UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Image Captioning using pre-trained GPT-2 models

End of Degree Project

Bachelor's Degree in Data Science

AUTHOR: García Gilabert, Javier

Tutor: Casacuberta Nolla, Francisco

ACADEMIC YEAR: 2021/2022

Acknowledgments

To my family, for their unrivaled love and support; to my college professors for their advices; and to all of my friends for being there!

Resum

L'objectiu del subtitulat d'imatges és descriure el contingut d'una imatge en llenguatge natural. A causa de l'èxit de diverses arquitectures d'aprenentatge profund, aquest repte que combina el processament d'imatges i el llenguatge ha despertat molta atenció en els darrers anys. L'objectiu principal d'aquest projecte de fi de grau és crear models basats en xarxes neuronals més precisos per subtitular imatges.

S'han desenvolupat diversos models basats en xarxes neuronals a partir de la xarxa neuronal CLIP, que ofereix representacions similars atesa una imatge i la seva descripció. Això, juntament amb GPT-2, un model de llenguatge, s'utilitza per proposar diversos dissenys de xarxes neuronals. El conjunt de dades MSCOCO, que consisteix en escenes quotidianes complexes amb descripcions en llenguatge natural, es farà servir per comparar diferents arquitectures.

Paraules clau: Descripció d'imatges, models preentrenats d'aprenentatge profund, GPT-2, CLIP

Resumen

El objetivo del subtitulado de imágenes es describir el contenido de una imagen en lenguaje natural. Debido al éxito de varias arquitecturas de aprendizaje profundo, este reto que combina el procesamiento de imágenes y el lenguaje ha despertado mucha atención en los últimos años. El objetivo principal de este proyecto de fin de grado es crear modelos basados en redes neuronales más precisos para el subtitulado de imágenes.

Se han desarrollado varios modelos basados en redes neuronales a partir de la red neuronal CLIP, que ofrece representaciones similares dada una imagen y su respectiva descripción. Esto, junto con GPT-2, un modelo de lenguaje, se utiliza para proponer varios diseños de redes neuronales. El conjunto de datos MSCOCO, que consiste en escenas cotidianas complejas con descripciones en lenguaje natural, se utilizará para comparar diferentes arquitecturas.

Palabras clave: Descripción de imágenes, modelos de redes profundas pre-entrenados, GPT-2, CLIP

Abstract

The objective of image captioning is to describe the content of an image in natural language. Due to the success of various deep learning architectures, this challenge that combines picture and language processing has aroused a lot of attention in recent years. The key goal for this end grade project is to create more accurate neural machine models for image captioning.

Several neural network-based models are built based on the CLIP neural network, which offers similar embeddings given an image and a descriptive caption. This, in conjunction with GPT-2, a language model, is used to propose various deep learning designs. The MSCOCO dataset, which consists of complex everyday scenes with natural language descriptions, will be used to compare different architectures.

Key words: Image captioning, pre-trained neural models, GPT-2, CLIP

Contents

List of Figures

List of Tables

Notation

Symbol	Description
$\mathbf{A}, \ldots, \mathbf{Z}$	Matrices.
$\odot$	Element-wise product.
$*$	Product.
$\langle \cdot, \cdot \rangle$	Scalar product.
$\lvert \cdot \rvert$	Cardinality of a sequence.
$\mathrm{p}(\cdot)$	Probability distribution.
ℓ	Loss function.
$\nabla_w \, \ell(w)$	Gradient of the function ℓ with respect to the parameters w.
$\texttt{CLIP}_{IE}$	CLIP image encoder
$\texttt{CLIP}_{TE}$	CLIP text encoder

Acronyms

AI artificial intelligence

BLEU bilingual evaluation understudy

CIDER consensus-based image description evaluation

CLIP contrastive language–image pre-training

CNN convolutional neural network

CV computer vision

GA genetic algorithm

LSTM long short-term memory

METEOR metric for evaluation of translation with explicit ordering

ML machine learning

MLP multilayer perceptron

NLP natural language processing

RNN recurrent neural network

ROUGE recall-oriented understudy for gisting evaluation

SPICE semantic propositional image caption evaluation

CHAPTER 1
Introduction

The task of producing a human-readable description of an image is known as image captioning. This can be a difficult task, as it requires the ability to understand both the image content and the linguistic structure of the language used to describe it. Image captioning is a popular artificial intelligence (AI) field that falls somewhere between computer vision (CV) and natural language processing (NLP). Due to the rising availability of vast amounts of data and the development of powerful algorithms in the field of AI, both fields have gained a lot of attention in recent years and as a result image captioning has unsurprisingly seen a rapid growth.

Image captioning can be used for different tasks such as automatic image indexing which is the task of creating a database of images that can be searched and retrieved quickly. It can also be used for image annotation, which is the task of adding information about an image to a data corpus. Image annotation can be used for a variety of purposes, big companies such as Facebook or Google generate descriptions from images for use in social media, search results or product labels.

Apart from being used for specific tasks, image captioning can also be used for helping people with disabilities. Image captioning, for example, can be used to supplement the visual experience of those who are blind or have impaired vision.

The human eye evolved to comprehend light waves that strike the retina and create images on the back of the eye, the brain then interprets these images to create what we see. This ability to see, dates back to the early days of human evolution. Other animals evolved differently, for example, bats use sound waves to interpret their surroundings via a sonar-like system. The South American weakly electric fish live in muddy waters where vision is limited, they developed a system to perceive electric fields in the water in order to navigate. This difference in how different animals evolve has led to the different ways they perceive and use their surroundings. Computers, on the other hand, require a separate method to comprehend visuals that try to mimic the human brain.

For a computer, understanding an image is primarily reliant on extracting important image features from the image. A specific property of an image, such as brightness, color, or spatial placement, is referred to as an image feature. After obtaining picture attributes, the computer can utilize them to recognize objects, sceneries, and patterns in the image. Image features can be obtained in a variety of ways, which can be split into two categories: machine learning-based approaches and deep machine learning-based techniques.

In machine learning-based approaches, hand-crafted characteristics are retrieved from a picture and fed into a classifier that will classify an object. These strategies, are extremely dependent on the task at hand and do not have a great degree of generalization. In deep machine learning-based techniques, on the other hand, a model automatically learns fea-

tures from training data. This type of technique has shown to be more accurate than machine learning-based strategies at generalizing to new tasks. However, deep learning models, are more computationally expensive to train.

In this work several deep learning models are developed for the task of image captioning. The models are trained on the MSCOCO dataset which consists of images and captions of various daily activities. Because deep learning models are costly to train, some of the proposed models are based on pre-trained models, which means that training them is less expensive than training a model from scratch.

In conclusion, image captioning is a field that is constantly growing, as new methods are developed to improve accuracy and efficiency. The deep learning models developed in this work provide a promising approach to tackling the task of image captioning.

1.1 Motivation

Last year, I began a Research Collaboration Fellowship on image captioning in the Pattern Recognition and Human Language Technology research center, which is also the main focus of this work. I decided to chose this topic for my final degree work as I have always been interested in the field of natural language processing and computer vision and I wanted to explore in more depth what I have seen during my degree about both fields.

In addition, numerous image captioning models have lately been employed to assist blind persons in understanding their environment. This type of model can help those with vision impairments enhance their quality of life. By improving the accuracy of these models, we will be able to improve the quality of life of persons who are blind.

1.2 Objectives

The primary goal of this project is to develop different AI models for image captioning. To this end, we will present different deep learning architectures for image captioning. These models will be validated and trained by using a dataset of images containing every day scenes.

Moreover, during this work it will be studied the state of the art models in the image captioning field based on artificial neural networks. The aim is to better understand the working principles of these models in order to improve the deep learning architectures.

In addition, the project will also investigate the use of pre-trained models for image captioning. To this purpose, different architectures will be proposed in order to take advantage of the pre-trained models. Finally, the results of this project will be compared in order to better understand the strengths and weaknesses of the different approaches.

1.3 Structure

The rest of this work is organized as follows:

- **Chapter 2:** Background

  This chapter includes a literature review of image captioning architectures that have recently been published and reflect the state of the art in this field. In addition, we will introduce some of the open problems in the field.

- **Chapter 3:** Dynamic Neural Networks

  In this chapter, we will go over what neural networks are, how they work, and how they are trained. We will also go through the most advanced deep learning architectures, which are the cornerstones of this final project's work.

- **Chapter 4:** Image Captioning

  In this chapter, we will go over several image captioning architectures that have been proposed. From this chapter, we will be able to understand the different approaches that have been used for image captioning.

- **Chapter 5:** Experimental framework

  This chapter covers all of the work's experimental aspects, including the data that was used, as well as the metrics and architectures that are proposed.

- **Chapter 6:** Results

  In this chapter, we will present the results of the experiments for the models proposed in chapter 5. We will compare the different models and analyze the results.

- **Chapter 7:** Conclusions

  This chapter summarizes the work done in this final degree project and points out the future lines of research that could be done in this area.

CHAPTER 2
Background

As previously said, around the dawn of the twenty-first century, there was a lot of study in the field of machine learning (ML), particularly deep learning (DL). Specifically, there have been countless articles on this field over the last decade, which, combined with the large amounts of data available and the computational power of current-generation machines, has led to rapid advancements in the field of ML. These advancements have, in turn, led to the development of powerful deep learning models that are able to learn from data in a more natural and effective way than traditional machine learning models.

Furthermore, deep learning models have found their way into a variety of other fields, including computer vision, natural language processing, sentiment analysis, recommender systems, and so on. And image captioning is no exception. In the following chapter we will present the main contributions in this field.

2.1 What is machine learning?

Machine learning (ML) is a subfield of AI that is focused on the development of algorithms that allow computers to learn from data without being explicitly programmed. Some of the main types of problems solved by ML according to the output domain are [19]:

1. **Unstructured prediction:** where the output is an atomic element, this can be a single value or a vector ($f : \mathcal{X} \rightarrow \mathbb{R}^d$). Depending on the form of this value, we can divide the unstructured prediction into two other categories:
    - **Regression:** the output domain is a real value. Generally, the aim is to predict the value of a target variable ($\mathcal{Y} \in \mathbb{R}$) based on a set of input variables ($\mathcal{X} \in \mathbb{R}^d$).
    - **Classification:** the output domain is finite . The aim is to assign an object to a class where classes are known beforehand.
2. **Structured prediction:** where the output has a structure. We understand a dependency between the object's elements as structure. Sequences, trees, and graphs are examples of structured objects.

Structured prediction is particularly relevant in the context of this final degree work. Sequence to sequence is a subclass of structured prediction where the purpose is to predict an output sequence from a sequence of inputs. Several NLP problems, such as machine translation, parsing, summarization, and speech recognition, can be expressed as

sequence-to-sequence tasks. Image captioning can also be seen as a sequence to sequence task where the input sequence is the image and the output sequence is the caption of the image.

2.2 Natural Language Processing

Natural language processing (NLP) is an area of linguistics, computer science, and artificial intelligence concerned with computer-human interaction, or more specifically, computers' capacity to interpret human language and extract information from text. The use of deep learning has been one of the major revolutions in NLP. Deep learning is being utilized in NLP to obtain state of the art results on a variety of tasks, including machine translation, sentiment analysis, and document classification.

The main architectures that are being used for NLP are recurrent neural networks (RNN), specifically long short-term memory (LSTM). RNN are more commonly used in NLP because they are able to capture the dependencies in a sequence of words. For instance, in a sentence such as "The cat sat on the mat", the word "cat" depends on the word "the" and the word "mat" depends on the word "sat". However, convolutional neural networks (CNN) are also used in the field of NLP albeit they are more used in the field of computer vision. Another architecture that is recently taking the lead is the Transformer architecture which was proposed in 2017 by Vaswani et al. [20]. The Transformer architecture has been employed in numerous NLP tasks with great success. For instance, it was able to achieve a new state of the art result on the WMT English-to-French translation task [21].

2.3 State of the art in Image Captioning

Image captioning was largely done using traditional machine learning-based techniques prior to the deep learning revolution. In these approaches, image features were extracted from the image using methods such as the Histogram of Oriented Gradients (HOG) [22]. Image features were used to feed a classifier (e.g. Support Vector Machine (SVM)) to classify the objects in the image. However, these approaches are no longer used for image captioning after deep learning methods were found to be much more effective.

Many image captioning approaches are now inspired by translation systems making use of a visual encoder and a linguistic decoder. Translation models in the field of automatic translation use attention-based architectures to translate a sentence, which means that the context of the sentence is taken into account when producing a translation. When these models were proposed, the models might better understand the meaning of a word by looking at other words in the phrase, this led to the development of models that could attend to different parts of the input sentence at the same time such as Transformer models [20] which are the state of the art in the field of translation. This is also utilized in image captioning systems since the system must identify which are the most significant items of an image to look at in order to offer an accurate description of it. Image captioning models resemble translation architectures as the task of describing an image is similar to "translating" an image to a sentence [10].

In prior work, the decision on what to look at an image was made during the encoding stage by using a convolutional neural network, and a recurrent neural network with an attention mechanism was employed as a decoder to generate a caption [10, 23, 24, 25]. These family of approaches are known as *encoder-decoder* methods as the input image is encoded into an intermediate representation of the image contents and then decoded to words. For instance, Vinyals et al. [23] employed a CNN pre-trained on ImageNet

[26] to encode an image and a LSTM network, a type of RNN, to decode the image and generate the caption. The encoding of the image can be obtained in different ways, in [23] the authors use a single feature extracted by using a CNN. However, this approach can ignore different regions of the image, other authors propose to use multiple features from different regions of the image. The regions can be uniformly sampled [10] or selected by an object detector [25].

These architectures, on the other hand, were unable to successfully relate the various parts of an image. Moreover, RNN decoders for image captioning have difficulty memorizing the inputs in earlier steps because to their recurrent nature, and as a result, RNN decoders for image captioning tend to generate captions without regard to the visual cues [27].

In the field of natural language processing, a similar challenge of relating words in a sentence was partially solved by the Transformer architecture [20]. In the encoding stage, some techniques have recently utilised the vanilla Transformer architecture to relate the regions of an image [28, 29, 27]. In [29] the authors use a Inception-ResNet-v2 [30] as the CNN that takes a image and outputs a sequence of image embeddings that are used to feed the Transformer model.

Previous techniques have relied on an *encoder-decoder* paradigm, in which an encoder extracts the image's visual content and a textual decoder generates the image's caption. Despite the fact that these strategies have shown to be quite effective, they require days of training on a large dataset in order to achieve good results. Recently, a new paradigm for image captioning was proposed, in which pre-trained language models are used to generate the caption of an image referred to as *lightweight* captioning models.

Language models are a type of AI model that are trained on large text datasets in order to predict the next word in a sentence. For instance, GPT-3 [31], proposed by OpenAI in 2020, has demonstrated to be the state of the art in natural language understanding. In [32, 11, 33, 34] a language model (e.g. GPT-2 [35]) is used to generate the next word of a caption from a set of initial context tokens and CLIP (Contrastive Language–Image Pre-training) [7] is used to extract the visual content from the image. CLIP was also proposed by OpenAI in 2021 which joinlty represents images and text descriptions and is based on the Transformer architecture. It is trained to embed images and text into vectors that will be similar if the image is related to the text. CLIP has shown to be able to match the accuracy of the state of the art models in object classification by finding the best match among statements of the type "This is an image of X" given an image. CLIP has also been useful in text-to-image task, as shown recently in [36] where they used a Generative Adversarial Network (GAN) [37] to produce images from text descriptions.

In [32] CLIP is used to extract the visual information from an image which is used to generate different context tokens by using a mapping network (e.g. Multi-Layer Perceptron, Transformer). These context tokens will be used to tame the language model to generate the caption of the image. Similarly, in [11] GPT-2 is used as the language model, however, in this case context tokens are not generated by a mapping network but a genetic algorithm (GA) is used to find the best context tokens starting from an initial population randomly initialized.

Image captioning systems are traditionally evaluated with reference based metrics, this is, metrics that use sentences that have been annotated by different human annotators after seeing the images. The most common metric is the Bilingual Evaluation Understudy metric (BLEU) [13] which was proposed in 2002 in the field of neural machine translation. Other metrics have been proposed specifically for image captioning such as the Semantic Propositional Image Caption Evaluation measure (SPICE) [12] or the Consensus-based Image Description Evaluation score (CIDEr) [16]. However, these measures necessitate

human references, which is a time-consuming task and also limits the number of references that can be obtained for each image. The MSCOCO dataset [38], for example, has five references per image. Recently, a novel approach to evaluate image captioning frameworks without the need of references was proposed in [17]. In this work, a new metric called CLIPScore is proposed which outperforms existing reference-based metrics like CIDEr and SPICE in terms of correlation with human judgments. CLIPScore is an unsupervised metric of the agreement between the predicted and the true captions of an image. The higher the score, the more accurate the predicted captions are. CLIPScore uses the CLIP model to embed the image and the text, then the cosine similarity between the embeddings is computed. The supervised version of CLIPScore is called RefCLIPScore which is also presented in [17] and achieves even higher correlation with human judgments.

In conclusion, the deep learning revolution has led to the development of more effective image captioning models that are now inspired by translation systems. These *encoder-decoder* models use a visual encoder and a linguistic decoder, and attention-based architectures are used to relate the regions of an image. Recently a new paradigm for image captioning was proposed, in which pre-trained language models are used to generate the caption of an image referred to as *lightweight* captioning models. This in conjunction with the new CLIPScore metric, a new approach to evaluate image captioning frameworks without the need of references, are the key building blocks for this final degree project.

2.4 Problems in Image Captioning

Image captioning systems are frequently challenged with a variety of problems. In this section we will discuss some of them.

Traditional image captioning systems are trained to guess the next word based on the ground truth's previous words. When a new image is provided to the model to generate a description, the model depends on its own previous predictions. As a result, the model's output will become progressively erroneous over time since the model is exposed to training data rather than its own predictions, this problem is known as the Exposure Bias Problem [39]. In [40], Gu et al. attempted to solve this problem by optimizing the process of caption generation in testing time using a reinforcement learning approach. In [11] this problem was partially solved by minimizing a loss function that resembled the CLIPScore metric in testing time.

Models for image captioning are still susceptible to recognizing objects that are not in the scene, this problem is referred to as Object Hallucination [41]. MacLeod et al. [42] looked at how visually impaired persons react to image captions generated by image captioning systems. They discovered that Object Hallucination is an evident worry for many visually impaired people who value accuracy over coverage.

The Vanishing Gradient Problem is a phenomena that has been found in neural networks. The problem is that in the backpropagation method, which is used to train neural networks, the neural network's weights do not change after certain iterations because the value of the gradient is too small. The Exploding Gradient Problem is another issue related to gradients, where the gradient becomes too large and causes the weights in the neural network to change too quickly, often resulting in the network's learning becoming unstable.

2.5 Proposals

In this section we are going to present the different neural network architectures for image captioning that will be employed in this work.

Firstly, we will start with architectures based on the Transformer architecture. These architectures will take as input the output of several CNN. Specifically, the following CNN will be used: EfficientNetB0 [43], VGG16 [44] and ResNet50 [45].

Secondly, in order to address the Exposure Bias Problem, we will implement three different modifications based on the ClipCap paper [32] that will improve the generated caption by using the unsupervised CLIPScore metric in testing time.

The 2014 MSCOCO dataset [38] will be used to train and validate all of the models, and the results will be compared using different image captioning metrics.

CHAPTER 3
Dynamic Neural Networks

In this chapter, we will go over the most important aspects of neural networks. We present the most common neural network topologies, which serve as the basis for the models employed in this final degree project. Finally, we will define what is a language model.

3.1 Introduction

Artificial neural networks can be seen as a mathematical function that takes an input and produces an output. Neural networks were loosely inspired by the inner workings of the human brain. These models are composed of interconnected processing nodes called *neurons* that work together to solve specific problems.

The pioneering work of Warner S. McCulloch and Walter Pitts [46] in 1943 gave birth to neural networks. Since then, neural networks have undergone periods of disinterest and obscurity until the deep learning era. The history of neural networks can be divided in three waves.

The first wave occurred in the 1950's when Frank Rosenblatt proposed the first neural network called the Perceptron in 1958 [47], which was a more general computational model than McCulloch–Pitts work. Bernard Widrow and Ted Hoff proposed the Adeline method in 1960 [48], which was the first time that learning by gradient descent was used in the context of neural network models.

The second wave ocurred during the 1980s which was called *connectionism*. At that time, the development of neural networks was on the verge of stopping, but in 1986 the back-propagation algorithm [49] changed the scene. This algorithm allowed neural networks to learn faster and more accurately. In 1989, Yann LeCun and his peers proposed the first neural network for handwritten digit recognition, the convolutional neural network [50]. Also, recurrent neural networks were proposed in 1990 [51]. Albeit there was a lot of research during the second wave, the interest in neural networks decreased mainly because of the limitations in computational power.

The third wave, or deep learning era, started in mid 2000s and it is still ongoing. Neural networks showed a resurgence in popularity as a result of the use of graphics processing units (GPU) for training deeper neural networks. In 2012, Alex Krizhevsky and his peers proposed AlexNet [52], a deep convolutional neural network that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which is an annual competition that evaluates algorithms for object detection and image classification at large scale. The performance of AlexNet was significantly better than all the previous algorithms proposed

for this challenge. Since then deep learning has dominated the field of CV, NLP and many other fields.

3.2 Linear Discriminant Functions

Discriminant functions are the fundamental building blocks of neural networks. A linear discriminant function is a scalar product between an input vector x and a vector of parameters w which will be referred as weights. An independent term w_0 called bias is added to the scalar product.

$$\phi(x) = w^t x + w_0 = \sum_{i=0}^{d} w_i^t x_i + w_0 \tag{3.1}$$

When modeling complex relationships between variables, linear discriminant functions are limited to linear relationships, which might be a limiting factor. Nonlinear discriminant functions can be utilized to create more expressive models. A nonlinear discriminant function is a nonlinear function $\psi : \mathbb{R} \to \mathbb{R}$ that is applied to the output of the linear discriminant function.

$$\psi(\phi(x)) = \psi(w^t x + w_0) \tag{3.2}$$

Sometimes the bias term can be added to the vector of parameters w by modifying the input vector x (equation 3.3).

$$\hat{x} = (\mathbf{1}, x) \quad \hat{w} = (w_0,\ w) \tag{3.3}$$

Therefore, the linear discriminant function can be rewritten as follows:

$$\phi(x) = \hat{w}^t \hat{x} \tag{3.4}$$

3.3 Multilayer Perceptron

A single-layered multilayer perceptron (MLP) is shown in figure 3.1. It is composed of three different types of layers: input, output and hidden layers. The input layer is the layer that takes the input data. The output layer gives the final result. The hidden layer is responsible for learning the relationship between input and output and each hidden layer is composed of several discriminant functions. The output of each discriminant function is a linear combination of the inputs, followed by a nonlinear function, called activation function.

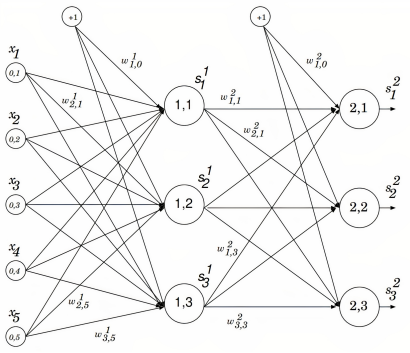


Figure 3.1: MLP with one hidden layer. **Image source:** [1]

The weights of the MLP in layer k which connect the neurons are defined as follows:

$$W^k = (w^k_{1,0}, \ldots, w^k_{M_k,\, M_{k-1}}) \ :\ w^k_{M_k, M_{k-1}} \in \mathbb{R},\ 1 \leq k \leq L \tag{3.5}$$

where L denotes the number of layers and M_k is the number of nodes in layer k. In figure 3.1, the MLP is composed of a single hidden layer. The output and hidden neurons are computed as follows:

$$s_j^1 = \psi(\phi_j^1) = \psi(\sum_i w_{j,i}^1 \, x_i), \; 1 \leq j \leq M_1 \quad s_j^2 = \psi(\phi_j^2) = \psi(\sum_i w_{j,i}^2 \, s_i^1), \; 1 \leq j \leq M_2 \quad (3.6)$$

where s_j^1 denote the output of the j^{th} neuron in the hidden layer, s_j^2 denote the output of the j^{th} neuron in the output layer, M_1 and M_2 denote the number of neurons in the hidden and output layers respectively and ψ is the activation function.

The computation of s_j^2 entails the feedforward part of the neural network. The gradient of a loss function is computed using the neural network's outputs, and this is utilized to update the parameters.

3.4 Activation Functions

Activation functions are linear or nonlinear transformations which are used in neural networks to determine the output of a neuron. These activation functions must be differentiable, as neural networks are generally tuned via gradient-based approaches. The most common activation functions are presented in figure 3.2. For instance, the logistic function takes a real number and outputs a value between 0 and 1.

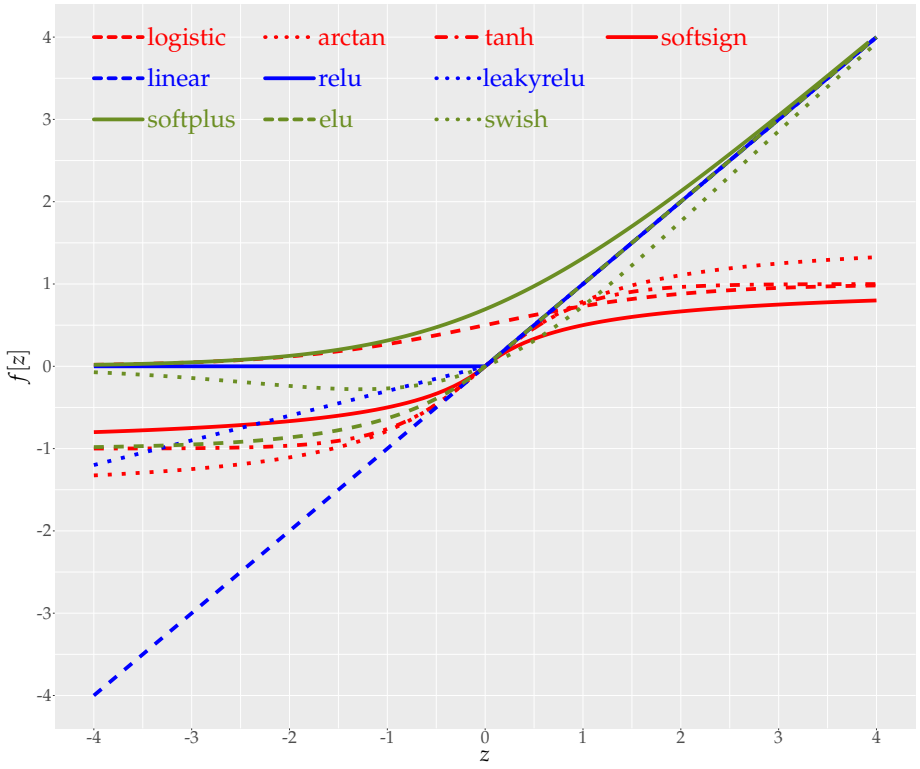


Figure 3.2: Most common activation functions. **Image source:** [2]

In this final degree project, language models will be used to approximate probability distributions. Language models estimate the probability of words in a vocabulary V given a sequence of context words. This is usually done by adding as many neurons as words in the vocabulary in the output layer of a neural network. Each node will denote

the probability of the i^{th} word. In order to generate a probability, that is, all elements must be between 0 and 1 and the sum of the elements must be 1, the softmax function is usually applied to the output layer. The softmax function is defined as:

$$\phi(x) = \frac{e^{x_i}}{\sum_{k=1}^{|x|} e^{x_k}},\ 1 \leq i \leq |x| \tag{3.7}$$

3.5 Parameter estimation

When training a neural network, an optimization algorithm is used to find the optimum parameters that minimize a loss function ℓ. The loss function takes into account the difference between the predicted output and the actual output. The selected loss function depends on the type of problem to be solved. For regression problems, the most common losses are the mean squared error and the mean absolute error. For a classification problem, the cross entropy loss is commonly used as the loss function.

The backpropagation step in a neural network is used after the loss function is computed to update the parameters. The gradient of the loss function with respect to the parameters of the network is computed and weights are updated as follows:

$$w = w - \alpha \cdot \nabla_w\, \ell(w) \tag{3.8}$$

where w denote the weights of the model, $\nabla_w\, \ell(w)$ the gradient of the loss function with respect to the weights and α is the learning rate which controls the size of the update step. The gradient descent algorithm is used to minimize the loss function. The algorithm works by iteratively computing $\nabla_w\ \ell(w)$ for the entire training set and updating the weights in the direction of the negative gradient by using equation 3.8. Each step is called epoch and the number of epochs has to be carefully chosen to avoid overfitting the training data.

The stochastic gradient descent algorithm (SGD) [53] is a variant of the gradient descent algorithm. The algorithm works by computing the gradient of the loss function with respect to an individual observation from the training set. The mini batch gradient descent is another algorithm that computes the gradient for a subset of samples called mini-batch.

3.5.1. Update rules

The update of the parameters of the network presented in equation 3.8 was proposed in 1951. This update rule, however, does not guarantee good convergence and can slow down the training of a network. Other update rules have been proposed which are widely used. In this section, we will discuss some alternative update rules that have been proposed in the literature.

SGD with momentum

The momentum method [54] adds a new term v called *momentum* or velocity that accumulates an exponentially decaying average of past gradients. The algorithm is inspired by Newton's law of motion. In a physical system, the negative gradient would be a force moving a particle in the parameter space [4]. The update rule is given by:

$$v_t = \beta(t) \cdot v_{t-1} - \alpha \cdot \nabla_w\, \ell(w) \tag{3.9}$$

$$w_{t+1} = w_t + v_t \tag{3.10}$$

where $\beta(t) \in [0, 1)$ is a hyperparameter that controls how much prior gradients contribute to v_t. It can be smaller for the first few iterations and larger as the number of iterations increases. For instance, we may have:

$$\beta(t) = \begin{cases} 0.5 & t \leq 250 \\ 0.8 & t > 250 \end{cases} \tag{3.11}$$

AdaGrad

The AdaGrad algorithm [55] adapts the learning rate to the weights. Weights with greater partial derivatives receive larger updates in AdaGrad, whereas weights with lower partial derivatives have smaller updates. AdaGrad rule for weight w_i is given by the following expression:

$$w_{t+1,\,i} = w_{t,\,i} - \frac{\alpha}{\sqrt{\mathbf{G}_{t,ii} + \epsilon}} \odot \nabla_w \, \ell(w_{t,i}) \tag{3.12}$$

where $\mathbf{G}_{t,ii}$ is a diagonal matrix containing the sum of the squares of the gradients with respect to w_i and ϵ is a small constant that is added to prevent division by zero.

One of the main advantages of AdaGrad is that the learning rate is no longer a hyperparameter of the model as it is adapted during training. However, one of the key issues with AdaGrad is that gradients tend to be large at first iterations, and the total sum of gradients grows while training. This may lead the model to stop learning or slow down the training process prematurely.

RMSProp

RMSProp [56] is a variation of AdaGrad that computes an exponentially weighted moving average of squared gradients instead of acummulating them. The update rule is the following:

$$\mathbf{S}_t = \lambda \mathbf{S}_{t-1} + (1 - \lambda)(\nabla_w \, \ell(W_t) \odot \nabla_w \, \ell(W_t)) \tag{3.13}$$

$$w_{t+1,\,i} = w_{t,\,i} - \frac{\alpha}{\sqrt{\mathbf{S}_{t,i} + \epsilon}} \odot \nabla_w \, \ell(w_{t,i}) \tag{3.14}$$

where λ is a term that controls the contribution of past gradients to create S_t which solves the main problem of AdaGrad.

Adam

Adaptive Moment Estimation (Adam) [57] is a hybrid of RMSProp and momentum algorithms. Adam computes an exponentially weighted moving average of squared gradients (equation 3.15) like RMSProp. Similarly to momentum, it also computes an exponentially decaying average of gradients (equation 3.16).

$$v_0 = 0, \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla_w \, \ell(w) \odot \nabla_w \, \ell(w)) \tag{3.15}$$

$$m_0 = 0, \quad m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_w \, \ell(w) \tag{3.16}$$

m_t and v_t are initialized at zero. As a result, at initial iterations their values are close to zero. To overcome this problem, the m_t and v_t terms are corrected as follows:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.17}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.18}$$

Finally, the weights are updated using the following rule:

$$w_{t+1} = w_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \tag{3.19}$$

3.6 Most advanced architectures

3.6.1. Recurrent Neural Networks (RNN)

Recurrent neural networks (RNNs) are similar to MLPs, but they have key differences that allow them to cope with sequential input more effectively. Its formulation arises from a problem of simple neural networks, which is the inability of one input to affect the next output. The problem can be seen when trying to apply neural networks to some NLP tasks in which every word in the text depends on the previous and subsequent words.

RNNs take a sequence of T m-dimensional input vectors $X = x_1, \ldots, x_T$ and output a sequence of r-dimensional vectors $Y = y_1, \ldots, y_T$. At each time step, a hidden state of size p ($h \in \mathbf{R}^p$) is created to model the temporal sequence. In the literature, several RNN designs have been presented. The vanilla RNN is the simplest RNN and is defined as:

$$h_t = F(x_t, h_{t-1}) = \phi(\mathbf{U}x_t + \mathbf{W}h_{t-1} + b), \quad y_t = h_t \tag{3.20}$$

where $\mathbf{U} \in \mathbb{R}^{p \times m}$, $\mathbf{W} \in \mathbb{R}^{p \times p}$ and ϕ is an activation function. As shown in equation 3.20, in the vanilla RNN the output vectors are the hidden state. The benefit of RNNs over MLPs is that they can share parameters over several time steps. In figure 3.3 an illustration of the vanilla RNN is shown.

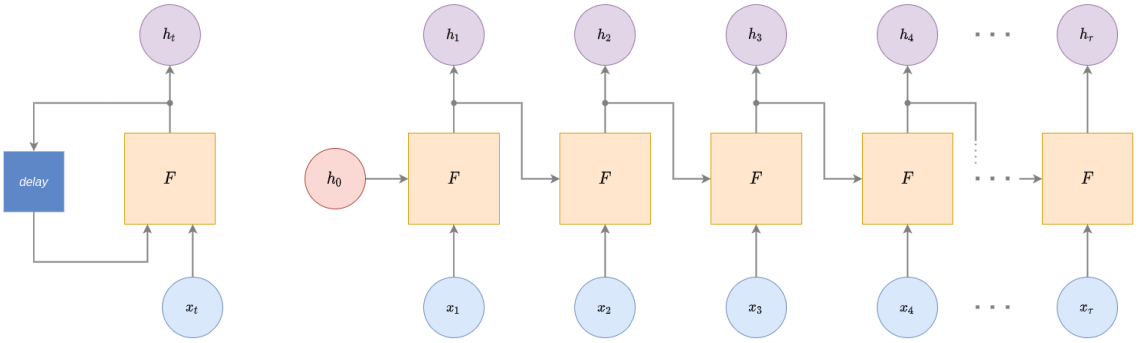


Figure 3.3: Recurrent neural network topology. **Image source:** [3]

The most common approach for training RNNs is back-propagation-through-time (BPTT), which is an extension of the backpropagation algorithm. The BPTT algorithm applies the backpropagation step to the unfolded version of the RNN.

The vanishing gradient problem is one of the main problems with vanilla RNNs which occurs when the gradient's norm is close to zero, thus, when the gradients are propagated

to prior time steps, they drop exponentially to zero. This limits their training to very short sequences. The long short-term memory (LSTM) [58] is one of the most successful variants of RNNs that alleviates the vanishing gradient problem. Aside from LSTMs, the gated recurrent unit (GRU) [59] is a popular alternative to LSTMs. It may be thought of as a simpler form of LSTM units.

Long Short-Term Memory (LSTM)

Long short-term memory (LSTM) are a special type of RNN, capable of learning long-term dependencies. They were proposed by Sepp Hochreiter and Jürgen Schmidhuber in 1997 [58]. Figure 3.4 shows an illustration of the LSTM cell.

The main idea of LSTMs is to not forget information that has been seen before. Each LSTM cell has a forget and an output gate, which determine how much information should be kept from the previous time step and how much information should be given to the next time step. LSTMs have shown to be very powerful in many different applications, such as text generation, video classification and time series prediction.

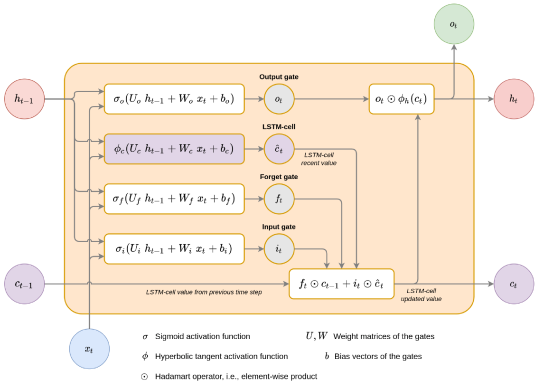


Figure 3.4: LSTM **Image source:** [3]

In more detail, LSTM networks store an extra state, called memory: $c_t \in \mathbb{R}^p$, in addition to the hidden state: $h_t \in \mathbb{R}^p$. The hidden state is computed according to:

$$h_t = o_t \odot \phi_h(c_t) \tag{3.21}$$

where $\odot$ denotes the element-wise product, c_t is the memory state, ϕ_h is the hyperbolic tangent activation function and o_t is an output gate: $o_t \in \mathbb{R}^p$. The output gate o_t controls how much the memory of the LSTM contributes to create h_t at time step t.

The memory state is computed by adding the previous time-memory step's state, multiplied by a forget gate ($f_t \in \mathbb{R}^p$), to the current time-updated step's memory state ($\hat{c}_t \in \mathbb{R}^p$), multiplied by an input gate ($i_t \in \mathbb{R}^p$):

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{3.22}$$

The input, forget, output gates and the time-updated memory state are computed as follows:

$$\hat{c}_t = \phi_c(\mathbf{U}_c h_{t-1} + \mathbf{W}_c x_t + b_c) \tag{3.23}$$
$$o_t = \sigma_o(\mathbf{U}_o h_{t-1} + \mathbf{W}_o x_t + b_o) \tag{3.24}$$
$$f_t = \sigma_f(\mathbf{U}_f h_{t-1} + \mathbf{W}_f x_t + b_f) \tag{3.25}$$
$$i_t = \sigma_i(\mathbf{U}_i h_{t-1} + \mathbf{W}_i x_t + b_i) \tag{3.26}$$

where $\mathbf{U}_c, \mathbf{U}_o, \mathbf{U}_f, \mathbf{U}_i \in \mathbb{R}^{p \times p}$, $\mathbf{W}_c, \mathbf{W}_o, \mathbf{W}_f, \mathbf{W}_i \in \mathbb{R}^{p \times m}$ and b_c, b_o, b_f, b_i are the weight matrices and bias vectors of the gates and the updated memory state. ϕ and σ functions denote the hyperbolic tangent and sigmoid activation functions respectively.

Albeit LSTMs partially solve the vanishing gradient problem, they can still suffer from the exploding gradient problem. The exploding gradient problem occurs when the norm

of the propagated gradients is large, in this case the gradient grows exponentially to infinity, causing numerical instability and causing the model to cease learning. There are several methods to mitigate the exploding gradient problem, including gradient clipping, which scales the gradient when its norm is greater than a threshold.

3.6.2. Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) [50] are a type of neural networks designed to process input data with a grid-like topology [4] (e.g. time series data, image data, etc.). They have shown great success in a variety of image recognition tasks, including classification, segmentation or image captioning. CNNs are created by stacking convolutional layers and pooling operations.

The convolution operation is the application of a filtering function f_{conv} known as kernel, filter or convolution filter to an input followed by an activation function g. The convolution filter slides across the input, performs an element-wise multiplication and sums up the results. In figure 3.5 an illustration of a convolution with a 2-dimensional filter is shown. As can be seen, the filter of size 2 is applied to input subsections of the same size. These subsections are moved by a predetermined amount until the entire input is represented.

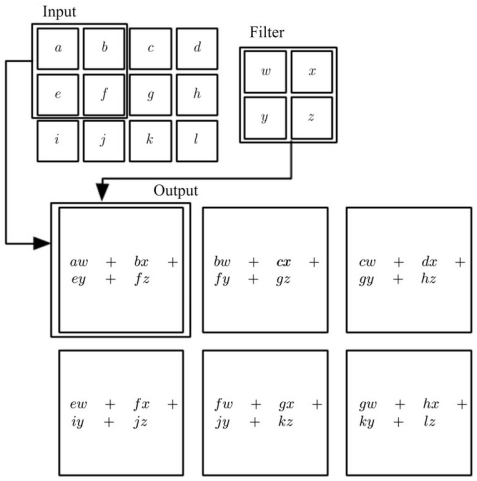


Figure 3.5: An example of 2-D convolution. **Image source:** [4]

Kernel weights are learned during training using the backpropagation algorithm. The same kernel weights are applied to all input's subsections. Thus, for a given input of size n and a filter of size h, the convolution operation will produce a feature map $c = c_1, \ldots, c_{n-h+1}$, where c_i is computed as follows:

$$c_i = g(w^t x_{i:i+h-1} + b) \tag{3.27}$$

where w is the filter, g is an activation function and b is a bias term. Multiple filters are applied to an input in a convolutional layer obtaining several feature maps.

The pooling operation, also known as subsampling, is used to reduce the size of the feature maps. It is widely used in CNNs and aims to reduce the computational cost and the number of parameters that need to be learned. The pooling layer consists in applying a pooling operator (ϕ) to feature maps.

$$\hat{c} = \phi(c_1, \ldots, c_n) \tag{3.28}$$

The most common pooling operators are the max and average pooling. The max pooling operator selects the maximum value, while the average pooling operator averages the values. An illustration of the max and average pooling operators is shown in figure 3.6.

3.6.3. Transformers

An *encoder-decoder* architecture is used in the Transformer model [20] , as it was in many preceding models. However, the Transformer architecture has been shown to outperform

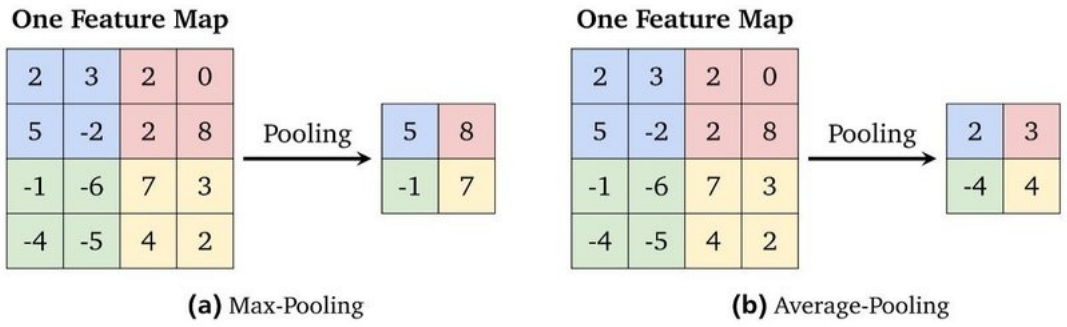


Figure 3.6: Example for the max-pooling and the average-pooling.**Image source:** [5]

many other architectures on a wide range of text generation tasks and has become the standard baseline for many NLP tasks.

Intuition

Figure 3.7 depicts the basic elements of the Transformer architecture. As can be seen, the Transformer takes one sentence as input and produces another. The input in translation could be a spanish phrase, and the output could be a translated sentence in another language. However, as we will see later, the input and output may not be sentences. The encoder and the decoder are the two major components of the transformer. Each component is made up of a stack of N encoder and decoder layers.

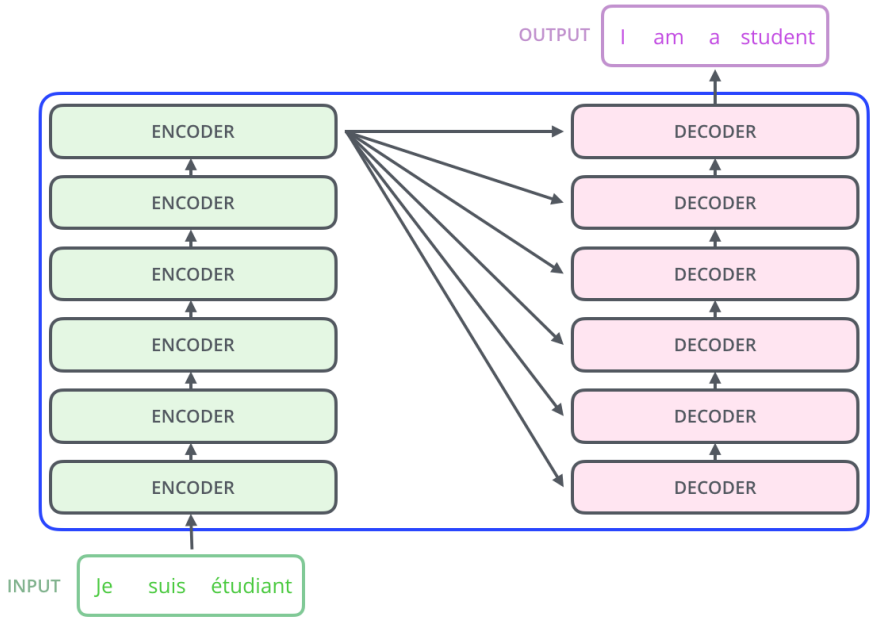


Figure 3.7: Transformer's Architecture. **Image source:** [6]

Two further layers, a self attention module and a feedforward neural network, make up the encoder layers (figure 3.8). When the model is encoding a certain word, the first layer is utilized to assist it identify which are the most significant terms in the phrase to look at. The outputs from the self attention module are fed into a feedforward neural network that is applied to each word separately. The decoder uses the same layers as the encoder, but it adds a new one to assist it focus on relevant sections of the input phrase.

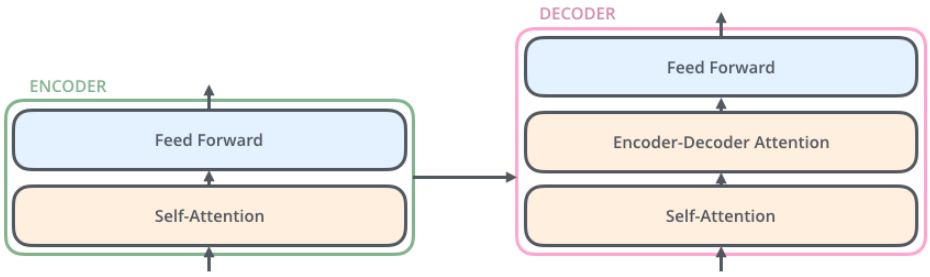


Figure 3.8: Transformer's encoder and decoder layers. **Image source:** [6]

Attention

An attention function $F_{att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ takes key ($\mathbf{K}$), queries ($\mathbf{Q}$) and values ($\mathbf{V}$) matrices as inputs and outputs weighted average vectors $\mathbf{V}'$. Firstly, it computes the similarity between keys and queries. A softmax function is then applied on each s_{ij} so that the resulting vector adds up to one.

$$s_{ij} = f_{sim}(q_i, k_j) \tag{3.29}$$

(q_i: i^{th} query vector; k_j: j^{th} key vector)

$$\alpha_{ij} = \frac{e^{s_{ij}}}{\sum_j e^{s_{ij}}} \tag{3.30}$$

where q_i is the i^{th} query and k_j is the j^{th} key. There are different approaches to compute the similarity function. The most common one is the scaled dot product. Other approaches might be:

$$f_{sim} = \begin{cases} q_i^t \, k_i & \text{Dot product} \\ \frac{q_i^t \, k_i}{\sqrt{d}} & \text{Scaled dot product} \\ \mathbf{w}_v^\top \tanh(\mathbf{W}_q \, q_i + \mathbf{W}_k \, k_i) & \text{Additive Attention} \end{cases} \tag{3.31}$$

Finally, an output vector is computed for each query q_i as a weighted average by multiplying each value vector by α_{ij}. This vector is commonly known as the context vector as it is telling how similar this query is with regard to all other queries.

$$v_i' = \sum_j \alpha_{ij} v_j \tag{3.32}$$

In matrix form this can be seen as:

$$F_{att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q} \, \mathbf{K}^t}{\sqrt{d}})\mathbf{V} \tag{3.33}$$

where d is a hyperparameter and the scaled dot product has been considered as the similarity function.

Word embeddings

Machine learning models do not "see" data in the same way that humans do. We can readily understand the term 'dog', but ML models cannot; they require feature vectors. Word embeddings, are representations of words that can be fed into ML models. A word embedding is a numeric vector that represents a word that may be taught using a number of different language models.

Positional encodings

Because the self-attention operation is permutation invariant, meaning it does not consider the order, it is critical to employ suitable positional encoding to give the model order information. The positional encoding is a vector of the same size as the word embeddings, then it can be directly added to the word embedding. These vectors follow a pattern that the model uses to calculate the position of each word in the sequence, as well as the distance between them.

Encoder layer

Given a sequence of tokens $t_1, \ldots, t_L$, learned word embeddings are used in order to convert input tokens to vectors of dimension d_m. Positional information of each token is added using positional encodings (figure 3.9).

$$\mathbf{A} = [E_t(t_1) + e_1, \ \ldots, \ E_t(t_L) + e_N], \ \mathbf{A} \in \mathbb{R}^{L \times d_m} \tag{3.34}$$

where $E_t(x)$ is a function that returns the corresponding word embedding for the token x, e_i is the corresponding positional encoding for the i^{th} token and $\mathbf{A}$ is a matrix of dimensions $\mathbf{A} \in \mathbb{R}^{L \times d_m}$ where L is the number of tokens and d_m the dimensionality of each token.

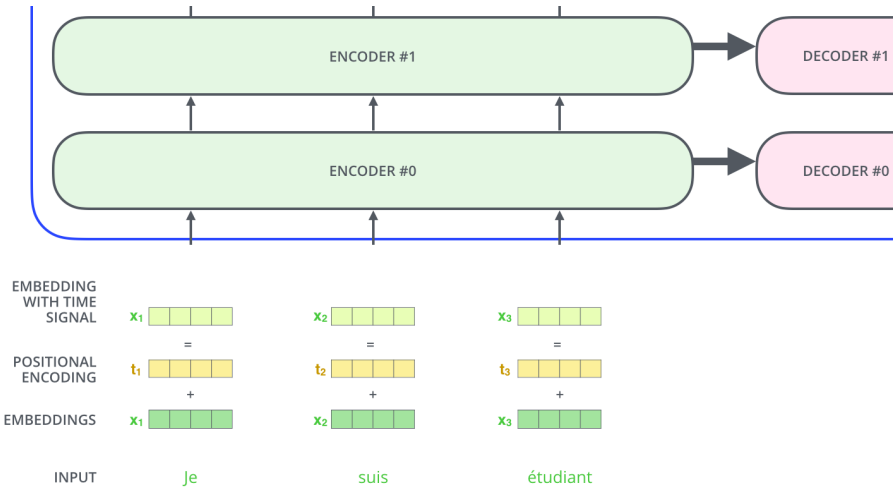


Figure 3.9: Input of the first encoder layer of the Transformer architecture. **Image source:** [6]

$\mathbf{A}$ is only used in the encoder's first layer. The outputs from the layer directly below will be used as input for the subsequent layers. Matrix $\mathbf{A}$ is passed to the encoder as it is illustrated in figure 3.10.

In the self-attention module, firstly, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ matrices are computed by multiplying $\mathbf{A}$ with $\mathbf{W}^Q$, $\mathbf{W}^K$ and $\mathbf{W}^V$ trainable matrices. However, multiple attention heads are used to learn distinct representational subspaces at different positions, boosting the model's ability to focus on diverse positions. The multi-head self-attention layer consists in s heads where each head learns different representations at different positions. In each head, $\mathbf{Q}_i$, $\mathbf{K}_i$ and $\mathbf{V}_i$ matrices are computed as follows:

$$\mathbf{Q}_i = \mathbf{A}\,\mathbf{W}_i^Q, \quad \mathbf{W}_i^Q \in \mathbb{R}^{d_m \times d_k} \tag{3.35}$$

$$\mathbf{K}_i = \mathbf{A}\,\mathbf{W}_i^K, \quad \mathbf{W}_i^K \in \mathbb{R}^{d_m \times d_k} \tag{3.36}$$

$$\mathbf{V}_i = \mathbf{A}\,\mathbf{W}_i^V, \quad \mathbf{W}_i^V \in \mathbb{R}^{d_m \times d_v} \tag{3.37}$$

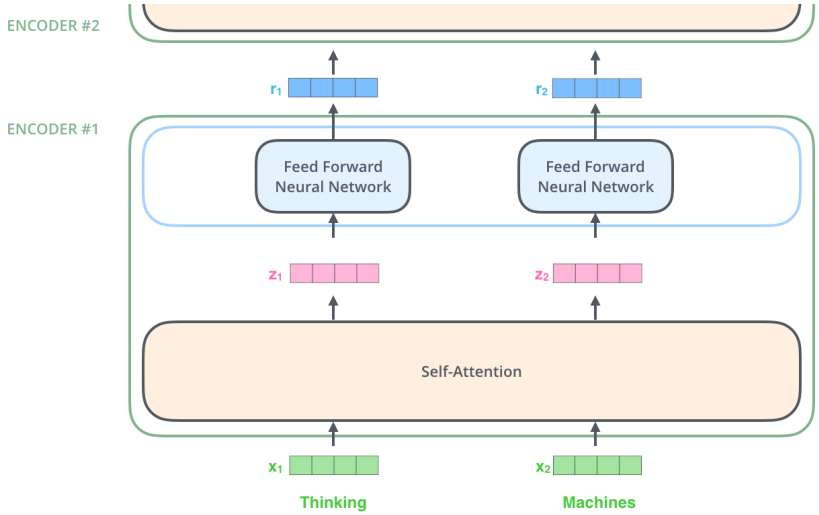


Figure 3.10: Sublayers of the encoder layer of the Transformer. **Image source:** [6]

Where d_k and d_v are the dimension for the key/query vectors and value vectors respectively. And $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$ are trainable matrices. In the original paper 8 heads are used ($s = 8$) and $d_v = d_k = d_m/s$.

After $\mathbf{Q}_i, \mathbf{K}_i$ and $\mathbf{V}_i$ are computed, an attention function is used to obtain a weighted average vector. The scaled dot product is used as the similarity function.

$$\text{head}_i = F_{att}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \tag{3.38}$$

Resulting vectors for each head are concatenated and then projected again using a matrix $\mathbf{W}^O \in \mathbb{R}^{s\,d_k \times d_m}$ so that resulting final vectors are d_m-dimensional.

$$\mathbf{Z} = \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \texttt{concat}(\text{head}_1, \ldots, \text{head}_s)\mathbf{W}^O \tag{3.39}$$

The resulting outputs from the self attention module is a matrix $\mathbf{Z} \in \mathbb{R}^{L \times d_m}$ where each row is d_m-dimensional and represents the i^{th} word. The self attention module is summarized in figure 3.11.

Then, each row of $\mathbf{Z}$ is passed separately to a fully connected feedforward layer with a ReLU activation function which is defined as follows:

$$\text{FFN}(z_i) = \max(0,\ z_i\,\mathbf{M_1} + b_1)\mathbf{M_2} + b_2 \tag{3.40}$$

In the original paper, the number of nodes in the inner layer of the fully connected feedforward layer is $d_{ff} = 4 * d_m$. Output vectors from the fully connected feedforward layer are embedded into a sole matrix and passed to the next encoder layer.

To optimize the flow of information across the network, residual connections are employed after each sub-layer, followed by a layer-normalization step (figure 3.12).

Decoder layer

The decoder layer, which is similar to the encoder but differs in several ways, is used to generate the model's output. As it is shown in figure 3.12, the decoder is composed of the same modules as the encoder but it adds a new attention module which is provided only with the query $\mathbf{Q}$ from the layer below and receives its keys $\mathbf{K}$ and values $\mathbf{V}$ from

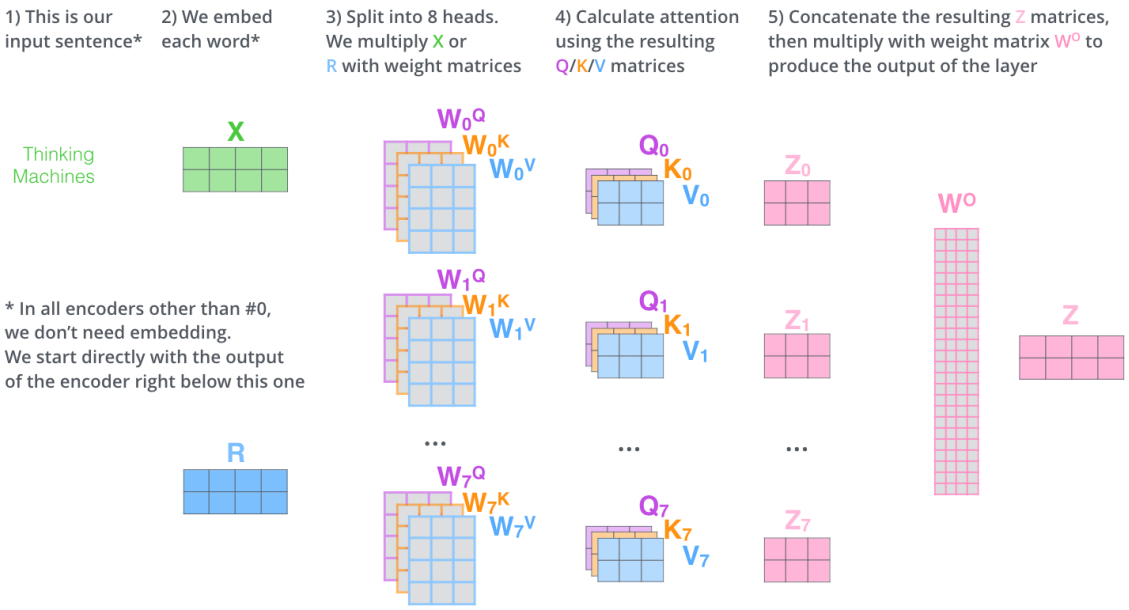


Figure 3.11: Summary of the encoder layer. **Image source:** [6]

the final layer of the encoder. The output from the final layer of the encoder is mapped to two matrices **K** and **V** which are the ones used in this new attention module.

The decoder outputs one token at a time when generating text. The prior prediction and the output of the encoder's final layer are sent to the decoder at each step. The self-attention modules, on the other hand, are not identical to those utilized in the encoder. Only earlier tokens in the output sequence are allowed to be attended to by the decoder self attention modules. This is done by masking out the future tokens in the output sequence when generating the keys and values, which are parts of the matrix product. Tokens are placed as inputs to the decoder as they are predicted, and the position occupied is unmasked.

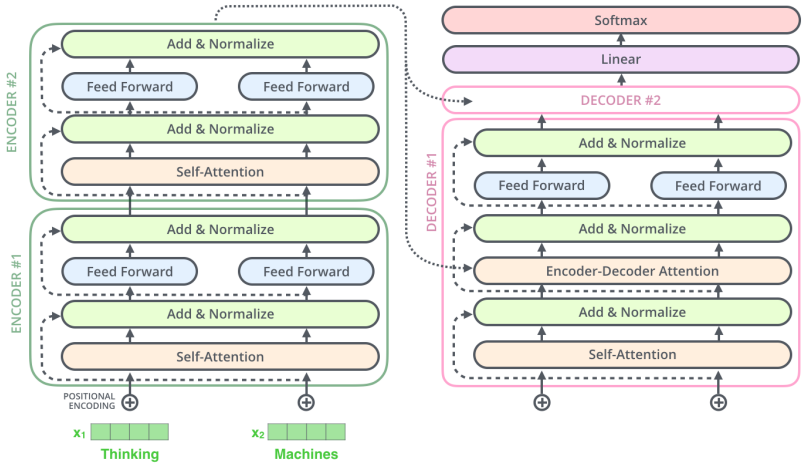


Figure 3.12: Transformer's encoder and decoder sublayers. **Image source:** [6]

Linear and Softmax layers

The decoder outputs a matrix **Z**. The final layers of the decoder (figure 3.12) are used to transform the matrix **Z** into a logit vector p whose elements are the probability of the

token corresponding to each index position. This is done by using a fully connected neural network which is followed by a softmax function. The token with the highest probability is chosen and used in the next step of the decoder.

3.6.4. Contrastive Language–Image Pre-training (CLIP)

Contrastive Language–Image Pre-training (CLIP) [7] is a deep learning model that mixes picture and text representations. It was released by OpenAI in early 2021 and consists in two parts: a text encoder and an image encoder which are trained such that they produce similar outputs when the image and text inputs are similar. CLIP is based on the work of Zhang et al. [60] who proposed a similar method for medical images with text annotations. For instance, if we feed the image encoder with the image of a pangolin, the image encoder will return a vector that represents the image. If we now give the text encoder sentences like "an image of a pangolin", "a sweet pangolin walking in the jungle" and "the most attractive pangolin" the text encoder will produce a vector that is similar to the vector produced by the image encoder.

Motivation

Most current vision models are trained on standard datasets such as ImageNet, Open Image Dataset, and others, and while these datasets attempt to contain a wide selection of pictures, doing so is quite challenging. These datasets are useful for testing the performance of vision models, but they do not generally contain the variety needed for training. CLIP, on the other hand, was trained on a dataset made up of 400 million (image, text) pairings scraped from the internet in order to perform a contrastive learning task.

Contrastive learning

Contrastive learning is based on the idea of learning an embedding space where similar samples are close together and dissimilar samples are far away. The contrastive loss introduced by Chopra et al. in 2005 [61], is one of the early training objectives used in contrastive deep metric learning. Given a set of samples with its corresponding class $\{(x_i, y_i), x_i \in \mathcal{X}, y_i \in \{1, \ldots, L\}\}_{i=1}^{N}$, we want to learn a function $f : \mathcal{X} \to \mathbb{R}^d$ that encodes x_i into an embedding vector l_i such that samples from the same class are close together in the embedding space, while samples from other classes are far apart. Therefore, when two samples (x_i, x_j) belong to the same class, contrastive loss decreases the embedding distance, but maximizes it otherwise [62].

Training

CLIP jointly trains a text encoder and an image encoder in a contrastive fashion. Both the text encoder and the image encoder produce similar embeddings if the caption is related to the image.

In figure 3.13 it is shown the contrastive pre-training approach. Given a set of N (image, text) pairs, CLIP computes the cosine similarity matrix between all $N \times N$ possible (image, text) pairings within this batch. The text and image encoders are jointly trained to maximize the similarity between N correct pairs of (image, text) associations while minimizing the similarity for $N(N - 1)$ incorrect pairs.

In more detail, CLIP transforms each x_v and x_u input picture and text into d-dimensional vector representations $\mathbf{v}$ and $\mathbf{u}$, respectively. The encoder function f_v turns each picture

x_v into a vector of dimensions h_v, which is transformed into vector $\mathbf{v}$ via a non-linear projection function g_v. Similarly, we get a text representation $\mathbf{u}$ with a text encoder f_u and a projection function g_u for each text input x_u.

$$\mathbf{v} = g_v(f_v(x_v)) \quad \mathbf{u} = g_u(f_u(x_u)) \tag{3.41}$$

where $\mathbf{v}, \mathbf{u} \in \mathbb{R}^d$. For contrastive learning, the projection functions g_v and g_u project representations from the encoder space to the same d-dimensional space.

A minibatch of N input pairs (x_v, x_u) is sampled from training data at training time, and their representation pairs $(\mathbf{v}, \mathbf{u})$ are computed. The i-th pair is denoted by $(\mathbf{v_i}, \mathbf{u_i})$. For the i-th pair, the training goal contains two loss functions: an image-to-text contrastive loss ($\ell_i^{(v \to u)}$) and a text-to-image contrastive loss ($\ell_i^{(u \to v)}$):

$$\ell_i^{(v \to u)} = -\log \frac{\exp(\langle \mathbf{v_i}, \mathbf{u_i} \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle \mathbf{v_i}, \mathbf{u_k} \rangle / \tau)} \quad \ell_i^{(u \to v)} = -\log \frac{\exp(\langle \mathbf{u_i}, \mathbf{v_i} \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle \mathbf{u_i}, \mathbf{v_k} \rangle / \tau)} \tag{3.42}$$

where $\langle \mathbf{v_i}, \mathbf{u_i} \rangle$ denotes the cosine similarity and $\tau \in \mathbb{R}^+$ indicates a loss hyperparameter.

After that, the final loss is calculated as a weighted average of the two losses across all positive image-text pairs in each minibatch:

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(\lambda \ell_i^{(v \to u)} + (1 - \lambda) \ell_i^{(u \to v)} \right), \tag{3.43}$$

where $\lambda \in [0, 1]$ is a scalar weight.

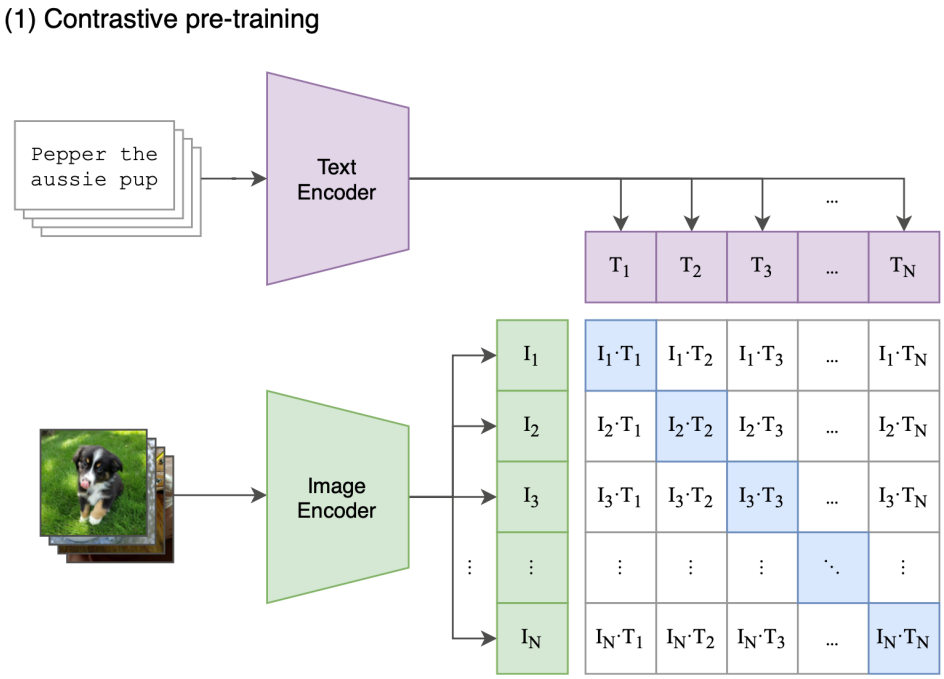


Figure 3.13: Summary of CLIP contrastive pre-training over text-image pairs. **Image source:** [7]

Encoders

Two different models can be used as image encoders: the ResNet-50 model [45] and the recently introduced Vision Transformer [63]. For the text encoder they used a GPT-style

Transformer with 63 million parameters and 12 layers with 512 nodes with 8 attention heads.

3.7 What is a language model?

The objective of language modeling is to estimate the probability distributions of words in a vocabulary given a sequence of context words. Language models use context, this is, a sequence of previous words, to estimate the probability distribution of the next word or token (equation 3.44).

$$\mathrm{p}(w_i \mid w_1, \ldots, w_{i-1}) \tag{3.44}$$

Language models can also compute the probability of a sequence of words of length L by factorizing the probability using the chain rule (equation 3.45).

$$p(w_1, \ldots, w_L) = \prod_{i=1}^{L} \mathrm{p}(w_i \mid w_1, \ldots, w_{i-1}) \tag{3.45}$$

Language models are frequently trained with next word prediction tasks to minimize the negative log likelihood (equation 3.46) of large amounts of written text.

$$NLL = -\frac{1}{L} \sum_{i=1}^{L} \log \mathrm{p}(w_i \mid w_1, \ldots, w_{i-1}) \tag{3.46}$$

3.7.1. N-gram based Language Models

N-gram language models compute $\mathrm{p}(w_i \mid w_{i-1}, \ldots, w_1)$ by counting the number of times a token appears after a certain sequence of tokens (context). Suppose that we want to compute the following probability:

$$\mathrm{p}(\text{'banana'} | \text{'the', 'man', 'is', 'eating', 'a'}) \tag{3.47}$$

A 5-gram language model would use a very large text corpus and count the number of times that the sequence "the man is eating a" appears, as well as how many times it appears followed by "banana". We can compute these counts and estimate the probability as follows:

$$\mathrm{p}(\text{'banana'} | \text{'the', 'man', 'is', 'eating', 'a'}) = \frac{c(\text{'the', 'man', 'is', 'eating', 'a', 'banana'})}{c(\text{'the', 'man', 'is', 'eating', 'a'})} \tag{3.48}$$

While this method can work fine sometimes, it has one major problem. If the given context is not seen during training, the language model will give that sequence a probability of 0. This problem is referred to as the zero-frequency problem. One way to solve it is to use smoothing, which adds a small probability to every observed sequence. Another issue with these methods is that they are unable of determining the semantic similarity of sequences that have the same meaning but are written using different terms.

3.7.2. Generative Pretrained Transformer 2 (GPT-2)

The Transformer architecture seen in section 3.6.3 was adapted for language modeling tasks in the Generative Pretrained Transformer 2 model (GPT-2) [35]. GPT-2 is a language model that was published in 2019 by OpenAI and it was trained on a 40GB corpus named WebText.

The vanilla Transformer was composed of two main parts: the encoder and the decoder. In the GPT-2 models, the researchers decided to work just with the decoder block and proposed a language model formed from stacked decoder blocks. The decoder blocks that they used resemble the ones proposed in the vanilla Transformer, with the exception that the encoder is not used, hence the second self attention sub-layer is removed. The GPT-2 model generates sentences in the same way as it was explained in section 3.6.3 for the vanilla Transformer. In addition, a set of context tokens can be added to generate the text at the initial step, this will tame the model to generate the text starting from the given context. The size of the context tokens and the output differs depending on the GPT-2 model architecture (figure 3.14). In the large model, which is the one used in this work, the size is 1280.

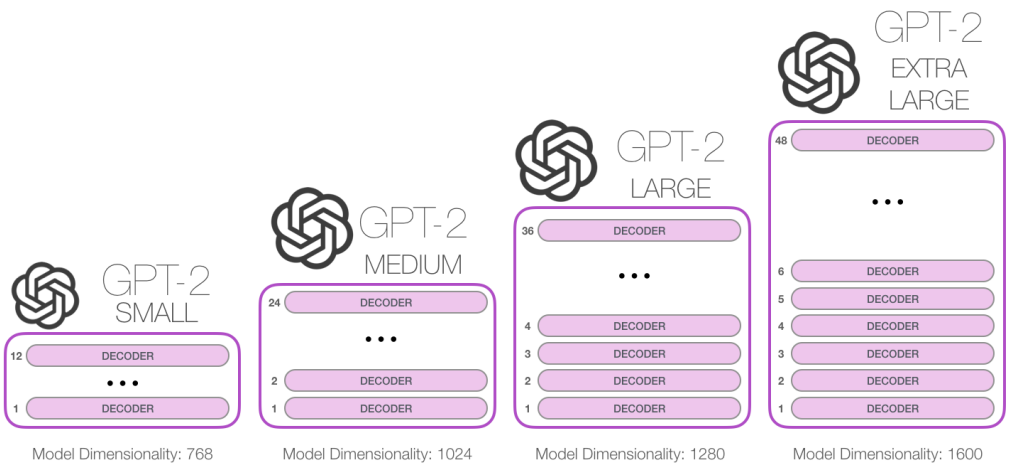


Figure 3.14: GPT-2 architectures. **Image source:** [8]

3.7.3. Decoding methods

Due to the advent of huge transformer-based language models trained on millions of webpages, including as OpenAI's GPT-2 model, there has been a renewed interest in the task of automatically generating text from a given model. GPT-2, XLNet, and BERT all generate text by sampling from a distribution over sequences of words sampled from the model. This approach is simple, efficient, and often produces good-quality text. Improved decoding techniques, as well as improved Transformer design and large amounts of unsupervised training data, have had a substantial influence. The GPT-2 model can decode using a variety of approaches, including greedy search, beam search, top-k sampling, top-p sampling, etc. In this section, three of the most common decoding strategies will be discussed: beam search, greedy search and top-k sampling.

Language models generate text in an autoregressive fashion by computing the probability of a sequence of words using the chain rule (equation 3.45). The language model normally determines the length L of the sequence of words, which corresponds to the timestep T. There are two special tokens: `[SOS]` and `[EOS]` which represent the start and

end of a sentence respectively. When the [EOS] token is predicted and selected by the decoding strategy, the language model stops generating any other tokens.

Greedy search

The greedy search is the most simple strategy. At each timestep t, greedy search just chooses the word with the highest probability as its next term.

$$w_t = \arg\max_w \mathrm{p}(w \mid w_1, \ldots, w_{t-1}) \tag{3.49}$$

This strategy has one main flaw, it can assign the highest probability to particular sets of words repeatedly. As a result, the model generates text with repeated words or sentences. Another problem of greedy search is that it can miss sentences with greater overall probability. For instance, in figure 3.15, the phrase "the dog has" has a probability of $0.4 * 0.9 = 0.36$ which is higher than the generated phrase's chance of $0.5 * 0.4 = 0.2$.

Beam search

Beam search solves the problem of missing phrases with a greater overall probability by keeping the most likely N beams of hypotheses at each time step and finally selecting the hypothesis with the greatest probability computed using equation 3.45. An example of the beam search strategy is shown in figure 3.16.

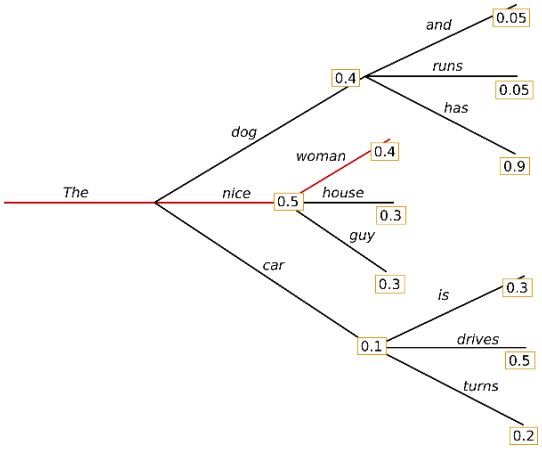


Figure 3.15: An example of a greedy search. **Image source:** [9]

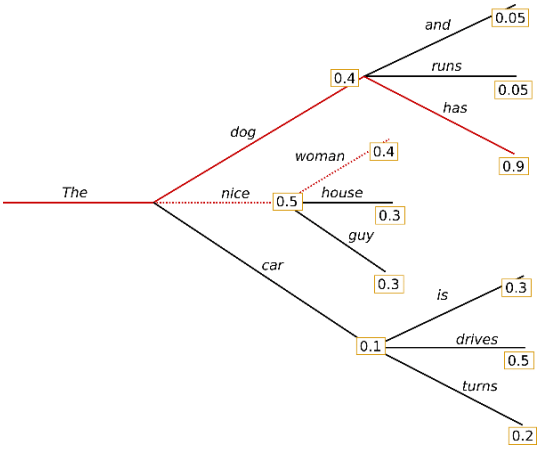


Figure 3.16: An example of a beam search. **Image source:** [9]

Top-K sampling

Top-k sampling has lately gained popularity as an alternative sampling method; GPT-2, for example, used this strategy. In top-k sampling, the top k next tokens ($V^{(k)}$) are sampled according to their respective conditional probability at each time step. After that, the distribution is rescaled as shown in equation 3.50 and the rescaled distribution is used for sampling.

$$\mathrm{p}'(w \mid w_1, \ldots, w_{t-1}) = \begin{cases} \frac{\mathrm{p}(w \mid w_1, \ldots, w_{t-1})}{\sum_{w \in V^{(k)}} \mathrm{p}(w \mid w_1, \ldots, w_{t-1})} & \text{if } w \in V^{(k)} \\ 0 & \text{otherwise.} \end{cases} \tag{3.50}$$

CHAPTER 4
Image Captioning

In this chapter we review and describe existing image captioning methods, which include template-based approaches, retrieval-based approaches, and deep learning-based image captioning models. Deep learning-based models are split up into single-stage attention models, two-stages attention models, transformer-based models, and based on pre-trained models.

4.1 Template-Based Methods

Methods based on templates to produce captions, utilize preset templates with a number of empty slots. In this approach, the important objects in the scene are detected and then the empty slots are filled in the preset template. For example, in [64] the authors use a Conditional Random Field (CRF) to infer the objects, attributes, and prepositions of the image, and then the gaps of the template are filled. Albeit template-based methods can produce captions that are grammatically correct, they are predetermined and unable to produce variable-length captions.

4.2 Retrieval-Based Methods

In retrieval-based methods, the caption of an image is generated by picking the best appropriate caption from a group of existing captions. The most similar pictures with its respective captions from the training set are found using retrieval-based techniques, and the selected captions are referred to as candidate captions. These methods use the similarity between the query image and the training images to generate captions and the caption with the highest score is chosen as the generated caption. These approaches generate captions that are grammatically correct. They cannot, however, create image-specific captions.

4.3 Single-Stage Attention Based Methods

In single-stage attention-based methods, an encoder is used to embed the image and then a decoder is used to generate the caption. When producing a word, the attention method described in section 3.6.3 is employed in the decoder to pay attention to the most informative sections of the picture. Xu et al. [10] used the Oxford VGGnet [44] pretrained on ImageNet to encode the image, then decoded the image into a squence of words using a LSTM [58] based language model with an attention mechanism. An

illustration of this framework is shown in figure 4.1. They were able to focus on the most significant regions of the image by applying an attention mechanism in the decoder. However, this technique is unable to determine the position of each picture section, and the position of the regions may be crucial when describing an image.

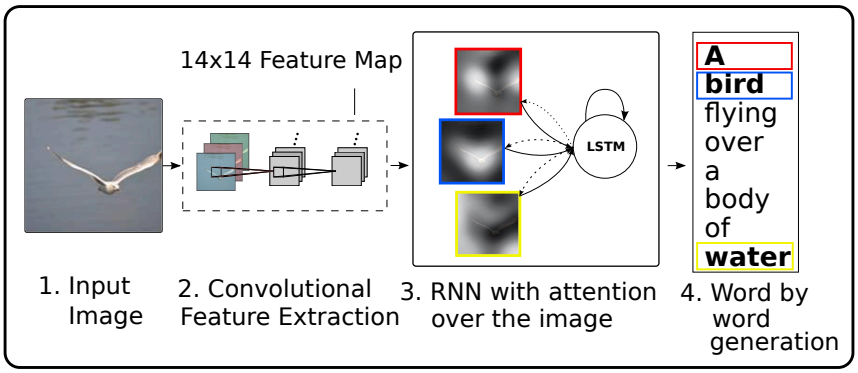


Figure 4.1: An example of a Single-Stage Attention Based framework. **Image source:** [10]

4.4 Two-Stages Attention Based Methods

In two-stages attention based methods, a *bottom-up* attention is done by using an object detection to detect several informative regions in the image. Then, a *top-down* attention attends to the most relevant detected regions when producing a word. Anderson et al. [25] used Faster R-CNN [65] to select the most informative regions of the image, then they used a LSTM [58] language model as a decoder with an attention mechanism.

The two-stage attention based image captioning models has shown to outperform single-stage attention based models. Each identified region, on the other hand, is isolated from the others and the method is not able to model the connection between different regions.

4.5 Transformer Based Methods

The transformer based methods use the architecture of the Transformer model explained in section 3.6.3. A CNN encoder and a Transformer model are the two basic components of these techniques. The Transformer model takes the image features retrieved from the CNN as input. The encoder component of the Transformer is utilized for self-attention on visual features, while the decoder part is used for visio-linguistic attention and for generating the caption. Huang et al. [28] proposed AoANet which uses the vanilla Transformer without adding the feedforward layer. In AoANet, similarly to [25], the image features can be extracted using a CNN or using Faster R-CNN.

Compared to other image captioning methods, the Transformer architecture is able to model the relationship of different regions of the image.

4.6 Based on Pre-trained Models

Novel image captioning methods based on the Contrastive Language-Image Pre-Training (CLIP) and the Generative Pretrained Transformer 2 (GPT-2) models have recently been proposed. Two of them are covered in this section.

4.6.1. ClipCap

The ClipCap model, recently introduced by Ron Mokady et al. [32], is a *lightweight* captioning approach that combines CLIP and GPT-2.

Problem statement

The goal is to learn how to provide a relevant caption for an unknown input image. The image encoder of CLIP is used to generate the embeddings of the input images. Given a dataset of images, captions and the CLIP visual embedding of the images $\{m^i, c^i, x^i\}_{i=1}^N$, we can use the log-likelihood as the training objective:

$$\max_{\theta} \sum_{i=1}^{N} \log \mathrm{p}_\theta(c^i | \, x^i) \tag{4.1}$$

where θ represents the trainable parameters of the model and x^i is the CLIP visual embedding of the input image which is used as a condition. The caption can be seen as a sequence of words: $c^i = c_1^i, \ldots, c_\ell^i$. Therefore, the objective can be rewritten as follows:

$$\max_{\theta} \sum_{i=1}^{N} \log \mathrm{p}_\theta(c_1^i, \ldots, c_\ell^i | \, x^i) \tag{4.2}$$

The condition is used as a prefix to the caption. We can decompose the expression $\mathrm{p}_\theta(c_1^i, \ldots, c_\ell^i | \, x^i)$ into the product of conditional probabilities by using the chain rule (equation 4.3). Then, we can use an autoregressive language model that predicts the next token without considering subsequent tokens because the required semantic information is captured in the prefix. As a result, our objective can be summarized as follows:

$$\max_{\theta} \sum_{i=1}^{N} \log(\prod_{j=1}^{\ell} \mathrm{p}_\theta(c_j^i | \, x^i, c_1^i, \ldots, c_{j-1}^i)) \tag{4.3}$$

$$\max_{\theta} \sum_{i=1}^{N} \sum_{j=1}^{\ell} \log \mathrm{p}_\theta(c_j^i | \, x^i, c_1^i, \ldots, c_{j-1}^i) \tag{4.4}$$

Design

In figure 4.2 it is shown how the model works. GPT-2 is the language model. Captions are converted into a sequence of word embeddings by using the GPT-2 tokenizer. Firstly, the image (m^i) is encoded by using the CLIP visual encoder:

$$x^i = \texttt{CLIP}_{IE}(m^i), \quad x^i \in \mathbb{R}^{d_c} \tag{4.5}$$

where d_c represents the dimension of the embedding and is equal to 512. Then, this embedding is projected into a higher dimensional space by using $\mathbf{W}$ which is a matrix, that will be learned during training, of dimensions $d_c \times (l * 768)$ where l is a hyperparameter of the model that represents the length of the sequence of tokens. The projected embedding is reshaped such that the final shape is $l \times 768$. Projected embeddings can be seen as a sequence of word embeddings of dimensions 768 without textual meaning:

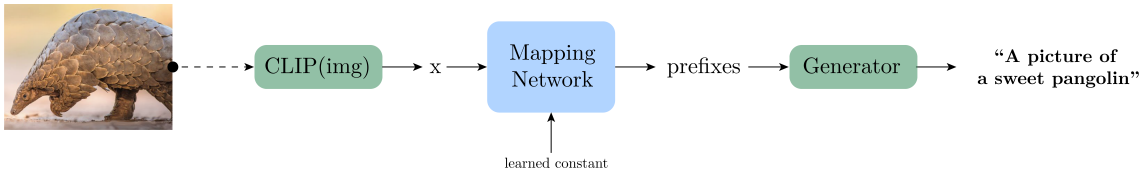


Figure 4.2: Architectural design of the ClipCap framework

$$x^i\mathbf{W} = e^i \in \mathbb{R}^{l*768} \longrightarrow \texttt{reshape}(e^i) \longrightarrow \mathbf{E}^i = [v_1^i, \cdots, v_l^i], \;\; \mathbf{E}^i \in \mathbb{R}^{768\times l} \tag{4.6}$$

Next, $\mathbf{E}^i$ is concatenated with learned vectors $\mathbf{B} \in \mathbb{R}^{768\times l}$ which are learned during training:

$$\mathbf{E'}^i = \texttt{concat}(\mathbf{E}^i, \mathbf{B}), \;\; \mathbf{E'}^i \in \mathbb{R}^{768\times 2*l} \tag{4.7}$$

$\mathbf{E'}^i$ is given to a mapping network F, which maps it to $2 * l$ embedding vectors of dimensions 768. From those $2 * l$ vectors, the first l vectors are used as prefixes for GPT-2. The mapping network might be the Transformer model or a MLP. The prefixes it generates in both cases have no textual meaning; their entire purpose is to tame the language model.

$$p_1^i, \cdots, p_{2*l}^i = F(\mathbf{E'}^i) \tag{4.8}$$

Prefixes are concatenated with the caption word embeddings and during training we feed the language model with the concatenation Z^i.

$$Z^i = p_1^i, \ldots, p_l^i, c_1^i, \ldots, c_\ell^i \tag{4.9}$$

The training objective is predicting the caption tokens conditioned on the prefix in an autoregressive fashion. Thus, the Transformer (F) is trained using the cross-entropy loss:

$$\mathcal{L}_X = -\sum_{i=1}^{N}\sum_{j=1}^{\ell} \log \mathrm{p}_\theta(c_j^i |\, p_1^i, \ldots, p_l^i, c_1^i, \ldots, c_{j-1}^i) \tag{4.10}$$

The mapping network can be trained in two different ways. To begin, fine-tune the language model, which entails training the language model's first layers. This method increases the number of training parameters for the model, after which a second method is presented, in which the language model is frozen throughout training.

Inference

The CLIP encoder and the mapping network F are used to extract the visual prefixes of an image m. Using these prefixes, the caption is generated by predicting the next tokens one by one guided by GPT-2. The GPT-2 model assigns a probability to each word in the vocabulary. These probabilities are then used to predict the next token by using a beam search strategy or a greedy method.

4.6.2. CLIP-GLaSS

The CLIP-GLaSS framework [11] is a novel image captioning framework that uses the Generative Pretrained Transformer 2 (GPT-2) to produce a caption given an image after

an exploration of the latent space performed by a genetic algorithm (GA) [1]. Moreover, this framework can also be used to generate an image from a caption by using Generative Adversarial Networks (GANs) [37].

Design

In GPT-2, context tokens are utilized to predict the next token. The main idea of this method is to find the context tokens that make the GPT-2 model produce a sequence of words that describe the content of the input image. This is accomplished by utilizing a GA to explore the latent space of z which is a vector of context tokens.

Figure 4.3 depicts the CLIP-GLaSS framework for the image-to-text task. The framework takes an image as input, which is encoded by using the CLIP visual encoder (equation 4.5). GPT-2 is seeded by z, a vector of context tokens, and outputs a text which is passed to the CLIP text encoder. The cosine similarity (s) between the embedding of the CLIP text encoder and the CLIP visual encoder are computed. The similarity s is used as the fitness function in the GA. The optimization problem of the GA can be summarized as follows:

$$\max_{z} \quad cos(\texttt{CLIP}_{IE}(G(z)),\ \texttt{CLIP}_{TE}(T)). \tag{4.11}$$

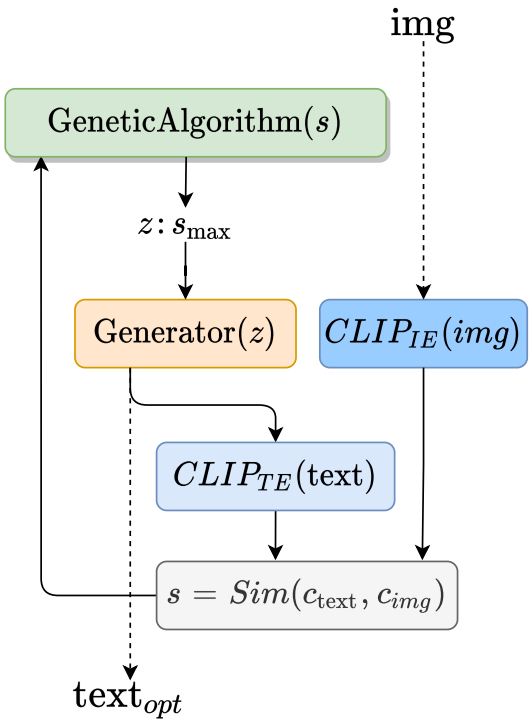


Figure 4.3: CLIP-GLaSS architecture design. **Image source:** [11]

[1] More details about genetic algorithms can be found in appendix A.

CHAPTER 5
Experimental framework

In this chapter we will cover all the experimental work of the project. To begin, we will describe the dataset that will be used to train and validate our models. Second, we will discuss some of the most common metrics in image captioning that will be used to validate our systems. In addition, we will detail the Python libraries for working with neural networks that have been used, as well as the hardware we had at our disposal during the project. Finally, we will detail the implementation of our proposals.

5.1 Dataset

The Microsoft COCO (MSCOCO) 2014 Captions dataset [38] is a large-scale dataset consisting of images and captions. It was originally created by Microsoft for computer vision tasks such as instance segmentation, semantic segmentation and object detection. Microsoft later added five human generated captions to each image to produce the MSCOCO caption dataset. Images were obtained from Flickr searching for pairs of eighty object categories and different scene types. Human annotators were not allowed to give names to the people in the photographs or to describe events that may or may not have occurred in the past or future. In addition, each sentence has at least eight words and only describes the scene's most important aspects. Some examples from the MSCOCO dataset are shown in figure 5.1.

It is composed of 123,000 images which are typically divided into three datasets which are known as the *Karpathy* splits and are widely utilized in other image captioning publications: 118,000 for train, 5,000 for validation and 5,000 for test.

Figure 5.1: Example images and captions from the MSCOCO dataset.

5.1.1. Analysis of the legal and ethical framework

Legality

The MSCOCO dataset is a large-scale dataset with roughly 123,000 photos. It is released under the Creative Commons Attribution 4.0 International License[1], which gives the dataset the maximum freedom and allows anybody to use it for research and study purposes. This work falls into both categories and thus the use of this dataset is permitted.

5.2 Metrics

The goal of image captioning metrics is to compute a score that captures the similarity between an image and a candidate caption. This task can be approached in two ways: reference free metrics and reference based metrics. While reference free metrics do not require a pre-existing reference set of captions, reference based metrics do.

5.2.1. Reference free

One of the issues with image captioning is that an image can have numerous captions that are appropriate. Traditionally, a set of reference captions is used to evaluate the caption's accuracy. There may, however, be other descriptions that are not included in the reference captions but are still acceptable. Reference-free measures seek to address this issue by obviating the need for reference captions.

CLIPScore

CLIPScore [17] makes use of CLIP in order to compute the similarity between the image and the caption generated by the model. In CLIP, the text and image encoders each produce a single vector, which is intended to represent the content of an input caption or image.

In order to compute the CLIPScore metric, the image and the caption are passed to the image encoder and the text encoder respectively. The cosine similarity of the resulting embeddings is then calculated and rescaled by a factor $w = 2.5$. The final score is computed as shown in equation 5.1.

caption textual embedding — visual embedding

$$\texttt{CLIP-S}(c, v) = \mathrm{w} * \max(cos(\mathrm{c}, \mathrm{v}), 0) \tag{5.1}$$

scale parameter

5.2.2. Reference based

The idea behind image captioning reference based evaluation is to compare the model's candidate sentences to the reference sentences and assign a score.

[1] https://creativecommons.org/licenses/by/4.0/legalcode

RefCLIPScore

RefCLIPScore [17] is an extension of CLIPScore that is used when references are available. Therefore, this is not a reference-free metric. To compute RefCLIPScore, CLIP's text encoder is used to extract the textual embeddings of each reference. The cosine similarity between the textual embedding of the caption generated by the model and each reference embedding is computed. Then, RefCLIPScore is calculated as the harmonic mean of CLIPScore and the maximal reference cosine similarity (equation 5.2).

caption textual embedding — visual embedding

$$\texttt{RefCLIP-S}(c, \mathrm{R}, v) = \text{H-Mean}(\texttt{CLIP-S}(\mathrm{c}, \mathrm{v}), \max(\max_{r \in R} cos(c, \mathrm{r}), 0)) \tag{5.2}$$

set of references — reference textual embedding

BLEU

The Bilingual Evaluation Understudy metric (BLEU) [13] was designed to assess the quality of machine-translated text and it is now the quintessential metric in image captioning assignments. The metric measures how close the machine translation is to the human translation by counting the number of identical n-grams in the two texts.

The BLEU metric for n-grams of size n (w_n) given a candidate sentence and a collection of reference sentences is computed as follows:

maximum **reference** counts of w_n — count of w_n in $\hat{y}$

$$p_n = \frac{\sum_{w_n \in \hat{y}} \min(m(w_n), \mathrm{c}(w_n, \hat{y}))}{\sum_{w_n \in \hat{y}} c(w_n, \hat{y})} \tag{5.3}$$

where $\hat{y}$ is the candidate sentence and $m(w_n)$ is the maximum reference counts for w_n which is computed as follows:

count of w_n in the j^{th} reference

$$m(w_n) = \max_{j=1,\ldots,|\mathrm{R}|} c(w_n, y_j) \tag{5.4}$$

number of references

METEOR

The Metric for Evaluation of Translation with Explicit ORdering (METEOR) [14] is based on the harmonic mean of unigram precision and recall. It incorporates stemming and synonymy matching, which aren't seen in other measures. METEOR is a language-dependent metric that seeks correlation at the sentence level, whereas BLEU seeks correlation at the corpus level.

The precision (P) and recall (R) of the unigrams must be calculated in order to compute the METEOR score. Given a reference sentence (r) and a candidate sentence (c), the precision and recall are computed as follows:

$$P = \frac{m}{\sum_{w \in c} 1} = \frac{m}{|c|} \tag{5.5}$$

$$R = \frac{m}{\sum_{w \in r} 1} = \frac{m}{|r|} \tag{5.6}$$

where m is the number of unigrams that appear in both the candidate and reference sentences and $|c|$ and $|r|$ denote the length of the candidate sentence and reference sentence respectively. The harmonic mean is then used to combine precision and recall:

$$F_{mean} = \frac{10 \cdot P \cdot R}{R + 9P} \tag{5.7}$$

The current criteria only account for congruity in terms of single words, not broader segments that appear in both the reference and candidate sentences. Longer n-gram matches are utilized to compute a penalty p for the alignment in order to account for the congruency of larger n-grams. The penalty will be increased by the number of mappings that are not adjacent in the reference and candidate sentences.

Unigrams are clustered into the fewest possible chunks to compute this penalty, with a chunk defined as a set of unigrams that are contiguous in both the candidate and the reference. The following formula is used to calculate the penalty p:

number of chunks

$$p = \frac{1}{2}\left(\frac{\mathrm{k}}{\mathrm{m}}\right)^3 \tag{5.8}$$

total amount of mapped unigrams

If there are no bigram or longer matches, the penalty has the effect of lowering the F_{mean} by up to 50%. The final metric is calculated as follows:

$$M = F_{mean}(1 - p) \tag{5.9}$$

ROUGE

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [15] includes measurements that compare a summary's quality to that of other (ideal) summaries generated by humans. Between the computer-generated summary to be evaluated and the ideal summaries written by humans, the measurements count the amount of overlapping units such as n-grams, word sequences, and word pairs. ROUGE-L: Longest Common Subsequence is the most common metric used to evaluate the results of the image captioning task.

In order to calculate the ROUGE-L score, precision and recall must be computed. Let $LCS(c, r)$ be the candidate sentence's and reference sentence's longest common subsequence. The precision and recall are then computed as follows:

$$P = \frac{|LCS(c, r)|}{|c|} \tag{5.10}$$

$$R = \frac{|LCS(c, r)|}{|r|} \tag{5.11}$$

The ROUGE-L score is then computed as shown in equation 5.12. The ROUGE-L score will fall between 0 and 1, and the closer it gets to 1, the closer the candidate sentence c resembles to the reference sentence r.

$$\texttt{ROUGE-L}(c, r) = 2 \cdot \frac{1}{\frac{1}{P} + \frac{1}{R}} \tag{5.12}$$

CIDER

The Consensus-based Image Description Evaluation score (CIDEr) [16] was proposed expressly for image captioning. When compared to other metrics, it gives more importance to relevant n-grams and has a higher correlation with human consensus scores.

Given an image I_i, a candidate caption c_i and a set of m reference captions: $S_i = \{s_{i1}, s_{i2}, \ldots, s_{im}\}$. Firstly, the words in both the references and the candidate description are mapped to their common stem or root form, this is, "fishing", "fished", "fisherman" would be rewritten as "fish". Sentences are represented as the set of n-grams present in the sentence. For instance, if just unigrams are used, the sentence: "a cat is playing" would be represented as {'a', 'cat', 'is', 'playing'} . The CIDEr score for n-grams of size n is computed as follows:

$$\texttt{CIDEr}_n(c_i, S_i) = \frac{1}{m}\sum_{j=1}^{m}\frac{g^n(c_i)\cdot g^n(s_{ij})}{||g^n(c_i)||\cdot||g^n(s_{ij})||} \tag{5.13}$$

where $g^n(c)$ represents the term frequency–inverse document frequency (TF-IDF) of all n-grams in c. By using the TF-IDF score, n-grams that occur across all images are given less importance. CIDEr is based on the assumption that n-grams that are relevant for an image will appear frequently in the image's set of references and n-grams that occur often in the reference captions of other images are less relevant and will be given less importance by using the inverse-document-frequency (IDF) term. The TF-IDF score for an n-gram w_k is computed as follows:

$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum_{w_l\in\Omega} h_l(s_{ij})}\log\left(\frac{|I|}{\sum_{I_p\in I}\min(1,\sum_q h_k(s_{pq}))}\right) \tag{5.14}$$

where Ω is the vocabulary of all n-grams, $h_k(c)$ is the frequency of the n-gram w_k in the sentence c and I is the set of images.

The final CIDEr score is computed as a weighted average of $\texttt{CIDEr}_n$ for $n = 1, 2, 3, 4$:

$$\texttt{CIDEr}(c_i, S_i) = \sum_{n=1}^{N} w_n \cdot \texttt{CIDEr}_n(c_i, S_i) \tag{5.15}$$

where w_n are set to $\frac{1}{N}$ and $N = 4$.

SPICE

Instead of computing the similarity between an image and a candidate caption based on n-gram similarity as other metrics like METEOR, BLEU, and CIDEr do, the Semantic Propositional Image Caption Evaluation measure (SPICE) [12] focuses on the semantic propositions of the text.

SPICE represents semantic propositional content using *scene-graphs*. A sentence s is parsed into a set of object classes C, a set of relation types R and a set of attribute types A. Formally, s is defined as follows:

$$G(s) =< O(s), E(s), K(s) > \tag{5.16}$$

where $O(s) \subseteq C$ is the set of object mentions in s, $E(s) \subseteq O(s) \times R \times O(s)$ is the set of hyperedges representing relations between objects, and $K(s) \subseteq O(s) \times A$ is the set of attributes associated with objects. The SPICE score is computed as the F1-score between the scene-graph tuples of the candidate caption and the reference sentences after the references and candidate caption have been turned into scene graphs. Similarly to METEOR, SPICE also uses synonyms for matching the tuples. Figure 5.2 illustrates an example of a parsed scene graph.

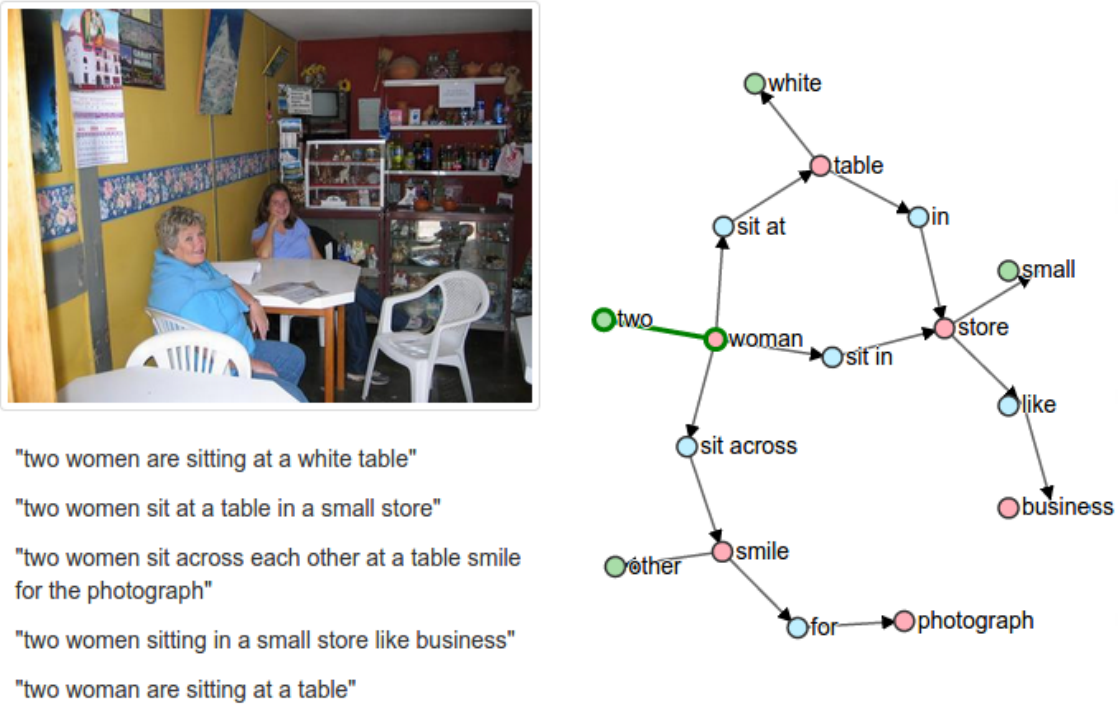


Figure 5.2: Example of a scene graph (right) parsed from a set of reference image captions (left)
Image source: [12]

5.3 Environment

We used two machines offered by the Pattern Recognition and Human Language Technology (PRHLT) research center of the Polytechnic University of Valencia to do this work.

On the hardware side, these machines are powered by two Nvidia GeForce RTX 2080 GPUs allowing us to speed up the training of our models. Regarding the software used, the PRHLT machines are running the Ubuntu 20 operating system and the Nvidia CUDA library for GPU acceleration. The runtime environment is composed of the free Anaconda 6 distribution of the programming language Python under the 3.9 version. To keep project requirements separate, virtual conda environments were employed.

Jupyter Notebooks are documents that combine code with text, images, and equations. Notebooks allow to run code, create visualizations and display them all in a single document, which is useful in research applications. In this work, Notebooks were used to create visualizations and perform several experiments.

Other libraries were installed as well. Namely:

- **Pytorch:** An end-to-end open source machine learning platform. Its API is based on the Torch library and is available for Python. The library provides a wide range of

high-level abstractions that speed up the creation of machine learning models and make code execution across a variety of computing resources easier. The Transformers library from Huggingface is built on Pytorch, which is where GPT-2 is implemented.

- **TensorFlow:** An open source deep learning library similar to Pytorch that allows for easy prototyping and deployment of machine learning models. It was developed by Google and may be used for a variety of applications, with an emphasis on deep neural networks.
- **Keras:** A high-level deep learning API that runs on top of TensorFlow and allows for rapid model creation and experimentation.
- **Transformers:** An open source library created by Huggingface that provides state of the art pre-trained models. These ready-to-use models work with Pytorch and Tensorflow.
- **CLIP:** An open source library created by OpenAI which implements the pre-trained CLIP neural network.
- **Pymoo:** An open source framework that implements state of the art single- and multi-objective genetic algorithms.
- **Numpy:** A library for scientific computing with Python.
- **Matplotlib:** A library for plotting data in Python.
- **Requests:** Allows to send http requests in Python.
- **Pandas:** An open source Python library for data analysis.

5.4 Architectures

5.4.1. CNN + Transformer

As detailed in section 4.5, the vanilla Transformer architecture may be used for image captioning by combining it with a CNN model. In this first approach, we suggest utilizing the vanilla Transformer with several CNNs that have previously been pre-trained using the ImageNet dataset [26] to encode the image. The architecture design is shown in figure 5.3. As it can be seen, the image is encoded into a set of feature vectors which are used as input for the Transformer model. The CNN architectures that have been used are the following: EfficientNetB0 [43], VGG16 [44] and ResNet50 [45].

Implementation

The original code for this approach's implementation was obtained from [66]. The original repository used the EfficientNetB0 as the CNN, the code was modified in order to also use the VGG16 and the ResNet50 CNNs [2].

[2]The modified repository can be found here: https://github.com/JAVI897/Image-Captioning

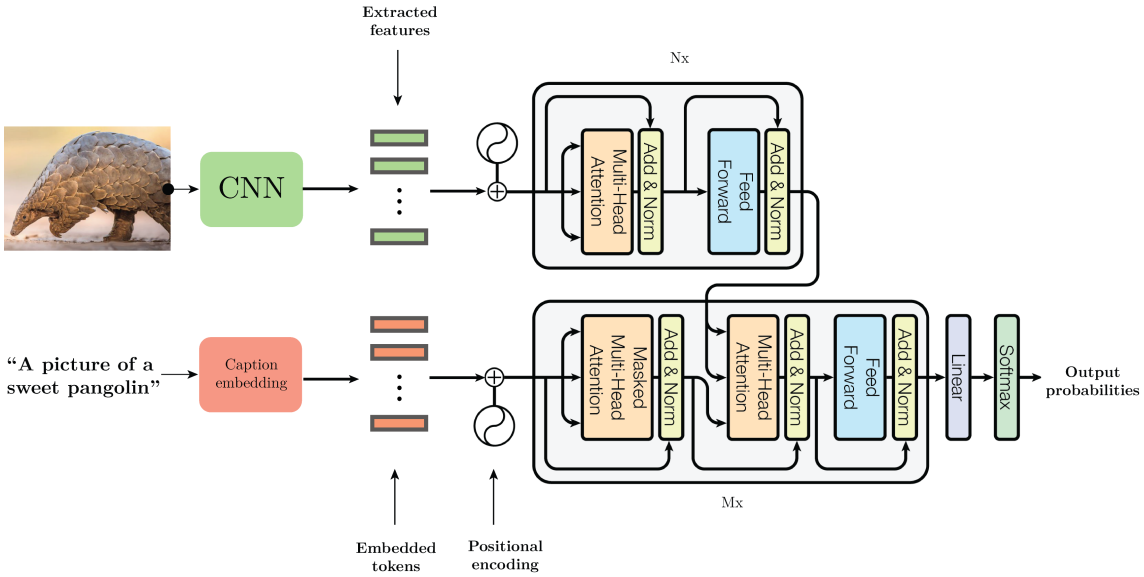


Figure 5.3: CNN + Transformer architecture

Training

In this section, the training adjustments of the aforementioned models will be described. The dataset is divided into three subsets: training, validation and test. The training set is used to train the model. The validation set is used in the validation process and the test set is used to evaluate the final model.

The optimization algorithm used was Adam (explained in section 3.5.1) and the selected loss was the cross-entropy loss (equation 4.10). Because the training times were long, the batch size was set to be as large as possible, as the graphics cards' capacity is restricted; in this case, the batch size was 64. The *EarlyStopping* callback was used, this is, the training of the model was stopped if after three iterations of the Adam algorithm, the loss did not improve in the validation set. In figure 5.4 it is shown the training loss for the CNN + Transformer models. As can be seen, the best model in the validation set is the one that uses the EfficientNetB0. Results on the test subset are discussed in chapter 6.

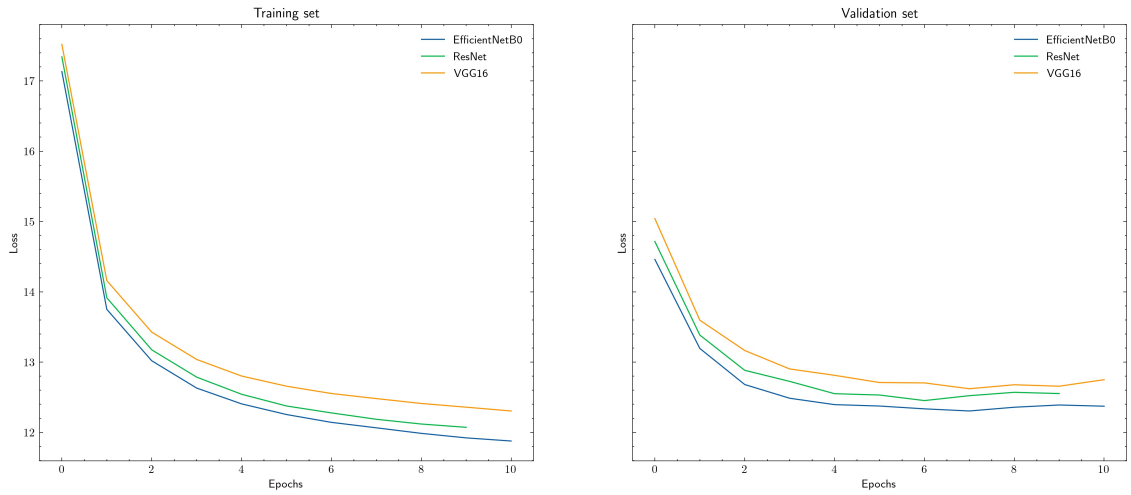


Figure 5.4: Graphs of the performance of the loss function in the training set (left) and validation set (right) of the models in each epoch.

5.4.2. Mapping Network

The Mapping Network architecture refers to the ClipCap model explained in section 4.6.1. Several modifications based on the vanilla ClipCap architecture are proposed in order to generate captions that maximize the CLIPScore metric when generating the caption, these methods are: Mapping Network + Genetic, Mapping Network Max Similarity and Mapping Network + Greedy Search based on CLIPScore.

Implementation

The code for these approaches was modified from the ClipCap publication [32] [3]. The model was trained from scratch with the MSCOCO dataset as it was already explained in section 4.6.1. The same configuration as the original publication for training was employed with some minor changes; the Transformer architecture was used as the mapping network, the l hyperparameter was set to 40, and the GPT-2 model was frozen during training, meaning no fine-tuning was done.

Mapping Network + Genetic

In the ClipCap architecture, the Transformer model outputs a set of l vectors of dimensions 768 which can be thought of as word embeddings without textual meaning that tame the language model. A matrix P can be used to represent those embeddings (equation 5.17). The main notion behind this variant of the vanilla ClipCap architecture is that by optimizing the CLIPScore metric during inference, those prefixes can be enhanced and therefore, the candidate caption can be improved.

$$P^i = [p_1^i, \ldots, p_l^i],\ \mathbb{R}^{768 \times l} \tag{5.17}$$

Inspired by the work of Federico Galatolo et al. [11], we propose to use a genetic algorithm to maximize the CLIPScore metric starting with the prefixes obtained by the Transformer (mapping network). The architecture is illustrated in figure 5.5.

When addressing a single objective optimization, a traditional GA can be used. In this approach, the NSGA-II algorithm [67] has been used. The advantage of using a GA is that it is independent from the type of architecture for generating text. More details about genetic algorithms can be found in appendix A.

The GA is provided with the prefixes generated by the mapping network. The set of prefixes available in the GA's population feeds the generator. The text generated by the generator is used to feed the CLIP text encoder. Finally, the CLIPScore metric is computed by combining the textual and visual embeddings. The GA optimization problem consists in finding the best prefixes that maximize the CLIPScore metric.

In order to create the population for the GA, firstly the prefixes obtained from the mapping network are flattened into a vector z of dimensions $768 * l$. Then, 20 individuals are created by adding random vectors of dimensions $768 * l$ sampled from a standard normal distribution (mean 0 and variance 1) to z. The crossover and mutation operators used are the k-point crossover operator and the polynomial mutation operator respectively.

[3] The modified repository can be found here: https://github.com/JAVI897/CLIP_prefix_caption

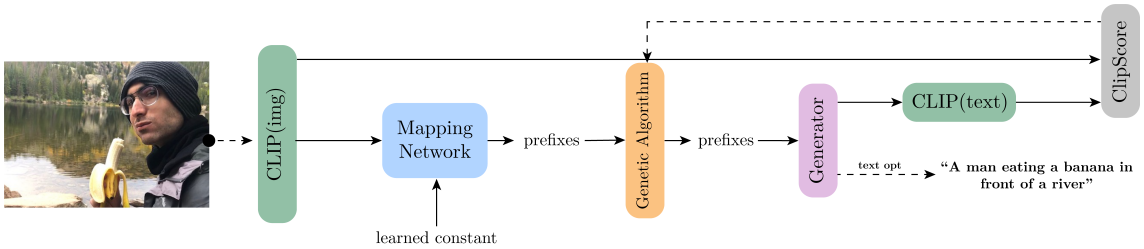


Figure 5.5: Genetic algorithm approach

Mapping Network Max Similarity

A beam search strategy can be used by the GPT-2 language model to generate a list of candidate captions of size N. The primary premise of this variant is that by choosing the hypotesis with the highest CLIPScore metric, the final candidate caption will better describe the input image. The objective function is illustrated in equation 5.18.

$$\max_{h \in H} \texttt{CLIP-S}(\texttt{CLIP}_{TE}(h), v) \tag{5.18}$$

candidate caption

CLIP visual embedding

set of hypotesis

Mapping Network + Greedy search based on CLIPScore

As explained in section 3.7.3, GPT-2 may generate text using a variety of strategies. We propose a novel greedy search that takes into account the CLIPScore metric. Because CLIPScore is a reference-free metric that does not require a reference sentence, it may be used to direct the greedy search to the tokens that best describe the input image. The complete pseudocode of this approach can be found in appendix B.

At each timestep t, greedy search just chooses the word with the highest probability as its next term.

$$w_t = \arg\max_{w} \mathrm{p}(w \,|w_1, \ldots, w_{t-1}) \tag{5.19}$$

Instead, we propose to select the next word that maximizes a linear combination between the CLIPScore metric and the conditional probability $\mathrm{p}(w \,|w_1, \ldots, w_{t-1})$. We do not use the CLIPScore metric itself to direct the search since the CLIPScore would always assign more probability to non-empty tokens.

Let L be a list containing the N next tokens with the highest probability.

$$L = \{o_1, \ldots, o_N\} \tag{5.20}$$

For each next token o_i in L, the CLIP textual embedding is computed for the sentence already generated but adding the token o_i.

$$c_i = \texttt{CLIP}_{TE}(w_1, \ldots, w_{t-1}, o_i) \tag{5.21}$$

Then, for each next token o_i in L, the linear combination is computed as follows:

$$s_i = \beta \, \mathrm{p}(o_i \mid w_1, \ldots, w_{t-1}) + (1 - \beta) \, \frac{1}{2.5} \, \texttt{CLIP-S}(c_i, \, v) \tag{5.22}$$

where v is the visual embedding of the image obtained with the CLIP visual encoder. The importance of the conditional probability of the token o_i is controlled by the hyperparameter β. It is worth noting that as β falls, the decoder prioritizes the CLIPScore measure. The N hyperparameter controls the amount of tokens to select. These hyperparameters will be chosen using a training set of 300 pairs of sentences and images extracted from the MSCOCO dataset and it will be validated using the Karpathy test partitions of MSCOCO.

Finally, the objective function would be the following:

$$w_t = \underset{i=1,\ldots,N}{\arg\max} \, s_i \tag{5.23}$$

This approach is summarized in figure 5.6.

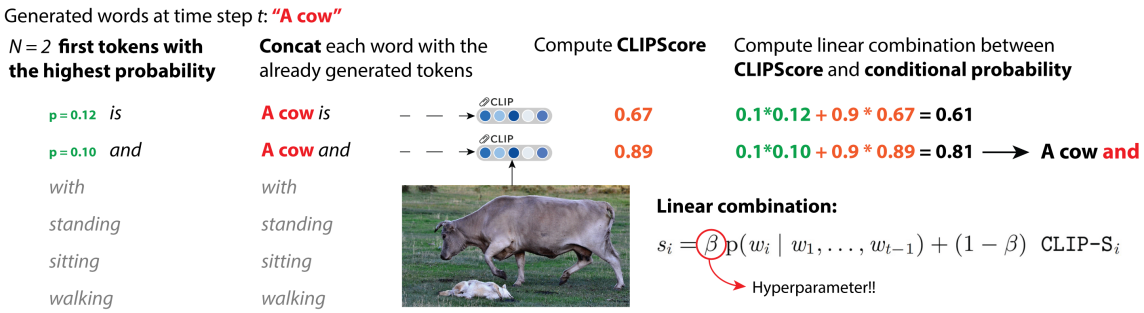


Figure 5.6: Example of the greedy search based on CLIPScore

CHAPTER 6
Results

In this chapter, the architectures proposed in section 5.4 will be evaluated using the test subset of the Karpathy et al. splits [18] with the metrics presented in the previous chapter[1] . Results are compared with a baseline model which consists in using the ClipCap architecture with the beam search strategy described in section 3.7.3. Finally, the proposed methods will be compared with the state of the art image captioning models and some examples will be discussed.

6.1 CNN + Transformer

6.1.1. Quantitative evaluation

The results for the models that combine a CNN with the Transformer architecture are given in table 6.1. The model that employs the EfficientNetB0 CNN outperforms the baseline model in the BLEU-1 score. Also, it obtains the same score for the BLEU-2 metric. However, for the remaining supervised metrics, the baseline model obtains better results. The transformer-based models perform worse than the baseline model in the unsupervised CLIPScore metric which can be due to the fact that the baseline model uses CLIP to encode the image, therefore their captions could be more correlated with the CLIPScore metric. In comparison to the baseline model, the results show that the model which uses the EfficientNetB0 CNN can better predict the captions' unigrams. However, as compared to the baseline model, it performs worse in terms of caption semantic structure metrics such as CIDEr or SPICE.

Model	B@1	B@2	B@3	B@4	M	R	C	S	CLIP-S	CLIP-S^{ref}
Beam search	0.66	**0.50**	**0.37**	**0.27**	**0.26**	**0.53**	**0.95**	**0.20**	**0.77**	**0.81**
EfficientNet + Transformer	**0.69**	**0.50**	0.35	0.25	0.23	0.50	0.81	0.16	0.70	0.76
VGG + Transformer	0.65	0.46	0.32	0.22	0.21	0.47	0.67	0.14	0.66	0.72
ResNet + Transformer	0.66	0.48	0.33	0.23	0.22	0.48	0.73	0.15	0.68	0.74

Table 6.1: Metrics for CNN + Transformer approach. We report supervised metrics (those that require human references): B@1 = Bleu 1, B@2 = Bleu 2, B@3 = Bleu 3, B@4 = Bleu 4 [13], M = METEOR [14], R = ROUGE [15], C = CIDEr [16], S = SPICE [12]. Finally, we report semantic relatedness to the image (CLIP-S [17]), and to the human references (RefCLIPScore [17]) based on CLIP's embeddings.

[1]Metrics were computed using the following framework: https://github.com/jmhessel/clipscore

6.2 Mapping Network Max Similarity

6.2.1. Quantitative evaluation

In table 6.2 it is shown the results for the max similarity approach in the Karpathy split. We first compare the results for the following supervised metrics: BLEU, METEOR, ROUGE, CIDEr and SPICE. As can be observed, the gap widens as the beam size grows. Because there are more hypothesis to choose from as the beam size increases, the first hypothesis, which is the one chosen by the beam search with a beam size of 5, has a lower likelihood of being chosen and as a result other hypothesis with lower values in these supervised metrics are selected. Regarding the unsupervised CLIPScore metric, the max similarity approach is better than the baseline model (0.77 vs 0.80). Finally, the max similarity approach also outperforms the baseline in the RefCLIPScore metric (0.81 vs 0.83).

Model	Beam ↓	B@1	B@2	B@3	B@4	M	R	C	S	CLIP-S	CLIP-S^{ref}
Beam search	5	**0.66**	**0.50**	**0.37**	**0.27**	**0.26**	**0.53**	**0.95**	**0.20**	0.77	0.81
Max Sim	5	**0.66**	**0.50**	0.36	0.26	**0.26**	0.52	0.93	**0.20**	**0.80**	**0.83**
Max Sim	10	0.59	0.45	0.32	0.23	0.26	0.51	0.86	0.20	**0.80**	**0.83**
Max Sim	15	0.54	0.40	0.29	0.20	0.25	0.49	0.80	0.20	**0.80**	**0.83**
Max Sim	20	0.50	0.37	0.26	0.18	0.25	0.48	0.75	0.20	**0.80**	0.82

Table 6.2: Metrics for the max similarity approach. We report supervised metrics (those that require human references): B@1 = Bleu 1, B@2 = Bleu 2, B@3 = Bleu 3, B@4 = Bleu 4 [13], M = METEOR [14], R = ROUGE [15], C = CIDEr [16], S = SPICE [12]. Finally, we report semantic relatedness to the image (CLIP-S [17]), and to the human references (RefCLIPScore [17]) based on CLIP's embeddings.

6.3 Mapping Network + Genetic

6.3.1. Quantitative evaluation

In the genetic approach, the population size was set to 20 individuals and the number of generations was set to 3. The results of the Karpathy split are shown in table 6.3. In metrics that are directly dependent on the references (e.g. BLEU, METEOR, CIDEr, etc.), the genetic method does not outperform the baseline model. As a result, the captions created by this method do not match the reference captions exactly. With the CLIPScore and RefCLIPScore measures, however, it outperforms the baseline model. This indicates that the approach produces accurate captions, although they differ from the reference captions.

Model	B@1	B@2	B@3	B@4	M	R	C	S	CLIP-S	CLIP-S^{ref}
Beam search	**0.66**	**0.50**	**0.37**	**0.27**	**0.26**	**0.53**	**0.95**	**0.20**	0.77	0.81
Genetic	0.61	0.43	0.29	0.19	0.23	0.47	0.70	0.16	**0.81**	**0.83**

Table 6.3: Metrics for genetic approach. We report supervised metrics (those that require human references): B@1 = Bleu 1, B@2 = Bleu 2, B@3 = Bleu 3, B@4 = Bleu 4 [13], M = METEOR [14], R = ROUGE [15], C = CIDEr [16], S = SPICE [12]. Finally, we report semantic relatedness to the image (CLIP-S [17]), and to the human references (RefCLIPScore [17]) based on CLIP's embeddings.

6.4 Mapping Network + Greedy Search based on CLIPScore

6.4.1. Hyperparameter search

An exhaustive search through a subset of the hyperparameter space of the greedy algorithm has been performed in order to find the optimal values for N and β. To perform the grid search, the following subsets of N and β were explored:

$$\beta = \{0, 0.05, 0.1, 0.15, 0.2, \ldots, 1\} \tag{6.1}$$
$$N = \{5, 10, 15, 20, 25, 30, 35, 40\} \tag{6.2}$$

The experiments were carried out with 300 photos chosen at random from the MSCOCO dataset's training set. The optimal hyperparameters were selected such that the RefCLIPScore was maximized. Nevertheless, other metrics were computed in order to better understand the impact of the hyperparameters in the resulting captions.

In figure 6.1 it is shown the results obtained for the CLIPScore and RefCLIPScore metrics. As can be observed, the CLIPScore measure rapidly drops as β increases because β controls the influence of the CLIPScore metric on the resultant caption. When β is greater than 0.25, the N hyperparameter does not appear to be related to the CLIPScore measure. When β is less than 0.25, however, the captions improve as N grows. This is because as N grows, more words can be chosen as the next word in the caption, increasing the probability of selecting the best word based on the CLIPScore measure. Regarding the RefCLIPScore measure, like the CLIPScore metric, drops as β increases. When β is set to zero, the greedy search, unlike the CLIPScore metric, produces the worst results. This is because when β is set to zero, the greedy search relies exclusively on the unsupervised CLIPScore metric, which can generate captions that are unrelated to the reference captions. This, however, does not imply that the caption created is incorrect; rather, it indicates that the caption does not match the reference captions.

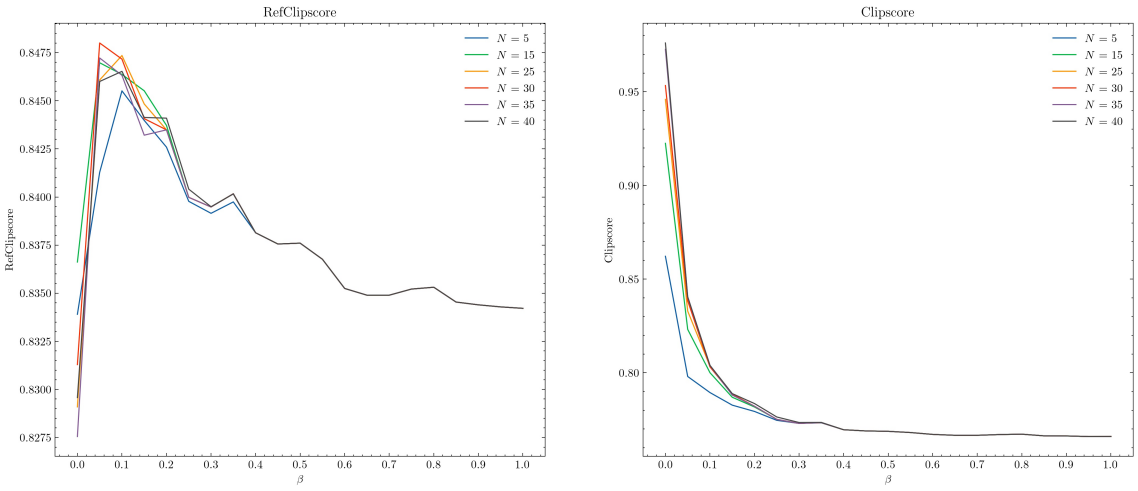


Figure 6.1: On the left the results for the RefCLIPScore metric for different values of β and N are presented. The unsupervised CLIPScore metric's results are presented on the right. Both results are computed using a sample of 300 photos from MSCOCO.

In figure 6.2 the results for different BLEU metrics are presented. As it is seen, unlike the unsupervised CLIPScore metric, in the BLEU metric as β grows the BLEU score increases. This is because the BLEU metric is supervised and as β increases, the greedy search is less directed by an unsupervised metric as it is the CLIPScore metric. As a result, the captions

are more similar to those seen in the training set, and hence to the reference captions used to compute the BLEU measure.

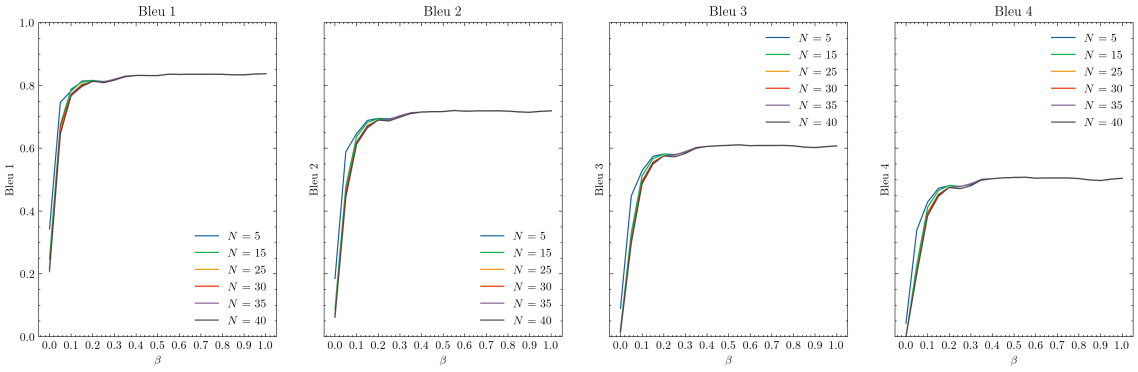


Figure 6.2: BLEU metrics for different values of β and N. Results are computed using a sample of 300 photos from MSCOCO.

The top three best results regarding the RefCLIPScore metric are presented in table 6.4. As can be seen, the best results are obtained with a value of β of 0.05 or 0.1. Moreover, in order to see if there are significant differences between the top three combinations of hyperparameters, we used approximation randomization testing (ART) [68] [2] with 10,000 repetitions and a p-value of 0.05. Results of the ART tests are presented in figure 6.3. As can be observed, the p-values are greater than the significance level (0.05), thus the top three combination of hyperparameters do not show significant differences for the RefCLIPScore metric.

β	N	B@1	CLIP-S	CLIP-S$^{\text{ref}}\uparrow$
0.05	30	0.669	0.839	0.848
0.10	25	0.772	0.804	0.847
0.10	20	0.782	0.802	0.847

Table 6.4: The top three best results for the RefCLIPScore metric for a sample of 300 images randomly selected from MSCOCO. We report supervised and unsupervised metrics: B@1 = Bleu 1 [13], CLIP-S = CLIPScore [17], CLIP-S$^{\text{ref}}$ = RefCLIPScore[17].

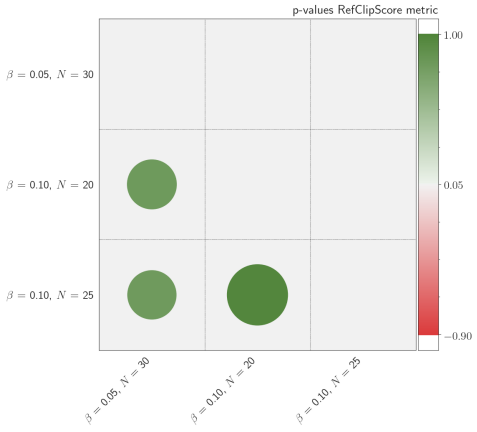


Figure 6.3: p-values of ART tests.

6.4.2. Length distribution

In order to understand how the β hyperparameter affects the length of the generated captions, the histograms of the lengths of the captions generated for different values of β were analyzed on the sample of 300 images randomly selected from the MSCOCO dataset. In figure 6.4 it is shown the histogram of the lengths of the captions for various values of β. When β is set to 0, the distribution is skewed to the right, which means that captions tend to be longer when β is set to 0. This is due to the fact that when β is set to zero, the search is completely guided by CLIP and it assigns more probability to not empty words. The length distribution, on the other hand, becomes centered around 10

[2] For the calculations, we utilized the following framework: https://github.com/midobal/mt-scripts/tree/master/art

words as β grows. It is interesting to see how the kurtosis changes when the N hyperparameter increases when β is set to 0.05. As it is shown in figure 6.5, as N increases, the variance of the distribution decreases, resulting in the generation of more phrases with a length of 10 words. This penhomena is just observed when β is set to 0.05. The length does not appear to be affected by the N hyperparameter when β is greater than 0.05.

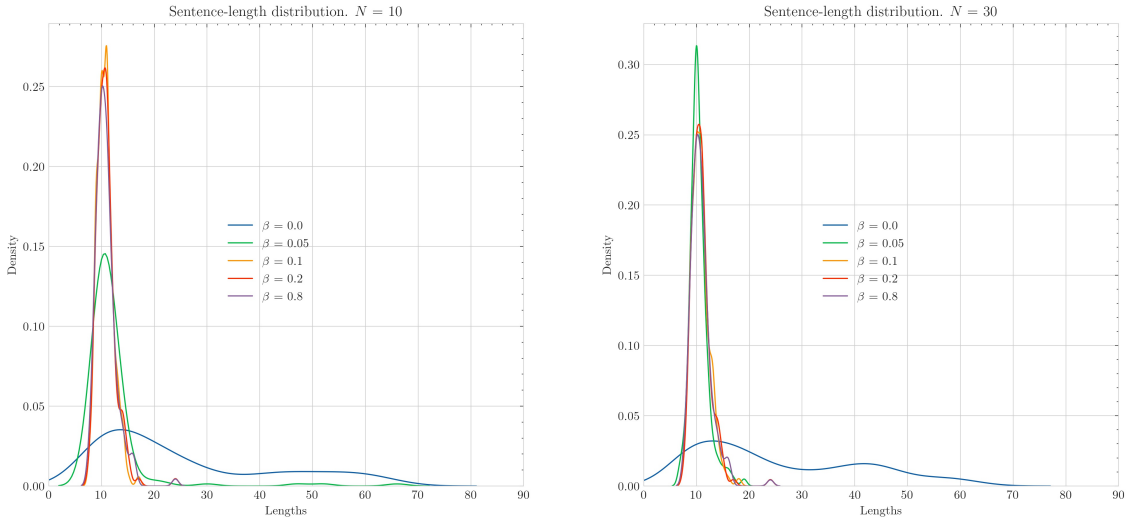


Figure 6.4: Caption's length distributions for different values of β. On the right, the results for $N = 10$ are shown. Results for $N = 30$ are presented on the left.

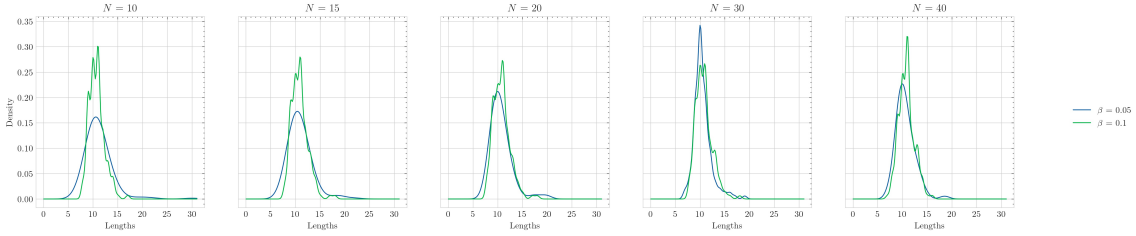


Figure 6.5: Caption's length distributions for different values of β and N.

6.4.3. Quantitative evaluation

The Karpathy split was used to validate the top three combinations of hyperparameters according to the RefCLIPScore metric because there were no significant differences between them. Results are shown in tables 6.5 and 6.6.

The results for the various BLEU metrics for n-grams of size n are shown in table 6.5. The proposed greedy search, as can be observed, outperforms the beam search in the Bleu 1 metric for β values of 0.1. This indicates that the technique accurately predicts the precise words in the reference captions. For the Bleu 2 metric, however, the greedy search does not outperform the Beam search despite the fact that the results are nearly identical. For more than 2 words (Bleu 3 and Bleu 4), the greedy search seems to obtain lower results than the beam search strategy.

Model	$\beta \uparrow$	$N \downarrow$	B@1	B@2	B@3	B@4
Beam search	-	-	0.6640	**0.5032**	**0.3692**	**0.2674**
Greedy search + CLIPScore	0.10	20	**0.6834**	0.4959	0.3438	0.2347
Greedy search + CLIPScore	0.10	25	0.6811	0.4927	0.3405	0.2316
Greedy search + CLIPScore	0.05	30	0.6078	0.4000	0.2488	0.1524

Table 6.5: BLEU [13] metrics for greedy approach. B@1 = Bleu 1, B@2 = Bleu 2, B@3 = Bleu 3, B@4 = Bleu 4.

In table 6.6 we present our results for other supervised and unsupervised metrics. We first consider the following supervised metrics: METEOR, ROUGE, CIDEr and SPICE. As can be seen, the greedy search lags in these metrics in comparison to the beam search approach. CLIPScore is an unsupervised metric which measures the relatedness between the image and the caption without references. Evidently, the greedy search outperforms the beam search approach in this metric (0.84 vs 0.77). The RefCLIPScore is used to measure the semantic distance from the references. In this case, the greedy search is much better in this metric than the beam search method.

Model	$\beta \uparrow$	$N \downarrow$	M	R	C	S	CLIP-S	CLIP-S^{ref}
Beam search	-	-	**0.2596**	**0.5278**	**0.9535**	**0.1959**	0.7729	0.8150
Greedy search + CLIPScore	0.10	20	0.2359	0.4942	0.8299	0.1749	0.8111	0.8339
Greedy search + CLIPScore	0.10	25	0.2343	0.4912	0.8198	0.1738	0.8135	0.8347
Greedy search + CLIPScore	0.05	30	0.2066	0.4348	0.6177	0.1453	**0.8452**	**0.8421**

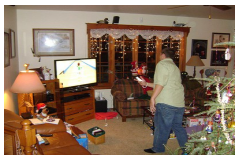
Table 6.6: We report supervised metrics (those that require human references): M = METEOR [14], R = ROUGE [15], C = CIDEr [16], S = SPICE [12]. Finally, we report semantic relatedness to the image (CLIP-S [17]), and to the human references (RefCLIPScore [17]) based on CLIP's embeddings.

6.4.4. Qualitative Analysis

In this section, we will compare the captions generated for several images acquired from the MSCOCO dataset's validation set for various β and N configurations. Results are shown in table 6.7. We discuss each picture from left to right.

First, the method with $\beta = 0.1$ properly detects the church's location, demonstrating CLIP's real-world knowledge whereas the method with $\beta = 0.05$ does not albeit it is more directed by CLIP's visual content. Next, the method with $\beta = 0.05$ attempts to describe the written text in the stop sign wheras the method with larger values of β does not. The ability of describing the text on an image has recently been explored by Tewel et al. in [33] where they changed the context tokens given to GPT-2 from "Image of a" to "Image of text that says." The following image is wrongly described by the method with $\beta = 0.05$ which thinks that the man eating a banana is naked. The following picture is wrongly described by all approaches, which do not appear to comprehend the scene's context. In the first image of the second table, the approach with $\beta = 0.05$ provides a more detailed description of the scene. Similarly, the approach that is more influenced by CLIP's visual content in the next image provides a more complete description of the picture. The following picture is correctly described by all approaches, albeit the method with $\beta = 0.05$ appears to be more concrete about the location. In the following picture when β is set to 0.05, the technique defines the plane as a "Deltajet 747" which is a plane

Table 6.7: Generated captions for ten images from the MSCOCO test set (Karpathy et al. [18] split).

				
Ground Truth	A very large castle off in the back ground behind some trees.	A stop sign with the phrase "hammer time" written on it.	A young man with a hat on trying to eat a banana.	Dog about to snap at a cow in an open pasture.
$\beta = 0.10$ $N = 20$	A picture of a church in Barcelona with a garden and a castle.	A stop sign that has been vandalized by a man.	A man with a banana taking a bite of a banana.	A cow biting a calf in the stomach.
$\beta = 0.05$ $N = 30$	A view of a medieval cathedral in the background.	A stop sign that has time on it.	A banana is being eaten by a man wearing nothing but hat.	A calf feeds a cow on a grassy field.
				
Ground Truth	The living room is all decorated for Christmas.	A group of people riding in a black raft.	A short train traveling through a rustic contryside.	A World Cargo 747 airline is taxiing down a runway.
$\beta = 0.10$ $N = 20$	A man playing Wii in a living room.	A group of people riding rafts in the water.	A train on a track near a grassy field.	A large white and blue plane sitting on a runway.
$\beta = 0.05$ $N = 30$	A Christmas decorated area with television games and man in shirt playing Wii.	A rafting group riding a river in the middle of a wet and turbulent water.	A train travelling through a rural area.	A Deltajet 747 sitting on a runway near a field.

that looks similar to the one in the image. As beta is increased, the method becomes less specific about the plane.

6.5 Comparison

In this section, the various systems will be quantitatively evaluated and compared to several state of the art models. Some visual examples are shown in the qualitative analysis.

The chosen criteria for those approaches that required hyperparameter tuning were to select the best model based on the RefCLIPScore metric. In the case of the proposed greedy search method, the β value was set to 0.05 and N was set to 30. Similarly, the beam size for the max similarity method was set to 5.

6.5.1. Quantitative evaluation

In table 6.8 it is shown the results for the the Karpathy split for different systems. Four recent baselines are compared: VinVL [69] which does not use CLIP to embed the image and it is a state of the art technique, ClipCap [32] which was presented in section 4.6.1, CLIP-VL [34] which uses spatial grid features extracted from CLIP and tewel2021zero [33] which is also based on CLIP. As can be seen, our methods lag in the METEOR, CIDEr and SPICE metrics in comparison with VinVL, ClipCap and CLIP-VL. Regarding the unsupervised CLIPScore metric, our methods based on maximizing the CLIPScore metric are much better than the CNN with a Transformer approaches. Moreover, these methods outperform the state of the art VinVL, CLIP-VL and ClipCap models in this metric. The tewel2021zero, on the other hand, achieves higher results in this metric than our

proposals. However, albeit the tewel2021zero method outperforms our approaches in the CLIPScore metric, the gap is narrower than in other state of the art models. Regarding the RefCLIPScore metric, our methods based on maximizing the CLIPScore metric outperform the state of the art models.

Model	METEOR	CIDEr	SPICE	CLIP-S	CLIP-S^{ref}
VinVL [69]	**0.311**	**1.409**	**0.252**	0.760	0.820
tewel2021zero [33]	0.115	0.146	0.055	**0.870**	0.790
CLIP-VL [34]	0.297	1.342	0.238	0.770	0.820
ClipCap [32]	0.271	1.083	0.201	0.770	0.810
Ours; Genetic	0.225	0.696	0.164	0.814	0.827
Ours; Greedy Search	0.207	0.618	0.145	0.845	**0.842**
Ours; Max Similarity	0.260	0.932	0.197	0.799	0.830
Ours; VGG + Transformer	0.208	0.672	0.135	0.657	0.722
Ours; ResNet + Transformer	0.218	0.732	0.146	0.677	0.739
Ours; EfficientNetB0 + Transformer	0.228	0.807	0.159	0.700	0.759

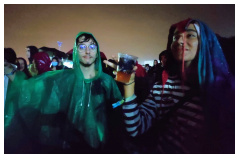
Table 6.8: Comparison of models. We report supervised metrics (those that require human references): METEOR [14], ROUGE [15], CIDEr [16], SPICE [12]. Finally, we report semantic relatedness to the image (CLIP-S [17]), and to the human references (RefCLIPScore [17]) based on CLIP's embeddings.

6.5.2. Image examples

In this section, the captions generated for different images will be compared. Firstly, table 6.9 shows four pictures taken with an smartphone. The results clearly demonstrate that models that combine a CNN with the Transformer architecture do not generalize well to the scenes presented. The VGG + Transformer model, for example, labels the first image as "A woman is holding a cell phone to her ear" which is incorrect because the image depicts two men in the middle of a throng. All of the ClipCap-based approaches produce proper captions in the first image, however the max similarity method and the genetic approach confuse the glass of beer with a glass of water. The greedy approach, which recognizes that the photo was shot during a music festival, yields the most correct caption for the first image. In the second image, the captions generated by the baseline model and the max similiarity approach do not produce accurate captions, as both describe the image as "A man is sitting at a table with his hand on his heart," which is incorrect because the man in the image does not have his hand on his heart. The genetic and greedy techniques provide the most accurate captions, albeit the first generates a more imaginative caption. The ResNet + Transformer technique provides an accurate caption in the following image. The genetic and greedy techniques, on the other hand, yield more accurate captions. The baseline model and max similarity technique, respectively, misidentify Santa Claus' statue as a basilisk and a teddy bear. In the last picture, all the ClipCap-based approaches generate accurate captions. However, the genetic approach mistakes the mask with a pillow.

Other images obtained from the internet are shown in table 6.10. In the first picture, all the models agree that the photo depicts teddy bears. However, whereas the ClipCap-based techniques indicate there are two teddy bears, the models that combine a CNN with the Transformer architecture do not. In the next picture, all the models incorrectly describe the image except for the greedy and genetic approaches. The greedy approach describes the image as "A newspaper advertisement for Spain featuring a man and a donkey". The inclusion of Spain in the description might be due to the fact that "El País" is

Table 6.9: Generated captions for smartphone photos.

				
Beam search	A group of people with umbrellas in the air.	A man is sitting at a table with his hand on his heart.	A group of people sitting next to a statue of a basilisk.	A man that is asleep in the back of a plane.
Greedy Search	A group of drinking fans outside a festival.	A man smiling while sitting at a table.	A Christmas tree with a man and a woman reading a book.	A man asleep breathing while wearing his mask.
Genetic	A group of people with different colors and sizes of glass bottles in the rain.	A young man with a smile on his face as he waits for his turn at a table.	A young boy reading a book in front of a statue of a man in a christmas costume.	A young man is sitting on the back of a train with his face covered in a pillow.
Max Similarity	A group of people with umbrellas and a jug of water.	A man is sitting at a table with his hand on his heart.	A group of people sitting next to a statue of a teddy bear.	A man that is asleep in the back of a plane.
EfficientNet + Transformer	A woman is holding a cell phone in her hand.	A woman holding a hot dog in her hands.	A woman sitting on a bench with a cell phone.	A woman is sitting in a chair with a laptop.
VGG + Transformer	A woman is holding a cell phone to her ear.	A man is eating a piece of pizza.	A group of people sitting around a table with a cake.	A man sitting in a chair with a laptop.
ResNet + Transformer	A woman and a man are holding a wii remote.	A man is holding a pair of scissors.	A woman and a man standing in front of a christmas tree.	A man is sitting in a car talking on his cell phone.

a spanish daily, however the image does not portray any advertisement involving a man and a donkey. Similarly, the genetic approach indicates that there is an advertisement in the newspaper which is not correct. In the third image, the models that combine a CNN with the Transformer architecture obtain less accurate descriptions whereas the ClipCap-based techniques give more specific captions. For instance, the greedy approach describes the image as "A horse flying through space with astronaut on top of him." Finally, with the exception of the greedy and genetic approaches, all of the models in the last image incorrectly describe the image. The greedy method defines the image as "A couple of people holding hands talking on a political crisis," which is more precise and recognizes that the couple in the photo are politicians.

Table 6.10: Generated captions for images obtained from the internet.

				
Beam search	A couple of teddy bears dressed in red and gold.	A portrait of a man in a white coat in front of a white wall.	A man riding a horse on the back of a spaceship.	A man standing next to a woman holding a phone.
Greedy Search	Two bears smoking spirits while posing for a photo.	A newspaper advertisement for Spain featuring a man and a donkey.	A horse flying through space with astronaut on top of him.	A couple of people holding hands talking on a political crisis.
Genetic	A pair of teddy bears of different ages standing in front of a bottle of whiskey.	A black and white image of an advertisement in a newspaper.	A picture of a silhouette of a man riding a horse in space.	A picture of two people standing next to each other in a room.
Max Similarity	A couple of teddy bears dressed in red and green.	A portrait of a man in a white coat and a flag.	A person riding a horse on the back of a spaceship.	A man standing next to a woman holding a phone.
EfficientNet + Transformer	A large brown teddy bear sitting on top of a table.	A sign that is on a pole with a sign.	A man riding a white horse on top of a field.	A man in a red shirt and a red tie.
VGG + Transformer	A teddy bear sitting on a table next to a glass of wine.	A sign that is on a pole in the middle of a street.	A man is playing a game of baseball.	A man in a suit and tie holding a cell phone.
ResNet + Transformer	A teddy bear with a glass of wine on its head.	A sign that is on a pole with a sign on it.	A horse that is standing up in the dirt.	A man and a woman are holding up a kite.

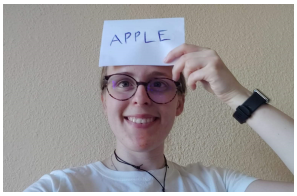
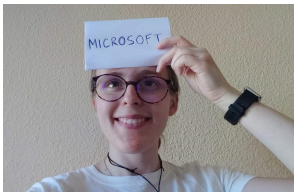
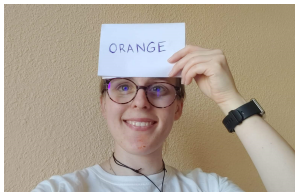
6.5.3. Textual cues

Goh, et al. [70] examined how CLIP can understand written text within an image. Image captioning models that make use of CLIP also take advantage of this capability as Tewel et al. showed in [33]. In order to understand our models' dependence on visual and textual cues we selected three images where in each of them a girl holds a paper with a written text. The greedy approach for different values of β and the baseline model are used to generate the caption for each image. Examples are shown in table 6.11. We discuss each picture from left to right.

The girl in the first image is holding a piece of paper with the word "APPLE" written on it. The baseline approach identifies the image as "A woman is holding up her iPhone." which is incorrect, but the confusion may result because the iPhone has an apple logo on it. The greedy technique for β values of 0.05 and 0.10 recognizes that the girl is holding an apple logo. However, when β is set to 0.20, the method describes the picture as "A woman with glasses holding up a sign." In the second image, the word written on the piece of paper is "MICROSOFT," the baseline model and the greedy method with $\beta = 0.05$ mistakenly interpret that there is a laptop in the image, which might be due to the fact that the word laptop is semantically related to Microsoft. When β is set to 0.10, the greedy approach correctly describe the image. However, when β is set to 0.20, the greedy approach misidentifies the term Microsoft as Microphone and identifies a laptop in the picture. The word in the last image is "ORANGE." The baseline model and the greedy approach for $\beta = 0.05$ and $\beta = 0.10$ utilize the word orange in the caption. They are, however, unable to produce an appropriate caption for the picture. For instance, the baseline model describes the image as "A person wearing a white shirt and orange shoes." which is not correct as the image does not show any shoes.

In conclusion, the models are able to understand the written text on an image, however, sometimes they are not able to correctly relate the written text to the context of the scene.

Table 6.11: Typography skills.

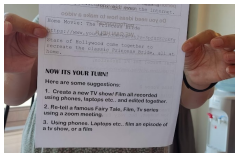
			
Beam search	A woman is holding up her iPhone.	A woman with a sign on her hip with a laptop on her lap.	A person wearing a white shirt and orange shoes.
$\beta = 0.05$ $N = 30$	A woman with apple logo on her face.	A woman with a laptop smiling with a sign on her forehead.	A person smiling with orange on a white turd.
$\beta = 0.10$ $N = 30$	A woman with glasses holding Apple logo on her head.	A woman with a sign that reads "Microsoft" on her wrist.	A person smiling with a orange on her t-shirt.
$\beta = 0.20$ $N = 30$	A woman with glasses holding up a sign.	A woman with a sign that reads "Microphone" on her laptop.	A person that is smiling on a white car.

Adversarial pixel perturbations

CLIP has been shown to be vulnerable to typographical attacks. Contradictory text and picture signals can lead CLIP to choose misleading (visual) alternatives, as Noever et al. demonstrated in [71]. They also concluded that the CLIP model prefers to read first and look afterwards, a phenomena they dubbed "reading isn't believing." We modified

several photographs that had phrases by eliminating the text from the image to see if our image captioning models were affected by this phenomena. In table 6.12, four images are shown with their respective caption generated by the greedy approach. The first example shows a woman holding a piece of paper with some text on it, the greedy approach tries to describe the content of the text. However, when the text is removed from the image, the model describes the image as "A woman holding a white paper towel in her hand." The second image depicts a yacht with the words "Dirty Money" written on it. The greedy approach describes the image as "A boat owned by a criminal is at the bank of a harbor." This is due to the inscription because when that text is removed from the image, the model describes it as "A white boat parked in a dock with a railing." Thus, in both cases the model's reading behavior is to describe the image by reading first and looking afterwards.

Table 6.12: Adversarial pixel perturbations.

				
Greedy Search $\beta = 0.05$ $N = 30$	A woman holding document saying "No" on a form.	A woman holding a white paper towel in her hand.	A boat owned by a criminal is at the bank of a harbor.	A white boat parked in a dock with a railing.

CHAPTER 7
Conclusions

We have seen several techniques to tackle image captioning via the study of state of the art architectures. In addition, the architectures discussed, ranging from the most basic to the most complex, have aided in a deeper grasp of the knowledge gained during the degree. Furthermore, this project has also contributed to the state of the art techniques by introducing novel methods that are able to improve the generated caption by using the CLIPScore metric.

Proposed methods based on the ClipCap architecture outperformed the state of the art models in image captioning in the RefCLIPScore metric. However, as demonstrated in section 6.5.2, those models that combined a CNN with the Transformer architecture did not perform well.

The greedy search based on the CLIPScore metric has shown a number of skills, including real-world knowledge and the capacity to comprehend written text on an image. In the last case, the written text is used to give more specific details of the image when generating a caption. The genetic approach obtained good results regarding the RefCLIPScore metric as well. However, one major drawback with this technique is that, due to the random nature of the genetic algorithm that is employed, the caption generated for an image might vary from iteration to iteration. Finally, the max similarity method behaved similarly to the baseline model, which chose the first caption produced by the beam search strategy.

7.1 Future work

As mentioned above, the CNN + Transformer models did not always perform well. In future work, it might be interesting to find out why this is the case. The first hypothesis is that the CNN model is unable to extract enough features. This might be due to the fact that the CNN models used were trained on ImageNet, where the content of the images is much more general than the content in the MSCOCO dataset. Probably, using CLIP as the image encoder would produce better results as CLIP has been trained with much more specific images.

The ClipCap architecture combined with the proposed methods were found to be the best performing ones. In future work, it might be interesting to explore other image captioning architectures based on pre-trained language models as a larger transformer-based language model, such as GPT-3 [31], is likely to have provided better results.

7.2 Relation with studies

The subjects taken during the degree have provided a solid foundation in the fundamental building blocks of this work. Some of them, on the other hand, are more directly related to this project. Namely:

- **"Técnicas Escalables en Aprendizaje Automático" (TEAA).** Scalable machine learning techniques are the main topics of this course. It focuses on approaches that can handle large datasets because their learning algorithms can be parallelized. This subject was quite helpful when writing Chapter 3 because it covers RNNs, CNNs, and NNs, as well as the numerous challenges associated with Neural Network's training. In addition, the beam search approach described in section 3.7.3 was addressed in the later part of this course.
- **"Optimización" (OPT).** This course covers the fundamentals of data science optimization techniques. This subject has been really useful when describing the several update rules for gradient descent (section 3.5.1). Genetic algorithms, which were employed in one of the proposed methods, are also covered in depth in this subject.
- **"Análisis de Imágenes y Vídeos" (AIV).** This course covers the fundamentals of image processing and image recognition. CNNs were covered in depth during this course which has proven to be really useful when writing section 3.6.2.
- **"Modelos descriptivos y predictivos II" (MDP II).** This course covered the fundamentals of machine learning techniques. The most relevant topics for this project that were covered in this subject are the introduction of linear discriminant functions, the gradient descent technique, and neural networks.
- **"Lenguaje natural y recuperación de la información" (LNR).** This course presents a variety of NLP techniques, with a particular emphasis on text. This course was really beneficial since it covered concepts such as n-grams, word embeddings, and language models. This course covers the majority of the material presented in section 3.7.

Aside from the specific knowledge seen in the aforementioned courses, the degree has strengthened key transversal abilities which have proven to be crucial in the completion of this project. Namely:

- **CT-01 Comprehension and integration.** Throughout this work, the state of the art in image captioning was studied as well as several image captioning models. Some of them were modified and replicated.
- **CT-02 Application and practical thinking.** During this work, several concepts addressed in the degree have been put into practice. For example, the genetic algorithm's implementation was significantly based on the GA theory acquired in the degree.
- **CT-04 Innovation, creativity and entrepreneurship.** In this project, it has been proposed novel modifications of the ClipCap architecture that aimed to improve the caption generated based on the recently introduced CLIPScore metric.
- **CT-10 Awareness of contemporary problems issues.** This project began after noticing various flaws in state of the art image captioning models, such as the Exposure Bias Problem described in section 2.4.

- **CT-11 Life-long learning.** During this project a significant amount of time was spent researching topics that were not seen during the degree. The Transformer architecture, CLIP, and the image captioning metrics employed in this work are some examples.

Bibliography

[1] F. Casacuberta Nolla and R. Paredes, "Chapter 2. multilayer perceptron," in *Artificial Neural Networks*, 2021. Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging - Polytechnic University of Valencia. https://www.prhlt.upv.es/~fcn/Students/rn/t2rna.pdf.

[2] J. Lederer, "Activation functions in artificial neural networks: A systematic overview," *Computing research repository*, vol. abs/2101.09957, 2021.

[3] J. A. Gómez Adrian, "Unit 4 – artificial neural networks recurrent neural networks (rnn)," in *Técnicas Escalables en Aprendizaje Automático (TEAA)*, 2021. Bachelor's Degree in Data Science - Polytechnic University of Valencia.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[5] A. E. Guissous, "Skin lesion classification using deep neural network," *Computing research repository*, vol. abs/1911.07817, 2019.

[6] J. Alammar, "The illustrated transformer." https://jalammar.github.io/illustrated-transformer/. Accessed: 2022-4-29.

[7] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the international conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[8] J. Alammar, "The illustrated GPT-2 (visualizing transformer language models)." https://jalammar.github.io/illustrated-gpt2/. Accessed: 2022-4-29.

[9] P. Platen, "How to generate text: using different decoding methods for language generation with transformers." https://huggingface.co/blog/how-to-generate. Accessed: 2022-4-29.

[10] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the international conference on machine learning*, pp. 2048–2057, PMLR, 2015.

[11] F. A. Galatolo, M. G. C. A. Cimino, and G. Vaglini, "Generating images from caption and vice versa via clip-guided generative latent space search," *Computing research repository*, vol. abs/2102.01645, 2021.

[12] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *Proceedings of the european conference on computer vision*, pp. 382–398, 2016.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

[14] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the second workshop on statistical machine translation*, pp. 228–231, 2007.

[15] C. Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proceedings of workshop on text summarization of the Association for Computational Linguistics*, pp. 74–81, 2004.

[16] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575.

[17] J. Hessel, A. Holtzman, M. Forbes, R. L. Bras, and Y. Choi, "Clipscore: A reference-free evaluation metric for image captioning," *Computing research repository*, vol. abs/2104.08718, 2021.

[18] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

[19] Á. Peris Abril, *Interactivity, Adaptation and Multimodality in Neural Sequence-to-sequence Learning*. PhD thesis, Universitat Politècnica de València, 2020.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] X. Liu, K. Duh, L. Liu, and J. Gao, "Very deep transformers for neural machine translation," *Computing research repository*, vol. abs/2008.07772, 2020.

[22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, pp. 886–893, 2005.

[23] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[24] J. Lu, C. Xiong, D. Parikh, and R. Socher, "Knowing when to look: Adaptive attention via a visual sentinel for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 375–383, 2017.

[25] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6077–6086, 2018.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 248–255, 2009.

[27] G. Li, L. Zhu, P. Liu, and Y. Yang, "Entangled transformer for image captioning," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8928–8937, 2019.

[28] L. Huang, W. Wang, J. Chen, and X.-Y. Wei, "Attention on attention for image captioning," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4634–4643, 2019.

[29] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th annual meeting of the Association for Computational Linguistics*, vol. 1, pp. 2556–2565, 2018.

[30] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the association for the advancement of artificial intelligence conference on artificial intelligence*, 2017.

[31] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[32] R. Mokady, A. Hertz, and A. H. Bermano, "Clipcap: CLIP prefix for image captioning," *Computing research repository*, vol. abs/2111.09734, 2021.

[33] Y. Tewel, Y. Shalev, I. Schwartz, and L. Wolf, "Zero-shot image-to-text generation for visual-semantic arithmetic," *Computing research repository*, vol. abs/2111.14447, 2021.

[34] S. Shen, L. H. Li, H. Tan, M. Bansal, A. Rohrbach, K. Chang, Z. Yao, and K. Keutzer, "How much can CLIP benefit vision-and-language tasks?," *Computing research repository*, vol. abs/2107.06383, 2021.

[35] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[36] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski, "Styleclip: Text-driven manipulation of stylegan imagery," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2085–2094, 2021.

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[38] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the european conference on computer vision*, pp. 740–755, Springer, 2014.

[39] T. Ghandi, H. Pourreza, and H. Mahyar, "Deep learning approaches on image captioning: A review," *Computing research repository*, vol. abs/2201.12944, 2022.

[40] J. Gu, J. Cai, G. Wang, and T. Chen, "Stack-captioning: Coarse-to-fine learning for image captioning," in *Proceedings of the association for the advancement of artificial intelligence conference on artificial intelligence*, vol. 32, 2018.

[41] A. Rohrbach, L. A. Hendricks, K. Burns, T. Darrell, and K. Saenko, "Object hallucination in image captioning," *Computing research repository*, vol. abs/1809.02156, 2018.

[42] H. MacLeod, C. L. Bennett, M. R. Morris, and E. Cutrell, "Understanding blind people's experiences with computer-generated captions of social media images," in *Proceedings of the 2017 CHI conference on human factors in computing systems*, pp. 5988–5999, 2017.

[43] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the international conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computing research repository*, vol. abs/1409.1556, 2014.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[46] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[47] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[48] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *Western electric show and convention record*, vol. 4, p. 96, 1960.

[49] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[50] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[51] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Artificial neural networks: concept learning*, pp. 112–127, 1990.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[53] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[54] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *Ussr computational mathematics and mathematical physics*, vol. 4, no. 5, pp. 1–17, 1964.

[55] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[56] T. Tieleman, G. Hinton, *et al.*, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computing research repository*, vol. abs/1412.6980, 2014.

[58] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[59] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *Computing research repository*, vol. abs/1409.1259, 2014.

[60] Y. Zhang, H. Jiang, Y. Miura, C. D. Manning, and C. P. Langlotz, "Contrastive learning of medical visual representations from paired images and text," *Computing research repository*, 2020.

[61] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, pp. 539–546, 2005.

[62] Z. Wu, "Contrastive representation learning." https://zeqiong-06.github.io/jessie-log/2021/05/31/contrastive-representation-learning.html. Accessed: 2022-2-14.

[63] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *Computing research repository*, vol. abs/2010.11929, 2020.

[64] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg, "Baby talk: Understanding and generating simple image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1601–1608, 2011.

[65] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[66] "Github image-captioning." https://github.com/Dantekk/Image-Captioning, 2021.

[67] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[68] S. Riezler and J. T. Maxwell III, "On some pitfalls in automatic evaluation and significance testing for mt," in *Proceedings of the Association for Computational Linguistics workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 57–64, 2005.

[69] P. Zhang, X. Li, X. Hu, J. Yang, L. Zhang, L. Wang, Y. Choi, and J. Gao, "Vinvl: Revisiting visual representations in vision-language models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5579–5588, 2021.

[70] G. Goh, N. Cammarata, C. Voss, S. Carter, M. Petrov, L. Schubert, A. Radford, and C. Olah, "Multimodal neurons in artificial neural networks," *Distill*, vol. 6, no. 3, p. e30, 2021.

[71] D. A. Noever and S. E. M. Noever, "Reading isn't believing: Adversarial attacks on multi-modal neurons," *Computing research repository*, vol. abs/2103.10480, 2021.

APPENDIX A
Genetic Algorithms

A.1 Introduction

Nature uses a variety of methods that have led in the emergence of new species that are more adapted to their surroundings. The principles that react to species evolution were discovered by Charles Darwin's study in the past century. Natural selection is the first of these rules, which claims that individuals that have higher reproductive success will be more numerous in future generations. The theory of evolution is built on this foundation. Genetic mutation is the second law. It states that organisms' DNA undergoes random mutations. These mutations can be passed down through the generations, resulting in new features that allow organisms to develop and adapt.

Genetic algorithms (GA) attempt to solve a problem by simulating the evolution process. To begin, a population of solutions is generated, and the fittest are chosen to reproduce; after that, a new population is created, consisting of the parents and their children. The procedure is repeated until a 'good' solution is obtained.

Combinatorial optimization problems are notoriously difficult to solve, and the majority of them are NP-hard, which means that finding the best solution in a reasonable amount of time is extremely tough. GA is an effective technique for resolving these problems. When this strategy is used with additional heuristics, the search space and search time can be greatly decreased. The knapsack problem, which entails filling a backpack with items so that the overall value of the items within is maximized while the total weight of the items does not exceed the backpack's capacity, is one of the most popular combinatorial optimization problems that may be solved using GA.

A.2 Chromosome

The representation of a solution using a GA is essential to the success of the algorithm. This means that the solution can be represented as a sequence of symbols, here called a chromosome. The chromosome is evaluated by a function known as the fitness function, which assigns a value to the chromosome based on how good it is. For instance, in the knapsack problem the chromosome may be a binary list of items that the backpack can carry and the fitness function may assign a value according to the total weight or the total value of the items in the backpack.

A.3 Population

The population is a set of chromosomes, each chromosome being a solution to the problem. The population is initialized with a set of random chromosomes. However, if there are some heuristics that are known to work well in the problem, it is useful to initialize the population with some of these heuristics.

A.4 Selection methods

Parent selection consists in choosing a pair of individuals from the population, which are then crossed and mutated. The two individuals are called parents. There are different methods of parent selection, each one with its advantages and disadvantages.

The tournament selection is one of the most used parent selection methods. The tournament selection involves the comparison of k randomly selected individuals (chromosomes). The individual with higher fitness wins the tournament and is selected as a parent. The tournament selection is performed until all the parents are selected.

A.5 Crossover operators

The aim of the crossover operation is to mix the characteristics of two solutions or parents in order to create a new population. This is done in order to introduce some diversity in the population and avoid the stagnation of the results. One of the most common crossover operator is the one-point crossover. This operator consists in selecting a random crossover point in the chromosome and then exchanging the information between the two parents.

A.6 Mutation operators

The mutation operation is performed on a chromosome in order to introduce some variability in the population. One of the most common mutation operators is uniform mutation. This operator selects a chromosome with a certain probability and then changes the value of a random gene of the chromosome.

A.7 Replacement techniques

The replacement technique consists in creating a new population from the parents and the children and then replacing the current population with the new one. There are different replacement techniques. One of the most used replacement techniques is elitism. The elitism technique consists in selecting the best individuals in the current population and placing them in the new population and then replacing the current population with the new one.

A.8 Stop criteria

The stop criteria consist in a condition or a set of conditions which stop the algorithm. There are different stop criteria. One of the most used stop criteria is the maximum

number of generations. This stop criterion consists in stopping the algorithm once the maximum number of generations is reached.

The general scheme of a GA is illustrated in algorithm A.1.

Algorithm A.1 General scheme of a Genetic Algorithm

```
Input: generations, population size
Output: best individual
  n = 0
  population = initial_population(population size)
  while n < generations do
    offspring = empty list
    for i in 1: population size / 2 do
      parent 1, parent 2 = parent_selection(population)
      child 1, child 2 = crossover(parent 1, parent 2)
      child 1, child 2 = mutation(child 1, child 2)
      add child 1 and child 2 to offspring
    end for
    population = replacement(population, offspring)
  end while
  return population.get(0) =0
```

APPENDIX B
Greedy search pseudocode

Algorithm B.1 Greedy search

Input: model, tokenizer, clip_image, clip_model, N, β,
prefix_embed, entry_length, temperature
Output: output_text

```
generated = prefix_embed
tokens = None
for i in 0 : entry_length do
  logits = GPT2(generated)
  logits = logits / temperature
  logits = softmax(logits)
  sorted_logits, sorted_indices = sort(logits)
  if tokens ≠ None then
    Z = zeros(logits.shape)
    for j in 0 : N do
      aux_next_token = sorted_indices[j]
      aux = concatenate(tokens, aux_next_token)
      aux_text = decode(aux)
      clip_text = clip_model(aux_text)
      clip_score = max(cos(clip_image, clip_text), 0)
      Z[aux_next_token] = clip_score
    end for
    logits = β · logits + (1 − β) · Z
  end if
  next_token = argmax(logits)
  next_token_embed = model.embed_token(next_token)
  if tokens = None then
    tokens = next_token
  else
    tokens = concatenate(tokens, next_token)
  end if
  generated = concatenate(generated, next_token_embed)
  if stop_token = next_token then
    break
  end if
end for
output_text = decode(tokens)
return =0
```

APPENDIX C
Sustainable Development Goals

The Sustainable Development Goals (SDGs) are a call for action by all countries to end poverty, protect the planet, and improve the lives and prospects of people around the world. In 2015, all members of the United Nations adopted 17 goals as part of the 2030 Agenda. This agenda establishes a plan to achieve the goal in 15 years. These goals are:

1. No Poverty
2. Zero Hunger
3. Good Health and Well-being
4. Quality Education
5. Gender Equality
6. Clean Water and Sanitation
7. Affordable and Clean Energy
8. Decent Work and Economic Growth
9. Industry, Innovation and Infrastructure
10. Reduced Inequality
11. Sustainable Cities and Communities
12. Responsible Consumption and Production
13. Climate Action
14. Life Below Water
15. Life On Land
16. Peace, Justice, and Strong Institutions
17. Partnerships for the Goals

Specifically, this final degree work is related to the following objectives:

- **SDG 3 - Good Health and Well-being**: Image captioning models can be used to assist blind people in comprehending their surroundings. This sort of model has the potential to improve the quality of life for those who have vision problems. This research proposes several image captioning approaches that might be utilized in the future to improve current models and aid in the development of more robust image captioning models to assist blind people.
- **SDG 8 - Decent Work and Economic Growth**: Image captioning models are increasingly used by companies such as Facebook or Google for the automatic generation of descriptions from images for use in social media, search results or product labels. By using this models, resources can be saved as the need for manual data annotation is eliminated and the efficiency of the company can be increased. Moreover, the development of new models can create new job positions in the field of artificial intelligence.
- **SDG 9 - Industry, Innovation and Infrastructure**: The image captioning field is continuously growing, as new methods are being developed to improve accuracy and efficiency. The methods proposed in this work provide a promising approach to tackling the task of image captioning.