UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

Boosting Image Captioning Using ConvNeXt Deep Neural Networks

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingenieria de Tecnologías de la Información y la Comunicación

Autor:

Ramos Granda Leo Thomas

Tutor:

Morocho Cayamcela Manuel Eugenio, Ph.D.

Urcuquí, Junio 2023

Autoría

Yo, **Leo Thomas Ramos Granda**, con cédula de identidad 1805311923, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Asimismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Junio 2023.

LEO
THOMAS
RAMOS
GRANDA

Digitally signed
by LEO THOMAS
RAMOS GRANDA
Date: 2023.06.23
13:38:36 -05'00'

Leo Thomas Ramos Granda
CI: 1805311923

Autorización de publicación

Yo, **Leo Thomas Ramos Granda**, con cédula de identidad 1805311923, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, Junio 2023.

LEO
THOMAS
RAMOS
GRANDA

Digitally signed
by LEO THOMAS
RAMOS
GRANDA
Date: 2023.06.23
13:38:54 -05'00'

Leo Thomas Ramos Granda

CI: 1805311923

Dedication

To my beloved parents, who have been my greatest inspiration and support throughout my life. I dedicate this work to you with all my heart, as a tribute to everything you have done for me.

To my dear teachers, who have been my guides on the path of knowledge. I dedicate this work to you as a demonstration of my gratitude and respect for everything you have taught me.

To my friends, who have been my emotional support in difficult times and my companions of joy in good times. I dedicate this work to you as a testimony of our friendship and the value of your presence in my life.

Leo Thomas Ramos

Acknowledgment

To my beloved family, thank you for always being by my side, for your love, patience, and understanding throughout this entire process. I cannot find the right words to express all the love and gratitude I feel towards you. From the beginning, you have been by my side, giving me your love, support, and encouragement, no matter the obstacles that came our way. You have been my rock and my refuge in moments of uncertainty, and my source of inspiration in moments of challenge.

Thank you for believing in me and for being my constant motivation to push my limits and achieve my goals. This thesis is just a small reflection of everything you have done for me, and it would not be possible without your unconditional love and support.

Verónica, José, Roberto, my family, your dedication and effort have been a constant inspiration to me. Thank you for teaching me the value of hard work, perseverance, and commitment. Your love and example have driven me to be the best version of myself. Every achievement I have made is because of and for you.

To my friends, who have been by my side through thick and thin, sharing laughs, tears, and dreams. Thank you for your words of encouragement, your company, and your unwavering friendship. I could not have made it this far without your support. I extend a special thanks to Mike, my brother and study buddy, partner in adventures and craziness. Also, thank you to Dánely, Vincent, Ariana, Nicolás, Jean, and Andrés, wonderful people I had the opportunity to meet and learn from.

To my dear professors, who have been my guide and inspiration throughout my entire academic career. Thank you for your wisdom, patience, and dedication to teaching us everything you know. I feel fortunate to have had the opportunity to learn from such brilliant individuals as yourselves. This work is a small tribute to your immense legacy.

A special thanks to my advisor Manuel Eugenio Morocho, and professor Fracklin Rivas for their tremendous support and inspiration in the completion of this work. And thanks to Edmundo Casas and Cristian Romero for their important collaboration.

Resumen

Este trabajo propone un modelo basado en ConvNeXt para generar subtítulos de imágenes. Específicamente, se integra el modelo convolucional ConvNeXt, una arquitectura de visión por computadora de última generación, con una red de memoria a corto y largo plazo que incluye un módulo de atención visual. Se realizaron diversos experimentos para evaluar la viabilidad de ConvNeXt en esta tarea. En primer lugar, se estudió el impacto de usar cuatro versiones de ConvNeXt para la extracción de características. Además, se probaron dos tasas de aprendizaje diferentes durante la etapa de entrenamiento del codificador para analizar el impacto de esto en el rendimiento. Asimismo, se analizó el efecto de la inclusión y exclusión de teacher-forcing en el decodificador durante el entrenamiento. Se utilizó el conjunto de datos MS COCO 2014, y se adoptaron la pérdida, top-5 accuracy y BLEU-n como métricas de rendimiento. Los resultados muestran que nuestro modelo propuesto supera el modelo de referencia en un 43.04% y un 39.04% para los modelos de atención suave y atención dura, respectivamente, en términos de BLEU-4. Nuestro modelo también supera en un 4.57% y un 0.93% a los enfoques equivalentes basados en transformador de visión y transformador de imagen con eficiencia de datos, respectivamente, en términos de BLEU-4. Además, nuestro modelo mejoró a alternativas que utilizan codificadores basados en arquitecturas ResNet-101, ResNet-152, VGG-16, ResNeXt-101, y MobileNet V3, en un 6.44%, 6.46%, 6.47%, 6.39% y 6.68%, respectivamente, en términos de precisión en top-5 accuracy, y en un 18.46%, 18.44%, 18.46%, 18.24% y 18.72%, respectivamente, en términos de pérdida.

Palabras Clave:
Subtítulado de imágenes, ConvNeXt, Redes neuronales artificiales, Visión computacional, Procesamiento del lenguaje natural, Aprendizaje profundo

Abstract

This work proposes a ConvNeXt backbone-based model for image captioning. Specifically, the ConvNeXt convolutional model, a state-of-the-art computer vision architecture, is integrated with a long short-term memory network enclosing a visual attention module. Diverse experiments were performed to evaluate the feasibility of ConvNeXt in this task. First, the impact of using four versions of ConvNeXt for feature extraction was studied. Additionally, two different learning rates were tested during the training stage of the encoder to analyze the impact of this on performance. Furthermore, the effect of inclusion and exclusion of teacher-forcing at the decoder during training was analyzed. The 2014 MS COCO dataset was used, and the loss, top-5 accuracy, and BLEU-n were adopted as performance metrics. The results show that our proposed model outperforms the benchmark by 43.04% and 39.04% for soft-attention and hard-attention models, respectively, in terms of BLEU-4. Our model also surpasses equivalent approaches based upon vision transformers and data-efficient image transformers by 4.57% and 0.93%, respectively, in terms of BLEU-4. Moreover, it outperforms alternatives that use ResNet-101, ResNet-152, VGG-16, ResNeXt-101, and MobileNet V3 network-based encoders, by 6.44%, 6.46%, 6.47%, 6.39%, and 6.68%, respectively, in terms of top-5 accuracy, and by 18.46%, 18.44%, 18.46%, 18.24%, and 18.72%, respectively, in terms of loss.

Keywords:
Image captioning, ConvNeXt, Artificial neural networks, Computer vision, Natural language processing, Deep learning

Contents

Dedication v

Acknowledgment vii

Resumen ix

Abstract xi

Contents xiii

List of Tables xv

List of Figures xvii

1 Introduction 1

1.1 Background 1

1.2 Problem Statement 3

1.3 Objectives 4

1.3.1 General objective 4

1.3.2 Specific objectives 4

2 Theoretical Framework 5

2.1 Artificial Intelligence 5

2.1.1 Machine learning 5

2.1.2 Deep learning 8

2.2 Artificial neural networks 8

2.2.1 Recurrent neural networks 12

2.2.2 Convolutional neural networks 14

2.2.3 Transformer networks 17

3 State of the Art 23

3.1 Image captioning 23

3.2 ConvNeXt 26

4 Methodology 29

4.1 Data Set Description and Preprocessing 29

4.2 Model Proposal 30

4.2.1 Benchmark model 30

4.2.2 ConvNeXt-based approach 32

4.3 Experiments 34

4.3.1 Performance Metrics 36

5 Results and Discussion 39

5.1 Limitations 47

6 Conclusions 51

Bibliography 53

List of Tables

2.1 Types of machine learning. 7

3.1 Summary of image captioning related works. 25
3.2 Summary of ConvNeXt related works. 28

4.1 Example of how captions are encoded 31
4.2 ConvNeXt variants used in this work. 34
4.3 Final encoding size produced by each of the ConvNeXt variants used. . . . 34

5.1 BLEU metrics obtained using a learning rate of $4e - 4$. 40
5.2 BLEU metrics obtained using a learning rate of $1e - 4$. 40
5.3 Average inference time (seconds) 45
5.4 Average loss 45
5.5 Top-5 accuracy 46
5.6 Comparison of our results according to the top-5 accuracy and loss metrics 46
5.7 Comparison of our results according to the BLEU metrics 47

List of Figures

2.1 Relationship between artificial intelligence, machine learning, and deep learning. 6
2.2 ML model workflow. 6
2.3 Machine learning and Deep learning comparison. 8
2.4 The perceptron. 9
2.5 The multi-layer-perceptron. 10
2.6 The gradient descent process. 11
2.7 The Long short-term memory architecture. 13
2.8 LeNet-5 architecture. 15
2.9 Comparison of Swin Transformer, ResNet, and ConvNeXt block structures 17
2.10 The Scaled Dot-Product block. 18
2.11 The Multi-head Attention block. 19
2.12 The transformer architecture. 20

4.1 Examples taken from the MS COCO dataset 30
4.2 Encoder-decoder architecture for image captioning used as benchmark. . . 31
4.3 Proposed ConvNeXt backbone-based architecture 33
4.4 Overall workflow followed in this study. 36

5.1 Visual results of image captions using a learning rate of $4e-4$. 41
5.2 Visual results of image captions using a learning rate of $1e-4$. 42
5.3 Visual results of image captions using a learning rate of $4e-4$. 43
5.4 Visual results of image captions using a learning rate of $1e-4$. 44
5.5 Image captioning results using the best model 49

Chapter 1

Introduction

1.1 Background

Image captioning is the task of generating a natural language description of the content within an image. It is a field that combines both, computer vision and natural language processing [1]. Specifically, image captioning involves image understanding by generating a description for an image [2]. The first stage involves object detection and recognition; understanding the type of scene, the properties of objects, and the interactions between them. The latter comprises the generation of syntactically and semantically well-formed sentences [3].

The range of image captioning applications is broad and it is used in wide variety of areas, including education, digital libraries, biomedicine, military, commerce, and web search [2]. For instance, image captioning-based assistive technology can help visually impaired people to understand their environment [4]. Image captioning can also be applied in search engines and social networking platforms for content-based image retrieval [2, 4, 5]. It is also helpful in generating descriptions for augmented reality images, virtual reality experiences, and video games [6]. Image captioning can also be utilized in sophisticated recommendation systems and visual assistance systems [5].

Image captioning has been a challenging task in the field of computer vision with significant difficulties and challenges [7]. For example, image captioning systems must deal with a wide variety of images, from simple scenes with few elements to complex scenes with a large amount of detail [2, 4]. In addition, objects in the scene may vary in shape, size, and appearance, as well as include noise, blur, or occlusion [8]. Also, these systems often have to deal with new or rare objects for which they were not trained [4]. This makes it difficult to accurately identify the scene or objects in the image and their relationships to each other. Moreover, an image may contain multiple scenes, making it even more difficult to describe accurately [9].

Template-based and retrieval-based techniques have been considered early approaches for image captioning [5, 10]. The former methods use fixed-dimension empty-space templates to generate captions [2]. These methods first use classifiers to recognize the objects, attributes, and relationships in an image, and then fill in the spaces in the template [4]. These methods can generate grammatically correct captions; however, they cannot generate variable-length description captions, nor do they correctly express the visual context of the image [11]. The latter find visually similar images and their captions from the training dataset [2, 12]. The caption for the query image is selected from this set of captions, called candidate captions. Although these methods produce grammatically correct captions, they cannot generate image-specific captions because the results are significantly general [4].

With the advancement of technology, superior systems that are able to handle the complexities and challenges of image captioning have been proposed. Currently, most methods for image captioning use deep learning, specifically artificial neural networks [2]. Although there are different approaches based on neural networks for image captioning, the encoder-decoder architecture is the most commonly used. This approach allows the task to be divided into two parts; the encoding phase, in which visual features are extracted from the input image, and the decoding phase, which creates the output concerning the encoded or extracted features [5]. Commonly, a convolutional neural network (CNN) is in charge of feature extraction, and a recurrent neural network (RNN) converts this information into words and phrases that produce the image's caption [1, 4].

However, classical encoder-decoder architectures cannot analyze the image over time while generating the image descriptions. Moreover, they generate the caption considering the scene as a whole without the image's relevant spatial aspects [2]. Therefore, attentional mechanisms have emerged as a tool to overcome these limitations. The main difference between methods based on attentional mechanisms and other schemes, is that the former can focus on the salient parts of the input image while generating the output sequences simultaneously [1, 4, 7]. Thereby, the model can focus on the important regions of the image at each timestep and dynamically updates the caption until the final result is obtained [2, 13]. This allows the generation of captions that are more accurate and closer to those generated by a human.

1.2 Problem Statement

Modern technology has allowed us to have better techniques to tackle different computer vision tasks. We moved, for example, from the use of CNNs to the new transformers. These new architectures have become a standard in natural language processing and computer vision tasks [14]. They have gradually replaced CNNs and pushed them into the background [15], even though one of their disadvantages is their high time and space complexity, which can increase training time.

Although neural networks have enabled image captioning systems to achieve impressive results, generating accurate descriptions is still a challenge [16]. Unlike other tasks, such as image classification, the field of image captioning still has room for improvement, particularly when the task involves challenging conditions. Furthermore, image captioning systems are bound to become more efficient and effective due to their wide range of applications. For all these reasons, the study and development of new image captioning systems and algorithms is a field that has attracted much attention and remains an exciting challenge for the research community [5]

In this work, we propose a system for image captioning using a new architecture called ConvNeXt for feature extraction. In summary, ConvNeXt is a modernization of CNNs, whose redesign was inspired by the transformer architecture. So far, this architecture has proven to achieve similar or even superior performance to the transformer while maintaining the simplicity and efficiency of standard CNNs. Overall, our ConvNeXt feature extractor combines with a long short-term memory (LSTM) with attentional mechanisms to generate the image descriptions.

In addition, to evaluate the feasibility of ConvNeXt in diverse contexts, we have conducted different experiments that include using four variants of ConvNeXt, from lightweight to robust models to evaluate the impact on the performance of caption generation. We have also evaluated the use of teacher-forcing and the effect on the system's performance. Finally, we have tested the impact of two different learning rates during the training stage. To evaluate our proposal, we have used the top-5 accuracy and BLEU-n metrics, proving that our system is capable of generating accurate descriptions improving the state-of-the-art.

1.3 Objectives

1.3.1 General objective

The general objective of this work is to improve the performance achieved by the architecture of the paper *Show, Attend and Tell* [17], by modifying the encoder with a state-of-the-art approach and varying some hyperparameters.

1.3.2 Specific objectives

- Modify the original architecture by replacing its encoder with a ConvNeXt backbone.
- Use different versions of ConvNeXt and compare the performance obtained by each of them.
- Use different learning rates to train the proposed architecture and compare the performance obtained using each.
- Analyze the effect of applying teacher forcing during architecture training and compare the performance when this is not applied.

Chapter 2

Theoretical Framework

2.1 Artificial Intelligence

Artificial intelligence (AI) refers to systems capable of performing problem-solving tasks by imitating human intelligence [18]. This concept term was first mentioned in 1950 by Alan Turing in his book *Computing Machinery and Intelligence* [19, 20]. In that book, he asked, "can machines think?" and introduced the Turing Test method to determine whether a machine can be considered intelligent from the point of view of imitation [19]. Then, in 1956, John MacCarthy officially established the term *artificial intelligence* as *"the science and engineering of making intelligent machines, especially intelligent computer programs"* [19, 20].

Over the years, AI algorithms have evolved from simple sets of "if-then" rules to today's complex algorithms [19]. The development of AI has brought great advances in the industry as it allows solving tasks that typically require human intelligence [21, 20]. As a result, there have been economic and social benefits that have improved people's lifestyle [20].

2.1.1 Machine learning

Machine Learning (ML) is a subdivision of AI (see Fig. 2.1) that focuses on solving tasks and learn relationships and patterns using examples and observations from sample data [20, 21, 22]. In order to build these predictive models, a process of selection or generation of a set of convenient features is performed; this procedure is known as feature engineering [21]. Once the model identifies the patterns or features that describe a particular problem, it "learns" to apply these relationships in future similar scenarios [19]. Not all problems involving ML are approached in the same way; however, there are some essential and generalizable steps that ensure the proper functioning of ML models [23]. These are shown in Fig. 2.2.

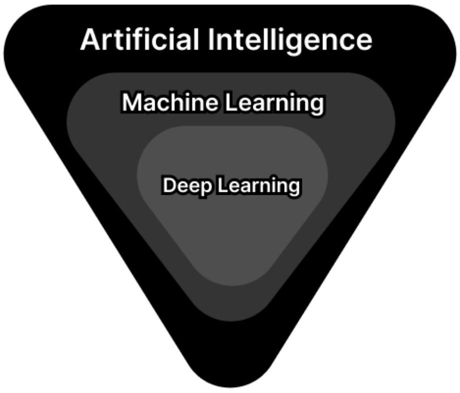


Figure 2.1: Relationship between artificial intelligence, machine learning, and deep learning.

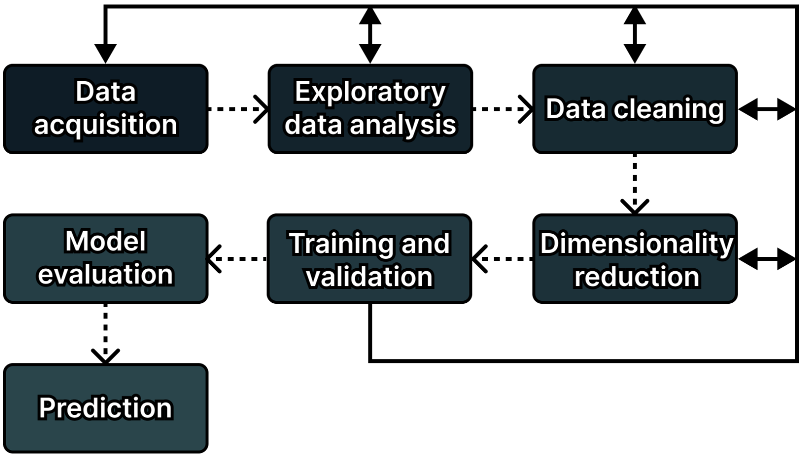


Figure 2.2: ML model workflow.

ML can be classified into four types according to the problem and the available data: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [20, 22, 24, 25, 26, 27], as shown in Table 2.1.

Type	Description
Supervised learning	As the name implies, these techniques require supervision [23]. This means that in order to train a model, labeled data is used [23, 24]. In other words, there is an output associated with an input, both labeled. This process is known as calibration since the model maps an input x to an output y [22]. Once trained, the model can predict a target from new or previously unseen data [22, 25].
Unsupervised learning	Unlike supervised learning, the data used to train unsupervised models is not labeled [26, 27]. That is, there are no outputs associated with inputs. In this case, the model is in charge of finding some relationship, characteristic, or pattern among the input data [23]. An example of unsupervised learning is clustering, a technique that groups elements with common properties and is applied in different tasks such as customer segmentation [22, 26].
Semi-supervised learning	Semi-supervised learning can be seen as a mix of supervised and unsupervised learning [23, 26]. The data used for these models is mixed; that is, it contains both labeled and unlabeled data [26, 28]. Generally, the models learn from the unlabeled data and use what is learned to solve supervised tasks [20]. One of the uses of this type of models is in medical image processing [26].
Reinforcement learning	This technique is based on a system of rewards and penalties [22]. Here an agent interacts with its environment through trial-and-error actions that maximize the rewards for each correct decision [22, 29]. In other words, the agent receives a reward each time it performs a correct action and a penalty when it fails to do so. In this way, the model learns by itself to make the most appropriate decisions to perform a task without a set of prior cues successfully [22, 26].

Table 2.1: Types of machine learning.

ML techniques have been enhanced with the technological development of computing and the increase in existing data volume [23, 30]. Some of the tasks in which ML has special applicability are classification, regression, and clustering [22, 23]; for this reason, it is widely used in areas such as fraud detection, customer segmentation, and credit scoring [22].

2.1.2 Deep learning

As illustrated in Fig. 2.1, Deep learning (DL) is another subset of AI [21, 29], which is within ML. Unlike ML, DL uses more complex and sophisticated methods based on deep neural networks [29]. The main difference between ML and DL lies in feature engineering. ML requires to indicate to the algorithm the features which it must look at, which needs expertise and is a time-consuming procedure. In contrast, DL algorithms extract the features themselves and search for the optimal set of these that allow them to obtain the most accurate output possible [29, 31], as seen in Fig. 2.3. In other words, DL deals with unstructured data and "learns" from it. For this reason, DL is considered the closest technique to human learning [19, 31].

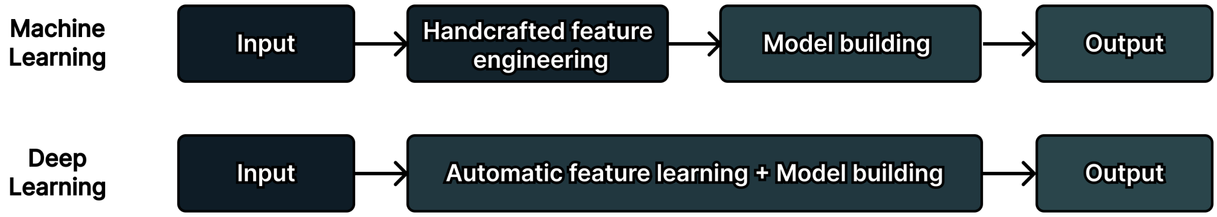

Machine Learning
Input
Handcrafted feature engineering
Model building
Output
Deep Learning
Input
Automatic feature learning + Model building
Output

Figure 2.3: Machine learning and Deep learning comparison.

2.2 Artificial neural networks

In 1957, Frank Rosenblatt, known as the "father of deep learning," created the *perceptron*, the smallest computational unit of DL [25]. The *perceptron*, also called artificial neuron, is inspired by the biological neuron [26]; it is shown in Fig. 2.4. The output of the *perceptron* y can be mathematically defined as:

Mathematically it is defined as:

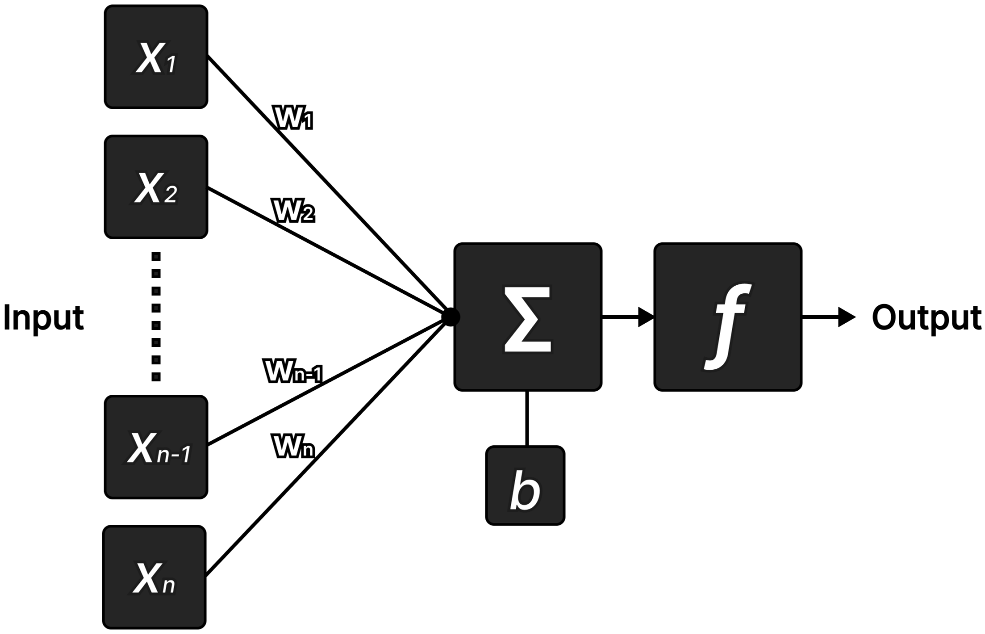


Figure 2.4: The perceptron.

$$y = f(\sum_{i=1}^{D} w_i x_i + b) \tag{2.1}$$

where D is the dimension of input space, x is the input vector, w is a set of weights corresponding to the input vector, b is the bias, and $f(\cdot)$ is a function called "activation function". It works as follows: first, each input value is multiplied by a corresponding weight; then, the sum of the resulting products is calculated, and the bias is added. Subsequently, this value is passed to the activation function. This function generates a binary value, 0 or 1, as a binary classifier [25, 26, 30].

Although the *perceptron* marked the beginning of the revolution in AI-based technologies [32], it could not solve nonlinearly separable problems since the core of a *perceptron* is, basically, a linear regression [25]. The solution to this problem was to combine multiple perceptrons in parallel, forming layers as in Fig. 2.5. This grouping was called *multi-layer-perceptron* (MLP) and is the most basic artificial neural network (ANN) [23, 25, 30]. The functionality of the MLP is based on that of the *perceptron*, with the difference that the outputs of the neurons in one layer are the inputs of the neurons in the next [30].

After this, the *backpropagation* (BP) algorithm came to solve the biggest problem of MLP. BP is an optimization algorithm that uses the chain rule to calculate how slightly changing each weight would change the network error [33]. Its name is since the algorithm

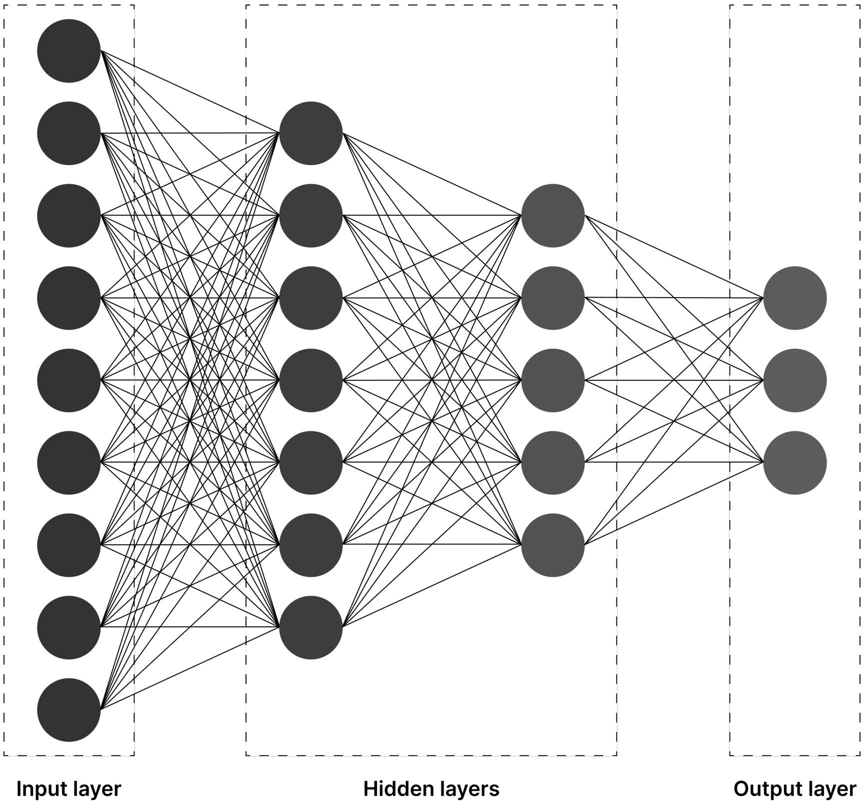


Figure 2.5: The multi-layer-perceptron.

first calculates the error value of each neuron in the last layer and propagates it backward [25, 33]. In this way, the global error is minimized. Before BP, MLPs could not automatically derive network weights and biases; in other words, the network could not be trained [25, 30]. The error function in classic *backpropagation* is the *sum of squared error* (MSE) over all instances n [25], which is defined as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{2.2}$$

where y_i is the target value and $\hat{y}_i$ is the predicted output of the network on input x_i.

Variations of the BP algorithm have appeared over the years. One of them is the widely used *gradient descent* algorithm [25]. This algorithm allows finding the minimum of a function automatically. For this, it uses the gradient (or derivative) of the function, which guides the algorithm to approach the minimum ideal of said function progressively [34, 35].

Mathematically it is defined as in Equation 2.3, where θ_{t+1} is the parameter that is updated by subtracting its current value θ_t and the product of the *learning rate* α and the gradient (∇f). We use the gradient to know which way to "move." If the gradient is negative, we will go in the current direction, and if the gradient is positive, we will go in the opposite direction [35, 36]. Thus, iteratively, we will arrive at a point where the gradient is close to zero, which will indicate that we are at a minimum value [36], as in Fig. 2.6.

$$\theta_{t+1} = \theta_t - \alpha \nabla f \tag{2.3}$$

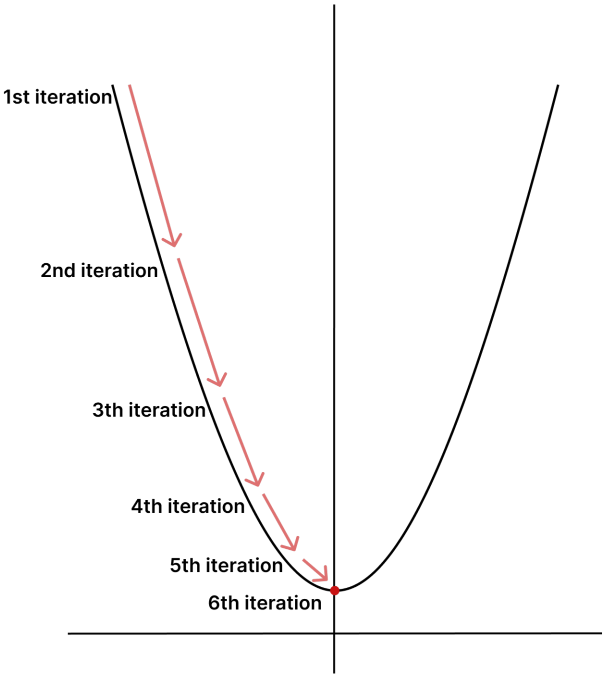


Figure 2.6: The gradient descent process.

The *learning rate* is a parameter that defines the proportion in which we advance in each iteration of the descending gradient algorithm [35, 36]. It is important to define this parameter correctly. On one hand, if we choose a *learning rate* that is too small, the algorithm will be inefficient, and it will take many iterations to reach a minimum. On the other hand, if we assign a large value, the "steps" of the algorithm will be very large, and it will be impossible for it to converge to a minimum [28, 36]. Fortunately, there are different techniques to adjust this parameter, such as Stochastic gradient descent (SGD), Momentum, Nesterov accelerated gradient (NAG), among others [34, 35, 36].

Due to its excellent results, the descending gradient algorithm is the most used to train almost any architecture, from the classic multi-layer networks to the most advanced transformer networks.

2.2.1 Recurrent neural networks

Classical neural networks allow the processing of only one piece of data at a time, but if the input is a sequence of data, such as a dialog, this type of architecture will not be able to process this type of sequence [30]. The *recurrent neural network* (RNN) is a neural network specially designed to work with data whose elements have a temporal relationship, such as time series or ordered data sequences [22, 30, 37]. In RNNs, information can be passed to the nodes of the following layer or the nodes of a previous layer. This property gives RNNs a kind of artificial memory that allows previous inputs to influence the current output [26].

Since NLP is word order dependent, language processing requires memory about previous items. For example, to analyze a sentence, interpreting a particular word requires knowledge of the surrounding words to obtain the correct meaning of the whole sequence [37]. A common example of this idea is the auto-correct text generation of cell phones. These programs generate the next word according to the context provided by the user's previously typed words.

RNNs respond to a very intuitive idea, given that their functioning is similar to that of a human. For example, when a person reads, he processes words sequentially, relying on the context of all the previous words to give a concrete meaning to the information. For these reasons, RNNs were widely used to address various tasks in this field [30, 37].

However, RNNs present a problem: they suffer from loss of context if the input to be processed is large [22]. The more words a recurrent neural network processes, the more the first words lose importance within the context of the sentence concerning the last ones being processed [22, 37]. In other words, the network forgets which were the first words in a sequence. This, of course, leads to serious errors of interpretation and understanding. Fortunately, over the years, more sophisticated architectures have been developed that allow us to deal with this problem better [37].

Long short term memory

Although RNNs allow processing sequences of data, they have the disadvantage of lacking long-term memory. This significantly limits the usefulness of RNNs, since they are generally intended to process large sequences of data.

One of the best-known architectures and one of the first that tried to control this problem is the *Long short-term memory* (LSTM). This network can "store" sequence-relevant data and preserve it in both the short and long term [37, 38]. Unlike a classic RNN, it contains an additional input and output, called "cell state" [30], as seen in Fig. 2.7, which allows it to add or remove information from the neural network's memory [22, 30]. This cell state works like a conveyor belt. Then, the "forget" gate allows to decide which information to discard; the "update" gate allows to update the state cell by adding or changing information; finally, the "output" gate filters the contents of the cell state, and the output is obtained [30, 37, 38].

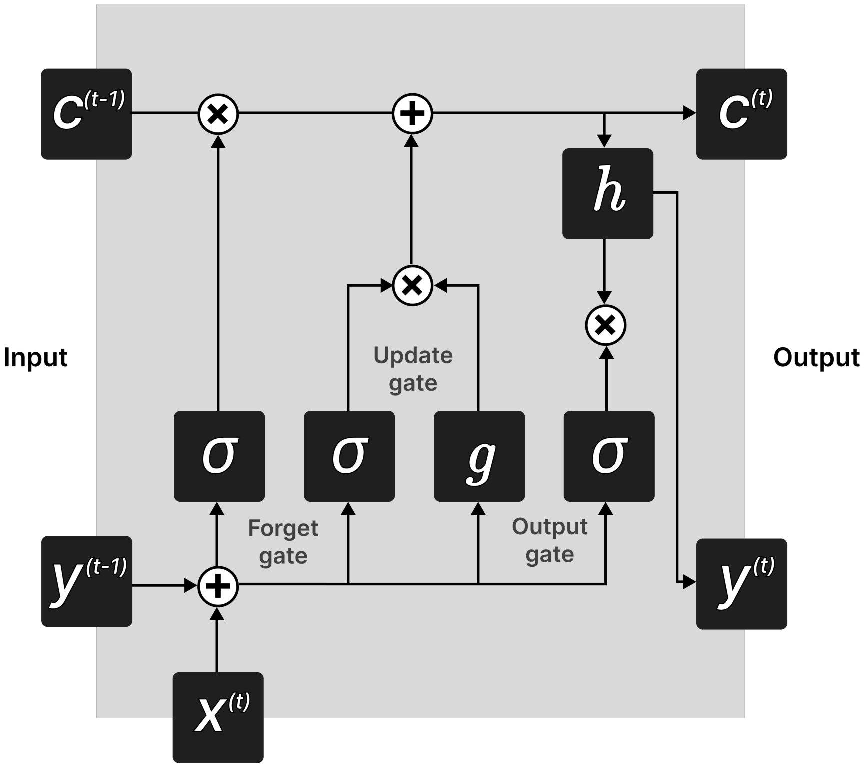


Figure 2.7: The Long short-term memory architecture. σ: sigmoid activation function; h: tanh activation function; g: tanh activation function; $\bigoplus$: sum over all inputs; $\bigotimes$: multiplication; x^t: input vector; y^{t-1}: previous hidden state; y^t: current hidden state; c^{t-1}: previous cell state; c^t: current cell state.

2.2.2 Convolutional neural networks

Although classical multilayer neural networks have been widely used in image processing, they have some disadvantages. Each input in a classical ANN corresponds to a pixel; in other words, the model input is a one-dimensional vector. This means that the spatial context of the image features is lost. That is, the possible relationships between the elements or pixels of an image are lost. For example, pixels close to each other in an image are likely to be more correlated than pixels on opposite sides of the image, but this is not taken into account in a feed-forward neural network [26].

Convolutional neural networks (CNN) are a special class of ANNs that specialize in working with images as input data. They comprise a series of processes that allow the extraction and learning of features from an image [22]. In this way, the aim is to generate a data matrix whose values contain the relevant information, taking into account the spatial distribution of the data.

Feature extraction begins with a process called **convolution**. This process consists of mathematically operating a group of image pixels with a convolution matrix called a **kernel**. The operation performed is the scalar product. This kernel operates through the image from left to right and from top to bottom until a new data matrix is generated [28]. For feature extraction, not one but several kernels are applied; the set of kernels is known as **filters**. Each of them runs through the image and generates an output matrix. The set of output matrices generated by the filters is known as a **feature map** [26].

Subsequently, an operation known as **pooling** is performed. This is used to reduce the dimensionality of the data, taking only the most representative pixels of the matrix. Two types of pooling are the most commonly used, **average pooling** and **max pooling** [31]. As the name suggests, average pooling calculates the average of the values of a feature map. On the other hand, max pooling selects only the highest value of all. In this way, we reduce the dimension of the data and keep only the most relevant.

This convolution and pooling processes are repeated several times until the most complex features of the image can be extracted. Once this is done, it is possible to **flatten** the data to introduce it into a classical feed-forward network and generate the corresponding prediction [22]. Fig. 2.8 shows the architecture of the LeNet-5 model, considered the first convolutional neural network created.

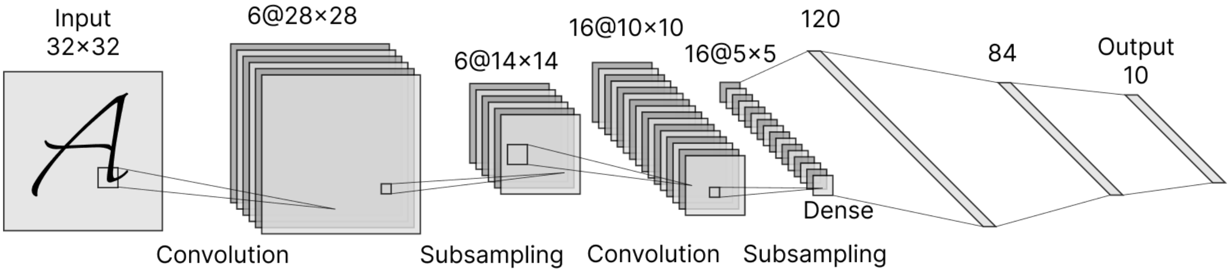


Figure 2.8: LeNet-5 architecture. Subsampling is another term for pooling.

ConvNeXt arquitecture

The ConvNeXt architecture emerges as an alternative to the, for the time being, state-of-the-art transformers. It was developed by researchers at Facebook AI Research (FAIR) in 2022. The idea of this development focuses on 'modernizing' CNNs. The researchers used a ResNet-50 architecture as a basis and made changes to its design and configuration.

First, they changed the classical way CNNs were trained and trained the ResNet with techniques similar to those used to train vision transformers [39]. This first change already yielded significant results since this procedure improved the performance of the original ResNet-50. Subsequently, they applied several design modifications divided into five aspects:

1. Macro design
2. ResNeXt
3. Inverted bottleneck
4. Large kernel size
5. Layer-wise micro designs

For the macro design stage, they analyzed the design of the Swin Transformers. Specifically, they analyzed and applied the multi-stage design, in which each stage has a different feature map resolution. For this, they made changes in the stage computation ratio and the stem cell structure.

Then, they tried to adopt the idea of ResNeXt [40], which has a better FLOPs/accuracy ratio than a conventional ResNet. To do this, they use the concept of grouped convolution, in which the convolutional filters are separated into different groups. In other words, the aim is to ”use more groups, expand the width,” which significantly reduces FLOPs.

After, they replicated the use of a transformer’s inverted bottleneck. This makes the hidden dimension of the MLP block four times larger than the input dimension. This change reduces the FLOPs of the entire network and produces a slight improvement in its performance.

The next part consisted of analyzing the performance of large convolutional kernels. Basically, they increased the kernel size, motivated by transformers. They used as an example the Swin Transformers, whose kernel size is 7x7, which is larger than the 3x3 kernel of the ResNetXt. This resulted in a performance improvement without affecting the FLOPs of the network too much. However, they found that starting with a 7x7 kernel, increasing the kernel size does not generate significant improvements.

After this, they focused on a micro-scale analysis, most of which was at the last layer’s level. Specifically, they analyzed different configurations of activation functions and normalization layers. The most significant changes were: replacing the activation function ReLU with GELU, reducing the number of activation functions, reducing the number of normalization layers, replacing batch normalization with layer normalization, and separating the downsampling layers. Fig. 2.9 shows the difference between the ConvNeXt block designs and the reference models.

After this, they obtained what they called ConvNeXt. After being trained and evaluated, ConvNeXt showed similar and, in some cases, better performance in image classification, object detection, and segmentation tasks than state-of-the-art models such as Swin Transformers. The full explanation of the design, operation, and training of the ConvNeXt can be read in the original paper in [41].

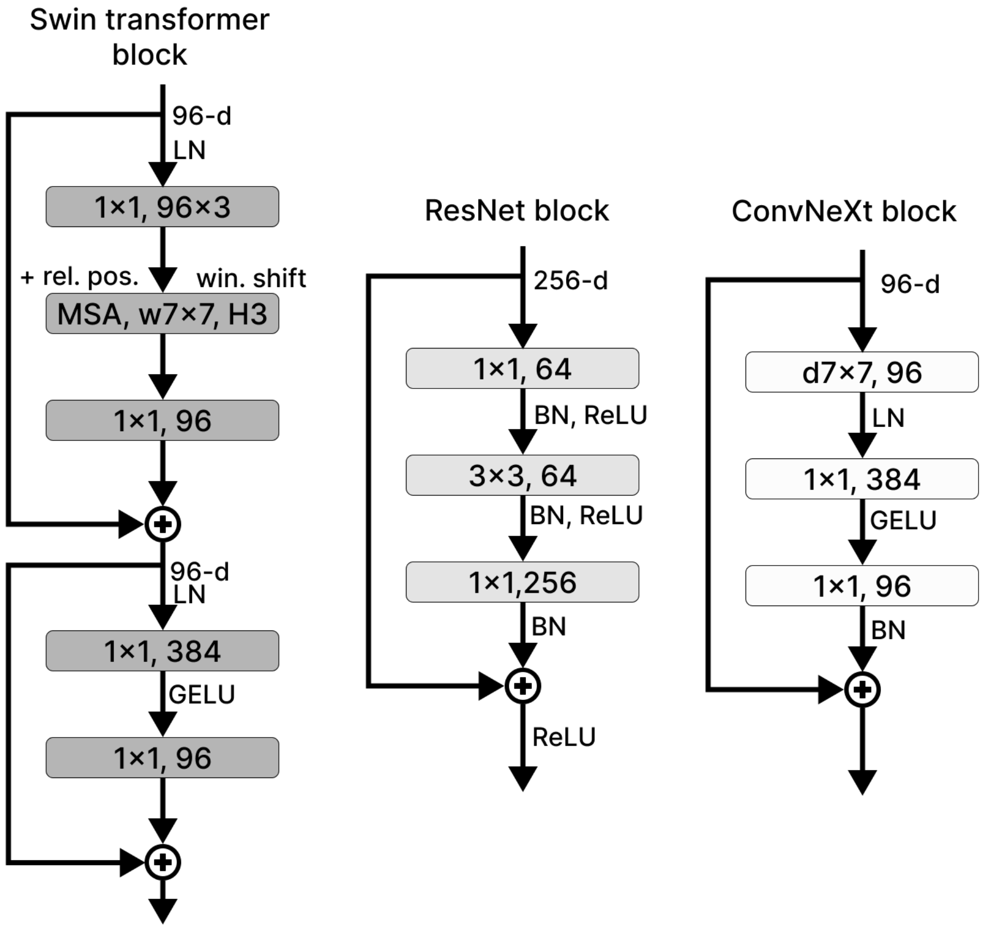


Figure 2.9: Comparison of Swin Transformer, ResNet, and ConvNeXt block structures

2.2.3 Transformer networks

Self-attention

The big problem of RNNs lies in their lack of long-term memory [22]. When the sequences are long, the first processed words become unimportant in the sequence context, and the network starts to fail [37]. Even networks such as LSTM are not robust enough to support the large volume of data and the complexity of tasks demanded by today's world. A mechanism that pays "attention" to a complete sequence was proposed to address the limitations of RNNs [37]. This is known as *self-attention*.

Self-attention is a mechanism that pays "attention" to all words in a sequence [42, 43]. In this mechanism, first, the input is transformed into three new vectors: the query vector q, the key vector k, and the value vector v [43, 44]. The query vector and key refer to the key-lock principle, similar to that found in databases for retrieving information. The vector q can be understood as the set of characteristics or representative properties of a word, the vector k represents the properties that the word looks for in another word, and the v is the vector value of the word. In this way, it is possible to analyze whether a word has has a stronger or weaker relationship with another word [44].

In order to know the strength of this relationship, the q vector of one word must be multiplied by the k vector of another, using the dot product [43]. The higher the value of this operation, the greater the compatibility between the words. Then, this attention vector serves as a mixing factor for each v vector; these vectors are summed, and an output vector is obtained. This vector contains information about all the words that make up a sequence, giving greater importance to those to which more "attention" has been paid [43, 44, 45]. This approach is known as Scaled Dot-Product Attention and is illustrated in Fig. 2.10.

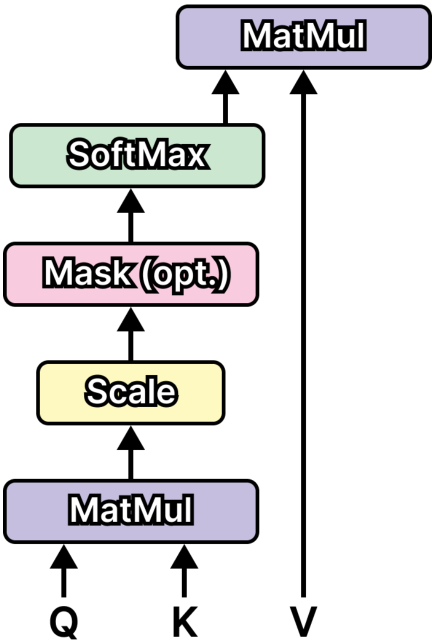


Figure 2.10: The Scaled Dot-Product block.

Calculations are commonly performed by grouping the vectors derived from the different inputs into matrices: Q, K and V. Mathematically, the output calculation is a vector of attentional weights of the values by applying a *softmax* function as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.4}$$

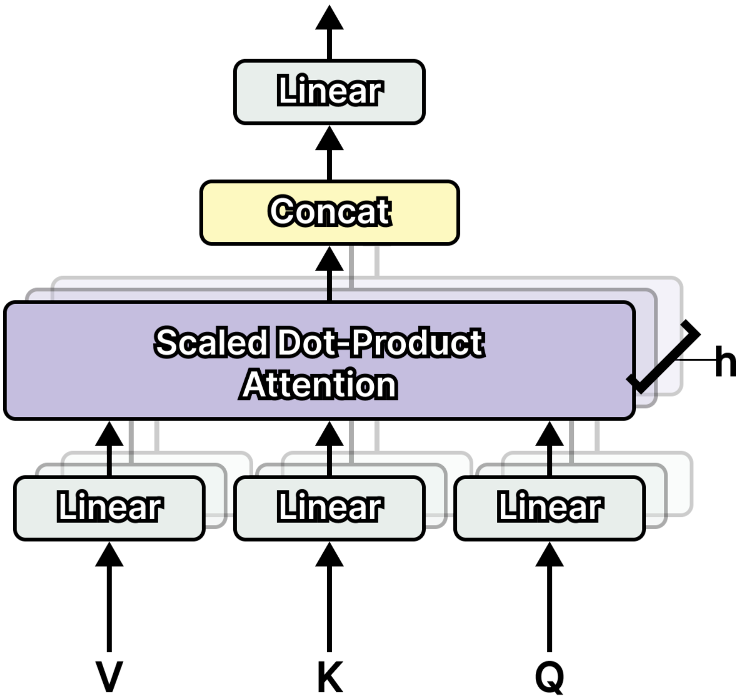


Figure 2.11: The Multi-head Attention block.

where d_k is the embedding size that represents the words in the data set.

In this way, it is possible to know the context of a word concerning any other word in a sequence, regardless of the distance between them [42]. In other words, this approach solves the problem of RNNs lack of long-term memory [37]. These attention mechanisms have revolutionized the world of NLP since their creation has allowed the development of giant models that have been successfully applied to many natural language tasks [42].

Transformer architecture

The attentional mechanisms allowed to solve the problem of lack of memory of the RNNs, and thanks to these the recurrent neural networks and others were benefited. However, in 2017, Vaswani et al. [46] proposed the idea that it is not necessary to combine attention mechanisms with RNNs to perform well. These authors mention that attentional mechanisms by themselves are enough to achieve similar or even superior performances to classical RNNs. The proposal of these authors was a new architecture called *transformer*.

The *transformer* is the first artificial neural network based entirely on attention mechanisms [42]. This architecture consists of two main blocks: the encoder and the decoder [43, 45], as shown in Fig. 2.12. The encoder block extracts the most relevant information from a sequence. The output of this block is connected to a decoding block that takes this result and generates a new output, which is the one with the highest probability according to the context of the sequence [43].

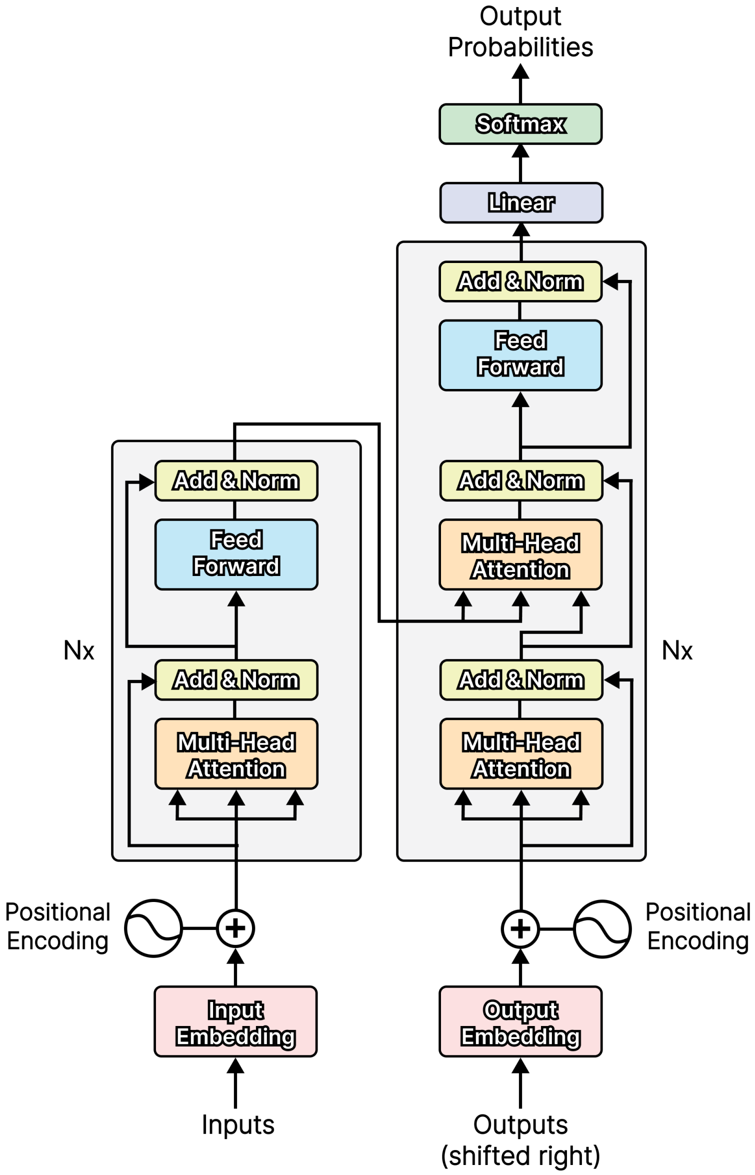


Figure 2.12: The transformer architecture.

The encoder and decoder blocks' main components are the *feed forward neural network* and the *multi-head attention module* [45]. A feed forward network is a classical neural network in which information flows in one direction only, from the input nodes to the output nodes [26]. A multi-head attention module is a mechanism that works with several self-attention components in parallel, as illustrated in Fig. 2.11. The independent attention outputs are concatenated and linearly transformed into the expected dimension [43, 45].

Unlike RNNs, which process an input sequentially, the Transformer processes the input in parallel [43, 44]. A priori, this has a disadvantage: in a language sequence, the order is important. By processing all the components of a sequence simultaneously, the contextual information that allows us to know its meaning is lost [43]. For this, the creators of the Transformer propose a special type of encoding called *positional encoding.*

This is a non-absolute positional encoding that uses an intuition based on sine and cosine waves. Mathematically the positional encoding is defined as follows:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{2.5}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{2.6}$$

where pos is the position of the word within the input sequence, i is the dimension, and d_{model} the embedding dimension. The constant term is because the wavelengths form a geometric progression from 2π to $10000{\cdot}2\pi$ [46].

In this way, it is possible to tell the Transformer the order of the words that make up an input sequence [43]. The attention mechanisms and this parallel processing make the transformer a superior architecture to the RNN since it is possible to process a larger amount of data efficiently.

Chapter 3

State of the Art

3.1 Image captioning

A large number of systems have been proposed for image captioning. A recognized architecture is the one proposed by *Xu et al.* [17]. This architecture is, in short, an encoder-decoder architecture. First, a convolutional network is used to perform feature extraction. Then, the decoder, powered by an attentional mechanism, focuses on specific parts of the image and generates an output sequence, the image caption. The authors used the MS COCO data set and the BLEU and METEOR metrics to train the architecture to measure performance. As result, they obtained better numbers in all metrics concerning the models with which they were compared.

The work of *Liu et al.* [47] is quite similar to the previously mentioned. It also comprises an attentional mechanism, a feature extractor, and an LSTM decoder. However, the architecture incorporates a merging gate that adaptively adjusts the weight between the visual information and the conceptual information for generating the caption. For training, they used the MS COCO and Flickr30k data sets. The metrics SPICE, $CIDE_r$, METEOR, ROUGE-L, and BLEU-4. The results showed an improvement concerning other architectures; for example, the one proposed in [17].

Continuing with the trend of using attentional mechanisms, the work of *Qiu et al.* [48] proposes an encoder-decoder architecture for image captioning. In this case, the idea is similar to thw work seen in [17] but applied to the captioning of terrain images. For this purpose, they used the MICD data set containing Martian data. Furthermore, to evaluate the performance, they used the metrics BLEU, METEOR, ROUGE-L, and $CIDE_r$. As a result, they obtained state-of-the-art results and demonstrated that this type of network can and has scope for application in planetary and astronomical domains.

As technology advanced, new components were adopted to enhance and improve existing architectures. For example, we have the work of *Lei et al.* [49] which incorporates a transformer module. First, feature extraction is performed by adapting a Region Based CNN. In this way, they could perform a multi-level encoding that allowed them to have more accurate information about the image. Then, they modified a transformer architecture, incorporating a sparse module between the scale and softmax functions of this one. This module is then connected to an LSTM to form the decoder. For training, they used the data sets MS COCO and Flickr30k and measured the performance using BLEU, METEOR, $CIDE_r$, and ROUGE-L. The results they obtained showed slight improvements over the baseline architectures.

The work of *Li et al.* [50], in the meanwhile, proposes a modification to the classical encoder-decoder networks used for image captioning. In addition to containing the image captioning network, it includes a multi-level similarity fusion network. This allows to compute the local and global semantic similarities and merges them to build the multilevel similarity-based reward. The reward is back-propagated to guide the parameter update of the caption generation network by a reinforcement learning algorithm, to achieve better matching semantics. To train the model, they used the MS COCO data set. The metrics BLEU, METEOR, ROUGE-L, $CIDE_r{-}D$, and SPICE. As a result, they obtained numbers that reflected a state-of-the-art performance of their proposal. They outperformed the classical models combining CNN and RNN; however, they were slightly inferior to the transformer-based proposals.

Table 3.1 shows a summary of the works analyzed above.

Table 3.1: Summary of image captioning related works.

Architecture	**Data input**	**Data set**	**Loss function**	**Optimizer**	**Performance metrics**	**Reference**
CNN + LSTM	Images	MSCOCO	Cross entropy	Adam	BLEU, METEOR	[17]
CNN + LSTM	Images	MSCOCO, Flickr30k	Cross entropy	Adam	SPICE, $CIDE_r$, METEOR, ROUGE-L, BLEU	[47]
Visual attention + LSTM	Images	MICD	Cross-entropy	Adam	BLEU, METEOR, ROUGE-L, $CIDE_r$	[48]
R-CNN + Transformer + LSTM	Images	MSCOCO, Flickr30k	Cross-entropy	Adam	BLEU, METEOR, CIDEr, ROUGE-L	[49]
Faster R-CNN + Attentional LSTM	Images	MSCOCO	Cross-entropy	Adam	BLEU, METEOR, ROUGE-L, $CIDE_r$-D, SPICE	[50]

3.2 ConvNeXt

The inclusion of transformers generated a revolution and meant a significant improvement in many neural network architectures at the time, especially those applied to images. As a result, they gradually replaced CNNs as the base architecture for any Computer Vision task. Moreover, the transformers inspired many researchers to propose more and better architectures to compete with them. One of them is the ConvNeXt architecture which is directly inspired by the design of the Transformers. This is an architecture proposed in 2022 that claims to match and, in some cases, surpass the transformers' performance.

Despite being a new architecture, many researchers are already incorporating it into their research areas. According to Table 5, there are many fields in which this new architecture is being applied. For example, we have the work of *Yang et al.* [51], which applies ConvNeXt in remote sensing. Specifically, the authors made several modifications to a YOLOv4 architecture, including incorporating ConvNeXt to replace the original feature extractor. This, along with the other modifications, allowed the authors to outperform the original architecture and other state-of-the-art options. Two data sets, RSOD and NWPU VHR-10, were used for training. The metrics used were the F1 score and mean average precision.

In another field, the work of *Zhu et al* [52]. proposes the use of ConvNeXt in the medical field. Specifically, the authors propose a network for predicting the origin for bone metastatic cancer. For this purpose, they used a network composed of an encoder and an attention-pooling mechanism. Once feature extraction is performed, the features are merged into a representative vector by attention-based pooling. They used a proprietary data set that they collected from eight healthcare centers in China for training and testing. They also used recall, precision, ACC, and ROC metrics to measure the performance of the approach. They obtained state-of-the-art results and demonstrated the effectiveness of the model for detecting cancer and some types of cancer.

A similar work is proposed by *Hassanien et al.*. [53], which uses ConvNeXt as part of an architecture for breast tumor malignancy prediction. The architecture consists of three main elements, a backbone, a pooling mechanism, and a visual explanation mechanism. The backbone is responsible for the feature extraction, the pooling mechanism generates the malignancy score for each input sequence, and the visual explanation mechanism helps interpret the deep learning decisions to generate a prediction. They used a data set of malignant and benign sequences for training, part of a clinical database of ultrasonic radiofrequency strain imaging data created by the Engineering Department of Cambridge University. Likewise, the metrics used were accuracy, precision, recall, and F1 score. It should be noted that, in addition to ConvNeXt, they tested other architectures for feature extraction; however, ConvNeXt proved to provide the best numbers in conjunction with the other mechanisms that make up the complete architecture.

In the work of *Xu et al.* [27], a visual tracking architecture is proposed. It comprises three components. First, a ConvNext is used as a feature extractor. Then, a feature fusion network suppresses the responses of similar targets by applying a 3D C-Max fusion network to enhance the network's feature recognition capability. Finally, a multibranch prediction network comprises classification and regression branches for foreground and background classification. The data sets used in this work were ILSVRC-VID/DET, COCO, and GOT-10k. The metrics used to measure performance were AUC, precision, success rate graph, average overlap, frames per second, and degree success rate. As a result, they obtained competitive numbers in all metrics and even outperformed other variants, such as ResNet-based architectures.

In the work of *Hoo et al.* [54], the authors' proposal is focused on using and modifying a ConvNeXt for face recognition. The authors propose some variations of the ConvNeXt, whose differences are centered on different combinations of downsampling approaches in the stem partition and the bottleneck partition. Moreover, the downsampling approaches differ in kernel size for convolutional operation. Specifically, they introduced a non-overlapping patch in addition to the typical overlapping approach. The authors used the UMD Faces data set to train all models. Also, the metric used to evaluate performance was accuracy. As a result, they obtained that their models present a state-of-the-art performance. However, they did not outperform other alternatives by a short margin. Even so, it should be noted that the authors' proposal presented lighter models than the existing ones.

Table 3.2: Summary of ConvNeXt related works.

Architecture	**Task**	**Data input**	**Data set**	**Loss function**	**Optimizer**	**Performance metrics**	**Reference**
Backbone-Neck-Head	Remote sensing	Images	NWPU VHR-10, RSOD	EIoU	N/A	Precision, F1-score	[51]
Encoder + Attentional pooling	Bone metastatic cancer prediction	Images	Chinese health-care centers	Cross entropy	Adam	recall, precision, ACC, ROC	[52]
Backbone + Pooling + Visual explanation	Breast tumor malignancy prediction	Images	Clinical dataset Cambridge University	Cross entropy	Adam	Accuracy, precision, recall, F1-Score	[53]
Backbone + Siamese network	Visual tracking	Images	ILSVRC-VID/DET, COCO, GOT-10k	Cross entropy	Gradient descent	AUC, precision, average overlap, frames per second, success rate graph, and degree success rate	[55]
Modified ConvNeXt	Face recognition	Images	UMD Faces	ArcFace	Stochastic Gradient descent	Accuracy	[54]

Chapter 4

Methodology

4.1 Data Set Description and Preprocessing

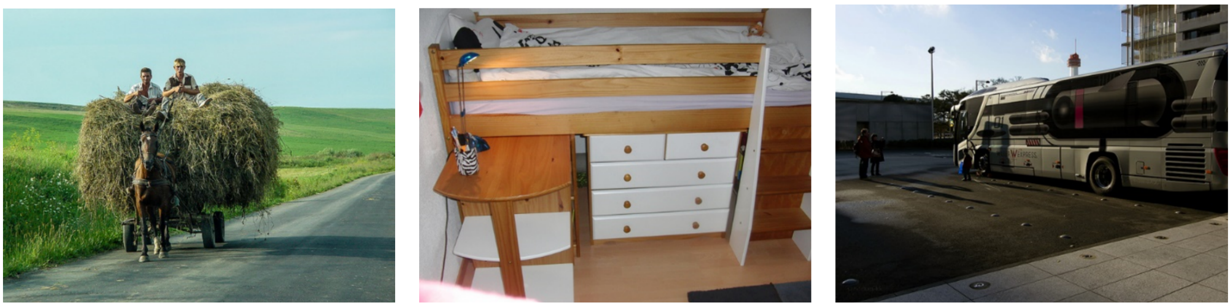
This work uses the 2014 MS COCO [56] dataset, the variant intended for image captioning. Moreover, we report results using the splits described in the work of *Karpathy and Li* [57]. The dataset contains 123,287 images in total. The splits provide ready-made datasets for training, validation and testing. There are 5,000 images designated for the validation set, 5,000 for testing and the remaining for training. Fig. 4.1 shows some example images of this dataset. It is also necessary to indicate that the model receives three inputs: the images, the captions, and the lengths of the captions.

Since we use a pre-trained model, the input images must have the structure required by it to work correctly. For this reason, the images are resized to 224 × 224 pixels. Likewise, it is necessary to normalize the images; for this, the mean ($\mu = [0.485, 0.456, 0.406]$) and standard deviation ($\sigma = [0.229, 0.224, 0.225]$) of the ImageNet dataset are used.

Now, the captions are used both as the model's target and the decoder's input. The latter needs to know where the caption begins and ends; for this reason, it is necessary to add two characters to delimit its extension. These are represented as $< start >$ and $< end >$ 'signals.' With the addition of these characters, a caption would look like this:

$$< start > \text{a large bus sitting next to a very tall building} < end >$$

Also, since not all captions are of a fixed length, filling the missing spaces in the caption with a padding character is necessary. This is represented by $< pad >$. With the inclusion



(a) Caption: "a horse carrying a large load of hay and two people sitting on it"

(b) Caption: "bunk bed with a narrow shelf sitting underneath it"

(c) Caption: "a large bus sitting next to a very tall building"

Figure 4.1: Some examples taken from the MS COCO dataset and their respective ground-truth captions.

of this character, a caption would look as follows:

$$< start > \text{ a large bus sitting next to a very tall building } < end >< pad >< pad >$$

Furthermore, given that every neural network model does not receive letters as input but numbers, it is necessary to encode the input. Each word is assigned an identification number for this purpose and by means of a dictionary-based mapping system. In this way, each of the captions is represented as a numerical array. For the example in Fig. 4.1c, the input caption after all the preprocessing would look something like the one shown in Table 4.1.

Regarding caption lengths, these are just numerical arrays containing the number of words that make up the description of each image.

4.2 Model Proposal

4.2.1 Benchmark model

Our proposal is inspired by the architecture of *Xu et al.* [17], and will be used as the benchmark. This can be seen in Fig. 4.2.

The authors of the mentioned work proposed an encoder-decoder architecture with visual attention mechanism. The model takes a raw image and generates a caption y

Table 4.1: Example of how captions are encoded. The caption is the one in Fig. 4.1c.

Word	**Encoding**
$< start >$	9876
a	1
large	2
bus	3
sitting	4
next	5
to	6
a	7
very	8
tall	9
building	10
$< end >$	9877
$< pad >$	9878
$< pad >$	9878
...	...

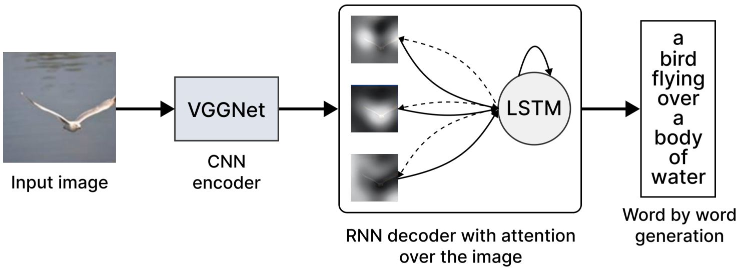

Input image
VGGNet
CNN encoder
LSTM
RNN decoder with attention over the image
a bird flying over a body of water
Word by word generation

Figure 4.2: Encoder-decoder architecture for image captioning used as benchmark.

encoded as a sequence of 1-of-K encoded words as follows:

$$y = \{y_1, y_2, \cdots, y_C\}, \quad y_i \in \mathbb{R}^K$$

where K is the vocabulary size and C is the length of the caption.

For feature extraction, they used a CNN VGGNet pre-trained on ImageNet. This produces L vectors (known as annotation vectors), each of which is a D-dimensional representation corresponding to a part of the image.

$$a = \{a_1, a_2, \cdots, a_L\}, \quad a_i \in \mathbb{R}^D$$

For the encoder, they used an LSTM that produces a caption by generating a word at each decoding step. For each annotation vector at time t, the attentional mechanism computes the corresponding weights using the previous hidden state (h_{t-1}) and passing this output through a softmax function:

$$w_{ti} = Att(a_i, h_{t-1})$$

$$\alpha_{ti} = \frac{exp(w_{ti})}{\sum_{K=1}^{L} exp(\lambda_{tk})}$$

Once the weights have been calculated, the context vector $\tilde{z}_t$ is calculated as follows:

$$\tilde{z}_t = \phi(\{a_i\}, \{\alpha_i\}),$$

where ϕ is a function that returns a single vector given the set of annotation vectors and their corresponding weights. The context vector $\tilde{z}_t$ is nothing more than a dynamic representation of the relevant parts of the input image at time t.

Finally, the probability of the output word is calculated using the context vector, the previously generated word, and the previous hidden state as follows:

$$p(y_t \mid a, y_1^{t-1}) \propto exp(L_0(Ey_{t-1} + L_h h_t + L_z \tilde{z}_t))$$

where $L_0 \in \mathbb{R}^{Kxm}$, $L_h \in \mathbb{R}^{mxn}$, $L_z \in \mathbb{R}^{mxD}$, and E are learned parameters initialized randomly. Note that m and n denote the embedding and LSTM dimensionality respectively. For a more detailed explanation of this architecture see [17].

4.2.2 ConvNeXt-based approach

Given the above context, our proposal focuses on studying the alternative of replacing the encoder of the original architecture with a current state-of-the-art model. Specifically, we propose the use of the ConvNeXt architecture as the encoder. Originally, ConvNeXt was trained and tested for image classification tasks. In that field, ConvNeXt has already demonstrated its viability and effectiveness as it reported superior performance to the CNNs of the time and also outperformed vision transformers. For this reason, and given the outstanding performance shown by ConvNeXt, our experiments are focused on evaluating the performance of this architecture for image captioning tasks.

Following a methodology similar to the benchmark work, we use transfer learning employing pre-trained models without finetuning to implement our proposal. In this way, we avoid training the entire architecture from scratch. Since we use pre-trained models, it is necessary to modify them to use them as feature extractors. Specifically, it is necessary to remove the last layer of the ConvNeXt, which contains the MLP classification head, since we will not perform classification tasks.

Once the feature extraction is done, this information is passed to the LSTM-based encoder, with attentional mechanisms over the image. It is important to note that, in this work, we use the 'soft attention' approach, stated by the benchmark authors, for which we calculate a weighted soft attention annotation vector ($\tilde{z}_t$) defined as:

$$\tilde{z}_t = \sum_{i=1}^{L} \alpha_{t,i} a_i$$

and described above.

Finally, the encoder generates one word at a time until the final caption is obtained. The general scheme of the proposed architecture is shown in Fig. 4.3.

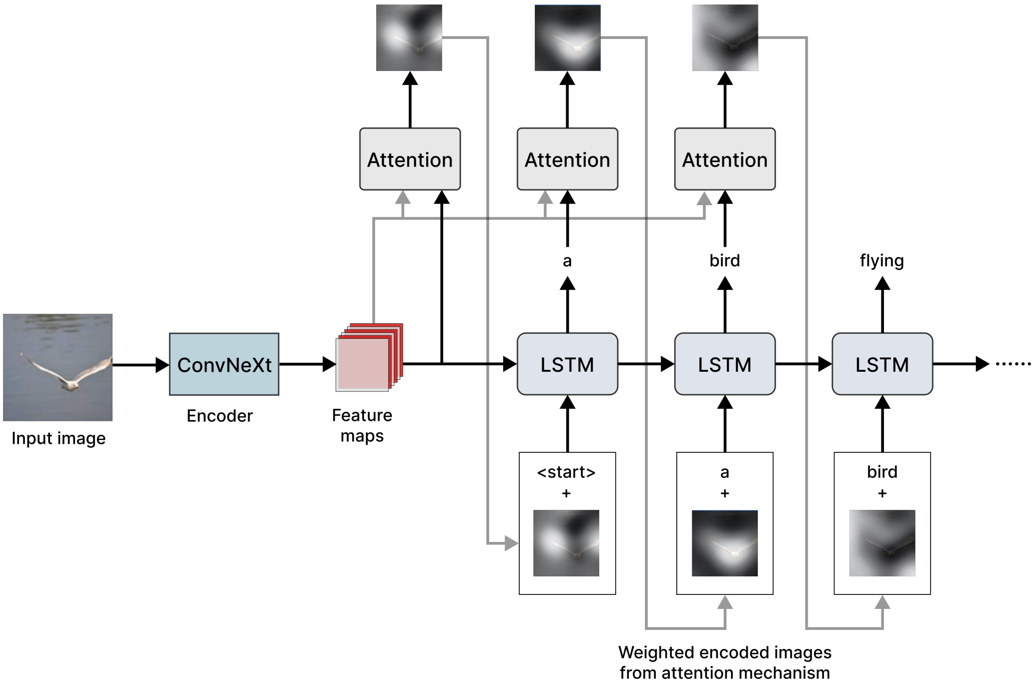


Figure 4.3: Overall representation of the proposed ConvNeXt backbone-based adaptation for image captioning.

4.3 Experiments

As mentioned above, the main objective of this work is to study the effect of replacing the encoder of the benchmark architecture with the new ConvNeXt architecture. Nevertheless, we also want to perform different experiments and analyze the proposal's performance in each of them.

Therefore, the first case study we analyzed is to evaluate the proposal's performance using different versions of ConvNeXt. These range from the lightest version to a more robust version with a larger number of parameters. The details of each of these are shown in Table 4.2.

Table 4.2: ConvNeXt variants used in this work. All version are pre-trained on ImageNet-1K dataset; accuracy results shown were also measured on this dataset.

Model	**Resolution**	**Acc@1**	**Acc@5**	**Params**	**FLOPs**
ConvNeXt tiny	224x224	82.52	96.15	28M	4.5G
ConvNeXt small	224x224	83.62	96.65	50M	8.7G
ConvNeXt base	224x224	84.07	96.87	89M	15.4G
ConvNeXt large	224x224	84.42	96.98	198M	34.4G

It should be noted that some implementation modifications are necessary depending on the version of ConvNeXt used. Specifically, all ConvNeXt versions used generate a final encoding of size 7x7, but each one generates a different number of channels, as shown in Table 4.3.

Table 4.3: Final encoding size produced by each of the ConvNeXt variants used.

Model	**Encoding size**
ConvNeXt tiny	(768, 7, 7)
ConvNeXt small	(768, 7, 7)
ConvNeXt base	(1024, 7, 7)
ConvNeXt large	(1536, 7, 7)

The next point we decided to evaluate is to vary some of the parameters with which the architecture is trained. Specifically, we decided to use two different learning rates (α) to train the proposal and analyze the impact of each of them on its performance. We did this since performing experiments with smaller learning rates is recommended when working with pre-trained models [58]. The learning rates used are shown in Equations 4.1 and 4.2.

$$\alpha = 4e - 4 \tag{4.1}$$

$$\alpha = 1e - 4 \tag{4.2}$$

It is important to remember that we decided to use different versions of ConvNeXt for our proposal, so each architecture variant was trained with each of the mentioned learning rates.

The last aspect we decided to evaluate was the use of teacher-forcing (TF) during the architecture training. In a nutshell, teacher-forcing is a technique for training RNNs in which, at each time step, the ground-truth captions are used as the input to the decoder and not the word that the decoder generated at the previous time step [59, 60, 61]. The use of TF usually helps the training to be faster and more efficient [62]; however, the model may present some instability in the evaluation since it learned to depend on the ground-truth captions. Along with the previous experiment, each variant of the architecture was trained with and without TF to evaluate the difference in performance.

The entire workflow used in this work can be seen in Fig. 4.4.

The implementation was performed using the PyTorch framework, while the pre-trained models were obtained from the torchvision library. The loss function and optimizer used were Cross entropy and Adam, respectively. Also, the training was implemented in such a way that this is stopped when there is no improvement in the metrics for 20 epochs in a row or when 120 epochs have been reached. In another aspect, training was performed using an HPC node with a 32-core AMD EPYC 7742 processor and a Nvidia A100 SXM4 graphics card of 40 GB.

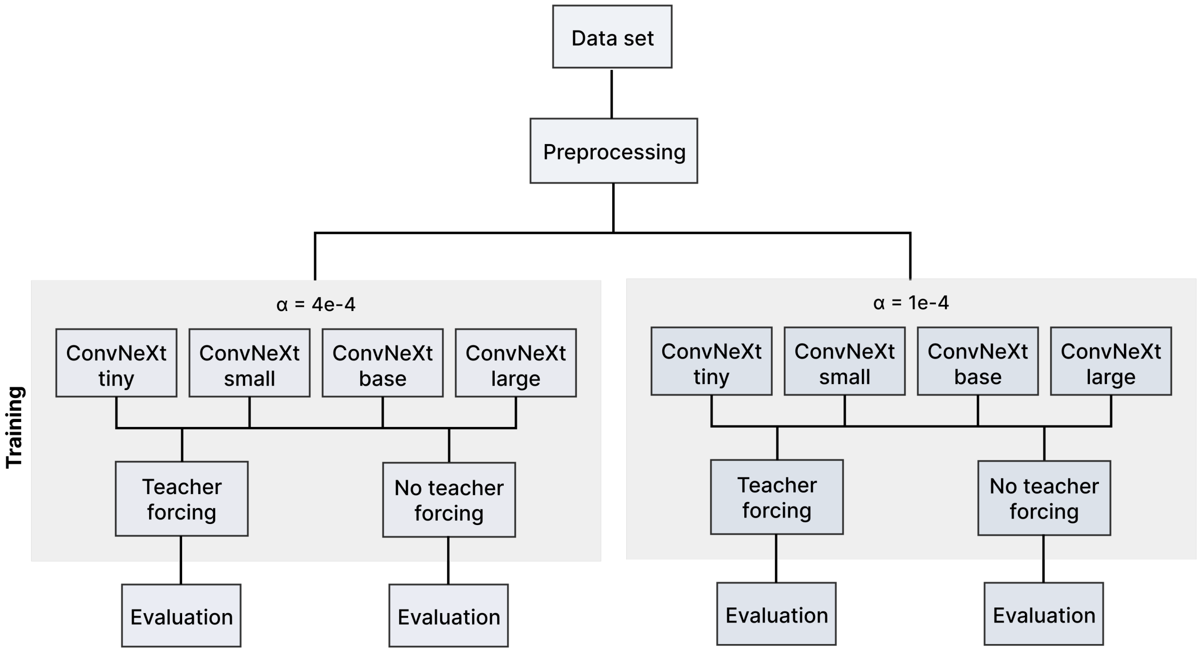


Figure 4.4: Overall workflow followed in this study.

4.3.1 Performance Metrics

In addition to measuring the loss of the trained models, we use two metrics to evaluate the performance of our approach.

BLEU-n

BLEU-n is a metric that evaluates machine-generated texts by comparing them to one or more reference texts created by humans [63, 48]. This evaluation is performed at the n-gram level, which means that BLEU-n measures the overlap between the machine-generated sequences and the reference texts based on consecutive sequences of n words [48]. The BLEU-n score is calculated by taking the geometric mean of the n-gram precisions, where each precision value is weighted according to the length of the n-gram [64]. Furthermore, a brevity penalty factor is applied to discourage the machine-generated texts from being significantly shorter than the reference texts [65]. In mathematical terms, we can define it as:

$$\text{BLEU-n} = \text{BP} \cdot \exp\left(\sum_{i=1}^{n} w_i \log P_i\right)$$

where BP is the brevity penalty factor, w_i are the weights for each level of n-gram, and P_i is the precision of each level of n-gram.

Top-5 accuracy

The next metric used is top-5 accuracy. This is a specific type of top-n accuracy. In general terms, the top-n accuracy metric measures the proportion of times the model correctly predicts the first n elements in a set of possible outcomes [66, 67]. In this case, we measure the proportion of predictions that match the correct labels within the top five prediction options [68]. Mathematically, we can define it as:

$$\text{top-5 accuracy} = \frac{1}{N}\sum_{i=1}^{N}[y_i \in \text{top-5}(f(x_i))]$$

where N is the total number of examples, x_i is the $i-th$ example, y_i is the true label for the $i-th$ example, $f(x_i)$ is the model making predictions on the $i-th$ example, and top-5($f(x_i)$) is the set of the top five predictions according to the model $f(x_i)$.

Chapter 5

Results and Discussion

Table 5.1 shows the results obtained from the models trained with a learning rate of $4e-4$, regarding the BLEU metrics. From this, the first evident thing is that the models that do not use TF show better performance than those that do. The gap between models that use TF and those that do not is, in all cases, greater than 8 points concerning the BLEU-4 metric.

Likewise, the ConvNeXt-base models are the ones that show the best performance among all the alternatives tested both with and without TF. The next model is the ConvNeXt-large with the second-best performance shown. However, the difference in performance between the base and large variants is more significant when TF is not used. Specifically, the difference between the base and large variant is 0.03 and 0.44 in TF and no TF, respectively, relative to the BLEU-4 metric. Regarding the ConvNeXt-small and ConvNeXt-tiny variants, they show lower performance than the more robust alternatives. However, it is repeated that the differences in performance are smaller when TF is used.

Moving on to training using the learning rate of $1e-4$, a similar behavior to that seen with the larger learning rate is observed, as shown in Table 5.2. However, the gap between the models with and without TF is more significant in this case. Specifically, all models without TF outperform those using TF by no less than 9 points in terms of BLEU-4. Likewise, the Convext-base variants again show the best performance whether or not TF is used, over the other variants. In second place are again the Convext-large variants, and finally, the Convext-small and Convext-tiny variants.

It is evident that avoiding the use of TF produces better results. In all the experimental

Table 5.1: BLEU metrics obtained by the best checkpoint generated using a learning rate of $4e-4$.

	Model	**BLEU-1**	**BLEU-2**	**BLEU-3**	**BLEU-4**
Teacher-forcing	ConvNeXt-tiny	72.90	52.70	35.63	24.03
	ConvNeXt-small	73.09	52.99	36.07	24.55
	ConvNeXt-base	73.17	53.02	36.11	**24.60**
	ConvNeXt-large	73.35	53.20	36.14	24.57
No teacher-forcing	ConvNeXt-tiny	74.37	57.79	44.03	33.59
	ConvNeXt-small	73.81	57.16	43.44	33.20
	ConvNeXt-base	74.61	57.88	44.48	**34.20**
	ConvNeXt-large	74.30	57.71	44.05	33.76

Table 5.2: BLEU metrics obtained by the best checkpoint generated using a learning rate of $1e-4$.

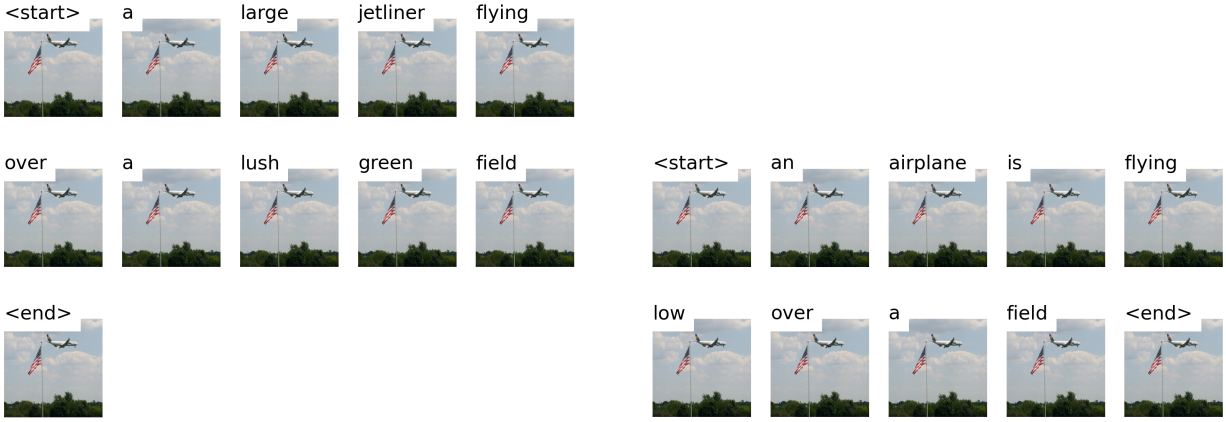
	Model	**BLEU-1**	**BLEU-2**	**BLEU-3**	**BLEU-4**
Teacher-forcing	ConvNeXt-tiny	72.75	52.62	35.58	24.04
	ConvNeXt-small	73.22	52.96	36.13	24.54
	ConvNeXt-base	73.17	53.16	36.17	**24.63**
	ConvNeXt-large	73.32	53.13	36.16	24.58
No teacher-forcing	ConvNeXt-tiny	74.59	57.92	44.10	33.61
	ConvNeXt-small	74.37	57.69	43.95	33.59
	ConvNeXt-base	74.79	58.07	44.61	**34.76**
	ConvNeXt-large	74.39	57.92	44.41	34.24

scenarios analyzed, the models that did not use TF were superior in performance. For this reason, we will focus our further analysis on this set of models.

In order to graphically analyze the performance of the models, Fig. 5.1 and Fig. 5.2 were made. The first corresponds to the models with the highest learning rate, and the second corresponds to the models with the lowest learning rate.

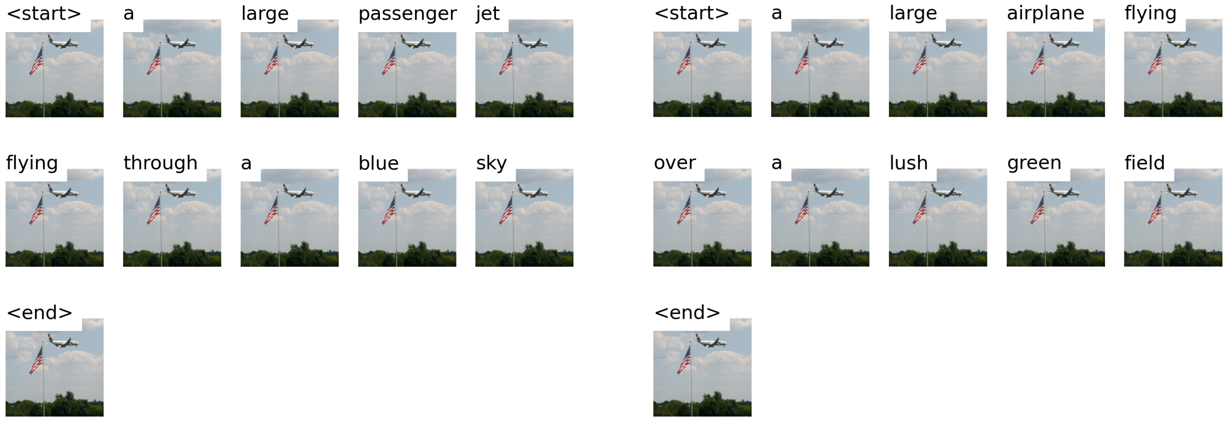
Regarding Fig. 5.1, it is evident that all models deliver coherent captions; however, none sufficiently match the ground-truth. The ConvNeXt-tiny, ConvNeXt-small, and ConvNeXt-large variants delivered similar results but did not describe all relevant objects in the image. The ConvNeXt-base variant generated a different, consistent caption but did not accurately reflect the entire scene, relative to the ground-truth. In none of the captions generated was the flag mentioned as part of the scene.

Turning to Fig. 5.2, on the one hand, it is evident that the ConvNeXt-tiny and



(a) Image caption generated by ConvNeXt-tiny model.

(b) Image caption generated by ConvNeXt-small model.



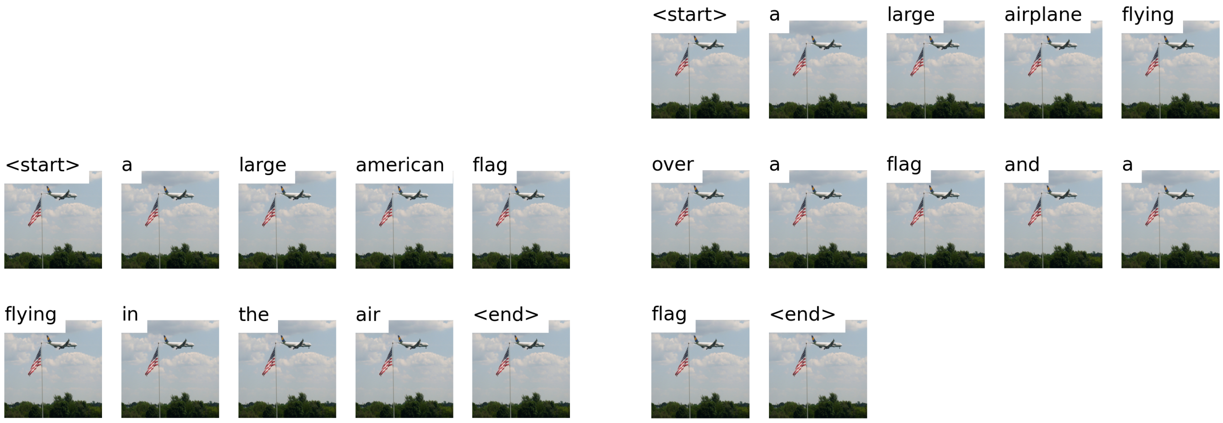
(c) Image caption generated by ConvNeXt-base model.

(d) Image caption generated by ConvNeXt-large model.

Figure 5.1: Visual results of image captions using a learning rate of $4e-4$. **Ground-truth**: *"The plane was flying high near the flag in the air"*

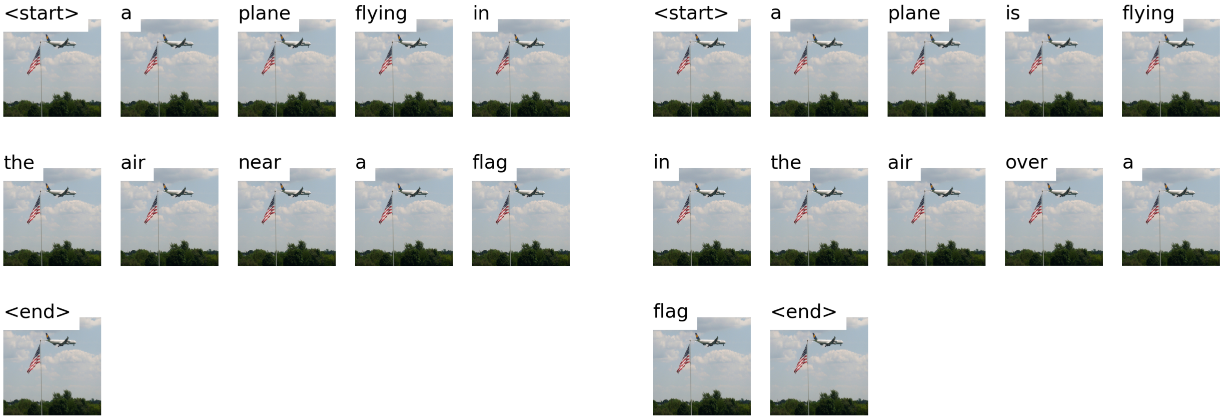
ConvNeXt-small variants give erroneous results, both in coherence and in the match with the ground-truth. On the other hand, the ConvNeXt-base and ConvNeXt-large models give very accurate results similar to the ground-truth. The ConvNeXt-base variant stands out as the model that comes closest to ground-truth in terms of matching. The ConvNeXt-large variant differs slightly from the ground-truth, but the caption generated by it is still significantly accurate and correctly describes the scene.

For this example, it is clear that the ConvNeXt-base and ConvNeXt-large models trained with the lowest learning rate produce the best results overall. Likewise, some superiority is observed for ConvNeXt-tiny and ConvNeXt-small models trained with the lower learning rate over those trained with the higher learning rate.



(a) Image caption generated by ConvNeXt-tiny model.

(b) Image caption generated by ConvNeXt-small model.



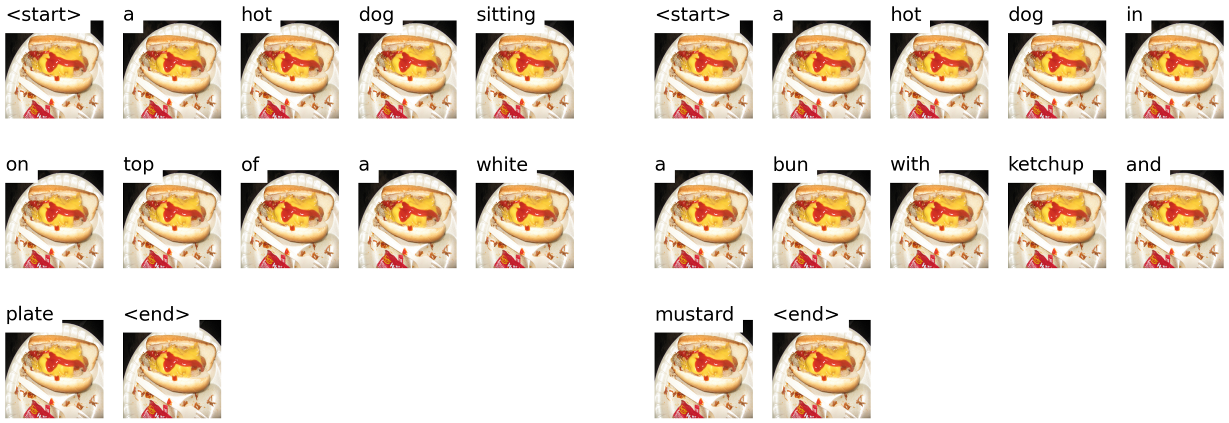
(c) Image caption generated by ConvNeXt-base model.

(d) Image caption generated by ConvNeXt-large model.

Figure 5.2: Visual results of image captions using a learning rate of $1e-4$. **Ground-truth**: *"The plane was flying high near the flag in the air"*

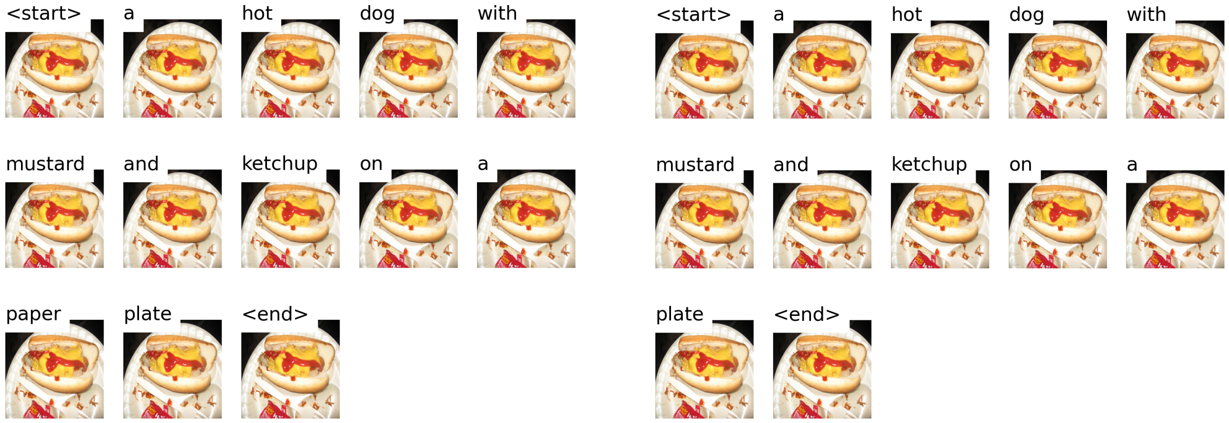
The second example is shown in Fig. 5.3 and Fig. 5.4. Similar to the previous example, the first corresponds to the models with the highest learning rate, and the second corresponds to the models with the lowest learning rate.

Concerning Fig. 5.3, and using the ground-truth as a reference, the ConvNeXt-tiny variant stands out as the only model that does not produce an adequate caption. Although this variant generated a coherent description, it does not describe the main object of the scene. Concerning the other variants, all generated adequate captions. However, the ConvNeXt-base and ConvNeXt-large variants stand out over ConvNeXt-small for including more details. These two variants have a good match with the ground-truth, describe the main object, and, in addition, include more details about the context.



(a) Image caption generated by ConvNeXt-tiny model.

(b) Image caption generated by ConvNeXt-small model.



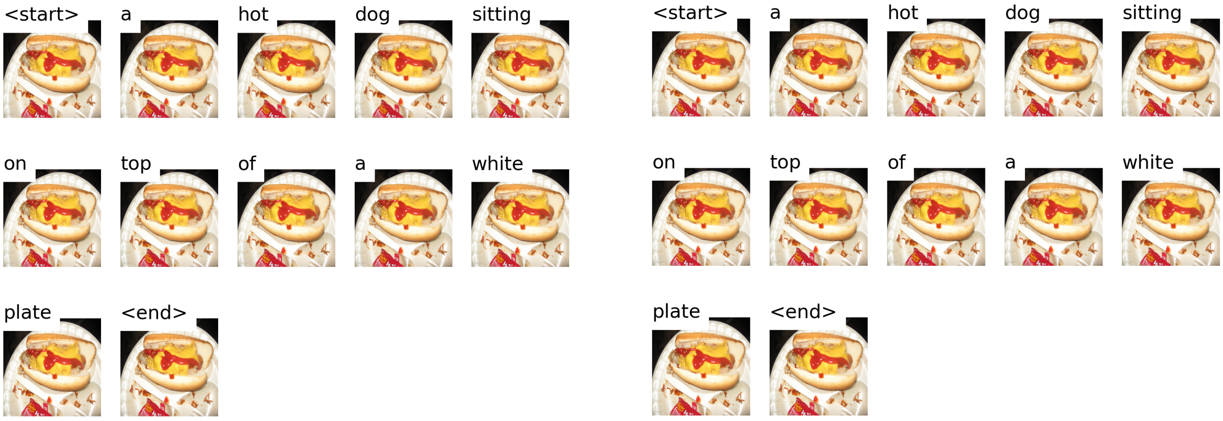
(c) Image caption generated by ConvNeXt-base model.

(d) Image caption generated by ConvNeXt-large model.

Figure 5.3: Visual results of image captions using a learning rate of $4e-4$. **Ground-truth**: *"A hot dog covered in mustard and ketchup."*

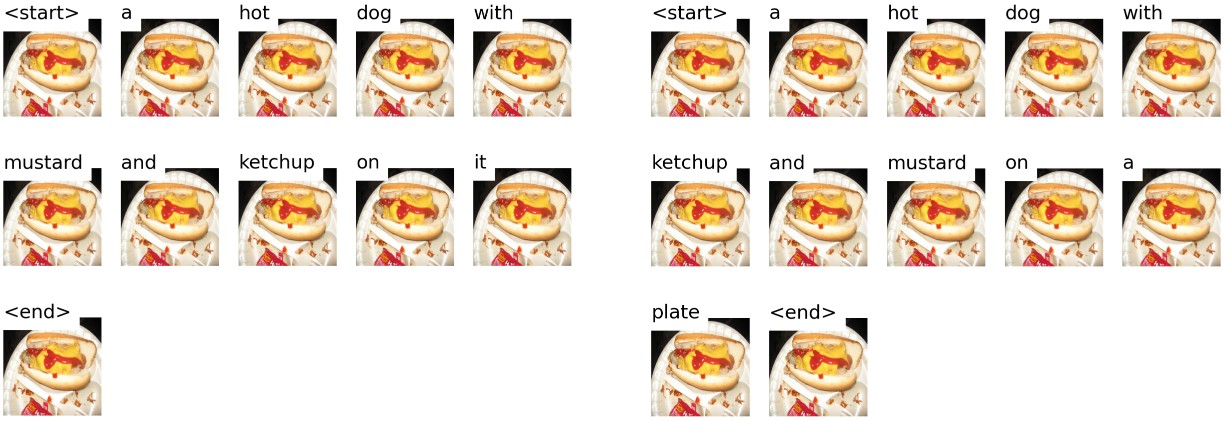
Turning to Fig. 5.4, the ConvNeXt-tiny and ConvNeXt-small variants stand out for having generated the captions that match the ground-truth the least. These two variants do not describe the most important elements of the scene. The ConvNeXt-base and ConvNeXt-large variants, on the other hand, generated coherent captions with a good match to the ground-truth. However, it should be noted that although both variants generated accurate captions, the ConvNeXt-large variant includes more detail in its description.

From these two examples, it is possible to highlight that the models that give the best overall results are the two most robust variants, ConvNeXt-base and ConvNeXt-large, trained with the lowest learning rate. Likewise, when we analyzed only the lighter



(a) Image caption generated by ConvNeXt-tiny model.

(b) Image caption generated by ConvNeXt-small model.



(c) Image caption generated by ConvNeXt-base model.

(d) Image caption generated by ConvNeXt-large model.

Figure 5.4: Visual results of image captions using a learning rate of $1e-4$. **Ground-truth**: *"A hot dog covered in mustard and ketchup."*

variants, ConvNeXt-tiny and ConvNeXt-small, they gave better results when trained with the largest learning rate. In short, on the one hand, for robust models, training with a lower learning rate produces better results; on the other hand, training with a larger learning rate generates better performance for the lighter models.

Moreover, considering the graphical results, and the numerical results in Table 5.2, the best model we obtained in overall performance is the ConvNeXt-base variant trained with the lowest learning rate and without TF. The second best model obtained is the ConvNeXt-large variant, trained under the same conditions as the first one.

Moving on, Table 5.3 shows the average inference times for each variant and with each learning rate used. From this table, it is possible to highlight that, in general, all models take a short time for inference. However, the models trained with the largest learning rate are shown to be slightly faster than those trained with the lowest learning rate. Likewise, the ConvNeXt-tiny variants are those that take the least time compared to their more robust alternatives during the inference process.

Table 5.3: Average inference time (seconds). Best checkpoints with no teacher-forcing.

Model	$\alpha = 4e-4$	$\alpha = 1e-4$
ConvNeXt-tiny	**0.0286**	**0.0362**
ConvNeXt-small	0.0367	0.0381
ConvNeXt-base	0.0453	0.0455
ConvNeXt-large	0.0461	0.0489

On another point, Table 5.4 shows the average loss of the models. From this table, as expected from what we have seen above, the ConvNeXt-base variants stand out as those with the lowest loss, concerning both learning rates used. In second place are the ConvNeXt-large variants, and then, the lighter variants. In general, a small loss is observed in all models.

Table 5.4: Average loss. Best checkpoints with no teacher-forcing.

Model	$\alpha = 4e-4$	$\alpha = 1e-4$
ConvNeXt-tiny	2.838	2.8383
ConvNeXt-small	2.8398	2.821
ConvNeXt-base	**2.7947**	**2.783**
ConvNeXt-large	2.8154	2.82

Moving on to the next point, Table 5.5 shows the results of the top-5 accuracy of the models according to each learning rate used. From this table, the ConvNeXt-base variants stand out as those with the best numbers, both for the smallest and the highest learning rate. Then there are the ConvNeXt-large variants. The ConvNeXt-base and ConvNeXt-tiny variants have similar numbers to each other.

Finally, we compared our best models' performance with the alternatives available in the literature. For this, we use the best model for each learning rate used. It should be remembered that we consider only the models without TF since they are the ones that give the best performance, as explained above.

Table 5.5: Top-5 accuracy. Best checkpoints with no teacher-forcing.

Model	$\alpha = 1e - 4$	$\alpha = 4e - 4$
ConvNeXt-tiny	77.399	77.322
ConvNeXt-small	77.298	77.419
ConvNeXt-base	**77.763**	**77.797**
ConvNeXt-large	77.556	77.487

Table 5.6 shows a comparison between our best models and some approaches from the literature in terms of loss and top-5 accuracy. This table shows that our best models outperform the other approaches by a considerable margin, both in loss and top-5 accuracy. Our ConvNeXt-base variant trained with the lowest learning rate stands out as the one with the best numbers among all.

Table 5.6: Comparison of our results with other approaches according to the top-5 accuracy and loss metrics. We compare our best models with both learning rates: $*\alpha = 4e - 4$, $**\alpha = 1e - 4$.

Encoder	**Top-5 accuracy**	**Loss**
ResNet-101[69]	73.092	3.413
ResNet-152[69]	73.077	3.412
VGG-16[69]	73.069	3.413
ResNeXt-101[69]	73.128	3.404
MobileNet V3[69]	72.928	3.424
ConvNeXt-base*	77.763	2.794
ConvNeXt-base**	**77.797**	**2.783**

Turning to Table 5.7, this compares our best models with the benchmark. Nevertheless, we also compare our proposal with another state-of-the-art approach. Specifically, we compare our proposal with the approach proposed in [69], in which transformers are used as feature extractors to perform image captioning. As mentioned above, transformers are modern and powerful architectures that replaced the classical CNNs, and are a great challenge to overcome.

The results show that our proposal outperforms the benchmark and transformer-based models in BLEU-4. Specifically, our best model, the ConvNeXt-base trained with a learning rate of $1e - 4$ and no TF, outperforms the state-of-the-art from [17] by 43.04 % for the soft-attention model and by 39.04 % for the hard-attention model. Likewise, our proposal outperforms the state-of-the-art from [69] by 4.57 % for the ViT-based model and by 0.93 % for the DeiT-based model. Furthermore, our models were not fine-tuned, in contrast to the models proposed in [69]. In other words, there is still room for improvement for our proposal.

Table 5.7: Comparison of our results with state-of-the-art approches according to the BLEU metrics. We compare our best models with both learning rates: $*\alpha = 4e-4$, $**\alpha = 1e-4$.

	BLEU-1	**BLEU-2**	**BLEU-3**	**BLEU-4**
SAT (Soft)[17]	70.70	49.20	34.40	24.30
SAT (Hard)[17]	71.80	50.40	35.70	25.00
ViT[69]	-	-	-	33.24
DeiT[69]	-	-	-	34.44
ConvNeXt-base*	74.61	57.88	44.48	34.20
ConvNeXt-base**	**74.79**	**58.07**	**44.61**	**34.76**

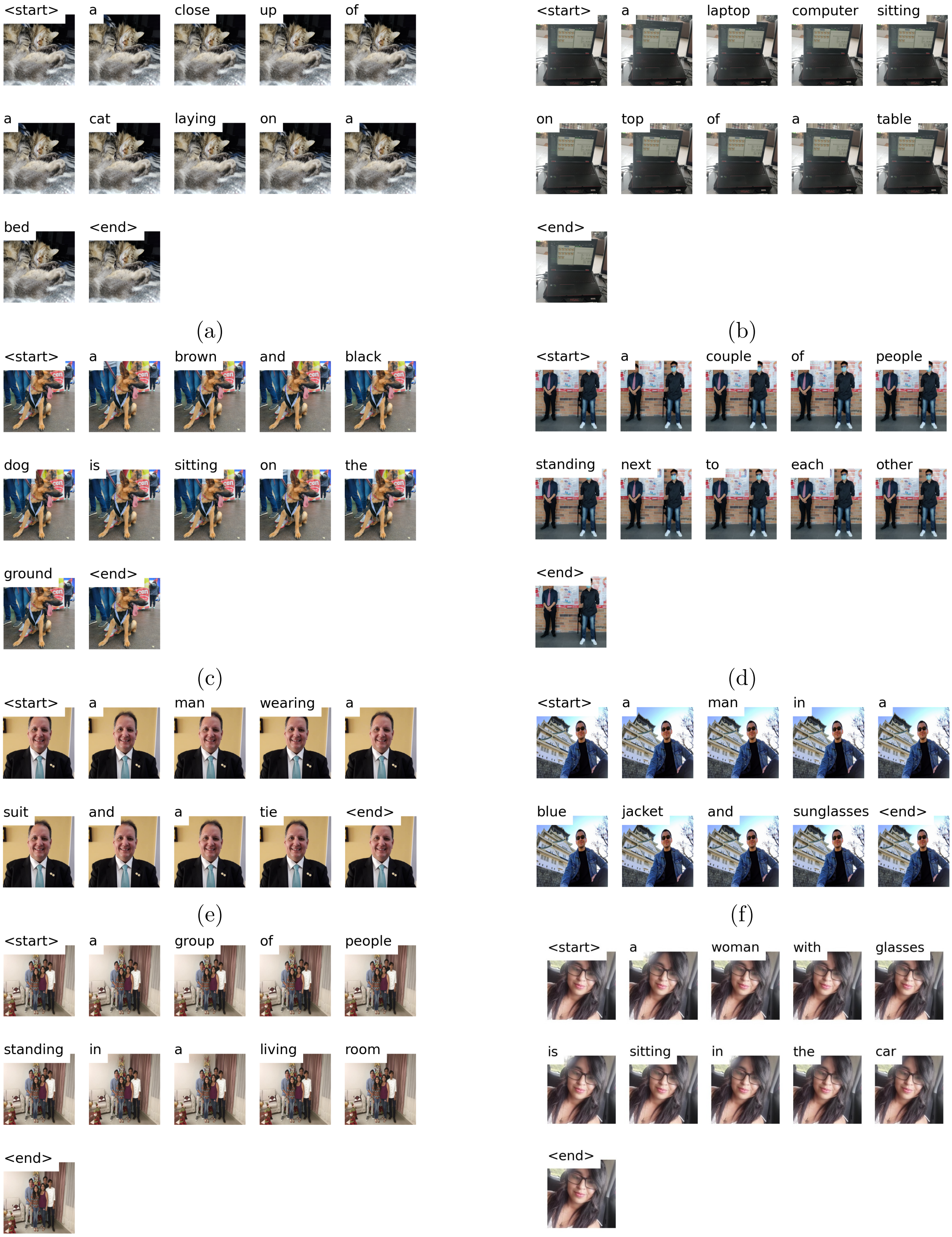
As a final experiment, we performed some tests using our best model on images captured by the authors of this paper using their cell phones. The images correspond to different contexts, including pets, friends, and objects. All images used are publicly available. The results can be seen in Fig. 5.5. These results show that our model is able to generate coherent descriptions with a good level of detail in different contexts. In all tests, we obtained a description that correctly reflected the scene, even considering specific details such as colors and accessories. This was especially evident in the example in Fig. 5.5h. In this example, it is difficult to identify that the person is inside a car; however, the model correctly detected this situation and described it.

We have successfully evaluated the approach in different contexts along with the latter tests. The tests performed included scenes with different numbers of entities and different types of things, including food, people, animals, simple objects, and complex objects. It was also shown that adequate descriptions of varying lengths could be generated.

5.1 Limitations

Some limitations of our work include the need for more knowledge of the proposal's performance in more specific contexts or applications. For example, in medicine, to describe objects and situations in medical images, in satellite images to describe geographic features such as rivers and mountains, and in wildlife images to describe a specific animal or plant species.

Given the nature of the dataset used to train the model, it is foreseeable that the proposal will underperform in some specific context in its current state. However, with an adequate re-training or fine-tuning of the proposal in a specific dataset, it is expected to perform similarly to the one reported in this work.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5.5: Image captioning results using the best model. The images were taken with cell phones in different contexts.

Chapter 6

Conclusions

This paper proposes an image captioning system that uses ConvNeXt, a state-of-the-art computer vision architecture, as a feature extractor. The feature extractor is combined with an LSTM block with a visual attention module to generate accurate and coherent descriptions. We conducted various tests in various situations to assess the efficiency of ConvNeXt. These experiments included testing four variants of ConvNeXt to analyze the impact on caption generation, training the system with two different learning rates to analyze the effect on the performance. And finally, we have analyzed the inclusion of teacher-forcing during the training process, and compared its performance without teacher-forcing.

As a result, we claim that teacher-forcing should be avoided to produce a better overall performance of the system. Likewise, it was shown that using a small learning rate can produce performance improvements, mainly when robust models are used as feature extractors. The opposite was demonstrated with lightweight models, which produce slightly better results when trained with larger learning rates.

Numerical results shows that our proposal is able to not only generate comprehensive captions, but also surpass the transformer-based benchmark's metrics. In particular, our leading model surpassed the state-of-the-art BLEU-4 by 43.04 % for its soft-attention model and 39.04 % for its hard-attention model. In addition, our best model outperformed the ViT and DeiT-based BLEU-4 by 4.57 % and 0.93 %, respectively. Similarly, it outperforms alternatives that utilize ResNet-101, ResNet-152, VGG-16, ResNeXt-101, and MobileNet V3 network-based encoders by 6.44%, 6.46%, 6.47%, 6.39%, and 6.68%, respectively, in terms of top-5 accuracy, and by 18.46%, 18.44%, 18.46%, 18.24%, and 18.72%, respectively, in terms of loss.

The significance of accurate image descriptions becomes evident, as even a minor misdescription of an object can potentially lead to severe consequences. Any additional percentage of accuracy, even a mere 1%, enables us to enhance the overall user experience and ensure the effective utilization of image captioning technology in real-world scenarios.

Additionally, the numerical findings were reinforced by the visual results, indicating that the suggestion produces logical descriptions of varying lengths. The experiments conducted encompassed various scenarios, including diverse quantities of entities and types of objects, such as food, individuals, animals, simple objects, and complex objects. The proposal was not only tested with images from the dataset but also with unfamiliar images taken by people. In this scenario, the proposal also demonstrated good performance, as it generated suitable descriptions.

For future works, it is suggested to extend the analysis of this proposal by expanding the number of experiments. For example, it is possible to perform hyperparameter tuning to verify if a different configuration allows for better performance. Also, it is possible to train and test this proposal with other datasets to verify its behavior.

Similarly, it is advisable to fine-tune the model by retraining some decoder layers, which is likely to improve the metrics. And finally, the impact of modifying further components, such as the decoder, with available state-of-the-art architectures can be studied to discuss if this update can complement the ConvNeXt enconder improving the overall performance of the system. The code and material developed in this work are available in a GitHub repository[1].

---

[1] `https://github.com/Leo-Thomas/ConvNeXt-for-Image-Captioning-.git`

Bibliography

[1] S. Herdade, A. Kappeler, K. Boakye, and J. Soares, “Image captioning: Transforming objects into words,” in *Proceedings of the 33rd International Conference on Neural Information Processing Systems.* Red Hook, NY, USA: Curran Associates Inc., 2019, pp. 11 137–11 147.

[2] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, “A comprehensive survey of deep learning for image captioning,” *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–36, feb 2019.

[3] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: Lessons learned from the 2015 mscoco image captioning challenge,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 652–663, 4 2017.

[4] S. Liu, L. Bai, Y. Hu, and H. Wang, “Image captioning based on deep neural networks,” *MATEC Web of Conferences*, vol. 232, p. 01052, 11 2018.

[5] Z. Zohourianshahzadi and J. K. Kalita, “Neural attention for image captioning: review of outstanding methods,” *Artificial Intelligence Review*, vol. 55, pp. 3833–3862, 6 2022.

[6] Z. Zeng and X. Li, “Application of human computing in image captioning under deep learning,” *Microsystem Technologies*, vol. 27, pp. 1687–1692, 4 2021.

[7] R. Sharma, A. Kumar, D. Meena, and S. Pushp, “Employing differentiable neural computers for image captioning and neural machine translation,” *Procedia Computer Science*, vol. 173, pp. 234–244, 2020, international Conference on Smart Sustainable Intelligent Computing and Applications under ICITETM2020.

[8] P. Dognin, I. Melnyk, Y. Mroueh, I. Padhi, M. Rigotti, J. Ross, Y. Schiff, R. A. Young, and B. Belgodere, “Image captioning as an assistive technology: Lessons learned from vizwiz 2020 challenge,” *Journal of Artificial Intelligence Research*, vol. 73, pp. 437–459, 1 2022.

[9] G. Xu, S. Niu, M. Tan, Y. Luo, Q. Du, and Q. Wu, “Towards accurate text-based image captioning with content diversity exploration,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 12 637–12 646.

[10] L. Ke, W. Pei, R. Li, X. Shen, and Y. W. Tai, “Reflective decoding network for image captioning,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, pp. 8887–8896, 10 2019.

[11] P. Kuznetsova, V. Ordonez, A. Berg, T. Berg, and Y. Choi, “Collective generation of natural image descriptions,” in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Jeju Island, Korea: Association for Computational Linguistics, Jul. 2012, pp. 359–368. [Online]. Available: https://aclanthology.org/P12-1038

[12] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig, “From captions to visual concepts and back,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 1473–1482, 10 2015.

[13] 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 (ICCV)*, October 2019, pp. 4634–4643.

[14] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, “Transformers in vision: A survey,” *ACM Comput. Surv.*, vol. 54, no. 10s, sep 2022. [Online]. Available: https://doi.org/10.1145/3505244

[15] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, “Twins: Revisiting the design of spatial attention in vision transformers,” in *Advances in*

Neural Information Processing Systems, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 9355–9366. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/4e0928de075538c593fbdabb0c5ef2c3-Paper.pdf

[16] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[17] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML’15. JMLR.org, 2015, p. 2048–2057.

[18] L. Ramos, “Artificial intelligence for cancer detection using medical image: Highlights and limitations,” *Green World Journal*, vol. 5, 2022.

[19] V. Kaul, S. Enslin, and S. A. Gross, “History of artificial intelligence in medicine,” *Gastrointestinal Endoscopy*, vol. 92, pp. 807–812, 10 2020.

[20] C. Zhang and Y. Lu, “Study on artificial intelligence: The state of the art and future prospects,” *Journal of Industrial Information Integration*, vol. 23, p. 100224, 9 2021.

[21] N. Muthukrishnan, F. Maleki, K. Ovens, C. Reinhold, B. Forghani, and R. Forghani, “Brief history of artificial intelligence,” *Neuroimaging Clinics of North America*, vol. 30, pp. 393–399, 11 2020.

[22] C. Janiesch, P. Zschech, and K. Heinrich, “Machine learning and deep learning,” *Electronic Markets*, vol. 31, pp. 685–695, 9 2021.

[23] F. Maleki, K. Ovens, K. Najafian, B. Forghani, C. Reinhold, and R. Forghani, “Overview of machine learning part 1: Fundamentals and classic approaches,” *Neuroimaging Clinics of North America*, vol. 30, pp. e17–e32, 11 2020.

[24] D. H. Murphree, P. Puri, H. Shamim, S. A. Bezalel, L. A. Drage, M. Wang, M. R. Pittelkow, R. E. Carter, M. D. Davis, A. G. Bridges, A. R. Mangold, J. A. Yiannias, M. M. Tollefson, J. S. Lehman, A. Meves, C. C. Otley, O. Sokumbi, M. R. Hall, and N. Comfere, "Deep learning for dermatologists: Part i. fundamental concepts," *Journal of the American Academy of Dermatology*, 5 2020.

[25] S. Razavi, "Deep learning, explained: Fundamentals, explainability, and bridgeability to process-based modelling," *Environmental Modelling & Software*, vol. 144, p. 105159, 10 2021.

[26] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational Vision Science & Technology*, vol. 9, pp. 14–14, 1 2020.

[27] Y. Xu, Y. Zhou, P. Sekula, and L. Ding, "Machine learning in construction: From shallow to deep learning," *Developments in the Built Environment*, vol. 6, p. 100045, 5 2021.

[28] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 5 2021.

[29] A. Mathew, P. Amudha, and S. Sivakumari, "Deep learning techniques: An overview," *Advances in Intelligent Systems and Computing*, vol. 1141, pp. 599–608, 2021.

[30] W. T. Le, F. Maleki, F. P. Romero, R. Forghani, and S. Kadoury, "Overview of machine learning: Part 2: Deep learning for medical image analysis," *Neuroimaging Clinics of North America*, vol. 30, pp. 417–431, 11 2020.

[31] X. Wang, Y. Zhao, and F. Pourpanah, "Recent advances in deep learning," *International Journal of Machine Learning and Cybernetics 2020 11:4*, vol. 11, pp. 747–750, 2 2020.

[32] W. Buntine, "Machine learning after the deep learning revolution," *Frontiers of Computer Science 2020 14:6*, vol. 14, pp. 1–3, 5 2020.

[33] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nature Reviews Neuroscience 2020 21:6*, vol. 21, pp. 335–346, 4 2020.

[34] P. Cheridito, A. Jentzen, and F. Rossmannek, "Non-convergence of stochastic gradient descent in the training of deep neural networks," *Journal of Complexity*, vol. 64, p. 101540, 6 2021.

[35] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with tensorflow: A review:," *https://doi.org/10.3102/1076998619872761*, vol. 45, pp. 227–248, 9 2019.

[36] P. Netrapalli, "Stochastic gradient descent and its variants in machine learning," *Journal of the Indian Institute of Science 2019 99:2*, vol. 99, pp. 201–213, 2 2019.

[37] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 604–624, 2 2021.

[38] G. V. Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artificial Intelligence Review*, vol. 53, pp. 5929–5955, 12 2020.

[39] W. Liu, J. Liu, Z. Luo, H. Zhang, K. Gao, and J. Li, "Weakly supervised high spatial resolution land cover mapping based on self-training with weighted pseudo-labels," *International Journal of Applied Earth Observation and Geoinformation*, vol. 112, p. 102931, 2022.

[40] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, 2016.

[41] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* Los Alamitos, CA, USA: IEEE Computer Society, jun 2022, pp. 11 966–11 976.

[42] I. Lauriola, A. Lavelli, and F. Aiolli, “An introduction to deep learning in natural language processing: Models, techniques, and tools,” *Neurocomputing*, vol. 470, pp. 443–456, 1 2022.

[43] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, “A survey on vision transformer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[44] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, “Transformers in vision: A survey,” *ACM Computing Surveys (CSUR)*, 3 2021.

[45] A. Galassi, M. Lippi, and P. Torroni, “Attention in natural language processing,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 4291–4308, 10 2021.

[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 2017-December, pp. 5999–6009, 6 2017.

[47] F. Liu, X. Ren, Y. Liu, H. Wang, and X. Sun, “simNet: Stepwise image-topic merging network for generating detailed and comprehensive image captions,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.* Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 137–149.

[48] D. Qiu, B. Rothrock, T. Islam, A. K. Didier, V. Z. Sun, C. A. Mattmann, and M. Ono, “Scoti: Science captioning of terrain images for data prioritization and local image search,” *Planetary and Space Science*, vol. 188, p. 104943, 2020.

[49] Z. Lei, C. Zhou, S. Chen, Y. Huang, and X. Liu, “A sparse transformer-based approach for image captioning,” *IEEE Access*, vol. 8, pp. 213 437–213 446, 2020.

[50] J. Li, N. Xu, W. Nie, and S. Zhang, “Image captioning with multi-level similarity-guided semantic matching,” *Visual Informatics*, vol. 5, no. 4, pp. 41–48, 2021.

[51] X. Yang, J. Zhao, H. Zhang, C. Dai, L. Zhao, Z. Ji, and I. Ganchev, “Remote sensing image detection based on yolov4 improvements,” *IEEE Access*, vol. 10, pp. 95 527–95 538, 2022.

[52] L. Zhu, H. Shi, H. Wei, C. Wang, S. Shi, F. Zhang, R. Yan, Y. Liu, T. He, L. Wang, J. Cheng, H. Duan, H. Du, F. Meng, W. Zhao, X. Gu, L. Guo, Y. Ni, Y. He, T. Guan, and A. Han, “An accurate prediction of the origin for bone metastatic cancer using deep learning on digital pathological images,” *eBioMedicine*, vol. 87, p. 104426, 1 2023.

[53] M. A. Hassanien, V. K. Singh, D. Puig, and M. Abdel-Nasser, “Predicting breast tumor malignancy using deep convnext radiomics and quality-based score pooling in ultrasound sequences,” *Diagnostics*, vol. 12, no. 5, 2022.

[54] S. C. Hoo, H. Ibrahim, and S. A. Suandi, “Convfacenext: Lightweight networks for face recognition,” *Mathematics*, vol. 10, no. 19, 2022.

[55] Q. Xu, H. Deng, Z. Zhang, Y. Liu, X. Ruan, and G. Liu, “A convnext-based and feature enhancement anchor-free siamese network for visual tracking,” *Electronics*, vol. 11, no. 15, 2022.

[56] 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,” *Lecture Notes in Computer Science*, vol. 8693 LNCS, pp. 740–755, 2014.

[57] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 664–676, 2014.

[58] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” *Advances in neural information processing systems*, vol. 27, 2014.

[59] N. Benny Toomarian and J. Barhen, “Learning a trajectory using adjoint functions and teacher forcing,” *Neural Networks*, vol. 5, no. 3, pp. 473–484, 1992.

[60] M. Pratama, C. Za'in, E. Lughofer, E. Pardede, and D. A. Rahayu, "Scalable teacher forcing network for semi-supervised large scale data streams," *Information Sciences*, vol. 576, pp. 407–431, 2021.

[61] M. Sangiorgio and F. Dercole, "Robustness of lstm neural networks for multi-step forecasting of chaotic time series," *Chaos, Solitons & Fractals*, vol. 139, p. 110045, 2020.

[62] J. W. Chen, X. K. Sigalingging, J.-S. Leu, and J.-I. Takada, "Applying a hybrid sequential model to chinese sentence correction," *Symmetry*, vol. 12, no. 12, 2020.

[63] 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 on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, 2002, p. 311–318.

[64] I. Mandal and A. Dwivedi, "Deep learning algorithms for accurate prediction of image description for e-commerce industry," in *Data Management, Analytics and Innovation*, N. Sharma, A. Chakrabarti, and V. E. Balas, Eds. Singapore: Springer Singapore, 2020, pp. 401–418.

[65] M. A. Haidar and M. Rezagholizadeh, "Textkd-gan: Text generation using knowledge distillation and generative adversarial networks," in *Advances in Artificial Intelligence*, M.-J. Meurs and F. Rudzicz, Eds. Cham: Springer International Publishing, 2019, pp. 107–118.

[66] B. Mobasher and J. Cleland-Huang, "Recommender systems in requirements engineering," *AI Magazine*, vol. 32, pp. 81–89, 6 2011.

[67] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 39–46.

[68] Y. Wu, L. Liu, J. Bae, K.-H. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, and Q. Zhang, "Demystifying learning rate policies for high accuracy training of deep neural net-

works," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 1971–1980.

[69] R. Castro, I. Pineda, W. Lim, and M. E. Morocho-Cayamcela, "Deep learning approaches based on transformer architectures for image captioning tasks," *IEEE Access*, vol. 10, pp. 33 679–33 694, 2022.