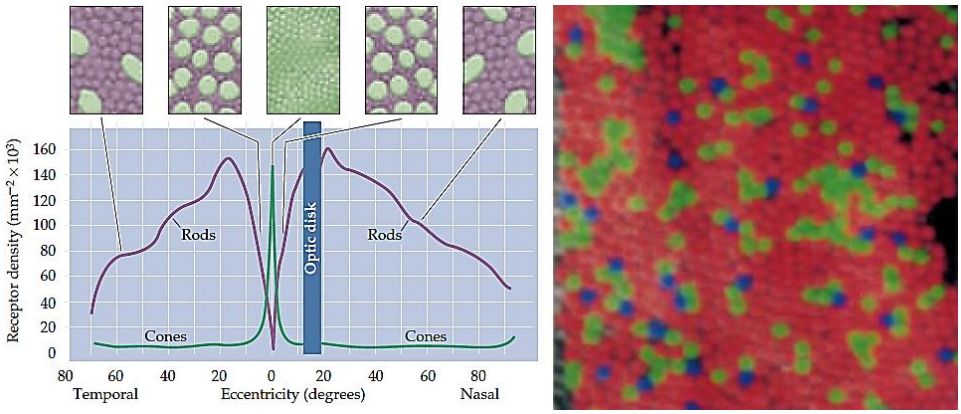

Figure 40: Left: The rod and cone photoreceptors densities in different eccentricities in the retina [184] according to Osterberg [189], Right: Imaging that illustrates the distribution of different types of cones, sensitive to red, green and blue colors in the center of the retina [185].

Another kind of photoreceptors are the *rod* cells that do not discriminate color, but detect tiny differences in illumination having a much greater sensitivity than cones [184].

After the photoreceptor the visual information is processed by a neural network and it exits the eye through the axons of neurons that form the *optic nerve* [184]. There are four kinds of neurons that are found in two tissue layers. The *outer plexiform layer* (OPL) consists of *horizontal cells* (HC) and *bipolar cells* (BP). The BP responses are propagated to the *inner plexiform layer* (IPL) where *amacrine cells* (AC) and retinal *ganglion cells* (GC) are found. The axons of BP neurons form synapses with the dendritic receptive fields of the GC neurons that are sensitive to specific color and spatially mapped at the underlying cone input [190].

The photoreceptor-BP-GC is the main vertical pathway in the retina. There are also lateral interactions that are implemented by the HCs and the ACs. The HCs regulate the amount of photoreceptor input that reaches the dendrites of the BPs. They are considered to measure the average level of illumination in a region, in order to implement a local gain control which enhances edges [191]. The ACs regulate the information flow from BPs to GCs by lateral inhibition that maximizes the effectiveness of the network while reducing the number of synapses [192]. This natural scheme can be reproduced in an artificial neural network with four successive layer of neurons. The BP layer as the second layer will receive input from the first HC layer and transmit its output to the third AC layer, where it will be processed and send to the fourth and last layer of artificial GC neurons.

Visual Neuroscience has provided with more knowledge on the later stages of visual processing. The GC axons that form the optic nerve are extended into a part of the brain called *lateral geniculate nucleus* (LGN) [193], where the neural signal is adapted and relayed to the *primary visual cortex* or *V1 cortex* [194]. It has been discovered that a category of cells in the V1 have a receptive field organization that is sensitive to oriented lines of light. These *hypercolumns* of *V1 simple* cells [195] are part of the ice cube model that is described in the

early work of Hubel and Wiesel on the macaque monkey visual cortex [196]. A second category are the *V1 complex* cells that are thought to integrate responses of V1 simple cells [42], [197] and a third category the *V1 hypercomplex* cells.

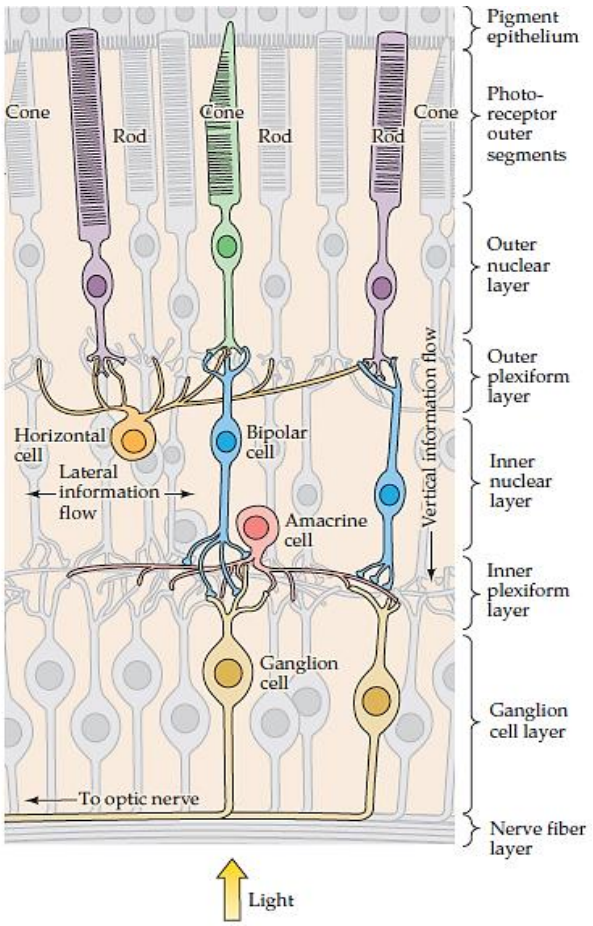

Figure 41: The human retina [184]. The photoreceptors are at the innermost side of the eye globe's convex sphere near the pigment epithelium, a cell tissue that absorbs scattered light and plays important role on the maintenance of the retina.

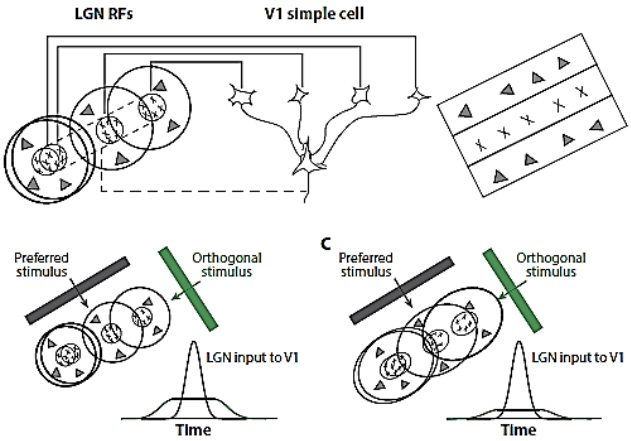

Figure 42: Top left: Lateral geniculate nucleus (LGN) to primary visual cortex (V1) neural circuitry. Top right: Orientation selective V1 receptive field. Bottom: LGN adaptation of a non-matching orthogonal stimulus [194].

3.4.3 Foundations of the Bio-inspired Convolutional Neural Network

Leveraging this knowledge we can design a deep convolutional neural network that imitates the biological neural network. Its layers will include the key functionality of the human visual system. Using convolutional modules with more than one layer or parallel layers, the CNN macro-architecture will resemble the stratification of the biological neural network. The functionality of the visual pathway from photoreceptor input to the V1 output can be engineered into the micro-architecture of the convolutional modules, using suitable mathematical functions in combination with convolution operations. This is a completely different approach from state-of-the-art architectures, proposing pre-defined weights for the convolutional kernels in addition to the learnable weights. Moreover all the primitive feature extraction on the image will not be performed by the first convolutional layer. Color distinction, fine-grained features like points, blobs and edge primitives, oriented lines and patterns will be extracted by different layers of the deep CNN, with their respective values placed in different activation maps.

Color distinction stands for the ability to detect contrasting intensities of light like white and black and contrasting colors like red and green, blue and yellow, a mechanism in human color perception known as *color opponency* [198]. Trichromatic human vision and RGB are both *additive color systems* [159], in which colors are formed by adding different wavelengths of light. The opposite is subtracting, when wavelengths are absorbed by paint used in artwork and printing systems. Opponent colors cannot be perceived at the same time due to the additive nature of the human color system. Red and green are opponent due to the existence L-cone and M-cone photoreceptors that are respectively sensitive to red and green wavelengths of light. The blue color is perceived by S-cone responses and the opponent yellow color is derived from the L-cone and M-cone responses, forming the blue-yellow opponency. For simplicity the white/black contrasting intensities will be considered a third opponent color pair.

With the term fine features we can distinct the most fine-grained local features that are extracted from the colors of pixels. A point or a circular blob with opponent colors at its surrounding is the most primitive feature. A transition from one color to its opponent or between different levels of illumination in the same color, indicates edge primitives, or *edgelets*. Combining fine features, lines can have two properties, directionality and length, and can be extracted at a second stage of processing. Clusters of biological V1 neurons are orientation-selective and the V1 hypercolumn of neurons covers all possible angles of a line. Kernels of V1 convolutional layers can have orientation-selective weights, engineered in their receptive field maps.

These primitive feature categories can be detected in any visual information and for any image collection, and when combined in the later layers they would create more complex features. The intuition that these features can have universal application, drives an effort to create a new type of CNN. In the VFL abstraction such features could be considered as the base for visual language semantics, transferable to different natural image datasets or to different domains of visual information.

3.4.4 Models Inspired by the Human Visual System

3.4.4.1 Historic attempts and models not based on artificial neural networks

Several ideas that were inspired by the human visual system existed before the proposal for this Master's thesis. Its inception dates back to the Bachelor degree thesis, which implemented a fully connected MLP network with several micro-networks. The earlier and current approaches found in bibliography have not modeled into a CNN architecture the complete functionality of the human retina as described in 3.4.2. Since the research sub-domain of Deep Neural Networks is one of the most dynamic, similar ideas may have been proposed at the same interval with this thesis.

One of the early attempts to create a neural network model that imitates the functionality of a biological retina and including all four main categories of neurons, was made by Eeckman, Colvin and Axelrod [199]. Many aspects of their system like DoG for receptive fields, overlapping receptive fields and directional sensitivity with filters of different spatial sizes can be found also in this thesis. The neural network was not convolutional and its purpose was not for image recognition, but for motion detection on monochrome telescope stills. It was evaluated as a noise reduction filter using the signal-to-noise ratio as its error metric. The layers of this architecture were the photoreceptor, HC cells, BP cells with DoG receptive fields, directionally sensitive AC cells and GC cells. This architecture was later implemented in a VSLI circuit, the Vision Processor Hardware (ViP) [200] .

In Virtual Retina [201] there is a five layer architecture implemented for monochrome images, that includes a photoreceptor layer. It implements HC neurons with Gaussian functions that simply smoothen the input and the BP functionality using a Difference-of-Gaussian function. It has a convolution operation followed by an activation function for BPs and GCs. ACs are used for contrast gain control and inhibition of BPs. Nevertheless no neural network implementation and performance evaluation can be found in the report. There is also an image encoding scheme based on Virtual Retina that uses a DoG filter bank and compresses the resulting format using the stack-run algorithm [202], [203].

A sophisticated photoreceptor layer is introduced in [204] which takes into account the Osterberg's distribution [189] of rod and cone cells according to the eccentricity of the retina and the Dowling's spectral sensitivities [187] for the red, green and blue cones. [204]. It proposes a color space transformation [75] from RGB to XYZ in order to reconstruct the original wavelengths that have been captured as RGB by the digital imaging sensor. A complete non-neural bio-inspired model of the retina is presented in [205] using the photoreceptor layer of [204]. The ACs are implemented as a co-adaptation function, emulating the information exchange between AC and BP with a fixed exchange rate of 20%. This makes the model difficult to port to a feed-forward neural network.

Another non-neural model proposed in [206] models the spectral sensitivities of cones to red, green, blue in conjunction to the distribution of photoreceptors according to the eccentricity in the retina. A recent bio-inspired model which models the retina and additionally the LGN and the V1 [207], implements a color opponency mechanism according to physiological evidence The retina is simplified into cone and GC layers only and the calculation for deriving a yellow color from RGB does not take into account the color sensitivities of the biological photoreceptors.

3.4.4.2 Related bio-inspired CNN models

A approach inspired by the on/off pathways of the retina uses a modified AlexNet with convolutional layers that have dual activations functions, considered as an ON/OFF ReLU activation function [208]. The ON is $y = ReLU(x)$ while the OFF is $y = ReLU(-x)$. It does not introduce a new CNN architecture, nor models the biological vision pathway.

The DERF feature extractors introduced in [142] use the Different of Gaussian (DoG) [209] and the neuroscience model of the retinal parvocellular GCs [210]. In a recent work the DERF feature descriptors were used with a CNN architecture [211]. A Category-specific Salient Region (CSSR) detection is performed to find interest points on the images which are clustered per category. Then DERF features are extracted for the cluster-quantized regions and a CSSR image representation is generated. These are concatenated with features extracted from a CNN and then used to train a classifier for scene classification. The method reports higher performance than the pre-trained Caffe reference model (AlexNet) [212] for the places dataset [213].

In [214] a five-layer CNN presents biological equivalences to V1 cells, namely simple and complex cells. In a similar to this thesis approach, the network has pre-defined parameters in the first layer, while learning is implementing at a later stage. It incorporates the use of data augmentation not only for training, but also as part of the network's recall

operation. It uses five different sizes of pre-defined Gabor filters on the same input image, in order to achieve scale invariance. It implements the spike timing-depended plasticity (STDP) [215] and it is trained with gradient descent using normalized weight increases/decreases to speed up the convergence. The same authors in [216] use DoG filters to model GCs in the first layer. Both works focus primarily on STDP and the bio-inspired approach is restricted to one kind of biological neuron in a shallow network implementation.

A Deep CNN model in [217] draws inspiration from the ventral stream that starts from the primary visual cortex V1, passes through upper visual cortex layers and ends in the temporal lobe of the brain [184], which is responsible for memory. It has a total of 11 layers with five $1x1$ NiN convolutional layers inspired by the human ventral stream. Biological findings are used for the selection of the layers' architectural hyperparameters, primarily the d_{input}/d_{output} ratio of the convolutional kernel hyperdepths. Using mean RGB centering and on-the-fly data augmentation to train the network, this architecture reports competitive accuracy on classification.

Concurrently with this thesis, the idea of using fixed receptive fields inside the convolutional neural network is proposed by [218] as the structured Receptive Field Neural Networks (RFNN). It implements convolutions with weighted Gaussian derivatives combining CNN's and scattering networks [219] in order to achieve scale invariance. A bio-inspired extension of the RNNN is found in [220] which uses Gabor functions inside convolutional kernels, something designed by the current thesis in parallel. The Gabor RFNN takes into account the organization of the V1 receptive fields and propose the Gabor functions as the closest implementation. These last two CNNs and the one proposed by this thesis in the following chapter may constitute a new meta-group of Deep Neural Networks that have a hybrid set of engineered and learned convolutional kernels.

4 The Bio-inspired Convolutional Neural Network (BioCNN)

4.1 Introduction

The CNN introduced in this work is a model that consists of both pre-defined and learnable parameters and thus considered a *hybrid CNN*. It models the complete stream of biological visual processing from the photoreceptors to V1 cells, implementing its fundamental concepts that were explained in section 3.3.3. According the author's best knowledge, this thesis is the first to propose a hybrid Deep Convolutional Neural Network together with a complete architecture inspired by the first stages of human vision. The neural network includes the photoreceptors and four types of neurons in the retina, the adaptation of retinal activations in the LGN and the functionality of V1 cells. It emulates the functionality of cone and rod photoreceptors by using a color space transformation before information is processed by the artificial OPL. The convolutional subsystem that is inspired by the retina, implements the color opponency mechanism and the V1-like convolutional modules respond to lines of different lengths and orientations, using parallel convolutional layers of different receptive field sizes. The whole structure from photoreceptor layer to V1-LGN output is the stem in the complete CNN architecture that has a subsequent stack of a few convolutional modules with relatively small feature hyperdepth. In section 4.2 there is a comprehensive overview along with the reasons that lead to the inception of this idea. Section 4.3 presents the layers of the Artificial Retinal Neural Network (ARNN) and their mathematical model in details. The modeling and engineering decisions taken for the Artificial Primary Visual Cortex (APVC) are described in section 4.4

4.2 Overview of the Bio-Inspired CNN

4.2.1 Inspiration

The inspiration for this work has been drawn from the field of Biomimetics [221], [222], that studies and imitates nature's models and processes. This can include natural occurring solutions that best serve the living organism sensory needs, like the case of human vision. Biomimetics suggest that when a novel technical solution is designed in a man-made system, its engineers should search for similar biological systems and evaluate them as potential prototypes [151]. This idea has worked well for some applications, like the Speedo Fastskin swimsuits, which have a surface that imitates the shark's skin and had been proven successful at the 2008 Summer Olympics [223].

Using this bio-inspired approach a question that arises is, can we mimic the neural networks of the human visual system to create successful deep neural networks for Computer Vision? The recent success of the CNNs that were originally inspired by biological neurons and the high performance of the bio-inspired SIFT features provide a first hint in that direction. Nevertheless, as the creator of the CNN recently highlighted [224], it is only an inspiration drawn from basic Neurophysiology and not exact models of the brain, that would fall in the field of Computational Neuroscience. This is exactly the approach of this Master's thesis that proposes a novel kind of deep neural network the *Bio-Inspired Convolutional Neural Network* (BioCNN), having first acquired key knowledge about the human visual system through a small study in Visual Neuroscience.

4.2.2 Motivation

Extracting features from still two-dimensional images is only a component in a greater vision module of an intelligent robotic agent that can operate in real-world conditions. Solving image recognition is just the start to build such systems, but at the same time it can provide complete solutions to domains like Medical Imaging.

The 2D image recognition component is a part of the 3D computer vision module with the ability to recognize objects in real-world scenes. Depth in three dimensions can be derived using stills from two cameras that are symmetrically positioned and have a correlated angular displacement. The RGB-D hardware like Microsoft Kinect and the more recent Intel RealSense imaging systems [225] incorporate depth as the fourth feature D of each pixel. The computer vision module should additionally be able to detect direction of light from color and depth. Moreover this module can belong to a sensory-motor assembly that generates feedback, for example adapting the field of view (FoV) and focusing on the scene

by rotating or converging the two cameras to target an object. A second module that operates in parallel can detect motion and identify moving objects. All modules for the artificial sense of vision will be part of an intelligent robotic agent operating in the real world, which can assist humans in places out of their reach, thus able to react faster than human in emergency situations like in natural disasters.

The second motivating reason would be the integration of a Content-Based Image Retrieval Software agent in medical imaging solutions like UT, CT/MRI PET-CT/MRI which will operate in real-world clinical environment [226]. Designed to additionally provide an advanced user experience (UX), these future CBIR systems could assist the medical scientist. The daily visual activity of a medical doctor has a trivial part that can be delegated to an intelligent CBIR agent, providing more time to concentrate on important decisions. Moreover if this technology becomes successful, the AI could identify an object on a medical image that has fallen out of attention. Recently CNN has been used in this domain, but the question of how effective is transfer learning has still to be answered [227]. Can we use a CNN trained on millions of natural images for image recognition with medical images, which have completely different sampling technology hiding the latent variables of a diagnosis in 1-3 features per pixel? Image recognition tasks for both Robotics and Medicine can be theoretically based on image retrieval with a universal Visual Features Language.

4.2.3 Hypotheses

The CNN architecture has been proven successful for certain image recognition tasks in lab conditions. Visualizing the learned receptive fields of AlexNet gives the first insight that the feature extractors learned on the ILSVRC2012 dataset converge to solutions found in biological visual systems. These are experimentally considered as reusable through transfer learning to other domains. Seeing that from a point of view in the destination domain, the weights of a pre-trained convolutional kernel are pre-defined and need a minor fine-tuning. This point of view can be extended with the proposal that some pre-defined parameters can be calculated using mathematical models that are inspired by corresponding biological models of the human vision. These parameters could be non-learnable in order to eliminate the need of fine-tuning for the layers that contain them.

The significance of gaining insight on the inner functionality of the convolutional layers has been made apparent by the design of ZFNet which was aided by visualization. An intuition is that better CNN architectures can be discovered when there is already an understanding of the inner function. Using simple math functions to generate pre-defined weights in combination with r*eceptive field map engineering* of the convolutional kernels, can create a new ground of study for deep CNNs, which could explore more formal reasons for

their performance and steer decisions in the design of networks. The question whether new CNN models should require image preprocessing or dataset augmentation during training, could be potentially be answered by embedding these needs the architecture itself, having found an optimum set of hyperparameters. The design space exploration process can use objective functions according to the mathematical models that generate the convolutional kernel weights, which are a-priori known. The proper combination of network depth, feature hyperdepths, and engineered RFMs, along with learnable parameters through optimization, can potentially embed in a CNN architecture an invariance to label-preserving transformations.

We make the following hypothesis: It is possible to model a set of pre-defined parameters for the stem of a CNN architecture, keeping the same accuracy with state-of-the-art models and using a fraction of total learnable parameters. There are already such proposals in the direction of minimizing the computation cost without losing quality, most notably SqueezeNet [109]. Non-trainable parameters at an early stage can potentially be a regularizer for the entire CNN architecture, mitigating overfitting and restraining the vanishing/exploding gradient problems.

A second hypothesis is that the convolutional layers of such a hybrid architecture will be useful extractors for fine, local and global features used in the CBIR context. Keeping less layers and smaller feature hyperdepths can create a shallow depth and narrow width geometry for the CNN architecture. A shallow-narrow CNN can be experimentally proven sufficiently accurate, while fitting at the same time in a smaller amount of memory.

Both hypotheses are based on the study of human vision and some of its fundamental properties. Human vision is the best solution in nature for visually sensing the environment. This claim is made not for the visual system itself, but for its supporting role in higher intelligence. The basic functionality of biological vision is implemented in a shallow neural network inside the retina, which is connected with the more complex and deep networks in the brain. There could be as few as 8-10 biological neural network layers of distinct classes of neurons before the first point, line and texture features of a natural scene are sensed in the visual cortex.

4.2.4 Architectural Overview

This Master's thesis introduces a reference architecture and the basic concepts for a BioCNN deep neural network, consisting of three subsystems and a classifier. The first subsystem which processes the input image is the *Artificial Retinal Neural Network* (ARNN) whose output if forwarded to the *Artificial Primary Visual Cortex* (APVC). These two subsystems form the hybrid stem, a term used in Inception architecture [98], of the BioCNN where the convolutional kernel weights are mostly pre-defined. The third subsystem is the *Artificial Higher Visual Cortex* (AHVC) which is a stack of convolutional modules on top of the stem and before the classifier, as a typical CNN architecture with learnable parameters.

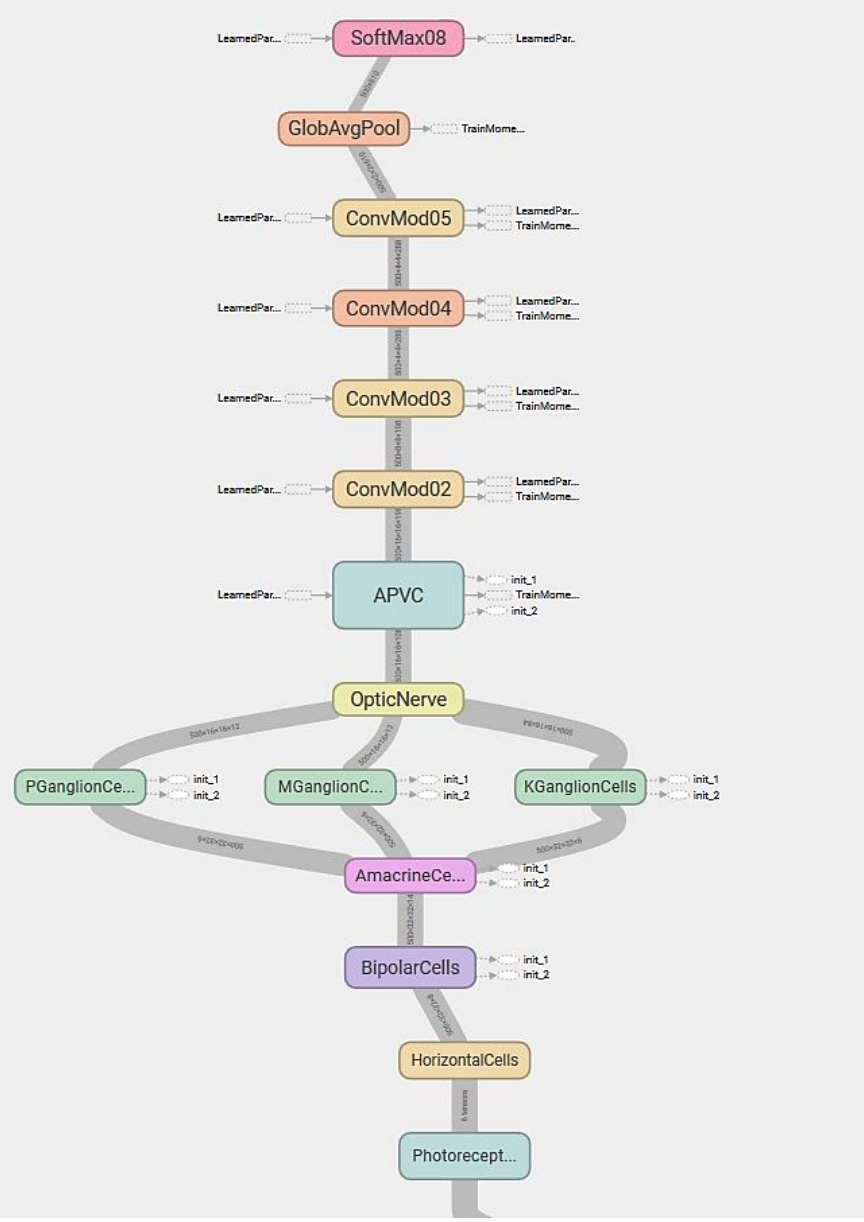

Figure 43: Reference macro-architecture of the BioCNN (visualization using Google Tensorboard). The stem of the network starts at the photoreceptors layer and ends with the APVC subsystem. The convolutional modules of the AHVC are numbered from 2 to 5. After module 5 a Global Average Pooling layer connects each output feature with the class logits of the Softmax layer.

The ARNN subsystem contains of three main parts. The Photoreceptors Layer (PRL) converts the trichromatic color representation of the input image into *chroma* and *luma* components. The outer plexiform layer (OPL) stack consists of the Horizontal Cells Layer (HCL) and the Bipolar Cell Layer (BPL), which implement sensitivities to target colors and levels of illumination. The inner plexiform layer (IPL) stack has the Amacrine Cells Layer (ACL) that implements color opponency and non-linear brightness. These activations are input to three parallel layers of ganglion cells (GC) with receptive fields of different spatial sizes. Activation maps of parvocellular GC (p-GC) layer, magnocellular layer (m-GC) and koniocellular layer (k-GC) are concatenated in a similar fashion like in an Inception module [34] [98], forming the 3D output tensor of the A*rtificial Optic Nerve*. The layers in the ARNN subsystem were engineered as one-to-one porting of their biological counterparts.

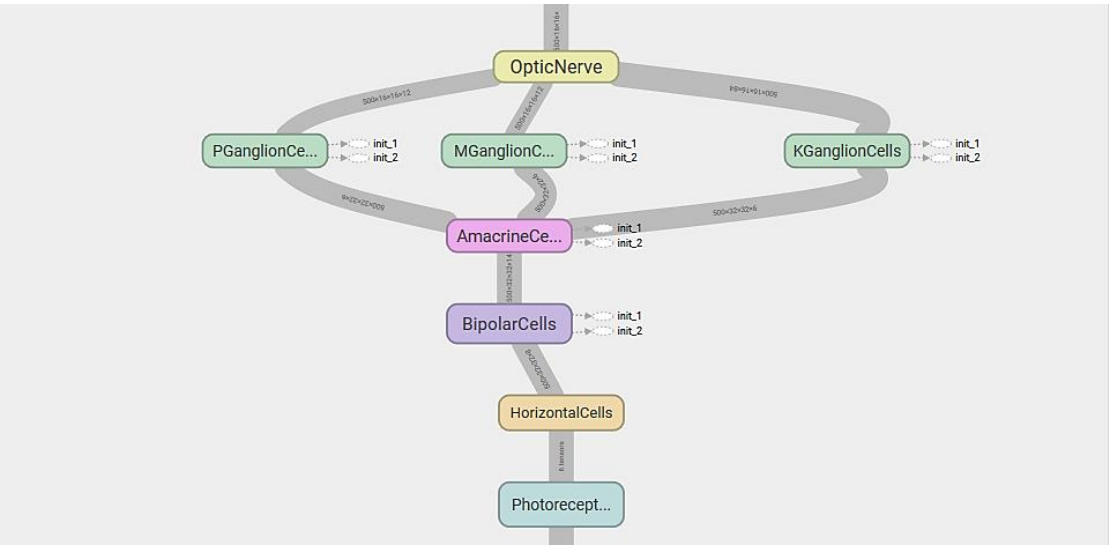

Figure 44: Layers of the Artificial Retinal Neural Network (ARNN)

The APVC subsystem is a hybrid module that implements shortcut connections introduced in [95] and utilized by the ResNets [35], containing layers with both pre-defined and learnable parameters. In the lateral geniculate nucleus (LGN) abstraction, the optic nerve output of the ARNN is normalized and fed as input to two kinds of layers. The V1 Gabor filter layers model the responses of simple and complex cells and the LGN-V1 learnable layers are inspired by the adaptation done by the LGN before the stimuli reach the V1 visual cortex. The fine features of the ARNN are processed in V1 in order to extract line and pattern primitives. Additionally the optic nerve input of the subsystem is forwarded by a shortcut connection to the output of the APVC without implementing a residual functionality. Thus this output will be a concatenation of ARNN fine features and APVC primitive features extracted from them, providing an augmented feature tensor to the AHVC.

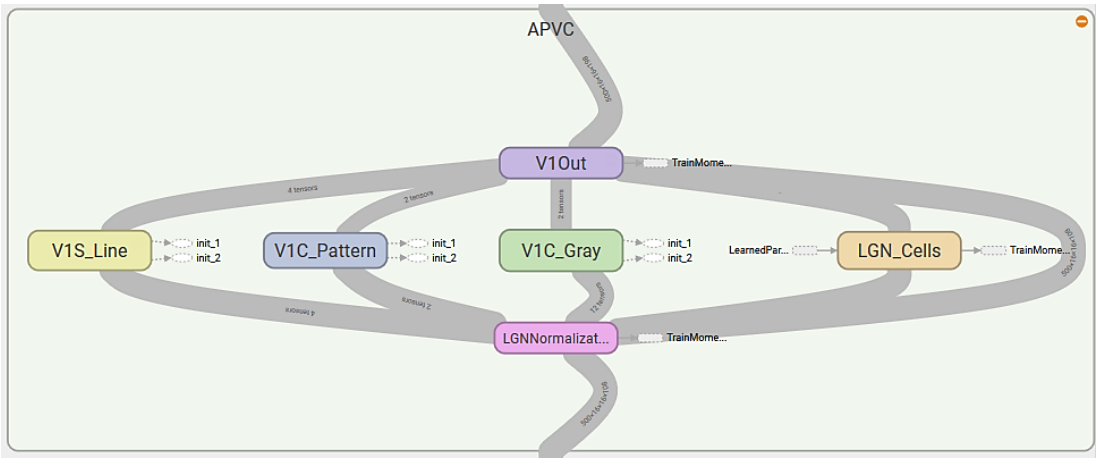

Figure 45: Groups of convolutional layers in an Artificial Primary Visual Cortex (APVC)

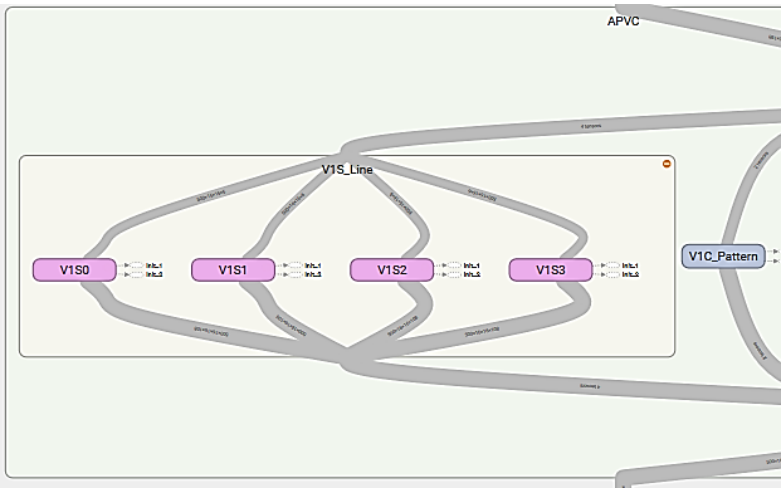

Figure 46: Layers in the group of V1 simple cells with different spatial window sizes operating as feature extractors for oriented line primitives.

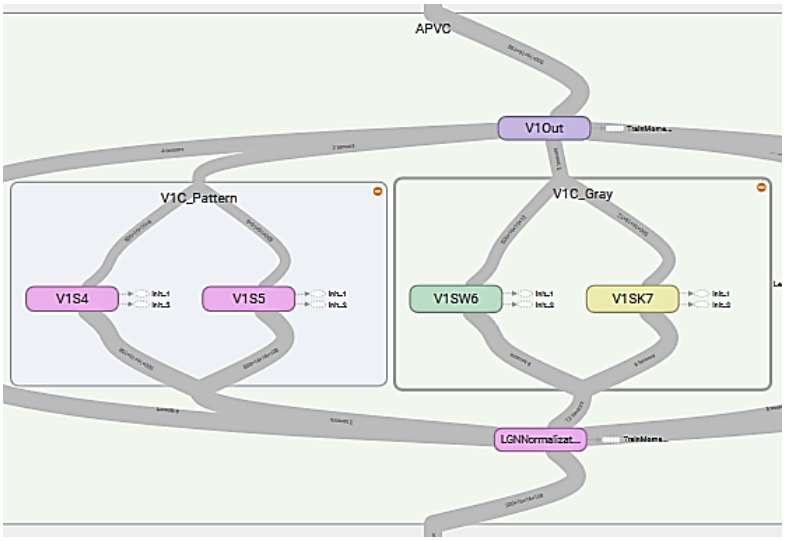

Figure 47: Layers of V1 complex cells with different spatial window sizes. Left: the group of pattern feature extractors and right the layers for the color ignorant V1C_Gray group.

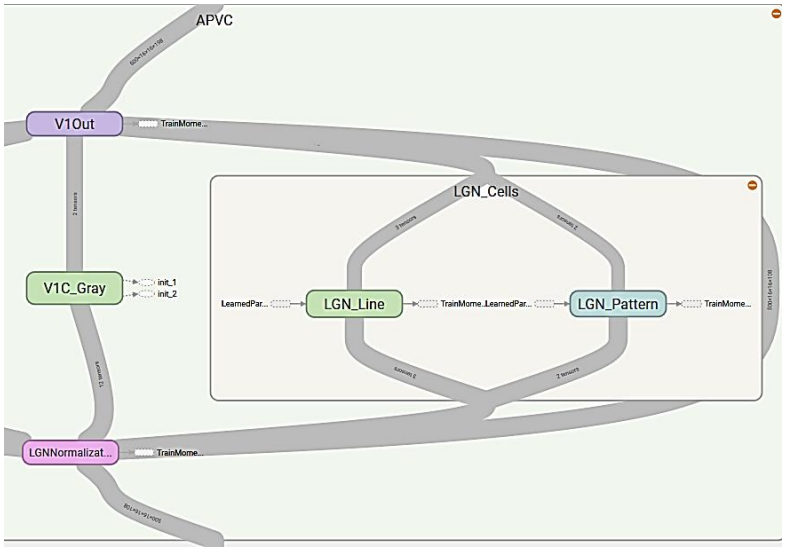

Figure 48: Layers of LGN-V1 complex adaptive cells, which are initialized with the pre-defined RFMs of V1 cells and fine-tuned during the learning process.

The AHVC subsystem is nothing more than a stack of trainable convolutional modules that have CONV and optionally MAXP reduction layers. The classifier can have FC layers and a SMAX exit layer, or alternatively a GLAVP layer that is connected to the SMAX. Several architectures for AHVC could be evaluated to achieve better accuracy, currently out of this Master's thesis scope. Using Inception modules, BN, and residual connections in a combination with the BioCNN stem could be the subjects of further research.

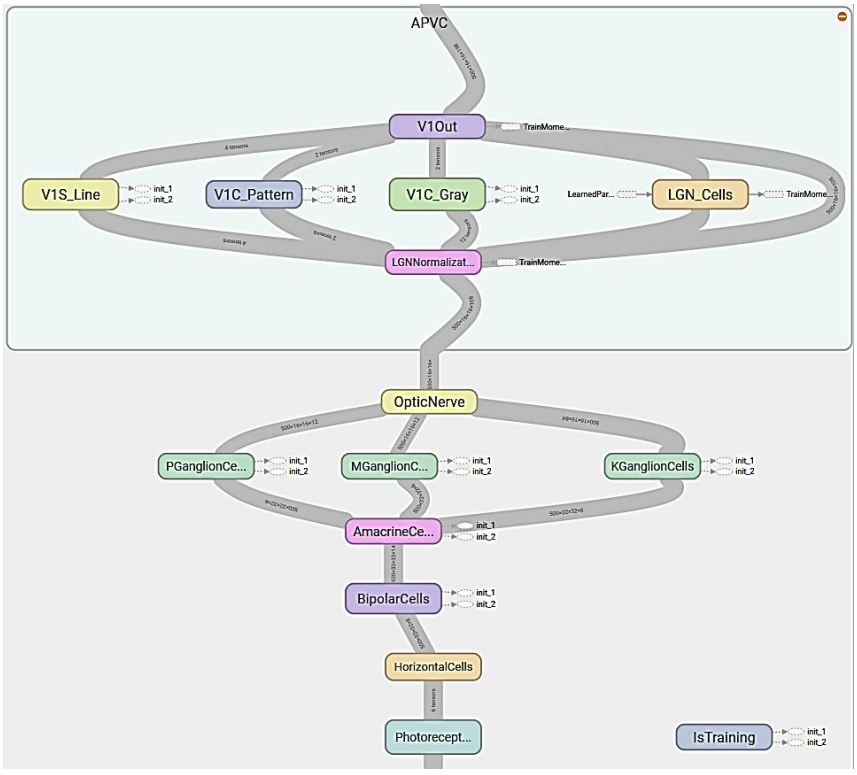

Figure 49: Reference macro-architecture for the BioCNN stem that can be used in any convolutional neural network architecture.

4.3 Artificial Retinal Neural Network (ARNN)

4.3.1 Color Space Transformation with the Photoreceptors Abstraction

In the first stage of the BioCNN processing, the Photoreceptor Layer (PRL) is an abstraction of the biological photoreceptor functionality. They are implemented as a color space transformation [75] [228] and a set of one-dimensional Gaussian functions [136], expressing peak cone responses to specific color and rod non-linear sensitivity to luminance. The RGB values are converted into HSV [228] in order to use the *hue* (H) component for the cone functionality. Hue expresses the degrees that a color is found in the color wheel [229] by encoding them in the range of $[0,1]$. Both 0 and 1 corresponds to the red color, creating a circular representation.

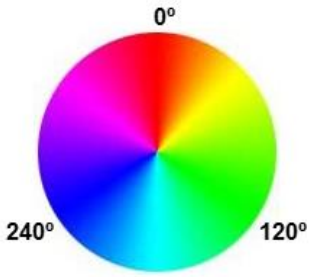

Figure 50: Color disc with angles for the red, green and blue.

The peak spectral sensitivities of cones are modeled as peak responses of Gaussian functions that target specific hues. This however is not implemented directly in the PRL but as part of the color detection functionality in the OPL, thus the photoreceptors abstraction spans across the first and the third layer of the BioCNN neural network. The *brightness* (V) component in HSV, or B in HSB, is not correlated with changes in *saturation* (S) [230]. The saturation invariance can be achieved by discarding the S component at the PR layer.

The grayscale colors are perceivable when $R \approx G \approx B$. According to the RGB-to-HSV color transformation functions [231] [75], hue for grayscale colors is either undefined and replaced by zero which represents the red color, or $H \approx 0$. This is inappropriate when considering the hue as input to a color discriminating function and for this reason a negative value is introduced to express a *null hue*. For a pixel at (x, y) the vector or RGB values is $p^{\{x,y\}} = [\, r(x,y), g(x,y), b(x,y) \,]$ and the function c_μ returns a positive value in the range $[0,1]$ for valid hues or a negative value in $[-\mu, 0]$ for gray-like colors:

$$c_\mu(x,y) = h(x,y) - \mu \cdot (\tanh(\left\| p^{\{x,y\}} \right\|_1) - 1) \tag{39}$$

The corresponding hue of the pixel is $h(x,y) \in [0,1]$ and $\mu < 0$ is the *null hue constant* that is typically set to $\mu = -2$. The component C for *chroma* is calculated from the function

$c_\mu(x, y)$ as a feature which expresses the combined responses of cone photoreceptors to a range of the visible color spectrum. The responses are inhibitory for white light that is captured by the digital imaging sensors with small differences of the RGB components.

A choice for modeling the rod photoreceptors could have been the V component that is independent from saturation, calculated as the maximum of the RGB values. Nevertheless a feature extractor using only the hue H/C and brightness V could not be able to find edges between different saturations of the same hue. A second choice will be to use the *lightness* L of the HSL color space that is the average of the maximum and minimum of the trichromatic RGB component. The component the *luma* Y [232] of the YIQ color space, which takes into account the human perception [233], uses a weighted sum of all three RGB components, thus encapsulating saturation differences in the same hue.

In the central region of the human retina there are substantially more red and green than blue cones, as presented in section 3.3.2. The selection of the Y component suits intuitively as more close to their distribution in the human retina [185]. Additionally it worked well for human perception of monochrome video, because YIQ was used for TV broadcasting since 1953, when the monochrome receivers used Y [230]. Luma is calculated by a simple matrix multiplication of the 3D input tensor of RGB features with the vector of the luma coefficients.

$$w_{luma} = [\, 0.2989, 0.5866, 0.1145 \,]$$

The resulting 2D matrix will emulate the rod responses for the image, creating a monochrome representation by mixing approximately 30% red, 59% green and 11% blue for each pixel. The dominating green color and the red color are also compliant with the distribution of colors in the Bayer CFA pattern, described in section 2.2.1, with small differences in mixing percentages $[-5\%,\ -9\%,\ +10{,}75\%]$. A more formal approach could take into account the CFA distribution of the digitization equipment. Furthermore it is known that cone and rod population varies in the retina depending on the eccentricity, finding mostly cones in the *fovea centralis* and rods in the outskirts of the retina [184]. In this baseline ARNN model none of these are implemented, but the sparsity of photoreceptor values is considered to gain significant computational benefits for future BioCNN designs.

Putting them all together a color transformation from RGB to the custom color space CSY (chroma-saturation-luma) is made by the photoreceptor layer of the ARNN as the basis of the color discriminative functions of the later stages. The system ensures saturation invariant color detection and identification of color absence keeping only C and Y. Differences in saturation or illumination for the same color are encoded in the Y component. The PR layer performs a dimensionality reduction $3 \rightarrow 2$ transforming RGB into two features.

4.3.2 Modeling the Outer Plexiform Layer

4.3.2.1 The Gaussian Convolutional Layer

The CY tensor is used as input for the two-layer OPL module, discarding the saturation component S. The OPL implements the spectral sensitivities of cones and the primary color streams that will be used for the opponent color functionality. The response function of a cone to a specific target value of chroma C at a pixel $z = c(x, y)$ is implemented by a one-dimensional Gaussian function $gauss(c(x, y))$. For three types of cones each one will have a different standard deviation σ_z and mean μ_z .

$$gauss_{\sigma,\mu}(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(z-\mu)^2/2\sigma^2} \tag{40}$$

This functionality is split into the two layers of the OPL the Horizontal Cell Layer (HCL) and the Bipolar Cell Layer (BPL). The output values of the 1D Gaussian function $gauss_{\sigma,\mu}(x)$ with standard deviation σ and mean μ are normalizing into the range of [0,1]:

$$y = f(x) = \frac{gauss_{\sigma,\mu}(x)}{gauss_{\sigma,\mu}(\mu)} : x \in [0,1], y \in [0,1] \tag{41}$$

In order to facilitate a NiN 1×1 convolutional layer that will produce peak activation at specific values of chroma $x \in [0,1]$, the normalized Gaussian function is transformed into a weighted sum that suits the artificial neuron model:

$$y = \frac{gauss_{\sigma,\mu}(x)}{gauss_{\sigma,\mu}(\mu)} \quad \Rightarrow$$

$$y = \frac{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\mu-\mu)^2}{2\sigma^2}}} = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{e^0} = e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \Rightarrow$$

$$lny = lne^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \Rightarrow$$

$$lny = -\frac{(x^2 - 2\mu x + \mu^2)}{2\sigma^2} \qquad w_1 = \frac{1}{2\sigma^2} \qquad \Rightarrow$$

$$lny = -(w_1 x^2 - 2w_1 \mu x + w_1 \mu^2) \quad w_2 = -2w_1\mu \ and \ b = w_1\mu^2 \quad \Rightarrow$$

$$e^{lny} = e^{-(w_1 x^2 + w_2 x + b)} \qquad w_2 = -\frac{\mu}{\sigma^2} \ and \ b = \frac{\mu^2}{2\sigma^2} \quad \Rightarrow$$

$$y = e^{-u} \qquad u = w_1 x^2 + w_2 x + b$$

(42)

Equation 42 expresses the exponential activation function $f(u) = e^{-u}: y \in [0,1]$ and the synaptic sum u of an artificial neuron which generates the peak values of $y = 1.0$ when $x = \mu$ and uses an input vector $X = [x^2, x]$. The synaptic weights vector $W = [w_1, w_2] = [\frac{1}{2\sigma^2}, -\frac{\mu}{\sigma^2}]$ and the bias $b = \frac{\mu^2}{2\sigma^2}$ depend upon the choice of the hyperparameters σ, μ.

The *Gaussian Convolutional Layer* is a 1×1 convolutional layer that represents multiple neurons sensitive to certain values of input features, responding with activations that belong to a Normal or Gaussian distribution. A 4D convolutional kernel G noted as $[1 \times 1 \mid 2 \rightarrow h\,]$ will include weight vectors and bias values $w_i, b_i : i = 1..h$ for each one of the output features, implementing neurons with different sensitivities on the same input.

$$f_{\sigma,\mu}(\boldsymbol{U}) = e^{-conv2D(\boldsymbol{G}\cdot \boldsymbol{X})+\boldsymbol{B}} \tag{43}$$

This layer is tightly coupled with the previous layer which provides its activation values squared in the 3D tensor X. The parameters for the G weights and B biases are the width of the bell curve σ and its peak activation value μ. For a learnable Gaussian convolutional layer the error minimization process could adjust σ_i and μ_i according to their contribution in the layer's combined error. This perspective can be explored in an in independent study.

BioCNN neurons are spatially mapped to different regions of the artificial retina. Using a convolution operation with a common stride for both horizontal and vertical directions their mappings will implement a grid-like *artificial retinotopic map*, a term inspired by the biological neuron mappings to locations of the retina [234].

This type of layer resembles Radial Basis Functions (RBF) networks [235] but it differs from them by not weighting the Gaussian function output. It can be considered as RBFs inside a convolutional kernel. This is a completely different operation compared to the Convolutional RBF Network described in [236], where a smoothened RBF network is produced from a convolution with a smoothing kernel.

4.3.2.2 Horizontal Cell Layer

The HCL provides the necessary input for the BPL which is a Gaussian convolutional layer. It simply packs the values for chroma and luma for each pixel into the 3D tensor $O_{HC} = \{C^2, C, Y^2, Y\}$ that will be fed to the convolution operation of BPL where the color matching function is implemented. The HCL can be considered a pass-through layer with dual activation functions for both C and Y:

$$f_{HC1}(\boldsymbol{u}) = \boldsymbol{u}\,, \qquad f_{HC2}(\boldsymbol{u}) = \boldsymbol{u}^2$$
$$f_{HC} = f_{HC1}(\boldsymbol{u}) \cup f_{HC2}(\boldsymbol{u}) \tag{44}$$

It could be a convolutional layer using $[3 \times 3]$ or large spatial dimensions for a kernel that will average the luma of a region, imitating a property of biological HCs, which average input from many photoreceptors. A weighted average over a region of Y values can be implemented using a receptive field map W_{HC} for the luma input channel with its weights set to $w_{ij} = \beta_{ij}\,(1/m * n)$ where $m \times n$ the spatial dimensions of the input region and β_{ij} the corresponding weight for the location (i, j) in the map. We define the function $y(i, j)$ that returns the Y component for a pixel at coordinates (i, j) and the brightness smoothing operation of the HCL can be expressed as:

$$\boldsymbol{u} = u_{HC1}(\boldsymbol{p}) = u_{HC2}(\boldsymbol{p}) = conv2d\big(\boldsymbol{W}_{HC} \cdot y(i,j)\big) = conv2d\Big(\beta_{ij} \cdot \frac{1}{m * n} \cdot y(i,j)\Big)$$
$$f_{HC1}(\boldsymbol{u}) = \boldsymbol{u} \qquad f_{HC2} = \boldsymbol{u}^2$$
$$f_{HCL}(\boldsymbol{u}) = f_{HC1}(\boldsymbol{u}) \cup f_{HC2}(\boldsymbol{u}) \tag{45}$$

HCL layers can also perform downsampling of the image with a convolution $[3 \times 3 \sim s\,|2 \to 4] : s > 1$ reducing complexity at the earliest stage. Images with higher than 300×300 resolution can be handled by the HCL layer in order to be transformed to a more compact CY representation. Moreover HCL can implement interleaving in the receptive field mappings with operations like $[1 \times 1 \sim s \mid 2 \to 4]: s > 1$, RFMs that have zero weights or "holes" and atrous convolutions [237]. In this first implementation of an ARNN no such functionality is implemented by the HCL layer. Furthermore non-neural interpolation methods that use chroma and luma can be used for downsampling, like the nearest neighbor interpolation [238].

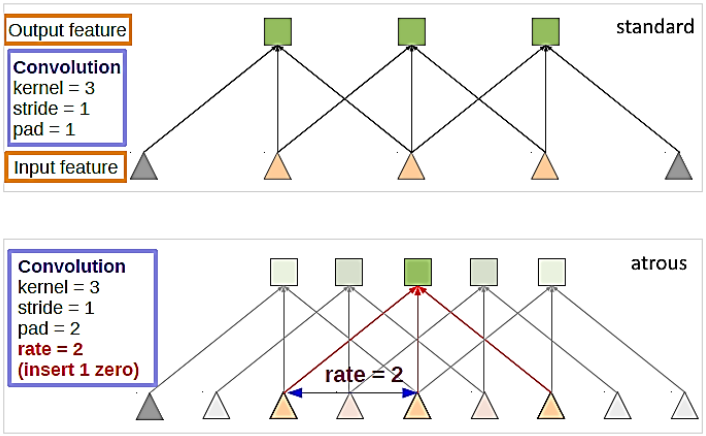

Figure 51: Atrous convolution [237]

Summarizing we can consider four different types of HC neurons as operating on the CY output of the PR layer and adjusting the input of BP neurons, inspired by the biological horizontal cells. The concatenated output of the HCL has a feature hyperdepth of $h = 4$ and the layer can be considered in the simplest form as $[\,1 \times 1\,|2 \rightarrow 4]$. Various methods can be used for future HCL implementations with respect this layer feeds color and luma to the second convolutional layer of the OPL module.

4.3.2.3 Bipolar Cell Layer

The BPL is an $[\,1 \times 1\,|\,4 \rightarrow 14\,]$ Gaussian convolutional layer with 10 color features and 4 brightness features. Its activation function is $f_{BPL} = f_{\sigma,\mu}(u) = e^{-u}$ and the convolutional kernel $g(x, y, z)$ for color BPs will have a receptive field map $W_z = \left[w_{c^2}, w_c, w_{y^2}, w_y\right] = [\frac{1}{2\sigma^2}, -\frac{\mu}{\sigma^2}, 0\,, 0]$ for each color feature $z = 1\,..\,10$ using only the chroma channels $\{C^2, C\}$. For a value of C that equals to the Gaussian mean μ_z the convolution sum u used in the activation function f_{BPL} will produce the peak value. This value will represent the peak spectral sensitivity of the cone photoreceptors and the minimum value of 0 will represent the lack of target color. There are BP neurons for all the hues of the opponent pairs, red-green and blue-yellow.

With respect to the color circle, the "peak" manifestations of each color are found at specific degrees, if we consider using the extreme values of 0 and 1 for their normalized RGB components. These can be referred as *pure colors*, a group that includes any RGB combination with one or two components set to the maximum value and the rest to the minimum. They are used in context as the most representing samples, but this can be differently interpreted in humans in psychometric experiments. The XKCD color naming

survey [239] discovered that colors which are perceived as "perfect" red, green, blue and yellow colors do not match the current models of color naming [240] , like the HTML color names. The angles of pure colors in the color circle are represented as hues in the [0,1] interval and are used for the mean μ that produces the peak Gaussian neuron activation.

0^0 60^0 120^0 240^0 360^0

Figure 52: The pure colors and their degrees on the color circle. Depending on the medium the reader can perceive the intensity of color opponency in the numbers that represent the R, G, B values.

The red color will have two targets one for the range of reds that is close to 0 and another one for the range close to 1, implementing the circle or colors.

$$\mu_{red1} = 0 \quad \mu_{yellow} \simeq 0.1666 \quad \mu_{green} \simeq 0.3333 \quad \mu_{blue} \simeq 0.6666 \quad \mu_{red2} = 1$$

The second hyperparameter for the pre-defined BP weights is the standard deviation σ that is selected in accordance to a degree span over the color circle. An empirical choice for each corresponding color is presented at the following table

Bipolar Cell type	**Color presence**	**Peak excitation at Chroma (C)**	**Width of the Gaussian bell curve**
R-On detector 1	red	$\mu_{red1} = 0.0$	$\sigma_{red\_on1} = \frac{30^0}{360^0} \simeq 0.0833$
Y-On detector	yellow	$\mu_{yellow} \simeq 0.1667$	$\sigma_{yellow\_on} = \frac{24^0}{360^0} \simeq 0.0667$
G-On detector	green	$\mu_{green} \simeq 0.3333$	$\sigma_{green\_on} = \frac{30^0}{360^0} \simeq 0.0833$
B-On detector	blue	$\mu_{blue} \simeq 0.6667$	$\sigma_{blue\_on} = \frac{64^0}{360^0} \simeq 0.1778$
R-On detector 2	red	$\mu_{red2} = 1.0$	$\sigma_{red2\_on} = \frac{33^0}{360^0} \simeq 0.0917$

Table 7: Bipolar cell target-color-on detectors.

An intuition on the color circle and the order of appearance of colors, suggests that the yellow-on BP must have more specificity to its color compared to BPs for red and green colors. BP-Y-On should not be activated by colors perceived as reddish or greenish, so a 20% narrower bell curve was selected. For its opponent color blue, which lies in the opposite side

of the color circle, a wider bell curve has been intuitively chosen to cover the larger angular span.

The biological BP cells have excitatory and inhibitory responses to the presence of a target color (ON) and the presence of opponent color (OFF). The BPL has two kinds of color detectors target-color-on and target-color-off for each one of the five pure color hues in a total of 10 color activations. The set of μ hyperparameters of BP-Color-On neurons is reused for BP-Color-Off neurons with a different selection of standard deviations:

Bipolar Cell type	**Opponent color presence**	**Peak excitation at Chroma (C)**	**Width of the Gaussian bell curve**
R-Off Detector	green	$\mu_{green} \simeq 0.3333$	$\sigma_{red\_off} = \frac{33^0}{360^0} \simeq 0.0917$
Y-Off Detector	blue	$\mu_{blue} \simeq 0.6666$	$\sigma_{yellow\_off} = \frac{55^0}{360^0} \simeq 0.1528$
G-Off Detector 1	red	$\mu_{red1} = 0.0$	$\sigma_{green\_off1} = \frac{33^0}{360^0} \simeq 0.0917$
G-Off Detector 2	red	$\mu_{red2} = 1.0$	$\sigma_{green\_off2} = \frac{33^0}{360^0} \simeq 0.0917$
B-Off Detector	yellow	$\mu_{yellow} \simeq 0.1667$	$\sigma_{blue_{off}} = \frac{33^0}{360^0} \simeq 0.0917$

Table 8: Bipolar cell target-color-off detectors

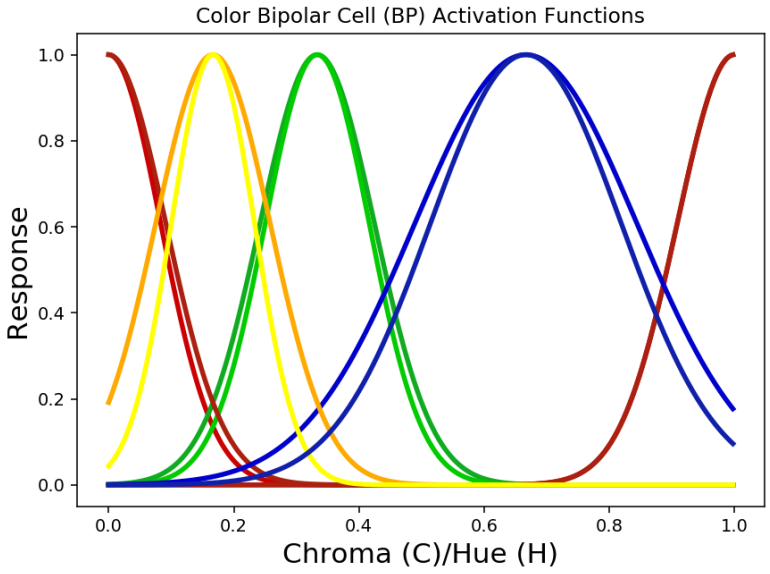

Figure 53: Activation functions of On/Off Bipolar Cells for red, yellow, green and blue. The BP-On color detectors have a darker line for their respective colors, while the BP-Off a lighter.

Additionally to color detection, the BPL detects illumination levels for light and darkness, based on the knowledge that biological BP receive input from both rods and cones.

The same neuron model is used, with a RFM map that discards the color stream $W_z = [w_{c^2}, w_c, w_{y^2}, w_y] = [0, 0, \frac{1}{2\sigma^2}, -\frac{\mu}{\sigma^2}]$ using the luma pair $\{Y^2, Y\}$. The exponential function transforms the linear scale of Y intensities introducing non-linearities in the BPL output. The peak brightness value $Y = 1.0$ is used as the target for the white-on detector and the peak darkness value $Y = 0.0$ for the black-on detector. The hyperparameter table for all types of colorless BP cells is:

Bipolar Cell type	**Luminance Level**	**Peak excitation at Luma (Y)**	**Width of the Gaussian bell curve**
W-On detector	light	$\mu_{white} = 1.0$	$\sigma_{white\_on} \simeq 0.3804$
W-Off detector	dark	$\mu_{black} = 0.0$	$\sigma_{white\_off} \simeq 0.0176$
K-On detector	dark	$\mu_{black} = 0.0$	$\sigma_{black\_on} \simeq 0.1216$
K-Off detector	light	$\mu_{white} = 1.0$	$\sigma_{white\_off} \simeq 0.3804$

Table 9: Bipolar cell white/black on/off detectors

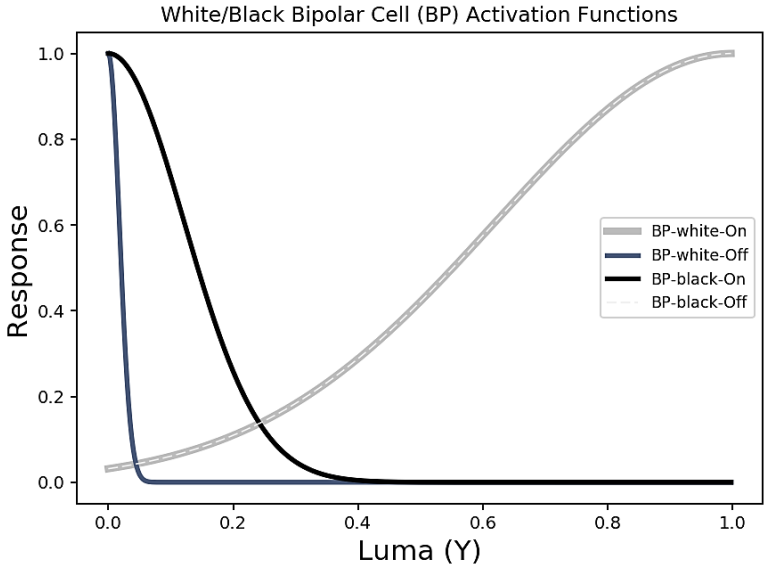

Figure 54: Activation functions of On/Off Bipolar Cells for white and black

The curve for BP-white-off has been intuitively crafted according to the brightness levels where a color cannot be perceived.

The functional properties of the hyperparameters set for the color and luminance detectors in the BPL implement a very large search space that can be explored in future research. Moreover an optimization process can be used so that the ARNN learns the optimum values of μ and σ which are used in the generating functions for the convolutional kernel weight. For simplicity in the current thesis these have been empirically chosen. The BPL and the tightly coupled HCL form the OPL micro-network that that falls under the NiN

paradigm. OPL is placed between the PR layer and the Inner Plexiform Layer (IPL). The full functionality of the color opponency that starts with the BPL neurons is completed in the IPL.

4.3.3 Modeling the Inner Plexiform Layer

4.3.3.1 Amacrine Cell Layer

The retinal amacrine cells are still a subject of research in the field of Visual Neuroscience with many of their properties not fully known. They form synapses at the axons of BP cells, adjusting their activations before they are transmitted to the ganglion cells. In the Inner Plexiform Layer (IPL) of the ARNN they are implemented as an 1×1 convolutional layer between the BPL and the Ganglion Cell Module (GCM). The activations of the Amacrine Cell Layer (ACL) express the color opponency with positive values when the target color is detected and negative values for the opponent color. They implement a weighted Difference of Gaussian (DoG) function [138] [209] between target-color-on and target-color-off BPL activations. The output activation tensor B of the OPL has only positive values in the range of $[0,1]$. A weighted sum of all excitatory BP-on activations minus the inhibitory BP-off activations has two formulas for red and green and a common formula for blue, yellow, white and black colors.

$$A_{RED}(x,y) = W_{R\_ON}B_{R\_ON1}(x,y) + W_{R\_ON}B_{R\_ON2}(x,y) - W_{R\_OFF}B_{R\_OFF}(x,y) \tag{46}$$

$$A_{GREEN}(x,y) = W_{G\_ON}B_{G\_ON}(x,y) - W_{G\_OFF}B_{G\_OFF1}(x,y) - W_{G\_OFF}B_{G\_OFF2}(x,y) \tag{47}$$

$$A_Z(x,y) = W_{Z\_ON}B_{Z\_ON}(x,y) - W_{Z\_OFF}B_{Z\_OFF}(x,y) \quad : z \in \{blue, yellow, white, black\} \tag{48}$$

The activation tensors of the respective BP neurons are noted as B and the kernels of the ACL convolutional layer as W. According to equation 46 all incoming BP-red-on activations for the target color are weighted equally and added and the same is done in 47 for all incoming BP-green-off activations. Equation 48 supports an additional opponency mechanism for white and black. In the ACL there are 6 output features for 3 opponent color pairs and the NiN convolutional layer can be noted as $[1 \times 1 \mid 14 \rightarrow 6]$.

The weights for the AC kernels are designed so that negative and positive values contribute unevenly in the amount of excitation or inhibition for a specific color opponency detector. As a basis two constants d and a are used for shaping the opponent weights w_{on} and w_{off} and they control the mixture of the BPL activations:

Excitation mixture hyperparameter	Inhibition mixture hyperparameter	
$d = e - 1 = 1.7183$	$a = \frac{1 - e}{e} = -0.6321$	(49)
Excitation weight	Inhibition weight	
$w_{on} = d \cdot \beta_{on}$	$w_{on} = a \cdot \beta_{off}$	(50)
$\beta_{on} > 0$	$\beta_{off} > 0$	

The analogy $|d/a| = e$ has been intuitively chosen to ensure that excitation is sufficiently stronger than inhibition. Specific adjustments to this are supported by the β_{on}/β_{off} ratio. The ACL synaptic sum is the output of the convolution operation without any bias and its activation function is the basic linear:

$$f_{AC} = f(\boldsymbol{U}) = conv2d(\boldsymbol{W_{AC}} \cdot \boldsymbol{X_{BP}}) \tag{51}$$

An AC cell is a weighted difference of 1D Gaussians using the target color activations minus the opponent color activations, which are e times smaller. It results into a combined response that expresses color opponency in the range of $[-0.6321, 1.7183]$.

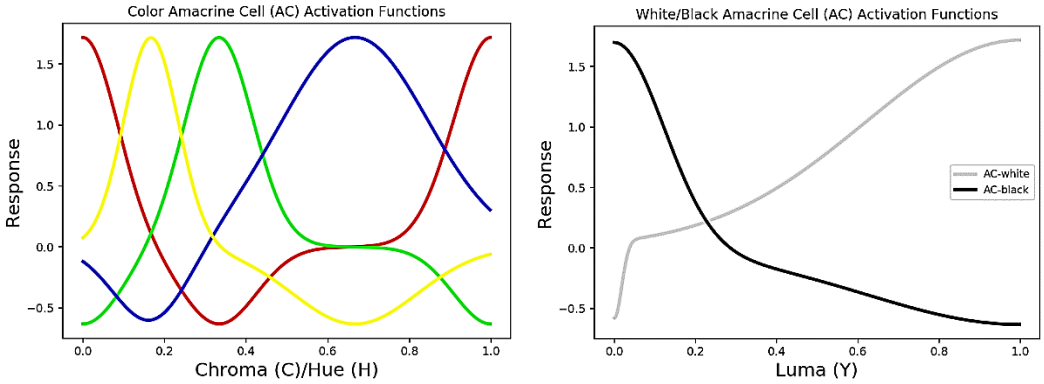

Figure 55: Activation function of the color Amacrine Cells (left) and the white/black Amacrine Cells (right). Each color represent the respective neuron in the ACL.

A combined visualization of both BPL and ACL activations for all the possible values of chroma C, provides insight about the BioCNN implementation of color opponency:

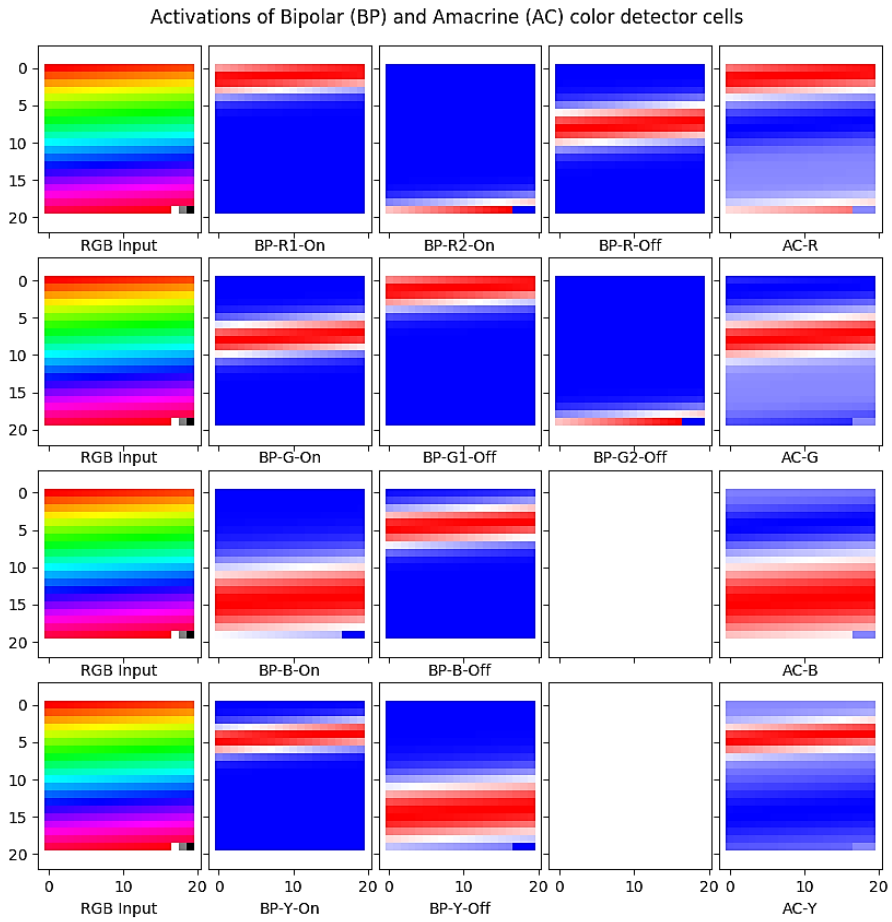

Figure 56: Color opponency in the BioCNN. Column 1 has 1024 different hues of RGB colors. Columns 2, 3 and 4 are visualizing the activations of the BP neurons depending on their target color and function. Blue indicates BP activations close to zero and red close to one. In the last column there are activations of AC neurons that combine two or three BP neurons. Blue values are negative, white close to zero and red positive.

The color opponent mechanism in the first layers of the BioCNN attempts to decorrelate the RGB input components and at the same time create a redundancy of color representation for the image, which is consistent with the biological findings. It can be beneficial for natural scene recognition, following the suggestion that the color opponency mechanism may not be a sole consequence of the trichromatic additive color system of cones, but a property of the natural spectra [241]. Furthermore the BPL-ACL interactions in the IPL introduce a type of non-linearity in the CNN which can be controlled by their hyperparameters, allowing a formal study.

4.3.3.2 Ganglion Cell Layers

The Ganglion Cell Module (GCM) is imitating the receptive field (RF) organization of the retinal ganglion cells in the primates' retina [242]. The fundamental knowledge from Visual Neuroscience is that the RFs of biological ganglia implement an antagonism between the input stimuli that illuminates the center and the input stimuli in the surrounding area.

According to the *center-surround antagonism* model, a color illuminating the RF center and its opponent color illuminating the surround of the RF, generate the strongest excitation of the neuron. When a color illuminates the whole RF it creates a baseline activation. A diffuse opponent color over the RF has an inhibitory response. When opponent color is hits only the center without spreading to the surround, the strongest inhibitory response is generated.

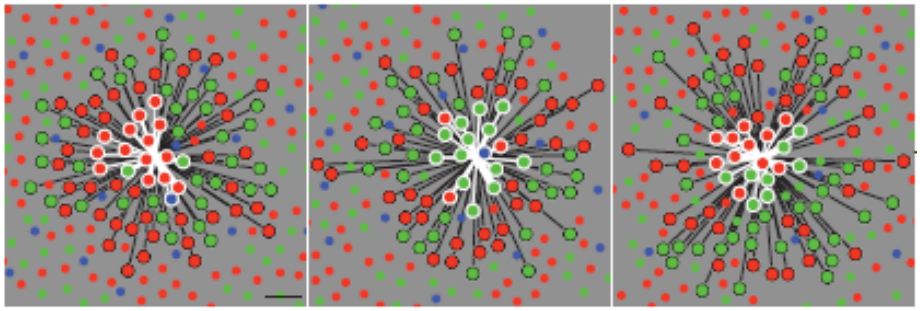

Figure 57: Model of connectivity for the biological ganglion cells with respect to their source cone photoreceptors. This cell implements red-green antagonism. The cones with white oulines are mapped to the center of its receptive field and those mapped in the surround black. This figure also depicts the difference in red,green and blue cone distribution [190].

This center-surround antagonistic functionality is implemented by the convolutional layers of the GCM module, which have specially constructed kernels in order to convolve with a selected pair of opponent BP activations and occasionally with white or black BPs. Through receptive field map engineering, the center and surround RFMs are pre-defined using the Difference of two-dimensional Gaussians (DoG) function [137] [138] as an approximation to the Laplacian of Gaussian (LoG) function [137] that exist in the biological ganglion cells model. The 2D Gaussian function and the DoG are:

$$gauss2d_{\sigma}(x,y) = \left(\frac{1}{2}\pi\sigma^2\right)e^{-(x^2+y^2)/2\sigma^2}$$
$$DoG_{\sigma_c,\sigma_s}(x,y) = gauss2d_{\sigma_c} - gauss2d_{\sigma_s} \quad (52)$$

where σ_c and σ_s correspond to standard deviations for the center and surround Gaussian functions. For a $n \times n$ square, the following 2D Gaussian function places the peak value on its center coordinates $(x_0 = n/2\,, y_0 = n/2)$: $x0, y0 \in \mathbb{N}$ having an λ as the constant for the *full width at half maximum* (FWHM) [243]:

$$g_{\lambda}(x,y) = gauss2D_{\lambda}(x,y) = e^{-4\log 2\cdot\frac{(x-x_0)^2+(y-y_0)^2}{\lambda^2}} \;:\; \lambda = 2\sigma\sqrt{2\cdot\ln(2)} \quad (53)$$

An odd dimension for the square is implied and typically $n \in \{5,7,11,13\}$. The above formula ensures that $\sum_{y=1}^{n}\sum_{x=1}^{n} gaus2D_{\lambda}(x,y) = 1$. For calculating the weights at each position of the RFM a weighted DoG is used in order to model a different contribution of the center's excitation and the periphery's inhibition.

$$WDoG = w_{inh} \cdot (1+\varphi) \cdot g_{\lambda_{exc}}(x,y) - w_{inh} \cdot g_{\lambda_{inh}}(x,y) \; : \lambda_{inh} = \varphi \cdot \lambda_{exc} \qquad (54)$$

The weighting is controlled by the BioCNN layer-specific hyperparameters w_{inh}, λ_{exc} and φ which ensure a larger peak of the excitatory weights concentrated in the center and a wider spread of smaller inhibitory weights in the surround. The inhibition FWHM is φ times larger than excitation FWHM and the amount of excitation $1+\varphi$ times larger than inhibition. The three hyperparameters are inspired by the related Computational Neuroscience model for ganglion cells. They control the balancing of positive and negative values in the 3D center-surround organization for one feature in the 4D convolutional kernel.

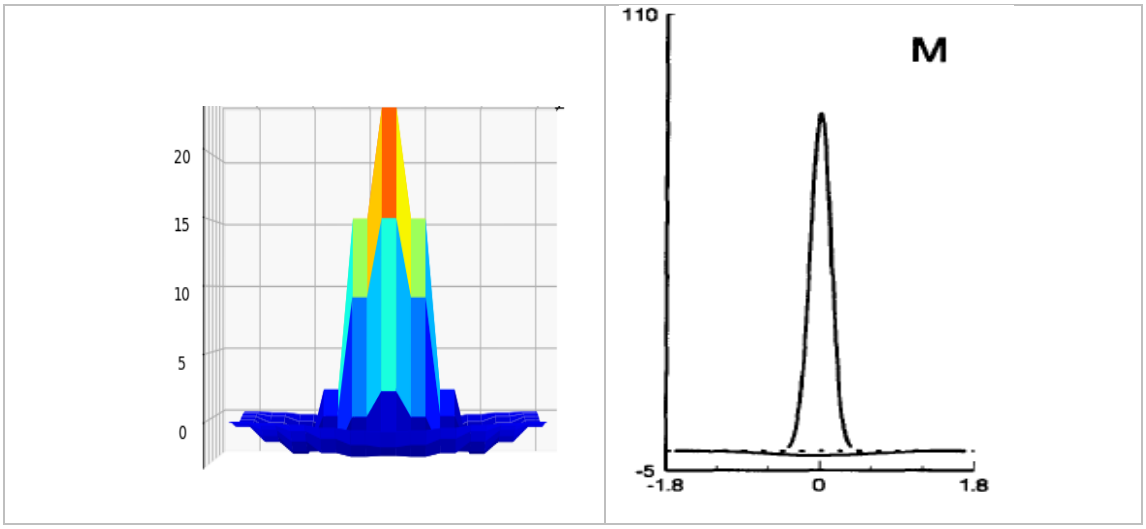

Figure 58: Center-surround model of the artificial ganglion cell (GC) in BioCNN (left) in comparison to the model of the biological magnocellular ganglion cells presented in [242].

Two distinct genders of GCs can be found in a GC layer, according to their response in the presence of a target color in the center or the surround. The color-on-center ganglion cells (GC+) are maximally excited when a blob of target color covers the center and at the same time its opponent color fills the surrounding area. The GC+ are designed so that their center-surround RFMs implement the Mexican hat, or Marr wavelet [137] [244], when they are combined in three dimensions. The second gender is the color-off-center (GC-) which are modeled to be maximally inhibited when the target color hits the center and not inhibited when a blob of opponent color hits the center. Two RFMs are designed so that the GC- creates a baseline excitation when the target color illuminates the surround. Also the maximum excitation occurs, when the opponent color is near the center-surround boundary. The functions that generate the weights for the RFMs in each GC gender are based on the $WDoG$ function that generates the $RFM_{+target}$.

$$RFM_{+target} = WDoG(x, y) \tag{55}$$

$$RFM_{+opponent} = -WDoG(x, y) \cdot \omega \tag{56}$$

$$RFM_{-target} = -ReLU(WDoG(x, y)) \tag{57}$$

$$RFM_{-opponent} = ReLU(-WDoG(x, y)) \cdot \omega \tag{58}$$

The constant ω is another hyperparameter of the BioCNN network that represents a ratio between the amount of excitation caused by the target color's map and the mount of inhibition caused its opponent color's map.

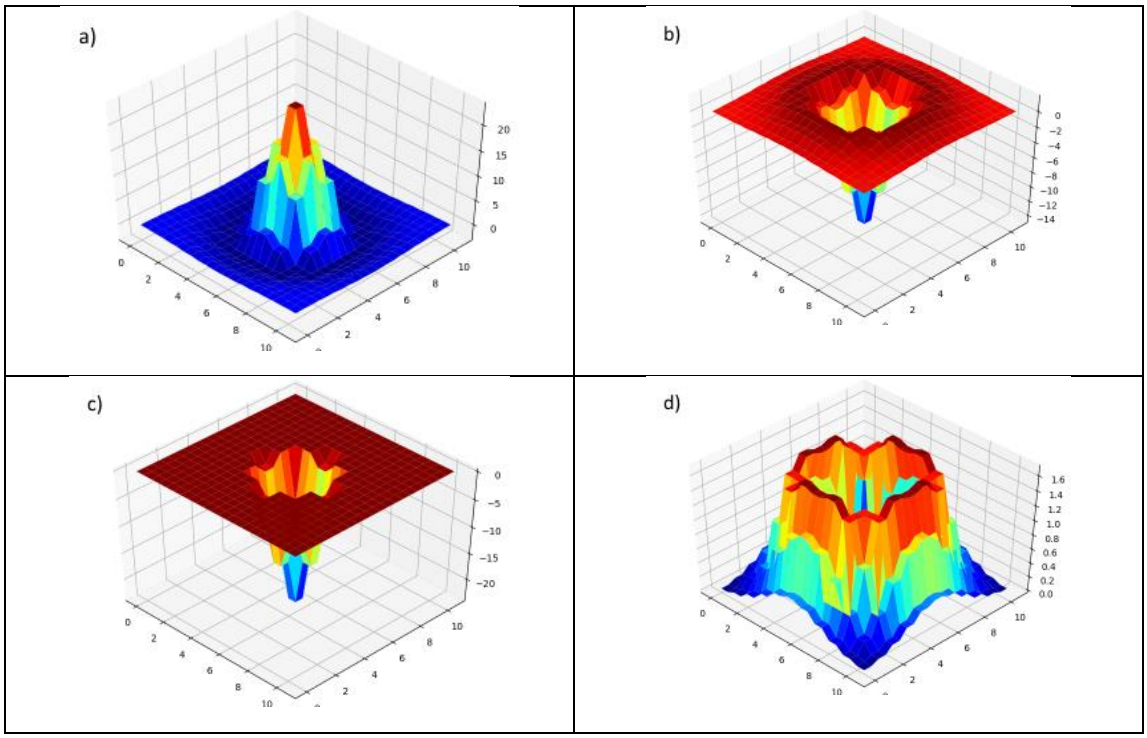

Figure 59: Visualization of Ganglion Cell receptive field weights in a $[11 \times 11]$ convolutional kernel. Blue represent weights close or below zero and red the strongest activations in the map. a) RFM for target color that is ON at center b) RFM for opponent color that is ON at surround. c) RFM for target color that is OFF at center d) RFM for opponent color that is ON at the center-surround boundary and does not inhibit the center.

The pre-defined weights for the center-surround RFMs are generated for each color feature creating a 4D kernel of non-trainable filters. Despite the square-shape of the maps, the center-surround organization is modeled in BioCNN using a rasterized circular disc for the center and the concentric disc that is cropped by the map's square for the surrounding area. The notation for the convolutional kernel is extended to include the center-surround diameters:

4D Ganglion Cell Kernel Notation	$[\, a \times b \sim s \mid m \rightarrow h \,]$: ((c) s)

For the color opponency pair red/green there are 4 different GC types GC-Red+, GC-Red-, GC-Green+, GC-Green- which are convolving on the AC-Red and AC-Green activations. For blue/yellow there are 4 types that are convolving with AC-Blue, AC-Yellow and additionally with a third channel AC-Black for blue and AC-White for yellow. The RFM weights for the third input channel have a constant value of $b = e - 2$ thus adding excitation or inhibition that is caused only by different brightness levels inside the same color. For GC-Blue+ and GC-Blue- the darkness level from AC-Black is convolved and the presence of bright light inhibits their output. For GC-Yellow+, GC-Yellow- the brightness level of AC-White is convolved so that darkness will turn off the yellow color. The choice for using black as input to the convolution of GC-B+ and GC-B- is inspired by the increase in blue sensitivity during adaptation to darkness, known as the Purkinje shift [245].

Additionally to the color sensitive GC types there are grayscale center-surround blob detectors which operate on the AC-Black and AC-White channels. These GCs are implementing a stream of grayscale visual information processing that starts from the rods.

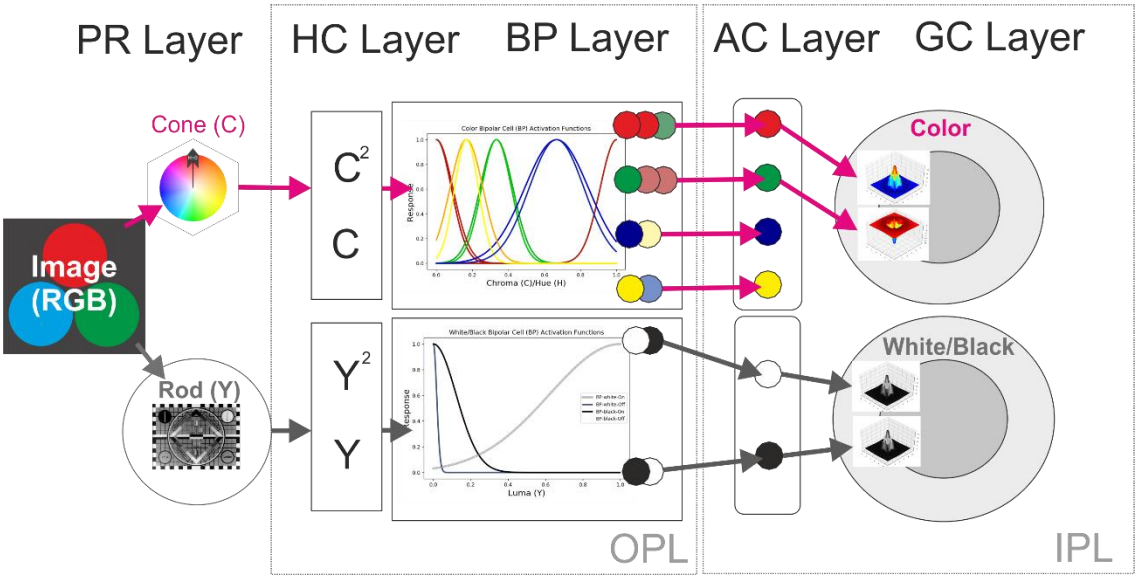

Figure 60: Processing of visual information in the various layers of the ARNN neural network. There are two streams, the color stream that is depicted with magenta arrows and the grayscale stream that is depicted with gray arrows.

In the Ganglion Cell Module there are 12 different clusters of ganglion cells grouped by their color and their center-surround operation which are presented in table 10:

Base Type	Target Color / Input Feature / RF Map			Opponent Color / Input Feature / RF Map			Illumination / Input Feature / RF Map		
GC-R+	red	AC-R		green	AC-G				
GC-R-	red	AC-R		green	AC-G				
GC-G+	green	AC-G		red	AC-R				
GC-G-	green	AC-G		red	AC-R				
GC-B+	blue	AC-B		yellow	AC-Y		black	AC-K	
GC-B-	blue	AC-B		yellow	AC-Y		black	AC-K	
GC-Y+	yellow	AC-Y		blue	AC-B		white	AC-W	
GC-Y-	yellow	AC-Y		blue	AC-B		white	AC-W	
GC-W+	white	AC-W		black	AC-K				
GC-W-	white	AC-W		black	AC-K				
GC-K+	black	AC-K		white	AC-W				
GC-K-	black	AC-K		white	AC-W				

Table 10: Clusters of GC neurons with their corresponding input channels and convolutional kernels.

In each cluster two additional functional properties categorize further their role as feature detectors, sensitivity to direction and spatial extend of the receptive field. It has been found that certain biological ganglia in the mammalian retina, exhibit stronger responses to specific orientations of light hitting their RF centers than to blobs of light. These *directionally selective ganglion cells* (DSGC) are ported into the BioCNN by slicing the convolutional 4D kernel into n_{slices}. The two-dimensional RFMs for center and surround of each color have circular sectors of the inscribed disc at specific angles. The angular step of the circular sectors is calculated as $\theta = 2\pi/n_{slices}$ and can be considered a hyperparameter of the BioCNN network. As feature extractors these DSGCs extract edgelets which correspond to transitions of a color to its opponent at a specific angle.

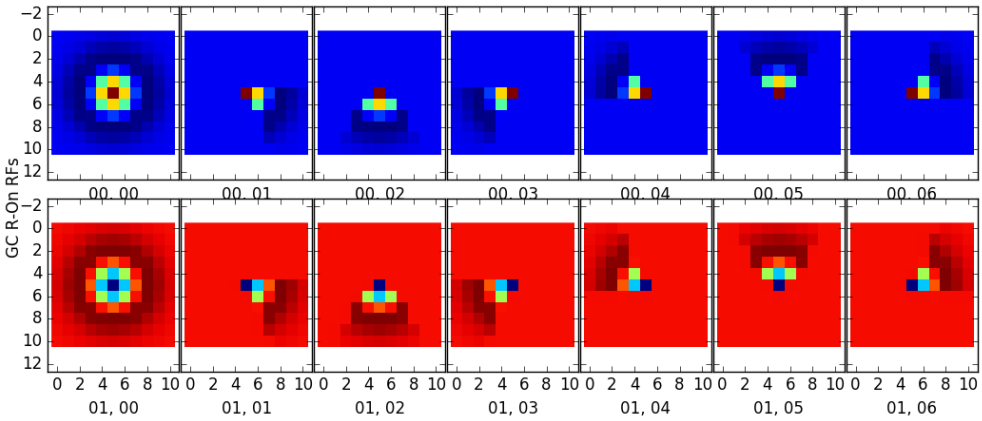

Figure 61: Receptive field maps $[11 \times 11]$: $((3)11)$ for the GC-R+ focused on the red color input channel (top) and on the opponent green color input channel (bottom). Blue depicts values close or below zero and red positive ones. The blob detector has a center of 3 pixels and a total diameter of 11 pixels. The edgelet detectors have a shape of a circular sector of angle $\theta = 60^0$

Furthermore Visual Neuroscience has discovered three groups of biological ganglia that are named after their role in the visual stream. In BioCNN the *parvocellular* GCs (p-GCs), the *magnocellular* GCs (m-GCs) and the *koniocellular* GCs (k-GCs) have different spatial sizes in their receptive fields. In the reference architecture these groups are implemented as three different convolutional layers p-GC, m-GC and k-GC with receptive field sizes of 3×3 , 5×5 and 11×11. Their center diameters in pixels are $d_{p-GC} = 1$, $d_{m-GC} = 1$ and $d_{k-GC} = 3$. The directional selectivity is only implemented for a center of 3 pixels that exist in the largest spatial size, hence the k-DSGC layer.

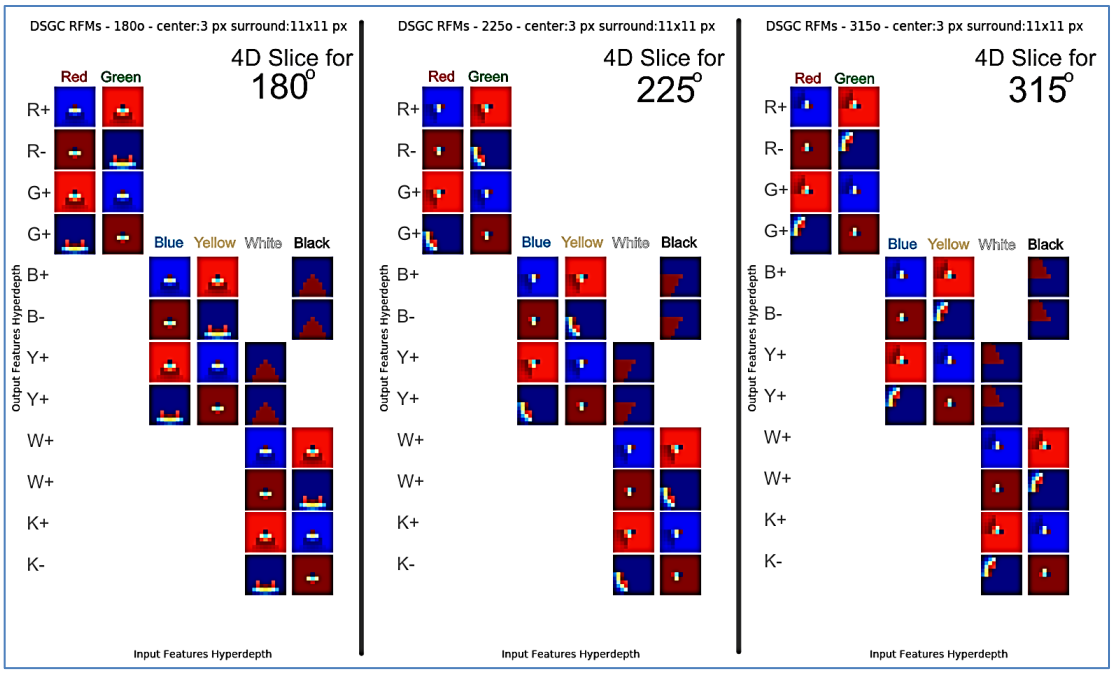

Figure 62: Directionally Selective Koniocellular Ganglion Cells (k-DSGC) for all 12 color clusters. There are three 4D slices showing the RFMs for three different directions of edges. X axis represents the input feature hyperdepth with ACL colors red, green, blue, white, black. Y axis represents the output feature hyperdepth and a respective RFM is visualized for each combination.

The total number of output features of the GCM module, which is the output of the ARNN subsystem, is calculated using the count of features in each group c_{parvo}, c_{magno}, c_{konio}:

$$h_{ARNN} = c_{parvo} + c_{magno} + c_{konio} \cdot (1 + n_{slices})$$

$$c = c_{parvo} = c_{magno} = c_{konio} \xrightarrow{\theta = \frac{2\pi}{n_{slices}}} \tag{59}$$

$$h_{ARNN} = c \cdot (3 + \frac{2\pi}{\theta})$$

For the reference ARNN there are $c = 12$ and DSGC of $\theta = 60^0$ that calculates to a hyperdepth of 108 features extracted from pixel resolution by the pre-defined convolutional kernels. The corresponding layers with center-surround diameters for the respective ganglion cell groups are:

- p-GC : $[3 \times 3 \sim s \mid 6 \rightarrow 32] : ((1)\ 3)$
- m-GC : $[5 \times 5 \sim s \mid 6 \rightarrow 12]\ ((1)\ 3)$
- k-GC : $[\ 11 \times 11 \sim s \mid 6 \rightarrow 84\ ] : ((3)\ 11)$

These are operating in parallel on the same region of the image and their activations are concatenated. The activation function for all the layers is the ReLU and the synaptic sum is the convolution result without any added bias. The use of a rectifier function ensures that a range of positive values will represent the different amounts of excitation of the neuron when target/opponent colors are present on the center-surround RFM. No negative values are generated, but the bias shift problem could be mitigated by the redundancy of opponent pairs like GC-R+ / GC-G+, where only one of the two clusters will generate a strong response.

The retina outputs a 3D tensor of all features for each spatially mapped region of the input image. This image patch can be considered a retinal element or *rexel* that is a squared shape region in a typical CNN which is mapped to pixel elements or pixels of the input image. A rexel is the first image descriptor which contains extracted features from colors and can be of any size. In AlexNet/ZFNet the rexels are found on the activation map of the first convolutional layer. In BioCNN the rexels can be considered the spatial elements of the ARNN output. They express the fine features around a pixel or inside a diameter of pixels with blobs of red/green, blue/yellow, white/black and edgelets of contrast in specific directions inside them. Since a rexel is simply an abstraction of the mapped region of pixels, other shapes beside squares can be evaluated in a future study.

4.3.3.3 Design of GC layers and downsampling in the IPL

There are two different stages where downsampling operations can be performed in the ARNN. In the HCL an early downsampling can reduce a higher image resolution. Nevertheless early downsampling can decrease the quality of the extracted features, discarding fine details that could have been captured by the GCs. The BPL and ACL functionality extract colors for every incoming region without skipping regions and are considered inappropriate for reducing the spatial size.

The proper stage for downsampling is the GCM, where its hyperparameters regulate the spatial reduction and the feature extraction quality. The center diameter in pixels d_{center} and the surround diameter $d_{surround}$ are using odd numbers to place the center in a rasterized RFM. The center-surround analogy $cs = d_{center}/d_{surround}$, takes typically the values $cs \in \{\frac{1}{3}, \frac{1}{5}, \frac{3}{7}, \frac{3}{11}, \frac{3}{13}, \dots\}$.Choosing a stride $s > 1$ is a standard choice that improves the recall speed of the whole CNN network. Having in mind that DSGC functionality operates in a more fine-grained scale, a convolution with a medium stride $s = 3$ and a diameter $d_{kDSGC} = 3$ can create a grid of rexels with k-DSGC centers covering the whole image. This feature extractor can be generally designed as $[n \times n \sim c \mid 6 \rightarrow h]\ ((c)\ n)$ where the ratio n/c creates the Mexican hat wavelets and n controls the downsampling.

Implementing k-DSGC of large sizes is beneficial for detecting larger blobs additionally to small points detected by $[3 \times 3]$ and $[5 \times 5]$ maps. Loss of point detail can occur when $s = 3$ because the p-GCs will not overlap and a choice for a $[3 \times 3 \sim 2 \mid 6 \rightarrow h]$ convolutional layer might increase accuracy in expense of computational efficiency. As depicted in table 10 there are sparse connections between the GCs and the ACs. The feature extraction requires only 2 or 3 out of the 6 input channels. In a more elaborate ARNN architecture a concatenated output of more than 3 convolutional layers each one as $[n \times n \sim s \mid 2 \rightarrow h]$ can be tested for its computational efficiency.

4.4 Artificial Primary Visual Cortex (APVC)

4.4.1 Lateral Geniculate Nucleus

The Artificial Primary Visual Cortex (APVC) is subsystem that is inspired from the biological equivalents of the lateral geniculate nucleus (LGN) [193] and the primary visual cortex or V1 [42], [184]. As in the human visual pathway, the ARNN output stream reaches the LGN cells where it is adapted and fed to the receptive fields of the V1 cells. In the BioCNN model the LGN and the V1 functionality is combined into an assembly of parallel operating convolutional layers. With respect to the LGN, APVC performs the following tasks:

a) Normalizes the GC activations before they reach the upper layers, typically by using min-max normalization to the interval $[0,1]$.
b) Forwards the normalized GC activations directly to the output of the APVC without any further processing, through a shortcut connection.
c) Feeds the normalized GC activations to the V1 simple and V1 complex cell layers.
d) Adapts the normalized GC activations in the learnable LGN convolutional layers.
e) Implements spatial reduction on the assembly's concatenated activation maps.

End-to-end the APVC subsystem handles a concatenated input of fine features from the ARNN, in order to augment it with primitive features. It uses both pre-defined and learned weights and it is a hybrid subsystem placed between the non-learnable ARNN and the fully learnable AHVC. The convolutional layers of the APVC operate in parallel and can implement sparse connection with the previous layers.

The LGN convolutional layers are assembled into a mini-module inside the APVC. There are three different spatial sizes for the convolutional kernels and every input region is convolved with a stride $s = 1$.

$LGN_{tiny}:\quad [3 \times 3 \sim 1 \mid 108 \rightarrow 12]$
$LGN_{small}:\quad [5 \times 5 \sim 1 \mid 108 \rightarrow 12]$
$LGN_{medium}:\quad [7 \times 7 \sim 1 \mid 108 \rightarrow 12]$

Table 11: Convolutional layers of the LGN mini-module. The convolution stride is always one and the input for these layers consists of points, blobs and edgelets.

In each layer a bias is added to the convolution result, and the synaptic sum is used with the ELU activation function, that is described in section 2.4.4.

$$f_{LGN} = f(\boldsymbol{U}) = ELU(conv2d(\boldsymbol{W_{V1}} \cdot \boldsymbol{X}) + \boldsymbol{B}) \quad (60)$$

These layers are the first in the BioCNN architecture that have learnable weights. They are initialized by cloning the pre-defined receptive field maps of the V1 convolutional kernels of the same size. The training process is equivalent to fine-tuning since the weights are transferred to the LGN cells. They start as redundant units on the same input and gradually shaped into different feature extractors through the learning process. The LGN weights are modified by the gradients that are back-propagated down the APVC concatenated output. The output hyperdepth for the LGN mini-module d_{LGN} depends upon the selection of hyperparameters for the V1 mini-module.

4.4.2 Gabor V1 Simple Cells

4.4.2.1 Design of V1 Simple Cells and the 2D Gabor Function

Convolutional neural networks are inspired by the receptive fields of biological neurons found in the V1. The *V1 simple cells* (V1S) generate a strong response to simple line primitives that illuminate their receptive fields and are selective to a specific orientation of the line [246]. All possible line orientations for a region in the retina are handled by a hypercolumn of V1 cells each one expressing having a different directional sensitivity.

The V1S are convolutional layers in BioCNN that detect oriented line patterns formed by points, blobs and edgelets of the ARNN activation maps. There are four different spatial sizes in the V1S mini-module with the corresponding convolutional kernels:

Name	Convolutional Kernel
$V1S_{tiny}$	$[\,3 \times 3 \sim 1 \mid 108 \rightarrow 6\,]$
$V1S_{small}$	$[\,5 \times 5 \sim 1 \mid 108 \rightarrow 6\,]$
$V1S_{medium}$	$[\,7 \times 7 \sim 1 \mid 108 \rightarrow 6\,]$
$V1S_{large}$	$[\,11 \times 11 \sim 1 \mid 108 \rightarrow 6\,]$

Table 12: Convolutional layers of the V1S mini-module. The convolution stride is always one and the input for these layers consists of points, blobs and edgelets. The V1S mini-module extracts line primitives for 6 orientations and 4 lengths.

The stride is typically $s = 1$ is so that all incoming rexels are used as input for the line detectors. The weights in the receptive field maps of V1 kernels are engineered using the normalized two-dimensional Gabor kernel function described in [247].

$$g_{f,\theta,\gamma,\eta,}(x_o, y_o) = \frac{f^2}{\pi\, \sigma_x\, \sigma_y}\; e^{-\left(\left(\frac{f^2}{\sigma_x^2}\right)x'^2 + \left(\frac{f^2}{\sigma_y^2}\right)y'^2\right)} e^{j2\pi f x'} \tag{61}$$

$$x' = x_0\, cosa + y_o\, sina$$
$$y' = -x_o\, sina + y_o\, cosa$$

The hyperparameters for the creation of the V1 directionally selective kernels are the frequency f of the sinusoidal wave plane, a the angle of rotation for the line extractor inside the RFM, σ_χ^2 the variance of the Gaussian function for the x axis and σ_y^2 the variance of the Gaussian function for the y axis. Properly selecting the values of the Gabor function hyperparameters generates RFMs for detecting oriented line patterns or texture patterns,that are used in the pre-defined V1 cells and as cloned as initial values for the trainable LGN cells.

4.4.2.2 V1 Color Sensitive Simple Cells

The input of the V1S cells is activations of the GC cells, which due to the use of the ReLU function and after the normalization are restricted in the positive interval $[0, 1]$. Each V1S hypercolumn of a specific spatial size is sensitive to all possible orientations of a line for an angular step of θ and the count of different cells is $c = \frac{\pi}{\theta}$. For the n^{th} orientation the value of the Gabor function angle hyperparameter will be $a = n\frac{\pi}{\theta}$**.** For reducing the computation cost only $c = 6$ different orientations are detected, with $\theta = 30^{\,0} = \frac{\pi}{6}$.

The V1S kernel has a common RFM map for all input features in order to detect line patterns in any color. The color opponency restricts the co-existence of lines in two opponent colors, e.g. the GC-R+ and GC-G+ cannot have simultaneous strong activations for the same region. On the contrary a line of magenta color will generate responses from both GC-R+ and GC-B+ activations and a line of teal color from GC-G+ and GC-B+ activations.

The V1S module has parallel convolutional layers of different RFM sizes to support different line lengths. The analogies between the square kernel dimensions can specify four different line lengths from the smallest $l = 1.0\ for\ [3 \times 3]$, $l \simeq 1.67\ for\ [5 \times 5]$, $l \simeq 2.33\ for\ [7 \times 7]$ and the largest $l \simeq 3.67\ for\ [11 \times 11]$ almost four times the smallest length. In all V1S layers no bias is added to the result of the convolution operation and the ELU activation function is used:

$$f_{V1S} = f(\boldsymbol{U}) = ELU(conv2d(\boldsymbol{W_{V1}} * \boldsymbol{X})) \tag{62}$$

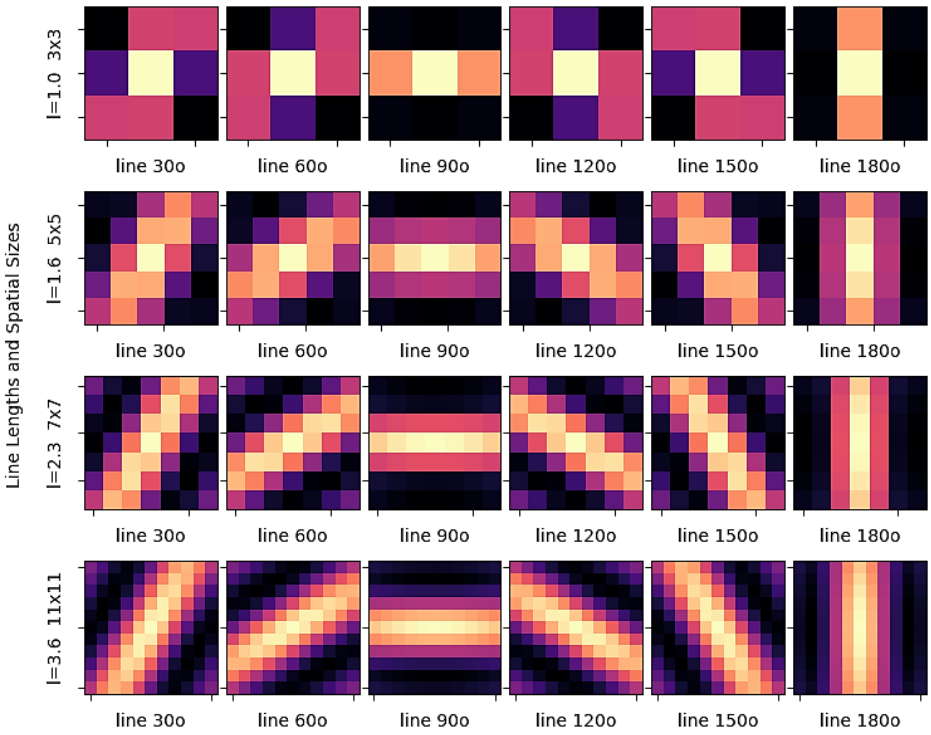

Figure 63: Receptive field maps for 4 different lengths of the V1S line detectors. Line-detectors are designed as Gabor filters and tuned to be selective to 6 possible directions with an angular step of $\theta = 30^0 = \pi/12$

4.4.2.3 V1 Gray Simple Cells

A different group of V1S convolutional layers uses only the grayscale stream from the optic nerve activations implemented by the GC-W+, GC-W-, GC-K+, GC-K-. This ensures line detection with color invariance. There are common RFMs for all input channels with 4 V1S_Gray clusters each one with different spatial sizes and a directional sensitivity to white and black lines. Generally a V1S_Gray convolutional layer is $[n \times n \sim 1 \mid 4 \rightarrow 6]$. This scheme introduces sparsity between input and output features that reduces computational complexity in a processing stage where there is large input hyperdepth and a high spatial resolution. More sparse connections can be implemented in the APVC subsystem, to further reduce its computational cost.

4.4.2.4 V1 Complex Cells

A second group of V1 convolutional layers is designed to detect scatter patterns of GC activations that might occur due to the texture of a region. The *V1 complex cells* (V1C) in the BioCNN respond to entropy in a spatially mapped region and correspond to the second type of neurons found in the biological V1. Using a small frequency $f = \frac{\pi}{2}$ for the Gabor generating function, texture patterns occur in the receptive field maps:

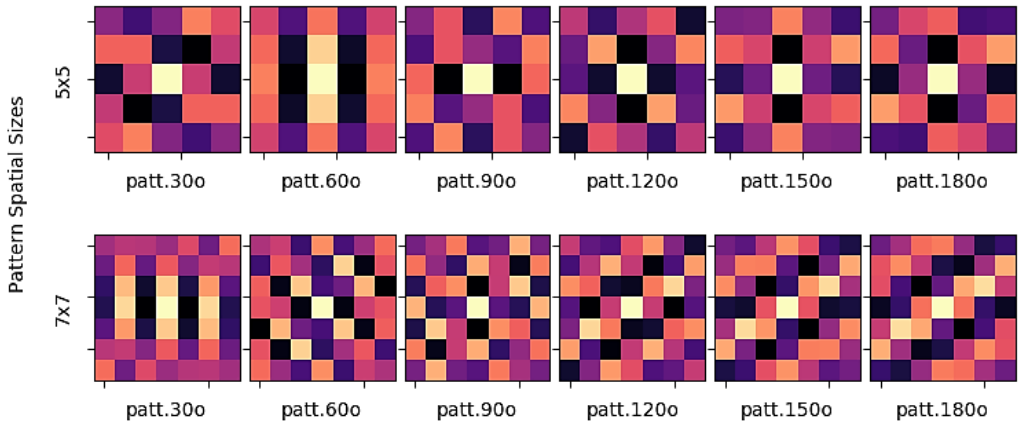

Figure 64: Receptive field maps of the V1C scatter pattern detectors. Patterns are created by manipulating the parameters of the Gabor 2D kernel function.

The V1C layers follow the same convolutional neuron model with the V1S. They serve the purpose of producing an increased activation value for an input region of increased entropy.

These texture extractors are a simplistic approach taken by this thesis that can be extended using more powerful methods that use Gabor or other kernel that extract texture. The logic can be extended to other primitives like boxes, triangles adding a third category of neurons the *V1 hypercomplex cells* (V1C) that are also found in the human brain. The V1C functionality is not following biological equivalences, but is based on the intuition that having a texture filter bank might prove beneficial for the later processing.

4.4.3 Characteristics of BioCNN stem output

The ARNN in combination with the APVC are the stem of the BioCNN network. The upper convolutional layers receive their concatenated activations in order to extract the local, higher and global features of a given image. Through the feature extraction operations of the hybrid stem, there are several features that emerge by design. Using a count of $c_{LENGTHS}$ for different spatial sizes of V1S RFMs, an angular step θ, a count of $d_{GRANULARITIES}$ of different texture patterns, the hyperdepths can be calculated as:

$$\begin{aligned} h_{ARNN} &= 108 \\ h_{V1S} &= (c_{LENGTHS} + d_{GRANULATIES}) \cdot \frac{\pi}{\theta} \\ h_{LGN} &= (c_1 + d_1) \cdot \frac{\pi}{\theta} \qquad c_1 \leq c_{LENGTHS},\ d_1 \leq d_{GRANULATIES} \\ h_{V1GRAY} &= (c_2 + d_2) \cdot \frac{\pi}{2} \qquad c_2 \leq c_{LENGTHS},\ d_2 \leq d_{GRANULATIES} \end{aligned} \tag{63}$$

The incoming ARNN tensor is augmented with the APVC output using a non-residual shortcut connection.

$$D_{APVC} = D_{ARNN} \cup D_{V1S} \cup D_{LGN} \cup D_{V1GRAY} \quad (64)$$

The total features extracted by the stem are h_{APVC} and equal to the sum of:

$$h_{APVC} = h_{ARNN} + h_{V1S} + h_{LGN} + h_{V1GRAY} \quad (65)$$

The reference APVC has the following selection of structural hyperparameters $c_{LENGTHS} = 4, c_1 = 3, c_2 = 1$, angle $\theta = 30^0 = \frac{\pi}{6}$ and $d_{GRANULARITIES} = d_1 = d_2 = 2$. This creates a hyperdepth of $h_{APVC} = 108 + 36 + 30 + 24 = 198$ features in the 3D output tensor of the APVC.

The slice of output that corresponds to GCM activations, contains tiny features like a pixel of a target color that is surrounded by 8 opponent color pixels or crosses of 5 pixels that are surrounded by 20 opponent color pixels in a 5×5 area. Additionally blobs of 3 pixel diameter in specific color with opponent surrounding color are detected in an 11×11 area. Furthermore a transition from target to opponent color at the boundaries of the center-surround receptive field can be detected in 6 directions. The parallel V1S layers extract primitive oriented line features of four different sizes. The V1C layers generate activations on texture patters that are constructed using the Gabor function. Due to the time restrictions of this thesis, a visualization of the LGN layer activation maps using the technique in [80] was not made possible. This could have provided insight about the learned features in the BioCNN stem output and compare their initial state with their final. Spatial downsampling can be performed on the APVC output, before a computationally expensive convolution on its increased hyperdepth.

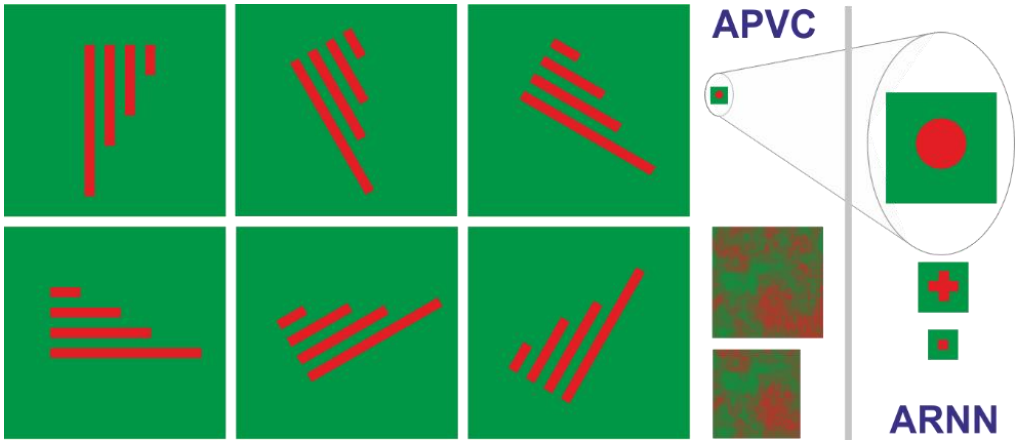

Figure 65: Features extracted by the BioCNN stem. Left: Primitive features extracted by the APVC. Left: Fine features extracted by the ARNN.

4.4.4 Higher Visual Cortex Convolutional Layers

The Artificial Higher Visual Cortex (AHVC) subsystem in the BioCNN architecture is built on top of the hybrid stem with ordinary convolutional layers or modules. Until this level the spatial resolution has not been significantly reduced, potentially keeping important fine details from the input resolution of the image. The first convolutional module that receives the APVC output has a potential input hyperdepth of $h_{APVC} = 198$ features and should implement a significant downsampling that is needed for computational efficiency. There can be a NiN convolutional layer $[1 \times 1 \sim 1 | 198 \rightarrow h_{small}]$ before a CONV layer $[3 \times 3 \sim 2 | 198 \rightarrow h_{big}]$ followed by a MAXP $[3 \times 3 \sim 2]$ where $h_{small} < h_{big}$ for reducing the computational cost.

The total number of trainable convolution modules can be minimized to 3-4 before the classifier. These higher-level feature extractors can be considered to model the higher visual cortex starting from V2 [184]. The first layers of the AHVC are local feature extractors while the later cover broader areas and the last one global features of the image. The quality of the extracted features in these different resolutions could be evaluated for use in CBIR. A study could compare them against features that are extracted by layers in state-of-the-art architectures. Using more than one image collection and transfer learning can help to reach safer conclusions.

Furthermore various architectures can be tested using a BioCNN stem in a future research effort that will investigate potential benefits. Very deep networks, wider feature hyperdepths, Inception architectures and ResNets can be tested as the AHVC subsystem with the target of reducing computation costs while keeping the same performance.

5 Creating a VFL Vocabulary from BioCNN Features

5.1 Introduction

From the perspective of software engineering, a complete DN-CBIR system can be considered to have three parts or tiers. The tier that organizes the machine learning logic has three distinct modules implementing methodologies from the respective domains. The Deep Neural Network Module or Visual Features Extraction Module (VFE) has a CNN which extracts features from images and is described in section 5.2. The Data Mining Module or Visual Features Language Module (VFL) in section 5.3 creates the language using a hierarchy of clusters that translates the regional feature descriptors into visual words. In section 5.4 the Information Retrieval Module of Visual Memory Module (VMM) converts an image, which has been recalled through VFE and quantized through VFL, into visual documents and booklets of fabricated words. These textual documents can be processed by a text IR pipeline and stored in the reverse-index of a search engine.

5.2 Convolutional Layer Activations as Features

5.2.1 Semantics of Neural Activations

The image is recalled through a BioCNN and features are extracted from the RGB values of its pixels. The learned state of the neural network, which consists of pre-defined and learned parameters, has been previously saved in the data tier of the system and is loaded by the VFE module. The resulting activations for a given image should be deterministic to

support the generation and operation of a Visual Features Language and for this reason LCN and BN normalizations that have thresholds in the denominator are avoided. Due to the depth of the neural network architecture cumulative arithmetic errors can emerge at the last layers, who are typically selected to provide features due to their lower spatial resolution. Each location in an activation map of the CNN has a vector of activation values which is used as the feature descriptor of a squared-shape region. The hyperdepth of features in each activation map creates different semantics. In the BioCNN stem the semantics of the engineered features are known by design. In the AHVC they express composite visual information and are more difficult to comprehend even when visualized.

Different semantics exist for different layers with regard to the *spatial granularity* [248]. This term defines how fine or how general is the image representation at a given layer of the CNN. In the first layer there is a pixel-level granularity and fine-features are extracted. In the last ones there are large square areas of pixels containing global features. A rexel, as defined in section 4.3.3, contains the fine features of a mapped region of pixels. These are extracted by the GCs and augmented with the V1-LGN features. At the APVC output the mapped spatial area will be still that of a rexel, due to convolutions operations with stride $s = 1$. The features of the BioCNN stem can be fine-grained depending on the choices for downsampling. The ratio of around $1/4$ is a typical choice found in AlexNet and ZFNet using $s = 4$ or two consecutive layers with $s = 2$. Halving the image resolution at the APVC output resolution comes with an increased computational cost but more accurate representations. A middle solution suitable for the design of the DSGC neurons in BioCNN is a downsampling ratio of $1/3$ in combination with a center diameter of 3 pixels which keeps some details as part of the DSGC activations.

A redundancy of visual information in the rexel descriptor is created by the combined point and line features that are extracted at different levels of the BioCNN. These are extracted by a variety of filters and non-linear transformations that increase the expressiveness of the internal representation and forwarded to the first fully-learnable convolutional module of the AHVC. The broader area features can be found in the middle layers of the architecture and the global features of the image at the last convolutional layer. The values of global features would be tightly coupled to the architecture of the classifier and related to the categorical probabilities expressed by the Softmax output layer.

The AHVC layers may have different spatial granularities and completely different feature semantics. Feature fusion can aggregate these values into descriptors or create multiple VFL representations of the image, each one for a different convolutional layer.

5.2.2 Assemblage of the Descriptor Set

The 3D activation volume Y_i: [$n_i \times n_i$ | h_i] of a specific convolutional layer i is used to provide the feature descriptors for all regions of an image at a specific level of granularity. Flattening the $n_i \times n_i$ activation map assembles a set of n_i^2 descriptor vectors for the image at this spatial granularity. The total count of descriptors for a collection of m known images are $c_l = m \cdot n_i^2$. The spatial resolution at the last layer l of a CNN architecture is typical decreased to $n_l \leq 10$, but the combination with a medium-sized image collection for example $m = 3000$ images can lead to a large count of $3000 \cdot 100 = 300\,000$ descriptors of dimensionality h .

This brings into focus the problems that might be caused by *the curse of dimensionality* and additionally the computational expenses for the creation of the VFL vocabulary. The solution is to reduce the dimensionality with the use of PCA so the resulting dimension d_i will be used for Hierarchical k-Means clustering. The implementation of feature fusion in the BioCNN-CBIR is not aggregating the activations of different layers in the descriptor vectors, but has three different descriptor sets that correspond to the last three convolutional modules l , $l-1$ and $l-2$:

$$\begin{aligned} G &= \{\, d_l \,\} && : |\{\, d_l \,\}| = m \cdot n_l^2 \\ M &= \{\, d_{l-1} \,\} && : |\{\, d_{l-1} \,\}| = m \cdot n_{l-1}^2 \\ L &= \{\, d_{l-2} \,\} && : |\{\, d_{l-2} \,\}| = m \cdot n_{l-2}^2 \end{aligned} \tag{66}$$

The description sets G, M and L will be used to create three different VFLs the $GVFL, MVFL$ and $LVFL$ respectively. The letter G stands for global, M for medium and L for local spatial granularities.

5.3 High Dimensionality Clustering on Large Descriptor Sets

5.3.1 Hierarchical k-Means

An unsupervised clustering method is a standard choice to create the VFL vocabulary. The learned cluster model will act as the layer-specific vocabulary for the visual words. At an early stage of the current work the problems from a high count of descriptors of high-dimensionality have emerged. The "vanilla" k-Means algorithm could not be applied and the use of the mini-batch version of k-Means described in [249] was decided.

The research for suitable variations of k-Means followed the requirements of computational efficiency for high dimensions, fair quality and simplicity in their application. It concluded that the Hierarchical k-Means (HkMeans) [151], which is an extension of the Bisecting k-Means originally proposed in [165], is appropriate for creating the VFL from the activation maps of a CNN layer. With a constant and relatively small branching factor b the clustering model generated by HkMeans resembles a tree and the set of descriptors is split to b branches. For a given descriptor a different cluster is assigned in each level of the hierarchy, creating a path to the leaf cluster, the *cluster path*.

If we think of natural language words, the most significant syllable is the first and the word's suffix is the less significant syllable. On the contrary the first assigned cluster in a cluster path represents a more general grouping of descriptors. The formation of clusters in HkMeans uses the most common characteristics in the root and as it climbs on the branches specificity increases in the sub-clusters. Thus the first cluster in a cluster path is the less significant for the visual word and the last cluster in the path the most significant. Following the paradigm of a natural language the visual word would be the reversed cluster path.

A bio-inspired abstraction for this hierarchy of clusters with increased significance in the innermost level, is the natural process of forming roots, which expand in the ground to rootlets, in order to collect the needed nutrients. Observing a plant's root it is apparent that there is no constant branching factor and it is usually very large at its top. A variation of the HkMeans method is to have a different branching factor for each level of the hierarchy. The algorithm can be named as *Rooting k-Means* (RkMeans) as an abstraction of the natural rooting process. The hierarchy of clusters starts with the first k clusters that are learned by k-Means where $k = RF_1$ the first level's *rooting factor*. Then it expands into rootlets and sub-rootlets with $RF_2, \dots, RF_n$ up to the level n of the hierarchy where the terminal node with the most significance is the *nutrient node.*

5.3.2 Rooting k-Means

The following pseudocode illustrates the recursive algorithm of Rooting k-Means (RkMeans) that is used to create a cluster path. It starts from the least significant circular-shaped clusters to the most significant terminal nodes in the hierarchy.

Algorithm 1 : Rooting k-Means Clustering

```
function RootingkMeans( Data, VFLVocabulary ):
    RecurseClustering ( 0, Data, Hierarchy = VFLVocabulary)
-------------------------------------------------------------------------------
function RecurseClustering ( Level, Data, Hierarchy ):
    if Data.Count > RootingFactor[ Level ]:
        Clusters  ← MinibatchKMeans( Data, k = RootingFactor[ Level ] )

        Hierarchy.AddChildrenAtLevel( Level, Clusters )

        for I = 1 to RootingFactor[ Level ]:
            DataForCluster[ I ] ← Clusters[ I ].AssignAndSegregate( Data )
            RecurseClustering( Level + 1, DataForCluster[ I ], Hierarchy )
    else:
        for I = 1 to Data.Count:
            Hierarchy.AddChildrenAtLevel( Level, Data.Indexes )
```

In each level l of the recursion, k-Means clustering with $k = RF_l$ is performed on a subset $Data_i$ of the dataset adding k child nodes to the hierarchy. Clusters are assigned to $Data_i$ and the data points are segregated into $Data_{ij}$ for each child node $j = 1..k$ .The resulting fragments of data are pushed onto the next level of recursion for each corresponding child node. When the remaining data points at a level of the hierarchy are less than the number k of the k-Means algorithm, their respective indices $i = 1,2, \dots k-1$ are used as the cluster labels. This condition terminates the recursion and may lead in the creation of an unbalanced tree, having terminal nodes below the maximum depth of the hierarchy.

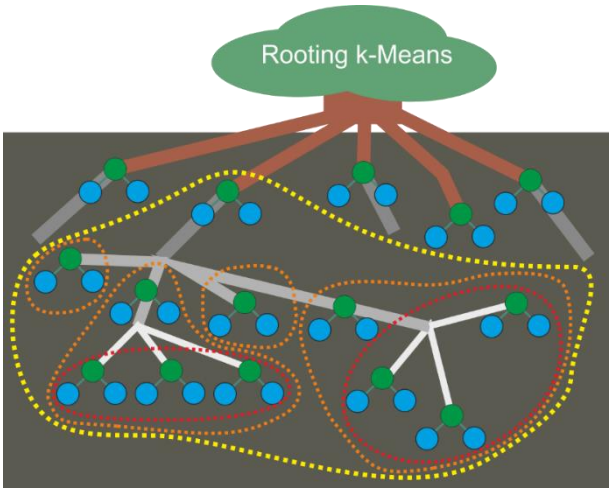

Figure 66: Illustration of the Rooting k-Means hierarchical clustering model with 3 levels and rooting factors $RF = \{5,4,3\}$. The 2nd rootlet from left (brown) expands to 4 sub-rootlets (light gray) and the 2nd sub-rootlet expands to 3 terminal nutrient nodes (white).

5.3.3 Visual Word Assignment

The VFL vocabulary is the set of all possible cluster paths in the hierarchical cluster model. A visual word is the item in this collection which represents a specific cluster path. A given descriptor can be assigned a visual word of the vocabulary through a recursive cluster assignment algorithm.

Algorithm 2 : Visual Word Assignment using a Hierachy of Clusters

```
function VisualWord( Desciptor, VFLVocabulary ):
    VWord = { }
    RecurseAddingSyllable ( 0, Desciptor, VWord, Node = VFLVocabulary.Root)
    VWord = VWord.Reverse()
    return VWord
-------------------------------------------------------------------------------
function RecurseAddingSyllable( Level, Descriptor, byref VisualWord, Node ):
    if  Level < Hierarchy.MaxDepth:
       ClusterIndex ← Node.Clusters.AssignTo( Descriptor )

       if  ClusterIndex is NotFound:
           VisualSyllable ← Null

           for I = Level to Hierarchy.MaxDepth:
               VisualWord = VisualWord ∪ VisualSyllable
       else:
           VisualSyllable ← Node.Clusters[ ClusterIndex ].Label
           VisualWord = VisualWord ∪ VisualSyllable
           ChildNode = Node.Clusters[ ClusterIndex ].Node

           RecurseAddSyllable( Level + 1, Descriptor, VisualWord, ChildNode)
```

Given a descriptor vector and a node in the hierarchy, the algorithm assign the index with the least distance between the cluster centroid and the vector. The label of the cluster is added to the cluster path that is an ordered set of the visual word's syllables. The corresponding child node is used to determine the next and more significant cluster for the same descriptor. If no sub-clusters exists in a level of the hierarchy, the visual word is right padded with a syllable representing a non-existing cluster or *null cluster*. This is repeated to the maximum depth of the hierarchy in order to create visual words with the same number of syllables. After the end of the recursion the order of clusters in the path is reversed.

Formally the descriptor $d_{x,y} = [v_1, v_2, \dots v_h\,]$ holds the values v_i of features in a hyperdepth h for specific point (x, y) of the feature map. It is assigned a cluster path through the recursive algorithm 2, where l is the outmost level in the hierarchy. The visual syllables are represented by the clusters c_l and are ordered in the visual word vw by their significance:

$$vw = f_l(f_{l-1}(\,d_{x,y})) = \{\, c_l\big(d_{x,y}\big), c_{l-1}\big(d_{x,y}\big), \dots, c_2(d_{x,y}), c_1(d_{x,y})\} \tag{67}$$

5.3.4 Extensibility into a Visual Features Language Grammar

5.3.4.1 Handling images as written language.

The ordered set of clusters that is assigned to each descriptor can create interesting analogies with natural languages. Clustering with k-Means create multi-dimensional circular based-shapes because of the distance criterion for the clustering algorithm [250] . These first clusters are formed by descriptors which have small distances, while the later are formed by differences between descriptors of the same group. The analogy to natural language for the first cluster is the suffix of a word, which is the most common part. In text retrieval *stemming* [128] discards common suffixes an can keep the root of a word-term. This is done to increase the retrieval performance of an IR system. A hypothesis for visual words is that they can potentially have "roots" that are somehow captured by the VFL vocabulary. Thus stemming of visual words will group similar features under the same visual word terms in the IR model, impacting the term frequencies used by the weighting scheme. This is an interesting aspect that can be researched in order to discover a universally applicable VFL.

Moreover, the text retrieval takes into account he Zipf's law and ignores the common words in a language from both the collection's index and the queries. The removal of *stopwords* [128] can bring objectiveness into the calculation of term weights. The list of words ignored by the IR system is called stopwords list. The common visual words in image representations can be considered as *visual stopwords* and their exclusion from the CBIR index can be investigated whether it plays a role to the image retrieval performance.

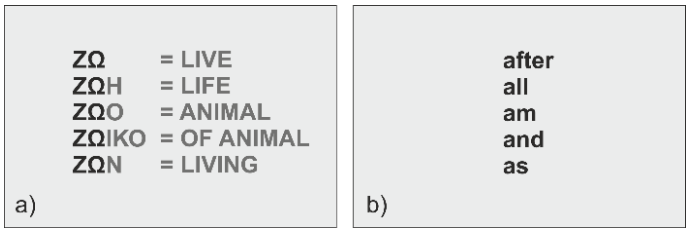

Figure 67: Left: Stemming of Greek words that belong to the same subject using their common root word, the verb "live". Right: English stopwords starting from "a".

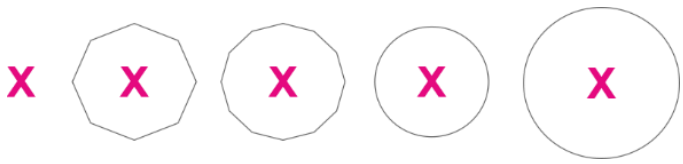

Figure 68: Conceptual stemming of visual words keeping a common root visual feature. The magenta X-shape is the common feature in all 5 samples and visual word stemming would hypothetically remove the surrounding gray shapes of a hexagon, a dodecagon and two circles of different radius.

5.3.4.2 Detection of Visual Stopwords from VFL Statistics

In the BioCNN-CBIR pipeline visual stopwords are determined using the VFL statistics of the indexed image collection. A total count of n visual words exists in the whole image collection. For each word its frequency wf_i in the language is counted and a set of frequencies $wf = \{wf_i\}_{i=1..n}$ is constructed. The mean $\mu = \overline{wf}$ and the standard deviation $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(wf_i - \mu)}$ of the frequencies are calculated for the VFL. Visual stopwords can be considered a count of s words that satisfy the following criterion:

$$sf_i \geq (\alpha \cdot (\mu + \sigma) - 1) \quad : sf_i \in \mathbb{N} \quad \mu, \sigma \in \mathbb{R} \tag{68}$$

The constant a is a factor that defines how many times more frequent is the visual stopword compared to uncommon visual words observed in the VFL statistics.

The set $sf = \{sf_i\}_{i=1..s}$ is the visual stopwords list and will be used to re-index the image collection. Removing the visual stopwords equals to ignoring regions of convolutional layer activations, that are clustered under the same rootlet of the RkMeans hierarchy. This can have potentially positive and negative effects. If the regions are common background patterns like for example water texture in sea, floor or wall the remaining representing features significance in the collection will increase. On the other hand they can be useful features that the clustering process has organized into the same group, for example a circular shape of a wheel that is common in photographs of vehicles. Removing such stopwords will hurt the overall performance of the CBIR system.

Figure 69: Conceptual visual stopwords removal. The downsampled image representation has 96×96 regions each one represented by a visual word. If the features on the wall (lime) and the floor (cyan) are considered visual stopwords their removal will make these areas ignored by the system. The remaining visual information of the scene will have increased significance in the image collection.

5.4 Indexing the Image as a Visual Document

5.4.1 Composing a Visual Document

A *Visual Document* (VDOC) created by the VFL module is a 2D matrix of visual words, each one of them describing the visual information in the underlying image region of its position in the matrix. The $a \times b$ regions of a BioCNN layer activation map are described by the visual words in the VDOC:

$$VDOC = \begin{bmatrix} vw_{11}, vw_{12} & \cdots & vw_{1b} \\ \vdots & \ddots & \vdots \\ vw_{a1}, vw_{a2} & \cdots & vw_{ab} \end{bmatrix} \quad (69)$$

This image representation can support spatial relationships between the visual words, which can potentially be exploited for discovering relations of neighboring visual features. The visual document in this thesis is simply used as an orderless list of visual words, according to the BoVW model. The VMM module supports multiple VDOCs per image as an implementation of CNN feature fusion. Each VDOC is composed in a different VFL that has been derived from the activations of a different layer in the CNN architecture. A booklet, for example a user manual, contains instructions on the same topic in different languages. The *visual booklet* (VBOOK) will contain a section for each VDOC in a different VFL. The GVFL, MVFL and LVFL are the three standard languages created by the activations of the last three convolutional modules of the AHVC. The VDOC sections of a VBOOK will be indexed as different fields by the IR engine.

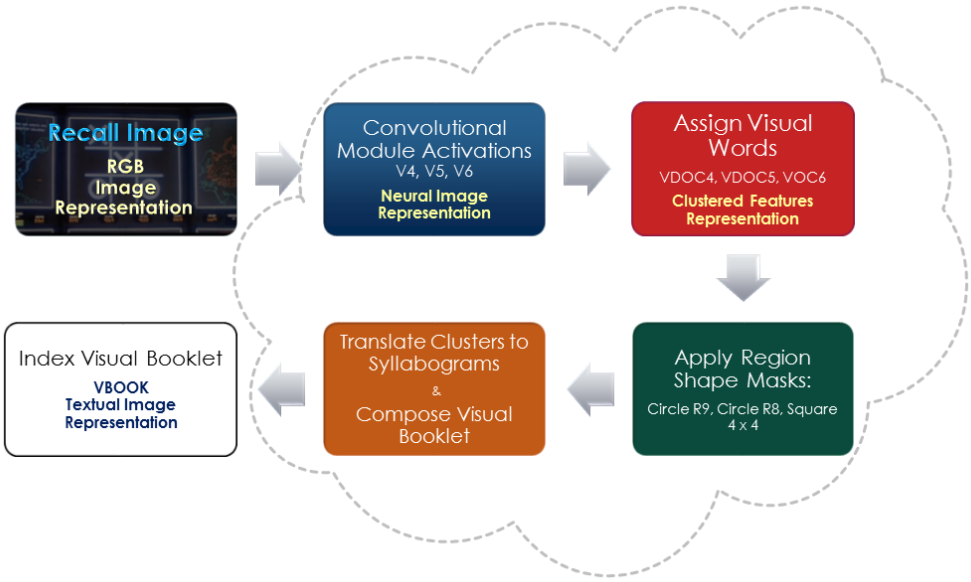

Figure 70: The BioCNN-CBIR pipeline. An RGB color image will be translated by the modules: VFE (blue), VFL (red) and VMM (green) into a text document containing visual words. VDOCs 4, 5, and 6, correspond to the activations of the convolutional modules V4, V5 and V6.

5.4.2 Using text for the Visual Document image representation

Textual descriptions are easier to understand and fit in our perception of natural language. Using visual documents as 2D matrices of ordered integer sets provides little insight about the VFL. As an improvement the visual words in the VMM module are represented by textual words, thus the image is converted from its initial pixel representation into a text document. The clusters at each level of the hierarchy are given textual labels from a set of fabricated natural language syllables. To enrich this representation, these *syllabograms* contain at least one consonant and a vowel and different sets exist for different positions in the visual words. The clusters that belong to the root of the hierarchy are assigned syllabograms that resemble natural word suffixes. Simple key-value pairs $\left\{ c_i, syllabogram_i \right\}$ can be used for the cluster labels c_i. In a more elaborate scheme these pairs can be created using a combination matrix with vowels and consonants. The matrix in the following example has 5 vowels and 6 consonants, supporting 30 clusters with 30 syllabograms.

	k	t	p	m	r	s
a	$c_1 = "ka"$	$c_2 = "ta"$	$c_3 = "pa"$	$c_4 = "ma"$	$c_5 = "ra"$	$c_6 = "sa"$
e	$c_7 = "ke"$	$c_8 = "te"$	$c_9 = "pe"$	$c_{10} = "me"$	$c_{11} = "re"$	$c_{12} = "se"$
i	$c_{13} = "ki"$	$c_{14} = "ti"$	$c_{15} = "pi"$	$c_{16} = "mi"$	$c_{17} = "ri"$	$c_{18} = "si"$
o	$c_{19} = "ko"$	$c_{20} = "to"$	$c_{21} = "po"$	$c_{22} = "mo"$	$c_{23} = "ro"$	$c_{24} = "so"$
u	$c_{25} = "ku"$	$c_{26} = "tu"$	$c_{27} = "pu"$	$c_{28} = "mu"$	$c_{29} = "ru"$	$c_{30} = "su"$

Table 13: Syllabogram hash for the labels of 30 clusters

A three-syllable visual word $vw = \{ c_3(d_{x,y}), c_2(d_{x,y}), c_1(d_{x,y}) \}$ will resemble a word of a natural language, for example $vw = \{8, 5, 24\} = \{\text{te}, \text{ra}, \text{so}\} = te - ra - so$. The VDOC representation is a text document with a grid of image regions reproduced by using white space separators between visual word columns and carriage returns for the different rows. The VDOC representation for a 3×3 activation map would read as:

$$vw_{11} \;\; vw_{12} \;\; vw_{13}$$

$$vw_{21} \;\; vw_{22} \;\; vw_{23}$$

$$vw_{31} \;\; vw_{32} \;\; vw_{33}$$

This representation has a dual purpose. The first is to gain insight for similar features in regions of the image by simply inspecting the document for same or similar visual words. The

second and more important is that an image representation in text can be used by standard text retrieval search engines and weighting schemes. This also gives the ability to automatically retrieve the statistics of the VFL after the image collection is indexed, in order to detect visual stopwords.

Furthermore textual VFL can empower the BioCNN-CBIR model with additional features that can be implemented in the VMM module. The most notable of them is taking into account the position of the visual word in context with the use of sequences of words, the *n-grams* [128]. The *visual bigrams* could be composed by a visual word and its neighboring left, right, top and bottom visual words. Rotating the grid of regions by 90^0 to use the visual document's transpose matrix $VDOC^T$ before indexing an image, is a simple way for extracting the vertical visual bigrams. Taking into account the proximity of visual features opens an interesting perspective for future research.

5.4.3 Indexing an Image Collection

Image retrieval can be classified to two approaches, the *discrete* that is analogous to text information retrieval (IR) and the *continuous* approach that is similar to nearest neighbor classification [120]. The first is using inverted indexes and similarity measures from the IR domain and the second vector representations of the image compared with distance metrics. This thesis follows the discrete approach and reuses the established text-IR methodology.

The image collection is converted into n VDOC textual representations suitable to be indexed by a text search engine. The engine creates an *inverted index* [128] that is stored in the data tier of the BioCNN-CBIR system. It is called reverse, because straightforward words are belong to documents and an inverted index implements the opposite, a lists of documents for each word-term. For CBIR systems each visual word is a term and the key-value pair entry of the inverted index can be defined as:

$$tl_i = \{\, vw_{i\,,} \{p_1, p_2, \dots, p_n\}\} \quad i = 1..c \tag{70}$$

For each key visual word vw_i there is list of the visual documents that contain it, expressed by their respective *postings* p_j [128]. There are c index entries for a visual vocabulary that consists of c distinct visual words. For indexing efficiency the postings list is usually sparse containing postings only for documents that contain the terms. A posting holds the pointer to the visual document [128] d_{ij} for a visual word vw_i, a count of its occurrences c_{ij} inside this document and the location of the word's appearance in the text l_{ij}.

$$p_j = [d_{ij}, \quad c_{ij}, \quad l_{ij}] \tag{71}$$

For implementing CNN-feature fusion in indexing, we consider a VBOOK as a structured text document with VDOC fields. Each section corresponds to a specific field in the IR index. The postings have an additional f_{ij} which indicates the field that the term is found in the document.

$$p_j = [d_{ij}, \quad c_{ij}, \quad l_{ij}, \quad f_{ij}] \tag{72}$$

In order to handle different semantics of features in different convolutional layers, a different VFL will be indexed in a different field, like multiple translations of a natural language text. This can be easily implemented in the VFL module, using distinct syllabogram hashes at the cluster labeling process, ensuring that each VFL has totally different visual words.

This approach to feature fusion is different to the standard fusion of features inside the descriptor. The computational overhead in the clustering process is avoided and the redundancy of visual information in different levels of spatial resolution is delegated to the IR model, namely the Vector Space Model described in section 3.2.3. The experiments in the next chapter will try to identify the aspects that affect the quality of image retrieval in the BioCNN-CBIR model.

6

BioCNN-CBIR Evaluation

6.1 Introduction

To evaluate the BioCNN-CBIR model, the combination of various quantifiable qualities for its three modules VFE, VFL and VMM compose a complex evaluation process. Metrics from the respective domains of Deep Neural Networks, Data Mining and Information Retrieval are used for individual evaluation of the CNN model, clustering model, and IR model. An additional factor of difficulty for optimizing the performance of the pipeline is the choice of data. There are two image subsets for training and evaluating a CNN classifier. A part of them can be indexed by the CBIR engine as the image collection, while some images will remain unknown to the system in order to be used for the test queries. The retrieved set of images will provide an indication for the performance of the overall system.

This chapter systematically organizes an evaluation methodology for CNN-CBIR and presents experimental results. Section 6.2 describes the standard classification metrics used to evaluate the VFE module and introduces two custom metrics, Accuracy per Error (APER) and Overfit Margin (OFM). Additionally it presents the Silhouette metric for unsupervised clustering and the standard Information Retrieval metrics for CBIR. The choices for the experimental environment are described in section 6.3. Section 6.4 analyzes the characteristics of the selected datasets and 6.5 presents the set of hyperparameters for the reproducibility of the experiments. The results for image classification are presented in section 6.6 and for image retrieval in 6.7. The chapter ends at section 6.8 where the findings from the experimental process are discussed.

6.2 Metrics for evaluating CNN-CBIR models

6.2.1 The Complexity of Evaluating CNN-CBIR

The target for the VFE module, which contains the CNN network model, is to achieve good classification accuracy and minimized classification error. Visual similarities between the different samples of an object should be captured by the learnable parameters. Generalization capabilities are desired for the network in order for its layers to be used as feature extractors for the CNN-CBIR pipeline. In the VFL module, the codebook creation uses an unsupervised clustering algorithm, not including any forward knowledge for the clusters. The cluster model can only be evaluated by measuring how strong is the belong-to relationship of the data points within each cluster. Using a hierarchy of clusters each data point has a path of assigned clusters and the evaluation of the graph cannot be done with standard clustering metrics. In the VMM module the values of retrieval metrics are calculated on image queries and have multiple dependencies on the combined performance of the previous two modules.

The following factors contribute to the overall retrieval performance: a) the classification accuracy and the selection of the layers that will provide the features, b) the quality of the cluster model that is used to assign the visual words c) the weighting scheme used for retrieval. The possible combinations create a large search space for identifying the best CNN-CBIR model. Handling these factors individually a targeted evaluation on the respective models includes the following tasks:

a) Evaluate a CNN architecture that will accurately classify object by their images. An optimal classifier might be a generic feature extractor.
b) Evaluate a suitable hierarchy of clusters that will partition the numerous feature descriptors to homogeneous clusters. This might generate a well-functioning VFL that will accurately represent the image as text.
c) Evaluate retrieval models for a given VFL that will place the most relevant visual documents in the highest ranks of the retrieved image set.

Considering that each one of them is an open problem, an attempt to reach the state-of-the-art performance for CBIR systems is outside the time constraints and the scope of this Master's thesis. The current targets are a) to create a functioning BioCNN-CBIR prototype in order to explore the complexity through evaluation and b) report the various factors that contribute to the quality of image retrieval. Additionally it seeks to organize an evaluation system with metrics and best practices simple enough for the current needs, but at the same time a baseline for future research that will target increased performance.

6.2.2 Supervised Classification Metrics

6.2.2.1 Metrics for evaluating a CNN-based image classifier

In the VFE Module a classifier is trained on known samples of classes in a binary classification problem. We denote $t_{members}$ the number of samples that belong to a class and are correctly classified (true positives), while $t_{non\_members}$ the number of samples that are not members of a class and are correctly not classified as such (true negatives). With regard to a *null hypothesis* the Type I errors are $e_{members}$ that is the count of samples predicted as members of class which they are actually not (false positives). Type II errors are $e_{non\_members}$ as the count of class members that are not correctly classified (false negatives). The *recall*, *precision* and *F-measure* metrics for each individual class c are:

$$recall_c = \frac{t_{members}}{t_{members} + e_{non\_members}} \tag{73}$$

$$precision_c = \frac{t_{members}}{t_{members} + e_{members}} \tag{74}$$

$$F_c = (1 + \beta^2) \cdot \frac{precision_c \cdot recall_c}{(\beta^2 \cdot precision_c) + recall_c} \tag{75}$$

Recall is intuitively the ability of the classifier to find all the samples that belong to class and precision is the ability of the classifier not to classify a non-member sample. The generalized version of the F-measure for a class c includes a parameter β in a harmonic mean expression. It can also be expressed using the counts of samples as:

$$F_c = \frac{(1 + \beta^2) \cdot t_{members}}{(1 + \beta^2) \cdot t_{members} + \beta^2 \cdot e_{members} + e_{non\_members}} \tag{76}$$

For the *F1-measure* or *F1-score* we set $b = 1.0$ forcing equal weights to Type I and Type II errors, thus equal significance of precision and recall. The F1 for the class is a harmonic mean of precision and recall and a more objective metric in extreme differences of precision and recall:

$$F1_c = \frac{2 \cdot precision_c \cdot recall_c}{precision_c + recall_c} \tag{77}$$

For n classes a weighted average of the F1-score is used as the primary metric to evaluate the generalization capability of a CNN classifier.

$$avg\_F1 = \frac{1}{n}\sum_{i=1}^{n} support_i \cdot F1_i = \frac{1}{n}\sum_{i=1}^{n} \frac{t_{members}}{t} \cdot F1_i \tag{78}$$

The factor $support_i$ is the fraction of the total samples t of the class that are correctly classified or the percentage of true positives over all samples. Averaging and weighting with the support provides two additional classification metrics for evaluating the learned state of a CNN classifier: The first is *average recall* which equals to classification *accuracy* and is the reverse of *classification error*, and the second is *average precision*.

$$\begin{aligned} accuracy = avg\_recall &= \frac{1}{n}\sum_{i=1}^{n} support_i \cdot recall_i \\ classification\_error &= 1 - accuracy \end{aligned} \tag{79}$$

$$avg\_precision = \frac{1}{n}\sum_{i=1}^{n} support_i \cdot precision_i \tag{80}$$

For a more objective measurement of the generalization capability, k learned states of the same CNN architecture are used, each one one trained by excluding some images from the training set. The method of *k-fold Cross Validation* will be discussed in section 6.3.2. For multiple learned states of an architecture there are three sets of metrics $a = \{accuracy_j\}$ $p = \{avg\_precision_j\}$ $f = \{avg\_F1_j\} : j = 1..k$. The *mean*, *median* and *average absolute deviation* of the sets are considered metrics of the generalization capability of the architecture.

$$\begin{aligned} mean\_acc = \bar{a} \quad mid\_acc = \tilde{a} \quad avgdev\_acc &= \langle |accuracy_j - mean\_acc| \rangle \\ mean\_pre = \bar{p} \quad mid\_pre = \tilde{b} \quad avgdev\_pre &= \langle |precision_j - mean\_pre| \rangle \\ mean\_F1 = \bar{f} \quad mid\_F1 = \tilde{f} \quad avgdev\_F1 &= \langle |F1_j - mean\_F1| \rangle \end{aligned} \tag{81}$$

6.2.2.2 Confusion matrices

The confusion matrix is a tool that illustrates misclassifications in each individual class and can be used to detect issues on the discriminative abilities of the model. For increased usability, the tabular confusion matrix is augmented with the metrics of per class recall, precision, F1-score and the total count of test samples for the class that is useful for unbalanced test sets.

Recall	Precision	F1-score		0	1	2	3	4	5	6	7	8	9	10	Samples
0.600	0.714	0.652	0	15	0	0	1	0	0	0	0	0	0	0	25
0.900	0.918	0.909	1	0	45	0	0	0	1	0	0	0	0	0	50
0.083	0.125	0.100	2	0	0	1	0	0	0	0	1	0	0	0	12
0.250	0.091	0.133	3	0	0	0	3	0	0	0	0	0	0	0	12
0.176	0.103	0.130	4	0	0	0	0	3	0	0	0	0	0	0	17
0.333	0.421	0.372	5	1	0	0	0	0	8	0	0	0	0	0	24
0.063	0.100	0.077	6	0	0	0	0	0	0	1	0	0	0	0	16
0.667	0.118	0.200	7	0	0	0	0	0	0	0	2	0	0	0	3
0.360	0.486	0.414	8	0	0	0	0	1	0	0	0	18	0	0	50
0.520	0.650	0.578	9	0	0	0	0	0	0	0	0	0	26	0	50
0.308	0.400	0.348	10	0	0	0	0	0	0	0	0	0	0	4	13

Table 14: Augmented confusion matrix for the first 10 out of 101 classes of Caltech101, sliced at the first 10 classes.

Additionally a 2D visualization of a confusion matrix is used to depict misclassifications as noise or points of intense color that do not belong in the diagonal. This is particularly useful for a large number of classes and can help detect issues at a first glance, without scrolling through the tabular view, thus providing increased usability to the researcher.

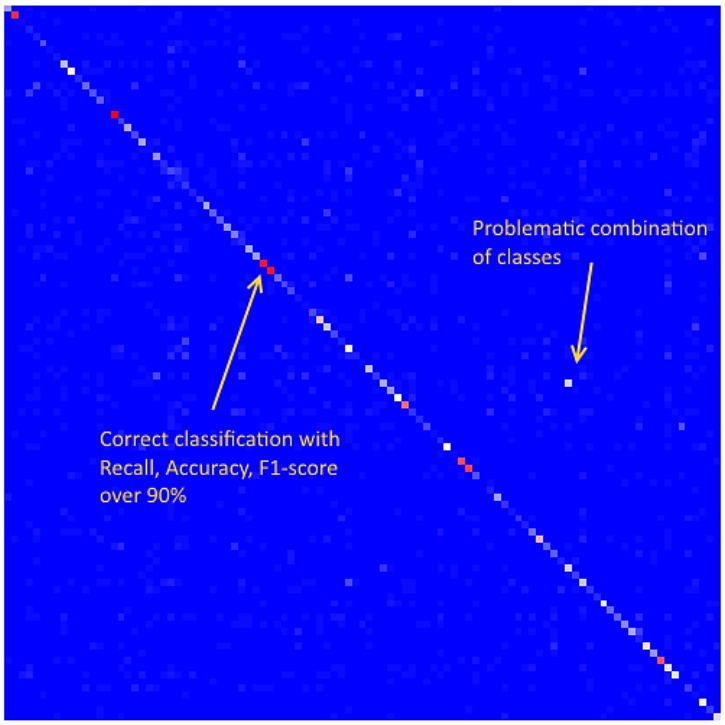

Figure 71: 2D Visualization of a confusion matrix for the combinations of 101×101 classes

6.2.2.3 Metrics for comparing a CNN classifier on ImageNet

The metrics *top-1 error* and *top-2 error* are mentioned here due to their use in evaluating classifiers competing in the ILSVRC classification task. They were not used in the final experiments because training on the ILSVRC2012 dataset was not made possible. For m images of an unknown test set that is used to evaluate a trained classifier, there are c_1

images where the top predicted probability p_1 in a Softmax output corresponds to a class $y_{PREDICTED}^{(k)}$ that is different from $y_{ACTUAL}^{(k)}$

$$top1error = \frac{c_1}{m} \ , \qquad c_1 = |\left\{k: \left(y_{ACTUAL}^{(k)} \neq y_{PREDICTED}^{(k)}(p_1)\right)\right\}| \tag{82}$$

For a count of c_5 images, which the actual class is not in the top-5 predicted classes, the top-5 error metric is:

$$top5error = \frac{c_5}{m} \ , \qquad c_5 = |\left\{k: \left(y_{ACTUAL}^{(k)} \notin \left\{ y_{PREDICTED}^{(k)}(p_i)\right\}_{i=1..5}\right)\right\}| \tag{83}$$

The generalization of the top-1 and top-5 metrics is the top-K error metric. It expresses the fraction of samples in an unknown test set where the sample's actual class does not exist in the set of the top-K classes predicted by the model.

6.2.3 Custom Metrics for Monitoring the Training Process

The multi-class categorical cross entropy (CCE) is the error function minimized by gradient descent training. It can be an additional metric for evaluation of the learned state.

$$CCE = \sum_{i=1}^{n}\sum_{k=1}^{c} y_{ACTUAL}^{(k)} \log(softmax_k(y_{PREDICTED}^{(k)})) \tag{84}$$

The CCE can act as an indicator for the convergence of the training process to optimal solutions. In various CNN training sessions for 10 and 100 classes using images normalized to [0,1] and Xavier initialization, it was observed that a typical value at the start of the session is $5 > CCE > 4$ that is decreased to $CCE < 1$ when the network converges to a solution. In overfitting conditions the CCE is minimized to $0.01 < CCE < 0.2 : accuracy \approx 1$. The following indicator can give an estimate in a given context for the convergence of the supervised training.

$$SCC = convergence = e^{-CCE/\beta} \tag{85}$$

For 10 and 100 classes the constant is set to $\beta = 1$ and the corresponding empirical markers can be used to monitor the training process:

- $SCC < 0.20$ Start of the training process. If SCC does not double in the first epochs the CNN architecture is difficult to train.
- $SCC > 0.50$ A good convergence without overfitting should have up to this point similar training and validation accuracies.

- $SCC > 0.80$ The potential for further improvement of validation accuracy is small.
- $SCC >= 0.99$ The model has over-fitted on the training data with accuracy that reaches 100%

A quality indicator for the learned state at each epoch is the custom metric *Accuracy per Error* (APER). It is calculated at the end of each training epoch on a small validation set that is not used to train the model.

$$APER = \frac{accuracy}{CCE} \Leftrightarrow$$

$$APER = \frac{\frac{1}{n}\sum_{i=1}^{n} support_i \cdot recall_i}{-\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ACTUAL}^{(k)} \log(softmax_k\left(y_{PREDICTED}^{(k)}\right))} \quad (86)$$

At the last epochs the accuracy might slightly surpass the peak values that been observed in previous epochs. These are not representative of the best learned state because the CNN model has a tendency to over-fit on the training data. The APER ratio is a more objective indicator for the best learned state that might have occurred before a seemingly best validation accuracy. It is used to compare the learned states of two different epochs inside the training session and determine the best.

When training a CNN with a small number of classes and without dataset augmentation, overfitting occurs from an early stage. For this reason a metric to compare two architectures and determine which is less prone to overfit is needed. In finance the calculation of the profit margin gives the percentage of the total revenues that is pure profit.

$$margin = \frac{Revenues - Costs}{Revenues} \quad (87)$$

Porting this into the problem of CNN overfitting, the increased training accuracy is based on a profit earned from biasing on the training samples. This *bias profit* is simply the difference $TrainingAccuracy - ValidationAccuracy$ and the corresponding *Overfit Margin* metric (OFM) can be expressed as

$$OFM = \frac{TrainingAccuracy - ValidationAccuracy}{TrainingAccuracy} \quad (88)$$

The value of the metric is the percentage of the training accuracy that is bias profit gained from overfitting, added over the actual ability to generalize that is objectively calculated by the validation accuracy. Lower values of OFM indicate lower overfitting and higher values tend to appear at the last epochs of training.

6.2.4 Metrics for the Creation of a Visual Features Language

6.2.4.1 PCA dimensionality reduction metric

For measuring the quality of a dimensionally reduced features descriptor set, a typical measure is the *Mean Squared Error* (MSE). For compressing n original vectors x_i into $y_i = W \cdot x_i$ of smaller dimensionality, the reconstruction is made by using the transpose matrix $z_i = W^T \cdot y_i$. The error that is introduced from reconstruction is calculated as:

$$mse = \frac{1}{n}\sum_{i=1}^{n}(x_i - z_i)^2 \tag{89}$$

6.2.4.2 Unsupervised clustering metric

The creation of the VFL vocabulary does include any forward knowledge for the clusters. A metric for unsupervised clustering is the *Silhouette Coefficient* [251] that shows either that the feature descriptors suit well in their clusters, or that they reside in the border regions between clusters. It gives an indication about the cohesion inside a cluster by measuring the Euclidean or Manhattan distance of all belonging data points to the cluster separation [252].

$$s(y_i) = \frac{b(y_i) - a(y_i)}{\max\big(a(y_i),\ b(y_i)\big)} \quad , \qquad s(y_i) \in [-1, 1] \tag{90}$$

For a cluster C_k the average dissimilarity $a(y_i)$ of a data point $y_i \in C_a$ compared to all other data points $y_j \in C_a : j \neq i$ is subtracted from the minimum dissimilarity $b(y_i)$ of the data point to the members $y_k \in C_n$ of other clusters $y_i \notin C_n : n \neq a$. The result is normalized by the maximum of the dissimilarities, thus the value of Silhouette is in the range of $[-1,1]$. There are three indications for the cohesion of the feature descriptors inside a cluster according to the value of Silhouette:

- $s(y_i) \cong 0$: The descriptor can be assigned to another cluster without improvement or deterioration of both clusters cohesion.
- $s(y_i) < 0$: The descriptor lowers the cohesion of its assigned cluster.
- $s(y_i) > 0$: The descriptor suits well inside its assigned cluster

The Silhouette index for the whole cluster C_k is the average of its n belonging descriptors Silhouette Coefficients:

$$silhouette(C_a) = \frac{1}{n} \cdot \sum_{i=1}^{n} s(y_i) \quad , \qquad y_i \in C_a \tag{91}$$

6.2.5 Information Retrieval Metrics

The standard metrics for text-retrieval are used to evaluate CBIR. For a single image query that retrieves q images from a collection of c images the precision, recall and F1-score are defined for IR on an unordered set of retrieved images. These will be mentioned from this point on as *PrecisionR*, RecallR (R for retrieval), to disambiguate with the precision and recall metrics of classification.

$$precisionR_q = \frac{t}{q} \tag{92}$$

A number t of retrieved images are relevant to the category of the image used in the query. When using the same class labels for both classification and retrieval, the relevant images are these whose label matches the class label of the image-query.

$$recallIR_q = \frac{t}{r} \tag{93}$$

There is a total number r of relevant images retrieved for the given query in the collection where $r \leq q$. The recall metric is simply the fraction of the relevant items retrieved, out of all relevant items in the collection. Increasing the number q of the returned images the recallR will gradually grow closer to 1 until $t = r$ when $q = c$, that is when the query retrieves the whole collection.

Taking into account the order by which the IR method has retrieved the images, the rank of the images, an interpolated version of the precisionR is used for the top k retrieved documents [253].

$$iprecisionR = \max_{r' \geq r} \big(precisionR\,(r') \big) : r, r' \in [0\,,1] \tag{94}$$

The level of the recall value is r that can be interpolated with a constant step, for example $r_1 = 0.0, r_2 = 0.1, \ldots r_{11} = 1.0$. The expression will return the highest precision for any recall level $r' > r$. A graph curve of the $iprecisionR$ metric represents the function $f: RRecall \rightarrow RPrecision$ and is called a *Precision-Recall (P-R) graph.*

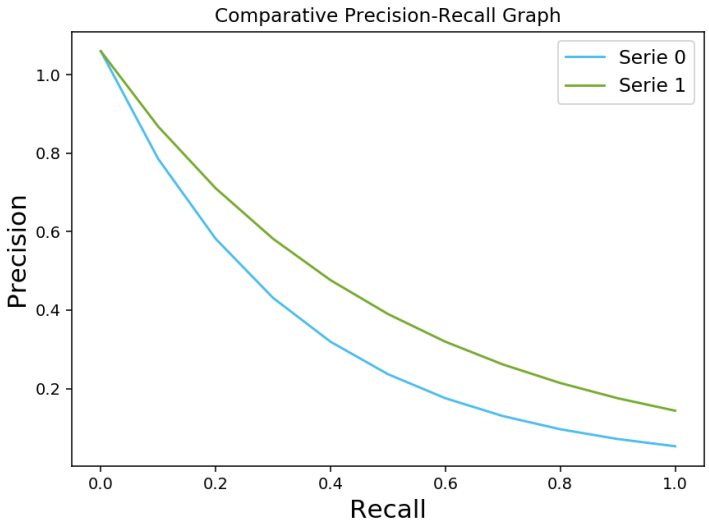

Figure 72: Precision-Recall comparative graph

This graph is important for the comparison of two retrieval models. Different properties of each retrieval method can be identified by the area enclosed under the P-R curve. As the count q of retrieved documents increases the recall level tend to reach 1.0 and when $q = c$ all the documents in the collection have been retrieved.

Additionally the precision metric for subset of retrieved documents up to the rank k, defines a set of metrics for *PrecisionR at* k or $p@k$. Depending on the desired q and the collection size there could be P@5, P@10, P@50, P@100, P@1000 and other metrics.A more stable metric for ranked retrieval is the *mean average precision* (mAP) defined for the set of relevant documents $D = \{d_1, d_2, .. d_{m_j}\}$ and the set of ranked documents $R_{j,k}$ that has been retrieved.

$$mAP(D) = \frac{1}{|Q|} \cdot \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} RPrecision\left(R_{j,k}\right) \tag{95}$$

The value of mAP will be 0 if no relevant documents are retrieved [253].

6.3 Evaluation System for Experiments

6.3.1 Training, Validation and Test Sets

For the BioCNN-CBIR evaluation the sets of images used for training and testing the CNN are reused as part of the image collection. Standard approach in ML is to have a *ground truth set* (GT) for training and an *unknown test set* (UT) for testing the model's generalization capability. For a dataset that this is already split into GT and UT parts, this will be preserved to provide a common ground of comparison with published papers. The GT set is further partitioned into two sub-sets the *training set* (TS) and the *validation set* (VS) used to monitor the training process. The VS can help detect overfitting during training and implement an early stopping mechanism. The usual choice for partitioning a given GT, will be 90% of its samples as TS and the remaining 10% for VS, in order to facilitate 10 different combinations of TS / VS sets.

In the case of an image dataset without a predefined GT / UT partitioning, random samples for each set will be chosen. The number of samples for the GT or the percentage of the dataset used as GT, will conform to the setup in published papers as a minimal ground of comparison. The counts of samples in each one of the training and validation sub-sets are calculated from the count of samples in the ground truth set and the percentage p_{VS} of samples chosen for validation:

$$n_{TS} = n * (1 - \frac{p_{VS}}{100}) \qquad n_{VS} = n_{GT} \cdot \frac{p_{VS}}{100} \tag{96}$$

6.3.2 10-fold Cross Validation

For safest conclusions on the generalization capabilities of a BioCNN architecture, the method of Cross-Validation (CV) [254] is used to evaluate multiple learned states. There are different types of CV that have advantages and disadvantages [254]. The *Hold-Out* method partitions all the available data to GT and VS sets, trains the model with GT and holds the VS set outside the training process for testing. Repeating this and randomly choosing each time a different part of the data with the same p_{VS} percentage of samples as VS, is called *Repeated Hold-Out Validation*. Using this method the results are highly depended on the partitioning. There is the possibility throughout the repetitions, that all the samples would eventually be handled as known and used to train the models. This approach does not ensure a completely unknown set of images that will objectively evaluate a trained model.

An improvement to the previous method is the *k-Fold Hold-Out Cross-Validation* (CVF-k) by which that dataset is split to GT and UT that is kept to test the model. The GT set

is split into k fragments. The model is trained k times using $k-1$ fragments for the TS set and holding out as the VS a different fragment in each training process. In the final k resulting models, all the GT samples have been used to train the model but UT remains unknown. A selection of $k = 10$, mentioned from now on as the *CVF-10 method*, provides accurate performance estimation. It makes predictions for the validation set using a model that is trained with the 90% of the ground truth data [254] . Each one of the 10 resulting models will be tested using a common unknown set of samples UT, which has been kept aside and was not presented during training.

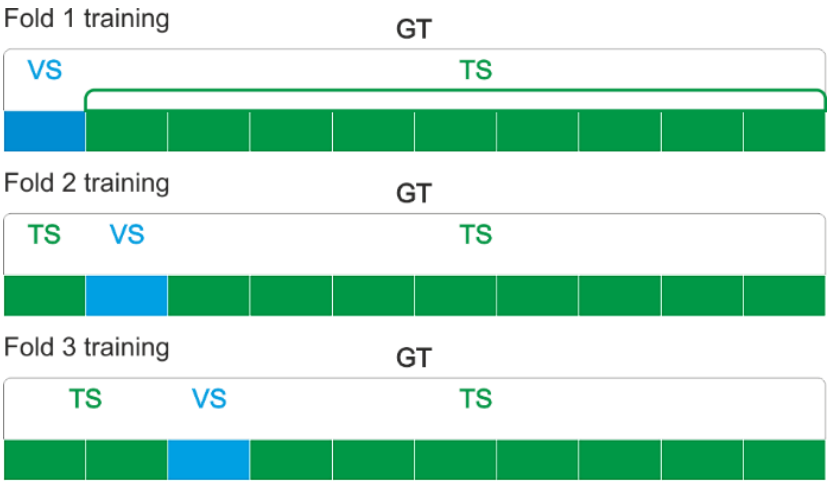

Figure 73: 10-fold Hold-Out Cross Validation with 3 different training sessions for the 3 first folds. The ground truth set (GT) is partitioned into different test set (TS) and validation set (VS) in each training fold.

Another type of CV is the *Repeated k-Fold Hold-Out Cross-Validation* (RCVF-k) that holds out a different fragment from k folds is in each training epoch and uses the remaining $k-1$ to train the model. The RCVF-10 uses 100% of the GT inside a training session. The GT set is shuffled after the end of each epoch and split again into 90% TS and 10% VS with potentially be n different TS sets used in n epochs of training.

6.3.3 Gradient Descent Learning

6.3.3.1 Throttled Gradient Descent (TGD)

The error minimization method used to train the BioCNN is simple gradient descent with momentum. Various approaches in training CNNs, consider to reduce the learning rate (LR) as the training advances, typically with exponential decay. A variation introduced here is that the variable LR can be either reduced or increased depending on the outcome of the validation at the end of each training epoch. A useful abstraction is made by thinking LR as

the throttle on a wheeled vehicle that must be adjusted to control its ascent or descent on slopes. Hence the name *Throttled Gradient Descent* (TGD) that is given to this training method.

When in an epoch i the validation error ve_i decreases insignificantly compared with the previous epoch a step on the accelerator increases the LR to the next throttle level. For this a minimum threshold is used to compare the difference of the validation error between two consecutive epochs $|\Delta ve| = |ve_i - ve_{i-1}| < \theta_{SMALL} : \Delta ve < 0$. When an increase in the validation error occurs $\Delta ve \geq 0$, a step on the break decreases the LR to the previous throttle level. For a total number of throttle levels l with fixed analogy $a > 1$ between two levels, the value of LR is calculated as:

$$LR_i = LR_{i-1} \cdot a = LR_{MIN} \cdot a^{i-1} \ : \ i \in [1, l] \tag{97}$$

Training starts at level $i = 1$ using the minimum learning rate LR_{MIN} and accelerates up to the maximum $LR_{MAX} = LR_{MIN}\ a^{l-1}$. It has been experimentally observed that in the first epochs the LR throttle increases and reaches max, allowing large drops of training error. Then it decreases gradually to reach minimum again due to increases in validation error that are caused by overfitting.

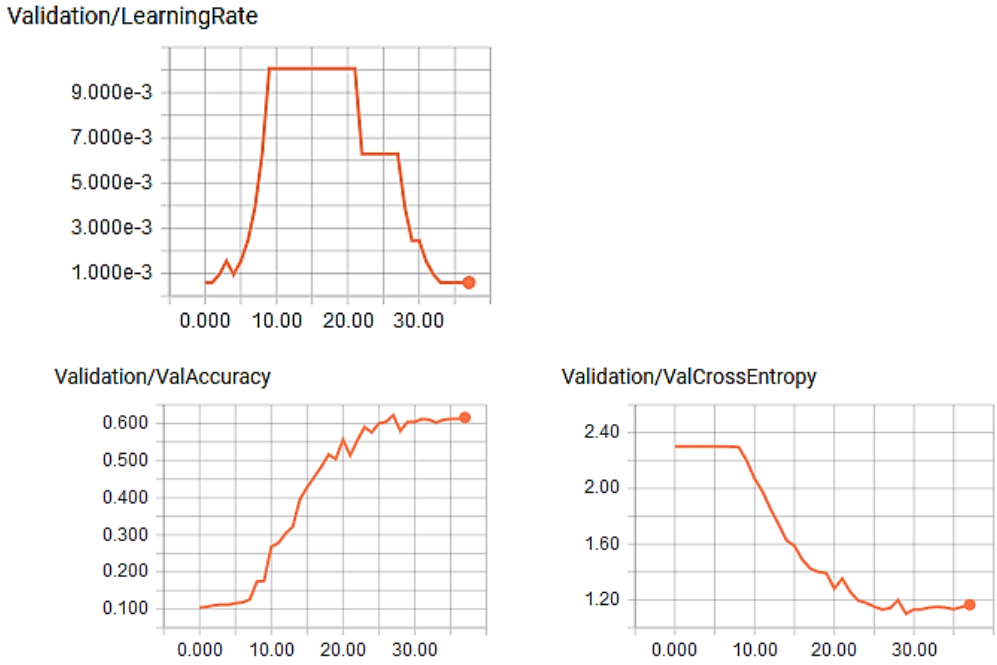

Figure 74: Throttled Gradient Descent (TGD) for 37 epochs of training. Graphs are generated by Google Tensorboard for the learning rate (top left), validation accuracy (bottom left) and validation categorical cross entropy (bottom right). Plateaus and peaks of the validation cross entropy error can increase and decrease the learning rate.

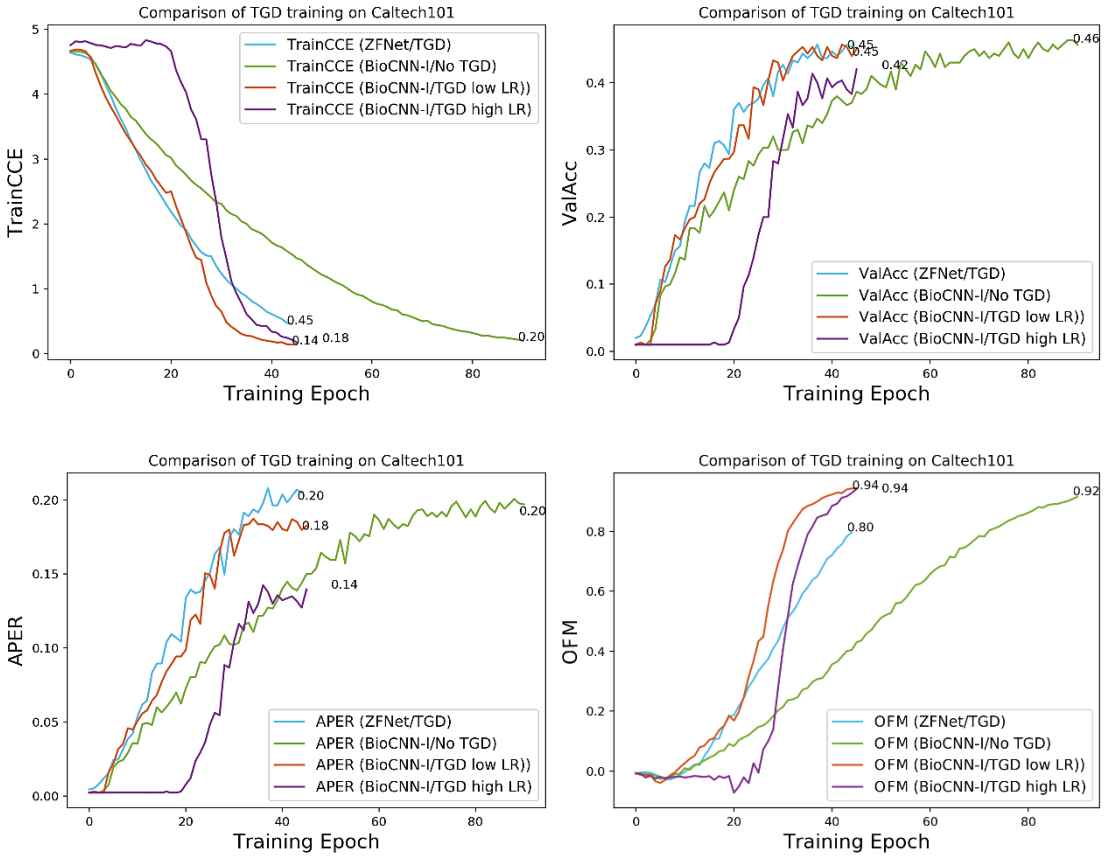

Figure 75: Gradient descent training on Caltech101 with ZFNet and the same BioCNN architecture. There are apparent differences in the models' convergence and the amount of overfitting in four different scenarios of fixed learning rate (green), TGD with nominal settings (blue), low (orange) and high learning rates (purple)

The TGD training method was initially tested for proof-of-concept on the MNIST dataset [17] using the AlexNet architecture for 10 handwritten digits classes, resulting in a classification error of 1.36%. A comparison between this method and other gradient descent methods or other variable learning rate schemes can be conducted in a future research, aiming to testify whether TGD helps CNN models to converge to better solutions and in a smaller amount of epochs.

Additional throttle control conditions can be modeled like checking the difference of the validation error Δva. Furthermore the starting point can be at a different throttle level for example LR_{mid} so that a smaller LR_{MIN} will be used only during the last epochs. This can be thought of as beneficial, because more detailed corrections of the model weights will occur in the last epochs. The TGD logic can be extent to other gradient descent methods that were mentioned in section 2.4.5.

6.3.3.2 Overfitting monitoring and early stopping

For preserving generalization when the model starts to overfit, a custom early stopping mechanism is implemented. For determining the proper stop epoch, a key performance indicator (KPI) for the quality of the learned state is needed. The *Learning Quality* indicator LQ_i of an epoch is compared with its maximum value LQ_{max} of all previous epochs to determine whether the current epoch is the best. The following formula calculates the LQ indicator from the average training accuracy ta , the validation accuracy va and the validation error ve:

$$LQ_i(ve, va) = \frac{va_i}{ve_i} - p_i(ta_i, va_i)$$
$$p_i(ta_i, va_i) = \begin{cases} \beta \cdot (1 - \tanh\left(\frac{ta_i - va_i}{ta_i} - \theta\right)) & , p_i(ta_i, va) \geq 0 \\ 0 & , p_i(ta_i, va) < 0 \end{cases} \quad (98)$$

The ratio $va\ /\ ve$ is the APER ratio introduced in equation 86 and it is penalized by the function p_i which contains the OFM metric of equation 88. In equation 98 that weight $b \in [0,1]$ controls the significance of overfitting and $\theta > 0$ is a bias for the margin function result.

The best value of LQ_i is kept in LQ_{max} and the corresponding best epoch number in $b = i$. When the quality of the current epoch is $LQ_i < LQ_{max}$ then the distance between the two epochs is $ed = i - b$. If this has reached the maximum allowed distance from the best epoch, defined as a training hyperparameter $ed \geq \mathrm{ed_{MAX}} : \mathrm{ed_{MAX}} > 1$ the training session is terminated. A value of $ed_{MAX} = 10$ is typically used. Additional stopping criteria on abnormal training events are also implement. For an epoch before the maximum limit of epochs i_{MAX} and after a minimum number of epochs i_{MIN} the following conditions will stop the training:

- A tiny increase in validation error that is considered as no change $\Delta ve \approx 0$ $\Delta ve = |ve_i - ve_{i-1}| < \theta_{TINY} : ve_i < ve_{i-1},\ \theta_{TINY} > 0$
- A large decrease in validation accuracy $\Delta va = va_i - va_{i-1} \leq \theta_{DROP} : \theta_{DROP} < 0$
- A maximum count of failures that decrease performance of the model, for example 10 increases of the Δve without an increase.

6.3.4 Minibatches, Sample Shuffling and Subfolds

The training is done with minibatches of data due to the large amount of data. A minibatch size of $n_{MINIBATCH}$ is used so that an integer $\frac{n_{TS}}{n_{MINIBATCH}}$ number of steps will run in each training epoch. For m epochs needed to train the model there is a total of $s = \frac{n_{TS}}{n_{MINIBATCH}} \cdot m$ mini-batch training steps and a granularity of $n \cdot m$ data points for the series of training error and training accuracy metrics.

Mini-batch training can have a bias on the data, affecting the occurring gradients and the corresponding weight-updates. The typical solution is to shuffle the samples before each epoch, so that a different sequence of samples is used. The large count of images forbids shuffling of the whole dataset. By shuffling only the sequence of minibatches, each one containing a fixed set of images, good mixing properties are not ensured. For this reason a more efficient shuffling scheme is used together with the CVF-10 method:

- The training set TS has 9 out of 10 folds of the ground truth set GS and each fold is divided into k subfolds, totaling $9 \cdot k$ subfolds. The number of samples in a subfold is $n_{SUBFOLD} = \frac{n_{TS}}{9 \cdot k}$
- A shuffling of the subfolds is made before each epoch, resulting in $(9 \cdot k)!$ possible permutations to randomly select from.
- Images of subfolds are fed into a FIFO queue with random order so that $n_{MINIBATCH}$ samples are ready for a training step. This decreases even more the possibility for the same sequence of samples to occur inside a minibatch.
- Having an additional ratio $\frac{n_{MINIBATCH}}{n_{SUBFOLD}} = a : \ a \in \mathbb{N}, \ a > 1$ ensures that further mixing is done by composing a minibatch out of randomly enqueued images of a randomly selected subfold.

6.3.5 Visual Features Language Setup

The same choice of rooting factors $\{RF_i\}$ for the RkMeans algorithm is used for all VFL creation experiments. In each experiment the learned state of a BioCNN is kept the same and different convolutional modules are chosen to extract the features. Dimensionality reduction using PCA is performed before the clustering process. Feature fusion is implemented using the activations from the last 3 convolutional modules to generate 3 different VFLs. The global features are extracted by the last convolutional module l as the *GVFL* language, the previous $l-1$ as the *MVFL* for medium-resolution features and $l-2$ convolutional model extracts the more local features as the *LVFL* language. A single retrieval

experiment on an image collection uses 6 different VFL vocabularies to compose a VBOOK textual image representation, each one containing 3 different VDOCs in GVFL, MVFL and LVFL. The 3-leveled hierarchy of clusters will create 3-syllable visual words. For the textual representation there are 3 syllabogram hashes h_1, h_2, h_3 for each level in each VFL. There is a total of 9 hashes ensuring different visual words for different VFLs. Generalizing this experimental setup when using n-syllable visual words there will be n rooting factors for clustering and $l \cdot n$ syllabogram hashes for the l convolutional modules. In the hash the syllable can be fabricated using a combination of v vowels, c consonants and s semivowels. Possible setups for combinations are $n = c \cdot v,\ n = 2c \cdot v,\ n = c \cdot s \cdot v$. For example $v = 5$ vowels $c = 10$ consonants and $s = 3$ semivowels will create unique syllabograms for 150 clusters.

6.3.6 Image Retrieval Environment

6.3.6.1 Known Image Collection and Indexing

The common practice in CNN-based CBIR is to use a network that is pre-trained on the large scale image dataset ILSVRC2012 [9], which contains millions of images. Then this is fine-tuned on the image collection that will be used for retrieval. In BioCNN-CBIR the network is trained on a subset of the image collection used for retrieval. This requires images to have labels for supervised classification. A single label will be used as the class and multiple labels should be converted into classes for each possible combination. Training the neural network on the image collection attempts to recall the images through a learned CNN state that is more related to the visual semantics of the collection. This is substantially more difficult to achieve competitive retrieval performance and attempting this is a starting point for similar research.

For experiments, the images collection or collection set (CS) will be indexed by the Terrier IR search engine that is presented in section 7.2.4. Samples from both partitions GT and UT of an image dataset can be used as the image collection $GT \cup \text{UT} \subseteq \text{CS}$, or reversing this a CS can be split into ground truth and unknown image sets $GT \subseteq \text{CS},\ UT \subseteq \text{CS},\ GT \cup \text{UT} = \text{CS}$. The VFLs will be created using all the images in the CS that are recalled through the trained BioCNN. The experimental setup includes leaving out some images of the UT set from the collection $CS \cap UT = \emptyset$ to use them as test queries.

To evaluate the CBIR performance a set of image queries (QS) will be run against the indexed collection. The QS can be a subset of the images that are not used for training $QS \subset DS:\ QS = \{x : x \notin \text{GT}\}$ or a set of completely unknown external images that belong

to the classes $QS \cap DS = \emptyset$. Constructing a QS using the human expert's perception of the topic, can have deliberate visual differences from the ones found on the dataset. The datasets may have been collected with automated tools and can include an amount of dataset bias [255] [256]. Despite having visual differences, handpicked image samples are ensured to be absolutely relevant to their class. They are used to provide objective hints of the generalization capabilities of the overall system. Both CS and QS sets will be translated into visual documents, by recalling them through the trained CNN and assigning the visual words of the respective VFL.

The relevance relations between the visual documents in QS and the visual documents in CS are recorded in the *QRELS* key-value dictionary. For each key combination $\{VDOC_q, VDOC_c\}: q \in QS, c \in CS$ there is a flag value 1 to indicate that is relevant and 0 that is irrelevant. These lists are automatically generated using the classes of the image collection, thus a class is the topic for the visual document in this CBIR approach.

6.3.6.2 Image Search Queries

A VDOC image representation has $m \times n$ visual words organized in columns and rows, for the respective $[\, m \times n \mid h \,]$ activation map of a convolutional module. An image search query can use all n_i^2 visual words inside the corresponding $VDOC_i$ in a single field search that will be performed in the respective VFL language $i \in \{GVFL, MVFL, LVFL\}$. Feature fusion is implemented with a fielded search. A query can also split the image into equal squared-shaped regions of visual words and use them in a combined search into the same destination field. Another querying scheme is to keep the center part of each VDOC using circular and squared masks to search only with the central regions' visual words. There are three standard choices for the queries in experiments:

- Using the whole spatial extend of an image by including all the visual words of the VDOC in the query
- Using the visual words that represent the center square of the image
- Using a rasterized circular mask to keep the visual words in the center of the VDOC.

The combinations can be numerous and new querying schemes can be constructed that will take account the location of the visual words in the documents. Future querying schemes can be designed to approximate object localization and detection tasks.

6.4 Image Datasets

6.4.1 CIFAR10 and CIFAR100 dataset

The CIFAR10 dataset is a subset of the tiny images dataset of MIT and NYU [68], that has been labeled using human experts during the research conducted by Alex Krizhevsky for its Master's thesis. The tiny images dataset was assembled using the Google, Flickr and Altavista image search engines. The CIFAR10 has 10 classes with 60000 RGB color images of 32×32 resolution. The small image size is suitable for benchmarking classification on a CNN architecture in a reasonable amount of time, allowing the use of thousands of images per class during training. A predefined partitioning into GT and UT exists as a downloadable file and it is used here without any modification. There are 5000 samples per class as ground truth and 1000 samples per class for testing, totaling 6000 tiny image samples per class. The existing partitioning of the GT has five batches that are merged into one, in order to shuffle the dataset and then split it to 10 equal folds following the CVF-10 validation method.

There is also the CIFAR-100 dataset which contains samples from 100 classes. There are 6000 color images per class with 32×32 dimensions resulting in the same total number of samples with CIFAR-10. The predefined split for each class is 500 images for ground truth and 100 images for testing.

6.4.2 Caltech 101 dataset

The Caltech101 dataset has 101 categories of objects and an additional background clutter category [257], that was first used for Bayesian and One-Shot Learning by Fei-Fei Li. There are unequal counts of image samples per class, which have been assembled from queries performed in the Google Image Search engine. A characteristic of the dataset that simplifies classification is that objects are centered in images, in most cases not occluded and without heavy background cluster. The image shapes are not squared, with the largest dimension around 300 pixels. There are both grayscale and colored images in the dataset and there are some images with noise from JPEG compression artifacts. There are also some images that have been rotated from their original rectangular orientation, with their new extends filled with black color. The most difficult aspect is that classes can have as low as 31 samples that imposes training on a very small number of samples per class.

Unlike CIFAR it has no default partitioning into GT and UT sets and these are created by randomly selecting samples of each class. For a total count of samples S in a class a count N is randomly chosen for the GT set and the remaining $S - N \leq M$ are kept for the UT set. The limit M is the maximum number of samples per class for the unknown test set.

The typical approach in papers that use Caltech101 is $N = 30$ and $1 \leq M \leq 50$ remaining images. An unbalanced UT set is assembled due to the distribution of samples in the dataset.

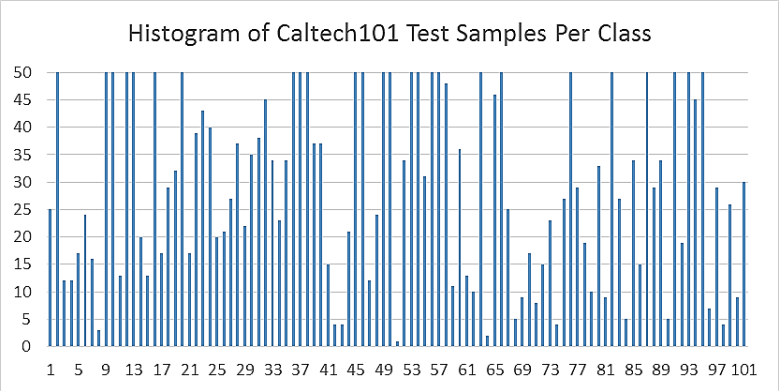

Figure 76: The unbalanced test set that occurs using the standard $1 \leq M \leq 50$ setup for Caltech 101.

There is an inherent extra difficulty in a random division of Caltech101, because some classes with low M can have an impact on the classification metrics. A misclassification of only one sample in classes with the least test samples can decrease the class accuracy to 0%, 50%, 66.66%, 75%, having an impact on the overall accuracy for all classes. For comparison all CNN architectures are all trained on the same random partitioning of Caltech101 using 10 folds of the CVF-10 validation method.

Caltech 101 class	Number of remaining test samples
inline skate	1
metronome	2
binocular	3
Garfield, gerenuk, platypus, wild cat	4

Table 15. Classes with very few test samples occurring from the $N = 30,\ 1 \leq M \leq 50$ setup

6.4.3 The Caltech101-2017 randomly chosen image collection

All classification experiment will use the same fragment of samples from the Caltech101 dataset. Given a random seed rnd a process will randomly choose 30 samples from each class for the GT set, and the remaining 1 up to 50 samples of the class for the UT set. The integer value of the random seed will represent the suffix in the name of the derivative dataset, thus Caltech101-2017 is named after the value $rnd = 2017$. For each class the filenames of Caltech101 images have the form $image\_\{4\ digit\ number\}.jpg$ and they are sorted in ascending order before the random pick. All images have been adjusted in the square 227×227 resolution of the AlexNet reference model in the Caffe framework [212], an input dimension that reproduces the convolutional kernel spatial sizes for both AlexNet and ZFNet.

This derivative of Caltech101 image dataset has been created using *stratification* [254] of the 101 class samples into the 10 training folds. In each training fold a maximum of 3 out of 30 samples will be used for each one of the classes. Because of the inconvenient total number of samples $101 * 3 * 10 = 3030$ a trimming of 30 samples is done to have a total of 3000 samples. One sample from the 30 top classes is removed, which are ordered according to the count of their representing samples in Caltech101. Moreover it is ensured that only 3 classes will be trimmed in each fold. This GT set is more difficult to train than the standard setup because of the 29 samples per class that is further reduced by partitioning 90% of the total 3000 samples into a training set.

For image retrieval experiments the 101 classes will represent 101 topics. A relevant document is retrieved when its class matches the class of the image-query. The number of the image class from 1 to 101 is also indexed by the IR search engine, for facilitating *unit testing* [258] of the retrieval operation in the VMM module.

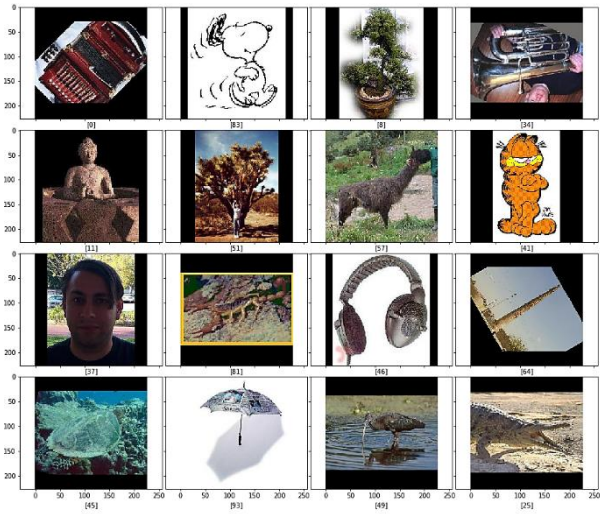

Figure 77: Caltech101-2017 image collection, the first 16 images of fold 2 / 10

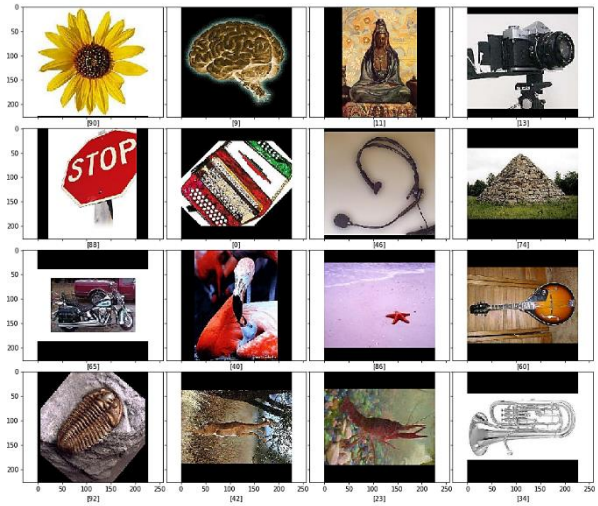

Figure 78: Caltech101-2017 image collection, the first 16 images of fold 4 / 10.

6.4.4 The HIQ-1 Handpicked Images Set

One sample was handpicked from the results of the Google Image Search engine for each class name of the Caltech101 dataset. The choice introduces deliberate difficulties but at the same time ensures that samples are very representative of their class. For example the image for "cup" depicts a magenta cup with a white handle on a white background and for "yin-yang" the symbol is emanating blue light rays. The handpicked image queries (HIQ-1) will run on the indexed samples of the Caltech101 collection in order to test the overall capability of retrieving relative images. Additionally they could provide objectiveness by not involving a potential dataset bias.

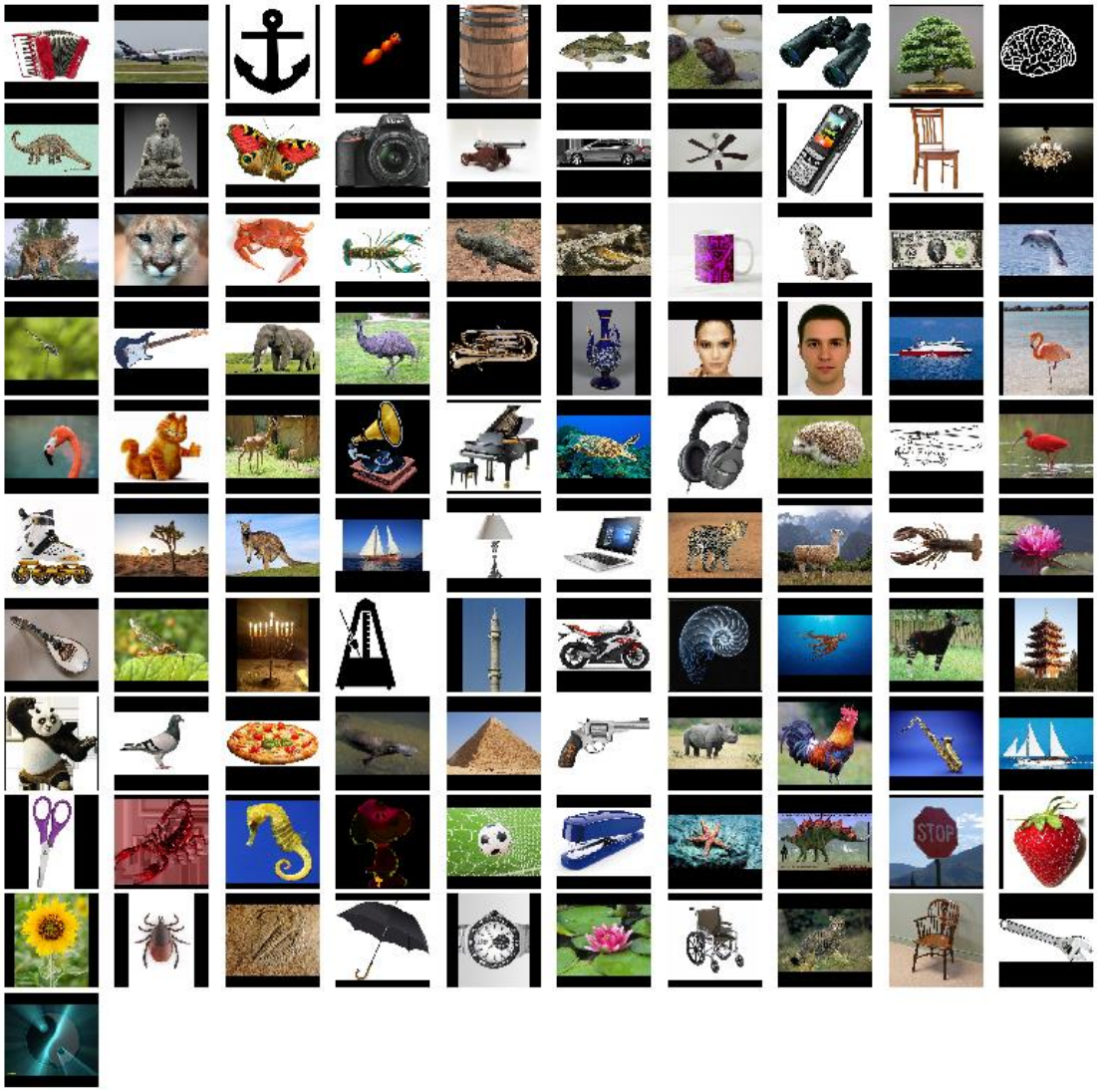

Figure 79: The HIQ-1 handpicked images samples for the 101 classes of Caltech101.

6.5 Experiments Setup

6.5.1 Classification Training Process

A "vanilla" training process is used for the classifications experiments on both CIFAR10 and Caltech101-2017 by which:

a) No preprocessing is done on the images.
b) No augmentation of the image dataset was performed during training.
c) No batch normalization is used inside the convolutional layers.

This scheme is used to compare the CNN architectures and experiment with various hyperparameters that define them, not pursuing to reach maximum accuracy or beat the state-of-the-art on the image datasets. Various hybrid BioCNN designs are compared with the AlexNet / ZFNet CNNs using the same learning process.

The Throttled Gradient Descent algorithm is used to train all the networks and the training session is terminated by the early stopping mechanism presented in section 6.3.3. For a given dataset a common mini-batch size is used for all architectures that is 500 images for CIFAR10 and 30 images for Caltech101. The weights in the convolutional kernels for all architecture are initialized using the Xavier initialization that was described in sub-section 2.4.5.7, using $a = 1,\ b = 1$ for the hyperparameters of the initialization boundaries expressed in equation 27.

6.5.2 Nominal TGD Gradient Training Hyperparameters

For TGD training the nominal values used for all networks are:

- Momentum between 0.8 and 0.95, typically 0.85 or 0.92.
- Three different throttle scales were used as defined by LR_{MIN} and LR_{MAX} :

1. $LR_{MIN} = 0.0001$ and 14 throttle scales with $a = 1.6$

$$LR \in \{ \begin{matrix} 0.00010\ ,0.00016\ ,0.00026\ ,0.00041\ ,0.00066\ ,0.00105\ ,0.00168 \\ ,0.00268\ ,0.00429\ ,0.00687\ ,0.01100\ ,0.01759\ ,0.02815\ ,0.04504 \end{matrix} \}$$

2. $LR_{MIN} = 0.0006$ and 11 throttle scales with $a = 1.6$

$$LR \in \{ \begin{matrix} 0.00060\ ,0.00096\ ,0.00154\ ,0.00246\ ,0.00393\ ,0.00629 \\ ,0.01007\ ,0.01611\ ,0.02577\ ,0.04123\ ,0.06597 \end{matrix} \}$$

1. $LR_{MIN} = 0.001$ and 10 throttle scales with $a = 1.6$

$$LR \in \{ \begin{matrix} 0.0010\ ,0.0016\ ,0.0026\ ,0.0041\ ,0.0066 \\ ,0.0105\ ,0.0168\ ,0.0268\ ,0.0429\ ,0.0687 \end{matrix} \}$$

- No early stopping will occur before $i_{min} = 30$ epochs of training and the process will stop if reaches $i_{max} = 110$ epochs.
- The training will stop when the distance to the best epoch, using the LQ performance indicator, reaches a maximum of:
    - $\mathrm{ed}_{\mathrm{MAX}} = 8$ for CIFAR10
    - $\mathrm{ed}_{\mathrm{MAX}} = 10$ for Calech101-2017

6.5.3 AlexNet / ZFNet architecture reproduction

For comparing the BioCNN with known architectures, a custom reproduction of the AlexNet architecture described in [3] and its improved sibling ZFNet described in [80] were both implemented in the Google Tensorflow framework (see chapter 7).

The narrower ZFNet geometry has a maximum $h = 384$, thus the hyperdepths of the convolutional kernels between AlexNet and ZFNet are the same. Input image size is 227×227 for Caltech101 and 32×32 for CIFAR10. The total parameters of both networks are calculated to 21.6 million for 10 target classes of CIFAR10 and 72.3 million for 101 target classes of Caltech 101. In the ZFNet implementation a local response normalization (LRN) is used before each convolutional layer, so that the only differences between the reproduced architectures would those introduced by ZFNet in CONV1: $[\,7 \times 7 \sim 2 \mid 3 \rightarrow 96\,]$ and CONV2: $[\,5 \times 5 \sim 2 \mid 96 \rightarrow 256]$. The AlexNet and ZFNet are trained from scratch on the same dataset in order to make comparisons with BioCNN architectures.

6.5.4 BioCNN Architectures for CIFAR10

Multiple BioCNN hyperparameter configurations have been trained on CIFAR10 for selecting candidate architectures. There are two distinct types of BioCNN architectures:

- Type-I has a classifier with two fully connected layers that use 50% dropout on the $f(u) = \tanh(u)$ activations and a Softmax layer.
- Type-II has a global average pooling layer without dropout that feeds directly to the Softmax output layer.

In each type a different count of stacked convolutional modules in the AHVC subsystem are performing spatial reduction with max pooling. The used hyperparameters for creating the pre-defined RFMs, the connectivity between modules, their hyperdepths and the depth of the CNN compose the *genotype* for the BioCNN. Different hyperparameter configurations are labeled as Deep Network Architecture (DNA) with a trailing number. The architectures are described using the 4D convolution notation that was introduced in section 2.4.1.

BioCNN Type-II Deep Network Architecture 5 (BioCNN-II DNA5) for CIFAR10

Subsystem	Module		Convolutional Layer				Pooling / Normalization /
Name	Name	Input	ID	Name	Subtype	Kernel	Regularization Layer
ARNN	OPL	[32 x 32 | 3]	1	Cones Rods		[1 x 1 | 3 → 2]	
			2	Horizontal Cells		[1 x 1 | 3 → 8]	
			3	Bipolar Cells		[1 x 1 | 8 → 14]	
	IPL	[32 x 32 | 14]	4	Amacrine Cells		[1 x 1 | 14 → 6]	
			5i	Ganglion Cells	p-GC	[3 x 3 ~ 2 | 6 → 12]	
			5ii		m-GC	[5 x 3 ~ 2 | 6 → 12]	
			5iii		k-DSGC	[11 x 11 ~ 2 | 6 → 84]	
				Optic Nerve (ONRV)		{ p-GC, m-GC, k-DSGC }	U → LRN
APVC	V1	[16 x 16 | 108]	6i	V1 Simple Cells		[3 x 3 ~ 1 | 108 → 6]	
			6ii			[5 x 5 ~ 1 | 108 → 6]	
			6iii			[7 x 7 ~ 1 | 108 → 6]	
			6iv			[11 x 11 ~ 1 | 108 → 6]	
			6v	V1 Complex Cells		[5 x 5 ~ 1 | 108 → 6]	
			6vi			[7 x 7 ~ 1 | 108 → 6]	
			6vii	V1 Complex Grayscale M. Cells		[7 x 7 ~ 1 | 6 → 24]	
	LGN	[16 x 16 | 108]	6viii			[3 x 3 ~ 1 | 108 → 6]	
			6ix			[5 x 5 ~ 1 | 108 → 6]	
			6x	LGN-V1 Simple Cells		[7 x 7 ~ 1 | 108 → 6]	
			6xi			[5 x 5 ~ 1 | 108 → 6]	
			6xii			[7 x 7 ~ 1 | 108 → 6]	
	ONRV		6xiii	ONRV Shortcut Connection		[108 →108]	U → LRN
AHVC	V2	[16 x 16 | 198]	7	V2 NiN		[1 x 1 ~ 1 | 198 → 144]	
			8	V2 Reduction		[3 x 3 ~ 1 | 144 → 233]	maxpool [3 x 3 ~ 2] → LRN
	V3	[8 x 8 | 233]	9	V3 Reduction		[3 x 3 ~ 1 | 233 → 377]	maxpool [3 x 3 ~ 2] → LRN
	V4	[4 x 4 | 377]	10	V4		[3 x 3 ~ 1 | 377 → 233]	
	V5		11	V5 Reduction		[3 x 3 ~ 1 | 233 → 144]	maxpool [3 x 3 ~ 2] → LRN
CLS	GLAVP	[2 x 2 | 144]				[→ 144]	
	SMAX					[144 → 10]	

Table 16: The BioCNN Type-II Deep Network Architecture 5 (BioCNN-II DNA5) for CIFAR10

AHVC and CLS Subsystems of BioCNN Type-I DNA1 for CIFAR10					
AHVC	V2	[16 x 16 | 198]	8 V2 Reduction	[3 x 3 ~ 1 | 198 → 89]	maxpool [3 x 3 ~ 2] → LRN
	V3	[8 x 8 | 233]	9 V3 Reduction	[3 x 3 ~ 1 | 89 → 144]	maxpool [3 x 3 ~ 2] → LRN
	V4	[4 x 4 | 377]	10 V4	[3 x 3 ~ 1 | 144 → 144]	
	V5		11 V5 Reduction	[3 x 3 ~ 1 | 144 → 89]	maxpool [3 x 3 ~ 2] → LRN
CLS	FC1	[2 x 2 | 144]		[576 → 256]	
	FC2			[256 → 256]	
	SMAX			[256 → 10]	

Table 17: Differences in the AHVC between BioCNN-I DNA1 and BioCNN-II DNA5. The geometry of DNA1 is very narrow with only 89 features in the last convolutional module V5.

Comparing the differences in the AHVC subsystems between the genotypes DNA1 and DNA5, the second has a $[1 \times 1 \sim 1|198 \rightarrow 144\ ]$ NiN convolutional layer in its input, and a GLAVP before its output. Also DNA1 has a narrower CNN geometry with hyperdepths of 89 and 144 features and uses FC layers in its classifier.

6.5.5 BioCNN Architectures for Caltech101-2017

For the classification of images to 101 classes of the Caltech101-2017 there are three types of BioCNN architectures that were tested:

- Type-I has a classifier consisting of two fully connected layers that use 50% dropout on the $f(u) = \tanh(u)$ activations and a Softmax layer.
- Type-II has a global average pooling layer without dropout that feeds directly to the Softmax output layer.
- Type-III has a global average pooling layer without dropout that feeds a fully connected linear layer with 50% dropout on the $f(u) = u$ activations that feed the Softmax output layer.

Type-II and Type-II architectures could not be fully trained on Caltech101 because the early stopping mechanism detected 10 epochs without a change in their learning quality. Nevertheless the two types showed minimum overfitting during training, indicated by the OFM curves and can be investigated in future attempts.

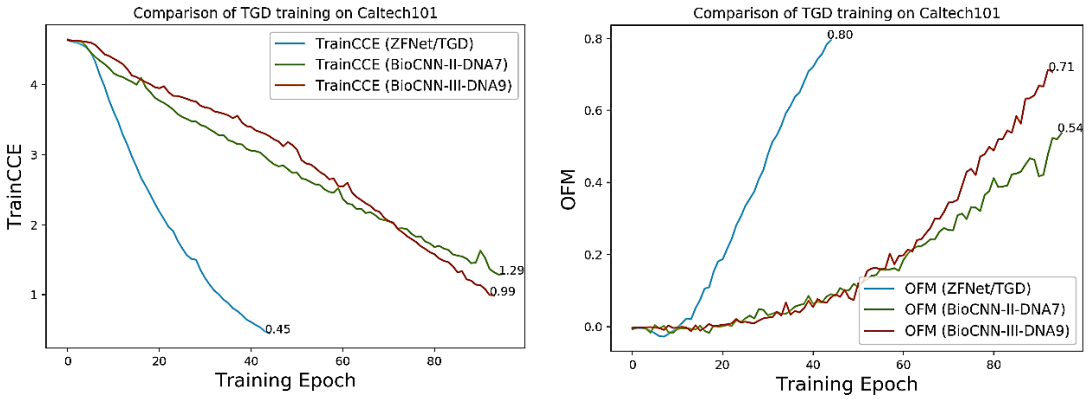

Figure 80: Early stopping when training Type II and Type III architectures on Caltech101. The OFM metric indicates low overfitting compared to ZFNet and the training CCE a slow convergence.

The Type-I architecture has less V1 convolutional layers with an APVC output hyperdepth of $h = 150$ and 4 convolutional modules in its AHVC without a NiN layer. Moreover a 5×5 spatial size exists in the AHVC as CONV module V3.

BioCNN Type-I Deep Network Architecture 12 (BioCNN-I DNA12) for Caltech101-2017

Subsystem	Module		Convolutional Layer				Pooling / Normalization /
Name	Name	Input	ID	Name	Subtype	Kernel	Regularization Layer
ARNN	OPL	[227 x 227 | 3]	1	Cones Rods		[1 x 1 | 3 → 2]	
			2	Horizontal Cells		[1 x 1 | 3 → 8]	
			3	Bipolar Cells		[1 x 1 | 8 → 14]	
	IPL	[227 x 227 | 14]	4	Amacrine Cells		[1 x 1 | 14 → 6]	
			5i	Ganglion Cells	p-GC	[3 x 3 ~ 3 | 6 → 12]	
			5ii		m-GC	[5 x 3 ~ 3 | 6 → 12]	
			5iii		k-DSGC	[11 x 11 ~ 3 | 6 → 84]	
				Optic Nerve (ONRV)		{ p-GC, m-GC, k-DSGC }	U → LRN
APVC	V1	[76 x 76 | 108]	6i	V1 Simple Cells	l=1	[3 x 3 ~ 1 | 108 → 6]	
			6ii		l=1.67	[5 x 5 ~ 1 | 108 → 6]	
			6iii		l=2.33	[7 x 7 ~ 1 | 108 → 6]	
			6iv		l=3.67	[11 x 11 ~ 1 | 108 → 6]	
			6v	V1 Complex Cells		[5 x 5 ~ 1 | 108 → 6]	
	LGN	[76 x 76 | 108]	6vi			[5 x 5 ~ 1 | 108 → 6]	
			6vii	LGN-V1 Simple Cells		[7 x 7 ~ 1 | 108 → 6]	
	ONRV		6viii	ONRV Shortcut Connection		[108 → 108]	U → LRN
AHVC	V2	[76 x 76 | 150]	7	V2 Reduction		[3 x 3 ~ 1 | 150 → 233]	maxpool [3 x 3 ~ 2] → LRN
	V3	[38 x 38 | 233]	9	V3 Reduction		[5 x 5 ~ 1 | 233 → 377]	maxpool [3 x 3 ~ 2] → LRN
	V4	[19 x 19 | 377]	10	V4		[3 x 3 ~ 1 | 377 → 233]	
	V5	[19 x 19 | 233]	11	V5 Reduction		[3 x 3 ~ 1 | 233 → 144]	maxpool [3 x 3 ~ 2] → LRN
CLS	FC1					[14400->2048]	
	FC2					[1440->1024]	
	SMAX					[144 → 101]	

Table 18: The BioCNN Type-I Deep Network Architecture 12 (BioCNN-I DNA12) that is trained for classification on the Caltech101-2017 random subset.

6.6 Image Classification Results

6.6.1 Classification Results on CIFAR10

For classification on CIFAR10, 4 architectures were trained with Throttled Gradient Descent learning on the standard training set with the following hyperparameters:

Architecture	Learning Rate Range	Momentum	Distance from best epoch for early stopping	Mini-batch Size
AlexNet	[0.0006 , 0.06597]	0.92	10	500
ZFNet	[0.0006 , 0.06597]	0.92	10	500
BioCNN-I DNA1	[0.0006 , 0.06597]	0.85	10	500
BioCNN-II DNA5	[0.0006 , 0.06597]	0.85	10	500

Table 19: TGD learning hyperparameters for each architecture trained on CIFAR.

For fairness, the comparisons on CIFAR10 are made between BioCNN and ZFNet implementations, since the stride of 4 in CONV1 is very large for the tiny images. The AlexNet metrics for CIFAR10 are only reported to indicate the decrease in performance. Two BioCNN architectures BioCNN-I DNA1, BioCNN-II DNA5, and reproductions of AlexNet and ZFNet, were trained on the CIFAR10 using the CVF-10 method. The Type-II BioCNN network which uses global average pooling, outperforms the ZFNet-Narrow architecture in all metrics of each individual fold:

Accuracy				
Fold Number	AlexNet	ZFNet	**BioCNN-I DNA1**	**BioCNN-II DNA5**
1	59.48	62.86	60.33	**65.62**
2	58.53	63.80	61.20	**65.59**
3	58.26	62.52	61.62	**64.97**
4	59.31	64.48	58.75	**65.88**
5	59.48	62.36	59.90	**65.99**
6	58.79	62.92	61.52	**65.88**
7	59.05	63.67	61.93	**66.53**
8	60.25	64.65	60.08	**65.35**
9	59.68	64.35	63.52	**65.09**
10	57.78	63.17	58.78	**64.75**
Average Abs. Deviation	0.58	0.71	1.20	**0.42**
Median	59.18	63.42	60.77	**65.61**
Mean	59.06	63.48	60.76	**65.57**

Table 20: CVF-10 comparison of classification accuracy for BioCNN, AlexNet and ZFNet architectures on standard CIFAR10 setup. Training was performed without data augmentation or preprocessing.

Precision				
Fold Number	AlexNet	ZFNet	**BioCNN-I DNA1**	**BioCNN-II DNA5**
1	59.97	63.85	60.65	**65.99**
2	58.67	64.43	62.01	**66.13**
3	58.36	63.32	62.30	**64.98**
4	59.99	64.63	58.80	**66.36**
5	59.97	63.17	60.69	**66.28**
6	58.99	63.63	61.92	**66.10**
7	59.52	64.30	62.76	**66.80**
8	60.75	65.09	60.71	**65.75**
9	59.92	64.57	64.07	**65.66**
10	58.34	63.65	58.81	**65.25**
Average Abs. Deviation	0.69	0.54	1.34	**0.42**
Median	59.72	64.08	61.32	**66.05**
Mean	59.45	64.06	61.27	**65.93**

Table 21: CVF-10 comparison of classification precision for BioCNN, AlexNet and ZFNet architectures on standard CIFAR10 setup, without data augmentation or preprocessing.

F1-Score				
Fold Number	AlexNet	ZFNet	**BioCNN-I DNA1**	**BioCNN-II DNA5**
1	59.76	63.42	60.70	**65.97**
2	58.70	64.32	61.75	**66.10**
3	58.44	63.00	62.19	**65.09**
4	59.75	64.70	58.95	**66.30**
5	59.76	62.91	60.37	**66.36**
6	59.02	63.43	61.95	**66.16**
7	59.41	64.19	62.54	**66.88**
8	60.62	65.09	60.59	**65.68**
9	59.94	64.59	64.03	**65.58**
10	57.98	63.48	58.96	**65.15**
Average Abs. Deviation	0.64	0.67	1.29	**0.44**
Median	59.58	63.84	61.23	**66.04**
Mean	59.34	63.91	61.20	**65.93**

Table 22: CVF-10 comparison of classification F1-Score for BioCNN, AlexNet and ZFNet architectures on standard CIFAR10 setup, without data augmentation or preprocessing.

The dimensions and stride [$11 \times 11 \sim 4$] of AlexNet are proven by this experiment as its disadvantage, when used on the tiny image size of 32×32. An interesting observation can be made about the average absolute deviation of accuracy, precision and F1-score. The BioCNN-II DNA5 has smaller average deviations than the second best ZFNet in all three metrics. For accuracy it is 0.42 for 10 folds, which is 40% smaller than ZFNet, indicating a stability of the architecture on different image subsets of the CIFAR10 standard training set.

Number of Epochs For Best Performance				
Fold Number	AlexNet	ZFNet	**BioCNN-I DNA1**	**BioCNN-II DNA5**
1	42	39	**35**	51
2	44	40	**29**	54
3	46	42	**31**	62
4	45	35	**30**	44
5	42	48	**38**	57
6	43	44	**37**	46
7	40	38	**31**	50
8	38	35	**35**	61
9	39	34	**36**	59
10	55	50	**37**	56
Average Abs Deviation	3.3	4.0	**2.9**	5.0
Median	43	40	**35**	55
Mean	43	41	**34**	54

Table 23: Indication of needed epochs to reach the best performance when training BioCNN, AlexNet and ZFNet architectures on standard CIFAR10 setup, without data augmentation or preprocessing.

The very narrow configuration of BioCNN-I DNA1 can be trained with TGD in approximately 35 epochs. An increased number of TGD training epochs is needed for BioCNN-II compared to ZFNet. The extra epochs could be caused by the additional 1×1 convolutional and the use of the GLAVP layer as its exit layer. Nevertheless, the baseline accuracy is reached in around 30 - 35 epochs of training as can be observed in the process monitoring graph for validation accuracy (ValAccuracy).

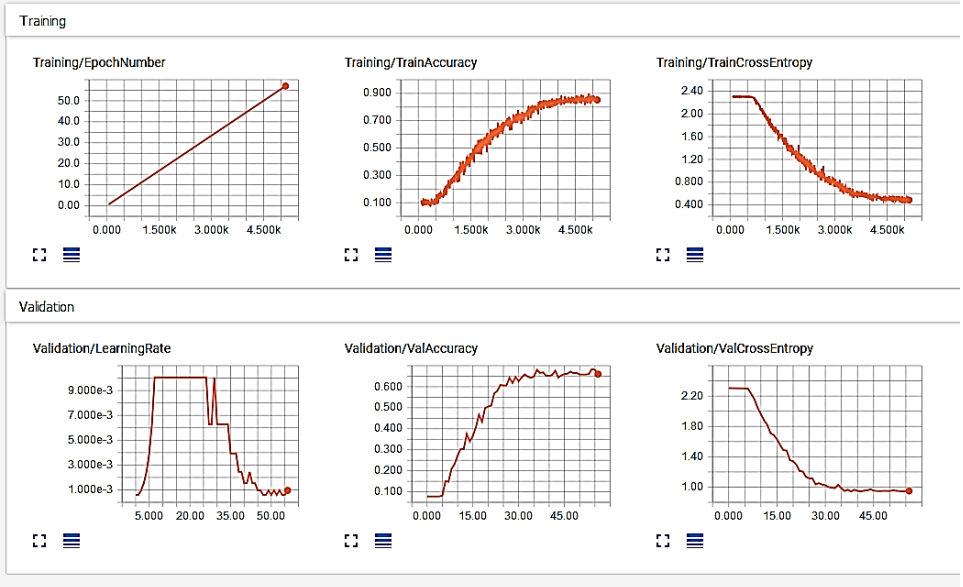

Figure 81: Google Tensorboard monitoring of TGD training process for the BioCNN-II DNA5, fold k=2, of the standard CIFAR training set. The granularity for the training panels is in training steps and for the validation panels in epochs. Data series are plotted without smoothing.

6.6.2 Classification Results on Caltech101

For classification on the random subset Caltech101-2017, three architectures are compared which have been trained with TGD using the following hyperparameters:

Architecture	Learning Rate Range	Momentum	Distance from best epoch for early stopping	Mini-batch Size
AlexNet	[0.0006 , 0.06597]	0.92	8	30
ZFNet	[0.0006 , 0.06597]	0.92	8	30
BioCNN-I DNA12	[0.001 , 0.0687]	0.92	12	30

Table 24: TGD learning hyperparameters for each architecture trained on Caltech101-2017.

Numerous attempts using BioCNN architectures were done on the 1st fold of the GT set to choose an architecture. The limited computational resources restricted the full CVF-10 evaluation to only one candidate architecture. The BioCNN-I DNA12 is a Type-I architecture with two FC layers that was compared against AlexNet and ZFNet.

The custom reproduction of the ZFNet architecture was the best model and displayed an accuracy 56%, significantly better than the one mentioned by the authors in [80]. The comparison between AlexNet and ZFNet shows small differences in all metrics, where in fold $k = 3$ the Precision and F1-score of AlexNet are slightly better. The BioCNN has the best Precision and F1-score only in fold $k = 7$.

Accuracy				
Fold Number	AlexNet	ZFNet Narrow	**BioCNN-I DNA12**	
1	54.26	**55.28**	53.89	
2	55.38	**56.13**	52.36	
3	56.13	**56.23**	54.43	
4	55.08	**55.72**	53.99	
5	54.50	**58.17**	53.24	
6	55.69	**55.99**	54.09	
7	53.82	**55.31**	55.04	
8	53.34	**54.30**	52.80	
9	55.96	**56.54**	54.26	
10	55.93	**56.33**	54.67	
Average Abs. Deviation	0.82	0.68	**0.65**	
Median	55.23	**56.06**	54.04	
Mean	55.01	**56.00**	53.88	

Table 25: CVF-10 comparison of classification accuracy for BioCNN, AlexNet and ZFNet architectures on Caltech101-2017. Training was performed without data augmentation or preprocessing.

Precision				
Fold Number	AlexNet	ZFNet Narrow	**BioCNN-I DNA12**	
1	60.14	**61.69**	59.83	
2	60.94	**61.82**	58.01	
3	**62.09**	61.64	59.51	
4	61.05	**61.08**	58.77	
5	60.19	**63.02**	58.56	
6	61.37	**61.39**	58.60	
7	59.96	60.09	**60.90**	
8	59.85	**60.33**	57.46	
9	61.53	**61.93**	59.71	
10	61.40	**61.70**	60.34	
Average Deviation	0.65	**0.60**	0.78	
Median	61.00	**61.67**	59.14	
Mean	60.85	**61.47**	59.17	

Table 26: CVF-10 comparison of classification precision for BioCNN, AlexNet and ZFNet architectures on Caltech101-2017, without data augmentation or preprocessing.

F1-Score				
Fold Number	AlexNet	ZFNet Narrow	**BioCNN-I DNA12**	
1	56.02	**57.04**	55.84	
2	57.08	**57.70**	53.87	
3	**57.86**	57.66	55.91	
4	56.81	**57.34**	55.60	
5	56.07	**59.58**	54.88	
6	57.42	**57.61**	55.42	
7	55.35	56.61	**56.88**	
8	55.18	**55.90**	54.02	
9	57.49	**58.17**	56.06	
10	57.52	**57.96**	56.46	
Average Deviation	0.82	**0.67**	0.76	
Median	56.95	**57.64**	55.72	
Mean	56.68	**57.56**	55.49	

Table 27: CVF-10 comparison of classification F1-Score for BioCNN, AlexNet and ZFNet architectures on Caltech101-2017, without data augmentation or preprocessing

The average absolute deviation of ZFNet is an indicator of a more stable architecture compared with AlexNet when they are both trained on a very small number of samples per class. The BioCNN Type-I architecture has competitive but lower performance when compared to ZFNet, indicating that optimization is most likely needed with regard to the chosen set of hyperparameters

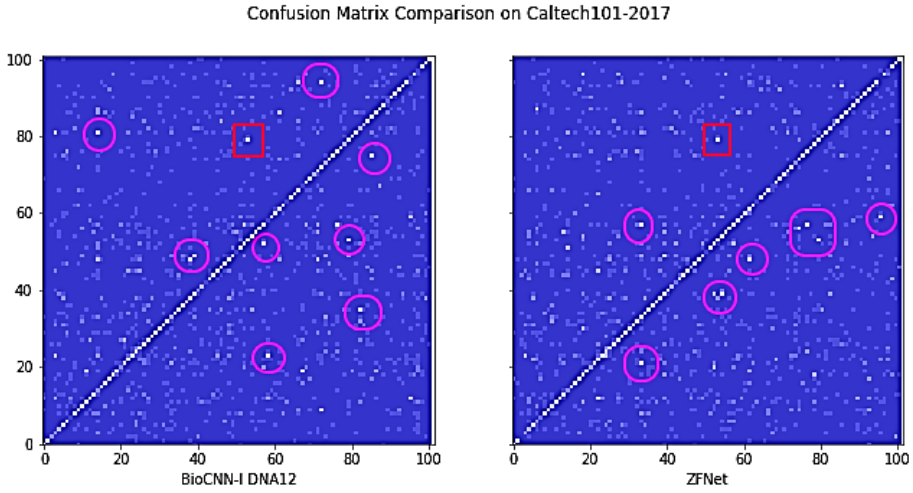

Figure 82: Confusion matrices of BioCNN and ZFNet when trained on Caltech101-2017 dataset using only 27 samples per class.

Comparison of the two confusion metrics, indicates a completely different convergence of the two models with only one common misclassification issue marked in figure 82 inside a red square. Monitoring the learning process with the use of CCE, Accuracy, APER and OFR can provide additional insight about the convergence and overfitting:

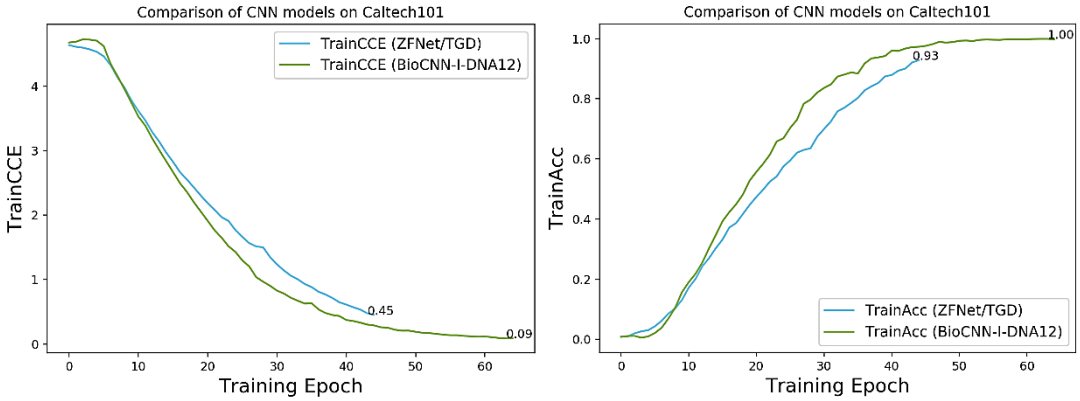

Figure 83: Comparison between the learning processes of BioCNN Type-I DNA 12 and ZFNet on the Caltech101-2017 random subset. Left: The average categorical cross entropy (CCE) error per epoch for the training session. Right: The training average accuracy per epoch.

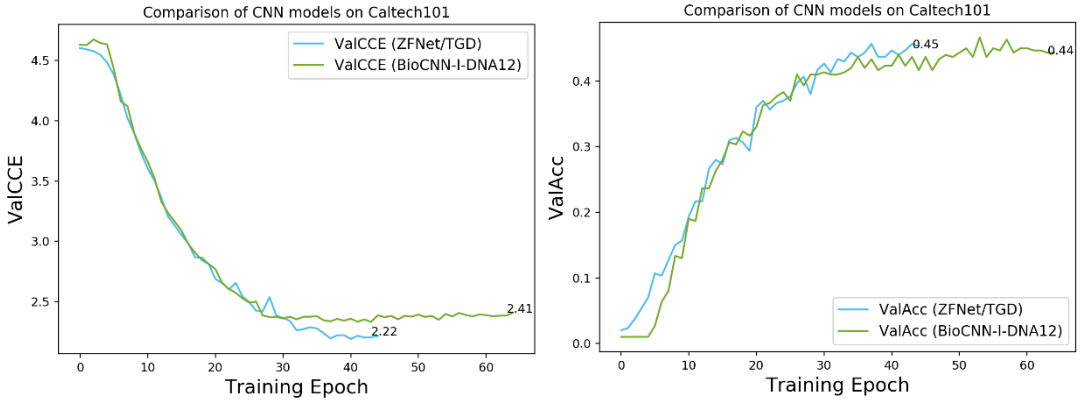

Figure 84 Left the validation CCE per epoch and right the validation accuracy per epoch.

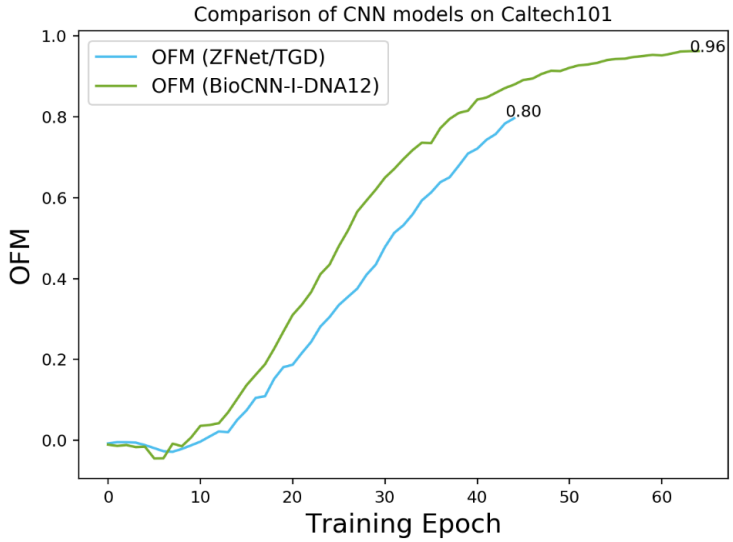

Figure 85: The custom metric Overfit Margin. A higher overfitting curve is noticed for the BioCNN architecture, an indicator of a non-optimized architecture

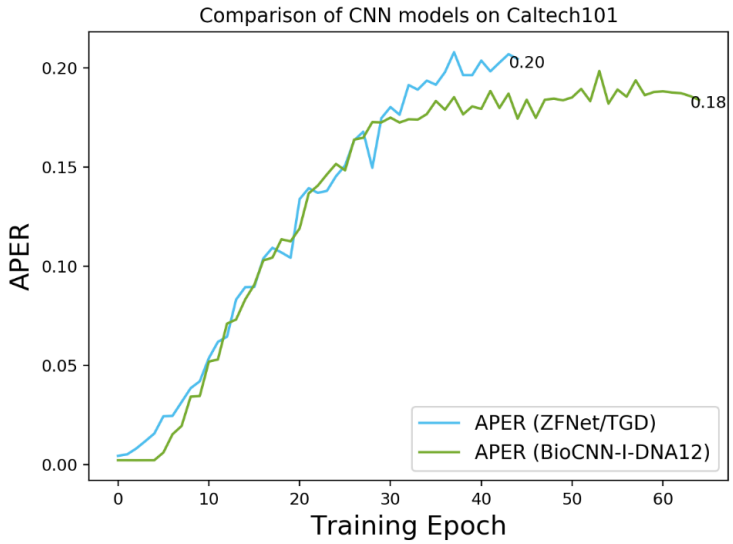

Figure 86: The custom metric average per error (APER). There is a crucial point around epoch 30 where the BioCNN architecture stops to improve its accuracy even though the CCE continues to decrease.

In the above figure there is an apparent point in the training of BioCNN where it can no longer increase its validation accuracy, thus fails to catch the ZFNet performance. This observation combined with the increased overfitting shown as the large span between the OFM curves in figure 85 leads to the following hypothesis: Overfitting helps the model to reach a validation accuracy above 40% but it has acquired a sub-optimal learned state in its convolutional kernel weights.

6.7 VFL Creation and Image Retrieval

6.7.1 VFL Creation Experiments

6.7.1.1 Dimensionality Reduction with PCA

The convolutional modules V3, V4 and V5 of the BioCNN-I DNA12 are used as feature descriptor sets for the creation of LVFL, MVFL and GVFL respectively. A series of PCA reductions has been performed to select two candidates based on the MSE reconstruction error, one for a drastic reduction and a second with a minimal reconstruction error. The choices for components were in the range of [16,144] with a step of 8, totaling 17 different reduction models. The chosen reduction methods were PCA with 24 and 144 components, where the last is actually a whitening operation on V5 activations, since this convolutional module has an output hyperdepth of $h = 144$ features.

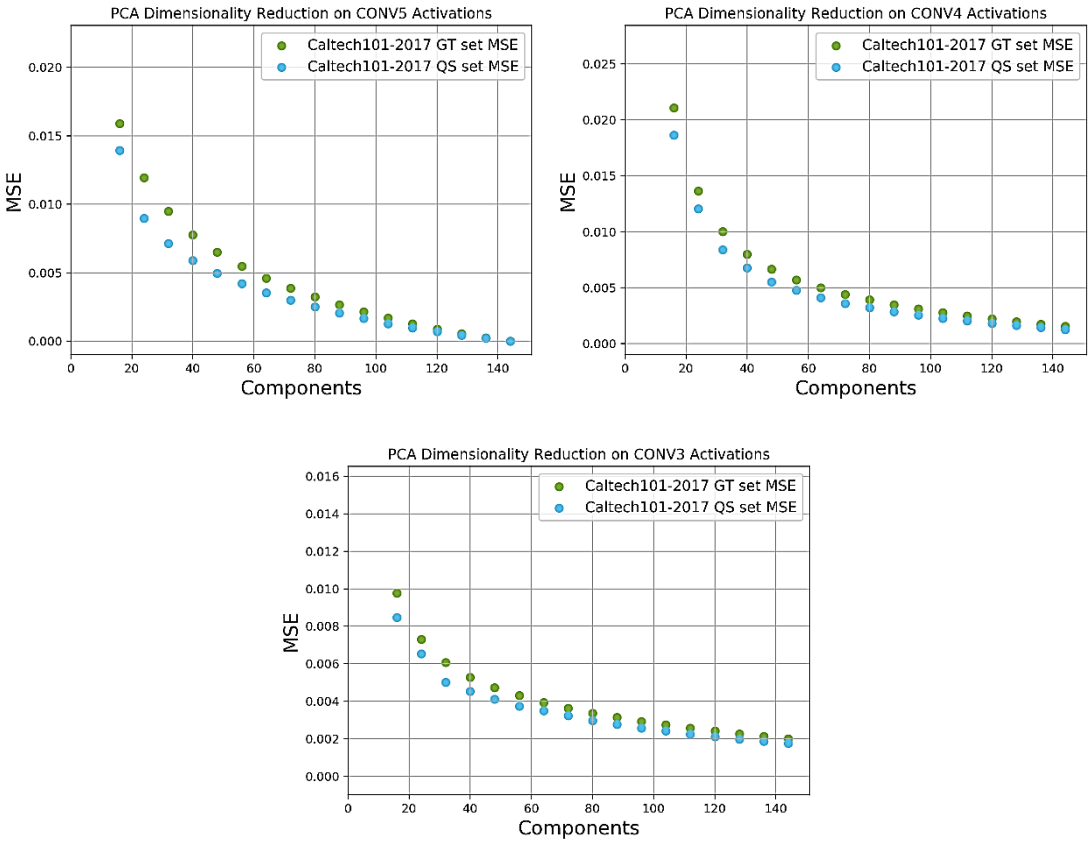

Figure 87: PCA dimensionality reductions on activations from BioCNN-I DNA12 modules for the ground truth set images and the query set images. Top left: Mean squared error (MSE) when using convolutional module V5, top right for V4 and bottom for V3 activations.

6.7.1.2 Clustering with RkMeans

The reference setup for clustering with Rooting k-Means has three levels in the hierarchy in order to generate 3-syllable visual words. The rooting factors are $RF_1 = 89$, $RF_2 = 34$ and $RF_3 = 13$ for the highest depth of the hierarchy which holds the most significant nodes in the cluster path. The corresponding 3-syllable visual word has $\{ 13, 34, 89 \}$ different syllables for the antepenultimate, penultimate and the last syllables respectively. Below is the complete setup for the creation of 6 different VFL vocabularies with a maximum of 39338 different visual words in the vocabulary.

	Deep Neural Network			Dimensionality Reduction		Clustering		
Lang.	Conv. Mod.	Activation Map [Size | Features]	Descriptors Count	PCA Drastic Reduction	PCA Whitening Reduction	RkM L1	RkM L2	RkM L3
LVFL	**V3**	**[19x19|377]**	**1083000**	**377 ≅ 24**	**377 ≅ 144**	**89**	**34**	**13**
MVFL	**V4**	**[19x19|233]**	**1083000**	**233 ≅ 24**	**233 ≅ 144**	**89**	**34**	**13**
GVFL	**V5**	**[10x10|144]**	**300000**	**144 ≅ 24**	**144 ≅ 144**	**89**	**34**	**13**

Table 28: Setup for the creation of VFL languages LFVL for V4, MFVL for V5 and GVFL for V6

No additional clustering experiments were run that would have needed a multiple of image retrieval experiments in order to evaluate them. Additionally the Silhouette metric provided only an indication of the clusters cohesion in each level of recursive process. It could not calculate an overall value for the descriptor set because the used implementation in SciKit-Learn (see chapter 7) has a limitation on the number of data points.

6.7.1.3 Zipf's law in Visual Features Language

The clustering process has created 39278 visual words for the GVFL and the word with the maximum frequency has 259 occurrences. For MFVL and LVFL there are 39338 visual words with the respective maximum frequencies of 1710 and 3829. Visualizing the rank of the visual words in the VFL by their frequencies, an interesting observation is made:

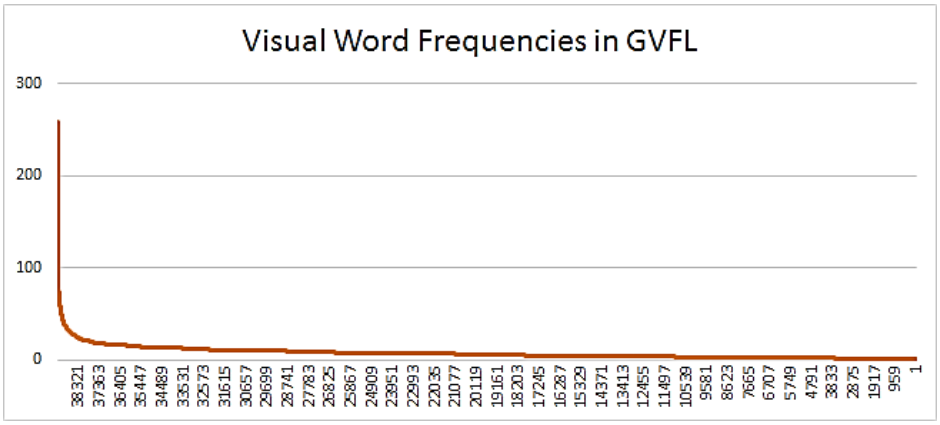

Figure 88: Frequencies of visual words in the GVFL vocabulary of V5 activations.

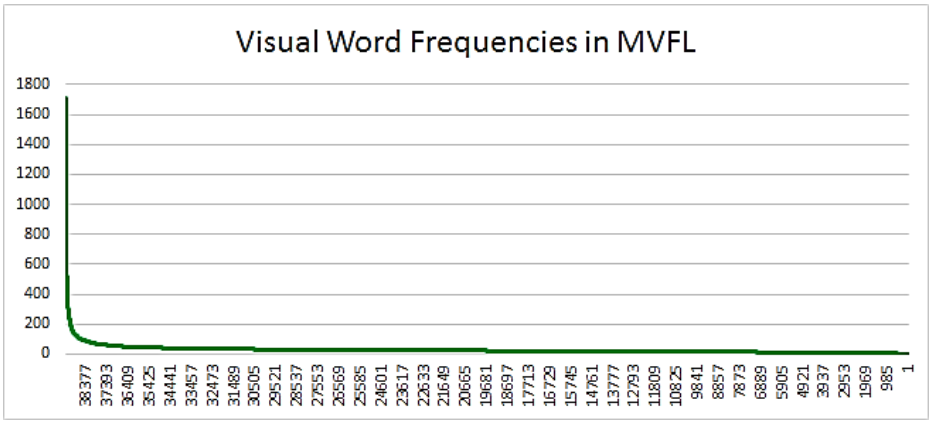

Figure 89: Frequencies of visual words in the MVFL vocabulary of V4 activations.

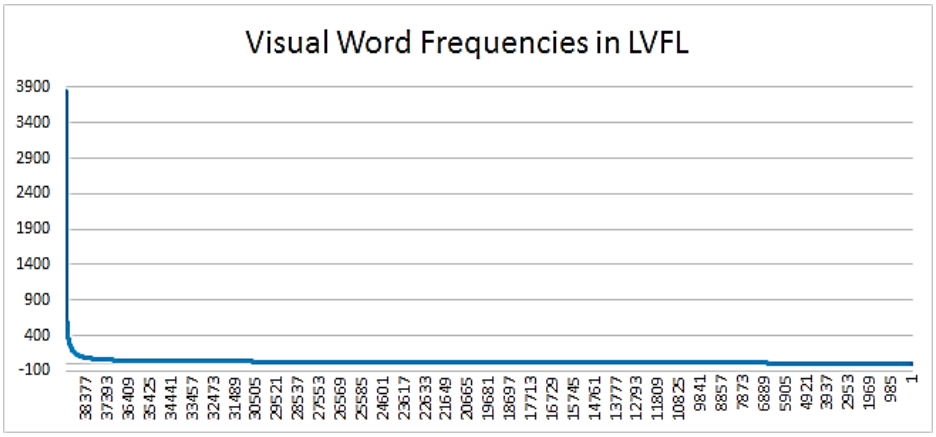

Figure 90: Frequencies of visual words in the LVFL vocabulary of V3 activations.

An appearance of the Zipfian distribution that is found in natural languages is observed for VFLs that have been generated indirectly from image pixels on a small collection of images, through Rooting k-Means clustering on BioCNN convolutional module activations. This is an intriguing observation that gives linguistic properties to the Visual Feature Language and supports the theory that common semantics of visual information that could be captured by a universal VFL. The appearance of Zipf's law in visual words has previously been observed for SIFT-based CBIR and the BoVW model [259]. Considering the Zipf's law applicability, a visual stopwords list can be created for each VFL using the method described in subsection 5.3.4.2. Depending on the statistics, low values of a might use up to $1/5$ of the vocabulary as visual stopwords.

	GVFL	MVFL	LVFL
Visual Words in VFL Vocabulary	39278	39338	39338
Average Visual Words Frequency (VWF)	7.64	27.53	27.53
Standard Deviation of VWF	7.86	40.97	61.96
wf_i Limit Multiplier a	0.7	0.5	0.4
wf_i Limit for Stopwords	10	33	35
Visual Stopword Count	10456	8110	6745
Visual Stopwords % of VFL	26.62%	20.62%	17.15%

Table 29: Stopword removal on GVFL, MVFL and LVFL with different values

6.7.2 Image Retrieval Experiments

6.7.2.1 Selection of classes for test queries

ClassID	ClassName	Sample Count	F1-Score
1	airplanes	50	95.8%
9	brain	50	68.8%
15	car_side	50	90.6%
36	faces	50	78.4%
37	faces_easy	50	96.1%
56	leopards	50	80.0%
62	menorah	50	75.8%
64	minaret	46	93.5%
65	motorbikes	50	90.9%
92	trilobite	50	82.2%

Table 30: Selected classes for retrieval queries

One image for each one of the above classes was used in the test query set for content-based image retrieval experiments. Even though the CNN learned state do not generalize adequately, these classes have a fair F1-Score and a large representation in the Caltech101 collection and are close to the 50 samples per class limit. This selection ensures a good representation of relevant topics in an imbalanced image collection like Caltech101. For retrieval experiments 10 random images from the Caltech101-2017 UT compose the random query images mini-set (RQS). Additionally images for the same 10 classes in the HIQ-1 mini-collection compose the handpicked query images mini-set (HQS).

6.7.2.2 Image retrieval on the Caltech101-2017 ground truth image collection

The retrieval of the RQS mini-set was repeated for multiple weighting schemes that are supported by TerrierIR and for the 6 different VFLs that were indexed. The count of the retrieved images for each query was 50 out of 3000 for the Caltech101-2017 collection. There is probability of being randomly relevant around 1% because each class contains 29 or 30 samples in the collection. Tables with the recorded values of PrecisionR, RecallR, mAP, Precision@10, Precision@20, Precision@30 and Precision@100 metrics are presented in the following pages. The best performing retrieval method is the Dirichlet Language Model (DLM) [260], a Bayesian language smoothing method [261] that displayed in all VFL experiments the best PrecisionR, RecallR and mAP. The top performing combination for image retrieval is using DLM with LVFL-144, a visual language that corresponds to activations from convolutional module V3 of BioCNN-I DNA12 with reduced dimensionality through PCA from 377 to 144 features.

Images from RQS	PrecisionR					
	BB2	BM25	DFR BM25	DLH	DLH13	DPH
GVFL-24	3.20%	2.40%	2.40%	2.40%	2.40%	2.40%
MVFL-24	2.60%	2.80%	2.80%	3.00%	3.00%	2.80%
LVFL-24	2.20%	2.00%	2.00%	2.00%	2.00%	2.00%
GVFL-144	2.25%	2.24%	2.24%	2.24%	2.24%	2.24%
MVFL-144	3.00%	3.20%	3.20%	3.20%	3.20%	3.20%
LVFL-144	3.60%	3.60%	3.60%	4.00%	4.00%	3.80%
	DFRee	Hiemstra LM	Dirichlet LM	IFB2	Inexp B2	Inexp C2
GVFL-24	2.40%	2.40%	**3.40%**	3.20%	3.20%	3.20%
MVFL-24	**3.20%**	3.00%	**4.20%**	2.60%	2.60%	2.40%
LVFL-24	2.00%	2.00%	**4.00%**	2.20%	2.20%	2.40%
GVFL-144	2.24%	2.24%	**2.44%**	2.24%	2.24%	2.24%
MVFL-144	3.20%	3.20%	**3.80%**	3.00%	3.00%	2.80%
LVFL-144	4.20%	4.20%	**6.00%**	3.40%	3.40%	3.60%
	InL2	Lemur TF-IDF	LGD	PL2	TF-IDF	
GVFL-24	2.40%	2.60%	2.40%	2.40%	2.40%	
MVFL-24	2.80%	2.80%	**3.20%**	2.80%	2.80%	
LVFL-24	2.00%	2.00%	2.00%	2.00%	2.00%	
GVFL-144	2.24%	2.24%	2.24%	2.24%	2.24%	
MVFL-144	3.20%	3.00%	3.20%	3.20%	3.20%	
LVFL-144	3.80%	3.60%	4.00%	4.00%	3.60%	

Table 31: PrecisionR metric for CBIR experiments using BioCNN-I DNA12 as feature extractor. Image queries use the RQS mini-set.

Images from RQS	RecallR					
	BB2	BM25	DFR BM25	DLH	DLH13	DPH
GVFL-24	5.52%	4.14%	4.14%	4.14%	4.14%	4.14%
MVFL-24	4.48%	4.83%	4.83%	5.17%	5.17%	4.83%
LVFL-24	3.79%	3.45%	3.45%	3.45%	3.45%	3.45%
GVFL-144	3.79%	3.79%	3.79%	3.79%	3.79%	3.79%
MVFL-144	5.17%	5.52%	5.52%	5.52%	5.52%	5.52%
LVFL-144	6.21%	6.21%	6.21%	6.90%	6.90%	6.55%
	DFRee	Hiemstra LM	Dirichlet LM	IFB2	Inexp B2	Inexp C2
GVFL-24	4.14%	4.14%	**5.86%**	5.52%	5.52%	5.52%
MVFL-24	5.52%	5.17%	**7.24%**	4.48%	4.48%	4.14%
LVFL-24	3.45%	3.45%	**6.90%**	3.79%	3.79%	4.14%
GVFL-144	3.79%	3.79%	**4.14%**	3.79%	3.79%	3.79%
MVFL-144	5.52%	5.52%	**6.55%**	5.17%	5.17%	4.83%
LVFL-144	**7.24%**	**7.24%**	**10.34%**	5.86%	5.86%	6.21%
	InL2	Lemur TF-IDF	LGD	PL2	TF-IDF	
GVFL-24	4.14%	4.48%	4.14%	4.14%	4.14%	
MVFL-24	4.83%	4.83%	5.52%	4.83%	4.83%	
LVFL-24	3.45%	3.45%	3.45%	3.45%	3.45%	
GVFL-144	3.79%	3.79%	3.79%	3.79%	3.79%	
MVFL-144	5.52%	5.17%	5.52%	5.52%	5.52%	
LVFL-144	6.55%	6.21%	6.90%	6.90%	6.21%	

Table 32. RecallR metric for CBIR experiments using BioCNN-I DNA12 as feature extractor. Image queries use the RQS mini-set.

Images from RQS	mAP					
	BB2	BM25	DFR BM25	DLH	DLH13	DPH
GVFL-24	0.70%	0.72%	0.72%	0.79%	0.79%	0.73%
MVFL-24	0.60%	0.86%	0.83%	0.99%	0.98%	0.95%
LVFL-24	0.72%	0.27%	0.26%	0.25%	0.25%	0.25%
GVFL-144	0.93%	1.22%	1.22%	1.28%	1.28%	1.28%
MVFL-144	0.67%	1.09%	1.07%	1.06%	1.18%	1.01%
LVFL-144	0.57%	0.54%	0.51%	0.59%	0.59%	0.54%
	DFRee	Hiemstra LM	Dirichlet LM	IFB2	Inexp B2	Inexp C2
GVFL-24	0.95%	1.09%	**1.47%**	0.70%	0.70%	0.64%
MVFL-24	1.03%	1.00%	**2.10%**	0.59%	0.59%	0.58%
LVFL-24	0.23%	0.23%	**1.01%**	0.65%	0.65%	0.67%
GVFL-144	**1.33%**	1.28%	**1.34%**	0.83%	0.83%	0.83%
MVFL-144	1.19%	1.17%	**1.71%**	0.67%	0.67%	0.60%
LVFL-144	0.61%	0.62%	**2.05%**	0.55%	0.55%	0.55%
	InL2	Lemur TF-IDF	LGD	PL2	TF-IDF	
GVFL-24	0.72%	0.80%	0.96%	0.73%	0.72%	
MVFL-24	0.85%	0.87%	0.90%	0.95%	0.82%	
LVFL-24	0.26%	0.31%	0.24%	0.25%	0.26%	
GVFL-144	1.22%	1.19%	**1.29%**	1.25%	1.22%	
MVFL-144	1.09%	1.03%	1.17%	1.00%	1.07%	
LVFL-144	0.54%	0.49%	0.56%	0.59%	0.51%	

Table 33. Mean average precision (mAP) metric for CBIR experiments using BioCNN-I DNA12 as feature extractor. Image queries use the RQS mini-set.

Images from RQS	P@10					
	BB2	BM25	DFR BM25	DLH	DLH13	DPH
GVFL-24	2.00%	2.00%	2.00%	2.00%	2.00%	2.00%
MVFL-24	2.00%	4.00%	4.00%	4.00%	4.00%	4.00%
LVFL-24	2.00%	2.00%	2.00%	2.00%	2.00%	2.00%
GVFL-144	6.00%	**7.00%**	**7.00%**	**7.00%**	**7.00%**	**7.00%**
MVFL-144	3.00%	5.00%	**4.00%**	6.00%	6.00%	4.00%
LVFL-144	1.00%	1.00%	1.00%	1.00%	1.00%	1.00%
	DFRee	Hiemstra LM	Dirichlet LM	IFB2	Inexp B2	Inexp C2
GVFL-24	2.00%	2.00%	**3.00%**	2.00%	2.00%	2.00%
MVFL-24	4.00%	4.00%	**7.00%**	2.00%	2.00%	2.00%
LVFL-24	1.00%	1.00%	**4.00%**	2.00%	2.00%	2.00%
GVFL-144	**7.00%**	6.00%	5.00%	5.00%	5.00%	5.00%
MVFL-144	5.00%	5.00%	**4.00%**	3.00%	3.00%	3.00%
LVFL-144	2.00%	1.00%	**6.00%**	1.00%	1.00%	1.00%
	InL2	Lemur TF-IDF	LGD	PL2	TF-IDF	
GVFL-24	2.00%	2.00%	2.00%	2.00%	2.00%	
MVFL-24	4.00%	4.00%	4.00%	4.00%	4.00%	
LVFL-24	2.00%	2.00%	2.00%	2.00%	2.00%	
GVFL-144	**7.00%**	**7.00%**	6.00%	**7.00%**	**7.00%**	
MVFL-144	5.00%	4.00%	5.00%	4.00%	4.00%	
LVFL-144	1.00%	1.00%	1.00%	1.00%	1.00%	

Table 34. Recall at 10 metric for CBIR experiments using BioCNN-I DNA12 as feature extractor. Image queries use the RQS mini-set.

Images from RQS	P@20						
	BB2	BM25	DFR BM25	DLH	DLH13	DPH	
GVFL-24	3.00%	2.50%	2.50%	2.50%	2.50%	2.50%	
MVFL-24	2.50%	**4.00%**	3.50%	**4.00%**	**4.00%**	**4.00%**	
LVFL-24	1.50%	2.00%	1.50%	1.50%	1.50%	1.50%	
GVFL-144	**4.50%**	**4.50%**	**4.50%**	4.00%	4.00%	4.00%	
MVFL-144	4.00%	4.00%	4.00%	3.50%	4.00%	4.00%	
LVFL-144	2.50%	2.50%	2.00%	2.00%	2.00%	2.00%	
	DFRee	Hiemstra LM	Dirichlet LM	IFB2	Inexp B2	Inexp C2	
GVFL-24	2.50%	**4.00%**	**5.50%**	2.50%	2.50%	3.00%	
MVFL-24	**4.00%**	**4.00%**	**4.00%**	2.50%	2.50%	2.50%	
LVFL-24	1.50%	1.50%	**4.50%**	1.50%	1.50%	1.50%	
GVFL-144	4.00%	4.00%	4.00%	4.00%	4.00%	4.00%	
MVFL-144	4.00%	3.50%	**5.00%**	3.50%	3.50%	3.50%	
LVFL-144	2.00%	2.00%	**5.50%**	2.00%	2.00%	2.50%	
	InL2	Lemur TF-IDF	LGD	PL2	TF-IDF		
GVFL-24	2.50%	2.50%	3.00%	2.50%	2.00%		
MVFL-24	**4.00%**	2.50%	**4.00%**	**4.00%**	3.50%		
LVFL-24	1.50%	2.00%	1.50%	1.50%	1.50%		
GVFL-144	**4.50%**	4.00%	**4.50%**	4.00%	**4.50%**		
MVFL-144	3.50%	4.00%	4.00%	4.00%	4.00%		
LVFL-144	2.00%	2.00%	2.00%	2.00%	2.00%		

Table 35: Recall at 20 metric for CBIR experiments using BioCNN-I DNA12 as feature extractor. Image queries use the RQS mini-set.

Images from RQS	P@30						
	BB2	BM25	DFR BM25	DLH	DLH13	DPH	
GVFL-24	3.67%	3.67%	3.67%	3.67%	3.67%	3.67%	
MVFL-24	2.33%	3.33%	3.00%	3.33%	3.33%	3.33%	
LVFL-24	2.00%	2.00%	2.00%	2.67%	2.67%	2.00%	
GVFL-144	3.67%	3.33%	3.33%	3.33%	3.33%	3.33%	
MVFL-144	3.00%	3.33%	3.00%	3.33%	3.33%	3.00%	
LVFL-144	3.00%	4.00%	3.33%	3.67%	4.00%	3.67%	
	DFRee	Hiemstra LM	Dirichlet LM	IFB2	Inexp B2	Inexp C2	
GVFL-24	3.67%	3.67%	**5.00%**	3.67%	3.67%	3.33%	
MVFL-24	3.33%	3.33%	**4.00%**	2.33%	2.33%	2.33%	
LVFL-24	2.67%	2.67%	**4.00%**	1.67%	1.67%	1.67%	
GVFL-144	3.33%	3.33%	**3.67%**	3.33%	3.33%	3.33%	
MVFL-144	3.00%	3.33%	**5.00%**	3.00%	3.00%	3.00%	
LVFL-144	4.00%	4.00%	**5.67%**	3.00%	3.00%	3.00%	
	InL2	Lemur TF-IDF	LGD	PL2	TF-IDF		
GVFL-24	3.67%	3.67%	3.67%	3.67%	3.67%		
MVFL-24	3.00%	3.00%	3.00%	3.33%	3.00%		
LVFL-24	2.00%	2.00%	2.67%	2.00%	2.00%		
GVFL-144	3.33%	3.33%	3.33%	3.33%	3.33%		
MVFL-144	3.33%	3.00%	3.67%	3.00%	3.00%		
LVFL-144	3.67%	3.00%	4.00%	3.33%	3.33%		

Table 36: Recall at 30 metric for CBIR experiments using BioCNN-I DNA12 as feature extractor. Image queries use the RQS mini-set.

Images from RQS	P@100					
	BB2	BM25	DFR BM25	DLH	DLH13	DPH
GVFL-24	1.60%	1.20%	1.20%	1.20%	1.20%	1.20%
MVFL-24	1.30%	1.40%	1.40%	1.50%	1.50%	1.40%
LVFL-24	1.10%	1.00%	1.00%	1.00%	1.00%	1.00%
GVFL-144	1.10%	1.10%	1.10%	1.10%	1.10%	1.10%
MVFL-144	1.50%	1.60%	1.60%	1.60%	1.60%	1.60%
LVFL-144	1.80%	1.80%	1.80%	2.00%	2.00%	1.90%
	DFRee	Hiemstra LM	Dirichlet LM	IFB2	Inexp B2	Inexp C2
GVFL-24	1.20%	1.20%	**1.70%**	1.60%	1.60%	1.60%
MVFL-24	1.60%	1.50%	**2.10%**	1.30%	1.30%	1.20%
LVFL-24	1.00%	1.00%	**2.00%**	1.10%	1.10%	1.20%
GVFL-144	1.10%	1.10%	**1.20%**	1.10%	1.10%	1.10%
MVFL-144	1.60%	1.60%	**1.90%**	1.50%	1.50%	1.40%
LVFL-144	2.10%	2.10%	**3.00%**	1.70%	1.70%	1.80%
	InL2	Lemur TF-IDF	LGD	PL2	TF-IDF	
GVFL-24	1.20%	1.30%	1.20%	1.20%	1.20%	
MVFL-24	1.40%	1.40%	1.60%	1.40%	1.40%	
LVFL-24	1.00%	1.00%	1.00%	1.00%	1.00%	
GVFL-144	1.10%	1.10%	1.10%	1.10%	1.10%	
MVFL-144	1.60%	1.50%	1.60%	1.60%	1.60%	
LVFL-144	1.90%	1.80%	2.00%	2.00%	1.80%	

Table 37: Recall at 100 metric for CBIR experiments using BioCNN-I DNA12 as feature extractor. Image queries use the RQS mini-set.

The results for the Mean Average Precision metric indicate an inadequate performance of the BioCNN-CBIR pipeline on a collection of 3000 random images from Caltech101 dataset. The P@10 and P@20 show ambiguous results. For P@30 and P@100 the superiority of using DLM with LVFL-144 over other combinations of weighting schemes and VFLs is confirmed. Findings in experiments indicate that the CBIR pipeline is far from optimal and indicate that the VFE and VFL modules need improvements.

Dirichlet LM	**PrecisionR**		**RecallR**		**mAP**	
	RQI	HQI	RQI	HQI	RQI	HQI
GVFL-144	2.44%	**3.89%**	4.14%	**6.55%**	1.34%	**1.51%**
MVFL-144	**3.80%**	2.60%	**6.55%**	4.48%	**1.71%**	0.32%
LVFL-144	**6.00%**	1.80%	**10.34%**	3.10%	**2.05%**	0.16%
Dirichlet LM	**P@20**		**P@30**		**P@100**	
	RQI	HQI	RQI	HQI	RQI	HQI
GVFL-144	4.00%	**6.50%**	3.67%	**4.67%**	1.20%	**1.90%**
MVFL-144	**5.00%**	2.50%	**5.00%**	2.67%	**1.90%**	1.30%
LVFL-144	**5.50%**	1.00%	**5.67%**	1.33%	**3.00%**	0.90%

Table 38: Comparison of Dirichlet LM retrieval for RQI and HQI mini-sets

Nevertheless an interesting observation is made on the retrieval results when querying the image collection with the completely unknown HQS images. The GVF-144 performs better for handpicked images than for randomly selected images from Caltech101. This could be based on the fact that the V5 weights are more discriminative to the global visual

characteristics of a class, or simply caused by the different frequency distribution of the visual words in GVFL. However with such low values for the metrics no safe conclusions can be drawn.

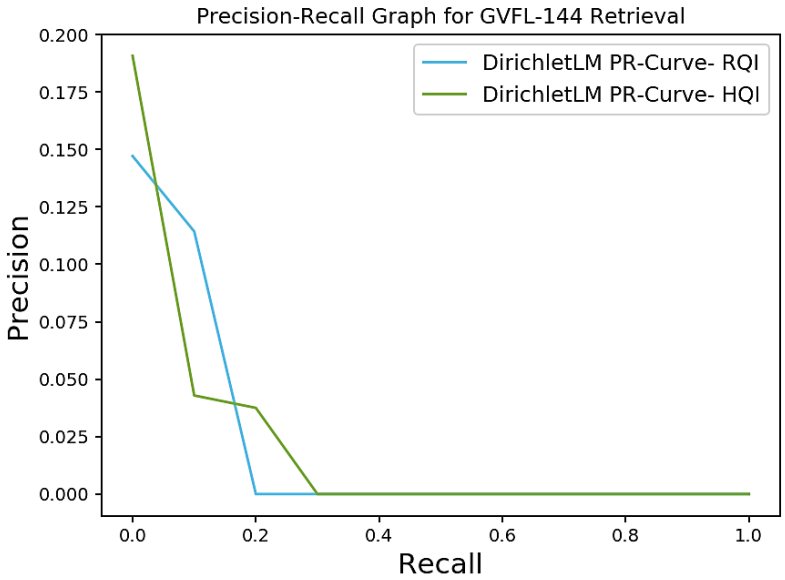

Figure 91: Comparing the P-R curves of RQS and HQS mini-sets for Dirichlet LM retrieval on GVFL-144.

6.8 Discussion

6.8.1 Best Performance on CIFAR10 Using a Fraction of Parameters

In the CIFAR10 experiments the number of parameters for the best performing BioCNN architecture are an order of magnitude less than the second best ZFNet. The BioCNN-II DNA5 has a total of 2.19 million parameters, with 1.94 million learnable and 250 thousand pre-defined parameters. The compared AlexNet / ZFNet architectures require 21.6 million learnable parameters for 32×32 image input. The BioCNN uses 10 times less parameters for reaching and surpassing the level of ZFNet performance, without preprocessing and augmentation. This finding is promising for reducing memory requirements of convolutional neural networks in order to use them in devices with limited capacity. A rough estimation of memory requirements can be made by inspecting the file size of the saved states, which are in Tensorflow binary format. The BioCNN-II DNA5 saved model occupies 19.5 MB of disk space, while the ZFNet 164 MB. Small memory footprint is a requirement for using deep neural networks in mobile devices for the Internet of Things (IoT). They could have a minimal hardware design with a few gigabytes of memory and a massively parallel processing unit with hundreds of ALUs.

The use of ELU activation functions in BioCNN helps the convergence in combination with the Xavier initialization of the weights. In future works the BioCNN weights could be initialized with the LSUV method and test the combination of ELU + LSUV for better TGD training convergence. The combined power of a better activation function and a better initialization technique, could also be a reason for the performance of the BioCNN architectures. To reach a safe conclusion whether the speed and accuracy are related to the use of ELU, or are a result of the architectural and functional novelties introduced by BioCNN, or both of them, a more intense research effort is required. The classification experiments on CIFAR10 can only be considered as indicators about potential performance.

6.8.2 Confirming the Transferability of BioCNN Features.

The 10 different models which resulted from 10 training folds on CIFAR10, have displayed smaller average deviation than ZFNet. This should be further studied with a 10-fold training on the ImageNet dataset to assess the stability of the BioCNN architecture. With a BioCNN pre-trained on ImageNet the quality of its convolutional feature extractors could be evaluated through transfer-learning experiments. A secure method for evaluating the transferability of any CNN feature extractor would be to train with CVF-10 on ILSVRC2012 and fine-tune 10 times on the destination dataset. This will provide 100 different models in a

method that could be described as fine-tuning with k^2 *Hold-Out-Combination Cross Validation.* A CNN model with decreased average absolute deviation in 100 folds could be a candidate general purpose feature extractor for CBIR.

There is also an important question that needs to be answered through future transfer learning experiments. Are the weights learned on ImageNet simply a better initialization for gradient descent learning on the destination dataset, analogous to an optimal DBN pre-training, or constitute universal feature extractors?

6.8.3 Competitive Performance on Caltech101-2017

The BioCNN Type-I architecture could not reach the performance of ZFNet in Caltech101 dataset but it used 35.6 million parameters, that is the half of ZFNet for 101 classes. Moreover using a stride $s = 3$ is not experimentally proven to be inferior as a choice and with more optimization could potentially reach the same performance as $s = 2$. BioCNN performs non-overlapping $[3 \times 3 \sim 3]$ convolutions in the p-GC convolutional layer, which is inspired from the small-sized parvocellular ganglion cells. Design space exploration of the numerous hyperparameters that exist for the Artificial Retinal Neural Network could discover better receptive fields for the GC kernels, and propagate more pixel-level detail to the next layers.

The reproductions of the AlexNet and ZFNet architecture were trained on the randomly chosen subset Caltech101-2017 with the standard setup $30\ train\ /\ 1 \leq test \leq 50$ without preprocessing and data augmentation. The accuracy was very low compared to transfer learning, for AlexNet 55.01% and for ZFNet 56%. Surprisingly this is significantly better than the reported $46.5\% \pm 1.7$ in table 4 of the original paper [80] when the model was also trained from scratch. A first reason could be the Xavier initialization that was used for all the architectures. It might also be an indicator of the potential benefits of TGD for training CNN classifiers. This needs to be testified by running a comparative study with other training methods and other variable learning rate schemes. Porting TGD to other domains of problems could provide indications whether it is as generally applicable variable learning rate method.

The needed time for a training epoch of AlexNet with the used hardware (see section 7.2.1) was 24 seconds, ZFNet needed approximately $\times$ 1.6 times more than AlexNet and BioCNN Type-I $\times$ 2.5 more than ZFNet. The speed of BioCNNs can be improved by re-examining the dense connections between the ARNN and APVC. Additionally the sparsity inside the ARNN can be improved, considering 4-5 receptive field maps in the GC kernels that are pre-defined to zero, because the GCM convolves only with 2-3 out of 6 ACL color channels. Moreover the V1 cells which have with an input hyperdepth of 108 can be sparsely connected with GCM according to the V1C grayscale paradigm.

Clusters for red, green, blue and yellow can detect lines of specific color using only the corresponding opponent GC's. Slicing of the incoming GC feature tensor could be implemented in more elaborate artificial LGN implementation.

6.8.4 Feature Quality of Sub-Optimal Classifiers

The Type-II architectures without the fully connected layers have performed better than Type-I in CIFAR10 classification. For Caltech101, the training of Type-II and Type-III architectures was terminated prematurely, but at the same time the custom OFM metric indicated that the models didn’t overfit on the training data. This could be due to the regularization properties of GLAVP layers when used directly or through a linear recombination as input to a Softmax output layer. Observing this metric in conjunction with the fact that early stopping occurred before BioCNN architectures overfitted an intuitive claim can be made: There might be a point in the gradient descent training of CNN where feature extraction in the kernels is more universal than the one learned on a successful ending, which facilitates high classification accuracy. This is a claim that has to be investigated with transfer learning and image retrieval experiments performed with the intermediate learned states of a CNN that are sub-optimal for the classification task.

6.8.5 Benefits from the Visual Features Language Concept

Counting the frequency of occurrence for each VFL visual word on the indexed text representation of the Caltech101-2017 collection helped to make an interesting observation. Descriptors which are assembled from CNN activation of image regions, dimensionally reduced through PCA and quantized through a hierarchy of clusters seem to follow the Zipf's law. According to this visual stopwords were be determined and assembled in a stopwords list exactly like in text retrieval. This is a benefit from converting digital image representations into VFL textual representations as the CBIR practice proposed by this thesis. This list can be processed by a text search engine without any modifications, allowing CBIR to be performed with existing text retrieval tools.

Abstracting the combinations of multi-dimensional feature values into actual readable words can provide insight for the image content. It could help researchers in the field of Information Retrieval discover other analogies natural languages and VFLs that are languages which describe visual information.

6.8.6 Understanding of the CNN-CBIR complexity

The retrieval experiments aimed to understand and explore the multiplicity of design choices for a CNN-CBIR model. The baseline image collection of 3000 images and a mere 2 sets of 10 image queries could not possibly provide any safe conclusions. The reported metrics show that the model could not achieve adequate image retrieval precision and accuracy. The benefit by this attempt is the creation of an experimental process which unveiled the complexity of optimizing a CNN-CBIR pipeline. Based on this experience the combination of the following factors is considered to affect CBIR retrieval performance for any used CNN architecture:

1. Selection of the convolutional layers that are best performing for retrieval. These can be determined only by the retrieval results at the end of the pipeline.
2. Number of principal components used in the PCA transformation, or completely different methods of dimensionality reduction.
3. Choosing a different vector quantization method, or determining the selection of the RkMeans clustering hyperparameters:
    a. Depth of the hierarchy.
    b. Rooting factors of each level and the ratios between them.
    c. Minibatch size for k-Means clustering.
4. Decision whether the ground truth images or an unknown set of images will be recalled through the CNN to provide the descriptor set for the creation of the VFL.

Combined efforts of research on the fields of Deep Neural Networks, Data Mining and Information Retrieval could discover with the best performing CNN-CBIR models.

7 Software Engineering Details

7.1 Three Tier Software Architecture

The BioCNN-CBIR system can be described in terms of Software Engineering (SE) and fit in the context of its parent domain of web intelligence. The CBIR system will provide the final user experience (UX) of an intelligent image search engine. In the world of business software, the three-tier architecture [262] has several benefits for the development and maintenance of complex software projects. There is typically a database as the *data tier* and a user interface in which the data are presented, the *presentation tier*. In-between them, the *business logic tier* holds the core logic of the business application that is decoupled from the data medium and the user interface implementation. We can see the BioCNN-CBIR implementation through the same prism as a *three-tier intelligent architecture* for the web 4.0 that has the following tiers:

a) The *Data Memory Tier* (DMT) contains the image dataset, the saved stated of a CNN model, the derived VFL cluster model, configuration and other data. The storage medium can be a database, a file system or a cloud storage service.
b) The *Machine Learning Intelligence Tier* (MLI) implements the intelligent logic that will empower the web 4.0 application. It can also be named as *Artificial Intelligence Tier (AIT)*, which is a more general term that includes symbolic AI approaches.
c) The *User Experience Tier* (UXT) that implements the user interface and the human-computer interaction (HCI) like input, presentation and user feedback.

As described in Chapter 5 the MLI tier in the BioCNN-CBIR system has the VFE, the VFL and VMM modules. The CNN, hierarchical cluster model and IR search engine are loosely coupled with the implementation of DMT and decoupled from UXT. In this chapter some details of technical nature are described through the Software Engineering prism. The following section 7.2 describes the environment that has been used by the process of Machine Learning Research and Development (ML R&D)

7.2 Machine Learning R&D Environment

7.2.1 Accelerated Computing with CUDA

In order to train the computationally expensive deep CNNs, hardware accelerated computations are used. The solutions are based on the NVidia CUDA parallel computing platform [263] that operates in CUDA-enabled hardware [264] and provides Application Programming Interfaces (APIs) [265] for general purpose computations hosted on a Graphics Processor Unit (GPU).

The hardware used for this Master's thesis is an off-the-shelf graphics card that has an NVidia GeForce GTX 960 GPU and 4GB of RAM. This architecture has 1024 mini-ALUs the *CUDA cores* which can parallelize expensive computations like convolution, when an eight-core CPU has only 8 ALUs. The enormous performance gains over CPU makes GPU computing the default choice for Deep Learning on desktop computers.

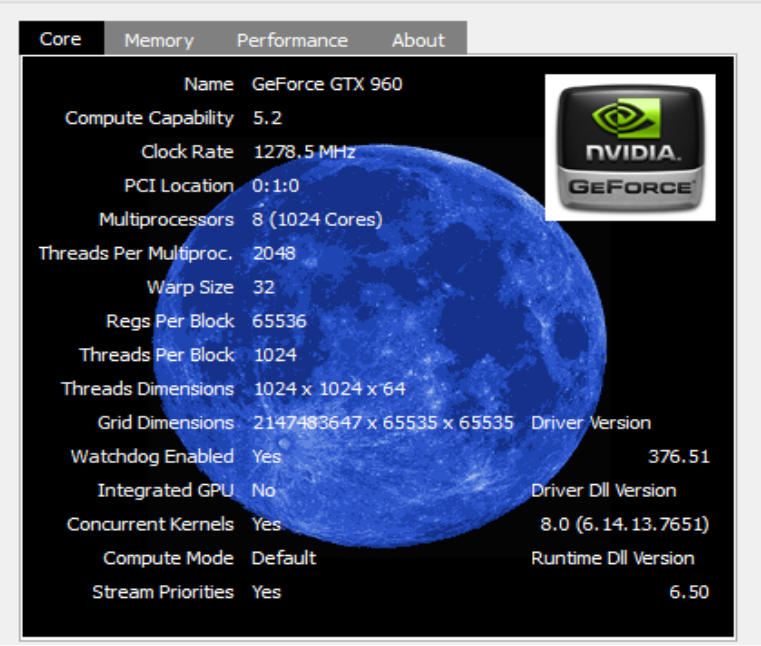

Figure 92. CUDA-enabled GPU. Hardware information is provided by CUDA-Z [266]

7.2.2 Google Tensorflow, Python

The chosen Machine Learning computational framework for his thesis is the open-source *Google Tensorflow* [267], which is built on top of the CUDA API for Python and reuses common math and scientific libraries like *numpy* and *scipy*. Python procedure calls to Tensorflow methods declare internal objects like sessions and *tensors*. A tensor is a computational node that can handle various computations or supporting functions for an image recognition model. The programming style of Tensorflow can be considered declarative, used to define a computational graph of tensors which runs in a parallel-enabled session. A deep neural network is declared as a computational graph with an input node and an output node. The execution of the graph is handled by a session that receives the input and requests the result of the output node executing the computations of all the prerequisite nodes. Tensor nodes can have various roles in the graph:

- value tensors, constants or variables, vectors, matrices, 3D tensor, 4D tensors.
- math functions, series, aggregate functions
- conditional branching
- convolution and pooling operations
- activation functions
- gradient descent optimization methods
- slicing and concatenation of tensors
- FIFO queues
- image processing operations
- color space transformations

A variety of tensors can be found in the extensive online documentation [268] and many answers and solutions by a growing user community [269]. Below is the simple math function $a = b + c \cdot d$ implemented in Tensorflow, with a constant b and the variables c, d for single precision floating point arithmetic:

```
oGraph = tf.Graph()
with oGraph.as_default():

   # Declaration
   tBeta   = tf.constant(5.0, tf.float32)
   tGamma  = tf.Variable(3.0, tf.float32)
   tDelta  = tf.Variable(2.0, tf.float32)
    tAlpha  = tf.add(tBeta, tf.multiply(tGamma, oGraph, "mul"), name="add")
```

A new computational graph `oGraph` is created and 5 tensor nodes are added in its context, a constant tensor `tBeta`, variable tensors `tGamma` and `tDelta`, multiplication tensor named *`"mul"`* and addition tensor named *`"add"`* as the output node that calculates `tAlpha`. In order to get the result of this calculation, the tensor runs in aparallel session, which uses the constant and the current values of the variables.

```
# Execution
with tf.Session().as_default() as oSession:
    oSession.run(tf.global_variables_initializer())
    a = oSession.run(tAlpha)
    print(a)
```

The `oSession` executes the underlying graph which calculates to `tAlpha`, and returns the value 11 to the Python variable `a` which is printed in the stdout. In the above example the GPU processing is done inside the session and the main-memory to GPU-memory transfers are completely hidden. The tensor objects reside in the Python main memory space, while the constant and the variable tensors load their values into the GPU memory. The session runs the computation in CUDA intermediate code on the GPU and the result is returned to Python variables. Google Tensorflow supports multi-GPU and distributed implementations that can be hosted on a computation cloud.

Power features of Tensorflow are the monitoring, logging and visualization capabilities through *Tensorboard*, a web server that provides a UX to Tensorflow log data. The current version offers capabilities for monitoring the training process through graphs, embedding images and audio into the log data, plotting distributions and histograms. The most helpful feature in the design of a model is the auto-generated visualization of the computational graph. Each tensor context can expand to show its child nodes down to the most detailed level.

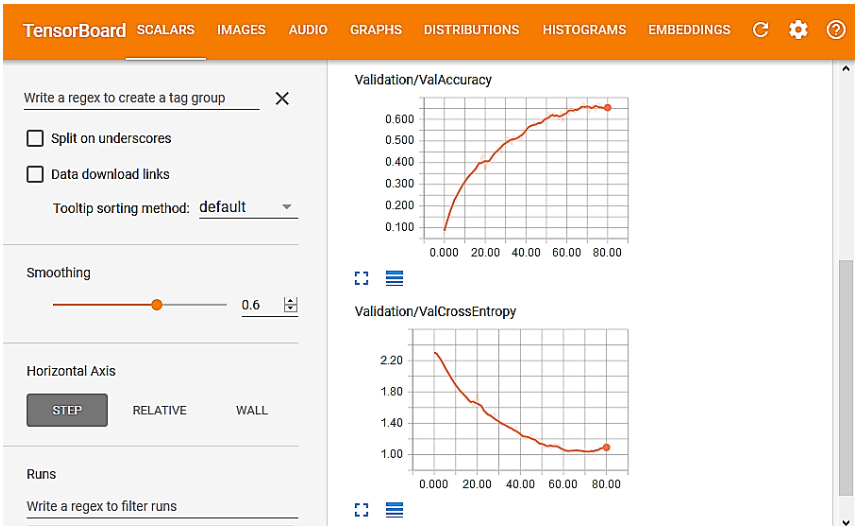

Figure 93. Tensorboard graphs for validation accuracy and validation categorical cross entropy that are updated during the training process.

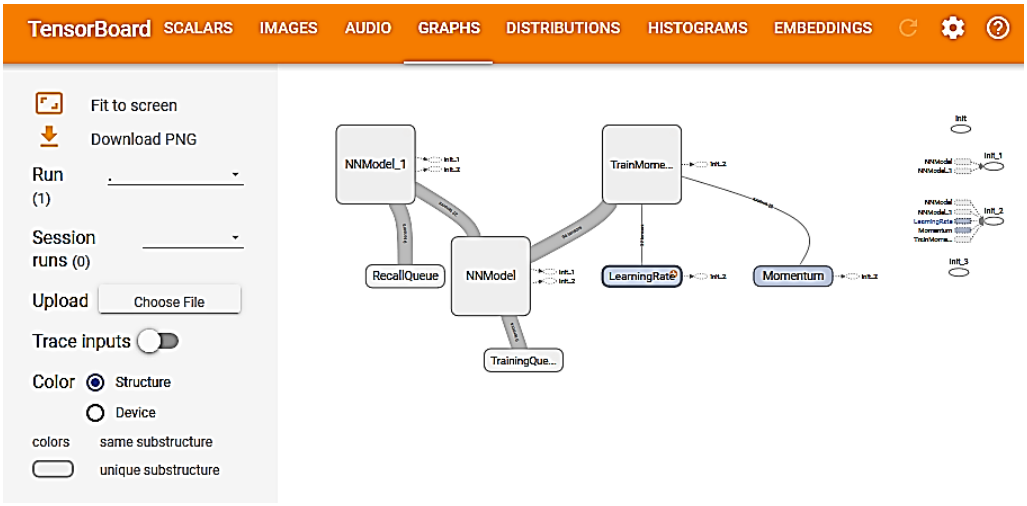

Figure 94. Top level graph of the Tensorflow implementation that has been used for traing BioCNNs

At the start of the thesis' development process the pre-release version 0.10.0 was available for Ubuntu 14.04 Linux and gradually made available for Windows platforms. At the time of writing the Tensorflow version 1.0.1 was used on Windows 64bit and Python 3.5. The migration from 0.x to 1.01 required minor refactoring due to finalized API changes. The complete Python-based machine learning environment of this thesis, including all other libraries and custom developed code is portable to both Windows and Linux platforms.

7.2.3 SciKit-Learn, MatPlotLib

The *SciKit-Learn* (sklearn) version 0.18.1 machine learning library for Python [270], provides additional functionality that is not currently supported by Tensorflow. It provides an implementation of the mini-batch k-Means algorithm in the `sklearn.cluster` package, which was reused as part of the Rooting k-Means algorithm. The PCA method from `sklearn.decomposition` was also used for preprocessing the feature descriptors before clustering. Additionally sklearn provides metrics in `sklearn.metrics`, like weighted Average Precision / Recall / F1-score and Silhouette.

The *MatPlotLib* (pyplot) version 2.0.2 python library [271] is used for rendering graphs and displaying grids of RGB images. Additionally it is used for 2D and 3D visualizations of the convolutional layer receptive field maps and the output activation maps.

7.2.4 Terrier IR Search Engine

The search engine used in retrieval experiments is the Terrier IR 4.2 open-source platform [272] that is created and maintained by the University of Glasgow. It is a cross-platform IR system written in Java that offers increased retrieval performance and a variety of options for weighting schemes, notably TF-IDF, BM25, DLH, IFB2 and many more that are found in its online documentation [273]. Fielded search and query expansion are supported by the TerrierIR engine, which is accessible through command line or a Java API. Furthermore it can execute a batch of retrieval queries that are assembled into a single structured text file, which requires to be in a TREC-compatible structured file format.

```
<doc>
<docno>003001</docno>
  <text>
    <classID>0</classID>
    <className>accordion</className>
    <GVFL>
      visualword visualword visualword
      visualword visualword visualword
      visualword visualword visualword
    </GVFL>
    <MVFL>
      visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword
    </MVFL>
    <LVFL>
      visualword visualword visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword visualword visualword
      visualword visualword visualword visualword visualword visualword visualword
    </LVFL>
  </text>
</doc>
```

Figure 95: TREC style format used for the Visual Booklets (VBOOK) in BioCNN-CBIR. Each field GVFL, MVFL, LVFL is a different Visual Document (VDOC) written in visual words of its respective VFL vocabulary. In this example the GVFL was generated on a $[3 \times 3]$ output map, MVFL $[5 \times 5]$ and LVFL $[7 \times 7]$.

7.2.5 TALOS Framework

Various useful methodologies of Software Engineering were used for the implementation needs of the thesis. A complex machine learning model like a deep CNN needs levels of abstraction, to hide the complexity of its inner specifics, gain in reproducibility, maintainability and implementation speed. Using the principles of *Object Oriented Analysis and Design* (OOAD) an object-oriented framework was created in Python for this thesis. It abstracts the details of the tensor-based neural network implementation and simplifies the declaration and use of CNN architectures, for the target of reducing need time for R&D. The *Tensor Abstraction Layer ObjectS* or *TALOS* framework contains reusable machine learning components, dataset operations, utilities for organization and evaluation of experiments. It implements various research artifacts like saved models, training stats, evaluation results and comparison graphs. Neural networks are declared as classes and convolutional modules that may contain multiple convolutional and pooling layers, are added to the computational graph with a single method call.

Extra features that increase the utilization of hardware resources are the multi-threaded data queues for training and validation. The samples are fed into queues that belong to the Tensorflow session thread by the queue feeder threads and not the main UI thread. This increases the GPU utilization and accelerates training by having a constant supply of samples in queue for the Tensorflow thread.

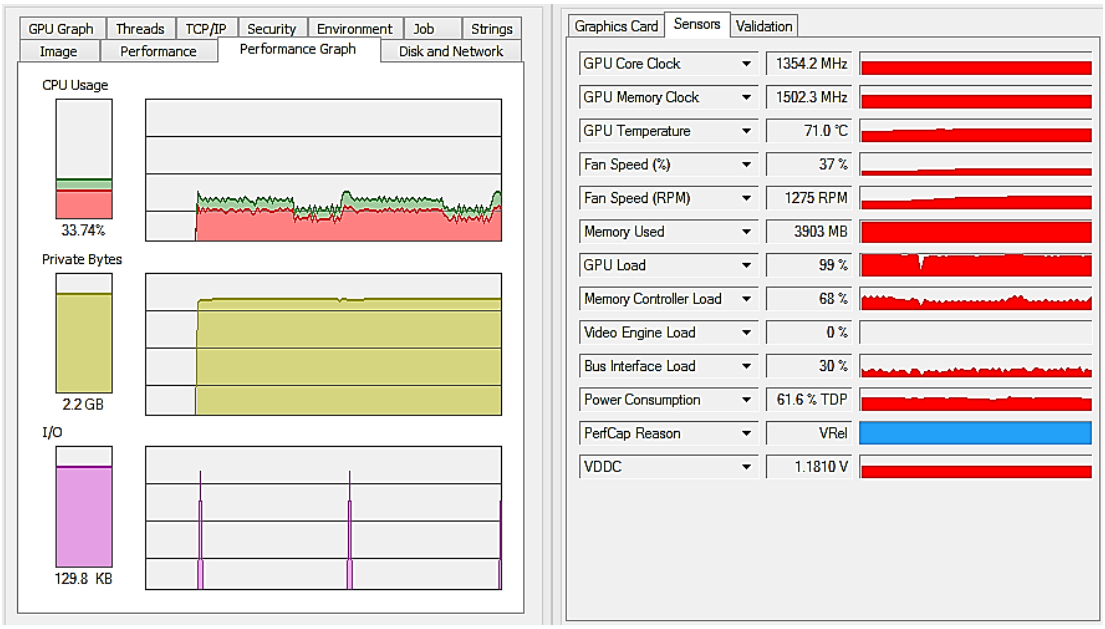

Figure 96. Utilization of CPU, main memory, I/O, GPU, memory controller of graphics adapter, during the training of a CNN network in TALOS. Tools that have been used to visualize the computational resources are: SysInternals Process Explorer [274] and GPU-Z [275].

For having a responsive UX a “heartbeat” message displays training error and accuracy providing at the same time a monitor of the training process. The frequency of the “heartbeat” in epoch steps is automatically calculated from TALOS given a desired period of seconds, by benchmarking the total time in seconds needed to train the first epoch.

All BioCNN models are implemented using TALOS, but their parameters can be loaded into custom Tensorflow implementations, provided that the names of contexts and tensor nodes in the computational graph are the same. Furthermore TALOS has reusable objects for the CIFAR10 and Caltech101 datasets that support folds, subfold shuffling and queuing. Future plans include making TALOS available on GitHub, as an open-source tool for practical Machine Learning research.

7.2.6 Data Memory Tier

The Data Memory Tier (DMT) tier is placed on a file system that has binary files and structured text files in both XML and JSON formats. The TALOs and Python objects and are *serialized* to files [276], known as *pickle* files. The images are originally stored in JPG 24-bit color format or PNG 24-bit color with alpha channel. They are loaded into Python lists, bundled into dictionaries and exported to pickle object serialization format. These files are loaded faster in Python and can be considered as caches for images. In a distributed implementation of the DMT, image caches and other data could be stored in databases either RDBMS or OODBMS [277] .

Some downloaded image datasets like CIFAR10 are in older versions of the pickle format. They are converted once from Python 3.5 and stored in the newest supported format. The learned states of CNNs are stored using the Tensorflow checkpoint binary format [268] that saves the values of variable tensors in the computational graph. Support objects that exist in the TALOS framework like training stats are also serialized as pickle files.

8 Conclusion

8.1 Summary

8.1.1 Introducing the BioCNN Deep Neural Network

Deep Neural Networks have created new prospects for solving image recognition problems with Convolutional Neural Networks as a suitable model to capture visual information in images. Nevertheless there is still a great distance to cover in order to reach the destination of using them as universally applicable visual feature extractors. The various architectures and methods of the current state-of-the-art are connected with increased computational costs. Moreover their superior accuracy is achieved in controlled conditions with a predefined number of categories and hundreds of samples for each one of them. The experiments in this thesis have confirmed that when training a state-of-the-art CNN classifier using a small number of samples per class, it fails to generalize.

In order to use CNNs in real-world applications, like robot agents and medical imaging equipment, they should become scalable allowing the addition of new samples of unknown categories, without the need of fine-tuning the entire model. Also a small number of samples for each category used to train the network should be sufficient for accurate and precise prediction. New architectures and methods are needed which will be not overfit on the ground truth image set. A decreased overfitting during supervised gradient descent training, can be an indicator of a more generic CNN feature extractor. To address these problems an alternative direction for research is proposed by this Master's thesis, which follows the paradigm of Biomimetics.

It introduces the BioCNN as a new kind of Deep Neural Network inspired by the human biology and color perception, which extends the CNN with the power of handcrafted feature extraction. The pre-defined weights in the convolutional kernels are engineered with models that imitate the functionality of the biological cells in the retina and the visual cortex.

The reference architecture of a BioCNN has a stem with 7 stages of feature extraction: 1) The Photoreceptors processing layer (PRL) 2) The Horizontal Cells convolutional layer (HCL) 3) The Bipolar Cells Gaussian convolutional layer (BPL) 4) The Amacrine Cells NiN convolutional layer (ACL) 5) The Ganglion Cells convolutional module (GCM) 6) The V1 cluster of convolutional layers with pre-defined Gabor filters 7) the V1-LGN cluster of learnable convolutional layer with kernels initialized with Gabor filters and fine-tuned through training. The PRL processing transforms the RGB color space into the two-dimensional color space of chroma C and luma Y, which represent the respective cone and rod responses to light. These are processed by the closely coupled HCL and BPL, which are stratified as the outer plexiform layer found in a biological retina. The BPL generates peek responses for 6 colors red, green, blue, yellow, white and black modeling the sensitivities of cones and rods to different wavelengths and levels of illumination respectively. The ACL implements the opponent color process of human color vision and the GCM detects blobs in a center-surround opponent color antagonism. The V1 extracts lines of different directions and lengths that are supplemented with learned features. The BioCNN stems creates a high-dimensional activation output, combining the extracted features from multiple layers of neural processing.

Experiments on the CIFAR10 datasets have indicated that BioCNNs could potential reach state-of-the-art performances using a fraction of the parameters when compared with the ILSVRC2013 winning architecture of Zeiler and Fergus. BioCNN has greater accuracy, precision and F1-score than ZFNet in a 10-fold cross validation and additionally a smaller average absolute deviation on all metrics. Classification in Caltech101 reached competitive but not optimal levels of accuracy indicating that more research effort is needed to find the best values for architectural and functional hyperparameters. Training the ZFNet on the Caltech101-2017 random subset with Throttled Gradient Descent (TGD) resulted in a better accuracy than the one reported in the original paper, even though no data augmentation was used. This indicates that TGD might be a useful variable learning rate technique on problems with high overfitting. Early stopping that takes into account a bias profit and expressed by the Overfit Margin (OFM) ratio is also a vital aspect of the learning process.

8.1.2 Content-Based Image Retrieval with the Visual Features Language

This work has not sufficiently addressed the problem of discovering well performing CNN-based features for increased CBIR performance. The experimental results showed inadequate performance of the overall model. Nevertheless the evaluation system used for experiments could be considered a systematic approach to CBIR research. Moreover the complexity is fully explored and the factors that contribute to the CNN-based CBIR

performance are identified. This can be an appropriate starting point for a more targeted research that will aim in increasing the retrieval metrics.

A contribution may be considered the Visual Features Language that generalizes the Bag-of-Visual-Words model and extends it with characteristics of a natural language like syllables and stopwords. The creation of fabricated textual words may be proven beneficial, as the image retrieval problem is abstracted into the more comprehensible text retrieval problem. Existing image collections can be translated into VFL datasets and reused as text collections for IR research, without the need of the CNN and the clustering model.

The interesting observation that was made through the CBIR experiments, was the appearance of Zipf's law in visual features, which needs to be further studied in order to support a theory: Quanta of visual information inside images follow the Zipfian distribution. An evaluation of different clustering methods for the creation of the VFL could not fit in the time restrictions of this Master's thesis. The Rooting k-Means algorithm is fast, simple for the current needs and suitable for low computational resources. Its quality needs to be evaluated individually in different problem domains and compared with the constant branching factor of Hierarchical k-Means and the binary tree of Bisecting k-Means.

Creating a content-based image retrieval task in an annual competition that will a large scale image collection could bring advances in image retrieval, like the advances in image classification which have been achieved through ILSVRC.

8.1.3 Software Implementation of a Web 4.0 Intelligence Tier

The BioCNN-CBIR image retrieval pipeline is a practical software implementation which illustrates a Machine Learning Intelligence Tier in a multi-tiered web 4.0 software application. The speed of translating an image from 24-bit color RGB format to VFL format is proper to create a responsive web user experience. This has been implemented as the prototype of an image search engine, with the codename "Horasis", which has in its core the BioCNN-CBIR pipeline.

A software side-product that comes out of this research effort is the TALOS framework. It is a software abstraction layer that hides the specifics of the underlying tensor-based implementation and currently supports the Google Tensorflow framework. It provides the ability to focus on the higher-level research needs and can speed it up through reusability and simplicity. The training of CNNs in reasonable amounts of time was made possible by the multi-threaded queues in conjunction with CUDA hardware acceleration. Software engineering, breakdown of neural networks in graphs of computational nodes and accelerated computing can help Deep Learning research come up with complex solutions that were previously very hard to implement.

8.2 Future Perspectives

The further evaluation of the Bio-inspired Convolutional Neural Network could include training an architecture on the ImageNet dataset. A visualization of the learned convolutional kernels using the Zeiler and Fergus technique, in conjunction with monitoring of the gradients during the learning process could help optimize the architecture. The Throttled Gradient Descent could be compared to standard training methods and other variable learning rate schemes on a small subset of the large-scale image dataset. If proven beneficial it could be used to train a BioCNN in a future ImageNet competition.

The Artificial Retina and V1 could be tested as the stem of existing state-of-the-art networks like the Inception v4 and the ResNet-152, in order to discover modified versions of them with decreased memory requirements. Additional ideas for reducing complexity and increasing the recall speed of BioCNN, include trimming the amount of needed receptive field maps for the V1 and downsampling in the Horizontal Cell layer. The last could be tested with higher than 300×300 image resolutions and investigate whether the ARNN implements a kind of image compression with minimal loss. Additionally an alternative implementation of the color opponency mechanism in the ARNN could use the LAB color space (CIE L*a*b 1997). It has 3 components L for lightness, a with negative values for green and positive values for red and b for blue/yellow. The Bipolar Cell layer could be decoupled from the HCL and implement a different kind of non-linearity that will target -1 and opponent $+1$ input values. For the numerous hyperparameters used to generate the pre-defined kernels of BioCNN, a design space exploration can be performed using Evolutionary Algorithms. The optimum set of hyperparameters and the connectivity between the convolutional layers in the ARNN and APVC, could be discovered through the use of an objective function and recombination of the fittest values. Furthermore it could be examined whether the values of these hyperparameters could be learned through gradient descent optimization.

For improving Content-Based Image Retrieval based on CNN features, there is a vast search space that is shaped by many factors. Individual solutions may include an ensemble of high-accurate CNN classifiers trained on the same data, providing more than one feature descriptors for an image patch on the same spatial resolution. This increase in dimensionality could be handled by an Autoencoder that will embed dimensionality reduction directly in the deep architecture. The last convolutional module in this multi-functional Deep Neural Network would feed both the classifier and the Autoencoder. Investigating multiple depths and rooting factors for the Rooting k-Means clustering alone could lead to increased image retrieval performance. Testing on a variety of image collections from different problem domains could discover a universally applicable model as a solution for Intelligent Vision.

9 Bibliography

[1] N. Yadav, A. Yadav, and M. Kumar, *An introduction to neural network methods for differential equations*. Springer, 2015.

[2] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using CUDA," *J. Parallel Distrib. Comput.*, vol. 68, no. 10, pp. 1370–1380, 2008.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," DTIC Document, 1985.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] A. El Gamal and H. Eltoukhy, "CMOS image sensors," *IEEE Circuits Devices Mag.*, vol. 21, no. 3, pp. 6–20, 2005.

[7] K. Jack, *Video demystified: a handbook for the digital engineer*. Elsevier, 2011.

[8] HDMI Licensing, LLC, "High-Definition Multimedia Interface Specification Version 1.3a," 10-Nov-2006. [Online]. Available: http://www.microprocessor.org/HDMISpecification13a.pdf. [Accessed: 24-May-2017].

[9] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Apr. 2015.

[10] M. Trott, *The Mathematica guidebook for symbolics*. Springer Science & Business Media, 2007.

[11] D. Wade, *Symmetry: The ordering principle*. Bloomsbury Publishing USA, 2006.

[12] J. S. Stam, J. H. Bechtel, S. D. Reese, D. D. Tuttle, G. S. Bush, and H. C. Ockerse, "Light source detection and categorization system for automatic vehicle exterior light control and method of manufacturing," US6774988 B2, 10-Aug-2004.

[13] S. H. Khan, M. Bennamoun, F. Sohel, and R. Togneri, “Automatic Shadow Detection and Removal from a Single Image,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 3, pp. 431–446, Mar. 2016.

[14] S. Sarkar, V. Venugopalan, K. Reddy, M. Giering, J. Ryde, and N. Jaitly, “Occlusion Edge Detection in RGB-D Frames using Deep Convolutional Networks,” *ArXiv14127007 Cs*, Dec. 2014.

[15] Y. Zeng, F. Zhao, G. Wang, L. Zhang, and B. Xu, “Brain-Inspired Obstacle Detection Based on the Biological Visual Pathway,” in *Brain Informatics and Health*, 2016, pp. 355–364.

[16] “Johannes Eisele Stock Photos and Pictures | Getty Images.” [Online]. Available: http://www.gettyimages.com/photos/johannes-eisele. [Accessed: 09-Jun-2017].

[17] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[18] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, “A learning algorithm for Boltzmann machines,” *Cogn. Sci.*, vol. 9, no. 1, pp. 147–169, 1985.

[19] R. Salakhutdinov, “Learning Deep Generative Models,” *Annu. Rev. Stat. Its Appl.*, vol. 2, no. 1, pp. 361–385, Apr. 2015.

[20] S. Geman and D. Geman, “Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images,” *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 6, pp. 721–741, 1984.

[21] G. Casella and E. I. George, “Explaining the Gibbs sampler,” *Am. Stat.*, vol. 46, no. 3, pp. 167–174, 1992.

[22] S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, *Handbook of markov chain monte carlo*. Chapman and Hall/CRC, 2011.

[23] G. E. Hinton and R. S. Zemel, “Autoencoders, minimum description length and Helmholtz free energy,” in *Advances in neural information processing systems*, 1994, pp. 3–10.

[24] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, “Efficient learning of sparse representations with an energy-based model,” in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, 2006, pp. 1137–1144.

[25] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” *Adv. Neural Inf. Process. Syst.*, vol. 19, p. 153, 2007.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[27] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and Composing Robust Features with Denoising Autoencoders,” in *Proceedings of the 25th International Conference on Machine Learning*, New York, NY, USA, 2008, pp. 1096–1103.

[28] J. Xu *et al.*, “Stacked Sparse Autoencoder (SSAE) for Nuclei Detection on Breast Cancer Histopathology Images,” *IEEE Trans. Med. Imaging*, vol. 35, no. 1, pp. 119–130, Jan. 2016.

[29] Y. LeCun *et al.*, “Handwritten digit recognition with a back-propagation network, 1989,” in *Neural Information Processing Systems (NIPS)*, 1989.

[30] B. Osgood, “The Fourier transform and its applications,” *Lect. Notes EE*, vol. 261, p. 20, 2009.

[31] A. Deshpande, “A Beginner’s Guide To Understanding Convolutional Neural Networks,” 2016. [Online]. Available: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/. [Accessed: 26-May-2017].

[32] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *J. Physiol.*, vol. 160, no. 1, pp. 106–154, 1962.

[33] M. Lin, Q. Chen, and S. Yan, “Network In Network,” *ArXiv13124400 Cs*, Dec. 2013.

[34] C. Szegedy *et al.*, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *ArXiv Prepr. ArXiv151203385*, 2015.

[36] I. Goodfellow *et al.*, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[37] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” *ArXiv151106434 Cs*, Nov. 2015.

[38] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, “Stacked convolutional auto-encoders for hierarchical feature extraction,” *Artif. Neural Netw. Mach. Learn. 2011*, pp. 52–59, 2011.

[39] B. Yang, J. Yan, Z. Lei, and S. Z. Li, “Craft objects from images,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 6043–6051.

[40] W. Ouyang, X. Wang, C. Zhang, and X. Yang, “Factors in finetuning deep model for object detection with long-tail distribution,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 864–873.

[41] P. C. Bressloff and J. D. Cowan, “The functional geometry of local and horizontal connections in a model of v1,” *J. Physiol.-Paris*, vol. 97, no. 2, pp. 221–236, 2003.

[42] M. Carandini, “Area V1,” *Scholarpedia*, vol. 7, no. 7, p. 12105, Jul. 2012.

[43] I. Choi, S. Kim, M. S. Brown, and Y.-W. Tai, “A learning-based approach to reduce JPEG artifacts in image matting,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2880–2887.

[44] C.-Y. Lee, P. W. Gallagher, and Z. Tu, “Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree,” in *International conference on artificial intelligence and statistics*, 2016.

[45] J. Yeonan-Kim and M. Bertalmío, “Retinal Lateral Inhibition Provides the Biological Basis of Long-Range Spatial Induction,” *PLOS ONE*, vol. 11, no. 12, p. e0168963, Dec. 2016.

[46] S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. S. Kruthiventi, and R. V. Babu, “A Taxonomy of Deep Convolutional Neural Nets for Computer Vision,” *Front. Robot. AI*, vol. 2, Jan. 2016.

[47] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks.,” in *Aistats*, 2011, vol. 15, p. 275.

[48] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[49] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 6, no. 02, pp. 107–116, 1998.

[50] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs),” *ArXiv151107289 Cs*, Nov. 2015.

[51] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, “Incorporating second-order functional knowledge for better option pricing,” *Adv. Neural Inf. Process. Syst.*, pp. 472–478, 2001.

[52] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *Proc. ICML*, 2013, vol. 30.

[53] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[54] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical Evaluation of Rectified Activations in Convolutional Network,” *ArXiv150500853 Cs Stat*, May 2015.

[55] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *J. Big Data*, vol. 3, no. 1, p. 9, Dec. 2016.

[56] H.-C. Shin *et al.*, “Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning,” *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[57] J. S. Bridle, “Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition,” in *Neurocomputing*, Springer, Berlin, Heidelberg, 1990, pp. 227–236.

[58] R. A. Dunne and N. A. Campbell, “On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function,” in *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne, 181*, 1997, vol. 185.

[59] H. R. Thieme, “Differentiability of convolutions, integrated semigroups of bounded semi-variation, and the inhomogeneous Cauchy problem,” *J. Evol. Equ.*, vol. 8, no. 2, pp. 283–305, May 2008.

[60] S. Mannor, D. Peleg, and R. Rubinstein, “The Cross Entropy Method for Classification,” in *Proceedings of the 22Nd International Conference on Machine Learning*, New York, NY, USA, 2005, pp. 561–568.

[61] P. Golik, P. Doetsch, and H. Ney, “Cross-entropy vs. squared error training: a theoretical and experimental comparison.,” in *Interspeech*, 2013, pp. 1756–1760.

[62] Y. Nesterov, “A method for unconstrained convex minimization problem with the rate of convergence O (1/k2),” in *Doklady an SSSR*, 1983, vol. 269, pp. 543–547.

[63] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT’2010*, Springer, 2010, pp. 177–186.

[64] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *J. Mach. Learn. Res.*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[65] M. D. Zeiler, “ADADELTA: an adaptive learning rate method,” *ArXiv Prepr. ArXiv12125701*, 2012.

[66] S. Ruder, “An overview of gradient descent optimization algorithms,” *ArXiv160904747 Cs*, Sep. 2016.

[67] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *ArXiv14126980 Cs*, Dec. 2014.

[68] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” M.S. thesis, University of Toronto, 2009.

[69] A. Jeffrey, D. Zwillinger, I. S. Gradshteyn, and I. M. Ryzhik, Eds., “15 - Norms,” in *Table of Integrals, Series, and Products (Seventh Edition)*, Boston: Academic Press, 2007, pp. 1081–1091.

[70] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *ArXiv Prepr. ArXiv12070580*, 2012.

[71] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, “Regularization of neural networks using dropconnect,” in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1058–1066.

[72] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, “Disturblabel: Regularizing cnn on the loss layer,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4753–4762.

[73] A. Livnat, C. Papadimitriou, J. Dushoff, and M. W. Feldman, “A mixability theory for the role of sex in evolution,” *Proc. Natl. Acad. Sci.*, vol. 105, no. 50, pp. 19803–19808, Dec. 2008.

[74] K. I. Diamantaras and S. Y. Kung, *Principal Component Neural Networks: Theory and Applications*. New York, NY, USA: John Wiley & Sons, Inc., 1996.

[75] A. Ford and A. Roberts, “Colour space conversions,” *Westminst. Univ. Lond.*, vol. 1998, pp. 1–31, 1998.

[76] N. Pinto, D. D. Cox, and J. J. DiCarlo, “Why is real-world visual object recognition hard?,” *PLoS Comput Biol*, vol. 4, no. 1, p. e27, 2008.

[77] B. M. H. Romeny, “The Gaussian kernel,” in *Front-End Vision and Multi-Scale Image Analysis*, pp. 37–51.

[78] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks.,” *ICML 3*, vol. 28, pp. 1310–1318, 2013.

[79] K. Jarrett, K. Kavukcuoglu, and Y. LeCun, “What is the best multi-stage architecture for object recognition?,” in *Computer Vision, 2009 IEEE 12th International Conference on*, 2009, pp. 2146–2153.

[80] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, pp. 818–833.

[81] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *ArXiv Prepr. ArXiv150203167*, 2015.

[82] D. Mishkin and J. Matas, “All you need is a good init,” *ArXiv151106422 Cs*, Nov. 2015.

[83] S. Ioffe, “Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models,” *ArXiv Prepr. ArXiv170203275*, 2017.

[84] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks.,” in *Aistats*, 2010, vol. 9, pp. 249–256.

[85] D. Sussillo and L. F. Abbott, “Random Walk Initialization for Training Very Deep Feedforward Networks,” *ArXiv14126558 Cs Stat*, Dec. 2014.

[86] A. M. Saxe, J. L. McClelland, and S. Ganguli, “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks,” *ArXiv13126120 Cond-Mat Q-Bio Stat*, Dec. 2013.

[87] “BatchNorm numerical instability · Issue #3963 · BVLC/caffe,” *GitHub*. [Online]. Available: https://github.com/BVLC/caffe/issues/3963. [Accessed: 14-Jun-2017].

[88] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes Challenge: A Retrospective,” *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jun. 2014.

[89] “ILSVRC2016.” [Online]. Available: http://image-net.org/challenges/LSVRC/2016/. [Accessed: 16-May-2017].

[90] L. K. Hansen and P. Salamon, “Neural network ensembles,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, 1990.

[91] M. D. Zeiler, G. W. Taylor, and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 2018–2025.

[92] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,” *ArXiv13126229 Cs*, Dec. 2013.

[93] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *ArXiv Prepr. ArXiv14091556*, 2014.

[94] “Heuritech Le Blog.” [Online]. Available: https://blog.heuritech.com/. [Accessed: 14-Jun-2017].

[95] K. J. Lang, “Learning to tell two spirals apart,” in *Proc. of 1988 Connectionist Models Summer School*, 1988.

[96] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European Conference on Computer Vision*, 2016, pp. 630–645.

[97] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[98] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” *ArXiv160207261 Cs*, Feb. 2016.

[99] S. Zagoruyko and N. Komodakis, “Wide Residual Networks,” *ArXiv160507146 Cs*, May 2016.

[100] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[101] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.

[102] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[103] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, no. Feb, pp. 281–305, 2012.

[104] S. C. Smithson, G. Yang, W. J. Gross, and B. H. Meyer, “Neural networks designing neural networks: multi-objective hyper-parameter optimization,” in *Proceedings of the 35th International Conference on Computer-Aided Design*, 2016, p. 104.

[105] B. Baker, O. Gupta, N. Naik, and R. Raskar, “Designing Neural Network Architectures using Reinforcement Learning,” *ArXiv Prepr. ArXiv161102167*, 2016.

[106] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[107] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *ArXiv Prepr. ArXiv151000149*, 2015.

[108] R. Logeswaran and C. Eswaran, “Neural network based lossless coding schemes for telemetry data,” in *Geoscience and Remote Sensing Symposium, 1999. IGARSS’99 Proceedings. IEEE 1999 International*, 1999, vol. 4, pp. 2057–2059.

[109] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size,” *ArXiv160207360 Cs*, Feb. 2016.

[110] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, 2006.

[111] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum, “One shot learning of simple visual concepts.,” in *CogSci*, 2011, vol. 172, p. 2.

[112] G. Koch, “Siamese neural networks for one-shot image recognition,” University of Toronto, 2015.

[113] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra, “Matching networks for one shot learning,” in *Advances in Neural Information Processing Systems*, 2016, pp. 3630–3638.

[114] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst, “Faceted metadata for image search and browsing,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2003, pp. 401–408.

[115] J. Ashley, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic, “The query by image content (QBIC) system,” in *ACM SIGMOD Record*, 1995, vol. 24, p. 475.

[116] A. Pentland, R. W. Picard, and S. Sclaroff, “Photobook: Content-based manipulation of image databases,” *Int. J. Comput. Vis.*, vol. 18, no. 3, pp. 233–254, 1996.

[117] R. Datta, J. Li, and J. Z. Wang, “Content-based Image Retrieval: Approaches and Trends of the New Age,” in *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*, New York, NY, USA, 2005, pp. 253–262.

[118] Y. Rui, T. S. Huang, and S.-F. Chang, “Image Retrieval: Current Techniques, Promising Directions, and Open Issues,” *J. Vis. Commun. Image Represent.*, vol. 10, no. 1, pp. 39–62, Mar. 1999.

[119] T. Chang and C.-C. Kuo, “Texture analysis and classification with tree-structured wavelet transform,” *IEEE Trans. Image Process.*, vol. 2, no. 4, pp. 429–441, 1993.

[120] T. Deselaers, D. Keysers, and H. Ney, “Features for image retrieval: an experimental comparison,” *Inf. Retr.*, vol. 11, no. 2, pp. 77–107, Dec. 2007.

[121] H. Tamura, S. Mori, and T. Yamawaki, “Textural features corresponding to visual perception,” *IEEE Trans. Syst. Man Cybern.*, vol. 8, no. 6, pp. 460–473, 1978.

[122] “ISO/IEC 15938-1:2002 - Information technology -- Multimedia content description interface -- Part 1: Systems.” [Online]. Available: https://www.iso.org/standard/34228.html. [Accessed: 17-May-2017].

[123] S. E. Grigorescu, N. Petkov, and P. Kruizinga, “Comparison of texture features based on Gabor filters,” *IEEE Trans. Image Process.*, vol. 11, no. 10, pp. 1160–1167, 2002.

[124] W. Li, K. Mao, H. Zhang, and T. Chai, “Designing compact Gabor filter banks for efficient texture feature extraction,” in *2010 11th International Conference on Control Automation Robotics Vision*, 2010, pp. 1193–1197.

[125] D. G. Lowe, “Object recognition from local scale-invariant features,” in *The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999*, 1999, vol. 2, pp. 1150–1157 vol.2.

[126] M. Mitra and B. B. Chaudhuri, “Information retrieval from documents: A survey,” *Inf. Retr.*, vol. 2, no. 2–3, pp. 141–163, 2000.

[127] A. I. Saichev, Y. Malevergne, and D. Sornette, *Theory of Zipf’s law and beyond*, vol. 632. Springer Science & Business Media, 2009.

[128] B. Croft, D. Metzler, and T. Strohman, *Search Engines: Information Retrieval in Practice*, 1 edition. Boston: Pearson, 2009.

[129] G. Sidorov, A. Gelbukh, H. Gómez-Adorno, and D. Pinto, “Soft similarity and soft cosine measure: Similarity of features in vector space model,” *Comput. Sist.*, vol. 18, no. 3, pp. 491–504, 2014.

[130] Sivic and Zisserman, “Video Google: a text retrieval approach to object matching in videos,” 2003, pp. 1470–1477 vol.2.

[131] S. O’Hara and B. A. Draper, “Introduction to the Bag of Features Paradigm for Image Classification and Retrieval,” *ArXiv11013354 Cs*, Jan. 2011.

[132] R. Vieux, J. Benois-Pineau, and J.-P. Domenger, “Content Based Image Retrieval Using Bag-Of-Regions,” in *Advances in Multimedia Modeling*, K. Schoeffmann, B. Merialdo, A. G. Hauptmann, C.-W. Ngo, Y. Andreopoulos, and C. Breiteneder, Eds. Springer Berlin Heidelberg, 2012, pp. 507–517.

[133] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[134] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 3304–3311.

[135] R. Kiros, R. Salakhutdinov, and R. Zemel, “Multimodal neural language models,” in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 595–603.

[136] B. M. H. Romeny, “Gaussian derivatives,” in *Front-End Vision and Multi-Scale Image Analysis*, pp. 53–69.

[137] D. Marr and E. Hildreth, “Theory of edge detection,” *Proc. R. Soc. Lond. B Biol. Sci.*, vol. 207, no. 1167, pp. 187–217, 1980.

[138] L. G. Shapiro and G. C. Stockman, *Computer Vision*, 1 edition. Upper Saddle River, NJ: Pearson, 2001.

[139] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[140] K. Grauman and B. Leibe, “Visual Object Recognition,” *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 5, no. 2, pp. 1–181, Apr. 2011.

[141] C. Harris and M. Stephens, “A combined corner and edge detector in Alvey vision conference. 1988,” *Manch. UK*.

[142] D. Weng, Y. Wang, M. Gong, D. Tao, H. Wei, and D. Huang, “DERF: Distinctive Efficient Robust Features From the Biological Modeling of the P Ganglion Cells,” *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2287–2302, Aug. 2015.

[143] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-Up Robust Features (SURF),” *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[144] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image Vis. Comput.*, vol. 22, no. 10, pp. 761–767, 2004.

[145] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, 2005, vol. 1, pp. 886–893 vol. 1.

[146] E. Tola, V. Lepetit, and P. Fua, “Daisy: An efficient dense descriptor applied to wide-baseline stereo,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, 2010.

[147] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005.

[148] P. Azad, T. Asfour, and R. Dillmann, “Combining harris interest points and the sift descriptor for fast scale-invariant object recognition,” in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, pp. 4275–4280.

[149] H. Si, “TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator,” *ACM Trans Math Softw*, vol. 41, no. 2, p. 11:1–11:36, Feb. 2015.

[150] X. Wu *et al.*, “Top 10 algorithms in data mining,” *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.

[151] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,” in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, 2006, vol. 2, pp. 2161–2168.

[152] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Object retrieval with large vocabularies and fast spatial matching,” in *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, 2007, pp. 1–8.

[153] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.,” in *Kdd*, 1996, vol. 96, pp. 226–231.

[154] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, “Support vector clustering,” *J. Mach. Learn. Res.*, vol. 2, no. Dec, pp. 125–137, 2001.

[155] S. M. Van Dongen, “Graph clustering by flow simulation,” 2001.

[156] T. Kinnunen, J.-K. Kamarainen, L. Lensu, and H. Kälviäinen, “Bag-of-features codebook generation by self-organisation,” in *International Workshop on Self-Organizing Maps*, 2009, pp. 124–132.

[157] V. Viitaniemi and J. Laaksonen, “Experiments on selection of codebooks for local image feature histograms,” *Vis. Inf. Syst. Web-Based Vis. Inf. Search Manag.*, pp. 126–137, 2008.

[158] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Lost in quantization: Improving particular object retrieval in large scale image databases,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1–8.

[159] H. Jégou, M. Douze, and C. Schmid, “Improving bag-of-features for large scale image search,” *Int. J. Comput. Vis.*, vol. 87, no. 3, pp. 316–336, 2010.

[160] Y. Avrithis and Y. Kalantidis, “Approximate gaussian mixtures for large scale vocabularies,” *Comput. Vision–ECCV 2012*, pp. 15–28, 2012.

[161] A. Mikulik, M. Perdoch, O. Chum, and J. Matas, “Learning vocabularies over a fine quantization,” *Int. J. Comput. Vis.*, vol. 103, no. 1, pp. 163–175, 2013.

[162] D. L. Donoho, “Aide-memoire. high-dimensional data analysis: The curses and blessings of dimensionality,” *Am. Math Soc. Lect.-Math Chall. 21st Century*, 2000.

[163] J. P. Cunningham and Z. Ghahramani, “Linear dimensionality reduction: Survey, insights, and generalizations,” *J. Mach. Learn. Res.*, vol. 16, pp. 2859–2900, 2015.

[164] Y. Ke and R. Sukthankar, “PCA-SIFT: a more distinctive representation for local image descriptors,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 2004, vol. 2, p. II-506-II-513 Vol.2.

[165] M. Steinbach, G. Karypis, and V. Kumar, “A comparison of document clustering techniques,” in *KDD workshop on text mining*, 2000, vol. 400, pp. 525–526.

[166] L. Zheng, Y. Yang, and Q. Tian, “SIFT Meets CNN: A Decade Survey of Instance Retrieval,” *ArXiv160801807 Cs*, Aug. 2016.

[167] L. Piras and G. Giacinto, “Open issues on codebook generation in image classification tasks,” in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, 2014, pp. 328–342.

[168] A. Torralba, R. Fergus, and Y. Weiss, “Small codes and large image databases for recognition,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1–8.

[169] E. Hörster and R. Lienhart, “Deep Networks for Image Retrieval on Large-scale Databases,” in *Proceedings of the 16th ACM International Conference on Multimedia*, New York, NY, USA, 2008, pp. 643–646.

[170] A. Hasasneh, “Robot semantic place recognition based on deep belief networks and a direct use of tiny images,” Ph.D. dissertation, Université Paris Sud-Paris XI, 2012.

[171] A. Krizhevsky and G. E. Hinton, “Using very deep autoencoders for content-based image retrieval.,” in *ESANN*, 2011.

[172] R. Salakhutdinov and G. Hinton, “Semantic hashing,” *Int. J. Approx. Reason.*, vol. 50, no. 7, pp. 969–978, 2009.

[173] L. Zheng, Y. Zhao, S. Wang, J. Wang, and Q. Tian, “Good practice in CNN feature transfer,” *ArXiv Prepr. ArXiv160400133*, 2016.

[174] J. Donahue *et al.*, “Decaf: A deep convolutional activation feature for generic visual recognition,” *ArXiv Prepr. ArXiv13101531*, 2013.

[175] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Learning hierarchical features for scene labeling,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[176] L. Zheng, S. Wang, L. Tian, F. He, Z. Liu, and Q. Tian, “Query-adaptive late fusion for image search and person re-identification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1741–1750.

[177] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in *European Conference on Computer Vision*, 2014, pp. 346–361.

[178] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories,” in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, Washington, DC, USA, 2006, pp. 2169–2178.

[179] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, “Multi-scale orderless pooling of deep convolutional activation features,” in *European conference on computer vision*, 2014, pp. 392–407.

[180] A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, “Visual instance retrieval with deep convolutional networks,” *ArXiv14126574 Cs*, Dec. 2014.

[181] E. Mohedano, K. McGuinness, N. E. O’Connor, A. Salvador, F. Marqués, and X. Giró-i-Nieto, “Bags of local convolutional features for scalable instance search,” in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, 2016, pp. 327–331.

[182] F. Radenović, G. Tolias, and O. Chum, "CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples," in *European Conference on Computer Vision*, 2016, pp. 3–20.

[183] J. Dowling, "Retina," *Scholarpedia*, vol. 2, no. 12, p. 3487, Dec. 2007.

[184] D. H. Edwards, "Neuroscience. Third Edition. Edited by Dale Purves, George J Augustine, David Fitzpatrick, William C Hall, Anthony-Samuel LaMantia, James O McNamara , and S Mark Williams.," *Q. Rev. Biol.*, vol. 81, no. 1, pp. 86–87, Mar. 2006.

[185] A. Roorda and D. R. Williams, "The arrangement of the three cone classes in the living human eye," *Nature*, vol. 397, no. 6719, p. 520, 1999.

[186] A. Stockman and L. T. Sharpe, "Human cone spectral sensitivities: a progress report," *Vision Res.*, vol. 38, no. 21, pp. 3193–3206, Nov. 1998.

[187] J. E. Dowling, *The retina: an approachable part of the brain*. Harvard University Press, 1987.

[188] "NWS JetStream - The Color of Clouds." [Online]. Available: http://www.srh.noaa.gov/jetstream/clouds/color.html. [Accessed: 31-May-2017].

[189] G. Osterberg, "Topography of the layer of rods and cones in the human retina," *Acta Ophthalmol*, vol. 13, pp. 6–97, 1935.

[190] G. D. Field *et al.*, "Functional connectivity in the retina at the resolution of photoreceptors," *Nature*, vol. 467, no. 7316, pp. 673–677, Oct. 2010.

[191] R. H. Masland, "The Neuronal Organization of the Retina," *Neuron*, vol. 76, no. 2, pp. 266–280, Oct. 2012.

[192] W. N. Grimes, J. Zhang, C. W. Graydon, B. Kachar, and J. S. Diamond, "Retinal parallel processors: more than 100 independent microcircuits operate within a single interneuron," *Neuron*, vol. 65, no. 6, pp. 873–885, 2010.

[193] T. G. Weyand, "The multifunctional lateral geniculate nucleus," *Rev. Neurosci.*, vol. 27, no. 2, pp. 135–157, Feb. 2016.

[194] N. J. Priebe, "Mechanisms of Orientation Selectivity in the Primary Visual Cortex," *Annu. Rev. Vis. Sci.*, vol. 2, pp. 85–107, Oct. 2016.

[195] D. Y. Ts'o, M. Zarella, and G. Burkitt, "Whither the hypercolumn?," *J. Physiol.*, vol. 587, no. Pt 12, pp. 2791–2805, Jun. 2009.

[196] F. Lecture, "Functional architecture of macaque monkey visual cortex," in *Proc. R. Soc. Lond. B*, 1977, vol. 198, pp. 1–59.

[197] J. A. Movshon, I. D. Thompson, and D. J. Tolhurst, "Spatial summation in the receptive fields of simple cells in the cat's striate cortex.," *J. Physiol.*, vol. 283, p. 53, 1978.

[198] L. M. Hurvich and D. Jameson, "An opponent-process theory of color vision.," *Psychol. Rev.*, vol. 64, no. 6p1, p. 384, 1957.

[199] F. H. Eeckman, M. E. Colvin, and T. S. Axelrod, "A retina-like model for motion detection," in *International 1989 Joint Conference on Neural Networks*, 1989, pp. 247–249 vol.2.

[200] R. W. Means, “Neural Network Retinal Model Real Time Implementation,” DTIC Document, 1992.

[201] A. Wohrer, P. Kornprobst, and T. Viéville, “Virtual Retina : a biological retina model and simulator, with contrast gain control,” INRIA, report, 2007.

[202] K. Masmoudi, M. Antonini, and P. Kornprobst, “Streaming an image through the eye: The retina seen as a dithered scalable image coder,” *Signal Process. Image Commun.*, vol. 28, no. 8, pp. 856–869, Sep. 2013.

[203] K. Masmoudi, “Retina-Inspired Image Coding Schemes,” Ph.D. dissertation, Université Nice Sophia Antipolis, 2012.

[204] X. Guan and H. Wei, “Realistic Simulation on Retina Photoreceptor Layer,” in *2009 International Joint Conference on Artificial Intelligence*, 2009, pp. 179–184.

[205] H. Wei, X. Guan, and Q. Zuo, “Multilayer and Multipathway Simulation on Retina,” in *Artificial Neural Networks – ICANN 2010*, K. Diamantaras, W. Duch, and L. S. Iliadis, Eds. Springer Berlin Heidelberg, 2010, pp. 193–198.

[206] H. Wei and X. Guan, “A Computational Retina Model and Its Self-adjustment Property,” in *Artificial Neural Networks – ICANN 2009*, C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas, Eds. Springer Berlin Heidelberg, 2009, pp. 30–39.

[207] K.-F. Yang, S.-B. Gao, C.-F. Guo, C.-Y. Li, and Y.-J. Li, “Boundary Detection Using Double-Opponency and Spatial Sparseness Constraint,” *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2565–2578, Aug. 2015.

[208] J. Kim, O. Sangjun, Y. Kim, and M. Lee, “Convolutional Neural Network with Biologically Inspired Retinal Structure,” *Procedia Comput. Sci.*, vol. 88, pp. 145–154, 2016.

[209] H. R. Wilson and S. C. Giese, “Threshold visibility of frequency gradient patterns,” *Vision Res.*, vol. 17, no. 10, pp. 1177–1190, 1977.

[210] R. W. Rodieck and M. Watanabe, “Morphology of ganglion cell types that project to the parvocellular laminae of the lateral geniculate nucleus, pretectum, and superior colliculus of primates,” in *Society for Neuroscience Abstracts*, 1988, vol. 14, p. 1120.

[211] M. Qi and Y. Wang, “DEEP-CSSR: Scene classification using category-specific salient region with deep features,” in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 1047–1051.

[212] Y. Jia *et al.*, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[213] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, “Learning deep features for scene recognition using places database,” in *Advances in neural information processing systems*, 2014, pp. 487–495.

[214] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier, “Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition,” *Neurocomputing*, vol. 205, pp. 382–392, Sep. 2016.

[215] Y. Dan and M. Poo, “Spike timing-dependent plasticity of neural circuits,” *Neuron*, vol. 44, no. 1, pp. 23–30, 2004.

[216] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, “STDP-based spiking deep neural networks for object recognition,” *ArXiv161101421 Cs*, Nov. 2016.

[217] S. Zhang, Y. Gong, J. Wang, and N. Zheng, “A Biologically Inspired Deep CNN Model,” in *Advances in Multimedia Information Processing - PCM 2016*, 2016, pp. 540–549.

[218] J.-H. Jacobsen, J. van Gemert, Z. Lou, and A. W. Smeulders, “Structured receptive fields in cnns,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2610–2619.

[219] J. Bruna and S. Mallat, “Invariant scattering convolution networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, 2013.

[220] G. Verkes, “Receptive Fields Neural Networks using the Gabor Kernel Family,” B.S. Thesis, University of Amsterdam, 2017.

[221] Y. Bar-Cohen, *Biomimetics: Biologically Inspired Technologies*. CRC Press, 2005.

[222] J. F. Vincent, O. A. Bogatyreva, N. R. Bogatyrev, A. Bowyer, and A.-K. Pahl, “Biomimetics: its practice and theory,” *J. R. Soc. Interface*, vol. 3, no. 9, pp. 471–482, 2006.

[223] B. Bhushan, *Biomimetics: lessons from nature–an overview*. The Royal Society, 2009.

[224] L. Gomes, “Facebook AI Director Yann LeCun on His Quest to Unleash Deep Learning and Make Machines Smarter,” *IEEE Spectrum: Technology, Engineering, and Science News*, 18-Feb-2015. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/facebook-ai-director-yann-lecun-on-deep-learning. [Accessed: 29-May-2017].

[225] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, “Intel RealSense Stereoscopic Depth Cameras,” *ArXiv170505548 Cs*, May 2017.

[226] H. Müller, N. Michoux, D. Bandon, and A. Geissbuhler, “A review of content-based image retrieval systems in medical applications—clinical benefits and future directions,” *Int. J. Med. Inf.*, vol. 73, no. 1, pp. 1–23, 2004.

[227] N. Tajbakhsh *et al.*, “Convolutional neural networks for medical image analysis: full training or fine tuning?,” *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[228] N. A. Ibraheem, M. M. Hasan, R. Z. Khan, and P. K. Mishra, “Understanding color models: a review,” *ARPN J. Sci. Technol.*, vol. 2, no. 3, pp. 265–275, 2012.

[229] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu, “Color harmonization,” in *ACM Transactions on Graphics (TOG)*, 2006, vol. 25, pp. 624–630.

[230] C. Poynton, “Frequently asked questions about color,” FAQ, 1999.

[231] A. Hanbury and J. Serra, “A 3D-polar coordinate colour representation suitable for image analysis,” *Submitt. Comput. Vis. Image Underst.*, 2002.

[232] “EG 28:1993 - SMPTE Engineering Guideline - Annotated Glossary of Essential Terms for Electronic Production,” *SMPTE EG 281993*, pp. 1–45, May 1993.

[233] E. J. Giorgianni and T. E. Madden, *Digital Color Management: Encoding Solutions*. John Wiley & Sons, 2008.

[234] N. V. Swindale, “Visual map,” *Scholarpedia*, vol. 3, no. 6, p. 4607, Jun. 2008.

[235] J. Park and I. W. Sandberg, “Universal approximation using radial-basis-function networks,” *Neural Comput.*, vol. 3, no. 2, pp. 246–257, 1991.

[236] I. S. Lim, N. W. John, and K. A. Shore, “Smoothing irregularly sampled signals by convolutional RBF networks,” *Electron. Lett.*, vol. 41, no. 22, pp. 1252–1253, 2005.

[237] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs,” *ArXiv160600915 Cs*, Jun. 2016.

[238] T. M. Lehmann, C. Gonner, and K. Spitzer, “Survey: Interpolation methods in medical image processing,” *IEEE Trans. Med. Imaging*, vol. 18, no. 11, pp. 1049–1075, 1999.

[239] R. P. Monroe, “Color Survey Results,” *xkcd*, 04-May-2010. .

[240] J. Heer and M. Stone, “Color naming models for color selection, image editing and palette design,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012, pp. 1007–1016.

[241] T.-W. Lee, T. Wachtler, and T. J. Sejnowski, “Color opponency is an efficient representation of spectral properties in natural scenes,” *Vision Res.*, vol. 42, no. 17, pp. 2095–2103, 2002.

[242] L. J. Croner and E. Kaplan, “Receptive fields of P and M ganglion cells across the primate retina,” *Vision Res.*, vol. 35, no. 1, pp. 7–24, 1995.

[243] E. W. Weisstein, “Full Width at Half Maximum.” [Online]. Available: http://mathworld.wolfram.com/FullWidthatHalfMaximum.html. [Accessed: 14-Jun-2017].

[244] R. Wang, *Introduction to orthogonal transforms: with applications in data processing and analysis*. Cambridge University Press, 2012.

[245] S. Anstis, “The Purkinje rod-cone shift as a function of luminance and retinal eccentricity,” *Vision Res.*, vol. 42, no. 22, pp. 2485–2491, 2002.

[246] E. Y. Smirnova, E. A. Chizhkova, and A. V. Chizhov, “A mathematical model of color and orientation processing in V1,” *Biol. Cybern.*, vol. 109, no. 4–5, pp. 537–547, Oct. 2015.

[247] V. Kyrki, J.-K. Kamarainen, and H. Kälviäinen, “Simple Gabor feature space for invariant object recognition,” *Pattern Recognit. Lett.*, vol. 25, no. 3, pp. 311–318, 2004.

[248] I. Tsoukatos and D. Gunopulos, “Efficient mining of spatiotemporal patterns,” in *International Symposium on Spatial and Temporal Databases*, 2001, pp. 425–442.

[249] D. Sculley, “Web-scale k-means clustering,” in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 1177–1178.

[250] J. Béjar Alonso, “K-means vs Mini Batch K-means: a comparison,” Universitat Politècnica de Catalunya, May 2013.

[251] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, 1987.

[252] R. C. de Amorim and C. Hennig, “Recovering the number of clusters in data sets with noise features using feature rescaling factors,” *Inf. Sci.*, vol. 324, pp. 126–145, Dec. 2015.

[253] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, 1 edition. New York: Cambridge University Press, 2008.

[254] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," in *Encyclopedia of database systems*, Springer, 2009, pp. 532–538.

[255] A. Torralba and A. Efros, "Unbiased look at dataset bias," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011, pp. 1521–1528.

[256] B. Gong, F. Sha, and K. Grauman, "Overcoming dataset bias: An unsupervised domain adaptation approach," in *NIPS Workshop on Large Scale Visual Recognition and Retrieval*, 2012, vol. 3.

[257] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Comput. Vis. Image Underst.*, vol. 106, no. 1, pp. 59–70, 2007.

[258] P. Runeson, "A survey of unit testing practices," *IEEE Softw.*, vol. 23, no. 4, pp. 22–29, 2006.

[259] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, "Evaluating bag-of-visual-words representations in scene classification," in *Proceedings of the international workshop on Workshop on multimedia information retrieval*, 2007, pp. 197–206.

[260] D. J. C. MacKay and L. C. B. Peto, "A hierarchical Dirichlet language model," *Nat. Lang. Eng.*, vol. 1, no. 3, pp. 289–308, Sep. 1995.

[261] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to ad hoc information retrieval," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001, pp. 334–342.

[262] W. W. Eckerson, "Three tier client/server architectures: achieving scalability, performance, and efficiency in client/server applications," *Open Inf. Syst.*, vol. 3, no. 20, pp. 46–50, 1995.

[263] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable Parallel Programming with CUDA," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008.

[264] "CUDA GPUs," *NVIDIA Developer*, 04-Jun-2012. [Online]. Available: https://developer.nvidia.com/cuda-gpus. [Accessed: 08-Jun-2017].

[265] "Accelerated Computing," *NVIDIA Developer*, 16-Nov-2015. [Online]. Available: https://developer.nvidia.com/node/873551/revisions/11259/view. [Accessed: 08-Jun-2017].

[266] "CUDA-Z." [Online]. Available: http://cuda-z.sourceforge.net/. [Accessed: 08-Jun-2017].

[267] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *ArXiv160304467 Cs Softw. Available Tensorfloworg*, Mar. 2016.

[268] "Tensorflow -Programmer's Guide," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/programmers_guide/. [Accessed: 08-Jun-2017].

[269] "GitHub - tensorflow/tensorflow: Computation using data flow graphs for scalable machine learning." [Online]. Available: https://github.com/tensorflow/tensorflow. [Accessed: 17-Jun-2017].

[270] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, p. 2825−2830, Oct. 2011.

[271] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.

[272] C. Macdonald, R. McCreadie, R. L. Santos, and I. Ounis, “From puppy to maturity: Experiences in developing terrier,” *Open Source Inf. Retr.*, vol. 60, 2012.

[273] “Configuring Retrieval in Terrier.” [Online]. Available: http://terrier.org/docs/v2.2.1/configure_retrieval.html. [Accessed: 08-Jun-2017].

[274] M. E. Russinovich and A. Margosis, *Troubleshooting with the Windows Sysinternals Tools*. Microsoft Press, 2016.

[275] “TechPowerUp GPU-Z,” *TechPowerUp*. [Online]. Available: https://www.techpowerup.com/gpuz/. [Accessed: 08-Jun-2017].

[276] A. Troelsen, “Understanding Object Serialization,” in *Pro VB 2005 and the. NET 2.0 Platform*, 2006, pp. 555–571.

[277] C. Coronel and S. Morris, *Database systems: design, implementation, & management*. Cengage Learning, 2016.