University of Liège - Faculty of Applied Sciences

Master thesis : Efficient Interactive Annotation for Cytomine

Graduation Studies conducted for obtaining the Master's degree in Computer Sciences and Engineering by Sacré Loïc

Supervisor:
Marée Raphaël

Academic Year 2018-2019

Abstract

Labelling and annotating histological images requires expertise but above all precious time (e.g. clinical time for pathologists). This is why improving the tools to annotate is crucial. Fortunately, the development of technologies such as deep learning have already brought non negligible improvements to computer vision and imagery related tasks.

In this thesis, a study is conducted on the use of a patch-matching based method for annotations in multiple stained sections of a tissue. The final goal is to identify a significant structure thanks to a patch (or also known as a window) in other sections of the same tissue by making patch-to-patch comparison. It is aimed to avoid the repeating process of labelling several time an identical object. The task is not simple as multiple sections might suffer from deformations and difference in staining, among others.

In order to make the comparison, neural networks are used and two similar approaches are tested. The first one consists in extracting some features from the patches thanks to pre-trained networks and comparing them with a measure called the cosine similarity. The second one consists in doing the same thing but the networks and/or the similarity measure are now trained on a specific dataset. The first approach yields the best results. Even if the trained networks have learned to compare, they lack of a huge and diverse dataset to offer convincing results. At this state, a real application can not be built on the basis of the proposed method. This is why, at the end of the report, some suggestions are given to continue investigating it. The study still offers a good starting point.

Acknowledgements

I would like to express my deepest appreciation to all those who contributed in the completion of this thesis.

- I would like to acknowledge with much appreciation my supervisor Raphaël Marée, who kept encouraging and guiding me during the year as well as proofread the work.
- A special thanks to Romain Mormont for all his suggestions and the time he spent proofreading.
- I also wish to thanks Nathan Greffe and Sébastien Blondiau for the time they gave for going through the document.

Contents

Chapter 1

Introduction

Pathology is the study and diagnosis of illness and disease. It plays a crucial role in the development of treatments for infections, viruses like AIDS and diseases like cancer. The expertise goes among others through the microscopic examination of tissue which is a branch of biology referred to as histology. Over the last century, significant improvements have already been made but there is plenty of rooms for more. The birth and growing development of modern technologies have revolutionized and facilitated the work. For instance, deep learning has been a major breakthrough for computer vision and imagery linked tasks thanks to the emergence of convolutional neural networks.

In the field of digital pathology and biomedical research, there is a strong need for efficient annotation tools. A specific and typical use-case for pathologists is the annotation in tissues of significant structures such as abnormal cells. In order to highlight the object of interest, an annotation can be made for example by drawing a bounding box, building a polygon point by point or by painting over roughly. The Cytomine software is especially designed for these kinds of services.

Cytomine is "an open-source rich internet application for collaborative analysis of multi-gigapixel images"[1]. The Cytomine software aims to offer ease of shared and collaborative work when dealing with large pictures. The project was initiated in June 2010 by Raphaël Marée, Grégoire Vincke, Renaud Hoyoux and Christopher Hamilton. Cytomine consists of three main structures:

- **Open Source**[2, 3]: open-source rich internet application for collaborative analysis of multi-gigapixel images using machine learning.
- **Open Company**[4]: responsible for installation, maintenance and support of Cytomine and on-demand machine learning algorithms development co-founded by Raphaël Marée, Grégoire Vincke, Renaud Hoyoux and Christopher Hamilton.
- **Open Research**[5]: development of machine learning algorithms and big data software modules at the Montefiore Institute (University of Liège, Belgium).

Among many of the possible applications, the Cytomine software helps life scientists, teachers and students to deal with whole-slide (histopathology) images (WSI). A WSI refers to a digital scan of a section of some processed specimen placed onto glass slides. Cytomine provides tools for labelling and annotating this kind of images as it can be seen in Figure 1.1.

Nevertheless, it can happen that several consecutive sections of a tissue have to be visually compared and/or annotated. These sections may be stained with different products. The staining process helps to enhance contrast for a better examination and results in different colored appearances. Figure 1.2 shows the visualisation of multiple stained sections with

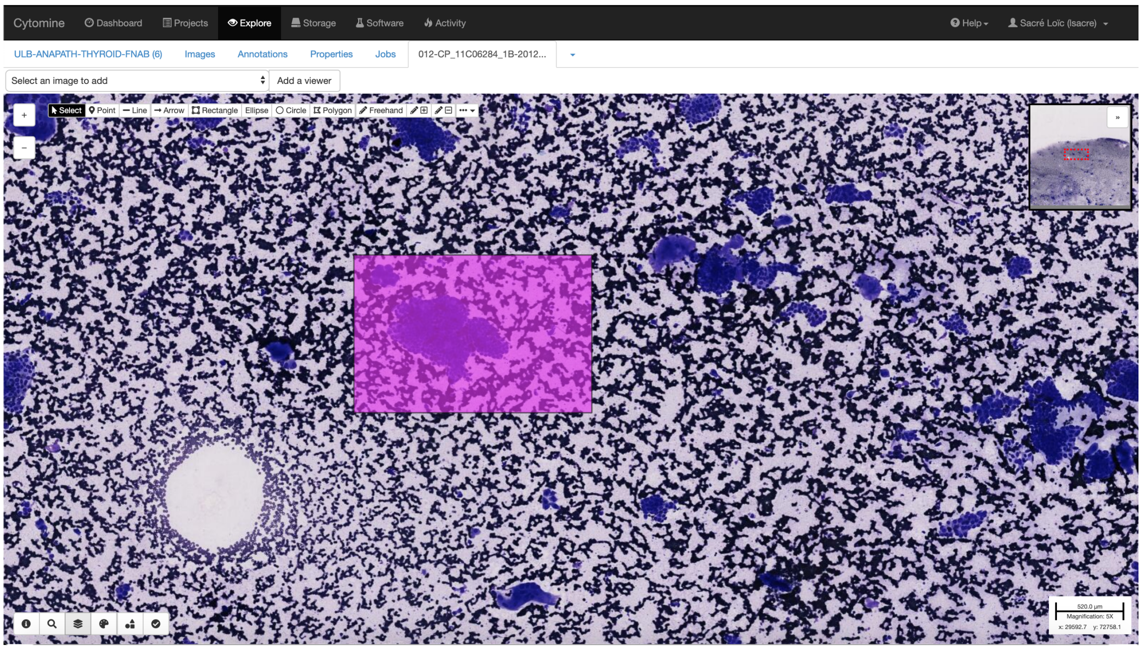


Figure 1.1: Bounding box annotation in the Cytomine Software

polygonal annotations in the Cytomine software.

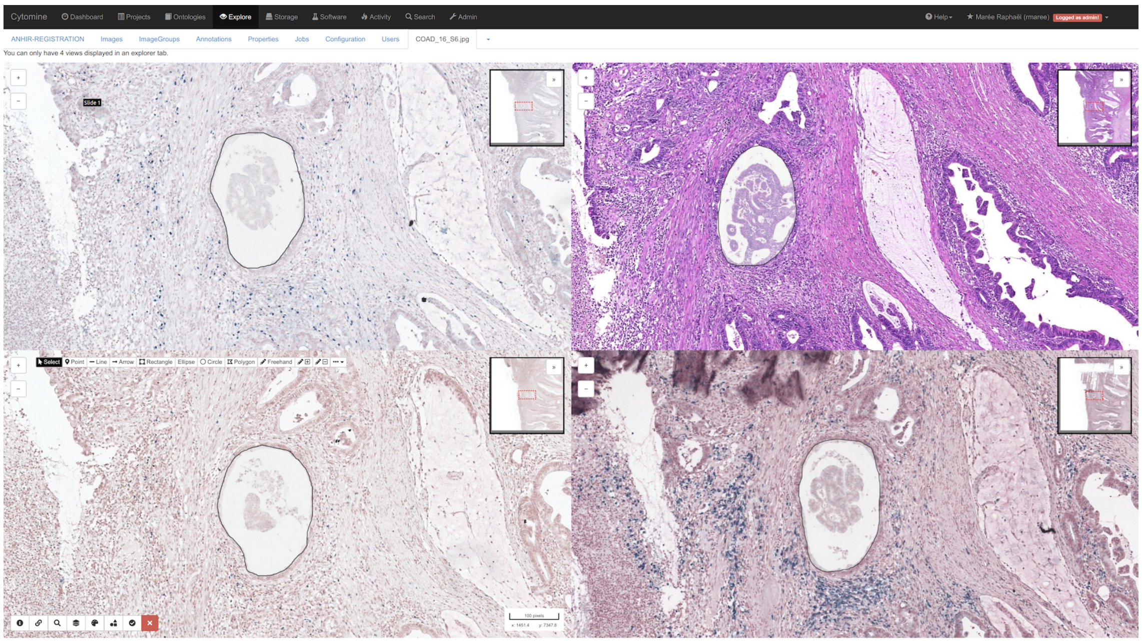


Figure 1.2: Visualisation of multiple stained sections with polygonal annotations in the Cytomine software

Manual annotation is extremely time consuming and could be even worse if the same structure have to be searched in different tissue sections. This is why it is imperative to develop intelligent tools for automating the process. Figure 1.3 illustrates a use-case of automatic annotation of two tissue sections with different stains. The left part of the figure (pink-colored section) is manually labelled thanks to a polygon tool while the right part of the figure (purple-colored section) illustrates an automatic labelling construction thanks to some software. This work aims at investigating different deep learning strategies to perform patch-matching in order to ease annotation across multiple tissue sections.

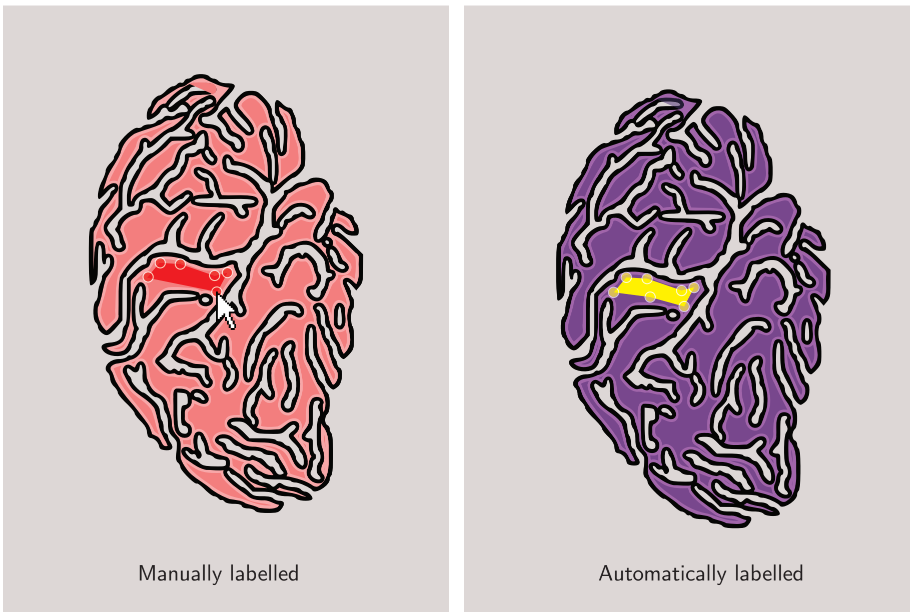


Figure 1.3: Use-case of automatic annotation with samples in multiple stains

Chapter 2

Problem Specifications and Dataset

In this chapter, the problem specifications and goals are going to be clearly exposed. Then, the dataset used for this study is going to be described, pointing out the choices made here.

2.1 Problem Specifications

This thesis is a preliminary study of methods distinct from image registration for automatic annotation of pathology images. Image registration is "the process of transforming different sets of data into one coordinate system"[6]. In other words, it is the process of aligning two identical or similar objects coming from two different images. The proposed and investigated approach here is to use **patch-matching** in order to obtain similarity likelihood. In more details, it consists in extracting a cropped window (called a patch) around a point of interest in a reference image and identifying the corresponding patch in a target image by making patch-to-patch comparison. A patch is exactly like a bounding-box annotation. This comparison is performed using deep neural networks. They will be introduced later on. Note that a pair of patches is said to be **similar** if the patches correspond to the same structure in the reference and target image. Otherwise, it is referred to as **dissimilar**.

Figure 2.1 illustrates the process of visualizing likely positions of a structure of reference from an image into a target one thanks to the use of a heat map. In the reference image, a patch is extracted (the blue box). In the target image, green colored patches indicate the patches that correspond the most likely to the one of reference (the more opaque, the better).

Even if the process seems very easy at first glance, the reality is far from being the case. Let's illustrate why with a simple example. Take a sphere and cut successively some slices from top to bottom. Each slice is a circle and the radius will grow until the middle of the sphere and start decreasing afterwards. The content of the slices is different and this is exactly what happens when sectioning a tissue.

In addition, due to processing of tissue and a pre-analytical phase, sections may also experience non-linear deformations. Factors such as the stain and the orientation of the tissue in the image also contributes to make the process of identifying similar structures challenging.

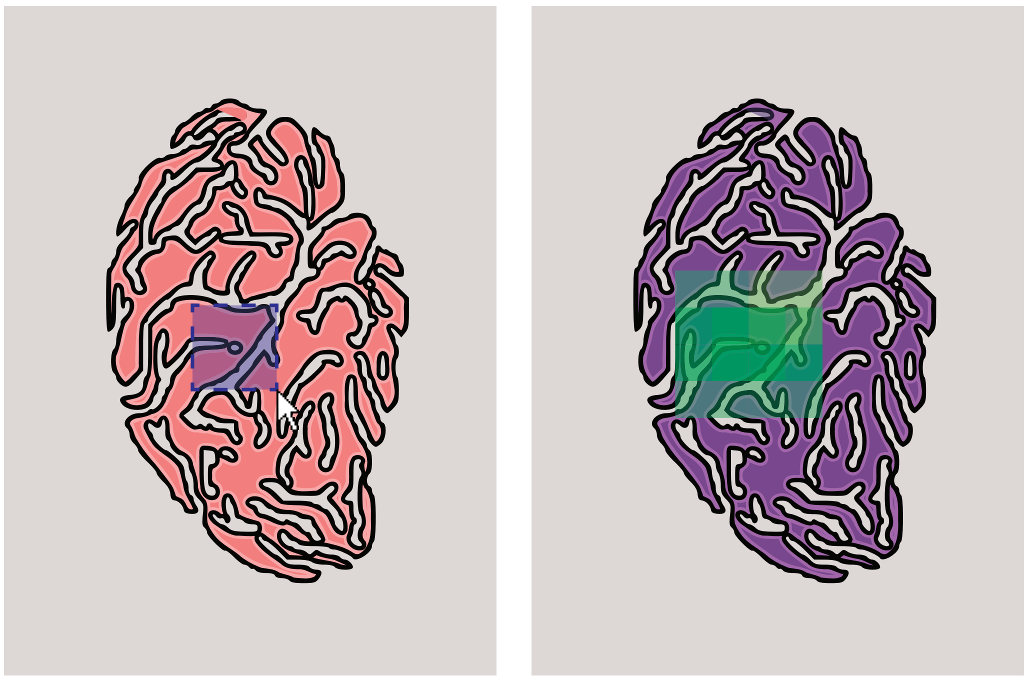


Figure 2.1: Likelihood similarity visualisation for a reference patch in a target image (left: reference - right: target)

2.2 Dataset

In order to study the patch matching approach, the dataset coming from the Automatic Non-rigid Histological Image Registration (ANHIR) challenge[7, 8, 9, 10, 11] is used. It has been chosen because it contains the position of matching structures across several sections of a same tissue.

The ANHIR challenge aims at the automatic nonlinear image registration of 2D microscopy images of histopathology tissue. The dataset consists of high-resolution (up to 40x magnification) whole-slide images of different types of tissue.

The images are organized in sets of consecutive sections of a same tissue where each section is stained with a different dye. An analogy can be the slices obtained after carving a ham. Significant structures are marked in the tissue with landmarks. These were manually identified in each image, with correspondences within each set. The tissues have been scanned using different magnitudes and resolutions (μm/pixel). A summary of the different tissues with their respective magnitude and resolution is given in Table 2.1.

The different dyes present in the dataset are:

- CC10 : Clara cell 10 protein
- CD31 : Platelet endothelial cell adhesion molecule (PECAM-1)
- CD4/CD8 : Cluster of differentiation 4/8
- EBV : IHC-staining for LMP-1 protein (Dako, clone CS.1-4) for Epstein-Barr virus
- ER : Estrogen receptor
- HE : Hematoxylin and eosin (H&E)
- HER2 : Human epidermal growth factor receptor 2 : HER2 (c-erbB-2/HER-2-neu)
- KI67 : Antigen KI-67

Name (Tissue)	**Magnitude**	**Resolution** (μm/pixel)	**Scale** (%)
lung-lesion (Lung lesion)	40x	0.174	50
lung-lobes (Whole mice lung lobes)	10x	1.274	100
mammary-glands (Mammary glands)	10x	2.294	25
mice-kidney (Mice kidney)	20x	0.227	25
COAD (COlon ADenocarcinoma (colon cancer))	10x	0.468	25
gastric (Gastric mucosa and gastric adenocarcinoma tissue fragments)	40x	0.2528	15
breast (Human breast)	40x	0.2528	20
kidney (Human kidney)	40x	0.2528	25

Table 2.1: Magnitude, resolution and scale of the images used for the different tissues

- MAS : Masson's trichrome stain
- PAS : Periodic acid-Schif
- PR : Progesterone receptor
- PRO-SPC prosurfactant protein C
- SMA : Smooth Muscle Actin
- S1 to S8: dye definition is unspecified but contains stains for the immune response (including CD4, CD68) and hypoxia as well as H&E

Note that the dyes denoted with Si ($i \in [1, ..., 8]$) are not stated. They are the dyes used to stain the COAD samples. This is problematic for making analysis on the different dyes of the dataset.

It can happen that an identical dye exists for different sections of the same tissue. For example, it is the case for the mice kidney samples. In this case, a hyphen followed by a number is suffixed to the name of the dye (e.g. PAS-1). Furthermore, as there are several samples corresponding to the same type of tissue, a hyphen followed by a number is also suffixed at the end of the name of tissue. These numbers for the tissues are the one given in the original dataset.

To ease the reading, some notations are introduced for referring to the samples from the dataset:

- $\tau_{\text{DYE}}^{\text{Tissue}}$: to refer to a tissue stained with a certain dye DYE
- $\tau_{\text{DYE}}^{\text{Tissue}}(i)$: to refer to a specific landmark i belonging to a tissue stained with a certain dye DYE

The whole dataset contains in total 53 different tissues, 151 images and 12736 landmarks. There are 191 pairs of slices of tissue and 16839 pairs of landmarks. Because our aim is to evaluate patch matching, this image registration dataset had to be converted into a

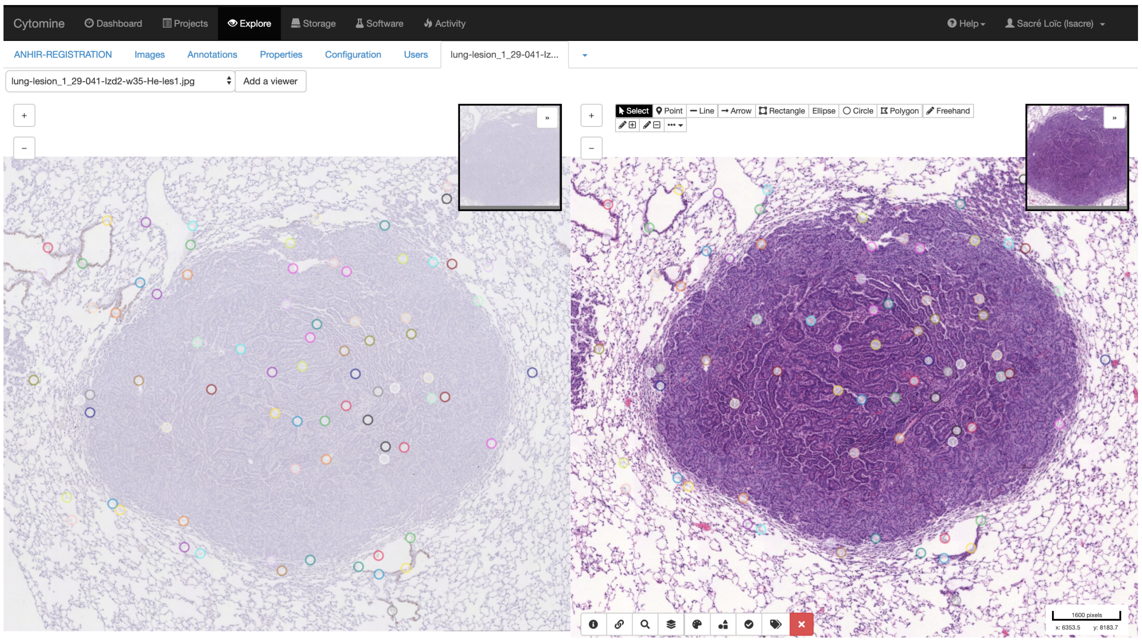


Figure 2.2: Two tissue slices $\tau_{\text{CC10}}^{\text{Lung lesion-1}}$ (left) and $\tau_{\text{HE}}^{\text{Lung lesion-1}}$ (right)

relevant format. For each landmark, a cropped window of the whole-slide image around the landmark has been extracted. This window will be called a patch in the rest of the document. The patches are of size 300×300 pixels (even if 100×100 and 600×600 patches will be used for some comparisons).

Examples of two tissue sections stained with a different dye as it can been seen in the Cytomine web software are shown in Figure 2.2. All the colored dots indicate a landmark. As it can be seen, corresponding (matching) landmarks are in the same color.

2.3 Choices Relative to the Dataset

For the ANHIR challenge certain images have been down-scaled for convenience and for keeping a similar size range. Even if sometimes they provide images at different scales, there are cases where the full scale one is even not available. In Table 2.1 are listed the scale that are used in the challenge and in this work. Furthermore, one may argue that the quality and quantity of the images content can vary due to the different magnitudes, resolutions and scales. Nevertheless, all patches during the experiments are extracted in the same way, i.e. by taking a window of a specific size around the landmark without caring about the resolution, magnitude or scale, to keep coherency. It still provides a lower bound clue on the performance. Resizing or even adapting the size of the patches, independently from the tissue, such that they all have the same resolution (i.e. that n pixels must correspond to the same length in the real specimen) is challenging and is not straightforward. To illustrate the problem, let's take two tissue samples with both 10x magnitude: $\tau_{\text{CC10}}^{\text{Lung lobes-2}}$ (the reference image) and $\tau_{\text{S7}}^{\text{COAD-1}}$ (the image whose size of the patches has to be adapted). Note that the images of COAD samples are only available at a scale of at most 25%. In order to obtain the adapted size of the patches, the following formula is used to compute the side size of the patch to extract (for a same magnitude):

$$s_{adapt} = s_{ref} \times \frac{res_{adapt}}{res_{ref}} \times \frac{scale_{adapt}}{scale_{ref}}$$

where s corresponds to the size of the side of the patch in pixels (in this case 300), res

(resp. *scale*) is the resolution (resp. scale) of the reference (resp. adapted) image denoted as ref (resp. $adapt$). For the $\tau_{\text{S7}}^{\text{COAD-1}}$ sample, the resulting size of the patches to be extracted is thus $300 \times \frac{0.468}{1.274} \times \frac{25}{100} = 27.55$ (~ 28) pixels.

Figure 2.3 shows a 300×300 pixels patch from the reference $\tau_{\text{CC10}}^{\text{Lung lobes-2}}$. It can be noticed that the content of the patch is rich enough to be identified in another section. Figure 2.4 illustrates the size of a patch from $\tau_{\text{S7}}^{\text{COAD-1}}$ when the size have been adapted according to the reference $\tau_{\text{CC10}}^{\text{Lung lobes-2}}$. The patch is indicated by a red arrow. In this case, it is clear that the content of the patch is far too small in comparison to the rest of the tissue structures. It comes from the fact that from a tissue to another, the real size of a significant structure may differ.

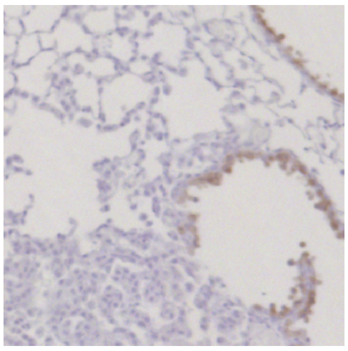


Figure 2.3: A 300×300 pixels patch from $\tau_{\text{CC10}}^{\text{Lung lobes-2}}$ (the reference)

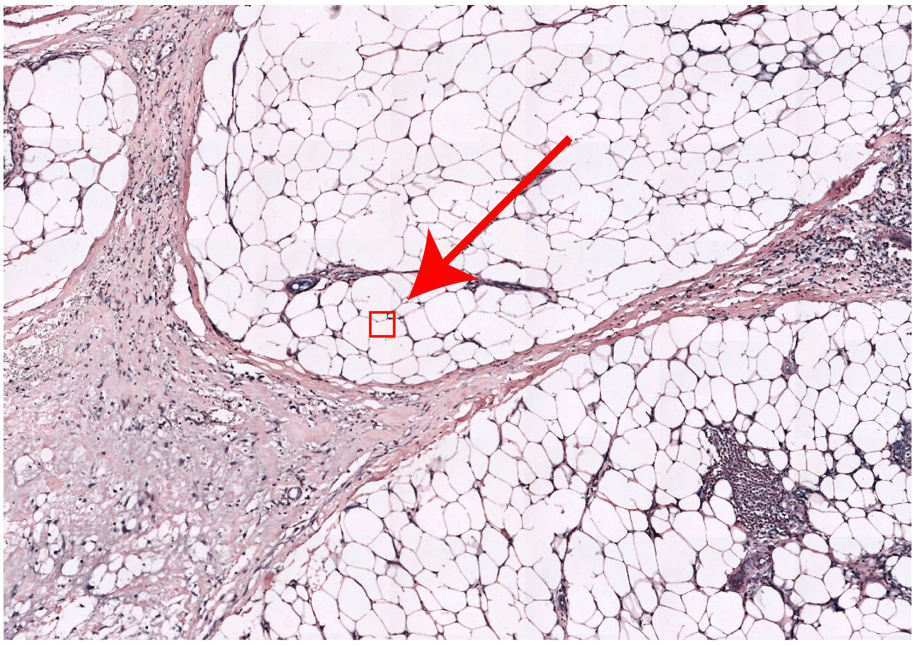


Figure 2.4: Illustration of a patch from $\tau_{\text{S7}}^{\text{COAD-1}}$ when its size is adapted according to the reference $\tau_{\text{CC10}}^{\text{Lung lobes-2}}$

Finally, some examples of **similar** patches are shown in Figure 2.5. Each group of patches is separated with a black horizontal line (the image has to be looked from top to bottom, then left to right). As it can be seen, the number of dyes vary from one tissue to another (there are between two and seven dyes per tissue). Furthermore, one can notice that the color as well as the patch content from one dye to another may strongly vary, which makes the task even more challenging.

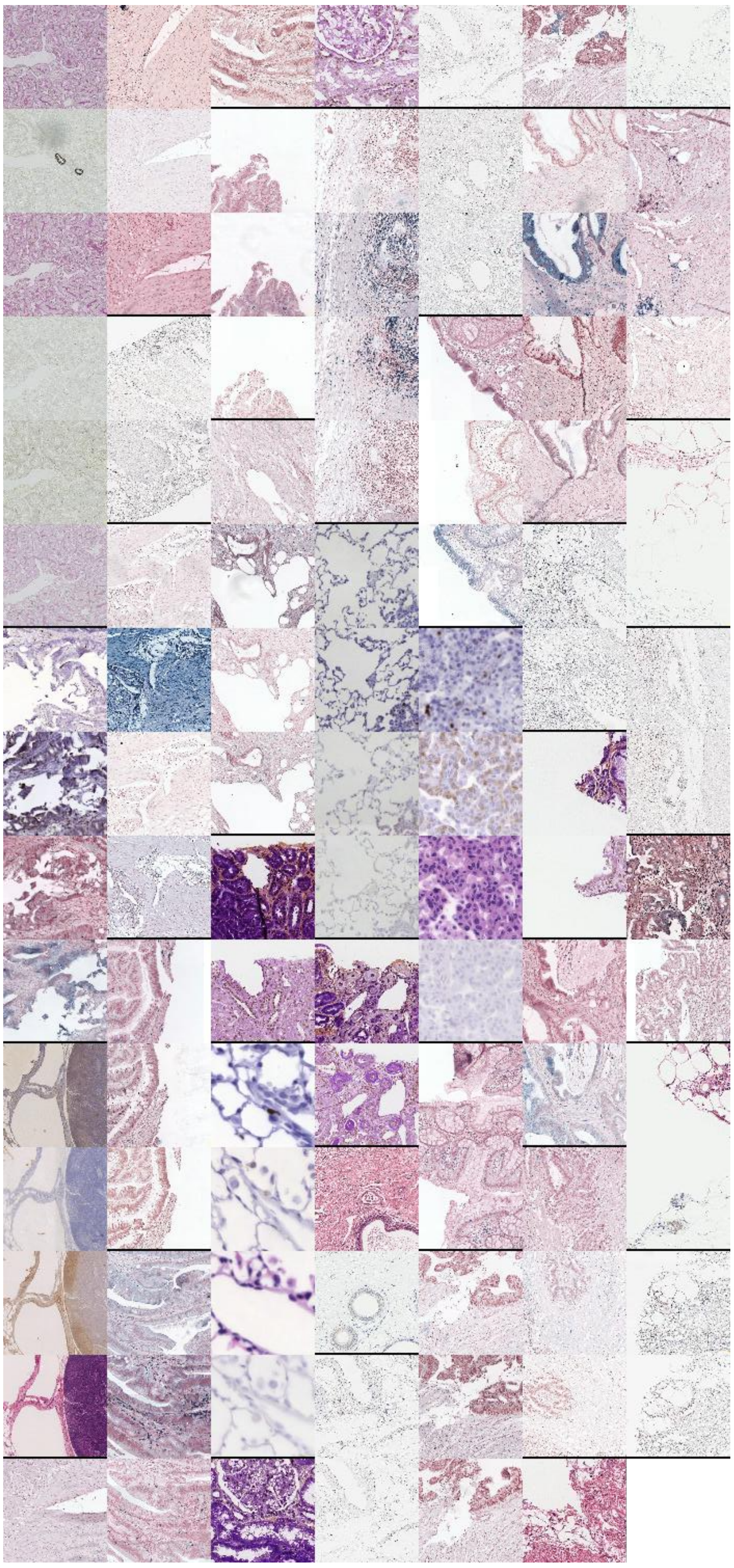


Figure 2.5: Examples of **similar** patches

Chapter 3

Theoretical Background

In this chapter, the theoretical background necessary to fully understand the approaches and strategies used in this work is given. The chapter is organized as follows. Firstly, the main concepts of machine learning and deep learning are going to be introduced. Then, state-of-the-art methods for image patch matching are going to be given. Finally, evaluation metrics used to assess the performance of the different approaches will be stated and described.

3.1 Machine Learning

Machine learning is a subset of artificial intelligence which involves any technique trying to mimic human behaviour. Machine learning is concerned with the use of statistical methods and the development of algorithms that allow the machine (i.e. the computer) to learn automatically from data and improve with experiences (by collecting more data).

3.1.1 Supervised and Unsupervised Learning

When working with machine learning algorithms, there are two main types of settings: supervised and unsupervised learning. There also exists semi-supervised learning which is a mix of both.

Supervised learning: the learning process is exclusively performed thanks to a dataset of labelled training examples. Formally, let $D = \{(\mathbf{x}_i, y_i), i \in [1, .., n]\}$ and $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ be a set of n independent and identically distributed (i.i.d.) pairs of training samples. $\mathbf{x}_i$ is the i^{th} input and y_i is the corresponding output. A machine learning algorithm tries to find a function $f^{\psi} : \mathcal{X} \rightarrow \mathcal{Y}$ ($f^{\psi} \in \mathcal{F}$) which minimizes the expectation of some loss function $\mathcal{L}$ over the joint distribution of input/output pairs:

$$\mathbb{E}_{\mathbf{x},y}[\mathcal{L}(f(\mathbf{x}), y)] \tag{3.1}$$

$\mathcal{L} \geq 0$ measures how close $f(\mathbf{x})$ is to y and $\mathcal{F}$ is the hypothesis space. Equation 3.1 is referred to as the expected risk. As the dataset is finite, the empirical risk minimization principle is used instead:

$$f_D^{\psi} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{(\mathbf{x}_i, y_i) \in D} \mathcal{L}(f(\mathbf{x}_i), y_i) \tag{3.2}$$

In the case of supervised learning, there exists a wide range of algorithms such as decision trees[12], k-nearest neighbours (k-NN)[13], artificial neural networks (deep learning)[14], support vector machines (SVM)[15], etc.

Unsupervised learning: on the other hand, in unsupervised learning the dataset consists of unlabelled training examples.

3.1.2 Regression and Classification

The type of problems that machine learning deals with can be categorised as regression and classification.

Regression: in the case of a regression problem, the output is numeric. The domain of the pairs of the dataset D is therefore $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \mathbb{R}$. A machine learning algorithm searches for the mapping $\mathcal{X} \rightarrow \mathcal{Y}$ such that an estimate for a new $\mathbf{x}$ is

$$\mathbb{E}[Y|X = \mathbf{x}]$$

Classification: in the case of a classification problem, the output is a class (for instance the output could be one of several animal species). The domain of the pairs of the dataset D is therefore $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \{1, ..., q\}$ (where q is the number of possible labels). A machine learning algorithm searches for the mapping $\mathcal{X} \rightarrow \mathcal{Y}$ such that for a new $\mathbf{x}$

$$\underset{y}{\operatorname{argmax}}\, P(Y = y|X = \mathbf{x})$$

The hypothesis space $\mathcal{F}$ will depend on the algorithm used to learn the most appropriate model, i.e. the function f which fits the best to the data.

3.1.3 Overfitting and Underfitting

During the training, it can happen that the learned model fits too well the training data and misses to generalize well to other input data. In that case, the network is said to be overfitting. On the other hand, when the model fits too poorly to the training data the model is said to be underfitting.

Let $\mathcal{Y}^{\mathcal{X}}$ be the set of all functions $f : \mathcal{X} \rightarrow \mathcal{Y}$. Then, the Bayes model f_B (the best model that can be obtained) is defined as:

$$f_B = \underset{f \in \mathcal{Y}^{\mathcal{X}}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x},y}[\mathcal{L}(f(\mathbf{x}), y)]$$

A model is said to be underfitting if

$$|\mathbb{E}_{\mathbf{x},y}[\mathcal{L}(f(\mathbf{x}), y)] - \mathbb{E}_{\mathbf{x},y}[\mathcal{L}(f_B(\mathbf{x}), y)| \gg 0$$

for any $f \in \mathcal{F}$. This means that the difference is large. When f_D^{ψ} is too specialised, then it is possible that:

$$\mathbb{E}_{\mathbf{x},y}[\mathcal{L}(f_B(\mathbf{x}), y)] \geq \frac{1}{n} \sum_{(\mathbf{x}_i, y_i) \in D} \mathcal{L}(f_D^{\psi}(\mathbf{x}_i), y_i)$$

f_D^{ψ} is thus overfitting the data.

3.1.4 Transfer Learning

Transfer learning is commonly used in machine learning. It consists in using the knowledge learnt from one task and applying it to a second one. A domain is a tuple of two elements, the feature space $\mathcal{X}$ and a marginal probability, $P(X), X = \{x_1, x_2, ..., x_n\} \in \mathcal{X}$. Let $\mathcal{Y}$ be the output space. A task T is learnt from the training data consisting of pairs $(x_i, y_i) \in \mathcal{X}_T \times \mathcal{Y}_T$ and is defined by $(\mathcal{X}_T, \mathcal{Y}_T, P_T(X, Y))$. The goal of transfer learning is solving a target task $(\mathcal{X}_t, \mathcal{Y}_t, P_t(X, Y))$ by using the source data $(\mathcal{X}_s, \mathcal{Y}_s, P_s(X, Y))$. The condition has to at least be $\mathcal{X}_s \neq \mathcal{X}_t$, $\mathcal{Y}_s \neq \mathcal{Y}_t$ (inductive learning) or $P_s \neq P_t$ (domain adaptation).

Note that $P_s(X, Y)$ and $P_t(X, Y)$ can differ in multiple ways:

- $P_s(X) \neq P_t(X)$ but $P_s(Y|X) = P_t(Y|X)$ (it is referred to as covariate shift)
- $P_s(X) = P_t(X)$ but $P_s(Y|X) \neq P_t(Y|X)$ (it is referred to as prior probability shift)
- otherwise it is referred to as general domain adaptation

3.2 Deep Learning

3.2.1 Artificial Neural Networks

Artificial neural networks are the most well-known and basic network architecture used in deep learning. They are composed of artificial neurons (also known as perceptrons[14]). An artificial neuron is a function $\mathbb{R}^p \to \mathbb{R}$ that takes as input a vector $\mathbf{x} \in \mathbb{R}^p$ and outputs a value $y \in \mathbb{R}$. The function consists in a weighted sum ($W \in \mathbb{R}^p$) and is often followed by a non-linear operation φ:

$$y = \varphi(W^T\mathbf{x}) = \varphi(\sum_{i=0}^{p-1} w_i x_i)$$

The first element x_0 of $\mathbf{x}$ is usually set to 1 such that with w_0 they form a bias. An artificial neuron can be visually expressed like a biological one as it can be seen in Figure 3.1.

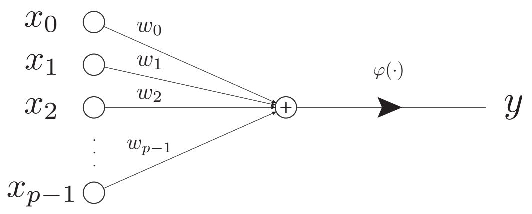


Figure 3.1: Artificial Neuron

There exists a wide variety of choices for the φ function. The most common ones are the hyperbolic tangent (tanh), the sigmoid (σ) or the rectified linear unit (ReLu), among others:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.3}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

$$\text{ReLu}(x) = \max(0, x) \tag{3.5}$$

The multi-layer perceptron (MLP) consists of at least an input layer, a hidden layer and an output layer. A layer is composed of parallel neurons (nodes) and each neuron in one layer is connected to all neurons in the next layer. A layer $\mathbf{h}$ generalizes the formula of a single neuron to form multiple q outputs with $W \in \mathbb{R}^{p \times q}$ and a bias term $b \in \mathbb{R}^q$:

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

where φ is now an element-wise function. For a MLP with L layers, the input layer $\mathbf{x} = h_0$, the hidden layers are $h_i, i \in [1, ..., L-2]$ and the output layer $\mathbf{y} = h_{L-1}$. A four-layer MLP is illustrated in Figure 3.2.

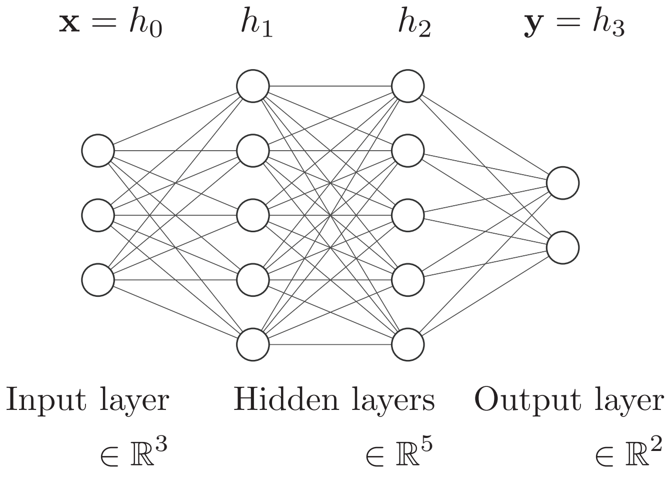


Figure 3.2: Fully-connected feedforward neural network

This model is also known as the fully-connected feedforward neural network and has parameters $\boldsymbol{\theta}$ (the weights and bias of all layers) that can be learnt thanks to a popular algorithm called back-propagation.

3.2.2 Training Neural Networks

Before describing the back-propagation algorithm, it is necessary to introduce the optimization problem behind MLP's. The focus is put on supervised learning with classification problems to explain the main principles.

To determine how well a function fits the data, a loss (or cost) function is used. Let $\hat{y} = f(\mathbf{x})$ be the output of neural network and y be the true label, the loss can for example be the 0-1 loss function

$$\mathcal{L}(\hat{y}, y) = \mathbb{1}(\hat{y} \neq y)$$

where $\mathbb{1}$ is the characteristic function or the Binary Cross-Entropy loss (BCE):

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y})) \tag{3.6}$$

The back-propagation algorithm is an algorithm that allows to efficiently compute the gradients of the loss with respect to all the parameters of a neural network. Back-propagation is following a gradient descent approach and is based on the chain rule. The gradient descent is an iterative optimisation algorithm which uses a first-order approximation of the gradient to find a local minimum (or a maximum depending on the step performed). The parameters can then be updated with:

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta \mathcal{L}(\theta_t)$$

where γ is the learning rate. Stochastic gradient descent (SGD)[16] follows the same principle but instead of evaluating over the whole dataset D, the evaluation is performed on a single sample. In the case of SGD, the gradient loss is therefore:

$$\nabla\mathcal{L}(\theta) = \frac{1}{n}\sum_{(\mathbf{x}_i, yi)\in D} \nabla l(y_i, \hat{y_i}) \text{ with } \hat{y_i} = f(\mathbf{x}_i;\theta)$$

A compromise between computing the true gradient and performing SGD is to use a *mini-batch.*

To fully understand how the back-propagation works, it remains to introduce the chain rule. The chain rule is a formula based on the composition of two functions which states:

$$(f \circ g)' = (f' \circ g)g'$$

Thus, for a function such as $f(g(x))$ with $g(x) = (g_0(x), g_1(x), ..., g_{n-1}(x))$ then the chain rule generalizes to:

$$\frac{df}{dx} = \sum_{i=0}^{n-1} \frac{\partial f}{\partial g_i}\frac{dg_i}{dx}$$

The computational graph of a simple function f with two inputs x and y and one output z is shown in Figure 3.3. During the forward pass, z is just the result of $f(x, y)$. During the backward pass, the gradient of the loss $\mathcal{L}$ with respect to x and y can be computed easily. Obviously, this can be generalized to more complex graphs.

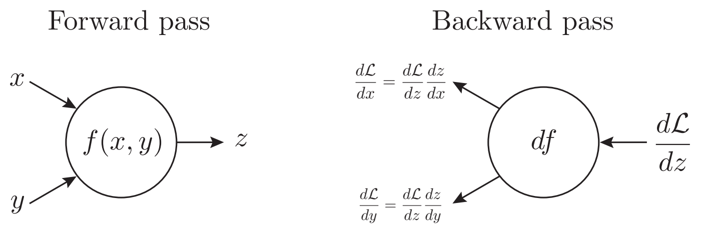

Forward pass
Backward pass
x
y
$f(x,y)$
z
$\frac{d\mathcal{L}}{dx} = \frac{d\mathcal{L}}{dz}\frac{dz}{dx}$
$\frac{d\mathcal{L}}{dy} = \frac{d\mathcal{L}}{dz}\frac{dz}{dy}$
df
$\frac{d\mathcal{L}}{dz}$

Figure 3.3: Forward and Backward pass

3.2.3 The Vanishing Gradient

A common problem occurring while training a neural network is the vanishing gradient. As the name suggests it, the gradient becomes so small that the evolution of the parameters stagnates. This effect worsens as the depth of the network (i.e. the number of layers) increases. For instance, the derivative of the sigmoid function (Equation 3.4) is

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

and is bounded between 0 and 1/4. If the weights are assumed to have a mean of 0 and a unit variance, they are then bounded between -1 and 1. Thus, for a fully-connected network with L layers (assuming that the input and weights $w_i \in \mathbb{R}$ and that there is no bias for simplicity):

$$\begin{aligned}
a_1 &= w_1 x \\
a_i &= w_i z_{i-1} \\
z_i &= \sigma(a_i) \\
\hat{y} &= z_{L-1} = \sigma(a_{L-1}) \\
\frac{d\hat{y}}{dw_j} &= \prod_{i=j+1}^{L-1} \frac{\partial z_i}{\partial a_i}\frac{\partial a_i}{\partial z_{i-1}} \times \frac{\partial z_j}{\partial a_j}\frac{\partial a_j}{\partial w_j} \\
&= \prod_{i=j+1}^{L-1} \underbrace{\frac{\partial \sigma(a_i)}{\partial a_i}}_{\leq 1/4} \underbrace{w_i}_{\leq 1} \times \underbrace{\frac{\partial \sigma(a_j)}{\partial a_j}}_{\leq 1/4} \frac{\partial a_j}{\partial w_j}
\end{aligned} \tag{3.7}$$

As it can be seen in Equation 3.7, $\frac{d\hat{y}}{dw_j}$ can quickly vanish to 0 as the number of layers grows. One solution to the problem is to use the ReLu (Equation 3.5) activation instead of the sigmoid. Indeed, the derivative of the ReLu activation is 1 when the input is positive and 0 otherwise.

3.2.4 Momemtum

When using the SGD, the value of the gradient may be too small, slowing down the training process. This is what momentum aims to overcome by adding some inertia into the process. The network parameters θ are updated as follows:

$$\begin{aligned}
v_t &= \alpha v_{t-1} - \gamma g_t \\
\theta_{t+1} &= \theta_t + v_t
\end{aligned}$$

The variable v_t is the velocity. It is a weighted sum over all the past gradients. The weights decrease exponentially:

$$\begin{aligned}
v_t &= \alpha v_{t-1} - \gamma g_t \\
&= \alpha^2 v_{t-2} - \alpha\gamma g_{t-1} - \gamma g_t \\
&\dots \\
&= -\gamma(\alpha^{t-1} g_1 + \dots + \alpha g_{t-1} + g_t)
\end{aligned}$$

The parameter α satisfies $0 \leq \alpha \leq 1$ and is usually set to 0.9. When $\alpha = 0$, momentum is equivalent to a traditional SGD.

3.2.5 Early Stopping

There exists some ways to reduce overfitting with neural networks. Early stopping is one of them. In order to properly train the network, the dataset is split into three subsets:

1. **Training set**: This set is the data which are used to adjust the parameters of the model.
2. **Validation set**: this set is used to prevent overfitting and underfitting. It serves as a reference for the training. If the error on the training set still continues to decrease while the one on the validation set increases, then training must stop. This process is referred as early stopping and is illustrated in Figure 3.4 (the image comes from one of the training with a neural network in Section 4.3). Before the red dotted line,

the model is said to be overfitting while after it is said underfitting. It is all about finding a good balance.

3. **Testing set**: This set serves for the final evaluation of the model.

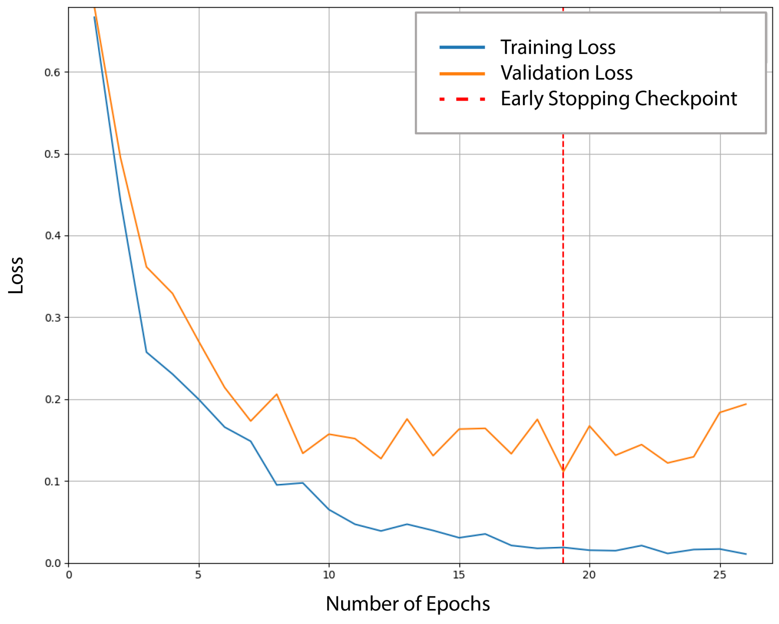


Figure 3.4: Illustration of early stopping

3.2.6 Dropout

Another way to prevent overfitting is the use of dropout[17]. It consists in freezing a random p fraction of the nodes in a layer during the training phase, making the network more robust. During the testing, all the activations are used but reduced by a factor p.

3.2.7 Batch-Normalization

Training neural networks can be difficult due to the fact that the activations of a layer is the input of the next layer (except for the output layer). Therefore, the input distribution of a layer is changing at each forward pass forcing the layer to constantly adapt itself. This phenomenon is known as covariate shift. To help overcoming this issue, the authors of [18] introduce a technique known as Batch-Normalization. It consists in shifting and re-scaling by respectively the mean and variance computed on the batch during the forward pass in the network. The full algorithm is given in Figure 3.5.

3.2.8 Convolutional Neural Networks

Convolutional Neural Networks (CNN)[19] are neural networks suitable for visual imagery analysis. The two main building blocks of a CNN are convolutional and pooling layers and it usually ends up onto a fully-connected network (see Figure 3.6). CNNs are very good at capturing spatial information in contrast to traditional neural networks.

Convolutional layers make use of a convolution operation which is based on the cross-correlation operator $\star$. For two-dimensional tensors (for example grayscale images), given an input vector $\mathbf{x} \in \mathbb{R}^{W\times H}$ and a convolutional kernel $\mathbf{k} \in \mathbb{R}^{w\times h}$, the discrete convolution

```
Input:  Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
        Parameters to be learned: $\gamma$, $\beta$
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

    $\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$                                  // mini-batch mean
    $\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$      // mini-batch variance
    $\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$   // normalize
    $y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$        // scale and shift
```

Figure 3.5: Batch Normalizing Transform, applied to activation x over a mini-batch [18]

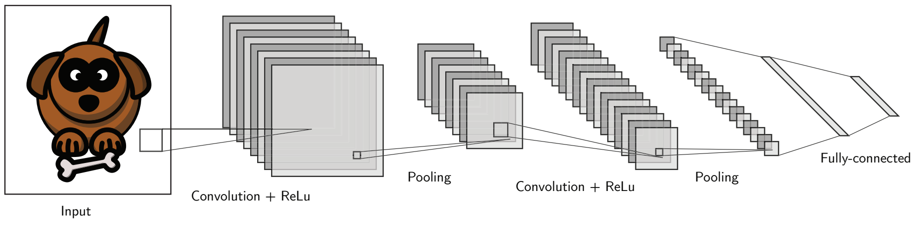


Figure 3.6: Typical CNN

$\mathbf{k} \star \mathbf{x}$ is a vector of size $(W - w + 1) \times (H - h + 1)$ such that

$$(\mathbf{k} \star \mathbf{x})[i, j] = \sum_{m=0}^{w-1}\sum_{n=0}^{h-1} k_{m,n}\mathbf{x}_{m+i,n+j}.$$

For three-dimensional tensors (for example colored images), i.e. $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$,the convolution operation is applied to the $C = 3$ channels and then the local results are summed up in the following way:

$$(\mathbf{k} \star \mathbf{x})[i, j] = w_0 + \sum_{c=0}^{C-1}(\mathbf{k} \star \mathbf{x}_{:,:,c})[i, j].$$

where w_0 and $\mathbf{x}_{:,:,c}$ corresponds respectively to some bias and to the c^{th} channel matrix of $\mathbf{x}$. The output tensor of a convolution is called a feature-map. If K kernels (also called filters) are applied, then the dimensions of the resulting tensors are $(W - w + 1) \times (H - h + 1) \times K$.

An issue that can happen when applying convolutions with a small kernel is losing information on the border of the input. A solution to this is to use zero-padding. It consists in adding p extra 0 all around the input. The use of stride is also very common. It corresponds to the shift of the kernel over the input (for a basic convolution the shift is equal to 1).

Pooling layers are often placed after convolutional layers or the non linearity. Pooling is a down-sampling operator. They progressively reduce the spatial size, decreasing the number of parameters in the convolution. There are two main types of pooling layers, max-pooling layers and average-pooling layers even though other operations such as L2-norm or sum can be performed. Considering a pooling area of size $h \times w$, the max-pooling operation consists in taking the maximum element within the area while average-pooling consists in taking the mean over the area. A pooling layer can be adaptive, this means that instead of specifying the size of the pooling area only the size of the output feature map is required. The size of the pooling window is then automatically computed. Adaptive pooling layers can be useful before an ending fully-connected network, in order to be able to use different data input size without changing the architecture of the CNN.

After each convolutional layer, ReLu (Equation 3.5) activation layer is conventionally used. It allows to introduce non-linearity within the network. ReLu had been shown to be more computationally effective than tanh (Equation 3.3) or sigmoid (Equation 3.4) function without significant change in accuracy.

3.2.9 ImageNet and Winner Networks

ImageNet[20] is "an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images"[21]. They do not own the images, they just provide thumbnails and URLs of the images. This database is designed for use in visual object recognition software research. Furthermore, from 2010 to 2017, the ImageNet project ran an annual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with two main competitions, object detection and localization.

Over the years, convolution neural networks showed themselves to be state-of-the-art for this challenge. Some of them are going to be briefly described (AlexNet, 2012[22], VGGNet, 2014[23], ResNet, 2015[24] and DensetNet, 2016[25])

These networks pre-trained on the ImageNet database are frequently used for transfer learning (see Section 3.1.4). They provide relatively good feature extractors with the advantage of not being trained again from scratch, which could be highly time-consuming and resources expensive.

AlexNet was designed by the SuperVision group. The architecture is not very complex and is mainly composed of 11×11, 5×5, 3×3, convolutions and max pooling operations. ReLU is used after each convolutional and fully-connected layers. They used dropout and data augmentation to reduce overfitting during the training.

VGGNet was introduced by Simonyan and Zisserman in 2014. It is known for its simplicity and depth (16 and 19 layers was considered as deep by then). It only uses 3×3 convolutional layers and pooling layers stacked onto each other.

ResNet was introduced by He et al. in 2015. Apart from obtaining a higher accuracy in image classification tasks, they accelerated the speed of training of the deep networks as well as reducing the effect of the vanishing gradient (see Section 3.2.3). Last but not least, this new network architecture addresses the counter-intuitive phenomenon known as the degradation problem. In short, experiments showed that stacking identity layers (i.e. $f(x) = x$) at the end of a network, thus adding depth to it, leads to a decrease in performance despite the fact that the identity function should have no effect on the results (as it just maps the input to the output). They introduced the deep residual learning framework. Instead of learning a direct mapping

$$x \to H(x),$$

the network learns a residual mapping

$$F(x) := H(x) - x$$

To do so, they introduced the building block of residual learning which contains a skip identity connection (see Figure 3.7). Therefore, for an image x_0, a non-linear transformation $H_l(\cdot)$ (l is the layer index), the information flow is given by

$$x_l = H_l(x_{l-1}) + x_{l-1}$$

in contrast to tradition convolutional feed-forward network where the x_{l-1} term is not present.

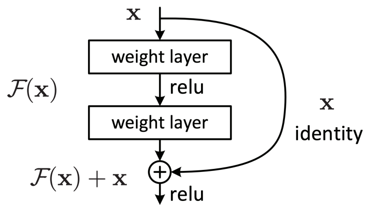

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3.7: Residual learning: a building block (source [24])

Deep residual networks have been shown to be particularly good at lots of computer vision tasks. In 2015, ResNet won the 1st places in all five main tracks of ILSVRC and COCO Competitions (ImageNet classification, ImageNet detection, ImageNet localization, COCO detection and COCO segmentation)

DenseNet was introduced by Huang et al. in 2016. The idea behind the Dense Convolutional Network framework is that each layer passes its own feature-maps to all the following layers. To improve the information flow between layers, the l^{th} layer receives the feature-maps of all preceding layers, $x_0, ..., x_{l-1}$, as input

$$x_l = H_l([x_0, x_1, ..., x_{l-1}])$$

where $[x_0, ..., x_{l-1}]$ is the concatenation of the feature-maps from in layers $0, ..., l-1$.

This means that instead of L connections for a traditional L-layers network, there are $\frac{L(L+1)}{2}$ connections. Furthermore, the network is divided into multiple densely connected dense blocks as the size of feature-maps may change. These blocks are then separated by a batch-normalization layer and a 1×1 convolutional layer followed by a 2×2 average pooling layer. If H produces k feature maps, the number of input feature maps in l^{th} layer is

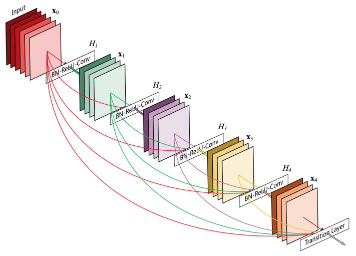
$$k_l = k_0 + k \times (l-1)$$

where k is the growth rate and k_0 is the number of channels in the input layer.

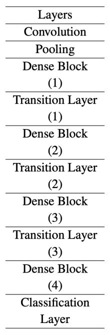
A 5-layer dense block with growth-rate $k = 4$ is illustrated in Figure 3.8a where $k_4 = 5 + 4 * (4-1) = 17$. The general architecture of DenseNet is shown in Figure 3.8a. The size of DenseNet network (for example 161) is equal to the sum of the number of layers in each dense block plus 5.

3.3 Siamese Networks

Siamese networks[26] are a kind of artificial network especially designed for similarity assessment. The architecture (see Figure 3.9) is composed of two identical neural networks f (working in tandem and sharing the same weights) and a similarity measure taking as input the outputs of these networks. The similarity measure can either be a distance measure such as Euclidean distance (Equation 3.8) (also known as the L2-distance) or



(a) 5-layer dense block with growth-rate of k=4



Layers
Convolution
Pooling
Dense Block (1)
Transition Layer (1)
Dense Block (2)
Transition Layer (2)
Dense Block (3)
Transition Layer (3)
Dense Block (4)
Classification Layer

(b) Architecture

Figure 3.8: DenseNet (source [25])

cosine similarity (Equation 3.9) for instance. For the definitions of both distances, two vectors $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^p \times \mathbb{R}^p$ are used. The Euclidean distance between the two vectors $\mathbf{x}$ and $\mathbf{y}$ is:

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{y} - \mathbf{x}||_2 = \sqrt{\sum_{i=1}^{p} (y_i - x_i)^2} \tag{3.8}$$

On the other hand, the cosine similarity measures the orientation of the two vectors $\mathbf{x}$ and $\mathbf{y}$ regardless of their magnitude. It is calculated by the dot product of two vectors divided by the product of the vector lengths (the Euclidean norm). A similarity close to 1 indicates high similitude. It is defined as follows:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||_2 \cdot ||\mathbf{y}||_2} \text{ with } ||\mathbf{x}||_2 = \sqrt{\sum_{j=1}^{p} x_j^2} \tag{3.9}$$

In the case where the inputs are images, the final objective of a Siamese network is not to classify but to learn to differentiate them.

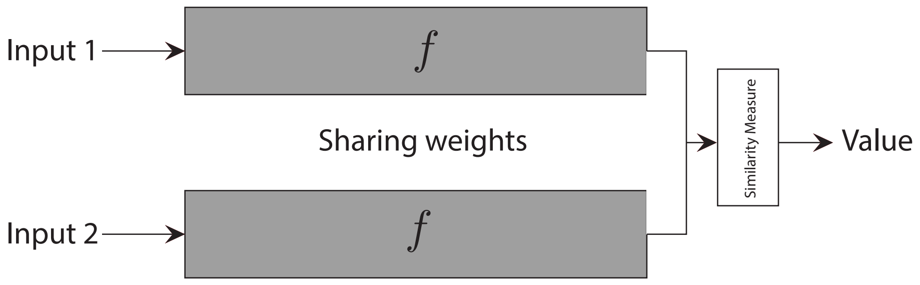


Figure 3.9: Siamese Network

For these specific networks, distance-based loss functions are usually preferred over prediction error-based loss functions like Logistic Loss or Hinge Loss used in classification for instance. The most popular known ones are the Contrastive Loss[27] and Triplet Loss[28]. They run over respectively pairs and trios of samples.

For a single pair, the Contrastive Loss is defined as follows:

$$\mathcal{L}(y, \mathbf{x}_1, \mathbf{x}_2; \boldsymbol{\theta}) = \frac{1}{2}(1 - y)d_{\boldsymbol{\theta}}^2 + \frac{1}{2}y[\max(0, m - d_{\boldsymbol{\theta}})]^2 \tag{3.10}$$

where $m > 0$ is the margin, y equals 1 if the two input vectors are similar and 0 otherwise and $d_{\boldsymbol{\theta}}$ is the Euclidean distance between the outputs

$$d_{\boldsymbol{\theta}}(\mathbf{x}_1, \mathbf{x}_2) = ||f_{\boldsymbol{\theta}}(\mathbf{x}_1) - f_{\boldsymbol{\theta}}(\mathbf{x}_2)||_2$$

For a triplet where $\mathbf{x}_p, \mathbf{x}_n, \mathbf{x}_a$ are respectively the positive, the negative and the anchor inputs, the Triplet Loss is defined as follows:

$$\mathcal{L}(\mathbf{x}_p, \mathbf{x}_n, \mathbf{x}_a; \boldsymbol{\theta}) = \max(||f_{\boldsymbol{\theta}}(\mathbf{x}_a) - f_{\boldsymbol{\theta}}(\mathbf{x}_p)||_2^2 - ||f_{\boldsymbol{\theta}}(\mathbf{x}_a) - f_{\boldsymbol{\theta}}(\mathbf{x}_p)||_2^2 + m, 0) \tag{3.11}$$

where $m > 0$ is the margin. The positive input describes the same entity as the anchor entity whereas the negative does not. L2-normalization (i.e. dividing a vector by its L2-norm) is generally used at the end of f when using the Triplet Loss (it is the case for example in the FaceNet[28] architecture). It has some nice properties. Firstly, The squared Euclidean distance between L2-normalized vectors is proportional to their cosine similarity:

$$||\frac{\mathbf{x}}{||\mathbf{x}||_2} - \frac{\mathbf{y}}{||\mathbf{y}||_2}||_2^2 = ||\frac{\mathbf{x}}{||\mathbf{x}||_2}||_2^2 + ||\frac{\mathbf{y}}{||\mathbf{y}||_2}||_2^2 - 2\frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||_2||\mathbf{y}||_2} = 2 - 2\frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||_2||\mathbf{y}||_2}$$

Furthermore, thanks to the normalization, the value of squared Euclidean distance is guaranteed to be within range $[0, 4]$, which helps making the choice of a margin.

For instance, Siamese networks can be used for face recognition[29] which consists in comparing faces such as FaceNet.

3.4 State-of-the-art Methods for Image Patch Matching

Image patch matching has been used as subroutine of many computer vision applications (super-resolution [30], image stitching[31], image denoising[32], etc). The state-of-the-art patch matching techniques [33, 34, 35] make use of convolutional neural networks, whether it is for learning to extract appropriate features, learning a similarity measure or both. Other techniques such as [36] use special encoding instead of CNN. In the remaining part of this section, the previously cited techniques and the dataset they used are going to be presented.

The UBC dataset (Brown et al. [37]) has been used as standard benchmark dataset for these papers. The dataset is composed of three subsets Yosemite, Notre Dame, and Liberty. Each of them contains more than 450,000 pair of patches. The patches are 64×64 pixels and grayscale. They were extracted from internet images thanks to either Difference of Gaussian (DoG)[38] interest point detector (which is used in the SIFT[39] algorithm) or multi-scale Harris corner detector[40]. The papers considered the ones extracted with DoG. An example of patches from the Liberty subset is shown in Figure 3.10 (one row corresponds to matching patches, i.e. from the same group).

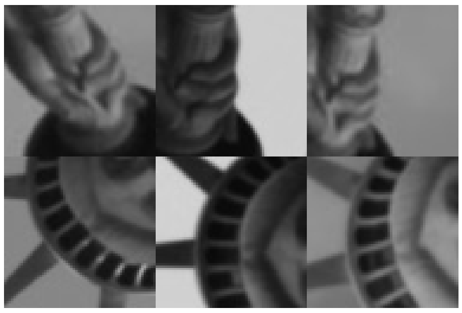


Figure 3.10: Samples from the Liberty subset

3.4.1 Learning to Compare Image Patches via Convolutional Neural Networks

In the approach suggested by Zagoruyko S. and Komodakis N.[33] in 2015, they have tested three main convolutional neural network architectures:

1. **Siamese**: as presented in Section 3.3, the networks share the same weights.
2. **Pseudo-siamese**: similar to the Siamese networks, the networks have the same architecture except that the weights are not shared.
3. **2-channel**: the input images are stacked before being fed into the network (each image corresponding to one channel).

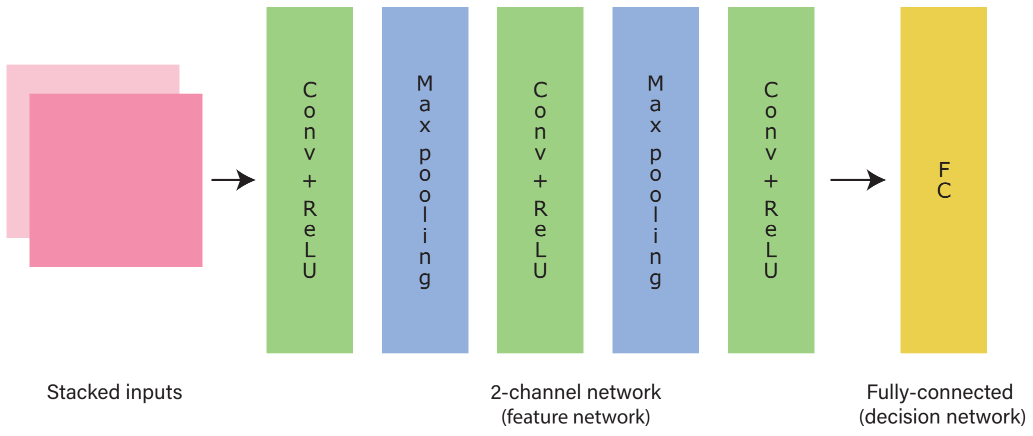


Figure 3.11: 2-channel architecture used in [33]

These architectures correspond to the branches for obtaining a feature descriptor. The branches are then merged at the bottom to make the comparison (thanks to a fully-connected network). An example of the 2-channel architecture network is shown in Figure 3.11. Their evaluations come to the conclusion that when the convolution layer was divided into small kernels of size 3 × 3, 2-channel architecture performs the best. The authors also tested 2-stream multi-resolution models which are worth mentioning. Coupled with the three presented architectures, they always show a significant performance increase, which allows to verify the importance of multi-resolution information when comparing patches. 2-stream multi-resolution models consist of two sub-convolutional networks (high and low resolution streams) and ending up in a fully connected network too. The inputs of the central high-resolution stream are generated by cropping (at the original resolution) the central part of the images while the inputs of the central low-resolution stream are generated by down sampling at half the original pair of input patches. The 2-stream model is illustrated in Figure 3.12.

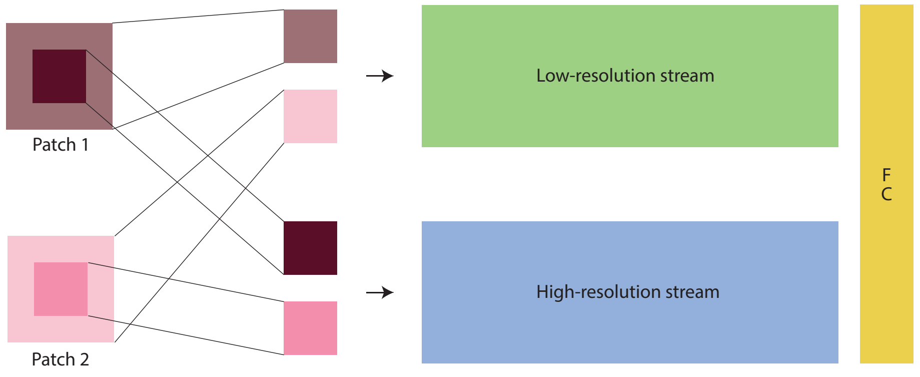


Figure 3.12: 2-stream architecture

3.4.2 MatchNet

The same year, Han X. et al. proposed a model called MatchNet[34]. MatchNet is a convolution neural network with a similar architecture to Siamese networks. It consists of two sub components, the feature and metric network respectively responsible for extracting the features of the patches and model the similarity between them. The feature network is inspired by the AlexNet[22] architecture. The output of the network is the probability that the patches are similar. To obtain this value, the final layer is normalized with the the softmax function. For $z \in \mathbb{R}^p$, the softmax function is defined as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{p} e^{z_j}}$$

The sum of the elements z_i, after the application of softmax function to z, is equal to 1. There are then two nodes v_0 and v_1 (v_1 being the similarity probability). The MatchNet architecture is illustrated in Figure 3.13. The Binary Cross-Entropy (BCE) loss has been used to train the network:

$$BCE(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \tag{3.12}$$

where $\mathbf{y}$ is the target label vector (0 for dissimilar pairs and 1 otherwise), $\hat{\mathbf{y}}$ is the output vector of the network and n is the batch size.

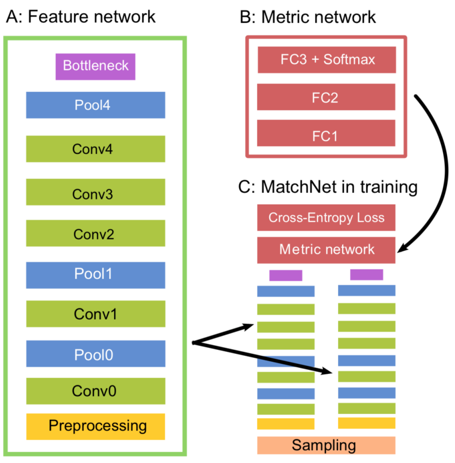


Figure 3.13: MatchNet (source [34])

3.4.3 Image Patch Matching Using Convolutional Descriptors with Euclidean Distance

In 2017, Melekhov I. et al.[35] proposed a Siamese network architecture using the euclidean distance. In contrast to CNN3 model[41] and the two models siam-l_2, pseudo-siam-l_2 proposed by [33], they decomposed convolutional layers with a big kernel size into several filters with smaller kernels 3×3 and separate them by ReLU activations. They used the Constrastive Loss (Equation 3.10) to train the network.

3.4.4 Sparse Over-complete Patch Matching

The usage of sparse coding to obtain more accurate encoding have shown successful progress in computer vision and signal processing applications. Based on this concept, Akila Pemasiri et al.[36] proposed in 2018 a new patch matching framework (sparse-coding had never been adopted for patch-matching processes). Their approach uses overcomplete sparse representation of image patch as the patch descriptor.

The basis of sparse-coding is the following. Given a dataset $X = [x_1, ..., x_n] \in \mathbb{R}^{m\times n}$ and the corresponding coding (latent representation) $C \in \mathbb{R}^{k\times n}$, the goal is to find a decoder $D \in \mathbb{R}^{m\times k}$ such that:

$$X \sim DC$$

and thus minimizing the reconstruction error:

$$\min_{D,C} ||X - DC||$$

When $m > k$ it is known as undercomplete, $m = k$ complete and $m < k$ overcomplete. Furthermore, sparse means that the coding has lots of zeros. Generally, the sparse coding algorithms relies on the following objective function:

$$\min_{D} \frac{1}{N} \sum_{i=1}^{n} \min_{c_i} ||x_i - Dc_i||_2^2 + \lambda ||c_i||$$

where λ is the sparsity penalty.

In [36], after learning the basis dictionary vector D, the encoded data using D are used to train a fully-connected neural network with sigmoid (Equation 3.4) activation. The training is done with the BCE loss (Equation 3.12).

3.5 Evaluation Metrics

All the techniques presented in Section 3.4 use the evaluation protocol proposed by [37] where the training is carried out on patches from one subset of the UBC dataset (e.g. Liberty) and the testing is carried out on a set of patches retrieved from another subset (e.g. Yosemite). ROC curves are calculated by thresholding the distance between feature pairs and the accuracy is measured with False Positive Rate at 95% recall (FPR95, also known as Error@95%).

The receiver operating characteristic (ROC) curve and the area underneath it (AUC) is a classical framework for analyzing the performance in classification problems. ROC curve is created by plotting the true positive rate (TPR) against the False Positive Rate (FPR) at various threshold settings.

The True Positive Rate (called Recall or Sensitivity) is defined as:

$$\text{True Positive Rate (TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The False Positive Rate (equals to 1 - Specificity) is defined as:

$$\text{False Positive Rate (FPR)} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

where TP, TN, FP and FN denote respectively the True Positives, the True Negatives, the False Positives and the False Negatives.

Let's say that one may wish to differentiate between **similar** (positive) and **dissimilar** (negative) pairs of patches (as the objective of this paper). In this case, the confusion matrix is given in Table 3.1.

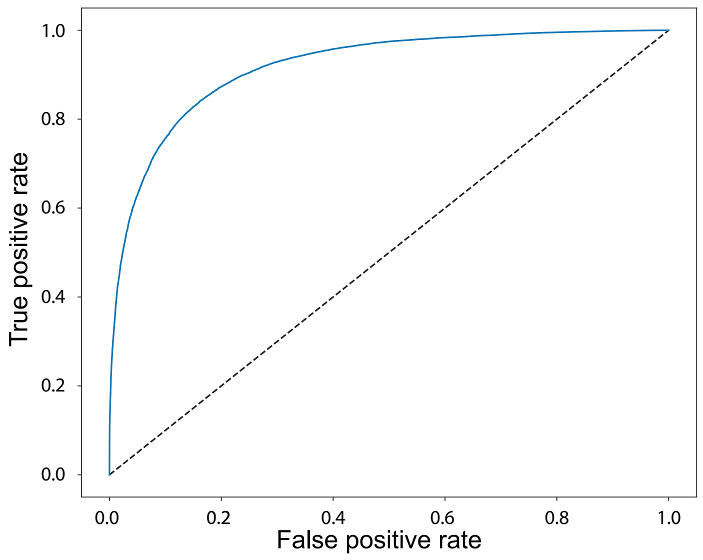


Figure 3.14: Example of a ROC curve

An example of a ROC curve is shown in Figure 3.14. The dotted line means that the proportion of correctly classified **similar** pairs is the same as the proportion of incorrectly classified pairs that are **dissimilar**. This line is equivalent to random guessing. Thus, the point at (1,1) means that even though the **similar** samples were all correctly classified, the **dissimilar** samples were all misclassified (i.e. all predicted as **similar**).

		Actual class	
		similar	**dissimilar**
Predicted class	**similar**	# TP	# FP
	dissimilar	# FN	# TN

Table 3.1: Table of confusion

The AUC is an indicator about how well a parameter can distinguish between both classes. The higher the AUC, the better. As previously presented, the False Positive Rate at 95% Recall is a significant clue about the capability to distinguish pairs too. Indeed, it is interesting to know how well the classifier performs at identifying correctly **dissimilar** pairs, when it is very good at recognizing **similar** ones (i.e. when the TPR is 95%). Of course, the lower this measure, the better.

Another evaluation metric that will be of great interest here is the top-k accuracy. Let n be the total number patches which are of interest (i.e. in this thesis which corresponds to a landmark and are wished to be identified in a target image), a_i a reference patch and b_j the j^{th} best predicted patch from the target image. Top-k accuracy is defined as follows:

$$\text{top-}k \text{ accuracy} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} d(a_i, b_j) \tag{3.13}$$

$$d(a_i, b_j) = \begin{cases} 1 & \text{if } a_i \text{ and } b_j \text{ are considered to be matching} \\ 0 & \text{otherwise} \end{cases}$$

For instance, the top-1 accuracy indicates the proportion of matches correctly identified.

Chapter 4

Results

In this chapter, several experiments and their respective results are going to be presented. All the experiments have been conducted with the dataset from the ANHIR challenge (see Section 2.2). The chapter is organized as follows. First of all, a naive segmentation technique is described (which is useful for analyzing patches from whole slide images by removing unnecessary background). Then, the performance of several pre-trained ImageNet networks as feature extractors are compared. After, network models (with different architectures) trained on the dataset are tested. Finally, the accuracy of the two main methods are going to be confronted. Note that all the implementations are written in Python using the computer vision and machine learning Pytorch[42] framework which is based on the Torch library. All the code for the project is available publicly on GitHub (`https://github.com/loicsacre/code-master-thesis`) and is accompanied by a complete description in the `README` file.

4.1 Pre-Processing

4.1.1 Segmentation

In order to make analyzis on the whole slide images, patches covering the entire tissue have to be extracted. To do so, a segmentation technique based on thresholding is used. The process is illustrated in Figure 4.1. The whole histological image is partitioned in several patches thanks to a sliding window of size $s \times s$ pixels. This window browses from the top-left corner of the picture by making a shift of $shift$ pixels horizontally at each step. When reaching a border at the right, the window is shifted by $shift$ pixels vertically and restarts to browse from the left border. A patch is not considered as part of the main object (the tissue) if the mean and the standard deviation of the pixel values over the 3 channels are respectively greater than 210 and less than 6 (this assumes that the background is mainly light and tends to be uniform). Some instances of the segmentation are shown in Figure 4.2. This segmentation technique has been applied to all the images available in the dataset. For the two combinations of the $s/shift$ parameters, 300/75 and 600/150, it requires a mean time of 22 seconds to get the full segmentation.

4.1.2 Data Normalization

Data normalization might play an important role in deep learning, whether it is for dealing with features whose range of possible values differ a lot for keeping equal variance within all the input data. The normalization operation consists in scaling data values with a

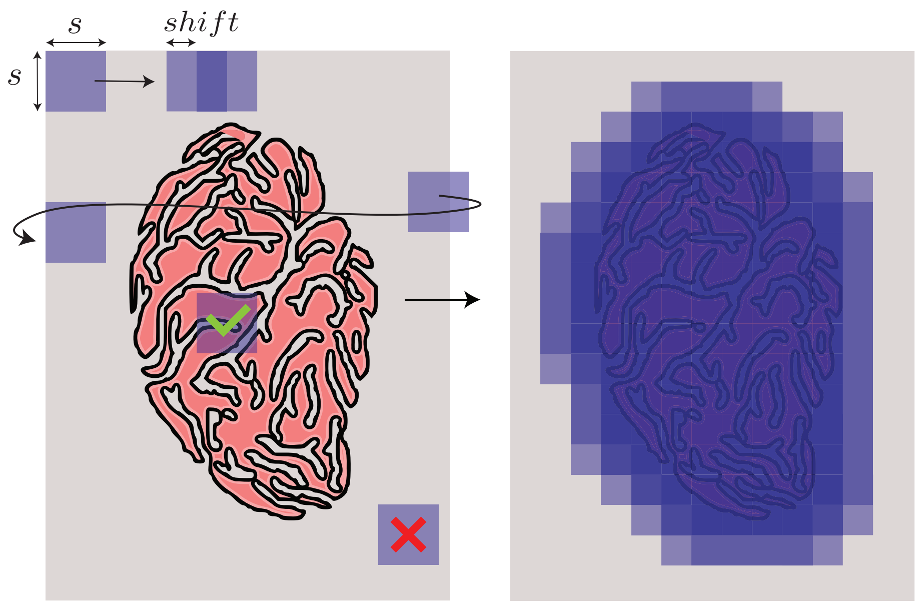


Figure 4.1: Segmentation workflow

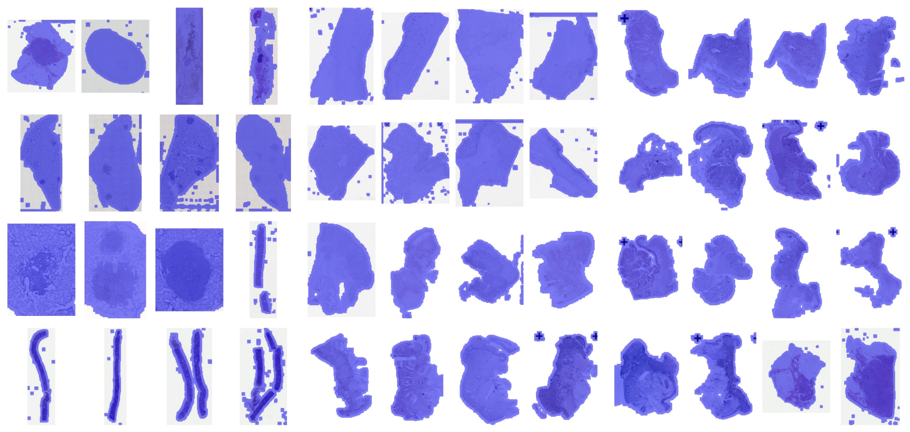


Figure 4.2: Examples of segmented tissues ($s = 300$ and $shift = 75$)

common factor. For instance, for a dataset $\boldsymbol{X}$, whose mean is μ and standard deviation (std) is σ then standard normalisation is performing:

$$\frac{\boldsymbol{X} - \mu}{\sigma}$$

which remaps all the values to a range such that they have together a mean of 0 and a standard deviation of 1.

For example, the networks pre-trained on the ImageNet database (within the Pytorch library) were trained on images in RGB format and using a per channel standard normalization:

$$\begin{aligned} \text{mean} &= [0.485, 0.456, 0.406] \\ \text{std} &= [0.229, 0.224, 0.225] \end{aligned}$$

Each value corresponds to a channel and were obtained by computing mean and standard deviation of pixel values. Note that the initial value of a pixel had been first re-scaled from a range of [0, 255] to [0, 1].

The first idea and most straightforward way to perform normalization with the dataset of whole slide images would be to normalize the patches per dye. Unfortunately, as stated in Section 2.2, the dyes following the Si (i being a number between 1 and 8) format are not specified. Nevertheless, a per image data normalization is still adopted. To obtain the values, the images are segmented using the technique described in the previous section with parameters $s = 300$ and $shift = 150$. Then, per-channel normalization parameters are computed by taking the mean and standard-deviation of pixel values over all the patches resulting from the segmentation. All the experiments will use this normalization (except when it it is clearly mentioned).

4.2 Transfer Learning via ImageNet Siamese Networks

4.2.1 Settings

The first approach for comparing the patches with each others consists in extracting a **feature vector** (i.e. a list of numbers taken from the output of a pre-trained neural network layer) for each patch and then compare them thanks to a similarity measure. For illustrative purposes, Figure 4.3 shows reshaped feature vectors extracted from three patches. It can be noticed that the vectors 4.3a and 4.3b look very similar. This approach is exactly what Siamese networks (see Section 3.3) do. Nevertheless, the function f is fixed (the network is not trained again) and the similarity measure is the cosine similarity (Equation 3.9).

(a) $\tau^{\text{Mice kidney-1}}_{\text{PAS-1}}(115)$

(b) $\tau^{\text{Mice kidney-1}}_{\text{PAS-2}}(115)$

(c) $\tau^{\text{Gastric-5}}_{\text{CD4}}(78)$

Figure 4.3: Feature vector from different 100×100 patches obtained with AlexNet pre-trained on the ImageNet database

In order to perform comparisons via feature extraction, several well-known architectures (the ones presented in Section 3.2.9) have been selected among the already implemented networks in Pytorch to compare their abilities in working on histological images.

The chosen networks are:

- **AlexNet**
- **VGGNet**: with 16 and 19 layers (with batch-normalization)
- **ResNet**: with 18, 34 and 50 layers
- **DensetNet** : 161 and 201

The feature vector is obtained by taking the output of the layer situated just before the fully connected network.

When the feature vectors (one dimensional vectors) are extracted for both patches, cosine similarity is used to compare the vectors and obtain a score (the closer to 1, the better). Table 4.1 summarises each of the network architectures with the number of parameters of the feature extractor part (i.e. the network after removing the fully connected end), the size of the feature vectors for patches of both sizes 100×100 and 300×300 pixels. The goal of using 100×100 pixels patches is to show that the use of too small image contents leads to poor performance.

Note that the 100×100 pixels patches were resized to 224×224 pixels due to architecture constraints. Indeed, the minimal input size required is 224×224 pixels because it it the same as the images from the ImageNet dataset used to train these networks.

Architecture	**Number of parameters**	**Feature vector size**	
		100×100	300×300
AlexNet	2,5M	9216	16384
ResNet-18	11,2M	512	8192
ResNet-34	21,3M	512	8192
ResNet-50	23,5M	2048	32768
VGGNet-16 (BN)	14,7M	25088	41472
VGGNet-19 (BN)	20M	25088	41472
DenseNet-161	26,5M	2208	19872
DenseNet-201	26,5M	2208	19872

Table 4.1: Number of parameters and the feature vector size (according to the size of the patches in pixels) for the pre-trained networks

As it can be noticed in Table 4.1, there is a large disparity among the size of feature vectors. This leads to biased results as will be shown in the following Section 4.2.2. Table 4.2 summarizes each of the network architectures Resnet and DenseNet (without their final pooling layer) with the size of the feature vector for patches of both sizes.

Architecture	**Feature vector size**	
	100×100	300×300
Resnet-18	25088	51200
Resnet-34	25088	51200
Resnet-50	100352	204800
DenseNet-161	108192	178848
DenseNet-201	94080	155520

Table 4.2: Feature vector size (according to the size of the patches in pixels) for ResNet and DenseNet (without their final pooling layer)

4.2.2 Evaluation of ImageNet Networks

The first experiment with feature extraction consists in testing the performance of the different ImageNet networks at identifying matching pairs. Note that only the patches coming from the available landmarks were used for both images (i.e. that a patch from a source image is searched among all the available patches from the target one). The final goal will obviously be to find a patch among the patches covering all the target images of the pair. Nevertheless, the results will already provide an upper-bound as it is much more easier to find a patch when there are far less comparisons to perform. The number of landmarks for an image is of the order of a hundred while the number of patches resulting from a segmentation is of the order of a thousand (considering a reasonable size for the patches and and $shift$ parameter for the segmentation). This analysis will allow to compare the different network architectures more quickly.

The top-k accuracy ($k \in \{1, 5\}$) (Equation 3.13) is used to assess the performance of the networks. As the number of landmarks is relatively low, each member of the pairs of images will be used as a reference and a target. Thus, n is equal to 2×16839 (see Section 2.2). Furthermore, two patches are considered as matching if they correspond to the same landmark number.

In Table 4.3 are shown the top-1 and 5 accuracy for both sizes. It can be noticed that AlexNet and VGGNet are performing best in comparison to the other architectures and as announced previously, the use of too small patches leads to a large performance decrease.

Another observation is that there is a strong correlation between the feature vector size and the accuracy. This is for that reason that the same experiment have been performed a second time for ResNet and DenseNet architectures but without their final pooling layer. The results are in Table 4.4. As it can be observed by comparing the two result tables, the performance of ResNet and DenseNet has increased considerably. In fact, the average pooling operation completely destroys the spatial information that the network could have extracted.

The experiments have also been conducted with the patches normalized with the same parameters used to trained the networks on ImageNet. The results are in Tables 4.3 and 4.4. It is quite interesting to observe that the results are similar, even a bit higher.

	Parameters from tissue images				Parameters from ImageNet			
	100×100		300×300		100×100		300×300	
	Top-1	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**
Architecture								
AlexNet	0.3935	0.6223	0.5874	0.7559	0.44	0.6599	0.6072	0.7641
ResNet-18	0.204	0.4308	0.2762	0.5286	0.268	0.5095	0.3866	0.6414
ResNet-34	0.1955	0.4199	0.2615	0.5091	0.261	0.4996	0.372	0.6282
ResNet-50	0.1791	0.3936	0.2393	0.479	0.2553	0.4928	0.3553	0.6144
VGGNet-16 (BN)	0.3467	0.5764	0.5218	0.7152	0.4389	0.6565	0.6114	0.7693
VGGNet-19 (BN)	0.3671	0.5937	0.5563	0.7342	0.4225	0.6416	0.6083	0.7713
DenseNet-161	0.2414	0.4772	0.4217	0.6748	0.2746	0.5091	0.4459	0.6834
DenseNet-201	0.2502	0.4892	0.4342	0.6848	0.2886	0.5311	0.4611	0.6967

Table 4.3: Top-1 and 5 accuracy of different networks trained on ImageNet. The inputs are either normalized with parameters obtained from the images of the dataset or the parameters used for training the networks on ImageNet

	Parameters from tissue images				Parameters from ImageNet			
	100×100		300×300		100×100		300×300	
	Top-1	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**
Architecture								
Resnet-18	0.3399	0.5681	0.5441	0.7311	0.4262	0.6451	0.6163	0.7785
ResNet-34	0.3365	0.571	0.5029	0.6962	0.4161	0.6421	0.6042	0.7745
ResNet-50	0.3236	0.5505	0.4989	0.6842	0.4245	0.6442	0.6171	0.7821
DenseNet-161	0.3852	0.6123	0.6233	0.7845	0.4175	0.6361	0.6423	0.7961
DenseNet-201	0.3907	0.618	0.6354	0.7957	0.4391	0.6545	0.6515	0.803

Table 4.4: Top-1 and 5 accuracy of different networks (without their last average pooling) trained on ImageNet. The inputs are either normalized with parameters obtained from the images in the dataset or the parameters used for training the networks on ImageNet

4.2.3 Evaluation on Segmented Whole-Slide Images

As previously stated, the goal is to find a matching patch without having any prior knowledge about its true position. In order to perform such an evaluation, it is required to segment the whole-slide images to separate the tissue from the background. Indeed, it is not necessary to analyze patches which do not contain any information.

The complete evaluation procedure is going to be described. First of all, every pair of images for a same tissue is retrieved (191 in total). Like the previous experiments, $2 * n$ ($n = 16.839$) is equal to 33678 as the two elements of the pair plays the role of the reference each one its turn. Refer to Section 2.2 for the numbers. Then for each pair, a specific size and a certain network architecture:

1. Get the n patches (from the landmarks only) from the reference image and extract the features. Let's call this feature matrix R (like **R**eference) and it is of size $n \times l$ (where l is the size of the feature vector).

2. Get all the m patches (resulting from the segmentation) from the target image and extract the features. Let's called this matrix T (like **T**arget) and it is of size $m \times l$

3. Compute cosine similarity of feature vectors from the matrix R and T. The result of the comparison C is a matrix of size $n \times m$.

4. For each row c_i of C, take respectively the column number(s) (corresponding to a patch in the second image) of the first or 5^{th} highest similarity scores for the top-1 and top-5 accuracy. The matching condition is now a bit different as the exact position of the matching patch is supposed to be unknown. The patch are matching if the center coordinate of the target patch lies within a radius smaller or equal to half the size of the patch around the true center coordinate. This condition is illustrated in Figure 4.4. The green patch is the true target patch, the purple patch is the target one (having the best score) and the blue circle has a radius of half the size of a patch.

As a reminder, AlexNet and DenseNet-201 (without the final max pooling) obtained the good results for the patches of size 300×300 pixels. This is why these two networks were considered (with VGG-19 (BN) used as benchmark). The procedure described above has been executed for the 191 pairs and a total top-k ($k \in \{1, 5\}$) accuracy has been computed by summing and weighting by the number of landmarks the accuracies obtained for each

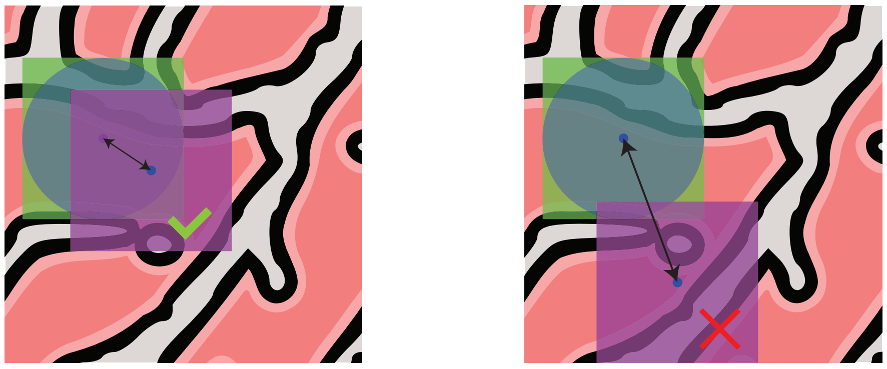

(a) Matching patches (b) Non-matching patches

Figure 4.4: Illustration of the matching condition

pair.

The results of the analysis for these three networks are shown in Table 4.5 for a value of shift equals to the size of the patch divided by 4. The time column is the time taken for extracting all the feature vectors from the patches of the target images used to obtain these results. DenseNet-201 still shows the best results.

Architecture	**Size**	**Shift**	**Top-k accuracy**		**Time to get features** (mean)
			$k=1$	$k=5$	
AlexNet	300	75	0.327	0.468	8:37:17 (0:01:21)
VGGNet-19 (BN)	300	75	0.279	0.423	13:53:41 (0:02:10)
DenseNet-201	300	75	0.448	0.587	2 days - 1:01:05 (0:07:41)
DenseNet-201	600	150	0.549	0.684	15:06:00 (0:02:22)
DenseNet-201	600 (resized to 300)	150	0.586	0.72	15:01:17 (0:02:21)

Table 4.5: Top-k accuracy of different networks trained on ImageNet when using a complete segmentation of the target image for different *size* and *shift* parameters as well as the total (and mean) time to get the features of the patches from the 382 target images

Now, some more specific results are going to be given. A first observation that can be made is that the networks are highly dependent to differences in orientation between the tissue. For instance, Figure 4.5 shows two sections of a COAD sample as they are given in the database from the ANHIR challenge. As it can be seen, the two sections are differently oriented. The results on this pair with DenseNet-201 are both 0 for the top-1 and 5 accuracy. There are 83 (resp. 8180) patches for the source (resp. target) stained with S7 (resp. S2). Nevertheless, if the target image is rotated by an angle of $-75°$, the results become respectively 0.3614 and 0.5181 for the top-1 and 5 accuracy. It is worth mentioning that the networks trained on ImageNet for the ILSVRC are not intended to be rotation invariant. The only transformation performed on the images in order to increase the size of the dataset is a horizontal flip. Furthermore, the images from ImageNet contain mostly objects that are vertical (humans, animals, etc). Specific architectures can however be built for handling rotation (see [43, 44, 45]).

In order to visualize the results on a specific pair, heat maps will be used. An example is shown in the bottom part of Figure 4.7. The patches of the segmented target images are highlighted in a specific color corresponding to the similarity score. The colours are following the color scale shown in Figure 4.6. The bluer the better. The lower bound corresponds to the 90^{th} percentile (this is an arbitrary choice) of the similarity score (this

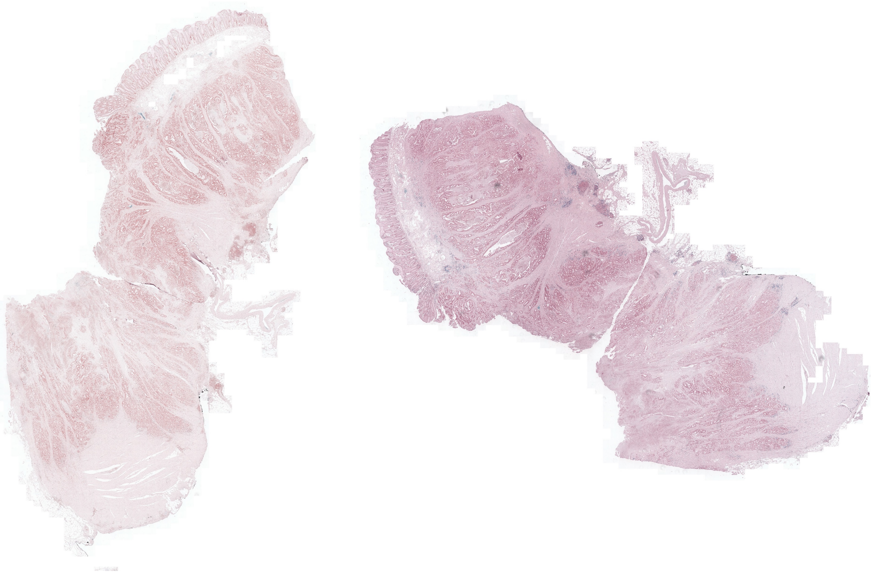


Figure 4.5: Two sections $\tau_{\text{S7}}^{\text{COAD-20}}$ (left) and $\tau_{\text{S2}}^{\text{COAD-20}}$ (right)

means that below this value lie 90% of the scores) while the upper bound is the maximum score. The green patches represent the 4 best matches before the maximum one which is highlighted in purple. The yellow cross represents the true position of the landmark that should be matched).

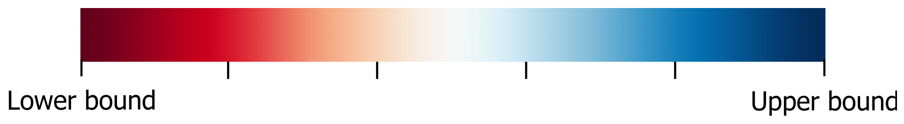


Figure 4.6: Color scale

For instance, two sections of a mammary glands sample are given in Figure 4.7. The results on this pair with DenseNet-201 are 0.948 (resp. 0.987) for the top-1 (resp. top-5) accuracy. There are 77 (resp. 2178) patches for the source (resp. target) stained with NEU (resp. HE). These great accuracy values result from the fact that the images are down-scaled at four times their original size and thus the patches contain much more information. Nevertheless, the color difference does not seem to affect the results.

In order to illustrate other observations, two pairs of sections from a lung lobes sample have been chosen. The reference section is stained with dye CD31 in both cases and have 107 reference patches. The two others are stained with PRO-SPC and CD10 and there are respectively 8562 and 8541 patches coming from the segmentation. The results obtained with DenseNet-201 for the first pair are 0.813 (resp. 0.925) for the top-1 (resp. top-5) accuracy and 0.486 (resp. 0.664) for the top-1 (resp. top-5) accuracy. In Figure 4.8 are shown four examples of matches for both of pairs and for the same reference patches. The three first examples show a correct match in both cases (the purple colored patch and the yellow cross are overlapping). In the first case, the reference patch is located on a border. Generally, the networks tend to have more ease to find the target patch when the reference one is situated in such location. This can be explained by the fact that a large portion of the patch is background content and the overall pattern of the patch is thus easily identifiable. The second and third examples have both a recognizable pattern too in contrast to the fourth with which even a human being would have some difficulties to spot the similarity.

Finally, as already briefly introduced in Section 2.2, the choice of the size of a patch is important and should normally be tuned on a case by case basis. For instance, Figure 4.9

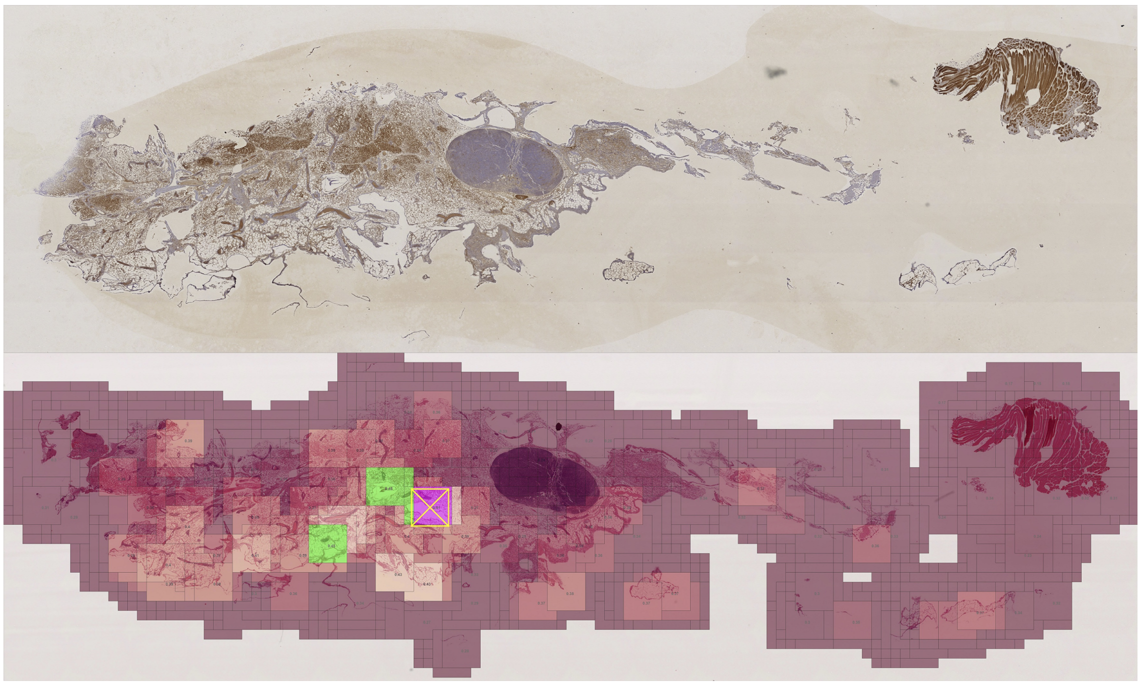


Figure 4.7: Two sections: $\tau_{\text{NEU}}^{\text{Mammary glands-1}}$ (top) and $\tau_{\text{HE}}^{\text{Mammary glands-1}}$ (bottom)

illustrates the same lung lesion sample but with different patch sizes. The reference (resp. target) is stained with CC10 (resp. PRO-SPC) and have 80 (resp. 10288) patches. The top-1 and 5 accuracy go from 0.463 and 0.675 to 0.675 and 0.7625 when the size of the patches goes from 300×300 to 600×600 pixels. Furthermore, the top-1 and 5 accuracy obtained with DenseNet-201 on segmented images with patches of size 600×600 pixels (using a shift of 150 pixels) can be found in Table 4.5. There is a 10% percent increase in both cases. It seems logical that the more information is used to compare, the greater the performance should be. This is why the same experiment has been done (patch size of 600×600 pixels and a shift of 150 pixels) but this time with the patches resize to 600×600 pixels are resized to 300×300 before going through the feature extractor. It can be observed that the results are even a little better (0.586 and 0.72) in comparison to keeping the true size of the patches. The patches should in general be larger than 300×300 pixels but it is not necessary to require on a bigger feature vector.

4.3 Models Trained on the Data

The second approach consists in building models trained on the dataset which is at disposal. This section is organized as follows. First of all, the data set will be presented, then the evaluation procedure will be set and the different architectures will be described. Finally, the results will be presented.

4.3.1 Dataset

In order to train new models, only the patches of size 300×300 pixels have been considered. As it has been shown previously, the smaller patches do not contain enough information to be compared effectively.

In total, there are 191 pairs of images and 16839 pairs of **similar** patches. Unfortunately, some of the patches do not have the exact 300×300 size and are not sized like a square (i.e. the width is different from the height). These patches have been discarded because

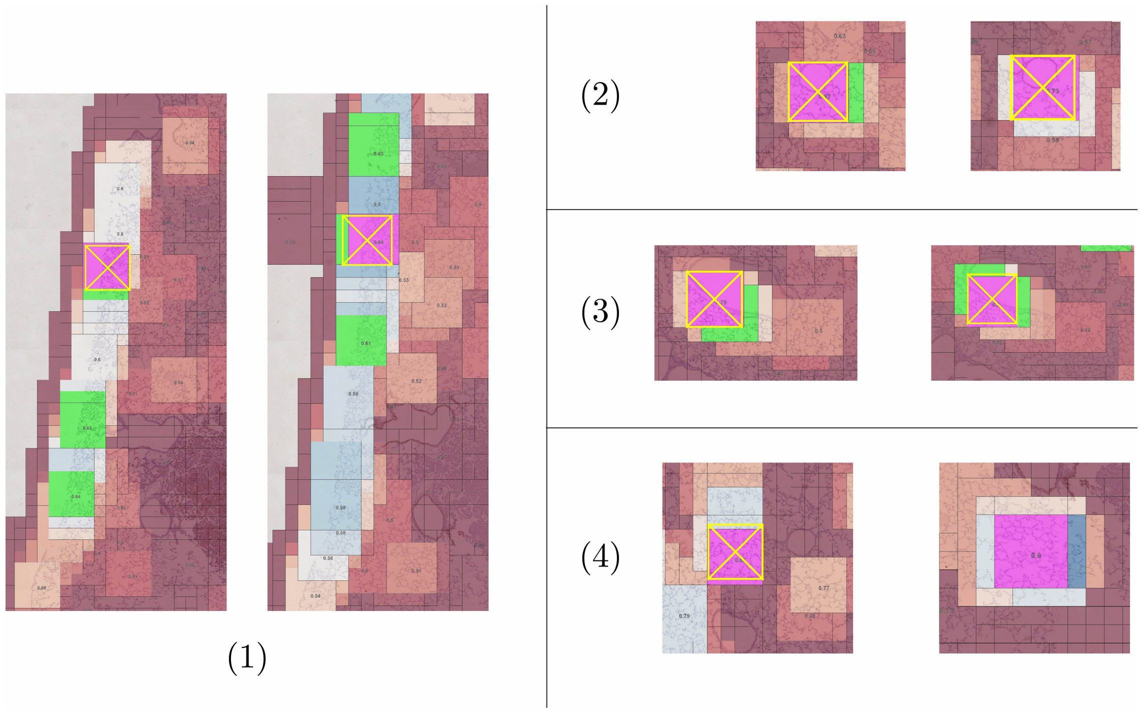


Figure 4.8: Example of matches in different sections of the same tissue with $\tau_{\text{CD31}}^{\text{Lung lobes-2}}$ as the reference. For each pair, the left (resp. right) corresponds to $\tau_{\text{PRO-SPC}}^{\text{Lung lobes-2}}$ (resp. $\tau_{\text{CC10}}^{\text{Lung lobes-2}}$)

resizing them would have caused the patches to be distorted, which is something to avoid as the tissues are already suffering from deformations as explained in Section 2.1. Note that it does not drastically reduce the size of the dataset which consists of a remaining number of 16788 pairs of patches. Then, in order to properly train a network thanks to early-stopping (see Section 3.2.5), the dataset needs to be split into three subsets: training, evaluating and testing.

The proportion of the dataset in each subset is respectively and approximately 65%, 15% and 20%. Furthermore, a constraint is added: a tissue can only be in one subset.

Obviously, a network is not only trained to recognize **similar** pairs but well to recognise **dissimilar** ones. Furthermore, the number of existing **dissimilar** pairs exceeds the number of **similar** ones. In order to increase diversity and prevent overfitting, the **dissimilar** pairs are sampled randomly at each epoch. The approach is inspired by [34] which uses a reservoir sampling during training. In addition, in order not to bias the training towards negative decisions, an equal number of **similar** and **dissimilar** pairs are sampled for each batch. The employed procedure is explained in Algorithm 1. For generating a triplet (when using the Triplet Loss (Equation 3.11)), the same principle is applied except that the anchor and the positive samples correspond to the **similar** pair and the negative sample is generated randomly according to the anchor (forming a **dissimilar** pair together).

As the dataset is quite small, data augmentation is used. For each **similar** pairs, 7 transformations are applied to generate more samples: rotation of 90°, 180°, 270°, horizontal and vertical flip as well as the transpose and the transverse. All these transformations are the ones proposed by the PIL library[46].

Finally, after data augmentation, the dataset is composed of $16,788 \times 2 \times 8 = 268,608$ pairs. It remains quite small in comparison to the UBC dataset presented in the Section 3.4 in which there are more 450,000 pairs of patches per subset (without taking into account any data augmentation!) .

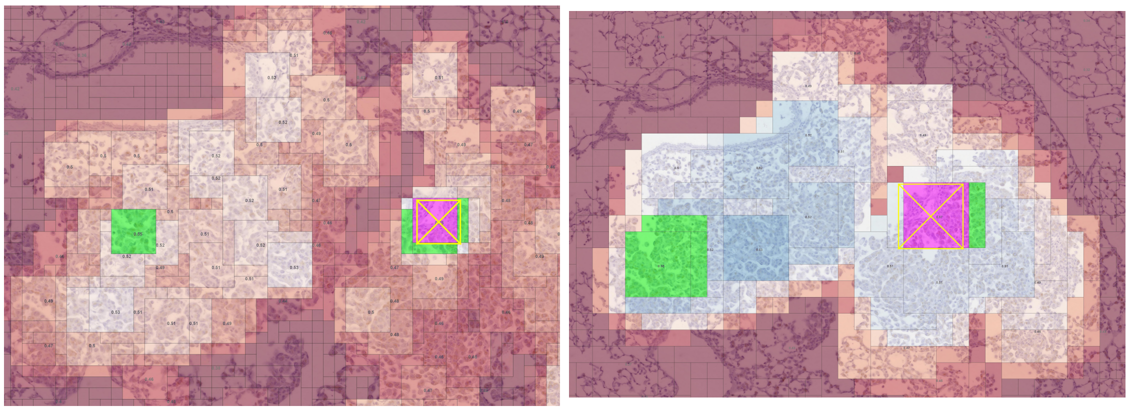

(a) 300 × 300 pixels (b) 600 × 600 pixels

Figure 4.9: Heat map with different patch sizes for a target $\tau^{\text{Lung lesion-3}}_{\text{PRO-SPC}}(76)$ with reference $\tau^{\text{Lung lesion-3}}_{\text{CC10}}(76)$

Algorithm 1: Sampling during training with BCE (Equation 3.12)

```
Data: Let $p_{sim}$ and $p_{dis}$ be two stacks initiated, at each epoch $ep$, with a differently
      shuffled list containing all the pairs of patches $T^t_{da,db}(i) = (\tau^t_{da}(i), \tau^t_{db}(i))$
Result: Generate a batch of size $2S$ at epoch $ep$
samples = [];
for $s$ in $S$ do
    samples[$2s$] = $p_{sim}$.pop() // similar ;
    $T^t_{da,db}(i) = p_{dis}$.pop();
    $\tau^t_{da}(i)$ = PICK_RANDOMLY_FROM_PAIR($T^t_{da,db}(i)$) ;
    $\tau^t_{db}(j)$ = GENERATE_RANDOMLY_DISSIMILAR($\tau^t_{da}(i)$) // da $\neq$ db & i $\neq$ j;
    samples[$2s$+1] = ($\tau^t_{da}(i)$, $\tau^t_{db}(j)$) // dissimilar;
end
Return samples;
```

A summary of what each subset contains is presented in Table 4.6. Note that if the Triplet Loss is used for training, the total number of triplets is equal to the half of the total number of **similar** pairs.

4.3.2 Architectures

Four different types of architecture have been tested and evaluated. The first one is the original **MatchNet**[34] network presented in Section 3.4. The reason of trying this architecture and working with 64 × 64 pixels grayscale patches is to investigate the possibility of using a more lightweight network.

The second and third network are both inspired from **MatchNet** and they will be referred as **TransferNet**. However, the feature networks are fixed and are respectively AlexNet and VGGNet-16 (BN) both pre-trained on the ImageNet database. The feature network is followed by a 1 × 1 convolution layer (with ReLu activation) in order to perform a dimensional reduction by reducing the number of feature-maps. It also has an adaptive average pooling layer. The metric network is a fully-connected network (with ReLu activation) ending with a softmax layer. The **TransferNet** architecture is shown in Figure 4.10. The details of the complete architecture for **TransferNet with AlexNet** (resp. **TransferNet with VGGNet**) are given in the Appendice A.2 (resp. A.3) following some conventions described in A.1.

Pairs of patches	Training	Evaluation	Testing	Total
All (%)	~65	~15	~20	100
# **similar**	10,960	2,493	3,335	16,788
All (with data aug.)	175,360	39,888	53,360	268,608
Tissues	39	4	6	59
Pairs of dyes	125	31	35	191

Table 4.6: Summary of the dataset

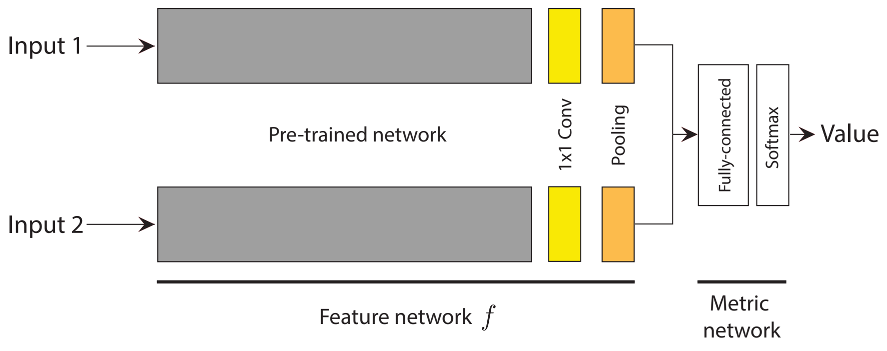


Figure 4.10: **TransferNet** architecture

The final architecture is a Siamese-like network (the similarity is learned too) which is trained thanks to the Triplet Loss (Equation 3.11). It will be denoted as **Siamese AlexNet**. The architecture is illustrated in Figure 4.11. It consists of the AlexNet feature network (which is this time, not pre-trained on ImageNet), followed by a 1×1 convolutional layer (to perform a dimensional reduction), an adaptive average pooling layer and a fully-connected layer. Finally, the output is L2-normalised. The details of the architecture, following the previous given conventions, can be found in A.4.

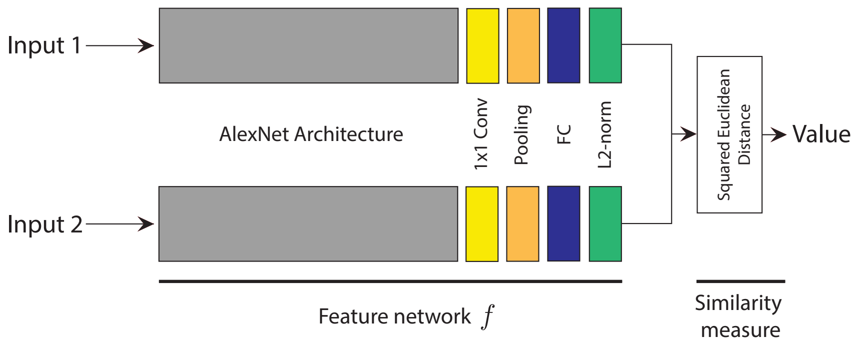


Figure 4.11: **Siamese AlexNet** architecture

4.3.3 Evaluation

The first three networks **MatchNet** and the two **TransferNet** networks were trained using the Binary Cross Entropy (Equation 3.12). On the other hand, the **Siamese AlexNet** network was trained with the Triplet Loss (Equation 3.11). They had all been trained using the (mini-batch) Stochastic Gradient Descent optimiser and with a batch size of 64.

Several parameters of learning rate as well as the presence of momentum (with a value of 0.9) or not had been tested. For the **Siamese AlexNet**, a margin value of 0.2, 0.5, 1.0 and 1.5 had been compared. A summary table of the parameters which gave the best results can be found in Table 4.7

Architecture	Loss	Batch size	Learning Rate	Momentum	Margin
MatchNet	BCE	64	0.005	0.9	/
TransferAlexNet			0.001		/
TransferVGGNet			0.001		/
Siamese AlexNet	Triplet		0.01	0	0.5

Table 4.7: Summary of the trained networks and their parameters

TransferAlexNet (**TransferNet with AlexNet**) and **TransferVGGNet** (**TransferNet with VGGNet**) have respectively 274,770 and 1,827,458 to be learned (just the similarity function is being tuned) while **MatchNet** (resp. **Siamese AlexNet**) have 9,674,738 (resp. 61,233,088) learnable parameters.

In order to assess the performance, ROC curve as well as the FPR95% will be used. The results may vary a little with the run as the batches are generated randomly. Anyway, they serve as a good indicator in the ability of the network to discriminate **similar** and **dissimilar** pairs. For each of the network architecture, the ROC curves resulting from the evaluation of the different models on the evaluating (resp. testing) set is given in Figure 4.12 (resp. 4.13). Note that the AUC values are rounded to the first decimal (except for **Siamese AlexNet**) due to a small implementation mistake. A value of 1 indicates a true value greater than 0.95 and less than 1 (an AUC of 1 is unlikely to be obtained). A value of 0.9 indicates a value less or equal to 0.95 and greater of equal to 0.9.

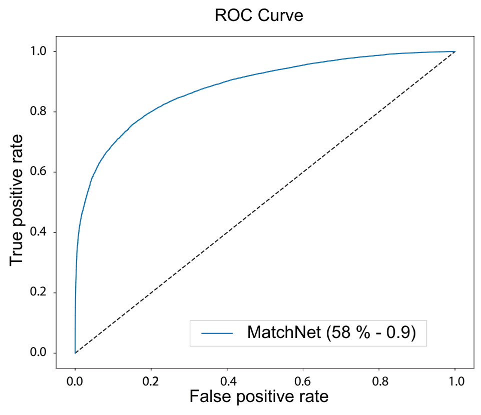
It can be observed, seeing Figures 4.13a and 4.13d, that the two best models are **MatchNet** and **Siamese AlexNet**. They respectively have a FPR95 of 19.7% and 17.1%. It is quite surprising MatchNet performs quite well in comparison to the **TransferNet networks**. As a reminder, it takes as input grayscale images of size 64×64. Nevertheless, when comparing Figures 4.12, it can be noticed that **MatchNet** performs very poorly with a FPR95 of 58%.

Finally, the FPR95 provides only a way to compare the networks quickly and assess their performance easily. It is important to remember that only the patches coming from the available landmarks were at stake during the training. All the patches that can be extracted from the segmentation were not taken into account in the dissimilar pairs or in the triplets as a negative input.

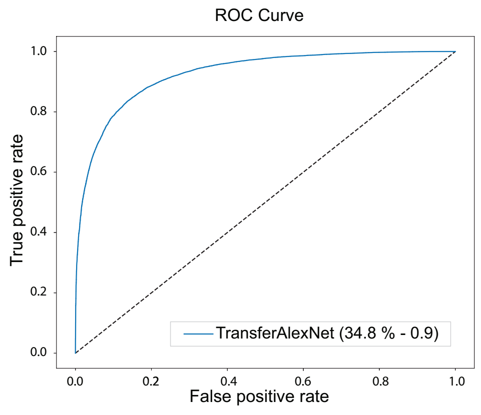
4.4 Comparison of both Approaches

In order to compare the two approaches (the use of pre-trained networks with a fixed similarity measure versus the use of networks trained on the dataset), an identical evaluation procedure must be set. All the patches are of size 300×300 pixels and the $shift$ parameter for the segmentation is 75.

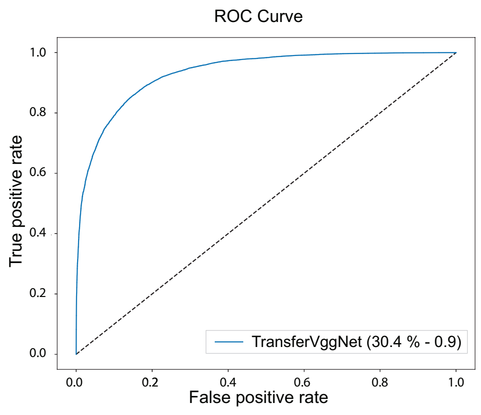
As the final goal is primarily to identify matching pairs, the top-k accuracy ($k \in \{1, 5\}$) will be used. In order to not biased the results in favor of the trained networks, only the tissues being part of the testing set are going to be used. It makes a total of 35 pairs of images and thus a total of 70 evaluations (as a each element of a pair plays the role of the



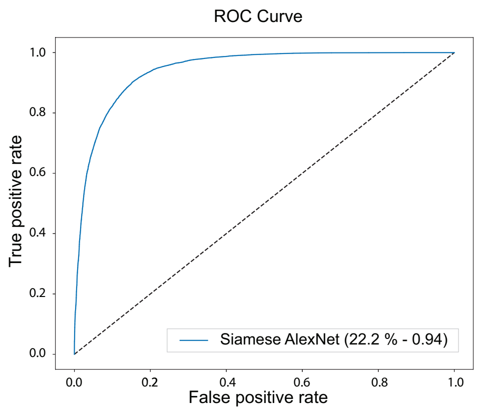
(a) MatchNet



(b) TransferNet with AlexNet



(c) TransferNet with VGGNet



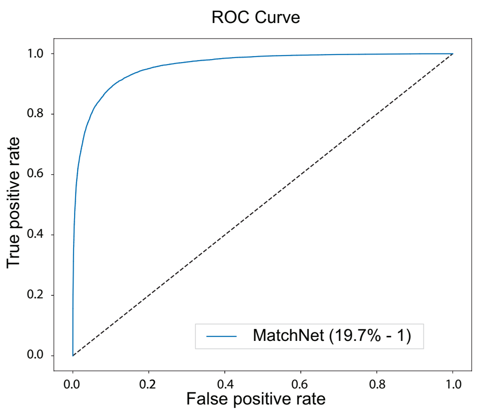
(d) Siamese AlexNet

Figure 4.12: ROC curves obtained on the evaluating set for the different models

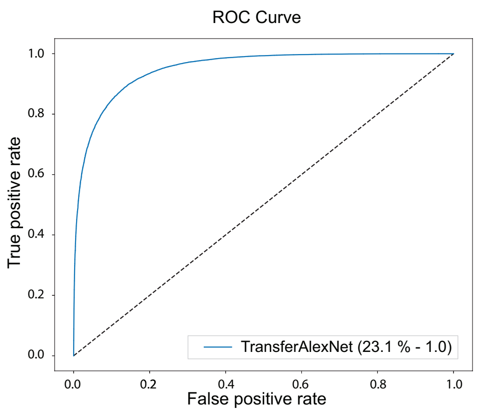
target). Furthermore, the accuracy will be calculated with a segmented target image. The results for the two approaches can be found in Table 4.8.

Without appeal, the results are in favour of the use of the first approach, i.e. the use of networks pre-trained on ImageNet with a fixed similarity measure. On the other hand, the results obtained with the second approach are quite disappointing. **Siamese AlexNet** obtained the best results among the generated models. It is consistent with the results given in the previous section. Despite the fact that **MatchNet** has only the second position among the generated models, it would be worth to investigate network architectures using grayscale and larger input images. It can be easily done by taking the feature network from AlexNet and reducing the number of input channels to one. The models had nevertheless learnt as they do not act as random guesser.

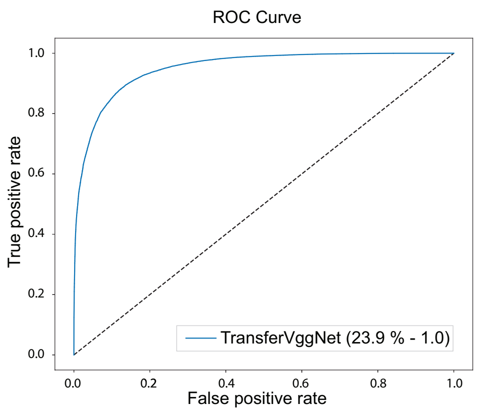
These poor results certainly come from a lack of data. The separation of the dataset into three sets has already been a bit challenging due to a lack of diversity in the tissues which, as a reminder, could not contain the same tissue in different sets. For instance, the training set is composed of 18% of similar patches coming from 6 sections (3 stained with PAS, 2 with CD31 and 1 with SMA) coming from a single mice kidney sample. An enlarged portion from two section samples with the landmarks (the colored dots) are shown in the 4.14. The deformations due to the sectioning process can clearly be observed. The results obtained with DenseNet-201 thanks to the first approach are catastrophic with a result of 0.173 (resp. 0.28) for the top-1-accuracy (resp. top-5-accuracy) on the $C_6^2 = 15$ pairs of images (thus a total of $15 \times 131 = 1965$ pairs of **similar** patches). 131 is the number of landmarks for a mice kidney sample.



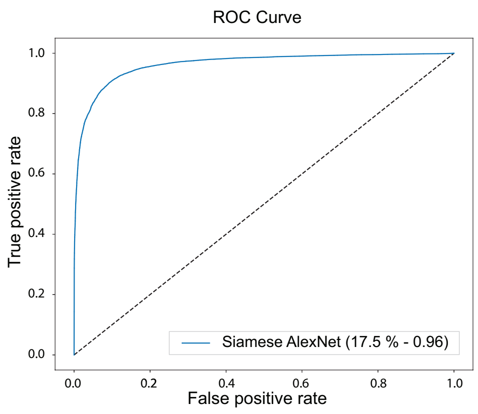
(a) MatchNet



(b) TransferNet with AlexNet

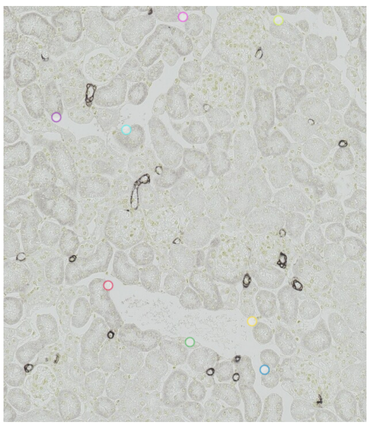


(c) TransferNet with VGGNet

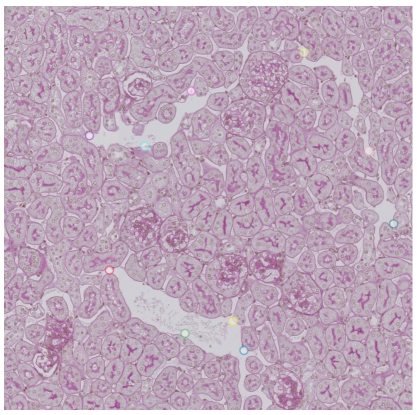


(d) Siamese AlexNet

Figure 4.13: ROC curves obtained on the testing set for the different models



(a) $\tau_{\text{SMA}}^{\text{Mice kidney-1}}$



(b) $\tau_{\text{PAS}}^{\text{Mice kidney-1}}$

Figure 4.14: Portion of $\tau^{\text{Mice kidney-1}}$ samples

Approach	**Architecture**	**Top-k accuracy**	
		$k = 1$	$k = 5$
1	AlexNet (BN)	0.344	0.501
	VGGNet-19 (BN)	0.312	0.46
	DenseNet-201	0.461	0.6
2	MatchNet	0.205	0.414
	TransferAlexNet	0.141	0.34
	TransferVGGNet	0.16	0.348
	Siamese AlexNet	0.339	0.5215

Table 4.8: Summary of the total top-k accuracy for both the pre-trained networks and the generated models considering only the tissues coming from the testing set

Chapter 5

Conclusion

In the medical and biological field, effective annotation tools have always been of a great importance. Indeed, manual annotation is a very time-consuming process. This time could be invested more effectively. Thanks to progress in the technological field especially, in computer vision, imagery and machine learning, these tools do not stop to be improved. Additionally, the development of these technologies has not reached its maturity in most cases.

In digital pathology, one may need to highlight significant structures in high resolution whole-slide images of the same tissue (this means different slices stained with an identical or different dye). The highlight can be done by drawing a bounding box or even a polygon tool around the element of interest. Instead of repeating the process manually in the different sections, this can possibly be automated. One way to solve this problem is to use image registration. Once the images aligned, the structures will overlap. Here a different approach known as patch-matching, which is not exclusive to image registration, is studied. The goal is to find matching bounding boxes (the patches) from two images (a reference and a target) thanks to a similarity score. It is worth recalling that this is a considerable challenge as there are differences in colors and content due to deformation when sectioning a tissue for instance. Furthermore, the dataset used in this study is strongly limited and misses some resources (such as full scale images or missing dyes information). This dataset has not been primarily designed for this study. Furthermore, each tissue should be carefully analysed as the significant structures in each of them does not have the same scale size which leads to differences in the patch information. This would require an expert or a person familiar with histological images.

The study have consisted in taking a reference patch from one image and comparing it to patches in a target segmented image. The pair of patches obtaining the best score is said to be matching. The study has been conducted in the following way.

First of all, some networks pre-trained on the ImageNet database have been used as the feature extractor of a Siamese network and compared. Cosine similarity has been used for the comparison between the feature vectors. The results obtained are not remarkable and require a significant feature vector size to perform at their best. The pooling operation at the end of certain networks completely destroys the information content of the patches. The size of the patches have also been shown to play an important role. The larger, the better (to a certain extent). Nevertheless, the feature vector size does not have to be proportional to the size of the patches (which can be resized) before going through the feature extractor.

Secondly, some models with a different architecture have been designed and trained on the available dataset. Unfortunately, the obtained results clearly show a great decrease in performance. The dataset can be one source of the problem. It lacks of samples and diversity, some tissue having more dyes than others. Then, these networks have small feature vectors while the networks trained on ImageNet require large ones to perform well. Finally, only a small variety of networks have been tested.

It is still necessary to remind that the results can be further improved. Indeed, the normalization parameters have been computed using patches coming from a naive per-image segmentation, the patches have been extracted from images at different scales and the networks have been trained using architecture with a fixed size of patch on a very limited dataset. Many improvements can be brought and the analysis is far from being complete. There is still plenty of rooms for improvement.

It could be interesting to try different color spaces such HSL (hue, saturation, lightness) or HSV (hue, saturation, value) and the use of grayscale images. Then, a better segmentation technique could be used for getting both a better normalization parameters and patches extraction. Furthermore, the feature networks (the networks responsible for extracting the features from an image) are not robust to rotation. Nevertheless, there exist rotation-invariant networks (see [43, 44, 45]). As the histological images of different sections are prone to suffer from a difference in orientation, it may be interesting to try these types of networks as feature extractors. The dataset comes from the Automatic Non-rigid Histological Image Registration (ANHIR) challenge. Only the training as well as the evaluating set have been provided (in order to prevent overfitting on the testing images). Since the challenge is now over, it would be of a great interest to add the testing set to the data in order to increase the small and present dataset. Concerning the network architectures, different ones should be investigated and the use of greater feature vector should be considered

Appendices

Appendix A

Architectures of the convolution neural networks used in the second approach

A.1 Conventions

Here is a small list of conventions used to describe the architecture of the convolutional neural networks:

- Convolution layer with ReLu: Conv(channel in, channel out, kernel size, stride, padding)
- Max Pooling Layer: MaxPool(kernel size, stride)
- Adaptive Average Pooling Layer: AdaptiveAvgPool(output size)
- Linear layer with ReLu activation: Linear(number of features in, number of features out)

A.2 TransferAlexNet

TransferAlexNet (TransferNet with AlexNet):

- Feature Network:
  - AlexNet: Conv(3, 64, 11, 4, 2) $\rightarrow$ MaxPool(3,2) $\rightarrow$ Conv(64, 192, 5, 1, 2) $\rightarrow$ MaxPool(3,2) $\rightarrow$ Conv(192, 384, 3, 1, 1) $\rightarrow$ Conv(384, 256, 3, 1, 1) $\rightarrow$ Conv(256, 256, 3, 1, 1) $\rightarrow$ MaxPool(3,2)
  - 1×1 Conv: Conv(256, 16, 1, 1, 0)
  - Pooling: AdaptiveAvgPool(8)
- Metric Network: Linear(2048, 128) $\rightarrow$ Linear(128, 64) $\rightarrow$ Linear(64, 2) $\rightarrow$ Softmax

A.3 TransferVGGNet

TransferVGGNet (TransferNet with VGGNet) (all convolution layers have a Batch Normalization layer before the ReLu activation and have a kernel size=3, stride=1 and padding=1 and all max pooling layer have a kernel size=2, and stride=2):

- Feature Network:
  - VGGNet: Conv(3, 64) → Conv(64, 64) → MaxPool → Conv(64, 128) → Conv(128, 128) → MaxPool → Conv(128, 256) → Conv(256, 256) → Conv(256, 256) → MaxPool → Conv(256, 512) → Conv(512, 512) → Conv(512, 512) → MaxPool → Conv(512, 512) → Conv(512, 512) → Conv(512, 512) → MaxPool
  - 1×1 Conv: Conv(512, 64, 1, 1, 0)
  - Pooling: AdaptiveAvgPool(6)
- Metric Network: Linear(4608, 384) → Linear(384, 64) → Linear(64, 2) → Softmax

A.4 Siamese AlexNet

- AlexNet: same as described in TransferNet with AlexNet (see A.2)
- 1×1 Conv: Conv(256, 128, 1, 1, 0)
- Pooling: AdaptiveAvgPool(8)
- FC: Linear(8192, 4096) → Linear(4096, 4096) → Linear(4096, 2048)
- L2-normalization

Bibliography

[1] Cytomine official website. https://cytomine.be/.

[2] Cytomine cooperative company. https://doc.cytomine.be/.

[3] 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. *Bioinformatics*, 32 9:1395–401, 2016.

[4] Cytomine. https://cytomine.coop/.

[5] Raphaël Marée, Romain Mormont, Remy Vandaele, Ulysse Rubens, Pierre Geurts, and Louis Wehenkel. Cytomine research @ montefiore, uliège. https://uliege.cytomine.org/.

[6] Wikipedia contributors. Image registration — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Image_registration&oldid=902788480, 2019. [Online; accessed 16-August-2019].

[7] Jiří Borovec, Arrate Muñoz Barrutia, and Jan Kybic. Benchmarking of image registration methods for differently stained histological slides. In *Conference: IEEE International Conference on Image Processing (ICIP)*, 10 2018.

[8] Rodrigo Fernandez-Gonzalez, Arthur Jones, Enrique Garcia-Rodriguez, Ping Yuan Chen, Adam Idica, Stephen Lockett, Mary Barcellos-Hoff, and Carlos Ortiz-de Solorzano. System for combined three-dimensional morphological and molecular analysis of thick tissue specimens. *Microscopy research and technique*, 59:522–30, 12 2002.

[9] L. Gupta, B. M. Klinkhammer, P. Boor, D. Merhof, and M. Gadermayr. Stain independent segmentation of whole slide images: A case study in renal histology. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 1360–1364, April 2018.

[10] Mikhailov I., Danilova N., and Malkov P. The immune microenvironment of various histological types of ebv-associated gastric cancer. In *Virchows Archiv*, page 473, April 2018.

[11] Bueno G. and Deniz O. Aidpath: Academia & industry: Collaboration for digital pathology. http://aidpath.eu/?page_id=279.

[12] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[13] Naomi S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *American Statistician*, 46(3):175–185, 1 1992.

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

[15] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[16] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[21] Imagenet. http://www.image-net.org/.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[25] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[26] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):43–57, Jan 2011.

[27] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, June 2006.

[28] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[29] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, June 2005.

[30] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *2009 IEEE 12th International Conference on Computer Vision*, pages 349–356, Sep. 2009.

[31] Matthew Brown and David Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74:59–73, 08 2007.

[32] A. Buades, B. Coll, and J. . Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65 vol. 2, June 2005.

[33] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. *CoRR*, abs/1504.03641, 2015.

[34] Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3279–3286, June 2015.

[35] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Image patch matching using convolutional descriptors with euclidean distance. *CoRR*, abs/1710.11359, 2017.

[36] Akila Pemasiri, Kien Nguyen, Sridha Sridharan, and Clinton Fookes. Sparse overcomplete patch matching. *CoRR*, abs/1806.03556, 2018.

[37] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):43–57, Jan 2011.

[38] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2001.

[39] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[40] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[41] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 118–126, Dec 2015.

[42] Pytorch. https://pytorch.org/.

[43] B. Fasel and D. Gatica-Perez. Rotation-invariant neoperceptron. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 336–339, Aug 2006.

[44] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. *CoRR*, abs/1604.06720, 2016.

[45] G. Cheng, P. Zhou, and J. Han. Rifd-cnn: Rotation-invariant and fisher discriminative convolutional neural networks for object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2884–2893, June 2016.

[46] Pillow. https://pillow.readthedocs.io/en/stable/.