UNIVERSITAT DE BARCELONA

Bachelor's Thesis

Double bachelor's degree in Mathematics and Computer Science

Mathematics and Computer Science faculty
Universitat de Barcelona

Unsupervised representation learning for medical imaging

Author: Albert Garcia Sanchez

Director:	**Dr. Santiago Segui**
Developed in:	**Computer Science department, Universitat de Barcelona**

Barcelona, **June 27, 2019**

Contents

Abstract

The aim of this project is to achieve good image representations through the use of Deep Learning technologies which are part of a broader family of Artificial Intelligence methods named Machine Learning. These image representations are vectors of representative float numbers called embeddings that, for the project, are focused on the entire digestive aparatus for medical purposes. Triplet loss is used altogether with a state of the art Convolutional Neural Network, ResNet, in order to achieve this goal. A series of tests are done in order to compare different training approaches of the ResNet model, seeking for the best image representations in the domain of the digestive aparatus. It is shown that ImageNet transfer learning underperforms with respect to not applying transfer learning for really specialized domains. To conclude, it is found that unsupervised representation learning through the use of Triplet loss enables transfer learning for specialized image domains such as the digestive aparatus.

Chapter 1

Introduction

1.1 Machine Learning and Deep Learning

Machine Learning, a subfield of Artificial Intelligence (shortened AI), provides the ability to develop techniques and models that learn from data yielded by the digital revolution. Due to the great improvement of such techniques and the data availability, Machine Learning has become broadly used in a diverse number of fields, e.g., medicine, marketing, robotics, consulting and many others.

Inside Machine Learning, it must be noted the recent success of Deep Learning techniques. These techniques have greatly enhanced the precision of results and have provided solutions to problems which up until now could not be proposed. This type of learning technique has yielded a great improvement in classification and segmentation of images, in the field of natural language processing and translation as well as in the diagnosis of diseases. More specifically, a Convolutional Neural Network (CNN) is a class of Deep Learning neural network commonly applied to analyzing visual imagery.

Even though Deep Learning could be thought as the solution to a large set of problems, specially in the field of medical imaging, Neural Networks still hold imperfections to be solved such as the following:

1. Although they tend to generate good results, there is no clear methodology to understand how and what they learn. For this reason, Neural Networks are used as a **black box**, i.e., given a certain input data the Neural Network processes it and generates the corresponding prediction, all of this without exactly knowing how the model reaches the output result.

2. The usage of any Machine Learning algorithm brings up a specific question: what happens with erroneous predictions? This question comes up due to the lack of **uncertainty metrics** in these algorithms. Given a correct prediction, there is no possible way to estimate the degree of confidence the Neural Network has regarding

its prediction. More specifically, this confidence can be low in certain situations such as when the data has notable noise or when the model has a determined structure that yields less confidence in its predictions.

3. In lots of cases, and specially in the medicine field, the data comes from a unique source captured with the same device. This fact makes the system sensitive to changes applied to the source of information and to the capturing device which can easily produce a decay in the precision of the results. That is why **domain adaptation**, a set of techniques used to adapt a model to a certain domain, must be applied.

These three points are, without any doubt, the current worries that anyone who wants to dive deeper into the development and research of Deep Learning has to face. Machine Learning and Deep Learning are outstanding trending technologies which will and have already revolutionized the world as we know it.

1.2 Motivation

Human health has been evolving from the beginning of the human being and since then a vast range of diseases, bacteria, viruses and other afflictions have been fought. Never before have we had so many resources and technology at the disposal of human health and medicine. For this reason, nowadays is when medicine is improving at its fastest pace and yet we still have to sand up to grave diseases such as cancer among others. Cancer is recognized as a major public health problem at worldwide scale. Even more impacting, cancer is expected to rank as the leading cause of death as well as to become the most important barrier in the increase of life expectancy in the 21st century [1].
A great percentage of humans around the world suffer from GastroIntestinal (GI) tract related diseases, one of them being colon cancer. In 2018, Global Cancer Statistics indicated that colorectal (9.2% of the total cases), and stomach cancer (8.2%) ranked as the third and fourth most common diagnosed cancer as well as being the third leading cause of death by cancer [1].

As with the vast majority of cancer cases, the detection of the previous GI related cancers in early stages drastically increases the probability of survival for the patient. In the case of detecting colorectal cancer, colonoscopy is used, while for stomach, esophagus and duodenum cancer upper endoscopy is the common method. Nevertheless, new and innovative methods are constantly emerging and improving.
In 2000, *Given Imaging* developed the Wireless Capsule Endoscopy (WCE) [2]. This capsule, provided with a camera, travels through the entire GI tract of the body and sends images wirelessly to a receiver for later image analysis. The two most important advantages are the following:

- Allows inner visualization of the GI tract
- Minimal invasion for the patient as well as low preparation and discomfort

WCE has been used for the examination of various GI diseases in the small intestine including Chron's disease, tumors, ulcers, polyps, bleedings and others. The WCE system is composed by:

- The WCE (Figure 1.1) is a disposable pill-shaped data recorder. The main components are: a lens, an image sensor, light sources, batteries and a radio transmitter system. After being swallowed, the capsule travels through all the GI tract, starting from the espohagus then dropping into the stomach, following to the intestines and finally being excreted from the anus.
- The workstation is where the video recorded by the WCE is downloaded for further analysis.

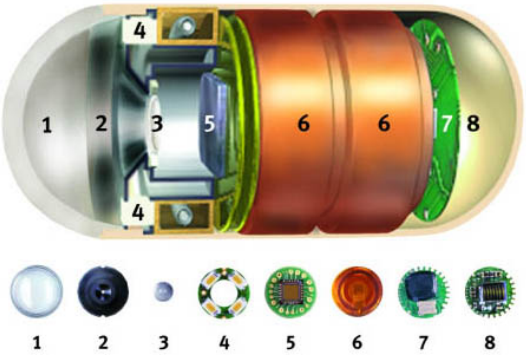

Figure 1.1: Components of the WCE. 1) Optical Cupule 2) Lens holder 3) Lens 4) LEDs illuminators 5) CMOS Image 6) Battery 7) Transmitter ASIC (Application Specific Integrated Circuit) 8) Transmitter.

Performing the WCE test spans for at least eight hours or until the batteries run out. In average, the number of frames per video yielded by this test is 55000. The time needed for the examination of the entire video is about three hours, which without a doubt is a time-consuming labor. For this reason, computer automated diagnosis systems are being developed for WCE videos, using state of the art Convolutional Neural Networks in addition to other techniques. Regrettably, the analysis of WCE videos still offers challenging difficulties such as low quality images, changes in illumination and variations in colours, textures and contents. But apart from these obstacles, there is also the problem of *Transfer Learning* and *domain adaptation* for this specific context.

Transfer learning is a machine learning technique where a model developed and trained for a task (called pre-trained model) is reused as the starting point for a model on a second task. In general, the former task aims for a broader goal such as the classification of every-day common items, objects, places, actions or ideas. On the other hand, the second model has a more specific objective and it is the one used for production or research purposes.

In the case of image transfer learning, the data used in the training of the pre-trained model could come from *ImageNet*, a popular public image database with more than 14 million images covering a huge range of common categories [3]. As described, this database is used for the training of the pre-trained model in order to provide the latter one with a starting point adapted to the most common images (common categories in everyday life). Nowadays, this method has almost become a must use for any project involving convolutional neurual networks. In most cases, the detected results comparing the usage and not usage of Transfer Learning are similar to the Figure 1.2.

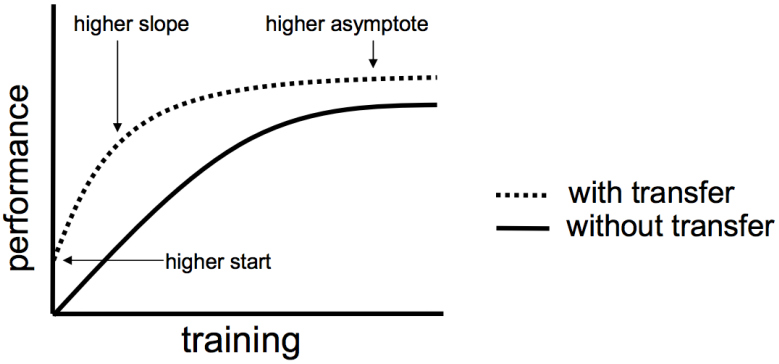

Figure 1.2: Comparison of performance between using and not using Transfer Learning.

Moreover, Transfer Learning can also decrease the time needed for the final model to be trained. In spite of all the advantages and improvements in using this technique, there are a number of situations in which Transfer Learning does not yield the expected improvements:

- When the pre-trained model has been trained with data from a specific domain not sufficiently related to the domain of the final model.

- When the pre-trained model has been trained with data from a broad and general domain while the second model trains with data from an extremely narrow and specific domain.

One case in which the second point occurs is in the field of medical imaging analysis due to its highly specialized domain [4]. This is exactly what arises when trying to use Transfer Learning in the development of the model for the WCE videos. Hence, data adaptation techniques must be used in order to properly implement Transfer Learning in the domain of WCE images. Therefore, the main motivation of this project is to further improve the results of the systems which will diagnose GI diseases from WCE data, consequently contributing to the enhancement of human health in the GI tract field including colon and stomach cancer.

1.3 Goals

This project is carried out under the supervision of a research team at the Mathematics and Computer Science faculty (Universitat de Barcelona) involved in the development of the previously mentioned diagnosis system for WCE data. In consequence, the implementation of the project's goal and the medical images used in the process are under a confidential agreement and their details cannot be fully disclosed.
The main objective of this project is to **make Transfer Learning possible in the domain of WCE data from a data adaptation perspective through the implementation of a CNN.** In order to fulfill this goal, the final trained model should meet a series of characteristics:

- Similar images should have similar representations regardless of their source, i.e, the patient, the device and other minor conditions.
- Non-similar images should have different representations regardless of the same factors as in the previous point.
- Given two similar images and a third one different from these two, the representation of the latter image should be different from the representations of the two first ones. In other words, there must be a continuous and clear distinction between a set of similar but not equal images and images not similar to these ones in conjunction. The representations must form clusters of similar images.

The difficulty of achieving these characteristics is stricly dependent on the data, which in this case comes from WCE videos. In this specific domain, the greatest challenges are unlabeled data and the undetermined amount of classes that appear in WCE videos.

Chapter 2

Plannification

In order to achieve the previous goals, as well as acquiring the needed knowledge, a continuous and gradual development process has been designed. This process follows a series of ordered tasks which are the following:

- Learning of the fundamentals about Machine Learning, Deep Learning and Convolutional Neural Networks
- Learning and understanding of the usage and implementation of the Triplet loss function
- Familiarization with the development environment
- Learning and practice of the programming tools used for the development
- Finding a fairly simple domain in which the goals can be achieved with a lighter implementation, implement, train and evaluate the model.
- Achieve the same goals in a more related domain, implement, train and evaluate the model in this data domain closer to the domain of WCE data.
- Implement and train the model for WCE data, test and evaluate different variations of this model.
- Memory writting.

The time allotted in days as well as the approximate start and end dates for each task are shown in the following Gantt diagram. Note that the start and end dates are not strictly fixed nor labelend in the diagram due to slight variations in the schedule during the actual development.

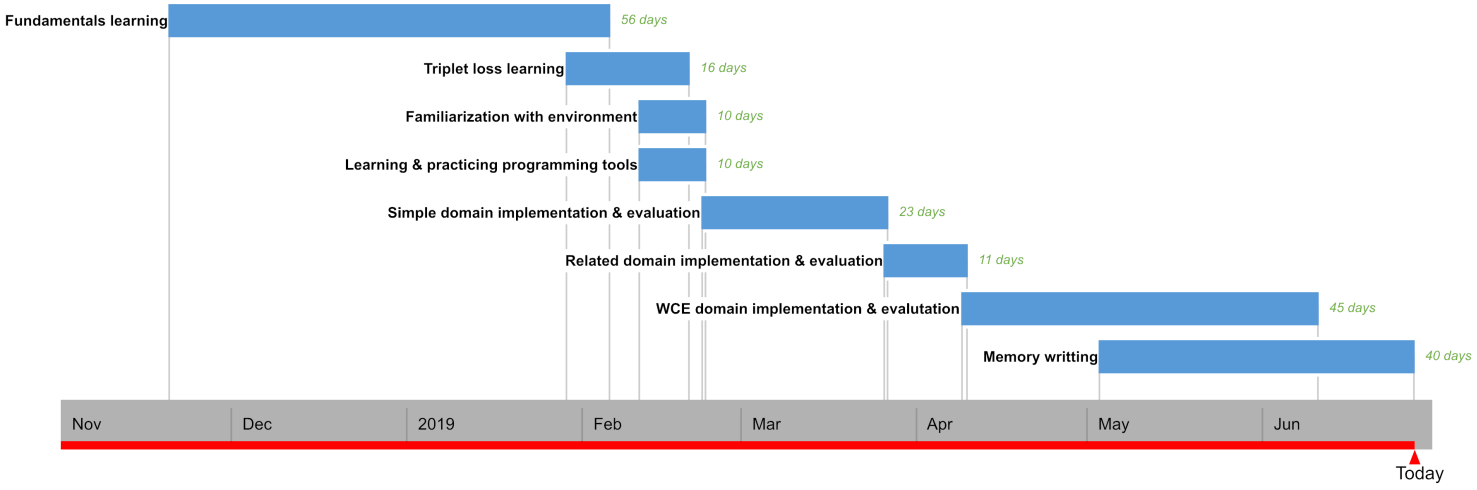

Figure 2.1: Gantt chart showing the allotted days and the ranging dates for each plannified task.

Chapter 3

Background

3.1 Neural Networks

Neural Networks have successfully accomplished excellent results in an unnumerable amount of tasks, such as in classification. Despite this, the training process of these networks involve thounds, or even millions, of labeled examples. Fortunately enough for the aim of this project, there is a large quantity of WCE videos available, each one containing a great amount of valuable frames.
Artifical Neural Network (ANN) is a biologically-inspired programming paradigm which provides a computer the ability to learn from observing data. In essence, ANN is a set of densely interconnected adaptive processing units (artificial neurons). These networks have a very important feature, they have an adaptive nature, where "learning by example" replaces the traditional methodology of "programming" for solving problems [5].

In order to understand the following chapters and the implementation of the project, the fundamentals of Neural Networks, and more specifically Convolutional Neural Networks, are explained in the following lines.

An **artificial neuron** is a computational processing unit which performs a specific operation:

$$y = g(\vec{w} \cdot \vec{x}) \tag{3.1}$$

where $\vec{x}$ is the input data, $\vec{w}$ are the weights which will continuously adapt to the viewed examples as the training process keeps going, $< \cdot >$ is the standard dot product, $g(\cdot)$ is the activation function and y is the output response of the neuron (also called stimulus or signal). The changes produced in the weights $\vec{w}$ during the training process have the objective of optimizing the neuron in order to generate signals similar to the expected ones for each viewed example. The more times a neuron sees a specific example, the more similar the output signal will be to the expected one.

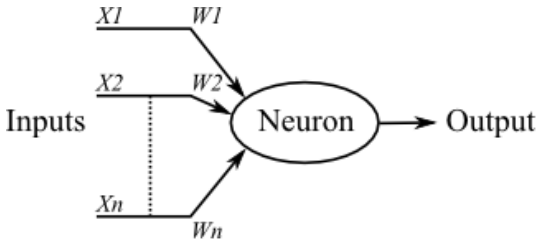

Figure 3.1: An artificial neuron with n inputs and n weights.

A neural network is a set of these artifical neurons structured in a network-like architecture, all of them with the same purpose which is to optimize (minimize in this case) a determined objective function called loss. The loss function represents how far the output is to the correct result. By convention, when the loss function evaluates to 0 means that the output of the model is equal to the correct result for a specific input. When this network is distributed by subsets of artificial neurons, one after the other, then each of these subsets is called layer. The relative position of each layer in the network provides each layer with a categorical name.

- The input layer is the first one of the network and it is the one which sees the input data unaltered.
- The output layer is the last layer and it is the one which provides the output result in a specific format.
- A hidden layer is any layer between the input and the output layers.

Apart from different layers, the connections of a layer with respect to the previous layer also determine the general structure of the neural network. When a neural network has more than 2 layers, i.e., when there exists at least one hidden layer, then it is considered as being deep.

3.1.1 Fully Connected Layer

A given layer is named Fully Connected, or dense, when each neuron of the layer is connected to all of the neurons from the previous layer. A neural network formed by only dense layers is a feedforward network, meaning that the data only flows forward and no cycles are present in the network. This type of layer implies dealing with a large amount of weights, consequently needing more memory for the network.

Having many weights in a layer implies that the layer has more capacity. Capacity, in the neural network context, can be seen as how much adaptive potential a certain model has. A problem with abusing the use of dense layers comes in the form of overfitting, which is when the network performs well in the training data but fails with unseen data of the same problem. This event usually occurs in two different situations: when the network has unnecessarily high capacity or when few data is available for training. Both of these situations are strictly dependent to the difficulty of the problem.

On the other hand, the opposite of overfitting can also occur. Called underfitting, is when a neural network performs badly in the training data as well as with unseen data. This mainly depends on the capacity of the network and on the time and data used for training. In both of these unwanted situations the model fails to generalize the solution. The desired capacity is the one which does not produce overfitting nor underfitting, therefore yielding the minimum generalization error.

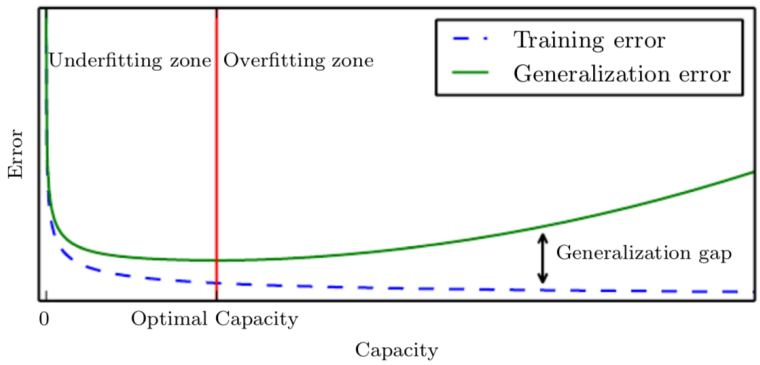

Figure 3.2: Plot showing the relation of training and generalization errors with respect to the capacity of the model.

Underfitting can be relatively easy to solve with more data, more training time or more capacity on the neural network. Overfitting on the other hand needs the use of specific techniques to be avoided. Regularisation is the technique of penalizing the neural network when using big values for the parameters. In order to achive this a regularisation term, which depends on the parameters of the network, is added to the objective function being optimized.
Another popular technique is called dropout. During the training process there is a certain probability (uniformly applied through all the neurons), usually between 10% and 70%, of totally disconnecting a given neuron from the network. In doing so, the rest of the network is forced to optimize the same problem but with less neurons, inducing the connected neurons to generalize better.

3.1.2 Convolutional Layer

A Convolutional layer is a specific type of layer used to process image data, or more generally, any data presented in the format of a tensor. The convolution operation is applied to the input data. This operation is based in sliding a filter, with a certain number of dimensions, over the input tensor and then calculating the element-wise product between the values of the input tensor and the values of the filter. These filters are usually defined to be squared (all of their dimensions equal) and relatively small. During the implementation of such layers the developer has to define the dimensions of the filters, the amount of filters to use per layer, the sride and the padding.

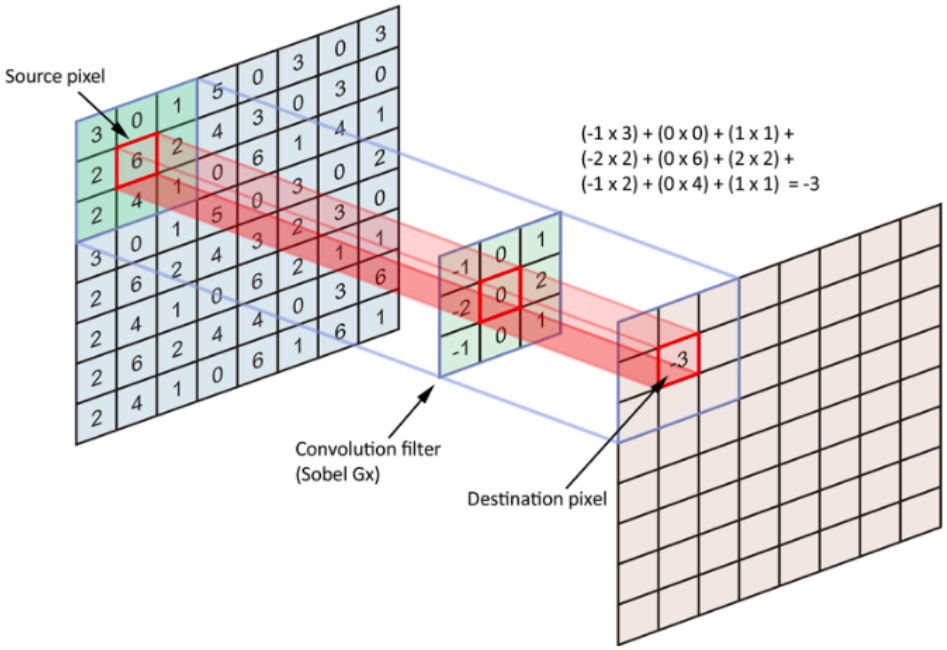

Figure 3.3: Convolution operation example with a 3×3 filter.

Less parameters are required compared to dense layers and more meaninguful data can be extracted from convolutional layers. This is due to the fact that the convolution operation can easily detect patterns, boundaries and spatial features. When dealing with image data and in conjunction with convolutional layers, pooling layers are normally used and in some cases batch normalization layers are also used.

3.1.3 Pooling Layer

Pooling layers follow Convolutional layers and substantially reduce the dimensions of a given input image in a non-linear way. The process applied maintains spacial properties and the relationss in the image. When defining a pooling layer the inner operation applied (maximum, minimum, average or stochastic) as well as the stride must be defined.

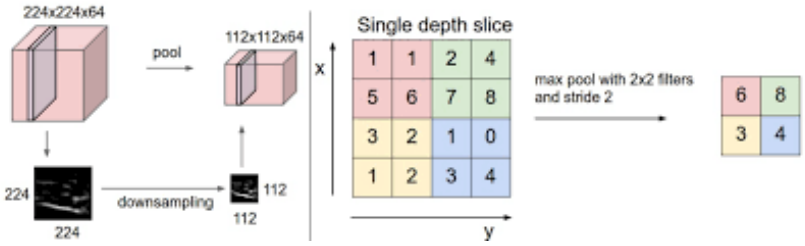

Figure 3.4: Pooling layer with 2×2 filters, using the maximum operation and a 2 stride.

3.1.4 Batch Normalization Layer

Batch normalization layers normalize their inputs by moving the mean value to 0 and the standard deviation to 1. These layers are introduced after a convolutional layer and before the activation function, even though they are not always used. The use of batch normalization provides stabilisation of the gradient, slightly reduces overfitting and reduces the training time.

A convolutional neural network (CNN) is a neural network formed by convolutional layers, pooling layers and in some cases batch normalization layers. A CNN may sometimes come with a Fully Connected layer as the output layer for the ultimate purpose of the network, e.g., for classification purposes. These are all the fundamental concepts of neural networks and convolutional neural networks needed for the implementation of the project's goal.

Chapter 4

Implementation

4.1 Preparation

Before anything else, the ideal environment and tools must be selected for the development of the desired CNN. For this project *Google Colaboratoy* has been selected as the best educational and easy-to-use environment for Neural Network development.
Google Colaboratory, shortened Google Colab or just Colab, is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. The developing interface is available through any browser, preferably with Google Chrome. Colab offers the possibility to use Python 2 or 3 as well as the possibility to select the hardware accelerator, TPU (Tensor Processing Unit) or GPU based. For the purpose of this project Python 3 and GPU accelerator have been selected. The resumed hardware specifications for Google Colaboratory are the following:

- CPU: Intel(R) Xeon(R) CPU @ 2.20GHz, 1 Socket, 1 Core per socket, 2 Threads per Core
- GPU: Nvidia Tesla T4 16GB VRAM GDDR6 (designed for Deep Learning applications)
- RAM: 12 GB
- Disk: 360 GB

Despite the free usage of this hardware there is a notable limitation. Whenever working with the free Colab version, a Virtual Machine (also known as a session) is created for hosting the notebook. After 12 hours of having this session open and running, Colab erases the Disk, RAM, VRAM and CPU cache alloted. Regardless of this limitation, the advantages of working in the Colab environment widely overcome the 12 hours session limit.

In combination with the Google Colaboratory environment, the *TensorFlow* Python API r1.13 has been selected for the programming development of the Neural Network. TensorFlow is a widely used end-to-end open source platform for machine learning development. Its core characteristics are:

- The **Tensor** is the basic data unit used and can be considered as the heart of the framework. It is defined as a mathematical object analogous but more general than a vector. A certain tensor has a property called rank which defines the number of dimensions of the tensor, e.g., a rank 0 tensor is a scalar, a rank 1 tensor is a vector, a rank 2 tensor is a matrix, etc.
- A combination of **Operations** create a neural network. These operations take a tensor as the input and output the processed tensor, therefore a neural network consistently works with tensors. CPUs and GPUs can be used for computation.
- All of the operations and their connections form a Computational **Graph** due to the fact that an operation is implemented as a class (storing useful information). The power of neural networks lies in the structured chained operations.
- **Auto-differentiation** is the technique of computing gradients from the graph, making possible the optimization of the learning parameters during training.
- The framework allows various **programming languages** to code neural networks, such as Python, C++, Java and others.

The main characteristics of a CNN must be defined before its implementation. These are:

- The objective function, generally called loss, which is the function to minimize during the training process in order to achieve the desired goal. The goal of the neural network is to minimize the loss in such a way (adapting the learning parameters) that it evaluates to 0, meaning that the output of the neural network exactly matches the desired output. Therefore, the loss function must be carefully chosen in a way that best reflects how good the output is with respect to the desired goal.
- The architecure used in a CNN strongly determines the training time (this is the time needed for a neural network to stabilize its loss function), the capacity of the model and consequently the probability of overfitting and underfitting. The architecure of a neural network can be understood as the specification of each layer, the total number of layers and the connections between them.

All the implementations and code developed in Python as well as all the disclosable data can be found in Annexed.1 .

4.2 Objective function: Triplet Loss

As described in the goals of the project, the aim of the CNN is to learn good representations for images coming from a very specific domain. The Triplet loss function is proposed as the objective function of the CNN for achieving the goal of the project. An embedding is a representative vector of real values with size n. The Triplet loss needs three embedding inputs to evaluate: an anchor (A), a positive (P) and a negative (N). The positive input has the same identity, class or label as the anchor while the negative has a different one. After processing each of these inputs through a conveniently structured CNN an embedding of size n is produced for each one of them. These embeddings can be represented in $\mathbb{R}^n$. Taking the standard distance function $||\cdot||$ in the $\mathbb{R}^n$ euclidean space, the Triplet loss is defined as:

$$\mathcal{L}_{Triplet-Loss} = max(||A - P|| - ||A - N|| + \alpha,\ 0) \tag{4.1}$$

where A, P, N are the anchor, positive and negative embeddings respectively and α is a defined margin value. (A, P, N) is the triplet of embeddings used for the Triplet loss.

When minimizing the Triplet loss value, the embeddings of the anchor and the positive inputs are being forced to be similar, while the embeddings of the anchor and the negative inputs are being forced to be different. The α margin is a constant defined to ensure that there is a minimum separation between the positive and the negative inputs with respect to the anchor input. The margin is usually a really small value but by default it is defined as 1. Because of the available time and the complexity of the goal, finding the best margin value for a specific task escapes the purpose of this project. For this reason α is always set to the default value. The distances computed in the Triplet loss could be squared yielding a different training process. For the same reasons as the ones explained for using the default margin, the distance used is not squared in any case.

Figure 4.1: Triplet loss minimizing the distance between the anchor and the positive as well as maximizing the distance between the anchor and the negative. Image font: *Cornell University*

The main reason as to why use Triplet loss as the objective function is due to its successes in face recognition and clustering of images [6]. Thorugh the use of Triplet loss in conjunction with a well structured CNN the goal of the project can be met. Hence, the three characteristics defined in the goals section reflecting the ultimate objective of the project can be met using this loss and a convenient CNN.

The Triplet loss has two different methodologies for evaluating a batch of inputs of size k. Depending on the methodology chosen and on the ultimate goal, the training time can vary as well as the final precision of the results.

- **Batch all** consists in averaging the loss over all the valid triplets of the batch. Remember that a triplet (A, P, N) is valid only when A and P have the same class and N has a different one.
- **Batch hard** consists in averaging the loss over all the hardest triplets for each anchor. Given an anchor, its hardest positive is the positive input which has the maximum distance to the anchor among all of its positives. On the other hand, the hardest negative of a given anchor is the negative input which has the minimum distance to the anchor among all of its negatives. Therefore, given an anchor A, its hardest positive P and its hardest negative N, the triplet (A, P, N) is called the hardest triplet (for the anchor A).

Regardless of the selection of the evaluation methodology, both of them must construct the needed triplets before the computation of the Triplet loss. The construction of the triplets for Batch all, i.e., the construction of all the valid triplets is a relatively easy task to perform. On the other hand, the construction of Batch hard triplets involve a bit more of computation since for each anchor the hardest positive and hardest negative must be found, i.e., for each anchor a maximum and a minimum among the distances must be found.

Apart from this, the main difference between these two methodologies is that Batch hard forces more extreme optimization due to the fact that it is exclusively looking at the hardest triplets, totally ignoring the rest of the valid triplets for a given anchor. Conversely, the Batch all approach takes into account all of the valid triplets for a given anchor and gives the same weight to all of them (standard averaging). Batch all makes Triplet loss optimize all the valid triplets of the batch while Batch hard focuses on the hardest triplets (extreme cases). Due to the high complexity, the high similarities and the significant noise of the images from WCE videos, the methodology selected is Batch all. In doing this, Triplet loss is way less sensitive to extreme and noisy triplets.

4.3 Data preprocessing

Once the objective function has been properly selected, there exists an issue that needs to be solved before designing, implementing and testing a certain CNN. As previously mentioned, the biggest challenges of working with data coming from WCE videos are the unlabaled data and the undetermined amount of classes that appear in a WCE video. These two challenges are deemed to be critical due to the fact that previous research projects which implemented Triplet loss used labeled data (categorized data). If the data is properly categorized with labels then valid triplets can be constructed for the Triplet loss. The issue with unlabeled data is that there is no methodology, a priori, of forming valid triplets.

An approach to solve this issue is designed and implemented in order to provide labels to all of the input images coming from a certain video. The process is as follows:

- Firstly, each frame of the video is normalized in terms of its image size, its colour range and the data type used for the pixels. While using the ResNet architecture, the input image size must be equal to $256 \times 256 \times 3$ pixels where the third dimension refers to the 3 colour channels RGB. On the other hand, during the first implementation, a simpler CNN architecture is used yielding different specification for the input image: the first architecure takes images of size 64×64 (note that these images are in gray scale). The colour range of a given image is normalized in order to obtain a final range of $[0.0, 1.0]$ pixel values represented with the data type Float32.
- Secondly, each frame is labeled with the number corresponding to its frame number. For example, if a given video is composed of 100 images then the first frame will be labeled with the number 0, the second frame with a 1, the third with a 2, until the last frame labeled with the number 99. When applying this step to several videos, a large enough offset number must be added to each label in order to have each frame with a totally different label.

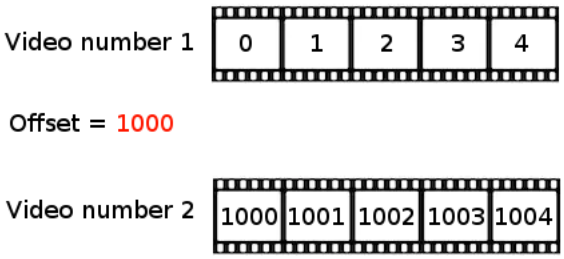

Figure 4.2: Labeling example process with 2 videos and an offset of 1000.

For this labels to be correctly used one more consideration must be addressed. Using

these labels with the previously defined Triplet loss means that each image belong to a different class (each image has a different label) and therefore for any anchor there will not be any positive image (and no valid triplets). The Triplet loss function is modified in such a way that for a given anchor labeled with a certain number x then all the positives images for this anchor are the ones labeled in the range $(x-\epsilon, x+\epsilon)$, where ϵ is a positive integer value defined as an hyperparameter of the model called "label frames tolerance". Given a pair of labeled images, checking whether these two are positive (belong to the same class) can be easily computed with a simple comparation:

$$|label1 - label2| < \epsilon \tag{4.2}$$

where label1 is the label of the first image, label2 is the label of the second image and $|\cdot|$ is the absolute value. This expression is satisfied if and only if the two images are positive.

Figure 4.3: Example using $\epsilon = 3$ of how an anchor image (gray dot) sees the rest of the images (green dot = positive image, red dot = negative image).

The justification of using this approach comes from the temporal characteristic of a video. If a video has been recorded with a certain continuity and without extreme jumps of scenes then this labeling process and identification of positives makes images temporarily close to be considered positive for the Triplet loss. How close two images must be to be considered as having the same class is determined by the ϵ hyperparameter, which in turn, must be cautiously chosen considering the framerate of the videos. This labeling process and identification of classes approach, in addition to the adapted Triplet loss, makes the CNN model learn embeddings that are similar when two images are close and different when two images are far from each other in the video. This methodology solves the two critical issues regarding the huge amount of unlabeled data.

Note that due to the largely enough offset, images from different videos will never be considered as having the same class. Due to this fact, for achieving a better training of the CNN, batches should be formed with images coming from the same video and even coming from the same location (forcing harder triplets and better learning).

4.4 First implementation: Simple domain adaptation

The simple domain chosen for the first implementation is Formula 1 races recorded from a Point Of View (POV) perspective. The main reasons for this selection are the following: relatively large amount of public POV videos of races and the fair amount of different images needed to learn. The second reason can also be seen as being a fairly simple domain in which the goals can be easily achieved, due to the high repetition of the image's characteristics. To make it even simpler for the CNN to achieve the goals, the images are converted into gray scale as well as resized to 64×64 as stated in the previous section. The architecture implemented for this task is shown in Figure 4.4.

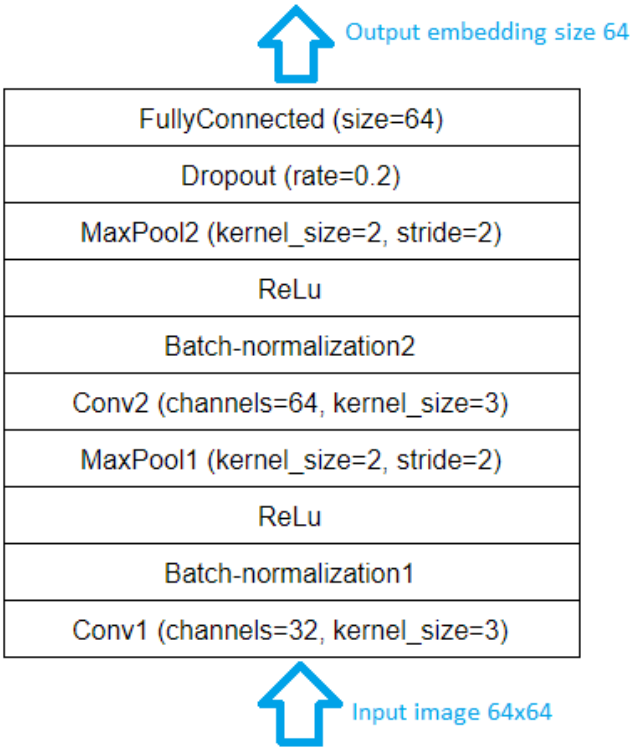

Figure 4.4: Architecture implemented for the Formula 1 representation learning.

The input of the CNN is a 64×64 gray image. The image is fed into a first block consisting of the following layers in order: convolutional layer with 32 channels and a kernel size of 3×3, a batch normalization layer, Rectified Linear Unit (activation function ReLu) and finally a max pooling layer of kernel size 2×2 and stride 2. The output of the first block is then fed into the second and last block with the same structure as the first one, except for the 64 channels used in the convolutional layer. At the end of the two blocks and after flattening the result a tensor of size $16,384$ appears. A dropout layer is applied to this tensor with a dropout rate of 0.2. Finally a fully connected layer outputs the resulting embedding of size 64.

The adapted Triplet loss in addition to the preprocessed images and a "label frames tolerance" of $\epsilon = 15$ (due to the high framerate of the Formula 1 videos) enable this CNN model to learn representative image embeddings of size 64 in the domain of Formula 1 races from a POV perspective in gray scale. The parameters of the network are optimized through minimization of the Triplet loss by an adaptive momentum optimizer (ADAM) with a fixed learning rate of 10^{-4} and a momentum exponential decay of 0.9.

4.5 Second implementation: Related domain adaptation

Colon images coming from Colonoscopy evaluations is the related domain selected for the second implementation. Even though this domain does not cover the entire GI tract, like the WCE data does, Colonoscopy Colon images are similar to the ones captured from the WCE device. This is the main reason for the selection of Colonoscopy images as the domain related to the WCE domain.

For implementation purposes, the architecture used for this task is a modification of the previous one. The architecture is modified so that the input image needs to be shaped $256 \times 256 \times 3$ (RGB channels included), therefore the input type of this architecture matches the input type of the final architecture. Other important modifications applied to the previous architecture are: deeper network (more hidden layers) and more dropout (higher dropout rate). The need for the second modification, which induces higher regularization, comes from the higher capacity of the model. The details of the implemented architecture are shown in 4.5.

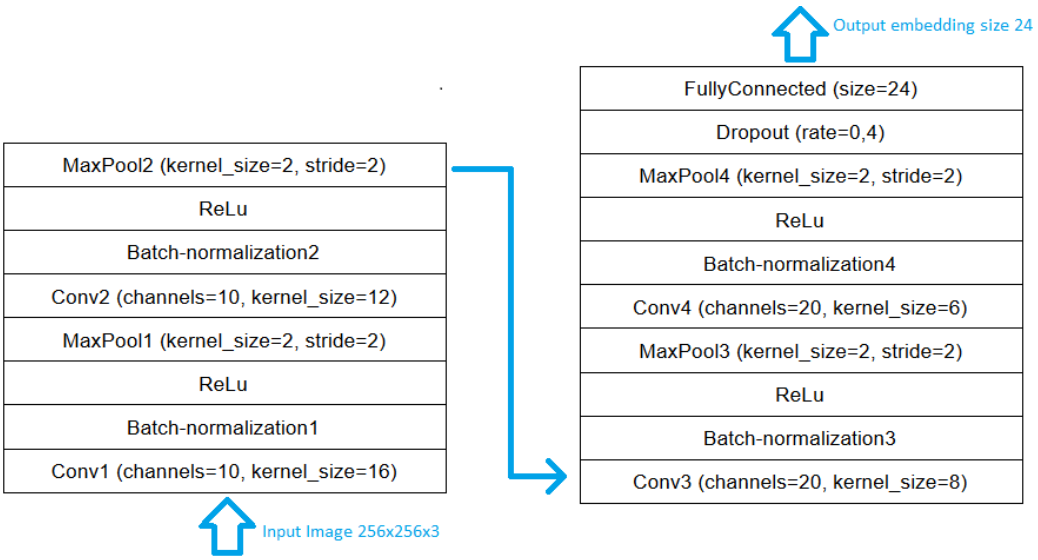

Figure 4.5: Architecture implemented for the Colonoscopy representation learning.

Note the higher dropout rate of 0.4 in comparison to the previous architecture. As previously stated, this architecture needs a stronger regularization due to the increase in its capacity, which comes from the increase in parameters. Also note the convolutional layers have different number of channels as well as kernel sizes. The two first layers have the bigger kernel sizes which induce the CNN to find bigger features in the images, while the last two kernel sizes are the smaller ones inducing small feature extraction during the last steps of the model.

Again, this architecture, the adapted Triplet loss, a "label frames tolerance" of $\epsilon = 4$ and the preprocessed images enable the CNN model to learn representative image embeddings of size 24 for the domain of Colonoscopy images in colour. The optimizer used for optimization of the parameters is the same one used in the previous implementation (ADAM), with the same configurations.

4.6 Final implementation: WCE domain adaptation

The previous implementations were designed to increasingly develop the knowledge and resources needed for the implementation of the final model, which is able to generate representatives embeddings satisfying the stated goals within the domain of WCE videos. Each video used for the training process is preprocessed as described in the Section 3.3 with a sufficiently large offset. On the other hand, the architecture used for the final model is completely different from the previous implementations. Due to the fact that the goal of the project is to implement and train a model to make able transfer learning in the specific domain of WCE data for medical purposes, this means that a state of the art CNN architecture must be used. The state of the art architecture selected is the one popularly known as ResNet, formally named as Deep Residual Network. This architecture has been widely used and exploited for its good results in image recognition among other tasks [7]. A new concept arises when using the ResNet model, the residual operation which is a technique used for solving the vanishing gradient problem. As more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero (the gradient vanishes), making the network hard to train.

The residual operation, also known as identity shortcut connection, is a network operation that takes an input tensor, skips a certain number of layers without modifying the tensor and finally adds it to the output of a specific layer. This technique effectively reduces the vanishing gradient problem although there are also other approaches for solving the same issue. The use of batch normalization and Rectified Linear Units (ReLu) also help to minimize this problem. When adding a residual operation to the network, the layers covered by this shortcut are known as a residual block. Therefore, in few words, ResNet is a deep CNN model with residual blocks.

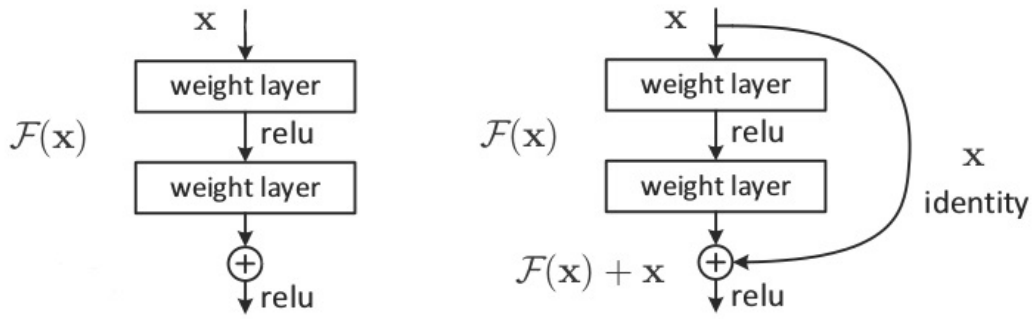

Figure 4.6: On the left a traditional block, on the right the equivalent residual block.

Since this architecture has been extensively used, it is easy to find implementations of it in open-source libraries. For practicality reasons, the implementation of the ResNet model is taken from the official open-source TensorFlow library of models [8]. In addition to the core implementation of the model in TensorFlow, some supporting functions and block implementations are taken from the open-source library called SenseTheFlow [9].

Originally, ResNet was designed to classify images, however, the goal of the project is to generate image embeddings. To achieve this a minor modification of the output layer is needed. The modified ResNet architecture implements a fully connected layer with n nodes, where n is the desired size of the embedding. Among the most popular Residual Networks, the one selected for this project is the one named ResNet50 which specifies the number of block layers used (in this case 3, 4, 6 and 3, each one consisting in a predefined residual block). The general proposed pipeline is shown in the Figure 4.7.

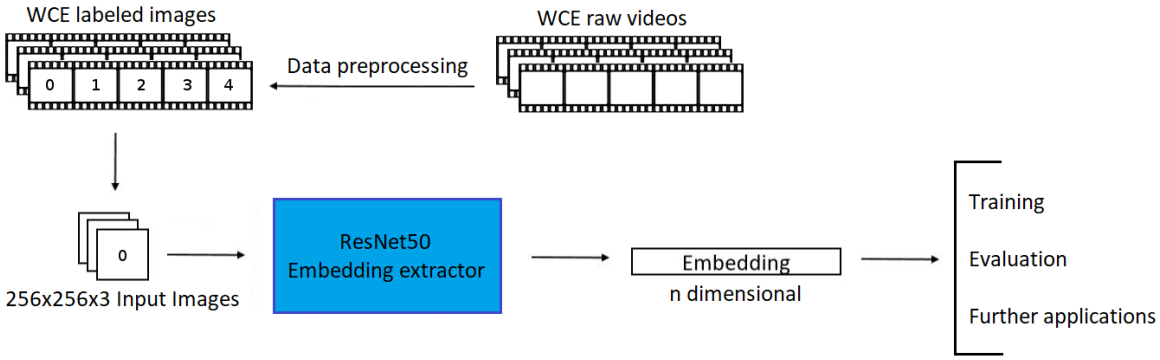

Figure 4.7: General pipeline implemented for the final task of representation learning in the WCE domain.

Note that depending on the purpose of each iteration (training, evaluation or further applications), the data preprocessing steps may be omitted starting directly from unlabeled input images. When not training the model, i.e., when evaluating or predicting results, labels are not needed for the input images. Also note that the prediction mode (exclusively getting the output embeddings) for this model is included in the "Further applications" output due to the fact that representative embeddings are used for other applications such as classification. Remember that in the project's context, the ultimate application of this embedding training is to make transfer learning possible in the WCE domain due to its high specialization.

Last specifications for the final model:

- The size of the embedding n is set to 2048 for experimental reasons and for convenience towards further applications.
- The "label frames tolerance" is set to $\epsilon = 4$ due to the low framerate of the WCE videos.
- The optimizer used is the same one as in the previous implementations (ADAM) with the same parameters.
- For the training of the ResNet50 using the adapted Triplet loss, transfer learning is performed from a ResNet50 model pretrained with the ImageNet image database. This is done for the acceleration of the training process.
- Data augmentation (DA) procedures are added as an hyperparameter (enable or disable all) for the training process. The DA image methods implemented are: 90 degree rotation, 180 degree rotation, 270 degree rotation, horizontal flip and vertical flip. If DA is enabled then when feeding training images to the CNN, each DA method will be applied to every image with a uniform probability of 50%.
- The implementation of a WCE image classifier with p classes through the use of embeddings just needs to freeze the embedding extractor model and add a Fully Connected layer at the output layer (embeddings layer) with p neurons. The loss function to optimize is the softmax cross-entropy popularly used for classification training. Given an image, after computing its embedding vector, the classifier outputs a vector of size p which in turn after applying the softmax function transforms into a vector of p probabilities (each one corresponding to the probability of being classified to a certain class). The maximum probability value corresponds to the classification of the image.
- Due to the Triplet loss being computed over relatively small batches of images (64 images per batch for this implementation), a way of ensuring that every anchor has at least a minimum number of positives within the same batch must be implemented for the training process (a minimum number of negatives is always achieved due to the low ϵ value and the size of the batches). Given an image set labeled by the preprocessing implementation (each label is the frame number) then the entire set is shuffled by blocks, which means that the set is organized in blocks of fixed size b and then all of these blocks are shuffled so that the content of each block is unaltered. Once shuffled, these blocks can be forgotten for the rest of the training process. For this implementation a value of $b = \epsilon = 4$ is chosen to always ensure a minimum number of $b - 1 = 4 - 1 = 3$ positives per anchor (the anchor is not considered a positive for itself, in fact comparing an anchor image to itself is never considered).

Chapter 5

Evaluation and results

5.1 Evaluation techniques

An incremental set of evaluation techniques have been applied from the first implementation up to the the last one. The evaluation methodologies must show or indicate how good the embeddings are with respect to the representation of an image within a specific domain. To achieve this, two general approaches have been used: a quantitative and a qualitative analysis. Taking into consideration both of these evaluation perspectives provides sufficient feedback for concluding whether the model performs as it should. The main evaluation techniques employed are the following:

- The Triplet loss itself is a direct indicator of how well the output embedding represents a unique image while comparing it to an anchor image. Remember that the Triplet loss evaluates a triplet of images (A, P, N), where the loss value shows how close the anchor-positive embedding's distance is and how far the anchor-negative embedding's distance is. Although this loss seems to be a good indicator for evaluating the results, it does not provide any insights of improvement when the CNN model is learning minor features (when the model is near its convergence).

- Different approaches of the K-Nearest Neighbours algorithm (KNN) are used for both quantitative and qualitative approaches. As stated in the goals, one characteristic that needs to be met by the trained model is the formation of clusters of similar (but not equal) images. Through the use of KNN, clusters can be formed, images with similar (or different) embeddings can be visualized and precision averages can be computed.

- Finally, one conclusive technique for evaluation comes from using the CNN trained model as the starting point for a CNN classifier within the same domain. Note that this evaluation technique is exactly the main goal of the project. The classification training and evaluation processes are strongly influenced by the embedding's representation quality, i.e., poor embedding representations yield poor classification accuracy while good embedding representations yield faster training and higher accuracy.

Further usage of the loss evaluation technique shows no clear way of concluding how good the model is performing when nearing convergence nor whether the model is improving or not. This can be seen in the Figure 5.1. Due to this lack of insights, the loss analysis technique is discarded. The last evaluation technique enumerated, using the trained embeddings for classification through transfer learning, is only applied to the last implementation because of the resources and data needed. The second evaluation technique described, the KNN based evaluation, is applied throughout all of the three implementations. The results are showed and analyzed in the following pages.

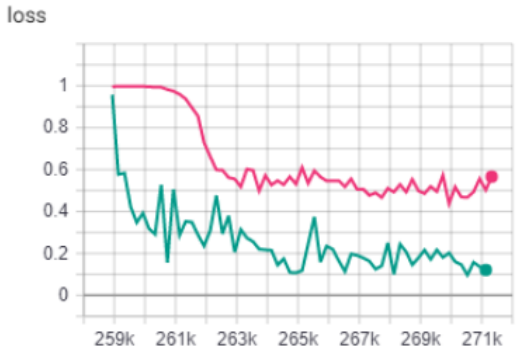

Figure 5.1: The x-axis shows the training step while the y-axis shows the Triplet loss value. The green plot represents the training loss and the pink plot represents the test loss. From the step 264k up until the last step there is no way of analyzing how the model is improving due to the test loss stabilizing around the 0.5 value. Further evaluations (such as the KNN based ones) show that after the step 264k the model keeps optimizing the embeddings.

5.2 Results and analysis

For better understanding of the results for each implementation, the training data as well as the test data used are briefly described. All the data available for a specific implementation is normally splitted into the training set and the test set in proportions around 75%-25% respectively.

5.2.1 First implementation results

For the Formula 1 domain adaptation a total of 5 POV videos from 5 different circuits have been gathered and preprocessed. The videos have lenghts between 1 minute to 2 minutes, yielding a total number of approximately 2.000 images per videos. For the training set 4 videos have been selected which means that a total number of 8.000 images are used for training. The last video is deemed as the test set with 2.000 images that the CNN will not train on. As previously explained, the Triplet loss on the test set stabilizes around the range $[0.4, 0.6]$ although it varies depending on the training process, the data available and the task. The qualitative evaluation based on KNN methods consists in taking a random image from a set of images and then showing the K nearest images based on the distances between the embeddings (in $\mathbb{R}^n$). The following pages show the qualitative evaluation performed in the test set and then in the train set.

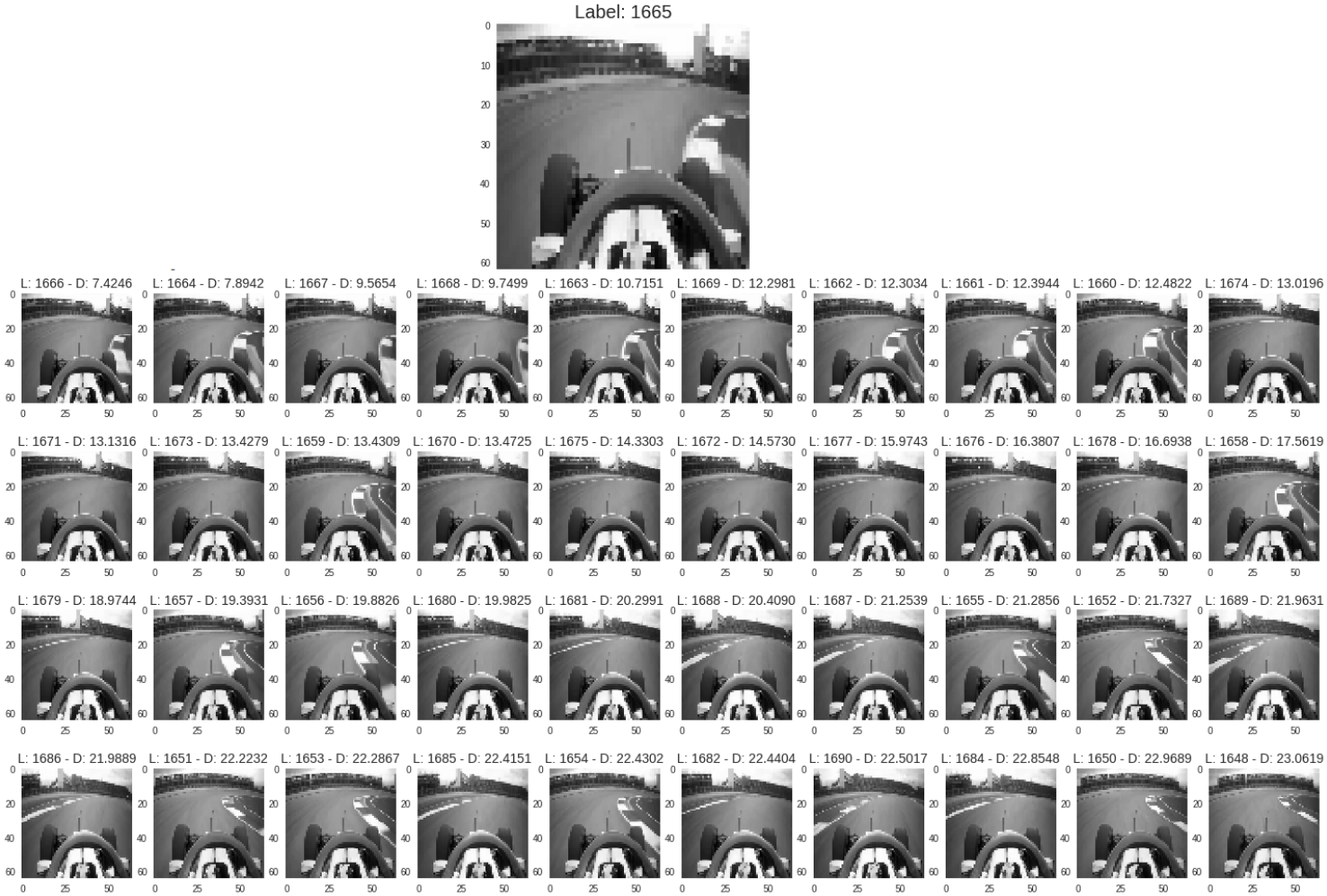

Figure 5.2: KNN qualitative evaluation on the test set. The top image is the selected image for the KNN method. Below it the 40 nearest neighbours sorted from closer to farthest (left to right and top to bottom). The 'L' stands for label and the 'D' stands for the distance between the embeddings.

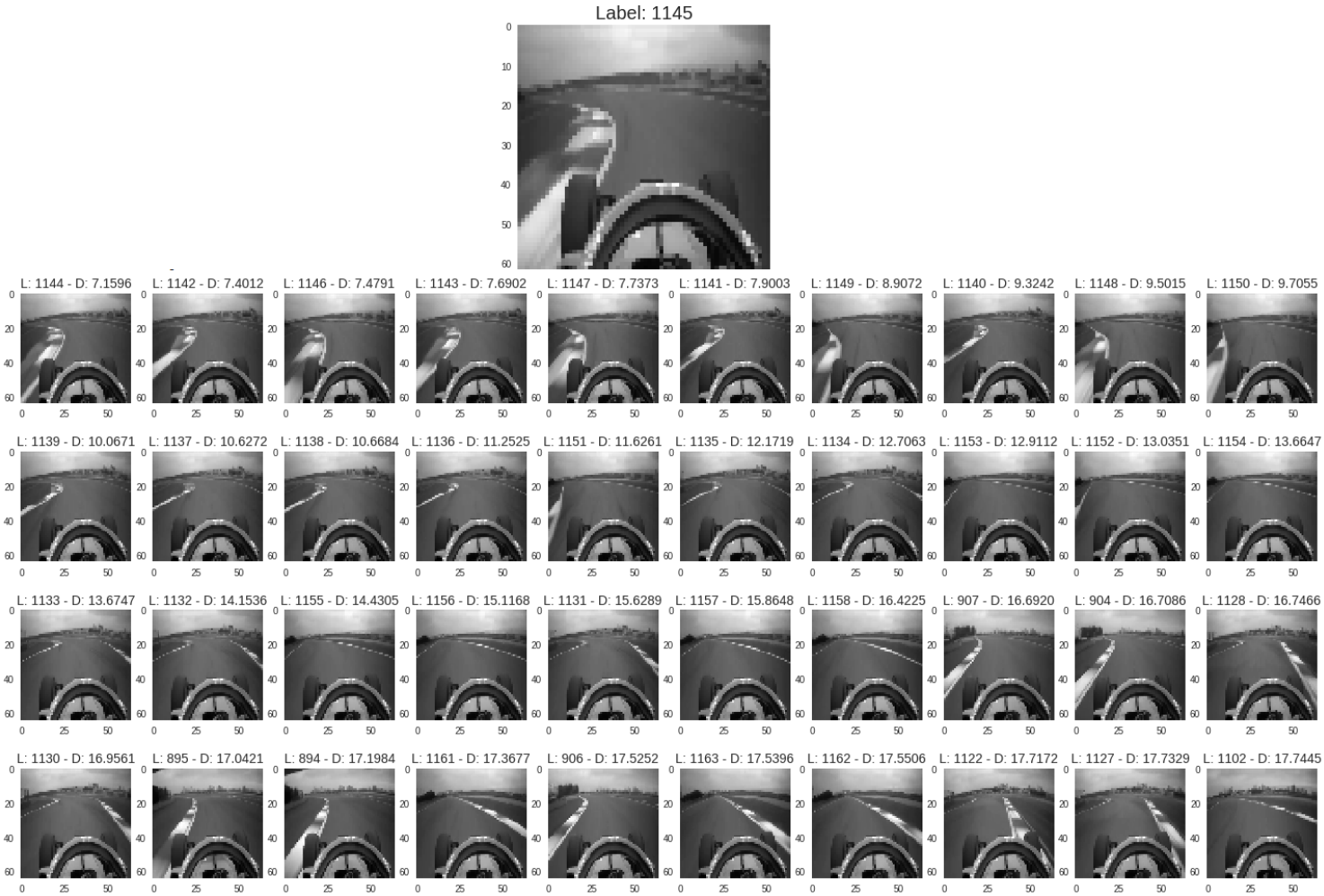

Figure 5.3: KNN qualitative evaluation on the train set. The same structure is used as described in Figure 5.2.

These images show pretty good results for around the top 20 images and from there outliers appear. Note that the labels of the nearest images are usually the ones that Triplet loss is suposed to optimize with respect to the selected image for the KNN method.

On the other hand, the quantitative evaluation implemented, called "simple KNN score", consists in computing the ratio of positive images in the $2\epsilon - 2$ nearest neighbours (the maximum range of positives) for each anchor image and then averaging all of this ratios. For example if a certain anchor image has 15 positives over the $K = 2 \times 15 = 30$ nearest neighbours then the ratio for this image results in $15/30 = 1/2$. Averaging all of the ratios over all the possible anchor images in a set provides a quantitative KNN based accuracy. The training set yields a score of 86.39% accuracy while the test set yields 69.96% accuracy. The "simple KNN score" over the test set in addition to the KNN qualitative tests show that the learned embeddings perform fairly well in the domain of Formula 1 races from a POV perspective in gray.

5.2.2 Second implementation results

A total number of 11.185 images from the colon have been fetched. These images come from an open database named CVC-ColonDB published by the Computer Vision Centre of the Universitat Autònoma de Barcelona (UAB) [10]. A total number of 1.500 images have been deemed as the test set while the rest of the images form the train set (9.685). Again, during the training process, the test Triplet loss stabilizes around the range $[0.4, 0.6]$, repeating the same experience as the one presented in the Figure 5.1. Due to the high number of images composing the training set, for memory reasons the evaluation techniques applied to the training set are restricted to the first 1.500 images (the same size as the test set). The following pages show the KNN qualitative results for the test and training sets.

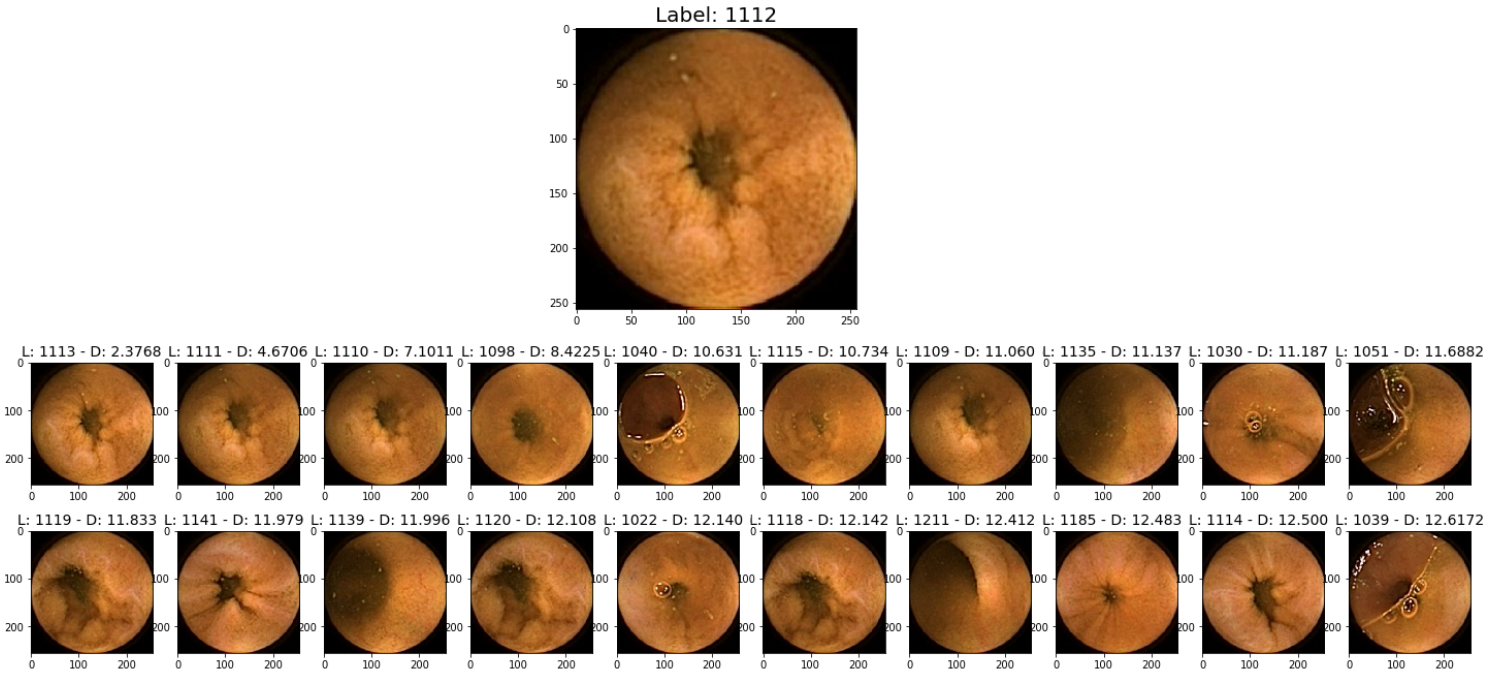

Figure 5.4: KNN qualitative evaluation on the test set. The top image is the selected image for the KNN method. Below it the 20 nearest neighbours sorted from closer to farthest (left to right and top to bottom). The 'L' stands for label and the 'D' stands for the distance between the embeddings.

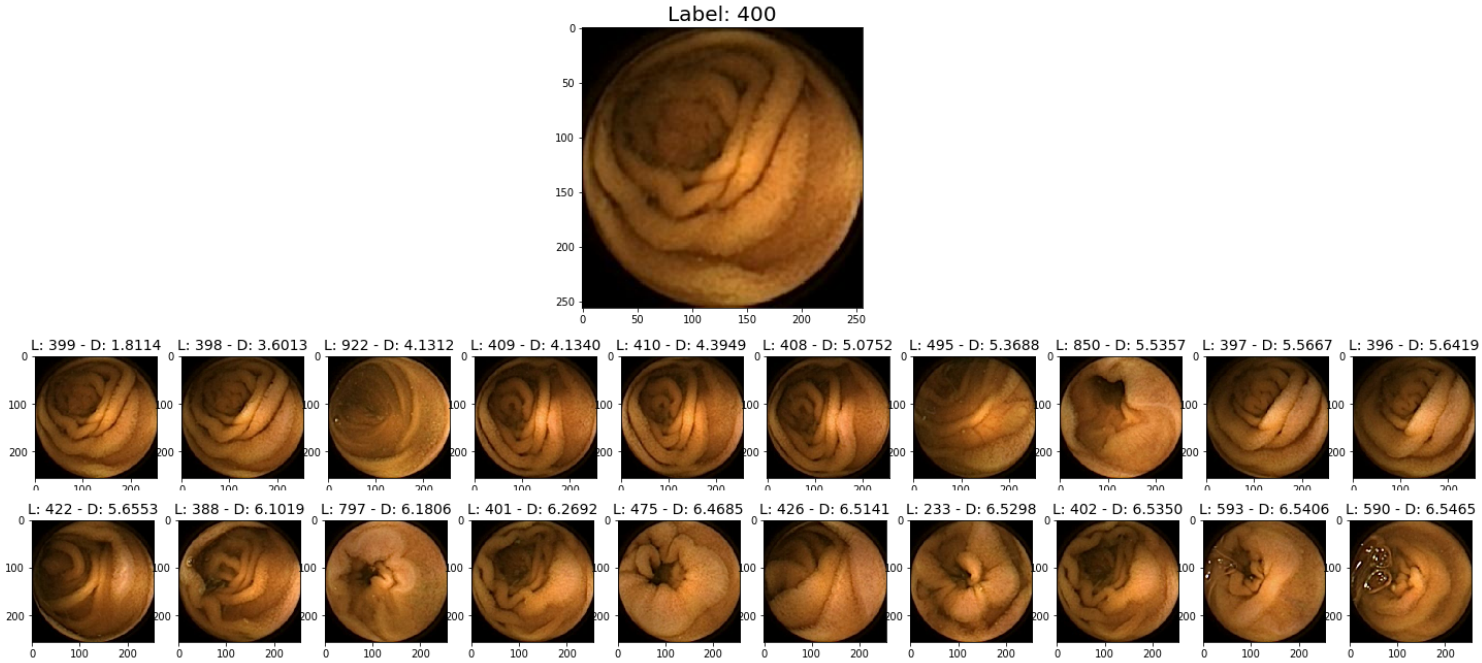

Figure 5.5: KNN qualitative evaluation on the train set. The same structure is used as described in Figure 5.4.

Apart from some outliers in the KNN test evaluation, the qualitative results are closely similar to the selected image or at least share some significant similarities. The quantitative KNN results ("simple KNN score") are the following: the training set yields an accuracy of 22.86% and the test set an accuracy of 13.77%. This incoherence between the qualitative and the quantitative KNN based results comes from the huge number of images in addittion to the labels frame tolerance of $\epsilon = 4$. Having a low ϵ explicitly imposes a stronger restriction when considering the positives of an anchor image. Given an anchor image, the KNN most similar images (by embeddings) may very possibly be outside the labels frame tolerance range and still look similar. Noise, in the sense of repeating images with certain separation and different images next to each other, amplifies this effect. For these reasons, an additional way of evaluating the embeddings from a global perspective must be applied.

A K-Means clustering evaluation is performed in a specifically prepared subset of the test set. The prepared subset is selected in a way in which the ideal clusters can be easily discriminated. K-Means clustering is just a KNN based clustering algorithm which given the number of desired clusters c and the configuration of the cluster's centers, generates c cluster in total around each center with the K nearest neighbours data point (the K value is optimized by the algorithm). The test subset selected is formed by the entire test set since 3 different types of images form the set, therefore, expecting to obtain 3 clear separated clusters if the embeddings are sufficiently representatives. The clusters obtained are the following:

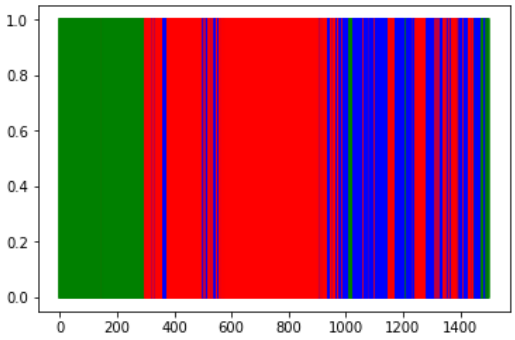

Figure 5.6: The x-axis represents the labels of the images (frame number) while the y-axis has no significant meaning. Each colour symbolizes a cluster. Each point in the x-axis has a coloured column which shows the cluster to which the image belongs.

The 3 expected clusters are distributed by the following labels: the first cluster goes from label 0 to 300, the second one goes from 301 to 900 and the last one goes from 901 up until the last label (1.500). The resulting K-Means clusters exactly identifies the first cluster, identifies the second cluster with some outliers and identifies the third cluster with a great number of noise coming from the other clusters. Due to the third ideal cluster being noisy and hard in terms of recognizing it as a whole, this result shows that the learned embeddings in the domain of coloured colon images perform fairly well.

5.2.3 Final implementation results

For the results of the last implementation a discrimination is made between having trained with Data Augmentation and having trained without it. Both of them use the same hyperparameters (except for the DA one) as well as the same data for training and evaluation. A total number of 7 videos coming from WCE devices are used, providing a total of 128.108 frames. The test set is composed from 2.796 images of the first video while the rest of the images are used for the training process (125.312). Due to the high number of training images, the evaluations performed over the training set are restricted to the first 2.796 (same size as the test set). The same evaluations are applied to both discriminations (with DA and without DA). To conclude, a final comparation analysis is made.

No Data Augmentation (NDA) results

During training, the Triplet loss of the test set stabilizes around the range $[0.2, 0.4]$ as seen in the Figure 5.7. Experiencing the same as during the second implementation, the "simple KNN score" yield low accuracies regardless of how good the embeddings are. For the training set the score is 36.36% and for the test set the score is 18.31%. The following pages show the qualitative results of the KNN evaluation.

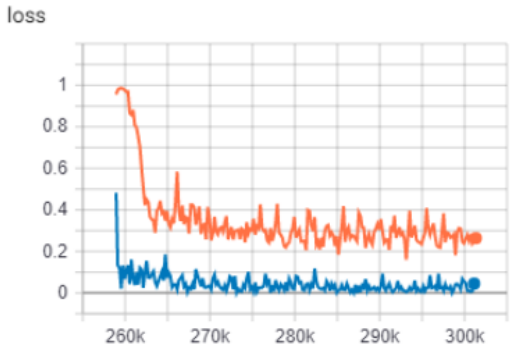

Figure 5.7: No DA Triplet loss for the training set (blue) and for the test set (orange).

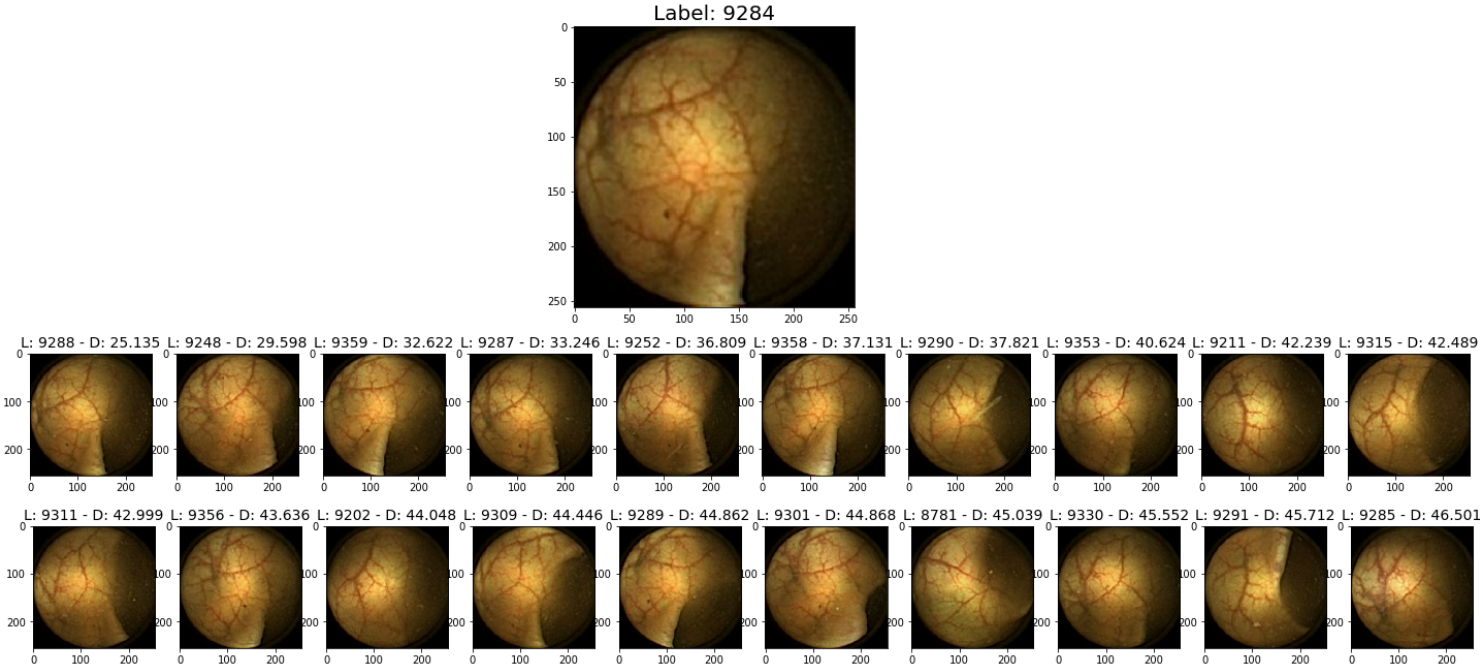

Figure 5.8: KNN qualitative evaluation on the test set. The top image is the selected image for the KNN method. Below it the 20 nearest neighbours sorted from closer to farthest (left to right and top to bottom). The 'L' stands for label and the 'D' stands for the distance between the embeddings.

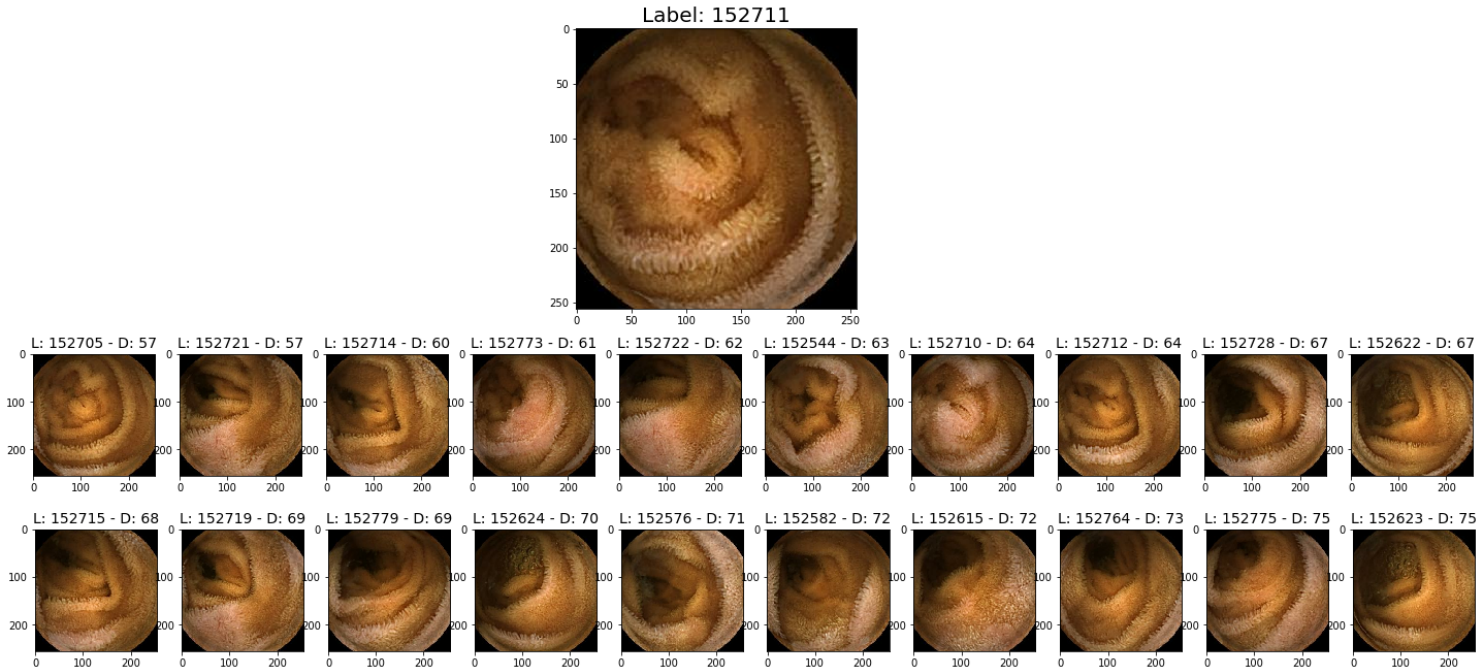

Figure 5.9: KNN qualitative evaluation on the train set. The same structure is used as described in Figure 5.8.

The qualitative analysis shows really good results in both train and test sets. Really few outliers, or none at all, appear when repeating the qualitative evaluation with different images. This indicates a pretty good embedding learning regardless of the artificial labels (seen during the qualitative analysis).

The predefined test subset for the K-Means clustering analysis has been selected in the following way: first cluster from label 0 up to 800, second cluster from label 801 up to 1000 and last cluster from 1001 up to 2796. The second cluster is fairly hard to identify as a whole due to its similarities with the first cluster (such as the colour). On the other hand, the first and last clusters are easily differentiated. A representative image from each cluster is shown in Figure 5.10.

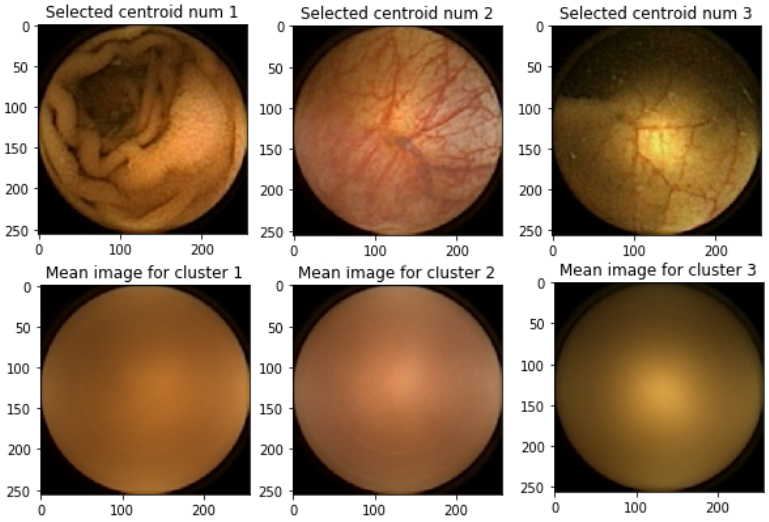

Figure 5.10: The three selected centres for each cluster at the top row. Bottom row shows the mean image of each cluster. Note the difference in colours between the mean images.

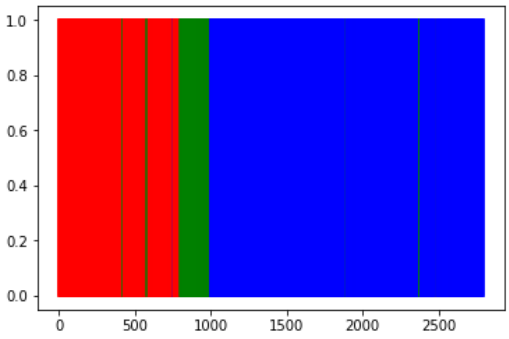

Figure 5.11: Resulting K-Means clusters for the trained model without Data Augmentation.

The well identified K-Means clusters, seen in Figure 5.11, support the hypothesis of good embeddings learned. An additional clustering evaluation is implemented in order to properly visualize if the three clusters correspond to the clusters produced by the embeddings in a reduced 2 dimensional space. An open-source Python library called Uniform Manifold Approximation and Projection (UMAP) is used for the clustering visualization. UMAP projects data from a high dimensional space to a lower dimensional space through topological properties of manifolds [11]. The visualization of the clusters is shown in the following Figure.

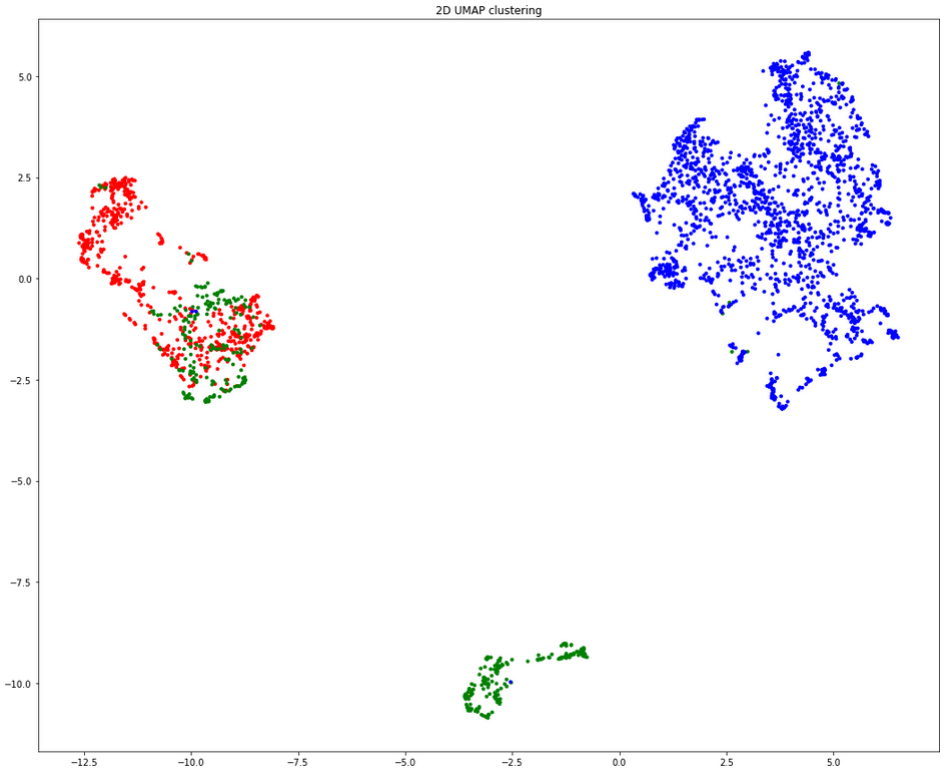

Figure 5.12: 2-dimensional projection of the embeddings. Red point = belongs to the first ideal cluster, Green point = belongs to the second ideal cluster and Blue point = belongs to the third ideal cluster. Distances between points are not representative of the distances between their embeddings.

The visualization of the embeddings forming clusters further confirms the good performance of the embeddings learned by the final model. The last and conclusive evaluation technique applied is the analysis of the results yielded by a WCE image classifier with the Triplet loss model pretrained as the starting point (transfer learning specialized in the domain of WCE data, as proposed for the goal of the project). This analysis is done in the last section of the Results chapter in order to compare it to the model trained with DA.

Data Augmentation (DA) results

As previously stated, the hyperparameters used (except for the DA/NDA) as well as the data used for the training and evaluation processes are exactly the same for comparison purposes. During training, the Triplet loss of the test set stabilizes around the range $[0.2, 0.3]$. The "simple KNN score" of the training set is in this case 28.07% while the score for the test set is 16.19%, slightly lower than in the no DA approach. This decrease is due to the Data Augmentation effects which induce the model to be invariant to rotations and flips over an image, which in turn force the embeddings to be similar to even more images throughout the same data. These properties, invariance to rotations and flips, as well as its effects in the embeddings can be clearly seen in the following Figures which show the KNN qualitative results.

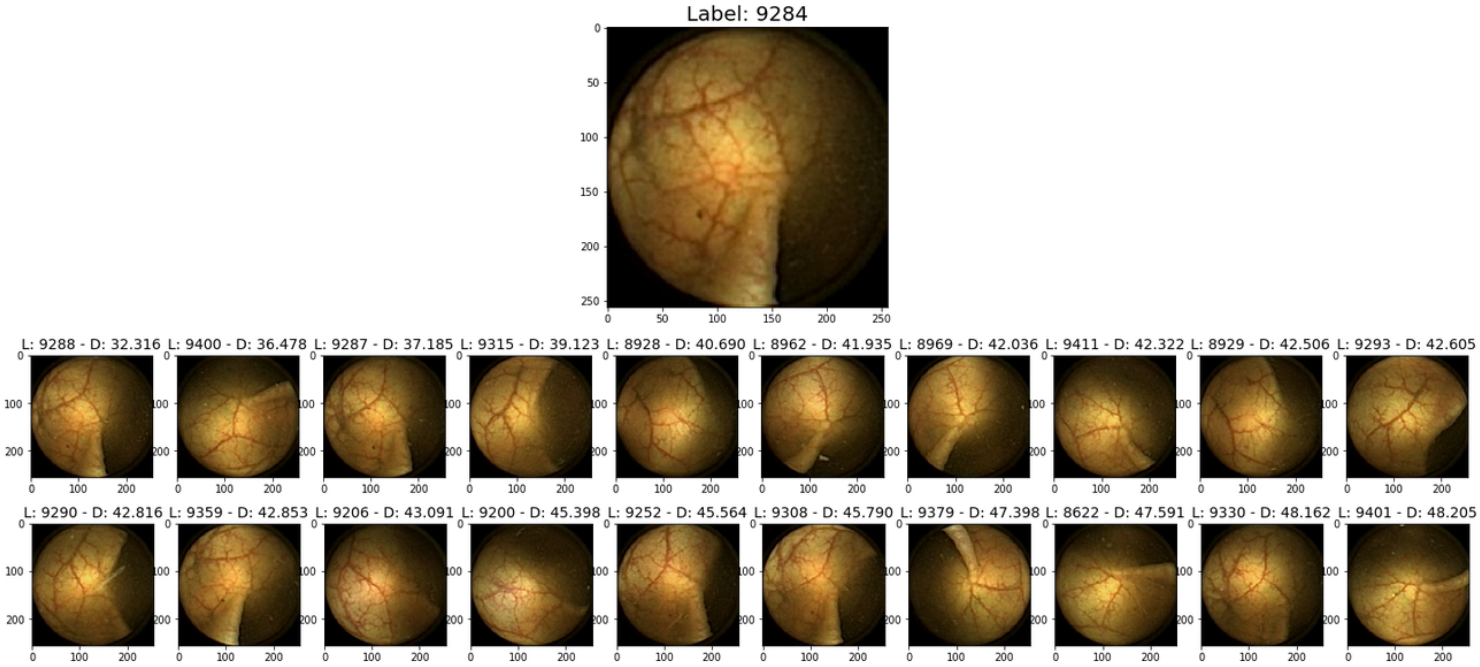

Figure 5.13: KNN qualitative evaluation on the test set. The top image is the selected image for the KNN method. Below it the 20 nearest neighbours sorted from closer to farthest (left to right and top to bottom). The 'L' stands for label and the 'D' stands for the distance between the embeddings. Note the difference of images with respect to the Figure 5.8.

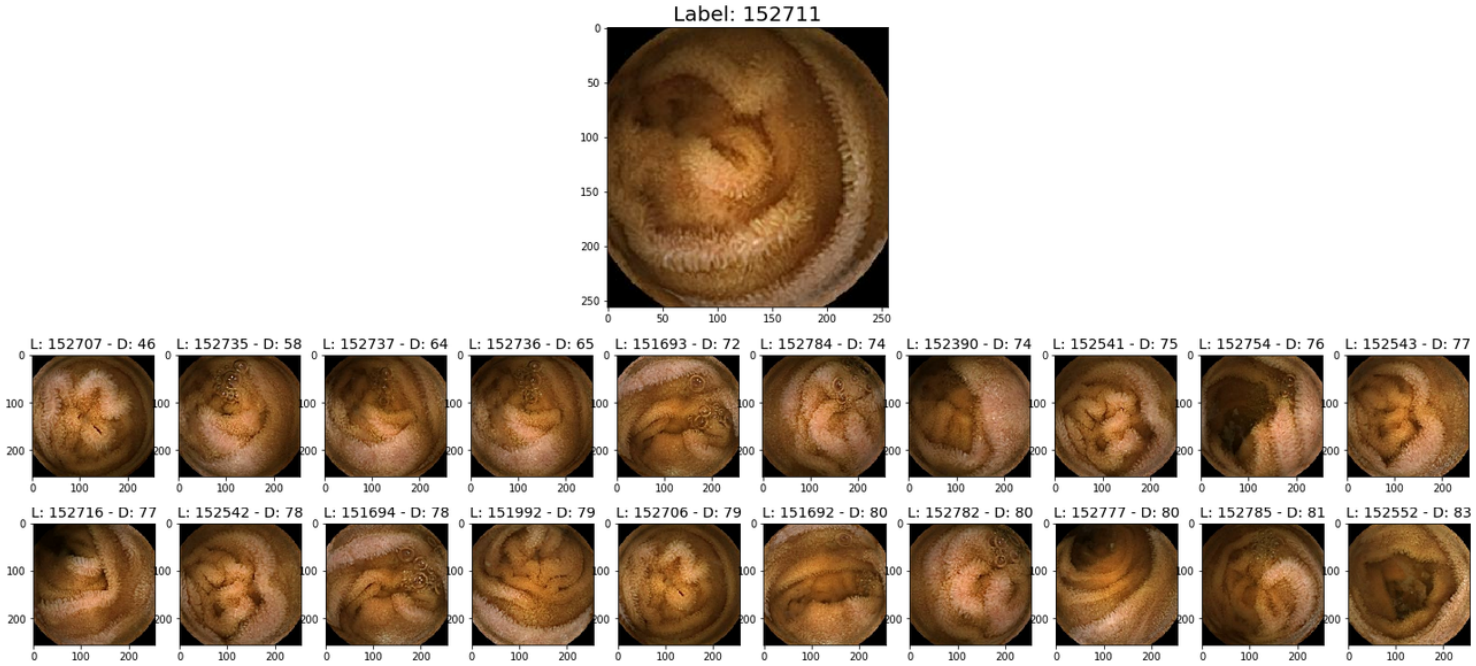

Figure 5.14: KNN qualitative evaluation on the train set. The same structure is used as described in Figure 5.13.

As with the no DA approach, the qualitative analysis shows really good results in both image sets. Really few or none outliers appear when repeating the qualitative evaluation indicating a pretty good embedding learning. The main difference with respect to the no DA approach is that now the closest images can be any image similar to the selected one regardless of its orientation and symmetries.

Using the same predefined test subset for the K-Means clustering analysis, the results are shown below. The representative images selected for each ideal cluster as well as the mean images are the same as in Figure 5.10.

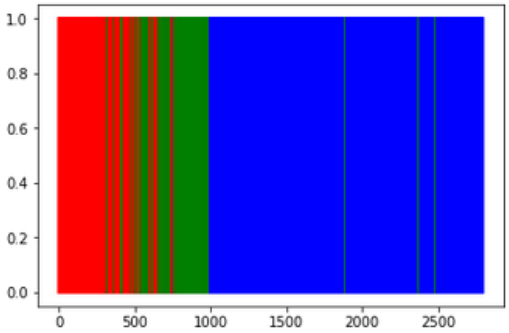

Figure 5.15: Resulting K-Means clusters for the trained model with Data Augmentation.

The three clusters are identified by the K-Means method but the frontier between the two first ones is significantly noisy in the sense of not detecting a clear frontier. This result shows good embeddings learning but, nonetheless, further analysis needs to be done in order to conclude which approach is better for the purpose of the project (with or without DA).

The following page shows the UMAP visualization of the clusters formed by the embeddings of the test images. Using the colours of each ideal cluster it can be seen what is happening in the frontier of the first and second clusters.

Note that the UMAP cluster visualization of the DA approach is strongly similar to the no DA approach. The only significant difference is the higher confusion of the DA approach when dealing with the frontier of the two first ideal clusters. The conclusive evaluation of the performance of this model, compared to other models such as the no DA model, is shown below.

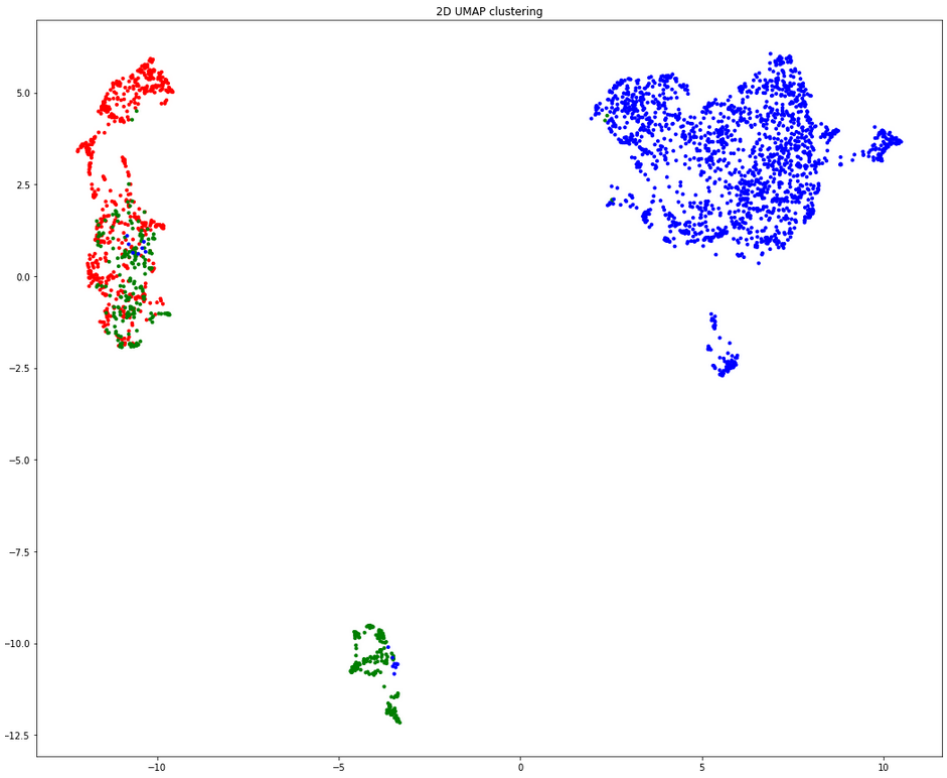

Figure 5.16: 2-dimensional projection of the embeddings. Red point = belongs to the first ideal cluster, Green point = belongs to the second ideal cluster and Blue point = belongs to the third ideal cluster. Distances between points are not representative of the distances between their embeddings.

5.2.4 Conclusive results

The conclusive evaluation consists in two accuracies. The first one is KNN-based which directly evaluates on the embeddings while the second one is a combination of the embeddings with a training process.

- A Rank-1 accuracy is implemented and evaluated which computes the average accuracy over a set of classified images from the WCE domain. Rank-1 computes the top 1 nearest neighbour in the embeddings space for a given image and checks whether the nearest image has the same class as the selected image, finally averaging all the accuracies yields the Rank-1 accuracy over a labeled set of images.
- The second accuracy is directly the accuracy obtained by training a classifier from the embeddings generated by the trained model. The embedding extractor model is not modified during the classifier training, therefore, the embeddings are unaltered and the hypothesis used is that better embeddingss yield better a classification accuracy. Also, through the classification evaluation, the motivation of the project can be tested.

The dataset used for these two techniques is composed by a total of 3.000 images from which some of these have also been used for the training of the embeddings model due to the low amount of labeled data in the WCE domain. Half of the labeled dataset is used for training and the other half for evaluation (1.500 for each purpose). During the training of the classifier, data augmentation is applied due to the low amount of labeled data in the training set, otherwise overfitting could easily appear. The following table shows the results for different embedding extractor models.

	Train accuracy	Test accuracy	Rank-1 accuracy
Random init	68.55%	41.12%	65.86%
ImageNet init	25.64%	35.47%	63.17%
Triplet loss NDA	74.14%	67.68%	77.52%
Triplet loss DA	**80.20%**	**70%**	**78.15%**

Figure 5.17: The first column describes the embedding extractor model. The second and third columns refer to the classification accuracies and the last column is the average of the Rank-1 accuracy over the training and test sets.

The first embedding extractor model consists in taking the architecture defined for the WCE domain adaptation task, randomly initializing its layer parameters and not training it. The second one consists in the same process with the difference of initializing the layer parameters from the same architecture trained with ImageNet database for a general purpose classification task. The two last embedding extractor models are the ones implemented, trained and evaluated in the previous sections.
Analyzing the results of the table the following insights can be made:

- The ImageNet initialization model underperforms in all of the accuracies in comparison with the randomly initialized model. This exactly reflects the motivation of the project since it is showing worse results when transfer learning from ImageNet in the WCE domain in comparison with no transfer learning at all. Further clustering analysis of these two models demonstrate the reason why ImageNet underperforms in Figure 5.18. The reason why the ImageNet pretrained model fails to train (detected by the low train accuracy) is because of the extremely poor representation of the images by embeddings. Since the ImageNet pretraining is designed for general purpose classification, the resulting embeddings of the WCE domain are really similar despite their visual differences and totally fail to form clusters (Figure 5.18).

- The Rank-1 accuracy is always higher than the classifier test accuracy due to the already used images for the Triplet loss models in addittion to the highly noisy classification of the Rank-1 approach.

- From these conclusive results, the DA training approach achieves slightly better accuracies in comparison to the NDA approach. Therefore, the **ResNet50 CNN trained by the Triplet loss with Data Augmentation and with an embedding size of 2048 would be the one chosen for transfer learning in the WCE domain**.

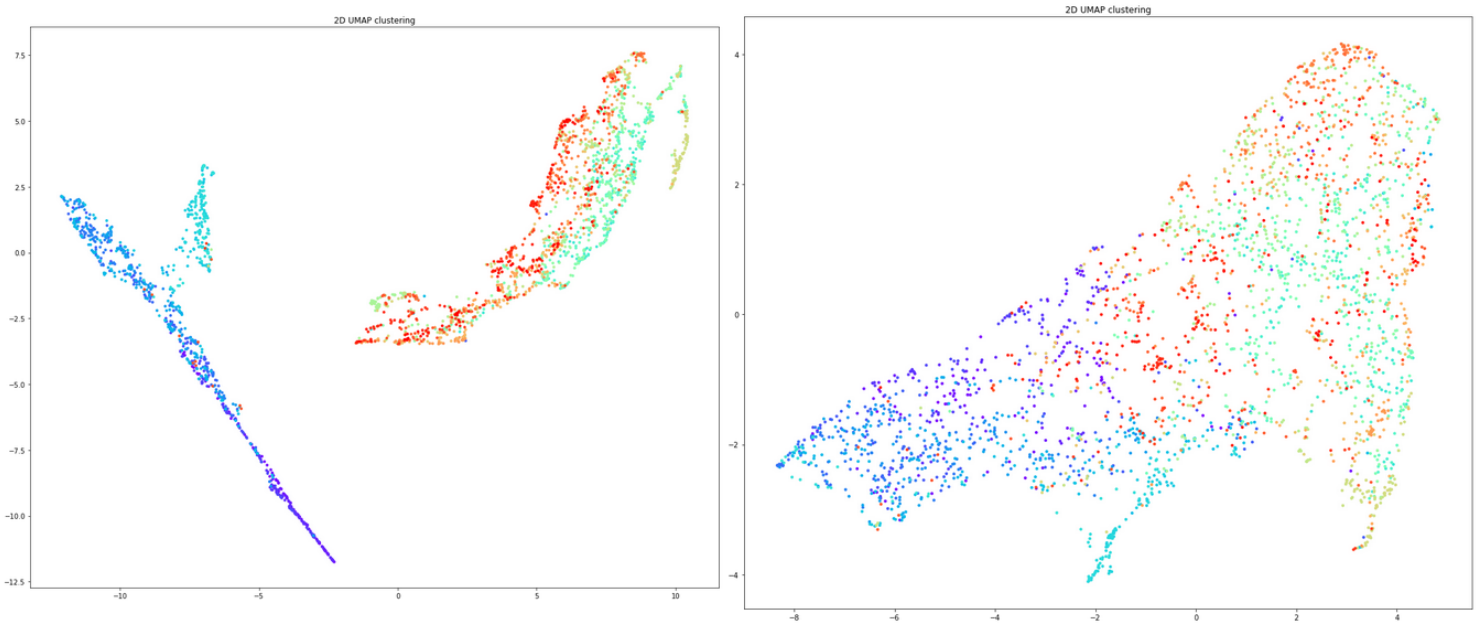

Figure 5.18: The left UMAP embedding projection corresponds to the randomly initialized embedding extractor while the right one corresponds to the ImageNet initialized embedding extractor. Note that the ImageNet initialization totally fails to cluster the images even though clear differences can be noted among them. This lack of sensitivity is due to the ImageNet image database being specifically designed for a broad range of image classifications. The randomly initialized model captures some differences among the images and thus clusters appear from the embeddings.

Chapter 6

Conclusions and future work

During the last evaluations performed and analyzed, the motivation of the project has been properly demonstrated. The goal of the project, to **make Transfer Learning possible in the domain of WCE data from a data adaptation perspective through the implementation of a CNN**, has been satisfactorily achieved through the use of the ResNet50 CNN architecture, a data preprocessing implementation, an adaptation of the Triplet loss and a total of 7 videos from WCE devices. Even though the worked implemented and tested throughout this project has resulted in good results, some issues and strengths have also been identified:

- The labeling data process implemented in addittion to the hyperparameter "label frames tolerance" (ϵ) produces false triplets during the computation of the adapted Triplet loss. A false triplet could be defined as a triplet (A, P, N) where either P is not a positive or N is not a negative even though their labels identify them as such. This can be seen in a simple example: suppose a given ϵ value chosen according to the framerate of the source of the data and supose that the data source does not produce perfectly continuous frames (noise and jumps from frame to frame), then following the labeling process there will be false triplets among all the valid triplets. If this situation were to happen regularly then the Triplet loss could be training in an undesired way. Fortunately enough, the results obtained for the three implementations done during the project show no sign of bad training. Either because the false triplets are so few that they do not affect in such a great scale or because the data source produces sufficiently continuous frames.
- An additional property detected when comparing the ResNet50 model trained with or without DA is that when training without DA then the learned embeddings are forced to look similar to embeddings of similar images including orientation and symmetries. In other words, when training the NDA model, the resulting embeddings are way more *strict* in terms of rotational and symmetric similarities.

Various lines of work are proposed from the results and implementations of this project:

- Finding the best configuration of hyperparameters which yields the best embedding representations after training. The most notable hyperparameters are: the learning rate, the batch size, the use of squared or non squared distances in $\mathbb{R}^n$, the use or non use of Data Augmentation, the "label frames tolerance" ϵ, the block shuffling size b, the margin of the Triplet loss α and the use of Batch all or Batch hard approaches of the Triplet loss.

- It is hypothesised that increasing the block shuffling size b hyperparameter (up to a maximum of $2\epsilon - 2$ which is the maximum number of positives per anchor) induces the aparition of harder triplets due to adding more positives to the batch which are more far to the anchor. If b is increased past $2\epsilon - 2$ then closer negatives will be added to each anchor which directly means harder triplets. If this hypothesis holds true with a given dataset then increasing b will make the training process harder and will force the CNN to learn more detailed and specific embeddings.

- Refering to the aparition of false triplets, an approach can be implemented and tested with the aim of minimizing these undesired triplets. After implementing, training and evaluating the ResNet50 implementation proposed, a KNN clustering process (or another process) could be used to *relabel* all the images from both the training and the test set being cautious about the outlier and noisy images. Once the images have been relabeled, the training and evaluation processes can be repeated with the new labels starting from an untrained (or from the previously trained) ResNet50 model. Iterating this process could lead to better labels which in turn could lead to better embedding learning. Even more, if this process yields good results then all the images from the dataset will end up being classified by c different classes which, a priori, were unknown. This last point would be the definitive solution to the critical issue presented and approached in the Section 3.3 Data preprocessing. In few words, if this iterative process converges to good results then a methodology for finding an undetermined number of classes from a big unlabeled dataset could be formally defined. The application of such methodology could solve the big problem approached through unsupervised learning, which is dealing with huge amounts of unlabeled/non-classified data, in a wide variety of tasks.

Bibliography

[1] Global cancer statistics 2018: GLOBCAN
Freddie Bray, Jacques Ferlay, Isabelle Soerjomataram, Rebecca L. Siegel, Lindsey A. Torre, Ahmedin Jemal
American Cancer Society, 12 September 2018
`https://onlinelibrary.wiley.com/doi/full/10.3322/caac.21492`

[2] Wireless capsule endoscopy
Gavriel Iddan, Gavriel Meron, Arkady Glukhovsky & Paul Swain
Nature, 25 May 2000
`https://www.nature.com/articles/35013140`

[3] ImageNet statistics
Stanford Vision Lab, Stanford University & Princeton University
2016
`http://image-net.org/about-stats`

[4] Deep Learning on Small Datasets without Pre-Training using Cosine Loss
Bjorn Barz & Joachim Denzler
25 Jan 2019
`https://arxiv.org/pdf/1901.09054.pdf`

[5] Fundamenals of Artificial Neural Networks
Mohamad H. Hassoun
MIT Press, 1995

[6] FaceNet: A Unified Embedding for Face Recognition and Clustering
Florian Schroff, Dmitry Kalenichenko & James Philbin
Cornell University, 12 Mar 2015
`https://arxiv.org/abs/1503.03832`

[7] Deep Residual Learning for Image Recognition
Kaiming He, Xiangyu Zhang, Shaoqing Ren & Jian Sun
Cornell University, 10 Dec 2015
`https://arxiv.org/abs/1512.03385`

[8] ResNet in TensorFlow
Google Brain Team
`https://github.com/tensorflow/models/tree/master/official/resnet`

[9] SenseTheFlow
GitHub user gpascualg
`https://github.com/gpascualg/SenseTheFlow/`

[10] CVC-Colon DataBase
Computer Vision Centre. Edificio O 08193 Bellaterra, Barcelona (Spain)
`http://mv.cvc.uab.es/projects/colon-qa/cvccolondb`

[11] Uniform Manifold Approximation and Projection (UMAP)
GitHub user lmcinnes
`https://github.com/lmcinnes/umap`

Annexed

.1

Source code

Source code and data at the GitHub repository:

`https://github.com/maxby12/TFG-UB`