Master Thesis

Active Learning for Semi-Automatic Dataset Labelization

Moreno La Quatra

Academic Supervisor
Elena Maria Baralis
Company Supervisor
Matthieu Ospici
Master in Software and Digital Systems

Politecnico di Torino
Italy, Turin
July 2018

Abstract

Lately, Convolutional Neural Networks have been successfully applied to solve several kinds of tasks in the context of Supervised Learning. Labelled datasets are a crucial element in those systems and, while the collection of unlabeled data is considered inexpensive, their annotation is usually the most expensive part of the process. Active Learning techniques aim to solve this problem by reducing the number of required annotations to obtain the desired performance. The master thesis explores the state-of-the-art in the domain of image classification, where most of the classical approaches are unsuccessful because of the nature of input data. The proposed method combines both the input space exploration, using CNN derived distance measure, and the refinement of the decision boundaries, using uncertainty based techniques. Moreover, Active Learning techniques have been integrated with Semi-Supervised Learning methods to improve model accuracy and perform automatic labelization.

Sommario

Durante gli ultimi anni, le Reti Neurali Convoluzionali sono state utilizzate con successo per compiere diversi tipi di compiti nel contesto dell'apprendimento automatico. Un elemento cruciale in questi sistemi sono le collezioni di dati classificati e, mentre la loro raccolta sia considerata come poco dispendiosa, l'annotazione manuale è spesso la parte più costosa del processo. Le tecniche di Active Learning mirano a risolvere questo problema riducendo il numero di annotazioni richieste per ottenere le performance desiderate. Questa tesi esplora lo stato dell'arte nel contesto della classificazione di immagini, ambito nel quale, la maggior parte degli approcci classici non hanno successo a causa della natura dei dati trattati. Il metodo proposto combina sia l'esplorazione della varietà immagini presenti nella collezione, usando delle distanze derivate dalle CNN, sia l'affinamento dei margini di decisione, utilizzando il valore di probabilità dato dalla rete neurale durante la classificazione. Inoltre, le tecniche di Active Learning sono state integrate con quelle di Semi-Supervised Learning per migliorare le performance della rete neurale e annotare automaticamente delle immagini specificatamente scelte.

Acknowledgements

I take this opportunity to thank all the people who have given me the necessary support for the success of this project.

First of all, I want to express my gratitude to the Atos company and all the people whom advice have made it possible. I want to especially thank my tutor Matthieu Ospici for his valuable suggestions during the internship, my friendly colleagues Arslen, Duc-Anh, Gabriel, Ismaïl, Loïc, Valentin, Victor for the inspiring discussions and all the other team members for their support.

Academically, I would like to acknowledge my supervisor Elena Maria Baralis for the notions learned during her university courses and for her assistance during this internship. Moreover, being part of a double degree program I want to express my sincere thanks to all Turin and Grenoble teachers and friends who have made this experience memorable not only from the educative but also from a personal perspective.

Special thanks go to my parents, for their support to follow my wishes, for their love and their continuous positive reinforcement. I express my gratitude to my sisters who have emotionally encouraged me during this journey. I am grateful to my aunt Benedetta who sustained me throughout my life and my grandparents Ignazio, Vincenza, Salvatore and Catena for their undying support. Last but not least, my heartfelt thanks to Viviana for her love, precious patience and steadfast encouragement during these years.

Contents

List of Figures

Chapter 1

Introduction

Over the last two decades, Machine Learning has become the core technology in a lot of intelligent systems. These algorithms make use of data jointly with statistical techniques to give the system the ability to *learn* without being explicitly programmed. Machine Learning techniques can be divided into three classes: *Unsupervised*, *Semi-Supervised* and *Supervised Learning*. The three classes can be distinguished by the kind of data required in the different cases. While unsupervised learning methods exploit data without any additional knowledge, the supervised learning methods require supplementary annotation associated to each sample, that will be used to teach the system. Finally, using semi-supervised learning, annotated data are used to teach the system combined with annotated data to gain additional knowledge.

1.1 Internship Context

This internship takes place in the *computer vision* R&D team of *Atos* in Grenoble. The team is part of the *Extreme Big Data (xBD)* division of the company, created after the acquisition of *Bull*, which specializes HPC (High-Performance Computing). The R&D department is composed of 10 teams, among which there is the *xBD* team that create and optimize Data Science tools to be used on supercomputers.

Recently, the team has been focused on the creation and improvement of computer vision based systems. Most of the technologies used in the team-projects apply deep learning methods and need valuable annotated data in order to have good performances. From these considerations stems the need of state-of-the-art active learning system, capable of minimizing human efforts to create annotated datasets.

1.2 Motivations behind Active Learning techniques

Recent approaches in machine learning are focused on learning from a massive amount of data. Unfortunately, most of the data available today do not contain the additional information needed by supervised techniques and so they require to be manually annotated by a human agent. *Active Learning*, the subject of this master thesis, addresses this problem trying to select the most *informative* samples (a subset of the complete dataset) for the labelization, in order to reduce human effort and obtain a model that matches as closely as possible the performance of the model trained on the fully-annotated dataset.

In the literature, there exists different applications for the active learning methods, for example in the domain of speech recognition where these methods are widely used and capable to obtain impressive results. Another classic but challenging application of this technique is in the domain of image classification, where the annotation can be tedious, especially for a large quantity of data. For this master thesis, image classification is the task that we address using *deep neural networks*[1] based classifiers. Most of the existing literature on active learning makes use of different type of classifiers such as SVMs[2]. The use of neural networks introduces new possibilities as well as new problems to take into account and they are detailed and addressed in this thesis.

A successful active learning method should be able to obtain a more accurate model with the same amount of annotated data or should obtain the same performance of a model learned using a larger amount of data. In this case, the reduction of the amount of annotations will lead to a reduction in the time spent by humans (often domain experts) to create competitive labelled datasets. Active Learning methods can be divided into two main categories presented hereunder.

Pool-Based Active Learning: is an approach in which there is the availability of a large pool of unlabelled samples. Active Learning algorithms, using a *query strategy*, iteratively select what are the best samples to teach the classifier.

[1]Neural networks are a set of algorithms that are designed to recognize patterns. They recognize numerical patterns contained in vectors, into which all real-world data be it images, sound, text or time series, are encoded.

[2]SVM is an acronym standing for *support vector machines*, which are supervised learning models used for classification or regression analysis

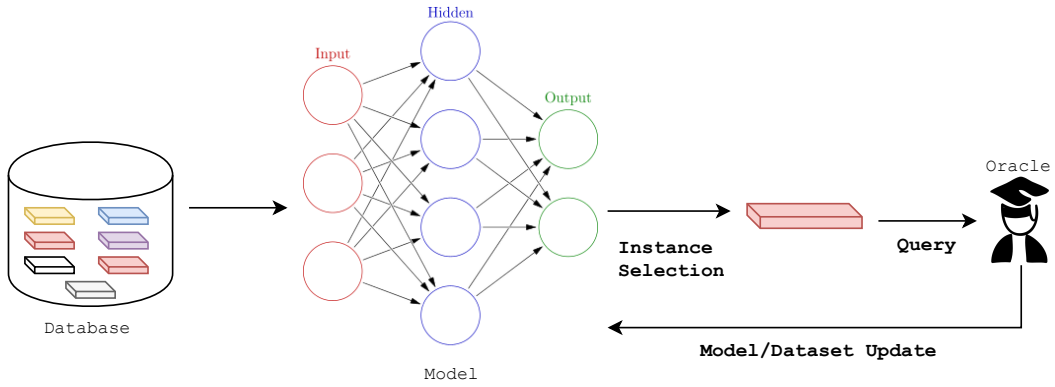


Figure 1.1: Pool-Based Active Learning Workflow

The Figure 1.1 graphically shows the workflow followed in this approach. As can be seen, the machine learning model is presented with a fixed pool of unlabelled instances. On each step, using the chosen strategy, the procedure selects one or more samples that it considers the *most informative* to speed up the learning. After being chosen, the instances are submitted to a human expert (the oracle) who provides the correct labels. The next step is to retrain the model, using additionally the newly available samples, and reiterate these steps till a condition on the maximum number of annotations or on the accuracy of the model is met.

Stream-Based Active Learning: is a slightly different approach with respect to the previous one. In this case, the learner is presented with a stream of unlabelled instances. For each incoming instance the model, using a given decision strategy, chooses whether to ask the oracle for annotation or not. The main difference is that, in the previous case the data samples are fixed, while in this case, there is a stream of samples and the learner should make a real-time decision on which samples could be useful to be part of the learning dataset. The Figure 1.2 explains graphically the case.

The algorithms belonging to this family differs from the pool-based ones in the information at their disposal. In the case of a fixed dataset, in fact, there is the possibility of ranking the samples and compare each one to the others while having a stream of data processed one-by-one, does not allow such ranking.

The context of this project is more oriented towards the pool-based approach but, some of the considerations can be extended to the stream-based methods. The next section will introduce the contribution of this thesis and details the concerned parts of the system.

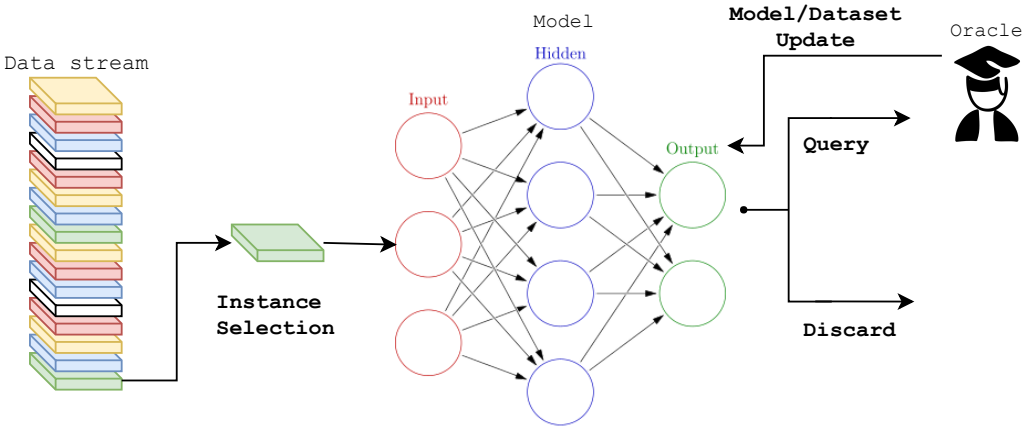


Figure 1.2: Stream-Based Active Learning Workflow

1.3 Contribution

This master thesis aims to explore state-of-the-art methods concerning the possible active learning strategies in the context of deep neural network classifiers. To better explain the contributions of this thesis it is necessary to sum up the fundamentals elements to define an active learning system, which are:

- The *model* (its class at least) used to perform the classification task.
- A *dataset* containing unlabelled samples.
- A query or *decision strategy* that is used to choose, between the unlabelled samples, the next one to be proposed to a human expert.
- The *human expert* (oracle) itself that will be used to obtain the correct labels.

Using a well-known architecture for the classifier and fully-labelled datasets for testing, different strategies have been analysed. In addition, a new method has been proposed that combines the information coming from different sources to create a unique score, used to rank each sample, according to the predicted contribution to the learning process. The methods will be validated by quantitative results on benchmark datasets.

Dealing with a partially annotated collection of data, in order to improve model's performances, semi-supervised learning methods have also been used. Under these conditions, multiple criteria have been used to choose and validate a set of pseudo-labelled samples, starting from the un-annotated ones. In conjunction with the improvement obtained using an efficient query strategy, those methods significantly improve the model's accuracy. Detailed and quantitative results can be found in chapter 5.

1.4 Report Structure

The remaining part of this thesis is divided into five chapters, the structure and content of which is detailed below:

- Chapter 2 is dedicated to an introduction of the context and an overview of the related works about different strategies for the active learning.
- Chapter 3 contains the details about the possibilities and challenges addressed in this project.
- Chapter 4 presents the proposed method along with the rationale for each choice.
- Chapter 5 is about the experiments and quantitative tests done to validate the decision for the implemented system.
- Lastly, chapter 6 is dedicated to final considerations and future development of the work done for this master thesis.

Chapter 2

Context and Related Works

As pointed out in the introduction, one of the most relevant aspects in active learning, from a research perspective, are the *query strategies*. They define the methodologies by which the samples are chosen from the full set of unlabelled ones. In the following sections the context for this master thesis is defined and the existing strategies are introduced, underlining their strengths and weakness in the context of deep learning and neural network based classifiers.

2.1 Deep Learning

Deep learning is a subfield of *machine learning* that makes use of a set of algorithms loosely inspired by the structure and function of the brain, called *artificial neural networks.* An artificial neural network is a collection of units called *neurons* linked by connections that allow the transmission of the signals. The Figure 2.1 illustrates this kind of architectures.

In the figure, the connections are represented (using black lines) and, in different colours, the input neurons that correspond to the input signals (in red), the hidden (in yellow) and the output ones (in blue). In this case, there can be one or several hidden layers[1] defining the depth of the network.

2.1.1 Neural Networks

The artificial neural networks are made of different components, the following paragraphs will introduce them in details.

[1] A layer is a set of nodes belonging to the same level in the network.

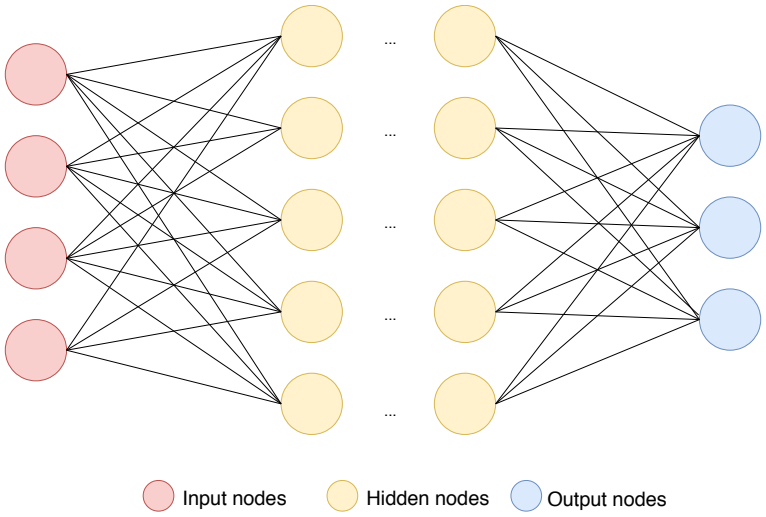


Figure 2.1: General Architecture for an Artificial Neural Network.

Neurons The *neurons* are the entities that correspond to the computational units. They receive the input signals (in the form of numerical vectors) and apply to them a transformation given by the *activation function* f that transforms them to the output signals. Input neurons have no predecessor but serves as input interface; similarly, the output layer neurons have no successor and serve as output interface.

Connections The *connections* are the links between different neurons and let the signal flow through them. Additionally, a weight w_{ij} is assigned to each connection, where i, j represent respectively the indexes of starting and ending neurons.

Propagation function: The *propagation function* is the function that computes the input for the next neurons starting from the output of the previous one. In this case, it is typically of the form $p(t) = \sum_i o_i(t) w_{ij}$

Learning rule: The *learning rule* corresponds to the algorithm used to modify the parameters of the neural network, in order to make the classifier learn patterns from the input data during the *training phase*which usually is the stage in which the network is taught, modifying the weights and thresholds.

Artificial neural networks are widely used to build classification systems for *Natural Language Processing* and *Computer Vision.* The project of this thesis is about this last topic. Over the years deep neural networks have outperformed previous state-of-the-art machine learning models in several fields and particularly in Computer Vision. In

this case, the most used variants are the *Convolutional Neural Networks* that will be introduced in the section below.

2.1.2 Convolutional Neural Networks

Not unlike traditional neural networks also *Convolutional Neural Networks* consist of an input, an output and multiple hidden layers. The hidden layers in a CNN[2] usually are of four different types, depending on the kind of operation they perform. The following paragraphs will introduce the operation performed by the different layers. For more details, the reader can refer to the Appendix A.

Convolutional Layers: The primary goal of this kind of layers, in the context of Computer Vision, is to extract features from the input image by applying the *convolution* operation over the input signals. In the case of CNNs, this type of layers consists of a set of learnable filters, each of which is a small window (in width and height) that extends through the full depth of the input volume. During the *forward pass* [3], each filter is convolved across the width, height and depth of the input volume and compute dot products between the entries of the filter and the input at any position.

Pooling Layers: This type of layers are often introduced between successive convolutional layers. The pooling layers operate independently on every depth slice of the input and resize it spatially. In the convolutional case, this layer can perform different operations using a sliding window of variable shape (usually squared). Common operations are $max(\cdot)$ or $avg(\cdot)$ that usually reduce the dimensions of the input.

Dense Layers: This kind of layers (often named also fully-connected layers) have the classic pattern that can be found also on regular neural networks. The neurons have full connections to all activations in the previous layer. This type of layer is usually the last one in a CNN. In the active learning strategy discussed in Algorithm algorithm 2, the output of this layer is used to define the input space and so, to compute distances between samples.

The categories above are the most used types of layers in the case of CNNs. This class of networks are used, instead of the classic ones, because of the scaling problem that CNNs resolve, in fact, using only the dense connection scheme, with the large size of the inputs (considering the resolution of the images), the number of parameters to handle can make the computational cost intractable.

[2] Acronyms standing for *Convolutional Neural Network*.

[3] The stage in which an image is passed through the network.

2.2 Learner's committee

One of the earliest strategies to choose most informative samples from a given pool makes use of a committee of voters and exploit their output to deduce that information. In this context, the voters are represented by a group of classifiers that give a confidence information for the input classification. The family of strategies called *Query by Committee*[21] makes use of the principle of *maximal disagreement* to choose the next sample to be labelled. The idea behind this technique is that the degree of disagreement among a committee of voters can serve as an estimate of the informativeness value. It is necessary to point out that more than one learner is required to implement this strategy because it checks the degree of disagreement using multiple probability scores. Moreover, the members of the committee should be trained, at each iteration, with all the available annotated samples.

The algorithm proceeds iteratively by selecting, each time, a single sample that maximizes the disagreement function, that can be defined in several ways. One of the most used function makes use of the *KL-Divergence* [4] and choose the next sample to label x^* as:

$$x^* = \underset{x}{\operatorname{argmax}} \left(-\frac{1}{|C|} \sum_{c=1}^{C} D(P_{\theta^{(c)}} \parallel P_C) \right)$$

Where $\theta^{(c)}$ represent one of the models belonging to the committee, C represents the complete set of voters and D is the function associated to the KL-Divergence. Under these circumstances, the most informative sample is the one that maximizes the average divergence between any committee member and all the others.

The committee of classifiers can be created in different ways depending on their type and on the resources available. In the context of neural networks, creating several classifiers could be unfeasible due to their parallel training. For this reason, the work by Ducoffe and Precioso [4] try to bypass the problem using a technique called *batchwise dropout*. This approach proposes to create the committee of *partial CNNs* adopting a version of the dropout[5] that, using a unique bernoulli mask[9], discards neurons for each sample in the batch. This technique allows a great reduction of the computational cost with the positive side-effect of having a committee whose members share the same architecture as the full network. Once having obtained the members of the committee,it

[4] *Kullback-Leibler divergence* is a measure used to estimate the difference between two probability distributions over the same variable.

[5] The dropout is a regularization technique for reducing overfitting in neural networks. The term refers to the action of *switching off* neurons in the network.

is possible to use the classical disagreement functions to choose one or more samples for the manual annotation. In the context of the cited work, each iteration chooses a batch of samples because of the requirements of neural network classifiers. The batched version is a straightforward implementation of the classic methods: choosing the first n samples in the ranking, obtained ordering them according to the disagreement score.

2.3 Expected model change

Considering that the goal is to select the samples that contain as much information as possible, some works focus on the estimation of the changes in the parameters of the model for each sample, selecting for annotation the ones that maximize this change.

From the previous principle starts the idea of Freytag, Rodner and Denzler [5] that uses the marginalized probability over all possible inputs and over its (yet unknown) labels to measure the expected change of the model. Unfortunately, this idea has a *myopic* approach, being able to select one instance at a time.

Another approach is the one named *expected gradient length* (EGL) that uses the norm of the training gradient to choose the next sample to query x^*. In this case, defining l_θ as the *loss function*[6] associated to the model and T as the previously annotated training set, the selected instance satisfies the following equation:

$$x^* = \underset{x}{\operatorname{argmax}} \left(\sum_{l=1}^{L} P_\theta(y_i|x) \parallel \nabla l_\theta(T \cup \langle x, y_i \rangle) \parallel \right)$$

Where, not knowing the correct label y, the expectation over all the possible ones is considered. Moreover, the norm of the training gradient can be approximated as $\nabla l_\theta(T \cup \langle x, y_i \rangle) \approx \nabla l_\theta(\langle x, y_i \rangle)$ because the training instances are assumed to be independent and, at query time, the value $\nabla l_\theta(T) \approx 0$ since l_θ is supposed to converge at the previous round of training. Similarly, the work of Cai et al. [2] extends this idea selecting a batch of samples at each iteration. Their work aims to achieve a faster convergence to the final expected model by selecting the points that significantly changes the current parameters. They exploit the *Stochastic Gradient Descent*[7] update rule, approximating the model change using the gradient of the loss function computed at the candidate instance. Having multiple instances to annotate at each iteration could be

[6] A *loss function* is a function defined for an optimization problem that is used to approximate the cost associated with a given event.

[7] *Stochastic Gradient Descent* is an approximated algorithm derived from gradient descent method. This iterative approach minimizes an objective function expressed as a sum of differentiable functions.

valuable, not only for the statistical relevance of the batch using neural networks, but also to parallelize the annotation (if multiple oracles available).

2.4 Expected Error Reduction

Following a similar principle of the previous class of algorithms, other active learning approaches target the reduction of the generalization error[8]. The idea behind this set of algorithms is to estimate the error for each label assigned to each unlabelled sample. In this case, the instance that minimizes the expected future error (also called *risk*) is chosen. To achieve this result a loss function can be defined and minimized. Considering, for example, the *logistic loss* is possible to formalize the problem as follows:

$$x^*_{\log} = \underset{x}{\operatorname{argmin}}\, P_\theta(y_i|x) \left(-\sum_{u=1}^{U}\sum_{j}^{L} P_{\theta^{+\langle x,y_i\rangle}}(y_j|x^{(u)}) \log P_{\theta^{+\langle x,y_i\rangle}}(y_j|x^{(u)}) \right)$$

Where $\theta^{+\langle x,y_i\rangle}$ refers to the new model after being re-trained with the training tuple $\langle x, y_i\rangle$ added to the set of labelled samples, U is the set of unlabelled samples and L the set of possible labels. It should be mentioned that, the true label is not known for each query instance, so it is approximated using expectation over all possible labels under the current model θ.

This family of algorithms have been widely explored starting from the work of Roy and McCallum [16] because of the twofold advantage of being both near-optimal in the choice of the samples and not dependent on the model class. Unfortunately, the computational resources required for this kind of approaches makes them unfeasible in most of the cases. More specifically, taking into account the complexity, these algorithms require not only the estimation of the expected future error for each sample in the unlabelled set but also the incremental retraining of the model for each possible label assignment.

2.5 Confidence based techniques

Machine learning models, when predicting the label associated to an input instance, are able to compute a posterior probability for a given sample and each possible label. Another family of methods that has been widely applied in the literature, exploits this confidence measure. The original idea derives directly from the work of Lewis and

[8]The generalization error is the measure that aims to estimate how an algorithm generalizes on real-world data. Due to the fact that a learning system is learned on a finite set of samples, they can be subject to sampling error and the accuracy on unseen data can significantly differ with respect to the one measured on training data.

Catlett [13] that implemented a simple but effective algorithm. The following works have defined different approaches to exploit this information and can be categorized as follows:

- *Least confident* [3]: rank each unlabelled instance using its highest posterior probability and select the least confident example $u \in \mathscr{U}$ (unlabelled set) to be annotated. Formalizing the method and defining y^+ as the label with the highest posterior probability under the model θ, the sample to annotate x^* is chosen as:

$$y^+ = \underset{y}{\operatorname{argmax}}\, P_\theta(y|x)$$

$$x^* = \underset{x}{\operatorname{argmax}} \left(1 - P_\theta(y^+|x)\right) \tag{2.1}$$

- *Smallest-margin* [17]: was introduced to take into account not only the information about the most probable label but also the other ones. It uses, as score, the difference between the most probable and second most probable label. Defining y_1^+ and y_2^+ as the first and second most probable class labels, under the model θ, it is possible to formalize the choice of x^* as:

$$x^* = \underset{x}{\operatorname{argmin}}\, P_\theta(y_1^+|x) - P_\theta(y_2^+|x) \tag{2.2}$$

- *Entropy* [1]: is the most general uncertainty sampling strategy and is defined as:

$$x^* = \underset{x}{\operatorname{argmax}} \left(- \sum_{i \in \mathscr{Y}} P_\theta(y_i|x) \log P_\theta(y_i|x) \right) \tag{2.3}$$

  Where $\mathscr{Y}$ represents the set of all possible class labels and $P_\theta(y_i|x)$ is the posterior probability given by the model, associated to the label i given the sample x. It is worth noting that the entropy-based approaches take all class label probabilities into consideration such that an interesting sample for the learning is associated with an high entropy value.

This family of approaches works well in a lot of cases but it is worth doing some remarks about:

- The value given by the classifier for the confidence is not always accurate for the active learning purposes.

- Most of the applications for this method select only one sample for each iteration. Creating a batched version of this kind of algorithm is not trivial. More details about this problem are given in section 3.1.
- This kind of approach tends to concentrate the choice to the samples that rely on the input space already explored by the labelled samples, choosing the most controversial one. At the end of method's iterations, in most of the cases, the input space has not been completely explored. This topic is better detailed in section 3.2.

The main drawback with respect to this method concerns the reliability of the confidence score given by modern neural network classifiers. In this case, due to their architecture, is possible to obtain very high confidence for misclassified samples, especially when the input space is not completely explored.

2.6 Active Learning and Neural Networks

In the last years, because of the success of these classifiers, a lot of works have been proposed regarding active learning techniques applied to neural networks.

Deep-Bayesian Active Learning

A considerable work is the one of Gal, Islam and Ghahramani [7] where they focus on high dimensional image data and a specific version of CNNs introduced for the first time by Gal and Ghahramani themselves [6] called *Bayesian Convolutional Neural Networks*. Using this type of classifiers, the uncertainty measure is more reliable and they obtain results that outperform the one obtained using the confidence score of standard CNNs.

Unfortunately, this technique has not been further explored because it does not use standard CNNs. Considering the type of classifiers used for this project and the constraint for which, in an active learning system, the used classifier should be as close as possible to the one that will be used to perform the final task, further development of this techniques will be unproductive for the project purposes.

Cost-Effective CNNs

The work of Wang et al. [24] proposes to apply standard uncertainty techniques to deep neural networks using the confidence score given by the CNN. The proposed method makes also use of unlabelled data using a *pseudo-labelling* technique embedded in the

algorithm. This sub-procedure can be classified as belonging to *semi-supervised* approaches because of the use of samples without additional knowledge in conjunction with already labelled instances. The complete method is illustrated in algorithm 1.

Algorithm 1: Cost-effective Active Learning procedure

```
function Cost-Effective-AL;
  Input : Unlabelled samples set 𝒰,
          Labelled samples set ℒ,
          High-confidence threshold t,
          Decay rate d,
          The number of maximum iterations M_i,
          The quota of manually labelable samples at each iteration q
  Output: parameter of the neural network W
          Updated labelled set ℒ
          Pseudo-labelled set ℋ
Initialization of the network using ℒ
Initialization of ℋ = ∅
do
    𝒮 = Obtain q samples belonging to 𝒰 ranking it using one of the confidence-based
      formulas (2.1, 2.2 or 2.3)
    Update ℒ asking the oracle for the annotation of samples in 𝒮
    ℋ = Obtain high confidence samples (classification confidence greater than t)
      belonging to 𝒰 along with their predicted label.
    Update W training the model using both the updated ℒ and the ℋ.
    Update t = t − d · i, where i is the index of the current iteration.
while not reach the maximum number of iterations M_i;
Return W, ℒ, ℋ
```

As can be seen from the pseudo-code, they propose a batched version of the standard uncertainty sampling techniques, ranking the samples according to the confidence score (obtained using one previously proposed methods) and selecting a batch of instances to label in agreement with their rank. In this case, they also apply the *pseudo-labelling* technique for the samples associated with an high-confidence score given by the classifier. This method, defining a minimum threshold t for the confidence and its decay rate d, allows the exploitation of not-annotated samples as part of the training set. The parameters are chosen in order to perform well with the obtained range of scores at different iterations.

It is worth noting that, using this method must be admitted an intrinsic error on the training data. For this reason, in the pseudo-code, the definition of $\mathscr{H}$ as a separate set with respect to $\mathscr{L}$ is done to allow different operations. In this context, in order to avoid error propagation, the pseudo-labelled samples are used during the current iteration and their label is then removed for the next one, treating them as part of $\mathscr{U}$. It is interesting that, despite the implicit error introduced using this procedure (maintaining it under a reasonable value), the model benefits from those annotations and increases its accuracy at each iteration.

Core-Set Selection

In the domain of image classification using deep neural networks, the recent technique proposed by Sener and Savarese [20] has demonstrated the effectiveness of input space exploration. The paper proposes the selection of a *core-set* of the input data such that, a model trained on this subset performs as closely as possible to the model trained on the entire dataset. The authors define a *robust*[9] version of the *greedy k-centers* algorithm and achieved state-of-the-art results in this domain. It is worth mentioning that, in order to define the space exploration, the authors defined as distance between samples the l_2 distance between the activations of the final fully-connected layer for each image. For more details about convolutional networks, the reader could refer to Appendix A. The *greedy k-centers* algorithm performs the batch selection illustrated in algorithm 2.

The proposed method, at each iteration, choose one sample that maximizes the distance with its nearest labelled one. The distances are updated at each iteration such that the samples already selected in previous iterations are considered as labelled. To perform the active learning, the function proposed is iterated multiple times t to choose multiple batches, adding, at the end of the procedure, $t \cdot s$ annotated samples to the training set. It is worth noting that, at each iteration of the procedure above, the model is retrained using the updated set $\mathscr{L}$.

[9] In this case *robust* refers to the fact that the method is not significantly affected by the problem of outliers.

Algorithm 2: Method used to choose the batch of samples using *greedy k-centers* method.

function Greedy-k-center;
Input : Unlabelled samples set $\mathscr{U}$,
Labelled samples set $\mathscr{L}$,
The quota of manually labelable samples at each iteration q
Output: a batch of labelable samples b
Initialize $s = \mathscr{L}$
Initialize $b = \emptyset$
do
    $i = \underset{i \in \mathscr{U}}{\text{argmax}}\ min_{j \in s} \Delta(x_i, x_j)$
    $b = b \cup i$
    $s = s \cup i$
while $|b| < q$;
return s

Chapter 3

Addressed challenges

3.1 Batch Acquisition

In most of the algorithm introduced in the previous chapter, the samples to be annotated are selected one for each iteration, using a *myopic approach*. In the context of deep neural networks instead, it is necessary to select, at each iteration, a batch of samples, mainly for the following reasons:

- A single point has no statistical relevance for the learning due to the batch training performed.
- Each selection iteration includes a training phase. It is infeasible to train as many models as the number of annotated instances because of the scale of the data.

Most of the algorithms proposed in chapter 2 are not easily adaptable for a batched version. The focus of this thesis, because of its context, is about batched active learning methods, in which the query strategy needs to choose a set of instances at each iteration. Moreover, choosing a batch of samples open the possibility of concurrent annotations, in case of multiple oracles available, reducing significantly the time required for the annotation phase.

3.2 Input space exploration

Because of the context of partially-labelled datasets, one of the most relevant challenges is the one related to the partial exploration of the input data space. In this case, is possible that the labelled samples are not representative of the full dataset. Under these

circumstances, the most effective techniques are the ones that allow the classifier to better cover the input space. In practice, to define the data exploration needs to be defined a distance measure between the samples of the dataset.

To better explain the issue is worth to visualize an example case. For the sake of simplicity, the following image represents data as 2D points, but the idea can be generalized for the multi-dimensional case, removing also the facilitation of linear classifier.

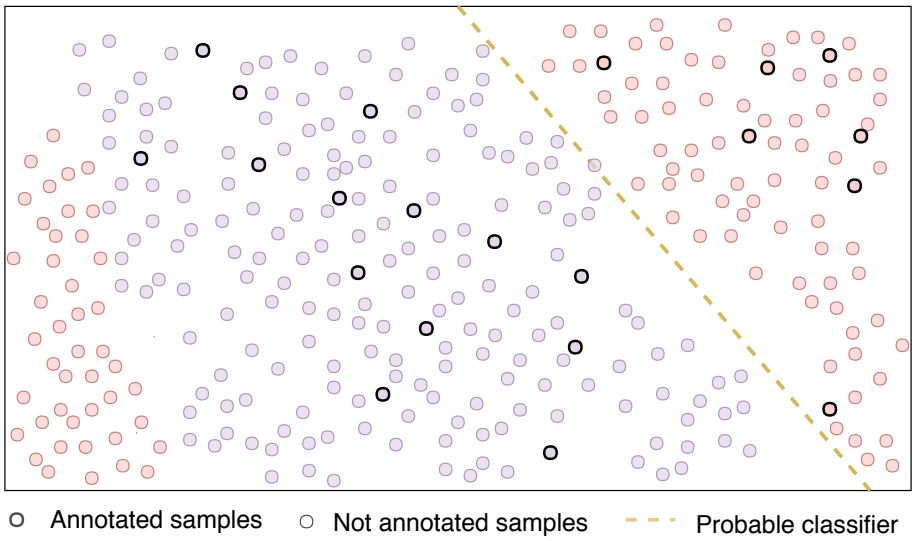


Figure 3.1: Exploration problem visualization

As visible in the chart, given that the annotated samples do not cover uniformly the input space, the classifier can produce a linear separation that performs poorly in not discovered spaces. Using an uncertainty-based technique as detailed in section 2.5 the next samples chosen will be the one located next to the borders of the current classifier. This type of techniques, in fact, are more suitable for the refinement of the classifier rather than the exploration of the space. Under these circumstances, the approach, in the first iterations, will not select samples in the unexplored zone and this will lead to a slow improvement of the model. Intuitively, in the simple case illustrated above, the classifier is unaware of the samples in red on the left-hand side of the image. Having at disposal only the annotated samples, the classifier will associate to the instances in the unexplored part of the space, the wrong label with an high confidence. In the paper *Learning Active Learning from Data*, Konyushkova and Sznitman [11] show that confidence-based methods become sub-optimal in case of partially explored input space.

Distance between images using CNNs Being in the domain of image classification and needing the definition of a distance measure between samples, is possible to take advantage of the idea proposed by Sener and Savarese [20] that applied the concept of

space exploration in the case of CNNs. The authors suggest the exploitation of the l_2 distance, not between two raw arrays representing the images but a transformed version of them, obtained forward-passing the input image through the network by the output of its last fully-connected layer. This choice is crucial because the output of the last fully-connected layer will define the input space that needs to be explored.

3.3 Decision boundaries refinement

Even though is necessary for an effective technique to explore the full input space, it is worth noting that very interesting results have been shown by confidence based techniques. Intuitively, this kind of approaches performs well in case of good coverage of the input space passing to the refinement of the decision boundaries, when it is worth looking for the toughest samples to classify. A different data distribution, with respect to the previous one in Figure 3.1 is shown in the image below:

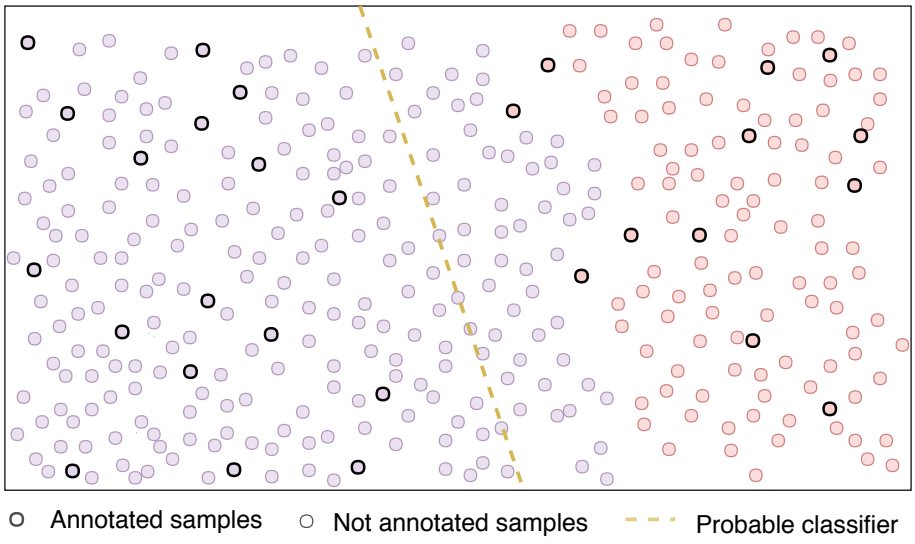


Figure 3.2: Refinement problem visualization

In this case, the input space is better covered by the labelled samples but, the classifier is not accurate due to the samples that are located near to the borders of the classifier. Under these circumstances, the uncertainty measure given by the network can be helpful to choose the most significative samples for the learning.

At this point, can be seen that a balance between the exploration of the input space and the refinement of the decision boundaries should provide a complete strategy for both the cases. In this master thesis it is proposed a method that combines both information in a unique score associated with each sample. The method is detailed in

section 4.1. It is necessary to remark that, the example above, is not realistic in the context of this thesis and is used only to intuitively illustrate the problem. Dealing with image classification, the type of data treated are of much higher dimension and the obtained classifier will not be linear.

3.4 Semi-Supervised Learning

In the context of partially annotated datasets, *semi-supervised learning* approaches include the necessary methods to exploit both the annotated and not annotated samples. In order to make any use of unlabelled data these approaches use at least one of the following assumptions:

- *Continuity assumption*: instances which are *close* one to each other are more likely to share a label.
- *Manifold assumption*: the data lie approximately on a manifold of much lower dimension than the input space.
- *Cluster assumption*: data tends to form discrete clusters, and points in the same cluster are more likely to share a label.

For this project is assumed that there is *continuity* between samples. This choice stems from the idea of having a valuable distance metric, based on the output of the last dense layer of the CNN classifier. Moreover, in case of using *confidence-based methods* (section 2.5), is possible to use this score to perform a pseudo-labelization as introduced by Wang et al. [24]. Also in this case, the two pieces of information are used jointly in order to reduce the possible error introduced in the set used to train the model. More details on the methods used for the implemented algorithms can be found in section 4.2.

3.5 Discussion

Being in the deep learning context and having identified the possible problems that can occur, the proposed idea is to create a method that can merge the two approaches. Intuitively, in the early steps, the methods should assure a proper coverage of the input space and, subsequently, the method should prefer the refinement of the classifier. Concerning instead semi-supervised techniques, they can be used to both improve model accuracy and refine the choice of the samples to manually label.

The main constraint, using semi-supervised techniques, is to obtain qualitative model and datasets. Choosing a *pseudo-labelling* procedure, it implicitly adds misclassified

samples in the training instances. For this reason, the goal is to create a procedure able to minimize them, perhaps at the cost of the number of automatically annotated samples . The complete procedure is explained in section 4.2.

Chapter 4

Implemented methods

To meet the requirements and better explore the possibilities offered by active learning algorithms, have been tested some methods that have been introduced in chapter 2 and a new one proposed for this master thesis. After the first phase of study, the problems related to *input space exploration* (section 3.2) were evaluated as prominent in this domain. Moreover, the great results obtained by *uncertainty-based techniques* suggest the integration of the two methods. The following section will introduce in details the approach used to merge distance and uncertainty information.

4.1 Balanced Uncertainty Exploration

The idea behind this method is to combine distance and confidence-based information to create a unique score able to adapt to the different conditions. At each unlabelled sample is associated a value obtained using the two scores. The first contribution d_i is defined as the distance between the sample itself and the nearest annotated one. The maximization of this value identifies samples that are located in parts of the input space that are not explored yet.

The uncertainty score instead, is retrieved forward-passing each unlabelled sample through the network. Doing so is retrieved the highest class probability for each sample $max(P_\theta(y|x))$ and computed u_i as follow:

$$u_i = 1 - max(P_\theta(y|x)) \tag{4.1}$$

Defined in this way, the best sample to choose for the labelization is the one that maximizes both the values d_i and u_i. The contributions have been merged using a weighted combination of them:

$$bue_i = \lambda\ u_i + (1-\lambda)d_i$$

Where bue_i is the score associated with the i^{th} sample and λ is the weighting parameter to which is possible to associate different functions. The primary idea is that λ should monotonically increase in order to give more weight to the confidence at each iteration. Concerning the implementation, naming s the index of the current iteration and q the number of total iterations, lambda is defined as:

$$\lambda = s/q$$

Such that, for the first iterations the main contribution is the one associated to the distance, while during the last, the uncertainty score gives a higher contribution. It is worth noting that, the distance score have been normalized (using min-max normalization) in order to fall in the interval $[0,1]$, while the confidence, being a probability measure, is retrieved from the network already in that range.

Intuitively, the idea behind the definition of this score is to start exploring the input space and then, increasingly, give a greater weight to the confidence-associated value choosing the toughest samples to refine the classifications. A related point to consider is that lambda can be defined with several kind of functions, for example it can profit from the information about the ratio of labelled samples in the entire dataset.

Distance computation: The proposed approach, as the *Greedy K-Centers* one, makes use of the distance computation. This measure, as introduced before, is computed between the features extracted from the final dense layer of the network. It is worth noting that, those features are the same used for the probability computation using the *softmax*[1] function. For this reason, both pieces of information are derived from a common source but treated in different ways. The output of the last fully-connected layer is the network representation of the input according to the demanded task and so an effective method to define the input space.

4.2 Pseudo-labelling procedure

Dealing with not completely labelled datasets and looking for a *semi-automatic* labelization procedure, active learning techniques have been used in conjunction with semi-supervised ones in order to improve the model performances and obtain more complete, annotated data collections. The used approach is detailed algorithm 3.

[1]The *softmax* is a function used to map a vector of real values into a vector of the same dimension where each entry contains the corresponding value in the original vector exponentially normalized in the range $(0,1)$.

Algorithm 3: Procedure illustrating the automatic soft-labelization.

```
function Automatic-soft-labeling;
  Input : Unlabeled samples set 𝒰,
          Labeled samples set 𝒟
          Minimum threshold for the uncertainty score t
𝒞 = set containing unlabeled samples satisfying the condition on the minimum
    threshold (t) with their associated label (given by the most probable one)
for i in 𝒟 do
    n = obtain the nearest sample to i in 𝒰
    l_d = get-label(𝒟, i)
    l_u = get-label(𝒞, n)
    if l_d is equal to l_u then
        assign label l_d to n
end
```

The procedure begins retrieving, for each unlabelled instance, its predicted label given by the most probable one. The prediction is obtained forward passing each sample into the network and needs to satisfy the condition concerning the minimum threshold for the confidence. Naming this set $\mathscr{C}$, for each labelled sample, the procedure continues retrieving the nearest unlabelled one (n in the pseudo-code). At this point, the function checks if the not annotated sample is present in $\mathscr{C}$ and if the predicted label matches the one of the near labelled sample. In the event that the condition on the threshold is not satisfied or the labels do not match, the pseudo-labelling is not performed for the sample under consideration.

This procedure is named automatic *soft* labelling because the assignment is not permanent, it is used to train the model at the current iteration and, for the subsequent ones, the samples are considered as not annotated. The labels automatically assigned are stored only at the end of the procedure, where their amount should be greater and the number of misclassifications lower. The choice to perform *soft* labelling is done to avoid that misclassified samples produce an error propagation that degrades the model performance. Referring to the introduction done in section 3.4 it is worth noting that, in this case, is used the *Continuity assumption*, selecting the samples using the distance with respect to the labelled one and validating the assignment of the label using the uncertainty measure. Combining the two pieces of information the number of errors is reduced obtaining more valuable annotated data.

4.3 Solution Analysis

The implemented procedure makes use of the proposed query strategy and the semi-supervised approaches detailed above. It distinguishes two training phases with respect to the hyperparameters used to train the model, respectively h and h_i for the complete and incremental train. In this case, those set of parameters contains the number of epochs[2], the learning rate and others.

Model training: Being the active learning an iterative procedure, is possible to proceed in two different ways to train the classifier. The first one is to re-initialize the model and proceeds with its training from scratch, while the second option is to perform an incremental training, keeping the previous model weights. The procedure uses both the approaches in different situations. The incremental training (*fine-tuning* the model) is used after the manual labelization, in order to make the network able to exploit the new annotated samples to refine the choice of pseudo-labelled instances. On the other hand, the complete training is used, at each iteration, when the full set of current annotated data is ready. The choice was guided by the fact that, doing an incremental training, in this case, could lead to the overfitting[3] of the network on images already available since the beginning of the active learning procedure.

The algorithm 4 illustrates the complete workflow applied using the selected active learning strategy.

Reduction of Automatic Labelization errors: As introduced before, the automatic labelization is implemented in a *soft* manner, such that those labels are considered only for the training of the model at the current iteration. In this case, is mandatory to consider a margin of error of the classifier. To reduce the error propagation and to reduce the influence of the previously auto-labelled samples, has been introduced a *fine-tuning* section using only human-labelled samples. This phase is useful both to exploit the lastly manually labelled samples and to reduce the cost with respect of a complete re-training of the classifier.

In the context of the test performed for the project, the datasets are completely annotated. For this reason, the tests have at disposal the true label and is simulated an ideal oracle that does not make any error. In real applications the oracle could incorrectly

[2] An *epoch* in the training phase of a neural network corresponds to a forward and backward pass over all the training examples.

[3] Overfitting, in machine learning, is a situation of a model when it becomes too specialized in the pattern found on training that and fails in the generalization on real-world data.

Algorithm 4: Pseudo-code for the implemented procedure.

function Active-Learning;
Input : Unlabeled samples set $\mathscr{U}$,
Labeled samples set $\mathscr{L}$,
Number of iterations n,
Number of annotation for each iteration b,
Training hyperparameters for complete train h,
Training hyperparameters for partial train h_i
Output: New set of model weights W,
Updated sets $\mathscr{L}$, $\mathscr{U}$, $\mathscr{A}$
Initialize the model and train it with previously annotated data (using parameters in h).
for *i in range* 0 ... n **do**
    $\mathscr{Q}$ = choose b samples to be annotated by the oracle using the chosen query strategy
    Update $\mathscr{L}$ and $\mathscr{U}$ using the new labels given by the oracle
    Train the model using parameters h_i and the updated $\mathscr{L}$
    $\mathscr{A}$ = soft-label automatically using algorithm 3
    Retrain the model using the parameters h and the updated sets $\mathscr{L}$ and $\mathscr{A}$
end
W = extract trained model weights
return W, $\mathscr{L}$, $\mathscr{U}$, $\mathscr{A}$

classify some instances. Is worth noting that, even for the datasets used during the test phase, the accuracy of a human classifier does not reach 100%.

4.4 Implementation tools

The methods introduced all along the thesis have been implemented using their theoretical definitions. The main programming language used is *python*. This language combined with all the available extension packages allow the prototyping of all kind of mathematical algorithms. For the project implementations have been used the following libraries:

- *Tensorflow*[23] is the principal library used to implement the classifier. It is an open-source project, initially created by *Google* that has rapidly become the most used extension to create deep learning models. In this case, it is used as backend for *Keras*[10], a higher-level library used to define the model architecture.
- *Numpy*[15] is a python package for scientific computing. It is used to define and easily manipulate the input signals, treating them as vectors. Moreover, this is the principal extension used for classical mathematical calculations.
- *Scikit-learn*[18] is one of the most widespread machine learning libraries in python. In the context of the project, it is used both to compute the accuracy of the model and, particularly, plays a key role in the distance computation, allowing its parallelization and reducing the time required.
- *Matplotlib*[14] and *Seaborn*[19] are two python libraries dedicated to the visualization. They support the creation of charts used to visualize the trend shown in the test section.

Being one of the most customizable parts of the project, the dataset management instead, has been done using standard parsing libraries and own-defined classes that allow the retrieval and storage of annotations.

4.5 Computational resources

Being an iterative procedure which involves multiple training phases (expensive in deep learning domain), the implemented procedure requires powerful computing systems. The test presented in the next chapter have been conducted using the following hardware configuration:

- *CPUs*: Dual Intel®Xeon®E5-2690 v4[4], having 28 core each.
- *GPU*: Single NVIDIA®Tesla®V100[5] in the 16GB configuration.
- RAM: 256GB

Those machines are dedicated to deep learning and used for the team purposes. It is worth noting that, as introduced before, most of the computation is needed for the training of the model, while, the remaining procedures do not introduce a significant overhead.

[4]CPU reference page: https://ark.intel.com/products/91770/Intel-Xeon-Processor-E5-2690-v4-35M-Cache-2_60-GHz

[5]GPU reference page: https://www.nvidia.com/en-us/data-center/tesla-v100

Chapter 5

Experimental Results

To test the effectiveness of the active learning methods, have been used some fully-labelled datasets. In this case, for the training set has been chosen an initial part of labelled samples and all the other ones have been considered as not annotated. The methods implemented have been tested using the same initially annotated instances in order to create tests as objective as possible.

5.1 Datasets description

To test the methods in different contexts have been chosen two standard datasets for image classification that differs for the number of classes. The next paragraphs will illustrate their structure.

Cifar-10 The CIFAR-10 dataset [12] consists of 60000 32x32 colour images belonging to 10 different classes. The classes are well balanced, in fact, there are 6000 images per class. The dataset is divided into training and test set with respectively 50000 and 10000 images. The classification task consists in the identification of the class for each image choosing between: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The classes are mutually exclusive so there is no overlap between them. The image below shows some examples of images belonging to the dataset.

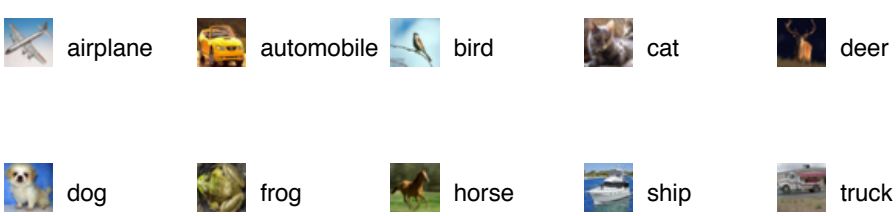


Figure 5.1: Examples of Cifar-10 images and their associated label

Cifar-100 The CIFAR-100 dataset consists of 60000 32x32 colour images which, in this case, belong to one of the 100 different classes. Also for this dataset, the samples are exactly balanced between classes, in fact, there are 600 images per class (500 in the training set and 100 in the test set). As in the previous case, the collection is split in two part of 50000 (training set) and 10000 images (test set). Each sample of the dataset is labelled with a superclass and a class. Each class belong to a superclass and a superclass contains multiple classes. For example, a superclass is of the type *aquatic mammals* and contains as classes *beaver, dolphin, otter, seal, whale*.

5.2 Evaluation methods

Dealing with fully-annotated datasets and willing to evaluate an active learning procedure, they have been split using the following rules:

- The test set has been maintained annotated and used only to measure the accuracy of the model at each iteration.
- The training set, considering N as its full size and n initially labelled samples, has been split in two parts, respectively of size $|\mathscr{L}| = n$ (annotated) and $|\mathscr{U}| = N - n$ (not annotated) using a uniform sampling procedure to select the initial set.

Under these circumstances, the only part of the dataset concerned by active learning is the training set. The model used to evaluate the methods refers to the classical VGG16 architecture that is detailed in subsection A.0.4. The parameters used to train the model are the same for all the test performed and are the following ones:

- The complete training is performed in $e = 300$ epochs, while the incremental one, when performed, is done for $e_i = 50$ epochs.
- The learning rate used is $\eta = 10^{-4}$ for both the fine-tuning phase and the complete train.
- The percentage of used dropout is $d = 25\%$.
- In the case of pseudo-labeling, the minimum threshold for the confidence score has been fixed to $t = 0.99$.

In order to compare the effectiveness of the different sampling methods, the following ones have been implemented and tested:

- *Random Sampling*, usually considered as the *baseline*, consists in choosing the samples to label in a randomized manner among the ones in the unlabelled set.

- *Uncertainty Sampling*, implements a batched version of the *least confident* case (Equation 2.1), detailed in section 2.5.
- *Greedy K-Centers Sampling* that uses the method introduced in algorithm 2.
- *Balanced Uncertainty Exploration Sampling*, the proposed approach, illustrated in section 4.1.

For each sampling method, at each iteration has been evaluated the test set accuracy of the current trained model. The accuracy is computed as the ratio between well-classified samples and the total number of classifications (the number of samples in the test set). In both the case of Cifar-10 and Cifar-100 the conditions of the test are:

- The number of initially labeled samples is 5000, corresponding to the 10% of the training set.
- The oracle is asked to annotate, at each iteration, 5000 samples proposed by the sampling strategy.
- The number of iterations is fixed to 4.

Apart from the initially labelled instances, the remaining ones are considered as not annotated for all the sampling methods. For the conditions given above, the ratio of labelled samples passes from 0.1 to 0.5. This choice was done to have, in the last iteration, one-half of unlabelled samples, both to make tests in realistic conditions and to allow the exploitation of semi-supervised techniques. Moreover, all the strategies have been tested multiple times, using three different initializations of the initial labelled set, in order to make as objective as possible the evaluation and not depending on the initial choice.

5.3 Cifar-10 results

The following charts compare the methods and their accuracy evolution obtained on the test set for Cifar-10:

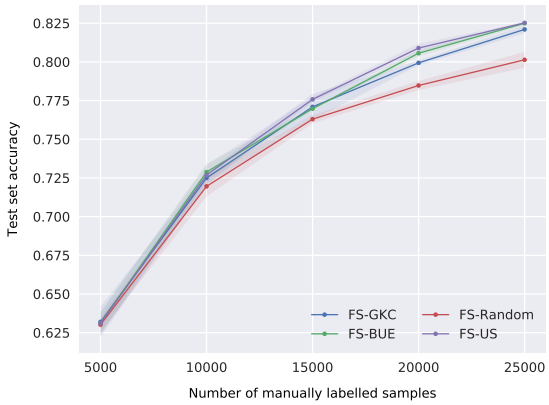


Figure 5.2: Fully-Supervised test

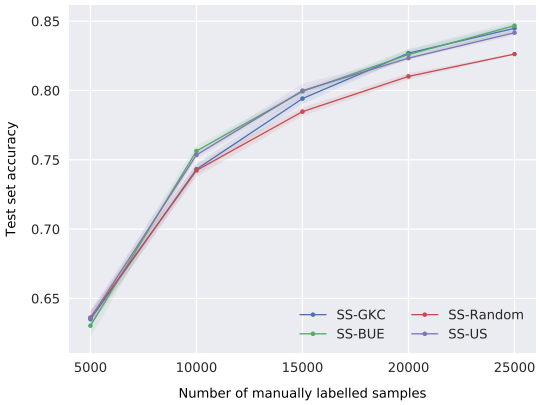


Figure 5.3: Semi-Supervised test

Cifar-10: test accuracy evolution and comparison of the implemented methods.

The images report the results obtained using only active learning techniques (Figure 5.2) or using active and semi-supervised learning (Figure 5.3). As can be seen in the figures, *Greedy K-Centers* (GKC), *Balanced Uncertainty Exploration* (BUE) and *Uncertainty Sampling* (US) methods obtain similar results that outperform the one obtained using the *random* sampling. Interestingly, both in the supervised and semi-supervised case, the straightforward implementation of Uncertainty Sampling performs well. The accuracy gap between the random strategy and the other ones increases at each iteration, showing that the other strategies allow a faster learning of the model.

5.4 Cifar-100 results

As in the previous case, the following charts show the accuracy trends in the supervised (Figure 5.4) and semi-supervised case (Figure 5.5) on Cifar-100:

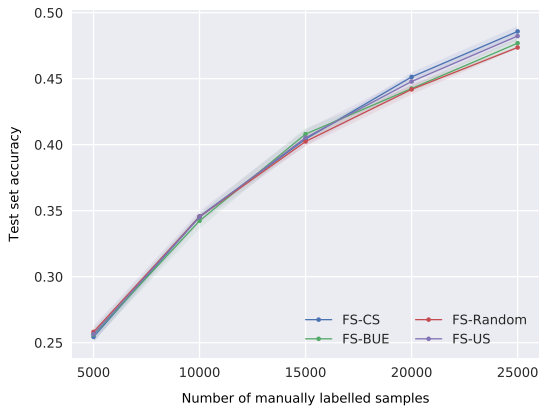


Figure 5.4: Fully-Supervised test

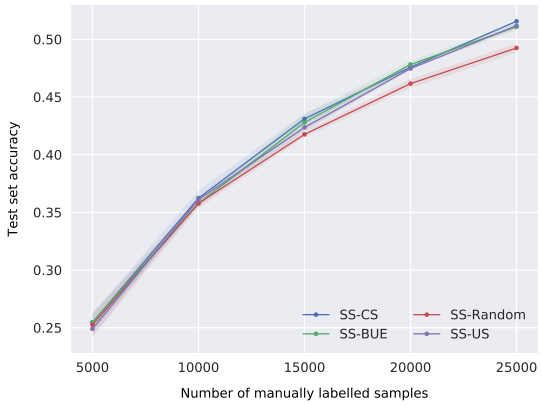


Figure 5.5: Semi-Supervised test

Cifar-100: test accuracy evolution and comparison of the implemented methods.

In this case, the gap between the methods is not as remarkable as in the Cifar-10 case. Dealing with more classes, the distance and uncertainty scores seem to do not represent well the informativeness of each sample. This consideration is validated by the semi-supervised test, where, having at disposal more samples to learn from, the gap between the baseline and active learning strategies increases, preferring by a small margin the *GKC* strategy.

5.5 Complete contribution

To illustrate the complete contribution of this master thesis, Figure 5.7 and Figure 5.6 illustrate the accuracy evolution of the different methods, in the fully and semi-supervised cases, on both Cifar-10 and Cifar-100 datasets.

The graphs show the improvements obtained using the procedure detailed in algorithm 3. Concerning Cifar-10, the accuracy on the test set outperforms by a consistent margin the one obtained using labelled samples only. Regarding the proposed active learning strategy, it performs similarly with respect to the other methods, specially in the semi-supervised cases. The future improvement could touch upon the score definition. A possible idea is to determine the weighting parameter λ (defined in section 4.1) as a direct function of annotated samples percentage in the dataset.

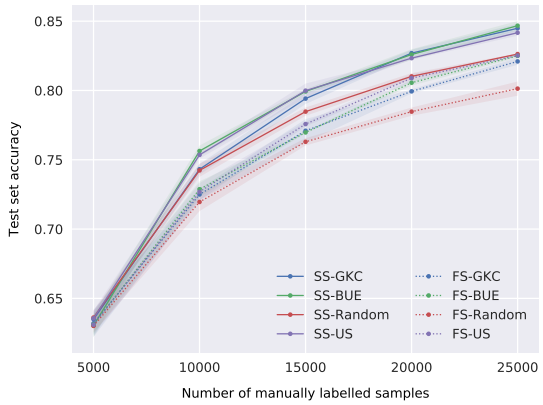


Figure 5.6: Cifar-10 complete test

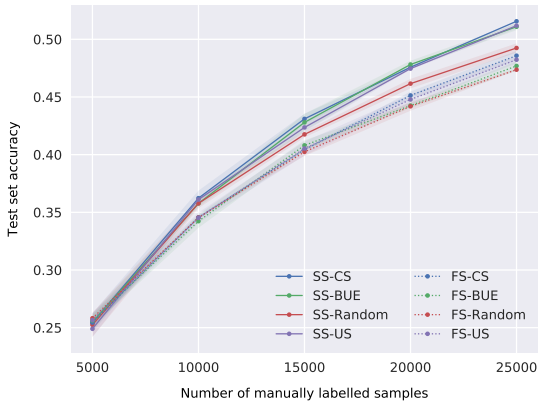


Figure 5.7: Cifar-100 complete test

5.6 Automatic Error Reduction

Since the goal is to obtain a semi-automatically annotated and qualitative datasets, some tests have been done in order to report the error introduced by the pseudo-labelling procedure. For all the sampling approaches tested in this section, the used semi-supervised technique is the one introduced in section 4.2. Moreover, as the tests in the previous section, they have been performed, for each method, with three different initializations of labelled instances. It is necessary to point out that, it was possible to perform these tests because the benchmark datasets are fully-annotated.

Cifar-10 The Figure 5.8 shows the evolution of the average error for each implemented method. The reported results show a decreasing monotonic trend for all the strategies adopted, except for the random sampling technique. Being related both to the confidence and distance metrics, the active sampling methods consistently improve the results obtained using the random technique. For this method, in fact, the graph shows that the average error decreases at the first iteration but start to increase in the following ones. Under these circumstances, the active learning methods improve significantly the performances.

Cifar-100 The Figure 5.9 report the mean error for each implemented method. While quantitatively the average error is higher than the one obtained in *Cifar-10*, its evolution reflects the same trend for all the techniques. This test validates, even more, the hypothesis according to which, the proposed semi-supervised approach takes advantage not only of the increasing accuracy but also of the distribution of the already labelled samples. In the random approach, having been aimlessly chosen, the samples suffer the

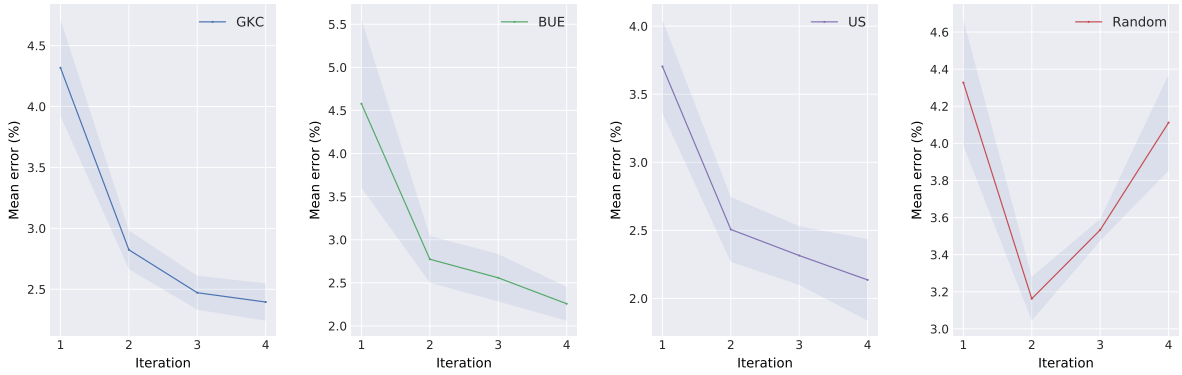


Figure 5.8: Cifar-10: From left to right, the pseudo-labelling average error respectively using *Greedy K-Centers*, *Balanced Uncertainty Exploration*, *Uncertainty* and *Random Sampling*.

exploitation of distance and confidence based information.

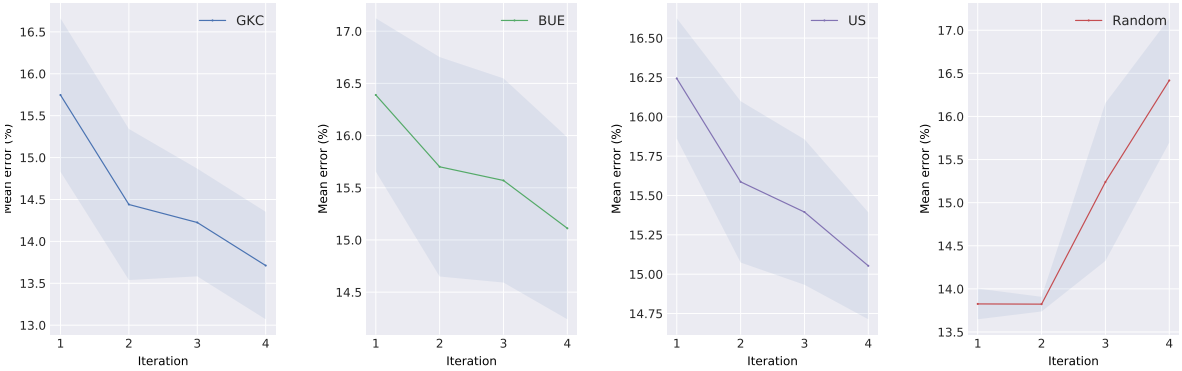
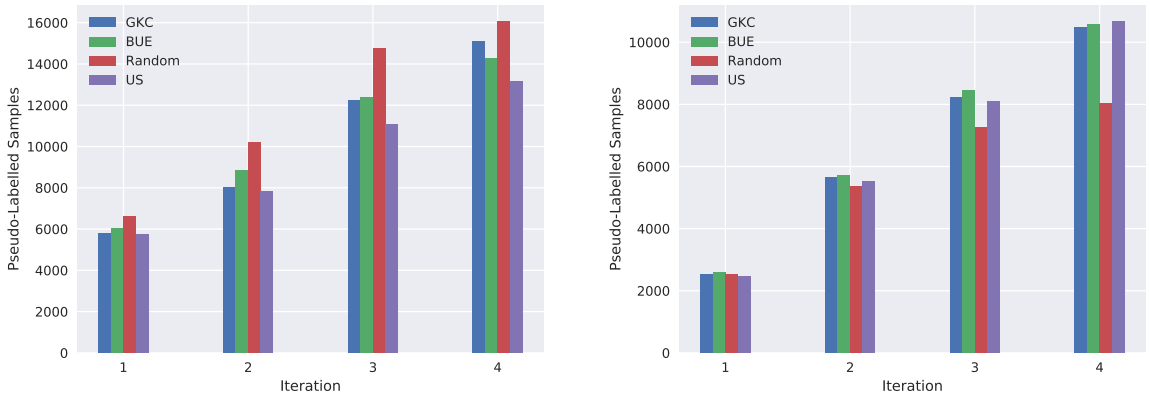


Figure 5.9: Cifar-100: From left to right, the pseudo-labelling average error respectively using *Greedy K-Centers*, *Balanced Uncertainty Exploration*, *Uncertainty* and *Random Sampling*.

It is possible to remark that the quantitative results depend on the initially labelled set of samples and on the learning performed by the network, in fact, due to the random dropout introduced during this phase, the features extracted and the samples representation used for the distance computation can slightly differ, despite the convergence of the model. For this reason, more than the quantitative results is more interesting to remark the trends obtained.



Cifar-10 Cifar-100

Figure 5.10: Number of automatically labelled samples per iteration for each tested technique.

At this point, to better understand the proposed procedure is possible to report the number of automatically labelled samples. The histograms in Figure 5.10 reports the averaged results for each sampling method at each iteration. As can be deducted, the number of soft-labelled samples grows at each iteration but,x the results differ on the two datasets. Specifically, the random technique labels more samples than the others for Cifar-10 as opposed to Cifar-100. This trend can be due to the representation learned by the model using this technique. Also in the case of not uniform input space exploration, having at disposal a lower-dimension vector as representation of the image, on Cifar-10, this strategy will more likely select a sample that can be validated by the model confidence, while in a 100-dimensional space, this is less probable. The other methods have similar performances on both the datasets, with the respective difference in terms of quantitative numbers.

Finally, merging the consideration about the percentage of mislabelling and the number of annotations assigned automatically, the random sampling is clearly the worst technique using the proposed approach. The other methods under test have the same performance with a slight preference towards the *Greedy K-Centers* algorithm.

Chapter 6

Conclusion

In this thesis have been introduced the main active learning approaches in the image classification domain. The human effort required and the deep learning models need of large volume of annotated data has brought the focus to active and semi-supervised techniques for the best exploitation of not annotated data. The new proposed procedure tries to merge a new active learning strategy with a pseudo-labelization technique to both minimize the human efforts and maximize the knowledge extracted from not annotated data. The main goal was to obtain a unified procedure that is able to semi-automatically label a given dataset.

The tests have shown that the baseline provided by the random sampling method has been significantly improved by the proposed methods both in terms of model accuracy and errors performing pseudo-labelization. The results obtained illustrate the importance of using active learning strategy for image classification tasks and the benefit provided by semi-supervised learning both to reduce human effort during labelization and to, even more, increase the deep learning model performance.

The possible use cases of active learning methods are varied and comprehend:

- The standard annotation of unlabelled datasets, taken into account during the tests.
- The use of active learning techniques in order to speed up the adaptation of a previously trained model in a real-world context.
- Change the intended purpose of an already existing dataset, in order to perform a different type of classification with respect to the one for which it has been conceived.

And many others that exist or may be required in the future. This technique, in fact, aims to exploit or predict the situation under which the model is not well performing

and, asking for human intervention could improve itself.

Lastly, it is possible to conclude these considerations, taking into account the requirement to implement the proposed approach using already existing deep learning models and datasets. In this case, relying only on standard methods such as the training of the model, the prediction and the extraction of the output of an intermediary layer should be not expensive to add a higher level system that, interacting both with the model and with the human annotator bring the benefit illustrated all along this thesis.

Future works

Considering the rapid growth of deep learning application and the interest in this subject of many research centres and companies, this project can be further developed in order to be tested in more contexts. Moreover, concerning the results obtained, the proposed method can be improved making the score adaptable to different circumstances. Specifically, we believe that the combination of multiple knowledge sources to define the contribution of each sample can be the right way to enhance the existing techniques.

More in details, further investigation can be done for the weighting parameter as introduced in section 4.1. This is a key component of the method and a possible idea is make it dependent from an adaptive function that relies on the ratio of labelled samples or on their distribution, in order to adjust, at each iteration, the correct balance between the exploration of the space and the refinement of the classifier.

Furthermore, as possible research paths, two interesting possibilities have been identified:

- To improve data-efficiency, active learning techniques can be used to select a small representative set of data and use them, in conjunction with GANs[1] to create many similar samples for the learning of the network.
- Explore *reinforcement learning*[2] techniques to adapt the active learning strategies, automatically, in several conditions. This idea is a higher level of abstraction that can be identified as *meta-active learning*.

[1] GANs, *Generative Adversarial Networks*, are deep neural net architectures that can learn to mimic any distribution of data.

[2] The *reinforcement learning* is an area of machine learning where virtual agents are learned to take the best decision in order to maximize a previously defined reward function.

It goes without saying that these are just two examples, but there exist more research opportunities to develop the work done for this project.

Finally, this thesis has not analyzed the interaction between the human and the algorithm. During the internship has been prototyped an interface, using *web-based* technologies, in order to make an annotator able to provide the label for the requested images. In this context, the front-end interface interacts with the algorithm to manage the different phases of the procedure, opening the dataset, initializing the model and providing the possibility to contribute to the annotation. The system has been conceived with the purpose of an easy integration with an existing deep learning model, acting as a higher level interface and exploiting the standard functions provided. In the future, this prototype can be further developed to create a collaborative platform, reducing the annotation time with multiple oracles available and maximizing the learning of the model with the techniques proposed in this thesis.

Bibliography

[1] Shlomo Argamon-Engelson and Ido Dagan. "Committee-Based Sample Selection for Probabilistic Classifiers". In: *CoRR* abs/1106.0220 (2011). arXiv: 1106.0220. URL: http://arxiv.org/abs/1106.0220.

[2] Wenbin Cai, Muhan Zhang, and Ya Zhang. "Batch Mode Active Learning for Regression With Expected Model Change". In: 28 (Apr. 2016), pp. 1–14.

[3] Aron Culotta and Andrew McCallum. "Reducing Labeling Effort for Structured Prediction Tasks". In: *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*. AAAI'05. Pittsburgh, Pennsylvania: AAAI Press, 2005, pp. 746–751. ISBN: 1-57735-236-x. URL: http://dl.acm.org/citation.cfm?id=1619410.1619452.

[4] M. Ducoffe and F. Precioso. "QBDC: Query by dropout committee for training deep supervised architecture". In: *ArXiv e-prints* (Nov. 2015). arXiv: 1511.06412 [cs.LG].

[5] Alexander Freytag, Erik Rodner, and Joachim Denzler. "Selecting Influential Examples: Active Learning with Expected Model Output Changes". In: *ECCV*. 2014.

[6] Y. Gal and Z. Ghahramani. "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference". In: *ArXiv e-prints* (June 2015). arXiv: 1506.02158 [stat.ML].

[7] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. "Deep Bayesian Active Learning with Image Data". In: *CoRR* abs/1703.02910 (2017). arXiv: 1703.02910. URL: http://arxiv.org/abs/1703.02910.

[8] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a.html.

[9] Benjamin Graham, Jeremy Reizenstein, and Leigh Robinson. "Efficient batchwise dropout training using submatrices". In: *CoRR* abs/1502.02478 (2015). arXiv: 1502.02478. URL: http://arxiv.org/abs/1502.02478.

[10] *Keras reference website*. https://keras.io/.

[11] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. "Learning Active Learning from Real and Synthetic Data". In: *CoRR* abs/1703.03365 (2017). arXiv: 1703.03365. URL: http://arxiv.org/abs/1703.03365.

[12] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[13] David D. Lewis and Jason Catlett. "Heterogeneous Uncertainty Sampling for Supervised Learning". In: *In Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 1994, pp. 148–156.

[14] *Matplotlib reference website*. https://www.matplotlib.org/.

[15] *Numpy reference website*. http://www.numpy.org/.

[16] Nicholas Roy and Andrew McCallum. "Toward Optimal Active Learning Through Sampling Estimation of Error Reduction". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 441–448. ISBN: 1-55860-778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655646.

[17] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. "Active Hidden Markov Models for Information Extraction". In: *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*. IDA '01. London, UK, UK: Springer-Verlag, 2001, pp. 309–318. ISBN: 3-540-42581-0. URL: http://dl.acm.org/citation.cfm?id=647967.741626.

[18] *Scikit-Learn reference website*. http://www.scikit-learn.org/.

[19] *Seaborn reference website*. https://seaborn.pydata.org/.

[20] O. Sener and S. Savarese. "Active Learning for Convolutional Neural Networks: A Core-Set Approach". In: *ArXiv e-prints* (Aug. 2017). arXiv: 1708.00489 [stat.ML].

[21] H. S. Seung, M. Opper, and H. Sompolinsky. "Query by Committee". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 287–294. ISBN: 0-89791-497-X. DOI: 10.1145/130385.130417. URL: http://doi.acm.org/10.1145/130385.130417.

[22] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

[23] *Tensorflow reference website*. https://www.tensorflow.org/.

[24] Keze Wang et al. "Cost-Effective Active Learning for Deep Image Classification". In: *CoRR* abs/1701.03551 (2017). arXiv: 1701.03551. URL: http://arxiv.org/abs/1701.03551.

Appendix A

Convolutional Neural Network Detailed

This part of the thesis will introduce in details the *Convolutional Neural Networks*. These models are often used in the frame of computer vision for their capability to extract relevant features from raw images, thanks to the convolution operation performed in some of their layers.zzzz

A.0.1 Convolutional Layers

This type of layers gives the name to CNNs. The convolution operation is motivated by its translation invariance. Each layer consists in a set of learnable filters with a small receptive field that extends through the full depth of the input volume. The operation computed consist in the swept of the filter over the image and, for each location, the output is computed. To better understand the operation is possible to define the following quantities:

- w, h and d respectively the *width*, *height* and *depth* of the input image (the dimensions of the image will be $w \times h \times d$).
- f_w, f_h and f_d respectively the *width*, *height* and *depth* of the considered filter.
- p as the *pixel padding* added at the borders of the input image. Usually, the literature refers to *0-padding* or *1-padding* in case 0 or 1 initialized pixels for padding.
- s_w and s_h the horizontal and vertical *stride*, that is the number of pixels of which the filter is moved at each sliding operation.

At this point is possible to define the output of a generic convolutional layer using the quantities defined above:

$$O_w = \frac{w - f_w + 2p}{s_w} + 1$$

$$O_h = \frac{h - f_h + 2p}{s_h} + 1$$

In this case, O_w and O_h represents the output width and height, while de depth remains unchanged because its value should match in the input image and the considered filter. The following picture visualizes the operations performed in the convolutional layer:

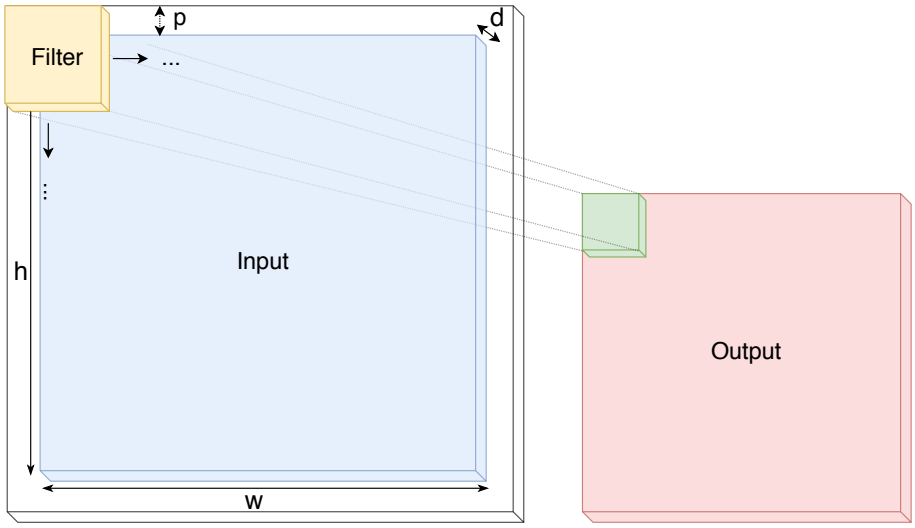


Figure A.1: Visualization of convolutional layer operation, in green, in this case, the result of the convolution between the selected part of the input and the filter.

In this case, is possible to see that there is a local connection between neurons, as happen for the human vision. These type of layers extract progressively more information about an image. The following image, using different colors for different positions of the filter, illustrates the operation with an example, in the case of $s_w = s_h = 1$, $d = 1$, $p = 0$, $f_w = f_h = 2$ and $w = h = 4$:

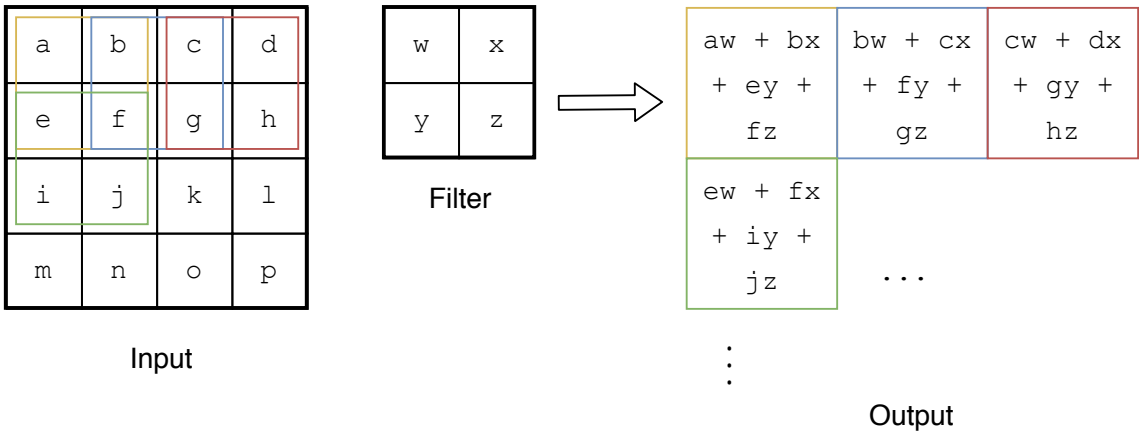


Figure A.2: A partial example of the convolution operation with a sliding filter over the input.

Given the parameters specified before is possible to compute the size of the output image that will be $O_w \times O_h \times d = 3 \times 3 \times 1$

A.0.2 Pooling Layers

In-between successive convolutional layers are often inserted *Pooling layers* that, as in the previous case, operate with a *sliding window* that is swept over the full width and height of the input. The operation performed is usually the computation of the average or the max of the input numbers. Using the same definitions in subsection A.0.1 is possible to illustrate with an example the case with $s_w = 2$, $s_h = 2$, $d = 1$, $p = 0$, $f_w = f_h = 2$ and $w = h = 4$:

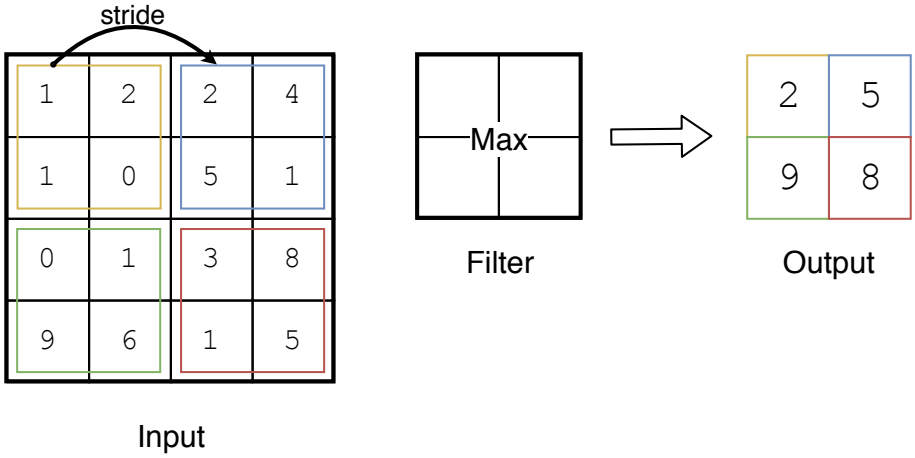


Figure A.3: Example of the max-pooling operation with a sliding filter over the input.

As possible to infer from the image above, the size of the output is usually smaller than the one of the input due to the nature of the operation. Also in this case is possible

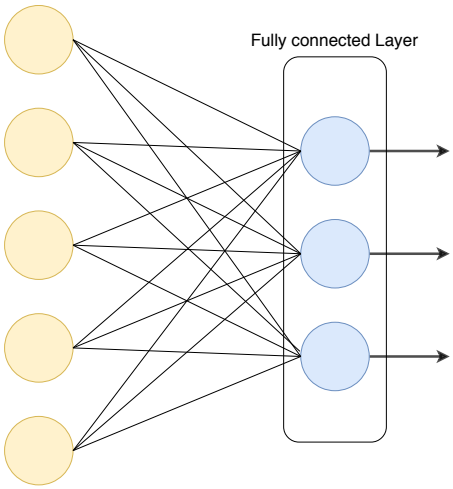


Figure A.4: Architecture and connections schema for a Fully Connected Layer.

to compute the output dimensions:

$$O_w = \frac{(w - f_w)}{s_w} + 1$$

$$O_h = \frac{(h - f_h)}{s_h} + 1$$

In this case, the output size will be $O_w \times O_h \times d = 2 \times 2 \times 1$, in fact, the depth do not change and the filter should have the same depth of the input.

A.0.3 Fully connected Layers

As can be deduced from the name, in fully-connected layers (often named also *dense layers*) each neuron in the previous layer is connected to the one of the next one. The outputs of previous pooling or convolutional layers are high-level features that, this type of layers, will use for the classification. The Figure A.4 shows the connection schema highlighting the fully-connected layer.

If this type of layers are placed as the last in the CNN, usually the used activation function is the *softmax*, and can be expressed as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \; for j = 1, ..., K$$

Using this function is ensured that the sum of the output probabilities from the fully connected layer is 1.

A.0.4 Used Architecture

The architecture used to conduct the experiments is the one introduced by Simonyan and Zisserman [22]. The network was first introduced in 2014 and demonstrates how the depth of the network is a crucial component for the performance. The image below shows a schema of the architecture:

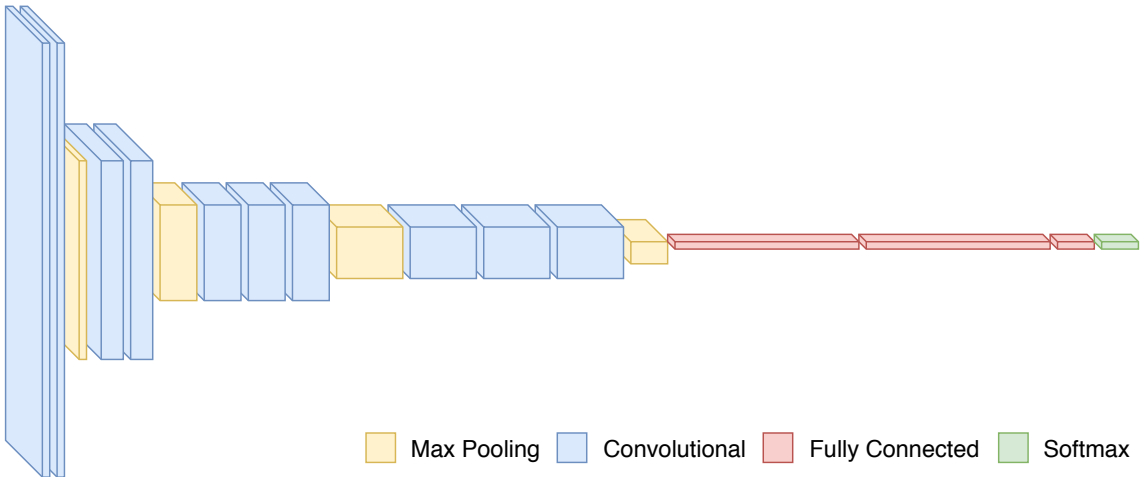


Figure A.5: VGG16 architecture schema with details of the layer types

This network was used because of its capability to generalize well on different datasets. Passing to implementation details, the network uses convolutional layers with 3×3 filters and extends the depth of the network to 16 layers. Each convolutional layer is designed to do not impact the width and height dimensions of the input, for example using stride equal to 1 and 1 pixel of *0-padding*. The pooling operation is designed to halve the dimension of the input image using a 2×2 spatial window and a stride of 2 (similarly to the example in the figure A.3).

The stack of convolutional and pooling layer is followed by two fully connected layers with a depth equal to 4096, followed by the last, the depth of which depends on the number of classes in the considered dataset. Each layer of the network is equipped with the ReLU activation function, that had rapidly become one of the most used in neural networks after the first introduction in deep learning context by Glorot, Bordes and Bengio [8].