



Escuela  
Politécnica  
Superior

# Image classification using embedded spaces generated by Siamese Networks



Bachelor's degree in Computer Engineering

## Bachelor's thesis

Author:

Guillermo Lloret Talavera

Supervisors:

Juan Ramón Rico Juan

Antonio Javier Gallego Sánchez



Universitat d'Alacant  
Universidad de Alicante

June 2019



# Image classification using embedded spaces generated by Siamese Networks

---

## Author

Guillermo Lloret Talavera

## Supervisors

Juan Ramón Rico Juan

*Department of Software and Computing Systems*

Antonio Javier Gallego Sánchez

*Department of Software and Computing Systems*



BACHELOR'S DEGREE IN COMPUTER ENGINEERING



Alicante (Spain) - June 2019



*Society is changing,  
one learning algorithm at a time.*

Pedro Domingos.



# Acknowledgements

I would first like to thank my supervisors Juan Ramón and Antonio Javier. They consistently allowed this paper to be my own work but steered me in the right direction whenever they thought I needed it.

I am also grateful to my colleagues for accepting nothing less than excellence for me. They shared with me these four years making this whole process much more bearable. I would also like to thank my partner, Silvia, for putting up with me for so long and helping me to grow as a person.

Finally, I must express my very profound gratitude to my parents for providing me with unconditional support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.





# Abstract

With the emergence of new algorithms and the increase in computing power, machine learning techniques are gaining enormous importance in a variety of fields. An important limitation of these systems is the need to have a very high amount of data to train them.

Siamese networks are a special type of neural networks that have certain advantages when dealing with problems where limited data is available. In this document, an analysis of this type of networks is carried out, which provides an insight into its true potential and its limitations. A comparison between existing architectures is presented and possible improvements are introduced. In addition, its parameters are exposed and the most appropriate configuration of them for each scenario is suggested.

The final objective of this project is to establish a reference that will help data scientists to deal with problems that were difficult to address due to the reduced amount of data available until now.



# Resumen

Con la aparición de nuevos algoritmos y el aumento de la potencia de computo, las técnicas de aprendizaje automático están cobrando una importancia enorme en gran variedad de campos. Una importante limitación de estos sistemas es la necesidad de disponer de una cantidad de datos muy elevada para llevar a cabo el entrenamiento de los mismos.

Las redes siamesas son un tipo especial de redes neuronales que presentan ciertas ventajas a la hora de tratar con problemas en los que se dispone de pocos datos. En este documento se llevará a cabo un análisis de este tipo de redes que permitirá conocer su verdadero potencial y sus limitaciones. Se presenta una comparación entre las arquitecturas ya existentes y propuestas de mejora para las mismas. Además, se exponen sus parámetros y se indica qué configuración de los mismos es la más adecuada para cada escenario.

Este proyecto tiene como objetivo establecer una referencia que ayude a futuros científicos de datos a tratar problemas que eran difíciles de abordar debido a la cantidad reducida de datos disponibles hasta ahora.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Objectives . . . . .	2
1.3	Document structure . . . . .	3
<b>2</b>	<b>State of the art</b>	<b>5</b>
<b>3</b>	<b>Technologies</b>	<b>7</b>
3.1	Software . . . . .	7
3.1.1	Python . . . . .	7
3.1.2	Keras . . . . .	8
3.1.3	TensorFlow . . . . .	8
3.1.4	Scikit-learn . . . . .	8
3.1.5	NumPy . . . . .	9
3.2	Hardware . . . . .	9
3.2.1	Local . . . . .	9
3.2.2	Google Colaboratory . . . . .	10
<b>4</b>	<b>Methodology</b>	<b>11</b>
4.1	Datasets . . . . .	11
4.1.1	MNIST . . . . .	11
4.1.2	Fashion-MNIST . . . . .	13
4.1.3	CIFAR-10 . . . . .	14
4.1.4	HISPAMUS dataset . . . . .	15
4.2	Model . . . . .	16
4.3	Pairing . . . . .	17

4.3.1	Pair pairing . . . . .	18
4.3.2	Triplet pairing . . . . .	18
4.4	Classification . . . . .	18
<b>5</b>	<b>Classical datasets experimentation</b>	<b>21</b>
5.1	Baseline architecture . . . . .	21
5.1.1	Convolutional neural network . . . . .	21
5.1.1.1	MNIST . . . . .	22
5.1.1.2	Fashion-MNIST . . . . .	22
5.1.1.3	CIFAR-10 . . . . .	23
5.1.2	Results . . . . .	23
5.1.2.1	MNIST . . . . .	23
5.1.2.2	Fashion-MNIST . . . . .	24
5.1.2.3	CIFAR-10 . . . . .	24
5.2	Pair siamese architectures . . . . .	25
5.2.1	Pair siamese network . . . . .	25
5.2.1.1	Parameters . . . . .	26
5.2.1.1.1	Optimizer . . . . .	26
5.2.1.1.2	Embedding dimensionality . . . . .	27
5.2.1.1.3	Pairing proportion . . . . .	28
5.2.1.1.4	Loss function . . . . .	28
5.2.1.1.5	Margin . . . . .	30
5.2.2	Pair siamese with classification network . . . . .	30
5.2.2.1	Parameters . . . . .	31
5.2.2.1.1	Loss contribution . . . . .	31
5.2.3	Results . . . . .	32
5.2.3.1	MNIST . . . . .	32
5.2.3.2	Fashion-MNIST . . . . .	33
5.2.3.3	CIFAR-10 . . . . .	34
5.3	Triplet siamese architecture . . . . .	35
5.3.1	Triplet siamese network . . . . .	35
5.3.2	Triplet siamese with classification network . . . . .	36
5.3.3	Triplet siamese with external memory network . . . . .	37

5.3.4	Results . . . . .	37
5.3.4.1	MNIST . . . . .	38
5.3.4.2	Fashion-MNIST . . . . .	39
5.3.4.3	CIFAR-10 . . . . .	40
5.4	Embedded spaces classification algorithm . . . . .	41
5.4.1	Average class distance . . . . .	41
5.4.2	Support Vector Regression . . . . .	41
5.4.3	$K$ -nearest neighbors . . . . .	41
5.4.4	Random decision forest . . . . .	42
5.4.5	Results . . . . .	42
<b>6</b>	<b>HISPAMUS dataset experimentation</b>	<b>45</b>
6.1	Results . . . . .	46
<b>7</b>	<b>Conclusions and future work</b>	<b>49</b>
	<b>References</b>	<b>51</b>





# List of Figures

2.1	Architecture proposed in Bromley et al. (1994) . . . . .	5
2.2	Omniglot examples . . . . .	6
4.1	Layers sharing configurations . . . . .	17
5.1	Convolutional architecture . . . . .	21
5.2	Baseline model of MNIST . . . . .	22
5.3	Baseline model of Fashion-MNIST . . . . .	22
5.4	Baseline model of CIFAR-10 . . . . .	23
5.5	Training of MNIST using convolutional neural network . . . . .	24
5.6	Training of Fashion-MNIST using convolutional neural network . . . . .	24
5.7	Training of CIFAR-10 using convolutional neural network . . . . .	25
5.8	Pair siamese architecture . . . . .	25
5.9	Pair loss functions . . . . .	29
5.10	Pair siamese with classification architecture . . . . .	30
5.11	Training of MNIST using pair siamese with classification network . . . . .	32
5.12	Embedded space of MNIST using pair siamese with classification network . . . . .	33
5.13	Training of Fashion-MNIST using pair siamese with classification network . . . . .	33
5.14	Embedded space of Fashion-MNIST using pair siamese with classification network . . . . .	34
5.15	Training of CIFAR-10 using pair siamese with classification network . . . . .	34
5.16	Embedded space of CIFAR-10 using pair siamese with classification network . . . . .	35
5.17	Triplet siamese architecture . . . . .	36
5.18	Triplet siamese with classification architecture . . . . .	36
5.19	Triplet siamese with external memory architecture . . . . .	37

5.20	Training of MNIST using triplet siamese network . . . . .	38
5.21	Embedded space of MNIST using triplet siamese network . . . . .	38
5.22	Training of Fashion-MNIST using triplet siamese with classification network . . . . .	39
5.23	Embedded space of Fashion-MNIST using triplet siamese with classification network . . . . .	39
5.24	Training of CIFAR-10 using triplet siamese with classification network .	40
5.25	Embedded space of CIFAR-10 using triplet siamese with classification network . . . . .	40
6.1	HISPAMUS pair siamese with classification embedded space . . . . .	47

# List of Tables

4.1	Data distribution of MNIST . . . . .	12
4.2	State of the art of MNIST . . . . .	12
4.3	Data distribution of Fashion-MNIST . . . . .	13
4.4	State of the art of Fashion-MNIST . . . . .	13
4.5	Data distribution of CIFAR-10 . . . . .	14
4.6	State of the art of CIFAR-10 . . . . .	15
4.7	Data distribution of HISPAMUS dataset . . . . .	16
5.1	Accuracy results of baseline . . . . .	23
5.2	Optimizers accuracy comparison . . . . .	26
5.3	Embedding dimensionalities accuracy comparison . . . . .	27
5.4	Pairing proportions accuracy comparison . . . . .	28
5.5	Loss functions accuracy comparison . . . . .	29
5.6	Margins accuracy comparison . . . . .	30
5.7	Loss contribution configurations accuracy comparison . . . . .	31
5.8	Pair siamese architectures accuracy comparison . . . . .	32
5.9	Triplet siamese architectures accuracy comparison . . . . .	37
5.10	Classification algorithms accuracy comparison . . . . .	42
6.1	Errors of HISPAMUS leave-one-out cross-validation . . . . .	46



# Chapter 1

## Introduction

This document presents the research carried out at University of Alicante focused on siamese networks. The project was led and supervised by professors Juan Ramón Rico Juan and Antonio Javier Gallego Sánchez.

### 1.1 Motivation

Nowadays, due to important advances in machine learning techniques, companies have realized about the importance of collecting all the information they generate for further processing. Big Data is the field of study that takes care of this whole process, and companies are being forced to invest in it if they do not want to be left behind. Recently, neural networks are being used to address a wide variety of problems due to their proven ability to deal with almost any type of data. The greater the amount of available data, the greater the performance of this type of systems.

However, there are still multiple areas where the amount of information available is not enough to use this type of techniques. An example of this is the recognition of endangered animals; it is very difficult to obtain images that can be used to train classification systems due to the extremely low number of specimens. The famous data science portal, kaggle, held a competition in 2019 to identify species of humpback whales using only images of their tail <sup>1</sup>. The final objective of this competition was to assist in conservation efforts by improving current classification techniques. Multiple participants opted to create models based on siamese architectures.

---

<sup>1</sup><https://www.kaggle.com/c/humpback-whale-identification>

## 1.2 Objectives

The main objective of this project is to obtain a detailed analysis of siamese networks, a type of neural network with a particularly good performance in problems with a limited data, applied to image classification following the few-shot approach. Another important objective is to know the behaviour of non-conventional techniques when used in embedded space classification.

In order to fulfil these objectives, the following tasks are required:

- **Datasets selection:** Datasets whose state of the art is well known will be used to facilitate comparisons. Adaptations will be made to comply with a few-shot approach. The potential and limitations of siamese networks will be determined by the difference in complexity among datasets.
- **Standard model implementation:** A model will be developed and trained according to the original architecture. It will be compared with traditional models in terms of accuracy and training time.
- **Hyperparameters tuning:** All possible parameters will be tested to find those that have the greatest impact on performance. The most appropriate configuration of these values will be identified for each scenario.
- **Research of alternatives:** Documents that present variants of the standard architecture will be searched. Models using these architectures will be implemented and trained.
- **Proposal for improvements:** Possible modifications that improve some aspects of existing models will be explored.
- **Model comparison:** The results of the most promising models will be compared to determine which one is the best among them.
- **Classification:** Different classification algorithms from embedded spaces will be implemented and they will be compared in terms of speed and accuracy.
- **Evaluation:** The best performing techniques will be applied to a real-world problem to verify the results.

## 1.3 Document structure

For ease of reading, the content of this document is organized into chapters and sections. Below is a description of the structure that will be followed.

- **Chapter 1: Introduction**  $\Rightarrow$  Covers the fundamentals of the project and describes the objectives to be achieved.
- **Chapter 2: State of the art**  $\Rightarrow$  Provides a general theoretical basis of the field in which the problem to be analyzed is found.
- **Chapter 3: Technologies**  $\Rightarrow$  Presents the main tools to use in order to efficiently analyze the problem.
- **Chapter 4: Methodology**  $\Rightarrow$  Exposes the details of the process that will be followed during the realization of the project.
- **Chapter 5: Classical datasets experimentation**  $\Rightarrow$  Provides the results of the experiments carried out on datasets usually used in machine learning.
- **Chapter 6: HISPAMUS dataset experimentation**  $\Rightarrow$  Shows the results of applying the best models obtained in the previous chapter to a real-world problem.
- **Chapter 7: Conclusions and future work**  $\Rightarrow$  Presents conclusions about the data obtained in the experiments and indicates possible future lines of research.





# Chapter 2

## State of the art

Siamese networks were introduced in the early 1990s by researchers at AT&T Bell Laboratories to solve the problem of signature verification (Bromley et al., 1994). They created a system consisting of two twin sub-networks (figure 2.1) in charge of extracting features and a joining neuron that computes the distance between them.

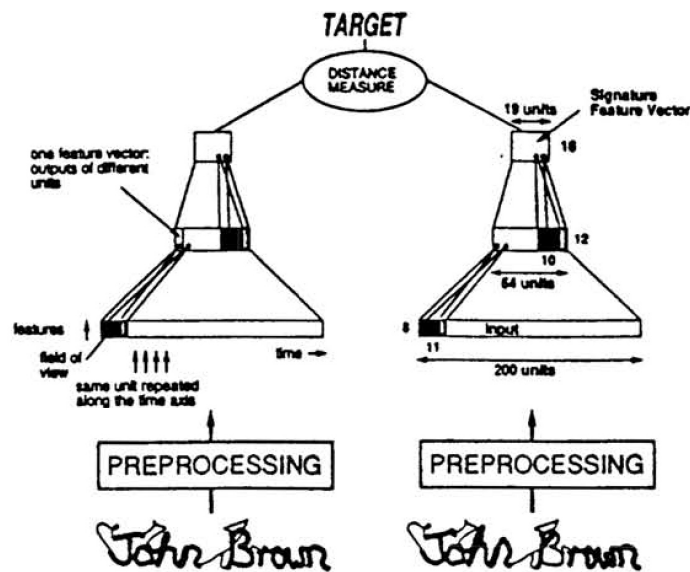


Figure 2.1: Architecture proposed in Bromley et al. (1994)

In recent years the popularity of siamese networks has increased due to their embedding capabilities. Although their use is not as widespread as that of other machine learning techniques, siamese networks can be found in multiple fields of computer vision like face recognition (Chopra et al., 2005; Sun et al., 2014; Schroff et al., 2015), image descriptors generation (Zagoruyko and Komodakis, 2015; Simo-Serra et al., 2015; Kumar et al., 2016), object tracking (Tao et al., 2016; Bertinetto et al., 2016; Tompson

et al., 2015), stereo matching (Luo et al., 2016) and image retrieval (Gordo et al., 2017).

Traditional neural networks learn to classify inputs by generating a probability distribution over all the potential classes. The best results are obtained when there are few possible classes and a large amount of data from each of them is available. They have another important limitation: once a network is trained it cannot classify inputs of new classes, to accomplish this it would be necessary to re-train the model.

On the other hand, siamese networks learn to determine how related are two or more inputs. They distribute each of the inputs using a meaningful embedding space where related items are placed close to each other. Siamese networks have been successfully employed in zero/one/few shot learnings applications where there is not enough data, the total number of classes is large and it changes over time. Both Koch et al. (2015) and Ye and Guo (2018) use siamese networks to classify images of Omniglot (figure 2.2), a dataset of 1623 characters belonging to 50 different alphabets widely used in one-shot experiments. Using siamese networks, better results are obtained than with traditional techniques, reaching an accuracy of 92%, which is very close to human rates.



Figure 2.2: Omniglot examples  
(Source: <https://github.com/brendenlake/omniglot>)

# Chapter 3

## Technologies

Choosing which technologies would be used to carry out the experiments was one of the most important objectives in the early phases of this project. In this chapter, we will describe the tools that are employed, directly or indirectly, in the development of this project disclosing the main motivations that have led us to choose them.

### 3.1 Software

All the experiments are implemented in Python along with the Keras library. We use Git, an open source version control system, to manage the project. The repository is accessible from any computer with internet access as it is hosted in Github. The code is written using different text editors among which Visual Studio Code stands out for its autocompletion and linting capabilities. Two online tools have been used to create graphs and diagrams: <http://alexlenail.me/NN-SVG> (convolutional architectures) and <https://www.draw.io> (siamese architectures).

#### 3.1.1 Python



Python is an interpreted, high-level programming language. According to the TIOBE Index<sup>1</sup> for April 2019, Python is the fourth most popular programming language and is seeing a steady rise. It can run on all major operating systems: Ubuntu 18, Windows 10 and macOS Mojave.

Python is widely used in machine learning due to the large collection of libraries and

---

<sup>1</sup><https://www.tiobe.com/tiobe-index/>

frameworks available. These tools, in conjunction with the simplicity of the language, reduce considerably the development time.

We use Python version 3.6.7 as it is compatible with most modern libraries. This version of the language is distributed under Python Software Foundation License, an open source license. To simplify the installation of the required libraries we use pip version 19.0.3, the native package manager.

### 3.1.2 Keras



Keras is a Python neural network framework capable of running on top of TensorFlow, CNTK, or Theano. According to their public statistics<sup>2</sup> for 2018, it has over 250.000 users. It implements the most commonly used neural network architectures and provides an intuitive API focused on fast experimentation. It offers an acceptable trade-off between abstraction and low-level detail.

We use Keras version 2.2.4, as it was the last release when the project started. This version is distributed under Massachusetts Institute of Technology (MIT) License, an open source license.

### 3.1.3 TensorFlow



TensorFlow is an open source library for machine learning developed by Google. It is used as the backend of Keras because of its flexibility and seamless integration<sup>3</sup>. It supports GPU acceleration using CUDA and cuDNN, which significantly reduces the time needed for training.

We use TensorFlow version 1.12.0 along with CUDA Toolkit version 9.0.

### 3.1.4 Scikit-learn



Scikit-learn is an open source machine learning library for Python. It includes the most common classification, regression and clustering algorithms. The documentation is detailed and contains good examples.

We use Scikit-learn version 0.20.0, as it was the last release when the project started.

---

<sup>2</sup><https://keras.io/why-use-keras/>

<sup>3</sup><https://www.tensorflow.org/guide/keras>

### 3.1.5 NumPy



NumPy is an open source Python library that supports multidimensional arrays operations and includes a large collection of high-level mathematical functions. Large operations are executed more efficiently using NumPy methods than built-in Python instructions. Most of the implementation of NumPy is written in C and wrapped in Python classes.

We use NumPy version 1.15.4, as it was the last release when the project started.

## 3.2 Hardware

Hardware is one of the most important parts when dealing with a large amount of data. Traditional approaches consume most of the time in training and very little time in evaluating results. However, in our experiments, the bottleneck was found in the classification and evaluation stages, as we have to work with the complete set of embeddings generated in the training phase.

### 3.2.1 Local

Most of the experimentation is carried out using a computer with the following specifications:

- CPU: Intel Core i7 - 7700K
  - Frequency: 4.20 GHz
  - Cache: 8 MB
  - Cores: 4
  - Threads: 8
- GPU: Gigabyte GeForce GTX 970 G1
  - Frequency: 1354 MHz
  - Memory: 4 GB GDDR5
  - CUDA cores: 1664
- RAM: 8 GB DDR4

- SO:       Ubuntu 18 (amd64)  
              Windows 10 (64-bit)

### 3.2.2 Google Colaboratory

Google Colaboratory, also know as Colab, is a free Jupyter notebook environment that works in the cloud. It supports Python 3 code including all the libraries stated in section 3.1. It offers the possibility of running any process as long as it does not exceed 12 hours of uninterrupted executing time. We have used Google Colaboratory at certain points when the local computer was not available for use.

The remote machine on which it runs has the following specifications:

- CPU:     Intel Xeon
  - Frequency:     2.30 GHz
  - Cache:         45 MB
  - Cores:         1
  - Threads:       2
- GPU:     Nvidia Tesla T4
  - Frequency:     1582 MHz
  - Memory:        16 GB GDDR6
  - CUDA cores:    2560
- RAM:     12.6 GB

# Chapter 4

## Methodology

In this chapter, we will briefly introduce the experimentation methodology which will be followed during the rest of this document. All implemented models will be included in the following repository:

[https://github.com/GuillerLT/siamese\\_neural\\_networks](https://github.com/GuillerLT/siamese_neural_networks)

### 4.1 Datasets

The choice of datasets was one of the first points addressed in the project. We decided to use three classical datasets, as their state of the art is known in detail, allowing for easy comparison of results. Most of these datasets have a large amount of data for training, however, the experiments follow the few-shot approach, so it is necessary to use a subsample of the available data. Once most of the experiments were completed, the most successful techniques were applied to a real-world problem. The available dataset is composed of unbalanced classes with a low number of examples.

#### 4.1.1 MNIST

MNIST (LeCun, 1998) is a large dataset of handwritten digits (0-9) frequently used in the field of machine learning. This dataset is natively supported by Keras, allowing us to import it easily. It was created as a combination of two National Institute of Standards and Technology (NIST) databases<sup>1</sup>:

---

<sup>1</sup><https://www.nist.gov/srd/shop/special-database-catalog>

- Special Database 1: Digits written by high school students.
- Special Database 3: Digits written by employees of the US Census Bureau.

The original data from NIST contains black and white images. In MNIST these images are normalized to fit in 20x20 boxes and centred in 28x28 images by computing the centre of mass of the pixels.

Digit	Samples		Examples
	<i>Train</i>	<i>Test</i>	
0	5.923	980	0 0 0 0 0 0 0 0 0 0 0 0 0 0
1	6.742	1.135	1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	5.985	1.032	2 2 2 2 2 2 2 2 2 2 2 2 2 2
3	6.131	1.010	3 3 3 3 3 3 3 3 3 3 3 3 3 3
4	5.842	982	4 4 4 4 4 4 4 4 4 4 4 4 4 4
5	5.421	892	5 5 5 5 5 5 5 5 5 5 5 5 5 5
6	5.918	958	6 6 6 6 6 6 6 6 6 6 6 6 6 6
7	6.265	1.028	7 7 7 7 7 7 7 7 7 7 7 7 7 7
8	5.881	974	8 8 8 8 8 8 8 8 8 8 8 8 8 8
9	5.949	1.009	9 9 9 9 9 9 9 9 9 9 9 9 9 9

Table 4.1: Data distribution of MNIST

Table 4.1 shows the number of samples for both training and test sets. Using all available data, error rates lower than human levels have been achieved (Simard et al., 1993). Table 4.2 shows some of the models that reach the best results.

Method	Error %	Reference
DropConnect	0.21	Wan et al. (2013)
Multi-column DNN	0.23	Cireřan et al. (2012)
Augmented Pattern Classification	0.23	Sato et al. (2015)
Maxout Network in Network	0.24	Chang and Chen (2015)
Pooling Functions in CNN	0.29	Lee et al. (2016)

Table 4.2: State of the art of MNIST

(Source: [https://rodrigob.github.io/are\\_we\\_there\\_yet/](https://rodrigob.github.io/are_we_there_yet/))

The amount of training data in the MNIST dataset (table 4.1) is too high for a few-shot approach. To accommodate these restrictions, only the first 100 samples of each class are selected for training. The dataset for testing remains unchanged.



### 4.1.2 Fashion-MNIST

Fashion-MNIST (Xiao et al., 2017) is a large dataset of clothing articles from Zalando<sup>2</sup>. This dataset emerged as a response to the needs of data scientists. Traditional MNIST is too easy for modern techniques and it is overused. Fashion-MNIST contains 28x28 pixel grayscale images associated with one of the ten possible classes. This dataset is also natively supported by Keras, so it can be easily imported.






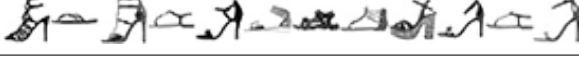




Label	Description	Samples		Examples
		<i>Train</i>	<i>Test</i>	
0	Top	6.000	1.000	
1	Trouser	6.000	1.000	
2	Pullover	6.000	1.000	
3	Dress	6.000	1.000	
4	Coat	6.000	1.000	
5	Sandal	6.000	1.000	
6	Shirt	6.000	1.000	
7	Sneaker	6.000	1.000	
8	Bag	6.000	1.000	
9	Boot	6.000	1.000	

Table 4.3: Data distribution of Fashion-MNIST

As it is a more complex dataset than MNIST, the error rate of the best models are slightly higher. Still, the results are close to human levels, as shown in table 4.4.

Method	Error %	Reference
FreezeOut	3.3	Brock et al. (2017)
Random Erasing Data Augmentation	3.7	Zhong et al. (2017)
DENSER	4.7	Assunçao et al. (2018)

Table 4.4: State of the art of Fashion-MNIST  
(Source: <https://github.com/zalando-research/fashion-MNIST>)

<sup>2</sup><https://www.zalando.com>

The amount of training data in the Fashion-MNIST dataset (table 4.3) is too high for a few-shot approach. To accommodate these restrictions, only the first 100 samples of each class are selected for training. The dataset for testing remains unchanged.

### 4.1.3 CIFAR-10

CIFAR-10 (Krizhevsky and Hinton, 2009) is a commonly used dataset created by the Canadian Institute For Advanced Research. It is a subset of the Visual Dictionary<sup>3</sup>, a collection of 80 million images. CIFAR-10 contains 32x32 pixel color images in 10 different classes. This dataset is also natively supported by Keras, and thus it can be imported to our project easily.

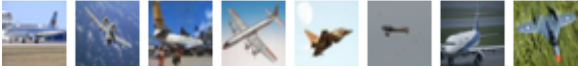

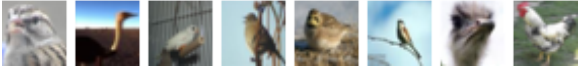

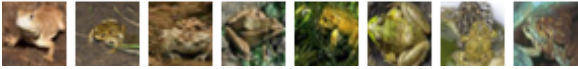
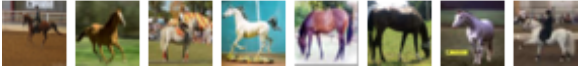
Label	Description	Samples		Examples
		<i>Train</i>	<i>Test</i>	
0	Airplane	5.000	1.000	
1	Automobile	5.000	1.000	
2	Bird	5.000	1.000	
3	Cat	5.000	1.000	
4	Deer	5.000	1.000	
5	Dog	5.000	1.000	
6	Frog	5.000	1.000	
7	Horse	5.000	1.000	
8	Ship	5.000	1.000	
9	Truck	5.000	1.000	

Table 4.5: Data distribution of CIFAR-10

CIFAR-10 is significantly more complex than MNIST and Fashion-MNIST. Initially the results did not reach 80% of accuracy, but in recent years there has been progress,

<sup>3</sup><http://groups.csail.mit.edu/vision/TinyImages/>

as shown in table 4.6. This difficulty is a result of some inherent characteristics of the dataset (Karpathy, 2011). The following aspects are some of the most problematic:

- **Variety:** The examples of a class are extremely diverse. For example, the *dog* class has images of significantly different breeds.
- **Perspective:** Examples of the different classes are shown from varying angles and magnifications.
- **Occlusion:** In some cases the examples are shown completely, while in others only a certain part is visible.

Method	Error %	Reference
Fractional Max-Pooling	3.47	Graham (2015)
Convolutional Net	4.41	Springenberg et al. (2014)
LSUV initialization	5.84	Mishkin and Matas (2015)
Pooling Functions in CNN	6.05	Lee et al. (2016)
Spatially-sparse CNN	6.28	Graham (2014)

Table 4.6: State of the art of CIFAR-10  
(Source: [https://rodrigob.github.io/are\\_we\\_there\\_yet/](https://rodrigob.github.io/are_we_there_yet/))

The amount of training data in the CIFAR-10 dataset (table 4.5) is too high for a few-shot approach. To accommodate these restrictions, only the first 100 samples of each class were selected for training. The dataset for testing remains unchanged.

#### 4.1.4 HISPAMUS dataset

HISPAMUS (Iñesta et al., 2018) project aims to provide smart access to archival manuscripts of music scores. Part of the used dataset has been extracted from Zaragoza cathedrals musical archives (Calvo-Zaragoza et al., 2016; Rizo Valero et al., 2018). Instead of using the complete dataset, we will select the images of the 5 less frequent classes. The model achieved in the HISPAMUS project has an error rate of 100% when classifying elements of these classes using  $K$ -nearest neighbors or convolutional neural networks approaches.

The images used for the training have been pre-processed by cropping the musical signs and converting them to grayscale. Since the images have different sizes, they have been normalized by centring them in a white canvas of 200x200 pixels.






Label	Description	Samples	Examples
0	Double whole stem	17	
1	Triple whole stem	11	
2	Longa	6	
3	Double whole	3	
4	Quadruple whole stem	2	

Table 4.7: Data distribution of the 5 less frequent classes of HISPAMUS dataset

Since the available data is limited, a resampling strategy is used. In particular, the leave-one-out cross-validation<sup>4</sup> (LOOCV) procedure is used as a method of evaluating the model. This approach involves using one observation as the validation set and the remaining observations as the training set. This is repeated for each of the elements of the set and the accuracy of all the repetitions is averaged.

## 4.2 Model

A siamese network consists of two or more twin neural network, each of them receiving one of the inputs. Figure 4.1 shows the possible layers sharing configurations. All experiments use full-share architectures because the paired samples always belong to the same domain. For each dataset, a well-known network with the softmax layer removed is used as a shared network. An implementation has been made as parameterized as possible to ease the testing of different parameter configurations.

The outputs from each subnetwork are used to calculate the similarity between elements. Distance metric functions ( $D$ ) are methods used to measure this similarity. Two elements are considered to be closely related if the value returned by any  $D$  is close to zero. These functions are characterized for satisfying the following properties<sup>5</sup>:

- Non-negativity:  $D(x, y) \geq 0$
- Identity of discernible:  $D(x, y) = 0 \iff x = y$

<sup>4</sup><https://academic.oup.com/bioinformatics/article/21/15/3301/195433>

<sup>5</sup>[http://slazebni.cs.illinois.edu/spring17/lec09\\_similarity.pdf](http://slazebni.cs.illinois.edu/spring17/lec09_similarity.pdf)

- Symmetry:  $D(x, y) = D(y, x)$
- Triangle inequality:  $D(x, z) \leq D(x, y) + D(y, z)$

We will experiment with various distance metric functions to determine which of them is most favourable for training the models.

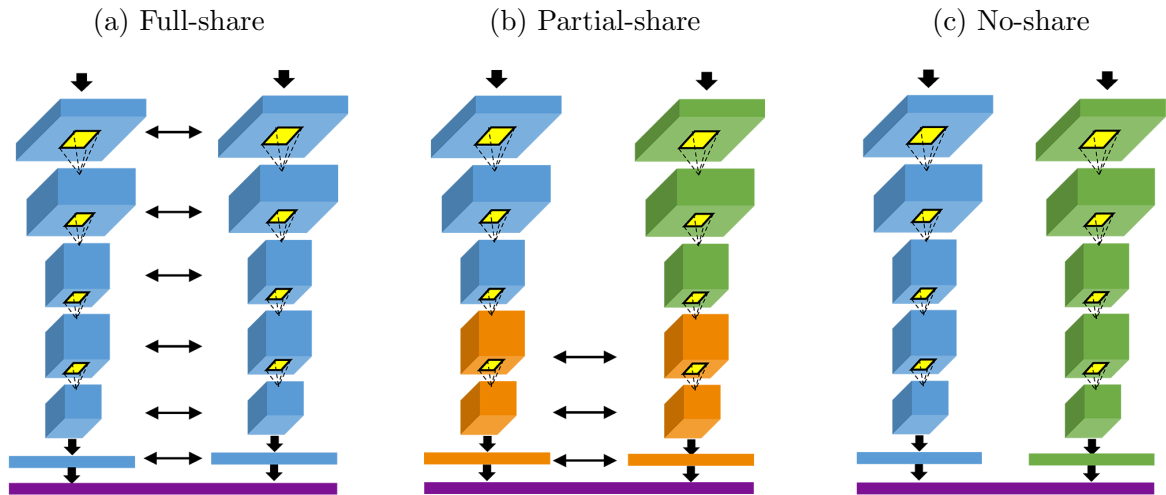


Figure 4.1: Layer sharing configurations  
(Source: <http://conteudo.icmc.usp.br/>)

## 4.3 Pairing

Siamese networks receive multiple inputs. According to the number of inputs, pairs (two inputs) and triplets (three inputs) can be distinguished. Each element of the dataset is composed of a single image, so it is necessary to carry out a pairing process that creates the necessary sets.

Keras offers the possibility of creating custom data generators, which allow the generation of batches on the fly. The first advantage of this tool lies in its memory usage: it is not necessary to load all data at the same moment, at any particular time only one batch is stored. The second advantage is given by the possibility of varying the pairings in each epoch, which improves training.

Generators for pairs and triplets siamese networks follow slightly different strategies motivated by the intrinsic characteristics of each architecture, but they share the same basic principles: Positive pairs, those formed by elements of the same class, will have an expected output of 0. In contrast, negative pairs, those formed by elements of a different

class, will have an expected output of 1. In certain variants of siamese networks, the data generator has been modified to include information on the class of each of the elements. Network training is significantly affected by data distribution. Better results are obtained when the number of elements of each class is equally distributed in a batch. Implementation details for each generator are detailed in the following sections.

### 4.3.1 Pair pairing

Pair siamese networks receive two inputs, so each image must be paired with another to form an element. Pairing is done randomly but some restrictions have been added to improve results:

- A positive pair cannot be formed by an element and itself.
- If more than one positive pair is made, the same couple of elements cannot be matched more than once.
- If more than one negative pair is performed, the elements with which it is paired must belong to different classes.

### 4.3.2 Triplet pairing

Triplet siamese networks receive three inputs. One of them is called anchor and is paired with two other elements, one positive and one negative. Both pairings are made randomly but in the positive case, it is ensured that the anchor does not pair with itself.

## 4.4 Classification

Natively, siamese networks return a similarity value between inputs. It would be possible to determine a limit above which two inputs are considered to belong to the same class. According to this premise, a new input could be compared with each of the training elements and the percentage of belonging to each class could be determined. The class with the highest percentage of belonging would be used as the output of the classifier. This process is very time-consuming and the results are not very precise.

In this project, the embedded spaces generated by the shared networks will be stored so they do not have to be computed every time they are required. Various techniques, such as random forest or k-nearest neighbours, will be applied to these embedded spaces to classify new inputs in an efficient way.

In addition, t-Distributed Stochastic Neighbor Embedding (Maaten and Hinton, 2008) will be used to graphically represent embedded spaces despite their high dimensionality. T-SNE<sup>6</sup> is a manifold learning technique that converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence (Kullback and Leibler, 1951) between the joint probabilities of the low-dimensional embedding and the high-dimensional data. Its function cost is not convex, which causes different results depending on the initialization.

---

<sup>6</sup><https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE>





# Chapter 5

## Classical datasets experimentation

Before testing on a real-life problem, tests will be carried out on classic datasets (MNIST, Fashion-MNIST and CIFAR-10) adapted to the few-shot approach according to section 4.1. Besides using the original siamese architecture, some alternatives proposed by us and others proposed by various authors will be used to see if they improve the initial approach. All of the tables in this chapter use accuracy as a measure of performance. Graphs showing the progress of the training phase are also included.

### 5.1 Baseline architecture

To estimate the performance of siamese networks it is necessary to know the accuracy of traditional approaches. There are many studies on these three datasets, but none have been found that use only a portion of the training collection. Rather, they all use all the available data in said collection.

#### 5.1.1 Convolutional neural network

To establish a realistic baseline, three convolutional neural networks have been implemented using the architecture shown in figure 5.1. Each model receives a single input, applies a series of convolutions and emits an output using a *softmax* layer. The loss is given by the categorical cross-entropy function.

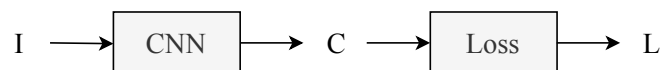


Figure 5.1: Convolutional architecture

After testing different models, the following ones have been selected for each dataset. Their complexity is consistent with the complexity of the data they are meant to classify. These models (without the *softmax* layer) will be used as shared networks for the siamese approaches.

### 5.1.1.1 MNIST

The model shown in figure 5.2 is the simplest model, as MNIST is the easiest of the three datasets. Using all available data of each class, this model achieves an accuracy of 99.25% after 12 epochs <sup>1</sup>.

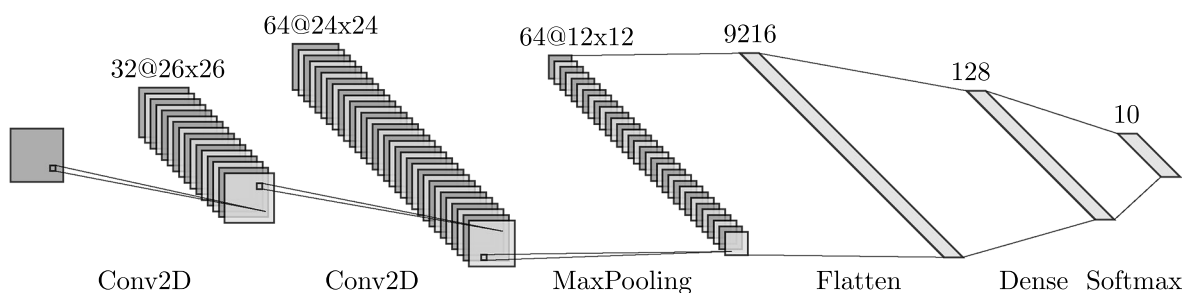


Figure 5.2: Baseline model of MNIST

### 5.1.1.2 Fashion-MNIST

The model shown in figure 5.3 is slightly more complex than the previous one, as it uses an extra *max-pooling* layer. Using all available data of each class, this model achieves an accuracy of 94% after 25 epochs <sup>2</sup>.

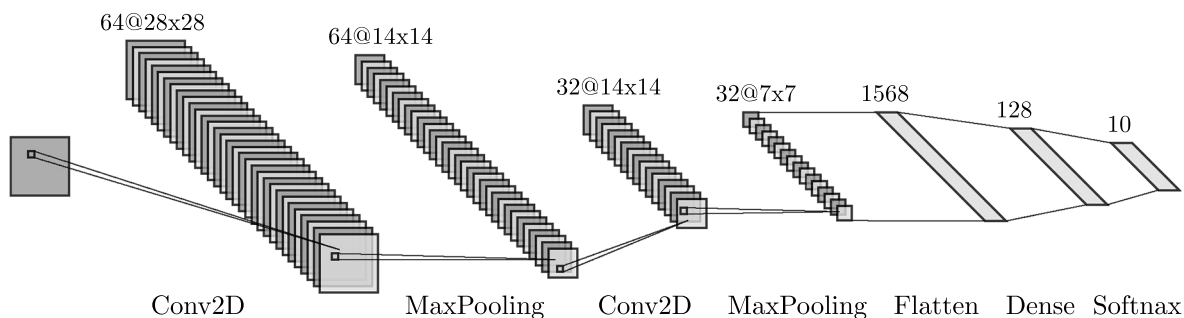


Figure 5.3: Baseline model of Fashion-MNIST

<sup>1</sup>[https://github.com/keras-team/keras/blob/master/examples/mnist\\_cnn.py](https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py)

<sup>2</sup><https://www.pyimagesearch.com/2019/02/11/fashion-MNIST-with-keras-and-deep-learning/>

### 5.1.1.3 CIFAR-10

The model shown in figure 5.4 is the most complex model of the three. It alternates between multiple convolutional layers and *max-pooling*. Using all available data of each class, this model achieves an accuracy of 79% after 50 epochs <sup>3</sup>.

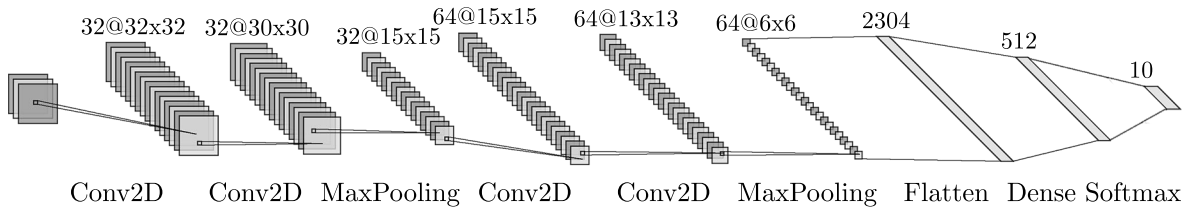


Figure 5.4: Baseline model of CIFAR-10

## 5.1.2 Results

The results of classifying each dataset with its corresponding model are shown in table 5.1. These values will be used as a baseline for the rest of the comparisons in this chapter.

Architecture	MNIST	Fashion-MNIST	CIFAR-10
CNN	0.92	0.82	0.42

Table 5.1: Accuracy results of baseline

In the following subsections, the results of each dataset are presented in detail. Special emphasis will be placed on the analysis of the impact on performance caused by not using all available data.

### 5.1.2.1 MNIST

Training this model with the reduced dataset gives results relatively close to its state of the art (section 4.1.1). Thanks to the simplicity of the architecture the training process is very fast and it achieves good results after only ten epochs (figure 5.5). The training and test losses evolve in parallel, so it can be said that the model is able to generalize based on the data it receives. No significant improvements in accuracy are achieved after 12 epochs.

<sup>3</sup>[https://keras.io/examples/cifar10\\_cnn/](https://keras.io/examples/cifar10_cnn/)

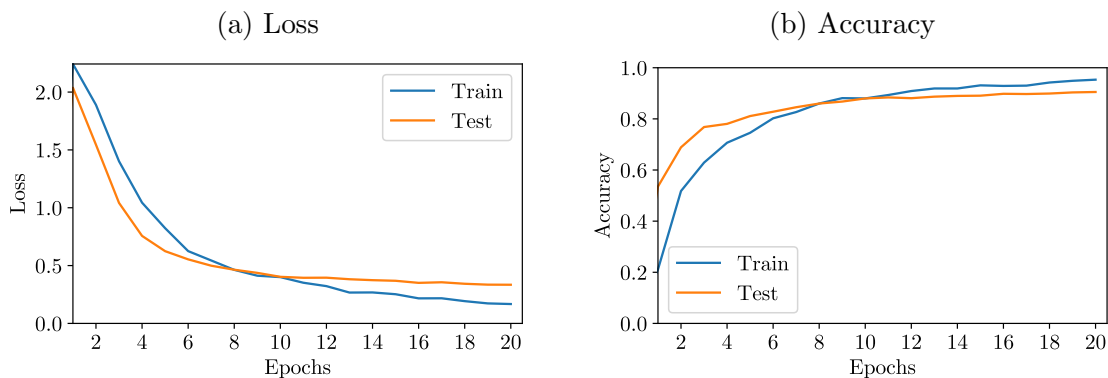


Figure 5.5: Training of MNIST using convolutional neural network

### 5.1.2.2 Fashion-MNIST

The training of this model with the reduced dataset has a greater impact on precision, which makes a significant difference compared to the state of the art (section 4.1.2). However, both loss and precision evolve in parallel during training (figure 5.6), which indicates that it is learning without excessive overfitting of the data it receives.

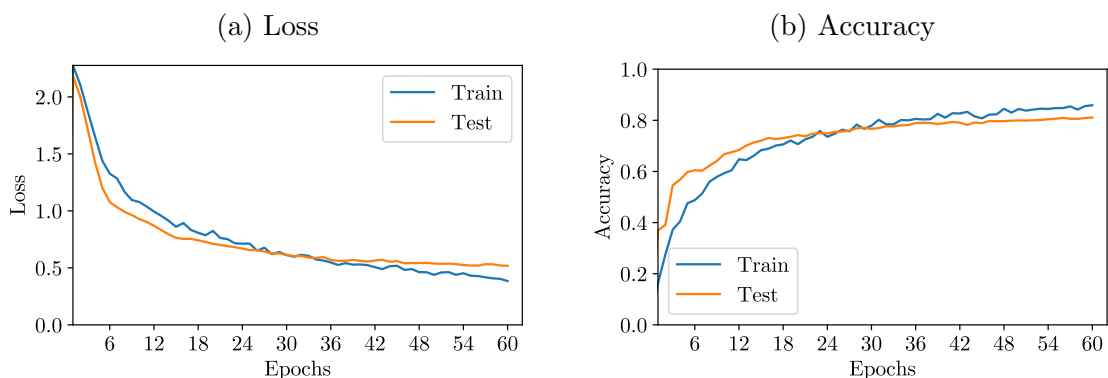


Figure 5.6: Training of Fashion-MNIST using convolutional neural network

### 5.1.2.3 CIFAR-10

The difference between this model, trained with a limited dataset, and the state of the art is notable since the accuracy is reduced by half. As the training phase progresses the test loss increases while the training loss decreases (figure 5.7) due to overfitting. At the end of the training, the model is able to perfectly classify the images it already has seen, but it has serious difficulties classifying new ones.

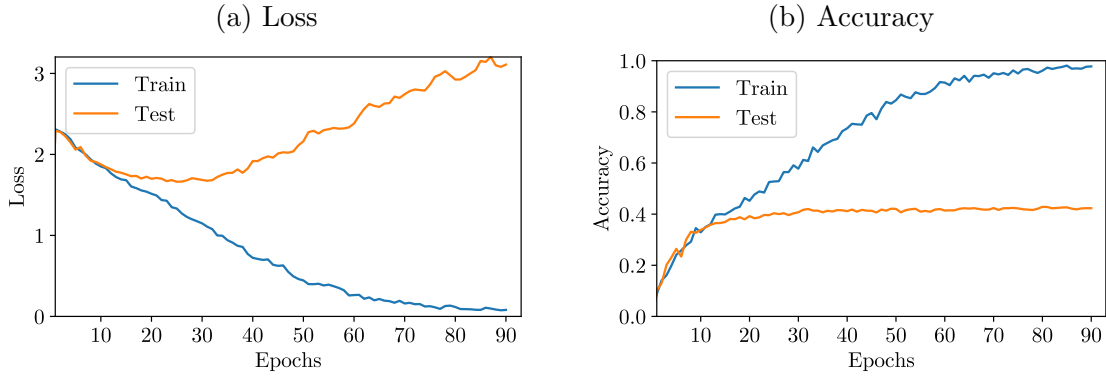


Figure 5.7: Training of CIFAR-10 using convolutional neural network

## 5.2 Pair siamese architectures

Pair siamese networks have two inputs that are processed by each of the two shared parts. In this section, we will analyze the results of the Keras implementation of multiple models based on the original architecture and a variant of it. The shared part of the network for each dataset is the same as the corresponding neural network of section 5.1.1 (removing the *softmax* layer).

### 5.2.1 Pair siamese network

The networks in this section are directly inspired by the architecture originally proposed in Bromley et al. (1994). Each model receives two different inputs which are processed by the shared part of the network. As shown in figure 5.8, two embeddings are generated after this process and one of the possible loss functions is applied to them. If the two inputs belong to the same class the distance between the embeddings is expected to be as close to zero as possible. On the other hand, if they belong to different classes, the distance is expected to be greater than a certain margin.

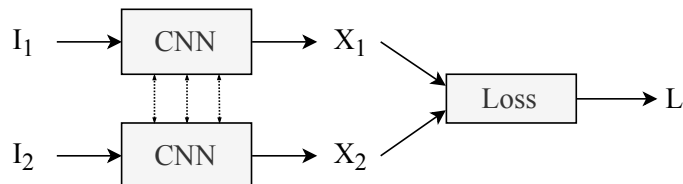


Figure 5.8: Pair siamese architecture

### 5.2.1.1 Parameters

In this section, each of the parameters will be analyzed to determine which is the best global configuration. The selected parameters will be locked for the rest of the experiments in this chapter.

#### 5.2.1.1.1 Optimizer

Optimizers are functions created to change the parameters of the network in order to obtain the best possible model. During training the optimizer updates the weights of each neuron using the error given by the loss function as a guide. As this error is minimized, accuracy of the model is expected to increase. Table 5.2 shows the results of the performed experiments. Adam will be used as optimizer for the remaining experiments, as it has been slightly superior in all three cases.

Optimizer	MNIST	Fashion-MNIST	CIFAR-10
SGD	0.702	0.553	0.221
RMSprop	0.946	0.786	0.416
Adadelta	0.913	0.671	0.326
<b>Adam</b>	<b>0.952</b>	<b>0.791</b>	<b>0.417</b>
Nadam	0.950	0.790	0.405

Table 5.2: Optimizers accuracy comparison. The best optimizer is highlighted in bold.

Details and references of each optimizer are given in the following subsections.

##### 5.2.1.1.1.1 SGD

Stochastic gradient descent (SGD) is an iterative optimizer proposed in Robbins and Monro (1951). It solves the problem of redundant information in batches by selecting a random sample per iteration. A high number of iterations is required and the noise affects the results excessively.

##### 5.2.1.1.1.2 RMSprop

RMSprop is a good, fast and very popular optimizer proposed in Tieleman and Hinton (2012). It uses an adaptive algorithm that computes the learning rate of a weight according to the magnitudes of recent gradients for that weight.

### 5.2.1.1.1.3 Adadelta

Adadelta, proposed in Zeiler (2012), is an extension of Adagrad that reduces its aggressive decreasing learning rate. It accumulates the prior gradients and uses this information to adapt the learning rate.

### 5.2.1.1.1.4 Adam

Adam was proposed in Kingma and Ba (2014) and has become one of the most popular optimizers. It uses the past squared gradients to estimate the first and second moments of the gradient. This information is used to adapt the learning rate of each parameter.

### 5.2.1.1.1.5 Nadam

Nadam is presented in Dozat (2016) as a variant of Adam. Instead of using the standard momentum to adapt the learning rate, it uses Nesterov accelerated gradient (NAG) which is considered to be better for slope adaptation.

### 5.2.1.1.2 Embedding dimensionality

The number of neurons in the last layer of the shared part of the siamese network determines the number of components of the resulting embedding, i.e. the dimensionality of the space. The higher this value, the more features can be reflected, but the complexity of the training process for the model increases.

Dimensionality	MNIST	Fashion-MNIST	CIFAR-10
2	0.545	0.591	0.296
4	0.729	0.677	0.322
8	0.856	0.744	0.383
16	0.900	0.768	0.387
32	0.919	0.765	0.390
64	0.949	0.767	0.384
<b>128</b>	<b>0.953</b>	0.777	0.393
256	0.949	0.788	0.402
<b>512</b>	0.952	<b>0.791</b>	<b>0.405</b>
1024	0.950	0.789	0.399

Table 5.3: Embedding dimensionalities accuracy comparison. The best dimensionality configurations are highlighted in bold.

Table 5.3 shows how embedding dimensionality affects the results. A small number of dimensions worsens the classification of embedded spaces. From 32 dimensions the

results improve slowly until they stall in 512 dimensions. For this reason, embedded spaces of 512 dimensions will be used in the rest of the experiments.

### 5.2.1.1.3 Pairing proportion

The proportion in which an element is paired with other elements of the same class (positive pairs) and with other elements of different classes (negative pairs) is one of the most relevant parameters of the training. The pairings are made randomly but maintain the restrictions described in section 4.3.

Pairs		MNIST	Fashion-MNIST	CIFAR-10
Positive	Negative			
4	1	0.938	0.731	0.269
3	1	0.946	0.751	0.331
2	1	0.948	0.766	0.379
3	2	0.948	0.781	0.396
4	3	0.950	0.791	0.420
1	1	0.952	0.791	0.405
3	4	0.951	0.810	0.425
2	3	0.951	0.803	0.444
1	2	0.949	0.796	0.446
1	3	0.953	0.807	0.472
<b>1</b>	<b>4</b>	<b>0.954</b>	<b>0.814</b>	<b>0.475</b>

Table 5.4: Pairing proportions accuracy comparison. The best pairing proportion is highlighted in bold.

The results are better when the number of negative pairs is greater than the number of positive pairs, as shown in table 5.4. The best results are obtained when an element is paired with only 1 element of the same class and with 4 elements of different classes. This 1/4 proportion will be maintained in the rest of the experiments of the project.

### 5.2.1.1.4 Loss function

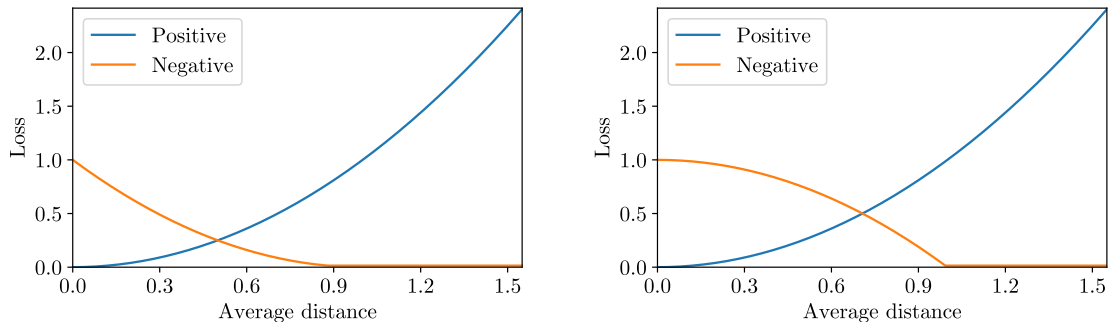
A loss function is a measure of how good a prediction model does in terms of being able to predict the expected output. If the prediction approximates the expected result the loss function will return a value close to zero. The optimizer (section 5.2.1.1.1) relies on this value to properly modify the weights of the neurons.

Figure 5.9 shows the three proven loss functions. Variable  $y$  is the expected output: 0 when the two elements belong to the same class and 1 when they belong to a different class. This variable cancels half of the equation according to its two possible values,



leaving only the loss corresponding to each pair. The variable  $D$  is the set of euclidean distances generated by the neural network, the first two functions use the average of the distances and the third uses the sum of them. Variable  $m$  is the margin from which it is considered that the distance between two elements of different classes is sufficiently large. The following section will address in depth how this parameter affects the training phase. Lastly, the variable  $N$  is the number of dimensions of the embedded space. As discussed in section 5.2.1.1.2, 512-dimensional spaces are used.

$$(a) L_a = (1 - y) \cdot D^2 + y \cdot \max(0, m - D)^2 \quad (b) L_b = (1 - y) \cdot D^2 + y \cdot \max(0, m - D^2)$$



$$(c) L_c = (1 - y) \cdot -\log\left(-\frac{D}{N} + 1\right) + y \cdot -\log\left(-\frac{N-D}{N} + 1\right)$$

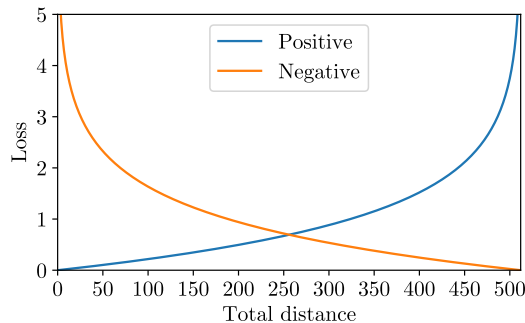


Figure 5.9: Pair loss functions

Table 5.5 shows the results of the tests with the different loss functions. The first two functions give similar results, but the first of them obtains better results in two of the three datasets, so it will be used in the rest of the siamese network experiments.

Algorithm	MNIST	Fashion-MNIST	CIFAR-10
<b>Loss function (a)</b>	0.954	<b>0.814</b>	<b>0.475</b>
<b>Loss function (b)</b>	<b>0.955</b>	0.801	0.409
Loss function (c)	0.647	0.532	0.180

Table 5.5: Loss functions accuracy comparison. The best loss functions are highlighted in bold.

### 5.2.1.1.5 Margin

The loss function chosen in the previous section has a parameter called margin that determines from what distance two elements of different classes are considered to be sufficiently spaced.

Margin	MNIST	Fashion-MNIST	CIFAR-10
0.25	0.946	0.734	0.411
0.50	0.950	0.796	0.460
<b>1.00</b>	<b>0.954</b>	<b>0.814</b>	<b>0.475</b>
2.00	0.948	0.813	0.465
4.00	0.941	0.782	0.248
8.00	0.917	0.759	0.247

Table 5.6: Margins accuracy comparison. The best margin configuration is highlighted in bold.

Tests performed with different margins are shown in table 5.6. Results worsen at both ends, both when the margin is very small and when it is very large. For the following experiments, the default margin of 1.0 will continue to be used.

## 5.2.2 Pair siamese with classification network

Several alternative architectures for pair siamese networks are presented in Bell and Bala (2015). According to the document’s own notation, architecture (A) corresponds to a traditional convolutional neural network (section 5.1) and architecture (B) corresponds to pair siamese network (section 5.2). In this section, we analyze the results of the implementation in keras of the architecture (C).

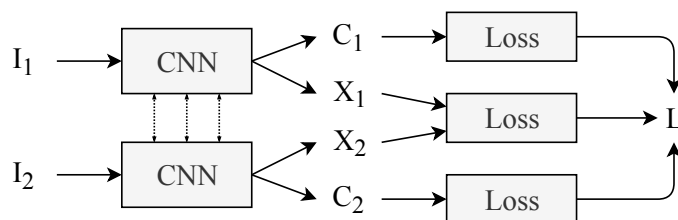


Figure 5.10: Pair siamese with classification architecture

The architecture shown in figure 5.10 has many points in common with the standard architecture, as it has two inputs that are processed to obtain the embeddings that will be used to calculate part of the loss. The main difference is that the classification of both inputs is calculated from these embedded spaces by adding *softmax* layers. The

categorical cross-entropy loss is calculated and added to the total loss as a greater or lesser percentage.

Although the inputs are similar, the expected outputs differ, since it is necessary to include information on the class to which both elements belong in addition to the similarity. The default data generator (section 4.3) did not include this information so some minors modifications are needed.

### 5.2.2.1 Parameters

All the parameters obtained empirically in section 5.2.1.1 can be used in these experiments, since this alternative architecture shares many features with the original architecture, as previously mentioned. However, due to the fact that in this architecture the total loss is calculated from different losses, a new parameter appears, which is studied below.

#### 5.2.2.1.1 Loss contribution

This architecture has three outputs with its corresponding associated loss. The first of them is the similarity between the embedded spaces of both inputs and uses one of the loss functions described in section 5.2.1.1.4. The other output corresponds to the predicted class and their loss is obtained through the categorical cross-entropy function. The contribution of each output to the total loss can be easily adjusted via the Keras interface <sup>4</sup>.

Categorical output contribution	MNIST	Fashion-MNIST	CIFAR-10
0.063	0.954	0.828	0.467
<b>0.125</b>	<b>0.962</b>	<b>0.845</b>	<b>0.507</b>
0.250	0.960	0.843	0.500
0.500	0.959	0.840	0.493
1.000	0.960	0.836	0.489
2.000	0.961	0.842	0.485
4.000	0.954	0.832	0.483

Table 5.7: Loss contribution configurations accuracy comparison.

The best loss contribution configuration is highlighted in bold.

The results of the tests with different contributions of the categorical outputs are

<sup>4</sup><https://keras.io/models/model/#compile>

shown in table 5.7. The differences between the tested configurations are very subtle. The worst results are obtained at extreme values, when the contribution is too low or too high. Since in the three datasets the best result has been obtained using a weighting of 0.125, this value will be used in the rest of the experiments.

### 5.2.3 Results

The results of classifying each dataset with pair siamese neural networks are shown in table 5.8. Both architectures slightly improve baseline performance. The alternative architecture is more effective than the original according to the experiments. In the following subsections, the progress during the training phase of this architecture will be analyzed in detail for each dataset.

Architecture	MNIST	Fashion-MNIST	CIFAR-10
Pair siamese	0.954	0.835	0.467
<b>Pair siamese with classification</b>	<b>0.962</b>	<b>0.846</b>	<b>0.505</b>

Table 5.8: Pair siamese architectures comparison. The best architecture is highlighted in bold.

#### 5.2.3.1 MNIST

Although the margin for improvement with MNIST was low because the results were very close to those obtained with the complete dataset, a model with a 4.6% better accuracy has been achieved. The progress of the training is shown in figure 5.11. There is rapid learning, especially in the first ten epochs. The training and test losses progress in parallel, which indicates that overfitting is not taking place.

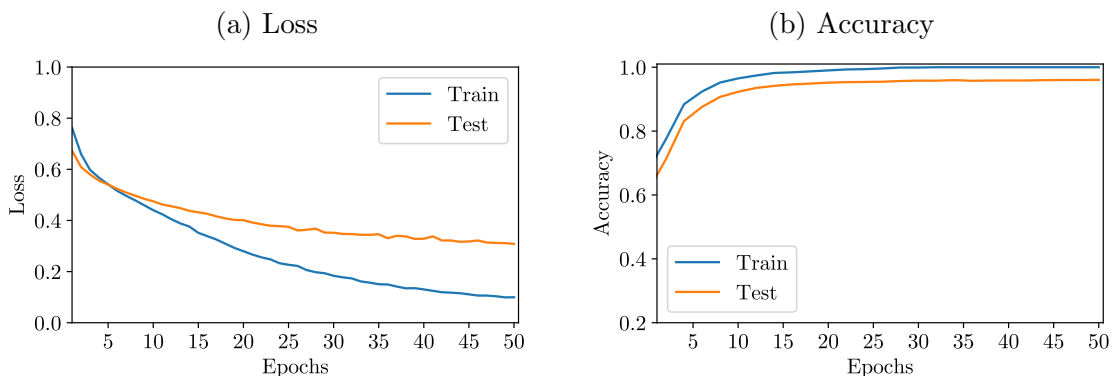


Figure 5.11: Training of MNIST using pair siamese with classification network

The embedded space generated with this model is shown in figure 5.12. It is perfectly structured since there is no overlap between classes as the separation between the areas they cover is high enough.

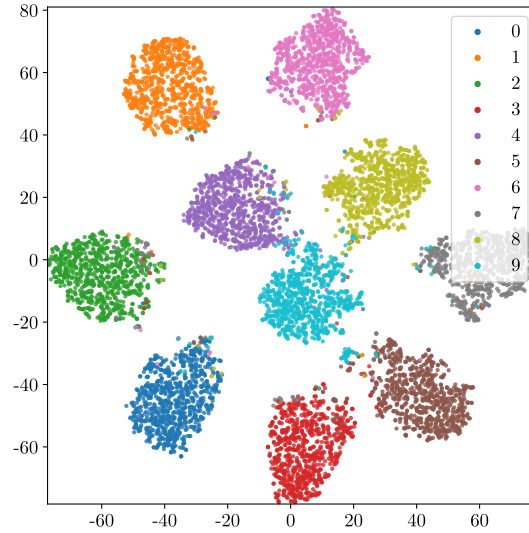


Figure 5.12: Embedded space of MNIST using pair siamese with classification network

### 5.2.3.2 Fashion-MNIST

The progress of the training is shown in figure 5.13. The results are 3.2% better than the baseline at the cost of a training that requires a slightly greater number of epochs. In the first 30 epochs, it experiences fast and constant learning, then it slows down. The difference between the loss of training and test is higher than in the baseline.

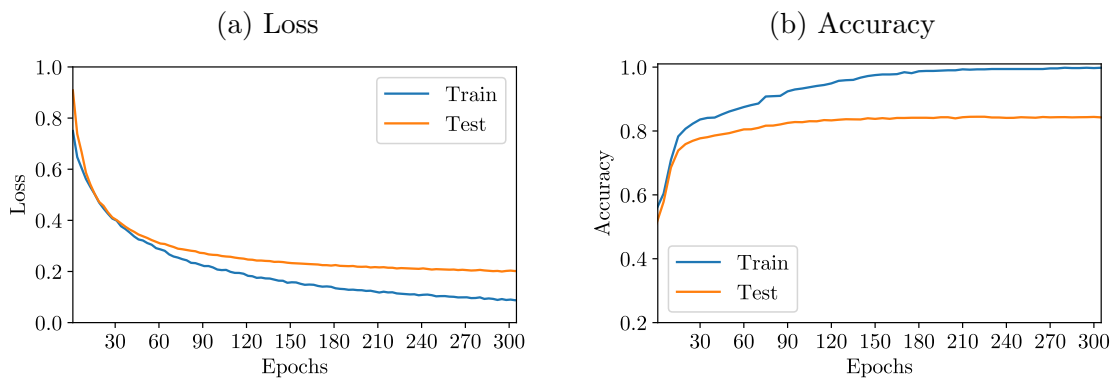


Figure 5.13: Training of Fashion-MNIST using pair siamese with classification network

The embedded space generated with this model is shown in figure 5.14 (the class associated with each label can be found in table 4.3). It is well structured, although there is some overlapping between class 6 (shirts) and classes 2 (pullovers) and 4 (coats).

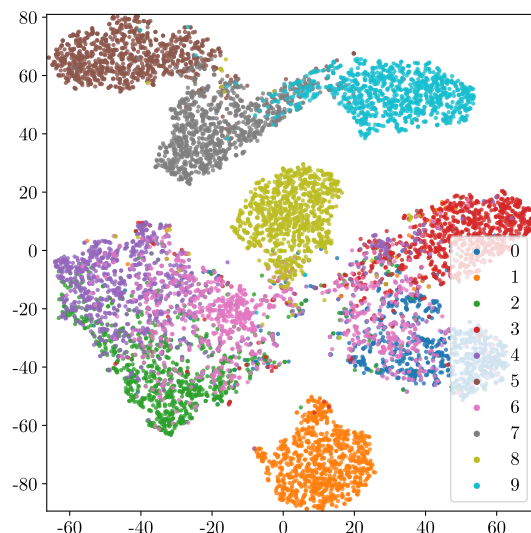


Figure 5.14: Embedded space of Fashion-MNIST using pair siamese with classification network

### 5.2.3.3 CIFAR-10

The progress of the training is shown in figure 5.15. In the first 60 epochs, slow learning takes place, after that moment only the accuracy in the training data improves. The difference between training and test losses is excessively high due to overfitting. In spite of this, the model based on the pair siamese architecture is 20,2% more precise than the baseline.

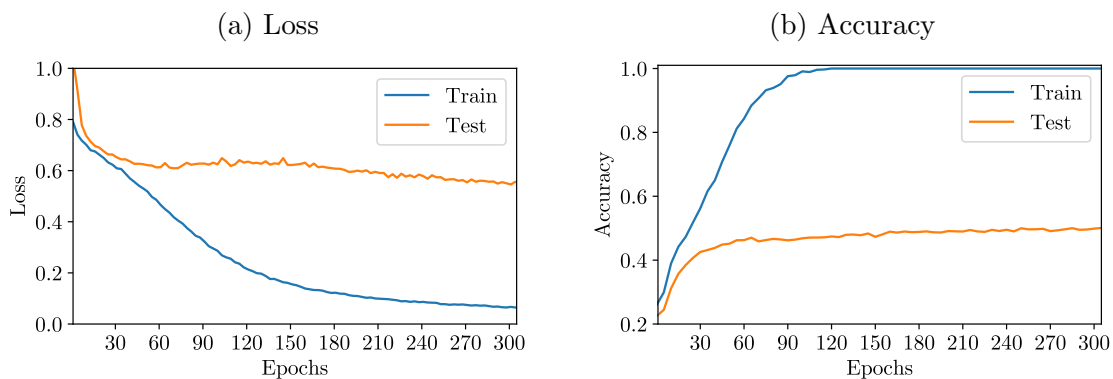


Figure 5.15: Training of CIFAR-10 using pair siamese with classification network

The embedded space generated with this model is shown in figure 5.16 (the class associated with each label can be found in table 4.5). It is completely unstructured, as classes tend to concentrate on certain areas but there is excessive overlap between most of them.

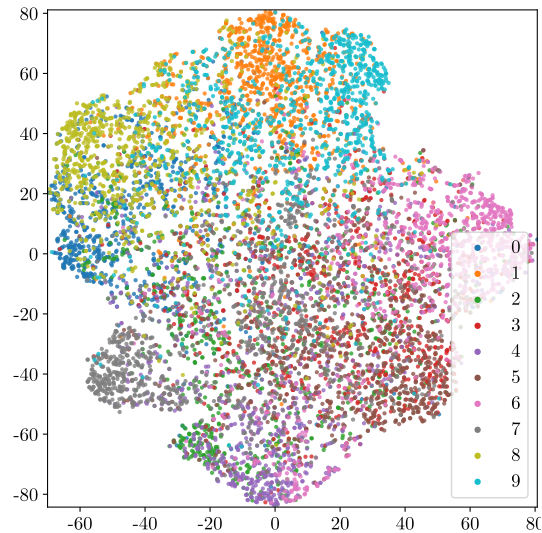


Figure 5.16: Embedded space of CIFAR-10 using pair siamese with classification network

## 5.3 Triplet siamese architecture

Triplet siamese networks are inspired by pair siamese architecture. They have three inputs that are processed by each of the two shared parts. In this section, we will analyze the results of the Keras implementation of the original model and two other alternatives. The shared part of the network for each dataset is the same as the corresponding neural network of section 5.1.1 (removing the *softmax* layer).

### 5.3.1 Triplet siamese network

The original triplet siamese architecture was proposed in Hoffer and Ailon (2015). According to the authors, this architecture is strongly inspired by pair siamese networks. As shown in figure 5.17, the model receives three inputs, two of which belong to the same class. One of these two inputs is used as an anchor and its embedded space is compared with the embedded space of the other two to compute the distances.

The objective of the training is to maximize the distance between the anchor and the element of different classes while minimizing the distance between the anchor and the element of the same class. To achieve this, the following loss function is used:

$$L_d = \max(0, D_+ - D_- + m)$$

As in section 5.2.1.1.4,  $m$  it is a margin whose value is arbitrarily chosen. The

margin obtained empirically in section 5.2.1.1.5 is used in the following experiments.  $D_+$  is the euclidean distance between the embedded spaces of the anchor and of the element of the same class. In the same way,  $D_-$  is the euclidean distance between the embedded spaces of the anchor and of the element of a different class.

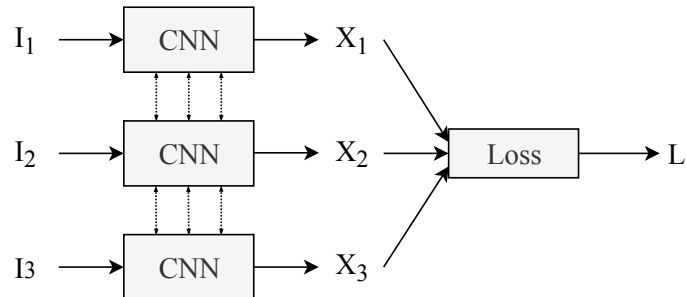


Figure 5.17: Triplet siamese architecture

### 5.3.2 Triplet siamese with classification network

This architecture arises as an adaptation of the modification made in section 5.2.2 but applied to a triplet siamese network instead of a pair siamese network. The number of inputs is the same but the class of each is predicted as well as comparing the embedded spaces. As shown in figure 5.18, the overall loss is obtained from the aggregation of several losses. The loss of class predictions is given by the categorical cross-entropy function.

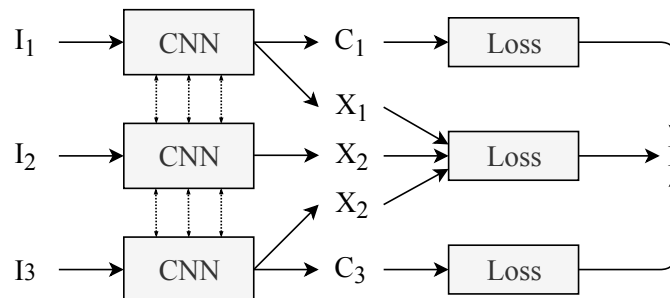


Figure 5.18: Triplet siamese with classification architecture

Keras allows assigning a weighting to each of the losses of the model. In the experiments of this architecture, the loss contribution obtained empirically in section 5.2.2.1.1 will be used. Since the default data generator (section 4.3) does not attach information about the class of each of the three images, it is necessary to make a small modification.



### 5.3.3 Triplet siamese with external memory network

This subsection presents a new triplet siamese architecture that uses external memory to store an embedded space representative of each class, as shown in figure 5.19. At the end of each epoch, this embedded space is readjusted according to the modifications of the network. These adjustments are obtained as a weighting between the stored point and the new point. This weighting initially favours the new point, allowing the representative embedded space to be adapted as the network is trained. With each epoch, this influence decreases, which forces the network to try to stick to the point stored in memory.

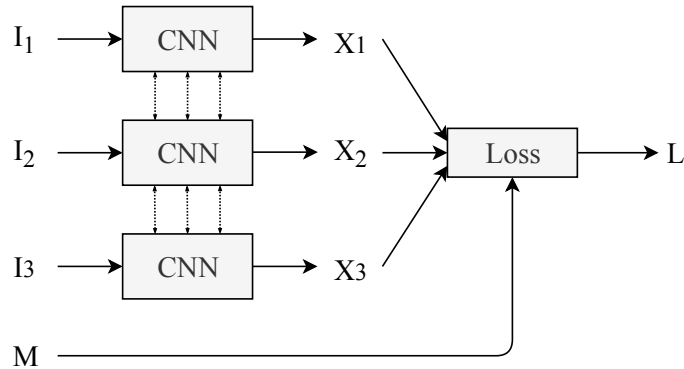


Figure 5.19: Triplet siamese with external memory architecture

### 5.3.4 Results

The results of classifying each dataset with triplet siamese neural networks are shown in table 5.9. These three architectures do not improve the results of the best pair siamese architecture despite their greater complexity. In the following subsections, the progress during the training phase of this architecture will be analyzed in detail for each dataset.

Architecture	MNIST	Fashion-MNIST	CIFAR-10
Triplet siamese	<b>0.948</b>	0.825	0.460
Triplet siamese with classification	0.935	<b>0.830</b>	<b>0.468</b>
Triplet siamese with external memory	0.936	0.827	0.467

Table 5.9: Triplet siamese architectures comparison. The best architectures are highlighted in bold.

## 5.3.4.1 MNIST

The standard triplet model obtained is 3% better than the baseline, but the pair siamese with classification model has even higher accuracy. The progress of the training is shown in figure 5.20. This architecture is trained more quickly than the rest of them, as it achieves very good results with only 6 epochs. In addition, overfitting does not take place, as training and test losses are very close throughout the whole process.

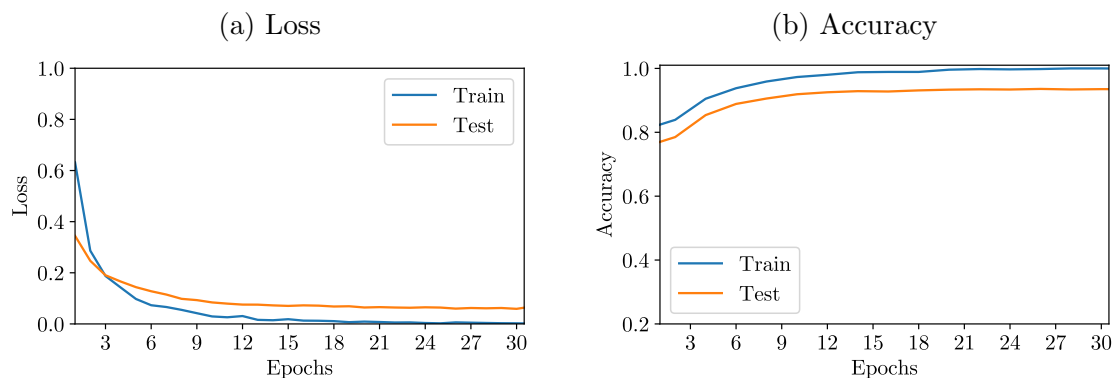


Figure 5.20: Training of MNIST using triplet siamese network

The embedded space generated with this model is shown in figure 5.21. The differences between this embedded space and the one generated by the pair siamese model are minimal. Classes are concentrated in well-separated areas and there is no overlap between them.

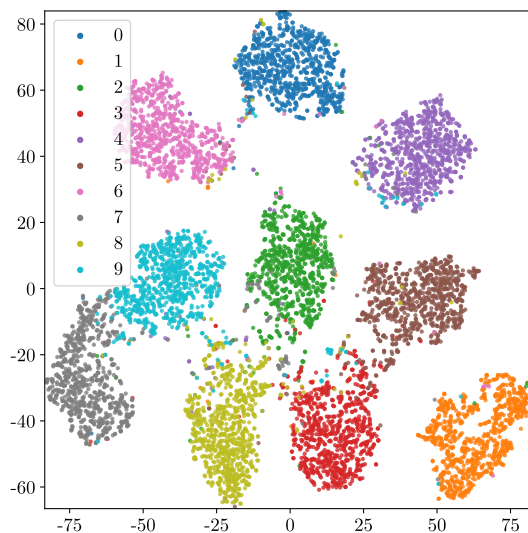


Figure 5.21: Embedded space of MNIST using triplet siamese network

### 5.3.4.2 Fashion-MNIST

The model obtained with the first alternative triplet siamese architecture is similar to the baseline and slightly worse than the pair siamese model. The progress of the training is shown in figure 5.22. The training phase is remarkably faster since the most substantial advances are made in the first 30 epochs. From that point on, there is a slight improvement until the process is finished. The difference between training and test losses is slightly lower than with pair siamese architectures.

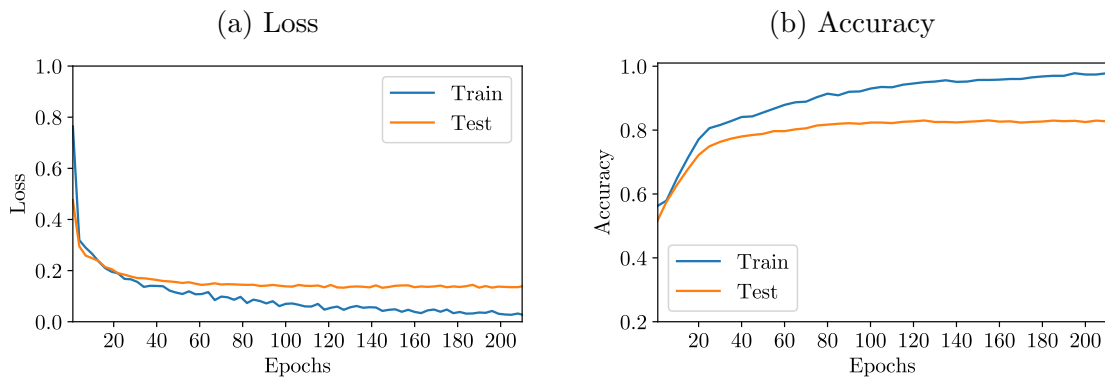


Figure 5.22: Training of Fashion-MNIST using triplet siamese with classification network

The embedded space generated with this model is shown in figure 5.23 (the class associated with each label can be found in table 4.3). It is fairly structured, with class 6 (shirts) being the most problematic due to overlap.

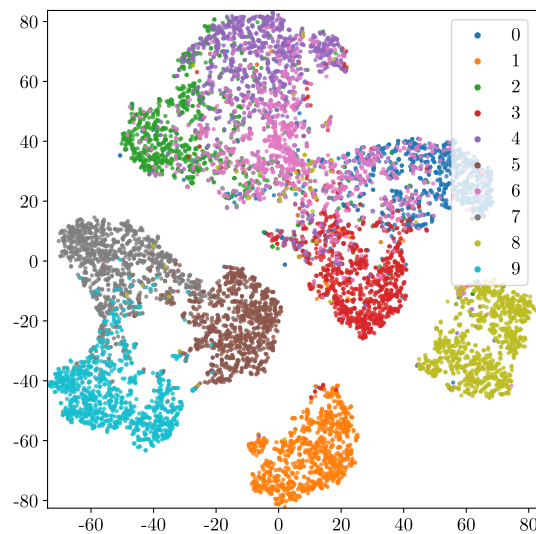


Figure 5.23: Embedded space of Fashion-MNIST using triplet siamese with classification network

### 5.3.4.3 CIFAR-10

Despite the greater complexity of the new architecture compared to the pair ones, it does not achieve a sufficiently good classification of the CIFAR-10 dataset. The progress of the training is shown in figure 5.24. Overfitting is even more evident in this model, as the difference between training and test losses is noticeably greater.

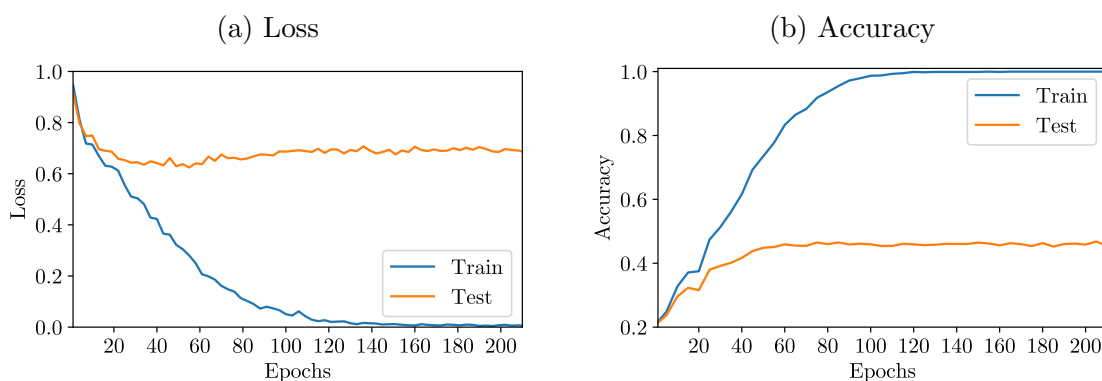


Figure 5.24: Training of CIFAR-10 using triplet siamese with classification network

The embedded space generated with this model is shown in figure 5.25. Some classes tend to concentrate on certain areas, however, samples are still excessively dispersed and overlapping takes place between most of them.

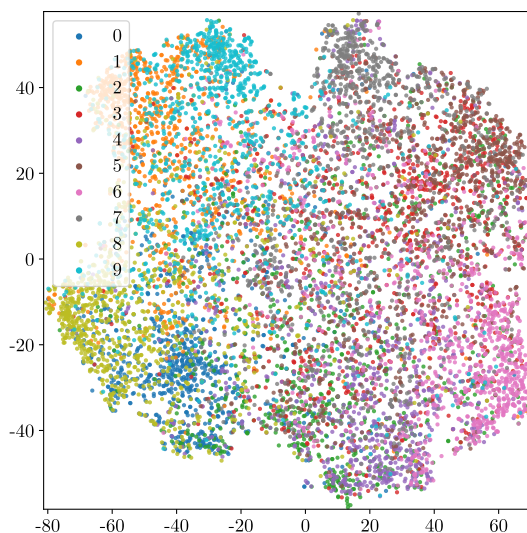


Figure 5.25: Embedded space of CIFAR-10 using triplet siamese with classification network

## 5.4 Embedded spaces classification algorithm

As explained in section 4.4, siamese networks output whether two elements belong to the same class or not. To obtain the classification of a single element it is necessary to employ techniques that use the embedded space generated during training in the shared part of the network. The following sub-sections present the most commonly used algorithm followed by other proposed techniques.

### 5.4.1 Average class distance

Average class distance technique is based on computing the euclidean distance between the embedded space of the element to be classified and all the labelled embedded spaces. A histogram is generated with the mean distances to each of the elements of the class. The class whose average distance is lower is selected as the output class.

### 5.4.2 Support Vector Regression

Support vector regression algorithms<sup>5</sup>, based in Cortes and Vapnik (1995), create non-probabilistic binary liner classifiers. They use kernel functions to map the input into high-dimensional feature spaces in order to find an optimal boundary between the possible outputs. Three of the most common kernels have been tested: linear, polynomial and radial (RBF).

### 5.4.3 $K$ -nearest neighbors

$K$ -nearest neighbors<sup>6</sup> is a non-parametric simple method introduced in Altman (1992). The algorithm is based on obtaining the  $K$  elements whose embedded space is closer to the embedded space of the element to be classified. Each of these elements votes the class to which it belongs. The class that receives the most votes is used as an output.

---

<sup>5</sup><https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR>

<sup>6</sup><https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier>

### 5.4.4 Random decision forest

Random decision forest<sup>7</sup> is a flexible learning technique for classification proposed in Ho (1995). The method operates by constructing a multitude of decision trees in the training phase. When growing the trees, instead of searching for the most important feature, it searches for the best feature among a random subset of them. The output is estimated as the average prediction of the individual trees. This algorithm has been widely used lately due to its good accuracy in classification and regression tasks with high dimensionality data sets.

### 5.4.5 Results

The results of the experiments with the different classification algorithms are shown in table 5.10. The methods that allow varying some parameter appear with the different tested configurations. The models with the highest accuracy (pair siamese with classification, section 5.2.2) have been used to generate the embedded spaces.

Algorithm	MNIST	Fashion-MNIST	CIFAR-10
<b>Average class distance</b>	<b>0.962</b>	<b>0.846</b>	0.505
Support vector regression (kernel: linear)	0.795	0.780	0.101
Support vector regression (kernel: polynomial)	0.058	0.069	0.001
Support vector regression (kernel: radial)	0.405	0.399	0.007
1-nearest neighbors	0.961	0.841	0.508
5-nearest neighbors	0.962	0.844	0.516
<b>15-nearest neighbors</b>	0.960	0.845	<b>0.518</b>
25-nearest neighbors	0.960	0.845	0.517
Random decision forest (Estimators: 25)	0.940	0.842	0.483
Random decision forest (Estimators: 50)	0.948	0.844	0.480
Random decision forest (Estimators: 75)	0.954	0.845	0.497
<b>Random decision forest (Estimators: 100)</b>	0.956	<b>0.846</b>	0.497

Table 5.10: Classification algorithms accuracy comparison. The best algorithms are highlighted in bold.

<sup>7</sup><https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier>

The average class distance algorithm, which is the method commonly used, works well for all three datasets as expected. Strategies based on support vector regression give a subpar performance. Of the three kernels tested, the only one that comes close to the accuracy of the average class distance algorithm is the linear one. Random decision forest gives good results, especially with MNIST and Fashion-MNIST. In the case of CIFAR-10, the most complex dataset, the  $K$ -nearest neighbors algorithm gives better results than other techniques, particularly when the value of  $K$  is large.





# Chapter 6

## HISPAMUS dataset experimentation

This chapter shows the results of the experiments on the dataset used in the HISPAMUS (Iñesta et al., 2018) project. In order to follow a few-shot approach, only samples of the 5 least frequent classes will be used as explained in section 4.1.4. Because the number of available data is excessively low, the leave-one-out cross-validation strategy will be used.

The most accurate architecture in previous experiments (pair siamese with classification, section 5.2.2) will be tested in this chapter. It will be configured according to the values of the parameters obtained empirically in sections 5.2.1.1 and 5.2.2.1. The model used to classify the CIFAR-10 dataset (section 5.1.1.3) will be used as a shared part of the siamese network.

Since the number of elements is so low, instead of using a random subset of the possible pairs, as in previous experiments, all possible combinations will be used. In addition, it is necessary to include information of the class of each of the elements of the pair due to the chosen architecture. Since this dataset is not natively supported in Keras, a procedure that reads images from memory and preprocesses them needs to be implemented.

The  $K$ -nearest neighbors technique will be used for the classification of embedded spaces, as it is more appropriate than the average class distance when the data is so highly unbalanced and there are classes with such a low number of elements. Another advantage of using this algorithm is that when an error occurs it is possible to identify

which image is closest to the element being classified. The value of  $K$  can only be 1 since class 4 (Quadruple whole stem) has only two elements.

## 6.1 Results

When leave-one-out cross-validation is applied, errors occur in 6 of the 39 (15.4%) images. Table 6.1 shows these errors indicating the original image and the closest image that causes the classification failure. No error occurs with elements of classes 2 (longa) and 3 (double whole). Classes 0 (double whole stem), 1 (triple whole stem) and 4 (quadruple whole stem) have an error rate of 11.8%, 18.2% and 100% respectively.









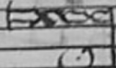



Image	Real class	Closest image	Predicted class
	0 (double whole stem)		3 (double whole)
	0 (double whole stem)		1 (triple whole stem)
	1 (triple whole stem)		0 (double whole stem)
	1 (triple whole stem)		0 (double whole stem)
	4 (quadruple whole stem)		1 (triple whole stem)
	4 (quadruple whole stem)		1 (triple whole stem)

Table 6.1: Errors of HISPAMUS leave-one-out cross-validation

The embedded space generated with the pair siamese with classification model in the first of the 39 iterations is shown in figure 6.1. It is fairly structured with well-differentiated areas for each of the classes. Although there is no overlap, the distance between elements of the same class is quite high.

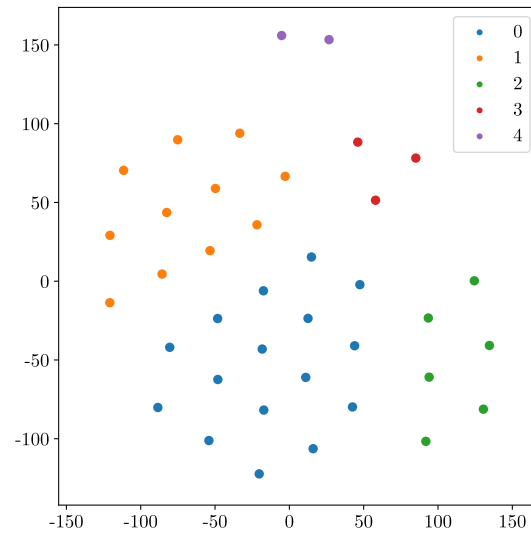


Figure 6.1: HISPAMUS pair siamese with classification embedded space



# Chapter 7

## Conclusions and future work

A detailed analysis of siamese neural networks applied to few-shot problems has been presented in this bachelor's degree thesis. This work can serve data scientists who face a problem where the amount of data available is very limited or classes are highly unbalanced. The experiments carried out prove the scope of this technology in terms of precision.

Firstly, a baseline was established using the most commonly used models for solving image classification problems, convolutional neural networks. Once the siamese architectures to be tested were selected, each one of the models was implemented, as well as the corresponding pairing mechanisms. Even with default parameters, siamese networks succeed in improving traditional approaches in all scenarios. Modifying these parameters achieves more refined configurations that return better results. The conclusions obtained for each of the parameters are presented below:

- **Optimizer:** Less sophisticated optimizers negatively affect the training. However, the difference between the most commonly used optimizers (RMSprop, Adam and Nadam) is minimal.
- **Embedding dimensionality:** Above 64 dimensions the exact value of this parameter has a trivial influence.
- **Pairing proportion:** It is one of the most influential parameters in the network. The higher the proportion of negative pairs, the better the results.
- **Loss function:** The traditional loss function of these networks continues to be the most effective.

- **Margin:** Slight modifications to the default value have very little impact but major changes worsen the results.
- **Loss contribution:** When the loss is obtained as the sum of several sources it is better to maintain a high influence of the similarity between pairs losses.

When the dataset has a low or medium difficulty the reduction of elements available for training has a smaller effect. On the other hand, when the dataset is complex, such as CIFAR-10, reducing the number of samples has a critical impact on the overall performance of the system. In both cases, the use of siamese networks over convolutional networks has been proven to be beneficial in mitigating the effects of using limited data at the cost of a slight increase in the complexity of the architecture.

The algorithm most commonly used for classification of embedded spaces gives very good results but it has been proven that other non-conventional techniques, such as random decision forests or  $K$ -nearest neighbors, also perform satisfactorily while reducing compute time.

Due to the time constraints of this project, some possible improvements and ideas were left out. Only 4 datasets with a relatively low number of classes have been experimented with. It would be interesting to extend the tests to datasets with a large number of classes. Although the proportion of pairings has been studied, they were always created randomly. There is the possibility of creating pairs following different strategies based on the current distance between the embedded spaces of the elements. Aside from image classification, another possible line of work would be to use this type of network in tasks such as language processing.

# References

- Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.
- Assunção, F., Lourenço, N., Machado, P., and Ribeiro, B. (2018). Denser: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines*, pages 1–31.
- Bell, S. and Bala, K. (2015). Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 34(4):98.
- Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. (2016). Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer.
- Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2017). Freezeout: Accelerate training by progressively freezing layers. *arXiv preprint arXiv:1706.04983*.
- Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a” siamese” time delay neural network. In *Advances in neural information processing systems*, pages 737–744.
- Calvo-Zaragoza, J., Rizo, D., and Quereda, J. M. I. (2016). Two (note) heads are better than one: Pen-based multimodal interaction with music scores. In *ISMIR*, pages 509–514.
- Chang, J.-R. and Chen, Y.-S. (2015). Batch-normalized maxout network in network. *arXiv preprint arXiv:1511.02583*.
- Chopra, S., Hadsell, R., LeCun, Y., et al. (2005). Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546.

- Cireřan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.
- Dozat, T. (2016). Incorporating nesterov momentum into adam. *OpenReview*.
- Gordo, A., Almazan, J., Revaud, J., and Larlus, D. (2017). End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124(2):237–254.
- Graham, B. (2014). Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*.
- Graham, B. (2015). Fractional max-pooling. *arXiv preprint arXiv:1412.6071*.
- Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.
- Hoffer, E. and Ailon, N. (2015). Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer.
- Iñesta, J. M., de León, P. J. P., Rizo, D., Oncina, J., Micó, L., Rico, J. R., Pérez-Sancho, C., and Pertusa, A. (2018). Hispamus: Handwritten spanish music heritage preservation by automatic transcription. *OpenReview*.
- Karpathy, A. (2011). Lessons learned from manually classifying cifar-10. *Published online at <http://karpathy.github.io/2011/04/27/manually-classifying-cifar10>*.
- Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2.
- Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.



- 
- Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.
- Kumar, B., Carneiro, G., Reid, I., et al. (2016). Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5385–5394.
- LeCun, Y. (1998). The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>.
- Lee, C.-Y., Gallagher, P. W., and Tu, Z. (2016). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, pages 464–472.
- Luo, W., Schwing, A. G., and Urtasun, R. (2016). Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703.
- Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.
- Mishkin, D. and Matas, J. (2015). All you need is a good init. *arXiv preprint arXiv:1511.06422*.
- Rizo Valero, D., Pascual Sánchez, B., Iñesta Quereda, J., Ezquerro Esteban, A., and González Marín, L. (2018). Towards the digital encoding of hispanic white mensural notation. *Anuario Musical*, 0(72):293–304.
- Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.
- Sato, I., Nishimura, H., and Yokoi, K. (2015). Apac: Augmented pattern classification with neural networks. *arXiv preprint arXiv:1505.03229*.
- Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

- Simard, P., LeCun, Y., and Denker, J. S. (1993). Efficient pattern recognition using a new transformation distance. In *Advances in neural information processing systems*, pages 50–58.
- Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., and Moreno-Noguer, F. (2015). Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126.
- Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.
- Sun, Y., Wang, X., and Tang, X. (2014). Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1891–1898.
- Tao, R., Gavves, E., and Smeulders, A. W. (2016). Siamese instance search for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1420–1429.
- Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.
- Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656.
- Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066.
- Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.
- Ye, M. and Guo, Y. (2018). Deep triplet ranking networks for one-shot recognition. *arXiv preprint arXiv:1804.07275*.

- Zagoruyko, S. and Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361.
- Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.
- Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2017). Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*.