Universidad de Sevilla
Escuela Técnica Superior de Informática

Máster en Lógica, Computación e Inteligencia Artificial

Trabajo de Fin de Máster

Prediction of head and neck cancer with Deep Active Learning

Author:

Enrique Arroyo Escribano

Directors:

Miguel Ángel Gutiérrez Naranjo

Miguel Cárdenas Montes

Seville, November 27, 2022

Contents

List of Figures

List of Tables

Acknowledgements

I wish to show my appreciation to the following people for helping me finalize the project.

To Miguel Ángel and Miguel, the directors of this master's work, for guiding and advising me. The enthusiasm they have shown has motivated me to carry out this project with great interest, and they have made me feel like one more member of a team.

To my parents, for supporting me and helping me to get to where I am now.

To my brother and my friends, for always being by my side and showing me how much they appreciate me.

Thanks to all of you.

Abstract

Collecting medical data is a costly task because the volume of data is often very large. This is even more complex considering that professionals need a lot of time to give a diagnosis for all these data. Commonly used Deep Learning techniques are not always enough to address this problem, since the number of attributes for each sample can be too large. Therefore, there is a need to find a way to analyze this information using as few instances as possible.

This work focuses on the use of Deep Active Learning to automatically label new samples. It uses the good classification capacity of Deep Learning models together with the sample study provided by Active Learning. Thus, the cost of labeling is reduced while seeking to maintain performance. Deep Active Learning splits the dataset into a labeled and an unlabeled set. The labeled set is used to initialize the Deep Learning model, which is used to extract features from the unlabeled pool.

To test the effectiveness of Deep Active Learning, we used a dataset with Head and Neck cancer samples. Our results show that it is possible to work with high dimensional data thanks to this method. Using 6 instances to classify per iteration, we were able to obtain an accuracy of 83% to 100%.

Chapter 1

Introduction

Artificial Intelligence (AI) is a discipline that seeks to automate tasks and make work easier for humans. Today, its applications are increasingly integrated in both leisure (simulations [4], streaming services [30], video games [29]...) and work (video surveillance [3], product inspection [6], company management [31]...). It began as a consequence of research into the automation of mathematics, initially applied to concepts that may seem more trivial, such as in strategy games. But this kind of work laid the foundations for the standard methodology used today: the adequate representation of the problem and the use of mechanisms for solving it. Over time, good results are being achieved in investigation, thanks to the development of new hardware, the creation of ontologies that reorganize larger and larger knowledge domains and the use of increasingly efficient algorithms.

Nowadays, AI use is widespread throughout the entirety of the tech world and bleeding-edge technologies. The evolution of this discipline in the last decades has led to numerous advances, being able to work with increasingly complex data such as images, audios, videos, texts... This gives rise to new areas whose projects are integrated into our daily lives, such as Natural Language Processing to study how human beings communicate, used for example in translators [1]; many of these areas are also combined to accomplish other tasks, such as autonomous vehicle driving [21].

AI is also an essential part of modern healthcare, as it is being used to support re-

searchers and medical professionals in hospitals and other clinical settings. It has a lot of advantages, speeding up the analysis and helping doctors to make better decisions. It can be used in a lot of ways, for example, clinical decision support helps professionals with medications, mental health advice and other kinds of recommendations depending on the patient's condition. Another common application for AI in medicine is computer vision, with tools that let us evaluate medical images (X-rays [13], magnetic resonance images [33], computed tomography scans [22]...) to find lesions and other discoveries that a specialist might miss.

A very important utilization of AI in this medical context is the study of cancer. It can help in its discovery, the diagnosis and in the therapy required. There are many different types of cancer, but in this work we will focus on **Head and Neck** cancer (see Figure 1.1 for details of the body area), which ranks sixth in incidence in Spain with more than 12000 cases diagnosed per year [5]. This kind of cancer can start in several places in the head or throat (not including the eyes and the brain):

- The larynx.
- The pharynx.
- The lips.
- Inside the nose, behind the nose and in the sinuses.
- In the oral cavity.
- In the salivary glands.

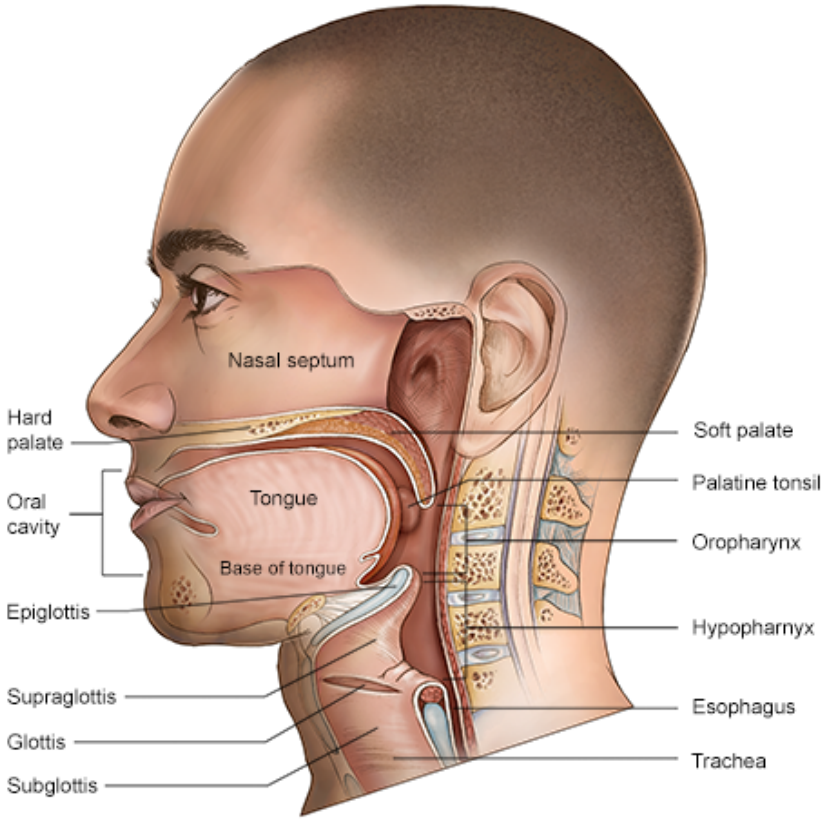

Figure 1.1: Main body parts affected by Head and neck cancer. Source: https://www.cancer.net/

This work aims at automatically labeling new instances. In this case, the samples used are from Head and Neck cancer cases, to demonstrate that it is applicable to real medical data. To achieve this, we will use **Deep Active Learning**, an approach that involves selecting unlabeled data items to best improve an existing classifier. Deep Active Learning is an ongoing active research sub-domain within *Deep Learning* space that is developed to help models make more accurate decisions.

Chapter 2

Problem definition

In this chapter we will explain the problem we want to solve, dividing it in two sections. The real problem is the establishment of what we want to achieve, while the technical problem are all of those solutions proposed to solve the real problem. The following chapter will feature the techniques used to solve it.

2.1 Real problem

The use of AI can be applied to a lot of areas in medicine, but the objective here is to classify patients with head and neck cancer. We would like to have a way to provide input data to a software, knowing that these data contains information about a patient, and get an output that tells us if the person has head and neck cancer or not. The idea is to automate this labelling process needing as few samples as possible, to assist doctors with the diagnosis.

We are dealing with a classification problem, which consists on assigning a label to an unseen instance, according to the analysis of the features of the instances and their classification in a known dataset. There will be two possible classes, a sample can be labeled as having head and neck cancer (**positive class**) or not (**negative class**). Machine Learning can be supervised or unsupervised. Supervised learning is defined by the use of labeled datasets to train algorithms into classifying data, to be able to calculate how

accurate our model is in making class predictions. In unsupervised learning, samples are grouped according to their features, finding patterns in unclassified data. This project uses methods for both types: for example, Random Forests are trained to calculate their performance in classifying instances (supervised learning), while DBSCAN clusters samples based on their similarities or differences (unsupervised learning).

2.2 Technical problem

2.2.1 Obtaining data

To make this study, we will need previous information about patients, whether they are diagnosed with head and neck cancer or not, so that we can explore the differences between a healthy and a diseased person. This is a very expensive task, because there are many characteristics that we can extract from a person depending on the context. The data is taken from the *Cancer Genome Atlas Program* [16], a repository containing more than 20000 cancer samples spanning 33 cancer types.

Our dataset has instances from head and neck cases, and it is divided in two classes, depending on whether the sample is diagnosed with cancer (**positive** label) or not (**negative** label). There are 564 instances in this dataset, and each one is defined by 60498 attributes (the dimensionality is 564×60498). Each attribute is a **quantitative representation of the expression of each gene**, unless a gene is inactive, in which case its value will be 0. The dataset has 519 samples from the positive class and 45 samples from the negative class, and this difference in proportions means that **we are dealing with an imbalanced dataset**. The dataset is available here: https://github.com/enaes05/DeepAL_mod.

2.2.2 Managing data

Once we have the dataset about the patients, we encounter two problems: the number of instances is small, and each of them is represented with a lot of attributes. This last issue means the data has high dimensionality, and the combination with a low sample size

is uncommon and hard to deal with. Reducing the dimensionality is one of the major challenges of this study, as it is necessary to work with the dataset with more ease.

2.2.3 Labelling process

The human task of labeling medical data becomes difficult because the process is slow and expensive, as several professionals are needed to make the diagnosis. This is why there is a need for studying an automatic way to label never-before-seen samples from previously labeled data. For this purpose we will use **Deep Active Learning** (see Chapter 3), a technique used to achieve great labelling efficiency while dealing with high-dimensional data. Deep Active Learning first extracts data features from already labeled samples, then selects unlabeled instances and tries to classify them with the information gathered from the first step. This strategy will be applied to the dataset by selecting random set of instances, and checking their predicted labels to see what percentage match their actual label.

Chapter 3

Deep Active Learning

Deep Active Learning comes from the combination of the strong learning capability in high-dimensional data that Deep Learning provides, and the potential of Active Learning to reduce labelling costs. This chapter will explain in depth the need to use Deep Active Learning and how we are going to apply it to our problem.

3.1 Deep Learning

Deep Learning [19] is a subfield of Machine Learning that uses algorithms inspired by the structure of the human brain. These algorithms are known as **Artificial Neural Networks**. To better exemplify what a Neural Network is, we can observe the behaviour of one of its possible architectures, the *perceptron*. As it can be seen in Figure 3.1, the inputs of the perceptron (the attributes of an instance) are multiplied with their weights, then these multiplied values are added. The bias values (w_0 in the previous image) is used to shift the final value calculated with the activation function.

In the above example, the perceptron only has one layer. One approach to the definition of Deep Learning is the use of perceptrons with many layers. There is no exact number of layers from which we refer to Deep Learning: it is the complexity of the treatment of these layers that differentiates it from other traditional multilayer neural networks.

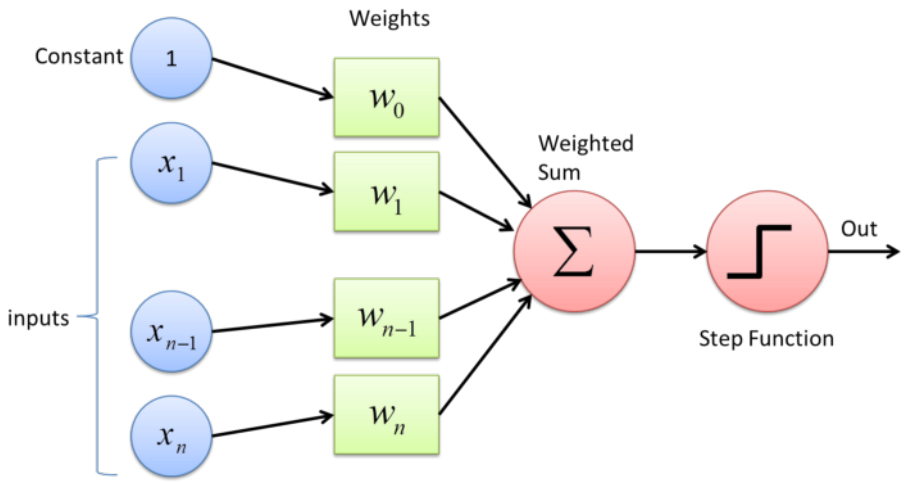

Figure 3.1: Basic structure of a perceptron. Source: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

Using more layers and neurons may increase the accuracy of a Neural Network, but that means that a model will need more computational resources. Due to the structure of the networks, many input parameters can be used. This allows the use of more complicated data, like images. Convolutional Neural Networks (see Figure 3.2) are used for this type of input data, applying operations to the matrix representation of the image, resulting in a multitude of filters (edge detection, blurring, darken, lighten...).

As neural networks are trained with more and more samples, the models may begin to overfit, losing generalization capability. *Regularization* is used to deal with this problem. The capacity of the model is controlled by adding a penalty function in addition to the error function:

$$\tilde{J}(w, X, y) = J(w, X, y) + \alpha\Omega(w),$$

where w represents the weights, X the instances and y the labels. α is a hyperparameter that controls the contribution of regularization, and the optimization algorithm must minimize both functions J and Ω. The objective is to try to keep the weights close to zero.

Through the years, Deep Learning has demonstrated great results in feature extraction,

allowing advances in fields such as computer vision [9] or natural language processing [23]. All of these solid learning capabilities come at the cost of a large number of manually labeled datasets. Obtaining all of these data is not hard, but it has to be manually labeled, which is expensive and takes too much time for professionals.

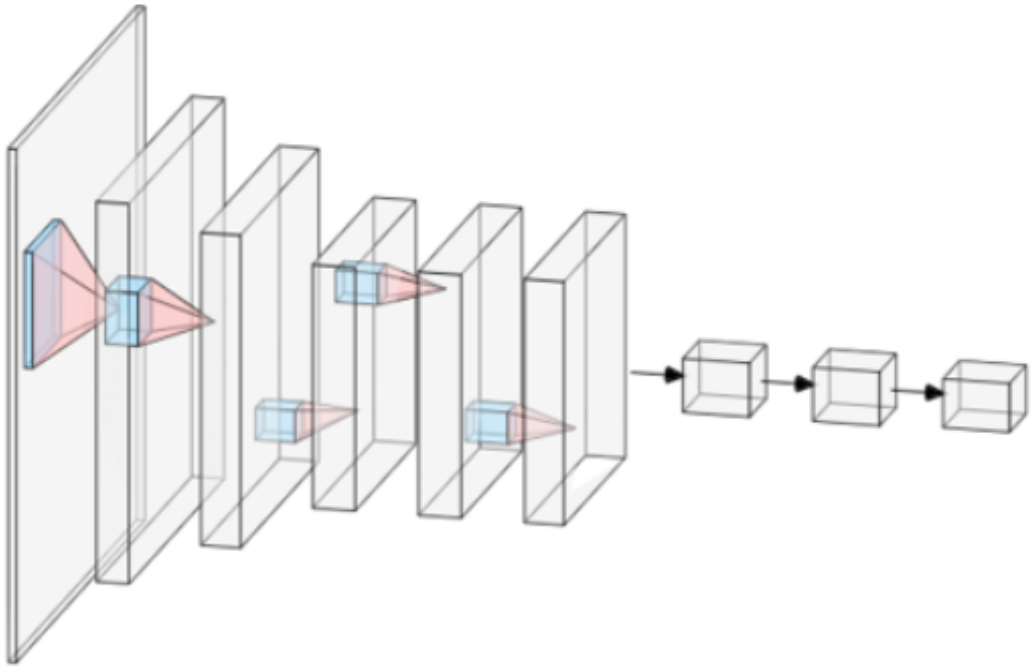

Figure 3.2: A Convolutional Neural Network. When used in images, it applies transformations to them to extract its features.

3.2 Active Learning

Active Learning [27] focuses more on the study of datasets, studying how to improve performance by labeling as few samples as possible. It selects the most useful unlabeled samples and hands it over to the oracle (for example, a human annotator) for labeling. This is done to reduce the cost of labelling while still keeping the performance. The selection of these instances is an important part of Active Learning, called query strategy, and depending

on that strategy it will select different kinds of samples. In Figure 3.3 we can see how Active Learning works. It selects unlabeled samples using a query strategy, gives them to the oracle to perform labelling, adds these instances to the labeled set and trains the model on this labeled set.

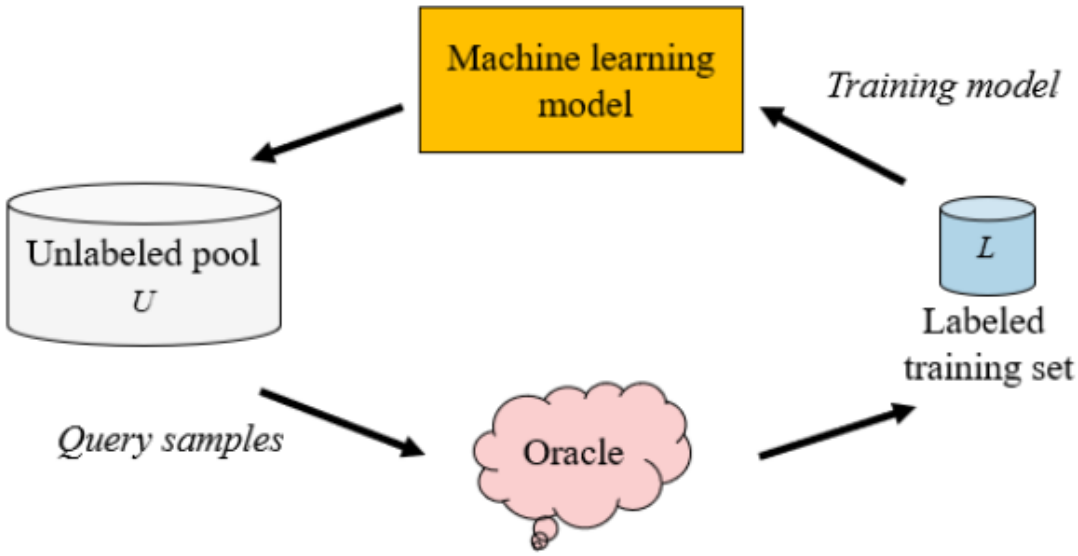

Figure 3.3: Framework diagram of the pool-based active learning cycle. It selects unlabeled samples, labels them with the oracles and includes the instances in a labeled set to train the model.

To evaluate the informativeness of unlabeled instances, there have been many proposed ways of formulating the query strategies: uncertainty sampling [20] (the active learner queries the instances about which it is least certain how to label), query-by-committee [28] (maintaining models that vote on labels of query candidates, with the most informative query considered to be the instance about which they most disagree), query strategies that attempt to minimize generalization error directly [25]...

3.3 Deep Active Learning

Deep Learning has excellent performance while dealing with lots of attributes, but the datasets need to be very large and that means that the labelling of instances is too expensive. Active Learning is done with a small annotation cost because it selects as few samples as possible, but it does not work well with high-dimensional data. It is an obvious approach to combine both techniques, and this combination is known as **Deep Active Learning** [8].

Figure 3.4 shows an example of application of Deep Active Learning. The Deep Learning model is initialized with the labeled training set L_0, and then the model is used to extract features from the unlabeled pool U. The samples from the unlabeled pool U are selected using a query strategy, and then the oracle is used to label these instances, being added to labeled set. The Deep Learning model is trained once again with the new labeled set, and the process is repeated until certain conditions are reached.

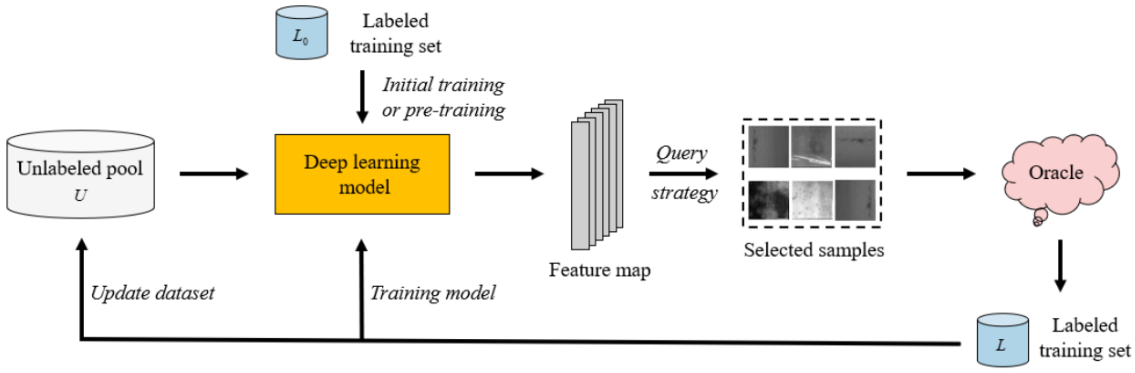

Figure 3.4: Example of Deep Active Learning. The Deep Learning model is initialized with the labeled training set, and it is used to extract features from the unlabeled pool. Samples are selected from the unlabeled pool using a query strategy, then they are labeled using an oracle and added to the labeled training set. The model is trained again with the new labeled set.

In Deep Active Learning, $U^n = \{X, Y\}$ is defined as an unlabeled set with n samples, while $L^m = \{X, Y\}$ is the labeled training set with m samples. X is the sample space, and

Y the label space. The main goal is to design a query strategy Q, $U^n \xrightarrow{Q} L^m$, using the Deep Learning model f. The optimization problem of Deep Active Learning can be expressed as:

$$\underset{L^m \subseteq U^n, (x,y) \in L^m, (x,y) \in U^n}{\arg\min} \mathbb{E}_{(x,y)}[\mathcal{L}(f(x), y)],$$

where $\mathcal{L}(\cdot) \in R^+$ is the loss equation, and it is expected that the number labeled samples is much smaller than the unlabeled ($m \ll n$). This means that the goal is to require as few labeled samples as possible. This is why the query strategy Q is so relevant here, as it is crucial to reduce the labelling cost.

A problem that often arises in Deep Active Learning is that the labeled training samples provided by the classic Active Learning method is insufficient for training in Deep Learning, because the model is greedy for data. Some previous Deep Active Learning methods [36] only trained on the labeled pool sampled by the query strategy, meaning that the training strategies are not fully utilized, ignoring the existence of unlabeled datasets. There are methods like CEAL (Cost-Effective Active Learning) [32], which assigns pseudo-labels to instance with high confidence in model prediction in addition to the labeled set sampled by the query strategy (see Figure 3.5).

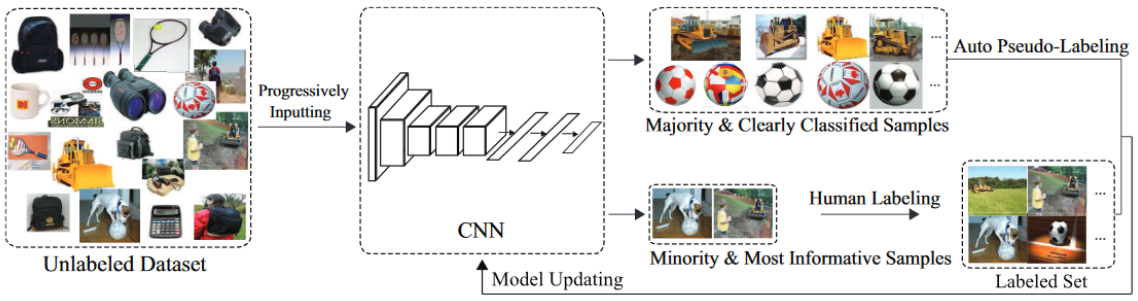

Figure 3.5: CEAL framework. The unlabeled dataset is fed to the Convolutional Neural Network, which gives two types of classification: samples with high confidence (which are given pseudo-labels) and samples with high uncertainty (labeled through the oracle).

Although it is a recent development, Deep Active Learning is already being applied to different fields. For example, as Deep Learning is widely used in computer vision, it is

expected that Deep Active Learning works as well. The previously mentioned technique CEAL is applied in image classification and recognition, because there is the problem of having to deal with high dimensionality while trying to keep the cost of labeling small. This area also expands to medical image analysis, like in [10] for tissue classification in oral cavity cancer. Other applications are object detection and semantic segmentation [26], video processing [15], text classification [35], wearable device data analysis [12]...

Chapter 4

Techniques applied

In addition to Deep Active Learning, other methodologies have been used in this work to develop the research. This chapter will be used to explain how they work and how they apply to our project.

4.1 Random Forest

Random Forest [7] is a method used for classification and regression, composed of a multitude of **decision trees**. A decision tree (see Figure 4.1) is an algorithm that creates a model able to classify an input by looking at the values of its features. The decision tree is constructed with a training set, recursively splitting our training samples using the features from the data that work best for the specific task.

By combining different decision trees we can build a **Random Forest** (see Figure 4.2), and the class with the most votes becomes our model's prediction. This is an ensemble method, because it combines multiple models (the decision trees, which individually are considered weak learners) to form a strong learner that obtains better performance than a single one. A Random Forest can be used to **get the most important features of the dataset**, which are the most important attributes when classifying an instance. This can be done with the **Gini Index**, which is determined by deducting the sum of squared of

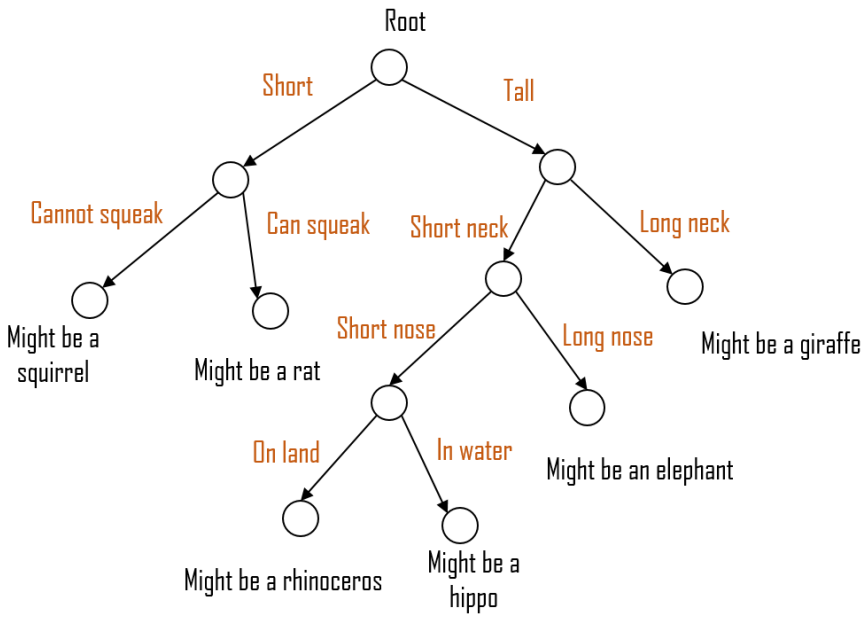

Figure 4.1: Example of a simple decision tree, classifying an animal depending on its features.

probabilities of each class from one:

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2$$

The Gini Index is basically a measure of variance, so lower values represent less missclassification. Selecting only the most important features allow us to reduce the dimensionality of the dataset.

4.2 Variational Autoencoder

Variational Autoencoders [18] (VAE) are another tool used for dimensionality reduction. A Variational Autoencoder is composed of an encoder and a decoder. The **encoder** produces the new features representation by compressing the data using a neural network,

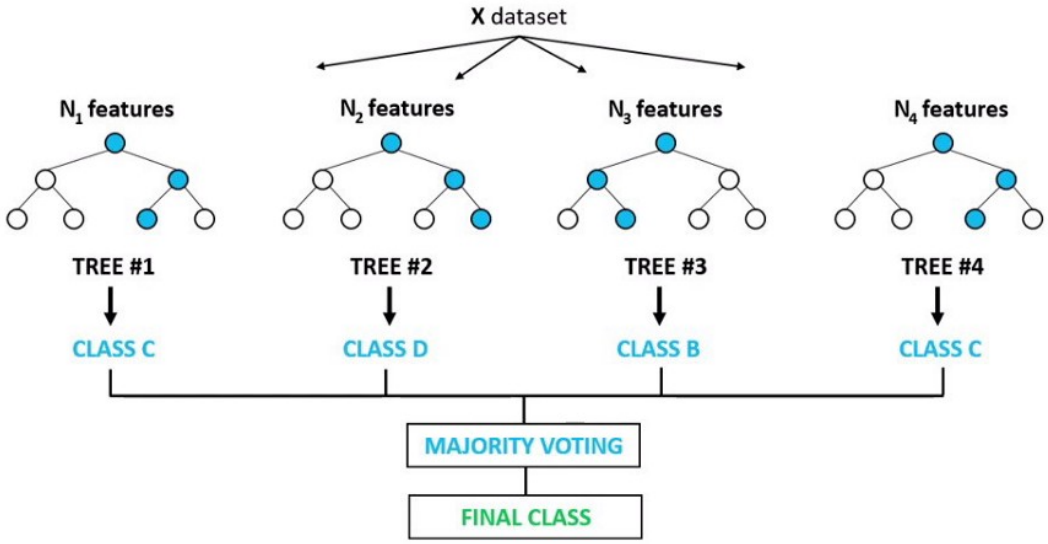

Figure 4.2: Random Forest behaviour. It trains different decision trees, and each one will give a class prediction. The final label of the Random Forest will be class with the most votes.

this new dimensionality is called the latent space. An encoder maps an input vector x to a hidden representation h [2], the expression for a single hidden layer would be:

$$h = \sigma(W_{xh}x + b_{xh})$$

W and b are the weight and bias of the neural network, and σ is the nonlinear transformation function. The **decoder** decompresses the information back to its original dimensionality, the expression for a single hidden layer is:

$$z = \sigma(W_{hx}h + b_{hx})$$

Variational Autoencoders have a **reconstruction loss** when the data is decompressed. This reconstruction loss is calculated with the *Kullback-Leibler* (KL) *divergence*. KL divergence is a way of measuring how much one probability distribution differs from another. In a VAE, we want to measure how different our normal distribution with *mean* μ and *variance* σ^2 is from the standard normal distribution. KL divergence can be expressed as:

$$\sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - log(\sigma_i) - 1$$

All of this process creates a bottleneck for data (see Figure 4.3) that ensures only the main structured part of the information can go through and be reconstructed. A recommended read to understand Variational Autoencoders is the book *Generative Deep Learning* [11], which exemplifies it in an "easy-to-understand" way.

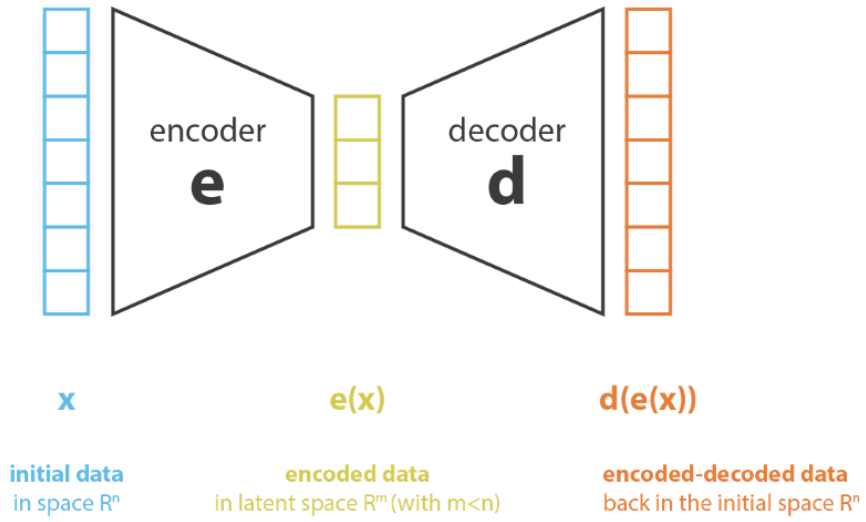

Figure 4.3: Structure of a Variational Autoencoder. The data is compressed using the encoder, and then decompressed with the decoder. Both are neural networks used to transform the data.

4.3 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (**DBSCAN**) [17] is a technique used to group samples. It works on the idea that if a particular point belongs to a cluster, it should be close to a lot of other points in that cluster. It is based on two parameters: radius and minimum points. For each object of a cluster, the neighborhood of a given radius (*Eps*) has to contain at least a minimum number of objects (*MinPts*):

$$N_{Eps} = \{q \epsilon D / dist(p, q) < Eps\}$$

In the previous expression, D represents the database of objects. The neighborhood of an object p has a certain number of q objects at a distance lower than the radius. The object p will be a core point if it contains at least a minimal number of points in its neighborhood (see Figure 4.4):

$$N_{Eps}(p) > MinPts$$

DBSCAN searches for the clusters by checking the neighborhood of each object in the dataset. If the neighborhood of an object p contains more than *MinPts*, a new cluster with p as a core object is created. The points that do not belong to any cluster are **outliers**. By knowing this, we can avoid misclassifying a sample from another type of cancer as a Head and Neck kind.

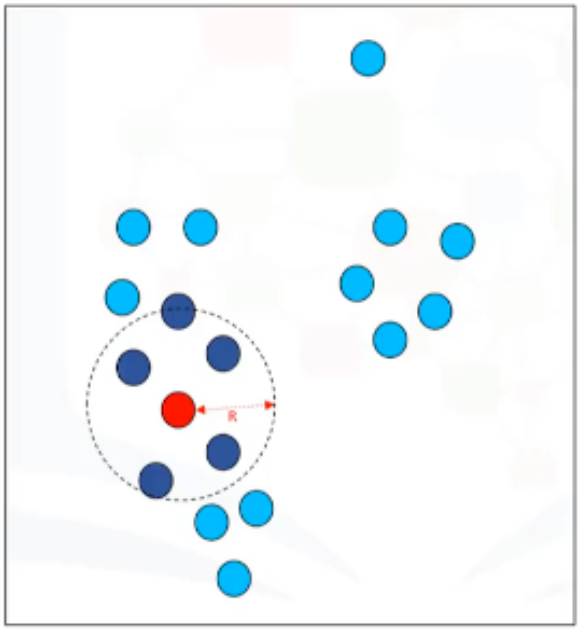

Figure 4.4: Example of DBSCAN. For a certain radius R and 6 minimum points, the red point would be a core point.

4.4 Applying Deep Active Learning to this project

In our work, Deep Active Learning will be applied in the following way (see Figure 4.5 for a representation of the algorithm used). The dataset is divided in two different sets:

- A labeled set, with instances that includes their classification.
- An unlabeled set, with instances without its classification.

To initialize the labeled set, a selection of random instances is made, which are half from the positive class and the other half from the negative class. This is done because the neural network will use the labeled pool to be trained, and the dataset is unbalanced so we do not want more priority on the majority class when labelling instances. The unlabeled pool is filled with the rest of instances. The neural network is trained with this labeled set, and will not be re-trained to save processing time and to simulate an environment where human labelling is too expensive (in a medical context).

After the pools are initiated and the neural network is already trained with good testing accuracy, it is time to iterate the labelling process. A query is done to select samples from the unlabeled set, and we use DBSCAN to create clusters from both the labeled set and the selected instances from the mentioned query. This way, we check whether the selected instances are similar or not to those that have already been categorized. Those instances that are similar enough will have a class provided by the neural network, and will be added to the labeled pool. If not, then it will be kept in the unlabeled set. We keep selecting random unlabeled instances and classifying them until we run out of them, or until we reach the maximum number of iterations.

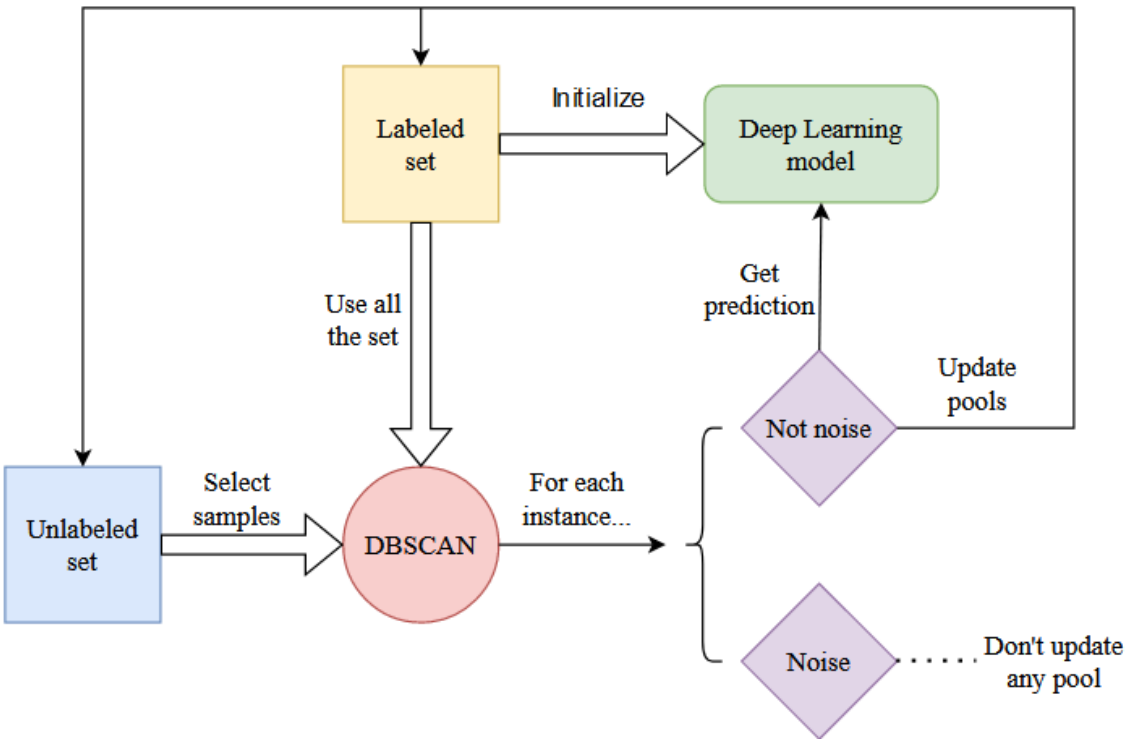

Figure 4.5: Our Deep Active Learning framework. At first, the labeled and unlabeled set are initialized, with the labeled set being used to train the Deep Learning model. In each iteration, a number of samples are selected from the unlabeled set, and they are analyzed with DBSCAN using also the labeled pool to check for outliers. For each instance selected from the labeled set, if the DBSCAN classifies it as noise, then the pools are not updated. If not, the instance is labeled with the Deep Learning model, and both pools are updated. The process ends when the number of iterations are reached, or if there are not any unlabeled samples to select.

Chapter 5

Objectives

5.1 Main objective

The objective of this project is to learn what **Deep Active Learning** is and how to apply it. To demonstrate its use, we will study the cell gene activation of patients that may have head and neck cancer. The main investigation focuses on the automatic labeling of instances by using Deep Active Learning, which will be applied to a dataset of head and neck cancer samples.

However, this dataset has both high dimensionality together with a low sample size, which stands as a tough challenge to analyze. Before applying Deep Active Learning we need to represent the data in a smaller way to make the work easier. To do this, we will apply techniques such as Random Forest and Variational Autoencoders.

DeepAL [14] is used as a base program for the application of Deep Active Learning, and the necessary changes will be added to adjust it to our requirements. The dataset used for this comes the *Cancer Genome Atlas Program* [16], having 564 samples from head and neck cancer cases, each instance being represented by 60498 attributes.

5.2 Secondary objectives

5.2.1 Learning of complex data management

During the Master's Degree, the student has worked with many datasets over the course, applying different techniques to study them. However, these datasets are often prepared and cleaned, so that they are more manageable to work with them and learn the basics of artificial intelligence. In the real world, the gathered data may have incorrectly entered values for its attributes, or may even have some missing values. In some cases, the class label could be wrong too. This gets more complex in a medical context, as the information is too large. By working on this project, the student will be able to learn to face these obstacles that are very common in investigation.

5.2.2 Investigation of articles and other sources

Deep Active Learning is a relatively recent technique, but there are already many articles about it that can be consulted, as can be seen in Chapter 6. It is important to know the sources where these reports can be found, and then there is the task of understanding the articles. This involves learning how other research may relate to our project, figuring out the approaches that could be applied in a similar way. The authors' justifications, whether graphs, mathematical expressions or other explanations, should be valued.

Another essential part of this work is Python programming, of which it is necessary to use libraries for experimentation. They are used to implement techniques needed to handle data, but they often have many adjustable parameters. These libraries have documentation that must be read in order to make good use of them.

Chapter 6

Background information

Other work related to this project is presented in this chapter. The software that has been used to help with our experiments will be presented first, and then different uses of Deep Active Learning that served as inspiration are displayed.

6.1 Deep Active Learning in Python

The article *DeepAL: Deep Active Learning in Python* [14] shows a great contribution by releasing a public library in GitHub that is able to apply Deep Active Learning in Python. This library can be used with many datasets, has a lot of adjustable parameters and it is easily modifiable. Because of its flexibility, **this software will be used as the basis of our project**.

Originally, the strategy followed by this software trains an initial classifier based on a labeled pool. For each round, the algorithm selects n samples from the unlabeled pool (this number is set by the user) according to a query strategy. DeepAL has many query strategies available, selecting instances randomly, with the least confidence, with largest entropy... Finally, when the unlabeled examples are classified, they are moved into the labeled pool, and the classifier is trained again based on this updated pool. The modifications made to the software, together with its operation and other auxiliary files are explained in Chapter

10.

6.2 Diagnosis of COVID-19 with Deep Active Learning

CT scans (Computed tomography) are used to get information about the lungs of patients, but the large scale of annotations demands too much time and effort. The article *COVID-AL: The diagnosis of COVID-19 with deep active learning* [34] explains how it overcomes the problem of having large data by using deep active learning. COVID-AL achieves great performance on COVID-19 diagnosis, while having only 30% of labeled data. In contrast to our project, a distinction is made here between three classes (coronavirus pneumonia, common pneumonia and normal controls) compared to the binary classification we are dealing with (cancer or not cancer). The dataset is also annotated differently because they are working with images, so other methods will be used, like convolution.

The COVID-AL framework is shown in Figure 6.1. COVID-AL selects the most informative samples with a hybrid sampling strategy, and they are later added to the labeled pool. This hybrid strategy is more complex in comparison to our random strategy, as it combines diversity-based and loss-prediction-based sampling strategies. Diversity-based sampling states that a patient can have several scans associated with him, and the classification network can make different decisions in the classification (which indicates high inconsistency), so a higher degree of inconsistency indicates that more information is contained in the data sample. The loss-prediction-based sampling strategy uses a regression network, previously trained with ground-truth labels, to compute the classification loss of the selected samples.

6.3 Deep Active Learning for classifying cancer

Saving annotation cost is still a crucial need in medical datasets, but the objective is to reach a good performance so that it can actually help humans efficiently. The article *Deep active learning for classifying cancer pathology reports* [24] uses active learning algorithms

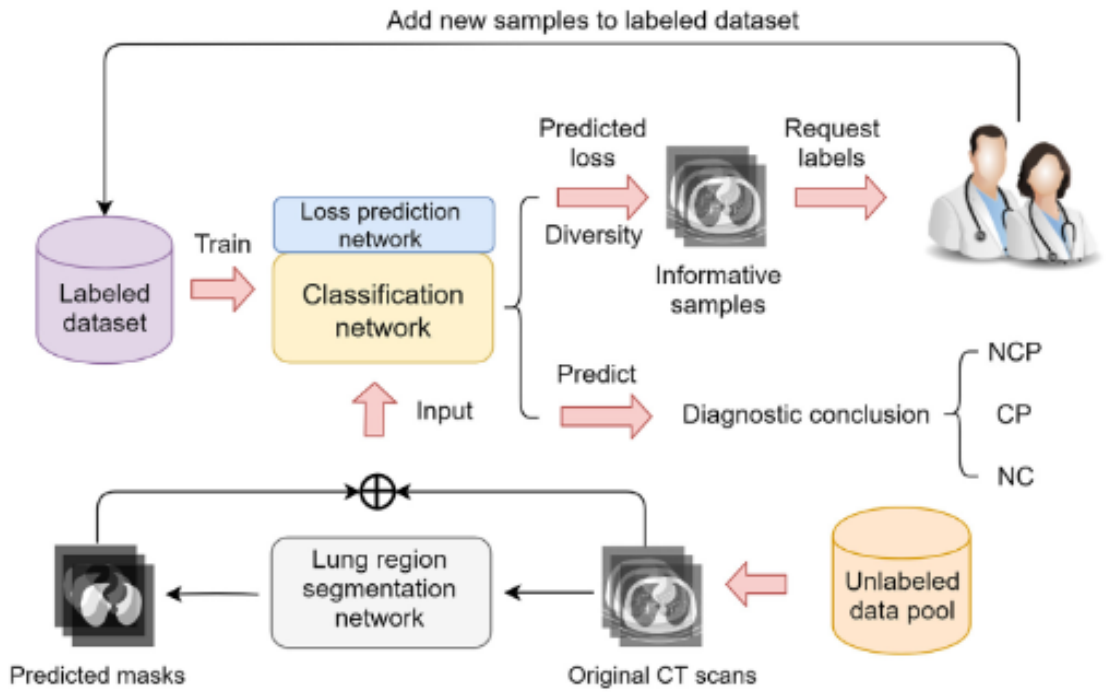

Figure 6.1: COVID-AL framework. The candidate samples are sorted in ascending order according to the predicted loss and computed diversity, respectively. Both rankings are summed and its weights are easily adjustable to adapt to the tasks.

to analyze cancer pathology reports. Although the mentioned work is focused in Natural Language Processing, it is still relevant to our project as it uses the same technique and covers the cancer disease. Another similarity is the fact that their datasets are imbalanced, as it is common in clinical text classification. Some classes are represented by less than a few hundred pathology reports, while others are represented by tens of thousands of reports. However, these datasets are much larger than the one we are using. One of them has 15000 instances, and a smaller version has 1000 instances, in comparison to our 564 sample size.

It also uses several sampling strategies:

- **Uncertainty sampling**: the lower the confidence of a given sample, the more informative it will be for the model.

- **Diversity sampling**: aims to maximize the diversity of the training dataset.
- **Query-by-comittee**: trains multiple predictive models (the committee), and samples are ranked based on how much disagreement there exists within the committee. Samples associated with the highest disagreement are selected and added to the training dataset.
- **Density-weight method**: attempts to select samples that the model is uncertain about but that are also similar to other samples in the dataset.

Figure 6.2 shows the flowchart of the computational pipeline used during the active learning experiments. First, a Convolutional Neural Network (CNN) is initially trained. Then, one of the strategies mentioned before is used to select instances, and the CNN is trained from scratch (it does not continue training the weights from the previous CNN). These instances are removed from the set in the following iterations, and the process is repeated until there are no more samples to select.

6.4 Survey of Deep Active Learning

Deep Active Learning has been used in a lot of investigations, but it still lacks an unified classification framework. The article *A Survey of Deep Active Learning* [8] does an excellent job at describing Deep Active Learning, explaining why the combination of Deep Learning and Active Learning achieves good results and showing examples of other works in different fields (text classification, image recognition, object detection...). It is a good read to fully understand Deep Active Learning.

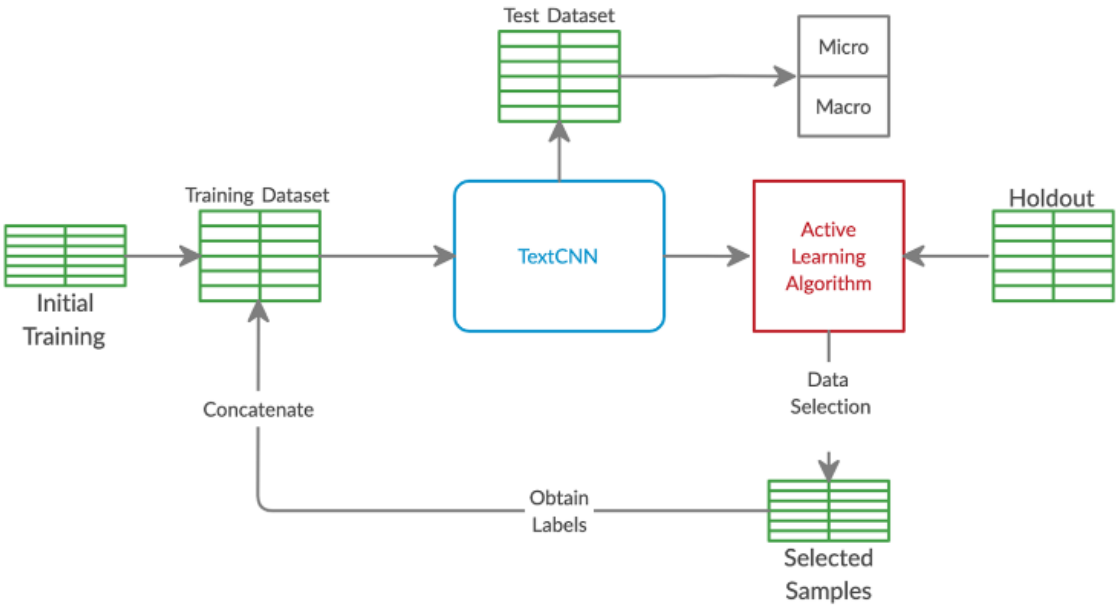

Figure 6.2: Framework for active learning applied to cancer reports. It initially trains a Convolutional Neural Network with a training set, and then a sampling strategy is used to select samples. The CNN is trained again with these instances, and the process is repeated until there are no more samples to select.

Chapter 7

Experiments design

The experimentation will be organized in two parts. In the first part, a **dimensionality reduction** will be performed on our dataset. In this way, the data will be more manageable and we will be able to proceed with the following steps. Data preprocessing will be applied beforehand, to make the dataset feasbale for the analysis. To achieve reduction, the techniques used are Random Forests and Variational Autoencoders (see Chapter 4 for more information). Random Forest are used first to select important attributes, and then a Variational Autoencoder applies its encoder to reach the latent space. Two architectures for encoder and decoder are tested, both starting from a dimensionality of 5000: one that reaches a latent space of 2000 (see Figure 7.1), and another one decreased until 500 (see Figure 7.2). The results are shown in Chapter 8.

In the second part, **Deep Active Learning** will be applied to the new, reduced dataset. DBSCAN checks if the unclassified instances are outliers, and if they are not, they are labeled with a Deep Learning Model. A dataset with randomly generated values is studied too, to check how DBSCAN behaves when comparing unlabeled samples to labeled ones. Chapter 9 shows the procedure and the parameters used for this experiment.

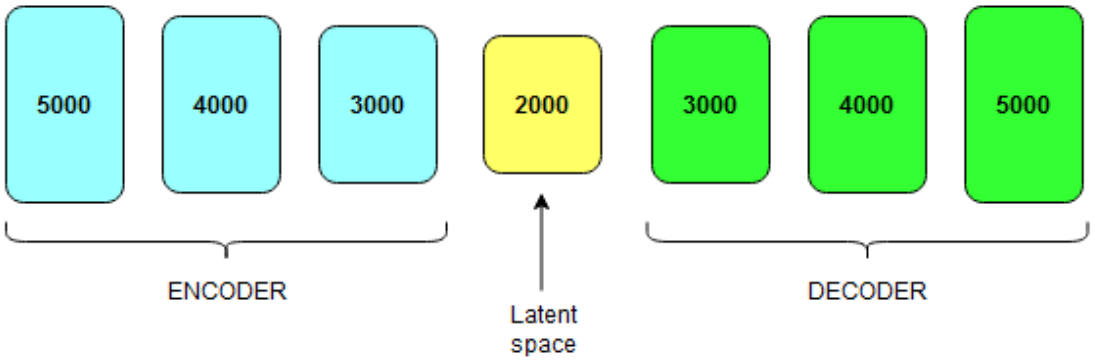

Figure 7.1: Architecture used for the Variational Autoencoder, with a latent space of 2000. Numbers represent layer widths, so we are therefore changing the dimension of the representation space.

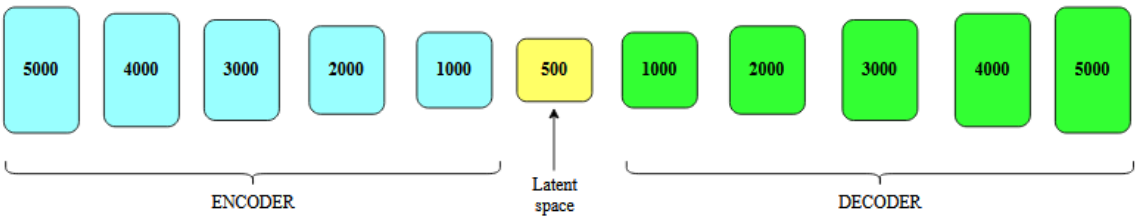

Figure 7.2: Architecture used for the Variational Autoencoder, with a latent space of 500. Numbers represent layer widths, so we are therefore changing the dimension of the representation space.

Chapter 8

Dimensionality reduction

In this first part of the experiments, we will detail how we managed to reduce the high dimensionality of the dataset we are working with.

8.1 Preprocessing data

Real-world data is messy and when created, the datasets can contain errors, miss values, have duplicate data... This is why it is important to preprocess data, to identify and rectify these problems as soon as possible. For example, in our dataset there are attributes that only contain zero values in all instances. These characteristics do not provide meaningful information, so they were deleted. In total, 8589 attributes were removed from the dataset, so now the dimensionality is 564×51909.

Another problem we can find is that the attributes can have different proportions, because the features represent distinct types of information. **Normalization** must be applied to overcome this. In our case, we want all attributes to have values between 0 and 1. This is done to use a common scale, without distorting the differences in value ranges or losing information. The normalization is done following the **MinMaxScaler**:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

8.2 Using Random Forests

Now, we proceed to use a **Random Forest** classifier with 100 trees, and using 70% of the dataset for training and 30% for test. The classifier is able to achieve a 68% accuracy on the test set. It is important to point out that although not applying normalization gives the classifier more accuracy (at around 83%), the Variational Autoencoder is not able to give good results without this preprocessing. Figure 8.1 shows the results of this classifier. We can see that the non-cancer class is the one that is classified better, as the positive class has more exceptions where the predicted label is closer to the opposite class.

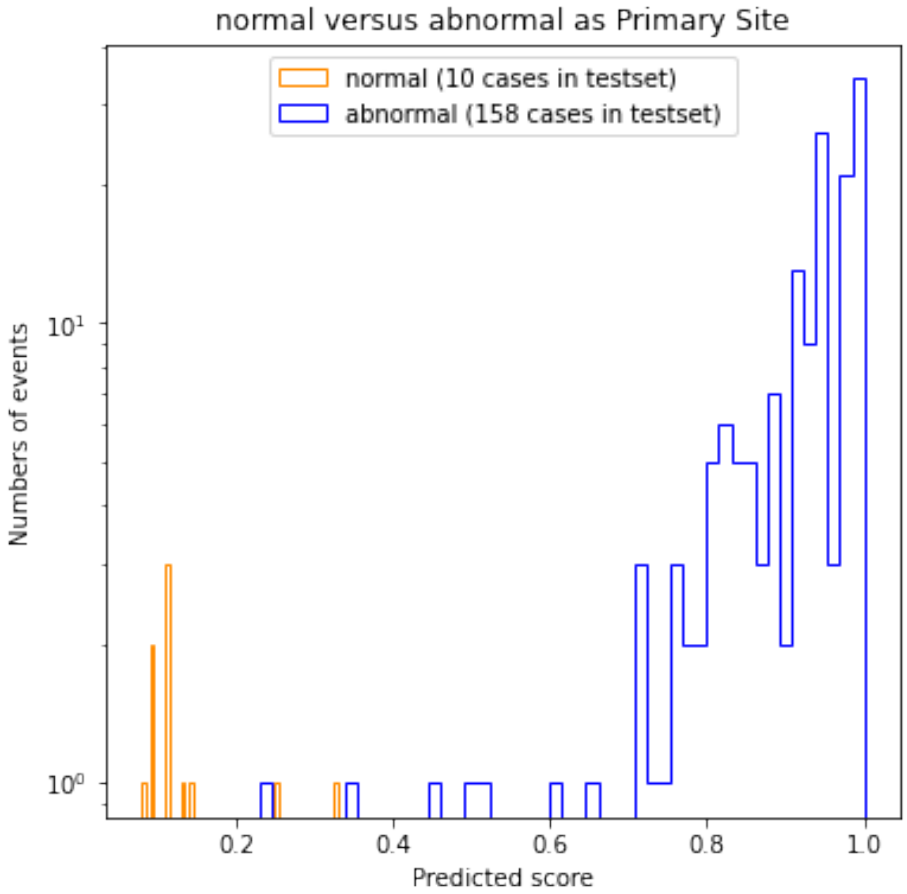

Figure 8.1: Results of the first Random Forest when classifying non-cancer instances (orange) and cancer instances (blue).

With these results, we select the most important attributes of the Random Forest classifier. Specifically, we select **5000 features**, meaning that the dimensionality now is 564×5000. This can be seen as a very big reduction (from 51909 attributes), but is a risk that should be taken, as the research can not progress with a dataset with such a high dimensionality. A second Random Forest is applied with this new dataset (5000 attributes), and the results can be seen in Figure 8.2. This time, the accuracy of the classifier is 74%, which demonstrates that the selection of important attributes is improving performance.

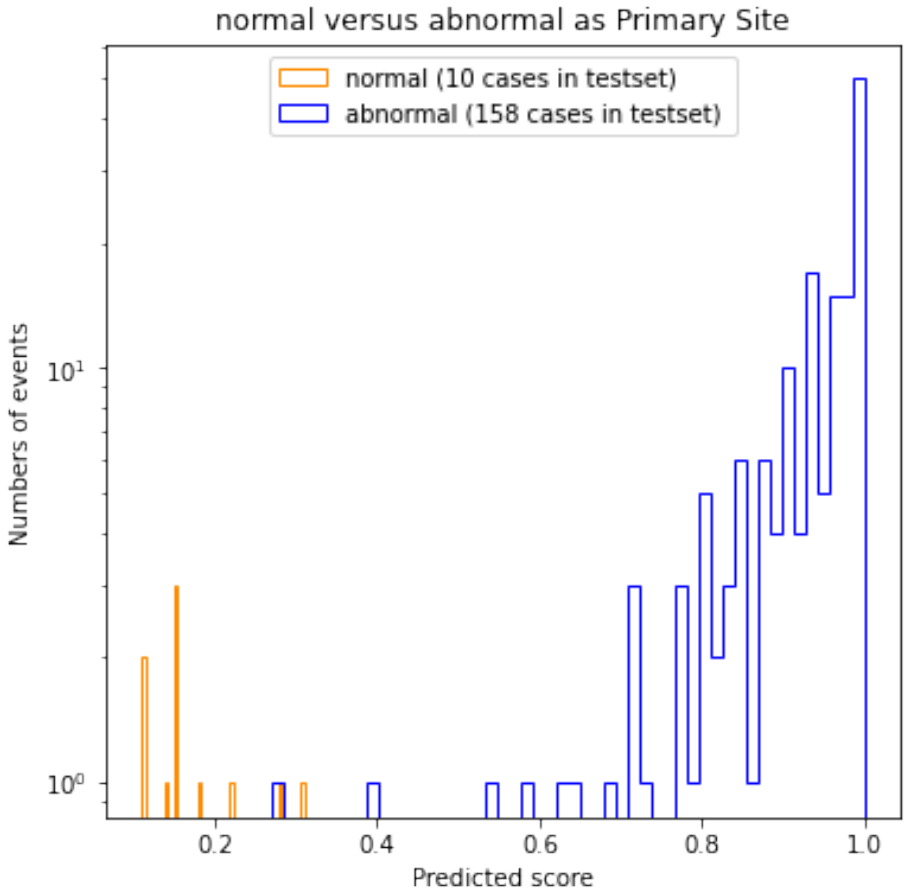

Figure 8.2: Results of the second Random Forest when classifying non-cancer instances (orange) and cancer instances (blue).

8.3 Using Variational Autoencoders

8.3.1 Pedagogic objective

The first application of a VAE was done with a reduction to two dimensions. This is done as a first approach to have a more intuitive vision of the problem, using this demonstration as a guide to observe the difficulty. The representation of this reduction can be seen in Figure 8.3, this shows us that we are working with a very complex problem. It is very hard to optimize a classifier with high dimensionality and unbalanced classes (significantly more positive than negative instances). This image displays a difference between a clear zone (where there are only positive classes) and a more confusing zone (where the positive and negative classes are more mixed). If an instance was classified as positive in the clear zone, then we will have to suspect a lot about that sample having cancer.

8.3.2 Performance objective

This time we are looking to a reasonable reduction to then be able to work with this new dimensionality. The experiments are done with two kind of architectures, both types use **ReLu** activation on each layer and **sigmoid** on the final layer of the decoder:

- One architecture goes from 5000 to 2000 dimensions on the encoder, then back to 5000 on the decoder.
- Another architecture goes from 5000 to 500 dimensions on the encoder, then back to 5000 on the decoder.

In all of the experiments, 200 iterations are done for the training of the VAE. We try changing the function loss between Binary Cross Entropy and Binary Focal Cross Entropy. This last one is usually useful for unbalanced datasets, but as we will see later it does not grant us good results. The parameter alpha of this loss function controls the weights of the majority class. The results are annotated by encoding and decoding the dataset, and then using the Random Forest classifier to get the classes on the instances of the reconstructed

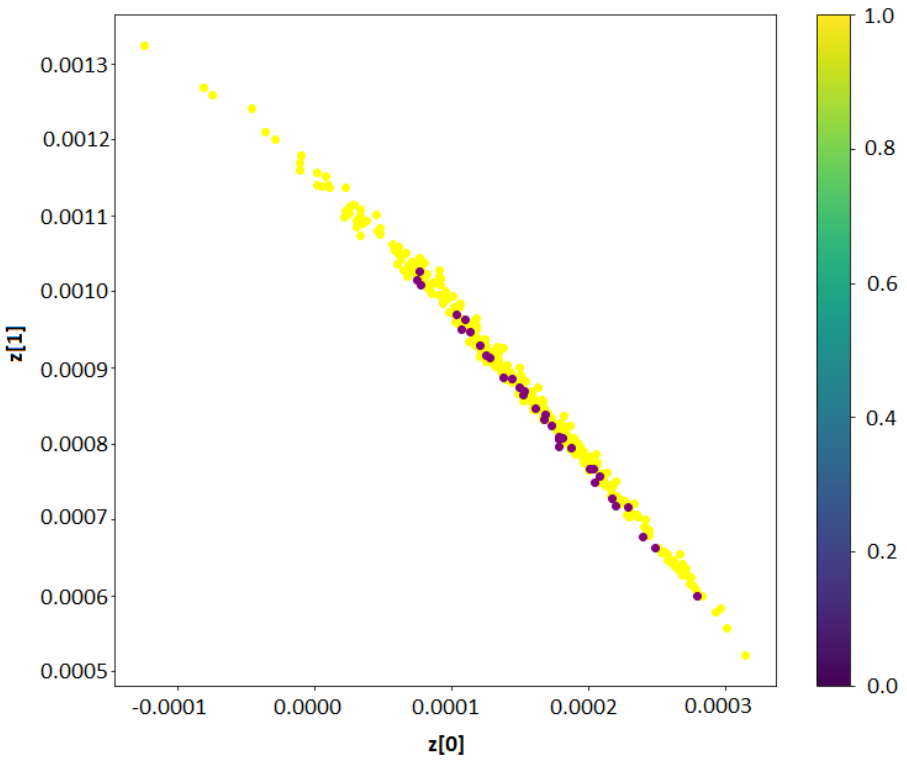

Figure 8.3: Dataset reduced to two dimensions. Yellow dots are cancer instances, purple dots are non-cancer

dataset. Table 8.1 shows the results of using the architecture 5000 to 2000, while Table 8.2 shows the results of the architecture 5000 to 500.

8.4 Results overview

We can see in the results shown in Table 8.1 and 8.2 that a lot of samples are classified as the positive class. This demonstrates that we are not only dealing with a very large dataset in terms of attributes, but the classes are unbalanced too, which makes it even harder to work with it. That is why Active Learning will be relevant in the following experiment, because depending on the query strategy chosen we can avoid this problem, as it is mentioned in [8]:

> *Furthermore, on tasks with a large number of unique labels with extreme class imbalance, active learning can significantly mitigate the effects of class imbalance and improve performance on the rare classes.*

The best results are obtained when using Binary Cross Entropy as a loss function and using the 5000 to 2000 architecture. More specifically, **the experiment in the forth row on the Table 8.1 gave us the best accuracy**. Binary Focal Cross Entropy was not helpful for this problem, even after changing its parameters, as it did not grant us good results for any architecture. For the next part of the experimentation, using Deep Active Learning, we will use the new dataset that arises from encoding the original one. This means that we will be using a dataset with a 564×2000 dimensionality.

ENCODER	DECODER	LOSS	LR	BATCH SIZE	RESULTS
5000 to 2000	2000 to 5000	Binary Cross Entropy	10^{-3}	12	All instances predicted as 1
5000 to 2000	2000 to 5000	Binary Cross Entropy	10^{-3}	128	All instances predicted as 1
5000 to 2000	2000 to 5000	Binary Cross Entropy	10^{-2}	12	Does not classify
5000 to 2000	2000 to 5000	Binary Cross Entropy	10^{-4}	12	Good accuracy on positive samples, negative samples not so much but still decent accuracy
5000 to 2000	2000 to 5000	Binary Cross Entropy	10^{-5}	12	Very inconsistent results, sometimes it classifies very well, sometimes it has really bad accuracy
5000 to 2000	2000 to 5000	Binary Focal Cross Entropy (gamma=2)	10^{-3}	12	All instances predicted as 1
5000 to 2000	2000 to 5000	Binary Focal Cross Entropy (gamma=5)	10^{-3}	12	All instances predicted as 1

Table 8.1: Experiments using 5000 to 2000 architecture

ENCODER	DECODER	LOSS	LR	BATCH SIZE	RESULTS
5000 to 500	500 to 5000	Binary Cross Entropy	10^{-3}	12	All instances predicted as 1
5000 to 500	500 to 5000	Binary Focal Cross Entropy (gamma=2)	10^{-3}	12	All instances predicted as 1
5000 to 500	500 to 5000	Binary Focal Cross Entropy (gamma=5)	10^{-3}	12	All instances predicted as 1
5000 to 500	500 to 5000	Binary Focal Cross Entropy (gamma=10)	10^{-3}	12	All instances predicted as 1
5000 to 500	500 to 5000	Binary Cross Entropy	10^{-4}	12	All instances predicted as a value between 0'79 and 0'9
5000 to 500	500 to 5000	Binary Cross Entropy	10^{-5}	12	All instances predicted as 1

Table 8.2: Experiments using 5000 to 500 architecture

Chapter 9

Application of Deep Active Learning

In this second part of the experiments, we will explain how Deep Active Learning was applied to our dataset.

9.1 Preparing the experiment

As we mentioned in the previous chapter, we will use a new reduced dataset. These data has been obtained after applying Random Forest to the original dataset, selecting the most important attributes, and then using the encoder of a Variational Autoencoder to reduce it down from 60498 characteristics to 2000. This means that we are now working with a 564×2000 dimensionality.

We will try using an "artificial" dataset too. These data will have the same dimensionality as the real one (564×2000), but the values will be randomly generated. This will be done to observe the behaviour of the algorithm.

We will be using DeepAL, a software that implements Deep Active Learning in Python. The strategy we want to follow differs slightly from the one implemented in DeepAL, as it is explained in Chapter 4.

9.2 Deep Active Learning applied to real data

First, we will apply Deep Active Learning to our real dataset, the one we previously reduced. The parameters used to run this experiment are:

- Number of instances initially labeled: **10**.
- Unlabeled instances selected per iteration: **6**.
- Iterations: **10**.

We can see that the parameters we decided have low values: this is because the dataset is not that big, and we are trying to select 50% of samples of each class in every iteration while the dataset is unbalanced. When DBSCAN is applied, its parameters are:

- Epsilon: **6**. This is the maximum distance between two samples for one to be considered as in the neighborhood of the other.
- Minimum points: **5**. Number of samples in a neighborhood for a point to be considered as a core point.
- Metric: **L2**. Metric to use when calculating distance between instances.

Epsilon has that value assigned just because it was found to be one of the better values after many trials. The minimum number of samples is small because there are not many instances to cluster per iteration, so the groups are not much bigger than that. L2 is the euclidean distance, and was the metric that better adjusted to the values of our samples.

Results are shown in Table 9.1. The neural network used in this experiment has an accuracy of 95'8% on test data.

9.3 Deep Active Learning applied to artificial data

Now, we will use the randomly generated dataset. These values will be used only for the query that occurs in each iteration, not for the initially labeled instances. The attributes

ITERATION	NUMBER OF NOISY SAMPLES OF DBSCAN	NEURAL NETWORK ACCURACY
1	0	100%
2	4	83%
3	1	100%
4	2	100%
5	3	100%
6	1	100%
7	2	83%
8	0	100%
9	0	100%
10	2	83%

Table 9.1: Results of applying Deep Active Learning with real data

values will also be between 0 and 1, to have the same scale as the original data. The parameters will be the same as the previous experiment, even for the DBSCAN. As the values are random, they do not have a real label class, so it does not make sense to measure the accuracy of the neural network on the selected samples: Instead, we will use it to watch how many instances are classified as positive and negative, even if they are not real values. The results are shown in Table 9.2.

9.4 Results overview

We can extract a lot of information from the results shown in Table 9.1. In principle, an interesting observation is the fact that some instances are perceived as outliers. Considering that we are dealing with real data, it is logical to think that they should only be classified as positive or negative classes. As mentioned in previous occasions, it should be noted that

ITERATION	NUMBER OF NOISY SAMPLES OF DBSCAN	SAMPLES CLASSIFIED AS POSITIVE	SAMPLES CLASSIFIED AS NEGATIVE
1	6	3	3
2	6	6	0
3	6	4	2
4	6	5	1
5	6	3	3
6	6	4	2
7	6	3	3
8	6	5	1
9	6	4	2
10	6	4	2

Table 9.2: Results of applying Deep Active Learning with random data

datasets may have values in their attributes entered incorrectly, or some samples may be mislabeled. It is not too far-fetched to think that there may be noisy instances, but it should also be noted that DBSCAN parameters may lead to unexpected results. We can also see that the neural network has great accuracy across all iterations. Selecting only 6 instances per iteration means that at most, the model fails to classify only one instance.

On the other hand, in Table 9.2 we can observe that all of the selected samples are viewed by the DBSCAN as outliers. This is a very important result, because this means that the Deep Active Learning algorithm is actually keeping the information from the real instances, so all the random data we provided is classified as outliers.

Chapter 10

Implemented software

This chapter is used to explained the software used to develop this project. All the code is available at https://github.com/enaes05/DeepAL_mod.

10.1 Dimensionality reduction

The reduction of the dataset is done with file *VAE_HeadAndNeck.ipynb*. The dataset is divided in two files, one with all the positive samples and another with the negative samples. At first, both files are loaded, then they are concatenated and divided into training and test (see Figure 10.1). After that, normalization is applied and the first Random Forest is trained. For the second Random Forest, the 5000 more important attributes are selected from the original datasets. The process is the same again, concatenating them and normalizing (see Figure 10.2) before starting the new training.

The Variational Autoencoder part comes next. A function is defined for sampling a point z in the latent space (see Figure 10.3). This is useful for wrapping any function into Keras layer, applying a function to a tensor that isn't already included as one of the out-of-the-box Keras layer types. The architecture for the model is loaded, both for encoder and decoder (see Figure 10.4), and once the parameters are set too, the VAE is trained. After proving that the data still has good performance after being encoded and decoded,

the reduced dataset is saved (564×2000 dimensionality).

10.2 Deep Active Learning application

As we mentioned in previous chapters, this is a modification of *DeepAL*. The main modifications are done in these files:

- *demo.py*: the main file, which allows to try experiments with different datasets and parameters. In our modification, the model is not re-trained and DBSCAN is used to check for noise. If a sample is seen as weird, then it is not updated in the labeled pool.
- *data.py*: used for loading the datasets and to initialize the labeled pool. The Head and neck cancer dataset is added, and when initializing the labeled set, it is intended that half of the instances are from one class (to overcome imbalance).
- *nets.py*: each dataset has a distinct architecture, so a new one is added for the cancer dataset.
- *random_sampling.py*: when selecting instances from the unlabeled pool, it ensures that classes are evenly divided by classes.

To try the experiment with the same parameters as the ones mentioned in Chapter 9:

```
$ python demo.py --dataset_name HeadAndNeck --n_init_labeled 10 --n_query 6
--n_round 10 --strategy_name RandomSampling
```

```
# Delete the first row for both sets, it only
# has identifiers that are not useful here
df_normal = df_normal.iloc[1:,]
df_normal = df_normal.drop(comun, axis=1)
df_abnormal = df_abnormal.iloc[1:,]
df_abnormal = df_abnormal.drop(comun, axis=1)
```

```
# Concatenate both datasets
data = pd.concat([df_normal,df_abnormal], ignore_index=True)
data = data.sample(n=data.shape[0], random_state=2)
# Number of trees for Random Forest
ntrees=100

# Selecting the last column as label
Y= data['label']
X= data.iloc[:,:-1]
X = np.asarray(X)
Y = np.asarray(Y)

# Training and test sets
test_size = int(np.floor(0.30*X.shape[0]) )
trainX, testX = X[:-test_size], X[-test_size:]
trainY, testY = Y[:-test_size], Y[-test_size:]
print(trainY.shape,testY.shape)
```

Figure 10.1: Concatenation of positives and negative files, and split into training and test sets

```
indexes = (-clf.feature_importances_).argsort()[:5000]
```

```
# Keep the 5000 important attributes
new_normal = df_normal.iloc[:, indexes]
new_normal['label'] = 0

new_abnormal = df_abnormal.iloc[:, indexes]
new_abnormal['label'] = 1
```

Figure 10.2: Selection of 5000 attributes

```
def sampling(args):
    """Reparameterization trick by sampling
        fr an isotropic unit Gaussian.
    # Arguments:
        args (tensor): mean and log of variance of Q(z|X)
    # Returns:
        z (tensor): sampled latent vector
    """

    z_mean, z_log_var = args
    # K is the keras backend
    batch = K.shape(z_mean)[0]
    dim = K.int_shape(z_mean)[1]
    # by default, random_normal has mean=0 and std=1.0
    epsilon = K.random_normal(shape=(batch, dim))
    return z_mean + K.exp(0.5 * z_log_var) * epsilon
```

Figure 10.3: Sampling method for the latent space in the Variational Autoencoder

```
# build encoder model
inputs = Input(shape=5000, name='encoder_input')
x = Dense(4000, activation='relu')(inputs)
x = Dense(3000, activation='relu')(x)
z_mean = Dense(2000, name='z_mean')(x)
z_log_var = Dense(2000, name='z_log_var')(x)

# use reparameterization trick to push the sampling out as input
# note that "output_shape" isn't necessary
# with the TensorFlow backend
z = Lambda(sampling,
           output_shape=(2000,),
           name='z')([z_mean, z_log_var])

# instantiate encoder model
encoder = Model(inputs, [z_mean, z_log_var, z], name='encoder')
encoder.summary()
```

```
# build decoder model
latent_inputs = Input(shape=(2000,), name='z_sampling')
x = Dense(3000, activation='relu')(latent_inputs)
x = Dense(4000, activation='relu')(x)
outputs = Dense(5000, activation='sigmoid')(x)

# instantiate decoder model
decoder = Model(latent_inputs, outputs, name='decoder')
decoder.summary()
```

Figure 10.4: Definition of both encoder and decoder

Chapter 11

Conclusions

11.1 Final conclusions

An extensive study has been carried out on a Head and Neck cancer dataset. These data presented many obstacles, being represented by a much larger number of attributes than the instances (564×60498 dimensionality), and having an imbalanced proportion of classes (519 for the positive class, 45 for the negative class). The dataset was reduced using a Random Forest classifier, selecting the 5000 most important attributes. Then, using a Variational Autoencoder the data was encoded to 2000 attributes. When this new dataset was decoded and tested on the Random Forest classifier, the accuracy was still similar to the original, proving that the reduction was done losing as little information as possible.

Deep Active Learning was applied to the new dataset (564×2000 dimensionality). This technique initializes a labeled pool to train a neural network, and another set with unlabeled samples. For a certain number of iterations, instances from the unlabeled pool are taken and their classes are predicted. Using DBSCAN, we check if those unlabeled samples are treated or not as noise: if not, then the predicted class is accepted and the labeled pool is updated. The results were successful, obtaining predictions of 83% to 100% accuracy, demonstrating that the framework was able to classify unlabeled samples that were never seen before. When trying a dataset with randomly generated values, all of the instances

were treated as noise, meaning that the model was able to identify fake samples that are not from real Head and Neck cancer.

This challenge has been useful as a reminder that real-world data is not easy to handle, requiring preprocessing and investigation of what techniques to use. There is no right approach for all cases, it is a matter of adapting to the specific problem.

11.2 Future improvements

We have applied Deep Active Learning by taking advantage of the combination that Deep Learning and Active Learning provide, needing few samples to achieve good learning performance. Applying this to the study of cancer is a further step to help in its diagnosis and research. The focus here was the Head and Neck cancer kind, but it would be interesting to see it applied to other types as well. When the artificial dataset was used to observe the behaviour of Deep Active Learning with random data, trying data from another type of cancer would be useful to see how they are classified: if they are labeled more close to the positive or the negative samples, or they are directly treated as noise.

If we still aim our attention to the Head and Neck dataset used, then it would be appropriate to further investigate more parameters, error functions and architectures of the models used. For example, trying other query strategies could result in a more efficient model, as it has only been tested with a randomized selection method.

Even if Deep Active Learning is a recent technique, the *state-of-the-art* is always evolving and bringing new discoveries. New articles and projects can be found and be extrapolated to this work. Other medical data could be used too, and it might even have better results if this information is represented with less attributes (lower dimensionality).

Bibliography

[1] Mina Abbaszade, Vahid Salari, Seyed Shahin Mousavi, Mariam Zomorodi, and Xujuan Zhou. Application of quantum natural language processing for language translation. *IEEE Access*, 9:130434–130448, 2021.

[2] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[3] HV Ravish Aradhya et al. Object detection and tracking using deep learning and artificial intelligence for video surveillance applications. *International Journal of Advanced Computer Science and Applications*, 10(12), 2019.

[4] Larry Campbell, A Lotmin, Marie MG DeRico, and Clark Ray. The use of artificial intelligence in military simulations. In *1997 IEEE international conference on systems, man, and cybernetics. Computational cybernetics and simulation*, volume 3, pages 2607–2612. IEEE, 1997.

[5] Genesis Care. Between 12,000 and 14,000 cases of head and neck cancer are detected in spain each year. https://n9.cl/p4ma2. Accessed: 2021-09-29.

[6] Amal Chouchene, Adriana Carvalho, Tânia M Lima, Fernando Charrua-Santos, Gerardo J Osório, and Walid Barhoumi. Artificial intelligence for product quality inspection toward smart industries: Quality control of vehicle non-conformities. In *2020 9th international conference on industrial technology and management (ICITM)*, pages 127–131. IEEE, 2020.

[7] Adele Cutler, D Richard Cutler, and John R Stevens. Random forests. In *Ensemble machine learning*, pages 157–175. Springer, 2012.

[8] Kevin De Angeli, Shang Gao, Mohammed Alawad, Hong-Jun Yoon, Noah Schaefferkoetter, Xiao-Cheng Wu, Eric B Durbin, Jennifer Doherty, Antoinette Stroup, Linda Coyle, et al. Deep active learning for classifying cancer pathology reports. *BMC bioinformatics*, 22(1):1–25, 2021.

[9] Andre Esteva, Katherine Chou, Serena Yeung, Nikhil Naik, Ali Madani, Ali Mottaghi, Yun Liu, Eric Topol, Jeff Dean, and Richard Socher. Deep learning-enabled medical computer vision. *NPJ digital medicine*, 4(1):1–9, 2021.

[10] Jonathan Folmsbee, Xulei Liu, Margaret Brandwein-Weber, and Scott Doyle. Active deep learning: Improved training efficiency of convolutional neural networks for tissue classification in oral cavity cancer. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 770–773. IEEE, 2018.

[11] David Foster. *Generative deep learning: teaching machines to paint, write, compose, and play.* O'Reilly Media, 2019.

[12] Gautham Krishna Gudur, Prahalathan Sundaramoorthy, and Venkatesh Umaashankar. Activeharnet: Towards on-device deep bayesian active learning for human activity recognition. In *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, pages 7–12, 2019.

[13] Miriam Harris, Amy Qi, Luke Jeagal, Nazi Torabi, Dick Menzies, Alexei Korobitsyn, Madhukar Pai, Ruvandhi R Nathavitharana, and Faiz Ahmad Khan. A systematic review of the diagnostic accuracy of artificial intelligence-based computer programs to analyze chest x-rays for pulmonary tuberculosis. *PloS one*, 14(9):e0221339, 2019.

[14] Kuan-Hao Huang. Deepal: Deep active learning in python. *arXiv preprint arXiv:2111.15258*, 2021.

[15] Ahmed Hussein, Mohamed Medhat Gaber, and Eyad Elyan. Deep active learning for autonomous navigation. In *International Conference on Engineering Applications of Neural Networks*, pages 3–17. Springer, 2016.

[16] National Cancer Institute. The cancer genome atlas program. https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga. Accessed: 2021-10-07.

[17] Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE, 2014.

[18] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[20] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

[21] Arzoo Miglani and Neeraj Kumar. Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges. *Vehicular Communications*, 20:100184, 2019.

[22] K Orhan, IS Bayrakdar, M Ezhov, A Kravtsov, and TAHA Özyürek. Evaluation of artificial intelligence for detecting periapical pathosis on cone-beam computed tomography scans. *International endodontic journal*, 53(5):680–689, 2020.

[23] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.

[24] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

[25] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. int. conf. on machine learning. 2001.

[26] Soumya Roy, Asim Unmesh, and Vinay P Namboodiri. Deep active learning for object detection. In *BMVC*, page 91, 2018.

[27] Burr Settles. Active learning literature survey. 2009.

[28] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[29] Geoff Skinner and Toby Walmsley. Artificial intelligence and deep learning in video games a brief review. In *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, pages 404–408. IEEE, 2019.

[30] Maria Torres Vega, Cristian Perra, Filip De Turck, and Antonio Liotta. A review of predictive quality of experience management in video streaming services. *IEEE Transactions on Broadcasting*, 64(2):432–445, 2018.

[31] Jaromír Vrbka and Zuzana Rowland. Using artificial intelligence in company management. In *International Scientific Conference "Digital Transformation of the Economy: Challenges, Trends, New Opportunities"*, pages 422–429. Springer, 2019.

[32] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, 2016.

[33] Shuihua Wang, Yin Zhang, Tianmin Zhan, Preetha Phillips, Yu-Dong Zhang, Ge Liu, Siyuan Lu, and Xueyan Wu. Pathological brain detection by artificial intelligence in

magnetic resonance imaging scanning (invited review). *Progress in Electromagnetics Research*, 156:105–133, 2016.

[34] Xing Wu, Cheng Chen, Mingyu Zhong, Jianjia Wang, and Jun Shi. Covid-al: The diagnosis of covid-19 with deep active learning. *Medical Image Analysis*, 68:101913, 2021.

[35] Ye Zhang, Matthew Lease, and Byron Wallace. Active discriminative text representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[36] Wencang Zhao, Yu Kong, Zhengming Ding, and Yun Fu. Deep active learning through cognitive information parcels. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 952–960, 2017.