Papyrus matching using machine learning

Pierre-Loup Nicolas

Supervisors: Raphaël Marée, Pierre Geurts

Thesis submitted for the degree of
MSc in Data Science

University of Liège
Faculty of Applied Science

Academic year 2019 — 2020

Abstract

Papyrus matching using machine learning

Thesis submitted for the degree of MSc in Data Science

Author: Pierre-Loup Nicolas
Supervisors: Raphaël Marée, Pierre Geurts

Academic year 2019 — 2020

For papyrologists, the task of reconstructing papyri from scattered fragments is a tedious and time-consuming, yet crucial task to uncover new texts that could provide meaningful information on a variety of subjects. With the advent of digital humanities, several tools have been developed to make papyrological studies more effective, in particular when it comes to the digitization and manipulation of papyri. But none of these tools can help papyrologists for the decision-making involved in the reconstruction of papyri.

In recent years, it has been shown that deep learning techniques can be successfully applied to a wide variety of problems, sometimes with impressive results. However, the application of such techniques to papyrology is pretty much novel.

In this thesis, deep learning techniques are applied to the task of papyrus fragment matching, to evaluate the possibilities offered by neural networks to tackle this complex task. Using a papyrus fragment dataset built from scratch using raw data from the papyrus collection of the Museo Egizio of Turin, several siamese neural network architectures are evaluated and compared. The best performing model is also evaluated on more a realistic use case, and gives encouraging results that could lead to practical use. Several improvements are proposed to further improve the results.

The implementation of the networks, as well as the instruction to retrieve the dataset and perform the experiments, can be found on GitHub: `https://github.com/plnicolas/master-thesis`

Contents

Chapter 1

Introduction

For many centuries, papyri have been the most common form of writing material in the ancient civilizations of Egypt, Greece and Rome. As a result, and because the dry climate of Egypt is optimal for their conservation, an extraordinary number of papyri have been found since the advent of archaeological studies in the nineteenth century. While this amount of data can be seen as a blessing for scholars studying these ancient civilizations, it is also a real challenge given the time and efforts needed to decipher, categorize and study such a large amount of documents.
More importantly, many papyri from egyptological collections throughout the world were retrieved in small pieces, or simply break down when unfolded after being stored rolled-up for more than two thousand years. The problematic of reconstructing those papyri from their fragments thus arose. It is a particularly difficult puzzle-solving task where some pieces were lost to time and the fragment boundaries are far from perfectly matching.

In recent years, the development of digital humanities provided scholars with a wide range of tools and techniques to digitize and analyse documents of all types. Papyrology in particular has been in the vanguard of the application of information technologies to humanities [1], and now benefits from advanced tools to make papyrological studies more effective. However, these tools are mainly directed at the digitization and manipulation of papyri, which is useful for the specialists and certainly increases their productivity, but does not *help* them per se.

Recent developments in the literature have shown that machine learning (and in particular deep learning) techniques can be successfully applied to a wide range of problems, sometimes with impressive results. However, the application of such techniques to papyrology is pretty much novel.

1.1 Context - Crossing Boundaries Project

The 'Crossing Boundaries' Project[1] is a collaboration between the Museo Egizio of Turin, the University of Basel (Switzerland), and the University of Liège (Belgium). It proposes an interdisciplinary approach to the written material produced by the ancient Egyptian community of Deir el-Medina [2]. This community consisted of the workmen who built the royal tombs in the Valley of the Kings during the New Kingdom period (c. 1350–1000 BCE), as well as their families, and

[1] http://web.philo.ulg.ac.be/x-bound/

produced an unparalleled quantity of texts and inscriptions.
The quantity of written material coming from Deir el-Medina can be explained by two main factors: first, the high level of literacy of the members of the community; second, the exceptional conditions for the preservation of the material itself. The village is located in foothills, protected from the Nile floods, and was abandoned during the reign of Ramesses XI (c. 1100 BCE), when the community moved away from the village.

The archives of the Museo Egizio contain more than 10,000 papyrus fragments, which are currently being digitized and curated as part of the TPOP (Turin Papyrus Online Platform) project [3]. Around 300 of these have already been reassembled into larger manuscripts by egyptologists. Some of these reassembled documents are more or less complete manuscripts, though some of their texts are partly unidentified. In addition to these, the museum holds thousands of (very) small, undocumented fragments, which may belong either to these ensembles or to others.
The project focuses on a particular category of documents from Deir el-Medina: the so-called 'heterogeneous' papyri. This label refers to a category of papyri combining texts and drawings of different types in a single manuscript. For instance, a single papyrus containing a copy of a letter to the authorities, a note about a judicial case and an hymn to a pharaoh.

Figure (1.1) Example of papyrus fragments from the Deir el-Medina corpus.

Many aspects of the scribal culture of ancient Egypt are still poorly understood, and most research done in this field focuses on the content of the texts itself, with the objective of reconstructing literary compositions, explaining historical events, or describing the administrative customs of ancient Egyptians. The Crossing Boundaries Project, on the other hand, adopts a contextualised and interdisciplinary approach to the written materials from Deir el-Medina, in order to get a grip on the scribal practices of the individual scribes who produced these papyri. The project aims at crossing the epistemological and methodological boundaries between traditional disciplines such as archaeology, papyrology, palaeography and text analysis, to study the heterogeneous papyri described above. Until now, these papyri have been studied individually, as textual sources for other thematics, and not as a coherant whole. But these papyri are particularly important for the synchronic and diachronic study of ancient scribes and their work, which is why the crossing boundaries project was created.

The Crossing Boundaries Project is expected to have a major impact on several levels: it will lead to the reconstruction of a sizable quantity of ancient Egyptian papyri, currently fragmented in collection of the Museo Egizio of Turin; new learning algorithms and a dedicated interface will be developed to piece together fragments of papyri, and this framework will be applicable to many other papyri collections; finally, the project will gain significant new insights into Egyptian scribal culture.

1.2 Goal of the thesis

The process of reassembling fragments into larger manuscripts is very time consuming and far from trivial; it takes a lot of efforts from experts to solve these "jigsaw puzzles", especially with so many small fragments to consider. But reconstituting papyri is of utmost importance to study them properly, therefore any effort to ease this process is welcomed.

The goal of the present thesis is to use machine learning techniques such as deep neural networks to help egyptologists in the task of papyrus reconstitution, by developing models to detect potentially linked fragments (i.e. originating from the same papyrus) and ease the reconstitution progress. These models could be used by the papyrologists to query a fragment database, returning the fragments considered closest to the one used for the query.
This piece of work will mainly revolve around the use of siamese neural networks using convolutional architectures (CNNs) to compare papyrus fragments and tell whether they come from the same original papyrus or not.

Chapter 2

Methods

In this chapter, the neural network architectures that will be used in the experiments are presented. A brief introduction to deep learning, and in particular convolutional and siamese neural networks (which are at the core of the architectures used in this work), is provided as a minimal theoretical background to understand the concepts used in the rest of the chapter. Then, each network architecture is detailed, and finally the concept of transfert learning, which will intervene in one of the experiments, is presented.

2.1 Introduction to deep learning

2.1.1 Artificial neural networks

Artificial neural networks are mathematical models vaguely inspired by biological neural networks. Their purpose is to learn to perform various tasks solely by considering examples, without task-specific programming.

In practice, they consist in a collection of neurons (also called *perceptrons*, introduced by Rosenblatt in 1957 [4]) arranged in layers. Each neuron is a simple mathematical function that combines its input vector $\mathbf{x} \in \mathbb{R}^m$ with a set $\mathbf{w} \in \mathbb{R}^m$ of weights and its bias b, in the form of a weighted sum $\sum_{i=0}^{m} w_i x_i + b$ to which an activation function σ is applied, for instance the rectified linear unit $ReLU(x) = max(0, x)$.
The general perceptron formula is thus given by

$$y = \sigma(\sum_{i=0}^{m} w_i x_i + b) = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

Figure 2.1 illustrates the general form of a perceptron (here w_0 is the bias b).

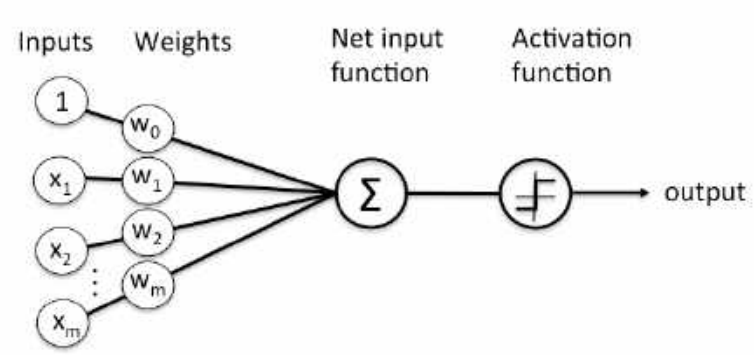

Figure (2.1) Perceptron.

A layer is composed of several neurons put in parallel, and produces q outputs by generalizing the perceptron formula:

$$\mathbf{h}(\mathbf{x}) = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b})$$

where $\mathbf{h} \in \mathbb{R}^q$ is the layer's outputs, $\mathbf{W} \in \mathbb{R}^{m\times q}$ is the weight matrix of the layer and $\mathbf{b} \in \mathbb{R}^q$ is the bias vector. Here, σ is an element-wise activation function.

Such layers can be composed in series, forming a multilayer perceptron (MLP - figure 2.2), which is the basic neural network architecture. It is also called a *fully-connected network*, because in most MLPs each neuron in one layer is connected to all neurons in the next layer.

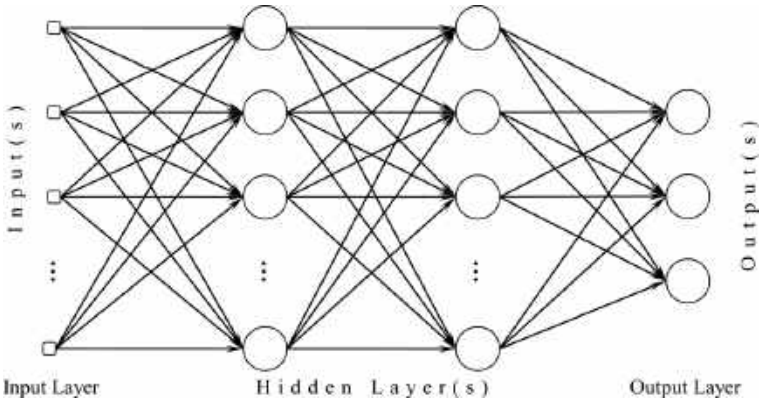

Figure (2.2) Example of multilayer perceptron, here with two hidden layers.

In essence, an artificial neural network is thus a very large mathematical function, whose goal is to best predict the output associated to a given input.
To determine how well a neural networks fits the data (i.e. how good its predictions are), a loss function L is used. Training a neural networks amounts to minimizing this loss function, which is done using training data (examples) and the backpropagation algorithm which will not be detailed here.

Because the perceptron model can work with any input vector $\mathbf{x} \in \mathbb{R}^m$ of real numbers, it can be applied to any task where the data can be represented in numerical form (e.g. images, sounds,...). However, fully-connected networks can be difficult to train due to their number of parameters

growing extremely large as their size, and most importantly the input size, increase.
Furthermore, MLPs do not take into account the spatial structure of data: in the case of images, input pixels which are far apart will be treated in the same way as pixels that are close together. This is completely ill-suited for tasks such as image recognition. For that reason, images often used with a distinct class of neural networks called *convolutional neural networks*.

2.1.2 Convolutional neural networks

Convolutional neural networks (CNN), first introduced in 1989 by LeCun et al. [5], are artificial neural networks specifically designed for visual data, that capture spatial information by using a succession of convolutional and pooling layers.

In general, a convolution takes as input a 3D tensor $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$ (for instance a RGB image of size $H \times W$, in which case $C = 3$), called the input feature map, and a kernel $\mathbf{u} \in \mathbb{R}^{C \times h \times w}$. The kernel slides across the input feature map, along its height and width, resulting in the 2D *output feature map* $\mathbf{o} \in \mathbb{R}^{(H-h+1) \times (W-w+1)}$ such that

$$\mathbf{o}_{ij} = \mathbf{b}_{i,j} + \sum_{c=0}^{C-1} (\mathbf{x}_c \star \mathbf{u}_c)[j, i]$$

where $\star$ denotes the cross-correlation operator, and $\mathbf{b}$ and $\mathbf{u}$ are shared parameters to learn, using backpropagation as usual. D convolutions can be applied successively to obtain a $D \times (H - h + 1) \times (W - w + 1)$ output feature map.

Pooling layers are often placed after convolutional layers to progressively reduce the original input size, which decreases the number of parameters in the subsequent layers of the network. Max-pooling over an area of size $h \times w$ consists in taking the maximum element in the area, while average pooling takes the mean over all elements in the area.

2.1.3 Siamese Networks

A Siamese neural network (sometimes called twin neural network) is an artificial neural network containing two or more identical subnetwork components that share their weights (hence the name "siamese"). This kind of neural network, first introduced by Bromley et al. in 1994 [6] to tackle the task of automatic signature verification, is specifically designed for tasks involving (dis)similarity.

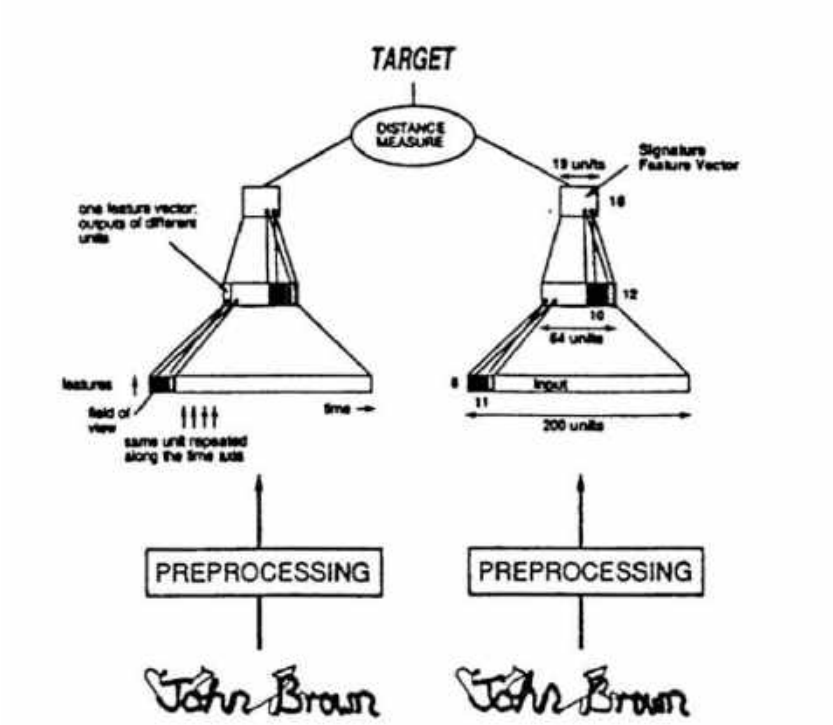

Figure (2.3) First siamese neural network architecture, proposed by Bromley et al. in 1994.

A typical siamese networks consists in two identical branches (the *twin networks*), each taking an input, that are later combined to compute a *similarity measure* (e.g. a distance metric such as the Euclidean distance) quantifying the similarity between the two branches' end vectors. This similarity measure can be directly outputted, or further used by extending the network (e.g. to perform classification).
It is important that not only the architecture of the twin networks is identical, but the weights have to be shared among them as well. The weight-sharing guarantees that two very similar inputs can not be mapped by their respective twin to very different points in the feature space, because each twin computes the exact same function.

Technically speaking, the twin networks can consist in any kind of neural network architecture, adapted to any kind of input data (fully-connected for numerical data, convolutional for images, LSTM for sequential data,...), though convolutional networks are often preferred, due to the fact that most applications of siamese networks involve image similarity. The twin networks' purpose is essentially to transform the input data they receive into a feature vector that be conveniently used to compute a similarity measure between the two inputs of the siamese network.

In the context of the present thesis, where the inputs will be images, siamese networks using convolutional neural networks (CNNs) are the architecture of choice. A simplified representation is given in figure 2.4.

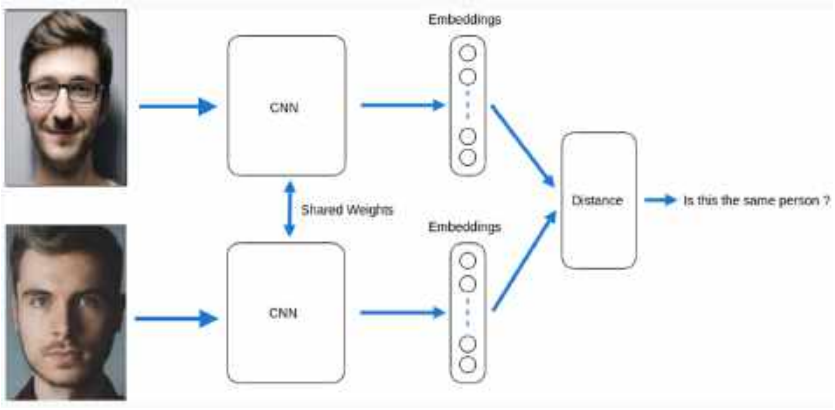

Figure (2.4) Simplified representation of a siamese neural network using a CNN.

In such an architecture, the convolutional part of the siamese network is used to get embeddings (called *output feature maps*) of the input images that capture fine-grained information about them. The embeddings are then used to compute a similarity measure. A simple and commonly used similarity measure is the Euclidean distance (also called L^2 norm).
Given two vectors $(x, y) \in \mathbb{R}^p \times \mathbb{R}^p$, the Euclidean distance between them is defined as:

$$d(x, y) = ||x - y||_2 = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

Another, more complex approach is to fuse the embeddings using, for instance, the absolute difference $|x - y|$ between them, before feeding the resulting vector to a fully-connected network. This approach results in a higher number of parameters to train, but exploits the output feature maps to a larger extent.

2.2 Architectures

Convolutional-based siamese neural networks can be broken down in two main components: the CNN architecture, which is responsible for the embedding of the input images; and the similarity measure, which determines the final output of the network. An optional third component can be used after the similarity measure, for instance to perform classification.

The choice of the CNN architecture will essentially determine the nature of the information encapsulated in the embedding of the input images; not only the depth of the network matters, but also the number and size of the convolutional kernels. These hyper-parameters define the receptive field of the network and the number of features it will extract from the images.
In the context of papyrus fragment matching, it may be interesting to investigate the impact of the CNN architecture and determine whether or not a deeper, more complex convolutional architecture is useful to the task.

For that reason, experiments will be conducted on a few convolutional architectures: first, the well-known ResNet architecture (with its ResNet50 variant - 50 layers); then, the Xception architecture; finally, the Papy-S-Net architecture (or rather, its convolutional part) proposed by Pironne et al. [9].

2.2.1 ResNet50

The ResNets (*Residual Networks*) were introduced by Kaiming He et al. [7] as a novel type of very deep architecture that can be efficiently trained using *residual learning blocks* implementing shortcut connections to mitigate the issue of vanishing gradients arising with deep networks.

Deep residual networks have been shown to be particularly good at many computer vision tasks; they are therefore interesting to consider for the present task.
The ResNet50 variant is a 50-layers deep convolutional architecture (figure 2.5) with over 23 million parameters. It achieved a 20.74% top-1 validation error (5.25% top-5) on the ImageNet classification task, and offers the advantages of very deep architectures while being reasonably fast to train.

In order to speed-up the training, as ResNet50 is a quite large network, transfer learning is used. This interesting approach consists in initializing the weights of ResNet50 with weights that were obtained by previously training it extensively on the ImageNet dataset. Using the "knowledge" obtained from the ImageNet dataset, it is possible to improve the learning process for the task of papyrus matching.

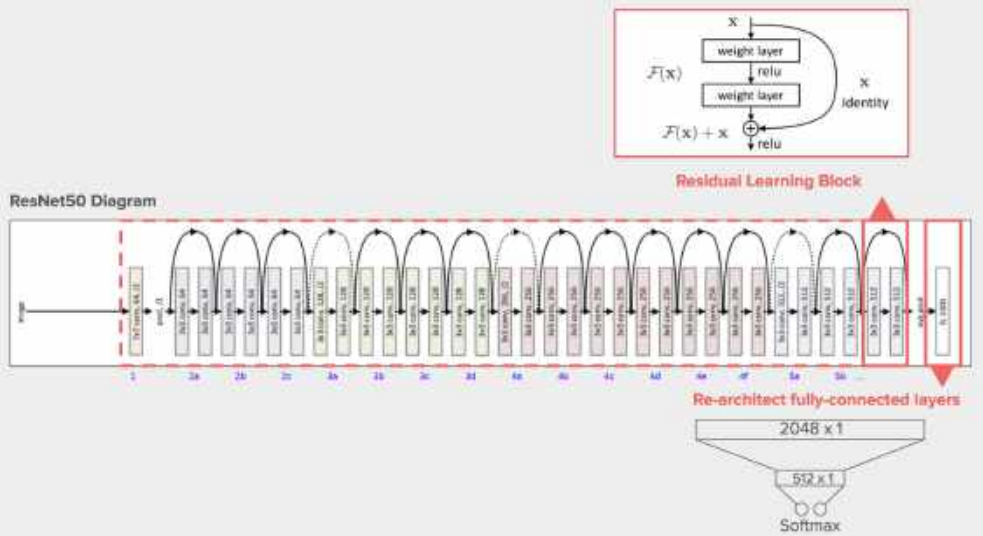

Figure (2.5) The ResNet50 architecture.

Using a very deep convolutional architecture, sure as ResNet50, allows to capture very "deep" visual information about the input images. Although in the current state of the art, researchers still fail to completely understand the information captured by CNNs (what they "see"), it is known

that deep networks learn particularly complex image patterns, that can be related to shapes, but also textures, for instance. Since papyrus scrolls are made from fibers of the papyrus plant, their texture and the structure of these fibers can be useful information to consider (and in practice, it is indeed considered by scholars). A deep convolutional network might capture this kind of information, although the resolution of the images needs to be quite high in order to clearly see the fibers.

In this thesis, a siamese architectures using ResNet50 as its convolutional part is considered, and will now be detailed.
The *ResNet50-Twin* network, inspired by the Papy-S-Net architecture (section 2.2.3), consists in a convolutional part using ResNet50 (with average pooling) to extract output feature maps from the images, followed by a merging layer that flattens the output feature maps and merge them using the absolute difference ($|x - y|$), and two fully-connected layers with 512 units each.

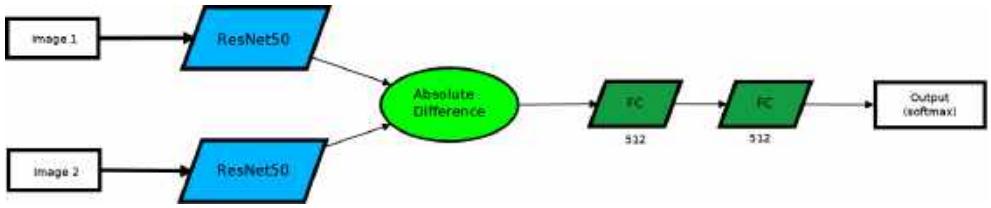

Figure (2.6) The ResNet50-Twin architecture.

2.2.2 Xception

Xception (short for *Extreme Inception*) is a deep convolutional architecture introduced by Chollet [8] in 2017 as an improvement of the InceptionV3 network, taking its principles to an extreme.

The main hypothesis of the Inception and Xception architectures is that for images, cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly. In practice, this means that while in a traditional CNNs the convolutional layers seek out correlations across both space (the pixels) and depth (the channels), in Inception and Xception space and depth are considered separately.

In the case of Xception, so-called *depthwise separable convolutions* are used; first, the spatial correlations of each channel are mapped separately, and then a 1x1 depth-wise convolution is used to capture cross-channel correlations (figure 2.7.)

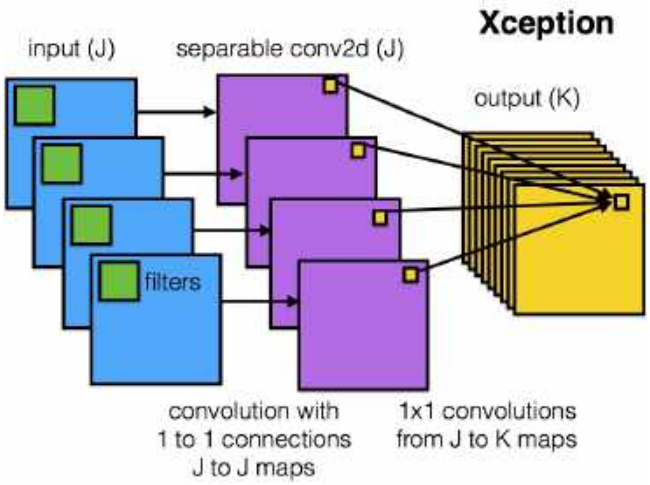

Figure (2.7) Depth-wise separable convolution module, as used in the Xception network.

Similarly to ResNet50, a siamese architecture using Xception is considered, with the same approach: The *Xception-Twin* network consists in a convolutional part using Xception (with average pooling) to extract output feature maps from the images, followed by a merging layer that flattens the output feature maps and merge them using the absolute difference ($|x - y|$), and two fully-connected layers with 512 units each.

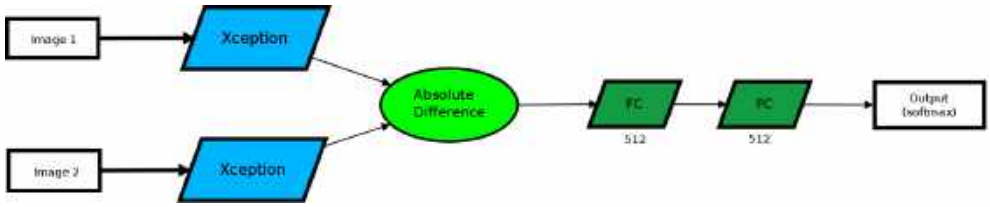

Figure (2.8) The Xception-Twin architecture.

2.2.3 Papy-S-Net

Papy-S-Net (*Papyrus-Siamese-Network*) is a siamese neural network introduced in 2019 by Pirrone et al. [9] to tackle a task which is very similar to the one addressed in the present thesis. Their paper focused on the matching of papyrus fragments coming from Egyptian mummy cartonnages, the main difference with the present work being that they considered a multilingual corpus (Greek, Latin, Coptic, Arabic and Hieratic) from the hellenistic period.

Their Papy-S-Net architecture uses a convolutional neural network consisting in three (CONV + MAXPOOLING) blocks, for a total of six convolutional layers which is considerably smaller than ResNet50. For the similarity measure, the absolute difference between the two image embeddings, followed by two fully connected layers, are used instead of the Euclidean distance. In essence, this means that Papy-S-Net focuses less on the image embeddings, and more on learning how to compute a similarity measure using those embeddings. It is quite different from usual convolutional siamese networks were the similarity measure is often a simple Euclidean distance.

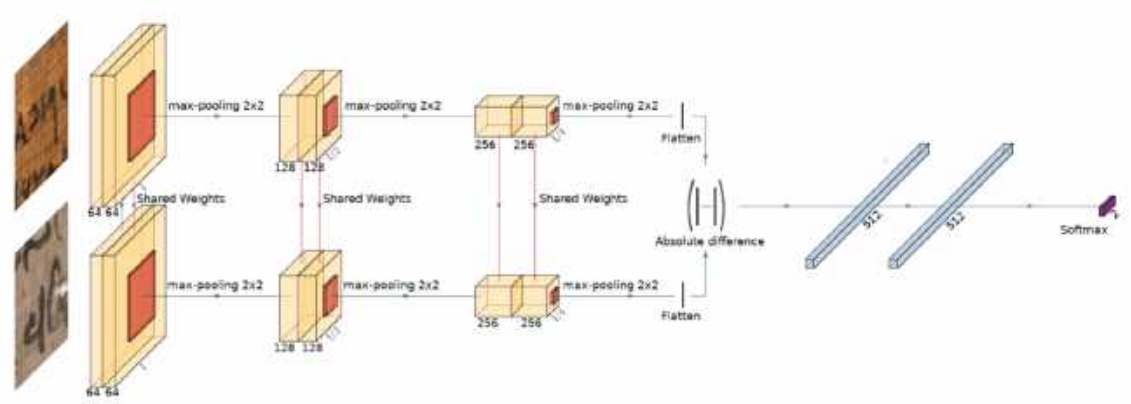

Figure (2.9) The Papy-S-Net architecture. (Pirrone et al. '19)

2.3 Transfer learning

Transfer learning [10] is a machine learning technique where a model developed for a task A is reused as the starting point for a model on a second task B, with the goal of speeding up the training process and potentially improving the results. This technique, first introduced by Lorien Pratt in 1993 [11] is particularly interesting when the amount of data for task B is lower than for task A; in that case, being able to transfer some "knowledge" from the first task to the other to compensate for the lack of data is interesting.
In the case of papyrus matching, the amount of data available is quite scarce (no large, "standard" dataset exists) and very few models have been developed to tackle this relatively niche task. On the other hand, image classification is a well studied problem with many large datasets and models (among which ResNet50 and Xception) already developed, with impressive results. Since the task of papyrus fragment matching also involves images, transfer learning could potentially be applied successfully.

ImageNet [12] is a vast image dataset consisting of more than 14 million images in 21,841 classes. For years, it has been one of the standard benchmarks to assess the performances of neural networks on the task of image classification, and many models have been developed to push back the state-of-the-art performances on ImageNet. But training neural networks on such a large dataset takes an important amount of time and computational resources. Therefore, it is difficult to train all models on it, but transfer learning can be (and is) used to still take advantage of the extensive "knowledge" that can be gained from this extensive image dataset [13].
In fact, most deep learning libraries nowadays propose several architectures (ResNet, Inception, Xception,...) that have been trained on ImageNet, with weights resulting from this training and ready to be fine-tuned on a new task.
In the experiments, the ResNet50 and Xception architectures presented in the previous sections will be used both with random weight initialization and with weights pre-trained on ImageNet, in a tentative to apply transfer learning to the task of papyrus matching.

Chapter 3

Dataset

In this chapter, the workflow utilized to build the dataset used in the present thesis is presented in details, from the raw data provided by the Crossing Boundaries project to the final dataset that is used in the experiments.
First, an overview of the raw data provided by the Crossing Boundaries team is provided, with various examples of papyri and an explanation of the limitations of the data. Then, the Cytomine platform, which is used to prepare the data, is briefly presented. Finally, the proper dataset creation process, resulting in the final dataset used in the experiments, is detailed.

3.1 Introduction: provided data & requirements

The more than 10,000 papyrus fragments in the Museo Egizio collection are not yet documented in the museum's existing database; their restoration and digitization are still a work in progress. Therefore, the quantity of data available gradually increased throughout the course of this thesis.

The papyrus fragments were directly provided by Mr. Stéphane Polis (University of Liège) and his team in the form of 400dpi scans of papyrus fragments of various sizes, along with larger, already reconstructed papyri. Figure 3.1 presents a collage of several images provided by the Crossing Boundaries team (these can be considered large-sized papyri). It is important to note that the digitization procedure can differ from one image to another (e.g. a colour reference chart may or may not be present, the background colour may differ,...), as figure 3.2 also illustrates.

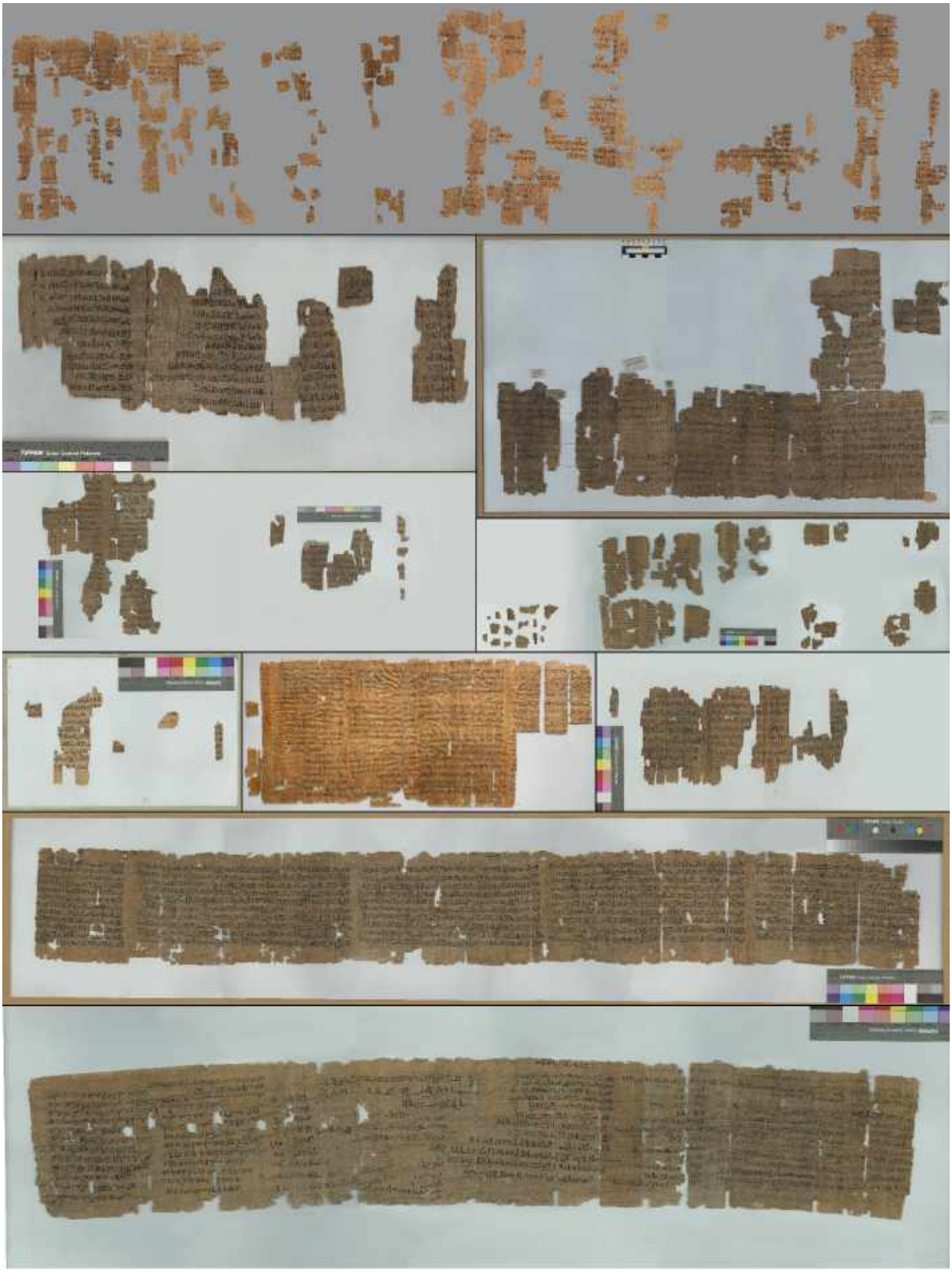

Figure (3.1) Example of images (sizes not to scale) provided by the Crossing Boundaries team.

Figure (3.2) Example of images provided by the Crossing Boundaries team. Notice the wooden frame in the second scan.

Furthermore, the papyri themselves come in all shapes, sizes, colors, and preservation state (figure 3.3). Depending on various factors, such as the storage conditions, the visual aspect of the papyri can be altered: while it can sometimes be useful to match fragments, it can also be misleading as different fragments from the same papyri may have been stored in different conditions, resulting in very different visual aspects.

Figure (3.3) Example of colour and texture differences.

Naturally, these high-resolution scans can not be used *as is* to train models. As explained in section 2.2, the models' inputs are pairs of fragment images. To create an exploitable dataset, it is therefore necessary to extract (fake) fragments from these high-resolution images to generate *ground truth*, and training data.
On that matter, unfortunately, while the amount of data provided by the team was considerable (thousands of images), most of it ended up being unusable due to the very small size of most fragments provided (figure 3.4): given their size, it is impossible to generate ground truth (i.e. 2 fake fragments) from them. In the end, only a handful of large papyri (such as those on figure 3.1) were useful to create a sufficient number of fake fragments.

In any case, for convenience, and as will be seen in the next section, because of the interesting tools provided, all the data has been stored on the *Cytomine* platform of the University of Liège, and the fake fragment creation process was done there.

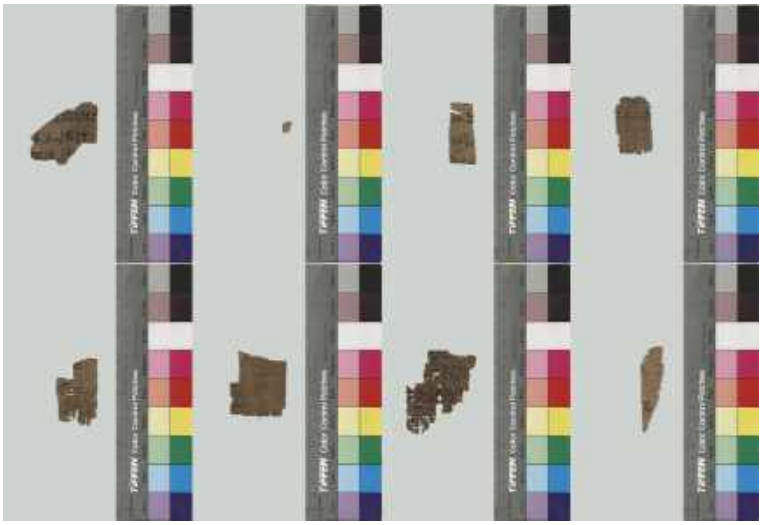

Figure (3.4) Example of unusable papyrus fragments.

3.2 Cytomine platform

The Cytomine open-source software[1], developed by Raphaël Marée et al. [16] is an application which fosters collaborative analysis of very large images and allows the semi-automatic processing of large image collections via machine learning algorithms. Although the main usage of Cytomine is medical research, the functionalities provided by the software can be used for any task involving large images.

In particular, Cytomine uses an annotation system that allows users to draw and save "regions of interest" inside the images, and associate values (text description, tags, properties,...) to them. This feature was particularly interesting in the context of the present thesis as it offered a convenient way to build a working dataset (figure 3.5).

The Cytomine web interface allows the user to manually create annotations of various shapes using different tools (freehand, circle, polygon,...), that can subsequently be retrieved in the form of images, either as plain, rectangular crops (thus discarding the original shape of the annotations if not rectangular) or as "alphamasks", i.e. using a transparency mask to preserve the original shape of the annotations (figure 3.6).

In the context of papyrus fragment matching, one could think that creating realistic shapes mirroring real fragments might be interesting to train the models; with the Cytomine software, testing such an hypothesis is quite easy (although time-consuming), and this will be investigated in the first experiment, presented in section 5.1.

3.3 Creation of the dataset

During the course of this master thesis, approximately 60 "large", already (partially) reconstructed papyri were sent by the Crossing Boundaries team, along with thousands of small fragments. For

[1] https://uliege.cytomine.org/

Figure (3.5) Example of annotations made with the Cytomine web interface.

Figure (3.6) Comparison between the same annotation extracted either as an alphamask (left) or a crop (right).

each papyrus/fragment, both the recto and the verso were provided. All the images were uploaded on the Cytomine platform of the university of Liège for storage.

From there, the annotation system of the Cytomine platform was used to manually create fake papyrus fragments using the larger papyri, in order to create ground truth.
A first batch of 1812 fake fragments, used in the first experiment, was created with realistic shapes, with the hypothesis that the shape of the fragments might be useful to train the models. However, and as will be seen in the first experiment (section 5.1), the impact is negligible. It is in fact quite easy to understand why: as explained before, for most papyri some of the fragments were lost to time, and furthermore the remaining fragments suffered from several depredations (insects,...),

meaning that the shapes of fragments belonging together will, perhaps counter-intuitively, hardly match in practice or present similar patterns.
Following this, a second batch of 2909 fake fragments was created using simple rectangular shapes, to increase the amount of data for the subsequent experiments (which only use rectangular crops).

In total, 4721 fake fragments were thus created in the form of Cytomine annotations on the large papyri's images. Using the Cytomine API, these annotations can be easily retrieved from the server as images using a simple Python script generating in parallel a CSV file listing, for each image (fake fragment), the papyrus it is coming from. This is the dataset that will be used in all of the experiments.

Since siamese neural networks take image pairs as input, the "fake fragment dataset" is not usable as-is; it is used to generate fragment pairs.
First, in order to avoid any bias between the training and test data, the fake fragments are split into two sets, according to their source papyrus: the training fragments set S_{train}, and the test fragments set S_{test}. 75% of the papyri are used for S_{train}, and the remaining 25% for S_{test}.

For both S_{train} and S_{test}, *positive pairs* (the two fragments come from the same papyrus) and *negative pairs* (the two fragments do not come from the same papyrus) are generated for each papyrus in the set, using a very simple procedure which simply consists in sampling fragments from the dataset and combining them (duplicate pairs are discarded).
This simple procedure makes it possible to generate an arbitrary large number of positive and negative pairs. Indeed, if the dataset consists in k distinct papyri, with n fake fragments each, there are $k \times \overline{C}_2^n = k \times \frac{n \times (n+1)}{2}$ distinct positive pairs, and $k \times [n \times ((k-1) \times n)]$ distinct negative pairs. Thus, assuming a relatively small dataset of $n = 50$ fake fragments per papyrus, with $k = 20$ papyri, 25500 positive and 950000 negative pairs are already possible.
However, the number k of different papyri should be as large as possible, to ensure sufficient diversity in the dataset; sampling a very large number of pairs from only a few papyri would undoubtedly lead to models that fail to generalize properly to never-seen papyri.

Chapter 4

Methodology

In this chapter, the methodology used to train, test and evaluate the models is detailed.
First, the evaluation metrics used to assess the performances of the models are defined. Then, several visualization techniques that will be used in the experiments are shown. Finally, the training and test protocol are presented, along with the methodology used for the assessment and visualization of the results.

4.1 Evaluation metrics

The evaluation metrics for the different models are the train and test accuracy, but also (and perhaps more importantly) the precision and recall, two metrics originating from the field of information retrieval that are particularly relevant in the context of the task of papyrus fragment matching.
Precision is the ratio of correctly predicted positive observations to the total predicted positive observations, and is computed[1] as

$$Precision = \frac{TP}{TP + FP}$$

The precision essentially represents the confidence we can put in the model when it predicts a case as positive. Recall (also known as sensitivity or true positive rate) is the proportion of positive observations that are detected among all positive observations, and is computed as

$$Recall = \frac{TP}{TP + FN}$$

The recall is a measure of how well the model recognizes positive observations.
Ideally, precision and recall should both be as high as possible, but usually there is a trade-off between them.

[1] TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

From these two measures, we can also define the F1-score, which is computed as a weighted average of precision and recall:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

These metrics are often presented in the form of ROC curves and precision-recall curves.

4.1.1 ROC Curves

ROC (*Receiver Operating Characteristic*) curves plot the recall as a function of the *1-sensitivity*, also called false positive rate [2] with varying confidence thresholds. It illustrates the predicting ability of a binary classifier as its confidence thresholds varies.
In practice, the predictions of the classifier are sorted from the most confident to the least confident, and the confidence threshold is varied. Each value of the threshold yields a different confusion matrix from which the recall and 1-sensitivity can be computed, corresponding to a point of the curve. Figure 4.2 gives an exemple of ROC curve for some binary classifier.

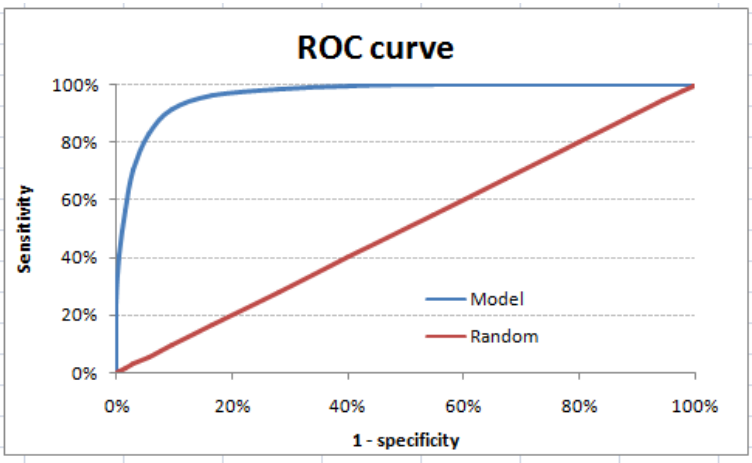

Figure (4.1) Example of ROC curve of some model (blue) and a random classifier (red).

Intuitively, at the point (0,0), the classifier always categorize an observation as a negative case: there is no false positive, but there is no true positive either. At (1,1), the classifier always categorize an observation as a positive: there is no true negative, but no false negative either. Thus a random classifier would give a flat ROC curve passing through the points (0,0) and (1,1).
On the other hand, a perfect classifier would yield a ROC curve passing through the point (0,1), i.e. no false positive and no false negative. A good model will thus be characterized by a ROC curve tending towards the point (0,1). For that reason, it is often convenient to summarize the ROC curve by a single value, called the *area under the curve* (AUC); the AUC is equal to 1 for a perfect classifier and to 0.5 for a random one. In general, the higher the AUC, the better the predictions.

[2] Defined as $\frac{FP}{FP+TN}$.

4.1.2 Precision-Recall Curves

Similarly, Precision-Recall (PR) curves plot the precision as a function of the recall when varying the confidence threshold. Here, a perfect classifier would pass through the point (1,1), and once again, the area under the curve, called MAP (for *Mean Average Precision*) can be used to summarize it; the higher the MAP, the better the predictions. One thing important to note, however, is that while ROC curves are independent of the ratio between positive and negative observations, PR curves are not; therefore, class imbalance can be an issue when considering Precision-Recall curves.

4.2 Visualization techniques

On top of these metrics, it can be interesting to visualize the predictions of the models. Because the task of papyrus fragment matching involve learning a (dis)similarity/distance measure, it is relatively well-suited for visualization techniques such as hierarchical clustering methods, multidimensional scaling and the t-SNE algorithm. While these methods are not necessarily a mathematically rigorous way of assessing the results, they represent a convenient way to look at them and could provide interesting information.

These visualization techniques only require a distance matrix giving distance measurements between objects; and such a distance matrix can easily be obtained from the neural networks considered in this thesis.
The architectures presented in section 2.2 end with a softmax layer, meaning that they output, for each input pair, a vector of two elements: the probability P_0 given by the network of the input belonging to class 0 (i.e. "similar pair"), and the probability P_1 of the input belonging to class 1 (i.e. "dissimilar pair"), which is of course $1 - P_0$.
The closer P_0 is to 1, the more confident the network is about the fragments in the input pair being similar (and belonging to the same original papyrus); conversely, if P_1 is close to 1, the network considers that the fragments are dissimilar and do not belong to the same papyrus. Therefore, P_1 can be seen as a *distance measure* between the fragments in the input pair, and this value can be used to build a distance matrix for a given set of fragments.

It is thus possible to build a distance matrix directly from a model's prediction for a set of fragment pairs.

4.2.1 Hierarchical clustering

Hierarchical clustering groups similar objects into entities called clusters. The objective is to iteratively define clusters that are distinct from each others, and for which the objects within each cluster are broadly similar to each other.
Several hierarchical clustering strategies exist, using either an agglomerative (bottom-up, i.e. start from clusters containing a single object and aggregate iteratively) or divisive (top-down, i.e. start with all objects belonging to a single cluster and divide it iteratively) approach.

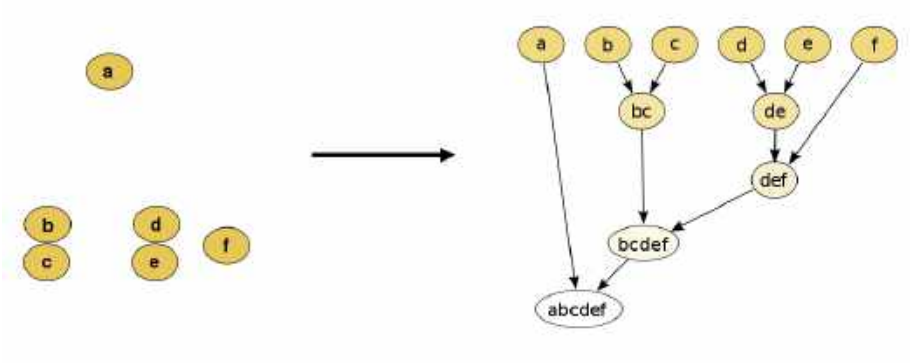

Figure (4.2) Illustration of agglomerative hierarchical clustering; raw data on the left, resulting dendrogram on the right.

The linkage criterion determines the distance between sets of observations as a function of the pairwise distances between observations. Hierarchical clustering has the distinct advantage that any valid measure of distance can be used. Even the observations themselves are not required: all that is used is a matrix of distances.

4.2.2 t-SNE

t-SNE (*T-distributed Stochastic Neighbor Embedding*) is a machine learning algorithm for visualization developed by Laurens van der Maaten and Geoffrey Hinton [17].
It is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. Specifically, it models each high-dimensional object by a 2D or 3D point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. The t-SNE algorithm can also work directly from a distance matrix, in which case it computes the data points based on the pairwise distances between objects.

4.2.3 MDS

MDS (*MultiDimensional Scaling*) [18], much like t-SNE, seeks a low-dimensional representation of the data in which the distances respect well the distances in the original high-dimensional space. Again, it can work directly from the high-dimensional data (in which case a distance matrix is computed by the algorithm) or from a pre-computed distance matrix.

Given an input distance matrix D, the general MDS algorithm works as follows:
1. Assign the objects to arbitrary coordinates in the p-dimensional space. (usually p=2 or p=3)
2. Compute the Euclidean distances among all pairs of points, to form the $\hat{D}$ matrix.
3. Compare the D and $\hat{D}$ matrix by evaluating the stress function. The stress function is a residual sum of the form

$$\sqrt{\frac{\sum_{i,j}(f(x_{ij}) - d_{ij})^2}{\sum_{i,j} d_{i,j}^2}}$$

where f is some function of the input data and d_{ij} is the Euclidean distance between points i and j

in the p-dimensional space.
4. Adjust coordinates of each point in the direction that minimizes stress the most.

Steps 2 through 4 are repeated until the stress is minimized.

There exists two types of MDS algorithm: metric and non-metric. The transformation of the input values $f(x_{ij})$ that is used depends on the type. In metric MDS, $f(x_{ij}) = x_{ij}$. In other words, the raw input data is compared directly to the mapped distances. In non-metric scaling on the other hand, $f(x_{ij})$ is a weakly monotonic transformation of the input data that minimizes the stress function. The monotonic transformation is computed via monotonic regression.

4.3 Test protocol

In order to conduct the experiments, as presented in section 2.2, the fake fragments dataset is used to generate training and test pairs, the former being generated from 75% of the papyri and the latter with the remaining 25%.
In total, 1812 fake fragments coming from 22 different papyri are used in the first experiment, and 4721 fragments coming from 69 papyri in the subsequent ones (see section 3.3). Although for the first experiment, 22 papyri is not a very large number, the selection is quite diverse in colors, preservation state, etc.
From the fake fragments, 103,591 training pairs and 42,859 testing pairs are generated in the first experiment, and 301,797 training pairs and 117,834 testing pairs in the others.

Unless noted otherwise in a specific experiment, each model is trained for 40 epochs with a batch size of 16, using the Adam optimizer with default parameters and categorical cross-entropy as the loss function. The input images' size and the learning rate vary depending on the network; these parameters are given in the section relative to each experiment. Furthermore, unless noted otherwise, data augmentation is used, in the form of random horizontal and vertical flips (probability of 0.5 for both). Everything is implemented using the Keras API with TensorFlow backend.

The train and test accuracy is reported at each epoch by Keras callbacks; as for the evaluation metrics and visualization techniques presented above, the following methodology is used:
A test-test approach is used. Considering test fragments only (i.e. fragments from papyri that were *not* used during training), all possible fragment pairs are generated. For the first experiment, done on the first, smaller iteration of the dataset, all test fragments (coming from 6 distinct papyri) are used; for the other experiments, that use the full dataset, only part of the test fragments is used, from 10 distinct papyri, otherwise the figures presented would be unreadable.
The trained networks are then used to produce predictions for these testing pairs, giving for each fragment pair a probability P_0 that the fragments are from a similar papyrus, and probability P_1 that the fragments come from different papyri.
As mentioned before, the probability P_1 can be used as distance measure, and therefore a distance matrix can be created using the P_1 probabilities given by the network for each fragment pair. This

distance matrix can then be used to generate ROC/Precision-Recall curves, run the t-SNE and MDS algorithms, and use hierarchical clustering to produce a dendrogram.

For comparison purposes, a simple, "naive" baseline is also considered, using the same test pairs, to have a better appreciation of the utility of deep learning models. This baseline simply consists in computing the colour (RGB) histogram of the fragment images of a given pair and using the χ^2 measure between them as a distance metric. This is done using the OpenCV[3] library.
Although this approach is very naive, as it only considers the colour composition of the images to compare them, it is still a meaningful approach as the colour of the fragments is quite important (as will be seen in the experiments). It also has the advantage of being simple and fast to use, contrary to neural network models, and colour histograms have been widely used in information retrieval and machine learning tasks before [14] [15]. Figure 4.3 provides an illustration of this baseline.

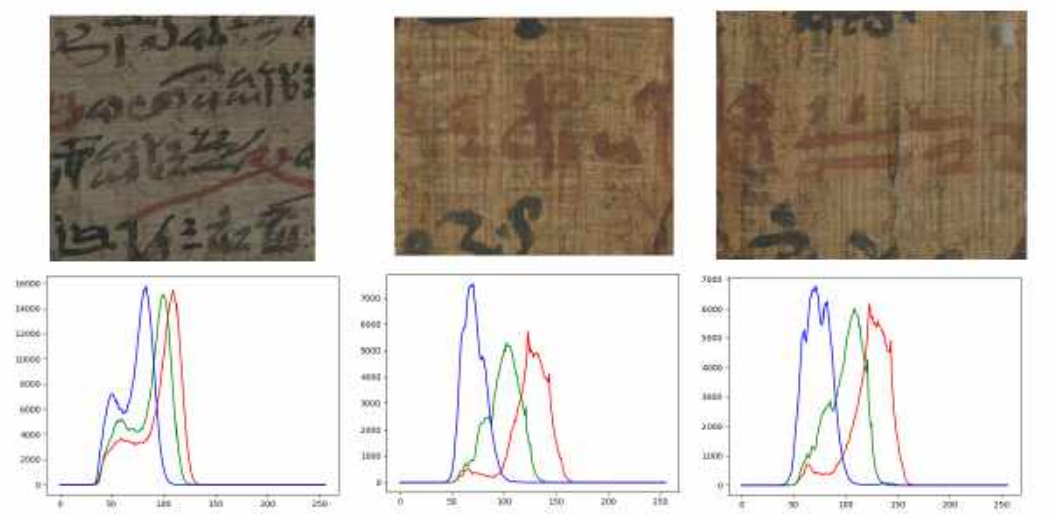

Figure (4.3) Three papyrus fragments and their respective RGB histograms. Notice the similarity between the histograms of the two rightmost fragments (which belong to the same papyrus).

[3]https://opencv.org/

Chapter 5

Experiments

In this chapter, several experiments are conducted using the architectures presented in chapter 2, and their results are presented.
In the first experiment, the Papy-S-Net architecture is trained using both rectangular crops of the papyrus fragments and alphamasks with realistic shapes (see figure 3.6), and the results are compared. The objective is to determine whether the shapes of the fragments are useful or not to achieve good performances on the task of papyrus fragment matching.
In a second experiment, several variants of the ResNet50 and Xception siamese architectures are trained on rectangular crops, and their results are compared. The goal of the experiment is to assess the impact of transfert learning on the performances, by confronting variants using weights pre-trained on the ImageNet dataset, and variants using random weight initialization. These larger architectures are also compared to the simpler Papy-S-Net architecture.
The third experiment further explores the capabilities of the Papy-S-Net architecture, which is only trained on a small part of the dataset in the first experiment, by using the full dataset and trying different enhancements for the model.
Finally, the performances of the neural network models are compared with those of the baseline presented in section 4.3, in order to evaluate the usefulness of deep learning for the task of fragment matching.

5.1 Impact of the fragment boundaries

The objective of the first experiment is to determine whether considering the exact shape of the fragments is useful to tackle the task, or not. As already mentioned in section 2.2, creating fake fragments of realistic shapes is a very tedious and time consuming process, thus evaluating their usefulness is particularly important. Furthermore, this question has not been addressed by the literature yet, as Pironne et al. ([9]) used rectangular crops in their experiments.

In this first experiment, only the Papy-S-Net architecture is considered. For both the crops and the alphamasks, the learning rate is set at 0.0001 for the first 20 epochs, and 0.00001 for the remaining 20 epochs.

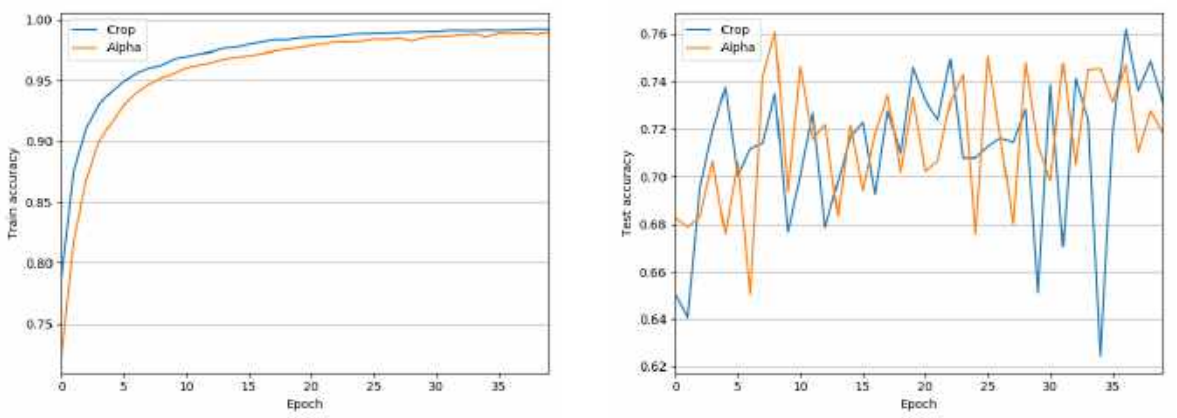

Figure (5.1) Train (left) and test (right) accuracy of the Papy-S-Net architecture when using regular crops or alphamasks.

	Precision	Recall	F1-Score
Papy-S-Net using crops	0.66	0.89	0.75
Papy-S-Net using alphamasks	0.66	0.79	0.72

Table (5.1) Evaluation metrics for Papy-S-Net.

Figure 5.1 shows the compared performances of the Papy-S-Net architecture when using crops or alphamasks, while tables 5.1 give the evaluation metrics of both approaches. At first glance, it seems that the impact of the alphamask is quite negligible, as in botch cases the train and test accuracy are very similar. The evaluation metrics confirm this observation, giving a slight advantage in terms of recall to the model using regular crops.

It is clear that in both cases, the network heavily overfits the data, as the train-set accuracy quickly stabilizes at over 95% while the test-set accuracy oscillates between 70 and 75%. The overfitting is mainly due to the small amount of data used (1812 distinct fake fragments). As a result, while it can be observed that the alphamasks (i.e. the shapes of the fragments) have a negligible impact, it is necessary to remain cautious as this observation come for a limited amount of data.
In order to assess the performances of both approaches in more details, the evaluation metrics and visualization techniques presented in chapter 4 are used on the test data.

First, it is interesting to look at the ROC and Precision-Recall curves obtained. Since to each fragment in the test data corresponds a single ROC/Precision-Recall curve (because each fragment is paired with every other fragment, yielding a distinct classification task), it is more convenient to plot the mean curves and look at the mean AUC/MAP to get a general idea of the performances of the networks.
Figure 5.2 shows the mean ROC curve obtained on the test pairs with Papy-S-Net when using crops (left subfigure) or alphamasks (right subfigure). Overall, there is no significant differences between the two approaches, Papy-S-Net using regular crops yielding a mean ROC of 0.80 (with a standard deviation of 0.15) versus 0.81 (standard deviation of 0.14) for Papy-S-Net using alphamasks.

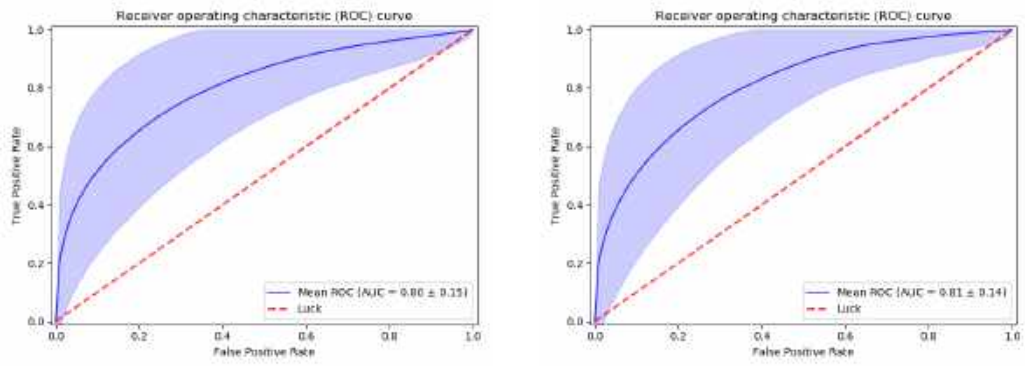

Figure (5.2) Mean ROC curve for Papy-S-Net when using crops (left) and alphamasks (right).

The mean Precision-recall curves (figure 5.3) are significantly worse than the ROC curves, but also, probably, a more realistic representation of the performances of the networks: because every test fragment is compared to every other fragments, dissimilar pairs clearly outnumber similar pairs (303,006 versus 60,603). Therefore, the test set is quite heavily unbalanced and Precision-Recall curves are more adequate. In any case, they exhibit the same tendency: the performances are pretty much identical.

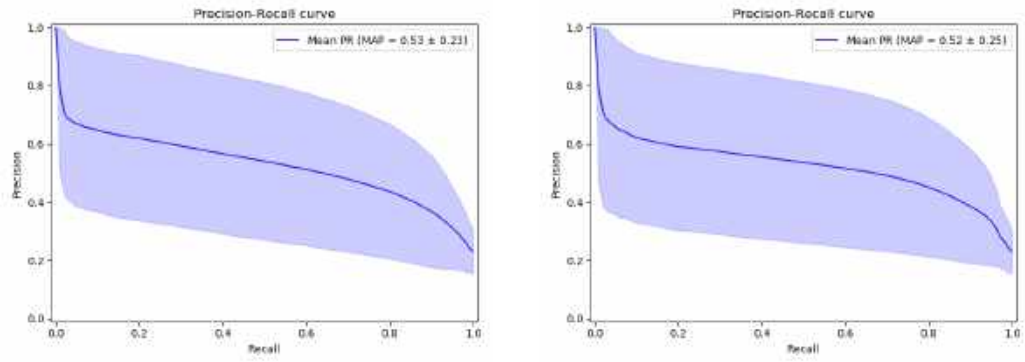

Figure (5.3) Mean Precision-Recall curve for Papy-S-Net when using crops (left) and alphamasks (right).

An example of output when using the t-SNE algorithm is given in figure 5.4. It offers a convenient visual representation of the predictions of both networks, though not absolute as different runs of the algorithm can provide significantly different outputs for the same predictions. Nonetheless, it appears that both approaches yield similar results, with most fragments clustering together with other fragments from the same original papyrus, though it appears that the networks are struggling to distinguish fragments from the papyri labeled 3, 4 and 5.
When using alphamasks, the clusters are looser, less distinct, except for papyrus 2 and 6.

The MDS algorithm produces similar results (figure 5.5), though the 2D representation is less convincing. Once again, papyrus 6 stands out, while papyri 3, 4 and 5 are difficult to distinguish.

A quick look at the fragments of papyri 4, 5 and 6 is sufficient to understand the results provided

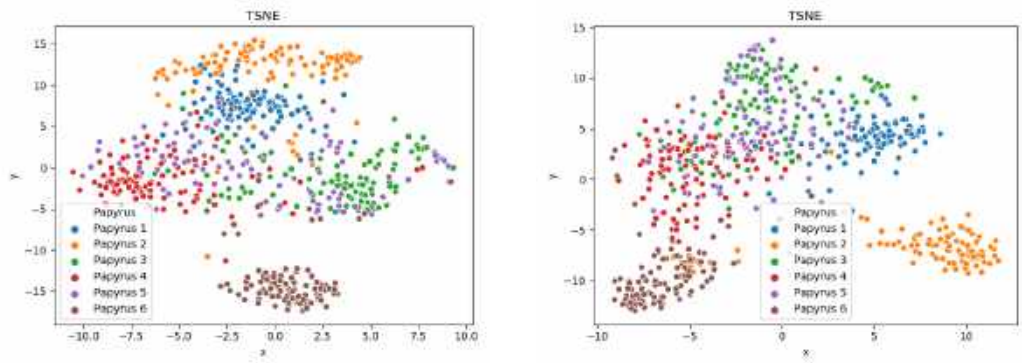

Figure (5.4) Example of t-SNE output for Papy-S-Net when using crops (left) and alphamasks (right).

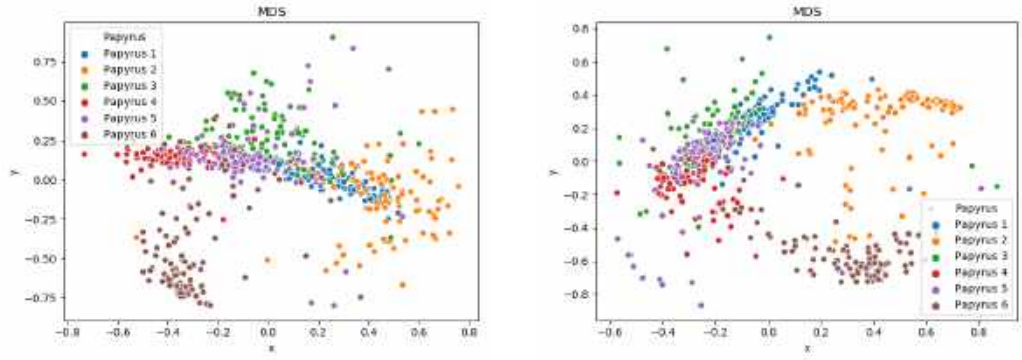

Figure (5.5) Example of MDS output for Papy-S-Net when using crops (left) and alphamasks (right).

by the t-SNE and MDS algorithms: it appears that papyrus 6 has a significantly more red hue compared to the other papyri, which makes it easily stand out from the others. On the other hand, papyri 4 and 5 have very similar hues and textures, meaning that it is very difficult to make a distinction between them without information about the writing/textual content, for instance. Figure 5.6 gives a comparison of fragments coming from papyrus 4, 5 and 6.

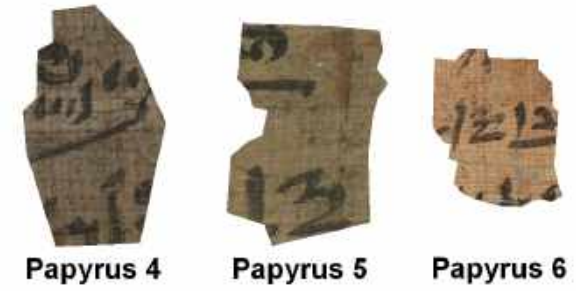

Figure (5.6) Example of papyrus fragments coming from papyrus 4, 5 and 6.

A final visual representation is given by the dendrograms obtained via hierarchical clustering (figure 5.7). Each coloured label on the x-axis represents a papyrus fragment, with the colour indicating

its papyrus of origin. Thus, labels with the same colour should be clustered together early in the dendrogram. Overall, labels of the same colour should be in the same (smallest) subtree, and a quick glance at the ordering of the labels on the x-axis can give a good idea of the quality of the results.
As can be seen in the figure, Papy-S-Net using crops fares better, with three main clusters highlighted (in green, red and cyan), but the same tendencies as with t-SNE and MDS can be observed: one colour clearly stands out, the others are quite mixed up together. Papy-S-Net using crops on the other hand highlights only two large clusters, and fragments with an orange label are present in completely opposite subtrees, which indicates bad performances for this papyrus.

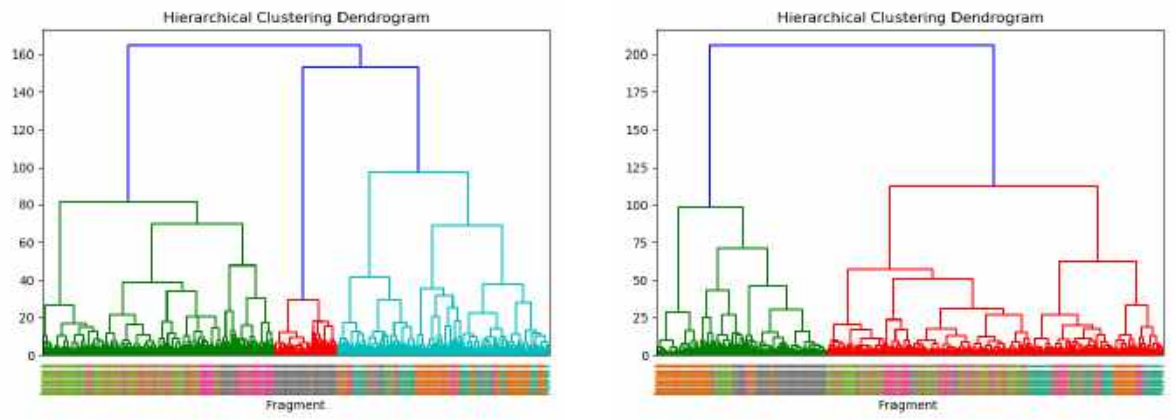

Figure (5.7) Dendrograms for Papy-S-Net when using crops (left) and alphamasks (right).

To conclude, it appears that the fragment boundaries have a negligible impact on the performances, as demonstrated by the various metrics and visualization techniques used, and in accordance to the initial intuition. Although the performances do not suffer from the usage of alphamasks with precise shapes either, it is probably not useful to consider them in a practical use case.

5.2 ResNet50 & Xception: results and impact of transfert learning

The second experiment is done using the ResNet50 and Xception siamese architectures, and evaluates both the performances of the networks and the impact of transfer learning using ImageNet on the training. For each architecture, two variants are used: the first one uses random weight initialization, while the second one uses weights pre-trained on the ImageNet dataset (cf. section 2.3). These variants will be respectively called "Random" and "IN".

All variants are trained for 40 epochs with a batch size of 16, using 224x224 images, and an initial learning rate of 0.0001 which is decayed by a factor x0.1 after 20 epochs.

5.2.1 ResNet50-Twin

Figure 5.8 presents the train and test accuracy of the ResNet50-Twin variants after training for 40 epochs. As in the first experiment with Papy-S-Net, overfitting is an issue despite the larger dataset (but indeed, ResNet50 is considerably larger than Papy-S-Net).
Nonetheless, in terms of test accuracy ResNet50-Twin using pre-trained weights outperforms Papy-S-Net by a small margin, at 76%. It is also apparent that transfer learning could be having a positive impact on the training, particularly in terms of test accuracy, with a clear gap between the two curves, notwithstanding the chaotic behaviour of the test accuracy; this impact should not be overestimated, as training ResNet50 from scratch is quite long and the curves indicate that the training hyper-parameters are probably not optimal, but on the other hand there is no sign of the test accuracy of ResNet50-Twin-Random getting any better with the number of epochs.

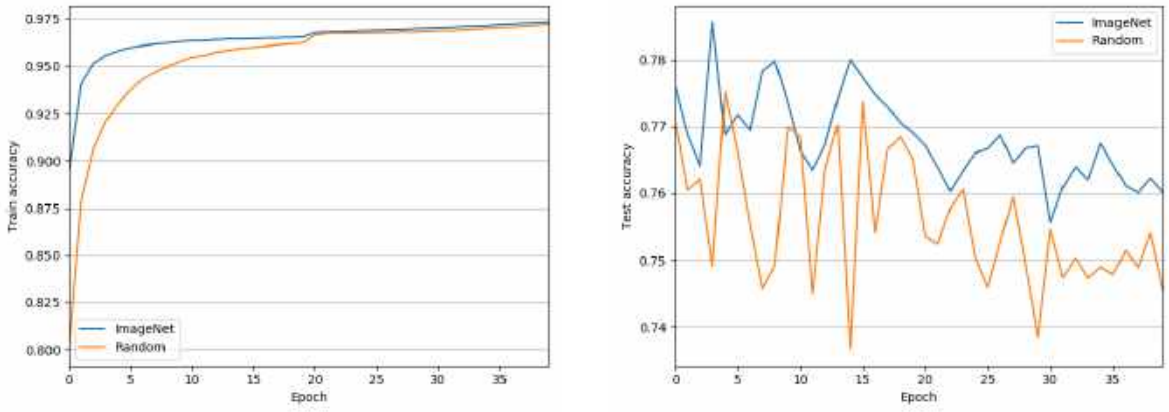

Figure (5.8) Train (left) and test (right) accuracy of the ResNet50 variants.

Table 5.2 gives the precision and recall of class 0 (i.e. similar pairs) for the ResNet50-Twin variants. The precision is slightly higher than for Papy-S-Net while the recall is quite poor, in the range 55-58%. Here, there is no clear difference between the two variants.

As with the first experiment using the Papy-S-Net architecture, the evaluation metrics and visualization techniques presented in chapter 4 are used on a test dataset consisting in all possible pairs generated from the fragments of 10 distinct test papyri. Figure 5.9 presents the mean ROC curves

	Precision	Recall	F1-Score
ResNet50-Twin-IN	0.80	0.55	0.65
ResNet50-Twin-Random	0.74	0.58	0.65

Table (5.2) Evaluation metrics for the ResNet50 variants.

for ResNet50-Twin with and without pre-trained weights. Here, the results are very similar to those obtained with Papy-S-Net, with a mean AUC of 0.80 for both variants.

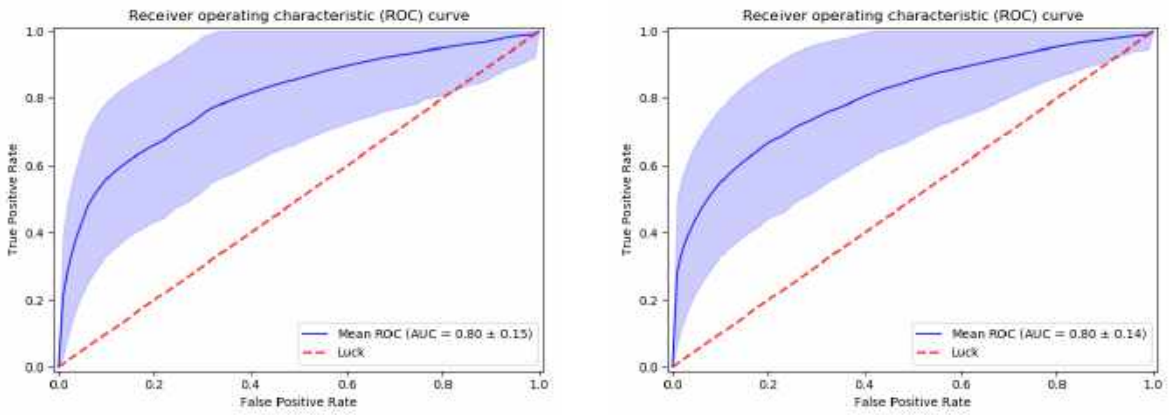

Figure (5.9) Mean ROC curves for ResNet50-Twin using pre-trained (left) and random (right) weight initialization.

Figure 5.10 shows the mean Precision-Recall curves for ResNet50-Twin. As with Papy-S-Net, the PR curves are significantly worse than the ROC curves, but here the results are even slightly worse, with a mean MAP of 0.49 for the variant using pre-trained weight and 0.50 for the variant using random weight initialization. However, the standard deviation over all curves is really high, once again: the network can perform quite well or pretty badly, depending on the fragment considered. Furthermore, the results of both variants are strikingly similar: apart from the test accuracy observed during training, there is no clear impact of transfert learning, judging from the other metrics.

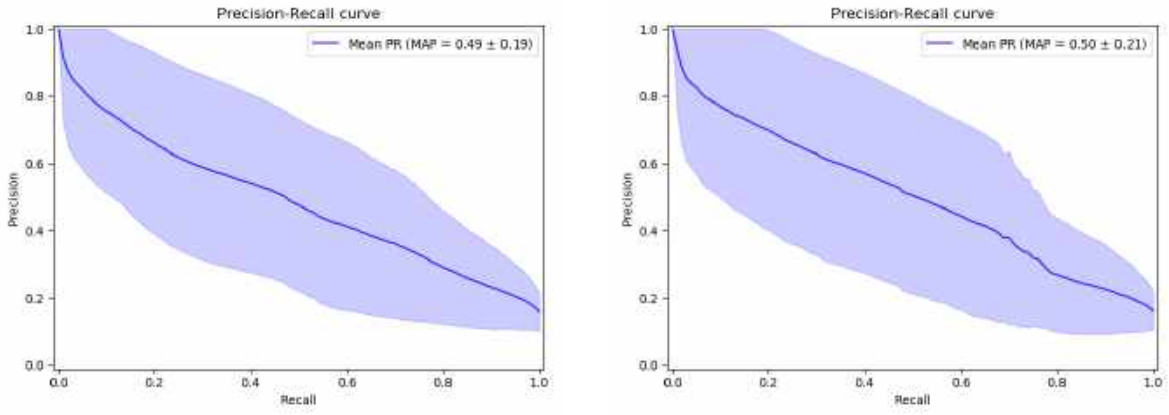

Figure (5.10) Mean PR curves for ResNet50-Twin using pre-trained (left) and random (right) weight initialization.

This is not necessarily surprising; the ImageNet dataset is one of image classification, with a wide range of objects and scenery depicted. Therefore, a pre-trained ResNet (or Xception) network will fare will at most image classification tasks, but Papyrus fragment matching is a very different problem mostly focusing on colours and fine-grained details on the textures, and the features that should ideally be extracted are quite different.

Some example outputs from the t-SNE and MDS algorithms are provided in figure 5.11 and 5.12 to visualize the results. Though it is not possible to compare directly the two architectures, given that different (and fewer) papyri were used to generate the t-SNE and MDS figures with Papy-S-Net, there is a flagrant difference in the case of the t-SNE algorithm. The output obtained with both variants of ResNet50-Twin present more compact and distinct clusters, though the results are not necessarily better. Indeed, while a portion of the fragments for each papyrus cluster correctly together, for both variants a large cluster made of fragments from all papyri mixed together can also be observed.
With the MDS algorithm on the other hand, the outputs are more similar to those already obtained using Papy-S-Net, but the results are not really satisfactory either.

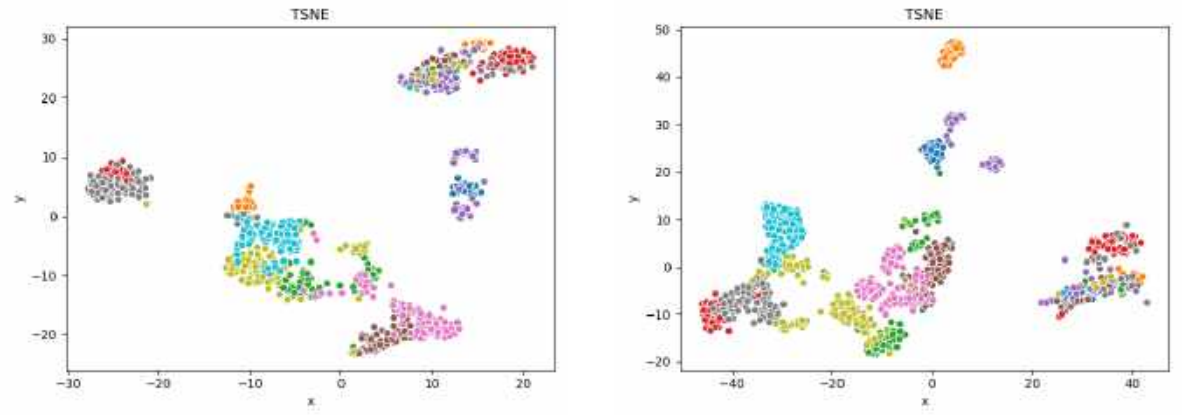

Figure (5.11) Example of t-SNE output for ResNet50-Twin using pre-trained (left) and random (right) weight initialization.

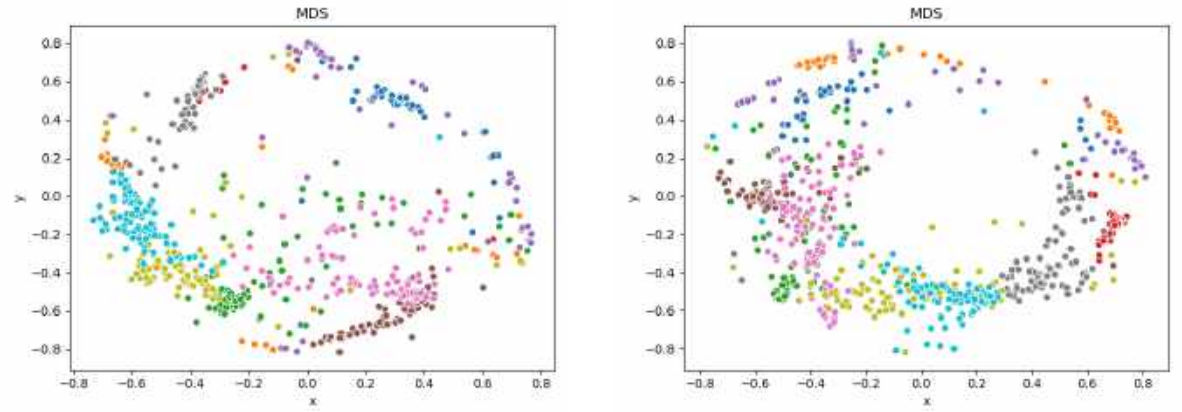

Figure (5.12) Example of MDS output for ResNet50-Twin using pre-trained (left) and random (right) weight initialization.

Finally, a quick look at the dendrograms obtained (figure 5.13). Both variants produced similar results, arguably not much worse than those of Papy-S-Net at first glance. But similarly to what was observed using t-SNE, a problematic group of fragments (probably the same) clearly stands out, constituting the entire left subtree, meaning that these fragments are considered to be clearly distinct from all the others, yet very close from one another. This subtree clearly mirrors the outlying cluster observed for t-SNE.

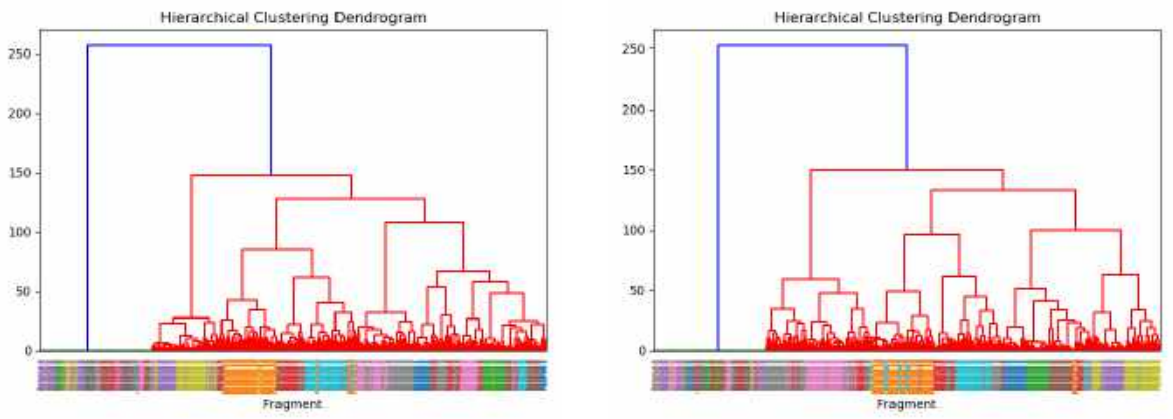

Figure (5.13) Dendrograms obtaines from ResNet50-Twin using pre-trained (left) and random (right) weight initialization.

5.2.2 Xception-Twin

As for the Xception-Twin network, figure 5.14 shows that overfitting is again a clear issue, with the variant using pre-trained weights reaching 95% training accuracy right after the first epoch and struggling in terms of test accuracy, its performances worsening as the training progresses. But while ResNet50-Twin-IN outmatched its Random counterpart in both training and test accuracy, here Xception-Twin-Random yields better results than its pre-trained counterpart and is even the best performing model thus far, with a test accuracy of 77%. This result further indicates that overfitting is the main issue, and that solving this problem could improve the performances of all networks.

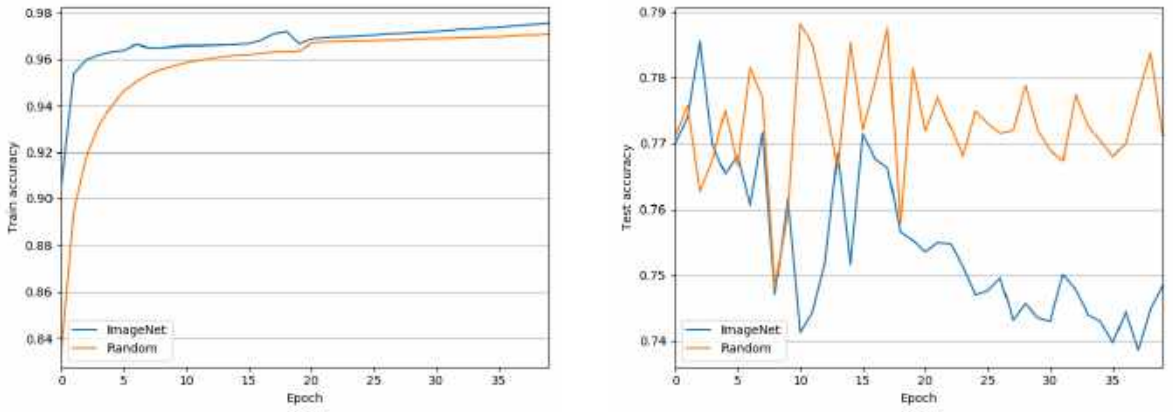

Figure (5.14) Train (left) and test (right) accuracy of the Xception variants.

As table 5.3 indicates, the overall metrics are similar to those obtained with ResNet50-Twin; again, though the precision is satisfactory the recall is quite low, which is quite disappointing given that the models' objective would be to help egyptologists select all promising fragments, rather than be 100% accurate in their matchings (since, at the end of the day, the egyptologists will manually verify everything).

	Precision	Recall	F1-Score
Xception-Twin-IN	0.76	0.55	0.64
Xception-Twin-Random	0.82	0.56	0.66

Table (5.3) Evaluation metrics for the Xception variants.

The ROC and Precision-Recall curves of the Xception variants (figure 5.15 and 5.16) are also similar to those obtained for ResNet50-Twin, with Xception-Twin-IN yielding the worst performances overall and Xception-Twin-Random being on-par with ResNet50-Twin-IN. Overall, the results are disappointing.

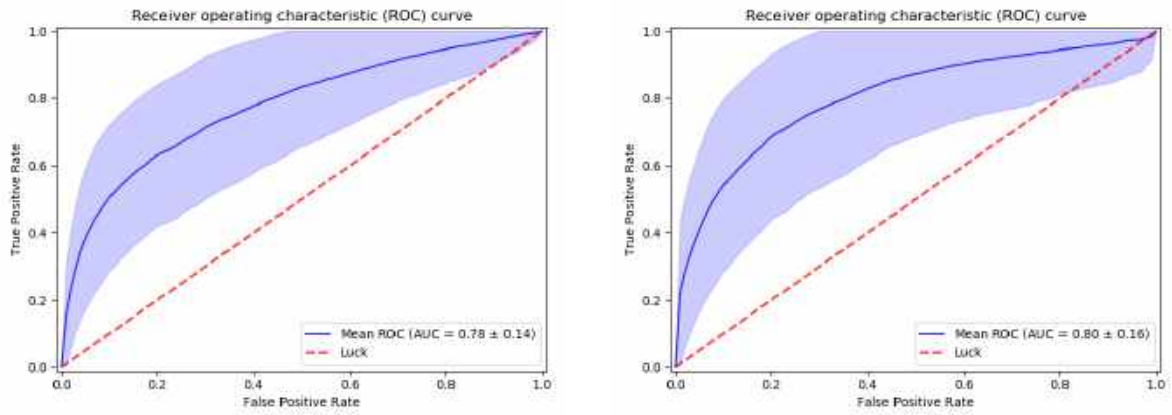

Figure (5.15) Mean ROC curves for Xception-Twin using pre-trained (left) and random (right) weight initialization.

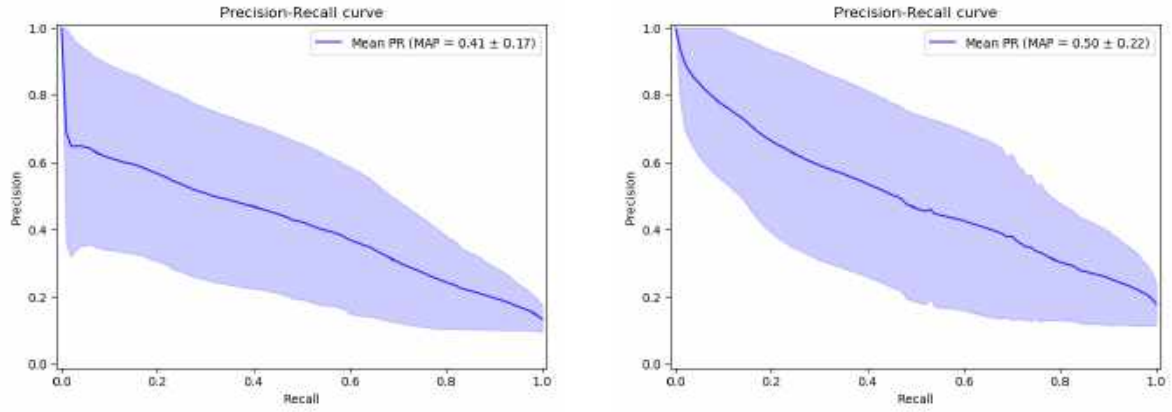

Figure (5.16) Mean PR curves for Xception-Twin using pre-trained (left) and random (right) weight initialization.

The visualization techniques did not show particularly interesting results, in the sense that they are very similar to those obtained using the ResNet50-Twin networks (the cluster of "problematic" fragments already observed when using ResNet50-Twin also appears here); they are therefore omitted to avoid unnecessary redundancies.

In the end, the ResNet50 and Xception siamese architectures achieve performances very similar to those previously obtained with Papy-S-Net, with two considerable differences however: first, the Papy-S-Net architecture has been trained using a smaller dataset than the one used to train ResNet50-Twin and Xception-Twin, and yet did not demonstrate as much overfitting. Second, the ResNet50 and Xception-based networks are considerably longer to train, as their convolutional parts are much larger. As a result, the smaller Papy-S-Net architecture seems more interesting.
Indeed, given that the task of matching papyrus fragments revolves mainly about colours, textures and writings, it seems reasonable that a larger convolutional network, aimed at detecting fine-grained details such as complex shapes, might be unnecessarily complex, to the point where it is counterproductive. A smaller convolutional network might be more appropriate for the task, on top of being generally faster to train.

Transfer learning proved to be impactful, though not necessarily in a positive way. The utilization of weights pre-trained on the ImageNet dataset systematically increased the training accuracy, allowing both architectures to almost perfectly predict the training data in a few epochs. This had a detrimental effect on the generalization capabilities of the network however, a typical case of overfitting; in the end, Xception-Twin fared better when using random weight initialization.
The ResNet50-Twin architecture, on the other hand, clearly benefited from transfer learning and achieved better results on every metric when using pre-trained weights. It is therefore difficult to give a definite answer, and additional experiments should be done; in particular, it would be interesting to use transfer learning with other papyrus collections, such as the multilingual corpus used by Pironne et al. The ImageNet dataset, though proved to be useful in a wide variety of contexts, is not appropriate in the present case.

Overall, the results of these two large networks are disappointing, and it seems reasonable to abandon them in profit of Papy-S-Net or another smaller model. As the next experiment will show, smaller convolutional architectures perform better and do not suffer as heavily from overfitting.

5.3 Additional experiments on Papy-S-Net

Overall, Papy-S-Net is the most "efficient" model, obtaining quite good results (with regards to ResNet50 and Xception) while being considerably faster to train, its convolutional architecture being smaller. On top of that, the first experiment did not make use of the whole dataset, since only fragments with realistic shapes (i.e. 1812 out of 4721) could be considered. Therefore, additional experiments were performed using Papy-S-Net and the whole dataset, in order to assess the real capabilities of the architecture and determine factors that could influence the performances.

5.3.1 Base

First, the "base" Papy-S-Net architecture, already used in the first experiment (section 5.1), has been trained on the complete dataset, using two different initial learning rates (0.0001 and 0.00005); these learning rates are divided by 10 mid-training, after epoch 20.

Figure 5.17 presents the train and test accuracies obtained after training: interestingly, the performances of Papy-S-Net on the full dataset are considerably better than those of the ResNet50 and Xception-based networks, with around 87% test accuracy (and nearly 100% train accuracy) when using the lower initial learning rate. Papy-S-Net thus clearly outperforms the bigger networks while being significantly faster to train and using smaller images. However, overfitting is still an issue with the enlarged dataset and after a certain number of epochs the test accuracy starts decreasing, even though using a very small learning rate seems to mitigate the problem.

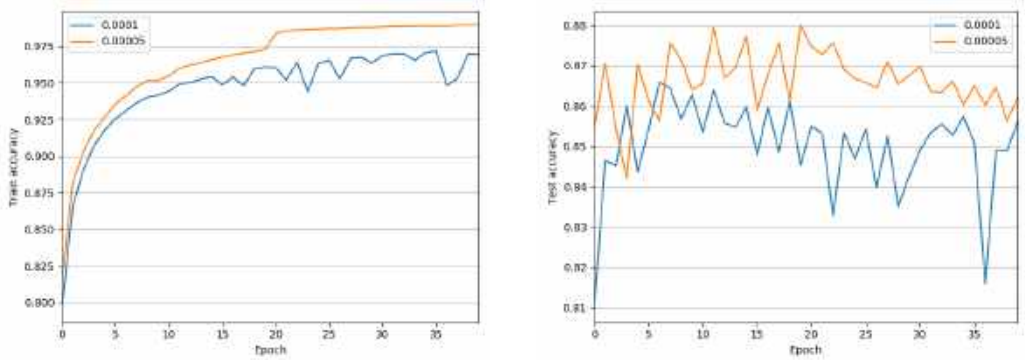

Figure (5.17) Compared train (left) and test (right) accuracies of the Papy-S-Net architecture with the two initial learning rates.

Given that the network trained using an initial learning rate of 0.00005 yielded the best performances, only its results will be presented in details. First, its mean ROC and Precision-Recall curves are shown in figure 5.18: they confirm the good performances already observed in terms of raw accuracy, with a mean AUC of 0.93 (+- 0.08) and a mean MAP of 0.72 (+- 0.20), which is considerably better than everything previously obtained. The increase is particularly spectacular for the Precision-Recall curve, though the standard deviation is again very high, meaning that the network's performances can vary considerably depending on the fragment considered. Nonetheless, these performances are quite encouraging and justify a proper test of the network (conducted in chapter 6).

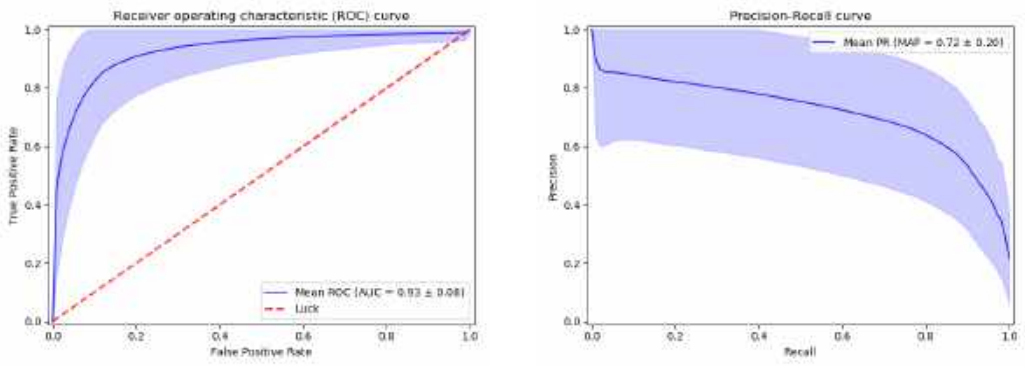

Figure (5.18) ROC and Precision-Recall curves of Papy-S-Net (lr = 0.00005)

The visualization techniques (figure 5.19 and 5.20) also provide interesting results, with very distinct clusters and few "problematic" fragments. The t-SNE algorithm in particular displays impressive results, and while some papyri's clusters are quite close from one another, the distinction between them is quite clear. The MDS algorithm shows very similar results.

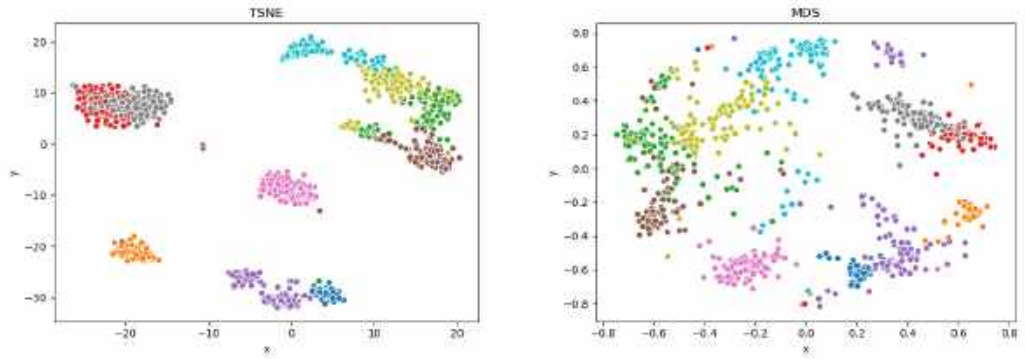

Figure (5.19) Example of t-SNE (left) and MDS (right) output for Papy-S-Net (lr = 0.00005)

The dendrograms obtained from the hierarchical clustering are also largely superior to those obtained using the bigger networks (or with Papy-S-Net on the smaller dataset); though the results are still not perfect, with some problematic fragments found on the far-right of the tree, most fragments are clustered correctly.

Overall, the results are thus pretty solid and quite surprising given the relative simplicity of the architecture; it is by far the best performing model thus far. Nonetheless, several tweaks were considered to try to obtain even better results.

5.3.2 Larger batch size

For practical reasons, the previous experiments used a batch size of 16, which is quite low (in comparison, a batch size of 64 is used by Pironne et al.); and although the batch size is generally not a key factor to improve performances, it is interesting to consider other values and compare the results. Generally speaking, larger batch sizes often lead to lower accuracies, especially in the case of the Adam optimizer (which is the optimizer used here). However, a batch size which is too small

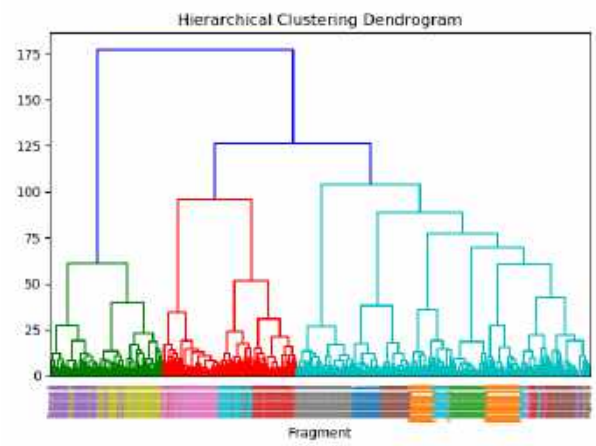

Figure (5.20) Dendrograms obtained with Papy-S-Net (lr = 0.00005)

can be counterproductive as well, and it might be the case with a batch size of 16.
Therefore, the Papy-S-Net architecture was trained using a batch size of 32 in order to see whether it had any impact on the performances, or not.
In terms of train and test accuracy, the impact is negligible: the values are roughly the same and the network seemingly overfits just as much, which is not surprising (figure 5.21).

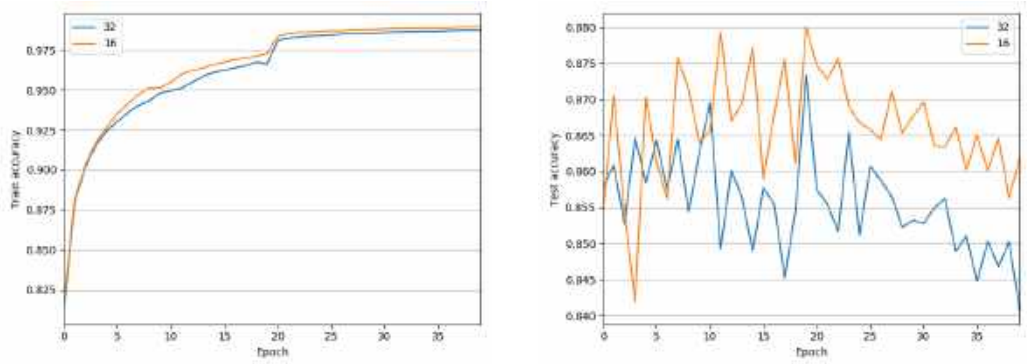

Figure (5.21) Train (left) and test (right) accuracy of the Papy-S-Net architecture with a batch size of 16 or 32.

	Precision	Recall	F1-Score
Batch size of 16	0.66	0.89	0.75
Batch size of 32	0.91	0.67	0.77

Table (5.4) Evaluation metrics for Papy-S-Net with a batch size of 16 and 32.

Figure 5.22 shows the ROC and PR curve of Papy-S-Net when using a batch size of 32. The mean AUC and MAP are slightly lower than what was observed with a batch size of 16, with a mean AUC of 0.91 (versus 0.93 when using a batch size of 16) and a mean MAP of 0.67 (versus 0.72). It thus appears that increasing the batch size has a slightly detrimental impact on the performances.

The outputs of the t-SNE and MDS algorithms (figure 5.23) are quite similar to what was previously

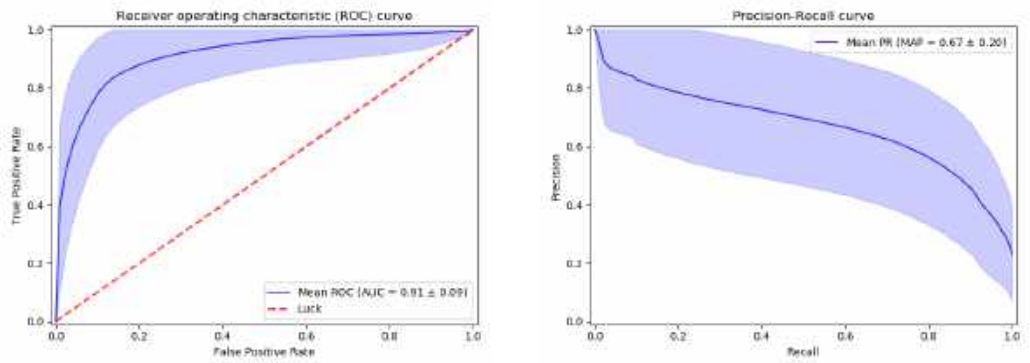

Figure (5.22) ROC and Precision-Recall curves of Papy-S-Net trained on crops using a batch size of 32

observed, and confirm that the impact of changing the batch size is minimal: this is quite logical as the batch size can influence the way the network learns, but not really what it learns (e.g. colours, textures,...). To really improve the abilities of the network, other methods must be used.

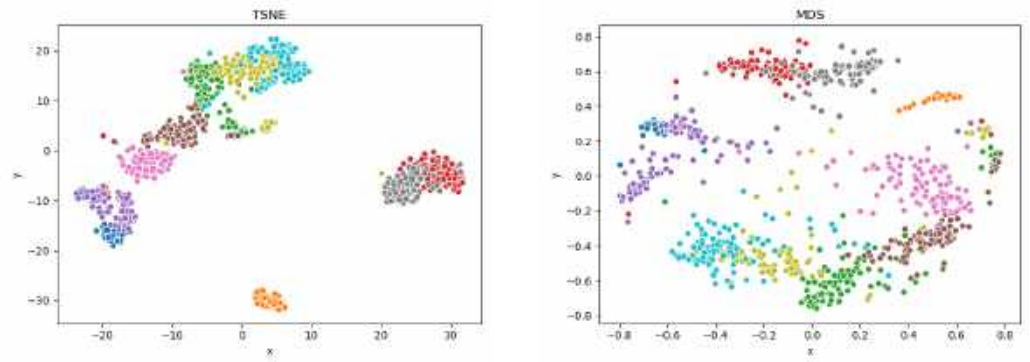

Figure (5.23) Example of t-SNE (left) and MDS (right) output for Papy-S-Net trained on crops using a batch size of 32

As a complement, the experiment was run again with a batch size of 64, but the results (according to all metrics) were on par with those obtained using a batch size of 16; therefore, 16 seems to be the ideal value (lower memory consumption than a batch size of 64, but similar results).

5.3.3 Bigger images

In their paper introducing Papy-S-Net, Pironne et al. use input images of size 128x128, which is quite low and potentially not sufficient to detect information such as entire characters, especially when the original images are in very high resolution. Therefore, increasing the input size could be a straightforward improvement, though increasing it too much can be counterproductive. An input size of 224x224 (identical to the input size of the ResNet50 and Xception networks used in the previous experiment) is thus tested, using a batch size of 16.

Quite surprisingly, while a slight improvement can be seen in the training accuracy when using bigger images, the test accuracy is considerably lower, at around 78% -versus 86%- (figure 5.24).

The two curves exhibit the same behaviour, but there is a net decrease in performances. Such a phenomenon is not easily explained; it is somewhat counter-intuitive, as one could assume that using larger images would have a positive effect.

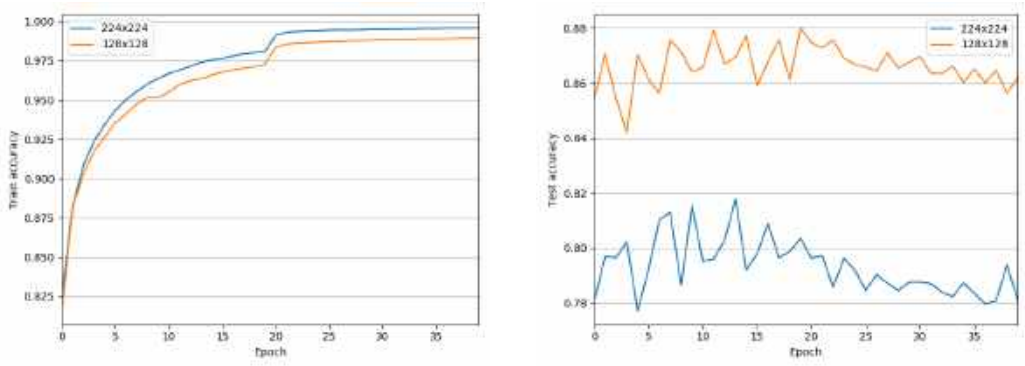

Figure (5.24) Train (left) and test (right) accuracy of the Papy-S-Net architecture depending on the input image size.

The ROC and PR curves (figure 5.25) confirm the decrease in performances, with a mean AUC of 0.88 (+- 0.10) and mean MAP of 0.56 (+- 0.19). The same goes for the outputs of the t-SNE and MDS algorithms.

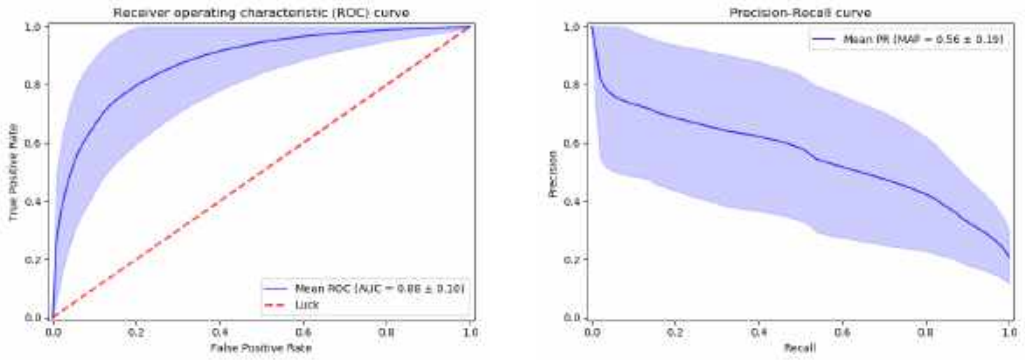

Figure (5.25) ROC and Precision-Recall curves of Papy-S-Net trained on crops using 224x224 images.

On top of the decreased performances, the main drawback is that the network takes considerably longer to train when considering larger images: while an epoch took $\approx$ 60 minutes using 128x128 images, with an input size of 224x224 more than 90 minutes are needed per epoch.

5.3.4 Additionnal data augmentation

Until now, random horizontal and vertical flips (50% of probability for each) were the only kind of data augmentation (DA) used when training the models. The fact that things such as the handwriting, line spacing or character size can provide relevant information to compare the papyri prevents the utilization of data augmentation techniques such as rotations or translations, that would destroy that kind of information. However, another data augmentation technique that could be useful is to manipulate the brightness of the images: as explained before, papyrus fragments that are related can end up in completely different conditions, depending on various factors. Fragments from the same papyrus could thus have different colours, for instance (figure 5.26). And depending on the digitization method, the colours/brightness of the images (e.g. the background) can also be different. It is thus interesting to add random shifts in brightness during training, so that the network learns to deal with these possible pitfalls.

Figure (5.26) Example of colour difference between fragments from the same papyrus.

The Papy-S-Net network was trained again, using both horizontal/vertical flips (with probability 0.5 each) and random shifts in brightness (with probability 0.5). As expected, the additional data augmentation lowers the training accuracy, as it makes it more difficult for the network to perfectly fit the training set. But it appears that this does not help the network generalize better, as the test accuracy remains inferior to what was obtained without this additional DA (figure 5.27).

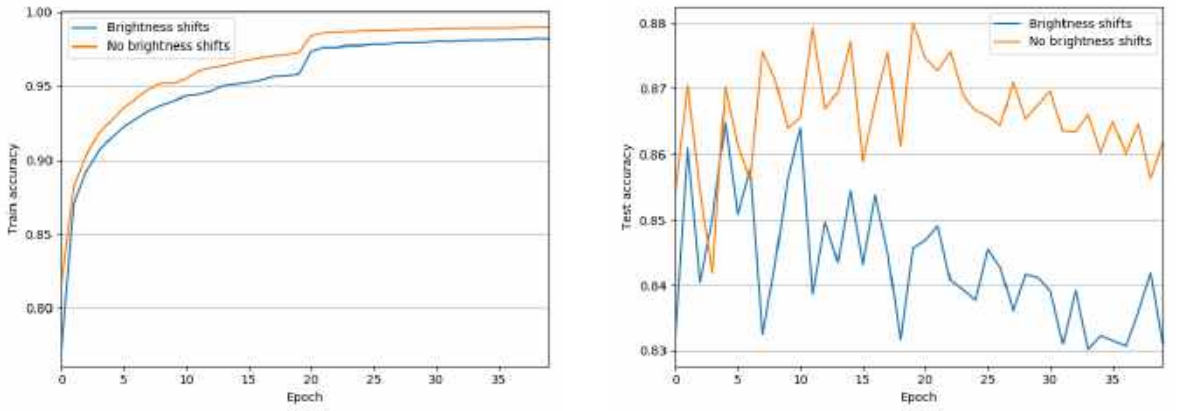

Figure (5.27) Train (left) and test (right) accuracy of the Papy-S-Net architecture when using random shifts in brightness.

The other metrics, computed on all possible test pairs, also show a decrease in performances (figure 5.28), with a mean AUC of 0.90, which is on par with the base Papy-S-Net model, but a mean MAP of 0.58, which is significantly lower.

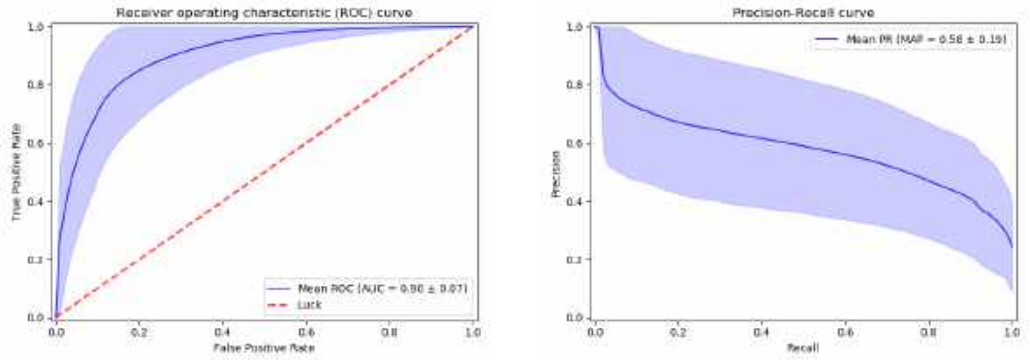

Figure (5.28) ROC and Precision-Recall curves of Papy-S-Net trained on crops using random shifts in brightness

In fact, the random shifts in brightness are probably counterproductive. Even though giving too much importance to the colour of the fragments can be a potential pitfall in some cases (such as illustrated in figure 5.26), in general the colour of the fragment is indeed of primary importance and possibly one of the main discriminators to perform the predictions. Therefore, it is perhaps not a good idea to "dissuade" the network to match fragments based on their colour.
The outputs of the visualization algorithms, shown in figure 5.29, can illustrate this fact. All in all, they are similar to those obtained without the random shifts in brightness. However, a noticeable difference appears with t-SNE: the clusters are closer to one another, which certainly reflects the lesser importance to given to colour.

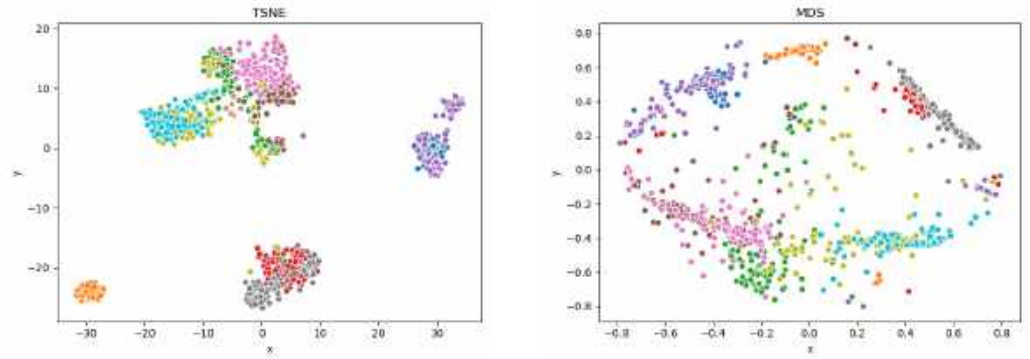

Figure (5.29) Example of t-SNE (left) and MDS (right) output for Papy-S-Net trained on crops using random shifts in brightness

5.3.5 Mixing up the approaches

Clearly, the various tentatives of enhancement for the base Papy-S-Net network led to disappointing, inferior results. A final tentative would be to combine these different approaches in order to obtain a hopefully better model. The Papy-S-Net architecture is thus trained again, using a batch size of 32, an input size of 224x224 and random shifts in brightness on top of the usual image flips. Figure 5.30 shows the training and test accuracy of the Papy-S-Net architecture as-is ("Base", blue line), and using the increased batch size, image size and data augmentation ("Mix", orange line). Again, the base network, not using any of the improvements, still performs significantly better than its "enhanced" counterpart.

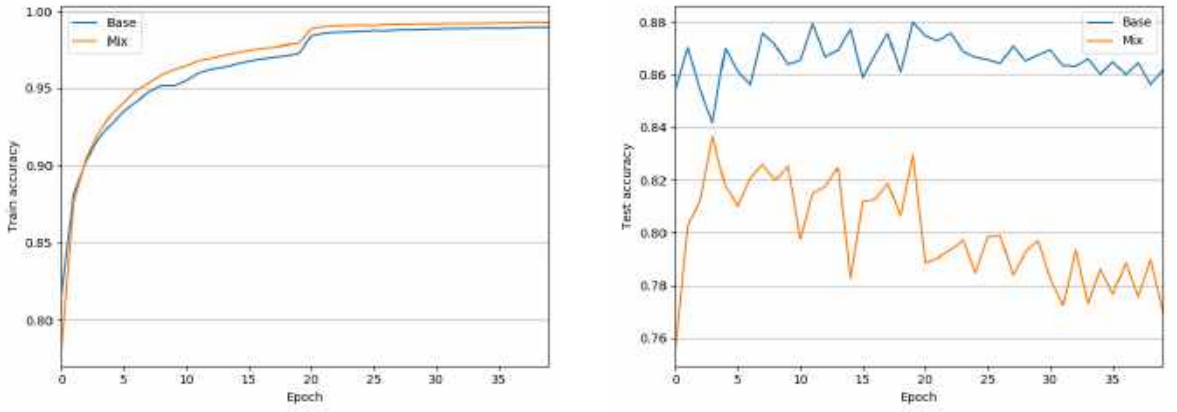

Figure (5.30) Train (left) and test (right) accuracy of the Papy-S-Net architecture when using a mix of all approaches, and when using no improvements.

As for the ROC and Precision-Recall curves and the visualization algorithms, their results are on par with what was obtained with the individual enhancements discussed above, which is disappointing. As a result, they will be omitted.

To conclude, this experiment makes it clear that Papy-S-Net is the best-performing architecture, while also being very simple. More generally, smaller convolutional networks seem to be well-suited for the task of papyrus-matching.
Given its performances, a practical use of the network could be considered, and the next chapter will constitute a first step in that regard. However, the network certainly suffers from overfitting, as the different approaches to enhance it led to very similar results. The amount of data might still be a limiting factor, even with the additional data considered compared to the first experiment conducted in section 5.1. Indeed, the increase in performances observed simply by increasing the amount of data after this first experiment is remarkable.

5.4 Usefulness of deep learning models

The results obtained in the previous experiments are promising, but one could question the usefulness of deep neural networks considering the time and computational resources they require. Therefore, the results of the Papy-S-Net, ResNet50-Twin and Xception-Twin architectures are compared to the baseline model presented in section 4.3, which uses the RGB histograms of the images to make predictions in the form of the χ^2 distance between the images' respective histograms.

The baseline does not have train and test accuracies, but using the usual methodology it is possible to generate ROC and Precision-Recall curves, and use the t-SNE, MDS and hierarchical clustering algorithms. Figure 5.32 presents the ROC and Precision-Recall curves of the baseline:

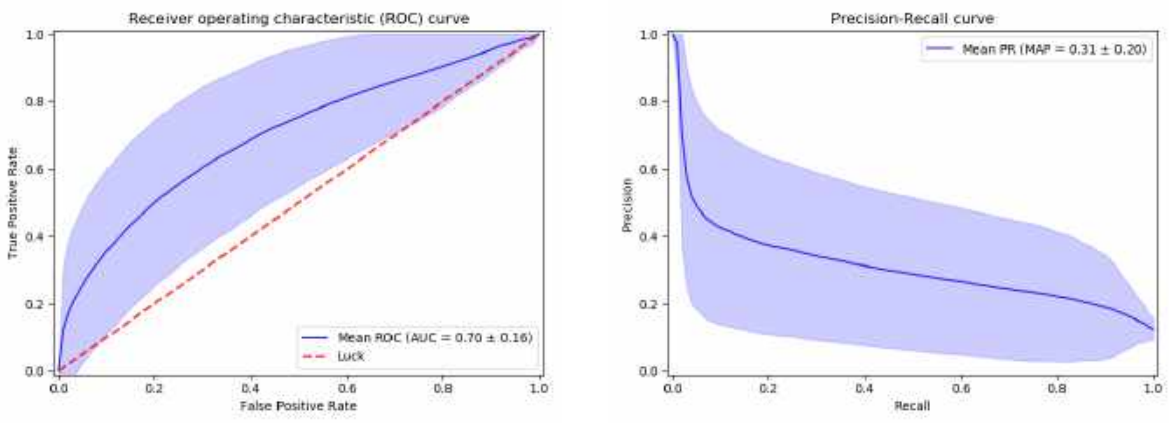

Figure (5.31) Mean ROC and PR curves for the RGB histogram baseline.

The results are quite mediocre, with a mean AUC of 0.70 (+- 0.16) and a very low MAP of 0.31 (+- 0.20). In a sense, given that the baseline only uses the colour distribution of the input images and is much quicker to use than deep learning model, these results are not that bad. Still, both the large ResNet50-Twin and Xception-Twin models and the smaller Papy-S-Net model vastly outperforms the baseline.

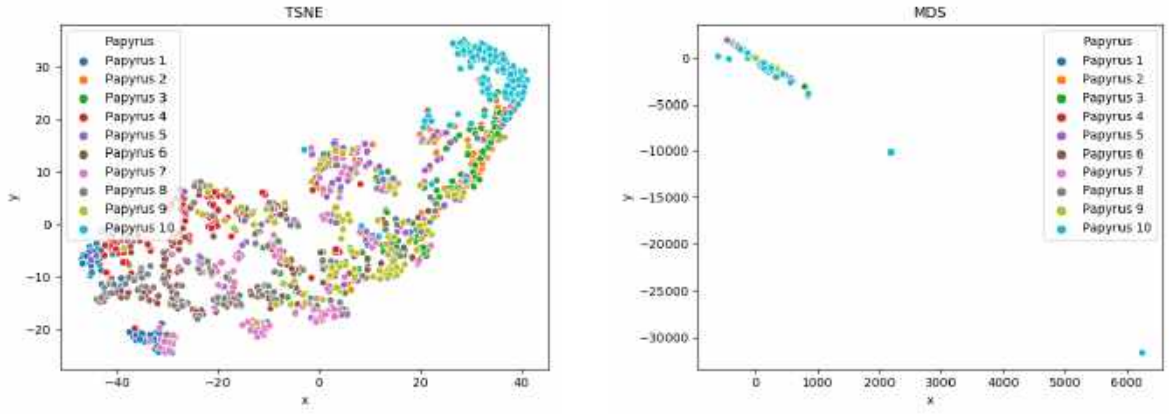

Figure (5.32) t-SNE and MDS outputs for the RGB histogram baseline.

The t-SNE and MDS algorithms yielded outputs quite different from what was obtained with the deep learning models, and the results are not satisfying either: with t-SNE, the obtained representation is dispersed and does not show any particular clustering, with the exception of papyrus 10 which clearly stands out (which is not really surprising given its distinctive greyish tone, illustrated in figure 5.33). The MDS algorithm does not provide any meaningful result either, with two particularly extreme outlying fragments.

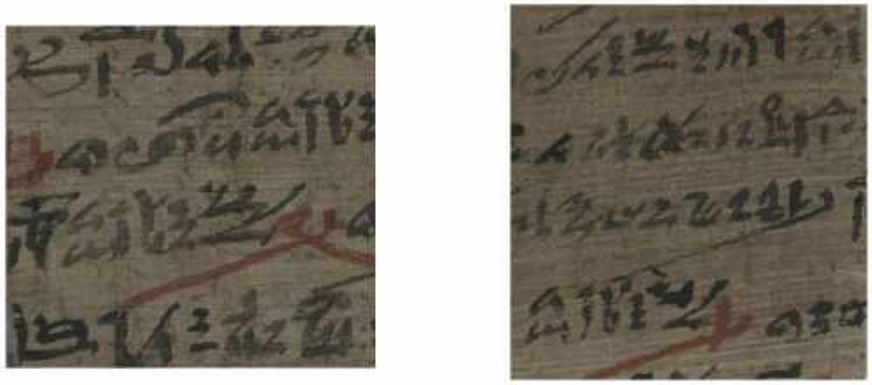

Figure (5.33) Example of fragments for papyrus 10.

To conclude, it appears that deep learning models are indeed useful and can lead to significantly better performances than basic computer vision approaches such as colour histograms, that can potentially perform really well depending on the dataset (e.g. a collection of papyri that for some reason have very different colour tones) but will most likely fail to generalize adequately given that they only consider the colours. Deep learning models base their predictions on more general features that should not be heavily-dependent on the test data, although colour is obviously one of the main features considered by neural networks as well.

Chapter 6

"Real" use case

Although the previous experiments provided interesting results, they were conducted in a rather theoretical setting, with the dataset being -out of necessity- artificially generated from already-reconstructed papyri. In practice however, the models will be used by egyptologists working on pairs of individual fragments coming from an unknown number of distinct papyri, often without much context.

Therefore, in order to evaluate the performances of the best performing model, Papy-S-Net, on a more realistic use case, a "blind test" was conducted in collaboration with Dr. Stéphane Polis, who is an egyptologist.
Mr. Polis selected several folders of fragments that had already been studied and matched by the Crossing Boundaries team but not yet reassembled, and provided the pictures without additional information. The fragment images' filenames indicated their folder of origin, and fragments were certainly not put inside the folders completely at random; however, this information was obviously not used to make the predictions, and the neural network model was thus completely agnostic with regards to the fragments.
From these folders, 150 individual fragments coming from 5 folders were selected to conduct the experiment: all possible pairs using these fragments were generated -with the exception of trivial (x,x) pairs and useless duplicates, i.e. having both (x,y) and (y,x)-, and the Papy-S-Net model was used to compute a prediction for each pair. Each prediction is a value between 0 and 1 which is the probability of the two fragments matching. To make it more readable for Mr. Polis, these probabilities were converted to integers values ranging from 0 to 9, a "score" of 0 indicating a probability of match $p < 0.1$, a score of 1 a probability of match $0.1 \leq p < 0.2$, etc.
The resulting list was sent back to Mr. Polis to be manually evaluated.

It is interesting to note that for most fragment pairs, the model is very "confident" in its predictions, with almost all pairs being attributed a matching probability < 0.1 or ≥ 0.9 (figure 6.1). Admittedly, this is very different from what is going on for the egyptologists working on such fragments, where it is very difficult to be certain about the matchings, except for very large fragments.

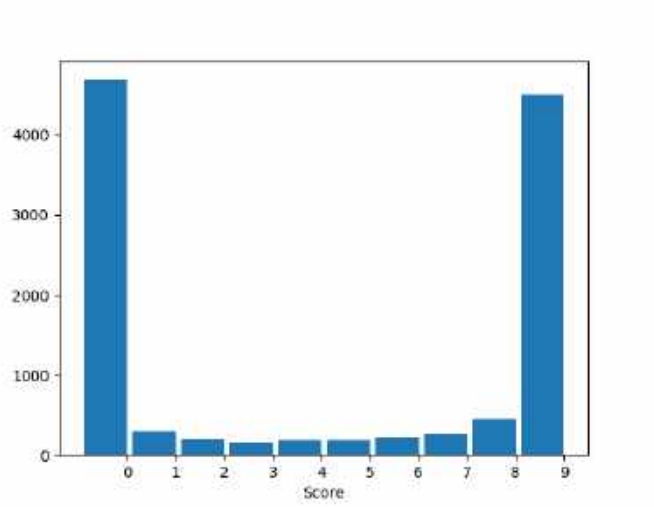

Figure (6.1) Histogram of the similarity scores attributed by the Papy-S-Net network.

With 150 distinct fragments, it is quite difficult to visualize the results in a convenient way; nonetheless, as unesthetic as it may be with such a large number of items, a similarity matrix is always interesting to consider. Figure 6.2 thus presents the similarity matrix obtained on the fragments, with the similarity being simply defined as the probability of the fragments matching (i.e. the prediction returned by the Papy-S-Net model for a given pair).

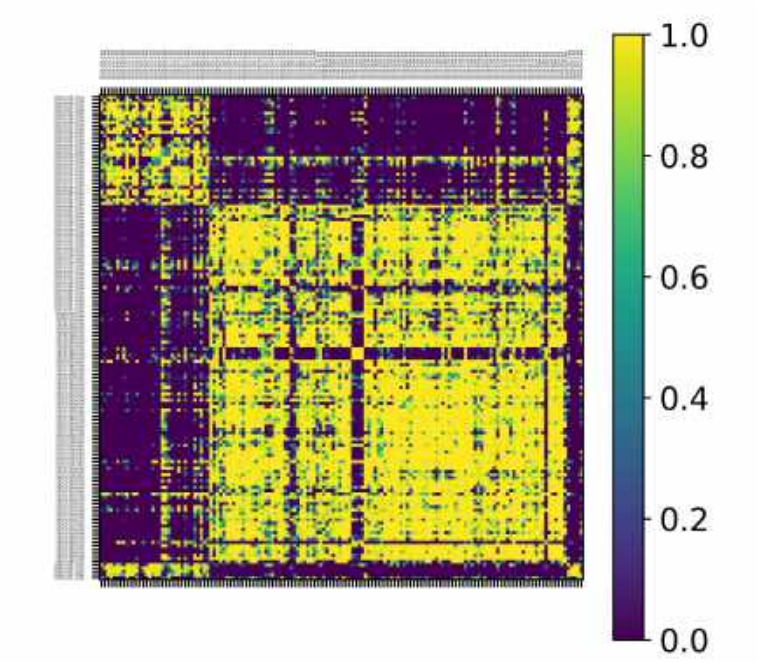

Figure (6.2) Similarity matrix obtained with Papy-S-Net on the 150 fragments.

As aforementioned, most of the predictions are close to 0 or 1. But what is most interesting is

the "squared pattern" centered around the diagonal, which indicates[1] that most fragments were deemed most similar to fragments from the same folder, a conclusion similar to the intuition mentioned above.

It is not always the case, however, and Papy-S-Net also found a lot of similarities between fragments coming from two particular folders, CP154 and CP045. A few fragments were also, quite curiously, considered very similar to most of the other fragments, or conversely, dissimilar to most of them. Upon closer inspection, it appears that this phenomenon is caused by two particular kinds of fragments (with some examples presented in figure 6.3).

Figure (6.3) Four of the problematic fragments.

On the one hand, very small fragments are deemed similar to most other fragments, probably because most of the image passed to the network for them is background (figure 6.4) and this somehow confuses the network. This is in fact one of the main weaknesses of the network. One solution would be to use some sort of edge detection pre-processing for the images, to isolate the fragment from its background and set its surroundings to a null value. Still, the exact reason why this would make the fragments seem similar to most others is unknown.

On the other hand, the fragments deemed dissimilar to most others present a distinctive colour and texture, which is in fact a "fixer" applied by egyptologists at the beginning of the twentieth century (figure 6.5). Though this kind of fixer is not used anymore, a small fraction of the fragments from the Turin collection presents this characteristic, which makes them immediately stand out visually. The dataset used in the present thesis does not contain any papyrus treated with such fixers, as they were deliberately avoided; given that a very small portion of the papyri is concerned, and that these fixers are not used anymore, it seemed reasonable to avoid using them, as it could potentially mislead the network. However, this question is up to debate.

[1]The fragments are sorted by filename, and thus by folder.

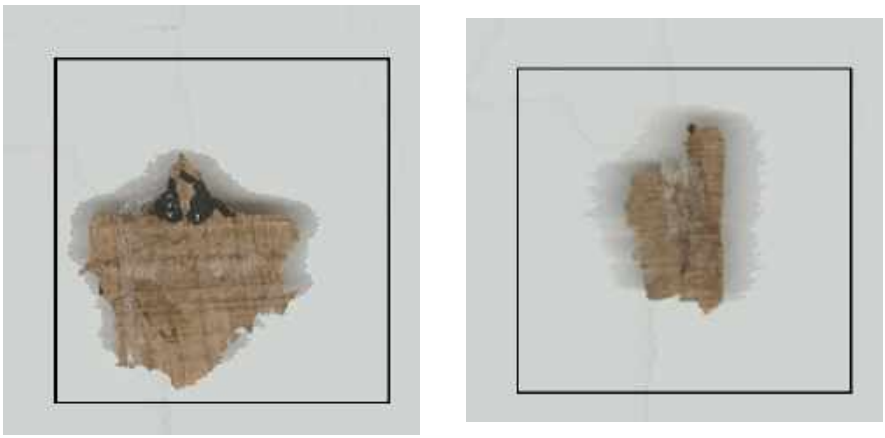

Figure (6.4) Crops of two of the smallest fragments. The image actually fed to the network is the area comprised inside the black square.

Figure (6.5) Papyrus fragments which were treated with a fixer.

Chapter 7

Conclusion & future work

For papyrologists, the task of reconstructing papyri from scattered fragments is a tedious and time-consuming, yet crucial task to uncover new texts that could provide meaningful information on a variety of subjects. With the advent of digital humanities, several tools have been developed to make papyrological studies more effective, in particular when it comes to the digitization and manipulation of papyri. But although these tools are certainly useful for papyrologists working on the reconstruction of papyri, in particular to preserve the fragile papyri from direct manipulations, they do not help them for the decision-making involved in this "puzzle-solving" task.
In the present piece of work, deep learning techniques have been applied to the task of papyrus fragment matching, to evaluate the possibilities offered by neural networks to tackle this complex task.

The first step consisted in building a dataset from scratch, using raw data provided by the Crossing Boundaries project team; using the Cytomine open-source software, a dataset of papyrus fragment pairs has been manually created from 69 distinct, large-size papyri.
Using this dataset, several siamese neural network architectures have been trained and evaluated using various metrics and visualization techniques to assess and visualize the results. A baseline model, consisting in the χ^2 distance between the RGB histograms of both fragment in the pair, was also considered for comparison purposes.

The siamese networks yielded interesting results, particularly the Papy-S-Net architecture recently proposed in the literature, which is quite small and fast to train.
Using this architecture, a first experiment showed that considering complex (i.e. semi-realistic) shapes for the papyrus fragments was not particularly useful to train the networks, as the results were similar to those obtained using only rectangular crops. This result is particularly interesting because it eases the dataset creation process (rectangular crops being much faster and easier to produce), it is thus good to know for future works.

The second experiment used larger convolutional networks and aimed at evaluating the impact of transfer learning (using weights pre-trained on ImageNet) on the performances. While the ResNet50 siamese architecture seemingly benefited from transfer learning, with a slight increase in performances, it proved to be counterproductive for the Xception-based architecture, which ended up

overfitting the data completely. It is thus difficult to give a definite answers. Additional experiments should be conducted, with additional data to reduce the risks of overffiting. It would also be particularly interesting to consider other papyrus collections to perform transfer learning, instead of "general" datasets such as ImageNet; but large papyrus image datasets don't exist yet, though the ongoing efforts for the digitization of papyrus collections across the world may change this in the near future.

Further experiments conducted using the Papy-S-Net architecture showed particularly good results according to all metrics, to the point where a practical use of the network could be considered. For that reason, a more "realistic" use case was considered with the help of Mr. Stéphane Polis, in the form of a blind-test performed on a new batch of papyrus fragments.

The naive baseline model using the χ^2 distance between the RGB histograms of the fragment images performed really poorly, which is not surprising given its extreme simplicity. Nonetheless, given that neural networks are considerably more expensive in terms of computational power and training time, the baseline fared well, at least on fragments coming from papyri with distinctive colour tones.
Still, it seems that deep learning models are indeed appropriate to tackle the task of papyrus fragment matching. But the limited amount of exploitable data makes it difficult to train deep neural networks: all three siamese architectures considered in the present piece of work showed signs of overfitting, which is not surpising given that only 69 different papyri were used to conduct the experiments.

Thus, the results could be improved in many ways. First and foremost, a larger dataset could be used, to reduce overfitting and see the real capabilities of the larger siamese architectures. Papyri from other collections (possibly in other languages, such as Coptic or Greek) could be used to increase the amount of data.
It would also be interesting to consider other loss functions to train the siamese architectures, several losses having been developped specifically for siamese networks learning distance metrics (such as the contrastive loss [19] [20] or the triplet loss). Another possibility could be to use triplet networks [21].

Bibliography

[1] Lucia Vannini, "Trends in Digital Humanities: Insights from Digital Resources for the Study of Papyri", in *Proceedings of the Digital Humanities Congress 2018*, 2019.

[2] S. Polis, K. Gabler, C. Greco, E. Hertel, A. Loprieno, M. Müller, R. Pietri, N. Sojic, S. Töpfer and S. Unter, "Crossing Boundaries. Understanding Complex Scribal Practices in Ancient Egypt", in *Rivista del Museo Egizio*, forthcoming 2020.

[3] Susanne Töpfer, "The Turin Papyrus Online Platform (TPOP): An Introduction", in *Rivista del Museo Egizio 2*, 2018.

[4] Frank Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain", in *Psychological Review, pages 65–386*, 1958.

[5] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network", in *Advances in Neural Information Processing Systems 2*, 1989.

[6] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Sickinger and Roopak Shah, "Signature Verification using a "Siamese" Time Delay Neural Network", in *Advances in neural information processing systems (pp. 737-744).* 1994.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] François Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] Antoine Pirrone, Marie Beurton Aimar, and Nicholas Journet, "Papy-S-Net : A Siamese Network to match papyrus fragment", in *The 5th International Workshop on Historical Document Imaging and Processing (HIP '19)*, September 20–21, 2019, Sydney, NSW, Australia.

[10] Sinno Jialin Pan and Qiang Yang, "A Survey on Transfer Learning", in *IEEE Transactions on Knowledge and Data Engineering*, October 2010.

[11] Lorien Y. Pratt, "Discriminability-based transfer between neural networks", in *NIPS Conference: Advances in Neural Information Processing Systems 5*, 1993.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database", in *CVPR09*, 2009.

[13] Minyoung Huh, Pulkit Agrawal and Alexei A. Efros, "What makes ImageNet good for transfer learning?", in *arXiv preprint.* arXiv:1608.08614, 2016.

[14] Jianzhong Fang and Guoping Qiu, "A colour histogram based approach to human face detection" in *International Conference on Visual Information Engineering.* 2003.

[15] Yunus Faziloglu, R. Joe Stanley, Randy H. Moss, William Van Stoecker and Rob P. McLean, "Colour histogram analysis for melanoma discrimination in clinical images", in *Skin Research and Technology, 9(2)*, 2003.

[16] Raphaël Marée, Loic Rollus, Benjamin Stevens, Renaud Hoyoux, Gilles Louppe, Remy Vandaele, Jean-Michel Begon, Philipp Kainz, Pierre Geurts, and Louis Wehenkel. " Collaborative analysis of multi-gigapixel imaging data using Cytomine", in *Bioinformatics, Volume 32, Issue 9.* 32 9:1395–401, 2016.

[17] Laurens van der Maaten and Geoffrey Hinton, "Visualizing Data Using t-SNE", in *Journal of Machine Learning Research.* 9: 2579–2605. November 2008.

[18] Joseph B. Kruskal, "Multidimensional Scaling", Sage, 1978.

[19] Raia Hadsell, Sumit Chopra and Yann LeCun, "Dimensionality Reduction by Learning an Invariant Mapping", in *Pattern Recognition (ICPR), 2016, 23rd International Conference on.* IEEE, 2016, pp. 378–383

[20] Iaroslav Melekhov, Juho Kannala and Esa Rahtu, "Siamese network features for image matching", in *Pattern Recognition (ICPR), 2016, 23rd International Conference on.* IEEE, 2016, pp. 378–383

[21] Elad Hoffer and Nir Ailon, "Deep metric learning using Triplet network", in *arXiv preprint.* arXiv:1412.6622, 2014. Deep metric learning using Triplet network