# POLITECNICO DI TORINO

**Master's Degree in Computer Engineering**


**Master's Degree Thesis**

# Cycle-consistent Deep Learning Architecture for Improved Text Representation and Translation

**Supervisors**

**Prof. LUCA CAGLIERO**

**Dr. MORENO LA QUATRA**

**Candidate**

**MICHELE D'ADDETTA**

**July 2021**

**Abstract**

This thesis presents CycleNLPGAN, a deep learning architecture that introduces an innovative approach to sentence encoding and alignment. It allows the definition of a latent vector space shared across a pair of languages. The model is jointly trained to perform neural machine translation from a source language $A$ to target language $B$. It generates a shared aligned vector space suitable for machine translation from a source language to a target one and vice versa. The architecture is based on a CycleGAN, a Computer Vision model that address image-to-image translation using cycle-consistent dynamics. It enforces the robustness of the resulting model and the quality of produced data. The architecture is defined using a cycle consistency loss, an approach used in neural machine translation and in domain adaptation models. The designed model aims at creating two mapping functions $G_{AB}$ and $G_{BA}$ such that the sentences $G_{AB}(A)$ generated by $G_{AB}$ translated from source language $A$ can be considered as indistinguishable original sentences in language $B$. The exact same principle holds for the inverse mapping function. The joint training of the two mapping functions and the introduction of the cycle consistency loss produces a model able to generate sentence embeddings that belong to the shared latent vector space. Two separate sentence embeddings are close in the latent space when the sentences are the same, or different languages sentences have similar meanings. At the same time, the model is able to produce accurate translation. The translation quality is enforced using the back translation, whose produces values defined as $G_{AB}(G_{BA}(B))$ and $G_{BA}(G_{AB}(A))$. Back translation values allow the definition of highly related mapping functions. In fact, during the training, each mapping function uses both real data and data defined from the opposite mapping function, in order to increase the relationship among them. The model is trained using an adversarial generative approach and uses a bilingual parallel dataset (Opus), which contains a large amount of aligned data for several language pairs. The effectiveness of CycleNLPGAN approach is demonstrated addressing both sentence alignment and neural machine translation tasks, obtaining results that are competitive against state-of-the-art models.

# Acknowledgements

Ringrazio la mia famiglia per essermi stata vicina durante questa esperienza fantastica. Grazie a mia madre e a mio padre, senza i quali non avrei mai potuto raggiungere questo traguardo così importante. Per tutto il supporto fornitomi, sia economico che morale, per tutti gli incoraggiamenti sia in caso di un buon voto, ma soprattutto quando un esame non andava come sperato. Non mi avete mai fatto sentire il peso di aver intrapreso questo percorso lontano da casa e che ha richiesto tanti sacrifici, neanche per un secondo in questi lunghi 6 anni, e ve ne sarò sempre grato.

Ringrazio le mie sorelle, i veri amori della mia vita, che mi hanno dato tutto quello che un essere umano può definire come amore. Siete state fondamentali durante questi anni, grazie alla vostra esperienza ma soprattutto grazie al vostro contributo nei momenti più difficili. Siete e sarete sempre le mie rocce, su cui potrò sempre contare nei momenti di difficoltà.

Grazie a Cecilia, la mia puzzola, perché nonostante sia un essere ancora così ingenuo e innocente, è in grado di mostrare amore e affetto come nessuno è mai riuscito a fare.

Ringrazio Francesca e Leonardo, i miei cug, per essere stati i miei punti fermi dal primo giorno, non solo durante il periodo universitario, ma da sempre. Abbiamo condiviso ogni genere di esperienza insieme, e avervi avuti con me durante questi anni a Torino è stato come non aver mai lasciato casa.

Infine ringrazio tutti i miei amici e parenti, in particolare Mattheo, Luca, Nicola, Michele, Giuseppe, Giulio, Aldo, Beatrice, Angela, Sergio, Edoardo Amico, Edoardo Giordano, Giorgio, Andrea, Igor, per essere stati parte della mia vita e per aver reso tutto più leggero e spensierato. Ognuno, a proprio modo, ha contribuito a rendere quest'esperienza fantastica, e non vi dimenticherò mai.

# Table of Contents

# List of Tables

# List of Figures

# Acronyms

**AI**
Artificial Intelligence

**CNN**
Convolutional Neural Network

**FCNN**
Fully Connected Neural Network

**GAN**
Generative Adversarial Network

**LSTM**
Long Short Term Memory

**ML**
Machine Learning

**NLP**
Natural Language Processing

**RNN**
Recurrent Neural Network

# Chapter 1

# Introduction

In last decades, Artificial Intelligence and Machine Learning research have seen increasing interest. At the time of writing, almost any life-related aspect could benefit of innovative methodologies developed in Artificial Intelligence (AI). However, AI history started a long time ago.

## 1.1 Artificial Intelligence History

By the 1950s, scientists, mathematicians, and philosophers have explored artificial intelligence and concepts behind it. A pioneering work of Alan Turing, a British polymath, suggested that humans use available information to solve problems and to take decisions. He strongly believed that machines, with adequate training, would do the same. Turing studies were stopped because of the scarcity of resources, in fact computers of that era couldn't store commands. Moreover, storage and computing were extremely expensive. Some years later, some mathematicians designed a program, the *Logic Theorist*, which emulated human problem solving skills. It's considered as the first artificial intelligence program. From '60s to '70s, computers could store more data, computational efficiency increased and became more accessible, so Artificial Intelligence and Machine learning algorithms improved, introducing a better interpretation of written and spoken language. These convinced government agencies to increase AI research, and the focus was on algorithms that could transcribe and translate spoken language. A new setback was in the late 70's. Computers were not fast enough and storage became a bottleneck, so research efforts diminished until 1990s, when machine learning techniques became popular, allowing computers to learn using experience. In 1997, a chess playing computer program produced by IBM, Deep Blue([1]), defeated world chess champion Gary Kasparov. In the same year, a speech recognition software was implemented. This was a huge step forward in language interpretation. The fundamental limits of

Artificial Intelligence were the computer storage and computational resources, but in year 2021, in most cases, this problem is mitigated.

With the advent of big data, it is possible to collect huge amounts of information in few hours using lots of sources, so even if algorithms are similar to the past, big data and massive computational resources allow artificial intelligence to use more data during the learning process, and this leads to the increase of models accuracy.

## 1.2 Machine Learning

Most of Artificial Intelligence success comes from Machine Learning. It uses algorithms and neural network models to let computer systems improve their performance in task such as prediction and generation of data. Machine Learning algorithms build a mathematical model using sample data to make decisions without being specifically programmed to make them. Machine Learning is based on a model of brain cell interaction. It implements models and algorithms, which are exploited in the definition of more complex systems, defining an Artificially Intelligent system. More recent Machine Learning algorithms use Deep Learning approaches in order to produce more accurate and performing models. Deep learning is a subfield of Machine Learning, and its main characteristic is the usage of algorithms that uses layers to model an artificial neural network. The resulting neural network is able to make decisions according to analysed data. The variety of tasks addressed by Machine Learning is multifaceted, but the most important ones concern the elaboration and understanding of visual (e.g., photos,videos and pictures) and natural language data (e.g., words or sentences in different languages).

Computer vision is a branch of Machine Learning which aims to understand, generate and classify images and videos. Applications of computer vision algorithms allow to recognize objects, people, animals, to segment images, differentiating objects and patterns in them. Nowdays, applications and research fields involving computer vision techniques are limitless, but the history of this branch started in the 50s and 60s, when a neurophysiological research conducted by D.H. Hubel and T.N. Wiesel on cats demonstrated that human vision is hierarchical [2]. Neurons detect edges and simple features, then shapes and detailed features are analysed. Finally more complex visual representations are detected. Strengthened by this knowledge, computer scientists tried to recreate human vision structure in a digital way. Even now, computer vision systems work as the human counterparts, so using a hierarchical approach to perceiving visual stimuli. Several systems were implemented during the second part of the past century, but the low amount of computational resources limited their results. Moreover, implemented models used shallow architectures, that produced inaccurate results. Starting from 2010s computer scientists gained access to more data than ever before and computing

hardware improved substantially, as costs went down. Algorithms developed in the 1980s-90s improved and computer vision had its breakthrough moment in 2012 during the ImageNet Large Scale Visual Recognition Challenge, also known as ILSVRC [3], an annual image classification competition. Research teams evaluate their algorithms on the ImageNet [3], data set, trying to achieve higher accuracy on visual recognition tasks. Until then the best-performing approach reached an error rate of around 26%, in 2012 a team from the University of Toronto published AlexNet [4], a deep neural network, that achieved an error rate of 16%. It was a game-changer innovation for the computer vision community and, in general, for artificial intelligence. From that moment, computer vision research increases, addressing more and more tasks and areas of application.

A similar trend happened in Natural Language Processing (NLP), the branch that addresses the understanding, the learning and the modeling of natural languages, considering their semantics and syntactic aspects.

### 1.2.1 Deep Natural Language Processing

The first study on NLP was made by Alan Turing in 1950. He wrote an article entitled *"Computing Machinery and Intelligence"* [5], in which he discussed how to let machines to be intelligent and proposed the Turing Test, a criterion to quantify machine intelligence exploiting the goodness of an artificial intelligence to understand languages. The proposed artificial intelligence should be able to understand and correctly use languages in order to make its speeches and sentences indistinguishable from human ones. Up to 1980s, NLP systems were based on hand-written rules and produced inaccurate results. Starting from 1990s, the approaches used in NLP changed and shallow machine learning algorithms have been used, introducing a more structured approach on learning. According to [6], this is the second wave of NLP, characterized also by the increasing availability of machine-readable data and of computational power. Discriminative models became the standard approach in most tasks, but the model shallowness reflected on the lack of constructed structures. This class of algorithms (i.e., decision trees [7] or shallow neural networks [8]) were able to infer rules directly from data. In the same period, translation knowledge increased significantly due to the availability of sentence-level alignments in the bilingual training data. In this way translation is not made by rules, but directly from data. In the 2010s, the third wave of Natural Language Processing started, and deep neural network ML methods became widespread in natural language processing, reaching state-of-the-art results in many natural languages tasks, such as language modeling, parsing, part of speech tagging and others. Deep learning can be viewed as a sequence of cell models, inspired by biological neural systems. If in the second wave the amount of training data wasn't

enough and learning methods were inaccurate, the third wave is marked by the huge amount of data. In fact, the lack of data let learning signals vanish exponentially while passing through deep neural networks layers, making it difficult to tune weights of them, especially for recurrent networks. These neural networks are able to discover intricate structures in high-dimensional data, so they have been applied to real-world tasks in artificial intelligence including speech recognition, image classification, and NLP. Another of the biggest advantages of deep networks is the backward algorithm definition, that defines how to update each weight in each layer of the network according to the difference among expected and predicted values.

Since 10 years, it has been shown that the availability of big data and faster computation is useful and beneficial. A large number of NLP applications, such as image captioning, visual question answering, speech understanding, web search, and recommendation systems, in addition to many non-NLP tasks, have reached unbelievable results thanks to deep learning. In machine translation and in text-based NLP application areas, deep learning impacts significantly. Advancing from the shallow machine translation algorithms implemented during the second wave of NLP, the current best machine translation systems are based on deep neural networks. Two important recent technological breakthroughs obtained applying deep learning to NLP problems are sequence-to-sequence learning and attention modeling. The sequence-to-sequence learning uses recurrent networks to carry out both encoding and decoding in an end-to-end manner. While attention was initially developed to overcome the difficulty of encoding long sequences, following developments significantly extended its capability to align two sequences in a flexible way. The key concepts of sequence-to-sequence learning and of attention mechanism boosted performances on tasks based on distributed word embedding. Popular techniques massively use word embeddings to capture semantic properties and context of words, increasing significantly learning of higher-level tasks. Word embeddings represent words as numerical vectors, allowing the definition of a latent space that includes and represents a language. First word embedding algorithms produced a representation of word using the entire word and subparts of it, but they didn't capture the more complex properties of words, such as their context and semantic aspects, so the results were not so accurate. In latest years, several approaches improve the definition of word embeddings, and this leads to a significant increase of learning results and language modeling quality. The popularity and the research fields embraced by Natural Language Processing increase more and more, leading to the creation of models capable of translating, generating sentences and transferring information from a language to another. After these successes, these concepts have been applied to several NLP-related tasks such as image captioning, speech recognition, one-shot learning, text understanding, summarization, and question answering.

## 1.3 Discriminative vs Generative models

Main architectures in Natural Language Processing belong to two groups of models: generative and discriminative models. Generative models are used to describe how a dataset is generated in a probabilistic way. They are able to generate data using the same model of given dataset. Having a dataset of sentences in a certain language, a generative model is able to generate new sentences in the same languages that don't exist in the dataset and it has never seen before, but that still appear similar to the real sentences. This is because the model learns the underlying data distribution and model the structure of the language. This approach require a large training dataset with many examples, so the model will learn to generate features and structure that look as if they have been created using the same rules as the original data. Generative model requires a stochastic random element in order to influence the generated samples, otherwise the model is nothing more than a fixed calculation. Without the random element, the generative model will produce the same value every time.

The counterpart of generative models are defined as discriminative. Given a dataset including sentences from different languages, a discriminative model learns to predict if they belong to English or to other languages, their sentiment, or to classify them, basing on words, syntactic and semantic structures in sentences. So a discriminative model learns features related to classes assigned to each input value. According to the input value and its class, the model is able to capture details from the input value and uses them as key characteristics of its class. Discriminative models use data that has labels during training, in order to learn how to discriminate between input groups and outputs the probability that an input example is associated to a label $l \in \mathcal{L}$, where $\mathcal{L}$ is the set of candidate labels. For this reason, discriminative modeling often requires supervised learning, while generative models are usually trained using unlabeled dataset. However, it is also possible to use labeled datasets for training generative models. In this case, labels are discarded and only raw data are used.

From a mathematic point of view, discriminative modeling attemps to calculate the probability that a data belongs to a certain class, meanwhile generative models attempt to estimate the probability of seeing a certain data without caring about the associated class. The key point is that even if a discriminative model can perfectly identify categories, it would still have no idea how to create a similar data. It can only output probabilities against inputs according to the task, as this is what it has been trained to do. A generative model, on the other hand, can output values that have a high possibility of belonging to the original training dataset.

Typically, generative models are more elaborate and address more complex tasks than discriminative models. In fact, generation may involve more complications during the training, and the more the data are complex, the more the model may

not obtain adequate results in generation. Some of the most common drawbacks in generative models are the collapse, in which the model produces a small set of outputs over and over during iterations, and the failure to converge, in which the quality of produced data is not adequate if compared to original dataset one. Examples of generative models are variational autoencoders (VAE) [9] and generative adversarial networks (GAN) [10].

Discriminative models instead are more robust and , in most cases, outperform generative models on classification error rates. Moreover, these models require a lower amount of data with respect to generative models. On the other hand, discriminative models don't explicit claims about how the data is generated, so they don't learn to reproduce and to model the structure and the composition of data, but only their classification according to the examined features. Examples of discriminative models are random forest [11], support vector machine (SVM) [12], convolutional neural network (CNN) and, more in general, classifier models.

## 1.4 Main Natural Language Processing Tasks

Natural Language Processing addresses several tasks, and their number increases over the years. Some of them have direct real-world applications, while others are used as subtasks and aid in solving larger tasks. Some of the most relevant tasks are enumerated below:

- **Automatic summarization** : this task aims to condense a piece of text to a shorter version. The source text is shorten in length while preserving the most relevant information. Text summarization methods fall into two different classes:

  1. Extractive approach, in which sentences are directly picked up from the document, and the quality is defined with a scoring function based on the coherence of the summary. Important section are identified and then they are cropped out to produce a condensed version. At the time of writing, most of research focus on this approach
  2. Abstractive approach, in which the summary is produced by interpreting the text with advanced NLP techniques, and a shorter text is generate as result. The text may not contain parts of the original text and keeps the most critical information of the original text, so rephrase and incorporation of information are required

  Some of the most common approaches to address automatic summarization are [13], [14] and [15].

- **Named Entity Recognition** : it is part of information extraction field and aims to identify and classify named entities in a text belonging to a set of predefined categories, such as people, places, objects, national authorities and so on. Named entity recognition bases its operation on knowledge base, that contains a list of named entities that can be extracted from a text. Most common Named Entity Recognition approaches are [16] and [17].
- **Neural machine translation** : it is a technology based on neural networks to translate text from a language to another. There are several variations of NMT currently being investigated and deployed in the industry. One of most stable and used versions of NMT is the Encoder Decoder structure, an architecture composed of two recurrent neural networks used in tandem to create a translation model. The encoder RNN takes a sentence as input in a number of consecutive time-steps and creates the hidden state vector, storing the information about the sentence. At each time-step, the hidden vectors adds information from the current word, while preserving the already stored information. At the final time-step, the whole input sentence is stored in the hidden state, which is given as input into the decoder. It predicts a word at each time-step until the complete sentence is produced. Most recent models introduce attention mechanisms, producing a more accurate input value for the decoder and achieving state-of-the-art results. Most of neural machine translation models use approaches such as [18] and [19]
- **Part-of-speech tagging** : a task that allows to convert a sentence or a text to list of tuples, where each tuple has the form $(word, tag)$. The tag represents a part-of-speech tag, such as noun, verb, adjective, etc. POS-tagging algorithms can be grouped into two distinctive categories:
  1. **rule-based tagging**, one of the oldest techniques that uses a dictionary for getting possible tags for each word. If more tags are possible for a word, then some hand-written rules are used to identify the correct tag.
  2. **stochastic**, a set of techniques that includes frequency or probability to define words tags.
- **Question answering** : a task which aims to automatically answer back to a certain posed question expressed in natural language. There are three major modern paradigms of question answering:
  1. IR-based Factoid Question Answering, which answer a user's question by finding short text segments on the Web or some other collection of documents.
  2. Knowledge-based question answering, that answer a natural language question by mapping it to a query over a structured database. The

question is structured in the form of a query or can easily be converted into using semantic parser

3. Using multiple information sources, in which models applies parsing, named entity tagging, and other processing step to extract information about the question. Then it is passed through the candidate answer scoring stage, that ranks candidates and defines the most correct answer

Some models that address this task are [16], [20] and [21].

- **Text categorization**: the process of categorizing a text into classes and groups. Some of the most common use cases in which text classification is included are sentiment analysis, in which models understand if the text is talking positively or negatively about something, topic detection, in which the model identify the tpoic of a text, and language detection, that aims to detect the language of a given text. Most accurate text categorization models are based on [20], [21] and [18].

- **Text segmentation** : the process of identifying the boundaries between words, phrases, or other linguistic meaningful aspects, such as sentences or topics. The resulting sentence is useful to help humans reading texts, and are mainly used to assist in more complex artificial processes.

- **Textual entailment** : the task of deciding, given two text fragments, whether the meaning of one text is entailed from another text. This technique is relevant for a wide number of more complex applications, giving a measure of correlation among text fragment.

All of them use specific architectures and models in order to reach best results. Sometimes it happens that models are shared among tasks. As instance, BERT [21] architecture has been applied in several tasks. It maintains the main structure while an additional subpart is stacked on top of it to address specific tasks

When models are trained in a cross-lingual or multilingual fashion, embedding alignment may be exploited. Embedding alignment is a task that aims to align languages and models among them, producing vector representations in a shared latent space. Not every task may exploit it because it is not appropriate for the specific task, such as Named Entity Recognition, or because of the structure of models, such as summarization. But one of the main advantage of text alignment is the possibility of transferring information from a language to another and vice versa, so even task that do not use multilingual data may benefit form alignment. As instance, a model specialized in English texts summarization, which is a rich language with lots of complex idiom structures and with a huge amount of available annotated data, may transfer information to a model which does the same in a low

annotated language, such as Hindi, that may have a lower accuracy. In this way, the Hindi summarization model may improve its outcomes exploiting information given by the more robust English model. Finally, embedding alignment can be seen as a task by itself, but most of its practical applications are as part of a broader pipeline.

## 1.5 Thesis contribution

It often happens that Computer Vision and Natural Language Processing have common techniques and approaches. By adapting the state-of-the-art models from a field to the other, it is possible to have great improvement for the final tasks. This is because of the similarity among tasks, so methods and techniques can be adopted in other fields. Even in generation task, both computer vision and natural language processing exploit GAN [10] architectures.These models take a latent random value as input and produce an output object. This output is built in order to be as similar as possible to data analysed from training datasets, as explained in previous sections. GANs are used in computer vision to generate images, meanwhile some GANs approaches have been applied in natural language processing to generate reliable and well defined sentences. Even in word embedding alignment, models like MUSE [22] use GANs to define a mapping function to align word latent space among languages.

One of the most popular tasks in last years is domain adaptation, which aims to transform images among domains, making them as similar as possible to data that actually belong to the target domain. More specifically, a domain is a data distribution with features that differ from other distributions. As instance, some domains can be real photos, drawings, sketches and synthetic images, that can represent the same objects, but in different style. Examples of images from different domains are shown in figure 1.1.

In the field of computer vision, a domain adaptation model that widely exploits GANs and achieves great results in style transfer is CycleGAN ([23]), a model that uses two GANs to transform images from a domain to another using a cycle-consistent approach. Examples of results obtained using CycleGAN are given in figure 1.2.

**Figure 1.1:** Domains in Computer Vision adaptation tasks

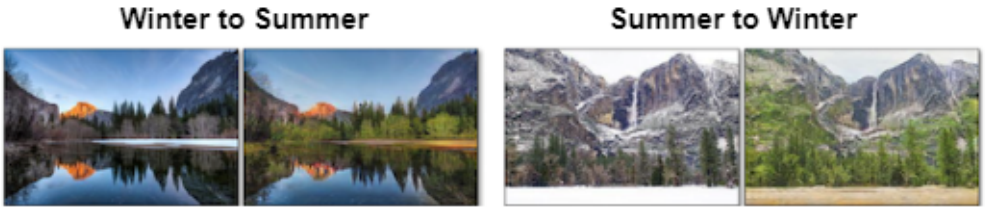


**Figure 1.2:** CycleGAN domain transfer examples

A natural language processing task that can be related to domain adaptation is neural machine translation. In fact, a language can be seen as a domain, where text features are different among languages (i.e. nouns, subjects, verbs), but their meaning can be the same in all of them. In this way, a language is considered as a domain, and the translation among languages has the same meaning of a transformation among domains.

In order to reproduce this behaviour in natural language processing, this thesis proposes to adapt CycleGAN to NLP, producing an architecture able to translate among languages using the cycle consistency approach. Neural machine translation is a very competitive task, and the main translation models are not available as open source code. **This thesis work focuses on using CycleGAN architectures to combine neural machine translation with domain adaptation.** In this thesis proposal, open source models have been used, even if they may not be as accurate as best developed architectures. The main target of this thesis is to avoid a translation-only task, and this is achieved introducing the sentence alignment objective in the thesis model, called CycleNLPGAN.

CycleNLPGAN main objective is to produce models that share the same embeddings latent space, trying to force them to produce close sentence embeddings for the same sentences in different languages. In this way, models are aligned and produce similar results for similar input values, improving the re-usability of the model and performances in models with lower accuracy with respect to most accurate models, such as English BERT [21] and BART [24]. At the same time, CycleNLPGAN translates sentences, trying to improve translation results introducing the cycle consistency.

The desired outcome of the CycleNLPGAN process is shown in figure 1.3.

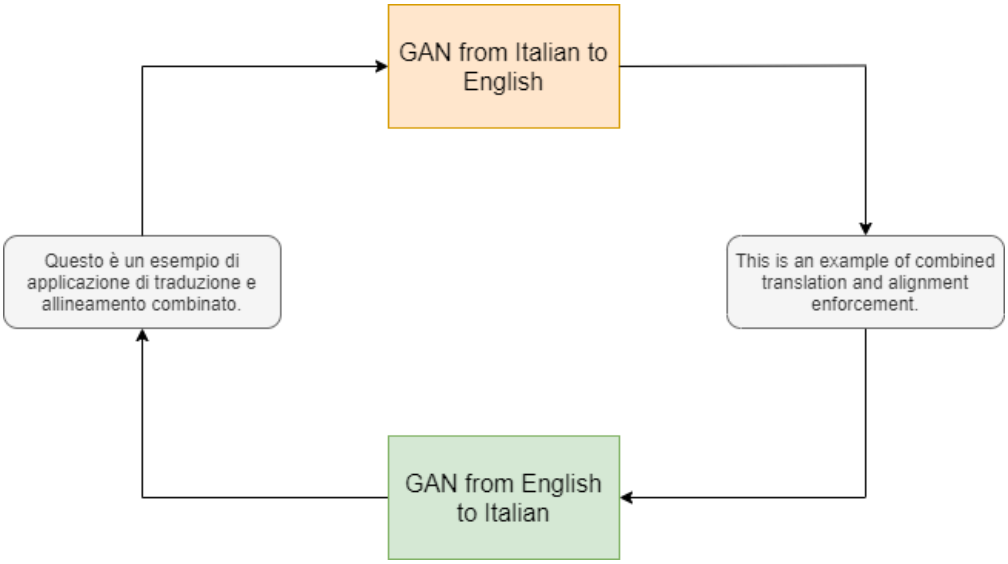


**Figure 1.3:** CycleNLPGAN results example

The result of the training phase is composed of two generators, that are able to produce a reliable translation from a source language to a target one, and two discriminators. Usually, discriminators are discarded when training is over, but those produced using this technique may be applied in other tasks, such as speech recognition. In fact, they learn to distinguish among real and fake sentences, but when the generators become more accurate and generate real-like sentence, then they learn how good is a sentence and if its composition is correct.

# Chapter 2

# Related Works

In the following chapter, the technologies used for this thesis are explained, in order to understand the underlying technologies and methodologies.

## 2.1 Deep Learning

### 2.1.1 Deep Learning Background

Deep learning is a branch of machine learning that uses algorithms based on the structure of the human brain. The main components of the human brain are the neurons, that produce signal transmitted by the synapses. An artificial neural network [25] is composed by elaboration units called neurons [8], which, after computation, transfer signals to the other units using the connection network between them. The composition of a neuron is represented in figure 2.1. The artificial neurons receive some input value and multiply them using a weight vector. The result is a weighted sum, so a scalar value. A bias value may be added, then the value goes through an activation function, and the result is the output of the neuron.

Neurons are linked between them, creating a network. Most of the neural network structures group the neurons in layers. In a fully connected neural network, each neuron in a layer is connected to the neurons in the previous and in the next layer. So when an input vector is given, the neurons in the first layer, called input layer, produce their results and propagate them to the next layer. The operation is iterated by the several layers in the neural network until the last layer, called output layer, is reached. The result produced is used to define the error of the calculation made by the neural network compared to the expected output. The error is back-propagated through the network layers and the neurons update their weights and bias values, also called offsets, according to it. The update value is

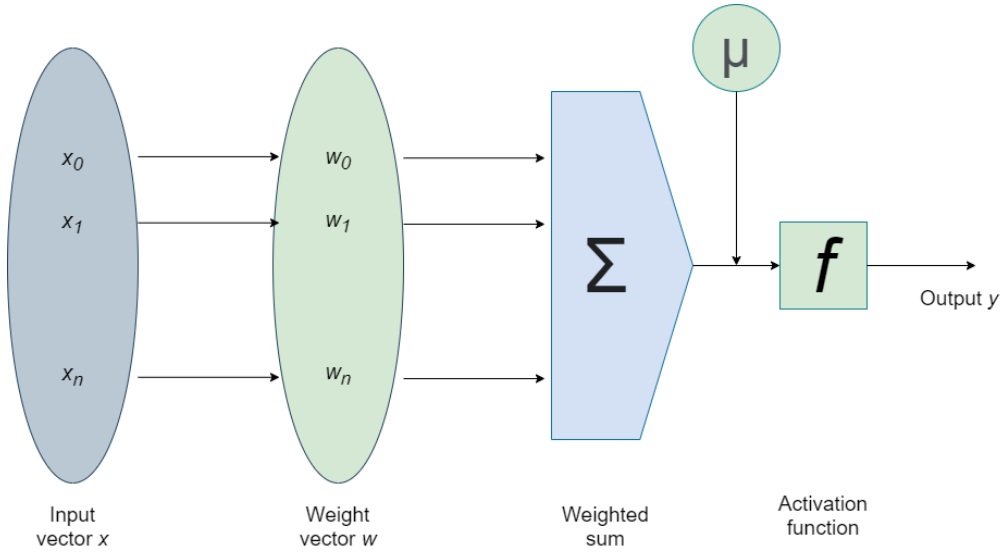


**Figure 2.1:** Composition of a neuron

defined according to learning rate, back-propagation function and loss value. The goal of this procedure is to find the most correct network parameters, in such a way that the network can improve its predictions.

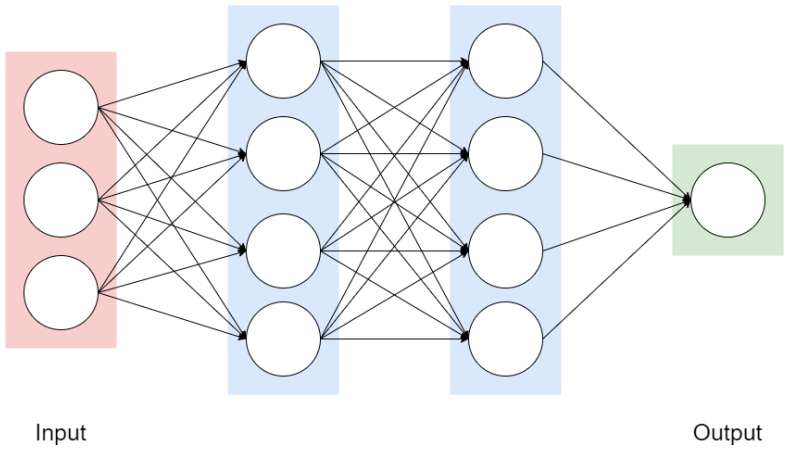


**Figure 2.2:** Neural network architecture

The architecture shown in figure 2.2 is the basic composition of a feed-forward neural network, however, several architectures have been implemented in machine learning during the years.

### 2.1.2 Recurrent Neural Network

Another important neural network architecture that has been successfully applied in Machine Learning and, specifically, in Natural Language Processing, is the recurrent neural network [26]. The recurrent net uses a hidden state vector to model the previous input and output data and uses it during the prediction of the current output. In this case, the final output depends also on previous inputs of the sequential series. Comparing the two architectures, both use the input value, but the feed-forward net does not take into account the sequence of the prior predictions and data. The main difference between feed-forward and recurrent neural networks (RNNs in short) is the use of the hidden state vector that represents the context based on prior inputs and outputs vectors. In most cases, the RNN architecture is mono-directional and uses previous data, but bidirectional RNNs can be used to look into future data too. In NLP, RNNs are used for a huge range of tasks, such as classification, summarization and translation.

The most used RNN architecture in Natural Language Processing is the Encoder Decoder RNN architecture, also called Sequence to Sequence RNN. In this architecture, both encoder and decoder are RNNs. The encoder updates its hidden states with the whole input sequence and produces a final context output vector. This sequence of values is fed into the decoder, which expands the context vector and gives a sequence of output vectors that compose the output sequence. This architecture is widely used in translation and alignment tasks and it is the base concept of more recent and more accurate approaches to them.

### 2.1.3 Long Short Term Memory Network

Another type of neural network architecture that is used in Natural Language Processing and, more in general, in Machine Learning, is the Long Short Term Memory architecture, often referred as LSTM [27]. This architecture was created to solve some issues that emerge using RNNs, such as the short-term memory. This issue mainly refers to the problem for which the recurrent neural network forgets first elements of the sequence when long sequences are given as input. LSTM architecture has an approach similar to RNN architecture one, but it computes inputs, outputs and hidden states in a different way. Figure 2.3 illustrates the difference between the core components of both the architectures. Instead of passing input and hidden state through an activation function, LSTM introduces additional gates and a cell state, used to address the problem of keeping or resetting the context. The gates are activation functions used to define if previous information should be kept or deleted (forget gate), if the cell state has to be updated or not (input gate) or the value of the next hidden state (output gate). Cell state keeps information during the processing of the entire sequence, reducing the effects of short-term memory.

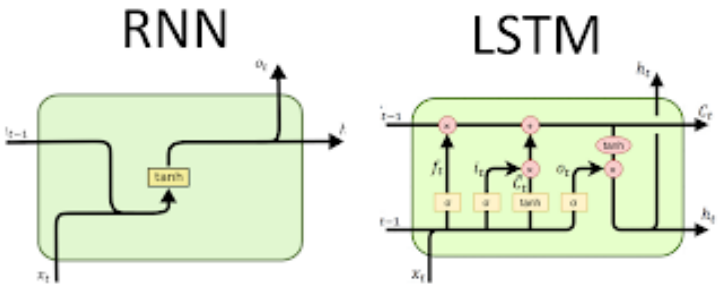


**Figure 2.3:** Differences between RNN and LSTM cell.

All these features characterize a LSTM cell and puts up a more stable architecture that is able to keep longer and less recent information.

### 2.1.4 Generative Adversarial Network

An important network architecture that is taken into account during the thesis project is generative adversarial net, GAN in short. It is a network presented in [10], which is trained to estimate generative models using an adversarial process. The model is made using two models:

- a generative model $G$, that generates data into the learnt data distribution. The generated data is used as negative training examples for the discriminator
- a discriminator model $D$, that estimates the probability that a data comes from training data rather than G. It penalizes the generator for producing implausible results

The $G$ network is trained to maximize the probability of $D$ making a mistake and to foolish it, meanwhile $D$ is trained to maximize the probability of making the correct prediction. This type of training follows the minimax two-player game approach. The whole model architecture is represented in figure 2.4.

Generator and discriminator have different training processes. During discriminator training phase, the generator is kept constant, and the opposite happens during the generator training phase. During first training steps, the generator produces data with poor quality, so the discriminator learns to detect fake data. As training goes on, the generator gets closer to real data producing data that fools the discriminator, running out for a worsening of the discriminator, that can't find differences between real and fake data.

As expected, a GAN uses two loss functions: one for generator and one for discriminator training. The formula that represents the objective is called minimax

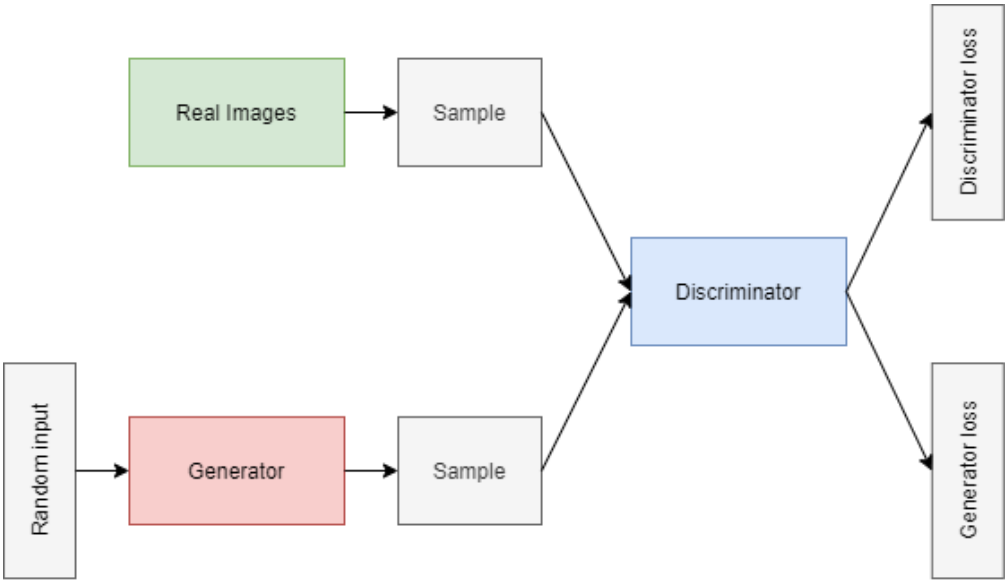


**Figure 2.4:** GAN architecture

loss and is

$$\mathbb{E}_x[logD_(x)] + \mathbb{E}_z[log(1 - D(G(z)))]$$

, where $D$ is the discriminator, $G$ is the generator, $\mathbb{E}_x$ is the expected value over real data, $\mathbb{E}_z$ is the expected value over the random inputs and $z$ is a random noise used as generator input. The goal of the generator is to minimize the minimax loss, while the discriminator tries to maximize it. This model has outperforming results in several tasks and machine learning branch, such as computer vision and natural language processing, improving and making easier to exploit unsupervised training.

On the other hand, models based on GAN structure may be unstable. Indeed it often happens that generators diverge and are not able to produce affordable data, so the correct configuration among hyperparameters must be found to achieve reliable results.

## 2.2 Natural Language Processing

The current section introduces the main technologies used in Natual Language Processing and their basic principles.

Natural Language Processing, or NLP in short, is a branch of Machine Learning, focused on the capability of understand, read, write and derive meaning from human languages. The algorithms used during the thesis workflow use the same approach to prepare and transform sentences in order to introduce them into the input layer of the network. The first pre-training step used is the tokenization, a

process that removes some characters as punctuation and segments the text into words or group of words called tokens. Once the tokens are created, they must be transformed in numerical values in order to feed them into the network. The dominant approach that represents the input text as a numerical representation is the word embedding.

### 2.2.1 Word Embedding

The word embedding is a numerical representation of a word, so a projection of the word into a vector space. The simplest word embedding algorithm is the one-hot encoding. Each world in a vocabulary is associated with a specific index. Given a vocabulary of $N$ words, a word is represented as an $N$-length vector, where all values are 0 except the one associated to the word index, that is equal to 1. This approach is sub-optimal because of several reasons:

- words that are semantically or syntactical similar may have completely different representations and it is not possible to relate them.
- the more the vocabulary size , the more the vector length, so the curse of dimensionality occurs.
- the word embedding is sparse and high dimensional, so the required memory is not negligible.
- frequent words influence the behaviour of the net when less frequent words occur. If the embeddings of a rare and of a frequent word are similar, then the network will handle them as similar instead of learn how to handle the rare one.

Better approaches used to produce embeddings that resolve partially the previous issues are Word2Vec, FastText and Glove. These approaches exploit stronger principles that allow to produce similar embeddings for semantically alike words.

#### Word2Vec

Word2Vec is an unsupervised learning method that computes continuous vectors as distributed representations of words. The technique presented in [28] tries to produce vector representation for a certain word that is close to similar words representations, exploiting syntactical and semantic similarity. The produced vectors can be used for algebraic operations and the result is close to the vector representation of the expected word with an high accuracy. The neural network language model used in the implementation can be trained using two different approaches, represented in figure 2.5:

1. Continuous Bag-of-Words Model (CBOW): the architecture is similar to a feed-forward neural network language model made by the input and the output layer. The projection layer of the network is shared for all words. In this model, a word is classified using the word itself, four history words and four future words in the sentence, that represent the context. The main difference with standard bag-of-word model is that it uses the continuous distributed representation of the context. In this approach, the order of the history and future word does not influence the projection.

2. Continuous Skip-gram Model: its architecture is similar to CBOW, but instead of predicting the current word based on the context, it tries to predict words in a range before and after the current word. Using this approach, the network tries to predict and understand the context of a word using the word itself.

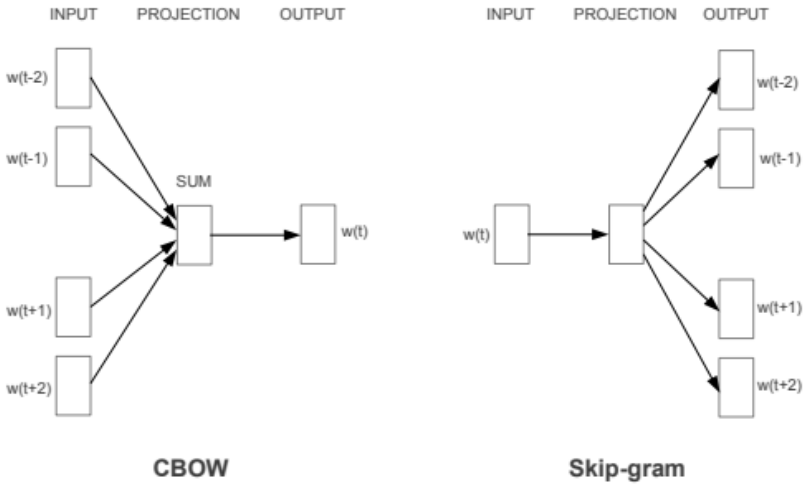


**Figure 2.5:** Word2Vec learning approaches [28]

The word embeddings obtained with this technique demonstrate an high closeness between similar words, but also for other type of similarities between words. The examples explained in [28] show that operation $vector(biggest) - vector(big)$ is close to the value obtained by the operation $vector(smallest) - vector(small)$. This result leads to the operation $vector(Italy) - vector(France) + vector(Paris)$, which gives a vector whose is the closest to $vector(Rome)$. The closeness of vectors is obtained using cosine distance. The main issues of this approach are (i) words that do not appear in the vocabulary cannot be embedded and (ii) words in vocabulary but never appeared in training set are projected into an embedding that may be inaccurate.

### FastText

FastText was presented in [29] to improve results obtained in [28]. It is a technique that generates word embeddings taking a word and representing it as a bag of character $n$-grams and using them to retrieve the entire word representation. In contrast to the other models, it does not ignore the morphology of words. In this way, even words that do not appear in the vocabulary can be represented as embeddings, because of their $n$-grams embeddings. Also the rare word representations improve, because the morphological aspect of the word, so each its sub part, is considered. The model is based on the skipgram approach of Word2Vec and generates a vector representation for each $n$-gram, then it sums them and obtain the word representation. Instead of training to predict words that appear in the context, it trains to predict the presence or the absence of context words. So for a certain word used in the training, some context words are given as positive examples and some random words from dictionary are used as negative examples. The net has to define if a word is or not in context.

FastText exploits a bag of character $n$-gram representation of each word. The representation is defined adding boundary symbols before and after the word, then n characters are taken in order to form a $n$-gram. Shifting right by one character, other $n$-grams are defined until the word ends, and the bag is defined. For each $n$-gram, the embedding is calculated, and the word embedding is the sum of the $n$-grams embeddings. FastText outperforms Word2Vec in most of the NLP tasks and it is able to train on a large amount of data in short times.

### GloVe

Global Vectors for Word Representation, or GloVe, is a model presented in [30] able to learn vector space representations of words capturing semantic and syntactic features. It tries to improve previous methods drawbacks, addressing their modest performance on word analogy tasks. It exploits statistics computed over the entire corpus focusing on a context window. GloVe model is trained on a global word-word co-occurrence matrix that contains, for each cell, the word-word counts and makes use of these statistics to define the space vectors. This matrix is referred as $X$. The $X_{ij}$ value is the number of times that word $j$ occurs in the context of word $i$, so the probability:

$$P_{ij} = \frac{X_{ij}}{\sum_k X_{ik}}$$

where $\sum_k X_{ik}$ is the number of times a given word appears in word $i$ context, can be defined as the probability that word $j$ appear in the context of word $i$. In this way, the co-occurrence probabilities can be used to find out some meaningful information about how words are related. The main usage of these probabilities

is to calculate ratio probabilities among two different words with respect to a probe word $k$, that gives information about the relationship of the two words. In this way, the relationship between two words can be used to distinguish relevant words from irrelevant words for a certain probe word and to discriminate between relevant words. The ratios are used as starting point for vector learning instead of probabilities themselves. The function that retrieves word vectors using words $i$, $j$ and $k$ takes the form

$$F(w_i, w_j, w_k) = \frac{P_{ik}}{P_{jk}}$$

where $w$ are the word vectors and $k$ is the context word vector.

GloVe outperforms other approaches (i.e., Word2Vec) but it require more computational resources to give significant improvements.

### 2.2.2 Contextualized word embeddings

The models presented in the previous section obtain good results in the definition of the semantic e syntactical relationship among words, but only in simple and shallow contexts. More complex characteristics about syntax and semantics can not be caught. Another issue of *shallow* models is when polysemous words occur. In this case, the embeddings obtained for a polysemous word are the same even the meanings are completely different, disregarding the context in which the word is used. In this way, the contextualized meaning of each word in a sentence is not captured. In order to increase performances and to create a deeper contextualized word representations, new models and techniques have been implemented.

Most of the improvements in NLP, and more specifically in contextualized embeddings, can be attributed to progress of general deep learning. In fact, all models explained in the next paragraphs use deep learning networks. Examples of neural network architectures are Transformer [18], BART [24], ELMo [31] and BERT [21]. Most of these architectures use the *attention* mechanism.

**Attention mechanism** In NLP, attention may be considered as a way to define the importance of words in a sentence with respect to a certain word in the text. For example, if the sentence is *"I am eating a green apple"* and the word *eating* is considered, the focus is to find a food word in the sentence, which is *apple*. The attention mechanism tries to give more importance to this word instead of the others, so it defines a value for each word, and the higher the value, the higher is the relevance for the word in the starting word context. There are several attention mechanisms that are used in Machine Learning and are chosen according to the task. The one used in Transformer model [18] and in most of other models explained in next sections is the self attention, which relates different positions of the input

sequence to the current word in order to compute the correlation between it and the previous part of the sequence. Instead of performing a single attention function, obtaining a $d_{model}$-dimensional vector, the result is obtained projecting the input values in $h$ different dimensions. The $h$ $d_v$-dimensional vectors are concatenated and projected, resulting in a final $d_{model}$-dimensional vector. This attention mechanism is called multi-head attention and gives information by different representation sub-spaces at different positions.

### Transformer

Transformer is a neural network architecture presented in [18] that combine sthe benefits of sequential models such as LSTMs and RNNs. It is able to model long term dependencies and use them during the training phase efficiently. Another improvement made by Transformer is the computational parallelization, while previous approaches preclude it for their sequential structure. The Transformer architecture is simpler than previous approaches and it is based on *attention* mechanisms.

It is an encoder-decoder architecture model that uses attention mechanism in order to model a representation of the entire sequence to the decoder at once instead of a sequential representation as in RNN. This is the main difference between this approach and RNN approach, because rather than building a single context vector starting from the hidden state, it creates a connection between the context vector and the source input.

The encoder is composed as a stack of six identical layers. Each layer has two sub-layers, which are a multi-head self-attention mechanism and a position-wise feed-forward network. Each sub-layer uses a residual connection and a layer normalization. As result, each sub-layer computes:

$$LayerNorm(x + Sublayer(x))$$

where $x$ is the input value. The output produced by each sub-layer is a 512-length vector.

The decoder has six identical layers as well. The basic composition is the same as the encoder, but the decoders adds another sub-layer, which performs multi-head attention over the output of the encoder stack. Also this layers use residual approach followed by layer normalization. The self-attention mechanism is quite different from the previous one, because it is masked in order to avoid to attend to subsequent positions. In this way, the $i^{th}$ word prediction depends on the $i-1$ previous outputs. The entire model architecture is shown in figure 2.6.

The Transformer outperforms previous approaches in several tasks, such as machine translation, summarization, but also in language modeling. In fact, it is able to model properly a language, capturing deeper details about words and

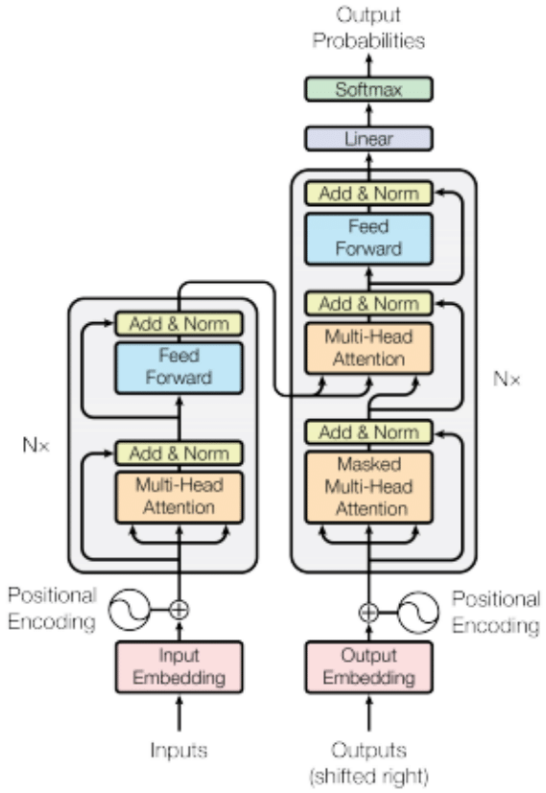


**Figure 2.6:** Transformer architecture [18]

their relationship, and it creates embeddings that are more accurate than previous approaches and that are defined according to the context.

### ELMo

Embeddings from Language Models representations, or ELMo representations, are deep contextualized word representations that model both complex characteristics of word use, such as syntax and semantics, and linguistic contexts. The deep contextualized word representation model that comes from [31] is used in several NLP tasks, such as question answering, textual entailment and sentiment analysis. The language model, presented in [31], derives embeddings from a bidirectional LSTM training on a large text corpus. It is trained in order to predict the next word in a words sequence, a typical approach used in Language Modeling tasks. It uses all the internal layers states in order to define the word representation and linearly combines them. It is demonstrated that higher-level states capture higher-level aspects of words, meanwhile lower-level ones model properly aspects of syntax. In

this way, ELMo word representations are functions of the entire input sequence and captures more details about it. Representations are computed using two bidirectional language models, which combine both forward and backward language models and and model the probability of token $t_k$ using history tokens, future tokens and contexts. Using this approach, ELMo looks at the entire sentence before assigning an embedding to a certain word. ELMO produces accurate embeddings and because of this it is applied to several NLP tasks. The main advantage of ELMo approach is the use of both previous and future context, but the use of LSTMs implies the related problems about the computational costs and the lower results obtained in comparison with attention-based approaches.

### BERT

Bidirectional Encoder Representations from Transformers, or BERT, is a language representation model designed to train deep bidirectional representations from unlabeled data using both left and right context in all layers. Once trained, BERT model can be fine-tuned on a specific task adding an additional output layer on the stack. The model obtains state-of-the-art results in eleven NLP tasks, becoming the de facto standard algorithm in most of them. The approach proposed in [21] uses the *masked language model* (MLM) pre-training objective. Using this technique, the model randomly masks some tokens from the input, and the goal is to predict the original masked words based on their contexts. This approach fuses completely left and right contexts, leading to a bidirectional Transformer. Another technique used during BERT training is the *next sentence prediction*, which significantly improves the context representation. According to [21], BERT framework is trained in two separated steps: pre-training and fine-tuning steps. During pre-training, the model is trained on unlabeled data over several tasks, then during fine-tuning, labeled data are used according to the defined task. In this way, the pre-trained parameters and the main architecture are the same for each task, and the proper fine-tuning model is applied. The model architecture is a multi-layer bidirectional Transformer encoder based on [18], made by 12 layers and a 768-length hidden state vector. BERT model hugely outperforms other contextualized models and becomes the standard model in most of NLP tasks. It is also used as starting point of several next researches, such as RoBERTa([32]), XLM-RoBERTa([33]), SBERT([34]), MobileBERT([35]), multilingual BERT ([21]) and ALBERT([36]). All of them make adjustment of base model according to specific tasks and use cases, based on computational costs, memory requirements and used languages.

Figure 2.7 shows main differences among Transformer, BERT and ELMo architectures. BERT structure is identical to Transformer one, but BERT use both previous and future tokens and exploited them in the training phase according to the masked language model objective, meanwhile Transformer masks off future

data. Both ELMo and BERT are bidirectional, so use left and right contexts, but ELMo implements it using two separated unidirectional networks.

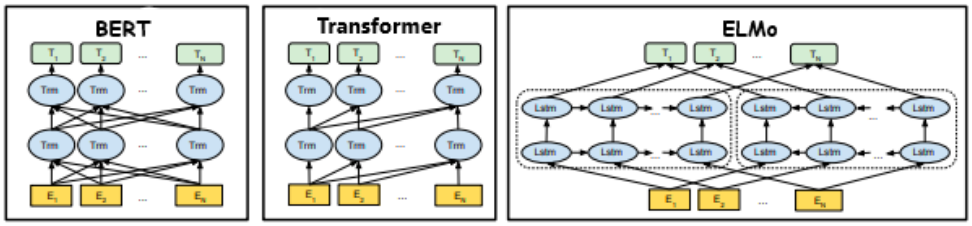


**Figure 2.7:** Differences among BERT, Transformer and ELMo modeling architectures

### BART

BART is a denoising autoencoder presented in [24]. The architecture is a standard sequence-to-sequence Transformer except for the modified activation function (GeLU instead of ReLU). The base model encoder and decoder have the same layers of Transformer architecture. BART model is related to BERT model as well, because it uses a similar approach during the training. The main differences from BERT are the addition of a cross-attention over the final hidden layer of the encoder and the removal of the feed-forward network used by BERT before the word prediction. The model is trained by corrupting text with a noising function and learning to reconstruct the original text. The noising function used in [24] includes token masking (random tokens are replaced with mask tokens), token deletion (random tokens are deleted and the model decides which positions are missing inputs), text infilling (a random number of consecutive words is replaced with a single mask token), sentence permutation (the document is divided in sentences then, they are shuffled) and document rotation(a token is selected randomly and the document is rotated so that it begins with it). Transformations applied by the noising functions are represented in figure 2.8.

BART can be used for several tasks such as sequence classification, token classification, sequence generation and machine translation.

BART, BERT and Transformer models do not retrieve directly contextualized word embeddings, but their encoders retrieve word embeddings that can be used to produce sentence embeddings. A sentence embedding summarizes the whole sentence given in input. If the sentence embedding is given as input to a generative model or to a decoder, then the result is a generated sentence. The sentence embedding encodes the meaning of the whole sentences and contains information

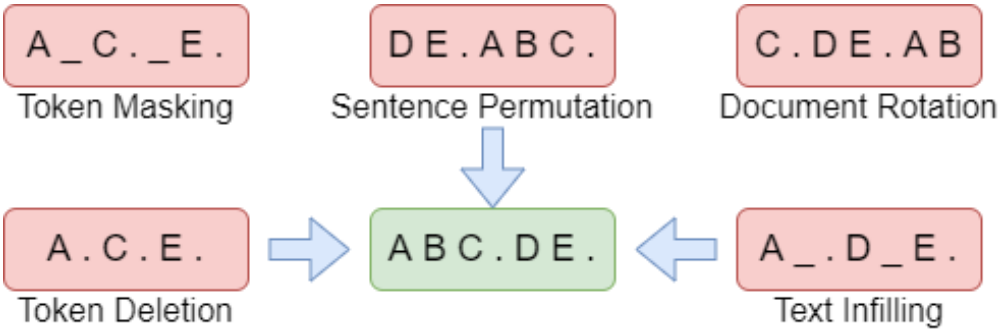


**Figure 2.8:** Transformations applied in noising function [24]

about the context in which each word is said. This type of embedding is useful in text alignment and in neural machine translation branches, that will be explained in the next sections.

### 2.2.3 Word and Sentence Alignment

Word alignment is a task of Natural Language Processing. It is a branch of Machine Learning applied on Natural Language Processing that is focused on aligning embeddings obtained for words in different languages with the same meanings and contexts. The problem is that the embeddings given by two different languages models on input sentences with the same meaning are not equal or similar in most of the cases. As instance, the embeddings obtained by an Italian model that takes the sentence "Io mangio una mela" as input are not similar to the ones obtained by an English model that compute the embeddings of the sentence "I eat an apple". This is due to the unsupervised learning used in all language modeling architectures, because there isn't a perfect starting value for a word embedding, so architectures learn how to model correctly a language without having a ground truth or a reference point that should be achieved. This causes zero relationship among models, which know a language but don't know how to model other ones and how to create a link between them. So the goal of word alignment is to find a technique to get a sentence embedding in a source language and obtain the corresponding sentence embedding in a target language that is as close as possible, and this is obtained by performing some operations. There are several techniques to perform word and sentence alignment, such as pseudo-multi-lingual corpora-based methods, that use monolingual word embeddings models and construct a corpus made with words from both source and target languages, or joint methods, that take parallel text as input and learn both source and target languages, or mapping-based methods, that take models trained on a single language corpus and a multilingual corpus is used to create mapping functions from a language model to another. An

example of joint method is multilingual BERT([37]), a BERT model pre-trained from monolingual corpora in 104 languages. The results of analysis made in [37] show that the model is robust and is able to generalize cross-lingual dependencies, but it is not able to learn specifically each language or a subset of them. Indeed this model is able to create a generic language vector space, but some specific language-related structures have not been captured.

The same task is addressed in sentence alignment. In this case, models produce an embedding that summarize the input sentence. Sentence alignment aims to make embedding for sentences with same meaning as similar as possible, in order to produce a shared vector space for all languages. Since sentence embeddings are defined using word embeddings, this task leads to word alignment too, including a more robust and fine-grained alignment at several levels of detail.

In the following section, the basic approaches and the methods compared to the thesis one will be explained. Most of the them belong to mapping-based approach because they reach outstanding results and because some of them use an approach that has been exploited as starting point in this thesis work.

### Word Alignment Techniques

A research published in 2013 in [38] explains the first technique aiming to align languages vector spaces. In this research, the goal is to use vector space representations of words and phrases of several languages, which define the models of languages, and to learn a linear projection between them. At first, the monolingual models are built, then a small bilingual dictionary is used to learn a linear projection between two languages. The projection is defined for each source-target embedding vectors pair, and the formulae is:

$$z_i = Wx_i$$

where $z_i$ is the approximated target language word vector, $x_i$ is the source language word vector and $W$ is the projection matrix. Applying the projection and obtaining the most similar word vector, the result is the translated word embedding in the target language. In order to increase the approximation, matrix $W$ values are trained in order to optimize

$$\min_W \sum_{i=1}^{n} ||Wx_i - z_i||^2$$

The idea of the projection came from the similarity of word vector representations of similar words in different languages obtained using PCA reduction, so the projection is a simple method that computes the source language word representation in the target language space in the shape of a word embedding. The technique was applied

on monolingual Word2Vec([28]) models trained on large corpus. The main problems of this approach are the need of a aligned bilingual dictionary, which is a limitation for the large amount of languages, and the need of an already trained model for each language,which is a limitation for languages with few data. Another issue related to this approach is the use of a translation matrix that is not orthogonal, and this may cause inconsistency and inaccuracy. This method was used as starting point of most of subsequent researches.

An example of improvement is the model defined in [39]. It is based on [38] implementation, but it improves some of its aspects introducing the normalization of word vectors, so all vectors are located on a hyper-sphere and the inner product falls back into it. In this way, the cosine distance measurement is more accurate than the one obtained in [38]. Another constraint that improves [38] was the use of an orthogonal transform matrix, so the normalization constraint on word vectors is respected.

An issue that still remains unresolved is the hubness problem. The methodology proposed in [40] aims to resolve this issue. Unsupervised methods that learn mapping functions from a source space to a target one associate the predicted approximated target word to the nearest neighbour vector in target space. The neighbourhoods are polluted by hubs, which are vectors that tend to be near to an high portion of items. In order to reduce the effect of hubness problem, pivots elements are defined as a set of vectors whose neighbours are retrieved, and target elements are defined as vectors which retrieve their neighbours. Target elements neighbours are ranked by downplaying the importance of elements that have a high hubness score, defined taking into account the number of times the word occurs in k-nearest neighbour lists of pivots.

Another improvement was made by authors of [41] research, which proved that linear transformation between vector spaces should be orthogonal and that the singular value decomposition (SVD) technique gives better results and a more robust approach. This technique also forms sentence vectors by summing and normalising over word vectors composing it and applying the SVD to sentence vectors. This demonstrates to improve alignment of the underlying word vectors. The method requires a dictionary of paired vectors and it is used to infer the orthogonal transformation.

The work proposed in [42] increases the accuracy of the mapping among language spaces. The dimension of bilingual dictionary is significantly reduced, and this increases usability for most language pairs. This methodology needs a 25 pairs vocabulary to initialize mapping among spaces. With the learned mapping, the dictionary is enlarged with new pairs obtained during the mapping itself. Using this self-learning fashion, the method exploits similarity of independently trained embeddings. Instead of taking a seed training dictionary with thousands pairs, learning mapping using it and evaluating the goodness on a evaluation dictionary,

this method uses a starting seed dictionary containing tens of pairs and uses it to produce its own output dictionary, considering it better than the original one. Once the new dictionary is generated, this method is iterated to produce a more accurate dictionary until the convergence criterion is reached. During each iteration, the generated vocabulary is used to increase alignment among embeddings, and it is considered as ground truth. This approach aims to obtain a better mapping based on real learned data and uses few starting information, allowing mapping among several languages pairs with few translation information. It is applied to pre-trained monolingual models such as Word2Vec([28]) and improves results of previous techniques.

Another approach that resolves the alignment task was presented in [43] and aims to completely remove the supervision in cross-lingual mapping. The approach takes inspiration from generative adversarial networks introduced in [10]. The model starts taking monolingual word embeddings trained separately on two languages and learns a mapping function $f$ which should find the nearest target embedding starting from the source one. The mapping function $f$ is learned and applied by the generator $G$ on input vector $x$. The generated value $f(x)$ is passed to the discriminator $D$, a binary classifier that tries to distinguish between fake values $f(x)$ and real values $y$. This approach does not require translation pairs to supervise the training and improves the usability of mapping for languages with few or no translation data. The method only requires monolingual models in order to align embeddings, so low resource languages models are not so accurate, but this is a problem in previous approaches as well. In order to increase accuracy and correctness of transformations, self-consistency constraint is required. It implies that if the source language embeddings are transformed in target ones using matrix $G$, then $G^T$ matrix should transform from target to source language embeddings. An additional reconstruction loss is added to measure the correctness of reconstruction of original embedding and is defined as:

$$L_R = -\cos(x, G^T G x),$$

where $x$ is the source language embedding and $G$ is the transformation matrix. This loss adds a method to control the capability of both translation and reconstruction because $G$ is updated to achieve a more accurate mapping. The model produce in [43], such that lots of upcoming researches, exploits the adversarial approach, as MUSE([22]) does.

MUSE model, presented in [22] merges most of the previous explained approaches in order to increase accuracy in unsupervised translation without using cross-lingual annotated data. The model uses two large monolingual corpora, one in source and one in target language, and learns a mapping function between the two embeddings spaces operating in two steps. The first step uses a player game approach similar to [43], with a discriminator that learns to distinguish between mapped source and

target embeddings, and a generator that tries to learn the best possible mapping in order to fool the discriminator. The mapping function uses an orthogonal matrix and is defined using the single value decomposition. Once it is done, the second step starts. In this phase, a dictionary is extracted from the resulting shared embedding space and it is used during the mapping fine-tuning, performed with the Procrustes solution. A Procrustes problem is a method which can be used to find out the optimal rotation and/or reflection of an object with respect to another one. The most frequent and best dictionary pairs are used as anchor points for Procrustes, which refines the mapping parameters. Finally, less frequent words performances are increased by changing the metric of the space. The previous steps are iterated until some convergence and stopping criteria are reached. A new unsupervised selection metric is introduced to define the mapping quality and to define the stopping criteria. To mitigate the hubness problem, a cross-domain similarity adaptation is introduced. In [22], source and target embeddings are trained using FastText model ([29]). The model obtained state-of-art results in word translation, cross-lingual semantic word similarity and sentence translation.

### Sentence Alignment Techniques

The approaches introduced in the previous sections use methods that exploit word-level details to align different languages. In this section, sentence alignment approaches are explained. These approaches lead to an alignment among languages taking into account more complex characteristics of words in languages, such as syntax, semantics and linguistic contexts.

A model that strongly improved alignment and multilingual sentence embedding is the model introduced in [44], called LASER. LASER is a joint learning model that learns and generates multilingual sentence representation for 93 languages written in 28 different scripts. The model uses a BiLSTM encoder with a shared vocabulary for all languages. It is language agnostic and is trained on a concatenation of all training corpora, so it has no explicit information or signals about the input language. This encourages the model to learn language independent representations. It is coupled with an auxiliary decoder trained on parallel corpora. The decoder takes a language ID embeddings that specifies the language that should be generated. The language ID embedding is concatenated to decoder inputs and to sentence embeddings. English annotated data are used to learn a classifier on top of resulting embeddings, so the English information can be transferred on other languages without modification. Composition of LASER model is shown in figure 2.9.

In [44], LASER is evaluated on XNLI task, cross-lingual classification, bitext mining and similarity search. The results outperform previous approaches and gives a more robust approach. It also gives a way to learn more languages together and to transfer information from a rich annotated language as English to a poor

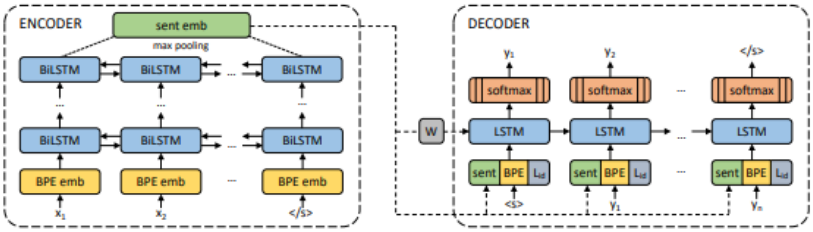


**Figure 2.9:** LASER architecture, [44]

annotated one, such as Hindi or some Chinese dialects. The models trained using this method achieve state-of-art results in sentence alignment tasks, and some of its techniques gives the basis for this thesis work. While LASER works well for identifying exact translations in different languages, it works less well for assessing the similarity of sentences that are not exact translations. This problem occurs because the model is trained in order to face alignment task and obtain the most aligned model, without caring of the similarity among non-parallel sentences.

Significant improvements in the approach used to resolve task are reached in [45]. The presented method extends sentence embedding models to new languages, creating monolingual models that map sentences to the same location in vector space. More precisely, an original monolingual model, called teacher model, is used to generate sentence embeddings and a new system, called student model, is trained miming the original model in order to generate sentence embeddings that should be as close as possible for the same sentences. The model built in [45] requires few sample sentences in the addressed languages, low hardware requirements and is extendable to more than 400 languages. The model is trained using Sentence-BERT ([34]), which is a BERT model with a mean pooling layer applied on the output that produce sentence embeddings.

A teacher model $M$, which maps sentences from a source language $s$ to a dense vector space, is required. Then, a parallel translated vocabulary is needed, in order to define a paired sentence set $(s_1, t_1), ..., (s_n, t_n)$, where $t_i$ is the $i^{th}$ translated sentence into target language $t$. A student model $\hat{M}$ is trained such that $\hat{M}(s_i) \approx M(s_i)$ and $\hat{M}(t_i) \approx M(s_i)$. The mean-squared loss over a batch $B$, defined as

$$\frac{1}{|B|} \sum_{j \in B} [(M(s_j) - \hat{M}(s_j))^2 + (M(s_j) - \hat{M}(t_j))^2]$$

is minimized in order to achieve the training objective. As it is, the student model $\hat{M}(s_i)$ learns the representation of the teacher model $M$. The training steps are represented in figure 2.10.

This training approach has the advantage that an embedding model can first be

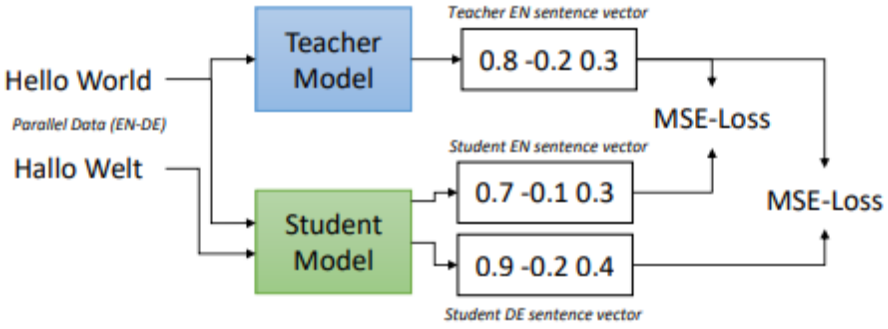


**Figure 2.10:** Distillation training method, [45]

trained on an high-resource language, then it can be extended to support further languages or to extend low-resource language embeddings using few aligned data. During [45] experiments, an English SBERT and XML-RoBERTa models are used as teacher and student models respectively. The models trained using this method achieve results that outperforms similar techniques, but don't reach LASER ones. The main difference with LASER is the usage of data during the training and the approach used to align models. In fact this technique is not completely focused on producing a language-agnostic model, but it wants to produce a shared vector space among languages, so that each language model can be aligned to it. Moreover, this approach has outstanding results even if it does not reach the state-of-the-art, and this is because it can reach high performance in both alignment and similarity search task. LASER, although, at he time of writing has state-of-the-art in aligment, but its performances in similarity search task are not comparable to [45] ones. Distillation approach is the one used as starting point in this thesis work.

The last alignment method presented is defined in [46] and proposes a contextual version of word retrieval to evaluate the degree of alignment. At first, multilingual BERT is taken and is pre-trained on sentences from Wikipedia, written in 104 languages, performing masked word prediction and next sentence prediction tasks. The next step is the definition of contextual alignment. Given two languages, two models are contextually aligned if they have similar representations for word pairs within parallel sentences. A parallel corpus $C$ containing $N$ parallel sentences is defined as $C = (s^1, t^1), ..., (s^N, t^N)$. Each sentence pair $(s, t)$ have word pairs denoted as $a(s, t) = (i_1, j_1), ..., (i_m, j_m)$, where a tuple $(i, j)$ define $s_i$ as the translation of $t_j$ and vice versa. The contextual alignment of a model $f$ can be defined as the accuracy in contextual word retrieval, and the formulae is

$$A(f; C) = \frac{1}{N} \sum_{(s,t) \in C} \sum_{(i,j) \in a(s,t)} \mathbf{1}(neighbor(i, s; f, C) = (j, t)),$$

where **1** represent the indicator function and *neighbor* function retrieves the most similar word to the $i^th$ word of source sentence $s$ in the target sentence $t$:

$$neighbor(i, s; f, C) = \underset{t \in C, 0 \leq j \geq len(t)}{s} im(f(i, s), f(j, t).$$

The same procedure is applied in the other direction, and the average between the two directions is used to define the degree of contextual alignment. Alignment accuracy is encapsulated in the loss function, but the squared error loss is applied instead of cosine similarity loss (CSLS):

$$L(f; C) = - \sum_{(s,t) \in C} \sum_{(i,j) \in a(s,t)} ||f(i, s) - f(j, t)||_2^2.$$

The approaches that mostly contribute to thesis proposal and that give points of comparison are [22], [43], [44] and [45]. Both [22] and [43] use adversarial approaches during the learning, which is the focus point of thesis method. [44] uses and produces an encoder-decoder model, which is the expected output structure of thesis work. [45] Approach uses sentence embeddings distance as focus in order to achieve multilingual alignment. The method proposed exploits the same distance to improve sentence alignment and to produce aligned language models.

## 2.3 Computer Vision and Domain Adaptation

One of the most important contribution to the method proposed in this thesis comes from an approach of Computer Vision field, that is CycleGAN model, used to address domain adaptation task. In next section, computer vision and domain adaptation will be briefly introduced. The main approaches to domain adaptation will be presented, and CycleGAN will be deeply explained.

### Computer Vision

Computer vision is a machine learning branch which makes computers gain high-level understanding from images and videos. Computer vision includes methods for acquiring, processing, analyzing, understanding and extracting high-dimensional data from digital images. Main computer vision models are built using convolutional neural networks, which are formed by several convolutional layers followed by some fully connected feed-forward layers. Each convolutional layer performs three operations, that are convolution stage, sliding filter activation stage and pooling stage. Convolution stage processes images and extracts features using filters, which contains the trainable weights of the neural network. Each convolutional layers contains several filters, and a feature is extracted by each of them. An activation function is applied during the activation stage simulating biological activation to

input stimulus and providing non-linearity to computation. The last operation is pooling, which performs feature down-sampling using sliding filters summarizing statistics of nearby outputs. According to the task, the output of convolutional layers may be given as input to some fully-connected layers. The convolutional layer output, which is represented as a matrix, is flatted down in a row vector, that is used as fully connected layers input. A typical convolutional layer structure which performs image classification is shown in figure 2.11.

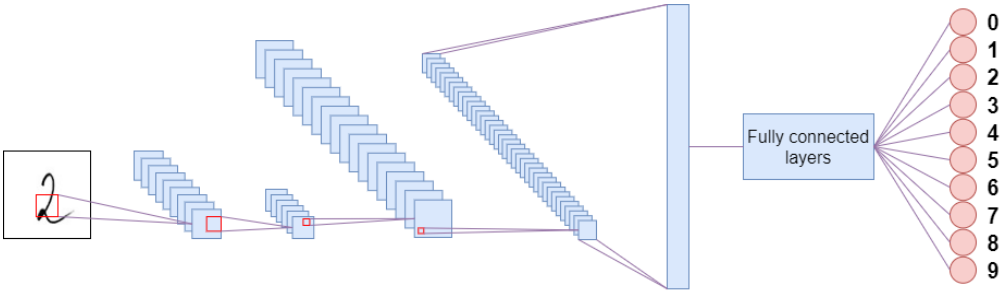


**Figure 2.11:** Convolutional Neural Network architecture

Computer vision include several sub-domains such as scene reconstruction, event detection, video tracking, image recognition, learning, motion estimation and 3D scene modeling. The sub-domain that is taken into account during this thesis is domain adaptation.

### Domain Adaptation

Some particular problems or domains may have few manually annotated datasets, so the CNNs trained from scratch on few data produces inaccurate results. This make the obtained models as unusable. Transfer learning comes to the aid giving a model pre-trained on an huge amount of data, so they may be used as pre-trained networks and can be fine-tuned on the small annotated dataset for the specific task. The problem is not completely solved because the input data may at test time may differ significantly from the training data, and this would lead to overfitting. The reason the model doesn't perform well is that the addressed domain changes, so it is not able to produce an accurate result. Domain adaptation comes to rescue. Domain adaptation is a machine learning sub-discipline. It deals with scenarios in which a model trained on a source domain or on a distribution of them is used in the context of a target domain that differs from the ones used in the training. In general, domain adaptation uses data from a source domain, that contains highly annotated data and in which the model achieves high results, to solve tasks in target domains, in which the amount of annotated data may be lower or the model has unacceptable results. This is useful in domains with few annotated data and

increases results accuracy in them. This is the reason why this task is helpful in thesis work. In fact, an highly annotated source language, such as English, can increase results in few annotated languages, such as Hind or some Chinese dialects.

Several approaches to domain adaptation were defined. At first, shallow algorithms were applied. A kernel function maps data from a source to a target domain. This method, presented in [47], is called Transfer Component Analysis and learns a low rank empirical kernel mapping. A geodesic flow kernel is defined in order to describe the entire path followed during the mapping from a source to a target subspace. The results obtained using shallow models are not so accurate, so deep domain adaptation approaches were defined.

### Domain-Adversarial Neural Network

The first deep adversarial domain adaptation is called Domain-Adversarial Neural Network, or DANN([48]). The neural network is expanded with a new branch made by fully connected layers with an output layer that produces two output values. The output values are used to define if the input value is from the source or target domain, and the resulting prediction is called domain label. So the input goes through the first part of the network, called feature extractor, which is composed by the convolutional layers of the network. Once the output of convolutional layers is computed, the intermediate output is forwarded through the two branches composed by fully connected layers, which are called label predictor and domain classifier respectively. The first one produces the task output and the loss $L_y$ is defined according to the task, then it is back-propagated through label predictor and feature extractor layers. The second branch generates the domain label, which is used to define the domain loss $L_d$, so that loss is back-propagated through domain classifier layers, then it goes through a gradient reversal layer and finally in the feature extractor layers.

### Adversarial Discriminative Domain Adaptation

Another deep approach is Adversarial Discriminative Domain Adaptation, called ADDA ([49]). This technique tries to achieve domain adaptation by using adversarial training. At first a source encoder CNN is trained on source data. Once the source encoder CNN is trained using annotated source images, adversarial adaptation is performed by learning a generator, composed by the target encoder CNN, to produce data such data a discriminator should not be able to distinguish among encoded source data and target data. The discriminator, which predicts the domain label of data, should not be able to distinguish data from different domains. According to the discriminator prediction, the target encoder updates itself to increase the accuracy. In testing phase, target images are mapped with the target

encoder to the shared feature space and the classification is performed using the source pretrained classifier. ADDA techinque is represented in figure 2.12.


**Figure 2.12:** ADDA [49] and CoGAN [50] architectures and training approaches.

### CoGAN

The CoGAN model [50] uses a similar approach, using two generator-discriminator pairs, one for the source and one for the target distribution. Some generators and discriminator weights are shared to learn a domain-invariant feature space. Using this approach, labeled target data can be generated and can be used in tasks such as classification. CoGAN structure is represented in figure 2.12.

### CycleGAN

The last approach explained, and the one which gives the bases to the thesis implementation, is the CycleGAN. CycleGAN, presented in [23], is a model implemented in order to resolve the image to image translation, a task in computer vision where the mapping between an input image and an output image is learnt using a training set of unaligned pairs. It learns to translate images from a source domain $X$ to a target one $Y$ without paired example, and this is one of the most important features of this approach. In fact, not all the image domains contain enough data to build a reliable and resilient model, and the amount of paired datasets is even lower. In this way, even if a domain does not have aligned data with another domain, the model is still able to address domain adaptation task.

A mapping function $G : X \rightarrow Y$ is learned using adversarial loss to make distribution of $G(X)$ images as indistinguishable from the distribution $Y$. An inverse mapping function $F : Y \rightarrow X$ is applied on a input image $y$ and translated image $F(y)$ is generated. A cycle consistency loss is introduced to enforce $F(G(X)) \approx X$ and $G(F(Y)) \approx Y$ constraints. The mapping functions are learned using adversarial loss, so generators $G$ and $F$ learn mapping function and inverse mapping function respectively. Two adversarial discriminators $D_X$ and $D_Y$ are introduced, where $D_X$ aims to distinguish between real images $x$ in domain $X$ and translated images $F(y)$ that should be as close as possible to domain $X$ images. The same is done in the opposite direction, so $D_Y$ aims to discriminate between $Y$ and $G(x)$ images. The architecture of a CycleGAN is represented in figure 2.13.

Two losses are introduced to reach the objective:

- Adversarial Losses, which are applied to both generators during the definition of the mapping functions. These losses aim to align the generated images distribution to the data distribution in the target domain. The objective of mapping function $G : X \rightarrow Y$ and its discriminator $D_Y$ is computed using a loss function, and its formula is

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_y[logD_Y(y)] + \mathbb{E}_x[log(1 - D_Y(G(x)))]$$

  The same objective is addressed by the mapping function $F : Y \rightarrow X$ and the discriminator $D_X$, and its loss function formula is

$$\mathcal{L}_{GAN}(F, D_X, Y, X) = \mathbb{E}_x[logD_X(x)] + \mathbb{E}_y[log(1 - D_X(G(y)))]$$

- Cycle Consistency Loss, which is used to obtain cycle consistency and to enforce coherence constraints on generators. In this way, for each image $x$ from domain $X$, the translation cycle should be able to bring back to the original image $x$. This leads to a forward cycle consistency defined as $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. Same considerations can be applied to satisfy the backward cycle consistency in the opposite direction, defined as $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. The cycle consistency loss is represented in figure 2.13. The cycle consistency loss is calculated as

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_x[||F(G(x)) - x||_1] + \mathbb{E}_y[||G(F(y)) - y||_1]$$

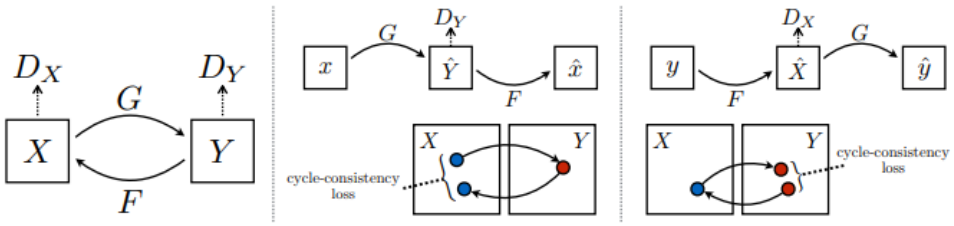


**Figure 2.13:** Model architecture and cycle consistency losses in CycleGAN [23]

The full objective followed during the training can be resumed as

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda\mathcal{L}_{cyc}(G, F)$$

,

where $\lambda$ is an hyper-parameter used to control the relative importance of the two loss types. Both loss types are necessary during the training, in order to introduce

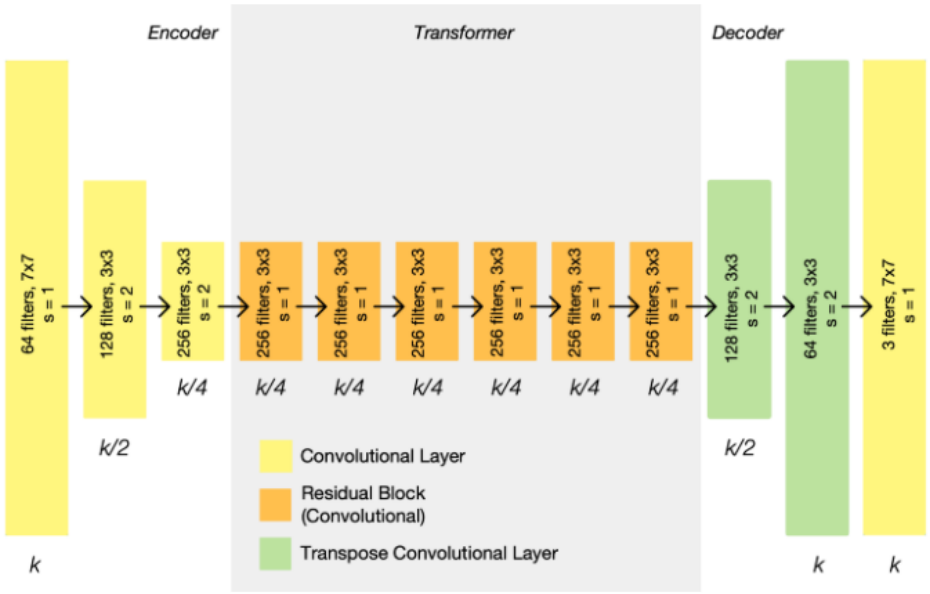


**Figure 2.14:** Generator architecture in CycleGAN model, *retrieved from towardsdatascience.com (last access: June 2021)*

the cycle consistency and because the only cycle loss is not sufficient to let the model converge.

Each CycleGAN generator has three sections: an encoder, a transformer and a decoder. The input image is forwarded through the encoder, and a shrinked representation of the image is obtained. It is passed to the transformer, made of six residual blocks, then it is expanded by the decoder, which enlarge the image and produce the final image in RGB. The generator architecture is represented in figure 2.14.

Discriminators are PatchGANs, fully convolutional neural networks that output the probability of the image being real. PatchGAN are computationally efficient and look at a patch of the image, focusing on a more surface-level features and producing a more effective prediction.

CycleGAN strongly outperforms previous approaches in domain adaptation and image translation, but the main feature introduced is the possibility to use both paired and unpaired datasets during the training. It works well on tasks that involve color or texture changes, or collection style transfer, but tasks that require substantial geometric changes fail. This approach is the core part of the implementation explained in this thesis. In fact, its approach can be adapted in a Natural Language Processing task such as the Neural Machine Translation because of the similarity of the tasks. Moreover, the usage of a cycle consistency loss might

introduce more stability in the translation process, increasing the quality as well.

# Chapter 3

# Implemented methods

This chapter aims to explain accurately the method implemented to address the task proposed for the thesis.

As preliminary work, several word and sentence alignment approaches have been analysed. Most of them exploit mapping-based techniques introducing several improvements, such as matrix orthogonality, hubness problem solutions and singular value decomposition technique, producing a more robust approach to map embeddings from a source to a target domain and vice versa. Their main criticism can be addressed tackling the shallowness of the mapping operation, indeed this leads to a weak understanding of more complex linguistic structures and contexts. Using those architectures, a polysemous word is mapped to the same word in the target space, even if its usage can be deeply different. Improvements have been introduced with [44] and [45] methods, which use deeper approaches to align languages. Both those approaches exploit contextualized word embeddings models, which produces sentence embeddings.

A sentence embedding represents a summary of the whole sentence, so it can be used by a decoder model to construct the sentence in the target space (e.g., the approach proposed by LASER[44]). As result, LASER encoder produces the same embedding for sentences which have the same meaning and are written in different languages. Conversely, this approach produces insufficient results for sentences or words that are not one the exact translation of the other, but have similar meanings. Moreover, the embeddings produced using two sentences in the same language that have similar meanings have weak o no connection between them.

This issue has been addressed in [45], that produces a separate model for each language. Given the knowledge induced into a language model for a given idiom (i.e. English BERT), the embeddings of two similar sentences in the same language have short distance in the embedding space. After the distillation training, language models associated to different languages produce similar representations for sentences which could be similarly translated.

An aspect that can be strongly exploited is the similarity among the characteristics of neural machine translation task in natural language processing and domain adaptation task in computer vision. During years, it happened that natural language processing and computer vision shared some techniques to address similar tasks, but no technique have been shared to address these tasks at the time of writing. The domain adaptation aims to transform an image from a source domain to a target one and the same approach is used in neural machine translation. In fact, in the NLP context, each language can be seen as a separate domain, therefore the translation of a sentence from a source to a target language can be seen as the adaptation of an image from one domain to another. This has led to the adaptation of a computer vision model to the world of natural language processing. Analyzing the most important approaches used in domain adaptation, CycleGAN ([23]) seems to be the most useful for the task proposed in this thesis.

The goal of this thesis work is exploit CycleGAN architectures to obtain a new sentence alignment approach by exploiting the main advantages of the previously explained methods. The model points to align embeddings for sentences written in different languages with the same meaning and jointly to translate sentence among languages. Taking a sentence from the source language, the model should be able to translate it in a target language, and the produced sentence embedding should be as close as possible to the embedding produced for the same sentence written in another language. In this way, a shared space is produced among languages, and each model produces embeddings in that space. Even similar sentences fall close to each other, taking into account context and complex linguistic structures. Starting from a sentence embedding in the shared space, a decoder, which is associated to a specific language, should be able to generate the proper sentence in it.

In next sections, the implemented architecture, called CycleNLPGAN, and each sub part of it, will be explained. The various aspects of training phase will be explained in detail as well.

## 3.1 Main architecture

CycleNLPGAN model is built following the CycleGAN model approach. The model is partially modified in order to create a shared vector space among languages, and jointly allowing it to translate sentences from a source language to a target one and vice versa.

CycleNLPGAN is an architecture made by two different GANs, and each of them carries out the transformation from a language into another. The whole architecture is represented in figure 3.1.

Each GAN is made by a generator $G$ and a discriminator $D$.

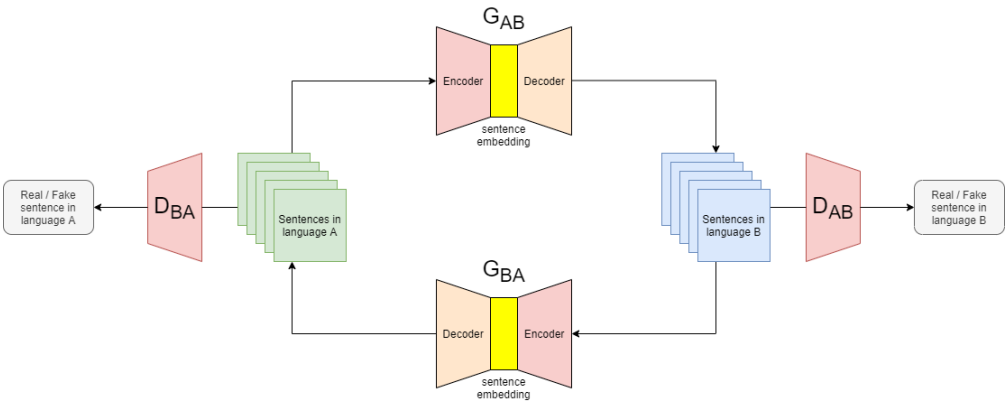


**Figure 3.1:** CycleNLPGAN structure

### 3.1.1 Generator

The generator leans a mapping function from a source language to a target one. Usually, the input of a generator is a random noise, which is expanded to produce a specific value, such as a sentence or an image. The generator of a CycleNLPGAN is quite unusual, because it is made by an encoder and a decoder part, so it requires a sentence as input, which is shrinked by the encoder part and then expanded to produce a sentence in a target language by the decoder part. In this way, the input of the generation phase is no longer a random value, but a summary of the starting sentence. This is the first main difference from a traditional GAN. In fact, the standard generator is made by a decoder-like model, meanwhile each CycleNLPGAN generator contains an encoder too.

Starting from an input sentence $x$, the generator $G$ applies tokenization, encodes the sentence and produces a sequence of word embeddings $e_x$ in a vector space $E_X$. Once $e_x$ are produced, the generator $G$ decodes the sequence and produces a sentence $y_{fake}$ in the target language. Moreover, the $e_x$ embeddings are forwarded through a mean pooling layer, that produces the sentence embedding for each sentence given as input. The sentence embedding is used to let models alignment among them, using techniques that will be explained later in this chapter. The generator structure and its use are shown in figure 3.2.

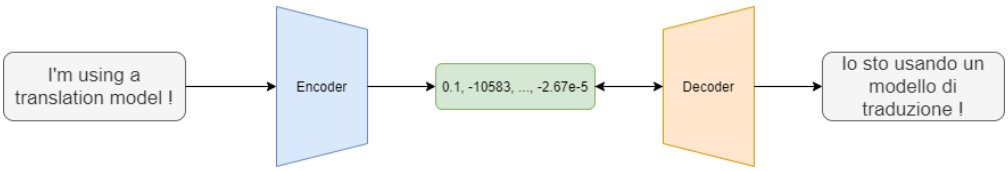


**Figure 3.2:** CycleNLPGAN generator

A MarianMT([51]) model is used as generator, but a generic encoder-decoder model can be used. MarianMT is a BART ([24]) model specifically trained for machine neural translation task. In fact, the whole architecture is defined considering both sentence alignment and neural machine translation tasks. The generator is made by a BART encoder, which produces a sequence of 512-length word embeddings, and a BART decoder, which uses the embeddings to generate the sentence in the target language. The number of produced word embeddings depends on the number of words in the input sentence. Word embeddings are given as input to a mean pooling layer, producing a 512-length summary of the whole sentence, called sentence embedding. This is the same approach applied by default in sentence BERT models, whom are the models exploited in the distillation method implemented in [45].

Each model is able to translate unilaterally from one language to another, i.e. from Italian to English. The reverse operation is made by a different network, trained exclusively on this task. The main disadvantage of this model is the misalignment among the several produced networks, so the model defined for each combination of language produces embeddings in its own vector space, causing no relationships among models trained on other languages pairs or on the same language pair, but in the opposite direction. This leads to the need of training the whole structure to produce a new language-pair translation model, and the same must be done in the reverse direction. Using the approach defined in the thesis, a single training phase is needed to train both translation directions. Furthermore, in order to increase the usability and the flexibility of the model, a shared vector space can be defined, and each $language-to-language$ model will use the shared vector space to produce its own sentence embeddings. In this way, the vector space is not specific to the individual language or a subset of languages, and each decoder will learn to produce embeddings in that vector space that is shared among different languages, making the approach highly reusable. In order to create a shared vector space, some fundamental operations must be executed. At first, each model have the use the same multilingual dictionary, in order to generate the same tokens for a certain word, regardless the source and target languages. At second, following the teacher choice made in [45], a model should be chosen as reference point for other languages, and its vector space is used as reference by other models. Since the part of each generator that produce the embeddings and consequently the language vector space is the encoder, this is the only part of the network that should be considered as ground truth by other languages models. In fact, even in [45], the network that produces embeddings, in that case SBERT[34], is considered as teacher model, and it has the structure of an encoder. In the same way, CycleNLPGAN might exploit the usage of an encoder as teacher, letting other languages to align to it and to produce embeddings in the same vector space. The main advantage of the usage of a shared vector space is that a decoder, trained

to produce sentences in a certain language starting from the shared space, should be able to produce the correct sentence without caring of the source language. This leads to the need, for each language, to have only a single encoder trained to produce embeddings in the shared space and a decoder able to construct a sentence starting from shared space embeddings. A new translation model from a source to a target language can be easily built by joining the source language encoder and the target idiom decoder.

In the thesis work, the shared vector space is not implemented because of the effort needed to implement this aspect. The model used during the training, indeed, has their own vocabulary related to the used language pair. The solution of this problem is the substitution of vocabularies in each model with a multilingual vocabulary followed by an intensive training phase, and this requires an effort that is not affordable and is out of the scope of this thesis. In conclusion, in the current implementation each languages pair produce a vector space that is shared among languages that compose the pair, but it is possible to produce a vector space that is shared among all languages. The decoders extracted by this implementation are able to produce the output sentence from the produced shared vector, without caring of the source language of the starting sentence and of the model that produces the embeddings.

### 3.1.2 Discriminator

The discriminator learns to distinguish between generated sentences $y_{fake}$ and sentences $y$ that actually belong to a certain language $Y$. A DistilBERT for sequence classification model is used as discriminator, but several models can be chosen. In particular, this DistilBERT implementation adds a randomly initialized fully connected layer on top of the model and produces a $n$-length vector as output. The $n$ value depends on the particular task. The objective in this model is to define if input values represent a real or a generated sentence, so two output values are needed. These values represent the probabilities defining whether the sentence is real or not. Applying softmax, a single output value is obtained, which is 0 for fake sentences and 1 for real sentences. The structure of the discriminator and its forward phase are represented in figure 3.3. When possible, DistilBERT trained only on the target language is used, otherwise the multilingual BERT is chosen. This choice is related to the availability of a model trained in the target language, and the presence of a DistilBERT model trained on a single language entails an higher accuracy of the discriminator during first training phase. Using a multilingual DistilBERT instead, it would recognize generic languages structures and would not be specialized during first training phases. However, during training, the models acquires more and more information about the specific target language, producing a fine-tuned DistilBERT.

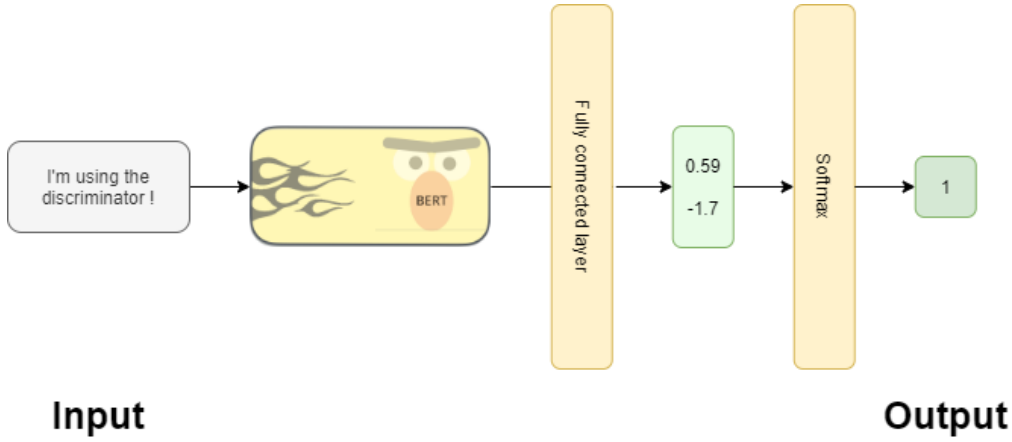


**Figure 3.3:** CycleNLPGAN discriminator

In GANs, the discriminator also plays a fundamental role in the training phase of the generator, as it defines the goodness of the produced data. On the other hand, the generator tries to foolish the discriminator, so it should be robust enough to recognize fake data. If the discriminator gets foolish, it becomes useless and the training should be stopped, or the quality of both networks will be compromised.

In traditional GANs, once the training finishes, discriminators are discarded. In thesis implementation, instead, they are kept and used to evaluate the goodness and the correctness of a sentence. In fact, using the whole model, it is possible to discriminate among well formed sentences and sentences that don't have a correct structure or meaning. Even removing the layer on top of the model, the underlying part is a BERT model that can be used as pre-trained model in other tasks.

## 3.2 Loss functions

Several loss functions are involved in the training of CycleNLPGAN. As in traditional GANs, both generator and discriminator losses are required. A cycle consistency loss is introduced to relate GANs, and more deeply generators. In next paragraphs, each loss function will be explained in detail.

### 3.2.1 GAN losses

As explained in previous section, a CycleNLPGAN is made by two separate GANs, and each GAN produces its own losses. A generator loss is produced to increase the quality of generated data, meanwhile discriminator loss tries to toughen up the discriminator.

#### Generator loss

The generator loss is typically produced defining how the generator is able to confuse the discriminator. Generated outputs are used as discriminator inputs jointly with real data. The discriminator, for each input data, defines if it is fake or real. The more the discriminator defines fake data as real, the more the generator loss decreases. In CycleNLPGAN implementation, a new loss is added to address the translation task. The introduced loss is the masked language modeling loss and is computed using parallel data. In fact, the source-target parallel sentences pairs are used during training, and the target sentences that should be produced are used as ground truth. The loss defines the discrepancy between the generated output and the desired one. The introduction of this type of loss implies the introduction of some supervised techniques in the training, which can be avoided in traditional CycleGAN, and the need of parallel datasets. This may be a limitation is certain languages with few data or with zero parallel datasets, but the robustness achieved using this approach increases. MarianMT implementation contains a built-in loss implementation in order to address translation task, so it is exploited in CycleNLPGAN training. Both traditional generator loss and the masked language modeling loss are used, with equal weight, allowing the model to address GAN task and translation task simultaneously.

#### Discriminator loss

The loss funtion used for discriminator training is the typical GAN loss, which defines how the discriminator is able to distinguish fake data. More in detail, the loss used in CycleNLPGAN model is cross entropy loss and is calculated on prediction results on both generated and real data.

### 3.2.2 Cycle consistency loss

One of the most relevant feature of CycleGAN is the introduction of a cycle consistency loss, and it is applied in CycleNLPGAN as well. This loss is a concept used to create a dependency between two mapping functions with inverted source and destination languages. In a standard approach, each GAN trains by itself, without knowing of the other one. This leads to a lack of relationship between the mapping from source to target domain and vice versa. Using cycle consistency loss in CycleNLPGAN, generators of both GANs are involved in the generation of data and in the backward phase, producing a relationship between them and inducing to the generation of more consistent and interrelated data. Traditional cycle consistency loss defines the quality of both generator to reconstruct the starting value. A sentence $x$ from language $X$ is traduced by generator $G_{XY}$ in language $Y$, producing sentence $y_{fake}$. This value is used as input in generator $G_{YX}$,

which translates from $Y$ language to $X$ one, producing reconstructed sentence $x_{rec}$. In order to evaluate translation goodness and to relate generators, the cycle consistency loss defines the similarity between $x$ and $x_{rec}$ sentences, that should be equivalent or as close as possible. According to loss value, both generators perform backward and try to find a way to produce a $x_{rec}$ sentence to be identical to $x$, increasing the quality of the $y_{fake}$ sentence as well. The same operation is performed in the opposite direction, producing a $y_{rec}$ sentence that should be as similar as possible to starting sentence $y$. This loss drives models to cooperate during the generation of values that are as coherent as possible, creating a relationship between them.

In addition to the traditional loss, a new loss function has been added to strengthen the alignment between the generators vector spaces. In fact, each generator produces its own vector space, and the cycle consistency loss by itself does not ensure the alignment between them. The new introduced loss forces alignment between models using an approach similar to distillation method introduced in [45], where one model teaches the other to produce similar embeddings. Sentence embeddings produced using mean pooling layer are compared and the loss is computed in order to penalize distant embeddings for sentences with the similar meaning or that are one the translation of the other. Cosine embedding loss is used in CycleNLPGAN and gives the main contribution in alignment task.

**A** ll these losses are grouped toghether to induce models to address both alignment and translation tasks, introducing a robustness given by the introduction of supervised steps in the traditional GAN and CycleGAN approaches.

## 3.3 Cycle-consistent process

During each training step, the whole model performs several forwards through the different parts of the architecture. At each step, two batches of parallel data are generated. The first batch $real_A$ contains sentences from language $A$, meanwhile the other batch $real_B$ contains sentences from language $B$.

The $G_{AB}$ mapping function is used to transform from a source to a target language, meanwhile the $G_{BA}$ function maps from target to source idiom. Taking batch $real_A$ from language $A$, the mapping function from $A$ to $B$ language is applied by generator $G_{AB}$ and produces sentences $fake_B$ similar to idiom $B$ sentences. Generator $G_{AB}$ results are used to compute loss value $lossG_{AB_1}$ based on the similarity of generated data to $real_B$ sentences, which are the exact translation of $real_A$ sentences and can be used as ground truth. Taking $fake_B$ as input, the mapping function from $B$ to $A$ idiom is applied by generator $G_{BA}$, which produces reconstructed sentence $rec_A$, that should be similar to $A$ phrases. The

same operation is performed in the opposite direction, so $real_B$ sentences from $B$ language are forwarded in $G_{BA}$ generator, in order to produce $fake_A$ sentences, which are the translation of $real_A$ sentences in the $A$ language, then it is forwarded through $G_{AB}$ generator to produce $rec_B$ sentences. Generator loss $lossG_{BA_1}$ is computed comparing $fake_A$ and $real_A$ sentences, that can be used as ground truth because are the exact translation of $real_B$ sentences. The steps performed in the training phase are represented in figure 3.4.

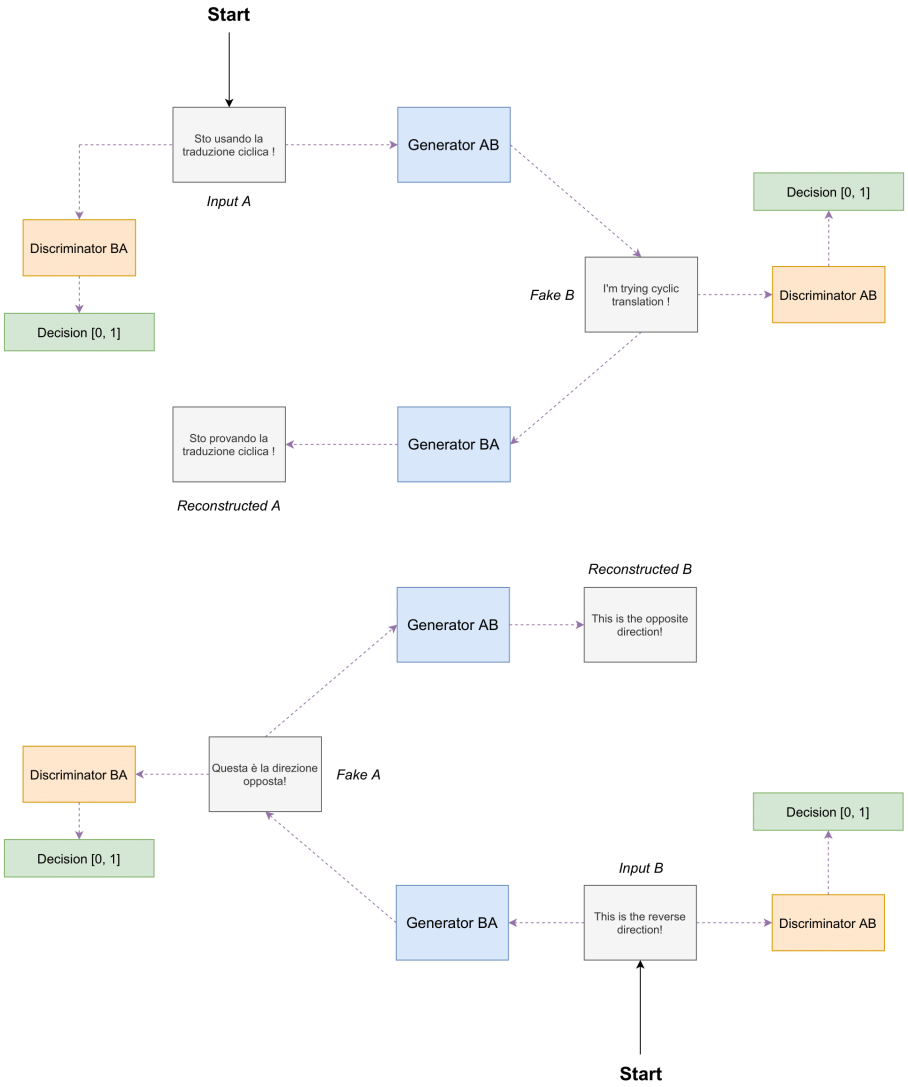


**Figure 3.4:** CycleNLPGAN training phase

Generator $G_{AB}$ takes $fake_A$ sentences as input and produces $rec_B$ sentences,

which should be similar to $B$ sentences, in particular to $real_B$ sentences. Loss value $lossG_{AB_2}$ is computed taking sentences $real_B$ as ground truth and $rec_B$ as generated values. Generator $G_{BA}$ takes $fake_B$ sentences as input and produces $rec_A$ sentences, which should be similar to $A$ sentences, in particular to $real_A$ sentences. Given $rec_A$ sentences and real sentences $real_A$, the loss value $lossG_{BA_2}$ is defined according to their similarity. Each of these losses contributes for a quarter to the total loss value of the generator to which they are related. Loss values $lossG_{BA_1}$ and $lossG_{BA_2}$ are involved in $lossG_{BA}$ loss value, meanwhile $lossG_{AB_1}$ and $lossG_{AB_2}$ are involved in $lossG_{AB}$ loss value.

The remaining half of either generator losses $lossG_{AB}$ and $lossG_{BA}$ is defined forwarding the generated value $fake_B$ and $fake_A$ sentences in $D_{AB}$ and $D_{BA}$ discriminators respectively and defining the generator loss $lossG_{AB_3}$ and loss $lossG_{BA_3}$ respectively. Those values are defined using the traditional GAN approach, so according to the quality of generated sentences.

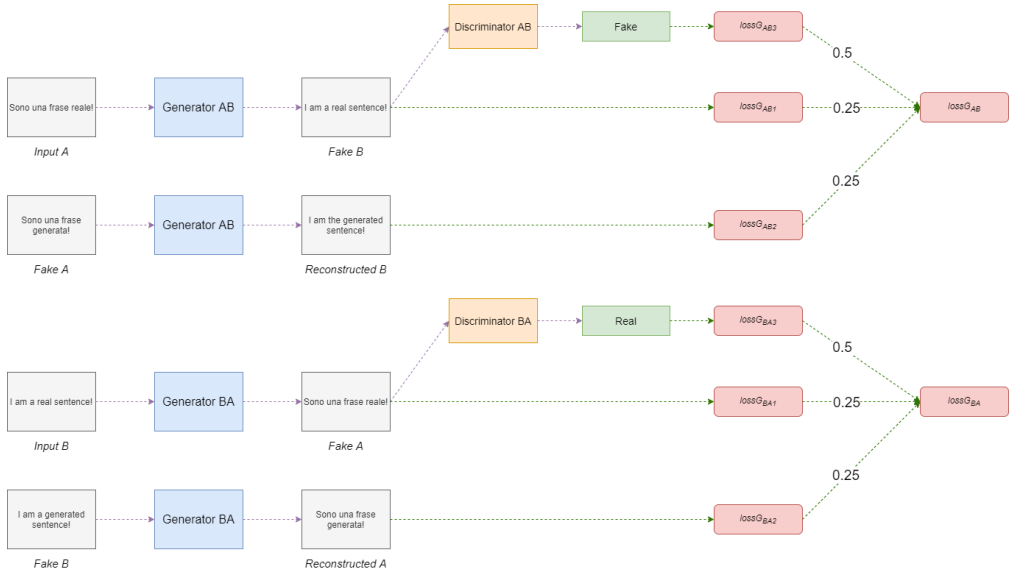


**Figure 3.5:** CycleNLPGAN generator loss

Discriminator loss $lossD_{AB}$ is defined forwarding both real $real_A$ and fake $fake_A$ sentences through the discriminator $D_{AB}$ and evaluating if the predictions are correct or not. The same is done for loss $lossD_{BA}$ in the opposite direction, and it is calculated forwarding both real $real_B$ and fake $fake_B$ sentences through the discriminator $D_{BA}$ and evaluating predictions goodness.

Once GANs losses are singularly computed, cycle consistency loss is jointly calculated. In order to define the cycle loss, combinations of real, generated and reconstructed sentence embeddings are compared in both languages in order to define $loss_{cycle_{ABA}}$ and $loss_{cycle_{BAB}}$. The two losses produced define the error during

the translation and the back translation in both directions. The first one represents the loss translating a sentence from language $A$ in language $B$ and re-translating it in language $A$, meanwhile the latter one defines the error translating a sentence from language $B$ in language $A$ and then re-translating it in language $B$. The sum of these losses produces the cycle consistency loss $loss_{cycle}$ that is back-propagated through both generators. MSE loss is used to evaluate the distance among sentence embeddings. This loss function gives a good measure of the inequality of sentence embeddings, so generators can improve transformation according to the distance among embeddings produced for parallel sentences.

The $loss_{cycle_{ABA}}$ is composed by four different losses. At first, the sentence embeddings of real sentences $real_A$ are compared to reconstructed embeddings $rec_A$, and this loss value is multiplied by the hyperparameter $lambda_A$, which define the weight of this specific loss. This is the most important part of the loss $loss_{cycle_{ABA}}$ because it contains information of either the first and the second translation and the comparison is done using values produced by the same generator $G_{AB}$. In fact, once a reconstructed sentence $rec_A$ is generated by generator $G_{BA}$, its sentence embedding can be computed forwarding the sentence through the $G_{AB}$ encoder, that is the same used to compute the $real_A$ sentence embedding.

The second part of the $loss_{cycle_{ABA}}$ is computed comparing $real_A$ and $real_B$ sentence embeddings. To obtain their sentence embeddings, $real_A$ and $real_B$ sentences are forwarded through $G_{AB}$ and $G_{BA}$ encoders respectively, and the result is multiplied by the hyperparameter $lambda_{C_1}$. This parameter is used to weigh the difference among the embedding produced by $G_{AB}$ in comparison with the one produced by $G_{BA}$ encoder. As explained in next sections, the latter one is used as ground truth, so this part of the loss defines the difference among the produced result and the expected result.

The last two quarters of the $loss_{cycle_{ABA}}$ are defined comparing the embeddings produced for sentences $real_A$ and $fake_B$ and the embeddings produced for sentences $fake_A$ and $fake_B$ respectively. Their values are multiplied by hyperparameters $lambda_{C_2}$ and $lambda_{C_2}$ respectively. Even if these values are not properly related to the cycle process, they are useful to give a quantification of partial results obtained during the cycle process. Moreover, they force the correctness of partial values, that are crucial for the decoding and the back-translation phases.

All those values are summed to achieve the $loss_{cycle}$ value, and each of them has the weight of a quarter of the total value. The $loss_{cycle_{ABA}}$ calculation process is represented in figures 3.6.

As for the loss, the $loss_{cycle_{BAB}}$ value is calculated as the sum of four equal weight losses that are calculated for the process in the opposite direction. As shown in figure 3.7, the second and the fourth part that compose the final value are the same used in the previous case. The first and the fourth part have an approach similar to the previous one, but the values used in the comparison change. More in

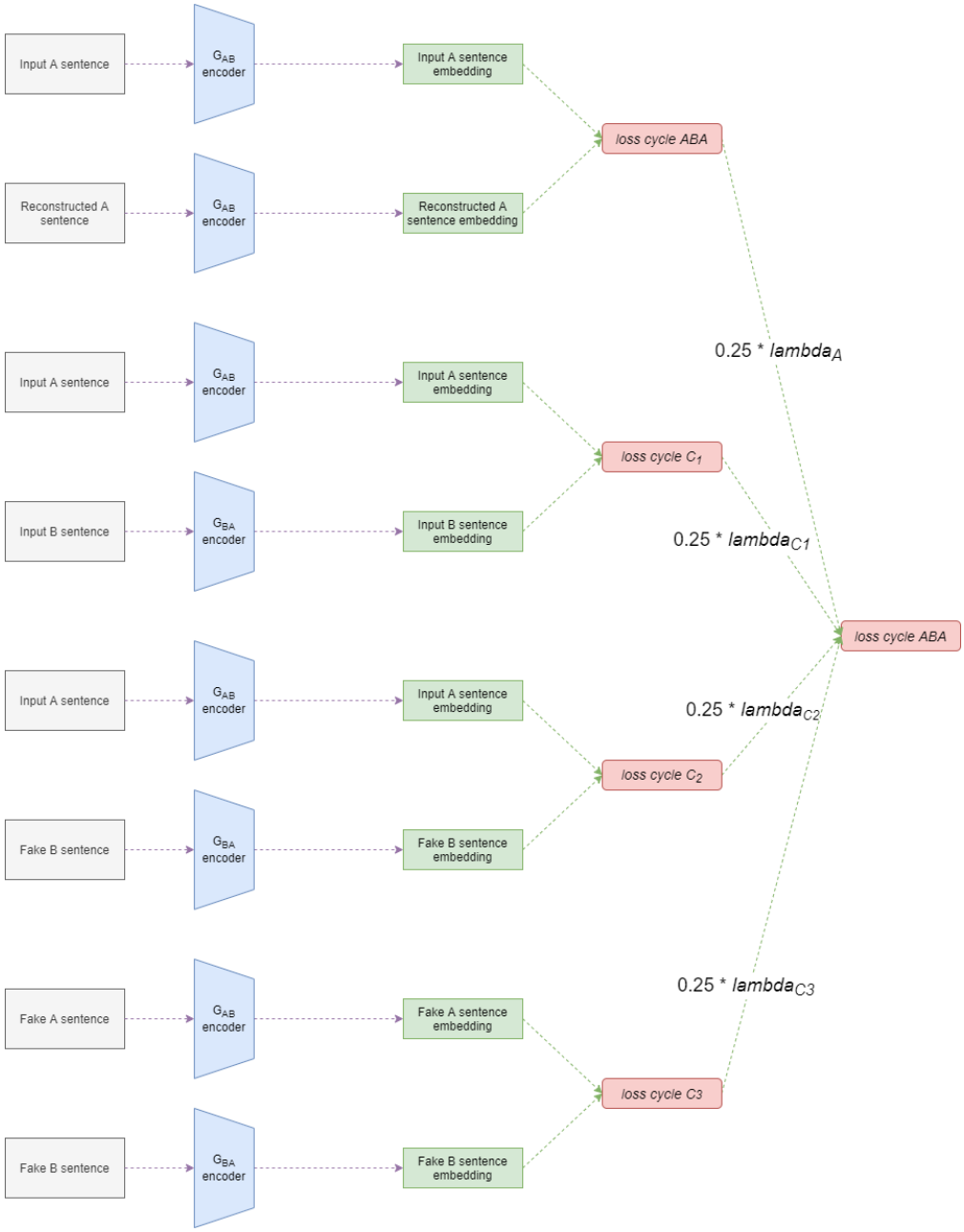


**Figure 3.6:** CycleNLPGAN ABA cycle loss

detail, the first part of the $loss_{cycle_{BAB}}$ value is calculated comparing the sentence embeddings of real sentences $real_B$ and of reconstructed sentences $rec_B$, and this loss value is multiplied by the hyperparameter $lambda_B$. The third part instead compares $fake_A$ and $real_B$ sentence embeddings produced by generators $G_{AB}$ and

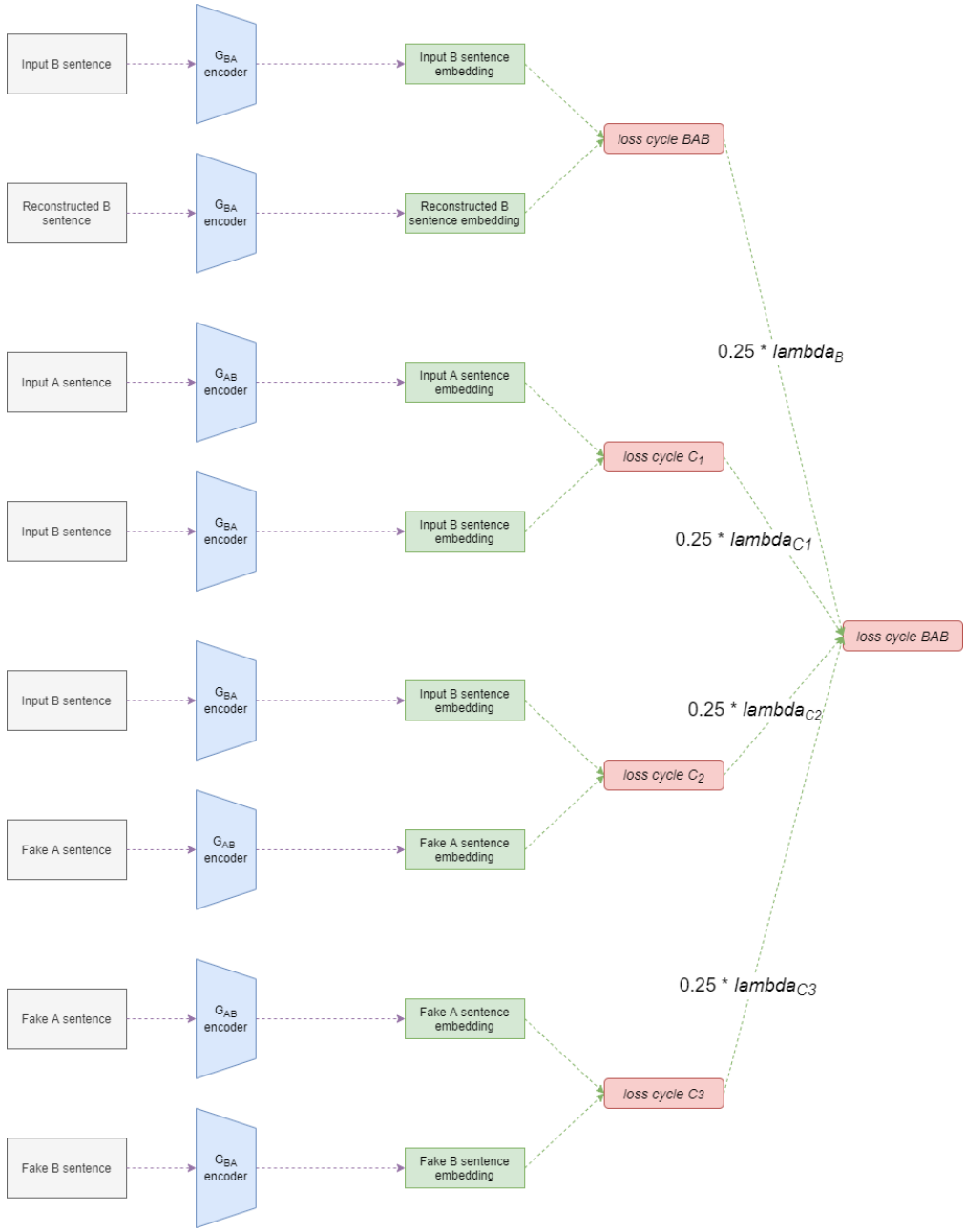


**Figure 3.7:** CycleNLPGAN BAB cycle loss

$G_{BA}$ respectively, and the result is multiplied by $lambda_{C_2}$.

The losses values $loss_{cycle_{ABA}}$ and $loss_{cycle_{BAB}}$ are summed with the same weight to produce the final loss value $loss_{cycle}$. Unlike the previous ones, this loss is back propagated through both generators.

## 3.4 Improvements

We explored several extensions and variants of the previously presented method before reaching the final structure described in earlier stages. The whole structure has been built using the CycleGAN([23]) as starting point, so the idea was to define the same structure in NLP tasks, more specifically in sentence alignment and neural machine translation tasks. The similarity among CycleGAN objective and the one described in this thesis leads to the definition of a model focused on translation task, but this is not the main objective of the thesis. In order to create a model that address alignment task, some attempts have been made to adapt the proposed model to the CycleGAN structure. At first, sentence BERT models have been chosen as generators models. Sentence BERT model produces a sentence embedding, but it does not contain a decoder part, so it is impossible to reconstruct the value and to forward it in discriminators and in the opposite direction generator. The mandatory requirement to correctly create the model is to use an encoder-decoder model. The choice falls on MarianMT model because it is a robust models and addresses source-target translation. Moreover, it is implemented as open source and several pretrained configurations are available. To avoid falling back into translation task, the sentence embedding loss and the new cycle consistency loss are introduced to move on the alignment task.

The model can be trained using every combination of languages, but during the whole project $B$ language is always English, in order to use it as reference.

At first, all modes were trained, so parameters update according to the couple of languages selected, but this leads to an alignment among them only. Changing languages, models are completely different from previous ones, so they aim to align themselves, neither caring of other models or using an external model as reference. Even using an external model as reference, the training is unfair, because this means use an English model as ground truth and forcing both generators in CycleNLPGAN to align simultaneously using English model as reference. This may produce an unstable training, so as result nets may be unusable. Moreover, the resulting architecture would have been more dependent from the reference model than from the CycleGAN structure itself. Another problem using this configuration is the alignment between two specific languages, and none of them is English. This is not the best solution, because it may be useful to align all languages models to an English model due to its popularity, the enormous amount of English data and the robustness of English models. So the idea is to use English model as focal point and each language model tries to emulate it.

Once English has been defined as reference language, another problem occurs, and this is about generated embeddings. Taking $English - to - language_A$ and $English - to - language_B$ encoders, the embeddings produced by their encoders are not equal, even if their source language is the same. After a detailed analysis,

this is because of the difference among the used vocabularies. The models trained on two different language pairs use different vocabularies that are related to the specific pair and contains only words from these languages. Even if two models share one language, it is not ensured that their vocabularies would be aligned. The possible solutions to this issue are:

- The alignment of vocabularies used for different language pairs by MarianMT
- The usage of a fixed $English - to - language_X$ encoder and of a shared vocabulary among all the languages

The first one is automatically discarded because it is a sub-optimal solution and is against the principles of the thesis work. In fact, it aims to align vocabularies more than models vector spaces, and this is not the objective of the thesis. The second solution is the best one because build a resilient and flexible model that can be used for a wide range of languages. The main disadvantage of this approach is related to the necessity of preforming impactful alterations to the starting structure of the MarianMT model, which would lead to an effort that is beyond the one required for this thesis work and that could lead to a more complex structure.

In order to avoid the effort required by the latter solution and to produce an usable model, another solution has been implemented. In this solution, the model is trained to align two languages during each training phase, but the alignment will be limited on the language pair only. Training the same model on a different language pair, the resulting generators will be aligned among them, but they are not aligned to other pairs models. This leads to a partial alignment among languages, that is constrained to the couple of languages used during the training. The proposed solution is implemented freezing the parameters of the encoder that is part of the English-to-other language MarianMT model, so it does not change during training. This choice has been done after several analysis reported in next chapter and introduce an enhanced stability. Moreover, the model as is is usable with models that already use vocabulary composed by several idioms, and this will produce a model that align all languages without the need of alteration to the architecture. Using this implementation, each $language - to - English$ model will use the English vector space as the objective of the training.

# Chapter 4

# Experiments and Results

In this section, details about training data and parameters are given. Moreover, several experiments have been conducted to evaluate results.

## 4.1 Training Data

Training dataset plays a big role in producing an affordable translation model with cross-lingual embeddings. Starting from first phases of design, the need of parallel data became immediately necessary. It seems the opposite assumption with respect to the typical approach used by CycleGAN and cycle consistency models. In fact, CycleGAN does not need parallel data to correctly learn because of the usage of a cycle consistency loss that defines how good is the reconstruction of the image. In this way, it defines how good both generators work and try to correctly fix them.

The same may happen to CycleNLPGAN translation part, but the main objective is to align embeddings produced by the decoder part of the generators, so parallel data are mandatory for the evaluation of the alignment goodness. This may be a challenging approach in low annotated languages, but the robustness of the reference English model may help and produce discrete results. Another problem that may occur using non parallel data is related to the instability of the networks during the training. Some attempts of unsupervised and non parallel training had been made, and as result networks didn't converge, increased their training time and produced inaccurate sentences. In some cases, networks completely diverged due to the instability, so networks generate sentence embeddings that fall into a fixed point and produce meaningless sentences.

Once the typology of dataset had been defined, several datasets and language pairs had been evaluated to correctly address both alignment and translation tasks. The choice fell on some datasets provided by OPUS([52]), a publicly available collection of parallel translated texts from the web. It provides parallel data for

hundred of language pairs.

The OPUS datasets used in this thesis project are:

- **JW300**: Parallel sentences from a collection of magazines [53]
- **Tatoeba:** A large dataset of sentences and translations used as support in language learning [52]
- **TED2020:** A collection of about 4000 translated TED talks available in more than 100 languages [54]
- **WikiMatrix:** Parallel sentences from Wikipedia in several languages [55]. This dataset contains a huge amount of sentences and often they are of bad quality, so only pairs with a quality score above 1.075 are used.

More details about complied languages, total amount of data and number of sentences for the addressed language pairs are shown in table 4.1.

| Dataset Name | Number of Languages | Number of Sentences | EN-DE | EN-FR | EN-ZH | EN-RU |
|---|---|---|---|---|---|---|
| JW300 | 417 | 109.08M | 2.1M | 2.3M | - | 1.0M |
| Tatoeba | 359 | 8.96M | 0.3M | 0.3M | - | 0.5M |
| TED2020 | 108 | 11.46M | 0.3M | 0.4M | 16.2k | 0.4M |
| WikiMatrix | 86 | 300.27M | 6.2M | 6.6M | 2.6M | 5.2 |
| Total Filtered Sentences | | | 2.9M | 2.6M | 89k | 1.9M |

**Table 4.1:** Datasets details

Either JW300, Tatoeba and TED2020 datasets contain mostly high quality parallel data, but it is not the same for WikiMatrix. In fact, during the definition of the datasets, sentences are evaluated to reach a minimum threshold of quality, otherwise they are discarded. Once the filtering phase is completed, the dimension of WikiMatrix dataset is heavly decreased. Even other datasets are filtered, in order to avoid the usage of low quality data. The number of sentences that are actually used in the training phase are shown in the last row of table 4.1. The amount of data is significantly lower than the starting dimension of datasets, and this will influence the quality of the training. This is especially true for the English-Chinese dataset, which contains a quantity of phrases that is not enough to allow the model to converge and obtain reliable results. In fact, as explained in following sections, the results obtained by English-Chinese CycleNLPGAN are not comparable to the ones achieved by competitor models or by CycleNLPGAN in other language pairs.

A minor part of the collection of selected dataset has been removed from the training set to perform evaluation during the training, in order to evaluate the model and define the correct set of hyperparameters.

## 4.2 Training configuration

The training is performed for four epochs. The first epoch has been executed using a constant learning rate equal to $2 * 10^{-5}$, then in the last three epochs a linear learning rate decay has been applied.

### 4.2.1 Hyperparameters configuration

The number of hyperparameters to fine-tune is quite large, so several configuration have been tried to achieve the best results. Experiments using all or a subset of parameters have been done in order to find the best configuration. The most important achievement obtained in this phase is the detailed discovery of each parameter contribution in the training, and so the best training approach to use during the training of the complete models.

The table 4.2 shows the value applied to each hyperparameter used during the training. A detailed explanation of each hyperparameter is reported to find out the correct contribution of each one.

Each hyperparameter has a specific role and is fundamental to find the correct equilibrium among them. In fact, GANs have the reputation of being unstable, and each uncontrolled oscillation may lead to a loss of performance or an impossibility of the network to converge.

The role of hyperparameters is fundamental, because it influences the training and the possibility of convergence. Several combinations have been tried empirically to find the most correct configuration. Using values defined in table 4.2, the training follows the typical steps of CycleGAN and, more in general, of GAN training. The losses values obtained training with the hyperparameters reported in table 4.2 are shown in figures 4.1, 4.2 and 4.3.

During the first phase, pretrained discriminators quickly find out differences between real and generated sentences, so discriminators losses are lower than generators ones. Instead, generators increase their losses during first steps because of the bad quality of generated sentences, reaching high peaks during this phase even because of the instability. In particular, $G_{AB}$ is very unstable in first iterations, and its loss reaches a peak value of 175. This is because both encoder and decoder update simultaneously, and, as consequence, unstable configurations may be frequent. It seems unusual because models are pretrained to perform translation, so it is expected that generated values have good quality, but the cycle consistency loss completely reshapes generators encoders parameters. At start, in fact, the distance among paired sentences embeddings is enormous and cycle loss tries to rapidly decrease it, forcing the $G_{AB}$ encoder to produce sentence embeddings as close as possible to $G_{BA}$ encoder embeddings, which are considered as ground truth. Doing it, generators decoders are unable to adapt to this impactful change in

| Hyperparameter | Value | Description |
|---|---|---|
| Learning rate | $2.0 * 10^{-5}$ | A tuning parameter that determines the step size at each iteration while trying to reach a minimum of a loss function. It represents the speed at which a machine learning model learns. |
| Lambda D | 10.0 | An hyperparameter that weights the importance of having correct discriminators predictions. Increasing it, the discriminator error has an higher impact and the models try to let it learn quickly. |
| Lambda G | 10.0 | An hyperparameter that weights the importance of generating reliable and good quality sentences.Increasing it, generators are more penalized when a bad quality sentence is generated. Sentence quality is defined using discriminator prediction on it and computing translation quality. |
| Lambda A | 50.0 | An hyperparameter that weights the importance of reconstructing sentences as similar as possible to the original ones starting from domain $A$ sentences. Increasing it, generators are more penalized when translation and back translation generates bad quality results. |
| Lambda B | 50.0 | An hyperparameter that weights the importance of reconstructing sentences as similar as possible to the original ones starting from domain $B$ sentences. Increasing it, generators are more penalized when translation and back translation generates bad quality results. |
| Lambda $C_1$ | 80.0 | An hyperparameter that weights the closeness of embeddings generated during the first translation step performed by both generators, using the same translated input sentence. Increasing it,generator $G_{AB}$ and $G_{BA}$ encoders are more penalized when $emb_{fake_A}$ and $emb_{fake_B}$ are distant. |
| Lambda $C_2$ | 15.0 | An hyperparameter that weights the closeness of embeddings generated during the first translation and the back translation by generators. Increasing it, $G_{AB}$ and $G_{BA}$ encoders are more penalized when $emb_{fake_B}$ and $emb_{rec_A}$ are distant and when $emb_{fake_A}$ and $emb_{rec_B}$ are distant. |
| Lambda $C_3$ | 5.0 | An hyperparameter that weights the closeness of embeddings generated during back translations performed by generators. Increasing it, $G_{AB}$ and $G_{BA}$ encoders are more penalized when $emb_{rec_A}$ and $emb_{rec_B}$ are distant. |

**Table 4.2:** Hyperparameters configuration and details

embeddings representation and produce inconsistent and meaningless sentences. As result, both discriminators are able to distinguish real sentences from inconsistent generated sentences, meanwhile generators produce meaningless sentences, but sentence embeddings rapidly become closer among languages.

Once the convergence of embeddings slows down, generators can adapt better on decoding sentences, so output sentence become better and better, and real sentences are reproduced with a good quality. This leads to a decrease of generators losses and a related rise of discriminator losses. In fact, discriminators were able to distinguish bad quality sentences, but in this phase generators output quality significantly increases. According to this, discriminators have to improve their results trying

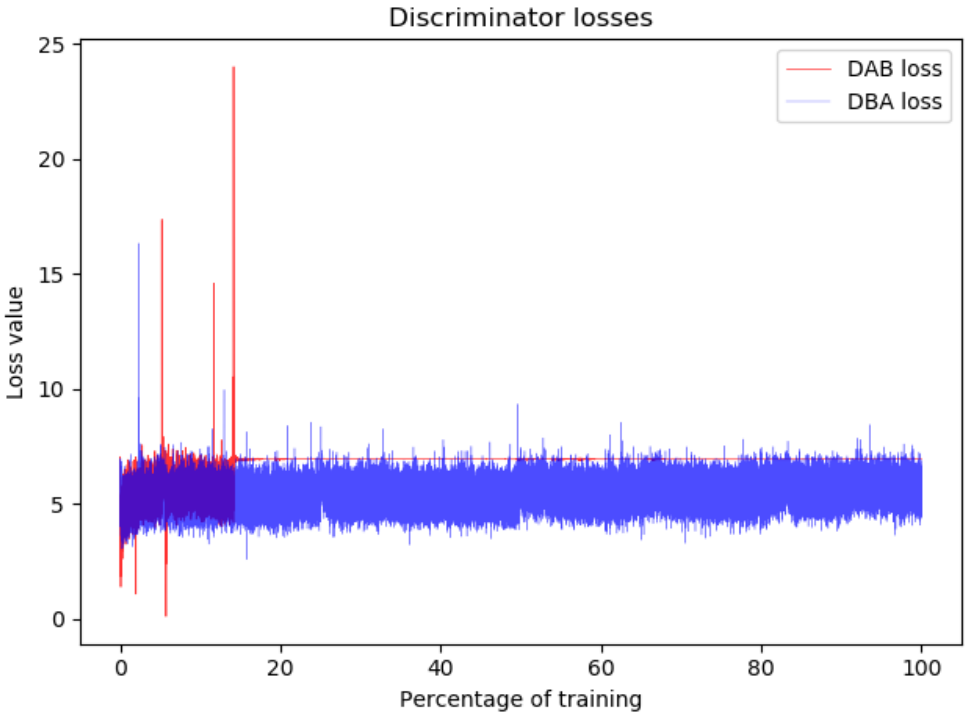


**Figure 4.1:** Discriminator losses during training

to distinguish more complex sentence structures and idiom characteristics. The increase of discriminators losses is not as amplified as in generators, and this is because the noise introduced by generators. In fact, even if discriminators learn to classify real and fake sentences, it often happens that fake sentences are still meaningless, and this is because the generators are improving translations gradually. As a consequence, the discriminators can gradually adapt back to the rise of quality in sentences.

The remaining part of the training is characterized by an increasing stability in losses values. The distance among similar sentences significantly reduces, keeping a stable value during epochs. Even the variance related to losses strongly decrease, producing loss values that can be grouped in small ranges of values. The same is for generators losses that, due to an increasing knowledge of language structures and features, decrease during time, reaching a stable value. Discriminators follow the inverse behaviour. In fact, during iterations discriminators partially worsen, but finally they start finding again differences among real and translated sentences. The only exception is the $D_{BA}$ behaviour, but the high instability of its loss is due to the usage of a multilingual DistilBERT, that contains information from several languages and needs more time to converge and specialize on a certain languages.

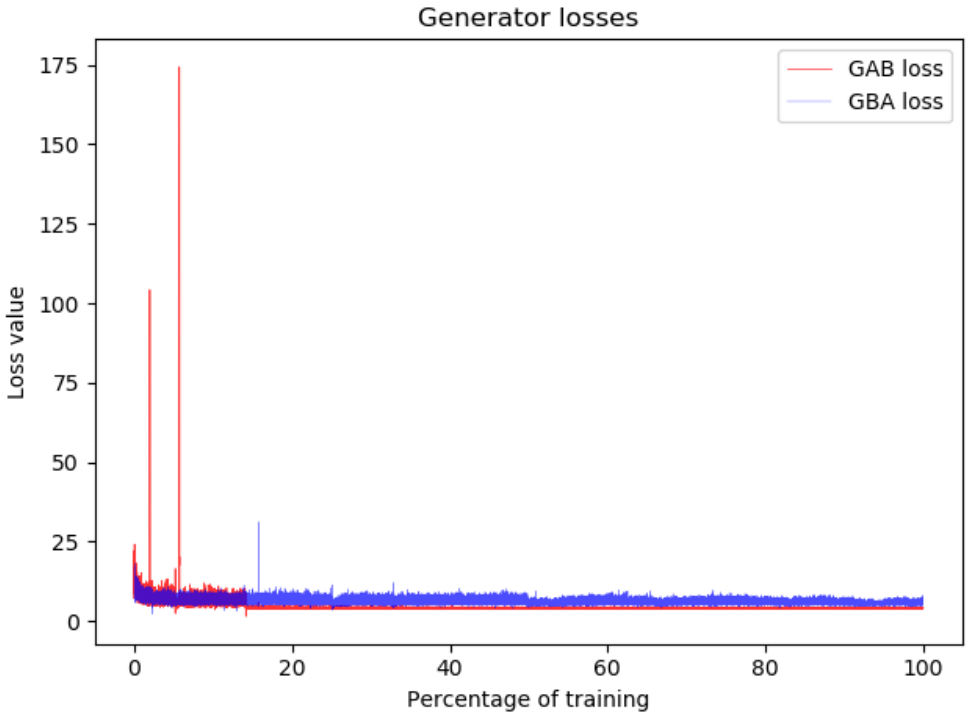


**Figure 4.2:** Generator losses during training

The choice of using a smaller discriminator than BERT has been done because of computational and memory limits. The same is for the usage of a multilingual discriminator, because the model is not pretrained on each language but on the most used ones. This require an additional pre-training phase to obtain a language specific discriminator, but it is an activity that is out of scope of the thesis.

## 4.3 Addressed tasks

Once the correct equilibrium among hyperparameters is found, several experiments have been conducted on both alignment and translation tasks. More in detail, bitext retrieval and WMT experiments have been conducted. Both of them are the most common challenges addressed in their respective tasks. They are used to classify and rank models developed to resolve alignment and translation respectively. CycleNLPGAN has been trained in several languages pairs, such as German-English, French-English, Russian-English and Chinese-English, and results obtained in both challenges are compared with state-of-the-art models in those languages. Moreover, results obtained using techniques similar to the one explained in the thesis are

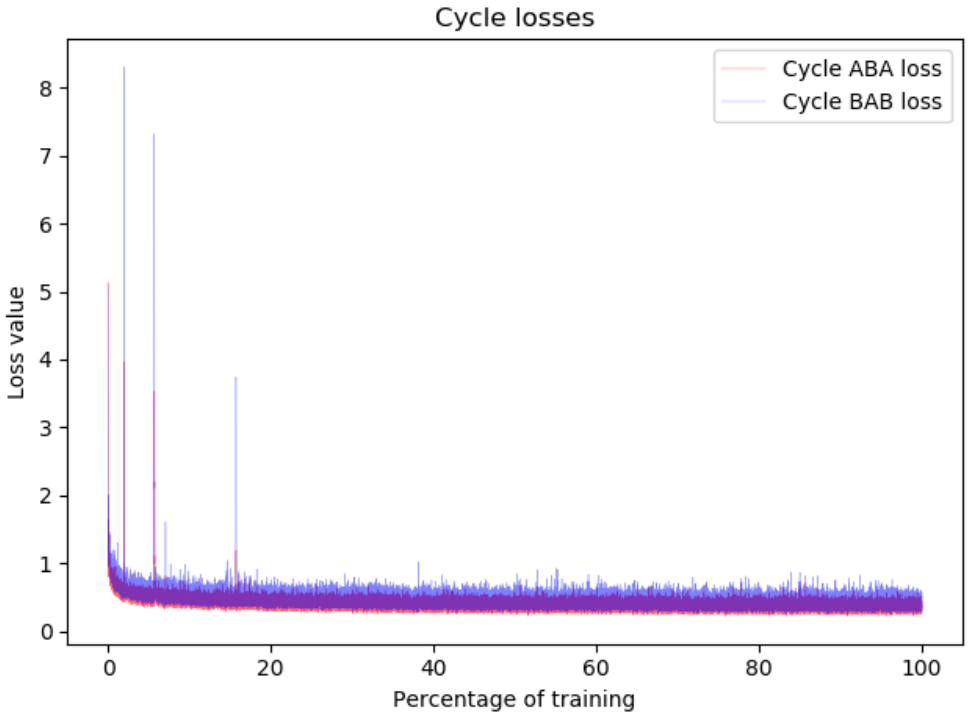


**Figure 4.3:** Cycle losses during training

used in the comparison. Finally, the CycleNLPGAN model is compared to native MarianMT results in WMT experiments, in order to define if the CycleNLPGAN is able to reach the translation results achieved before the training and eventually to improve them.

### 4.3.1 Bitext Retrieval : BUCC

Bitext retrieval aims to identify sentence pairs that are one the translation of the other inside two corpora written in different languages. In first implementations, the closeness among sentence was calculated using the cosine similarity, then the nearest neighbor was selected. This method has issues, so improvements have been done. A new scoring function has been implemented in LASER evaluation ([44]), and the formula is:

$$score(x, y) = margin(cos(x, y), \sum_{z \in NN_k(x)} \frac{cos(x, z)}{2k} + \sum_{z \in NN_k(y)} \frac{cos(y, z)}{2k})$$

where $x$ and $y$ are sentence embeddings of sentences from different languages and $NN_k(x)$ represents the k nearest neighbors of sentence embedding $x$ in the

other language. Margin function is represented as $margin(a, b) = a/b$.

The test is performed using the dataset from BUCC mining task, and the objective is to extract parallel sentences between English and a set of four target languages corpora, that are German, French, Russian and Chinese.

The dataset contains sentences for each language, going from 150k to 1.2M in each one, and 2-3% of them are actually parallel. A part of the dataset is used as training set in order to find a threshold for the score function, meanwhile the remaining part is used to test alignment measuring the $F_1$ score.

### 4.3.2 WMT : Workshop on Statistical Machine Translation

WMT is a workshop introduced in 2006 and is one of the most important challenges to evaluate neural machine translation models. The focus of this workshop it to use parallel corpora for machine translation. In this workshop researchers are encouraged to investigate models and approaches to improve the performance of Statistical Machine Translation systems for a wide set of languages, including morphologically complex languages, languages with partial free word order, and low-resource languages.

Four tasks are conducted in each workshop: a general translation task, a medical translation task, a quality estimation task, and a task to test automatic evaluation metrics. Since last editions, the number of tasks is increased to 11. These consisted of seven translation tasks, among which Machine Translation of News, Biomedical Translation and Machine Translation for Chats and four additional tasks not strictly related to translation (e.g., Automatic Post-Editing and Alignment for Low-Resource Conditions). The complete overview of the task can be found in [56].

Each year, a dataset, composed by training, evaluation and test data, is given to correctly train models, then results are calculated on the test set and announced at the conference. Most important translation models challenge each other on these tasks, especially on machine translation of news because it was introduced several editions ago and is the challenge that got more popularity during years.

## 4.4 Results

Given the computational resources required for training the complete model, we define *English-French* as evaluation language pair to analyze the impact of different parameters during the training phase. After finding the correct configuration of parameters, $English - Chinese$, $English - German$ and $English - Russian$ models have been trained.

Before defining parameters, several training configurations have been tried to define:

- **The loss function to use:** Experiments with MSE loss and cosine loss have been tried. MSE loss is used in similar approaches like [45] because of the computational efficiency. Cosine loss instead is a valid alternative to MSE, and apparently seems more correct for the thesis objective. This loss is used to calculate the error in sentence embeddings computation performed by generators encoders and is used to update models weights.

- **Training approaches:** This thesis work evaluates two different training approaches. The first approach consists in using MarianMT as is. There is no contraints in training the full set of weights of both the generators. This configuration is the standard training approach for this architecture design. However, considering the CycleNLPGAN architecture, it may lead to an unstable training, or in the worst case, to a unreachable convergence. This can happen because the encoders embeddings are way far in the beginning, and the generators try to get closer as soon as possible. Doing this, they may converge in single points or in close spaces, collapsing the models vector spaces and producing sentence embeddings in a subspace of the starting vector space. There is no guarantee of a convergence in this situation, or, in the best case, it may require a long training. The second approach, instead, consists in freezing the encoder part of the English-to-*language* generator. In this case, the English encoder acts as stable and fixed sentence embeddings model. Its predictions are used as ground truth by the *language* encoder available in the *language*-to-English, that is trained to produce sentence embeddings that get as close as possible to the parallel English ones. In this case, the vector space of the English-to-*language* encoder remains the same during the whole training, and this prompts the *language*-to-English encoder to produce a similar vector space, that is the main goal of this thesis. This approach uses the same assumptions proposed by [45]. However, both the architectures and addressed tasks differ. The authors of [45] designed a model that gets sentences from both languages during the same training and tries to map them in a shared space without addressing the translation task. Moreover, the teacher model is used as an auxiliary part of the training of the student model, meanwhile in CycleNLPGAN model it is a part of the trained model, even if it is fixed and frozen.

- **Hyperparameters settings:** The analysis aims to define if all defined hyperparameters should be used or if only a subspace of them is necessary. Some hyperparameters, such as $lambdaC_1$, $lambdaC_2$ and $lambdaC_3$, have been defined to speed up the training and to avoid the model collapse in a single point. Moreover, they may increase the closeness of similar sentence embeddings using information obtained by the back-translation. This increases the utility and the importance of the approach introduced in the thesis. To evaluate the

correctness of these hyperparameters. a configuration that does not use all of them has been used, then results are compared with the ones obtained using the whole set of parameters.

In order to define which choice is better, the comparison of each couple of configurations has been done using several values:

- BLEU score calculated on French to English and English to French sentences taken from evaluation dataset

- Average distance calculated on parallel sentences taken from evaluation dataset. In this way, the closeness of parallel sentences can be defined

- Average distances between a sentence and the non-parallel ones that compose the evaluation dataset in the same language and in the other language. These distances are defined as mutual distance in the following sections. Calculating the average distance of the distance of each sentence from the others in the same language, a dimension of the language vector space can be defined. In this way, an evaluation of the convergence and of the collapse of the language vector space can be performed during the training. Calculating the average distance of each sentence to the non parallel ones in the other language gives a measurement of the closeness and the alignment among languages vector spaces. Together with the alignment among parallel sentences, this value may help in the analysis of the effectiveness of the training. In the following sections, the average distance among each English sentence from the others in the French dataset is defined as *English vector space dimension*, meanwhile the average distance among each French sentence from the non parallel ones in the dataset is defined as *French vector space dimension*.

The evaluation set used in this analysis is part of the OPUS dataset, and it is composed by randomly selected sentences from the whole dataset. Sentences are selected before starting the training and then are removed from training data. Every configuration model trains for a subpart of the training dataset before being stopped and compared to the other approach. In following sections, a detailed explanation of each configuration shows the path followed to define the best configuration to use during the complete training with other language pairs. Remark that distances plots shown in next paragraphs report cosine distances, but this is not related the loss function. In fact, distances are calculated always using cosine distance, that is based on cosine similarity, meanwhile cosine loss defines the error that is backpropagated through the network according to the similarity among parallel sentence embeddings.

### 4.4.1 Loss function analysis

The first analysis is related to loss functions and aims to find the most correct loss to use during the training. In this case no information about freezing or not the $English - to - language$ encoder and about the set of hyperparameters to use was found. To take the models as similar as possible, both models have been trained freezing English to French encoder and using a subset of parameters, that involves all parameters defined in 4.2, except $LambdaC_2$ and $LambdaC_3$. The only difference is related to the loss function used to evaluate closeness of sentence embeddings. Plots of BLEU scores, average distances and mutual distances are reported in figures 4.4, 4.5 and 4.6.

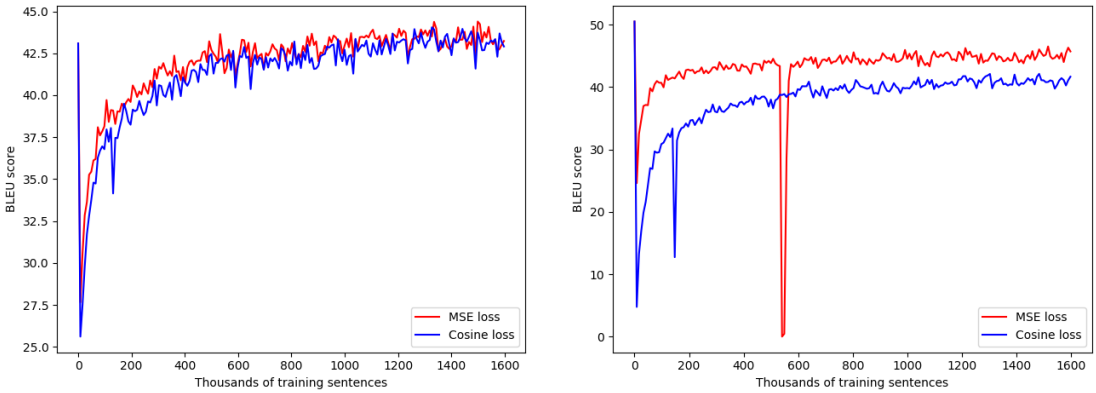


**Figure 4.4:** BLEU score obtained using different loss functions

Analyzing BLEU scores obtained using MSE loss and cosine loss, it seems that results are similar, especially for English to French ones, that are represented in the plot on the left in figure 4.4. In fact, BLEU values are similar and follow the same pattern, that is a rapid fall from starting scores in first steps, then a fast-paced increase, followed by a stabilisation and a balanced improvement. The same pattern is followed in French to English scores, reported in the right plot of figure 4.4, but in this case the difference among MSE and cosine is wider than before. Indeed, even if there are some unstable points, scores obtained using MSE translating from French to English are always greater than the ones obtained using cosine loss.

The analysis of average distances calculated on parallel sentences, represented in figure 4.5, reports a clearer overview of losses difference and effectiveness. Ideally, this value should be zero, and this means that sentence embeddings, and more in general languages vector spaces, are completely overlapped. Using MSE loss, the average distance decrease rapidly, reaching a 0.1 cosine distance after less than

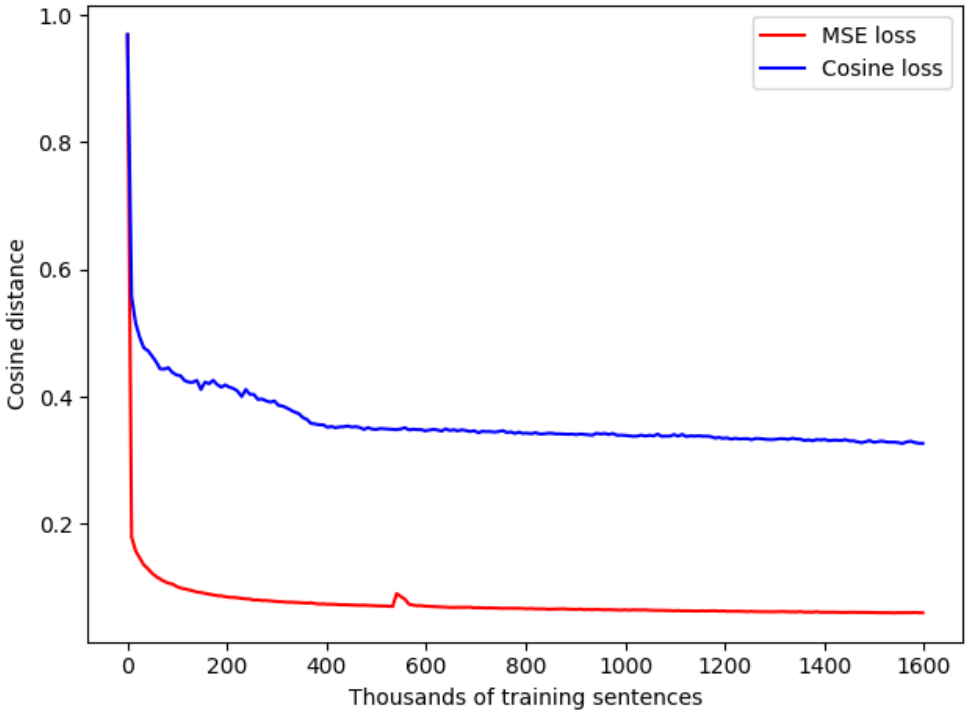


**Figure 4.5:** Average distance obtained using different loss functions

200 thousand iterations. After this phase, the cosine distance keeps decreasing, reaching a value of 0.06 at the end of the 1.6 million iterations. In contrast, using the cosine loss, the distance among parallel sentences decreases rapidly in first iterations, but once 200 thousand iterations have passed, it reaches a value of 0.4, that is much higher than distance obtained using MSE loss. Moreover, the speed of convergence to zero falls substantially, and after 1.6 million of iterations reaches a value of 0.35.

It is obvious that the speed of convergence and the closeness obtained using MSE loss are better than the ones achieved using cosine similarity, but mutual distances plots, figured in 4.6, give another visualization of training approaches. Indeed, using MSE loss, the dimension of English vector space is constant, and this is because of the usage of a frozen English to French encoder, meanwhile the French vector space asymptotically converges to the distance among English and French. This is not a bad information, that tells that the French vector space dimension increases during first iterations, then starts decreasing smoothly, but according to previous assumptions, it seems more plausible that both distance and French dimension would converge to English vector space. This is what partially happens using cosine loss. The distance among spaces and the dimension of the French

vector space follow a smooth convergence to English, but it is a slow process and may require a consistent number of epochs to converge.

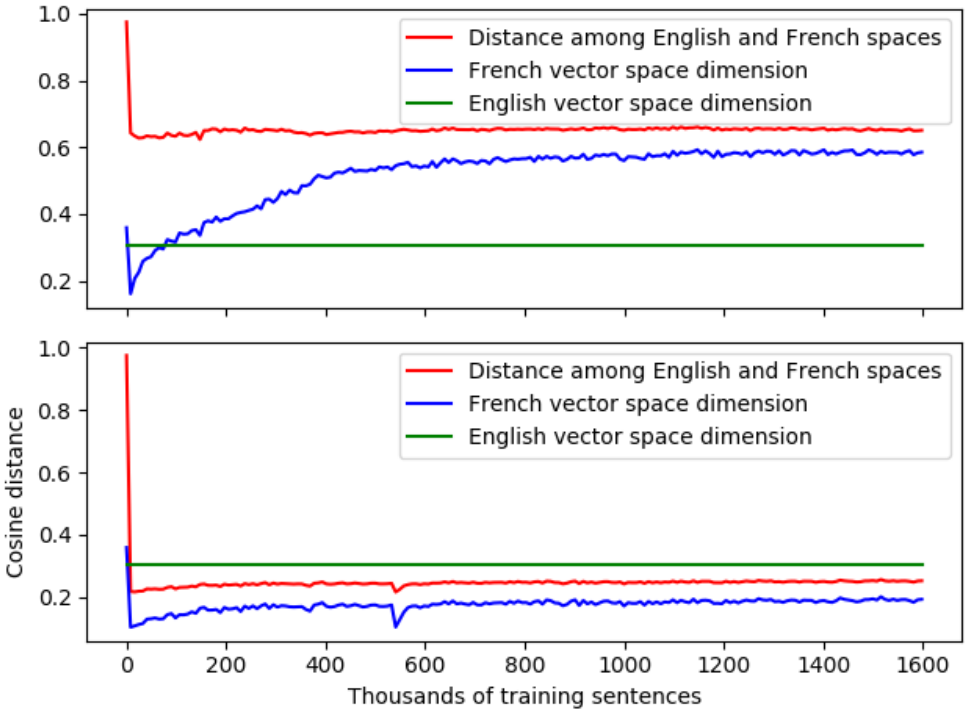


**Figure 4.6:** Average mutual distances obtained using different loss functions using MSE loss (on top) and cosine loss (below)

Due to the faster convergence of MSE loss and the lack of computational resources, MSE loss has been selected as loss function to compare sentence embeddings during training. Even if cosine loss seems to allow a better adaptation of French vector space to the English one, MSE loss obtains better results than cosine on translation quality and on speed of convergence.

### 4.4.2 English to *language* encoder training analysis

Once the loss function to use is defined, two configurations of model have been trained for 1 million iterations in order to define the best training approach for generators. As explained in previous chapter, the main objective of this thesis is to align two or more language vector spaces, and this requires to penalize generators encoders when far sentence embeddings are generated for parallel sentences. Calculating the loss among produced sentence embeddings and back-propagating it through both generators, they update their parameters. In this case, each generator

updates its encoder and its decoder using the loss value, and this produces a continuous variation of sentence embeddings of both generators. As a consequence, the vector spaces vary as well. Moreover, the decoder tries to reproduce correctly the sentence starting from the sequence of embeddings, but these values are not stable, especially during first iterations. This behaviour may be dangerous for the networks, because there is no control on language vector spaces, and this may lead to the massive reduction of vector spaces or, in the worst case, to the convergence of both generators in a single point. As far as decoders are concerned, the issue is similar, as they are not able to adapt to rapid changes and are not able to produce meaningful phrases when encoders collapse in individual points. This is because of the absence of a ground truth in the definition of sentence embeddings. A solution may be the use of anchor points, that represent some pre-defined sentence or word embeddings, that have to be used by the networks as points to enforce. In this way, the alteration of languages vector space is more controlled and their collapse may be avoided. Although this solution has positive aspects, it is not sure that the networks would converge in a shared space. Anchor points could be respected, but the rest of sentence embeddings may fall back in some isolated points or may assume an unpredictable behaviour.

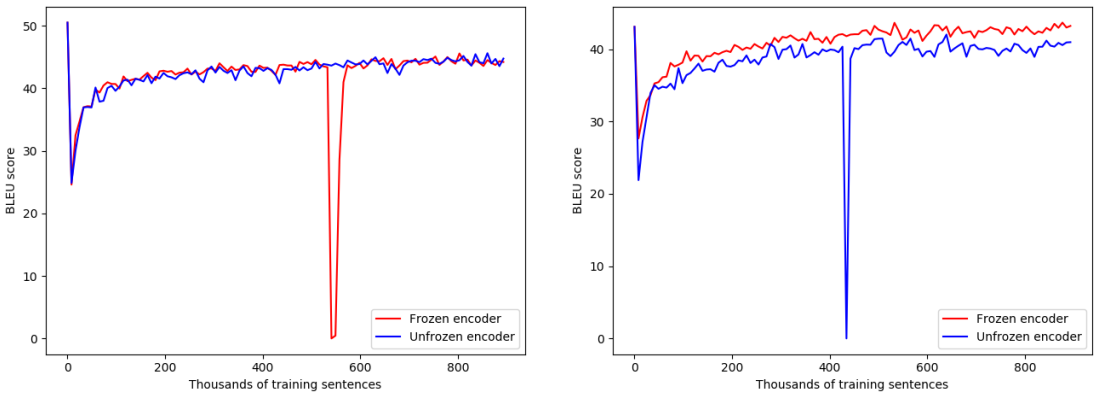


**Figure 4.7:** BLEU score obtained using different training approaches

The possible solution is the usage of an encoder as ground truth, that has frozen parameters. This means that the encoder does not update its parameters, but it remains the same during the whole training phase. Since the objective of thesis is to produce a shared vector space among languages using English as focal language, a training using a the English to French frozen encoder is performed. This will help convergence speed, convergence stability and will produce a French vector space that adapts itself to the predefined English vector space. Plots of results obtained with the training of both generators encoders and with the training of one encoder

only are reported in figures 4.7, 4.8 and 4.9.

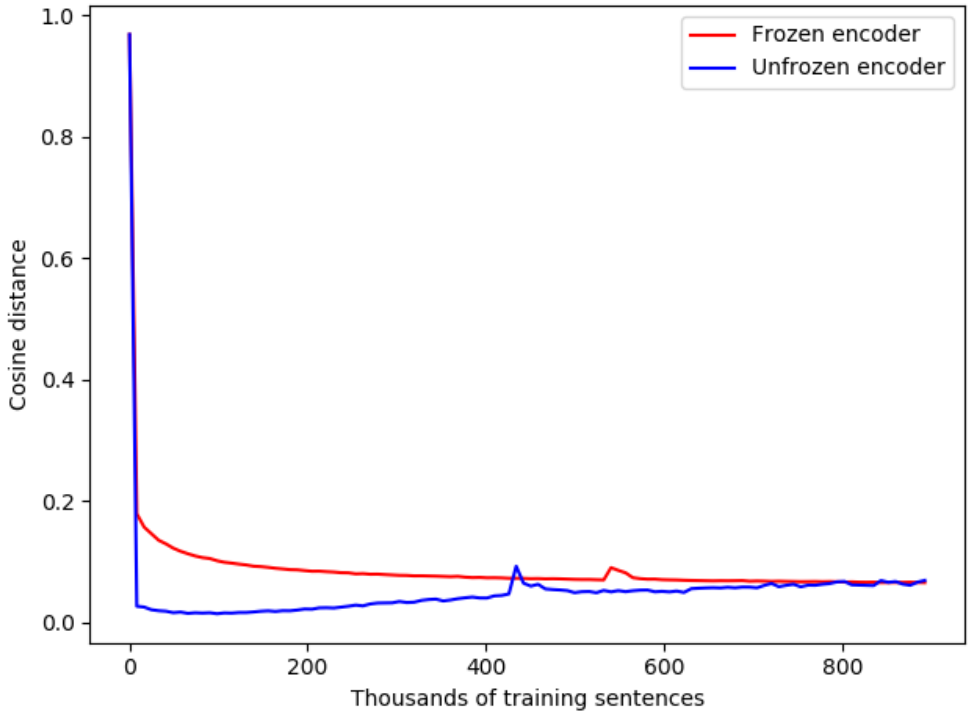


**Figure 4.8:** Average distance obtained using different training approaches

As shown in figure 4.7, the behaviour of BLEU scores follow the same pattern during iterations. Unless some unstable points, French to English BLEU scores are identical during the whole training. English to French BLEU scores show some differences. In fact, using a frozen encoder, the BLEU score is always higher than the trained one. This is because of the use of a fixed encoder, and this means that the only part of English to French network that is trained is the decoder. This lead to a more focused decoder training, and as result the frozen encoder configuration lightly outperforms the unfrozen one.

Analysing the plot in figure 4.8 that reports the average distances among parallel sentences, it is clear that the two configurations have different behaviours. Even if the unfrozen encoder configuration seems better, it is not that correct. In unfrozen encoder, the average distance among sentences drops in few iterations, reaching a value close to 0. This is exactly the worst case explained before, that is the collapse of the encoder sentence embeddings in some isolated points. In fact, analysing the upper plot in figure 4.9 that shows distances among non parallel sentences and languages vector space dimensions obtained training both encoders, both English and French vector space dimensions drops down to a value close

to 0. This means that the sentence embeddings related to the evaluation set, that contains heterogeneous sentences, fall in near points, regardless the sentence meaning. This behaviour may produce an irrecoverable configuration of generators, and the networks will not be able to translate and align. This is not the case, because plots show that during time the vector spaces increase their size, but the cannot reach back the starting English vector space dimensions. Even the average distance among parallel sentences, shown in 4.8, increases and converges to the same results of the configuration that uses the frozen encoder.

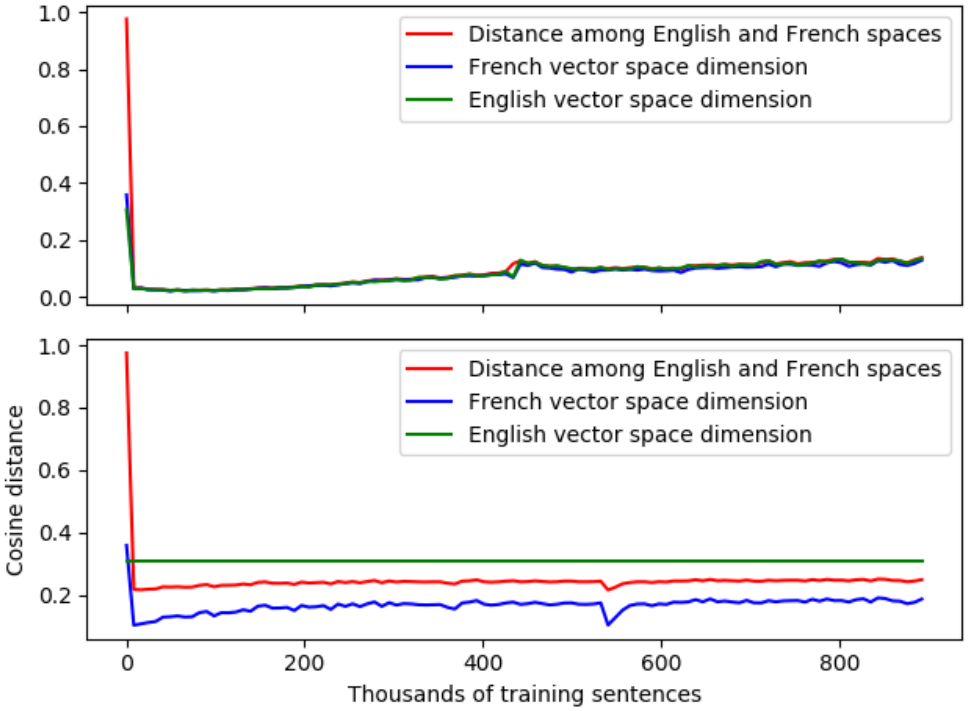


**Figure 4.9:** Average mutual distances obtained using different training approaches(top: unfrozen encoder, below: frozen encoder)

It seems that both configuration may be viable, but the frozen encoder configuration is chosen as best during the following steps. This choice is related to the instability of the configuration that simultaneously trains both generators. Even if not reported in the thesis, other experiments have been performed to understand the risks and the possible advantages related to the use of the unfrozen encoder configuration. The training of both encoders apparently seems the most correct, but it often happens that networks vector spaces fall in single points and generators are not able to recover. In fact, trying to produce sentence embeddings that are as close as possible and without using a ground truth, both models collapse and produce the

same sentence embedding vector, regardless the input sentences. This behaviour worsen during iterations, and produce a pair of generators that encode all sentences in the dataset in a limited amount of points in the shared vector space, and this leads to the definition of sentence embeddings that don't contain information about the input sentences and to the generation of meaningless sentences.

### 4.4.3 Hyperparameters usage analysis

So far, loss function and encoders training approach have been chosen. During previous experiments, only a subset of parameters were used. This decision was taken because the training using all hyperparameters requires a not negligible amount of computational time and resource. In fact, the use of all hyperparameters implies the increase of losses computed during each iteration, and the overhead of resources is not insignificant. Using a subset of them, instead, the number of losses calculated decreases, and the average time needed to complete an iteration decreases as well. Even without all the parameters, the network trained with MSE loss and a frozen English to French encoder gives outstanding results. In this section, a detailed analysis of differences between training using all hyperparameters or a subset of the is reported. More in detail, two hyperparameters were set to null in previous experiments, that are *lambda* $C_2$ and *lambda* $C_3$.

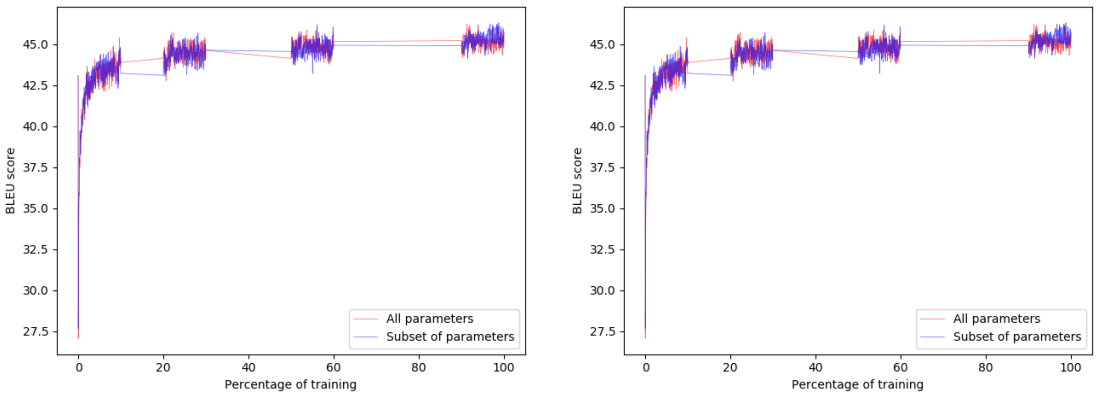


**Figure 4.10:** BLEU score obtained using different hyperparameters

*Lambda* $C_2$ defines the weight of the penalization related to the distance among translation and back-translation sentence embeddings, meanwhile *Lambda* $C_3$ is the parameters used to weight the penalization related to the distance among back-translation performed in both languages. These hyperparameters were introducted to enforce the sentence alignment comparing several combinations of the produced sentence embeddings, and to exploit the backtranslation. Doing this, the usage of

cycle consistency is more effective and becomes a fundamental part of the training and of the CycleNLPGAN approach.

The analysis has been performed training both the configurations for four epochs and using a linear decay for the learning rate. The value of each hyperparameter is reported in table 4.2. Plots of results obtained using all hyperparameters and using a subset of them are reported in figures 4.10, 4.11 and 4.12.

Unlike the previous cases, in this analysis the differences between the various metrics used to evaluate the models are much smaller. BLEU scores, reported in figure 4.10, are calculated in English to French and in French to English directions and have the same trend during the four epochs. Both configurations reach MarianMT's starting BLEU scores on the evaluation dataset at the end of the first epoch and improve them during the last three epochs. BLEU scores achieved using the whole set of hyperparameters are slightly higher than the others, but the difference is minimal.

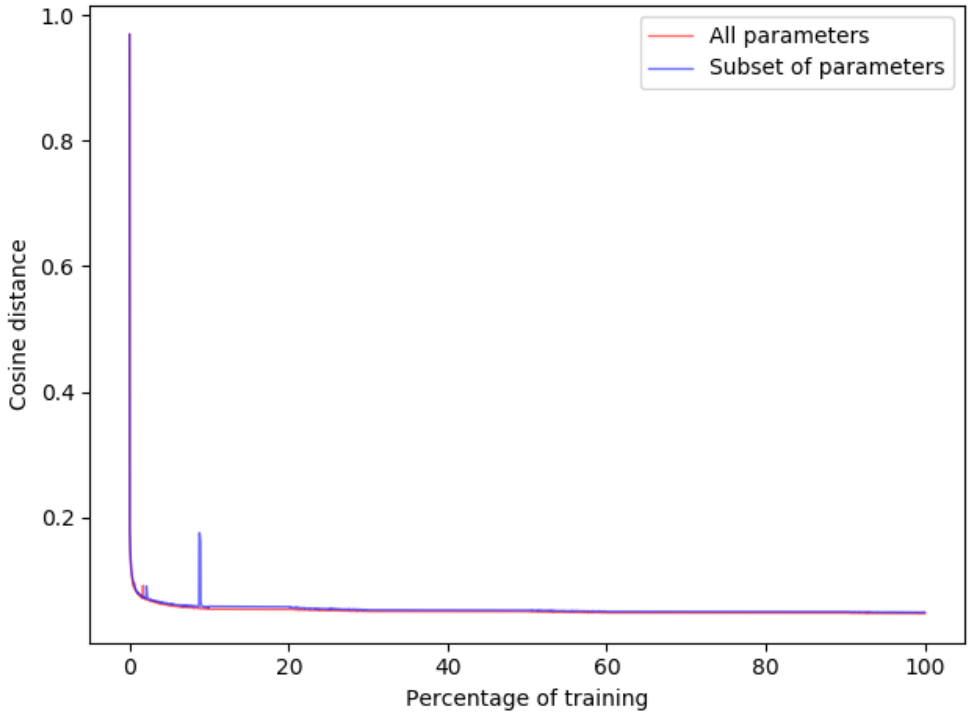


**Figure 4.11:** Average distance obtained using different hyperparameters

The same is for the average distance values achieved during the four epochs. Configurations obtains similar values during the whole training and reach a minimum value of 0.046 at the end of the training. According to values reported since now, the configurations are equivalent and produce sentences that have the same

meaning, structure and that have similar embeddings. In fact, the comparison of sentences produced during the training of both configurations shows the similarity among translations obtained starting from the same input sentences. The qualitative results obtained during this phase are not reported in this thesis.

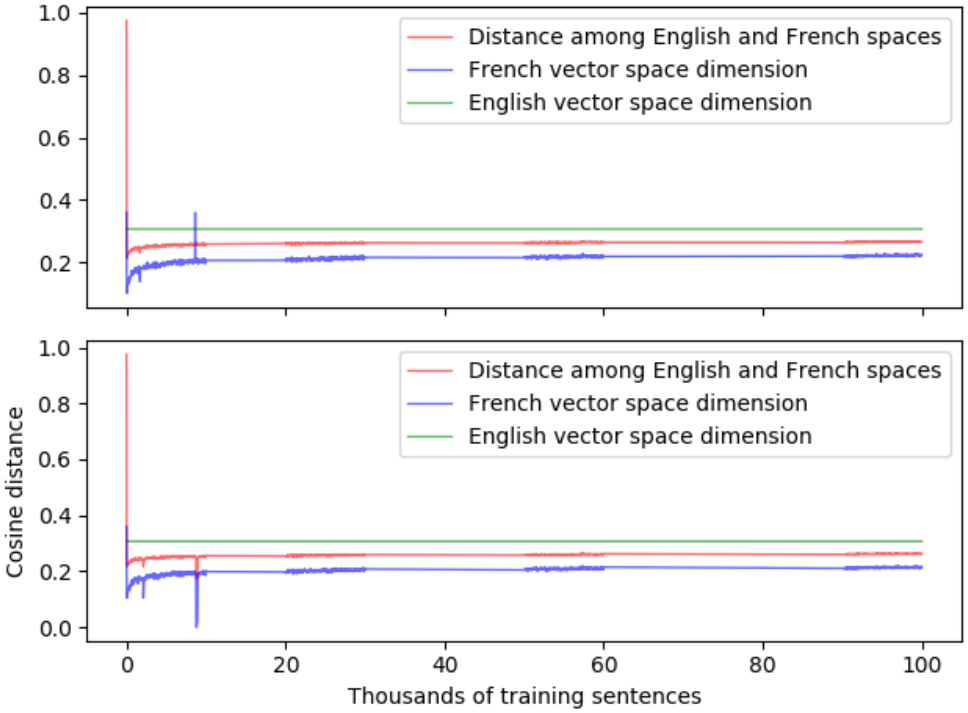


**Figure 4.12:** Average mutual distances obtained using different hyperparameters (on top: full set of hyperparameters, below: partial set of hyperaparameters)

The last plots used in the analysis, that are shown in figure 4.9, report the progress of average distances among vector spaces and the evolution of the French vector space in comparison with the English one. Even in them, the difference is almost nothing. Mutual distances are equivalent, the only difference is obtained comparing the vector spaces. The French vector space obtained using all the parameters is higher than the one obtained in the other configuration, and it gets a little closer to the English vector space. This means that French space is more similar to the English one, and that they are more overlapped.

Even if the computational effort and the time needed to converge is slightly higher using the whole set of parameters, this configuration has been chosen to perform the training in other languages. The additional time required by the configuration compared to the other is of the order of minutes, so it was considered as an highly affordable price to get slightly better results.

### 4.4.4 Results on addressed tasks

So far, performed analysis gives the best possible configuration that can be used during the training. Loss function, encoder training approach and the set of hyperparameters needed in the training have been defined comparing results obtained using several combinations of them. As result, the most resilient and performing configuration has been obtained and used to train models in several languages pairs, such as English-French, English-German, English-Russian and English-Chinese. Once the training phase is completed, the models are evaluated on the addressed tasks. In particular, each model is tested on BUCC bitext retrieval and on WMT news translation task, that address respectively sentence alignment and neural machine translation. In following sections, results are analysed and compared to other models. Moreover, qualitative results of models are reported in the last section.

#### BUCC bitext retrieval results

Results are compared to the state of the art models results obtained using LASER ([44]) and LaBSE([57]), and to other models, such as distillation training models ([45]), mUSE ([22]), XML-RoBERTa ([33]) and multilingual BERT ([37]). Results obtained by each model values are shown in table 4.3 and are used to compare models and training approaches to the one produced in this thesis work. Values related to the knowledge distillation have been obtained using a Sentence BERT ([34]) as teacher model.

| Model | DE-EN | FR-EN | RU-EN | ZH-EN | Average |
|---|---|---|---|---|---|
| mBERT [37] | 44.1 | 47.2 | 38.0 | 37.4 | 41.7 |
| XML-RoBERTa [33] | 5.2 | 6.6 | 22.1 | 12.4 | 11.6 |
| Distilled XML-RoBERTa [45] | 90.8 | 87.1 | 88.6 | 87.8 | 88.6 |
| mUSE [22] | 88.5 | 86.3 | 89.1 | 86.9 | 87.7 |
| LASER [44] | 95.4 | 92.4 | 92.3 | 91.7 | 93.0 |
| LaBSE [57] | 95.9 | 92.5 | 92.4 | 93.0 | 93.5 |
| **CycleNLPGAN** | 87.1 | 83.7 | 84.8 | 31.0 | 71.7 |

**Table 4.3:** $F_1$ score on BUCC bitext mining task

LASER and LaBSE obtain the best average values and the highest results in each language pair, and this is because their structures were built to have high performance in alignment among languages. Moreover, LASER is able to align more than two languages simultaneously, producing an agnostic vector space used by all languages. After them, Distilled XML-RoBERTa obtains the highest results. It is trained using the distillation training approach, that is the most similar to the one

used in this thesis. Unlike CycleNLPGAN, this technique takes advantage of data from both languages at every stage of the training, and this is demonstrated by the superior results compared to those of CycleNLPGAN. CycleNLPGAN results are comparable to those obtained by mUSE, a model uniquely built to align languages among them using a mapping function trained using an adversarial approach. Although CycleNLPGAN does not reach the level of direct competitors, the overall result is reasonable, as the model has been trained to reach a compromise between translation quality and sentence alignment. In fact, the model is able to produce aligned sentences and defines a latent space shared among languages. At the same time, it is able to produce meaningful sentences that are correct translation of original ones. Moreover, as shown in the next section, the model produces results that have the same quality as those produced by MarianMT generators before the training. The main difference to the starting MarianMT generators is the widely increased alignment among them according to the shared vector space. Before CycleNLPGAN training, models embeddings are much further than those obtained after the whole training phase. Starting from an average cosine distance value of 0.99 in all languages pairs on the evaluation set, the models reach a final average cosine distance value of 0.036, and this explains the strong contribution of the introduced approach.

Another aspect that should be considered to analyse CycleGAN results in BUCC is related to the training duration and to the datasets used in the training phase. In fact, the opposing models have been trained for many more epochs and with more data, which was not possible in CycleGAN due to time needed and due to lack of computational resources. This is also confirmed by the low results obtained by the English-Chinese languages pair model, as the number of available sentences were much lower than those required to converge and to obtain a fulfilling alignment result. Moreover, if it is necessary to achieve higher alignment results, it would be possible to increase the value of the hyperparameters that manage the impact of the alignment. However, this could lead to a less stable model or to a final configuration that has poor translation quality.

Another meaningful consideration that should be done analysing results is related to the dimension of the sentence embedding vector. In fact, the CycleNLPGAN model, and the MarianMT model as well, produces a $512 - length$ vector, that is considerably smaller than the one produced by LASER, that produces a $1024 - dimension$ fixed-size vector, and by sentence BERT, that generates a $768 - length$ vector. This may lead to the production of more compact vectors, and this significantly decrease performances on BUCC task. Other approaches, instead, produce more sparse vectors, and this mathematically would lead to a lower probability of having equal sentence embeddings or close sentence embeddings when two or more sentences are not perfectly aligned. It may be useful to compare different dimension sentence embeddings in order to quantify the difference among

produced embeddings and more in general among produced vector spaces. A technique that can be used to do this is related to the calculation of the average distance among sentences obtained using the different models. The more the distances are close among models embeddings, the more the models produce vector spaces that have similar dimensions, and therefore more sparse sentence embeddings.

Finally, results achieved by CycleNLPGAN in certain language pairs are slightly lower than competitors, such as distillation technique, but this may be related to the limited duration of the training and to the necessity of training both alignment and translation simultaneously. In fact, models like LASER are able to produce highly aligned vector spaces among languages, but these models have issues assigning meaningful similarity scores for sentence pairs that don't have identical meaning, but a similar one. On the other hand, mUSE and distillation approach produce vector spaces that make semantically similar sentences fall in close points. However, in the BUCC setup, similar sentences pairs are not labelled as parallel sentences, and this means that these models are penalized by BUCC score when sentences are not perfectly aligned. Since an approach similar to [22] and [45] has been used, a similar assertion could also be made for CycleNLPGAN model. To be certain of this, it should be useful to test the architecture in a semantic textual similarity task, but due to lack of computational resources and time it was not possible to do it during this thesis work.

### WMT news translation results

CycleNLPGAN model has been evaluated on WMT news test sets from 2014 to 2017, but the relevant dimension of WMT training datasets and the lack of computational resources don't allow the correct training of the model on the respective training sets. In fact, WMT datasets contain a huge amount of data, and completing the training on them would have required unaffordable time. This led to the use of the dataset described in previous sections, despite the expected results would not be totally comparable to those obtained training models on specific WMT datasets.

The most important analysis that has been carried out on the WMT task is related to the comparison between native MarianMT and CycleNLPGAN. As explained in the previous chapters, CycleNLPGAN architecture contains within it two MarianMT models. The first one is responsible for translating from English to another language, meanwhile the second follows the inverse direction. Tables 4.4, 4.5, 4.6 and 4.7 report the BLEU scores achieved by several models translating WMT test sets from a language to another. Except for MarianMT, mBART and CycleNLPGAN, the reported results are taken from the respective publication, so they may contain partial data with respect to those analysed in this thesis. Only results taken from official paper are reported for these models, meanwhile MarianMT, mBART and CycleNLPGAN have been tested on each

WMT dataset.

In most cases, the results obtained by the native MarianMT are higher than those obtained by the models obtained training the CycleNLPGAN. In fact, in most of the scenarios, MarianMT achieves results that are close to those obtained by the state-of-the-art models on WMT task, and CycleNPLGAN is not able to reach them. However, it sometimes happens that CycleNLPGAN outperforms MarianMT, increasing the quality of the translation that it had initially thanks to the techniques used in this thesis work. In fact, the native MarianMT is the architecture used to build the CycleNLPGAN model. MarianMT results correspond to the results obtained by CycleNLPGAN starting model, so this comparison gives a feedback on the effectiveness of the presented technique. Even if the MarianMT is trained on WMT training dataset, CycleNLPGAN is, able to get back to starting translation quality, and even to outperform it in certain scenarios. At the same time, the model is able to align languages vector spaces, so, under these assumptions, results obtained by CycleNLPGAN through years datasets is fulfilling.

Once the comparison between MarianMT and CycleNLPGAN is concluded, it is possible to make a deeper analysis of the results obtained using CycleNLPGAN compared to those achieved by the state-of-the-art models on WMT task. As shown in tables 4.4, 4.5, 4.6 and 4.7, in most cases CycleNLPGAN results are lower than other models, that are specifically designed to address neural machine translation task and that are trained on WMT datasets. Despite the use of a different training set, some language pairs models trained using CycleNLPGAN approach obtain results that are quite similar to those achieved by competitors on several years test sets, and in some cases CycleNLPGAN exceeds them.

| | WMT 14 | | | |
|---|---|---|---|---|
| **Model** | **EN-DE** | **DE-EN** | **EN-FR** | **FR-EN** |
| mBART [58] | 21.9 | 31.6 | 32.2 | 34.4 |
| MarianMT [51] | 23.8 | 29.5 | 39.8 | 38.0 |
| Transformer Cycle [59] | 35.14 | - | - | - |
| CMLM + LAT [60] | 27.35 | 32.04 | - | - |
| Transformer + BT [61] | - | - | 46.4 | - |
| SMT + iterative bascktranslation [62] | 14.08 | 17.43 | 26.22 | 25.87 |
| **CycleNLPGAN** | **25.43** | **28.26** | **34.7** | **33.0** |

**Table 4.4:** WMT '14 BLEU scores

However, over the years models have been designed to become more able to produce high quality translations. This is evident from the analysis of results. Indeed, models that have been implemented recently and than have been tested on WMT datasets far exceed those achieved using CycleNLPGAN model, meanwhile older models achieved results that are comparable to CycleNLPGAN ones.

| | WMT 15 | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **EN-DE** | **DE-EN** | **EN-FR** | **FR-EN** | **EN-RU** | **RU-EN** |
| mBART [58] | 24.7 | 31.1 | 29.0 | 32.0 | 25.9 | 29.3 |
| MarianMT [51] | 26.2 | 29.2 | 38.4 | 38.3 | 27.1 | 29.9 |
| ByteNet [63] | 26.3 | - | - | - | - | - |
| C2-50k Segmentation [64] | - | - | - | - | 20.9 | - |
| **CycleNLPGAN** | **27.7** | **29.1** | **35.1** | **33.1** | **21.0** | **25.0** |

**Table 4.5:** WMT '15 BLEU scores

| | WMT 16 | | | |
|---|---|---|---|---|
| **Model** | **EN-DE** | **DE-EN** | **EN-RU** | **RU-EN** |
| mBART [58] | 29.6 | 37.9 | 20.9 | 29.2 |
| MarianMT [51] | 31.4 | 35.7 | 26.2 | 30.0 |
| Multi-Agent Dual Learning [65] | 40.68 | - | - | - |
| Attentional encoder-decoder +BPE [66] | 34.2 | 38.6 | 20.9 | 28.0 |
| **CycleNLPGAN** | **27.7** | **29.1** | 18.7 | 24.1 |

**Table 4.6:** WMT '16 BLEU scores

Another aspect that should be analyzed carefully is related to the results obtained by the English-Chinese CycleNLPGAN model. Although the results are far lower than those obtained from other models, they can be used to understand the importance of having a suitable dataset in order to obtain sufficient translation quality. The dataset associated with the English-Chinese language pair, in fact, contains a very limited amount of data compared to those used for other language pairs, and even less than those provided by WMT for the training phase. A low amount of phrases is used for the training of this model because of the absence of Chinese data in some of the datasets used during the training. Those of them that contain Chinese-English parallel sentences are characterized by a not negligible amount of low quality data in them, and these sentences were discarded in the definition of the dataset. This filtering procedure has led to the use of only 87 thousand sentences during the training of English-Chinese model, that an inadequate number to obtain results comparable to those reported in table 4.7.

The same assumption can be reported to the other CycleNLPGAN models results reported in previous tables. Indeed, also for the other languages pairs, the amount of data used during the training phase is much lower than the corresponding data provided by the WMT dataset for the same language pair. The choice of a smaller dataset was made during the dataset definition and takes into account the time needed to complete the training and the related amount of computational resources required.

| | WMT 17 | | | |
|---|---|---|---|---|
| **Model** | **EN-DE** | **DE-EN** | **EN-ZH** | **ZH-EN** |
| mBART [58] | 23.0 | 33.0 | 1.43 | 23.6 |
| MarianMT [51] | 25.0 | 30.8 | 0.93 | 19.2 |
| OmniNetP [67] | 29.0 | - | - | 23.0 |
| T2R+Pretrain [68] | 34.2 | - | - | 23.8 |
| **CycleNLPGAN** | **26.1** | **29.9** | **2.5** | **12.6** |

**Table 4.7:** WMT '17 BLEU scores

Analysing the overall results achieved by CycleNLPGAN and the main competitors, it is clearly noticeable the difference among them. CycleNLPGAN is not able to reach state-of-the-art model in most of the language pairs. The two main causes of this verdict are the difference in the datasets used in the training phase and the specificity of the training of the other models, completely aimed at addressing the translation task. This is because models such as native MarianMT ([51]) perform better. Indeed they have been designed to exclusively translate corpora and have been trained on WMT datasets, meanwhile CycleNLPGAN has been trained on parallel datasets that does not include WMT dataset. Moreover, the amount of data used for CycleNLPGAN training is much lower, and the duration in epochs of the training is lower than other models.

Finally, these outcomes confirm the assumptions made initially, claiming that the model reaches a compromise and an equilibrium between alignment and translation tasks.

### Qualitative results

In the following section, some sentences taken from the abstract and their translation from English to French and from English to German are reported. In order to evaluate the cycle consistency contribution, the back-translation from German to English is shown as well. Moreover, a brief examination of the quality of both translation is given.

As shown in table 4.8, outcomes obtained traducing English sentences in German and French produce sentences that have the same meaning as in the starting sentences. The quality of translation is adequate and the overall meaning of each sentence is not altered. The effectiveness of CycleNLPGAN approach is confirmed by back-translation results. In fact, sentences obtained translating back in English generated German sentences shows that CycleNLPGAN maintains the English original sentences structure as unchanged. In some cases, the back translation retrieves a sentence that is identical to the starting one, and this is due to the introduction of the cycle consistency. In other sentences, the words used to build

them are different, but the meaning is identical.

However, in some cases, the translated sentences contain words that are not properly correct. Even if the resulting sentence holds his starting meaning, words used to produce it may be misleading and inaccurate. As instance, the term *"cycle-consistent"* is translated as *"bikekonsistenter"*, and finally it is back-translated as *"bike-consistent"*. The English to German translation produces a term that is the composition of cycle and consistent, but *cycle* term is translated as *bike*, that produces a misleading sentence. The back-translation produces a sentence that properly traduce the term *bike*, but it has not the same meaning as the starting sentence. The main cause of this type of translation errors is related to the generic nature of the training dataset, meanwhile this translation requires a dataset specialized on technical words and sentences. In fact, the model translates the word *cycle* in the the term corresponding to *bicycle* in German.

| ***Abstract excerpt*** |
|---|
| This thesis presents CycleNLPGAN, a deep learning architecture that introduces an innovative approach to sentence encoding and alignment. It allows the definition of a latent vector space shared across a pair of languages. The model is jointly trained to perform neural machine translation from a source language $A$ and to target language $B$. It generates a shared aligned vector space suitable for machine translation from a source language to a target one and vice versa. The architecture is based on a CycleGAN, a Computer Vision model that address image-to-image translation using cycle-consistent dynamics. It enforce the robustness of the resulting model and the quality of produced data. The architecture is defined using a cycle consistency loss, an approach used in neural machine translation and in domain adaptation models. |
| ***French Translation using CycleNLPGAN*** |
| Cette thèse présente CycleNLPGAN, une architecture d'apprentissage profond qui introduit une approche novatrice de l'encodage et de l'alignement des phrases. Il permet la définition d'un espace vectoriel latent partagé à travers une paire de langages. Le modèle est formé conjointement pour exécuter la traduction par machine neuronale à partir d'une langue source A et à la langue cible B. Il génère un espace vectoriel aligné partagé adapté à la traduction automatique à partir d'un langage source vers une cible et vice versa. L'architecture est basée sur un CycleGAN, un modèle de vision informatique qui aborde la traduction image à image en utilisant la dynamique conséquente. Il fait respecter la robustesse du modèle résultant et la qualité des données produites., L'architecture est définie à l'aide d'une perte de cohérence de cycle, d'une approche utilisée dans la traduction de la machine neurale et dans les modèles d'adaptation de domaine. |
| ***German Translation using CycleNLPGAN*** |
| Diese These präsentiert cycleNLPGAN, eine Deep-Learning- Architektur, die einen innovativen Ansatz zur Satzkodierung und Ausrichtung einführt. Es erlaubt die Definition eines latenten Vektorraums, der über ein Paar Sprachen geteilt wird. Das Modell wird gemeinsam trainiert, um neuronale maschinelle Übersetzungen aus einer Quellsprache A und zur Zielsprache B durchzuführen. Es erzeugt einen gemeinschaftlich ausgerichteten Vektorraum, der für die maschinelle Übersetzung von einer Quellsprache zu einem Zielraum geeignet ist und umgekehrt. Die Architektur basiert auf einem cycleGAN, einem Computer-Vision-Modell, das Bild-zu-Bild-Übersetzung mit bikekonsistenter Dynamik ansprechen. Es setzt die Robustheit des entstehenden Modells und die Qualität der produzierten Daten durch. Die Architektur wird durch einen Zykluskonsistenzverlust definiert, einen Ansatz, der in der neuronalen maschinellen Übersetzung und in Domain-Anpassungsmodellen verwendet wird. |
| ***Back Translation using CycleNLPGAN (from generated German sentences)*** |
| This thesis presents cycleNLPGAN, a deep learning architecture that introduces an innovative approach to setting coding and aligning. It allows the definition of a latent vector space which is shared over a pair of languages. The model is trained together to perform neural machine translations from a source language A and to the target language B. It creates a jointly oriented vector space suitable for machine translation from a source language to a target space and vice versa. The architecture is based on a cycleGAN, a computer vision model, addressing image-to-image translation with bike-consistent dynamics. It enhances the robustness of the resulting model and the quality of the data produced. The architecture is defined by a cycle consistency loss, an approach used in neural machine translation and in domain adaptation models. |

**Table 4.8:** Qualitative evaluation of CycleNLPGAN model by translating an excerpt of the abstract of this master thesis.

# Chapter 5

# Conclusions

Artificial Intelligence and Machine Learning research have seen an increasing interest during years. The most important and innovative techniques are related to Computer Vision and Natural Language Processing fields. Computer Vision is a branch of Machine Learning that aims to understand, generate and classify images, meanwhile Natural Language Processing addresses the understanding, the learning and the modeling of natural languages. Both of them resolve a wide range of tasks, and the produced models obtain astonishing results in most of them. It often happens that these two fields address similar tasks or tasks that can be related among them. In some cases, state-of-the-art models in both fields use similar techniques, as in some cases approaches used in one field are applied in the other one, producing impressive outcomes.

The model produced on this thesis, called CycleNLPGAN, addresses sentence alignment and neural machine translation, two sub-tasks of Natural Language Processing field. The main objective of this thesis is to produce a model able to align sentences with similar meanings using a cycle consistent approach. Jointly, the model performs translation among languages, trying to produce high quality translated sentences. Starting from a language couple, such as $French - English$ pair, the model is able to produce a shared latent vector space. Using this approach, the sentence embeddings produced by the resulting model for sentences in different languages that have a similar meaning, called parallel sentences, are close among them and belong to the same shared latent vector space. The resulting model exploits the cycle consistency loss, which is computed during the training phase in order to define the similarity among the starting sentences and the result of their translation in the other language followed by a re-translation, or back translation, in the starting language. The cycle loss aims to enforce the quality of the translation, penalizing the networks used during the translation and the back translation and updating them jointly. Moreover, this loss aims to increase the correlation among models that translate from a language to another and vice versa. This specific

approach is taken from a Computer Vision model called CycleGAN [23], that addresses domain adaptation task. In fact, the degree of similarity among the domain adaptation in Computer Vision and the neural machine translation in Natural Language Processing allows to reproduce the CycleGAN approach in Natural Language Processing. Both CycleGAN and CycleNLPGAN are trained using the generative adversarial approach, and contain two generators and two discriminators. Generators take a sentence in a source language as input, produce the sentence embedding and produce the sentence as output in a target language. Unlike traditional CycleGAN, this approach is trained in a supervised fashion in order to enforce the learning quality and to reduce the possibility of divergence during the training.

Several configuration of CycleNLPGAN have been trained for a limited amount of sentences in order to find out the best approach that should be used during the whole training. More in detail, the loss function, the training approach for generators and the set of hyper-parameters have been defined. Once the best configuration is chosen, some experiments aims to find the best value for each hyper-parameter.

The best configuration of hyper-parameters is used to train CycleNLPGAN using several language pairs, such as *French-English*, *German-English*, *Russian-English* and *Chinese-English*. After the training, each model is evaluated on both alignment and neural machine translation tasks. The addressed tasks are BUCC bitext retrieval and WMT news translation. The BUCC results are lower than the ones obtained by competitors model such as [45], [57] and [44]. One of the main disadvantage of CycleNLPGAN with respect to state-of-the-art models is the reduced amount of training data used and the lower number of iterations executed during the training. The Moreover, CycleNLPGAN produces lower dimension sentence embeddings with respect to competitors ones. In fact, CycleNLPGAN generators, that are MarianMT models, produce a 512-length sentence embedding vector, meanwhile competitors models produce higher dimensional vector that are composed by 768 and 1024 elements.

CycleNLPGAN achieves outstanding results with respect to state-of-the-art models on WMT task. Even if translation results are not always higher than competitors ones, the model can be considered adequate and accurate on this task despite not being trained on the reference dataset. In fact, CycleNLPGAN is trained on a set of data that does not contain WMT training dataset. This choice was constrained by the elevate dimension of the WMT training set. The training of CycleNLPGAN with this data would have required an unavailable set of computational resources and an unaffordable amount of time to complete the training for an adequate amount of epochs. The assumption related to the inadequate dataset and to the reduced amount of sentences is confirmed by results obtained for different language pairs. Models trained on a higher amount of data,

such as $German-English$ and $French-English$ models, reach results comparable to competitors, meanwhile models that uses a reduced number of sentences during the training, such as $Chinese-English$ model, obtain inaccurate results and low-quality translation.

Finally, CycleNLPGAN model is able to obtain a compromise among sentence and language alignment and translation quality, and the resulting model can be used as starting point for future works that aim to increase performances in both tasks.

## 5.1 Future Works

The master thesis work requires a significant effort, and several possible improvements have emerged during the implementation of the architecture described in previous chapters.

At first, several models may be used as generator. In this implementation, MarianMT was chosen as generator model because of the simple structure and the presence of several pretrained language pairs configurations. Other encoder-decoder model may be used to verify the robustness and the flexibility of CycleNLPGAN architecture. Moreover, another point of interest related to the generator choice is the dimension of the produced embedding. MarianMT model produces a 512-length embedding vector, but several models produce higher dimension vectors. Model like LASER [44] produce 1024-length vectors, and it may be possible that results are more sparse, meanwhile in MarianMT model produce embeddings that are more dense due to the limited dimension. The introduction of models that produce higher dimensional vectors may increase CycleNLPGAN results.

Another aspect that caught attention during implementation is related to results obtained during the loss function. During analysis, MSE loss seems more adequate to evaluate sentence embedding loss, but the graph reported in figure 4.6 obtained using cosine loss shows a behaviour during the training that is completely different if compared to the MSE loss one. Due to the limited amount of resources, the configuration cannot be trained for the whole amount of epochs. A more detailed analysis on the contribution given by the loss function, performed after training the model using the whole dataset for more than one epoch, may lead to a more accurate result.

The third implementation that may improve results is related to the massive usage of back-translation. During the training, the back-translation is used in certain phases to increase reliability and robustness, but its impact is partial and less impactful related to others. Analysing the quality of the back-translation of the real sentence, BLEU scores report a massive increase in the quality of the reconstructed sentences. Results obtained using the back-translation, that are not

reported in this thesis, are higher than the one produced during the first translation. Even if this aspect may be useful to increase the quality of the architecture, the amount of research related to this topic and the effort required to implement improvements are out of the scope of this thesis. It may be useful, in future works, to analyse more accurately this behaviour and to try to use back-translation in other training phases.

The last activity that may be useful in future works is related to the evaluation of the models. CycleNLPGAN architecture obtains results that are comparable to distillation approach [45] and mUSE [22] ones on the sentence alignment task. State-of-the-art models outperform them because they are built to address this specific task, but results obtained by models such as LASER [44] and [57] on semantic textual similarity task are lower than competitors. This is because they look for the perfect translation, meanwhile mUSE and distillation approach models look for sentences that have similar meanings. The same may be true for CycleNLPGAN, but to be certain of this, it should be useful to test the architecture in a semantic textual similarity task. The best way to address semantic textual similarity is performed evaluating the model on the multilingual STS 2017 dataset [69], which contains annotated pairs for English-English, Arabic-Arabic, English-Arabic, Spanish-Spanish, English-Spanish and English-Turkish. Given a pair of sentences, the model has to assign a score indicating their semantic similarity. A score of 0 indicates no relation among sentences, meanwhile 5 indicates semantically equivalent sentences. Due to lack of time it was not possible to evaluate CycleNLPGAN during this thesis work, but the results on this task may demonstrate its competitiveness and its flexibility on this task as well.

# Bibliography


[1] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. «Deep blue». In: *Artificial intelligence* 134.1-2 (2002), pp. 57–83 (cit. on p. 1).

[2] David H Hubel and Torsten N Wiesel. «Receptive fields, binocular interaction and functional architecture in the cat's visual cortex». In: *The Journal of physiology* 160.1 (1962), pp. 106–154 (cit. on p. 2).

[3] Olga Russakovsky et al. «Imagenet large scale visual recognition challenge». In: *International journal of computer vision* 115.3 (2015), pp. 211–252 (cit. on p. 3).

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105 (cit. on p. 3).

[5] Alan M Turing. «Computing machinery and intelligence». In: *Parsing the turing test*. Springer, 2009, pp. 23–65 (cit. on p. 3).

[6] Li Deng and Yang Liu. *Deep learning in natural language processing*. Springer, 2018 (cit. on p. 3).

[7] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984 (cit. on p. 3).

[8] Frank Rosenblatt. «The perceptron: A probabilistic model for information storage and organization in the brain». In: *Psychological review* 65.6 (1958), pp. 386–408 (cit. on pp. 3, 12).

[9] Diederik P Kingma and Max Welling. «Auto-encoding variational bayes». In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on p. 6).

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. «Generative adversarial nets». In: *Advances in neural information processing systems*. 2014, pp. 2672–2680 (cit. on pp. 6, 9, 15, 28).

[11] Tin Kam Ho. «Random decision forests». In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282 (cit. on p. 6).

[12] Corinna Cortes and Vladimir Vapnik. «Support-vector networks». In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on p. 6).

[13] Abigail See, Peter J Liu, and Christopher D Manning. «Get To The Point: Summarization with Pointer-Generator Networks». In: () (cit. on p. 6).

[14] Xingxing Zhang, Furu Wei, and Ming Zhou. «HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization». In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 5059–5069 (cit. on p. 6).

[15] Romain Paulus, Caiming Xiong, and Richard Socher. «A Deep Reinforced Model for Abstractive Summarization». In: *arXiv e-prints* (2017), arXiv–1705 (cit. on p. 6).

[16] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. «LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention». In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 6442–6454 (cit. on pp. 7, 8).

[17] Zhiheng Huang, Wei Xu, and Kai Yu. «Bidirectional LSTM-CRF Models for Sequence Tagging». In: () (cit. on p. 7).

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Advances in neural information processing systems*. 2017, pp. 5998–6008 (cit. on pp. 7, 8, 20–23).

[19] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. «Sequence to Sequence Learning with Neural Networks». In: () (cit. on p. 7).

[20] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. «XLNet: Generalized Autoregressive Pretraining for Language Understanding». In: () (cit. on p. 8).

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «Bert: Pre-training of deep bidirectional transformers for language understanding». In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on pp. 8, 11, 20, 23).

[22] Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. «Word translation without parallel data». In: *International Conference on Learning Representations*. 2018 (cit. on pp. 9, 28, 29, 32, 73, 75, 84).

[23] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. «Unpaired image-to-image translation using cycle-consistent adversarial networks». In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232 (cit. on pp. 9, 35, 36, 40, 52, 82).

[24] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. «BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension». In: *arXiv* (2019), arXiv–1910 (cit. on pp. 11, 20, 24, 25, 42).

[25] Warren S McCulloch and Walter Pitts. «A logical calculus of the ideas immanent in nervous activity». In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133 (cit. on p. 12).

[26] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985 (cit. on p. 14).

[27] Sepp Hochreiter and Jürgen Schmidhuber. «Long short-term memory». In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 14).

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. «Efficient Estimation of Word Representations in Vector Space». In: *arXiv preprint arXiv:1301.3781* (2013) (cit. on pp. 17–19, 27, 28).

[29] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. «Enriching word vectors with subword information». In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146 (cit. on pp. 19, 29).

[30] Jeffrey Pennington, Richard Socher, and Christopher D Manning. «Glove: Global vectors for word representation». In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543 (cit. on p. 19).

[31] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. «Deep contextualized word representations». In: *Proceedings of NAACL-HLT*. 2018, pp. 2227–2237 (cit. on pp. 20, 22).

[32] Yinhan Liu et al. «RoBERTa: A Robustly Optimized BERT Pretraining Approach». In: *arXiv* (2019), arXiv–1907 (cit. on p. 23).

[33] Alexis Conneau et al. «Unsupervised Cross-lingual Representation Learning at Scale». In: *arXiv* (2019), arXiv–1911 (cit. on pp. 23, 73).

[34] Nils Reimers and Iryna Gurevych. «Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks». In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: `https://arxiv.org/abs/1908.10084` (cit. on pp. 23, 30, 42, 73).

[35] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. «MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices». In: *arXiv* (2020), arXiv–2004 (cit. on p. 23).

[36] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. «ALBERT: A Lite BERT for Self-supervised Learning of Language Representations». In: *International Conference on Learning Representations*. 2019 (cit. on p. 23).

[37] Telmo Pires, Eva Schlinger, and Dan Garrette. «How Multilingual is Multilingual BERT?» In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 4996–5001 (cit. on pp. 26, 73).

[38] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. «Exploiting Similarities among Languages for Machine Translation». In: () (cit. on pp. 26, 27).

[39] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. «Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation». In: *HLT-NAACL*. 2015 (cit. on p. 27).

[40] Georgiana Dinu and Marco Baroni. «Improving zero-shot learning by mitigating the hubness problem». In: *arXiv preprint arXiv:1412.6568* (2014) (cit. on p. 27).

[41] Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. «Offline bilingual word vectors, orthogonal transformations and the inverted softmax». In: (2016) (cit. on p. 27).

[42] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. «Learning bilingual word embeddings with (almost) no bilingual data». In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 451–462 (cit. on p. 27).

[43] Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. «Adversarial training for unsupervised bilingual lexicon induction». In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 1959–1970 (cit. on pp. 28, 32).

[44] Mikel Artetxe and Holger Schwenk. «Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond». In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 597–610 (cit. on pp. 29, 30, 32, 39, 60, 73, 82–84).

[45] Nils Reimers and Iryna Gurevych. «Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation». In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2020. URL: `https://arxiv.org/abs/2004.09813` (cit. on pp. 30–32, 39, 42, 46, 62, 73, 75, 82, 84).

[46] Steven Cao, Nikita Kitaev, and Dan Klein. «Multilingual Alignment of Contextual Word Representations». In: *International Conference on Learning Representations*. 2019 (cit. on p. 31).

[47] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. «Domain adaptation via transfer component analysis». In: *IEEE Transactions on Neural Networks* 22.2 (2010), pp. 199–210 (cit. on p. 34).

[48] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. «Domain-adversarial training of neural networks». In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2096–2030 (cit. on p. 34).

[49] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. «Adversarial discriminative domain adaptation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176 (cit. on pp. 34, 35).

[50] Ming-Yu Liu and Oncel Tuzel. «Coupled Generative Adversarial Networks». In: () (cit. on p. 35).

[51] Marcin Junczys-Dowmunt et al. «Marian: Fast Neural Machine Translation in C++». In: *Proceedings of ACL 2018, System Demonstrations*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 116–121. URL: http://www.aclweb.org/anthology/P18-4020 (cit. on pp. 42, 76–78).

[52] Jörg Tiedemann. «Parallel Data, Tools and Interfaces in OPUS.» In: (cit. on pp. 54, 55).

[53] Željko Agic and Ivan Vulic. «JW300: A wide-coverage parallel corpus for low-resource languages». In: (2020) (cit. on p. 55).

[54] Nils Reimers and Iryna Gurevych. «Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation». In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2020. URL: https://arxiv.org/abs/2004.09813 (cit. on p. 55).

[55] Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. «Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia». In: *arXiv preprint arXiv:1907.05791* (2019) (cit. on p. 55).

[56] LoÃ¯c Barrault et al., eds. *Proceedings of the Fifth Conference on Machine Translation*. Online: Association for Computational Linguistics, Nov. 2020. URL: https://www.aclweb.org/anthology/2020.wmt-1 (cit. on p. 61).

[57] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. «Language-agnostic bert sentence embedding». In: *arXiv preprint arXiv:2007.01852* (2020) (cit. on pp. 73, 82, 84).

[58] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. «Multilingual denoising pre-training for neural machine translation». In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 726–742 (cit. on pp. 76–78).

[59] Sho Takase and Shun Kiyono. «Lessons on Parameter Sharing across Layers in Transformers». In: *arXiv e-prints* (2021), arXiv–2104 (cit. on p. 76).

[60] Xiang Kong, Zhisong Zhang, and Eduard Hovy. «Incorporating a Local Translation Mechanism into Non-autoregressive Translation». In: *arXiv e-prints* (2020), arXiv–2011 (cit. on p. 76).

[61] Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. «Very Deep Transformers for Neural Machine Translation». In: *arXiv e-prints* (2020), arXiv–2008 (cit. on p. 76).

[62] Artetxe Mikel, Labaka Gorka, Agirre Eneko, et al. «Unsupervised statistical machine translation». In: Brussels; Proceedings of the 2018 Conference on Empirical Methods in Natural ... 2018 (cit. on p. 76).

[63] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. «Neural Machine Translation In Linear Time». In: () (cit. on p. 77).

[64] Rico Sennrich, Barry Haddow, and Alexandra Birch. «Neural Machine Translation of Rare Words with Subword Units». In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 1715–1725 (cit. on p. 77).

[65] Yiren Wang, Yingce Xia, Tianyu He, Fei Tian, Tao Qin, Cheng Xiang Zhai, and Tie Yan Liu. «Multi-agent dual learning». In: *7th International Conference on Learning Representations, ICLR 2019*. 2019 (cit. on p. 77).

[66] Rico Sennrich, Barry Haddow, and Alexandra Birch. «Edinburgh Neural Machine Translation Systems for WMT 16». In: () (cit. on p. 77).

[67] Yi Tay, Mostafa Dehghani, Vamsi Aribandi, Jai Gupta, Philip Pham, Zhen Qin, Dara Bahri, Da-Cheng Juan, and Donald Metzler. «OmniNet: Omnidirectional Representations from Transformers». In: *arXiv e-prints* (2021), arXiv–2103 (cit. on p. 78).

[68] Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A Smith. «Finetuning Pretrained Transformers into RNNs». In: *arXiv e-prints* (2021), arXiv–2103 (cit. on p. 78).

[69] Daniel Cera, Mona Diabb, Eneko Agirrec, Inigo Lopez-Gazpioc, Lucia Speciad, and Basque Country Donostia. «SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation». In: () (cit. on p. 84).