Aalto University
School of Science
Degree Programme in Machine Learning and Data Mining

Miquel Perelló Nieto

Merging chrominance and luminance in early, medium, and late fusion using Convolutional Neural Networks

Master's Thesis
Espoo, May 25, 2015

Supervisor: Prof. Tapani Raiko, Aalto University
Advisors: D.Sc. (Tech) Markus Koskela, University of Helsinki
Prof. Ricard Gavaldà Mestre, Universitat Politècnica de Catalunya

Aalto University
School of Science
Degree Programme in Machine Learning and Data Mining

ABSTRACT OF THE MASTER'S THESIS

Author: Miquel Perelló Nieto

Title: Merging chrominance and luminance in early, medium, and late fusion using Convolutional Neural Networks

Date: 25.05.2015 **Language:** English **Number of pages:** 24+166

Professorship: Computer and Information Science **Code:** T-61

Supervisor: Prof. Tapani Raiko

Advisors: D.Sc. (Tech.) Markus Koskela, Prof. Ricard Gavaldà Mestre

The field of Machine Learning has received extensive attention in recent years. More particularly, computer vision problems have got abundant consideration as the use of images and pictures in our daily routines is growing.
The classification of images is one of the most important tasks that can be used to organize, store, retrieve, and explain pictures. In order to do that, researchers have been designing algorithms that automatically detect objects in images. During last decades, the common approach has been to create sets of features – manually designed – that could be exploited by image classification algorithms. More recently, researchers designed algorithms that automatically learn these sets of features, surpassing state-of-the-art performances.
However, learning optimal sets of features is computationally expensive and it can be relaxed by adding prior knowledge about the task, improving and accelerating the learning phase. Furthermore, with problems with a large feature space the complexity of the models need to be reduced to make it computationally tractable (e.g. the recognition of human actions in videos).
Consequently, we propose to use multimodal learning techniques to reduce the complexity of the learning phase in Artificial Neural Networks by incorporating prior knowledge about the connectivity of the network. Furthermore, we analyze state-of-the-art models for image classification and propose new architectures that can learn a locally optimal set of features in an easier and faster manner.
In this thesis, we demonstrate that merging the luminance and the chrominance part of the images using multimodal learning techniques can improve the acquisition of good visual set of features. We compare the validation accuracy of several models and we demonstrate that our approach outperforms the basic model with statistically significant results.

Keywords: Machine learning, computer vision, image classification, artificial neural network, convolutional neural network, image processing, Connectionism.

Preface

This thesis summarizes the work that I have been developing as a Master student on my final research project in the Department of Information and Computer Science in the Aalto Univeristy School of Science under the supervision of Prof. Tapani Raiko and Dr. Markus Koskela, and the remote support from Prof. Ricard Gavaldà.

I would like to thank the discussions and comments about topics related to my thesis to Vikram Kamath, Gerben van den Broeke, and Antti Rasmus. And for general tips and advises during my studies to Ehsan Amid, Mathias Berglund, Pyry Takala and Karmen Dykstra.

Finally, thanks to Jose C. Valencia Almansa for hi visits to Finland and his company in a trip through Europe. And a special thank to Virginia Rodriguez Almansa to believe in me and gave me support towards the acquisition of my Master degree, moving to a foreign country for two years and studding abroad.

"All in all, thanks to everybody,
since without anybody this work
will make no sense"
— Miquel Perelló Nieto (2012)

Otaniemi, 25.05.2015

Miquel Perelló Nieto

A note from the author

After the completion of this thesis, further experiments seem to demonstrate that it is possible to achieve the same validation accuracy by merging the chrominance and the luminance in early fusion with similar number of parameters. However, learning separate filters for the luminance and the chrominance achieved the same performance, while showing a faster learning curve, better generalization and the possibility of parallelize the training on different machines or graphics processor units. Furthermore the results of this thesis are extensible to other sets of features where the fusion level is not clear (e.g. audio, optical flow, captions, or other useful features).

Otaniemi, 25.05.2015

Miquel Perelló Nieto

Contents

Mathematical Notation

- This thesis contains some basic mathematical notes.
- I followed notation from [Bishop, 2006]
  - Vectors: lower case Bold Roman column vector $\mathbf{w}$ or $\mathbf{w} = (w_1, \ldots, w_M)$
  - Vectors: row vector is the transpose $\mathbf{w}^T$ or $\mathbf{w}^T = (w_1, \ldots, w_M)^T$
  - Matrices: Upper case Bold Roman $\mathbf{M}$
  - Closed interval: $[a, b]$
  - Open interval: (a, b)
  - Semi-closed interval: $(a, b]$ and $[a, b)$
  - Unit/identity matrix: of ize $M \times M$ is $\mathbf{I}_M$

Acronyms

Adaline Adaptive Linear Neuron or Adaptive Linear Element. 77, 157, 162

AI Artificial Intelligence. 4, 65, 157

ANN Artificial Neural Network. 3, 9, 21, 25, 35–37, 40, 43–46, 52, 54, 55, 59, 63, 65, 71, 74, 76, 78–83, 85–87, 94, 96, 104, 127, 157–161, 163–165

BM Boltzmann Machine. 73, 82, 85, 87, 158

BN Belief Network. 57, 85, 88, 158, 159

BoV Bag of Visual words. 15, 59, 158

BoW Bag of Words. 15, 158

BPTT backpropagation through time. 56, 158

CIE Commission Internationale de l'Clairage. 16

CNN Convolutional Neural Network. xviii, 2–4, 9, 37, 54, 56, 57, 59, 61–63, 86–89, 91–93, 95, 97, 99, 100, 103, 107, 112, 115, 132, 134, 135, 162, 163

DARPA Defense Advanced Research Projects Agency. 21, 84

DBN Deep Belief Network. 57, 88, 159

DoG Difference of Gaussian. 12, 13, 29

EDVAC Electronic Discrete Variable Automatic Computer. 70, 159

ELM Extreme Learning Machine. 36, 54, 55

ENIAC Electronic Numerical Integrator And Computer. 70, 159, 160

ESN Echo State Network. 88, 160

FNN Feed-forward neural network. 41, 160, 162, 165

GLOH Gradient location-orientation histogram. 14, 160

GPU Graphics Processing Unit. 61, 92, 95–97, 115, 116, 160

HM Helmholtz Machine. 88, 160

HOG Histogram of Oriented Gradients. 14, 59, 160

HoPS Histogram of Pattern Sets. 15, 160

HSL Hue-Saturation-Lightness. 93, 161

HSV Hue-Saturation-Value. 93, 161

ICA Independent Component Analysis. 86, 161

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 9, 89

Infomax maximum mutual information. 83, 86, 87, 161

K-cells Koniocellular cells. 30, 161

LGN Lateral Geniculate Nucleus. 27, 30, 31, 33, 132, 161–163

LoG Laplacian of Gaussian. 11–13, 29, 161

LRN Local Response Normalization. 62, 89, 162

M-cells Magnocellular cells. 29, 30, 161, 162

MFNN Multilayer Feedforward Neural Network. xvii, 36, 41, 44, 49, 50, 54, 55, 86, 162

MLP Multilayer Perceptron. 162

P-cells Parvocellular cells. 29–31, 161, 163

PCA-SIFT PCA-Scale-Invariant Feature Transform. 14, 163

PCA Principal Component Analysis. 14, 160, 163

PReLU Parametric Rectified Linear Unit. 38, 61, 163

RBF Radial Basis Function. 37, 86, 163

RBM Restricted Boltzmann Machine. 57, 88, 163

ReLU rectified linear unit. 37, 38, 59, 61, 63, 88, 89, 100, 163

RGB Red Green and Blue. 16, 18, 21, 93, 99–101, 103, 116, 132, 133, 165

RNN Recurrent Neural Network. 36, 55, 56, 85, 158, 160, 163

SGD Stochastic Gradient Descent. 52, 53, 164

SIFT Scale-Invariant Feature Transform. 12–14, 59, 160, 163, 164

SNARC Stochastic Neural Analog Reinforcement Calculator. xviii, 73, 164

SOM Self-Organizing Map. 37, 79, 82, 83

SURF Speeded-Up Robust Features. 14, 59, 164

SVM Support Vector Machine. 9, 59, 81, 87, 88, 164

TPE Temporal Propositional Expression. 70, 165

VC dimension Vapnik-Chervonenkis dimension. 81, 165

XYZ proportions of the RGB color space. 16, 20

YCbCr Y (luminance), Cb (blue difference), Cr (red difference). 18

YIQ Y (luminance) IQ (chrominance). 18, 20, 93

YUV Stands for luminance (Y) and chrominance (UV). 18, 21, 93, 101, 132–134, 165

List of Figures

List of Tables

Chapter 1

Introduction

"Do the difficult things while they are easy and do the great things while they are small. A journey of a thousand miles must begin with a single step"

— Lao Tzu

Machine learning and artificial intelligence are two young fields of computer science. However, their foundations are inherent in the understanding of intelligence, and the best example of intelligence can be found in the brain. The comprehension of the process of learning and the emergence of intelligence has been widely studied since humans had the capability to hypothesize what other people think. Philosophers, psychologists, sociologists, and physiologists, have attempted to solve these type of questions. However, with the development of the first computers, engineers, mathematicians, physicists and computer scientists started implementing the old ideas on electronic devices. After years of developing new ideas – and improving the computational power of these electronic devices – the terms machine learning and artificial intelligence have gotten further attention; and actually got their name. Nowadays, the tasks that researchers try to automate are usually very practical and less theoretical. Nevertheless, the biggest changes come with unexpected theoretical breakthroughs. Researchers explore new algorithms to solve important problems in a large number of domains: path planning, decision making, optimal control, game theory, regression, clustering, pattern recognition, information retrieval, logics, data mining, and in multiple applications that – we hope – can be learned automatically. In this thesis, we look at a narrow problem of pattern recognition in computer vision – the image classification problem – and evaluate state-of-the-art models to extract important conclusions that can help us to understand the process of vision. We analyze and extend the previous work, in a desire to contribute in that manner to the computer vision research community.

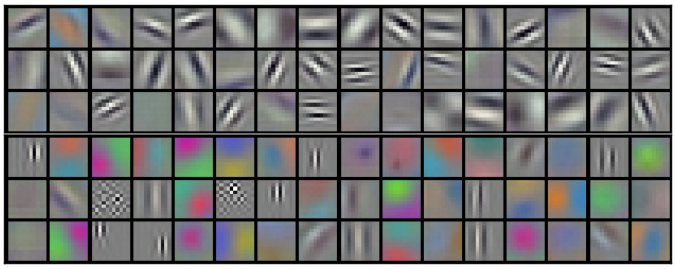
1.1 Motivation

In the computer vision community, researchers have been trying to improve image classification and object recognition algorithms since the late 80s. However, we still do not know what is the optimal way to solve these problems. Despite the fact that humans are able to solve these tasks from the early stages of our lives, we still do not comprehend which are the features that we use to succeed. For us, colors are a natural part of vision, but, the perception of colors is a subjective process for each brain. Furthermore, there are animals that only perceive the environment in a gray-scale of tones, while other animals can perceive ultra-light frequencies of color. The requirements of the retinal photoreceptors have changed with evolution, making different species to perceive a diverse number of colors.

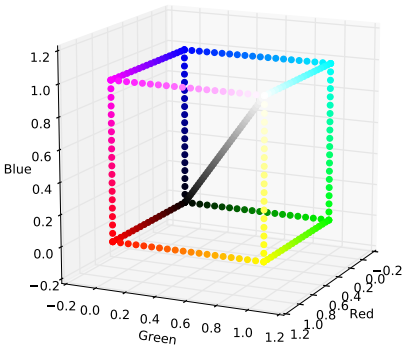
One of the most basic vision systems is the perception of one unique light wavelength. This is enough to perceive the luminance of the environment and determine edges, shapes, blobs and other patterns. For example, nocturnal animals usually have one type of photoreceptor and, therefore, can only see in a gray scale. The luminance is one of the most important features to recognize objects and scenes, while the perception of additional colors has been valuable to distinguish between poisonous fruits, venomous animals, and other important situations.

Although we still do not know completely how our brain understands the environment that we perceive, lately we have been able to train mathematical models that learn to solve image classification tasks automatically. Earlier approaches tried to create sets of hand-crafted features, usually driven by common sense or inspiration from biology. However, during the last decade, researchers achieved very good results by using mathematical models that learn to extract the sets of features automatically. The resulting features could be astonishing, as they for example discovered the importance of the luminance as an isolated feature, while the chrominance seemed to have less importance (although it contains 2/3 parts of the total information).

If we look at the initial filters that one of these models learned from a huge number of images (see Figure 1.1a) we can see that more than half of the filters are focused on the luminance, while the chrominance plays a secondary role for the image classification task. If we visualize all the possible values that a pixel can have, the gray-scale is only the diagonal of the resulting 3D cube. This small region in a cube with 256 values per color correspond to a portion of $256/256^3 = 1/256^2 \approx 1.5 \times 10^{-5}$ of the complete cube. However, the parameters of a Convolutional Neural Network (CNN) can take "continuous" values (double floats of 64 bits), decreasing the ratio of the diagonal exponentially to zero. This diagonal can be codified with one unique feature if we apply a rotation to the cube. Then, the other two components will represent the chrominance and can be disentangled. In the actual CNNs, the model needs to synchronize the three features together in order to work in the infinitely small luminance region, requiring large amounts of data and computational time. However, given that we know the importance of the luminance in the first stages, it should be possible to accelerate the learning if – as a preprocessing step – we present the images with the appropriate rotation, and train the filters separately.



(a) First filters of Alexnet. During the training the first 48 filters were trained in a separate machine from the bottom 48 filters.



(b) From all the possible colors in the RGB space only the diagonal axis is focused on the luminance.

Figure 1.1: **Motivation** Where the idea came from

With this in mind, we only need to investigate at which level to merge the color and the luminance information, in order to obtain the best results. This can be done by studying and validating several models; and it is the focus of this thesis.

1.2 Objective and scope

In this thesis, we propose to modify the architectures of previous state-of-the-art Convolutional Neural Networks (CNNs) to study the importance of luminance and chrominance for the image classification tasks. Although the initial idea was to try these experiments in deep networks and natural images, we opted to reduce the complexity of the problem to small images and reduced models. However, the results of this thesis can be extended to larger problems if the necessary computational capabilities are available.

In addition, the initial idea for this thesis was to create a self-contained report. For that reason, the necessary background to understand the topics related to Convolutional Neural Networks (CNNs) and image classification is summarized in the first chapters (identified as Background chapters). First, in Chapter 2 we present the problem of image classification, and which has been the most common approach to solve it. Then, Chapter 3 gives a small introduction on how the biological visual system works in humans and other mammals. The purpose of this chapter is to originate inspiration for future computer vision algorithms, and to explain the initial motivation that pushed the creation of Artificial Neural Networks (ANNs) and the Connectionism ideas. Chapter 4 introduces the concept of Artificial Neural Networks (ANNs), the mathematical model that describes them, how to train them, and series of methods to improve their performance. Next, Chapter 5 introduces the Convolutional Neural Network (CNN), one specific type of Artificial Neural Networks (ANNs) specially designed for tasks with strong assumptions about spatial or temporal local correlations. Finally, to end the background portion of the thesis, I chose to dedicate the complete Chapter 6 to explain the history of the Connection-

ism and all the important figures that have been involved in the understanding of the human intelligence, Artificial Intelligence (AI), and new techniques to solve a variety of different problems.

In the second part, we present the methodological contributions of the thesis. First, in Chapter 7, we explain the methodology followed in the thesis. We present the dataset, the color spaces, the architectures, different methods to merge the initial features, and finally the available software frameworks to test CNNs. Then, in Chapter 8, we discuss the first analysis to test different state-of-the-art architectures and how to evaluate their use of the colors after the training. Then, we present the set of experiments to test a diverse set of hypotheses about merging the colors. Each of the architectures is explained in the same chapter, as well as the evaluation criteria of each network. Next, we describe a statistical test to demonstrate the validity of our findings. Chapter 9 presents the results of the initial analysis, all the experiments and the statistical tests. Finally, we conclude the thesis with a discussion about our findings, a summary, and future work in Chapter 10.

Chapter 2

Image classification

"I can see the cup on the table," interrupted Diogenes, "but I can't see the 'cupness' ". "That's because you have the eyes to see the cup," said Plato, "but", tapping his head with his forefinger, "you don't have the intellect with which to comprehend 'cupness'."

— Teachings of Diogenes (c. 412- c. 323 B.C)

In this chapter, we give an introduction to the problem of image classification and object recognition. This has been one of the typical problems in pattern recognition since the development of the first machine learning models. First, we explain what is image classification and why is difficult in Section 2.1. We give some visual representations of how computers see images and briefly explain which has been the common approaches on computer vision to solve the object recognition problem in Section 2.1.1. Then, we present a different approach from the Connectionism field that tries to solve the problem with more general techniques in Section 2.1.2; this approach is the focus of the entire thesis.

After this small introduction we explain with more details the common approach of computer vision in the next sections. One of the most difficult problems to solve is due to the huge variability of a representation of a 3D object in a two dimensional image. We explain different variations and image transformations in Section 2.2. Then, we start solving the problem of recognizing objects first by localizing interesting regions on the image in Section 2.3, and then merging different regions to describe in a more compact manner the specific parts of the objects in Section 2.4. With the help of the previous descriptors, several techniques try to minimize the intraclass variations and maximize the separation between different classes, this methods based usually in data mining and clustering are explained in Section 2.5.

After the description of the common approaches, we explain the concept of color in Section 2.6, and show that there are better ways to represent the images using different color spaces in Section 2.7. Also, we highlight the importance of the luminance to recognize objects in Section 2.8.

Finally, we present some famous datasets that have been used from the 80's to the present day in Section 2.9.

A more complete and accurate description of the field of computer vision and digital images can be found in the references [Szeliski, 2010] and [Glassner, 1995].

2.1 What is image classification?

Image classification consists of identifying which is the best description for a given image. This task can be ambiguous, as the best description can be subjective depending on the person. It could refer to the place where the picture was taken, the emotions that can be perceived, the different objects, the identity of the object, or to several other descriptions. For that reason, the task is usually restricted to a specific set of answers in terms of labels or classes. This restriction alleviates the problem and makes it tractable for computers; the same relaxation applies to humans that need to give the ground truth of the images. As a summary, the image classification problem consists of the classification of the given into one of the available options.

It can be difficult to imagine that identifying the objects or labeling the images could be a difficult task. This is because people do not need to perform conscious mental work in order to decide if one image contains one specific object or not; or if the image belongs to some specific class. We also see in other animals similar abilities, many species are able to differentiate between a dangerous situation and the next meal. But, one of the first insights into the real complexity occurred in 1966, when Marvin Minsky and one of his undergraduate students tried to create a computer program able to describe what a camera was watching (See [Boden, 2006], p. 781, originally from [Crevier, 1993]). On that moment, the professor and the student did not realize about the real complexity of the problem. Nowadays it is known to be a very difficult problem, belonging to the class of inverse problems, as we try to classify 2D projections from a 3D physical world.

The difficulty of this task can be easier to understand if we examine the image represented in a computer. Figure 2.1 shows three examples from the CIFAR-10 dataset. From the first two columns it is easy to identify to which classes the images could belong. That is because our brain is able to identify the situation of the main object, to draw the boundaries between objects, to identify their internal parts, and to classify the object. The second column contains the same image in gray-scale. In the last column the gray value of the images has been scaled from the range $[0, 255]$ to $[0, 9]$ (this rescaling is useful for visualization purposes). This visualization is a simplified version of the image that a computer sees. This representation can give us an intuition about the complexity for a computer to interpret a batch of numbers and separate the object from the background. For a more detailed theoretical explanation about the complexity of object recognition see the article [Pinto et al., 2008].

In the next steps, we see an intuitive plan to solve this problem.

1. Because the whole image is not important to identify the object, we first need to find the points that contain interesting information. Later, this collection

Figure 2.1: **Digital image representation** with some examples from the CIFAR-10 dataset.The first column contains the original images, second column the gray-scale, and the last column contains the gray-scale values scaled to the range $[0, 9]$

of points can be used to define – for example – the object boundaries.

2. Group all the different collections of points to generate small objects or parts that can explain larger – or more complex – objects.

3. In some cases identifying the background can be a very good prior in order to classify the object. For example, if we are classifying animals and the picture is taken under the sea, then it is more probable to find a sea lion than a savanna lion.

4. All the extracted information conforms a set of features that we can use to compare with past experiences and use these experiences to classify the new images.

If the features that we are using are good enough and our past experiences cover a large part of the possible situations it should be easy to decide which is the correct class of the object. However, each of the previous steps has its own complexities and researchers work on individual parts to find better solutions. Additionally, this is a simplification of the common approach used in computer vision.

Although we saw a visual representation of the image stored in a computer, from the mathematical point of view it is easier to imagine the three channels of the image

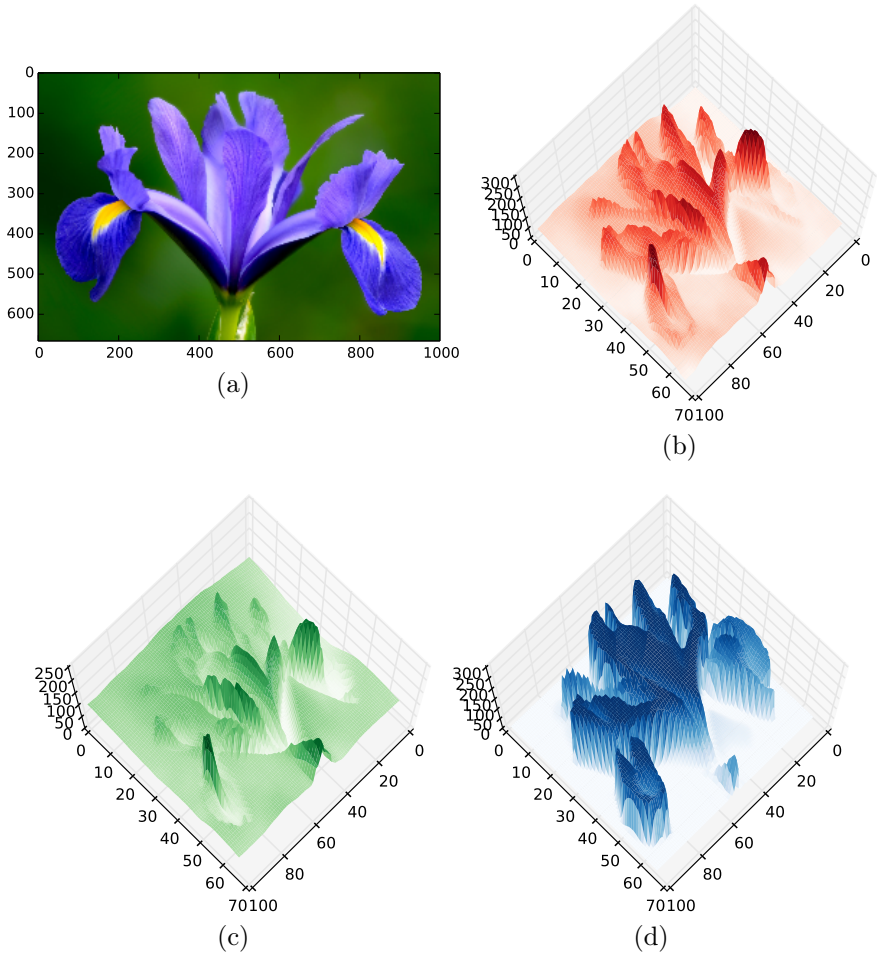


Figure 2.2: **Iris versicolor RGB example.** This is a representation of the three RGB channels as 3D surfaces

(red, green and blue) as surfaces, where the strength of the color in the specific pixel is represented with a higher peak on the surface. With this approach on mind it is easier to apply mathematical equations and algorithms that look for specific shapes in the surfaces. Figure 2.2 shows a picture of a flower with its corresponding three color surfaces.

2.1.1 Computer vision approach

We saw that image classification is a difficult task, and we explained one intuitive approach to solve the problem. But, what has been the common approach in the past?

First of all, we should mention that having prior knowledge of the task could help to decide which type of descriptors to extract from the images. For example, if the objects are always centered and use the same orientation, then we could not be interested in descriptors that are invariant to translation and rotation. This and

other transformations that can be applied to the images are explained in Section 2.2.

Once the limitations of the task are specified, it is possible to choose the features that can be used. Usually, these features are based on the detection of edges, regions and blobs throughout the image. These detectors are usually hand-crafted to find such features that could be useful at the later stages. Some region detectors are explained in Section 2.3, whilst some feature descriptors are shown in Section 2.4.

Then, given the feature descriptors it is possible to apply data mining and machine learning techniques to represent their distributions and find clusters of descriptors that are good representatives of the different categories that we want to classify. In Section 2.5 we give an introduction to some feature cluster representations.

Finally, with the clusters or other representations we can train a classification model. This is often a linear Support Vector Machine (SVM), an SVM with some kernel or a random forest.

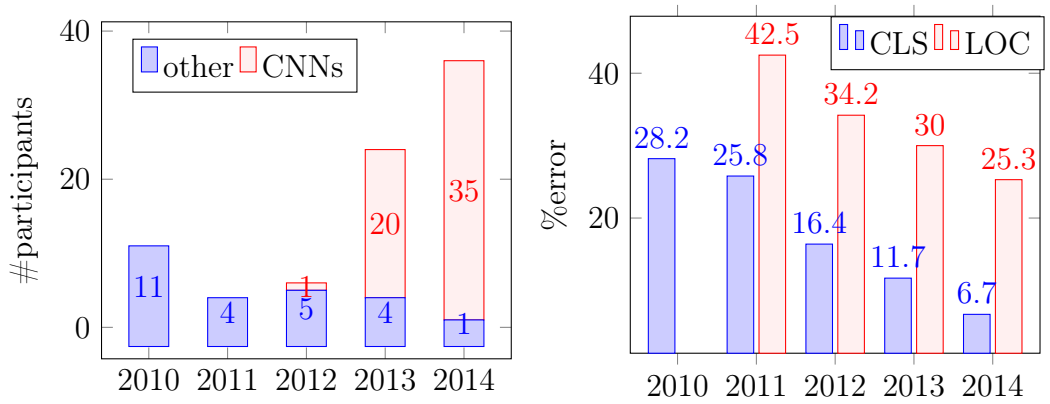
2.1.2 Connectionism approach

Another approach is to use an Artificial Neural Network (ANN) to find automatically the interesting regions, the feature descriptors, the clustering and the classification. In this case the problem is on which structure of ANN to choose, and lots of hyperparameters that need to be tuned and decided.

Convolutional Neural Network (CNN) have demonstrated to achieve very good results on hand-written digit recognition, image classification, detection and localization. For example, before 2012, the state-of-the-art approaches for image classification were the previously mentioned and explained in the next sections. However, in 2012, a deep CNN [Krizhevsky et al., 2012] won the classification task on ImageNet Large Scale Visual Recognition Challenge (ILSVRC)2012. From that moment, the computer vision community started to became interested in the hidden representation that these models were able to extract, and started using them as a feature extractiors and descriptors. Figure 2.3 shows the number of participants using CNNs during the period (2010-2014) in the ILSVRC image classification and location challenges, together with the test errors of the winners in each challenge.

2.2 Image transformations

We explained previously that image classification belongs to the class of inverse problems because the classification uses a projection of the original space to a lower dimensional one; in this case from 3D to 2D. To solve this problem one of the approaches consists in extracting features that are invariant to scale, rotation, or perspective. However, these techniques are not always applied, because in some cases, the restrictions of the problem in hand do not allow specific transformations. Moreover, the use of these techniques reduces the initial information, possibly deteriorating the performance of the classification algorithm. For this reason, prior knowledge of the task – or cross-validation – is very useful to decide which features



(a) number of participants

(b) Winners error results

Figure 2.3: **ILSVRC results** during the years 2010-2014

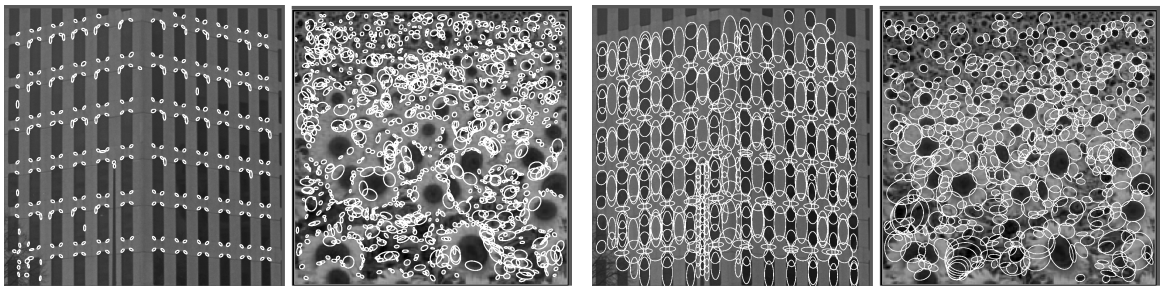
are better on each task.

Occasionally, the objects to classify are centered and scaled. One example of this is the MNIST dataset, composed of centered and re-scaled hand-written digits. In this case, it could be possible to memorize the structure of the digits and find the nearest example. However, this approach is computationally expensive at test time and better techniques can be used. On the other hand, some datasets do not center the objects, and they can be in any position of the frame. For example, in detecting people in an image, the persons can be standing in front of the camera or in the background, and we need a set of features invariant to scale and translation. Furthermore, some objects can appear with different rotations and adding rotation invariant features could help to recognize them. Also, the reflection is sometimes useful. Finally, we can combine all the previous transformations with the term affine transformations. These transformations preserve all the points in the same line, but possibly not the angles between the lines.

It is very useful to know the set of transformations that the dataset can incorporate. For example, if we want to detect people, the reflection of people can be used for training. This idea has been extended to increase the number of training samples; usually called data augmentation.

2.3 Region detectors

One of the most important prerequisites to find good image descriptors is to find good explanatory regions. To localize these regions, there are several approaches more or less suited to some specific invariances. Some interesting regions can be patches with more than one dominant gradient (e.g. corners or blobs), as they reduce the possibility of occurrence. On the other hand, there are regions that are usually not good (in the explanatory sense), for example regions with no texture are quite common and could be found in several different objects. Also, the straight lines present an aperture problem as the same portion of line can be matched at multiple



(a) Harris-Laplace detector (b) Laplacian detector

Figure 2.4: **Comparison of detected regions between Harris and Laplacian detectors** on two natural images (image from [Zhang and Marszalek, 2007])

positions. In general, the regions should be repeatable, invariant to illumination and distinctive.

Edge detectors are a subclass of region detectors. They are focused on detecting regions with sharp changes in brightness. The Canny detector [Canny, 1986] is one of the most common edge detectors. Other options are the Sobel [Lyvers and Mitchell, 1988], Prewitt [Prewitt, 1970] and Roberts cross operators. For an extended overview about edge detectors see [Ziou and Tabbone, 1998].

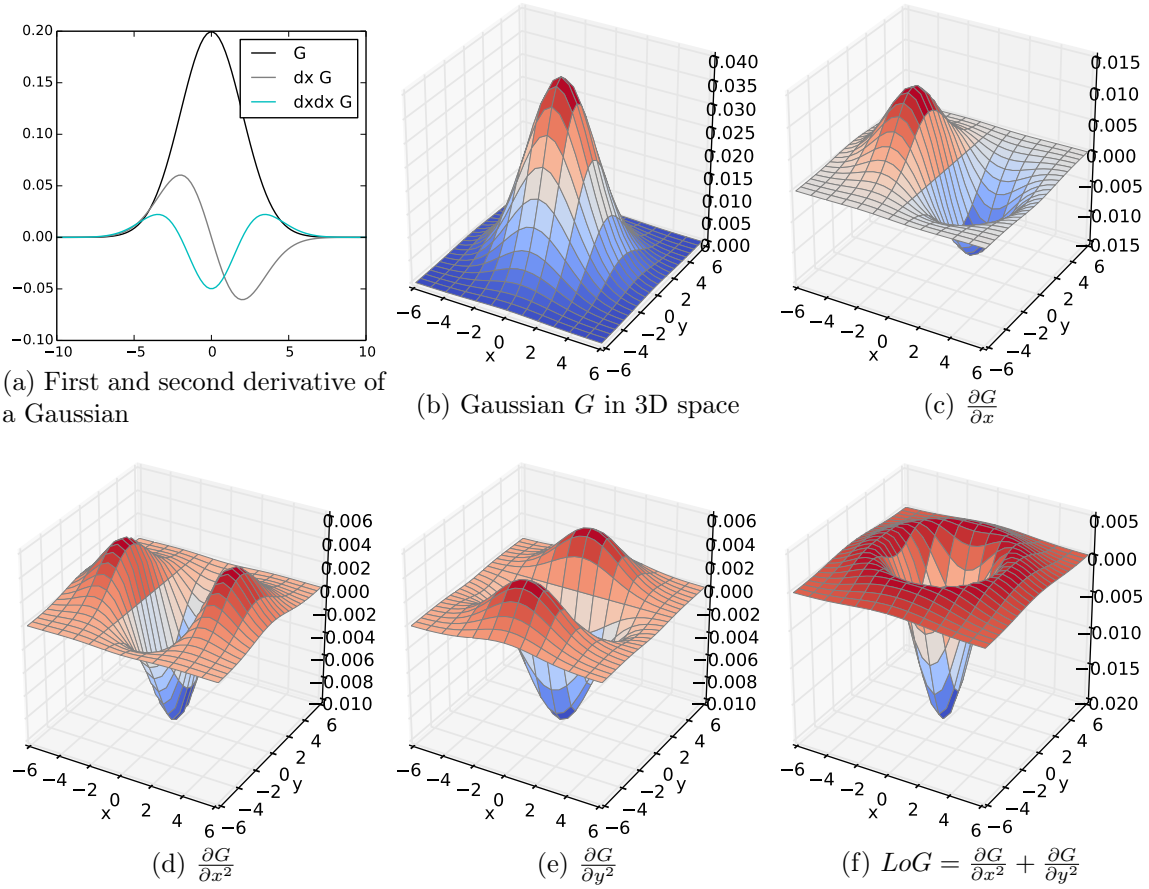
The Harris corner detector [Harris and Stephens, 1988] is a more general region detector. It describes one type of rotation-invariant feature detector based on a filter of the type $[-2, -1, 0, 1, 2]$. Furthermore, the Laplacian detector [Lindeberg, 1998] is scale and affine-invariant and extracts blob-like regions (see Figure 2.4b). Similarly, the Harris-Laplace detector [Mikolajczyk and Schmid, 2004; Mikolajczyk et al., 2005a] detects also scale and affine-invariant regions. However, the detected regions are more corner-like (see Figure 2.4a). Other common region detectors include the Hessian-Laplace detector [Mikolajczyk et al., 2005b], the salient region detector [Kadir and Brady, 2001], and Maximally Stable Extremal Region (MSER) detector[Matas et al., 2004].

The blob regions, i.e. regions with variation in brightness surrounded by mostly homogeneous levels of light, are often good descriptors. One very common blob detector is the Laplacian of Gaussian (LoG) [Burt and Adelson, 1983]. In gray-scale images where $\mathbf{I}(x, y)$ is the intensity of the pixel in the position $[x, y]$ of the image $\mathbf{I}$, we can define the Laplacian as:

$$L(x, y) = (\nabla^2 \mathbf{I})(x, y) = \frac{\partial^2 \mathbf{I}}{\partial x^2} + \frac{\partial^2 \mathbf{I}}{\partial y^2} \tag{2.1}$$

Because of the use of the second derivative, small noise in the image is prone to activate the function. For that reason, the image is smoothed with a Gaussian filter as a pre-processing step. This can be done in one equation:

$$LoG(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.2}$$



(a) First and second derivative of a Gaussian

(b) Gaussian G in 3D space

(c) $\frac{\partial G}{\partial x}$

(d) $\frac{\partial G}{\partial x^2}$

(e) $\frac{\partial G}{\partial y^2}$

(f) $LoG = \frac{\partial G}{\partial x^2} + \frac{\partial G}{\partial y^2}$

Figure 2.5: **Laplacian of Gaussian**

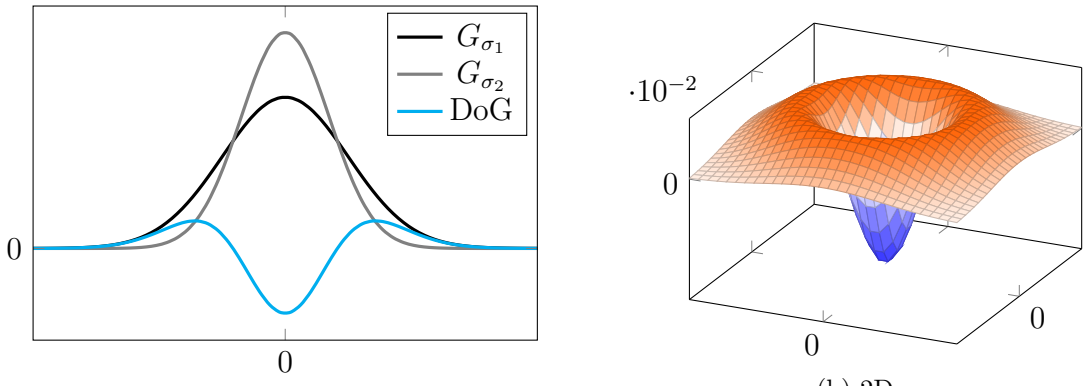
Also, the Difference of Gaussian (DoG) [Lowe, 2004] can find blob regions and can be seen as an approximation of the LoG. In this case the filter is composed of the subtraction of two Gaussians with different standard deviations. This approximation is computationally cheaper than the former. Figure 2.6 shows a visual representation of the DoG.

2.4 Feature descriptors

Given the regions of interest explained in the last section, it is possible to aggregate them to create feature descriptors. There are several approaches to combine different regions.

2.4.1 SIFT

Scale-Invariant Feature Transform (SIFT) [Lowe, 1999, 2004] is one of the most extended feature descriptors. All previous region detectors were not scale invariant, that is the detected features require a specific zoom level of the image. On the



(a) 1D (b) 2D

Figure 2.6: **Example of DoG in one and two dimensions**

contrary, SIFT is able to find features that are scale invariant – to some degree. The algorithm is composed of four basic steps:

Detect extrema values at different scales

In the first step, the algorithm searches for blobs of different sizes. The objective is to find Laplacian of Gaussian (LoG) regions in the image with different σ values, representing different scales. The LoG detects the positions and scales in which the function is strongly activated. However, computing the LoG is expensive and the algorithm uses the Difference of Gaussian (DoG) approximation instead. In this case, we have pairs of sigmas $\sigma_l = k\sigma_{l-1}$, where l represents the level or scale. An easy approach to implement this step is to scale the image using different ratios, and convolve different Gaussians with an increasing value of $k\sigma$. Then, the DoG is computed by subtracting adjacent levels. In the original paper the image is rescaled 4 times and in each scale 5 Gaussians are computed with $\sigma = 1.6$ and $k = \sqrt{2}$.

Once all the candidates are detected, they are compared with the 8 pixels that are surrounding the point at the same level, as well as the 9 pixels in the upper and lower levels. The pixel that has the largest activation is selected for the second step.

Refining and removing edges

Next, the algorithm performs a more accurate inspection of the previous candidates. The inspection consists of computing the Taylor series expansion in the spatial surrounding of the original pixel. In [Lowe, 1999] a threshold of 0.03 was selected to remove the points that did not exceeded this value. Furthermore, because the DoG is prone to detect also edges, a Harris corner detector is applied to remove them. To find them, a 2×2 Hessian matrix is computed and the points in which the eigenvalues are different with a certain degree are considered to be edges and are discarded.

Orientation invariant

The remaining keypoints are modified to make them rotation invariant. The surrounding is divided into 36 bins covering the 360 degrees. Then the gradient in each bin is computed, scaled with a Gaussian centered in the middle and with a σ equal to 1.5 times the scale of the keypoint. Then, the bin with the largest value is selected and the bins with a value larger than the 80% are also selected to compute the final gradient and direction.

Keypoint descriptor

Finally, the whole area is divided into $16 \times 16 = 256$ small blocks. The small blocks are grouped into 16 squared blocks of size 4×4. In each small block, the gradient is computed and aggregated in eight circular bins covering the 360 degrees. This makes a total of 128 feature descriptors that compose the final keypoint descriptor. Additionally, some information can be stored to avoid possible problems with brightness or other problematic situations.

2.4.2 Other descriptors

Speeded-Up Robust Features (SURF) [Bay et al., 2006, 2008] is another popular feature descriptor. It is very similar to SIFT, but, it uses the integral of the original image at different scales and finds the interesting points by applying Haar-like features. This accelerates the process, but results in a smaller number of keypoints. In the original paper, the authors claim that this reduced set of descriptors was more robust than the ones discovered by SIFT.

Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005] was originally proposed for pedestrian detection and shares some ideas of SIFT. However, it is computed extensively over the whole image, offering a full patch descriptor. Given a patch of size 64×128 pixels, the algorithm divides the region into 128 small cells of size 8×8 pixels. Then, the horizontal and vertical gradients at every pixel are computed, and each 8×8 cell creates a histogram of gradients with 9 bins, divided from 0 to 180 degrees (if the orientation of the gradient is important it can be incorporated by computing the bins in the range 0 to 360 degrees). Then, all the cells are grouped into blocks of 4×4 with 50% overlap with each adjacent block. This makes a total of $7 \times 15 = 105$ blocks. Finally, the bins of each cell are concatenated and normalized. This process creates a total of $105_{\text{blocks}} \times 4_{\text{cells}} \times 9_{\text{bins}} = 3780_{\text{features}}$.

Other features are available but are not discussed here, some examples are the Gradient location-orientation histogram (GLOH) [Mikolajczyk and Schmid, 2005] with 17 locations, 16 orientation bins in a log-polar grid and uses Principal Component Analysis (PCA) to reduce the dimensions to 128; the PCA-Scale-Invariant Feature Transform (PCA-SIFT) [Ke and Sukthankar, 2004] that reduces the final dimensions with PCA to 36 features; moment invariants [Gool et al., 1996]; SPIN [Lazebnik et al., 2005]; RIFT [Lazebnik et al., 2005]; and HMAX [Riesenhuber and Poggio, 1999].

2.5 Feature Cluster Representation

All the previouslydiscussed detectors and descriptors can be used to create large amounts of features to classify images. However, they result in very large feature vectors that can be difficult to use for pattern classification tasks. To reduce the dimensionality of these features or to generate better descriptors from the previous ones it is possible to use techniques like dimensionality reduction, feature selection or feature augmentation.

Bag of Visual words (BoV) [Csurka and Dance, 2004] was inspired by an older approach for text categorization: the Bag of Words (BoW) [Aas and Eikvil, 1999]. This algorithm detects visually interesting descriptors from the images, and finds a set of clusters representatives from the original visual descriptors. Then, the clusters are used as reference points to form – later – a histogram of occurrences; each bin of the histogram corresponds to one cluster. For a new image, the algorithm finds the visual descriptors, computes their distances to the different clusters, and then for each descriptor increases the count of the respective bin. Finally, the description of the image is a histogram of occurrences where each bin is referred to as a visual word.

Histogram of Pattern Sets (HoPS) [Voravuthikunchai, 2014] is a recent new algorithm for feature representation. It is based on random selection of some of the previously explained visual features (for example BoV). The selected features are binarized: the features with more occurrences than a threshold are set to one, while the rest are set to zero. The set of features with the value one creates a transaction. The random selection and the creation of transactions is repeated P times obtaining a total of P random transactions. Then, the algorithm uses data mining techniques to select the most discriminative transactions, for example Frequent Pattern (FP) [Agrawal et al., 1993] or Jumping Emerging Patterns (JEPs) [Dong and Li, 1999]. Finally, the last representations is formed by $2 \times P$ bins, two per transaction: one for positive JEPs and the other for negative JEPs. In the original paper this algorithm demonstrated state-of-the-art results on image classification with the Oxford-Flowers dataset, object detection on PASCAL VOC 2007 dataset, and pedestrian recognition.

2.6 Color

> *"We live in complete dark. What is color but your unique interpretation of the electrical impulses produced in the neuro-receptors of your retina?"*
>
> — Miquel Perelló Nieto

The colors emerge from different electromagnetic wavelengths in the visible spectrum. This visible portion is a small range from 430 to 790 THz or 390 to 700nm (see Figure 2.7). Each of these isolated frequencies creates one pure color, while combinations of various wavelengths can be interpreted by our brain as additional

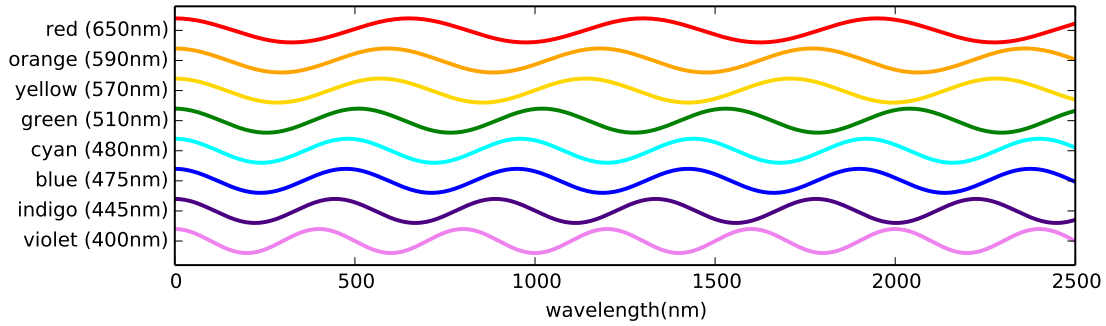


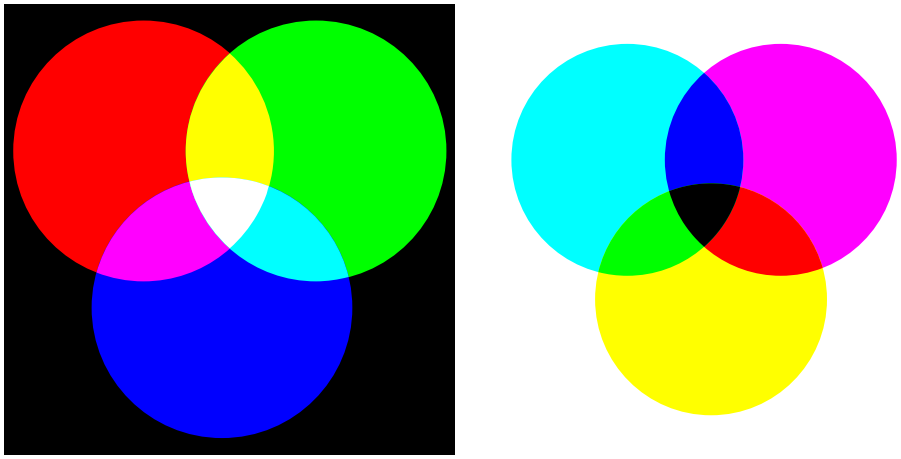
Figure 2.7: **Visible colors wavelengths**

colors. From the total range of pure colors, our retina is able to perceive approximately the blue, green, and red (see Chapter 3). While the co-occurrence of some adjacent wavelengths can accentuate the intermediate frequency – green and orange produce yellow and red and blue produce cyan – more separated wavelengths like red and blue generate a visible color that does not exist in the spectrum. This combination is perceived by our brain as magenta. By mixing three different colors it is possible to simulate all the different wavelenghts. These three colors are called primary colors. The red, green and blue are additive, this means that they act as sources of light of the specific wavelength (see Figure 2.8a). On the other hand, the cyan, magenta and yellow are subtractive colors, meaning that they act as filters that only allow a specific wavelength to pass, filtering everything else. For example, cyan filters everything except the range $520 - 490$ nm, yellow $590 - 560$ nm, and magenta filters mostly the green region allowing the light of both lateral sides of the visible spectrum. The subtractive colors can be combined to filter the additional wavelengths creating the red, green and blue colors (see Figure 2.8b). The colors black, white and gray are a source of light uniformly distributed in all the visual spectrum, they go from the complete darkness (absence of any wavelength), to the brightest white (light with all the visible wavelengths).

2.7 Color spaces

Although the visual spectrum is a continuous space of wavelengths, the techniques that we use to represent the different colors had to be discretized and standardized as a color space. It was William David Wright and John Guild that in the 1920s started to investigate the human perception of colors. Later in 1931 the CIE (Commission Internationale de l'Clairage) RGB and Expanded (XYZ) color spaces where specified. The RGB is one of the most used color spaces in analog and digital screens (see Figure 2.9a). XYZ, on the other hand, was a human sight version of the RGB that was standardized after several years of experiments on the human vision perception (see Figure 2.9c). In XYZ, the luminance (X) is separated from the chrominance (Y and Z).

Furthermore, during the transition between black and white and color television,



(a) RGB (b) CMY

Figure 2.8: **Primary colors** (a) RGB are additive colors, this means that the sum of the colors creates the white (an easy example is to imagine a dark wall where light focus of the three colors are illuminating. The combination of all the focus create the white spot). (b) CMY are subtractive colors, this means that they subtract light (an easy example is to imagine that the wall is the source of light and each filter stops all except one color; another example is to imagine the white paper and each of the colors absorbing all except its own color.

Reference standard	K_{ry}	K_{by}
ITU601 / ITU-T 709 1250/50/2:1	0.299	0.114
ITU709 / ITU-T 709 1250/60/2:1	0.2126	0.0722
SMPTE 240M (1999)	0.212	0.087

Table 2.1: **RGB to YCbCr standard constants**

it was necessary to create a codification compatible with both systems. In the black and white television the luminance channel was already used, therefore, the newly introduced color version needed two additional chrominance channels. These three channels were called YUV (see Figure 2.9b) in the analog version and nowadays the digital version is the YCbCr (although both terms are usually used to refer to the same color space). This codification has very good properties for compression of images (see Section 2.8). YIQ is a similar codification that also separates the luminance (Y) from the chrominance (I and Q) and was used in the NTSC color TV system (see Figure 2.9d).

It is possible to perform a spatial transformation from one of the previous color spaces to the others. In the case of RGB to YCbCr the transformation follows the next equations:

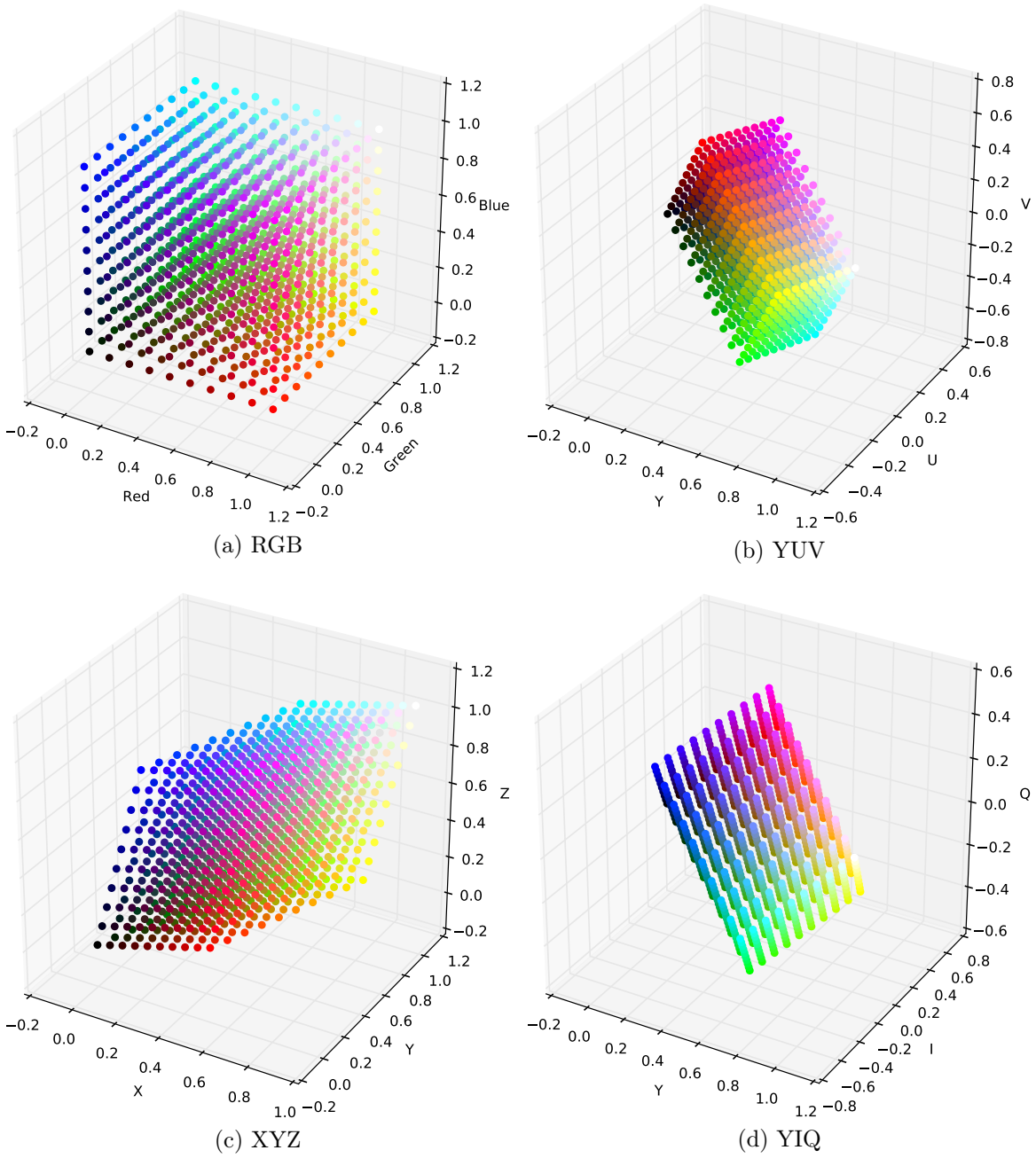
$$\begin{aligned} Y &= K_{ry}R + K_{gy}G + K_{by}B \\ C_b &= B - Y \\ C_r &= R - Y \\ K_{ry} + K_{gy} + K_{by} &= 1 \end{aligned} \tag{2.3}$$

$$\begin{aligned} Y &= K_{ry}R + K_{gy}G + K_{by}B \\ C_b &= K_{ru}R + K_{gu}G + K_{bu}B \\ C_r &= K_{rv}R + K_{gv}G + K_{bv}B \end{aligned} \tag{2.4}$$

where

$$\begin{aligned} K_{ru} &= -K_{ry} \\ K_{gu} &= -K_{gy} \\ K_{bu} &= 1 - K_{by} \\ K_{rv} &= 1 - K_{ry} \\ K_{gv} &= -K_{gy} \\ K_{bv} &= -K_{by} \end{aligned} \tag{2.5}$$

There are different standards for the values of K_{ry} and K_{by}, these are summarized in table 2.1.



(a) RGB (b) YUV

(c) XYZ (d) YIQ

Figure 2.9: **RGB represented in other transformed spaces**

Following the standard ITU601, we can express these equations using the next matrix transformation:

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix} \tag{2.7}$$

Also the matrix transformations for the XYZ and YIQ are presented here as as a reference:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.8}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.53715 & -0.498535 \\ -0.969256 & 1.875991 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.9}$$

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.595716 & -0.274453 & -0.321263 \\ 0.211456 & -0.522591 & 0.311135 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.10}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.9563 & 0.6210 \\ 1 & -0.2721 & -0.6474 \\ 1 & -1.1070 & 1.7046 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \tag{2.11}$$

2.8 The importance of luma

The luminance part of images is very important for the human visual system, as several experiments have demonstrated that our visual system is more accurate with changes in luminance than in chrominance. Exploiting this difference, video encodings usually have a larger bandwidth for luminance than chrominance, increasing the transfer speed of the signals. The same technique is used in image compression. In that case, chroma subsampling is a technique that reduces the amount of chroma information in the horizontal, vertical or both spatial dimensions. The different variations of chroma subsampling are represented with three digits in the form '$X_1 : X_2 : X_3$' where the variations of the X values have different meaning. The basic case is the original image without subsampling, represented by **4:4:4**. Then, if the horizontal resolution is reduced by half in the two chrominance channels the final image size is reduced by 2/3. This is symbolized with the nomenclature **4:2:2**. The same approach but reducing the horizontal resolution of the chrominance channels by a factor of 1/4 is symbolized with **4:1:1** and it achieves a reduction of size of 1/2. It is possible to achieve the same reduction of size also by decreasing by half the horizontal and vertical resolutions, symbolized with **4:2:0**. Table 2.2 shows a

subsample	Y	UV	YUV	factor
4:4:4	$I_v \times I_h$	$2 \times I_v \times I_h$	$3 \times I_v \times I_h$	1
4:2:2	$I_v \times I_h$	$2 \times I_v \times \frac{I_h}{2}$	$2 \times I_v \times I_h$	2/3
4:1:1	$I_v \times I_h$	$2 \times I_v \times \frac{I_h}{4}$	$\frac{3}{2} \times I_v \times I_h$	1/2
4:2:0	$I_v \times I_h$	$2 \times \frac{I_v}{2} \times \frac{I_h}{2}$	$\frac{3}{2} \times I_v \times I_h$	1/2

Table 2.2: **Chroma subsampling** vertical, horizontal and final size reduction factor

summary of these transformations with the symbolic size of the different channels of the YUV color space and the final size.

Figure 2.10 shows a visual example of the three subsampling methods previously explained in the YUV color space, and how it would look if we apply the same subsampling to the green and blue channels in the RGB color space. The same amount of information is lost in both cases (in RGB and YUV), however, if only the chrominance is reduced (YUV case) it is difficult to perceive the loss of information.

2.9 Datasets

In this section, we present a set of famous datasets that have been used from the 80s to evaluate different pattern recognition algorithms in computer vision.

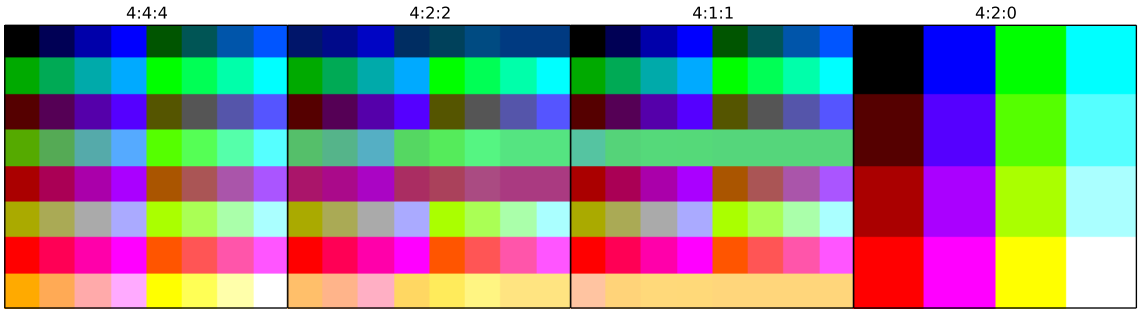
1989 In 1989, Yann LeCun created a small dataset of hand written digits to test the performance of backpropagation in Artificial Neural Networks (ANNs) [LeCun, 1989]. Later in [LeCun et al., 1989], the authors collected a large amount of samples from the U.S. Postal Service and called it the MNIST dataset. This dataset is one of the most widely used datasets to evaluate pattern recognition methods. The test error has been dropped by state-of-the-art methods to 0%, but it is still being used as a reference and to test semi-supervised or unsupervised methods where the accuracy is not perfect.

1996 The COIL Objects dataset [Nene et al., 1996] was one of the first image classification datasets of a set of different objects. It was released in 1996. The objects were photographed in gray-scale, centered in the middle of the image and were rotated by 5 degree steps throughout the full 360 degrees.

1998 From 1993 to 1997, the Defense Advanced Research Projects Agency (DARPA) collected a large number of face images to perform face recognition. The corpus was presented in 1998 with the name FERET faces [Phillips and Moon, 2000].

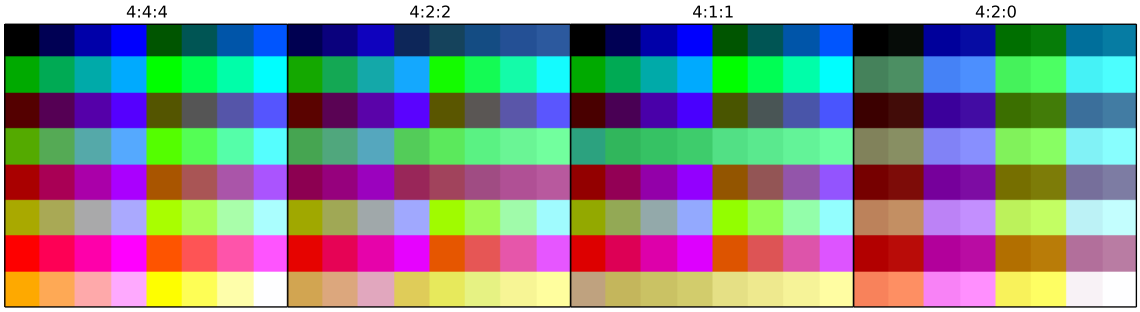
1999 In 1999, researchers of the computer vision lab in California Institute of Technology collected pictures of cars, motorcycles, airplanes, faces, leaves and backgrounds to create the first Caltech dataset. Later they released the Caltech-101 and Caltech-256 datasets.

2001 In 2001, the Berkeley Segmentation Data Set (BSDS300) was released for grouping, contour detection, and segmentation of natural images [Martin and Fowlkes, 2001]. This dataset contains 200 and 100 images for training and testing with 12.000 hand-labeled segmentations. Later, in [Arbelaez and Maire, 2011], the au-



(a) Subsampling in RGB color space

(b) Subsampling in RGB color space



(c) Subsampling in YUV color space

(d) Subsampling in YUV color space

Figure 2.10: **Comparison of different subsampling on RGB and YUV channels** - Although the chroma subsampling technique is only applied on YUV colors we apply the subsampling to Green and Blue channels on RGB codification only as a visual comparison. While in RGB channels applying this subsampling highly deteriorates the image, it is almost imperceptible when applied on chroma channels.

thors extended the dataset with 300 and 200 images for training and test; in this version the name of the dataset was modified to BSDS500.

In 2002, the Middleburry stereo dataset [Scharstein and Szeliski, 2002] was released. This is a dataset of pairs of images taken from two different angles and with the objective of pairing the couples. 2002

In 2003, the Caltech-101 dataset [Fei-Fei et al., 2007] was released with 101 categories distributed through 9.144 images of roughly 300×200 pixels. There are from 31 to 900 images per category with a mean of 90 images. 2003

In 2004, the KTH human action dataset was released. This was a set of 600 videos with people performing different actions in front of a static background (a field with grass or a white wall). The 25 subjects were recorded in 4 different scenarios performing 6 actions: walking, jogging, running, boxing, hand waving and hand clapping. 2004

The same year, a collection of 1.050 training images of cars and non-cars was released with the name UIUC Image Database for Car Detection. The dataset was used in two experiments reported in [Agarwal et al., 2004; Agarwal and Roth, 2002].

In 2005, another set of videos was collected this time from surveillance cameras. The team of the CAVIAR project captured people walking alone, meeting with another person, window shopping, entering and exiting shops, fighting, passing out and leaving a package in a public place. The image resolution was of 384×288 pixels and 25 frames per second. 2005

In 2006, the Caltech-256 dataset [Griffin et al., 2007] was released increasing the number of images to 30.608 and 256 categories. There are from 80 to 827 images per category with a mean of 119 images. 2006

In 2008, a dataset of videos of sign language [Athitsos et al., 2008] was released with the name American Sign Language Lexicon Video Dataset (ASLLVD). The videos were captured from three angles: front, side and face region. 2008

In 2009, one of the most known classification and detection datasets was released with the name of PASCAL Visual Object Classes (VOC) [Everingham and Gool, 2010; Everingham and Eslami, 2014]. Composed of 14.743 natural images of people, animals (bird, cat, cow, dog, horse, sheep), vehicles (aeroplane, bicycle, boat, bus, car, motorbike, train) and indoor objects (bottle, chair, dining table, potted plant, sofa, tv/monitor). The dataset is divided into 50% for training and validation and 50% for testing. 2009

In the same year, a team from Toronto University collected the CIFAR-10 dataset (see Chapter 3 of [Krizhevsky and Hinton, 2009]). This dataset contained 60.000 color images of 32×32 pixel size divided uniformly between 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck). Later, the number of categories was increased to 100 in the extended version CIFAR-100.

ImageNet dataset was also released in 2009 [Deng et al., 2009]. This dataset collected images containing nouns appearing in WordNet [Miller, 1995; Fellbaum, 1998] (a large lexical database of English). The initial dataset was composed by 5.247 synsets and 3.2 million images whilst in 2015 it was extended to 21.841 synsets and 14.197.122 images. 2015

Chapter 3

Neuro vision

"He believes that he sees his retinal images: he would have a shock if he could."

— George Humphry

This Chapter gives a brief introduction about the neural anatomy of the visual system. We hope that by understanding the biological visual system, it is possible to get inspiration and design better algorithms for image classification. This approach has been successfully used in the past in computer vision algorithms; for example, the importance of blob-like features, or edge detectors. Additionally, some of the state-of-the-art Artificial Neural Network (ANN) were originally inspired by the cats and macaques visual system. We also know that humans are able to extract very abstract representations of our environment by interpreting the different light wavelengths that excite the photoreceptors in our retinas. As the neurons propagate the electric impulses through the visual system our brain is able to create an abstract representation of the environment [Fischler, 1987] (the so-called "signals-to-symbol" paradigm).

Vision is one of the most important senses in human beings, as roughly the 60% of our sensory inputs belong to visual stimulus. In order to study the brain we need to study its morphology, physiology, electrical responses, and chemical reactions. These characteristics are studied in different disciplines, however, they need to be merged to solve the problem of understanding how the brain is able to create visual representations of the surrounding. For that reason, the disciplines that are involved in neuro vision extend from neurophysiology to psychology, chemistry, computer science, physics and mathematics. In this chapter, first we introduce the biological neuron, one of the most basic components of the brain in Section3.1. We give an overview of the structure and the basics of their activations. Then, in Section 3.2, we explain the different parts of the brain involved in the vision process, the cells, and the visual information that they proceed. Finally, we present a summary about the color perception, and the LMS color space in Section 3.3. For an extended introduction to the human vision and how it is related to ANN see [Gupta and Knopf, 1994].

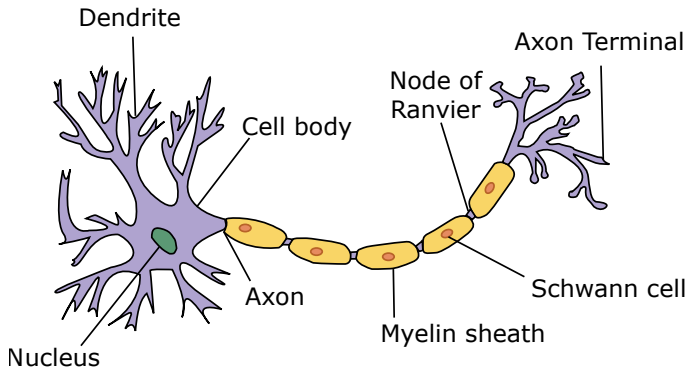


Figure 3.1: **Simplified schema of a biological neuron** [1]

3.1 The biological neuron

Neurons are cells that receive and send electrical and chemical responses from – and to – other cells. They form the nervous system, having the responsibility of sensing the environment and reacting accordingly. The brain is almost entirely formed by neurons, that together, generate the most complex actions. Humans have around 86 billion of neurons in all the nervous system. Each neuron is divided basically by three parts: dendrites, soma and axons. Figure 3.1 shows a representation of a biological neuron with all the components explained in this section.

The dendrites are ramifications of the neuron that receive the action potentials from other neurons. The dendrites are able to decrease or increase their number – and strength – when their connections are reiteratively activated (this behaviour is known as Hebbian learning). Thanks to the dendrites, one neuron in the cortex can receive on average thousands of connections from other neurons.

The soma (or cell body) is the central part of the neuron, and receives all the excitatory and inhibitory impulses from the dendrites. All the impulses are aggregated on time, if the sum of impulses surpass the Hillock threshold – in a short period of time – the neuron is excited and activates an action potential that traverses its axons.

The axons are the terminals of the neurons that start in the axon hillock. In order to transmit the action potential from the nucleus of the neuron to the axon terminals a series of chemical reactions creates an electrical cascade that travels rapidly to the next cells. However, some of the axons on the human body can extend approximately 1 meter, and to increase the speed of the action potentials, the axons are sometimes covered by Schwann cells. This cells isolate the axons with Myelin sheath, making the communication more robust and fast.

[1] Originally Neuron.jpg taken from the US Federal (public domain) (Nerve Tissue, retrieved March 2007), redrawn by User:Dhp1080 in Illustrator. Source: "Anatomy and Physiology" by the US National Cancer Institute's Surveillance, Epidemiology and End Results (SEER) Program.

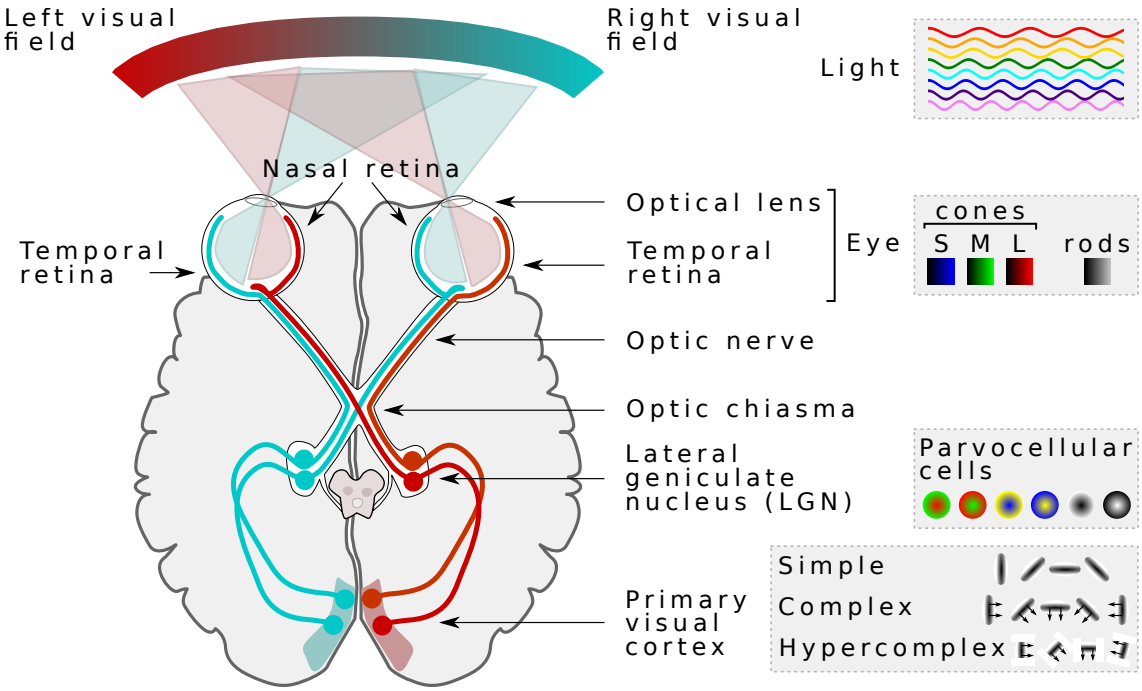


Figure 3.2: **Visual pathway**

3.2 Visual system

The visual system is formed by a huge number of nervous cells that send and receive electrical impulses trough the visual pathway. The level of abstraction increases as the signals are propagated from the retina to the different regions of the brain. At the top level of abstraction there are the so-called "grandmother-cells" [Gross, 2002], these cells are specialized on detecting specific shapes, faces, or arbitrary visual objects. In this section, we describe how the different wavelengths of light are interpreted and transformed at different levels of the visual system. Starting in the retina of the eyes in Section 3.2.1, going through the optic nerve, the optic chiasma, and the Lateral Geniculate Nucleus (LGN) – allocated in the thalamus – in Section 3.2.2, and finally the primary visual cortex in Section 3.2.3. Figure 3.2 shows a summary of the visual system and the connections from the retina to the primary visual cortex. In addition, at the right side, we show a visual representation of the color responses perceived at each level of the visual pathway.

3.2.1 The retina

The retina is a thin layer of tissue located in the inner and back part of the eye. The light crosses the iris, the optical lens and the eye, and finally reaches the photoreceptor cells. There are two types of photoreceptor cells: cone and rod cells.

Cone cells are cone shaped and are mostly activated with high levels of light. There are three types of cone cells specialized on different wavelength: the short cells are mostly activated at 420nm corresponding to a bluish color, the medium cells at

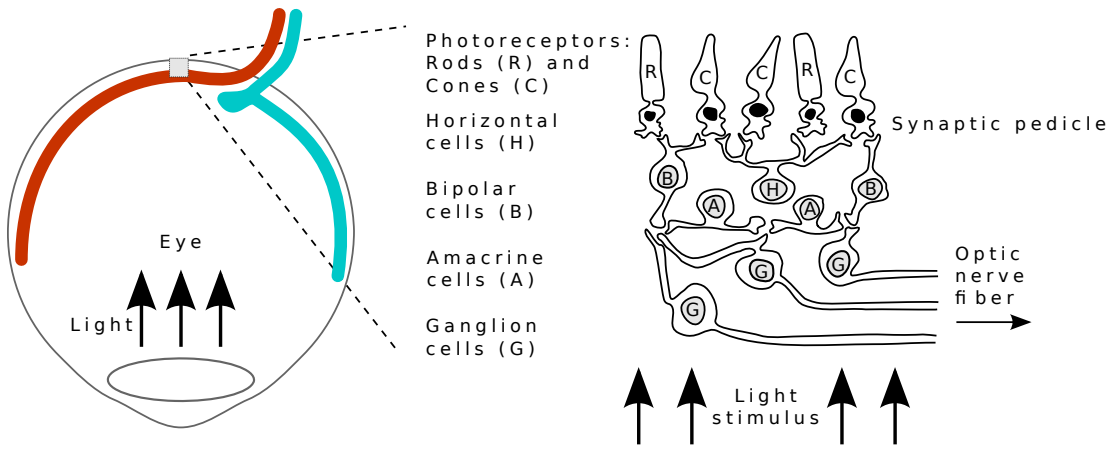
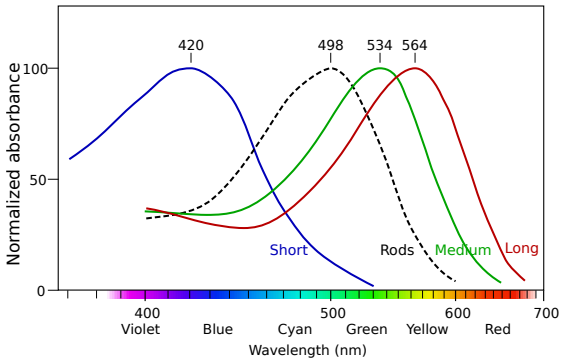
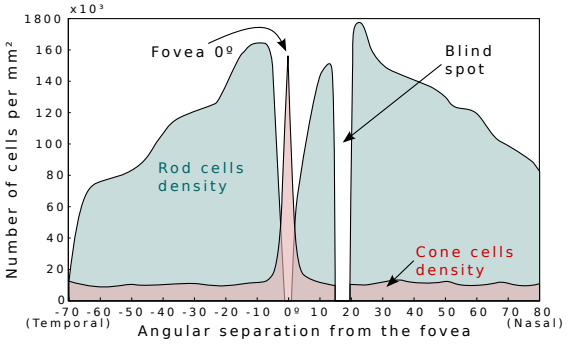


Figure 3.3: **A cross section of the retina** with the different cells (the figure has been adapted from [Gupta and Knopf, 1994]



(a) Human visual color response. The dashed line is for the rod photoreceptor while red, blue, and green are for the cones.



(b) Rod and cone densities

Figure 3.4: **Retina** photoreceptors

534nm corresponding to a greenish color, and the long cells at 564nm corresponding to a reddish color (see Figure 3.4a). These cells need large amounts of light to be fully functional. For that reason, cone cells are used in daylight or bright conditions. Their density distribution on the surface of the retina is almost limited to an area of 10 degrees around the fovea (see Figure 3.4b). For that reason, we can only perceive color in a small area around our focus of attention – the perception of color in other areas is created by more complex systems. There are about 6 million cone cells, from which 64% are long cells, 32% medium cells and 2% short cells. Although the number of short cells is one order of magnitude smaller than the rest, these cells are more sensitive and being active in worst light conditions, additionally, there are some other processes that compensate this difference. That could be a reason why in dark conditions – when the cone cells are barely activated – the clearest regions are perceived with a bluish tone.

Rod cells do not perceive color but the intensity of light, their maximum response corresponds to 498nm, between the blue and green wavelengths. They are functional only on dark situations – usually during the night. They are very sensitive, some studies (e.g. [Goldstein, 2013]) suggest that a rod cell is able to detect an isolated photon. There are about 120 million of rod cells distributed trough all the retina, while the central part of the fovea is exempt of them (see Figure 3.4b). Additionally, the rod cells are very sensitive to motion and peripheral vision. Therefore, in dim situations, it is easier to perceive movement and light in the periphery but not in our focus of attention.

However, the photoreceptors are not connected directly to the brain. Their axons are connected to the dendrites of other retinal cells. There are two types of cells that connect the photoreceptors to other regions: the horizontal and bipolar cells.

Horizontal cells connect neighborhoods of cone and rod cells together. This allows the communication between photoreceptors and bipolar cells. One of the most important roles of horizontal cells, is to inhibit the activation of the photoreceptors to adjust the vision to bright and dim light conditions. There are three types of horizontal cells: HI, HII, and HIII; however their distinction is not clear.

Bipolar cells connect photoreceptors to ganglion cells. Additionally, they accept connections from horizontal cells. Bipolar cells are specialized on cones or rods. There are nine types of bipolar cells specialized in cones, while only one type for rod cells. Bipolar and horizontal cells work together to generate center surrounded inhibitory filters with a similar shape of a Laplacian of Gaussian (LoG) or Difference of Gaussian (DoG). These filters are focused in one color tone in the center and its opposite at the bound.

Amacrine cells connect various ganglion and bipolar cells. Amacrine cells are similar to horizontal cells, they are laterally connected and most of their responsibility is the inhibition of certain regions. However, they connect the dendrites of ganglion cells and the axons of bipolar cells. There are 33 subtypes of amacrine cells, divided by their morphology and stratification.

Finally, **ganglion cells** receive the signals from the bipolar and amacrine cells, and extend their axons trough the optic nerve and optic chiasma to the thalamus, hypothalamus and midbrain (see Figure 3.2). Although there are about 126 millions of photoreceptors in each eye, only 1 million of ganglion axons travel to the different parts of the brain. This is a compression ratio of 126:1. However, the action potentials generated by the ganglion cells are very fast, while the cones, rods, bipolar, horizontal and amacrine cells have slow potentials. Some of the ganglion cells are specialized on the detection of centered contrasts, similar to the Laplacian of Gaussian (LoG) or a Difference of Gaussian (DoG) (see these shapes in Figures 2.5 and 2.6). The negative side is achieved by inhibitory cells and can be situated in the center or at the bound. The six combinations are yellow and blue, red and green, and bright and dark, with one of the colors in the center and the other on the bound. There are three groups of ganglion cells: W-ganglion, X-ganglion and Y-ganglion. However, groups of the three types of ganglion cells are divided in five classes depending on the region where their axons end: midget cells project to the Parvocellular cells (P-cells), parasol cells project to the Magnocellular cells (M-

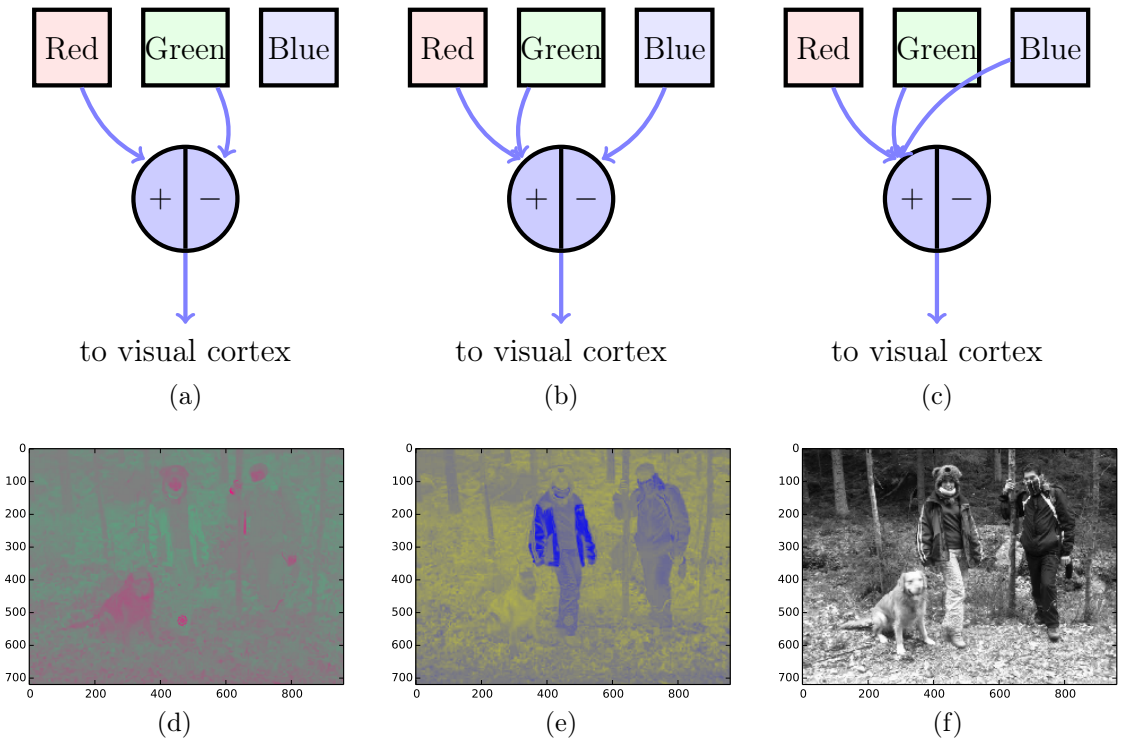


Figure 3.5: **Extitation of parvocellular cells** - The three different types of parvocellular cells; or P-cells.

cells), bistratified cells project to the Koniocellular cells (K-cells) (P-cells, M-cells and K-cells are situated in the Lateral Geniculate Nucleus (LGN)), photosensitive ganglion cells and other cells project to the superior colliculus.

3.2.2 The lateral geniculate nucleus

The ganglion cells from the retina extend around 1 million axons per eye through the optic nerve, and coincide in the optic chiasma. From the chiasma, the axons corresponding to the left and right visual field take different paths towards the two Lateral Geniculate Nuclei (LGNs) in the thalamus. Every LGN is composed of 6 layers and 6 strata. One stratum is situated before the first layer while the rest lie between every pair of layers. There are 3 specialized types of cells that connect to different layers and strata.

Koniocellular cells (K-cells) are located in the stratas between the M-cells and P-cells, composing 5% of the cells in the LGN.

Magnocellular cells (M-cells) are located in the first two layers (1st and 2nd) composing the 5% of the cells in the LGN. These cells are mostly responsible on the detection of motion.

Parvocellular cells (P-cells) are located in the last four layers (from 3th to 6th) and contribute to the 90% of the cells in the LGN. These cells are mostly responsible of color and contrast, and are specialized in red-green, blue-yellow and bright-dark

differences. Figure 3.5 shows a toy representation of the different connections that P-cells perform and a visual representation of their activation when looking at a scene.

3.2.3 The primary visual cortex

Most of the connections from the LGN go directly to the primary visual cortex. In the primary visual cortex, the neurons are structured in series of rows and columns of neurons. The first neurons to receive the signals are the simple cortical cells. These cells detect lines at different angles – from horizontal to vertical – that occupy a large part of the retina. While the simple cortical cells detect static bars, the complex cortical cells detect the same bars with motion. Finally, the hypercomplex cells detect moving bars of certain length.

All the different bar orientations are separated by columns. Later, all the connections extend to different parts of the brain where more complex patterns are detected. Several studies have demonstrated that some of these neurons are specialized in the detection of basic shapes like triangles, squares or circles, while other neurons are activated when visualizing faces or complex objects. These specialized neurons are called "grandmother cells". The work of Uhr [Uhr, 1987] showed that humans are able to interpret a scene in 70 to 200 ms performing 18 to 46 transformation steps.

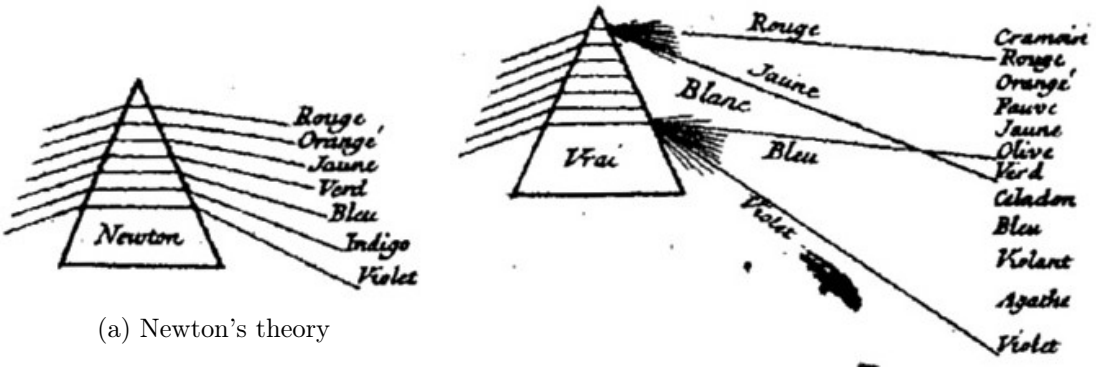
3.3 LMS color space and color perception

The LMS color space corresponds to the colors perceived with the excitation of the different cones in the human retina: the long (L), medium (M) and short (S) cells. However, the first experiments about the color perception in humans started some hundreds of years ago, when the different cells in our retina were unknown.

From 1666 to 1672, Isaac Newton[2] developed some experiments about the diffraction of light into different colors. Newton was using glasses with prism shape to refract the light and project the different wavelengths into a wall. At that moment, it was thought that the blue side corresponded to the darkness, while the red side corresponded to the brightness. However, Newton already saw that passing the red color through the prism did not change its projected color, meaning that light and red were two different concepts. Newton demonstrated that the different colors appeared because they traveled at different speeds and got refracted by different angles. Also, Newton demonstrated that it was possible to retrieve the initial light by merging the colors back. Newton designed the color wheel, and it was used later by artist to increase the contrast of their paintings (more information about the human perception and one study case about color vision can be seen in [Churchland and Sejnowski, 1988])

1666

[2]**Isaac Newton** (Woolshorpe, Lincolnshire, England; December 25, 1642 – March 20 1726) was a physicist and mathematician, widely know by his contributions in classical mechanics, optics, motion, universal gravitation and the development of calculus.



(a) Newton's theory

(b) Castel's theory

Figure 3.6: **The theory of Newton and Castel about the color formation** (Figure from Castel's book "L'optique des couleurs" [Castel, 1740])

1810 Later, in 1810, Johann Wolfgang von Goethe[3] got published the book Theory of Colours (in German "Zur Farbenlehre"). In his book, Goethe focused more in the human perception from a psychologist perspective, and less about the physics (Newton's approach). The author tried to point that Newton theory was not completely correct and shown that depending on the distance from the prism to the wall the central color changed from white to green. This idea was not new as Castel's book "L'optique des couleurs" from 1740 already criticised the work of Newton and 1740 proposed a new idea (see Figure 3.6). Both authors claimed that the experiments performed by Newton were a special case, in which the distance of the projection was fixed. Based on the experiments, they proposed that the light was separated into two beams of red-yellow and blue-cyan light in a cone shape and the overlapped region formed the green.

Goethe created a new colour wheel; in this case symmetric. In this wheel, the opponent colors were situated at opposite sides, anticipating the Opponent process theory of Ewald Hering.

1802 In 1802, T. Young [Young, 1802] proposed that the human retina should have three different photoreceptors, as there are three primary colors.

1892 In 1892, Ewald Hering proposed the opponent color theory, in which the author showed that yellow and blue, and red and green are opposites. Also, the combination of one pair can not generate another hue; there is no bluish-yellow or reddish-green. Figure 3.8 shows a representation of the intersection of the opponent colors and two examples where the intersection can be perceived as an additional color.

The two initial ideas were: either we have three photoreceptors for the primary colors, or the red-green, blue-yellow and white-black are the basic components of 1986 the human color perception. Later, physiologists [Nathans et al., 1986] discovered that both theories were correct in different parts of the brain. The LMS color space

[3] **Johann Wolfgang von Goethe** (Frankfurt-am-Main, Germany; August 28, 1749 – March 22, 1832) was a writer, poet, novelist and natural philosopher. He also did several studies about natural science some of them focused on color.

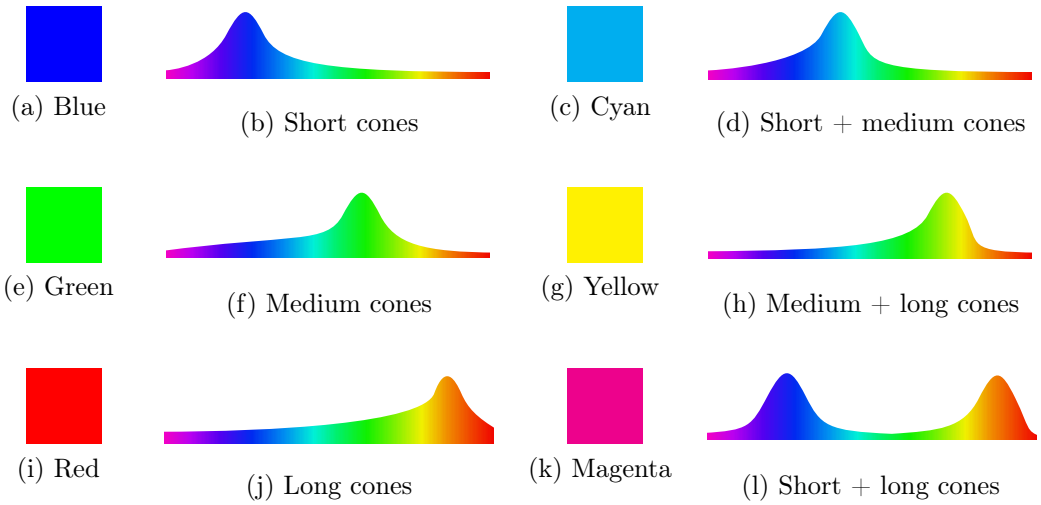


Figure 3.7: **Human perception of the primary colors** in the activation of the retinal cone cells.

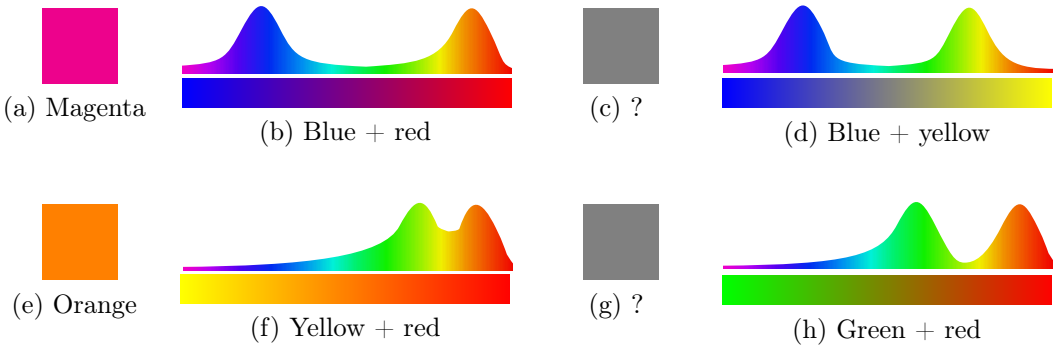


Figure 3.8: **Human perception of opponent colors** in the activation of the retinal cone cells, the center of the bars contain a color mixture of both sides at 50%. In the intersection of opposite colors there is no bluish yellow or redish green.

was detected in the retina, while the opponent process was performed in the LGN region of the thalamus.

Chapter 4

Artificial Neural Networks

"My mind seems to have become a kind of machine for grinding general laws out of large collections of facts"

— Charles Darwin

In this chapter, we give a brief introduction to Artificial Neural Network (ANN) and show how they can be used to solve pattern recognition problems. In the previous Chapter 2, we saw that image classification algorithms require a set of feature descriptors that best represents the classes and that discriminates between different categories. With the set of features, a classification algorithm can study the statistical distribution of each class and finally classify the images. In general, the features are designed in a preprocessing step, and later, their performance is validated by using a variety of datasets.

The design of the features is commonly laborious, and frequently hand designed. For that reason, it is difficult to find features that can be useful in a variety of situations, while sometimes it is impossible to find descriptors if these are counter-intuitive. For that reason, this approach does not scale to larger problems or tasks that are continuously changing.

On the other hand, ANN can solve the previous problems by learning automatically the set of feature representations that are good for the classification task, given a large number of image samples. Both parts of the problem are learned and solved simultaneously and ANNs have demonstrated to outperform other techniques in a large range of classification and regression problems.

We start the chapter by explaining what is an artificial neuron in Section 4.1. These neurons can be excited in different ways, and we simulate these activities by adding a variety of activation functions in Section 4.2. Then, we show the simplest architecture that is possible to create with a single neuron in Section 4.3. Three examples of simple networks are presented: one architecture to solve linear regressions in Section 4.3.1 and two architectures to solve pattern recognition tasks in Sections 4.3.2 and 4.3.3.

After explaining the simplest models, we extend the basic model with three architectures that are able to solve very complex tasks: in Section 4.4 we present

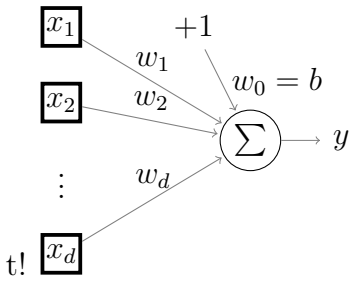


Figure 4.1: **Artificial neuron** computing the sum of their weighted inputs and bias term.

the Multilayer Feedforward Neural Network (MFNN), in Section 4.6 the Extreme Learning Machine (ELM) and in Section 4.7 a Recurrent Neural Network (RNN).

Once we shown a variety of architectures, we explain how to train an ANN in Section 4.5. Also, we explain how to control the potential of ANNs using regularization techniques in Section 4.5.5.

Finally, we present a small discussion about the new term deep learning that has been gaining attention in the machine learning community in Section 4.8

4.1 The artificial neuron

In this section we present the basic element of all ANNs: the neuron (often called in the context of machine learning a unit). Although this is not a biological neuron, we will use this term in all this chapter to simplify the reading (to see a description of the real biological neuron see Section 3.1). Similarly to biological neurons, artificial neurons receive signals trough the connections from other neurons. Depending on the strength of their input connections the neurons respond with more or less intensity to the incoming signals. The intensity is represented mathematically as a set of positive or negative factor weights $\mathbf{w}$ that amplifies or reduces the signals. The neuron also incorporates a bias b that increases or decreases the normal activity of the neuron. This description can be represented with the next equation:

$$a(\mathbf{x}) = b + \sum_{i=1}^{D} w_i x_i = b + \mathbf{w}^T \mathbf{x} \tag{4.1}$$

Where D is the number of incoming neurons and $\mathbf{x}$ is the activity of each neuron, in this case the $\mathbf{x}$ can represent the set of features of one sample. In order to simplify the equations it is common to add the bias term as another input with the fixed value of $+1$, and incorporate one weight that acts as bias. Using this simplification, we rewrite the summation 4.1 with the next simplified version.

$$a(\mathbf{x}) = w_0 1 + \sum_{i=1}^{D} w_i x_i = \sum_{i=0}^{D} w_i x_i = \mathbf{w}^T \mathbf{x} \tag{4.2}$$

Other types of ANNs compute the distance – instead of the scalar product – between the weights $\mathbf{w}$ and the input pattern $\mathbf{x}$, for example by computing $\| \mathbf{w}^T - \mathbf{x} \|^2$ (e.g. the Radial Basis Function (RBF) or the Self-Organizing Map (SOM)). In this case, activations with values close to zero are more similar to the training patterns. In this chapter, we focus on the version with the scalar product, however, the explanations can be extended to the versions of the distance with small modifications.

Additionally, it is possible to change the linear response of the neuron by adding an activation function $h(\cdot)$ after the summation:

$$z = h(a(\mathbf{x})) = h(\mathbf{w}^T \mathbf{x}) \tag{4.3}$$

Being able to change the activation function increases the range of possible patterns that the neuron can approximate. In the most basic case the function $h(\cdot)$ performs the identity function while more complex cases use non-linear functions. We will see that multiple hidden neurons using non-linear activation functions can approximate any pattern; given the sufficient number of neurons. For that reason, ANN are known to be universal approximators [Hornik et al., 1989].

4.2 Activation function

There is a large variety of activation functions that have demonstrated good performance on different tasks. Initially, when McCulloch and Pitts [McCulloch and Pitts, 1943] designed the first ANN only one activation was specified. As they were modeling biological neurons they choose an "all-or-none" law by using a Threshold Function; also referred as Heaviside function in engineering literature (Figure 4.2a). This function evaluates to one when the input is positive, and zero when the input is negative (the behaviour at the zero is commonly indifferent):

$$h(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{if } a < 0 \end{cases} \tag{4.4}$$

However, nowadays some of the most common activation functions are sigmoidal functions (functions with "S" shape). The reason is their behaviour at the "center". These functions can behave as linear or as step function depending on the parameters of the network. One example of sigmoidal function is the logistic function (see Figure 4.2f):

$$h(a) = \frac{1}{1 + \exp(-a)} \tag{4.5}$$

Also, the hyperbolic tangent is commonly used (see Figure 4.2e):

$$h(a) = \tanh(a) \tag{4.6}$$

The rectified linear unit (ReLU) was a fundamental part for the training of deep networks for image classification (see Figure 4.3e); specially in the case of CNNs. ReLU behaves as a linear function if the input is positive, and equals zero

otherwise. One of the benefits of this activation function is that it creates sparse hidden representations, that demonstrated to benefit the training. While training networks with other activation functions creates more complicated gradients.

$$h(a) = \begin{cases} a & \text{if } a \geq 0 \\ 0 & \text{if } a < 0 \end{cases} \tag{4.7}$$

More recently, a new form of rectification called Parametric Rectified Linear Unit (PReLU) improved the best results on ImageNet 2012 classification dataset. In the work [He et al., 2015], instead of forcing the negative activations to become completely zero, the authors added one parameter to control the steepness s on the negative side of a ReLU, allowing the left side to become zero or to adapt to a linear slope.

$$h(a) = \begin{cases} a & \text{if } a \geq 0 \\ as & \text{if } a < 0 \end{cases} \tag{4.8}$$

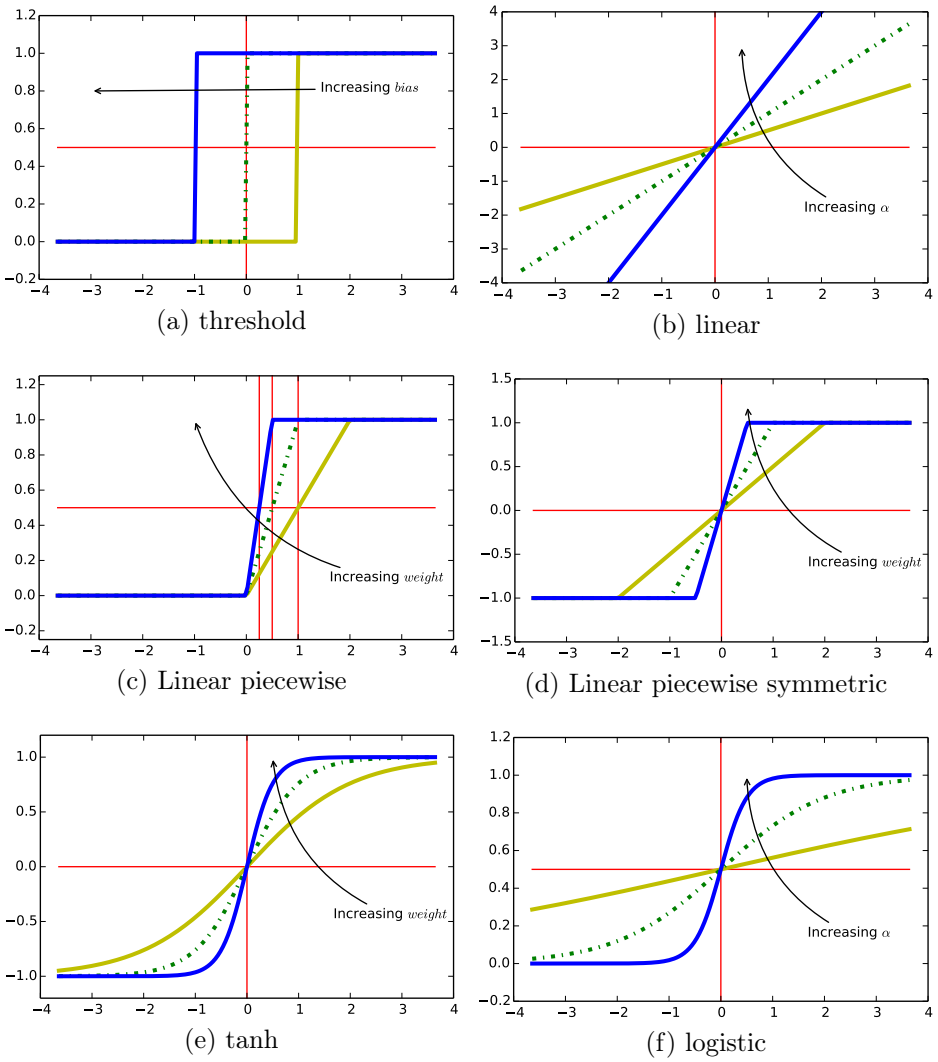
Figures 4.2 and 4.3 present a variety of activation functions and how they behave differently when some of the parameters are modified. Additionally, Table 4.1 shows some of the most common activation functions with their derivatives (this table can be useful for the training phase).

Activation function name	**function** $h(a)$	**derivative** $h'(a)$
Identity	a	1
Logistic sigmoid	$\frac{1}{1+e^{-a}}$	$h(a)(1-h(a))$
Hyperbolic tangent	$\tanh(a) = \frac{\sinh(a)}{\cosh(a)}$	$1-\tanh^2(a)$
Softsign	$\frac{a}{1+\lvert a\rvert}$	$\frac{1}{(1+\lvert a\rvert)^2}$
Sin	$\sin(a)$	$\cos(a)$

Table 4.1: **Activation function derivatives**

4.3 Single layer feed-forward neural network

In Section 4.1, we have seen the capabilities of one isolated neuron. Additionally, it is possible to aggregate different neurons to create more complex functions. These neurons can connect each other forming different architectures. In this section, we focus in the most basic architectures connecting the input features to one layer of neurons. With one single layer and with different activation functions it is possible to approximate a variety of functions. In the most basic case – with only one neuron and the linear activation function – we can perform a linear regression (see Section 4.3.1). Furthermore, the linear regression is not limited to predict one unique output signal, instead it could be formed by N neurons each one predicting some specific target output.



(a) threshold (b) linear

(c) Linear piecewise (d) Linear piecewise symmetric

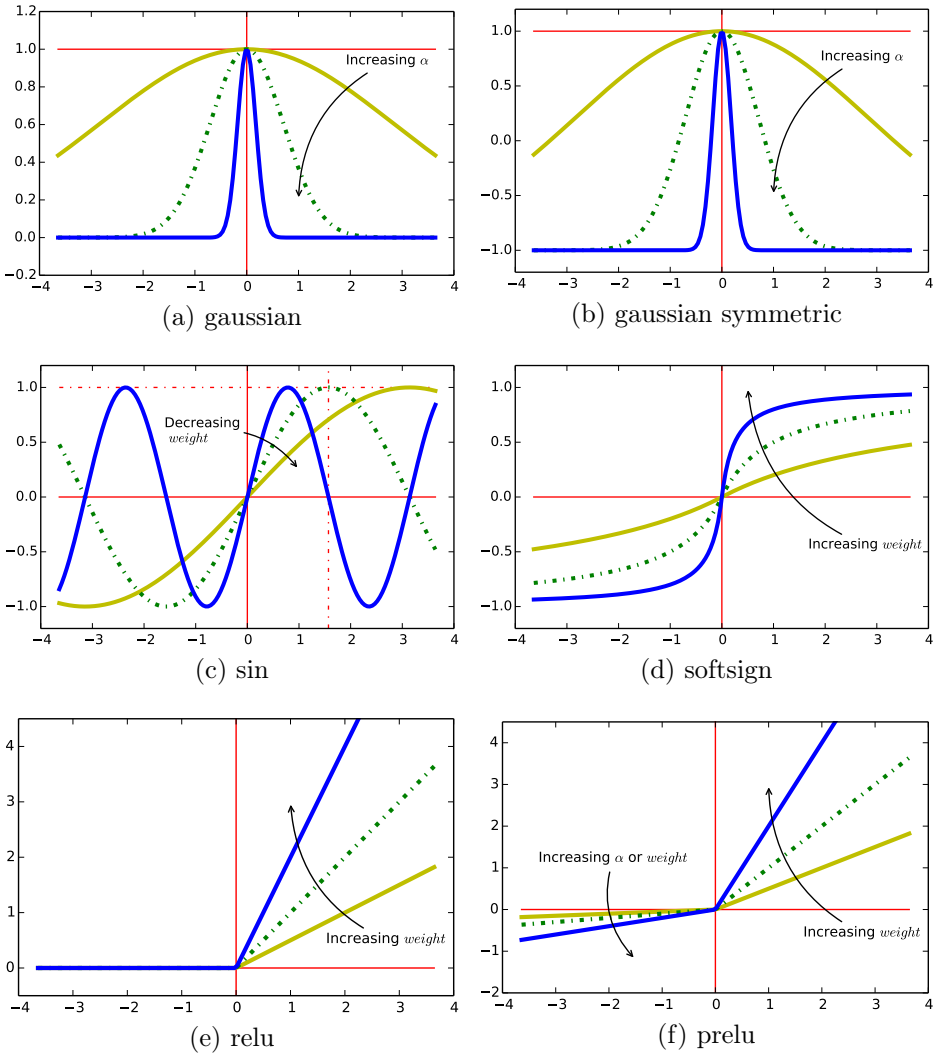
(e) tanh (f) logistic

Figure 4.2: **Example of activation functions (A)**

Another possibility is to use the last layer for pattern classification (see Sections 4.3.2 and 4.3.3). In this case, sigmoid activation function is commonly used. When the target is the classification of a pattern into multiple classes the output of all the neurons is normalized using Softmax, offering a probability distribution trough all the possible categories.

4.3.1 Linear regression

One of the most simple tasks can be performed with one unique neuron. We previously saw that one neuron can have one weight w_i per input x_i plus one bias term b. When only one dependent variable t is being predicted with only one explanatory variable x, the neuron can solve a simple linear regression. However, it can perform a multiple linear regression when more than one explanatory variables $\mathbf{x}$ are consid-



(a) gaussian (b) gaussian symmetric

(c) sin (d) softsign

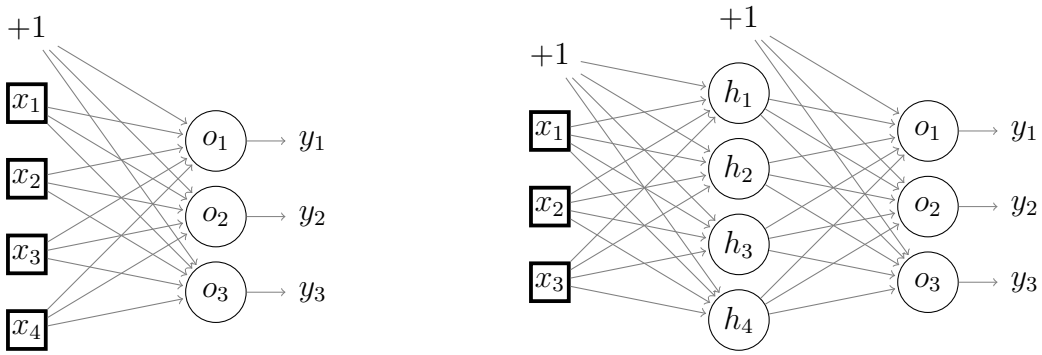
(e) relu (f) prelu

Figure 4.3: **Example of activation functions (B)**

ered as inputs. In this scenario and given that the bias is added as an additional weight the regression can be solved by the Least Squares method, which looks for the set of parameters $\mathbf{w}$ that minimizes the error between the prediction $y(\mathbf{x})$ and the target t.

$$\cap W = \arg\min_{w}(t - y(\mathbf{x}))^2 = \arg\min_{w}(t - \mathbf{w}^T\mathbf{x})^2 \tag{4.9}$$

Additionally, ANNs with one output layer can predict multiple dependent variables or outputs $\mathbf{t}$ at the same time. This type of regression is known as a multivariate linear regression, and is the same as performing various multiple linear regressions in parallel. It can be implemented by just augmenting the target variable t with a vector of targets $\mathbf{t}$ and the vector $\mathbf{w}$ by a matrix $\mathbf{W}$.



(a) Single layer (b) One hidden layer

Figure 4.4: **Single and Multilayer feedforward neural networks** (a) Example of a Single layer Feed-forward neural network (FNN) with 4 input features and 3 output units. (b) Example of a MFNN with 3 input features, 4 hidden units and 3 output units.

4.3.2 Perceptron

The Perceptron was one of the first neural networks that was able to classify binary patterns. Rosenblatt designed the neural network with several input nodes and one output node with a Heaviside step function 4.4. It was possible to classify binary patterns by finding a linear hyperplane that separated both classes on the positive and negative side. However, at the beginning, there was no algorithm to learn the set of correct parameters. Nevertheless, in 1958 Rosenblatt got published a report [Rosenblatt, 1958] on how to train the Perceptron using a set of pattern examples. Training consisted on testing the performance of the Perceptron on all the samples until all the patterns were correctly classified. At each prediction, if the target was wrongly classified the weights were updated with the following equation:

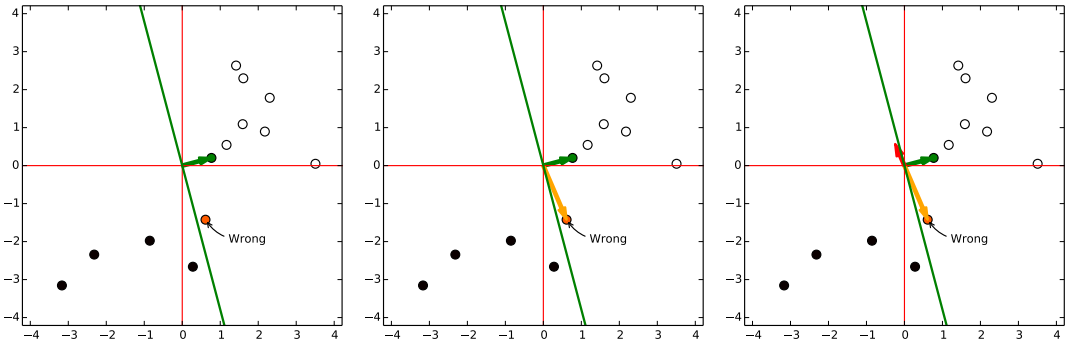
$$\mathbf{w}_{t+1} = \mathbf{w}_t + 2\alpha t\mathbf{x} \tag{4.10}$$

The same equation could be modified to be applied after each prediction (besides it was correctly or wrongly classified). In this case, when the prediction was right, the term $(t - y)$ was canceled and did not modify the weights:

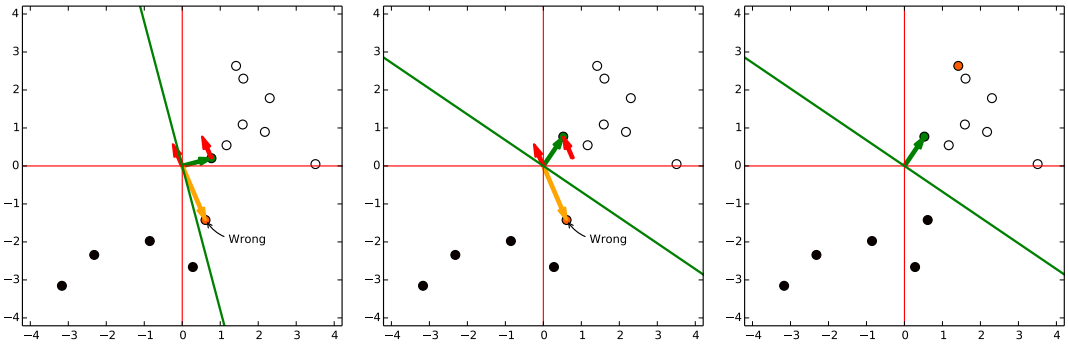
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha(t - y)\mathbf{x} \tag{4.11}$$

In both cases, the term α is a learning rate that is possible to tune for the specific task, t is the target value, y is the prediction of the network, and w_t are the weights at time-step t.

Figure 4.5 shows a training example of a Perceptron, in which, one sample is being misclassified as a false positive. In order to simplify the demonstration the parameters do not have the bias term, but just two weights; one per input. The weights are represented by the green vector and the green dot at the tip of the arrow. The orthogonal green line represents the hyperplane that separates both predictions



(a) False positive (b) Vectorial representation (c) Scale and invert



(d) Sum to the weights (e) Result of sum (f) Now is a true negative

Figure 4.5: **Example of Perceptron training** This is a toy example where a Perceptron tries to classify between black and white circles. The green dot represents the weight vector. The orthogonal green line corresponds to the hyperplane that separates the prediction between the positive and negative sides. The orange dot is the sample being analyzed, in the first figure the orange dot corresponds to a black point that is being missclassified as white. An extended description of the example can be found in the text.

between the positive and the negative side (predicts white and black circles respectively). In Figure 4.5a, the orange dot is being analyzed and it corresponds to a black dot that is being misclassified by the hyperplane. Figure 4.5b shows the vector associated to the misclassified sample in orange. Then, in Figure 4.5c, as the target class is negative, the corresponding vector is inverted and it is scaled by the learning rate α producing a smaller red vector in the opposite direction. In Figure 4.5d this vector is added to the weights vector and the result of the update is shown in Figure 4.5e. Finally, in Figure 4.5f the Perceptron classifies correctly all the samples. If some of the samples were still misclassified the same procedure could be continued. If the points are linearly separable this algorithm was demonstrated to converge to a correct solution; given a sufficiently small learning rate.

4.3.3 Logistic regression

ANNs can perform a logistic regression to classify patterns into two – or more – classes. This can be achieved in a similar way to the Perceptron but using a sigmoid function as an activation function. In this case, we do not assume Gaussian noise in the output distribution, but the dependent variable is assumed to follow a Bernoulli distribution. The output neuron usually performs a logistic function or a hyperbolic tangent function after its activation.

To perform a multi-class classification, the same model can be extended by adding one output per each class. To predict the class, the output of each neuron is usually normalized with a softmax function (also known as normalized exponential). After the normalization the output of each neuron remains in the interval $(0, 1)$ and all the outputs sum to one, creating a probability distribution over the different categories.

$$y(\mathbf{a})_o = \frac{e^{a_o}}{\sum_{o=1}^{O} e^{a_o}} \tag{4.12}$$

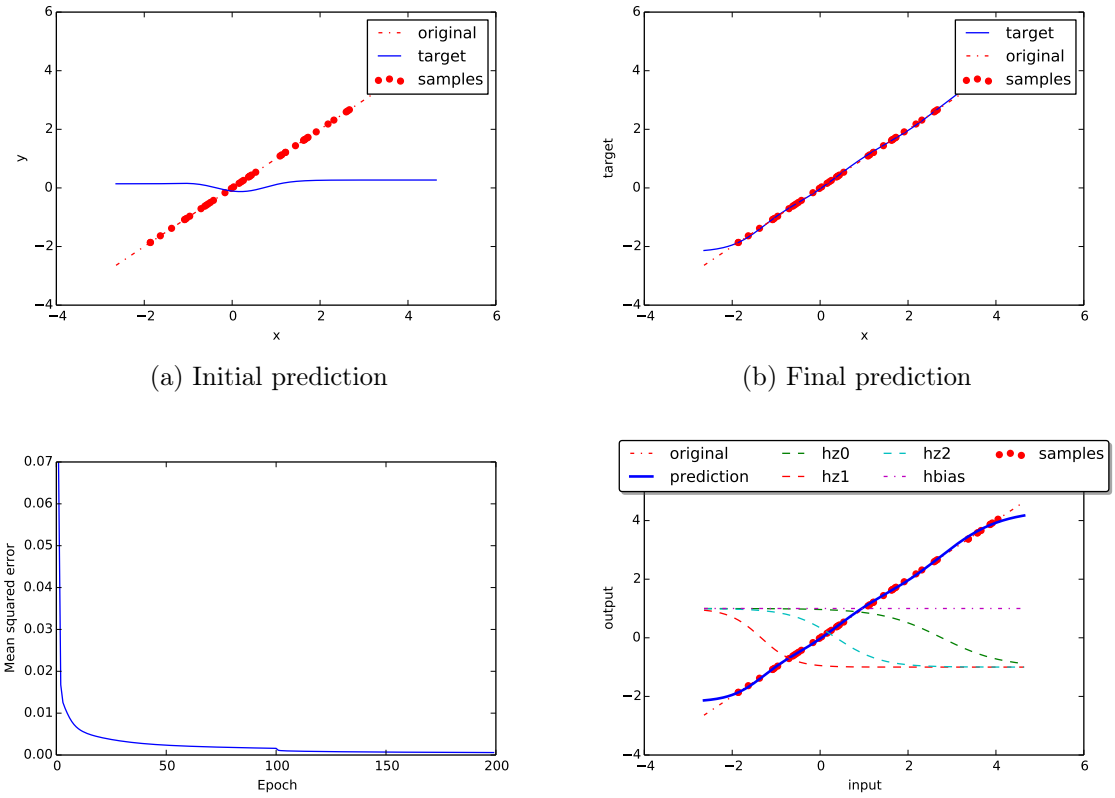
Where $\mathbf{a}$ are the activation functions of the last neurons, and O is the number of categories.

4.4 Multilayer feed-forward neural network

As we saw in the previous sections, a single layer feed-forward neural network can approximate various signals or classify different patterns. However, these networks can not solve more complex problems. In the late 60s, Marvin Minsky and Seymour Papert got published the book "Perceptrons" [Minsky and Papert, 1969], where the authors demonstrated that neural networks with only one input and output layer were not able to solve the Exclusive OR logical function and in general any non-linear classification problems. In order to solve these cases, a set of more complex features was required. Nevertheless, later, it was shown that it was possible to get the required non-linear features by adding an additional layer between the inputs and the outputs (see an example in Figure 4.4b). This layer was referred as a hidden layer, and it was demonstrated that by adding only one hidden layer an ANNs was able to approximate any function; given a sufficient number of hidden neurons. With only one hidden layer the output of the network can be represented with the next equation:

$$y(\mathbf{x}) = h(\mathbf{w}_o^T h(\mathbf{w}_h^T \mathbf{x})) \tag{4.13}$$

In which, there is one set of weights for the hidden layer $\mathbf{w}_h$ and one for the output layer $\mathbf{w}_o$. In this example, the same activation function $h(\cdot)$ is used on every neuron, but it is possible to assign different functions to each layer or neuron. Additionally, more hidden layers can be incorporated making the final function more complex and strongly non-linear. Additionally, networks that use sigmoid function



(a) Initial prediction

(b) Final prediction

(c) Training error

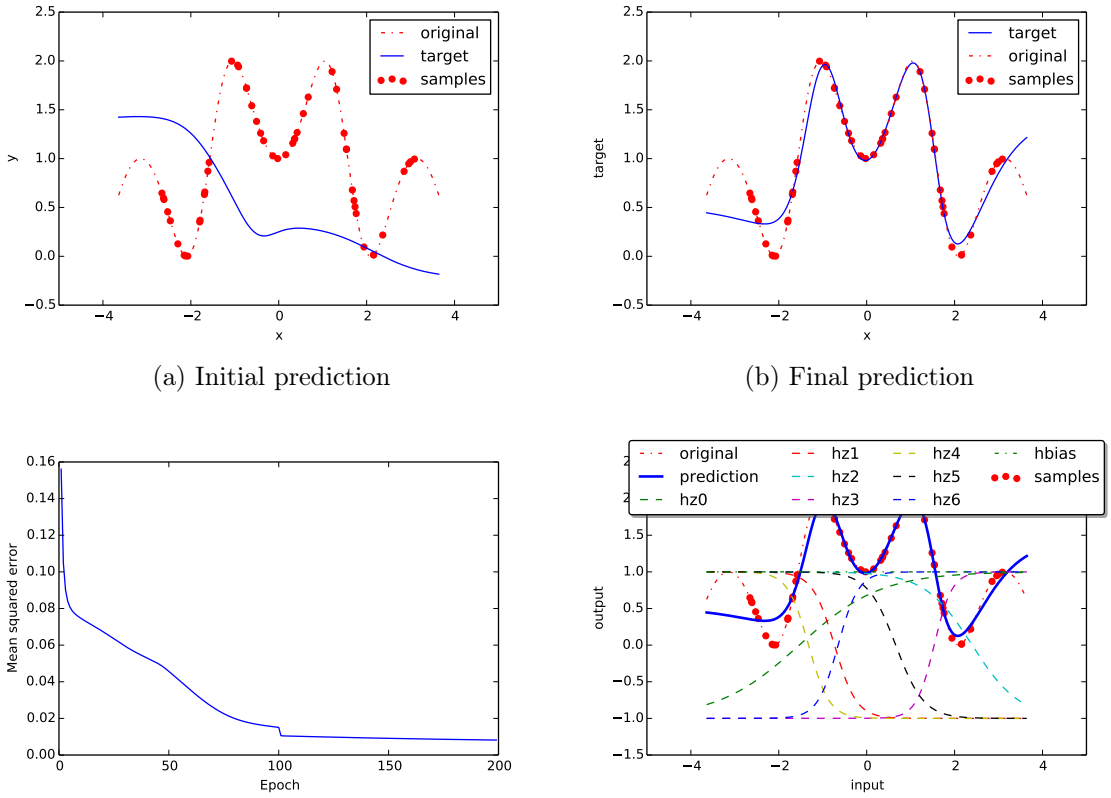
(d) Hidden representation

Figure 4.6: **Example of a MLP fitting a linear pattern**

can fit linear problems if the weights are kept in the central regime of the sigmoid function.

MFNNs are trained using the backpropagation algorithm (see details in Section 4.5.1). Figures 4.6 and 4.7 show two examples of MFNNs using the hyperbolic tangent activation function to fit linear and non-linear data respectively. If we focus in the linear pattern example, Figure 4.6a shows the original pattern with a dotted red line, the available random samples with large red dots and the actual prediction of the network with a blue line. The weights are first randomly initialized following a Gaussian distribution with zero mean and a small standard deviation. Figure 4.6b shows the final prediction after 200 training epochs (one epoch corresponds to all the sample set). Figure 4.6c depicts the training error during the 200 epochs. Finally, Figure 4.6d presents all the hidden representations learned by the hidden layer. The hidden representations are combined in the output layer to perform the final prediction. The same description can be applied to the non-linear example in Figure 4.7.

The performance of these networks can easily improve with the number of neurons and hidden layers. Furthermore, ANNs can approximate any function given



(a) Initial prediction

(b) Final prediction

(c) Training error

(d) Hidden representation

Figure 4.7: **Example of a MLP fitting a sin(cos(x)) pattern**

the sufficient number of hidden units. However, augmenting the number of neurons and layers also increases the complexity of the model, making the training computationally more expensive, and possibly increasing the generalization error. Figure 4.8 shows the common behaviour on the training and test error when the complexity of a model is progressively increased.

4.5 Training

Training ANNs has been one of the most important and difficult problems. The first ideas from McCulloch and Pitts in 1943 required the manual specification of the weights [McCulloch and Pitts, 1943]. It was not until 1957 that Frank Rosenblatt designed an algorithm to train the Perceptron to classify binary patterns [Rosenblatt, 1957] (we saw the training procedure of the Perceptron in Section 4.3.2). Nevertheless, this procedure was not able to train networks with one hidden layer, while Minsky demonstrated that without a hidden layer these networks were seriously limited. In another university, Bernard Widrow and Hoff designed a new architecture that was able to use the derivative of the error to update the weights,

1943

1957

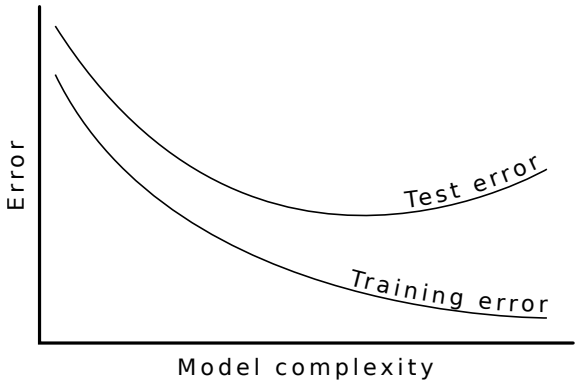


Figure 4.8: **Generalization error.** Common situation when the model complexity is increased

the technique was called "Delta Rule" (also known as "Widrow-Hoff", "Adaline Rule" or "Least Mean Squares Rule").

1974 In 1974, Werbos proposed the "dynamic feedback" algorithm to propagate the error from the output to the input of a network. This algorithm was actually the backpropagation, however, it did not gain the importance that it deserved. Later in 1986, Rumelhart, Hinton and Williams introduced by the first time the concept of 1986 backpropagation in the field of ANNs [Rumelhart et al., 1986]. Although it was not a new technique, it was one of the first papers that got the attention of the research community. From this point, backpropagation has been the most common method to train ANNs until the date. However, multiple tricks have been applied to improve the learning. For example, the initialization of the weights, the activation function, adding momentum, updating dynamically learning rates, weight sharing, and other modifications in the architecture.

4.5.1 Backpropagation

The problem of the Perceptron was that the step function did not allow the error to propagate from the output to the input. The step function is not differentiable and the error could not be propagated. Nevertheless, with a differentiable activation function like sigmoid functions it was possible to compute the gradient and propagate the output error. In Section 4.5.1 we introduce the equations for an ANN without hidden layer. Then, we extend the explanation to the case of multiple layers in Section 4.5.1.

Single layer network

Although in the single layer network the backpropagation algorithm is not applied, we give here the basic maths that will be used in the multilayer network. To train a single layer neural network first we need to compute an error value for all the

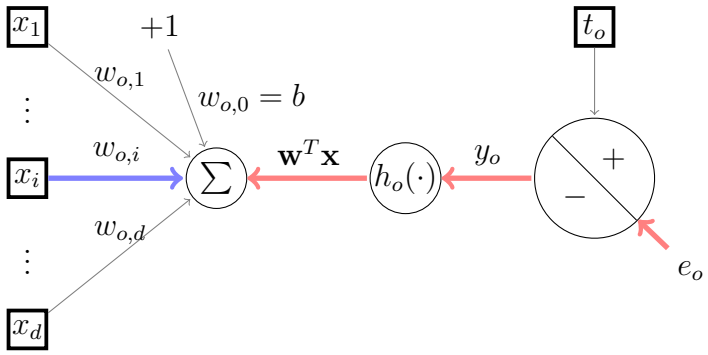


Figure 4.9: **Analogy of backpropagation illustrated in a single layer network**

predictions. We want to minimize the total error of the training samples:

$$\mathcal{E}_{TOTAL} = \sum_{n=1}^{N} \mathcal{E}_n \tag{4.14}$$

Where N is the number of training samples. However, to facilitate all the mathematical notation, in this section we will assume that minimizing the error of one sample $\mathcal{E}_n$ minimizes the total error $\mathcal{E}_{TOTAL}$. Then, in the next equations, the subscript corresponding to the sample n will be omitted. This description will be generalized and explained in Sections stochastic 4.5.2, mini-batch 4.5.4 and batch 4.5.3 gradient descent.

First, we need to choose an error function to derive all the equations. One of the most typical functions is the least squares, assuming that the error on the predictions follows a Gaussian distribution. In this method, the error is the sum of all the squared differences between the target outputs $\mathbf{t}$ and the predictions $\mathbf{y}$:

$$\mathcal{E} = \frac{1}{2}\sum_{o=1}^{O} e_o^2 = \frac{1}{2}\sum_{o=1}^{O}(t_o - y_o)^2 \tag{4.15}$$

Where O is the number of outputs; in regressions and binary classifications it is commonly one. The initial fraction $1/2$ is added to simplify the derivative during the backpropagation. This change do not affect the direction of the error gradient.

Then, given that our prediction is computed by a differentiable function $h(\cdot)$ of the activation $\mathbf{a}$, the prediction can be written as:

$$\mathbf{y}_o = h\left(\sum_{i=0}^{D} w_{o,i}x_i\right) = h(\mathbf{w}_o\mathbf{x}) \tag{4.16}$$

To update the input weight $w_{o,i}$ of an output neuron o for a given sample $\mathbf{x}$ we can compute the gradient of the error surface in the actual parameter space. If we apply the partial derivative respect the specific input weight $w_{o,i}$ on both sides of Equation 4.15 and apply the chain rule we get:

$$\frac{\partial \mathcal{E}}{\partial w_{o,i}} = \frac{\partial \mathcal{E}}{\partial a_o}\frac{\partial a_o}{\partial w_{o,i}} = \frac{\partial \mathcal{E}}{\partial e_o}\frac{\partial e_o}{\partial y_o}\frac{\partial y_o}{\partial a_o}\frac{\partial a_o}{\partial w_{o,i}} \tag{4.17}$$

function	partial derivative
$\mathcal{E}(\mathbf{e}) = \frac{1}{2}\sum_{o\in O} e_o^2$	$\frac{\partial \mathcal{E}}{\partial e_o} = e_o$
$e(y) = (t - y)$	$\frac{\partial e}{\partial y_o} = -1$
$y(a) = h(a)$	$\frac{\partial y_o}{\partial a_o} = h'(a)$
$a(\mathbf{x}) = \mathbf{w}\mathbf{x}^T$	$\frac{\partial a_o}{\partial w_{o,i}} = x_i$

Table 4.2: **Summary of derivatives for the backpropagation chain rule**

The first part of the derivation with the opposite sign is usually called the error or local gradient of one specific neuron and it is symbolized by δ_o.

$$\delta_o = -\frac{\partial \mathcal{E}}{\partial a_o} = e_o h'(\mathbf{w}^T\mathbf{x}) \tag{4.18}$$

Next we show the complete derivation step by step. Also Table 4.2 shows a summary of the partial derivatives.

$$\begin{aligned}
\frac{\partial \mathcal{E}}{\partial w_{o,i}} =& \frac{\partial \mathcal{E}}{\partial e_o}\frac{\partial}{\partial w_{o,i}}\left[\frac{1}{2}\sum_{o=1}^{O} e_o^2\right] \\
=& e_o \frac{\partial e}{\partial y_o}\frac{\partial}{\partial w_{o,i}}\left[\sum_{o=1}^{O}(t_o - y_o)^2\right] \\
=& e_o(-1)\frac{\partial y}{\partial a_o}\frac{\partial}{\partial w_{o,i}}\left[h(\mathbf{w}^T\mathbf{x})\right] \\
=& -e_o \cdot h'(\mathbf{w}^T\mathbf{x})\frac{\partial}{\partial w_i}\left[\sum_{i=0}^{D} w_i x_i\right] \\
=& -e_o \cdot h'(\mathbf{w}^T\mathbf{x}) x_i
\end{aligned} \tag{4.19}$$

The direction of the gradient indicates the direction that maximizes the error. As we are interested on minimizing the error, we can take a step into the opposite direction by changing the sign. The step length is determined by a learning rate η that is previously decided or adapted following some policy. Then, we can update the weight using the gradient of Equation 4.19:

$$\begin{aligned}
w_{o,i} \leftarrow& w_{o,i} - \eta \cdot \frac{\partial \mathcal{E}}{\partial w_{o,i}} \\
=& w_{o,i} + \eta \cdot e_o \cdot h'_o(\mathbf{w}^T\mathbf{x}) \cdot x_i \\
=& w_{o,i} + \eta \cdot \delta_o \cdot x_i \\
=& w_{o,i} + \Delta w_{o,i}
\end{aligned} \tag{4.20}$$

The update of the weights with the $\Delta w_{o,i}$ is commonly called delta rule. We will see that the local gradient is useful when backpropagation is applied.

Figure 4.9 shows an example of how the error must be back-propagated with a single input and output layer. The blue line correspond to the weight being updated, while the red lines are the errors being propagated from the output to the weight. Given that the activation function $h(\cdot)$ is differentiable.

Multilayer

It is also possible to use the same approach to train a MFNN. We saw in the last section that in order to compute the update of a specific weight we need: the input value of the specific weight, the activation value of the actual unit and the error made by this unit. Of these three values, the input and the activation value can be extracted with a forward pass in the network using a specific sample. However, the responsibility of the error by a hidden neuron is not that easy to know. This problem is known as the credit-assignment problem. In addition, the gradient can propagate through an exponential number of paths, however, the backpropagation algorithm uses dynamical programming to compute the gradient layer by layer, reducing the complexity to an acceptable level.

In MFNNs all the weights to be updated belong to one of the next three cases:

1. First hidden layer: The weight to be updated is in the first layer and its corresponding inputs are the original vector of features $\mathbf{x}$.

2. Intermediate hidden layer: The weight to be updated is in the middle of the network, in that case its inputs are the outputs of the previous hidden layer $\mathbf{z}_{l-1}$.

3. Output layer: To compute the gradient we have the output error. $\mathbf{e}_o$.

Then, we follow a similar procedure than the single layer network. First, we forward propagate the input values through all the network and store all the intermediate activation and output values. Then, we compute the error using the squared error or other loss function.

(3) The weights of the output layer can be updated using exactly the same method previously seen in the single layer network. With the exception that we need to substitute the $\mathbf{x}$ vector by the outputs of the previous layer namely $\mathbf{z}_{l-1}$ (see the similarity between the weight updating in Figure 4.9 and Figure 4.10).

(1,2) We saw previously in Equation 4.20 that in order to update one weight we need to compute first the local gradient or error δ_j and the input x_i associated to the weight $w_{j,i}$. In the case of hidden units the input value x_i is instead the output of the previous hidden unit, namely $\mathbf{z}_{l-1}$. In this case, we can get the input value $\mathbf{z}_{l-1}$ from the forward propagation, while the δ_j is missing. From the definition of δ_j – that we saw in Equation 4.18 – we can find the delta in a hidden layer expanding first the partial derivative:

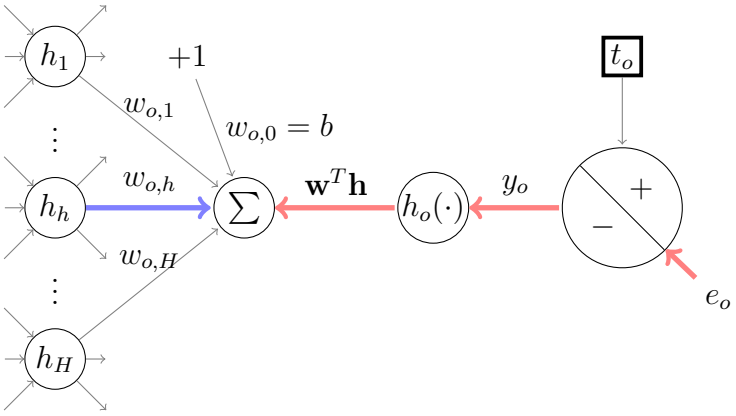


Figure 4.10: **Error backpropagation in an output layer of a MFNN**

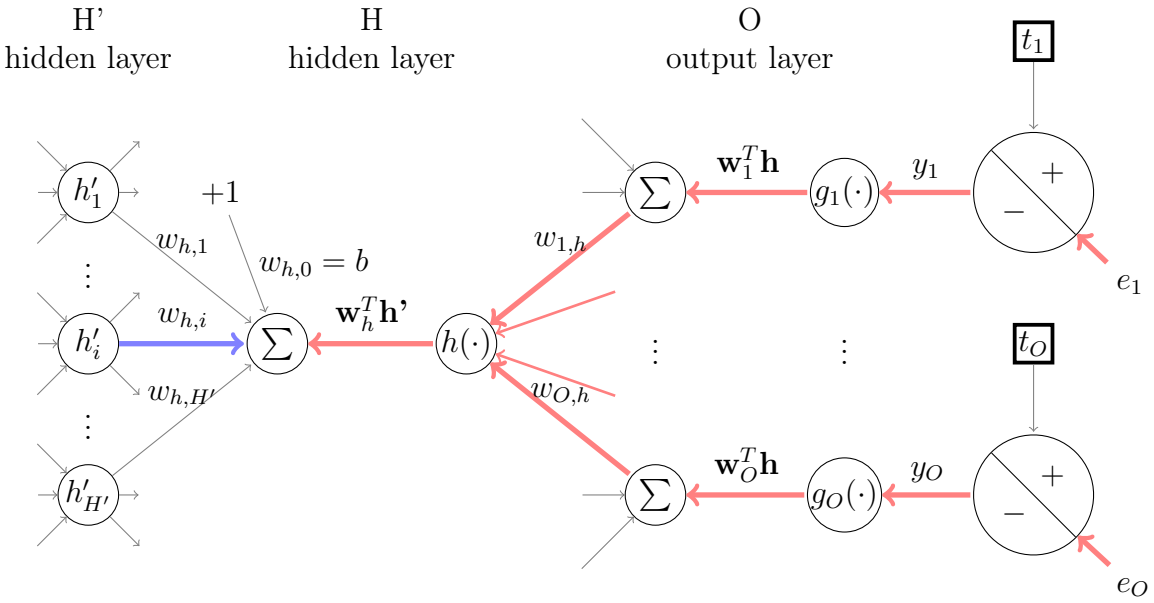


Figure 4.11: **Error backpropagation in a hidden layer of a MFNN**

$$\delta_j = -\frac{\partial \mathcal{E}}{\partial a_j} = -\frac{\partial \mathcal{E}}{\partial y_j}\frac{\partial y_j}{\partial a_j} \tag{4.21}$$

In this case the error caused by the neuron j affects several output errors. If we solve the first partial derivative of the error in the previous equation the summation over all the output errors remains in the equation; all of the errors contribute to the gradient.

$$\frac{\partial \mathcal{E}}{\partial y_j} = \sum_{o=1}^{O} e_o \frac{\partial e_o}{\partial y_j} \tag{4.22}$$

On the other hand, the second partial derivative of Equation 4.21 only depends

on the activation function of the actual neuron:

$$\frac{\partial y_j}{\partial a_j} = h'_j(a_j) \tag{4.23}$$

Next if we apply the chain rule to the partial derivative of Equation 4.22 we get:

$$\frac{\partial e_o}{\partial y_j} = \frac{\partial e_o}{\partial a_o}\frac{\partial a_o}{\partial y_j} \tag{4.24}$$

As in previous cases, the error e of the neuron o is:

$$e_o = t_o - y_o = t_o - h(a_o) \tag{4.25}$$

Where the target value t corresponds to the sample desired output and in a hidden layer corresponds to the desired output of the activation function. In both cases the first term is constant and the partial derivative is

$$\frac{\partial e_o}{\partial a_o} = -h'(a_o) \tag{4.26}$$

Then, given that the activation of the next layer is

$$a_o = \sum_{j=0}^{H} w_{o,j} y_j \tag{4.27}$$

the second partial derivative in Equation 4.24 is

$$\frac{\partial a_o}{\partial y_j} = w_{o,j} \tag{4.28}$$

Finally, we can combine the partial derivatives 4.24, 4.26 and 4.28 into Equation 4.22. Here is done step by step.

$$\begin{aligned}
\frac{\partial \mathcal{E}}{\partial y_j} &= \sum_{o=1}^{O} e_o \frac{\partial e_o}{\partial y_j} \\
&= \sum_{o=1}^{O} e_o \frac{\partial e_o}{\partial a_o}\frac{\partial a_o}{\partial y_j} \\
&= \sum_{o=1}^{O} e_o \left(-h'(a_o)\right) \frac{\partial a_o}{\partial y_j} \\
&= \sum_{o=1}^{O} e_o \left(-h'(a_o)\right) w_{o,j} \\
&= -\sum_{o=1}^{O} \delta_o w_{o,j}
\end{aligned} \tag{4.29}$$

Where in the last line we added the δ_o as described previously in Equation 4.18, however, in this case, the activations come from the previous layer instead of the input features.

Finally the local gradient of neuron j can be computed using the previous Equations 4.21, 4.23 and 4.29:

$$\begin{aligned}\delta_j =& -\frac{\partial \mathcal{E}}{\partial y_j}\frac{\partial y_j}{\partial a_j} \\ =& \left(\sum_{o=1}^{O} \delta_o w_{o,j}\right)\frac{\partial y_j}{\partial a_j} \\ =& \left(\sum_{o=1}^{O} \delta_o w_{o,j}\right) h'_j(a_j) \\ =& h'_j(a_j)\sum_{o=1}^{O} \delta_o w_{o,j}\end{aligned} \tag{4.30}$$

Then, the corresponding update in each hidden or output weight is

$$\begin{aligned}w_{j,i} \leftarrow& w_{j,i} - \eta \cdot \frac{\partial \mathcal{E}}{\partial w_{j,i}} \\ \leftarrow& w_{j,i} + \eta \cdot e_j \cdot h'_j(a_j) \cdot z_i \\ \leftarrow& w_{j,i} + \eta \cdot \delta_j \cdot z_i \\ \leftarrow& w_{j,i} + \Delta w_{j,i}\end{aligned} \tag{4.31}$$

where

$$e_j = \begin{cases} t_j - h_j(a_j) & \text{if neuron } j \text{ is an output} \\ \sum_k \delta_k w_{k,j} & \text{if neuron } j \text{ is hidden} \end{cases} \tag{4.32}$$

4.5.2 Stochastic gradient descent

Stochastic Gradient Descent (SGD) is a method to train an ANN where the gradient is computed and updated using individual samples. The method described in previous Section 4.5.1 used this approach. Every time that the complete set of training samples is used is considered an epoch. This technique makes the gradient to change direction more abruptly than other methods. This can be beneficial when the error space is very non-convex as it is more probable that the weights scape from some local minima. However, this technique do not exploit the parallelizable capabilities of some computer architectures. As the order of the samples is usually randomized, the final solution is not deterministic, making the solution of each execution possibly different.

4.5.3 Batch gradient descent

Batch gradient descent is a method that uses all the training data to compute the gradient of the error surface. The backpropagation method described in Section 4.5.1 can be used with slight modifications. We can compute the gradient for each sample and average all of them to perform one step. Or we can compute all the updates at the same time by changing every vector by a matrix. However, the computational complexity with respect to memory size becomes prohibitive with some datasets as it needs to load all the matrices in memory. In this case the gradient is more stable and it can be fast to find a local minimum if the error surface is locally convex. On the other hand, it is less probable to escape from any local minimum as it will follow directly the sum of all the gradients. One possible benefit of batch gradient descent is that it is deterministic, as it always uses all the training samples to perform an update.

4.5.4 Mini-batch gradient descent

Mini-batch gradient descent (some times called stochastic mini-batch gradient descent) is a method that aggregates the updates of a subset of the training samples. This approach allows to find directions of the gradient that average over several samples. This makes the direction of the gradient more stable than Stochastic Gradient Descent (SGD), while it exploits the computational capabilities of modern computers.

4.5.5 Regularization and other advices

Neural networks are commonly difficult to train. Some of the reasons are their high capability to approximate any function given the sufficient number of neurons and the large number of parameters that need to be learned. Several techniques have been designed to improve their generalization error, training time and convergence to local minima. In this section we give a brief description of some common techniques.

Weight decay

One typical technique for generalization is to use weight decay. This method adds a penalty to the size of the weights in the cost function.

$$\tilde{\mathcal{E}} = \mathcal{E} + \frac{\lambda}{2}\mathbf{w}\mathbf{w}^T \tag{4.33}$$

Then, the weight update is modified following the next equation:

$$\begin{aligned} w_{j,i} &\leftarrow w_{j,i} + \Delta w_{j,i} - \eta\lambda w_{j,i} \\ &\leftarrow w_{j,i}(1 - \eta\lambda) + \Delta w_{j,i} \end{aligned} \tag{4.34}$$

The penalty forces a normal distribution centered in zero in the values of the parameters, reducing the possible complexity of the network. In some cases it can help to generalize better, however it can limit the potential of the network.

Momentum

To speed up the training it is possible to use momentum. This method adds in each learning step a proportion of the previous weight update. Using this approach, it is possible to perform mini-batch gradient descent of certain size, while using the information of the previous batch to update the weights. It can help to find a local minima with a more stable direction for the gradients.

$$w_{j,i} \leftarrow w_{j,i} + \Delta w_{j,i,actual} + \alpha \Delta w_{j,i,old} \tag{4.35}$$

Adaptive learning

Adaptive learning consists on the modification of the learning rate during the training. It is common for ANNs to stop learning at some point during the training and oscillate in a local minima. These oscillations appear because the learning rate modifies the parameters with a specific step size that does not allow the error to reach the local minimum. Some techniques reduce the learning rate value in a fixed way, while other methods automatically adjusts the learning rate by observing the training error fluctuations.

Dropout

Dropout [Srivastava and Hinton, 2014] has demonstrated to improve the generalization of networks with fully connected layers. It is well known that the aggregation of multiple models is usually better than one unique model. Based on this idea, dropout is designed to learn multiple networks that occasionally share some of their weights and neurons, and finally, in the test phase, their predictions are averaged and aggregated. In order to do that, this technique inhibits randomly some neurons during the training. The number of neurons is usually a proportion of the neurons of a specific layer, and it is usually different depending on the task and deepness of the layer. After the training, in the test phase, the weights of the neurons need to be rescaled in order to compensate the larger training values, creating the aggregation of multiple networks.

Weight sharing

Occasionally, it is possible to reduce the number of parameters by adding a strong prior that allows the reuse of specific weights. CNNs are an extreme case, where we assume a translation invariance in all the input space. A more detailed explanation of CNNs can be seen in Chapter 5.

4.6 Extreme Learning Machines

An Extreme Learning Machine (ELM) [Huang et al., 2004] is a MFNN – usually – with one unique hidden layer, in which, the weights of the hidden layer are randomly chosen and fixed during the training process. Only the output weights are trained,

allowing a very fast learning procedure. This approach is very efficient and does not lose the generalization power of a MFNN. The first layer creates a highly non-linear hidden representation that can be used to train the last layer. Despite it is possible to train a MFNN with backpropagation, this algorithm is slow in comparison, and has problems on local minima and plateaus. On the other hand, Extreme Learning Machine (ELM)s is linear-in-the-parameter space and can learn the optimal solution numerically given the fixed hidden representation.

ELMs were used initially for regression and classification problems, but more recently, they are being used for clustering, feature selection, representational learning and other complex tasks. A more detailed description and additional information about ELMs can be found the recent review [Huang et al., 2015].

4.7 Recurrent Neural Network

Recurrent Neural Network (RNN)s are a specific type of ANN that incorporate cycles; usually in a hidden layer (see Figure 4.12). The first RNN appeared in 1982 in the work [Hopfield, 1982]. The Hopfield network was a network with stochastic units that tried to minimize an internal energy state, and was able to store memories in a biological inspired manner. Since then, multiple architectures have emerged: Simple Recurrent Networks (SRN) by Elman and Jordan [Elman, 1991], Continuous-time RNN (CTRNN) [Funahashi and Nakamura, 1993], Long Short Term Memory (LSTM) network [Hochreiter and Schmidhuber, 1997], Bi-directional RNN (BRNN) [Schuster and Paliwal, 1997], Echo State Network (ESN) [Jaeger and Haas, 2004].

The biggest problem of these networks is on finding a good way to train them. Some of them use backpropagation as a learning algorithm but the publications [Hochreiter, 1991] and [Bengio et al., 1994] have shown that the gradient decays (or explodes) exponentially in deep networks; we consider RNN to be very deep networks. Some solutions to this problem involve the use of sparsified connections and Hessian Free optimization (see [Martens, 2010; Martens and Sutskever, 2011; Sutskever et al., 2011]). These solutions have shown to avoid some gradient descent problems and demonstrated to be able to correctly predict some pathological synthetic datasets; which were previously arduous to learn with other techniques.

Recently in the report [Sutskever et al., 2011], the authors trained a special kind of RNN named Multiplicative RNNs (MRNN) that enabled each input unit to train their own hidden-to-hidden weight matrix, augmenting their expressiveness. Using this new architecture the authors got very good results, outperforming the standard RNNs on text generation at character level.

A RNN with one input, output and hidden layer and recurrent connections between the previous hidden state and the actual one can be modeled following the next equations:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{4.36}$$

$$\mathbf{o}_t = \mathbf{W}_{oh}\mathbf{h}_t + \mathbf{b}_o \tag{4.37}$$

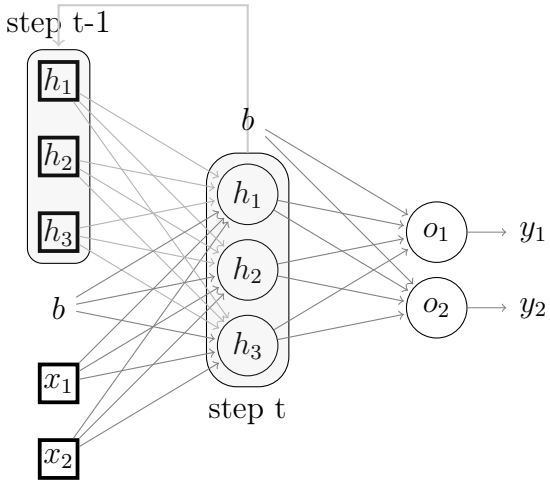


Figure 4.12: **Recurrent Neural Network** with three input features, three hidden neurons and two output neurons. The output of the hidden layer at the previous step is stored and used as an input. The network needs to learn an additional set of parameters for the previous step values.

Where $\mathbf{x}_t$, $\mathbf{h}_t$ and $\mathbf{o}_t$ are the input, hidden and output units values at time step t respectively, $\mathbf{b}_h$ and $\mathbf{b}_o$ are hidden and output biases, and W_{12} are the transformation matrices with the first $(_1)$ and second $(_2)$ subindex indicating destination and origin respectively.

To train RNNs, we can use backpropagation through time (BPTT) [Rumelhart et al., 1986]. This method consists of unfolding the network for some time-steps t and backpropagating the error through the various time steps. However, training a network with several layers could make the gradients to explode or vanish. In order to mitigate this problem we can sparsify the hidden-to-hidden connections and decrease the weights of each unit so as to ensure that the largest eigenvalue is smaller than one (this technique has demonstrated to improve the training).

4.8 Deep learning

During the last decade, the term deep learning has got a huge attention. This term refers to networks with multiple hidden layers that can extract a hierarchical representation of the input data. It is unclear which number of layers is considered to be deep, as this number is continuously increasing.

1980 We could consider one of the first deep architectures the Neocognitron from [Fukushima, 1980]. Fukushima designed an architecture simplifying some ideas from the visual system, and used them to recognize numbers in small images; sometimes adding noise. Three years later , Fukushima managed to use the same Neocognitron

1983 to recognize hand-written digits [Fukushima et al., 1983].

1989 Later in 1989, Yann LeCun et al. [LeCun et al., 1989] applied backpropagation to a Convolutional Neural Network (CNN) to classify handwritten digits that the authors compiled from the New York post office. The network that the authors

presented was adopted by the post offices in the U.S. to read the postal code of large amounts of letters.

From 1995, Hinton and Dayan et al. [Hinton et al., 1995; Dayan et al., 1995; Dayan and Hinton, 1996] were trying to train deep generative and recognition networks; that they called Helmholtz Machines. The authors designed an method called wake-sleep algorithm, that iterated during the training between the generative and the recognition model in an unsupervised manner. Later in 2006, Hinton et al. [Hinton et al., 2006] managed to train a Deep Belief Network (DBN) using a modified version of the wake-sleep algorithm to pre-train the network. After the pre-training, the authors fine-tuned the weights using backpropagation. The innovative idea was to generalize a stack of Restricted Boltzmann Machines (RBMs) to finally represent two Belief Networks (BNs). After this finding, the term Deep learning started to be used and the idea of unsupervised learning using deep networks has been broadly extended.

1995

2006

In 2012, Krizevsky, Sutskever and Hinton [Krizhevsky et al., 2012] trained a deep CNN in a purely supervised manner to classify large images. After the success of this approach, multiple research groups adopted similar ideas and designed neural networks with increasing number of layers [Szegedy et al., 2014; Simonyan and Zisserman, 2014; He et al., 2015].

2012

Chapter 5

Convolutional Neural Network

"Twice and thrice over, as they say, good is it to repeat and review what is good"

— Plato

CNNs are a specific class of ANNs that exploits the local correlations in the neighbourhood of the features, usually in the spatial or temporal dimension (e.g. images, audio or video). CNNs are designed to reduce the number of free parameters while maintaining a high level of performance. The trick consists on reusing most of the weights through the input space assuming local correlations in the features and translation invariance. In image classification, these networks are commonly composed by convolution layers, pooling functions, ReLU activation functions, fully connected layers and a final softmax function. The distribution of these components is designed by trying different compositions and validating their performance. Lately, CNNs achieved state-of-the-art results in computer vision, surpassing other techniques that used Bag of Visual words (BoV), Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), and applied a linear SVM to classify. These are some successful application examples on different tasks: Speech recognition [Abdel-Hamid and Mohamed, 2012], object recognition [Agrawal et al., 2014], image classification [Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2014; He et al., 2015]. For an extended description of CNNs see the book [Bengio et al., 2015].

5.1 Convolution layer

The convolution layer consist on a certain number of kernels that are convolved through all their inputs. Each kernel behaves as a filter that searches for strips, colors, circles, edges, and other patterns. Each kernel is assigned to a *feature map* that convolves the kernel trough all the input image and creates a map of activations, the stronger is the activation the smaller is the angle between the input values and the weight values; as it computes the dot product. These kernels are usually defined by a height and width, by specifying the number of pixels or input units; a stride, by specifying the number of pixels to jump when convolving the kernel; and a padding,

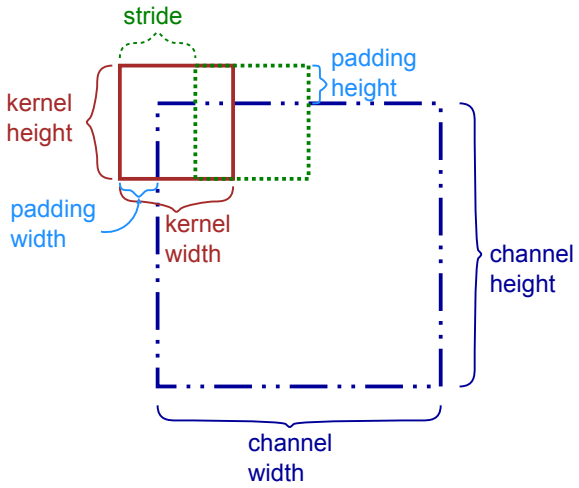


Figure 5.1: **Convolution parameters** kernel size, stride and padding in one channel (more details in the text)

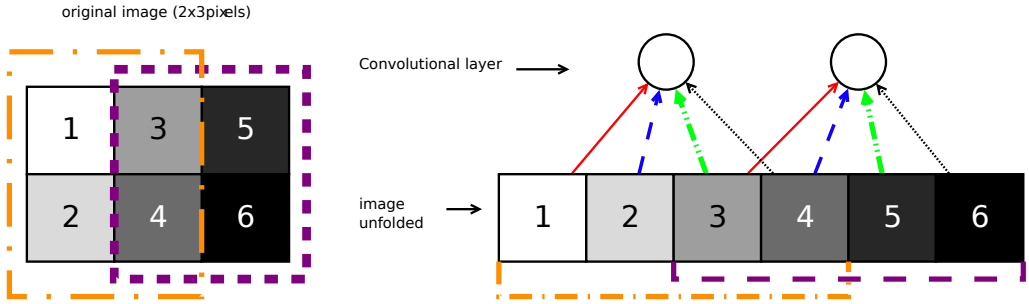


Figure 5.2: **Convolutional Network example**

by specifying the additional zeros that are added at the margins of the channel. Figure 5.1 shows an example of a kernel being convolved in a previous channel. The same idea can be generalized to multiple channels with a kernel moving through all the channels simultaneously. The red square represents the kernel, and the stride determines the number of pixels to displace when it is convolved. The same displacement size applies between the subsequent rows.

When the network is implemented, instead of moving the kernel trough all the channels, the feature maps are unfolded and the network can be seen as a complete network with local connection to the previous channels. Figure 5.2 shows a simplified representation of a convolution on a kernel with size 2×2 pixels, convolved through an image of size 3×2 pixels. The initial channel can be unfolded. Then, the feature map is locally connected to four adjacent pixels. The two neurons share the same set of 4 weights.

5.2 Grouping

Grouping is a technique to reduce the training complexity. This technique consists on the division of the feature maps into subgroups. This reduces the number of

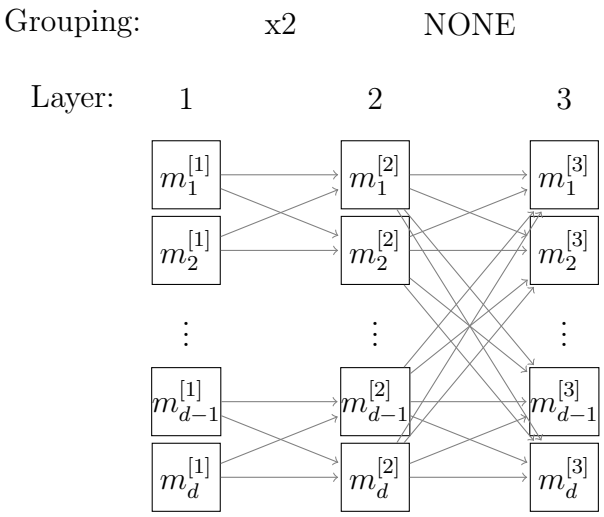


Figure 5.3: **Grouping in a CNN.** In this example, the first layer of feature maps is grouped into 2 groups in the second layer. The third layer is an example without grouping; a fully connected convolution. Although it is possible to use any number of groups, two groups is commonly used in order to distribute the computational resources into 2 different GPUs. This allows to train bigger CNNs and speed up the training.

connections, it also allows the distribution of the subgroups into different machines or Graphics Processing Unit (GPU)s. This technique was used in AlexNet network, and surprisingly the resulting filters were specialized into two distinctive filters, one with luminance and the other with chrominance. Although, the original reason of the grouping was to divide the computation complexity into two GPUs. Figure 5.3 presents an example of a grouping into two subgroups.

5.3 Rectification

One of the most common activation functions for CNNs is the rectified linear unit (ReLU) (see Section 4.2 on page 37). One of the reasons is that the typical sigmoid functions demonstrated a very slow rate of convergence, while ReLU has shown a fast convergence [Krizhevsky et al., 2012]. Furthermore, the absolute zeros at the negative side of the activation function create more sparse representations in the hidden layers. Sparse representations have demonstrated to be specially useful during the training.

Currently, the new Parametric Rectified Linear Unit (PReLU) [He et al., 2015] was shown to improve the results of ReLU, with the simple addition of one extra parameter (see Section 4.2). This is a generalization of the ReLU where the negative side has a parameter that determines the slope, in the ReLU case the slope is always equal to zero.

5.4 Pooling

The pooling function is occasionally incorporated after the rectification of the convolution layer. It consists on a kernel of a specific size that computes the maximum or average of the inputs; or chooses stochastically depending on the input values. The pooling is useful to alleviate possible noise on the input, and to reduce the dimensionality of the channels as the network becomes deeper. The pooling kernel also has a padding and a stride that can reduce the output size.

5.5 Local Normalization

Some works shown an increase on the performance of CNNs by adding local normalization of the output of the hidden layers. In AlexNet, a Local Response Normalization (LRN) was applied after each of the two first convolutions. In this case, the normalized response $b^i_{x,y}$ of the kernel i at the position (x, y) was equal to:

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta} \tag{5.1}$$

Where $a^i_{x,y}$ is the rectified activation of the kernel i at position (x, y); and the constants k, α, n, and β are hyper-parameters adjusted by cross-validation. Using Local Response Normalization (LRN), AlexNet got an improvement of 2% on the test error. In some cases the normalization is situated after the pooling.

5.6 Fully connected layers

A certain number of fully connected layers is commonly added on top of all the previous convolutions, poolings and normalizations (see AlexNet A.2 on page 136). These layers create a non-linear transformation of the final mappings, and the last layer reduces the dimensionality to the number of categories of the classification problem. The largest number of parameters is usually concentrated in this part. However, in new architectures the fully connected layers are being reduced or distributed into different outputs (see GoogleNet A.4 on page 138).

5.7 Soft-max

Finally, a soft-max function is commonly applied to the final activations. This is used to normalize the predictions of the different classes. The output of the soft-max can be interpreted as a probability density function of the given sample belonging to the various categories.

In order to compute the soft-max it is sufficient to compute the following equation for each output unit j.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{5.2}$$

5.8 Complete example

One possible architecture that uses all the previously mentioned parts to create a CNN can be seen in the Appendix A Figure A.5 on page 139. This CNN is a modified version of a network designed by Alex Krizhevsky and modified in the Caffe framework. It is composed by three convolution layers, three max-pooling, two normalizations, three ReLUs, one fully connected layer with ten outputs and a soft-max. This architecture is designed to classify the CIFAR10 dataset which contains ten different categories.

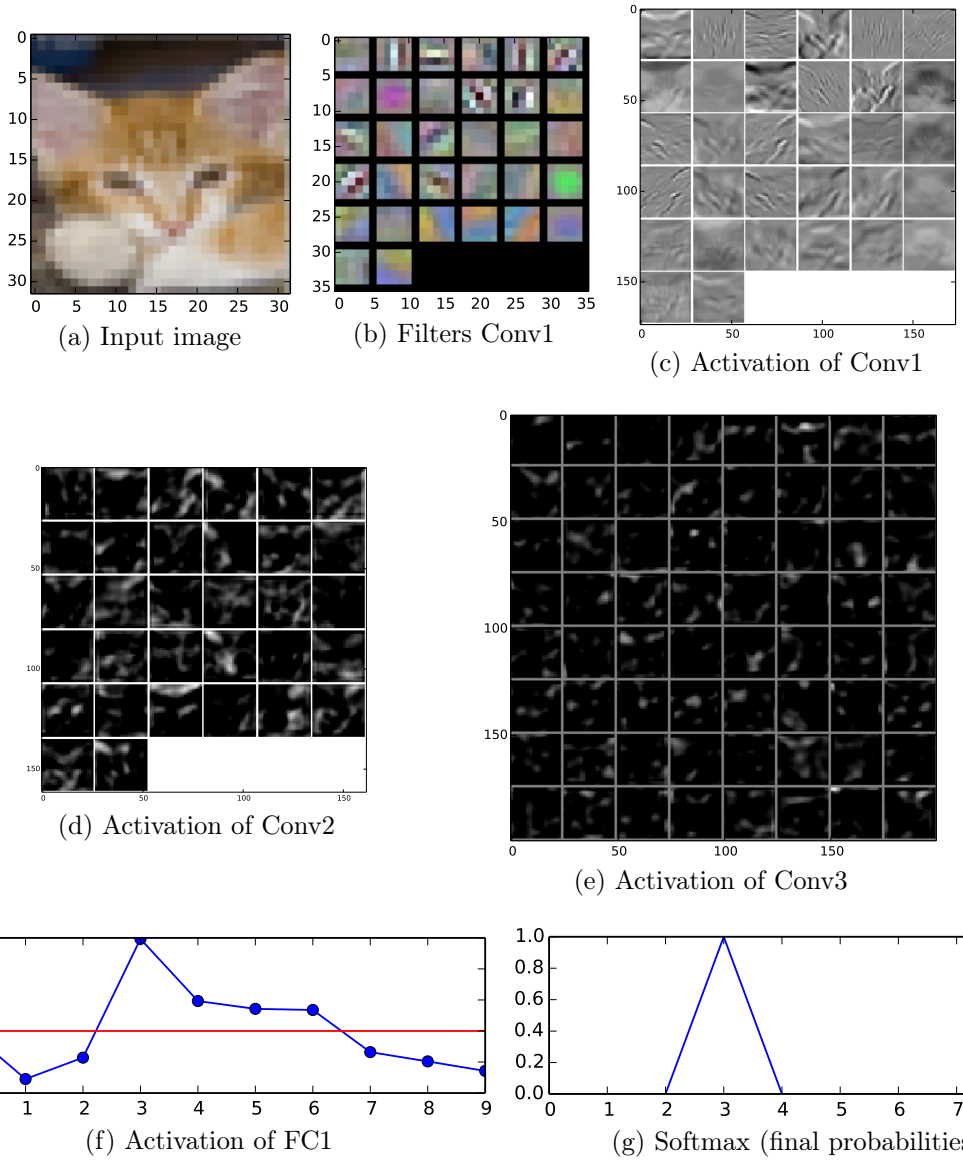
Figure 5.4 shows the activations of each layer when a picture of a cat is presented to the previous network. The caption of the image contains a complete description of each sub-figure.

5.9 Best practices

Training CNNs can be computationally expensive. Moreover, depending on the parameters and configuration the final performance can be bad. These are some recommendations before starting the training of one of these networks.

As it is common in every machine learning task, a large dataset can improve drastically the final performance of the model. In the case of CNNs this can be more accentuated, as it needs to learn the features with the only supervision of the label. For example, in image datasets, the training set can be augmented by cropping different sections, applying affine transformations, translations, scaling, rotations, or mirroring. The cropping is very efficient, as sometimes, training with the original images becomes computationally prohibitive. Also, in most of the cases, mirroring can be applied without loss of generality. In more specific cases it is possible to apply affine transformations that preserve the original properties. For example, in datasets of handwritten digits an affine transformation do not destroy the numbers. In more extreme cases, it is possible to apply elastic deformations (e.g. "elastic deformations that could resemble the uncontrolled oscillations of the hand muscles, damped by inertia" for MNIST dataset in [Simard et al., 2003]). Furthermore, the addition of structured random noise can increase the number of samples and force the network to find useful information.

Nevertheless, some of the typical regularization methods from ANNs are not needed when we limit the number of parameters, the structure, and the activation functions. For that reason, CNNs do not need usually momentum, weight decay, averaging layers, or fine-tuning.



(a) Input image (b) Filters Conv1 (c) Activation of Conv1

(d) Activation of Conv2 (e) Activation of Conv3

(f) Activation of FC1 (g) Softmax (final probabilities)

Figure 5.4: **Example of a forward pass in a CNN.** This examples shows all the activations of a CNN trained with CIFAR10 dataset. The input image 5.4a is convolved with the filters of the first convolution 5.4b giving the output 5.4d, then after a rectifier, a pooling and a normalization it is convolved again by the second convolution. Its outpu is 5.4d. The same process happens with the third convolution giving as an output 5.4e. Then a fully connected layer computes the inner product and outputs ten values 5.4f (one neuron per class). Finally a softmax is computed giving the final probabilities 5.4g where: 0: airplane, 1: automobile, 2: bird, 3: cat, 4: deer, 5: dog, 6: frog, 7: horse, 8: ship, and 9: truck,

Chapter 6

A brief history of Connectionism

> *"Do not believe in anything simply because you have heard it. Do not believe in anything simply because it is spoken and rumored by many. Do not believe in anything simply because it is found written in your religious books. Do not believe in anything merely on the authority of your teachers and elders. Do not believe in traditions because they have been handed down for many generations. But after observation and analysis, when you find that anything agrees with reason and is conducive to the good and benefit of one and all, then accept it and live up to it."*
>
> — The Buddha

In this chapter, I explain briefly my own perception about the history of Connectionism[1]. These facts may help to understand the different findings, ideas and advances that had been achieved in the field of Artificial Neural Networks (ANNs).

Although Connectionism is a novel paradigm of Artificial Intelligence (AI), the basic ideas started from the understanding of our brain, perception, and behaviour. For that reason, this history starts from ancient times when all the explanations were *foggy* philosophical speculations; at that time, it was impossible to make any empirial analysis on brains. Only three centuries ago, some psychologists and philosophers continued with some of the old ideas and created a movement called Associationism. This paradigm tried to theorize and explain the memories and mental states. But it was not after the 18th century, when the technology allowed the visualization of

[1]The first part of the history is more subjective, as it is almost impossible to find the roots of the original ideas. More recent facts are also more objective and easier to define. The majority of these facts are based on the references: large part between years 1749 and 1911 from [Wilkes and Wade, 1997], some historical notes between years 1890 and 1987 from [Anderson and Rosenfeld, 1988] and [Anderson et al., 1990], and additional references [Haykin, 1994; Garson, 1998; Mehrotra et al., 1997; Sutton and Barto, 1998; Gabbay et al., 2006; Vemuri, 1992; Fu, 2003; Boden, 2006]. Some of the information about the authors has been extracted from Wikipedia.

the nervous system and brain cells. From that moment, some prominent biologists become interested on how the brain and their most simple cells worked. Some of the first drawings of nervous cells date to the end of 18th century. At the beginning of the 19th century more biologist continued the work, but in late 30s, some mathematicians tried to model the brain with simple threshold logic. It was in 1943 when a very famous publication by McCulloch and Pitts got published: "A logical calculus of the ideas immanent in nervous activity". The authors developed a simplified mathematical model of a neuron, that in principle, could solve any logical function. From that point, we could say that the Connectionism paradigm started, and in a small amount of time, other psychologists, biologists, mathematicians, engineers and physicists became interested on these computational models.

Some parts of the history are not directly related to Connectionism but they become correlated afterward. These sections are highlighted with a different background and can be skipped during the reading. The format of the actual paragraph is an example.

6.1 First insights into the human perception

300 BCE

The philosophical idea of how our brain works started long time ago. Some of the first evidences of people theorising about "how we retain our ideas" and "what are our perceptions" date from the Ancient Greece. A well known example is the work of Plato[2] and his student Aristoteles. Plato developed the *theory of Forms* or *theory of Ideas* that tried to define the material and non-material objects, and their ideal representation. Although the connection is not clear, the idea of the creation of a universal abstraction of objects or patterns is an inherent part on some topics of Machine Learning. For example, techniques like feature extraction and selection or embedding try to find abstractions that can facilitate the tasks of pattern recognition. Another very useful concept by Plato was the Anamnesis, in which he claimed that our knowledge is impressed in our soul and that the act of learning something new, is nothing but retrieving and old knowledge. Aristotle continued similar work on the same line, he thought that similar images or memories could be somehow associated by evoking similar sensations. These ideas inspired some psychologists and philosophers on later centuries to develop Associationism ideas.

6.2 Human behaviour

1700

During the 17th century, some psychologists continued studding the same ideas about memories and their associations. It was John Locke[3] followed by David Hume, David Hartley and others that wrote about the "association of ideas". It was first

[2] **Plato** (Athens; 428/427 or 424/423 – 348/347 BCE) was a philosopher and mathematician from the Classical Greece. He founded the Academy in Athenes and was one of the important figures on science and the western philosophy.

mentioned in 1700 by John Locke in his third edition of "An Essay Concerning Human Understanding" [Locke, 1700], where he pointed that men associates different ideas together creating strong connections that makes them stack together. He also wrote that different men associate ideas in different manners depending on their education, interests, and other personal reasons (see p.3 [Howard C. Warren, 1921]). These ideas initiated the Associationism movement.

In 1749, David Hartley[4] – one of the members of the British "Associationist School" – got published his book "Observations on man" [Hartley, 1749]. In his book, Hartley proposed that the perceptions in our brain were mere vibrations, and memories were these same vibrations at a lower scale. 1749

1763

The Baye's theorem appears the first time in 1763 within the publication "An Essay towards solving a Problem in the Doctrine of Chances", after the death of Thomas Bayes[5] . It solved the problem of the inverse probabilities, and it played a crucial role in all the probabilistic approaches of machine learning and Connectionism.

6.3 The central nervous system

1800

During the 18th century various neuroanatomists developed different methods to stain some parts of the nervous system. These techniques allowed for the first time to visualize the neurons and all their parts. Some of these techniques are attributed to Gerlach 1858 (carmine), Nissl 1858 (methylene blue), Waldeyerin 1863 (hematoxylin), Golgi 1873 (silver). The better visualization of the nervous system served as an inspiration to later researchers.

1873

In 1873, Alexander Bain[6] wrote the book "Mind and body. The theories of their relation" [Bain, 1873]. In his book, Bain described perceptions and their reminiscence as currents in the nervous system. These currents were stronger when the real perceptions were originated and lower when they were remembered. A very good example of this explanation can be seen in the next paragraph extracted from the mentioned book:

> *"If we suppose the sound of a bell striking the ear, and then ceasing, there is a certain continuing impression of a feebler kind, the idea or memory of the note of the bell; and it would take some very good reason to deter us from the obvious inference that the continuing impression is the persisting (although reduced) nerve currents aroused by the original shock. And if*

[3] **John Locke** (Wrington, Somerset, England; August 29, 1632 – October 28, 1704) was an English philosopher and physician and one of the most influential Enlightenment thinkers.

[4] **David Hartley** (Halifax, Yorkshire, England; August 8, 1705 – August 28, 1757) was an English philosopher, psychologist and the founder of the Associationist school of psychology.

[5] **Thomas Bayes** (London, England; 1701 – April 6, 1761) was a statistician, philosopher and Presbyterian minister. Best known by his formulation of a specific case of the Bayes' theorem.

[6] **Alexander Bain** (Aberdeen, Scotland; 11 June 1818 – 18 September 1903) was a Scottish philosopher and educationalist. He founded the journal Mind, the first on psychology and analytical philosophy.

> *that be so with ideas surviving their originals, the same is likely to be the case with ideas resuscitated from the past – the remembrance of a former sound of the bell."* (p. 89-90 [Bain, 1873])

The idea was very similar to the ideas from David Hartley one century before, but he extended them. Another interesting idea that appeared in his book was that neurons are grouped in clusters because they are used simultaneously; the cell junctions grow when neurons perform the same task. Additionally, it is a necessary condition for the individual neurons to participate in different tasks, and react in a different manner depending on the context of other neurons.

1890 In 1890, William James in his book "The principles of psychology" [William James, 1890] suggested similar ideas. However, James did not know about Bain's work. He stated that the unique "elementary law of association" is the "neural habit" and it stands for the habituation of different neurons to be activated at the same time, reinforcing their connections. On the other hand, their connections become weaker if they are not used:

> *"When two elementary brain-processes have been active together or in immediate succession, one of them, on reoccurring, tends to propagate its excitement into the other."* (p. 566 [William James, 1890])

And he continued by indicating that the activation of a neuron depends on the sum of their input neurons, and that their strength depends on the number of times that they had been activated together:

> *"The amount of activity at any given point in the brain-cortex is the sum of the tendencies of all other points to discharge into it, such tendencies being proportionate (1) to the number of times the excitement of each other point may have accompanied that of the point in question; (2) to the intensity of such excitements; and (3) to the absence of any rival point functionally disconnected with the first point, into which the discharges might be diverted."* (p. 567 [William James, 1890])

1889 At the end of the 18th century and beginning of the 19th, Ramón y Cajal[7] was studying the biological structure of the brain and the nervous system. He was one of the most prominent researchers on neuroscience, and he is at the moment
1909 an important reference. Some of his most important postulates were: the *neuron doctrine*, neurons act as individual units in a discretized manner; and the *law of functional or dynamic polarization*, each neuron receives signals from other neurons to its dendrites and transmits the signal through their axons as action potentials [Cajal, 1909]. Figure 6.1 is an example of his drawings of neurons.

[7] **Santiago Ramón y Cajal** (Petilla de Aragon, Spain; 1 May 1852 – 18 Octover 1934) was a Spanish pathologist, histologist, neuroscientist and Nobel laureate. Famous by his work on neuroscience and hundreds of drawings of the intricate brain cells.

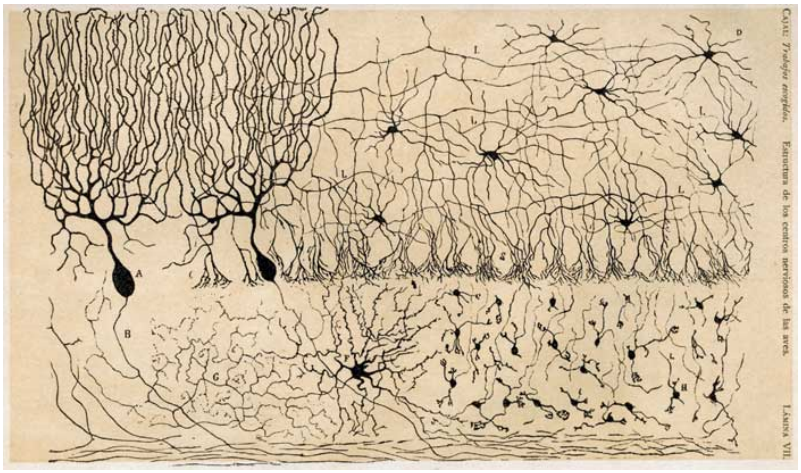


Figure 6.1: **Drawing of neurons by Ramón y Cajal**

6.4 Mathematical Biophysics

Prior to that moment, there were small mathematical foundations about Biology and more specifically the nervous system. It was in 1938, when Rashevsky[8] got published his book on mathematical biology and biophysics entitled "Mathematical Biophysics: Physico-Mathematical Foundations of Biology" [Rashevsky, 1938]. The intention of this book was to create fundamental equations and give a mathematical basis to the field of Biology. Some of the most important ideas were about the nervous system, and how to model it using linear differential equations, and describe the activations and propagations of the electrical impulses of neurons. He proposed to model the output of the neurons with binary threshold functions, and demonstrated the possibility to compute some logic functions. In 1940, Rashevsky founded the section of *Mathematical Biophysics* in the University of Chicago. This new multidisciplinary area involved a variety of fields like mathematics, physics, biology, zoology, chemistry and psychology (A review of the work done by Nicolas Rashevsky can be found in [Cull, 2007]).

1938

1940

In the same University of Chicago, in 1943, McCulloch[9] and Pitts[10] got published the influential work "A logical calculus of the ideas immanent in nervous activity" [McCulloch and Pitts, 1943] (at that time McCulloch was professor of psychiatry and psychology while Pitts was a student of Mathematical Biology). In this article, the authors designed a simplified mathematical model of a neuron. The simplification had several assumptions, as stated in the original article:

1943

[8]**Nicolas Rashevsky** (Chernigov, Russian Empire; November 9, 1899 – January 16, 1972) was a American theoretical physicist that pioneered the mathematical biology.

[9]**Warren Sturgis McCulloch** (Orange, New Jersey; November 16, 1898 – September 24, 1969) was an American neurophysiologist and "cybernetician". Known by his work on mathematical models of the neurons and one of the initiators of the Connectionism.

[10]**Walter Pitts** (Detroit. Michigan; April 23, 1923 – May 14, 1969) was a mathematician and logician. He was a brilliant mathematician from a young age, and was accepted as a member on Rashevsky's group being only 14 years old (p. 189 [Boden, 2006]). Some of his main contributions were in cognitive science, psychology, philosophy, neuroscience, computer science, artificial

1. *The activity of the neuron is an "all-or-none" process.*
2. *A certain fixed number of synapses must be excited within the period of latent addition in order to excite a neuron at any time, and this number is independent of previous activity and position on the neuron.*
3. *The only significant delay within the nervous system is synaptic delay.*
4. *The activity of any inhibitory synapse absolutely prevents excitation of the neuron at that time.*
5. *The structure of the net does not change with time.*

Following these assumptions, the authors discretized the time and binarized the output of the neurons. At each time step, a neuron was fired "only if" the sum of all its inputs from other binary neurons surpassed an *excitation threshold*. This fact was modeled with the next equation:

$$n_j(t) = I\left(\sum_{i \to j} w_{ij} n_i(t-1) > \theta_j\right) \tag{6.1}$$

Where the neurons n_i were connected to the neuron n_j at time t, w_{ij} were attenuation weights with values in the interval $[0, 1]$, and θ_j was the *excitation threshold* of the neuron n_j. With this model, neurons could perform some logical operations like the AND operation, if the threshold was set to 2; the INCLUSIVE OR, if the threshold was set to 1. They called this logic Temporal Propositional Expression (TPE), and a finite number of neurons – without circles – could compute any logical function (p.9 [Mühlenbein, 2009]); the authors claimed to have *"proved, in substance, the equivalence of all general Turing machines – man-made or begotten"* (p. 298 [Glimm et al., 2006]).

6.5 Machine intelligence

1944 In 1944, J.W. Mauchly[11] and J. P. Eckert[12] finished the design of the first electronic general-purpose computer ENIAC (Electronic Numerical Integrator And Computer). Before the construction of the ENIAC, Mauchly and Eckert started to design the EDVAC (Electronic Discrete Variable Automatic Computer) with the consulting help of J. von Neumann[13] . Neumann used to discuss with McCulloch and Pitts about their work, and inspired by some of the ideas in their work [McCulloch and Pitts, 1943] he finished the "First Draft of a Report on the EDVAC" [von Neumann, 1945] in the Spring of 1945 (p. 73 Chapter 2, [Norberg, 2005]). Von Neumann con-

1945

intelligence and artificial neural networks.

tinued his research career creating new automatas, inspired by the biological theories of McCulloch and Pitts and the computational theories of Alan Turing.

During that period of time, Alan Turing[14] was also interested in the creation of new computer machines able to learn by themselves. Turing designed one type of machines by using simple neurons randomly organized; what he called "unorganized machines". In 1948, Turing wrote a report in the National Physical Laboratory of London with the title "Intelligent Machinery". This paper was never published until 1965, because *"it was a schoolboy essay"* (this sentence is attributed to the headmaster director of the laboratory Sir Charles Darwin; see [Copeland and Proudfoot, 1999]). The manuscript contained some insights to the posterior ideas of Artificial Neural Networks (ANNs). One of the ideas was a type of neural network that he called "B-type unorganised machine". These networks were composed by set of neurons with some connections between them. The connections could be adjusted with connection-modifiers that enabled or disabled the output of the neurons (for more details see [Copeland and Proudfoot, 1996, 1999]). 1948

Norbert Wiener[15] was also interested on Turing's ideas about learning and Rashevsky's mathematical foundations on mathematical biology and biophysics. Wiener extended the work of Rashevsky and in 1948 published the book "Cybernetics or Control and Communication in the Animal and the Machine" [Wiener, 1948]. This book contained very useful mathematical tools for control, communication and statistical signal processing, amongst others (for more information see p. 332 [Copeland, 2012]).

In the same year, William Ross Ashby[16] created the "Homeostat" at Barnwood House Hospital [Ashby, 1949]. The Homeostat was one of the first electronic devices able to adapt automatically to external changes. It was controlled by four "Royal Air Force" bomb control units, and was connected with some loops. Figure 6.2 shows a

[11] **John William Mauchly** (Cincinnati, Ohio, United States; August 30, 1907 – January 8, 1980) was an American physicist, who co-designed the first electronic general-purpose computer ENIAC, followed by the EDVAC, BINAC and UNIVAC I.

[12] **John Adam Presper Eckert** (Philadelphia, Pennsylvania; April 9, 1919 – June 3, 1995) was an American electrical engineer, who co-designed the first electronic general-purpose computer ENIAC, followed by the computer EDVAC.

[13] **John von Neumann** (Budapest, Austria-Hungary; December 28, 1903 – February 8, 1957) was a Hungarian and later American pure and applied mathematician, physicist, and principal member of the Manhattan Project. He designed the Von Neumann computer architecture consisting on a Memory, and an Arithmetic Logic Unit (ALU) mediated by a Control Unit.

[14] **Alan Mathison Turing** (Warrington Crescent, London; June 23, 1912 – June 7, 1954) was an British mathematician, cryptanalyst, logician, computer scientist, and cognitive scientist. With the age of 22 he invented the abstract computing machines (Turing machines). He also helped on the decryption of the Enigma machine at the end of the World War II. In 1952 he declared to the police about his homosexuality, this made him to choose between imprisonment or some experimental treatment that he finally choose. In 1954 he was found died in his apartment. They autopsy reported that he died because of cyanide, and it was declared suicide, although it was never proven. In September 10, 2009 the British government apologized for their persecution for being homosexual.

[15] **Norbert Wiener** (Columbia, Missouri; November 26, 1894 – March 18, 1964) was an American mathematician, philosopher, and professor at MIT. Considered one of the precursors of cybernetics.

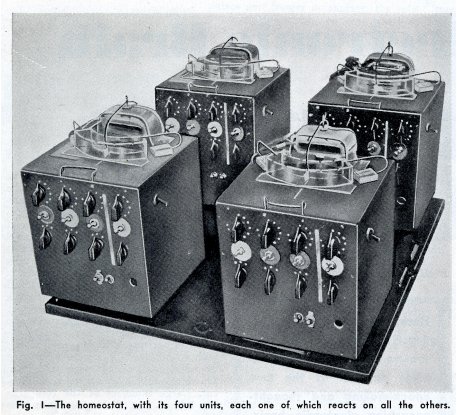


Figure 6.2: **The Homeostat created by William Ross Ashby.** Original text: The homeostat, with its four units, each one of which reacts on all the others (p. 78 "The Electronic Brain" [Ashby, 1949]).

picture of the Homeostat.

1949 After the second World War, in 1949, Donald Olding Hebb[17] inspired by Ramón y Cajal and Lashley[18] published the book "The Organization of Behavior" [Hebb, 1949]. In this book Hebb introduced one of the most used ideas about synaptic learning. His "learning postulate" (also known as "Hebb's postulate" and "Hebbian theory") proposed that biological neurons that are repeatedly activated increased their strength, in such a way that future activations became easier. On the other hand, if a pair of neurons were never activated together their connections became weaker, making their future responses more difficult. In Hebb's words:

> *"When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growhth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased"* (p. 50 Chapter 4 "The Organization of Behavior")

This was one of the first signs of neural networks learning their "weights" (short-term neural memory), however, it was a biological theory and – at that moment – it had little impact on engineering and computer science.

1951 In 1951, Marvin Minsky[19] was studding his PhD in mathematics in Princeton

[16] **William Ross Ashby** (Londres, England; September 6, 1903 – November 15, 1972) was an English psychiatrist and pioneer in cybernetics. He created the Homeostat in 1948; a device capable of adapting itself to the environment.

[17] **Donald Olding Hebb** (Chester, Nova Scotia, Canada; July 22, 1904 – August 20, 1985) was a Canadian psychologist, interested on learning processes. He influentiet in the area of neuropsychology, the function of neurons and developed the theory of Hebbian learning.

[18] **Karl Spencer Lashley** (Davis, West Virginia; June 7, 1890 – August 7, 1958) was an American psychologist and behaviorist. Known by his contributions on the process of learning and memory. He tried to localize memories in different parts of the brain (regions that he called engrams), however, after years of search he concluded that the memories had to be distributed across the cortex.

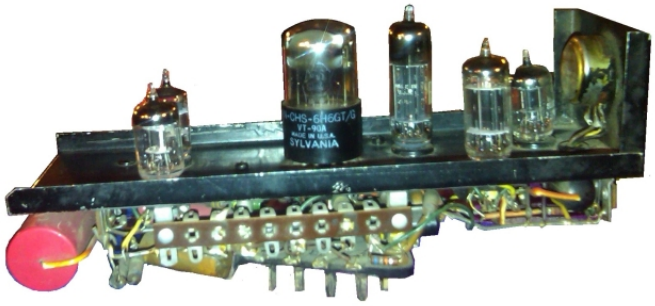


Figure 6.3: **One neuron from the SNARC.** Neuron of the maze-solving computer that Marvin Minsky shown to Gregory Loan.

University. Minsky got fascinated by the articles of McCulloch and Pitts, and decided to implement a self-learning device. Later, Minsky created one of the first physical neural net machines called the SNARC (Stochastic Neural Analog Reinforcement Calculator) . This device was composed by a set of vacuum tubes with some connections, and it was able to modify the connections automatically. Although it never carried any useful function, it provided inspiration to other researchers (see more about Minsky in [Bernstein, 1981]). Figure 6.3 shows a neuron of the SNARC constructed with vacuum tubes.

In 1952, Ashby published the book "Design for a Brain: The Origin of Adaptive Behavior" [Ashby, 1960].In his book, Ashby exposed the flexibility of our brain in order to learn and adapt to new situations. 1952

1953

In 1953, Metropolis[20] et. al. proposed the "Metropolis-Hastings algorithm", a Markov chain Monte Carlo (MCMC) method to obtain samples of complex probability distributions, in which is possible to obtain values proportional to the original density distribution [Metropolis et al., 1953]. This algorithm served as the basis to some Connectionism approaches like Boltzmann Machine (BM) in 1985.

In 1954, Minsky finished his doctoral thesis entitled "Theory of Neural-Analog Reinforcement Systems and Its Application to the Brain-Model Problem" [Minsky, 1954]. In his thesis, Minsky described an analog machine that learned using reinforcement learning, and was composed by smaller units also called SNARCs (p. 18 [Sutton and Barto, 1998]). 1954

In the same year, at the Massachusetts Institute of Technology, B.G Farley and W.A. Clark performed one of the first simulations of a neural network on a digital computer[Farley and Clark, 1954]. The network contained approximately 128 neurons, in two layers connected with randomly initialized weights (p. 441 [Gab-

[19]**Marvin Lee Minsky** (New York City, United States; 9 August 1927) is an American cognitive scientist, mostly interested on artificial intelligence, and co-founder of the Massachusetts Institute of Technology's AI lab.

[20]**Nicholas Constantine Metropolis** (Chicago, Illinois, USA; June 11, 1915 – October 17, 1999) was a Greek and American physicist. He and a group of researchers at Los Alamos designed the Monte Carlo method.

bay et al., 2006]). The authors designed a trial-and-error experiment and used a modified "Hebbian rule" to learn and recognize two different patterns. The authors noticed that in order to achieve the correct solution the different patterns had to be presented alternatively. At each step one of the patterns was shown, then the weights were increased or decreased depending on its correctness: if the prediction went into the correct direction the weights were increased, otherwise the weights were decreased.

From a physics perspective, G.G. Cragg and H.N.V. Temperly formuled from the first time the possible similarity between the interaction between neurons and the spin-glass problem in his article "Memory: the Analogy with Ferromagnetic Hysteresis" [Cragg and Temperley, 1955]. The spin-glass problem was well known in the physics community, and thanks to this analogy, some physicists shown interest on the use of Artificial Neural Network (ANN) to model physical systems (p. 74 [Neelakanta and DeGroff, 1994]).

In the discipline of communication theory, Gabbor[21] proposed the idea of a "nonlinear adaptive filter" that could be used to learn from past patterns, by using the mean squared error of the prediction and gradient descent [Gabor, 1954] (see also p. 24 [Liu et al., 2011]). Later, this idea was used to train Artificial Neural Networks (ANNs).

1956 In 1956, at the IBM Research Laboratory of New York, N. Rochester, J.H. Holland, L.H. Haibt and W.L. Duda made another simulation of a neural network on the IVM Type 704 Electronic Calculator [Rochester and Holland, 1956]. The researchers got inspired by the ideas of Brenda Milner[22] and the book of Hebb "The Organization of Behavior" about how the brain could work. They made two different experiments arranging neurons in one layer and connecting randomly to a second layer of neurons. The first experiment simulated 69 neurons, and tested the ideas of Hebbs in his monograph work. The second experiment simulated 512 neurons, and was addressed to test the theory of Milner; revision of Hebbs theory "F.M. Model". During the experiments the authors had to modify some aspects of the learning process. One of the problems was that using Hebbs rule the synaptic weights increased without bounds. They had to normalize the weights to sum up to some constant. The use of a normalization enabled to increase the desired weights while at the same time decreased the non-desired. Another conclusion was that it was necessary that the network contained excitatory and inhibitory neurons in order to respond to different patterns.

The same year in Princeton University, Uttley[23] demonstrated empirically the possibility to train a neural network to classify two different patterns as a binary classes, using the weights as conditional probabilities [Uttley, 1956b,a]. In his ex-

[21] **Dennis Gabor** (Ungaria; 5 June 1900 – 8 February 1979) was a Hungarian and British electrical engineer and physicist, who invented the holography. Received the 1971 Nobel Prize in Physics for his invention of the holographic method.

[22] **Brenda Milner** (Manchester, England; July 15, 1918) is a Canadian neuropsychologist with important contributions in the field of clinical neuropsychology.

periments, Uttley used Shannon's entropy measure.

Von Neumann found a solution to the reliability of the networks through redundancy in got published the paper "Probabilistic logics and the synthesis of reliable organisms from unreliable components" [von Neumann, 1956]. Neumann proposes the use of probabilistic neurons to form the network. However, he did not like the idea because these units were more reliable and simpler than the real biological neurons. He formalized the probabilistic logic, and then, he extended these ideas to create the "complicated automata" also called the "cellular automata. Nonetheless, the cellular automata had limited reliability and – at that time – there was a lack of logical theory of automata.

Another important work was initiated by Wilfrid K. Taylor, who applied the theory of Hebbian learning to create one of the first associative memories [Taylor, 1956]. In his work, he was computing the associations between different pairs of visual patterns (e.g. alphabetic letters). The network was composed by a layer of sensory inputs (mimicking the retina), and a second layer of associative neurons that received only the maximum signals on a neighbourhood of input neurons. This machine was able to learn the necessary features by itself, using Hebbian learning (see p.892 [Boden, 2006]).

M. Minsky, J. McCarthy[24] , N. Rochester[25] , C. Shannon[26] organized the first "International conference on artificial intelligence" – the Dartmouth Summer Research Project on Artificial Intelligence, sponsored by the Rockefeller Foundation.

6.6 The renaissance of Connectionism

In 1957, the psychologist Frank Rosenblatt wrote the first report about the Perceptron entitled "The Perceptron, a Perceiving and Recognizing Automaton" [Rosenblatt, 1957]. The idea was inspired by the visual nervous system. A "retinal" image was sent to the "Association" units (or "A-units"), these signals were scaled by some weight connections, then summed, and their output was sent to the "Response" units (or "R-units") that applied a threshold to the signal, producing a binary output. 1957

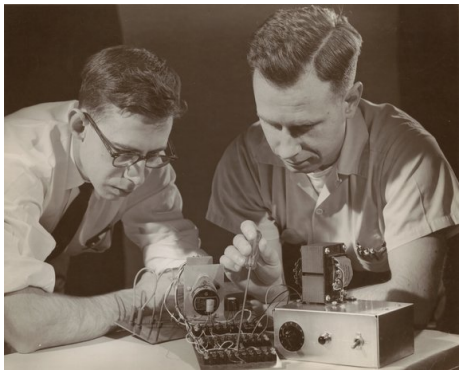
One year later, Rosenblatt extended his Perceptron and got published one of his most famous articles "The Perceptron: a probabilistic model for information storage and organization in the brain" [Rosenblatt, 1958]. Rosenblatt designed a supervised 1958

[23] **Albert M Uttley** (;) mathematician, computer scientist and experimental psychologist, who contributed to cybernetics movement.

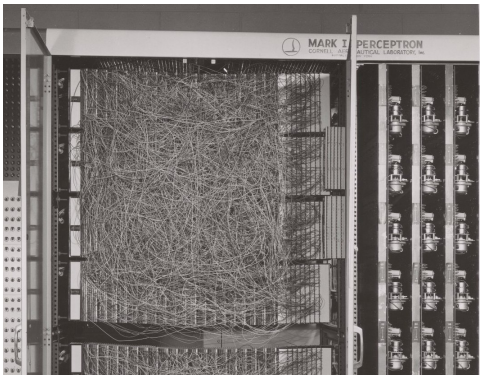
[24] **John McCarthy** (Boston, Massachusetts, United States; 4 September 1927 – 4 October 2011) was an American computer scientist, and cognitive scientist. He coined the term of "Artificial Intelligence" (AI) and developed Lisp programming language.

[25] **Nathaniel Rochester** (United States; 14 January 1919 – 8 June 2001) was an American electrical engineer. He designed the IBM 701, wrote the first symbolic assembler and participated in the founding of the field of Artificial Intelligence.

[26] **Claude Elwood Shannon** (Medford, Massachusetts, United States; 30 April 1916 – 24 February 2001) was an American mathematician, electronic engineer, cryptographer. He is known as the 'the father of information theory".



(a) Rosenblatt working on a A-unit



(b) MARK I Perceptron in cornell aeronautical laboratory

Figure 6.4: **Pictures of Rosenblatt work**

learning method to train the Perceptron to classify different patterns. The idea was to present all the sample patterns and after each pattern compare the network prediction with the correct class. If the result did not coincide then the weights were updated using a type of Hebbian learning equation:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + 2\alpha t\mathbf{x} \tag{6.2}$$

Where the $\mathbf{x}$ was the input vector, the α corresponded to the learning rate, t was the target class, and w was the connection weight. The same equation could be adapted to incorporate the correctness of the prediction by adding the actual prediction of the network y.

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha(t - y)\mathbf{x} \tag{6.3}$$

Applying the new equation after each pattern only modified the weights after the wrong predictions, while in the correct prediction the term $(t - y)$ was equal to zero, canceling the update.

Rosenblatt demonstrated with the "perceptron convergence theorem" that this algorithm always found a solution in a finite time; having some assumptions. The initial weights of the Perceptron were selected randomly without prior information, and the network was able to self-organize by adjusting its weights to memorize the correct output. See more about the history of the Perceptron in page 903 of the book "Mind as machine: A history of cognitive science. Vol.1" [Boden, 2006].

To test these ideas, Rosenblatt, Charles Wightman and other researchers created the Mark I Perceptron at Cornell University. It was used for character recognition of binary images of size $20 \times 20 = 400$ pixels. The network was composed by 512 weights (associator units, A-units) in a $8 \times 8 \times 8$ array of potentiometers driven by electric motors, 400 photosensitive receptors and 8 response units (R-units). See more detail about the experiment in page 5 of [Hecht-Nielsen, 1989].

In the same year, O.G. Selfridge[27] proposed the use of the "hill climbing" algorithm to learn the weights of an ANN that he called "Pandemonium" in his pub-

lication [Selfridge, 1958]. The idea was to change randomly all the weights of the network and evaluate the new performance, if the change improved the predictions it was accepted, if not the network was not modified. Using this algorithm, the network only improved or remained equal. With this approach, it was possible to learn systems with multiple layers of feature detectors, with very complex hidden features.

Also the unfinished book by John von Neumann "The Computer and the Brain" was finally published; one year after his dead. In the book, Neumann exposed the differences between the brain and computer machines in terms of computational power, processing speed and parallelism.

In , Bernard Widrow[28] and Hoff (one of his graduate students) designed one type of physical resistor that could be used as a memory. This resistor could be modified externally with electricity and could store one of ten possible values for long periods of time (from 1 to 10 ohms). They called this new type of resistor a memistor. Widrow and his student created an electrical device using multiple memistors and called it Adaline (Adaptive Linear Neuron or Adaptive Linear Element) [Widrow, 1960]. Adaline was composed by a single layer of memistors acting as weights. The authors also designed one method to teach the Adaline to classify different patterns, using a similar method as in the Perceptron. However, their method modified the weights using the derivative of the error, and followed the direction with the smallest error. This technique was called the "Delta Rule" and it converged to the least squares error (the "Delta Rule" is also known as Widrow-Hoff rule, Adaline Rule or Least Mean Squares (LMS) Rule). 1960

Roger Barron and Lewey Gilstrap founded the Adaptronics Corportaion. One of the first companies to sell neurocomputing applications.

In 1961, Minsky got published the report "Steps Toward Artificial Intelligence" in which he dedicates some sections to networks. The early version of this publication was finished on 1957 with the title "Heuristic aspects of the artificial intelligence problem" but it was not published, however, copies of the publication reached various research groups. 1961

Steinbuch[29] and Piske continued the work of Taylor about associative memories by using a "learning matrix"; a binary and an analog associative network [Steinbuch and Piske, 1963]. Additionally, the authors proposed some electronic circuits able to run the learning algorithms.

[27] **Oliver Gordon Selfridge** (London; May 10, 1926 – December 3, 2008) was an English and American mathematician. Studied in Massachusetts Institute of Technology from 1942–1950. After 2 years at Signal Corps Laboratories at Fort Montmouth, New Jersey, he joined Lincoln Laboratories in Group 34, Communication Techniques, were he was Group Leader (partially from the biographical note in p.512 [Sutherland, 1959]. He contributed on neural networks, pattern recognition and machine learning and created the Pandemonium.

[28] **Bernard Widrow** (United States; 24 December 1929) is an American professor of Electrical Engineer in Standford University. He is the co-inventor of the least mean squares filter (LMS) adaptive algorithm.

[29] **Karl W. Steinbuch** (Stuttgart-Bad Cannstatt, Germany; 15 June 1917 – 4 June 2005) was a German computer scientist, cyberneticist, and electrical engineer. Pioneer on his work on artificial

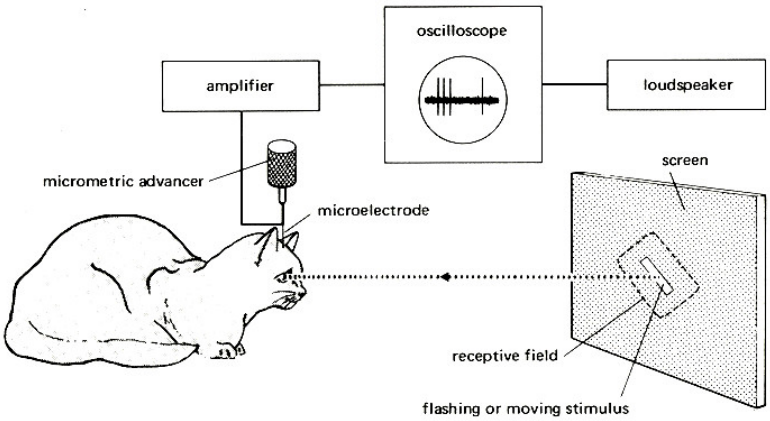


Figure 6.5: **Hubel and Wiesel experiment** about the visual cortex of a cat

Frank Rosenblatt wrote the huge report (626 pages) "Principles of Neurodymaics: Perceptrons and the theory of brain mechanisms" [Rosenblatt, 1961]. In which, Rosenblatt compiled all the ideas about the creation of models that were able to perceive the environment and to learn how to recognize objects. The author explains that Hebbian learning can be used to solve these tasks, and that the Perceptron and the brain were constrained in a similar way. Additionally, the report compiled and explained the the results of previous experiments of the Perceptrons. He also proposes a backpropagation algorithm to train the Perceptron, however, at that moment the gradient could not be computed – the activation functions were step functions – and the solution was just an heuristic (see page 292 from the original report [Rosenblatt, 1961]).

1962 In 1962, Novikoff got published the article "On convergence proofs on perceptrons" [Novikoff, 1962] and H.D. Block "The Perceptron: a model for brain functioning. I" [Block, 1962]. In both papers the authors proved the "Perception Convergence Theorem". For a detailed explanation of the algorithm and the convergence proof consult page 304 of [Bezdek et al., 2013].

In area of neurophysiology, Hubel and Wiesel discovered a more complex type of cells in the visual cortex of cats [Hubel and Wiesel, 1962]. Some of the complex cells were more responsable on detecting contrasts, edges, and rectangular bars at different orientations.

1963 In 1963, Winograd and Cowan proposed a distributed and redundant representation for ANNs following the initial ideas of von Neumann, McCulloch and Pitts [Winograd and Cowan, 1963]. The ideas indicated that it was possible to store and represent simple information with complex structures of large amounts of neurons. This should increase the robustness of the stored information towards small perturbations.

1965 In 1965, Nilsson[30] published the book "Learning machines: foundations of trainable pattern-classifying systems" [Nilsson, 1965]. The book made a great summary about previous work on pattern classification. The main topic was about linearly

neural networks.

separable problems and hypersurfaces.

In 1966, Arhur W. Burdks edited and completed the unfinished work of Von Neumann about automates, that Neumann gave in 1949 as a series of lectures in the University of Illinois with the title "Theory of Self-Reproducing Automata" [Neumann and Burks, 1966]. In these lectures, Von Neumann discussed some of the ideas of McCulloch and Pitts and their work on ANNs. 1966

In 1967, Marvin Minsky published the book "Computation: Finite and Infinite Machines", in which Minsky extended the theories of automatas based on von Neumann, McCulloch and Pitts [Minsky, 1967]. 1967

In the same year, S. Amari[31] used the stochastic gradient method to train a pattern classification model in his publication "A theory of adaptive pattern classifiers" [Amari, 1967]. He claimed that his learning method converged to the optimal set of weight even in the case of nonseparable patterns.

Grossberg[32] was developing mathematical models able to memoryze lists of terms or facts in his work "Nonlinear difference-differential equations in prediction and learning theory" [Grossberg, 1967]. This type of memorization was called short-term memory (STM) also known as Additive and Shunting models and became the Hopfield networks in 1984.

In 1968, Hubel and Wiesel continued doing research on the visual system. In the article "Receptive fields and functional architecture of monkey stritate cortex" they studied the macaque and monkey visual cortex, and discovered different types of cells that were strongly activated when a certain visual pattern was presented to the subject. These types of cells were the simple, complex, and low-order and higher-order hypercomplex cells. 1968

During this period of time, Minsky and Papert[33] carried some research about Artificial Neural Networks (ANNs) and the Connectionism ideas, and they wrote a technical manuscript that was not published. However, the manuscript started circulating at some scientific research groups and was finally published in 1969 with the name "Perceptrons" [Minsky and Papert, 1969]. In this book, the authors demonstrated mathematically some of the most basic limitations on single layer neural networks (e.g. it was not possible to learn the exclusive or logical function). However, they could not demonstrate that these limitations were extensible to multiple layers. The authors were discouraging the use of methods from the field of neurocomputing, as can be seen in some their quotes: 1969

"Our discussions will include some rather sharp criticisms of earlier work

[30] **Nils John Nilsson** (United States; 1933) is an American professor Emeritus of Engineering in Computer Science at Standford University. Particularly known by his contributions in the fields of search, planning, knowledge representation and robotics.

[31] **Shun'ichi Amari** (Tokyo, Japan; 1936) is a professor of mathematical engineering. He is known for "information geometry".

[32] **Stephen Grossberg** (Woodside, Queens, New York City; 31 December, 1939) is an American mathematician, cognitive scientist, neuroscientist, biomedial engineer and neuromorphic technologist. He contributed to the area of competitive learning and Self-Organizing Maps (SOMs) and developed the adaptive resonance theory (ART).

[33] **Seymour Aubrey Papert** (Pretoria, South Africa; February 29, 1928) is an American mathematician, computer scientis and educator. One of the inventors of Logo (programming language).

> *in this area. Perceptrons have been widely publicized as "pattern recognition" or "learning" machines and as such have been discussed in a large number of books, journal articles, and voluminous "reports". Most of this writing (some exceptions are mentioned in our bibliography) is without scientific value and we will not usually refer by name to the works we criticize. ...."* (p.4 [Minsky and Papert, 1969])

Despite of the title of the book "Perceptrons", the authors were not critizising only this model but others with several similarities. For example, the work of Bernard Widrow with his network "Adaline", and despite the fact that it is not mentioned in their book, almost everything could be applied to this model. From the words of Widrow:

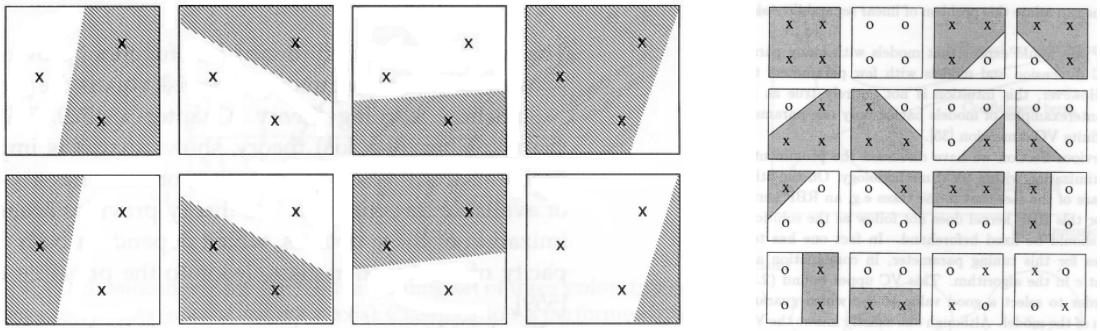
> *"When the Minsky and Papert book came out, entitled "Perceptrons", I somehow got a copy of it. Publishers send me zillions of books, so this one came into my office one day. I looked at that book, and I saw that they'd done some serious work here, and there was some good mathematics in this book, but I said, "My God, what a hatchet job." I was so relieved that they called this thing the perceptron rather than the Adaline because actually what they were mostly talking about was the Adaline, not the perceptron."* (interview by Edward Rosenfeld to Bernard Widrow p.60 [Anderson and Rosenfeld, 2000])

In the same year, Willshaw, Buneman and Longuet-Higgins got published a paper about "Non-holographic associative memory" [Willshaw et al., 1969]. The authors created an optical system capable of storing information in a similar manner than an ANNs. They shown a very simple experiment where it was possible to store visual information in a projected screen. There was a source of light (screen A) that traveled through holes in an opaque screen, the light diverged because of one lens, and hit a second screen B. The authors demonstrated that it was possible to retrieve the original image by projecting back the image from the second screen B. First they projected from screen A to screen B. Then, they marked in the screen B the points where the light hit. Next, they projected light from the points of the screen B through the intermediate opaque lamina and the lens. Finally, the light reached the screen A with a stronger intensity of light at the original points. By adjusting a threshold the level of light it was possible to visualize the original pattern. This model was able to associate multiple inputs to multiple outputs.

6.7 The winter of the Connectionism

1970

During the previous years, researchers approached the problems of neurocomputing by doing qualitative experiments without a strict analytical emphasis. Additionally, the economical and technological situation – and possibly the previous criticism of the Connectionism methods by Minsky and Papert – generated an bad impact in the field of Connectionism. Searching funding for projects was very difficult, and for



(a) Example with 3 points

(b) Example with 4 points

Figure 6.6: **VC dimension.** An hyperplane in a R^2 space can shatter up to 3 points.

that reason, some researchers left the field and moved to more theoretically proven topics. The area of Artificial Neural Networks (ANNs) required new ideas to keep the topic alive and get resources. Additionally, all the previous ideas were inspired by biological processes and claimed to achieve intelligent machines, this level of hype was hurting the area, making the situation worst. However, some researchers continued working on biological neural networks, psychology, and neuroscience, while some early pioneers on ANNs continued doing research in the same direction.

In 1971, Vapnik[34] and Chervonenkis developed a theory to measure the capacity of a model to classify training data and generalize to external data; what they called Vapnik-Chervonenkis dimension (VC dimension). This method was very useful to find upper bounds on the test error for some classification models. Figure 6.6 shows an example of a two dimensional space where 3 points belonging to two classes can always be separated, but not with four. 1971

Some researchers like T. Kohonen, J. Anderson and K. Nakano independently continued the work about associative memories. In 1972, Kohonen[35] got published the article "Correlation matrix memories" [Kohonen, 1972], in which, Kohonen proposed to approximate the data samples with a matrix of parameters. With the sufficient number of units it could store every sample pattern. The author did not use any analogy to neural networks although it could be seen as one, and he continued his research in neural networks. 1972

Anderson[36] got published "A simple neural network generating an interactive memory" with a more biological connotation [Anderson, 1972].

Nakano got published the work "Associatron – a model of associative memory" [Nakano, 1972], where, Nakano related the association mechanism in the human brain with his simplified version of a neural network with only local connections and

[34] **Vladimir Naumovich Vapnik** (Soviet Union; December 6, 1936) is a mathematician, statistician, and computer scientist. He is the main developer of the Vapnik-Chervonenkis theory and co-inventor of the Support Vector Machine (SVM).

closed loops. He also proposed a hardware implementation for the Associatron.

In the same year, S. Amari studied the stochastic states on macroscopic neuron-like elements with analog signals, and made a comparison with the discrete activation functions of McCulloch and Pitts in his work [Amari, 1972] .

1973 In 1973, L.N. Cooper[37] got published the work "A Possible Organization of Animal Memory and Learning" [Cooper, 2012] following the ideas of Anderson about association memories.

In the same year, Malsburg[38] got published the work "Self-organization of orientation sensitive cells in the striate cortex" about a plausible model of the visual cortex in vertebrates [von der Malsburg, 1973]. Malsburg implemented an artificial neural network with 338 neurones connected to 19 cells representing the retinal photoreceptors, and trained the network to recognize specific patterns. The learning algorithm self-organized each neuron to become specialized into some of the sample patterns. After the training, when a pattern from the sample was presented a few neurons were highly active while they become insensitive to patterns not present during the training.

1974 In 1974, P.J. Werbos[39] finished his PhD thesis in Harvard University in which Werbos proposed one kind of backpropagation – at that time called "dynamic feedback" [Werbos, 1974]. Later in 1994, it was recognized by the IEEE Neural Network Society with a Pioneer Award for developing the backpropagation algorithm, and his PhD thesis was reprinted with the title "The Roots of Backpropagation: From Ordered Derivatives to Neural Netwoks and Political Forecasting" [Werbos, 1994].

1975 In 1975, W. Little and G.L. Shaw[40] proposed a probabilistic model of a biological neuron [Little and Shaw, 1975]. In previous cases, the neurons where deterministic, firing when some specific threshold surpassed a certain value. However in their work, the authors point that there are several sources of noise in the biological neurons that can interfere in the deterministic firing of the neurons. For that reason, they proposed the use of stochastic activations that react with certain probability. This approach was used later in 1985 for the Boltzmann Machine (BM).

[35] **Teuvo Kohonen** (Finland; July 11, 1934) is a professor emeritus of the Academy of Finland. Prof. Kohonen made several contributions to the field of Artificial Neural Networks (ANNs), including work on the learning vector quantization and associative memories with his famous Self-Organizing Maps (SOMs) (also known as Kohonen maps).

[36] **James A. Anderson** (Detroit, Michigan; 1940) is an American physicist, psychologist, professor of cognitive science at Brown University. Interested in biology, neuroscience, computer science.

[37] **Leon N Cooper** (New York City, United States; February 28, 1930) is an American physicist and Nobel Prize laureate who co-developed the BCM theory; a physical theory of learning in the visual cortex.

[38] **Christoph von der Malsburg** (Kassel; 8 May 1942) is a German physicist, and neurobiologist. His main interests are in pattern recognition in the brain, neural networks and computer vision.

[39] **Paul John Werbos** (; 1947) is a scientist whose main interests are pattern recognition, artificial neural networks, machine learning. He has also written in areas of physics like quantum mechanics. He was recognized by the IEEE Neural Network Society with a Pioneer Award for developing the backpropagation algorithm in his PhD thesis.

[40] **Gordon L. Shaw** (Atlantic City, New Jersey; 1932 – 2005) is an American physicist, thesis

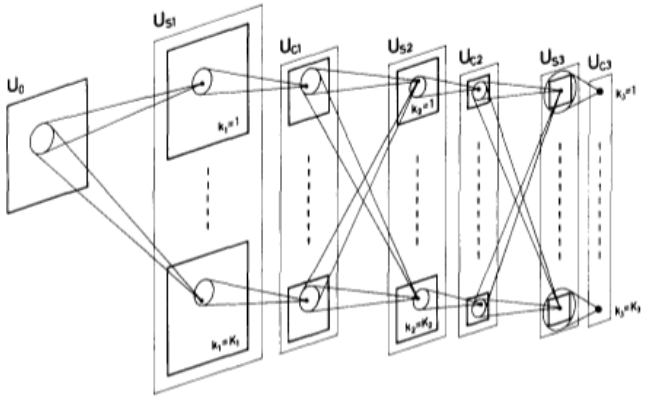


Figure 6.7: **Neocognitron** Schematic diagram illustrating the interconections between layers [Fukushima, 1980]

In 1976, C. von der Malsburg and D. Willshaw got published one of the first articles about Self-Organizing Maps (SOMs) with the title "A mechanism for producing continuous neural mappings: ocularity dominance stripes and ordered retino-tectal projections" [von der Malsburg and Willshaw, 1976]. 1976

In 1977, Kohonen got published the book "Associative Memory: A System-Theoretical Approach" [Kohonen, 1977], where he summarized the previous work about associative memories. 1977

In 1979, Uttley got published the book "Information transmission in the nervous system" [Uttley, 1979] in which he summarized his previous work on neural networks. One of the most known ideas was that the statistical correlation between neurons was determined by the fluctuation of states. Some of his ideas were important later for the maximum mutual information (Infomax). 1979

In 1980 K. Fukushima designed a type of Artificial Neural Network (ANN) that he called Neocognitron [Fukushima, 1980]. It was based on the visual nervous system that Hubel and Wiesel proposed in the 60s [Hubel and Wiesel, 1962, 1968]. The Neocognitron was composed by units that simulated simple and complex cells (S-cells and C-cells). These cells were arranged into different layers specialized in one of the types. The first layer only contained simple cells while complex cells followed at the next layer. The third and fourth layers corresponded to lower-order and higher order hypercomplex cells, but their structure was similar to the simple and complex cells (see architecture in Figure 6.7). Although the biological architecture was not specified for more than four layers, Fukushima stacked several pairs of these layers. The connections of the network were trained in an unsupervised manner, and after the training, the last layer of C-cells was specialized on one specific type of stimulus. Figure 6.7 shows a diagram of the architecture of the Neocognitron. 1980

In the same year, Grossberg got published the article "How does a brain build a cognitive code?" where he discussed a new principle for Self-Organizing Maps

on nuclear physics. After that he research on memory and learning. Famous by his experiment on using music for improving the spatio temporal reasoning. This experiment coined the term of "Mozart Effect".

(SOMs), the Adaptive Resonance Theory (ART); an unsupervised learning model for error correction [Grossberg, 1980].

6.8 Connectionism as a doctrine

1982 After several years of recession in the Connectionism area, researchers started coming back progressively as more applications demonstrated promising results. In 1982, Feldman and Ballard wrote an article that emphasized the benefits of using "distributed representations" instead of the "grandmother cell" approach [Feldman and Ballard, 1982]. In addition, the authors proposed the use of discrete values at the output of the neurons from -10 to 10 to encode binary classifications with the sign and – in addition – a sub-classification with the discrete values from 0 to 10. In their work, they applied the technique to recognize if a specific symbol was a character and in the positive case which was the type of font. The authors also introduced the Winner-Take-All (WTA) idea; if a neuron received various signals, the unit with the largest value inhibited the responses of the others.

Another important work came from J.J. Hopfield[41] . In his work "Neural networks and physical systems with emergent collective computational abilities" [Hopfield, 1982]. Hopfield presented a theoretical foundation for an addressable memory with binary threshold nodes. Each unit was connected to the rest with symmetric weights to guaranty the convergence of the network – after various iterations – into a stable low energy level state. The publication was written with terms from physics and was based on the principle of energy minimization. For that reason, the publication got the attention of physicists and they started showing interest on these type of models. This architecture was a special case of the Cohen and Grossberg neural network that was finally called Hopfield networks (see also [Hopfield, 1984] and a historical summary made by Grossberg [Grossberg, 1988]).

In the same year, Kohonen got published one of his important articles "Self-organized formation of topologically correct feature maps" [Kohonen, 1982]. In this work, Kohonen organized a set of neurons in a 1 or 2 dimensional array and trained the network to map the input signals. The resulting mapping in certain conditions was topologically correct, organizing the different events into different regions of the map. The visualization of the resulting map is usually very informative, although it needs a careful interpretation as the space can be highly complex, and non-linear.

1983 In 1983, the Connectionism ideas got more attention. For example, in the program of Defense Sciences Office at DARPA, the program manager Ira Skurnick opted to try the new approaches that neurocomputing researchers were proposing. From that point, Skurnick started funding research projects that applied these ideas. Also, other research organizations started funding similar projects.

Also, one of the first publications about reinforcement learning was published written by Barto, Sutton, and C.W. Anderson. The authors exposed a control problem to stabilize a pole over a movable cart, and solved the problem using a

[41] **John Joseph Hopfield** (Chicago, Illinois, United States; July 15, 1933) is an American physicists. He is known by his work in artificial neural networks and the Hopfield Networks.

neural network that learned by himself [Barto et al., 1983]. The authors needed to solve the problem of "credit assignment" to decide the influence of each parameter various steps before the error. They used two adaptive devices: an associative search element that matched all the inputs to an associative output, and an adaptive critic element that tried to predict the next reinforcement depending on the actual action. The learning rule of the associative search element used the expected prediction to modify its weights. Altogugh they only used one element of each, they pointed that the system should be extensible to multiple elements.

Fukushima et.al. [Fukushima et al., 1983] used his Neocognitron to classify handwritten characters invariant to position. The model worked successfully also by adding some noise to the input pattern.

Kirkpatrick, Gelatt and Vecchi developed the "simulated annealing" algorithm to find better local optima in optimization problems that could be applied to discrete cases [Kirkpatrick et al., 1983]. It was based on the metallurgy, where metals are cooled with a nonlinear and stochastic procedure; and on "Metropolis-Hastings algorithm", a "Monte Carlo method" designed in 1953.

Cohen and Grossberg had been working on addressable memories for some years and they finally got published an article about one type of ANN that could be used as a content addressable memory (CAM) [Cohen and Grossberg, 1983]. It included the Hopfield network as a special.

In 1985 Ackley, Hinton[42] and Sejnowski[43] designed the Boltzmann Machine (BM) [Ackley et al., 1985]. A Recurrent Neural Network (RNN) with stochastic units. It was based on statistical mechanics. 1985

Judea Pearl[44] developed the Belief Networks (BNs); a probabilistic graphical model with directed and acyclic connections [Pearl, 1985]. Three years latter he got published the book "Probabilistic reasoning in intelligent Systems: Networks of Plausible Inference" [Pearl, 1988], a very cited book about plausible reasoning under uncertainty.

In 1985 and 1986, Rumelhart, Hinton and Williams got published two of the most famous articles about backpropagation: "Learning internal representations by error propagation" [Rumelhart et al., 1985] and "Learning representations by back-propagating errors" [Rumelhart et al., 1986]. The authors claimed that the hidden units were able to represent some features of the original patterns that were not present in the input space. Although the idea of propagating the error was not new, this was one of the first times to apply the backpropagation algorithm to train a 1986

[42] **Geoffrey Everest Hinton** (Wimbledon, London; December 6, 1947) is a British and Canadian computer scientist and psychologist. His main interests are in artificial intelligence, artificial neural networks. He is the co-inventors of Boltzmann machines, backpropagation and contrastive divergence.

[43] **Terrence Joseph Sejnowski** (Cleveland, Ohio, United States; 1947) is an American physicist. He got the PhD from Princeton University in 1978 under the supervision of John Hopfield.

[44] **Judea Pearl** (Tel Aviv, British Mandate for Palestine (Israel); 1936) is an American electrical engineer, computer scientist and philosopher. His research interests is in artificial intelligence focusing on probabilistic approaches.

ANN. For example, the same concept was proposed in Werbos PhD thesis [Werbos, 1974], and was simultaneously applied by Yann Le Cun[45] in his PhD thesis [LeCun, 1985; Le Cun, 1986] to propagate "virtual target values" and one year earlier by Parker in his report "Learning-logic" [Parker, 1985]. Older applications of similar methods for optimal programming and control from Bryson et.al. date from the 60s the "Optimal programming problems with inequality constraints" [Bryson et al., 1963] and the book "Applied Optimal Control" [Bryson, 1975].

1987 In 1987, the International Neural Network Society (INNS) was formed, and the first open conference on Neural Networks was organized "IEEE International Conference on Neural Networks" in San Diego with 1700 attenders.

In the same year, Sejnowski and Rosenberg demonstrated the usefulness of MFNN with NETtalk [Sejnowski and Rosenberg, 1987], a neural network that learned to pronounce English text. The authors demonstrated empirically that the internal representation was distributed through all the neurons, making it robust small perturbations.

1988 In 1988, Broomhead and Lowe proposed the use of Radial Basis Function (RBF) networks in his publication [Broomhead and Lowe, 1988]; a type of ANN where the activation functions are radial basis (commonly a Gaussians).

Also, one of the most famous books of Rumelhart and McClelland "Parallel Distributed Processing, Volumes I and II" was published [Rumelhart et al., 1988]. The authors made a summary about all the Connectionism ideas, and how the biological nervous system is able to represent all the concepts in a distributed way throughout the neural network.

Linsker[46] got published an article exposing the idea of maximum mutual information (Infomax) [Linsker, 1988]. The idea relied on the maximization of the retention of information when some constrains are applied. Linsker focused on the visual perception of animals, but he made analogies to other problems. This idea was the basis for the posterior development of the Independent Component Analysis (ICA)

1989 In 1989, the academic journal Neural Computation was founded, focused on the topics of psychology, physics, computer science, neuroscience, and artificial intelligence, among others. .

In the same year, Yann Le Cunn collected a small dataset of hand-written digits (predecessor of the MNIST dataset) with 480 binary images of size 16×13 pixels, and demonstrated that constraining the parameters of a MFNN reduced the size of the network while improved its performance [LeCun, 1989]. This network was one of the first successful Convolutional Neural Networks (CNNs) and was called LeNet

[45] **Yann Le Cun** (near Paris, France; 1960) is a researcher in computer science, with special focus on machine learning, computer vision, mobile robotics, computational neuroscience. During his PhD he proposed one version of backpropagation. He designed one of the first Convolutional Neural Networks (CNNs) called LeNet and measured its performance with a handwriting recognition dataset with successful results. He participated in the creation of the image compression technology DjVu.

[46] **Ralph Linsker** (Canada; September 12, 1956) is a researcher in the IBM T.J. Watson Research Center. His main interests are in the field of neuroscience, machine perception and leanring.

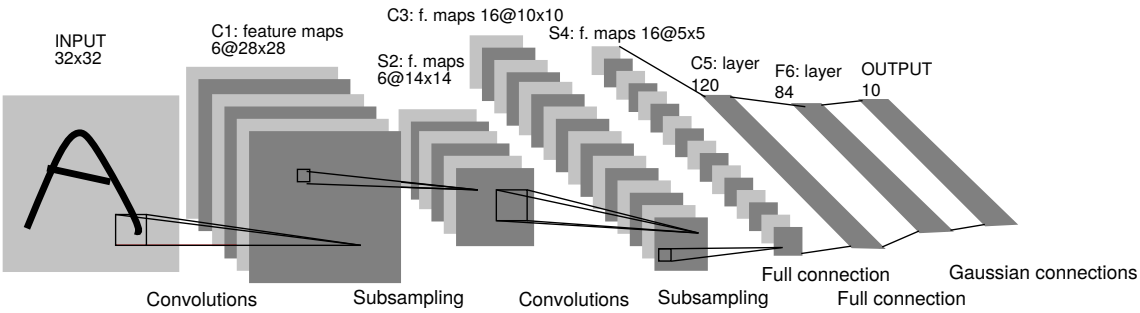


Figure 6.8: **LeNet-5.** One of the first implementations of a Convolutional Neural Network (CNN) for handwritten character recognition

(see Figure 6.8. After the success of the CNN on recognizing digits, Le Cunn et.al. collected 9298 hand-written digit images from the U.S. mail of New York post office [LeCun et al., 1989]. The model of this work was posteriorly used in the post offices of the U.S. Additional work was done in 1990 [LeCun et al., 1990]. In the same year, the IEEE Transactions on Neural Networks was founded. Figure 6.8 shows the architecture of LeNet-5. 1990

In 1992, Vapnik et.al. designed a pattern recognition technique called Support Vector Machine (SVM) [Boser et al., 1992]. This technique was able to find the optimal hyperplane that separated the input patterns in a transformed space. It was a non-parametric model that only required to store some of the sample points; the patterns that "support" the hyperplane that they called support vectors. The authors presented a very theoretical foundation and explained the boundaries of its generalization. Furthermore, when selecting a hyperbolic tangent as a kernel function, the resulting classifier could be interpreted as a ANN with one hidden layer and a variable number of hidden units that was chosen automatically during the training and made the separation optimum for the given task. 1992

The same year, R.M. Neal[47] developed the sigmoid belief networks, which he introduced in the article "Connectionist learning of belief networks" [Neal, 1992]. Neal shows empirically that the training of sigmoid belief networks can be faster than in a Boltzmann Machine (BM), as the directed connections of belief network do not need to compute the "negative phase".

In 1995, Bell and Sejnowski proposed to use the principle of Infomax to the blind separation problem [Bell and Sejnowski, 1995]; specifically in the cocktail party problem. The problem consisted on the separation of different voices and sounds that were previously recorded by various sources in a room with people talking and other noise. 1995

In the same year, Hinton, Dayan, Neal and Frey designed the "wake-sleep" algorithm to train simultaneously recognition and generation models with the same hidden representations [Hinton et al., 1995]. It consisted on two phases: during the wake phase (1), the recognition model was used to infer the latent variables from a

[47] **Radford M Neal** (Canada; September 12, 1956) is a professor of statistics and machine learning in the University of Toronto. He got his PhD in 1995 in Toronnto "Bayesian Learning for Neural Networks" under the supervision of Geoffrey Hinton.

real sample and the generative model was trained to recreate the hidden and visible variables, while in the sleep phase (2), the generative model created a sample and the recognition model was trained to recognize the hidden variables. The authors applied this technique to two directed graphical models that they called the Helmholtz Machine (HM) [Dayan et al., 1995]. More information about the HM and the wake-sleep algorithm can be found in [Dayan and Hinton, 1996; Dayan, 2000; Kirby, 2006].

1996 In 1996, Saul, Jakkolla, and Jordan were working with the problem of training sigmoid belief networks and they demonstrated the possibility to use mean field theory to train them. improves the training of sigmoid belief networks using mean field theory [Saul et al., 1996]. The authors shown the performance on the classification of hand-written digits.

2000 In 2000, the organizers of the Neural Information Processing System (NIPS) pointed that having the words "neural networks" in the title was negatively correlated with the acceptance of the paper, on the other hand Support Vector Machine (SVM), Belief Network (BN) and variational methods were encouraged (p. 1 [Simard et al., 2003]).

2004 In 2004, Jaeger and Hass got published the first paper about Echo State Network (ESNs) [Jaeger and Haas, 2004]. In this work, the authors shown the potential of these networks to predict very complex time series.

6.9 The birth of Deep learning

2006 In 2006, Hinton and Salakhutdinov developed an algorithm to train Belief Networks (BNs) with multiple hidden layers [Hinton et al., 2006]; what they called Deep Belief Network (DBN)s. The idea was to pre-train the network with the same algorithm used on Restricted Boltzmann Machines (RBMs) in each pair of layers, starting from the first pair of layers and ending at the output layer. Once the network was pre-trained, it was finetuned with some steps of the up-down algorithm (a variant of the wake-sleep algorithm [Hinton et al., 1995]), and Contrastive Divergence to speed-up the training [Carreira-Perpinan and Hinton, 2005].

2007 In 2007, after the success of training a Deep Belief Network (DBN) by using an unsupervised pre-training phase [Hinton et al., 2006], the interest on unsupervised learning of deep hierarchical features started growing. Some researchers compared the advances in deep learning with other methods like kernel methods [Bengio and LeCun, 2007].

2009 In 2009 Honglak Lee et.al. designed a probabilistic maxpooling to create a convolutional Deep Belief Network (DBN). This architecture could recognize and generate samples by performing a forward or backward pass [Lee et al., 2009].

Other researchers focused on the analysis of the deep architectures and found that in CNNs the ReLU was a very important activation function, the learned filters were better than hardwired, and multiple stages of features were better than one (deep architectures were better than shallow) [Jarrett et al., 2009].

In the same year, Yoshua Bengio published the book "Learning Deep Architec-

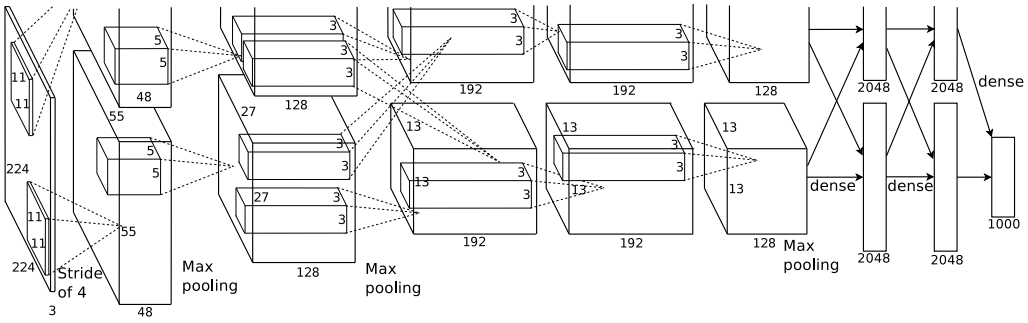


Figure 6.9: **AlexNet deep CNN.** Architecture that won the classification task in the ILSVRC 2012

tures for AI" [Bengio, 2009], in which he summarized the advances in deep learning and the representational power of the new architectures.

> One of the largest datasets of natural images was released with the name ImageNet, and the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2010 is celebrated with two challenges: the image classification and localization. 1.000 categories for the dataset where extracted from WordNet (a large lexical database of English). The first release contained 1.2 million of images but this number grown to 14 million images in 2015. The difference with other datasets was the number of images, categories, and sizes usually larger than 256×256 pixels.

2012

In 2012, researchers from Google published a famous article about large scale unsupervised learning to extract hidden representations from a large number of images (individual frames from YouTube videos). After the training, the researchers found neurons acting as "grandmother cells", showing a maximum activity when a specific object was presented to the network like cat faces, human faces, or human bodies [Le et al., 2012].

In the same year, Alex Krizhevsky and other researchers from University of Toronto participated in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 with a deep Convolutional Neural Network (CNN) reaching the first position on the classification task. It was one of the largest networks until date with multiple convolution, an fully connected layers. The training was fully supervised and most of the improvement was done by using the ReLU activation function, dropout and Local Response Normalization (LRN) (see architecture in Figure 6.9).

2014

In 2013 and 2014, the state-of-the-art approaches for image classifications were deep Convolutional Neural Networks (CNNs) with small changes in the number of layers, convolution sizes and a variety of heuristics [Szegedy et al., 2014; Simonyan and Zisserman, 2014; Ioffe and Szegedy, 2015; He et al., 2015].

Chapter 7

Method and Material

"It is common sense to take a method and try it. If it fails, admit it frankly and try another. But above all, try something."

— Franklin D. Roosevelt

In this chapter, we introduce the methodology followed to elaborate and test the experiments of this thesis. First, we give a brief introduction of the initial analysis and the experiments in Section 7.1 – however more details and a deep description can be found in Chapter 8; page 99. Second, we describe the datasets used in the initial analysis and the experiments in Section 7.2. Third, we discuss our color space selection in Section 7.3. Then, we describe how to merge particular color channels using techniques from multimodal learning in Section 7.4. Finally, we present a list of libraries and software available to train neural networks and explain our motivation to choose the framework used in this thesis in Section 7.5.

7.1 Initial analysis, experiments, and test

We divided the experiments of this thesis into three different phases. First, we evaluated the initial motivation and analyzed various CNNs. The basic idea was to study the importance of luminance and chrominance in the first convolution layer, and the distribution of the weights on each filter. Once the initial analysis was completed, we proceeded with various more specific experiments. Each experiment focused on one specific motivation, and multiple models were analyzed. Finally, we compared certain models having high validation accuracies using various statistical tests. During the statistical tests, we repeated the training of each model several times with random initializations and recorded their validation accuracies. With all the accuracy samples, we performed several statistical tests, each with particular assumptions about the real distribution. More details and an extensive description can be found in Chapter 8.

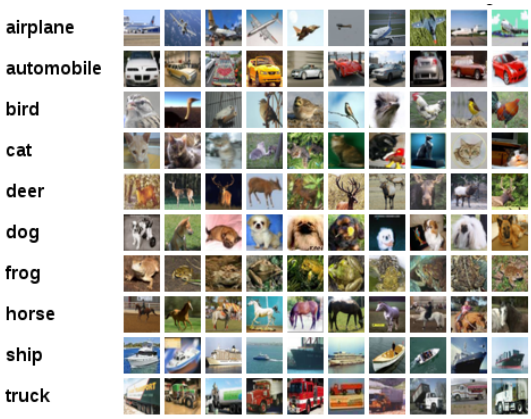


Figure 7.1: **CIFAR-10 examples**

7.2 Datasets

We saw in Section 2.9 a subset of the available datasets for image classification and object recognition. In order to select an appropriate dataset, we had various issues to take into account.

First, the initial motivation of the thesis was based on a CNN trained with the ImageNet 2012 dataset. However, the training of these networks is computationally expensive, thus requiring high performance GPUs and often more than one week of training. For that reason, the initial analysis was performed using the specification of some CNNs previously trained with ImageNet data, but in order to test all the hypotheses and run all the experiments of this thesis we chose a smaller dataset (we refer here as "small dataset" ones with less images, images of smaller height and with a smaller number of pixels, and a reduced number of categories). One smaller dataset is CIFAR-10, collected by Alex Krizhevsky, Vinod Nair and Geoffrey Hinton (see Chapter 3 of [Krizhevsky and Hinton, 2009]). It is composed of 60.000 small images of 32×32 pixels, with every image belonging to one of 10 classes. These classes do not overlap, thus simplifying the training. For example, the class "automobile" does not contain any big cars that could be confused with the class "truck". Figure 7.1 shows ten examples per class.

The reasons for choosing CIFAR-10 dataset include:

1. The reduced size of the images and the small number of classes allowed the size of the network to be reduced.

2. Despite having at our disposal several GPUs in our department, only one was powerful enough to train large models like AlexNet in a reasonable amount of time.

3. The training time of AlexNet required about 8 days, while training the models with CIFAR-10 took approximately 12 hours.

4. Our experiments involved training of multiple architectures and developing

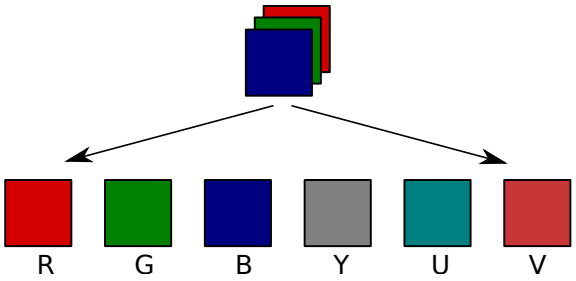


Figure 7.2: **Set of channels used in our experiments.** From the RGB channels it is possible to extract the YUV channels with a linear transformation

statistical tests repeating each experiment several times. This analysis would be unfeasible with large networks.

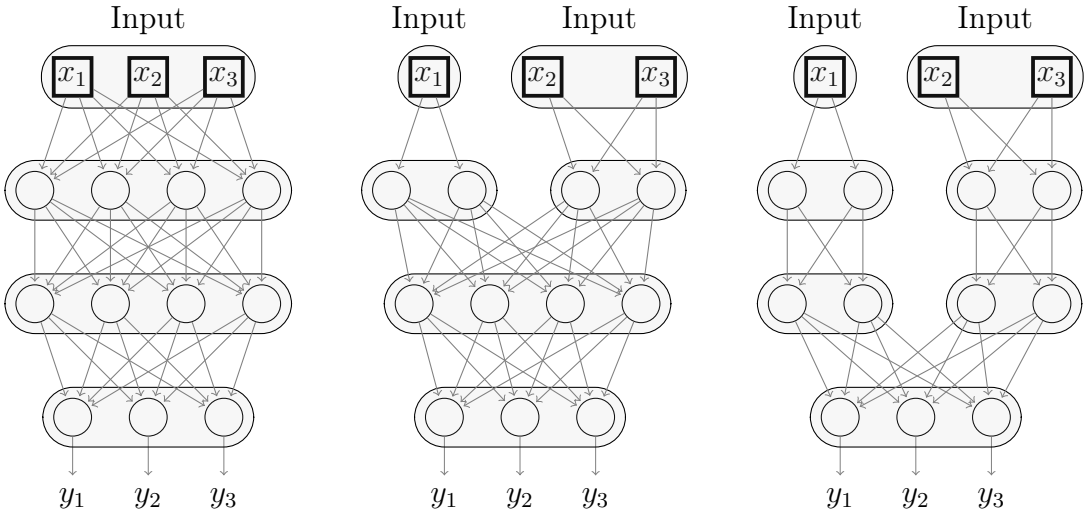
5. In case that the results on CIFAR-10 are successful, it should be possible to try a reduced set of larger architectures on bigger datasets.

7.3 Color spaces

As discussed in Section 2.7, there are multiple color spaces that can be used to represent images. However, our motivation was to exploit the separability of luminance and chrominance. Consequently, we needed a color space that divides the chrominance and luminance information into separate channels. Some available options were HSL, HSV, YIQ, and YUV. From these options, we focused on color spaces that need only a linear transformation from RGB. This restriction was decided for the re-usability of our framework and other implementations. The linear transformation can be a preprocessing step and only implies one matrix multiplication. The same transformation can be done to the average of the images in order to subtract the mean from every input image (this is done as a preprocessing step for each image to help the training of the network). We chose the YUV color space, as it fulfills all the mentioned requirements. Also, YUV has been used in previous works for skin color classification [Chai and Bouzerdoum, 2000] and segmentation [Chai and Ngan, 1999; Phung et al., 2005], facial region location [Chai and Ngan, 1998], and fast image segmentation for interactive robots [Bruce et al., 2000]. Other color spaces are left as a future work. The more common RGB color space was used in different contexts: (1) as a reference model, (2) to test ideas of multimodal learning, and (3) to combine different color spaces in the same network.

7.4 CNN architectures

During the experiments, we proposed and used several CNN architectures and configurations. In some cases, only the number of feature maps was modified. However, most of the experiments focused on learning specific filters for different channels.



(a) Early fusion (b) Medium fusion (c) Late fusion

Figure 7.3: **Multimodal learning example** with three different methods of merging the input features

This idea of merging the input features at different levels has been previously used in the context of multimodal learning.

7.4.1 Multimodal learning

Multimodal learning is a set of techniques that explore how to learn from different sources of features in order to solve a pattern recognition problem (see e.g. [Snoek et al., 2005]). Some examples of applying techniques are the classification of videos using audio, image, and motion features [Ngiam et al., 2011], and learning image representations and their corresponding textual captions [Fang et al., 2014]. As the number of features increase, the solution complexity also grow, and multimodal learning uses different approaches to simplify this problem. One of the approaches is to train individual models for each type of feature and then average all the model predictions. However, this approach does not exploit the possible correlations between the different features. On the opposite side, it is also possible to learn only one model using all the features. Nevertheless, this approach involves a larger number of parameters to learn, thus increasing the complexity of learning and occasionally the training time and the required number of samples. Moreover, in the case of ANNs, if the initial features are not correlated and we assigned random weights in their connections, the learning algorithm will need to suppress these connections (approaching the corresponding weights to zero), thus making the training slower and requiring additional epochs or data samples to converge. These different approaches can be named as *early fusion* when the features are not separated, *medium fusion* when the features are merged after some preprocessing steps, and *late fusion* when the features are merged in a later level of abstraction.

Figure 7.3 shows an example of the three merging methods used in this thesis. The example is a simplification where only three inputs are depicted. Figure 7.3a represents early fusion where all the inputs are merged at the first layer. Figure 7.3a represents medium fusion where the first layer learns individual features from features x_1 and from x_2 and x_3 together. The medium fusion can be seen in the context of this thesis with the luminance channel as a vector of inputs $\mathbf{x}_1$ and the chrominance channels as $\mathbf{x}_2$ and $\mathbf{x}_3$. In this case, the first layers of features learn specific filters for luminance and for chrominance. Figure 7.3c represents late fusion which merges the features in the output layer.

7.5 Software

The great success of CNNs on image classification has accelerated the development of new machine learning implementations and frameworks focused on speed and usability. In addition, the recent advances in computational power and modern GPUs have reduced the training time and the maximum number of parameters per model that can be learned. Several research groups and software developers have implemented excellent versions of CNNs. These implementations are able to explode the most recent computational capabilities. In order to develop these systems in an efficient and scalable way and in a reasonable amount of time, some of the systems are open source and available on GitHub. This enables rapid growth and reuse of the code.

Because of the complexity of these systems and the lack of time to evaluate the experiments during the thesis, we performed an initial analysis of the available frameworks. Some of the available options are described in this section.

7.5.1 OverFeat

Overfeat [Sermanet et al., 2014] is an implementation of the CNN that won the localization task in ILSVRC2013. Overfeat was initially developed by a research group at the New York University and was trained with Torch7 (an extension of the Lua programming language). It is able to use C/CUDA to accelerate the computation. This implementation offers a trained CNN and can be used to extract features or classify images. The authors offer a library in C++ and have developed wrappers for Python and Lua (they are also developing one for Matlab).

7.5.2 Caffe

Caffe [Jia et al., 2014] is a framework for rapid development of various kinds of neural networks. It is implemented in C++ and has wrappers for Matlab and Python. The definition of the neural networks is given in a plain text file, where every layer, activation function, and initialization is specified. Caffe was developed by Yangqing Jia during his PhD at Berkeley University. It can utilize the GPU to accelerate the training process and it has been one of the fastest CNN implementations.

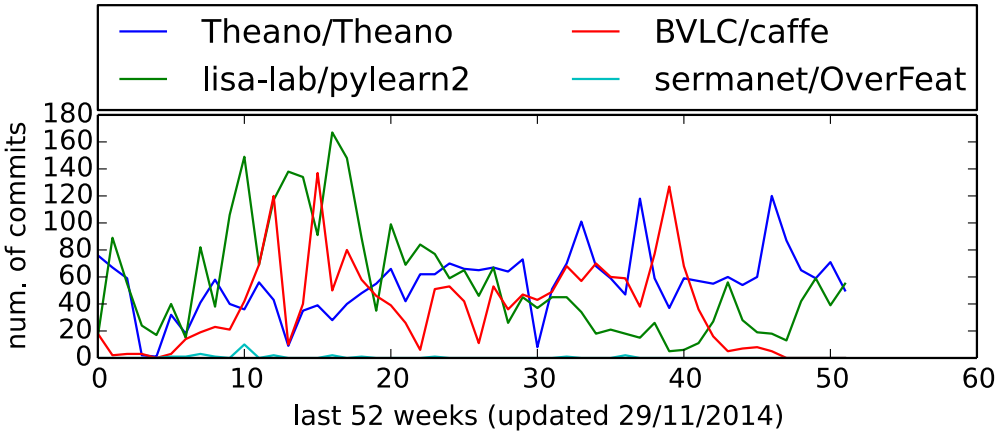


Figure 7.4: **CNN libraries comparison** in number of commits per week during my thesis

7.5.3 Theano

Theano [Bastien et al., 2012] is a Python library that optimizes some parts of the code in C to speed up the calculations. The implementation works on top of NumPy, one of the most extended libraries for Python for scientific programming. The library allows the use of the GPU in a transparent manner. It also implements automatic and efficient symbolic differentiation, given the specification of a function. Theano is being developed by members of the LISA Lab at Université de Montréal. The library is a exceptional tool to learn how different machine learning algorithms work. The authors also offer very good tutorials on how to implement common algorithms using this library.

7.5.4 Pylearn2

Pylearn2 [Goodfellow et al., 2013] is a library implemented on top of Theano that incorporates different machine learning algorithms ready to use. It offers a set of common tools for scientific experimentation, as well as an appropriate structure to use different datasets, models, training algorithms, and tests. It is also being developed in the LISA Lab at Université de Montréal.

7.5.5 Blocks

Blocks is a recent library implemented on top of Theano. It is similar to Pylearn2, as it incorporates already implemented machine learning algorithms. Also, it incorporates some facilities for training, monitoring and optimizing ANNs. It is also being developed in the LISA Lab at Université de Montréal.

7.5.6 In this thesis

In this thesis we chose to use the Caffe framework. The principal reasons for preferring Caffe were: **(1)** the number of parameters to train is huge. Training with a large number of parameters is computationally expensive and we required one of the fastest implementations of CNNs. The Caffe implementation is written in C++ and optimized for Cuda, which accelerate the training and allowed us to test a larger number of architectures. **(2)** We needed to specify a large variety of architectures and the Caffe implementation allowed us to modify a plain text file and run the same exact code with different configurations easily. **(3)** We wanted to try different state-of-the-art networks and some of them are available – or are easy to specify – in Caffe. **(4)** During the thesis the community developing Caffe was very active, improving the implementations and adding new features and architectures periodically (see Figure 7.4).

7.6 Computer hardware

CNNs usually require high computational resources during training. Modern GPUs can accelerate the training from 40 to 140 times with respect to CPUs. However, the memory size of the GPU limits the number of parameters of the network. Nowadays, state-of-the-art networks require a large amount of distributed computers, or high performance GPUs. In the department of Computer Science in Aalto University we had at our disposal one cluster of CPUs and GPUs. Table 7.1 list the GPUs that were available during our experiments. However, only one machine (named "gpu8") was powerful and fast enough to run the experiments with large networks; in a reasonable amount of time.

machine name	CPUs	Speed	GPU	c.c.	cores	Mem. (GB)
gpu001:gpu011	12	2666	Tesla M2090	2.0	512	6
gpu0:gpu7	4	1600	GTX480	2.0	448	1
gpu8	4	1600	GTX Titan	3.5	2880	6

Table 7.1: **Different machine architectures used during all the experiments.** The columns stand for: (machine name) name of the machine; (CPUs) number of CPUs; (Speed) Speed of the CPUs in MHz; (GPU) model of the NVIDIA CUDA GPU; (c.c.) CUDA computational capability; (cores) number of cores in the GPU; (Mem.) Memory size of the GPU in gigabytes.

Chapter 8

Experiments

"It's not an experiment if you know it's going to work."

— Jeff Bezos

In this chapter, we explain the motivation for each performed analysis and experiment. First, we give a description of the initial analysis in Section 8.1. This section discusses the original hypothesis about the importance of chrominance and luminance in the first convolution layer in two different networks.

Next, we present a detailed description of each individual experiment in Section 8.2. Each of these descriptions include: (1) the motivation for the experiment, (2) the color spaces that were used, (3) the architectures that were compared, and (4) details about the number of feature maps in each layer.

Finally, we compare the best architectures with multiple executions to check the statistical significance of the findings in Section 8.3. We also give a brief explanation about the different statistical tests in the same section.

8.1 Initial analysis

The initial analysis evaluated one of the principal motivations of this thesis: Do CNNs learn two different types of filters? Are they divided by luminance and chrominance? To answer these questions, we took several pre-trained networks [1] and analysed their weights at the first layer. These weights can be interpreted as visual filters or features. If we assume that the luminance filters do not contain chrominance information, then all the weights must be approximately in the diagonal of the RGB color space. The diagonal contains a grey-scale from the absolute white to the complete dark, and corresponds to all the pixels $p_{i,j,f}$ of the filter f represented as $p_{i,j,f} \in \mathbb{R}^3 = [w_r, w_g, w_b]_{i,j,f}$ where $w_r = w_g = w_b$. On the other hand, the filters that focus on chrominance should not expand excessively in the direction of the luminance axis. However, as the six remaining corners of the RGB cube –

[1] All the models are available for the Caffe framework in http://caffe.berkeleyvision.org/

red, green, blue, magenta, yellow and cyan – are not on the same plane, it is impossible to create an exact chrominance plane. For that reason, we do not compute the amount of chrominance in the features, but only the level of luminance.

We first analyze a modified version of AlexNet [Krizhevsky et al., 2012] (see original AlexNet in Figure A.2 and a modified version in Figure A.3). This CNN won the classification and localization tasks in ILSVRC2012 [Russakovsky et al., 2014]. We analyze this network once it has been pre-trained with the ImageNet 2012 dataset. AlexNet is composed of five convolutional layers – some of them followed by normalization and/or pooling – and of three fully connected layers including dropout. The first convolution layer contains 96 feature maps with a kernel size of 11 and a stride of 4. Although the original network separates the first convolution into two parts, in our case they were not separated.

The second network was designed in Berkeley University based on the network by Alex Krizhevsky and trained with CIFAR10 dataset (see Figure A.5). This network is composed of three convolutional layers, each one followed by pooling and the first two performing a normalization. The last layer is a fully connected layer with ten outputs, that is, one output per class. The first convolution contains 32 feature maps with a kernel size of 5 and a stride of 1.

Both networks perform a rectification of the convolutional layers with the ReLU activation function and both use a softmax layer on top of the last fully connected layer to predict the category of the input images.

Finally, we show a brief analysis of two recent state-of-the-art networks. However, we could not develop an extended analysis in this thesis, and it is left for future work.

8.2 Description of the experiments

In this section we describe the motivation and objectives of each experiment and the architectures that we used to evaluate each hypothesis. We expose a detailed description of each architecture concerning the color space, the number of feature maps on each convolutional layer, and the type of multimodal fusion used. Figure 8.1 summarizes the architectures used in each experiment with simplified diagrams, not including the variation in the number of feature maps.

The Berkeley network is used in all the experiments as a reference model. We found a set of hyperparameters to maximize its performance when using five random crops and mirroring in the RGB color space. Each of the modified architectures use the same hyperparameters with the exception of the number of units and layers. These are explained in detail in each individual section. However, the code that we implemented during the project (that incorporate the color transformations) does not support mirroring or cropping. Therefore, it must be possible to find a better set of parameters for all the architectures, that would improve the validation accuracies. Nevertheless, we leave this optimization as a possible extension of the current work.

Because of the large amount of architectures examined in this thesis, we assigned representative names to each of the networks. Each name contains all the information regarding the number of input channels, the color spaces, the number of layers,

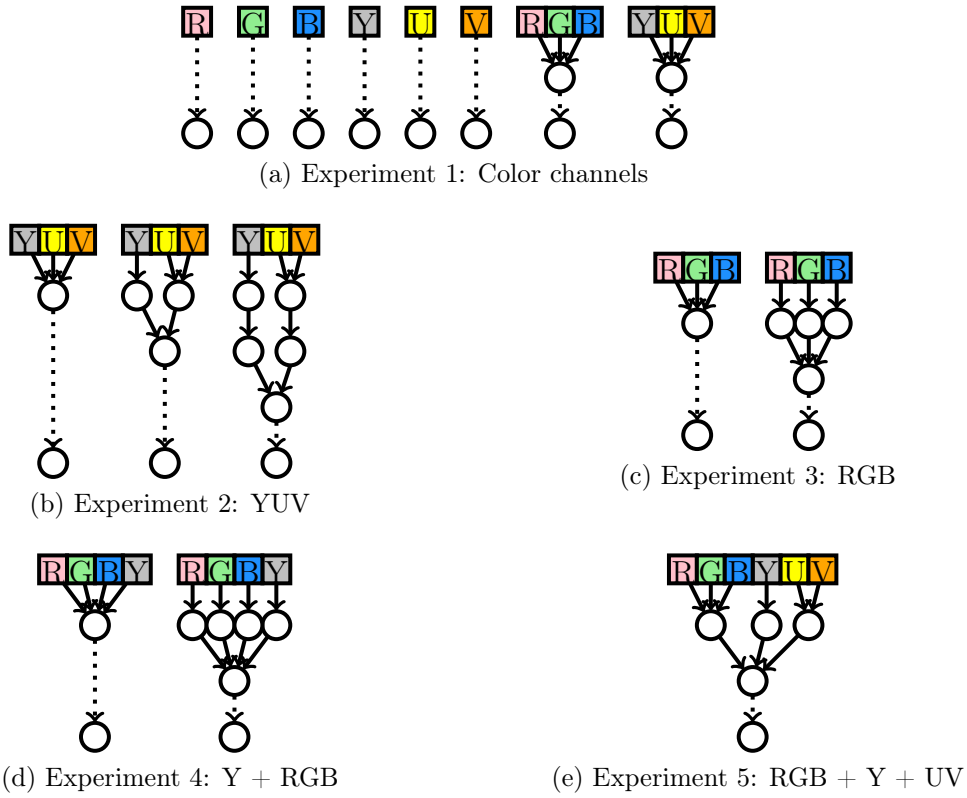
Example	Complete name	Simplified name	Fusion
(1)	rgb32-32-64_E	rgb32_E	Early
(2)	yuv32-32-64_E	yuv32_E	Early
(3)	y32_uv32_32-64_M	y32_uv32_M	Medium
(4)	y16-32_uv16-32_64_L	y16-32_uv16-32_L	Late

Table 8.1: **Examples of nomenclature used per each architecture**

the number of feature maps per layer, and the used multimodal fusion method. To understand the nomenclature we show four examples and explain each part of the name (Table 8.1 summarizes the four examples). **(1)** The reference architecture is the Berkeley University version trained with the RGB channels and symbolized as **rgb32-32-64_E**. The first part **rgb32** indicates that the channels **RGB** are merged in a convolution layer with **32** feature maps. Then each number separated by a dash (**-**) corresponds to a new convolution layer with the specified number of feature maps. The underscore (**_**) is used to finish the current path of convolutions and the character **E** corresponds to Early fusion, while in other cases it can be an **M** for Medium fusion or an **L** for Late fusion. Because all the architectures use this network as a base we simplified the nomenclature to **rgb32_E** and assumed always that there are a total of 3 layers of convolutions where if omitted the second layer contains 32 feature maps and the third layer 64 maps. **(2)** The next example uses the YUV color space and is merged in Early fusion, and is symbolized as **yuv32_E**. **(3)** A more complex example is **y32_uv32_M**, indicating that there are 32 feature maps for the Y channel and 32 feature maps for the UV channels. The next two convolutional layers of 32 channels are omitted (without omitting the last layers it would be represented as **y32_uv32_32-64_M**). **(4)** Finally one of the most complex examples is **y16-32_uv16-32_L**, containing one path of two convolutional layers for the Y channel, the first one with 16 feature maps and the second with 32 feature maps, and at the same time, the channels UV have their own convolutions with 16 and 32 channels. Only the last convolution is omitted here, corresponding to the last 64 feature maps (without omitting the last layer it would be represented as **y16-32_uv16-32_64_L**). As explained previously, the final **L** corresponds to Late fusion.

8.2.1 Experiment 1: Color channels

The first experiment evaluates the importance of each individual color channel for image classification. We test individually the colors red, green, blue channels, and the two chrominance components (U and V) from the YUV color space. In addition, the luminance channel (Y) is also included. We use two reference models to perform the comparison: one trained with all the RGB channels, and another using the YUV channels. The eight architectures use the same hyperparameters and number of feature maps per channel. However, due to the difference in the number of input



(a) Experiment 1: Color channels

(b) Experiment 2: YUV

(c) Experiment 3: RGB

(d) Experiment 4: Y + RGB

(e) Experiment 5: RGB + Y + UV

Figure 8.1: **Simplified diagram of different architectures per experiment.** Note that in some experiments the same architecture is used with different numbers of feature maps and these diagrams do not represent these changes. In addition, we only show the separated layers, merging layers, and output layers. The dashed lines indicate the remaining convolution layers.

channels the initial convolutions have different numbers of parameters. In the models with only one channel there are $1 \times 5^2 + 1 = 26$ parameters per initial feature map, while the models with three channels contain $3 \times 5^2 + 1 = 76$ parameters. Table 8.2 shows the different networks and their respective numbers of parameters.

model	**#parameters**
One feature channel: red, green, blue, Y, U, V	3.88e+05
Three feature channels: RGB, YUV	3.89e+05

Table 8.2: **Experiment 1: models and numbers of parameters**

8.2.2 Experiment 2: YUV early/medium/late fusion

This is the main experiment of the thesis and tries to answer the following question: Does merging the luminance and chrominance in medium or late fusion deteriorate or improve the performance of the network? If we assume that during training the

CNN creates two differentiated types of filters, for chrominance and luminance, then it should be possible to learn these filters and then merge them in an upper layer. To test this hypothesis, we train three different architectures using luminance as one feature and the two chrominance channels as another set of features. These features are merged in early, medium or late fusion. Aside from the merging level, we compare different numbers of feature maps for each architecture. Table 8.3 shows the different networks and their respective numbers of parameters.

model	**#parameters**
yuv32_E	3.89e+05
yuv64_E	4.17e+05
y12_uv20_M	3.88e+05
y20_uv12_M	3.88e+05
y16_uv16_M	3.88e+05
y32_uv16_M	4.01e+05
y32_uv32_M	4.15e+05
y16-16_uv16-16_L	3.75e+05
y16-32_uv16-32_L	4.4e+05
y32-32_uv32-32_L	4.66e+05

Table 8.3: **Experiment 2: models and numbers of parameters**

8.2.3 Experiment 3: RGB early and medium fusion

Experiment 2 demonstrated good performance with medium fusion, but not with late fusion (See the results on Section 9.2.2). In this experiment we study medium fusion in the RGB color space. However, if we separate the three basic channels at the beginning, the filters can only learn 1/3 of the luminance and 2/3 of the specific color in each channel. Then on the second layer, the models can combine the initial filters to create gray-scale filters. We use the RGB model with early fusion as a reference. For the medium fusion we try different numbers of feature maps (8, 11, 20, 30, 40 and 50 filters per channel). Table 8.4 shows the different networks and their respective numbers of parameters.

8.2.4 Experiment 4: RGB + Y early and medium fusion

The fourth experiment checks if it is possible to improve the performance of models that use RGB by adding an additional luminance channel. The Y channel should focus on the learning of luminance filters, while the RGB channels can focus entirely on the chrominance part. However, the RGB channels can still create some variations in the luminance component if it is strictly necessary. In this experiment the RGB

model	#parameters
rgb32	3.89e+05
r8_g8_b8_M	3.81e+05
r11_g11_b11_M	3.89e+05
r20_g20_b20_M	4.11e+05
r30_g30_b30_M	4.36e+05
r40_g40_b40_M	4.61e+05
r50_g50_b50_M	4.85e+05

Table 8.4: **Experiment 3: models and numbers of parameters**

channels are always merged in early fusion, while the Y channel is merged in early or medium fusion. Table 8.5 shows the different networks and their respective numbers of parameters.

model	#parameters
rgb	3.89e+05
yrgb32_E	3.9e+05
y16_rgb16_M	3.89e+05
y32_rgb32_M	4.16e+05

Table 8.5: **Experiment 4: models and numbers of parameters**

8.2.5 Experiment 5: RGB + Y + UV medium fusion

This experiment is an extension of the fourth one while the previous experiment we added a redundant luminance channel, in this experiment we further add two chrominance channels. In this case, the Y channel should focus on creating the luminance filters, the UV channels should focus on chrominance, and the RGB channels should create the remaining non-specialized filters; or specialized if required. In this case, the RGB channels are merged in early fusion, as well as the U and V channels in their own specific layer. Then, all the features are merged in medium fusion. The difference between the architectures resides in the number of feature maps in the first layers. Table 8.6 shows the different networks and their respective numbers of parameters.

8.3 Tests of Significance

All the previous experiments compared one training execution per each model. However, ANNs are randomly initialized making it uncertain if their performance dif-

model	**#parameters**
rgb32_E	3.89e+05
rgb11_y11_uv11_M	3.9e+05
rgb22_y11_uv11_M	3.99e+05
rgb11_y22_uv11_M	3.99e+05
rgb11_y11_uv22_M	3.99e+05
rgb22_y22_uv22_M	4.18e+05
rgb33_y22_uv22_M	4.27e+05
rgb22_y33_uv22_M	4.27e+05
rgb22_y22_uv33_M	4.27e+05
rgb33_y33_uv33_M	4.46e+05

Table 8.6: **Experiment 5: models and numbers of parameters**

ferences are caused by the architecture or a random component. Consequently, we perform a statistical tests with the models that demonstrated high validation accuracies by training ten models per architecture randomly initialized.

In general, statistical tests require independent observations in their samples. However, in our experiments the validation accuracies at different epochs are correlated. Therefore, we performed one statistical test per epoch. This was, obtain a curve per each model indicating the p-values during the training.

Some statistical tests assume that the samples follow a specific distribution. Nonetheless, we do not know the underlying distribution of the accuracies when we randomly modify the initial parameters. Consequently, we performed a set of tests that assume different distributions and compare if there exist discrepancies.

We use the following methodology to carry out the statistical tests:

1. We formulate the null hypothesis H_0 and the alternative hypothesis H_a
   - H_0 :Observations are the result of pure chance
   - H_a :Observations show a real effect combined with a component of chance variation
2. We identify a test statistic that fits our assumptions.
3. We choose an acceptance value α to refute or accept the hypothesis (in our case $\alpha = 0.04$).
4. We compute the p-value, which is the probability that the given samples have been generated given the assumptions and the null hypothesis.
5. We compare the p-value to the previously chosen α. If $p \leq \alpha$ then the observation is statistically significant. This means that the null hypothesis is not statistically plausible and the alternative hypothesis is accepted.

The next list summarizes the different statistical tests used, and their respective assumptions.

- Common in all tests
  - Individuals are randomly selected from the population
  - Each sample is independent from the rest
- Wilcoxon rank-sum
  - The samples are ordinal
  - Homoscedasticity (equal variance)
- Welch-Sapin Test (or Welch's t-test)
  - Two-sided test
  - Normally distributed
  - Sample sizes are likely the same size
- T-test of two independent samples
  - Two-sided test
  - Normally distributed
  - Homoscedasticity
  - Sample sizes are likely the same size
  - Recommended when less than 30 samples are available
- One-way ANOVA test
  - Reasonably normally distributed
  - Reasonably same standard deviation

Chapter 9

Results

"There are three principal means of acquiring knowledge... observation of nature, reflection, and experimentation. Observation collects facts; reflection combines them; experimentation verifies the result of that combination "

— Denis Diderot

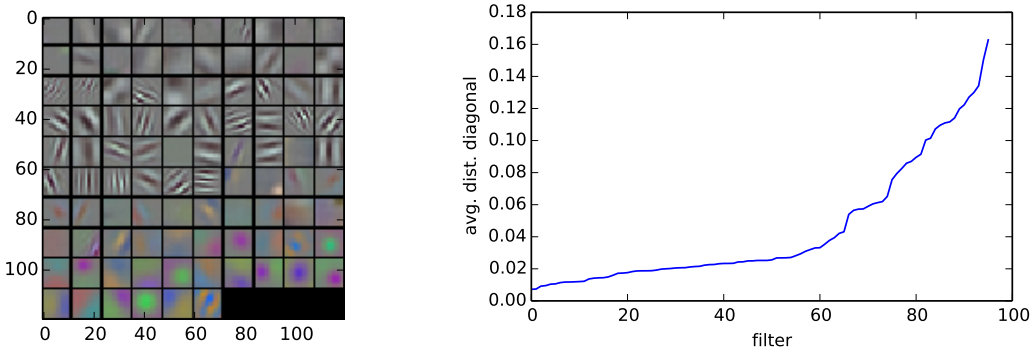
In this chapter we present all the results obtained during the thesis. The results are divided in three sections. Section 9.1 shows a detailed description of the first layers of filters in two CNNs. We also perform a superficial analysis of two state-of-the-art networks from ILSVRC2014. Next, in Section 9.2 we present the results of all the experiments and discuss the results with the initial motivation. Finally, Section 9.3 presents the results of the significance tests for the models with larger validation accuracy.

9.1 Initial analysis

In this section we analyze the first feature maps of two CNNs. Section 9.1.1 discusses the well-known Alexnet architecture trained on the ImageNet 2012 dataset to classify natural images. Section 9.1.2 focuses on an architecture by Berkeley University trained on CIFAR10 dataset to classify tiny images.

9.1.1 Alexnet filters for ImageNet dataset

Since 2012, Alexnet has been one of the most well-known CNNs. In this analysis we use a network by University of Berkeley based on AlexNet (see Figure A.2 for the original AlexNet and Figure A.3 for the University of Berkeley version). The first layer of this network contains 96 feature maps that are depicted in Figure 9.1a. We can observe that some of the filters contain luminance information in particular, while the rest contain different colors. In this figure, the filters have been sorted by the amount of luminance in a descending order. The measure used is the sum of



(a) First convolution filters

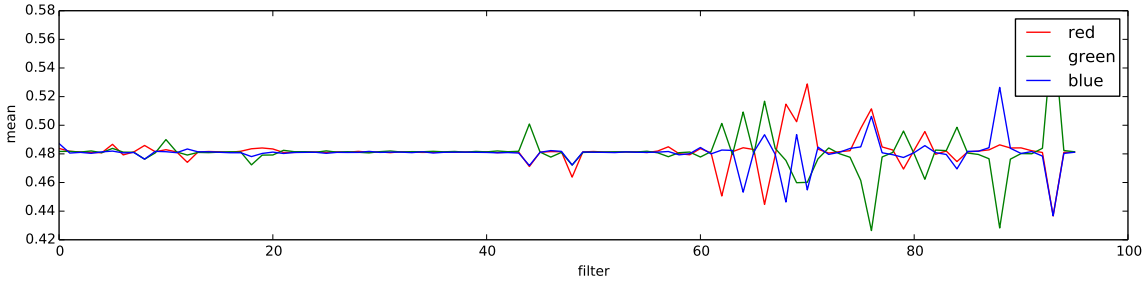
(b) Luminance information per filter

Figure 9.1: **First convolution filters in Alexnet** - In (a) the filters have been sorted by amount of chrominance in ascending order. They are sorted from left to right and top to bottom. Their respective amount of chrominance is depicted in figure (b)

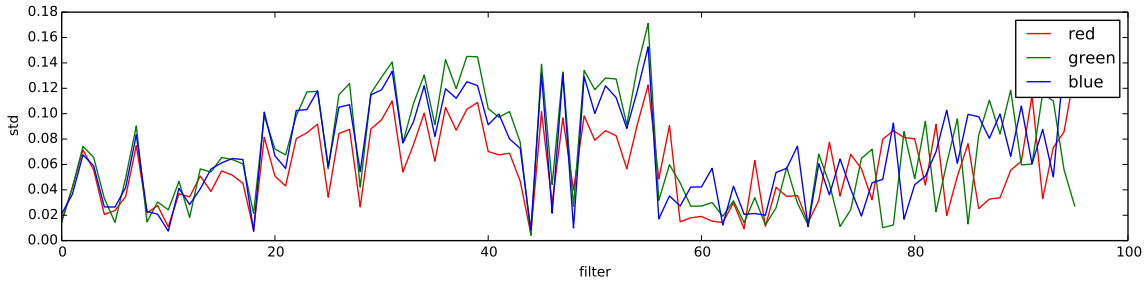
Euclidean distances from each "pixel" of the filter to the luminance diagonal line. The plot at the right side shows the value of the sum per each filter. We see that approximately the first 60 filters do not contain large amounts of color, but are not completely zero either. After about the 60th filter, the amount of color grows exponentially. It seems clear by this plot that there exist two different regions.

Figure 9.2 shows the mean and standard deviation per each channel and filter. The mean of the first approximately 59 filters is almost equal for all the channels. Also their standard deviation follows the same pattern in the three channels. Curiously, the green channel has a larger variance than blue, and blue more than red. (We will see later in the analysis and in Figure 9.3c that this could be the reason of the greenish/bluish white part and the reddish black part present in all the filters). The next filters show different means per each specific color, while their standard deviations are not even.

Finally, Figure 9.3 depicts the distribution of all pixels of the first filters. Although from the figure we can not see the real dependencies between the different pixels, their distribution and the previous filter visualization in Figure 9.1 give us some information about how to interpret them. It shows clearly that not the whole *RGB* color space is useful for the classification task. If we imagine all the possible values in an *RGB* cube, most of its inner space is empty and the filters seem to focus on some specific directions. Also, these directions show a large dependency in all the original coordinate axes and are not perpendicular to any of them. The most prominent axis is the luminance diagonal, containing the largest amount of points in a line. Figure 9.3c shows the first 56 feature maps, together with the boundaries of an *RGB* cube and the ideal diagonal. However, there is a small difference between the ideal diagonal of the cube and the diagonal made by the filters. The whitest color in the filters is not completely white but greenish or bluish, and the black is inclined to brown or redd color. This could be the cause of the previously mentioned difference in variances for the green, blue and red channels. In addition, Figure



(a) Mean weight values for the three channels



(b) Standard deviation for the three channels

Figure 9.2: **Mean and standard deviation of first filters of Alexnet in the RGB colorspace**

9.3d shows the last 40 filters. It shows that there are multiple components that are almost orthogonal to the luminance axis. They do not depict a perfect plane but a star with the vertices at different degrees. Nevertheless, these chrominance axes could be described using only a pair of orthogonal axis on a plane orthogonal to the luminance.

On the other hand, we can visualize the same filters after applying a linear transformation of the *RGB* color space into the *YUV* color space. Using this idea, we can evaluate the importance of each filter towards luminance or chrominance.

Figure 9.4 shows the mean and standard deviation of each channel. First we see that the transformation moves the center of the channels to different locations, the Y channel at approximately 0.63 while U and V at 0.27 (this translation is not important for our analysis). However, we can appreciate that the first filters have a more constant mean than the last ones. Regarding the standard deviation, approximately the first 58 filters exhibit a large variance in the luminance channel, while the chrominance channels have small variances. In contrast, the last filters generally present more variance in chrominance than in luminance.

Figure 9.5 shows the same cloud of filter pixels but in the *YUV* color space. After the transformation, almost all the points from the most prominent axis (black to white) is on the Y axis. The remaining filters then fall in the U and V plane. This clearly indicates that in the first filters the weights towards the chrominance channels tend to zero, whilst for the last filters the weights on the luminance axis tend to zero. All these almost zero weights are useless once the network has been trained, and the learning procedure needs to discover these zeros.

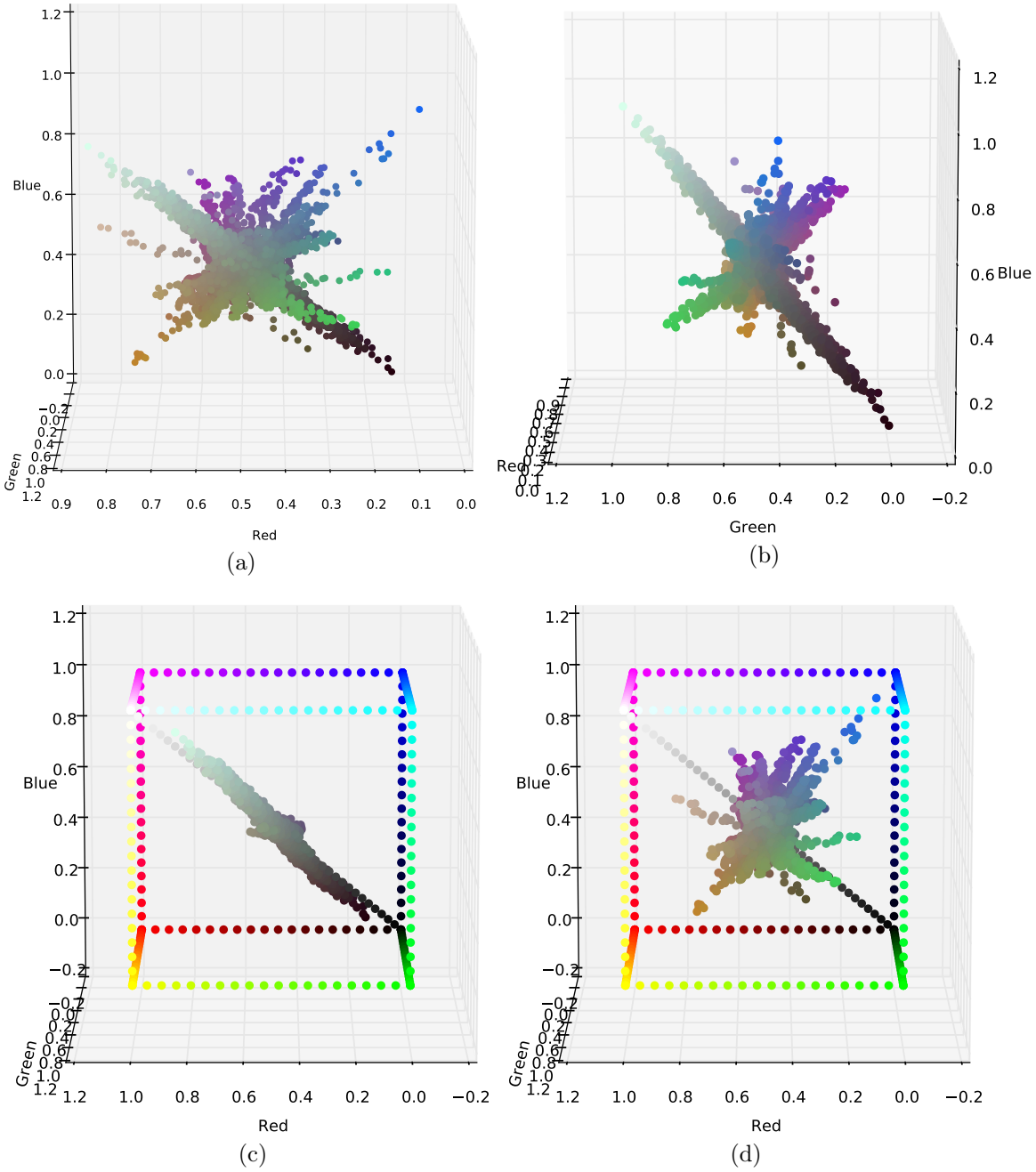


Figure 9.3: **Representation of the initial weights of Alexnet in the RGB colorspace** - (a,b) two different views of the weights, (c) first 56 filters with the boundaries and the diagonal of an *RGB* cube, (d) last 40 filters in the same cube

9.1.2 Berkeley filters for CIFAR10 dataset

We have seen in AlexNet analysis a clear separation between luminance and chrominance filters. Here we perform the same analysis for a smaller network. This network specified by University of Berkeley is originally based on one network by Alex Krizhevsky (see the architecture in Figure A.5).

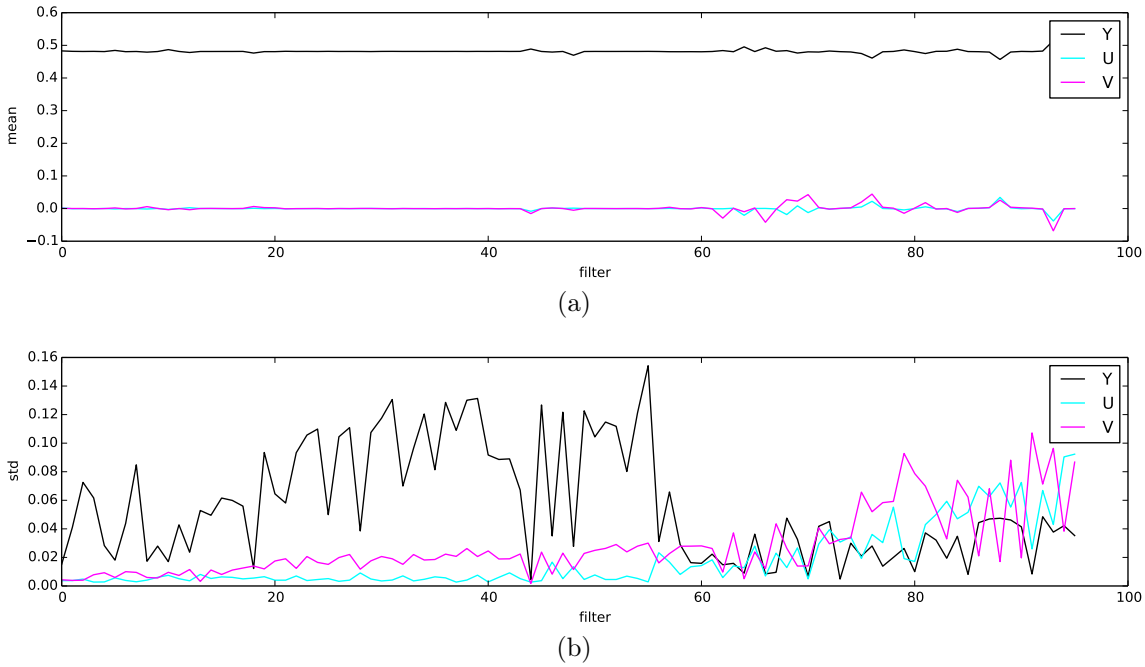


Figure 9.4: **Mean and standard deviation of first filters of Alexnet in the *YUV* colorspace**

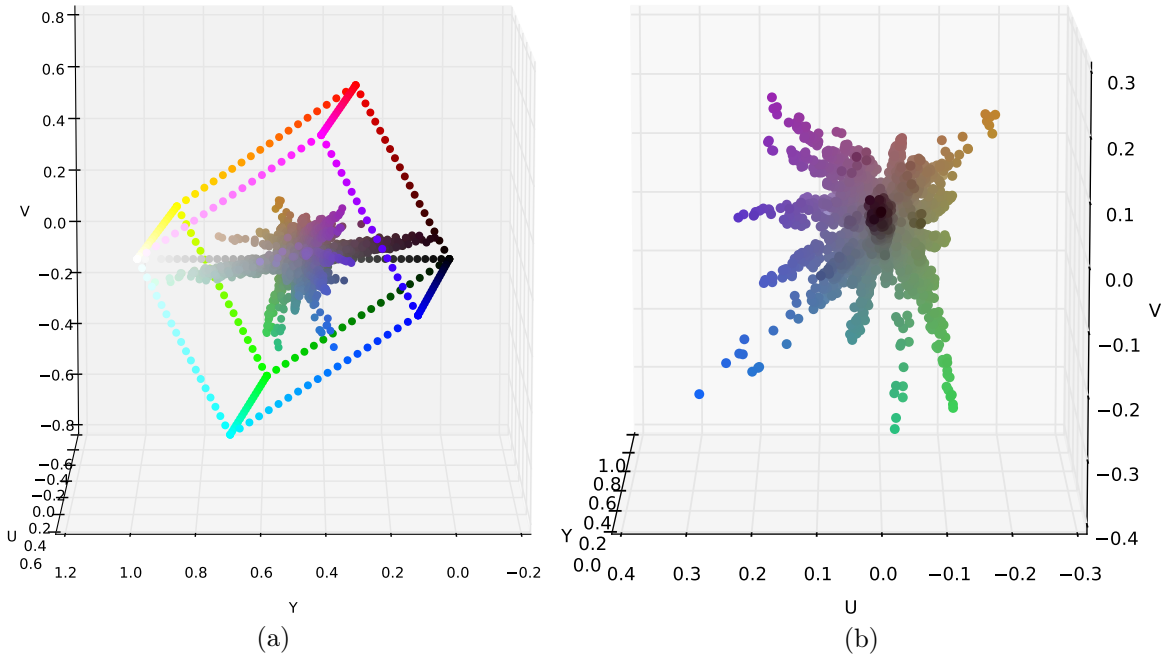
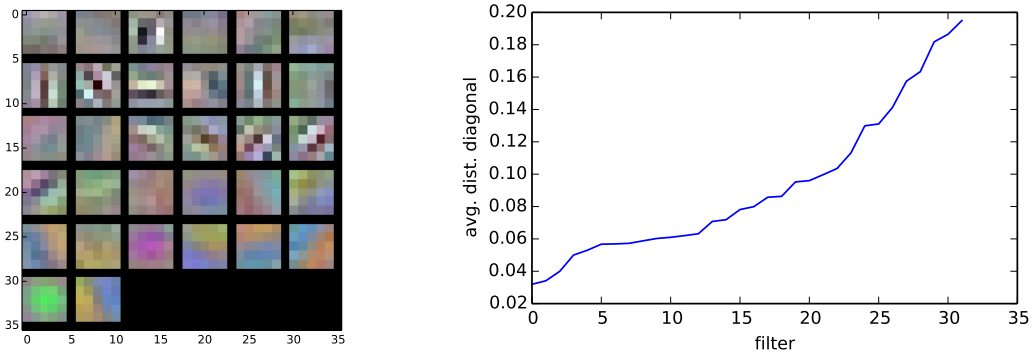


Figure 9.5: **Representation of the initial weights of Alexnet in the YUV colorspace**

Figure 9.6 shows the 32 feature maps of the first layer, sorted by the sum of Euclidean distances to the luminance diagonal. If in Alexnet the difference was



(a) (b)

Figure 9.6: **First layer of filters of Berkeley network trained with CIFAR10 dataset** sorted by the sum of all the Euclidean distances from each pixel to the luminance diagonal

clear, in this case the gradient is more subtle. Although there is still a slight visual difference.

We do not know the reason why in this case it is not separated, however one possible reason could be that the small number of filters force them to be reused on different situations. For instance, a reddish black filter with white strips could be used to detect red patterns or strips indifferently.

The weight distribution can be seen in Figure 9.7. Once more, there exist a principal axis corresponding to the luminance. However, the luminance and chrominance specialization in this case is less pronounced than in the case of AlexNet. If we rotate the pixels towards the YUV color space we see again some pixels distributed in the Y axis, while the rest is mostly spread in the U and V axes. However, in this case the specialization is not very clean. As in the previous case, the luminance axis created by the network is slightly greenish or bluish in the white region and reddish in the black region. Despite of these results, the experiments are focused on this network, and it is hoped they can generalize to AlexNet or newer state-of-the-art networks.

9.1.3 ILSVRC2014 state-of-the-art CNNs

Although the next networks are not intensely analyzed, we show here the first filters of the two winners of ILSVRC14 on classification, detection and localization tasks. Simonyan and Zisserman [Simonyan and Zisserman, 2014] designed a CNN that got the first and the second position in the localization and classification tasks respectively. The authors used very small kernel sizes at the first convolution layers, and stacked multiple convolutions without intermediate layers. Then, they added a maxpooling layer and repeated a similar configuration in the upper layers. The small kernel size and the multiple stacks of convolutions allowed the network to reduce the number of parameters while increasing the deepness and consequent level of abstraction. Because of the high non-linearities that this network involves from the

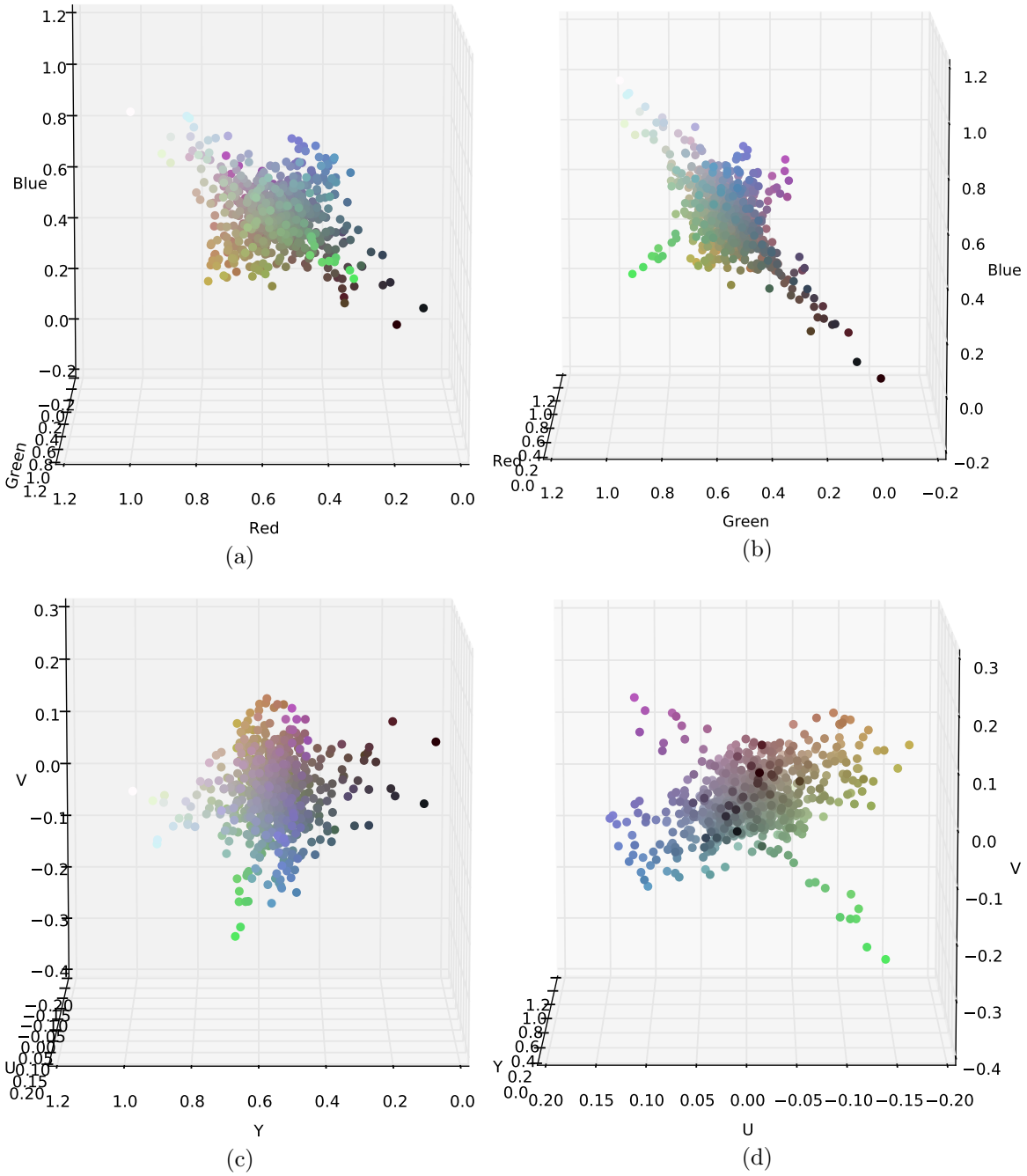


Figure 9.7: **First convolutional filters on Berkeley for CIFAR10.** Representation of the three component weights (red, green and blue) of each pixel of each feature map in the first convolution layer. The color of each point corresponds to the specific most responsive color

very beginning, it is not clear if the methods proposed in this thesis could work. Nevertheless, Figure 9.8b shows that again the filters have some specialization in luminance or chrominance, thus opening the possibility to try different levels for merging the *YUV* channels.

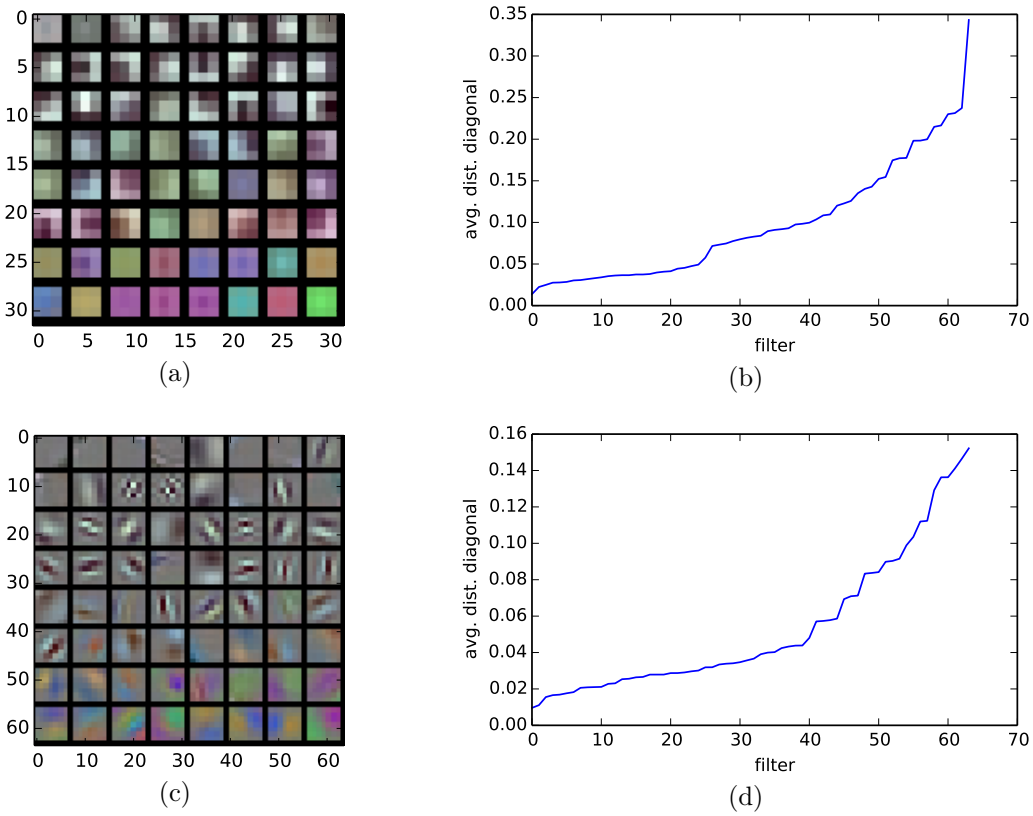
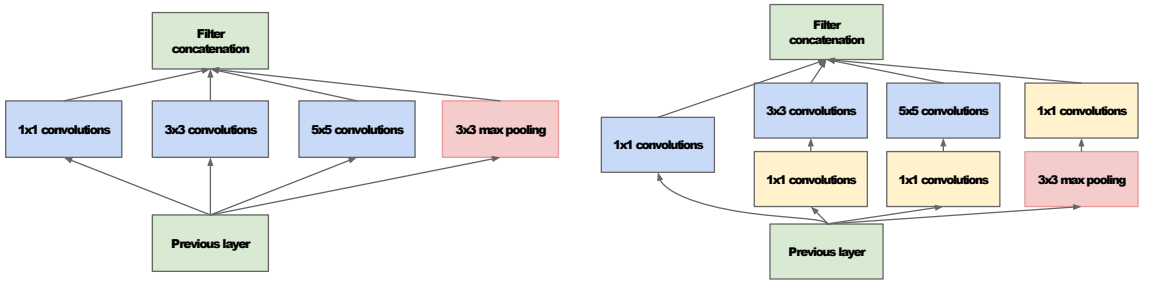


Figure 9.8: **First layer of filters of two state-of-the-art networks on ILSVRC14** - First column the visualization of the filters, second column the average of luminance of the kernels per filter (a,b) very deep neural network, (c,d) GoogleNet

The last network studied here is GoogleNet [Szegedy et al., 2014] (see Figure A.4). This network got the first position in the classification and detection tasks in ILSVRC2014. The authors reduced the number of parameters with respect to Alexnet by a factor of 12. To achieve this reduction, the authors designed a new handcrafted sub-module called "Inception" (see Figure 9.9). Each instance of this module incorporated several convolutions with kernel sizes 1×1, 3×3, and 5×5. On some levels, various "Inception" submodules were stacked without maxpooling layers in between, allowing the network to be deeper, with a total of 22 layers. Figure 9.8c shows the first filters sorted by luminance. Once more, there is a clear separation between luminance and chrominance filters that could be exploited with our approach.

Although the first analysis of the state-of-the-art networks shows the possibility of our method to improve their prediction, there is no place in this thesis to continue the analysis and posterior experiments. These evaluations are left for future work.



(a) Inception module, naïve version

(b) Inception module with dimension reductions

Figure 9.9: **Inception module** Figures from [Szegedy et al., 2014]

9.1.4 Conclusions

After this analysis, we can conclude that CNNs using the *RGB* color space need to coordinate the weights from the three channels to produce the luminance and chrominance filters. However, if we train the network in the *YUV* color space, a large part of the weights tends to zero at the end of the training. Consequently, these weights are useless for two reasons: (1) during the training, the network needs to learn that almost half of the weights are nearly zero, and (2) after the training, these weights do not contribute to the final classification, thus making them useless. For that reason, it must be possible to use all these computational resources and useless parameters to train additional filters or reduce the dimensionality of the network, thus accelerating the training.

9.2 Experiments

In this section we explain the results of all the different experiments. The analysis focuses on the training and test errors, the accuracy on test data, and the size of the models (number of parameters). Although on some occasions, the computation time is an important measure to compare different models, all the experiments that we carried out had similar training times. Moreover, we used a variety of different computer architectures and GPUs, making impossible to do most of the comparisons. Consequently there are no time comparisons in this thesis. However, the Table 9.1 depicts the different machines used during the experiments with their approximated computational times to train the models.

We used CIFAR10 dataset to evaluate the models. The original dataset is composed of 60.000 images, of which usually 10.000 are used for testing and 50.000 for training. We divided the training part into training and validation with 40.000 and 10.000 respectively. This division is commonly used on this dataset. In addition, the size of the validation set offers a very intuitive validation accuracy value as each 0.1 point of accuracy corresponds to 1.000 images correctly classified. Finally, the test data was not used. The reason is that it was not available from the dataset

machine name	GPU	Memory (GB)	time (hours)
gpu001:gpu011	Tesla M2090	6	12
gpu0:gpu7	GTX480	1	13
gpu8	GTX Titan	6	8

Table 9.1: **Different machine architectures used during all the experiments** - each column stands for: (machine name) name of the machine; (GPU) model of the CUDA GPU; Memory size of the CUDA GPU in GigaBytes; (time) approximated time of one training in hours.

source that we used[1].

The hyperparameters were chosen previously for a network trained with RGB channels, 32 feature maps in the first convolution and early fusion. The same hyperparameters were used for the rest of architectures. Every batch contained 100 images, the initial learning rate was 0.001 and it was decreased by a factor of 0.1 every 125 epochs. In addition, we choose a momentum of 0.9 and a weight decay of 0.004.

The basic architecture used is the predefined in Caffe software by Berkeley University shown in Figure A.5

9.2.1 Experiment 1: Color channels

In this experiment, we analyzed the difference between the basic colors and the luminance and chrominance to classify CIFAR images. The experiment was executed once per model during 500 epochs. During the training, the training error and the validation accuracy were periodically annotated. Figures 9.10 and 9.11 show partial information of the training, while Figure 9.12 and Table 9.2 contain a summary.

	Training error			Validation acc.			
model	**min**	**avg**	**std**	**max**	**avg**	**std**	**#params**
Red	0.108	0.36	0.291	0.768	0.725	0.0778	3.88e+05
Green	0.126	0.365	0.291	0.762	0.728	0.078	3.88e+05
Blue	0.123	0.396	0.337	0.761	0.725	0.079	3.88e+05
Y	0.108	0.36	0.294	0.773	0.735	0.0797	3.88e+05
U	0.547	0.868	0.323	0.611	0.584	0.0674	3.88e+05
V	0.439	0.813	0.349	0.597	0.561	0.064	3.88e+05
RGB	0.0571	0.265	0.256	0.803	0.765	0.083	3.89e+05
YUV	0.09	0.315	0.305	0.797	0.765	0.083	3.89e+05

Table 9.2: **Color channels: summary table of results**

The results show that using all available information from the three original channels (red, green and blue) achieved the best validation accuracy. There was a

[1]We got the dataset from www.kaggle.com

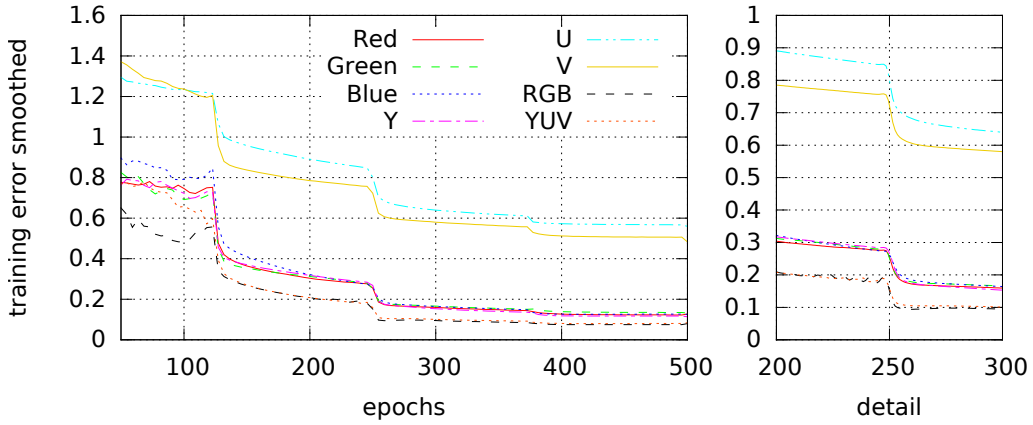


Figure 9.10: **Color channels: training error smoothed**

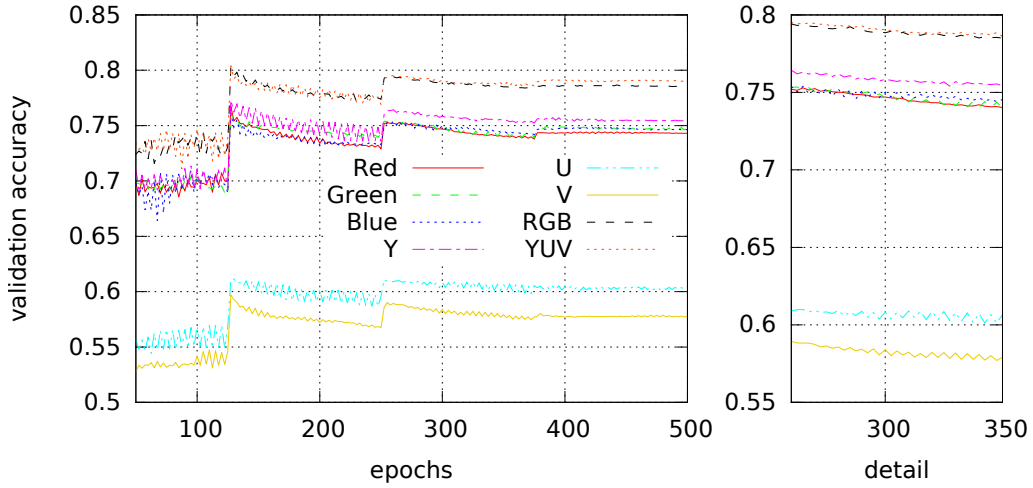


Figure 9.11: **Color channels: test accuracy**

small difference between RGB and YUV models in favour of RGB but the difference did not seem significant. The reason could be related to the random initialization of the parameters, as the YUV model was using simply a linear transformation of the data that could be reversed by the weights of the first layer during the training. The YUV and RGB models contained 1.600 more parameters than the rest, corresponding to the additional two channels in the first layer ($2 \times 5 \times 5 \times 32$).

The Y model got the third position with respect to the validation accuracy. This model used only the luminance of the image (gray scale image). We saw in Chapters 2 and 3 that one of the most important parts in vision is luminance, and some of the image compression techniques use this knowledge to reduce the size of the images without an appreciable difference.

The isolated colors (Red, Green and Blue) performed very similar to the Y channel. The difference was about 0.01 in the average validation accuracy; this difference corresponds to 100 images misclassified. The good performance of isolated colors could be related to the intrinsic luminance that they encode. The luminance is divided between the three channels, and possibly the learning algorithm was able

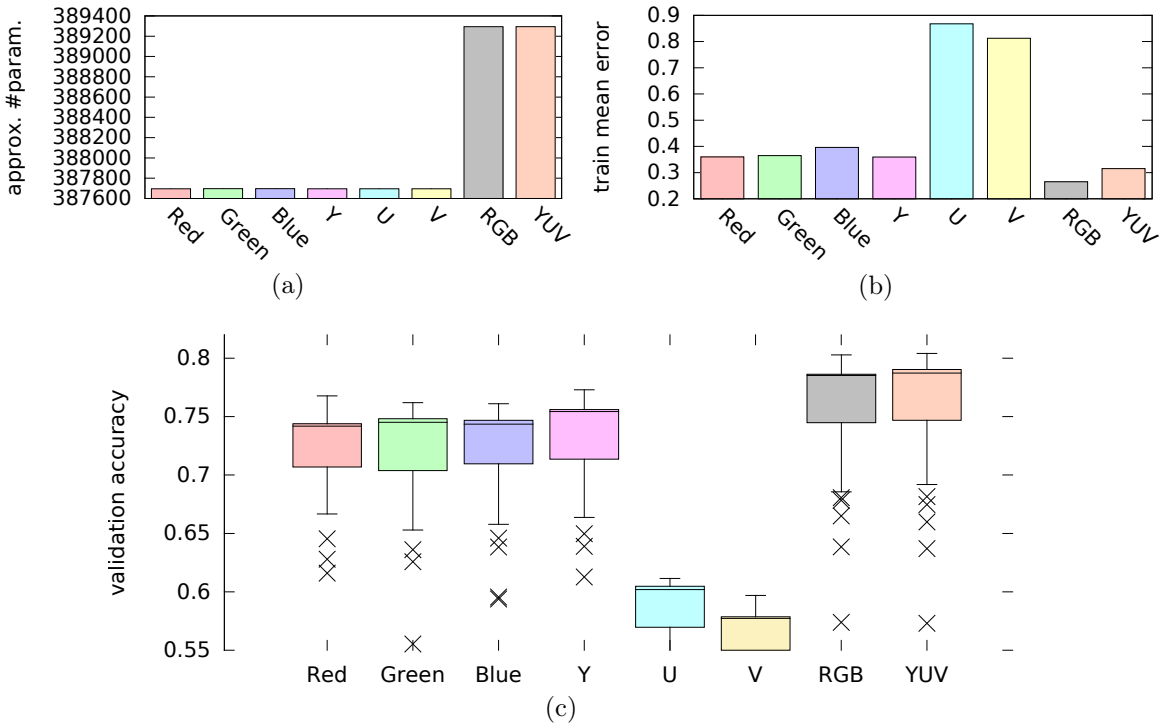


Figure 9.12: **Color channels: experimental results**

to use this information, together with an individual color to achieve reasonably good results.

Lastly, the chrominance part achieved the worst performance on validation accuracy. It is clear here the importance of the luminance for classifying images or objects. There was a small variation between the U (blue difference) and the V (red difference). In general, U was preferred to V on this task as, although the training error was larger, it generalized better. One possible reason could be the difference of their range values, $U \in (-0.436, 0.436)$ and $V \in (-0.615, 0.615)$. We did not however analyze the real reason of this disparity.

9.2.2 Experiment 2: YUV early/medium/late fusion

This experiment explored the difference of merging the luminance and chrominance at different convolution levels. We analyzed several architectures with an increasing number of feature maps and consequently number of parameters. Each model was trained once. Figure 9.13 shows the training error and Figure 9.14 shows the validation accuracy. Table 9.3 and in Figure 9.15 show a summary of the results.

	Training error			**Validation acc.**			
model	**min**	**avg**	**std**	**max**	**avg**	**std**	**#params**
yuv32_E	0.0848	0.278	0.275	0.798	0.767	0.0827	3.89e+05
yuv64_E	0.0783	0.328	0.283	0.798	0.767	0.083	4.17e+05
y12_uv20_M	0.0902	0.299	0.28	0.794	0.762	0.0816	3.88e+05
y20_uv12_M	0.0946	0.33	0.313	0.799	0.767	0.0825	3.88e+05
y16_uv16_M	0.1	0.313	0.279	0.803	0.77	0.0834	3.88e+05
y32_uv16_M	0.0804	0.308	0.284	0.8	0.764	0.0818	4.01e+05
y32_uv32_M	0.0869	0.294	0.288	0.809	0.771	0.0842	4.15e+05
y16-16_uv16-16_L	0.117	0.354	0.295	0.791	0.757	0.0834	3.75e+05
y16-32_uv16-32_L	0.0737	0.283	0.281	0.8	0.767	0.0835	4.4e+05
y32-32_uv32-32_L	0.0694	0.265	0.282	0.806	0.772	0.0827	4.66e+05

Table 9.3: **YUV E/M/L: summary table of results**

The results show that increasing the number of feature maps at the first layer with early fusion did not improve the validation accuracy, although the number of parameters increased and the training error decreased. However, merging the luminance and the chrominance on upper layers got a better validation accuracy, while the number of parameters decreased or increased slightly compared to the early fusion models.

If we focus on medium fusion, we see that y16_uv16_M and y32_uv32_M got the best results on validation accuracy. The first with the lowest number of parameters and with larger training error indicating that its generalization was higher; y16_uv16_M contains 800 parameters less than yuv32_M.

On the other hand, late fusion models improved with the number of parameters but their training error decreased very fast. This behaviour could be bad for

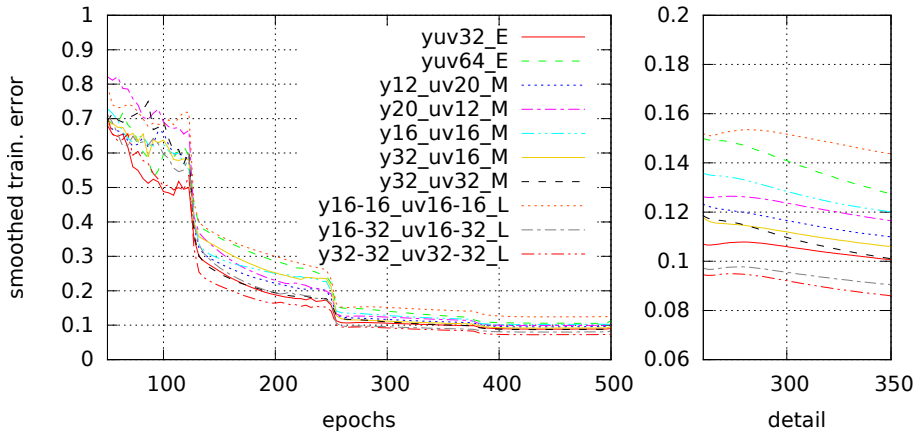


Figure 9.13: **YUV E/M/L: training error smoothed**

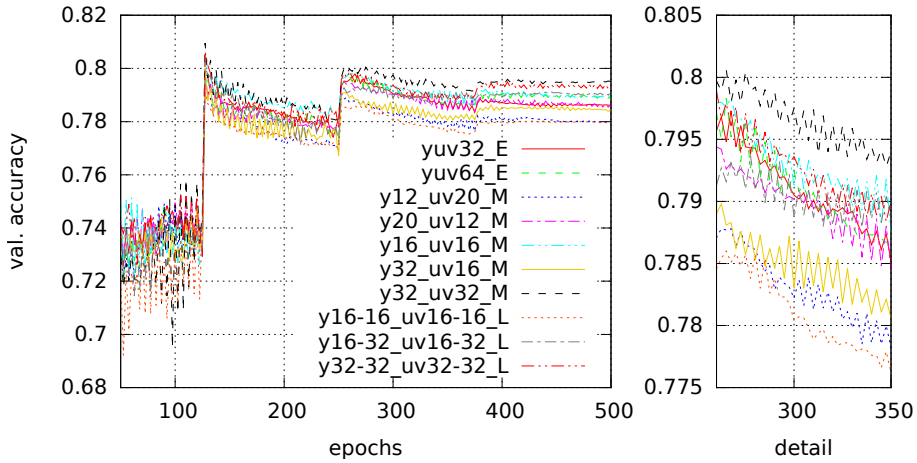


Figure 9.14: **YUV E/M/L: test accuracy**

generalization and it could need more strict regularization.

9.2.3 Experiment 3: RGB early and medium fusion

In this experiment we compared the difference of merging the *RGB* colors at the first layer (early fusion) and at the second layer (medium fusion). This restriction forces the first feature maps to focus on one unique color and one third of the luminance. We compared different number of feature maps at the first layer while the rest of the layers remained with the same number of feature maps.

Figure 9.17 shows the training error while Figure 9.16 shows the validation accuracy. A summary of this experiment can be seen in Figure 9.18 and Table 9.4.

The results show that the best validation accuracy corresponded to the early fusion. Merging chrominance and luminance at the first feature maps enabled the network to decide how to use each part of the colors with the possibility to create the whole gray-scale palette. On the other hand, medium fusion performed 0.01 points worse on average. The inability to merge the colors at the first layer deteriorated

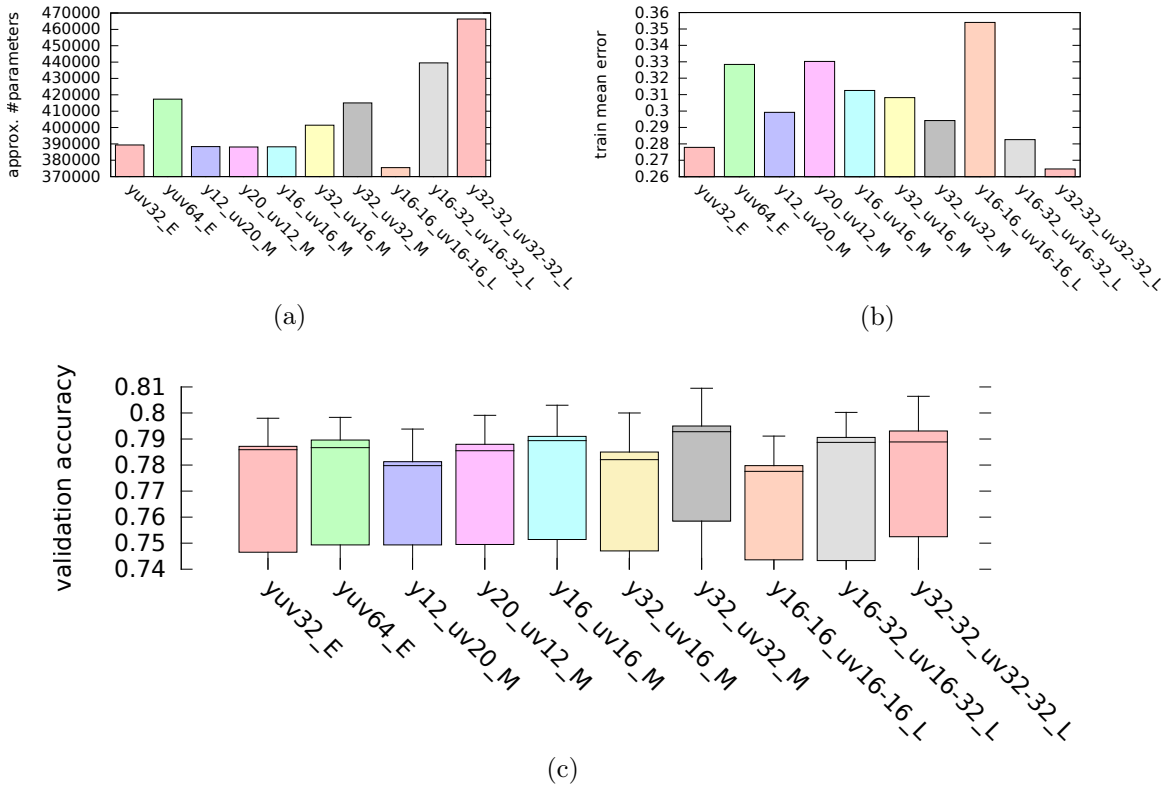


Figure 9.15: **YUV E/M/L: results of experiment**

	Training error			Validation acc.			
model	**min**	**avg**	**std**	**max**	**avg**	**std**	**#params**
rgb32	0.0571	0.265	0.256	0.803	0.765	0.083	3.89e+05
r8_g8_b8_M	0.0853	0.316	0.293	0.781	0.749	0.0805	3.81e+05
r11_g11_b11_M	0.0891	0.313	0.318	0.783	0.752	0.0814	3.89e+05
r20_g20_b20_M	0.081	0.298	0.299	0.788	0.757	0.0824	4.11e+05
r30_g30_b30_M	0.076	0.305	0.306	0.793	0.758	0.0818	4.36e+05
r40_g40_b40_M	0.0795	0.289	0.291	0.792	0.758	0.0824	4.61e+05
r50_g50_b50_M	0.0802	0.294	0.288	0.79	0.756	0.082	4.85e+05

Table 9.4: **RGB E/M: summary table of results**

their potential performance. Although the results were better than using only the luminance or the individual colors (see Section 9.2.1), medium fusion did not exploit the possibility of creating good feature maps at upper layers. Otherwise, the number of feature maps seemed to improve their performance, but they reached a limit when the number of feature maps per channel was larger than 30 (see Figure 9.18c).

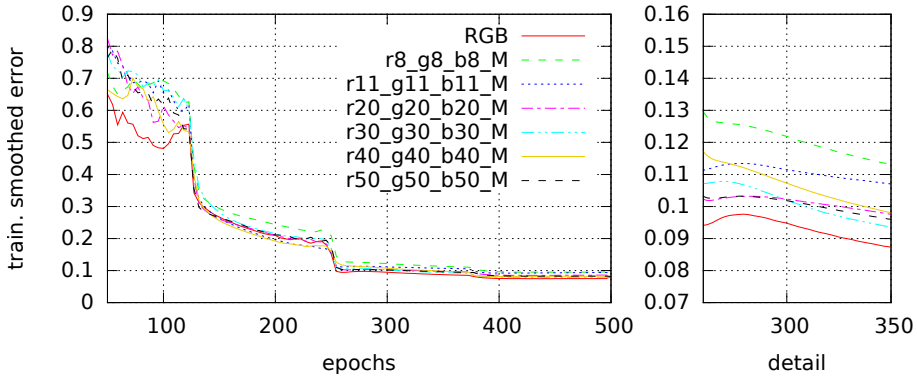


Figure 9.16: **RGB E/M: training error smoothed**

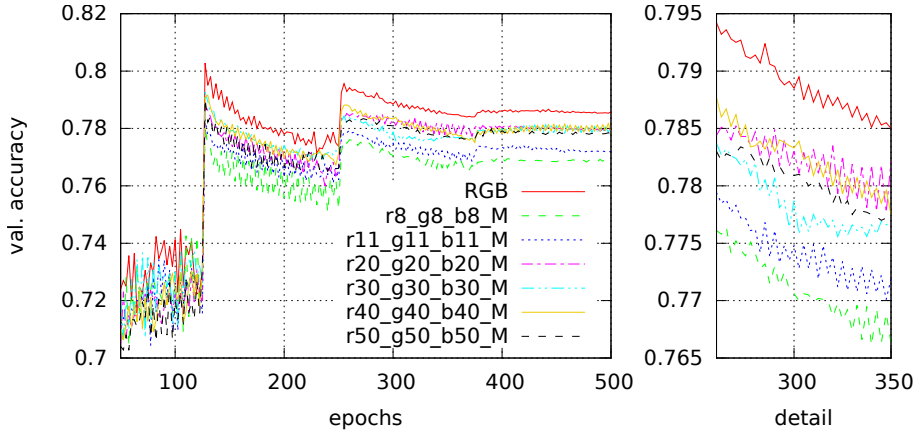


Figure 9.17: **RGB E/M: test accuracy**

9.2.4 Experiment 4: RGB + Y early and medium fusion

The previous experiments focused on different methods for merging the colors. However, in this experiment the luminance information was additionally incorporated to the *RGB* color space, and it was merged at different levels. This adds redundant information to the input, but this approach could create some specialized feature maps at different convolution levels. Table 9.5 shows a summary of the performance of each model.

	Training error			Validation acc.			
model	**min**	**avg**	**std**	**max**	**avg**	**std**	**#params**
rgb	0.0571	0.265	0.256	0.803	0.765	0.083	3.89e+05
yrgb32_E	0.0634	0.269	0.299	0.797	0.763	0.0822	3.9e+05
y16_rgb16_M	0.0821	0.304	0.284	0.79	0.757	0.0816	3.89e+05
y32_rgb32_M	0.0682	0.283	0.296	0.797	0.763	0.0824	4.16e+05

Table 9.5: **RGB+Y E/M: summary table of results**

The results show that the reference model (rgb or rgb32_E) achieved a better validation accuracy than the models that incorporated additional luminance infor-

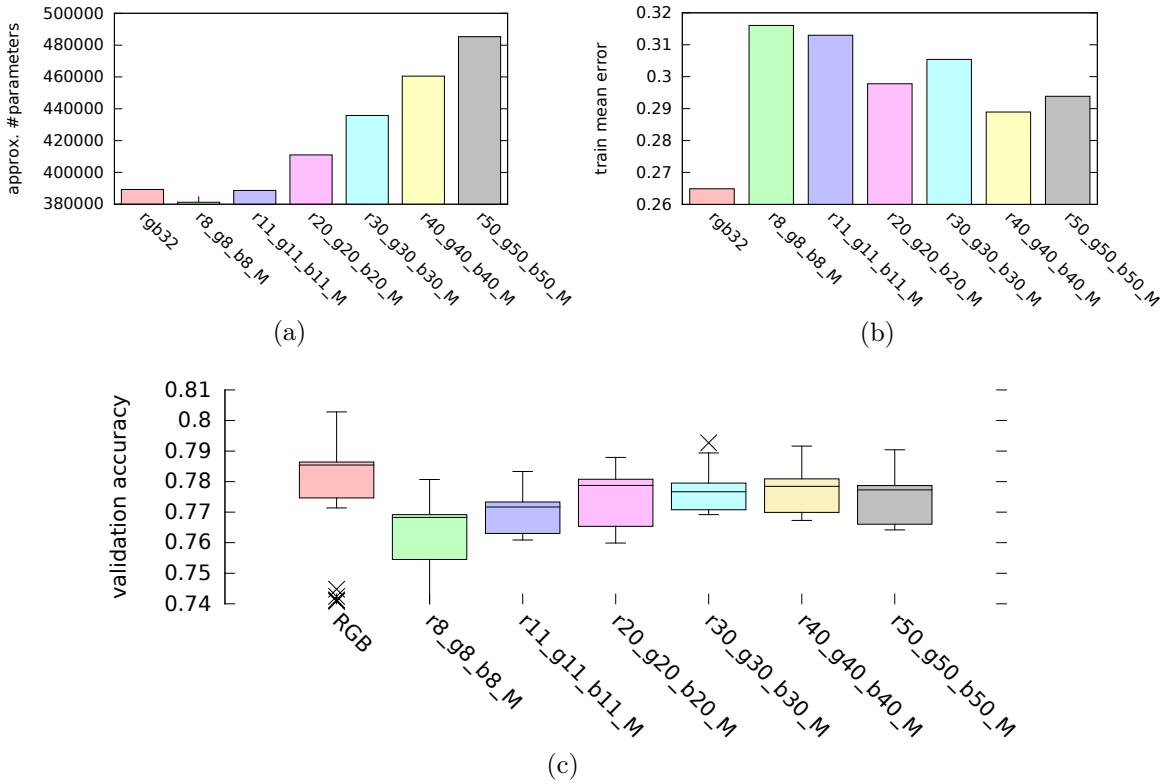


Figure 9.18: **RGB E/M: experimental results**

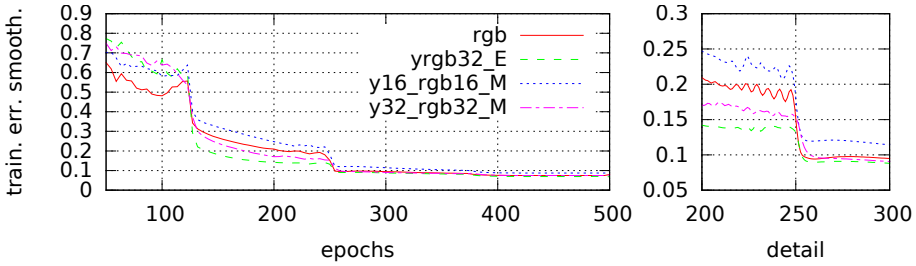


Figure 9.19: **RGB+Y E/M: training error smoothed**

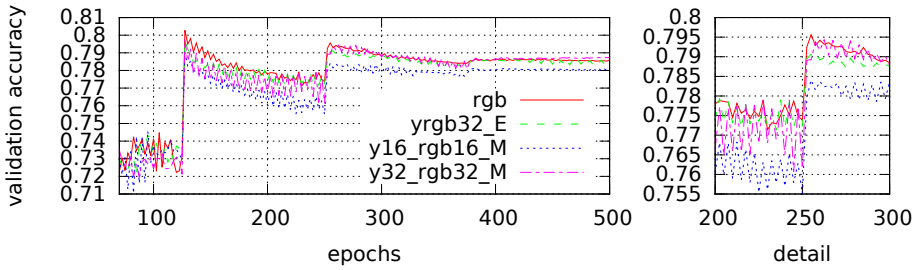


Figure 9.20: **RGB+Y E/M: test accuracy**

mation (see Table 9.5 and Figures 9.20 and 9.21c). Although the latter models

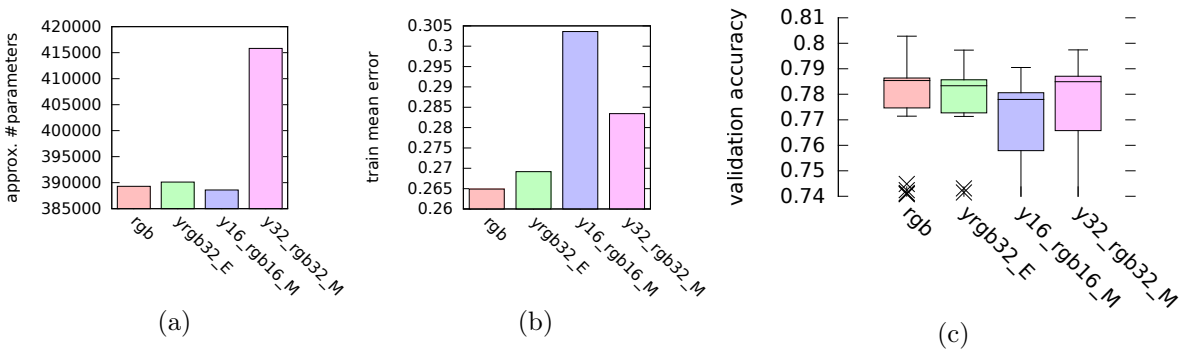


Figure 9.21: **RGB+Y E/M: experimental results**

were able to learn the same set of weights with the addition of some specific feature maps, they did not learn sufficiently good filters. One possible reason could be that the luminance channel did not add new useful information. Moreover, the redundant information increased the number of weights to train, thus increasing the learning complexity. However, their performance could be possibly improved by adjusting the hyperparameters. The tuning of the hyperparameters is open for future experiments.

9.2.5 Experiment 5: RGB + Y + UV medium fusion

In the last experiment, we tested different proportions and number of feature maps specialized on RGB, luminance, or chrominance. We merged these filters at the second convolution layer. The *RGB* model with early fusion was used as a reference model to asses the different performances. Table 9.6 and Figure 9.24 summarize the results. Figures 9.22 and 9.23 show the training and validation errors during some epochs.

	Training error			Validation acc.			
model	**min**	**avg**	**std**	**max**	**avg**	**std**	**#params**
rgb32_E	0.0571	0.265	0.256	0.803	0.765	0.083	3.89e+05
rgb11_y11_uv11_M	0.0771	0.306	0.304	0.797	0.762	0.0811	3.9e+05
rgb22_y11_uv11_M	0.0619	0.282	0.301	0.802	0.767	0.0831	3.99e+05
rgb11_y22_uv11_M	0.0823	0.292	0.295	0.801	0.765	0.0827	3.99e+05
rgb11_y11_uv22_M	0.0741	0.289	0.293	0.801	0.762	0.082	3.99e+05
rgb22_y22_uv22_M	0.0729	0.284	0.293	0.8	0.769	0.0833	4.18e+05
rgb33_y22_uv22_M	0.0633	0.274	0.288	0.801	0.768	0.0829	4.27e+05
rgb22_y33_uv22_M	0.073	0.293	0.295	0.804	0.769	0.0831	4.27e+05
rgb22_y22_uv33_M	0.0725	0.3	0.293	0.802	0.771	0.0834	4.27e+05
rgb33_y33_uv33_M	0.0696	0.287	0.286	0.799	0.769	0.0835	4.46e+05

Table 9.6: **RGB+Y+UV Medium: summary table of results**

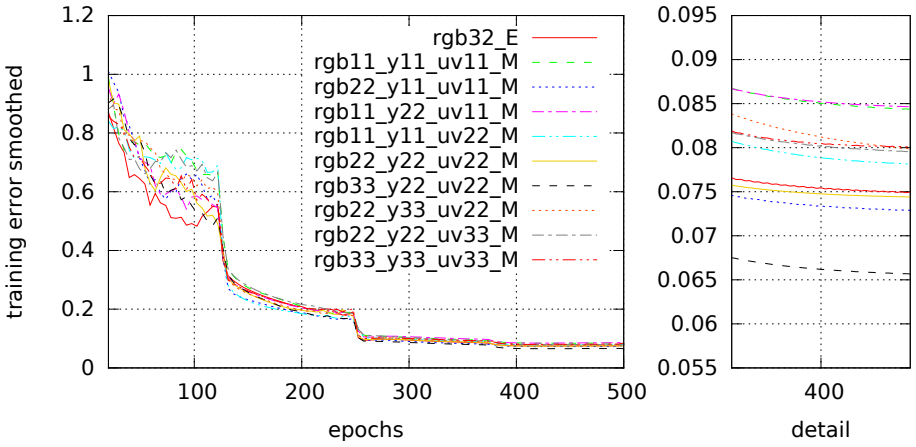


Figure 9.22: **RGB+Y+UV Medium: training error smoothed**

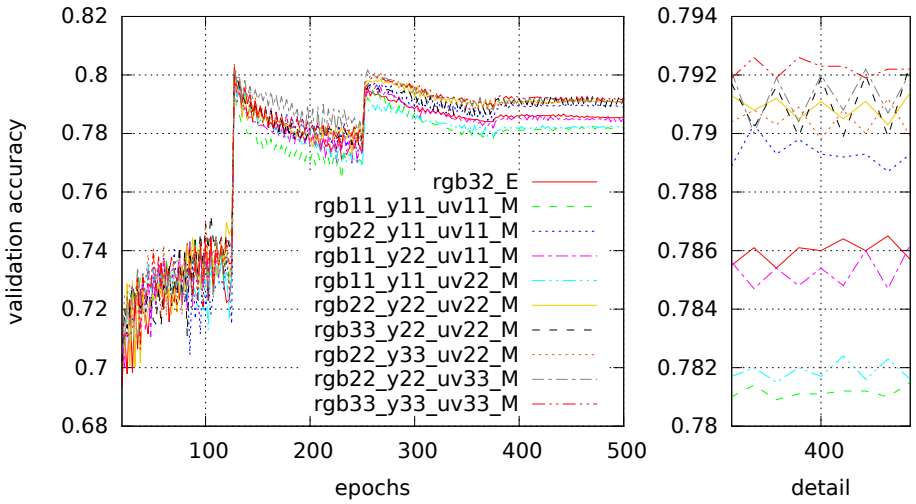


Figure 9.23: **RGB+Y+UV Medium: test accuracy**

The results show that in general the models slightly larger than the *RGB* one were not able to surpass the performance of the reference model. We saw in other experiments that the specialized features could work better than allowing the model to find the correct weights. However, from this experiment we saw that the model needed a larger number of feature maps if we restrict the number of connections. On the other hand, as we increased the number of feature maps the validation accuracy achieved comparable results. Surprisingly, the increase in the validation accuracy corresponded to an increase in the training error. This could mean that these models generalized better. The extreme case of these models was the rgb22_y22_uv33_M which achieved one of the best validation accuracies with one of the largest training errors. This model was selected for the test of statistical significance.

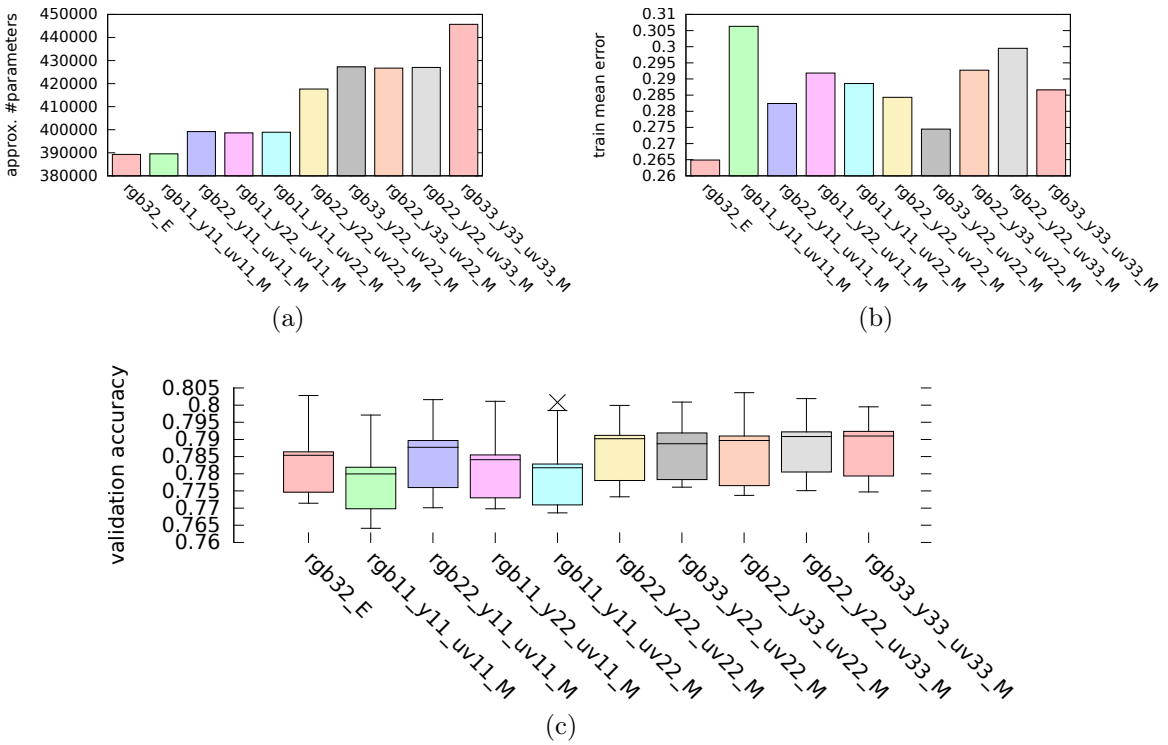


Figure 9.24: **RGB+Y+UV Medium: experimental results**

9.3 Statistical significance of the findings

As stated previously, all the experiments have been performed once per model. The reason was the time complexity of each training. In this section, we selected the models with the highest validation accuracies and inspected the statistical significance of the findings towards the initial randomization of the weights. After running the experiments ten times, we computed different statistics and show their p-values to ensure the reliability of the models. As the statistical tests require independent samples but each epoch was dependent from the previous, we computed the p-value during the training on each epoch for all the models. Table 9.7 shows the four models with the average performance of ten executions and their respective numbers of parameters.

	Training error			Validation acc.			
model	**min**	**avg**	**std**	**max**	**avg**	**std**	**#params**
yuv32_E	0.0803	0.308	0.29	0.803	0.769	0.0399	3.89e+05
y16_uv16_M	0.0761	0.317	0.296	0.807	0.77	0.0405	3.88e+05
y32_uv32_M	0.0699	0.291	0.285	0.809	0.775	0.0403	4.15e+05
rgb22_y22_uv33_M	0.0623	0.279	0.284	0.805	0.771	0.0408	4.27e+05

Table 9.7: **Statistical test: summary table of ten executions per model**

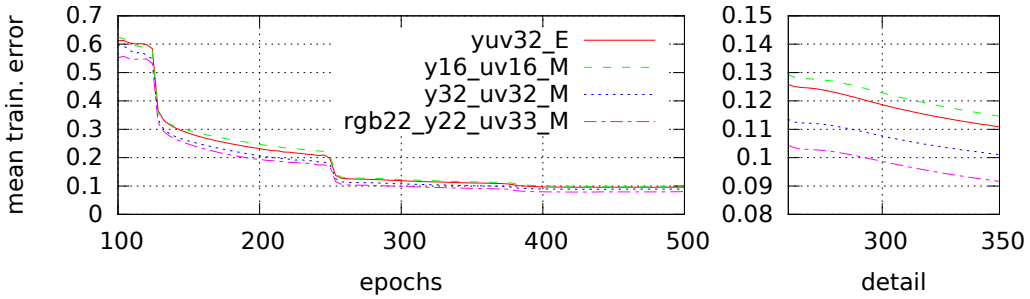


Figure 9.25: **Statistical test: mean training error** of ten executions per model

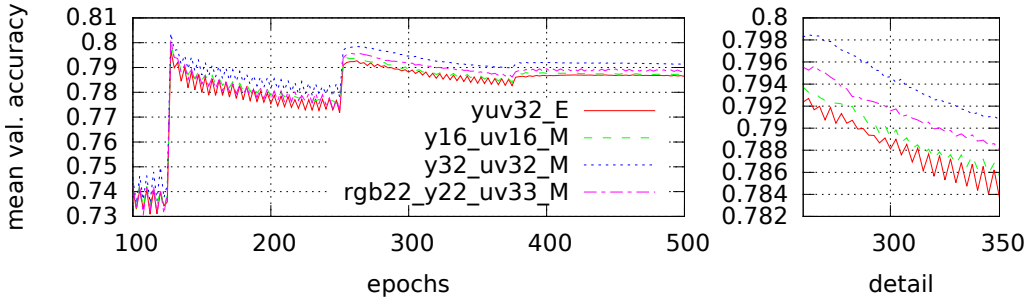
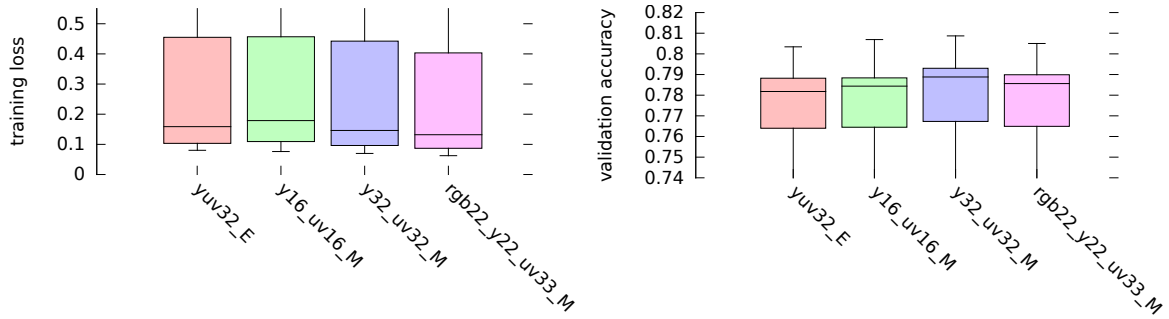


Figure 9.26: **Statistical test: test accuracy** of ten executions per model

The models performed similarly in each execution. Figure 9.25 shows the average of the training error for the ten executions, and Figure 9.26 shows the average of the validation accuracy per model. During training, the largest model (the model rgb22_y22_uv33_M that includes redundant information) got the smallest training error. However, it was possibly due to overfitting to the training data as can be seen from the poor validation accuracy. The model y32_uv32_M regularly got the best performance in the validation set. On the other hand, the basic network joining all the features in early fusion usually got a 0.006 points lower validation accuracy; corresponding to 60 misclassified images. Summaries of the training error and the validation accuracy can be seen in Figure 9.27. The boxplots show all values of each of the ten trainings, where it is possible to see the slightly better performance of the y32_uv32_M model over the yuv32_E one.

Every statistical test needed some assumptions about the sample distribution. However, the distribution of validation accuracies in an ANN strongly depends on the hyperparameters that we choose and the random initialization of the weights. Therefore, we can not be sure about the real distribution at the validation accuracy when we initialize randomly the parameters of the network. We could expect that the samples will follow a normal distribution, however the high nonlinearities involved in the predictions makes it very difficult to test. To overcome this situation we performed several statistical tests and compared their results. Each of the statistical tests had different assumptions (see a description of the tests with their assumptions in Section 8.3).

In all tests, we compared the reference model YUV32_E against the models that got the highest validation accuracies during the experiments. We tested if their



(a) Training loss

(b) Validation accuracy

Figure 9.27: **Statistical test: experimental results** of ten executions per model

.

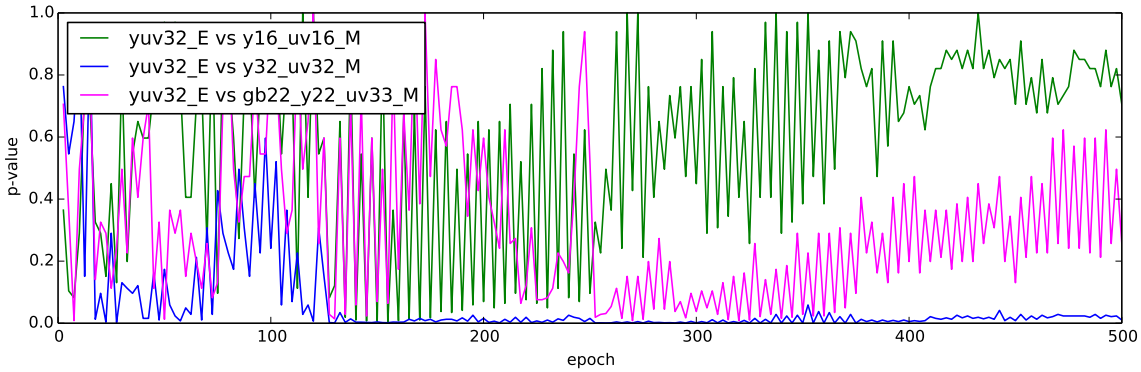


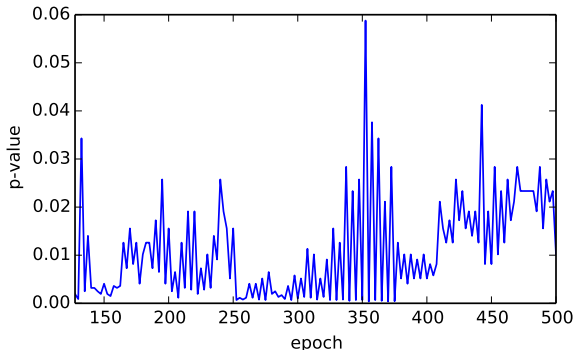
Figure 9.28: **Wilcoxon rank-sum test YUV32_E vs all**

means were significantly different. Hence, the null hypothesis H_0 and the alternative hypothesis H_a were:

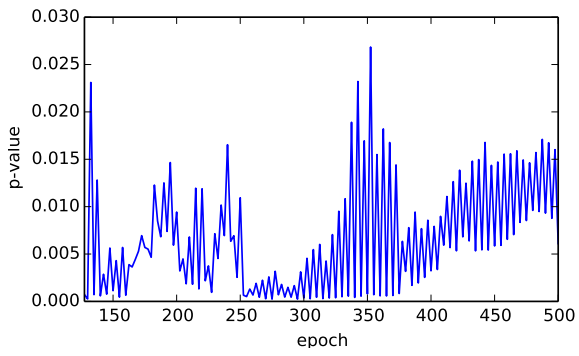
$$\begin{array}{lll} \text{Null hypothesis:} & H_0 : \mu_1 - \mu_2 = 0 & \text{The population means are equal} \\ \text{Alternative hypothesis:} & H_1 : \mu_1 - \mu_2 \neq 0 & \text{The population means are not equal} \end{array} \tag{9.1}$$

The first and most restrictive test was the Wilcoxon rank-sum test (Wilcoxon, 1945 [Wilcoxon, 1945]). The same test is also known as Mann-Whitney U test, as it was designed independently by Mann and Whitney in 1947 [Mann and Whitney, 1947]. This is a nonparametric test that does not assume normal distributions in the samples. However, it assumes homoscedasticity between both distributions (same variance). In a situation of heteroscedasticity the analysis could became invalid (for a detailed explanation see [Nachar, 2008]).

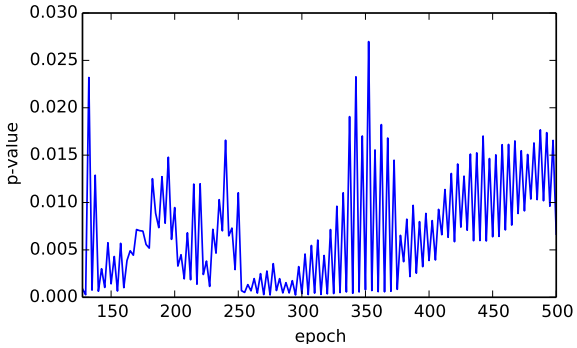
Figure 9.28 presents the p-value of the test for all the models during training. During the first 125 epochs there was no clear difference between the models. However, when the learning rate was decreased by a factor of 0.1 and all models improved their accuracy, there was a significant difference with respect to the model Y32_UV32_M. Figure 9.29a illustrates in more detail the last epochs, comparing only yuv32_E with y32_uv32_M. During the last epochs the p-value was almost



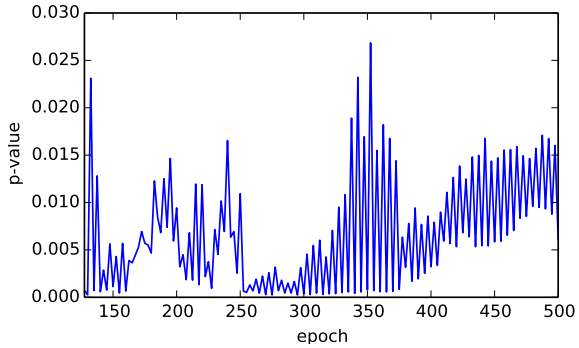
(a) Wilcoxon-Mann-Whitney two-sample rank-sum test p-values



(b) Student's t-test independent two-sample t-test



(c) Welch's t-test independent two-sample t-test



(d) One-way ANOVA test f-test

Figure 9.29: **Different statistical tests** comparing YUV32_E and Y32_uv32_M after the epoch 130

always smaller than 0.04. This indicated that the means of the distributions differ with a statistical significance of 0.04. Consequently, as the model Y32_UV32_M had always a larger validation accuracy, it should mean that it performed better.

We extended the analysis adding some assumptions to the distributions. Welch's t-test is a hypothesis test that assumes that the samples come from two normal distributions. It also requires similar number of samples from each distribution (in our case always ten samples). The results were similar to the Wilcoxon rank-sum for all the models, and for that reason only the last epochs are shown in Figure 9.29c. We got similar results when assuming homoscedasticity with the T-test (See Figure 9.29b), and performing a One-way ANOVA test that assumed the distributions to be reasonably normal and with similar standard deviation (See Figure 9.29d).

Chapter 10

Conclusions

"One thorn of experience is worth a whole wilderness of warning."

— James Russell

In the final chapter, we present a summary of our initial motivations, the context of our research, the methodology, the experiments, and the results in Section 10.1. Then we discuss about the findings and the main contributions of this thesis in Section 10.2. Finally, we conclude the chapter with an overview of all the possibilities that our contribution has opened, and give some guidances and directions of possible future work in Section 10.3.

10.1 Summary

The ideas of this thesis emerge from the extraordinary performance of the state-of-the-art models for image classification tasks. During the last years, we have been participating in the research of a new generation of machine learning models that have pushed the limits of pattern recognition. Deep learning has demonstrated the ability to learn very complex hidden representations from raw data, while hand-crafted methods that have been utilized for a long time usually require very complex solutions. On the other hand, deep convolutional neural networks have been shown to be able to learn very useful representations from large amounts of labeled data. Thanks to datasets like ImageNet these fully supervised models have been able to break the human level performance on classifying images to an extend of one thousand classes.

In this thesis, we took the state-of-the-art models and tried to understand some underlaying reasons of their good performance (see Section 8.1). We found very interesting the visual division of the learned filters into the luminance and chrominance filters (see Section 9.1.1). This fact guided our intuition to investigate whether the learned separation was actually necessary. If this was the case, it should be possible to add this prior knowledge to the network, thus making the learning easier and focused on the real problem of finding good feature descriptors. We did some

research about similar color spaces and found that the human visual system performs a similar transformation of three original signals (see Chapter 3). If our retina contains the short, medium and long photoreceptors – bluish, greenish and reddish wavelengths (see Section 3.2.1) –, in the Lateral Geniculate Nucleus (LGN) these signals are subtracted and merged in a particular way, creating one luminance signal and two chrominance signals (see Section 3.2.2). The reason is that this transformation allows a better separation of the colors and it seems to facilitate the visual process. In our work, we used a color transformation that is very similar to the biological one. It was also used during the transition from the black-and-white to the color television (see Section 2.7). In that case the luminance channel could be used for both television models, and the chrominance was added to the color versions. This compatibility was useful to facilitate the transition. Also the digital version of this transformation has been used more recently for image compression without an apparent loss of quality. The reason is that the luminance part seems more important for human vision than the chrominance, and using techniques like chroma subsampling some video and image coding systems allowed the reduction of bytes to encode images.

With this background, we found that separating the luminance from the chrominance channels could improve the learning of better hidden representations. Therefore, we first analyzed various state-of-the-art CNNs to observe if this color division was performed in other cases, or, on the contrary, if it was only an isolated case (see Section 9.1). After identifying other architectures that demonstrated the separation of the luminance and chrominance, we designed a methodology to test our ideas and to analyze an assortment of different architectures (see Section 8.2). The different experiments were inspired by the multimodal learning approach of merging different sets of features at different feature representation levels (see Section 7.4.1). Some approaches merge all the features and treat them as a mixed set. On the other hand, each feature can be preprocessed by some feature extraction and after that the hidden representation can be merged to perform the final classification. Using these ideas, our experiments used one or several color spaces and merged their individual channels at different layers. All the results were compared and the best models were selected (see Section 9.2). Finally, we run several trainings of the selected models and performed a statistical test of significance (see Section 9.3). The results showed that merging the luminance and the chrominance in the second layer and adding additional feature maps was better with a statistical significance than merging the channels at the first layer.

10.2 Discussion

We saw in Section 9.2.2 that incrementing the number of feature maps at the first layer did not improve the results when we used the YUV color space. And this color space generalizes to the RGB when the rest of the architecture remains equal. Also, we saw in Section 9.2.3 that merging the RGB channels at the second layer with varying number of feature maps did not improve the accuracy either. However, we

saw in Section 9.2.2 that when the model used the YUV color space and merged the luminance and the chrominance at the second layer the performance increase was statistically significant (with a significance level of 0.04). Although the size of the model increased by a factor of 1.07 (from 3.89e+5 to 4.15e+5 parameters), this performance was not achieved using other configurations with diverse numbers of parameters. This could indicate that it is necessary (or at least recommended) to learn separate filters for the luminance and chrominance. Additionally, we observed in Section 9.2.5 that trying to learn chrominance, luminance, and mixed filters in the same network hardly improved the results and the difference was not statistically significant.

Nevertheless, all the experiments used the same hyperparameters although the architectures were different. It would be recommended to adjust the parameters for each architecture, or at least the best models with respect to the validation accuracy. However, performing an adjustment of all the possible architectures would require loads of computational time.

The experiments were performed using the CIFAR10 dataset. We showed in Section 9.1.2 that previous networks trained with this dataset did not show a strong separation between the luminance and chrominance channels. However, the results in Section 9.3 indicated that learning the different filters separately improved the validation accuracy. We expect these results to generalize to state-of-the-art networks trained with ImageNet. As we saw in Section 9.1, these networks have a more clear separation between the filters, thus making our initial intuition more plausible. Nevertheless, larger networks would have required a prohibitive amount of training time for the scope of this thesis.

10.3 Future work

This thesis has been a small step towards the exploitation of distinctive filters for image classification. The successful results open the door to a deeper investigation in this direction. Although the tested architectures were not appropriately tuned, they outperformed the simple use of the RGB channels. However, because of the software implementation that we used, we could not incorporate several generalization techniques that have been shown to improve the results. For example, previous experiments with CIFAR10 and the original architecture have reported improved results by cropping the original 32×32 images into 28×28 crops and training with this aggregated dataset. Also, the use of mirroring of the input images in a random manner helped the generalization. Additionally, in larger networks like AlexNet using dropout at the last fully connected layers has been demonstrated to improve the generalization error.

First, (1) we recommend to add the cropping and mirroring, as it can be used in any dataset and architecture with shown generalization improvement. Next, (2) one should test a subset of the analyzed architectures to compare the performance when these features are incorporated. Then, (3) the best ideas can be used to train some of the state-of-the-art models with ImageNet dataset. Despite the models in

CIFAR10 demonstrating an increase of performance when merging the luminance and chrominance at the second layer, in larger networks a good merging could appear at upper layers.

Additionally, we chose the YUV color space because it was an easy linear transformation that completely separated the luminance from the chrominance. Nevertheless, when we analyzed the different CNNs we observed that better components can be used instead of the pure luminance. In our visualizations, there is no absolute white, as it has a bluish or greenish tone, nor absolute dark but a slightly reddish one. It is possible to find this axis performing principal component analysis, or use other methods to find better representations. The use of a linear transformation is recommended as it would be more portable to other implementations. However, non-linear transformations can be also used. The cost of using a non-linear transformation is in computing the mean of the images externally.

Appendix A

Architectures

In this thesis there are several representations of CNNs. Therefore, to facilitate the understanding of the different architectures, we designed a specific set of shapes to denote each layer of a typical CNN (see Figure A.1). Additionally, each layer in the posterior figures has a representative name.

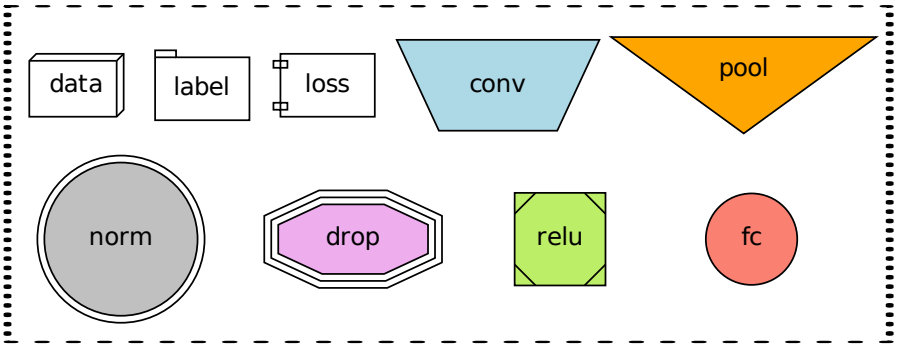


Figure A.1: **CNNs Keys** This representation of different class of layers can be used as a reference to better understand the different architectures shown in this thesis. These are from top down and left to right: data, label, loss function, convolutional layer, pooling layer, normalization layer, drop-out layer, rectifier as the activation function (or rectified linear unit (ReLU), and a fully connected layer

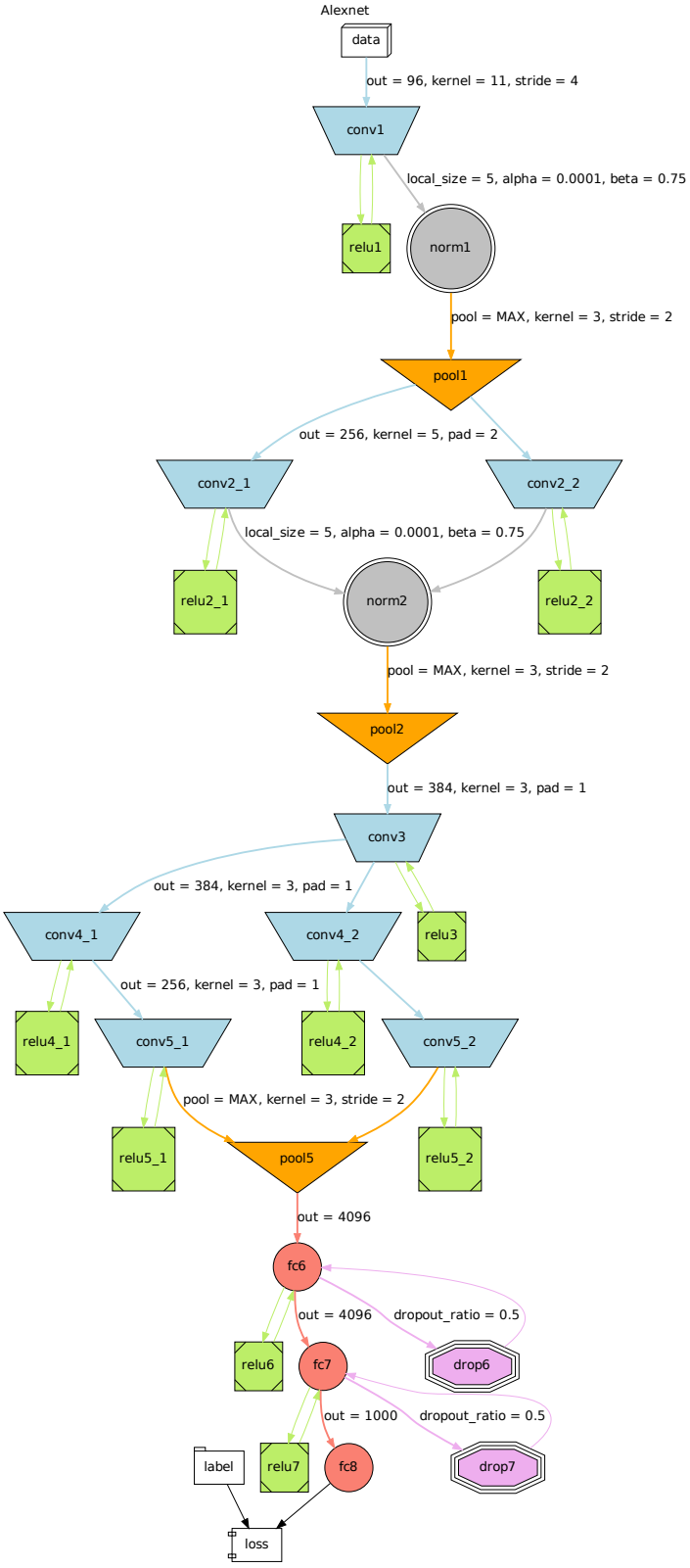


Figure A.2: **Alexnet**

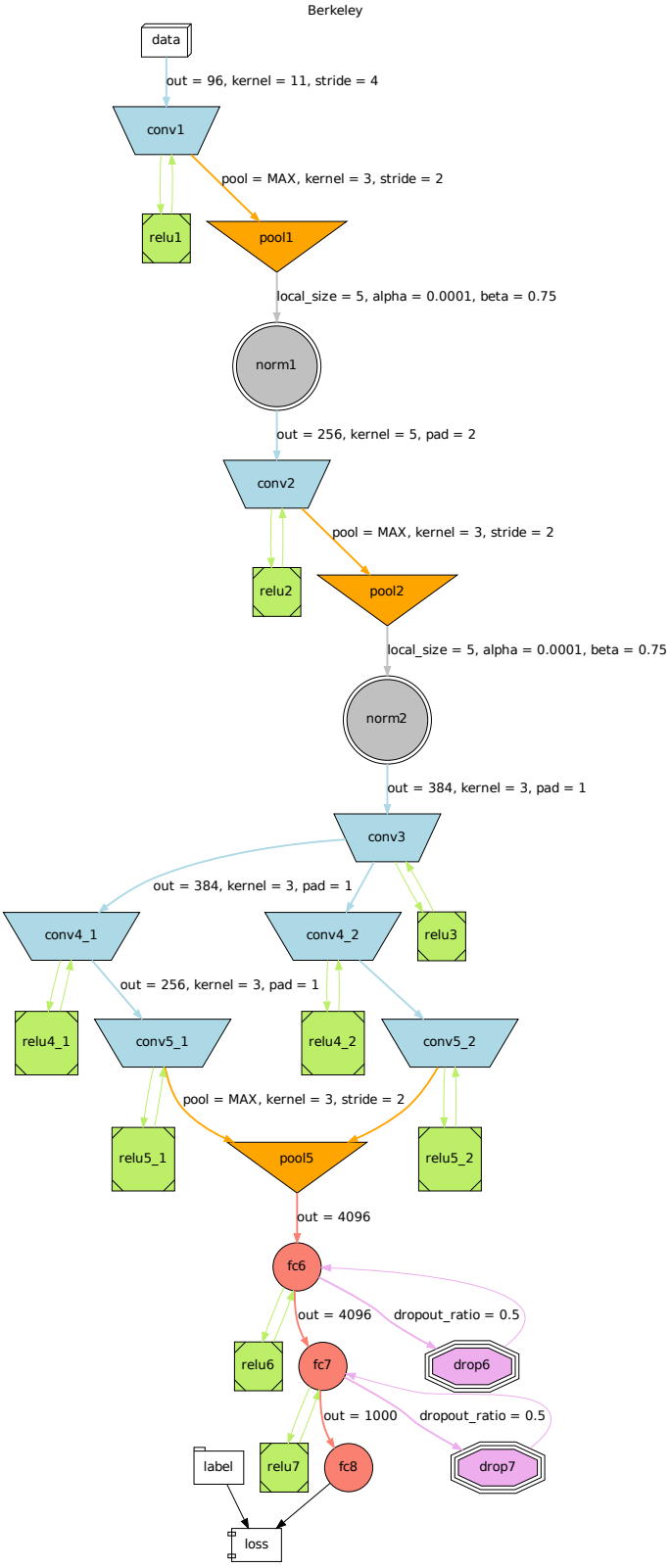


Figure A.3: **Berkeley version of Alexnet**

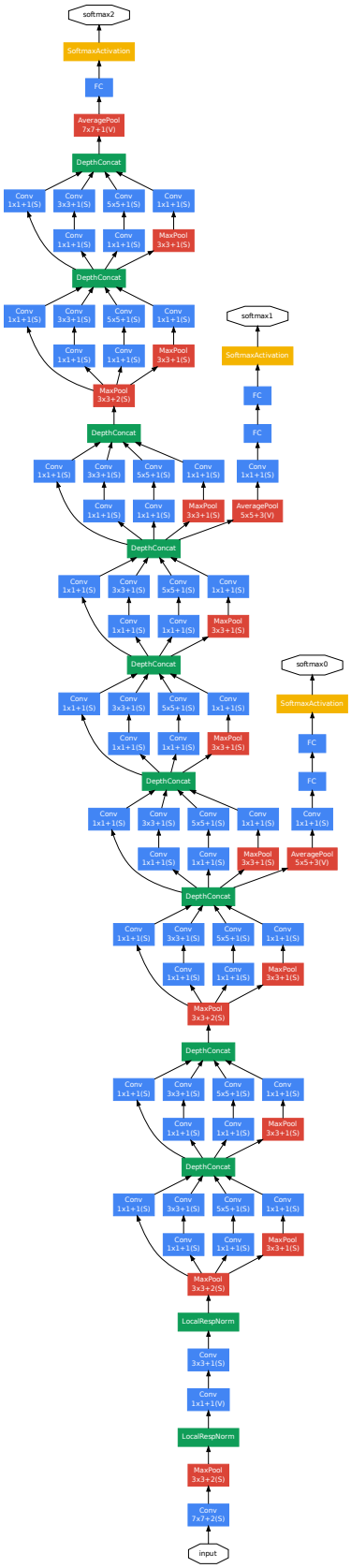


Figure A.4: **GoogleNet** Figure from [Szegedy et al., 2014]

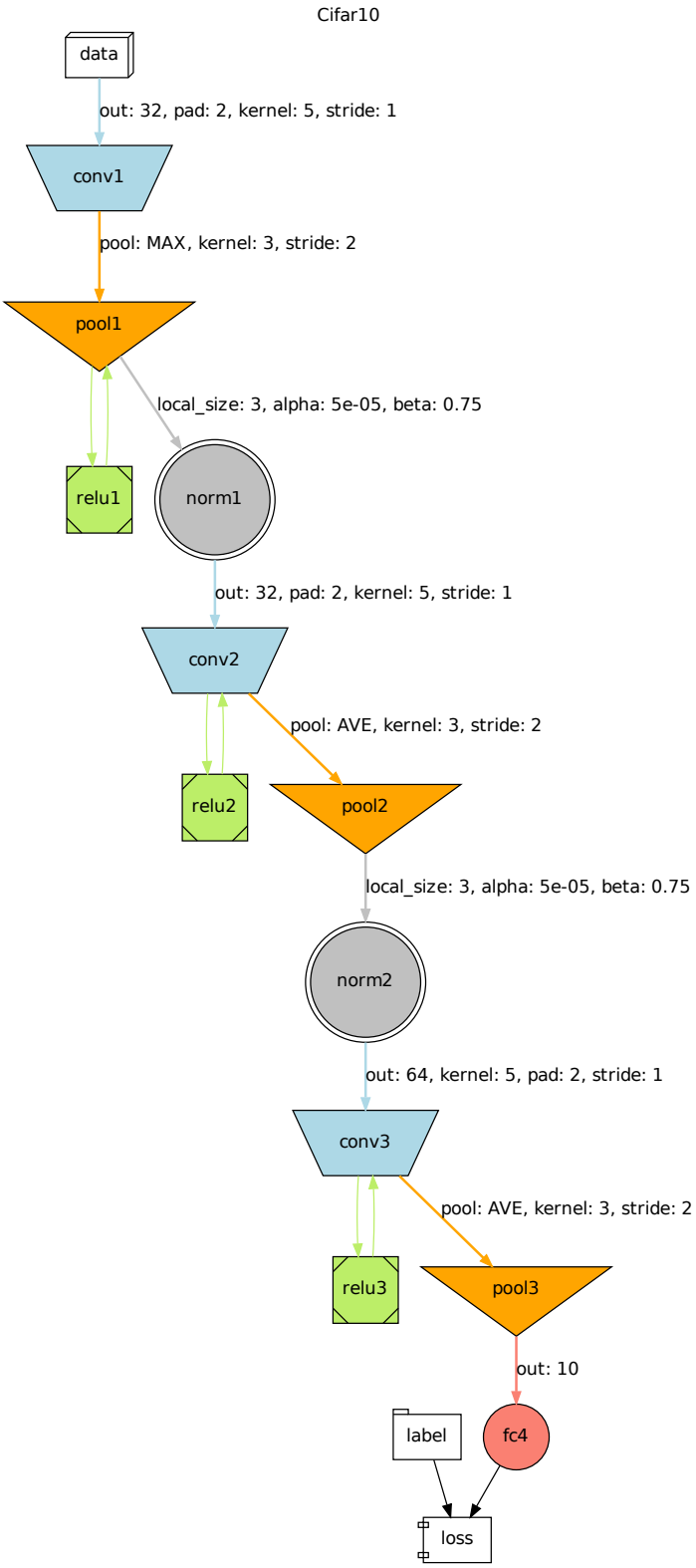


Figure A.5: **Early fusion: rgb32_E** or rgb32-32-64

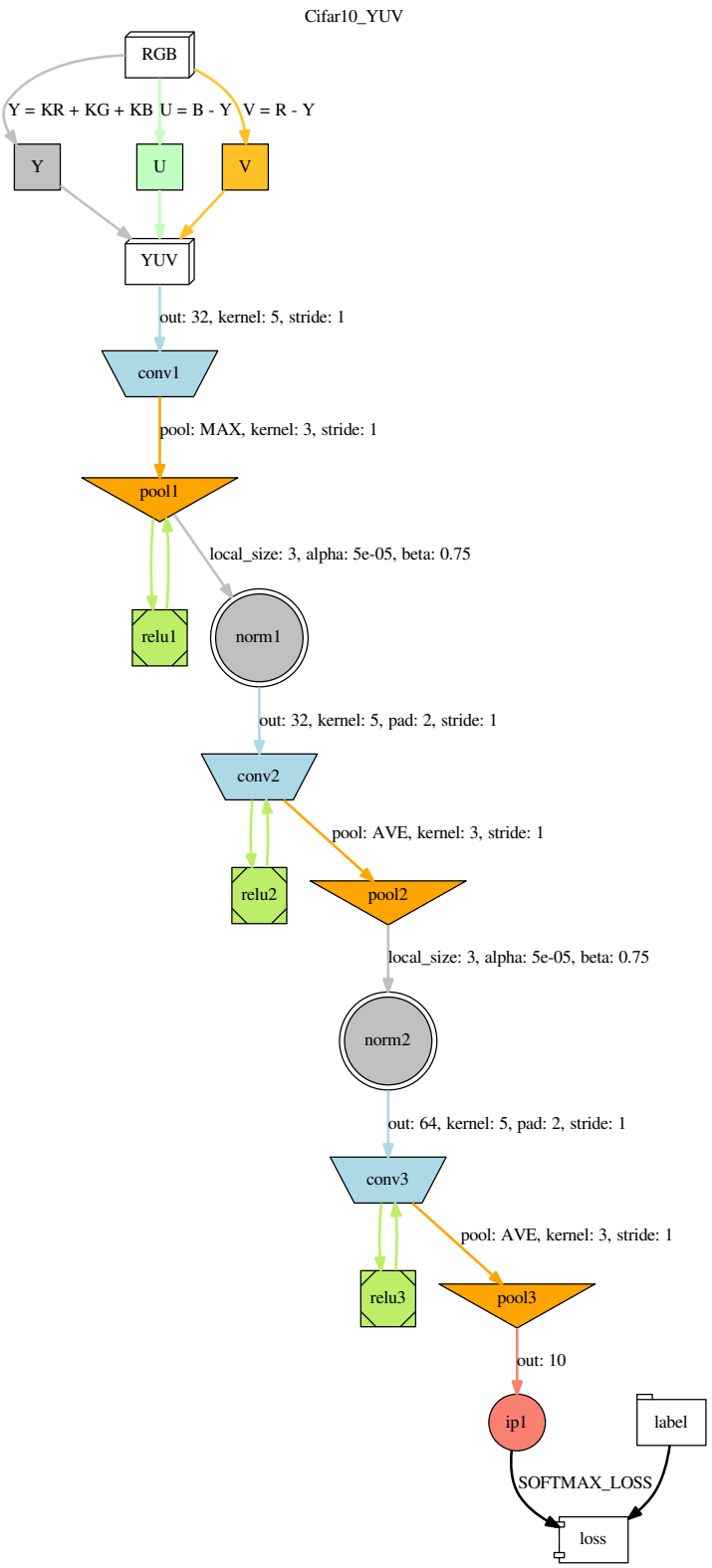


Figure A.6: **Early fusion: yuv32_E** or yuv32-32-64

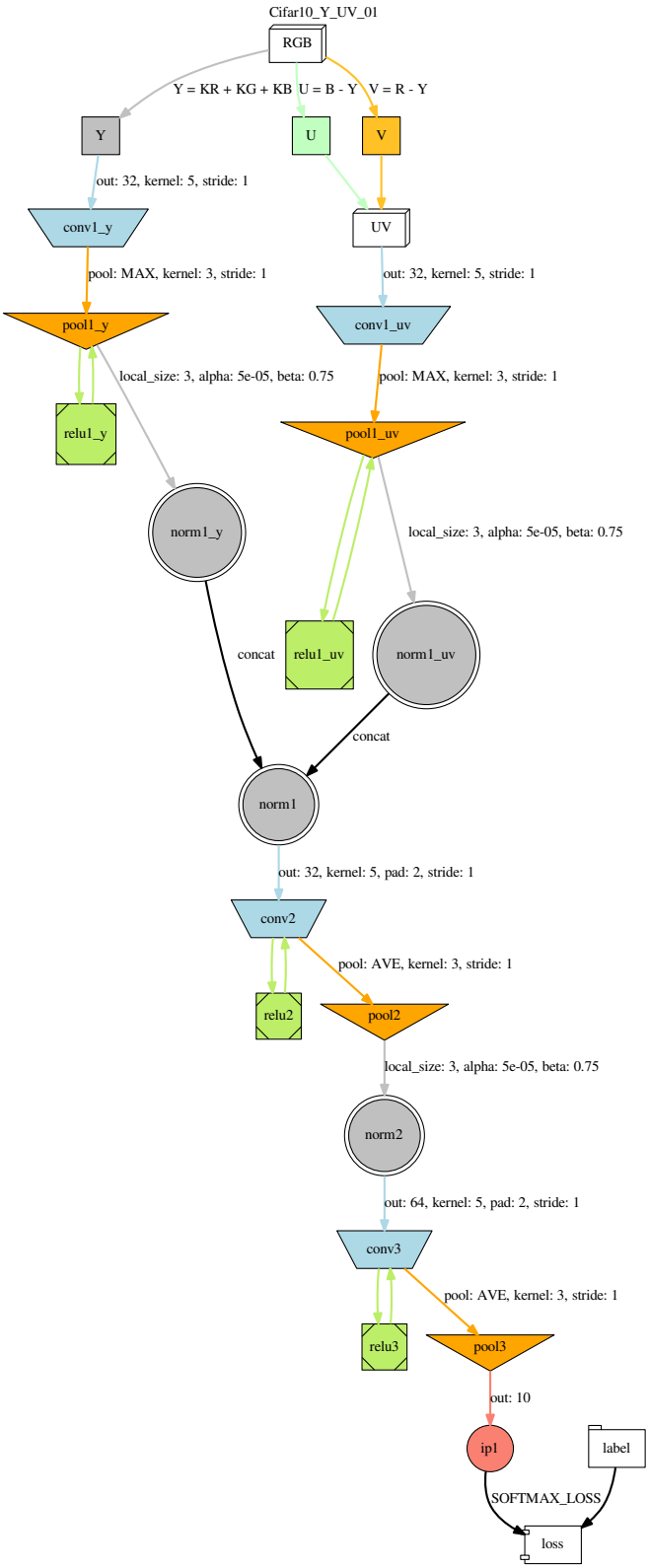


Figure A.7: **Medium fusion: y32_uv32_M** or y32_uv32_32-64

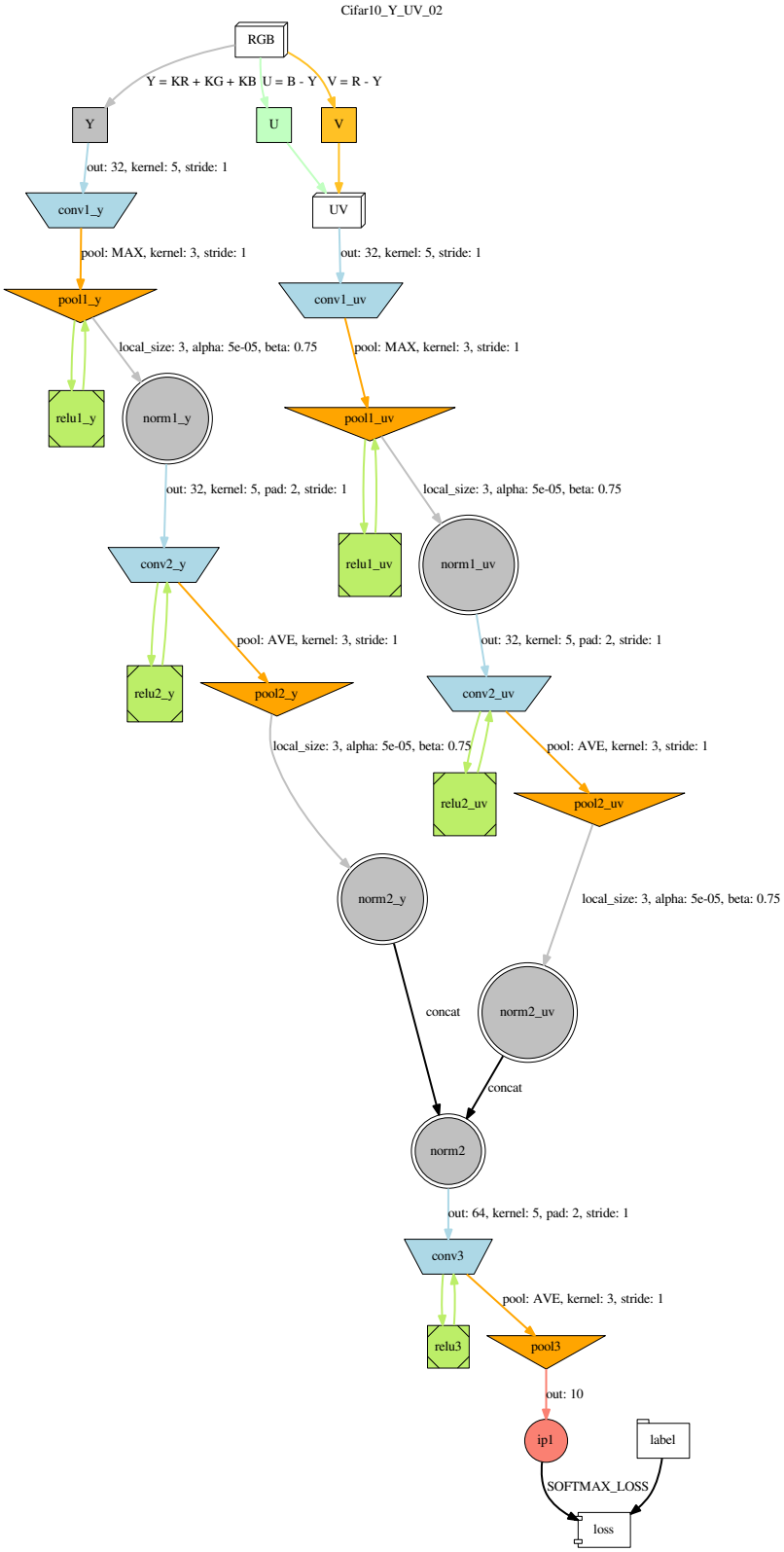


Figure A.8: **Late fusion: y32_uv32_L** or y32-32_uv32-32_64

Bibliography

Aas, K. and Eikvil, L. (1999). Text categorisation: A survey. *Raport NR*.

Abdel-Hamid, O. and Mohamed, A. (2012). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. *Acoustics, Speech and ...*.

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A Learning Algorithm for Boltzmann Machines*. *Cognitive Science*, 9(1):147–169.

Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. *Pattern Analysis and Machine ...*, 26(11):1475–1490.

Agarwal, S. and Roth, D. (2002). Learning a Sparse Representation for Object Detection. In Heyden, A., Sparr, G., Nielsen, M., and Johansen, P., editors, *Computer Vision-ECCV 2002*, Lecture Notes in Computer Science, pages 113–127. Springer Berlin Heidelberg.

Agrawal, P., Girshick, R., and Malik, J. (2014). Analyzing the performance of multilayer neural networks for object recognition. *Computer Vision-ECCV 2014*, pages 329–344.

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, pages 207–216.

Amari, S. (1967). A Theory of Adaptive Pattern Classifiers. *IEEE Transactions on Electronic Computers*, EC-16(3):299–307.

Amari, S.-I. (1972). Characteristics of Random Nets of Analog Neuron-Like Elements. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-2(5):643–657.

Anderson, J. (1972). A simple neural network generating an interactive memory. *Mathematical Biosciences*, 14(3-4):197–220.

Anderson, J. and Rosenfeld, E. (2000). *Talking nets: An oral history of neural networks*. MIT Press.

Anderson, J. A., Pellionisz, A., and Rosenfeld, E. (1990). *Neurocomputing (Vol. 2): Directions for Research*. MIT Press, Cambridge, MA, USA.

Anderson, J. A. and Rosenfeld, E. (1988). *Neurocomputing: foundations of research*. MIT Press, Cambridge, MA, USA.

Arbelaez, P. and Maire, M. (2011). Contour detection and hierarchical image segmentation. *Pattern Analysis and ...*, 33(5):898–916.

Ashby, W. (1960). Design for a Brain: The Origin of Adaptive Behavior.

Ashby, W. R. (1949). The Electronic Brain. *Radio Electronics*, pages 77–80.

Athitsos, V., Neidle, C., Sclaroff, S., Nash, J., Stefan, A., Yuan, Q., and Thangali, A. (2008). The American Sign Language Lexicon Video Dataset. In *Computer Vision and ...*, pages 1–8.

Bain, A. (1873). *Mind and body. The theories of their relation*. New York : D. Appleton and company.

Barto, A., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and ...*, SMC-13(5):834–846.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements. *arXiv:1211.5590 [cs]*.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (SURF). *Computer vision and image ...*, 110(3):346–359.

Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded up robust features. *Computer Vision-ECCV 2006*.

Bell, A. and Sejnowski, T. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159.

Bengio, Y. (2009). *Learning Deep Architectures for AI*, volume 2.

Bengio, Y., Goodfellow, I. J., and Courville, A. (2015). *Deep Learning*. MIT Press.

Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*, (1):1–41.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5(2):157–166.

Bernstein, J. (1981). A. I. *The New Yorker*, page 50.

Bezdek, K., Deza, A., and Ye, Y. (2013). *Discrete geometry and optimization*. Springer Science & Business Media.

Bishop, C. M. (2006). *Pattern recognition and machine learning.*, volume 1. New York: springer, 2006.

Block, H. (1962). The perceptron: A model for brain functioning. I. *Reviews of Modern Physics*, 34(1):123–135.

Boden, M. (2006). *Mind as machine: A history of cognitive science*, volume 1.

Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *... of the fifth annual workshop on ...*, pages 144–152.

Broomhead, D. and Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks.

Bruce, J., Balch, T., and Veloso, M. (2000). Fast and inexpensive color image segmentation for interactive robots. volume 3, pages 2061–2066 vol.3.

Bryson, A., Denham, W., and Dreyfus, S. (1963). Optimal programming problems with inequality constraints. *AIAA journal*, 1(11):2544–2550.

Bryson, A. E. (1975). *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press.

Burt, P. and Adelson, E. (1983). The Laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 31(4):532–540.

Cajal, S. R. y. (1909). Histologie du systeme nerveux de l'homme & des vertebres. page 1014.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698.

Carreira-Perpinan, M. and Hinton, G. (2005). On contrastive divergence learning. *... on artificial intelligence and ...*, 0.

Castel, L. B. (1740). *L'optique des Couleurs*. Chez Briasson (A Paris).

Chai, D. and Bouzerdoum, A. (2000). A Bayesian approach to skin color classification in YCbCr color space. volume 2, pages 421–424 vol.2.

Chai, D. and Ngan, K. (1998). Locating facial region of a head-and-shoulders color image. pages 124–129.

Chai, D. and Ngan, K. (1999). Face segmentation using skin-color map in videophone applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(4):551–564.

Churchland, P. and Sejnowski, T. (1988). Perspectives on cognitive neuroscience. *Science*, 242(4879):741–745.

Cohen, M. and Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *Systems, Man and Cybernetics, ...*, SMC-13(5):815–826.

Cooper, L. (2012). A possible organization of animal memory and learning. *Collective Properties of Physical Systems, eds. B. ....*

Copeland, B. (2012). *Alan Turing's Electronic Brain: The Struggle to Build the ACE, the World's Fastest Computer*. Oxford University Press.

Copeland, B. and Proudfoot, D. (1996). On Alan Turing's anticipation of connectionism. *Synthese*, 108(3):361–377.

Copeland, B. J. and Proudfoot, D. (1999). Alan Turing\'s forgotten ideas in Computer Science. *Scientific American*, pages 99–103.

Cragg, B. G. and Temperley, H. N. V. (1955). Memory: The Analogy with Ferromagnetic Hysteresis. *Brain*, 78(2):304–316.

Crevier, D. (1993). AI: The tumultuous history of the search for artificial intelligence.

Csurka, G. and Dance, C. (2004). Visual categorization with bags of keypoints. *Workshop on statistical ....*

Cull, P. (2007). The mathematical biophysics of Nicolas Rashevsky. *Biosystems*, 88(3):178–184.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. *... and Pattern Recognition, 2005. CVPR 2005 ...*, 1:886–893 vol. 1.

Dayan, P. (2000). Helmholtz machines and wake-sleep learning. *Handbook of Brain Theory and Neural Network. MIT ...*, 44(0).

Dayan, P. and Hinton, G. (1996). Varieties of Helmholtz machine. *Neural Networks*, 9(8):1385–1403.

Dayan, P., Hinton, G., Neal, R., and Zemel, R. (1995). The helmholtz machine. *Neural computation*, 904:889–904.

Deng, J., Dong, W., Socher, R., and Li, L. (2009). Imagenet: A large-scale hierarchical image database. *Computer Vision and ...*, pages 248–255.

Dong, G. and Li, J. (1999). Efficient mining of emerging patterns: Discovering trends and differences. *Proceedings of the fifth ACM SIGKDD international ...*, pages 43–52.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2-3):195–225.

Everingham, M. and Eslami, S. (2014). The pascal visual object classes challenge: A retrospective. *International Journal of ...*, 111(1):98–136.

Everingham, M. and Gool, L. V. (2010). The pascal visual object classes (voc) challenge. *International journal of ...*, 88(2):303–338.

Fang, H., Gupta, S., and Iandola, F. (2014). From Captions to Visual Concepts and Back. *arXiv preprint arXiv: ....*

Farley, B. and Clark, W. (1954). Simulation of self-organizing systems by digital computer. *... of the IRE Professional Group on*, 4(4):76–84.

Fei-Fei, L., Fergus, R., and Perona, P. (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70.

Feldman, J. and Ballard, D. (1982). Connectionist models and their properties. *Cognitive science*, 6(3):205–254.

Fellbaum, C. (1998). WordNet: an electronic lexical database. Cambridge, MA: MIT Press.

Fischler, M. A. (1987). *Intelligence: The Eye, the Brain, and the Computer.* Addison-Wesley.

Fu, L. (2003). Neural networks in computer intelligence. page 484.

Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. 202.

Fukushima, K., Miyake, S., and Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. *Systems, Man and Cybernetics, ...*, SMC-13(5):826–834.

Funahashi, K.-i. and Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806.

Gabbay, D., Woods, J., and Thagard, P. (2006). *Philosophy of Psychology and Cognitive Science: A Volume of the Handbook of the Philosophy of Science Series.* Elsevier.

Gabor, D. (1954). Communication theory and cybernetics. *Circuit Theory, Transactions of the IRE Professional ...*, CT-1(4):19–31.

Garson, G. D. (1998). *Neural Networks: An Introductory Guide for Social Scientists.* SAGE.

Glassner, A. (1995). *Principles of digital image synthesis: Vol. 1.* Elsevier.

Glimm, J., Impagliazzo, J., and Singer, I. (2006). *The legacy of John von Neumann.* American Mathematical Soc.

Goldstein, E. (2013). *Sensation and Perception.* Cengage Learning.

Goodfellow, I. J., Warde-Farley, D., Lamblin, P., Dumoulin, V., Mirza, M., Pascanu, R., Bergstra, J., Bastien, F., and Bengio, Y. (2013). Pylearn2: a machine learning research library. *arXiv:1308.4214 [cs, stat]*.

Gool, L. V., Moons, T., and Ungureanu, D. (1996). Affine / photometric invariants for planar intensity patterns. Lecture Notes in Computer Science, pages 642–651. Springer Berlin Heidelberg.

Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 Object Category Dataset.

Gross, C. (2002). Genealogy of the "grandmother cell". *The Neuroscientist*, 8(5):512–518.

Grossberg, S. (1967). Nonlinear difference-differential equations in prediction and learning theory. *Proceedings of the National Academy of Sciences of the United States of America*, 58(4):1329–1334.

Grossberg, S. (1980). How does a brain build a cognitive code? *Studies of Mind and Brain*, 87(1).

Grossberg, S. (1988). Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, 1(1):17–61.

Gupta, M. M. and Knopf, G. K. (1994). Neuro-vision systems: A tutorial. In *Neuro-Vision Systems: Principles and Applications*, pages 1–34.

Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, pages 23.1–23.6.

Hartley, D. (1749). *Observations on man, his frame, his duty, and his expectations*. 00 L : Scholars' Facsimiles and Reprints.

Haykin, S. (1994). *Neural networks: a comprehensive foundation*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852 [cs]*.

Hebb, D. O. (1949). *The Orgamization of Behavior a Neuropsychological Theory*. John Wiley & Sons Inc., New York.

Hecht-Nielsen, R. (1989). *Neurocomputing*. Addison-Wesley Publishing Company.

Hinton, G., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 1554:1527–1554.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995). The "wake-sleep" algorithm for unsupervised neural networks. *Science (New York, N.Y.)*, 268(5214):1158–61.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. *Master's thesis, Institut fur Informatik, Technische ....*

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10):3088–3092.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Howard C. Warren (1921). A history of the association psychology. page 355.

Huang, G., Zhu, Q., and Siew, C. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. *Neural Networks, 2004. ...*, 2:985–990 vol.2.

Huang, G.-B., Huang, G., Song, S., and You, K. (2015). Trends in extreme learning machines: A review. *Neural Networks*, 61:32–48.

Hubel, D. and Wiesel, T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, pages 106–154.

Hubel, D. and Wiesel, T. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, pages 215–243.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

Jaeger, H. and Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80.

Jarrett, K., Kavukcuoglu, K., Ranzato, M. A., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153.

Jia, Y., Shelhamer, E., and Donahue, J. (2014). Caffe: Convolutional architecture for fast feature embedding. *Proceedings of the ....*

Kadir, T. and Brady, M. (2001). Saliency, Scale and Image Description. *International Journal of Computer Vision*, 45(2):83–105.

Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: a more distinctive representation for local image descriptors. volume 2, pages II–506–II–513 Vol.2.

Kirby, K. (2006). A tutorial on Helmholtz Machines. (June).

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

Kohonen, T. (1972). Correlation matrix memories. *Computers, IEEE Transactions on*, 21(4):353–359.

Kohonen, T. (1977). *Associative Memory: A System-Theoretical Approach*. Springer, Berlin, Heidelberg, softcover edition.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. *... Science Department, University of Toronto, Tech. ...*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, pages 1–9.

Lazebnik, S., Schmid, C., and Ponce, J. (2005). A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278.

Le, Q., Ranzato, M., Monga, R., and Devin, M. (2012). Building high-level features using large scale unsupervised learning. *arXiv preprint arXiv: ...*.

Le Cun, Y. (1986). Learning Process in an Asymmetric Threshold Network. NATO ASI Series, pages 233–240. Springer Berlin Heidelberg.

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau a seuil asymmetrique (a Learning Scheme for Asymmetric Threshold Networks). In *Proceedings of Cognitiva*, pages 599–604, Paris, France.

LeCun, Y. (1989). Generalization and network design strategies. *Connections in Perspective. North-Holland, ...*.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Backpropagation applied to handwritten zip code recognition. *Neural ...*.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1990). Handwritten digit recognition with a back-propagation network. *Advances in neural ...*, pages 396–404.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116.

Linsker, R. (1988). Self-organisation in a perceptual network. *Computer*, 21(3):105–117.

Little, W. A. and Shaw, G. L. (1975). A statistical theory of short and long term memory. *Behavioral Biology*, 14(2):115–133.

Liu, W., Principe, J. C., and Haykin, S. (2011). *Kernel Adaptive Filtering: A Comprehensive Introduction*. John Wiley & Sons.

Locke, J. (1700). *An essay concerning human understanding*. Read Books.

Lowe, D. (1999). Object recognition from local scale-invariant features. volume 2, pages 1150–1157 vol.2.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Lyvers, E. and Mitchell, O. (1988). Precision edge contrast and orientation estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):927–937.

Mann, H. and Whitney, D. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 18(1):50–60.

Martens, J. (2010). Deep learning via Hessian-free optimization. *Proceedings of the 27th International Conference on Machine Learning*.

Martens, J. and Sutskever, I. (2011). Learning recurrent neural networks with hessian-free optimization. *... on Machine Learning (ICML-11 ...*.

Martin, D. and Fowlkes, C. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Computer Vision, 2001. ...*, 2:416–423 vol.2.

Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Mehrotra, K., Mohan, C., and Ranka, S. (1997). *Elements of artificial neural networks.*

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Mikolajczyk, K., Leibe, B., and Schiele, B. (2005a). Local features for object class recognition. *Computer Vision, 2005. ...*, 2:1792–1799 Vol. 2.

Mikolajczyk, K. and Schmid, C. (2004). Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *Pattern Analysis and Machine ...*, 27(10):1615–1630.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Gool, L. V. (2005b). A comparison of affine region detectors. *International journal of ...*, 65(1-2):43–72.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41.

Minsky, M. (1954). *Theory of neural-analog reinforcement systems and its application to the brain model problem.* PhD thesis.

Minsky, M. (1967). *Computation: finite and infinite machines.* Prentice Hall, Englewood Cliffs, NJ.

Minsky, M. and Papert, S. (1969). *Perceptrons.*

Mühlenbein, H. (2009). Computational Intelligence: The Legacy of Alan Turing and John von Neumann. *Computational Intelligence*, pages 23–43.

Nachar, N. (2008). The Mann-Whitney U: a test for assessing whether two independent samples come from the same distribution. *Tutorials in Quantitative Methods for Psychology*, 4(1):13–20.

Nakano, K. (1972). Associatron-a model of associative memory. *Systems, Man and Cybernetics, IEEE Transactions ...*, SMC-2(3):380–388.

Nathans, J., Piantanida, T., and Eddy, R. (1986). Molecular genetics of inherited variation in human color vision. *Science*, 232(4747):203–210.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113.

Neelakanta, P. S. and DeGroff, D. (1994). *Neural Network Modeling: Statistical Mechanics and Cybernetic Perspectives.* CRC Press.

Nene, S., Nayar, S., and Murase, H. (1996). Columbia object image library (COIL-20).

Neumann, J. and Burks, A. (1966). Theory of self-reproducing automata.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. (2011). Multimodal deep learning. *International Conference on Machine Learning*, 28.

Nilsson, N. J. (1965). *Learning machines: foundations of trainable pattern-classifying systems.* McGraw-Hill.

Norberg, A. L. (2005). *Computers and Commerce: A Study of Technology and Management at Eckert-Mauchly Computer Company, Engineering Research Associates, and Remington Rand, 1946 – 1957.* MIT Press.

Novikoff, A. (1962). On convergence proofs on perceptrons. *Proceedings of the Symposium on the Mathematical Theory of Automata, New York*, XII:615–622.

Parker, D. (1985). Learning-logic. Technical report, Invention Report S81-64, File 1, Cambridge, MA: Center for Computational Research in Economics and Management Science, MIT.

Pearl, J. (1985). Bayesian networks: A model of self-activated memory for evidential reasoning. Technical report.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann.

Phillips, P. and Moon, H. (2000). The FERET evaluation methodology for face-recognition algorithms. *Pattern Analysis and ...*, 22(10):1090–1104.

Phung, S., Bouzerdoum, A., and Chai, S. D. (2005). Skin segmentation using color pixel classification: analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):148–154.

Pinto, N., Cox, D. D., and DiCarlo, J. J. (2008). Why is real-world visual object recognition hard? *PLoS computational biology*, 4(1):e27.

Prewitt, J. (1970). Object enhancement and extraction. *Picture processing and Psychopictorics*, 10:15–19.

Rashevsky, N. (1938). Mathematical biophysics. Physicomathematical foundations of biology.

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, pages 1019–1025.

Rochester, N. and Holland, J. (1956). Tests on a cell assembly theory of the action of the brain, using a large digital computer. *Information Theory, IRE ...*, 2(3):80–93.

Rosenblatt, F. (1957). The Perceptron, a Perceiving and Recognizing Automaton. Technical report, Cornell Aeronautical Laboratory, Buffalo, NY.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408.

Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Laboratory, INC., Buffalo 21, N.Y.

Rumelhart, D., Hinton, G., and Williams, R. (1985). Learning internal representations by error propagation.

Rumelhart, D., McClelland, J., and Group, P. R. (1988). *Parallel distributed processing*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Russakovsky, O., Deng, J., and Su, H. (2014). Imagenet large scale visual recognition challenge. *arXiv preprint arXiv: ...*.

Saul, L. K., Jaakkola, T., and Jordan, M. I. (1996). Mean Field Theory for Sigmoid Belief Networks. *arXiv:cs/9603102*.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Sejnowski, T. and Rosenberg, C. (1987). Parallel networks that learn to pronounce English text. *Complex systems*, 1:145–168.

Selfridge, O. G. (1958). Pandemonium: A paradigm for learning. *Proceedings of the symposium*, 1:513–529.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv preprint arXiv: ...*, pages 1–16.

Simard, P., Steinkraus, D., and Platt, J. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. *ICDAR*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Snoek, C., Worring, M., and Smeulders, A. (2005). Early versus late fusion in semantic video analysis. *Proceedings of the 13th ...*, pages 399–402.

Srivastava, N. and Hinton, G. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine ...*, 15:1929–1958.

Steinbuch, K. and Piske, U. (1963). Learning Matrices and Their Applications. *IEEE Transactions on Electronic Computers*, EC-12(6):846–862.

Sutherland, G. B. B. M. (1959). Mechanisation of Thought Processes. In *Proceedings of a Symposium*, page 531, National Physical Laboratory, Great Britain. Her Majesty's Stationery Office.

Sutskever, I., Martens, J., and Hinton, G. (2011). Generating text with recurrent neural networks. *Proceedings of the ....*

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Szegedy, C., Liu, W., Jia, Y., and Sermanet, P. (2014). Going deeper with convolutions. *arXiv preprint arXiv: ....*

Szeliski, R. (2010). *Computer vision: algorithms and applications*.

Taylor, W. K. (1956). Electrical simulation of some nervous system functional activities. *Information theory 3*, pages 314–328.

Uhr, L. (1987). Highly parallel, hierarchical, recognition cone perceptual structures. *Parallel computer vision*.

Uttley, A. (1979). Information transmission in the nervous system. page 125.

Uttley, A. M. (1956a). Conditional probability machines and conditional reflexes. *Automata studies*, pages 253–276.

Uttley, A. M. (1956b). Temporal and spatial patterns in a conditional probability machine. *Automata Studies*, pages 277– 285.

Vemuri, V. R. (1992). Artificial Neural Networks: Concepts and Control Applications. *ieeexplore.ieee.org*, page 509.

von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14(2):85–100.

von der Malsburg, C. and Willshaw, D. J. (1976). A mechanism for producing continuous neural mappings: ocularity dominance stripes and ordered retino-tectal projections. *Exp. Brain Res*, 1:463–469.

von Neumann, J. (1945). First Draft of a Report on the EDVAC. Technical Report 1.

von Neumann, J. (1956). Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies*, pages 43–98.

Voravuthikunchai, W. (2014). Histograms of pattern sets for image classification and object recognition. *IEEE Conference on ...*, pages 224–231.

Werbos, P. (1974). Beyond regression: new tools for prediction and analysis in the behavioral sciences.

Werbos, P. (1994). *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. John Wiley & Sons.

Widrow, B. (1960). An Adaptive "ADALINE" Neuron Using Chemical "Memistors". Technical report, Solid-State Electronics Laboratory, Stanford Electronics Laboratories, Stanford University, Stanford, California.

Wiener, N. (1948). *Cybernetics or Control and Communication in the Animal and the Machine*. The Massachusetts Institute of Technology.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83.

Wilkes, A. L. and Wade, N. J. (1997). Bain on neural networks. *Brain and Cognition*, 33(3):295–305.

William James (1890). *The principles of psychology*. New York : Henry Holt and company.

Willshaw, D. J., Buneman, O. P., and Longuet-Higgins, H. C. (1969). Non-holographic associative memory. *Nature*, 222(5197):960–962.

Winograd, S. and Cowan, J. (1963). *Reliable computation in the presence of noise*. M.I.T. Press research monographs.22. M.I.T. Press, Cambridge, Mass.

Young, T. (1802). The Bakerian lecture: On the theory of light and colours. *Philosophical transactions of the Royal Society of ...*, 92:12–48.

Zhang, J. and Marszalek, M. (2007). Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of ...*, 73(2):213–238.

Ziou, D. and Tabbone, S. (1998). Edge detection techniques-an overview. *Pattern Recognition and Image Analysis C/C of ...*, pages 1–41.

Glossary

Adaline The Adaline (Adaptive Linear Neuron or Adaptive Linear Element) was one of the first implementations of an Artificial Neural Network (ANN) inspired by the work of McCulloch and Pitts. The same name was given to the physical device that simulated the network. It was designed to perform a summation of various inputs and bias, all of them scaled by a set of weights. In the physical device these weights were implemented using memistors. xiii, 77, 157, 162

AI Artificial Intelligence (AI) is a field of study that tries to create machines that are able to automatically solve particular problems. In most of the cases, finding the optimal solution is intractable and these algorithms try to find local optimums or good approximations. xiii, 4, 65, 157

ANN An Artificial Neural Network (ANN) is a mathematical representation of a biological neural network that simplifies its architecture and physical behaviour. It is used in Machine Learning to solve regression and decision problems. xiii, 3, 9, 21, 25, 35–37, 40, 43–46, 52, 54, 55, 59, 63, 65, 71, 74, 76, 78–83, 85–87, 94, 96, 104, 127, 157–161, 163–165

aperture problem In visual feature matching tasks, if the patch that is being analyzed contains only a straight line, it could be matched to multiple regions of a larger line. This problem makes impossible to choose wich is the real position. 10

Associationism It is the idea that mental processes are triggered by previous states. The connections between these states are called associations. Some of the first ideas have been seen in the work of Plato and Aristotle. In the 17th century the British "Associationist School" was funded, where John Locke, David Hume, David Hartley, James Mill, or Ivan Pavlov participated and used these ideas. 65–67

backpropagation It is a method of propagating the error backwards in an Artificial Neural Network (ANN) to perform credit assignment in the lower levels of the network. This is the common method to train a ANN toghether with an optimization method using gradient descent. xvii, 44, 46, 47, 49, 53, 55–57, 78, 82, 85, 86, 158

BM A Boltzmann Machine (BM) is a generative stochastic model with undirected connections. It is the stochastic version of the Hopfield network. xiii, 73, 82, 85, 87, 158

BN A Belief Network (BN) (also known as Bayesian network or Bayes network) is a probabilistic graphical model with acyclic and directed dependencies between a set of random variables. xiii, 57, 85, 88, 158, 159

BoV Bag of Visual words (BoV) is a technique used in computer to represent and image in a simplified manner. Based on the BoW, it describes the image with the number of occurences of certain visual features, without storing their position. xiii, 15, 59, 158

BoW Bag of Words (BoW) is a technique used in natural language processing and information retrieval to represent a text in a simplified manner. It consists on representing the text as a set of words with the number of occurences, without order. xiii, 15, 158

BPTT backpropagation through time (BPTT) is a technique to train Recurrent Neural Network (RNN)s by unfolding the network in time and applying the normal backpropagation. xiii, 56, 158

chrominance It is the color part of an image. In our vision sistem there are three type of Cone cells that are specialized in the chrominance. Glossary: Cone cells. 124

CIE Is the International Commission on Illumination (from French Commission Internationale de l'Clairage) and the international authority on light, illumination colour, and colour spaces. xiii, 16

CNN A Convolutional Neural Network (CNN) is a particular case of an ANN with some strong priors about the input signals. They exploits the assumption that there exist an important spatial or temporal connection between neighbor inputs. With this assumption it is possible to restrict the number of connections simulating a large amount of zero weights that are not actually represented in the network. For a larger explanation and a better understanding refer to the Chapter. 5. xiii, xviii, 2–4, 9, 37, 54, 56, 57, 59, 61–63, 86–89, 91–93, 95, 97, 99, 100, 103, 107, 112, 115, 132, 134, 135, 162, 163

Cone cell In our vision sistem there are three type of Cone cells, each one specialized in one of the light frequency domains. These are the short, medium and long frequencies. Altough they do not match one specific color, they are often classified as being specialized in blue, green and red colors respectively. 158

Connectionism "Connectionism is a movement in cognitive science which hopes to explain human intellectual abilities using artificial neural networks (also known as 'neural networks' or 'neural nets'). Neural networks are simplified

models of the brain composed of large numbers of units (the analogs of neurons) together with weights that measure the strength of connections between the units. These weights model the effects of the synapses that link one neuron to another. Experiments on models of this kind have demonstrated an ability to learn such skills as face recognition, reading, and the detection of simple grammatical structure" (definition by James Garson from the Standford Encyclopedia of Philosophy). 3, 65–67, 69, 73, 79, 80, 84, 86

credit-assignment problem This is a common problem on machine learning when assigning the responsability of the error to different parts of a model. For example, in Artificial Neural Network (ANN) it is difficult to assign the error of the neurons that are not directly connected to the output. 49

cross-validation Is a technique to compare models or their hyperparameters given a finite training set. The training is subdivided into smaller parts and the models are validated with one part and trained with the rest. The performance of the models in each validation section is averaged and compared with the rest of the models. 9

DBN A Deep Belief Network (DBN) is a Belief Network (BN) with multiple layers of lattent variables. See definition of Belief Network (BN). xiii, 57, 88, 159

deep learning It is a term being used lately to define Artificial Neural Networks (ANNs) that are not shallow. The hidden representation of these models is supposed to find hierarchical representations of the data. Also the hierarchical structure is more effective to reuse the features in lower layers. 36, 56

delta rule In machine learning, the delta rule is the update of the weights of a model that follow the gradient on the error surface. It is a technique to minimize the training error by performing several steps of gradient descent. 49

DoG The difference of Gaussians (DoG) detector is an edge detector that uses as a kernel the shape of a Gaussian substracting another one with a different variance. xiii, xvii, 12, 13, 29

dropout It is a regularization technique applied to Artificial Neural Networks (ANNs). It consists on the random inhibition of the output of certain neurons during the training, while keeping all the averaged outputs during the test phase. This technique has demonstrated empirically its efficacy for generalization. 54, 89

EDVAC The EDVAC (Electronic Discrete Variable Automatic Computer) was one of the first computers constructed in 1949. It was a predecesor of the ENIAC binary rather than decimal and larger computational capabilities. xiii, 70, 159

ELM An Extreme Learning Machine (ELM) is a particular case of an ANN with at least one layer of hidden units. xiii, 36, 54, 55

ENIAC The ENIAC (Electronic Numerical Integrator And Computer) was one of the first computers, constructed in 1946. xiii, 70, 159, 160

epoch In machine learning, when a model is being trained with a set of data, one epoch corresponds to one iteration over the complete set. 44

ESN An Echo State Network (ESN) is an Recurrent Neural Network (RNN) in which its connections are randomly choosen. The internal connections of the network are keept fixed during the training while the output connections are updated with a learning algorithm. This networks are eassy to train as the output weights are linear after the nonlinear parameter space, this makes the error surface quadratic and makes it solvable numerically. xiii, 88, 160

FNN A Feed-forward neural network (FNN) is a type of Artificial Neural Network (ANN) with directed and acyclic connections. xiii, 41, 160, 162, 165

GLOH The Gradient location-orientation histogram (GLOH) is an image descriptor used in computer vison. This technique is similar to SIFT but the created histogram contains more spatial bins and it is finally reduced with PCA. xiii, 14, 160

GPU A Graphics Processing Unit (GPU) is a dedicated electronic circuit focused on the visualization of images on a screen. It is designed for fast computations and modern GPUs are able operate with matrices in a very efficient manner. For that reason, they are being used for long scientific computations. xiv, 61, 92, 95–97, 115, 116, 160

Heaviside function Also known as a Threshold Function in the engineering literature, it is a discontinuous function with value zero in the negative side and one in the positive side. 37

HM The Helmholtz Machine (HM) is a type of Artificial Neural Network (ANN) with a recognition and a generative model that share the hidden variables but not the weight connections. It can be trained with the "wake-sleep" algorithm, an unsupervised method to train the recognition and the generative model simultaniously.. xiv, 88, 160

HOG The Histogram of Oriented Gradients (HOG) is an image descriptor inspired by SIFT, however, it extensively computes orientation gradients through all the image. xiv, 14, 59, 160

Hopfield network Is a type of Recurrent Neural Network (RNN) with symetric connections that is guaranteed to converge to a local minimum energy state. It can be used as a content-addressable memory. 79, 84, 85, 158

HoPS The Histogram of Pattern Sets (HoPS) is an image descriptor that randomly selects a set of other descriptors and reducing the total size. The random sets

are called transactions and are designed to maximize the separability of the descriptors on different classes while reduces the intra-class distances. xiv, 15, 160

HSL The HSL color space is a typical representation of colors in a cylindrical coordinate system. In this representation the lightness is codified in the height of the cylinder, the hue in the rotation of the rotation, and the saturation with the distance from the center axis. xiv, 93, 161

HSV The HSV color space similar to HSL, however, the lighting is represented with the variable value and is distributed in a different manner. While in the HSL the white is equaly distributed on the top surface of the cylinder, in the HSV the white is in the exact center of the top surface. xiv, 93, 161

hyperbolic tangent function It is a sigmoid function with negative and positive values on the negative and positive side respectively. It has asymptotic behaviour towards -1 and 1, and it can behave locally as linear or as a step function. For that reason, it has been extensively used as activation functions in Artificial Neural Networks (ANNs). 43

ICA The Independent Component Analysis (ICA) is a technique used in signal processing to find additive subsignals that are non-Gaussian and statistically independent. xiv, 86, 161

Infomax The maximum mutual information (Infomax) is an optimization principle that tries to minimize the amount of information needed to codify a match between a set of inputs and outputs. This principle has been studied in some biological systems and has been applied on information retrieval and machine learning algorithms. xiv, 83, 86, 87, 161

inverse problem It is a class of problems where the objective is to model a physical system given a set of measurements. 6, 9

K-cells The Koniocellular cells (K-cells) are a type of ganglion cells with their body in the retina of the eyes and with axons that extends to the LGN and the primary visual cortex. In the LGN they are situated in the strates between P-cells and M-cells. xiv, 30, 161

LGN The Lateral Geniculate Nucleus (LGN) is a section of the thalamus focused on the visual system. It receives the axons of ganglion cells from the retina through the optic nerve and optic chiasma and propagates their signals to the primary visual cortex. xiv, 27, 30, 31, 33, 132, 161–163

LoG The Laplacian of Gaussian (LoG) is a blob detector based on the Laplacian of a Gaussian. xiv, 11–13, 29, 161

logistic function Is a type of sigmoid function with values $[0, 0.5]$ on the negative side and values $[0.5, 1]$ on the positive side. 43, 162

logistic regression It is a probabilistic technique for pattern classification models for binary predictions. It uses the logistic function of a set of variables to generate the probability response. 43

LRN The Local Response Normalization (LRN) is a technique used in CNNs to normalize the local activity of neighbour neurons. xiv, 62, 89, 162

luminance It is the level of brightness (or light) of an image. By using only the luminance information of an image, only a grayscale from black to white can be represented. 122, 124

M-cells The Magnocellular cells (M-cells) are a type of ganglion cells with their body in the retina of the eyes and with axons that extends to the LGN and the primary visual cortex. In the LGN they are distributed in the first and second layers. xiv, 30, 161, 162

mean field theory It is a technique used in physics and probability theory to simplify the behaviour of large and complex systems with multiple individuals. It simplifies the complex interactions by an average value that is easier to compute. 88

memistor It is a electric component designed by Bernard Wirdrow in 1960 to store information in form of electrical impedance. It was the principal component for the posterior creation of the Adaline . 77, 157

MFNN A Multilayer Feedforward Neural Network (MFNN) is a particular type of a FNN with at least one layer of hidden units. xiv, xvii, 36, 41, 44, 49, 50, 54, 55, 86, 162

MLP A Multilayer Perceptron (MLP) is the most common name of a Multilayer Feedforward Neural Network (MFNN). See MFNN for the description. xiv, xvii, 44, 45, 162

momentum In the gradient descent algorithm, momentum is a technique used to update the parameters at a given step with the actual gradient and part of the previous update. 54

multimodal learning In machine learning, it is a set of techniques to agregate features from different sources in a variety of forms, in order to solve a pattern classification problem. 91, 93, 94, 132

multiple linear regression It is a technique to fit linearly a set of samples of a dependent variable given multiple explanatory variables. 39, 40

multivariate linear regression It is a technique to fit linearly a set of samples of multiple correlated dependent variables given multiple explanatory variables. 40

Neocognitron The Neocognitron is a type of Artificial Neural Network (ANN) with hierarchical connections and multiple layers designed to solve pattern recognition problems. It was designed by Fukushima, and inspired by the work of Hubel and Wiesel on the primary visual cortex in the 50s. 56, 83, 85

P-cells The Parvocellular cells (P-cells) are a type of ganglion cells with their body in the retina of the eyes and with axons that extends to the LGN and the primary visual cortex. In the LGN they are distributed in the last four layers (from the 3th to the 6th). xiv, 29–31, 161, 163

PCA The Principal Component Analysis (PCA) is a statistical technique to transform a set of sample points in an N-dimensional space into a new space where the new dimensions are orthogonal and sorted by variability. xiv, 14, 160, 163

PCA-SIFT The PCA-Scale-Invariant Feature Transform (PCA-SIFT) is an image representation based on SIFT that initially finds a larger feature representation and then applies PCA to reduce its dimensionality. xiv, 14, 163

Perceptron The Perceptron was one of the first Artificial Neural Networks (ANNs) developed in the 50s by Frank Rosenblatt. It was able to learn from examples and classify patterns into different classes. Altough it performs binary classifications, it can classify between multiple classes by using various output units. 41–43, 46, 75–78, 163

PReLU The Parametric Rectified Linear Unit (PReLU) is a function used in Artificial Neural Network (ANN) as an activation function. It is based on ReLU. However, it is negative or equal to zero in the negative side of the function, with a slope determined by a parameter; and grows linearly in the positive side with a different slope (commonly more pronounced). xiv, 38, 61, 163

RBF A Radial Basis Function (RBF) is a type of function that computes the distance to a specific center. One common type is the Gaussian function. xiv, 37, 86, 163

RBM A Restricted Boltzmann Machine (RBM) is a generative stochastic model with undirected connections between two layers, and without connections between units of the same layer. xiv, 57, 88, 163

ReLU The rectified linear unit (ReLU) is a function used in Artificial Neural Network (ANN) as an activation function. It is equal to zero in the negative side of the function and grows linearly in the positive side. The use of this activation function was an important part to classify images using Convolutional Neural Networks (CNNs). xiv, 37, 38, 59, 61, 63, 88, 89, 100, 163

RNN A Recurrent Neural Network (RNN) is a type of Artificial Neural Network (ANN) with loop connections between some of the units (usually between the hidden units). The activations between the loops are propagated in discrete

steps on time. These networks have been used succesfully for time series predictions. xiv, 36, 55, 56, 85, 158, 160, 163

SGD The Stochastic Gradient Descent (SGD) is an optimization method for minimizing the error of an objective function. It is based on minimizing the error in individual samples and estimates that the total error also decreases. One requisite to apply this technique is that the objective function has be differentiable. xv, 52, 53, 164

SIFT Scale-Invariant Feature Transform (SIFT) is an algorithm to extract a set of important points from an image invariant to scale and translation. See the Section 2.4.1 for an extended description. xv, 12–14, 59, 160, 163, 164

sigmoid belief network A sigmoid belief network is a directed and acyclic graph, where the output of each node computes a logistic function given the binary states of its parents. 87, 88

sigmoid function Mathematical functions with "S" shape, commonly monotonically increasing and with assymptotic behaviour at both ends. 43, 161

simple linear regression It is a technique to fit linearly a set of samples of a dependent variable given a single explanatory variable. 39

SNARC The SNARC (Stochastic Neural Analog Reinforcement Calculator) was one of the first hardware implementations of an Artificial Neural Network (ANN), developed by Marvin Lee Minsky in the 50s. It was a set of bacum tubes with random connections, and able to learn automatically. xv, xviii, 73, 164

softmax function The softmax function is a function that reduces a finite vector of real values to a vector of the same size with values in the interval $[0, 1]$. It uses the exponential function and a normalization factor to each element of the vector. 43

SOM Self-Organizing Maps (SOMs) are a type of ANNs trained using unsupervised learning to embed a complex feature representation in a reduced dimensionality space (commonly two dimensions). The embedding is achieved by using a competitive learning algorithm that computes a neighbourhood function between the samples and the different neurons, increasing the connectivity to the winning neuron and its neigbours and decreasing it to the rest. They are also known as Kohonen maps as the Finnish professor Teuvo Kohonen designed them in the 1980s. xv, 37, 79, 82–84

SURF The Speeded-Up Robust Features (SURF) is an image descriptor similar to Scale-Invariant Feature Transform (SIFT) but modified for a faster computation. xv, 14, 59, 164

SVM A Support Vector Machine (SVM) is a type of supervised binary classification model. It is designed to find a hyperplane (or a set of hyperplanes) that separates with the largest margin all the binary training samples. If there is no such hyperplane it accepts a cost to find a soft margin solution. The most basic model is the linear SVM while the kernelized method is non-parametric and finds support vectors on the training samples. The support vectors are training samples that are used to support the hyperplane. xv, 9, 59, 81, 87, 88, 164

Threshold Function Step function that takes the value of zero for inputs smaller or equal to zero and one otherwise. 37, 160

TPE The Temporal Propositional Expression (TPE) was a type of logic created in the 40s by McCulloch and Pitts to define the set of problems that were able to solve with artificial neurons activated on time steps. xv, 70, 165

unit In the context of Artificial Neural Network (ANN), one unit corresponds to one node of the network, it is also refered as an artificial neuron. 159, 162

universal approximator In the context of ANNs, it has been demonstrated that a Feed-forward neural network (FNN) with at least one hidden layer and a sufficient number of hidden units and some specific activation functions can approximate any continuous function. The set of activation functions that assures this fact are continuous sigmoidal functions; or nonconstant, bounded, and monotonically increasing continuous functions. 37

VC dimension The Vapnik-Chervonenkis dimension (VC dimension) is a measure of the capacity of a classification model to separate correctly a set of points in an N-dimensional space. For example, in a two dimensional space a linear classifier can classify up to three points in any spatial configuration, but not four. xv, 81, 165

weight decay The weight decay is a regularization technique designed for Artificial Neural Network (ANN) that adds a penalty to the size of the weights of the network into the cost function. 53

XYZ The XYZ is color space derived from the RGB color space but adapted to compensate the negative values perceived by the human visual system. xv, 16, 20

YCbCr The YCbCr is a color space used in digital images and video that encodes the luminance in the "Y" channel and the chrominance in the Cb (blue difference) and Cr (red difference). The YUV color space is the same space but for analog encoding. xv, 18

YIQ The YIQ is a color space that encodes the luminance in the Y channel while the chrominance is encoded in the orthogonal channels I and Q. It is used in the analog encoding NTSC (North America, Japan, some parts of Africa, South Korea, Taiwan, and others). xv, 18, 20, 93

YUV The YUV is a color space that encodes the luminance in the Y channel while the chrominance is encoded in the orthogonal channels U and V. It is used in the analog encoding PAL (used in Australia, Europe, except France, some parts of Africa, India, Brazil, Argentina, and others). xv, 18, 21, 93, 101, 132–134, 165