Politecnico di Torino

POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Leveraging the Visual Capabilities of Transformers in Multimodal Machine Translation

Supervisor:

Prof. Luca CAGLIERO

Co-Supervisor:

Ph.D Lorenzo VAIANI

Candidate:

Luca VILLANI

Academic Year 2023-2024

Abstract

Machine translation (MT) has come a long way since **Deep Neural Networks (DNNs) arrived**. The introduction of *Transformer architecture*, with its flexible data handling, opened the door to a new field: **Multimodal Machine Translation (MMT)**. MMT aims to combine text with other information, like images, to improve translation accuracy. While MMT is a rapidly growing field, there are still challenges. One is the lack of data that combines different modalities with translations. Another is how to represent different data types effectively and then combine them in a way that captures the overall meaning.

This thesis proposes a new architecture using *three transformers*: one for the *text*, one for a *general image representation*, and one for *detecting objects in the image*. The goal is to see if using both general and specific image features **improves translation quality**. Additionally, this research focuses on a lightweight architecture compared to the current trend of using increasingly complex models. Experiments were conducted using two Transformer sizes ("Tiny" and "Small") for translating *English to German, French, and Czech*. The results show that the proposed approach works well for German and French, but for Czech, only the general image representation led to improvements so far.

Table of Contents

Chapter 1

Introduction

Artificial intelligence (AI) is a vast field that studies how *we can simulate human behaviours with machines*. One particularly exciting subfield is **Natural Language Processing (NLP)**, focused on *combining techniques and knowledge from linguistics and computer science* to find an optimal way to make understandable, interpretable, and even generate human language to machines. This opens doors for a variety of applications and tasks, with **Machine Translation (MT)** being a prime example of it. Machine translation utilizes NLP techniques to automatically translate a text corpora from one language to another. Early MT systems relied on *rule-based approaches*, but today's most effective methods employ sophisticated statistical and neural network models. These models are trained on massive amounts of bilingual data, allowing them to capture the subtleties of language and produce increasingly accurate translations to become more and more like those made by humans.

Machine translation, thanks to **Neural Network's flexibility** of processing different kinds of data, has been developed another field, the one of **Multimodal Machine Translation (MMT)**. This innovative approach goes beyond just words, incorporating different types of data like images, speech, etc. to create a richer understanding of what's being communicated. Researchers worldwide have been pushing the boundaries of MMT (with usage of visual data) for years, focusing on three key challenges: *identifying the most relevant visual information for translation* , *developing efficient methods to extract it* , and *seamlessly combining it with text processing*.

The Transformer, an innovative *encoder-decoder architecture* introduced by *"Attention is all you need" (Vaswani, et al, 2017)* [1], marked a turning point in both NLP and Computer Vision. Unlike previous models, it forgoes sequential processing. Instead, it analyzes all parts of an input sequence, be it text or image (in this

case called *Vision Transformer*), simultaneously. This is achieved through a clever mechanism called *"Attention"*. Attention allows the Transformer to understand how different elements within the sequence relate to each other, even if they're far apart. This ability to capture long-range dependencies is crucial for tasks like machine translation, where understanding the relationship between words across sentences is key, or image captioning, where identifying objects and their interactions is essential.

This thesis presents an approach, inspired by the previous research *"Adding Multimodal Capabilities to a Text-only Translation Model" (Vijayan, et al. 2024)* [2] and *"Tackling Ambiguity with Images: Improved Multimodal Machine Translation and Contrastive Evaluation" (Futeral, et al 2023)* [3] that leverages three distinct Transformer architectures which aim to increase the accuracy of translations by adding two types of visual information collected from the same image. While the sequence-to-sequence Transformer is necessary for text processing, the true core of this model lies in the **two visual transformers**. These pre-trained Transformer models act as the foundation, tasked with extracting visual features from the accompanying image (always provided as a caption). The first visual encoder focuses on **global features**, extracting information that describes the overall context of the scene. Adding another layer of detail, the second visual encoder functions as an **object detector** extracting **regional/local visual information**. This fine-grained analysis provides crucial information to disambiguate context-dependent terms and capture the subtleties of the visual information.

We analyzed this architecture by studying its behaviour and the *quality of translations* on multiple levels: A *text-only analysis* that acts as a baseline to verify the impact of visual features, an analysis for *each type of visual feature* isolated from the other (for assessing its individual impact) and finally the *joint analysis* of these two to understand the true potential of using two types of visual features along with the text. Along with this study, we also analyzed performance differences between a configuration **"Tiny"** and **"Small"** for the sequence-to-sequence transformer.

To thoroughly assess the proposed model's capabilities, we conducted training and testing across three distinct target languages: *German, French, and Czech.* The **Multi30k** [4] dataset served as the foundation, providing *30,000 English-Target Language pairs*, each accompanied by a reference image. Employing three languages allows for a comprehensive analysis of the model's effectiveness on diverse linguistic structures. Three test sets were proposed in the dataset, containing 1000 sentence-image pairs each, namely *2016, 2017 and 2018.*

Chapter 2

Deep Learning

The aim of this introductory chapter is to discuss the foundations of the entire **Artificial Intelligence (AI)** context and the technological steps made to reach the state of the art of the tools commonly used nowadays. Starting from the various building blocks of a **Neural Network (NN)** up to the **Convolutional Neural Networks (CNN)** which are essential in my research of Multimodal Architectures, I will analyze in detail each of the steps required to train efficiently a **Deep Neural Network (DNN)**.
I will not cover the entire scope of **Machine Learning (ML)** , as it is not entirely relevant for my research despite Deep Learning being a subset of it; the purpose is to illustrate the concepts and methodologies necessary to delve into the relevant characteristics of the ML fundamental for the understanding of my research.

2.1 Neural Networks

In order to understand in detail the functioning of a **DNN** we must necessarily analyze its functioning. Inspired by the functioning of the human brain and the progress in understanding it, two researchers *Warren McCulloch*, a neurophysiologist, and *Walter Pitts*, a mathematician, layed the groundwork that will lead to the development of the first **Artificial Neural Networks (ANN)** illustrated in their paper *A logical calculus of the ideas immanent in nervous activity* [5] in which they explore the capabilities of performing computation by a simplified model of neurons.
Frank Rosenblatt, inspired by the previously cited work, engineered and built the first hardware implementation of an NN, *Mark I Perceptron* [6] designed for the Image recognition task.

This architecture, represents the fundamental of the *DNN*, which is characterized

by having three or more layers, one for the input, one for the output and an *"hidden layer"*, which is what differentiates a DL architecture concerning a traditional ML.

The most relevant building blocks of Mark I Perceptron are organized in 3 layers.

- An array of 400 photocells arranged in a 20x20 grid, named "sensory units" (S-units), or "input retina". Each S-unit can connect to up to 40 A-units.
- A hidden layer of 512 perceptrons, named "association units" (A-units).
- An output layer of 8 perceptrons, named "response units" (R-units).

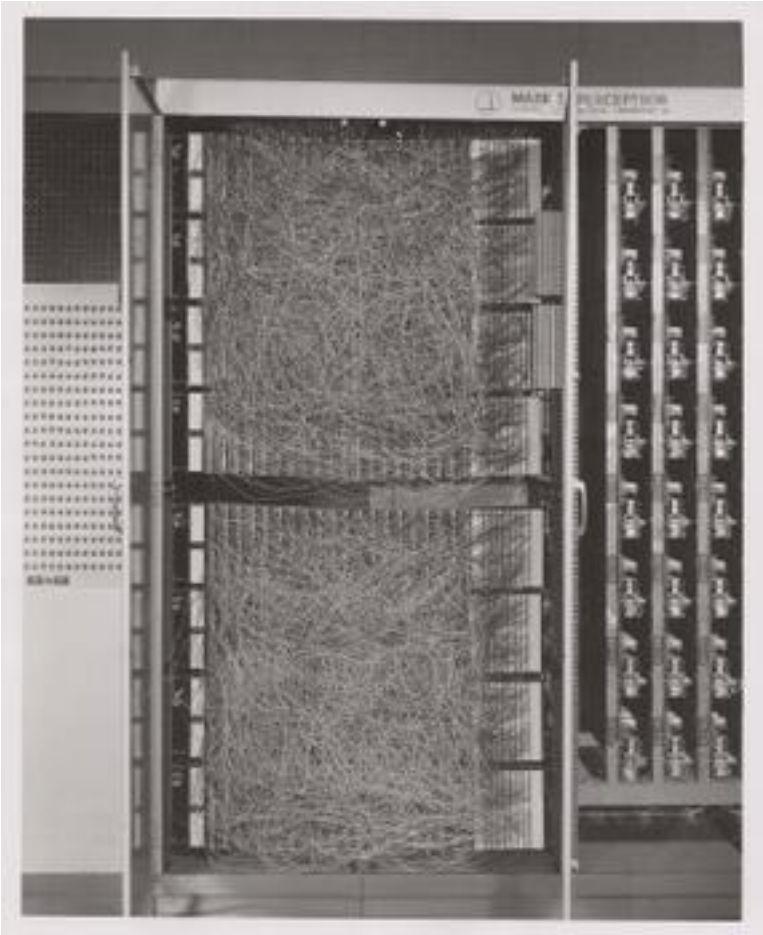

Figure 2.1: Mark I Perceptron - Source: Perceptron. (2024, June 18). In Wikipedia. https://en.wikipedia.org/wiki/Perceptron

2.1.1 Neurons

Going into detail, we want to analyze the functioning of a single artificial neuron to understand better how it is connected to its peers in the architecture of an ANN. Neurons are simple mathematical models, inspired by biological neurons, that take as input one or more numerical inputs (it could represent image pixel values, word embeddings, or any other data), multiply it by a **Weight** which represents the particular importance of that input to the neuron's activation; weighted inputs are then summed together. The weighted sum of the input is then passed to an **Activation Function, (a non-linear function)** (This aspect will be deeply analyzed in **Section 2.1.3**) which will determine if the neuron can switched on or not by a predefined **threshold**. The output of this function is then passed to other neurons in the network.

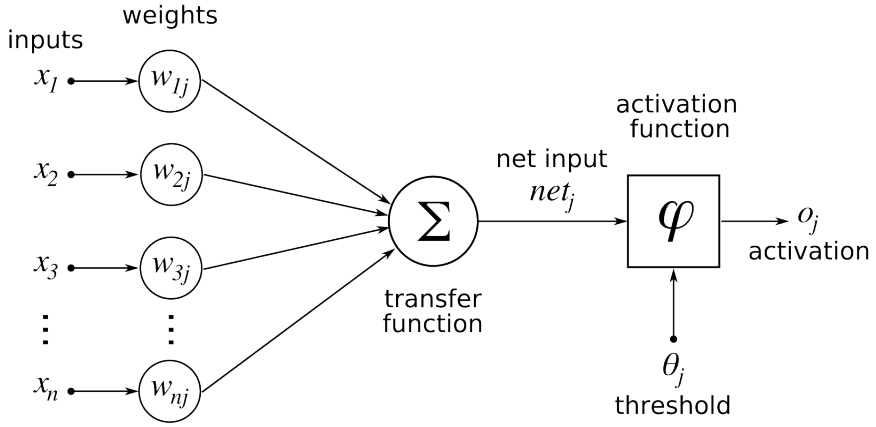

Figure 2.2: Functioning Schema of an Artificial Neuron - Source : [7]

This basic process is performed by all neurons in the NN, which through training learn to adjust weights and biases to perform activities such as detecting objects in an image, classifying images, text and more.

$$a = f\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{2.1}$$

- a (activation) : output of the neuron
- f : activation function
- N : number of inputs
- w_i : weight associated with each input (influence of each input on neuron's activation)

- x_i : individual input
- b : bias term, used to shift activation threshold

In the equation of a basic neuron, there is also included a **bias (b)** term which has been introduced to allow neurons to use different activation thresholds, helping NN to have an increased *generalization* capability. The generalization capability that stands for the ability of a neural network to train efficiently and identify increasingly complex patterns.

2.1.2 Multi-layers Perceptron

As stated at the beginning of this chapter, the first **ANN** was the *Mark I Perceptron*, nowadays what is commonly used by every DNN algorithm is a **MultiLayer Perceptron (MLP)** which is also called *Feedforward network*, because of the a-cyclic graph structure of the net, the information flows the input to the output, without any feedback connections.

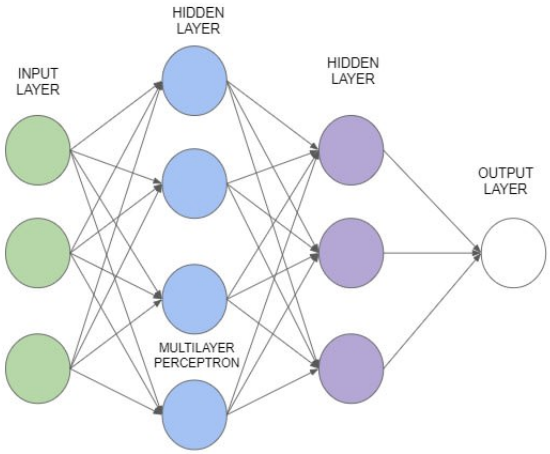

Figure 2.3: Simple graph of an MLP with only 1 Hidden Layer - Source: [8]

MLP leverage a multi-layer structure so it can exploit the efficiency of matrix-vector computations, essentially there are 3 kinds of layers:

- *Input layer* : It receives the input information, in different forms (text, image pixels, numbers and every data that can be numerable
- *Output layer* : It is the last layer of the network, and gives the result of the computations of the N middle layers

- *Hidden layer(s)* : Core of the network, they process the information received from the input layer and through very heavy computations extract patterns and other features from the data. Their number determines the *depth* of the network. In the case of a *Perceptron*, $n_{hidden} = 1$, in the case of a *DNN*, $n_{hidden} > 1$

2.1.3 Activation functions

In **Section 2.1.1** we talked about how the *Artificial Neuron* resembles the internal functioning mechanism of the one inside the brain, here's it is very important to dive deep into the concept of **activation**, which basically defines whether a weighted sum of input should be passed to the other neurons, namely *fire* or *activate* it.
Activation functions act as gatekeepers, introducing non-linearity in the network along with shaping the network's input into a specific range, a fundamental step to interpreting the network's output and make predictions. Now we're going to explore the most important:

- **Sigmoid/Logistic Function**: output a value between 0 and 1, suitable where output is a probability.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

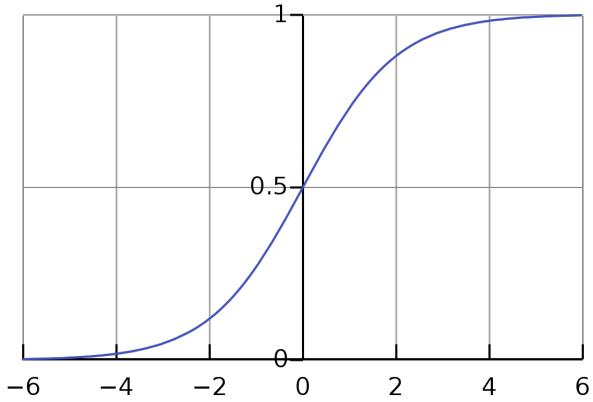

Figure 2.4: Sigmoid/Logistic Function - Source : [9]

- **Hyperbolic Tangent (Tanh)**: output a value between -1 and 1, computationally more expensive than Sigmoid

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.3}$$

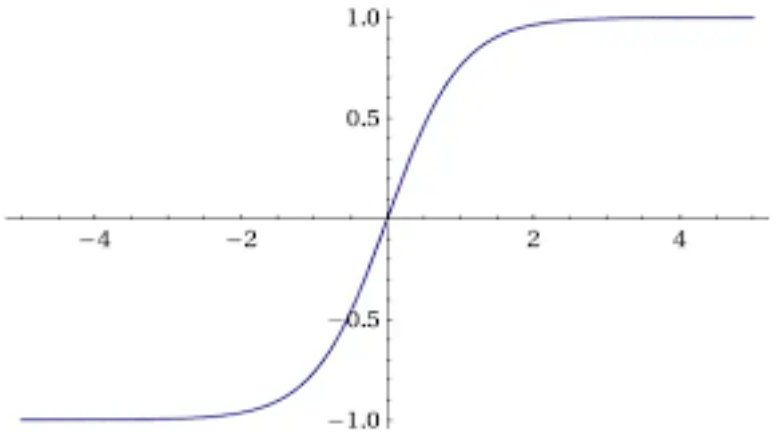

Figure 2.5: Hyperbolic Tangent Function - Source : [10]

- **Rectified Linear Unit (ReLU)**: most used in hidden layers, output the input if positive and 0 if negative, in this case, there is no scaling performed on the input.

$$f(z) = max(0, z) \tag{2.4}$$

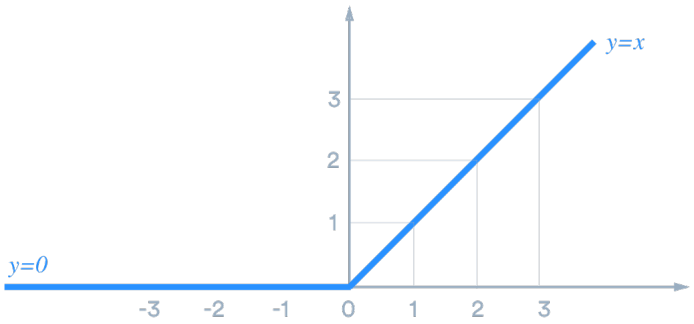

Figure 2.6: ReLU Activation Function - Source : [11]

- **Gaussian Error Linear Unit (GELU)**: combines a linear function with cumulative distribution function of standard normal distribution. Unlike ReLU, it has a smooth, non-monotonic transition around zero but with an increased computational cost.

$$gelu(z) = 0.5z(1 + erf(z/\sqrt{2})) \tag{2.5}$$

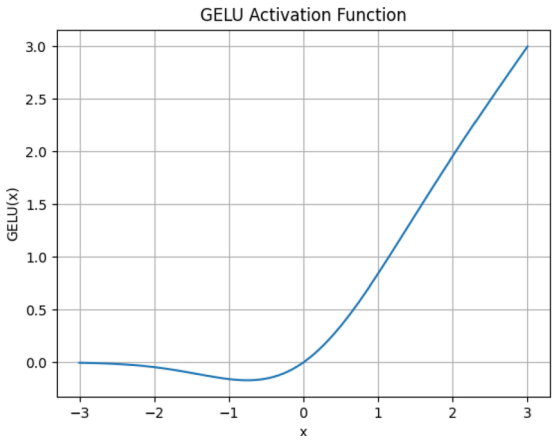

Figure 2.7: GELU Activation Function - Source : [12]

2.2 Training of a DNN

In the last section, we discussed the architecture and functioning of a typical ANN, but, what makes it relatable to ML and brings out its true potential is the possibility of adjusting the weights and biases of each neuron. In this section, we will discuss how the **training of a *DNN*** is done and how the network learns to identify patterns within the often huge amount of data we fed in.

2.2.1 Back-Propagation

We have already discussed the **forward pass**, data go into input neurons, and through a network-size dependent number of computations get out in the form of probability from the output neurons. We can now measure how precise the output is concerning our *ground truth*, this measure is called **Loss** and measures the error between the network's output and our data. This measure is then iterated backwards through the network using the mathematical tools of: **Partial Derivatives** and **Chain rule**.

Starting from the output layer, we compute the partial derivative of the Loss function for the activation function of that neuron, we then apply the *chain rule* to propagate backwards, in simpler terms, it helps to understand how previous layers contributed to error in the current layer so we can know how much each weight contributed to the overall error.

Now that we have all the information we need to adjust the weights we can apply another function called **Optimizer** to update the weight of each neuron or learned

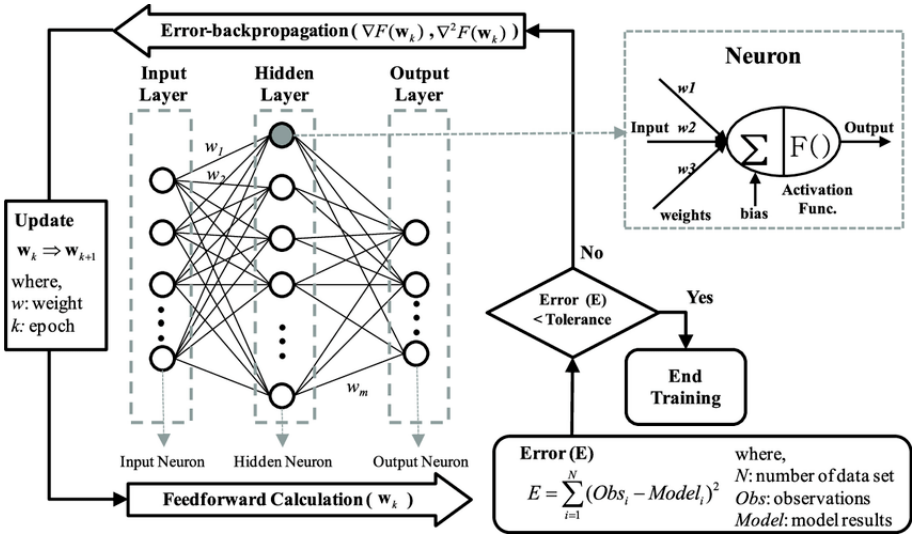

Figure 2.8: Training schema of a DNN - Source : [13]

parameter. This optimizer uses one of the most important *hyperparameters (tunable parameters)* of the whole DL context: **Learning rate (η)**. It controls the step size taken during weight updates, if chosen too large, it leads to faster learning but a possible overshoot of the minimum error, conversely, a smaller learning rate is more time-consuming but converges better to the minimum error.

This whole process:

1. Forward pass
2. Computing loss
3. Backward pass
4. Weights update (Optimizer step)

is repeated for a fixed number of steps (**Epochs**) in which the whole dataset is completely fed into the network, the objective is to reduce at minimum the loss to achieve the maximum accuracy of the network concerning the data we fed reaching a point in which there is nothing more to improve, so-called **Convergence**.
It usually means that the deeper and larger is the architecture, and the more are the data fed, the more the network will be accurate and capable of performing increasingly complex tasks. One of the goals of my study is to understand, through various training techniques, how to reach comparable results in the translation task with lighter architectures and smaller amounts of data.

2.2.2 Loss

Further exploiting the concept of **Loss function** (or *Cost function*), previously mentioned, and its contribution to the training phase and evaluation of a DNN. This measure comes from ML in which, for linear models, is a *convex function*, that will guarantee to find a global minimum (for convex domains) using a convex optimization algorithm.
For DL the situation is different since the Neural Networks introduce non-linearity, also some of the loss functions become non-convex and so losing the guarantee of finding a global minimum, in the optimization step the optimizer can get stuck in a local minimum that is not representative of the best performances.
There are various loss functions specific to the task the network has been designed to perform, some of the most important in the field are:

- **Mean Squared Error (MSE)**: the average square difference between predicted values and the ground truth, used mostly for regression tasks.

$$MSE = \frac{1}{n} * \sum_{i=1}^{N} (y_{pred_i} - y_{true_i})^2 \tag{2.6}$$

  Where:

  - n : number of samples
  - Σ : summation over all data points (from i to n)

- **Mean Absolute Error (MAE)**: the average absolute difference between predicted values and ground truth, used for regression, particularly with a high presence of outliers in the data.

$$MAE = \frac{1}{n} * \sum_{i=1}^{N} |y_{pred_i} - y_{true_i}| \tag{2.7}$$

- **Cross Entropy Loss (Log Loss)**: Measures difference between probability distributions of predicted values and ground truth. Specifically designed for classification tasks, in which the network computes different probabilities from different categories. It requires probabilities as the network's outputs

$$H(p, q) = -\sum_{i=1}^{C} q(y_i) * log(p(y_i)) \tag{2.8}$$

  Where:

- Σ : summation over all possible class labels (i from 1 to C, where C is the number of classes)
- $q(y_i)$: the true probability of correct class for data point i
- $p(y_i)$: network predicted probability for class i for data point i

- **Kullback-Leibler Divergence Loss (KLDiv Loss)**: like cross-entropy it measures the difference between two probability distributions, in particular how much one diverges from the other.

$$KL(P||Q) = \sum_{i=1}^{N} P(y_i) * log(\frac{P(y_i)}{Q(y_i)}) \tag{2.9}$$

Where:

- Σ : summation over all possible events (i from 1 to n, n number of events)
- $P(y_i)$: represents the probability of event y_i under distribution P
- $Q(y_i)$: represents the probability of event y_i under distribution Q

2.2.3 Optimizer

The objective of the training of a Neural Network is to minimize the *Loss function* to its global minimum, this is possible with the iterative approach proposed earlier and to adjust all the internal weights and biases of the network come in our help a set of Optimization techniques, namely **Optimizers**.
Most of them build upon the mathematical concept of **Gradient Descend (GD)**, an iterative approach for which the derivative (*Gradient*) of the loss function is computed with respect to each parameter of the network, then, guided by the learning rate, a small step in the negative direction of the gradient is taken towards reaching the minimum.

$$update_i = parameter_i - \eta * \nabla \tag{2.10}$$

This vanilla approach has quite some limitations, most important of all, it can get stuck in local minima. For this reason more sophisticated optimizers have been developed, here I will show you the most important ones:

- **Stochastic Gradient Descend (SGD)** : along with the gradient descent, it updates parameters based on a single data point (or small mini-batch) at a time.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla L(\mathbf{w}_t, \mathbf{x}_i, y_i) \tag{2.11}$$

Where:

- $\mathbf{w}_t$: weight vector at iteration t
- $\mathbf{w}_{t+1}$: updated weight vector
- η : learning rate
- $\nabla L(\mathbf{w}_t, \mathbf{x}_i, y_i)$: gradient of loss function L with respect to weights at iteration t, evaluated in a single data point (input x_i , target output y_i)

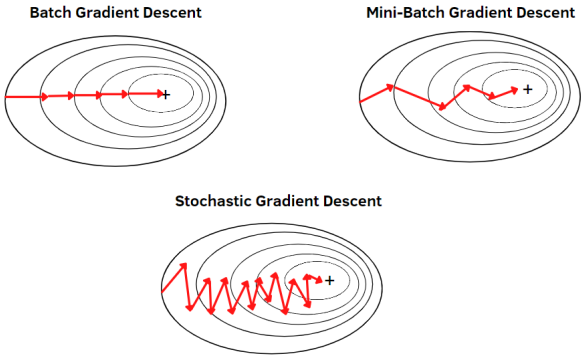

Figure 2.9: Visual representation of SGD variations - Source : [14]

- **RMSprop (Root Mean Squared Propagation)** : adapts the learning rate for each parameter based on its recent square gradients, addressing issues with gradients of varying magnitudes.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta}{\sqrt{\overline{\mathbf{g}_t^2} + \epsilon}} \mathbf{g}_t \tag{2.12}$$

Where:

- $\mathbf{g}_t$: gradient of loss function at iteration t
- $\overline{\mathbf{g}_t^2}$: exponentially decaying average of squared gradients up to iteration t, computed as:

$$\overline{\mathbf{g}_t^2} = \beta \overline{\mathbf{g}_{t-1}^2} + (1 - \beta)(\mathbf{g}_t \odot \mathbf{g}_t) \tag{2.13}$$

  with β as a hyperparameter controlling the decay rate (usually 0.9)
- ϵ : small value for numerical stability (usually between 1e-7 to 1e-9)

- **Momentum** : it introduces a velocity term, which captures direction and magnitude of past updates, accumulating gradients with a decay factor

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \beta(\mathbf{v}_t + \eta \nabla L(\mathbf{w}_t, \mathbf{x}_i, y_i)) \tag{2.14}$$

where $\mathbf{v}_t$ is the velocity term at iteration t

- **Adaptive Moment Estimation (Adam)** : combining the features of momentum and SGD, it estimates the first and second moments of the gradients using the exponentially decaying averages with correction factors to account for bias at the initial steps.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \odot \hat{\mathbf{m}}_t \tag{2.15}$$

  Where $\hat{\mathbf{m}}_t$ is the bias-corrected estimate of the first moment (mean) of gradients and $\hat{\mathbf{v}}_t$ is the second moment (variance) of gradients.
  There is also a slight variant called **AdamW** that incorporates the concept of *Weight decay*, a regularization technique that penalizes large weight values, this helps to prevent overfitting and improve generalization.

$$\mathbf{w}_{t+1} = (1 - \lambda)\mathbf{w}_t - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \odot \hat{\mathbf{m}}_t \tag{2.16}$$

  where $(1 - \lambda)$ is the weight decay term, applied to the weights before Adam update.

2.3 Convolutional neural networks

The nature of my research involves the use of multimodal data, specifically text and images. It is therefore natural to also talk about another type of neural network, specific to processing images, the so-called **Convolutional Neural Network (CNN) or ConvNet**. Exactly how the neural network was designed taking inspiration from the functioning of the human brain. The convolutional neural network was developed taking inspiration from the visual cortex, responsible for processing vision. The main characteristic in the biological counterpart is that some neurons are activated only in specific areas of the visual field; CNNs capture this behavior through layers of filters that scan the image extrapolating specific features. Like, CNNs are composed of Input, Output and FCN layers, along with two layers specific to these architectures, which we will delve into in the next sections:

- **Convolutional Layers** : applies filters (also called *Kernels* to the image through a sliding window, the filter extracts shapes and other features building a feature map
- **Pooling Layers** : Used to reduce the dimensionality of the data from convolutional layers (image processing tends to be high-dimensional), they help to reduce overfitting.

All the new layers inserted, working with the images, are distributed along the three spatial dimensions: *width, height and depth.* CNNs are commonly used to perform image-related tasks such as *Image Classification*, *Object Detection*, *Image Segmentation* and more.

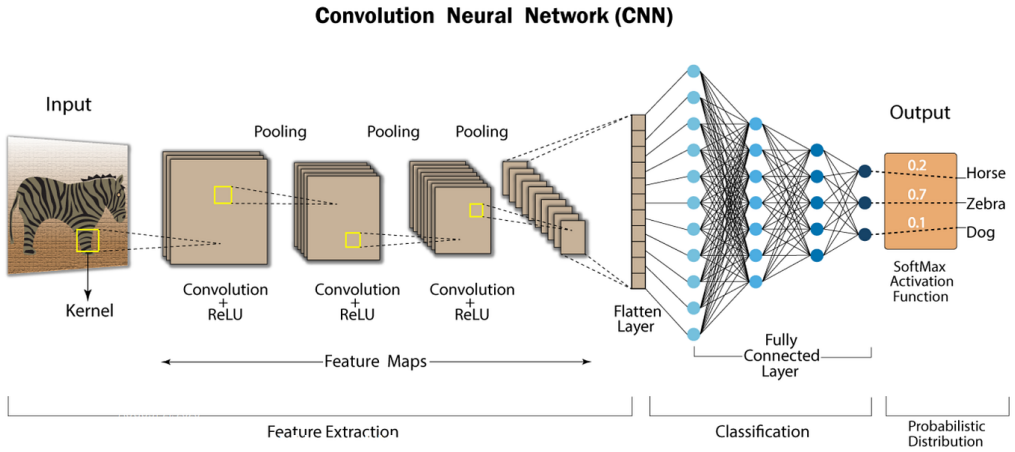

Figure 2.10: Simple CNN architecture for Image Classification - Source : [15]

2.3.1 Convolutional Layers

The **Convolutional Layer** is the core component for this architecture, it allows the network to extract features from the input images that can be processed to perform classification or, as the case of my research, combined with other kinds of data for different multimodal applications. They are based on the mathematical operation of *Convolution* :

$$Output[i,j] = \sum_{m,n}(Input[m,n] \cdot Filter[i-m, j-n]) + Bias \tag{2.17}$$

Where:

- **Output[i, j]**: represent the element at position (i, j) on the feature map.
- **Input[m,n]**: represent the element at position (m,n) of the input image.
- **Filter[i - m, j - n]**: represent the corresponding element in the sliding filter.

This operation is performed through a *Sliding window* that extrapolates features by performing element-wise multiplication with the input images, this is also called *Convolving*, the result of this operation is a 2-dimensional **Feature map** (or *Activation Map*) that stores the response of the filter at every spatial position. Since the filter is a learnable parameter, through each training step the network will learn to recognize which features are most important/useful to perform the task for which it was designed.
Along with the filter (or *Kernel*) size, that determines the so called **Receptive Field** of the neuron, another parameter is fundamental to operating these calculations,

the **Stride** which determines the space of moving of the filer, based on how large this parameter is, then we'll have a smaller feature map.
A very important technique related to this layer is the *Shared weights*, the image is analyzed with the same filter (and so retain weights and biases) in each region of it, this reduces the computational overhead of learning new parameters, and helps with overfitting.
The output of this layer is then passed through a non-linear activation function (such as $ReLU$), for the same reason that is used in the NN's classical layers, allowing to extract more and more complex patterns in the data.

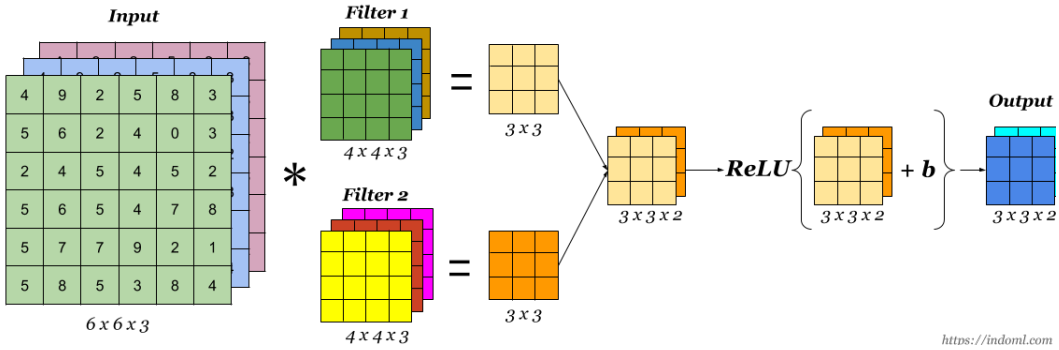

Figure 2.11: Convolutional Layer - Source : [16]

2.3.2 Pooling Layers

Along with the Convolutional Layer, in a CNN architecture, there is another essential component which is the **Pooling Layer**. It comes after every convolution and it is used in order to *reduce computational cost* by reducing dimensionality in the data and *control overfitting*, smaller data size can prevent the network from memorizing unnecessary or redundant data improving the generalizability.
Another fundamental characteristic is that Pooling Layers perform a feature summarization, retaining the invariant features extracted regardless of their exact location in the feature map.
Like the convolution operation, the pooling layer divides the input feature map into a grid, for each element in the grid is applied the pooling function and the result is a feature map with reduced width and height but with the same depth.
Since pooling is also a windowed function, the two main parameters are **Size** and **Stride** which works exactly as the previous layer. (Stride = 1 means no overlap between regions, while if stride >= 2 the resulting feature map will be smaller).
There are two kinds of Pooling functions:

- **Max Pooling**: the function takes the maximum value in the pooling window, emphasizing the most prominent features in the region.

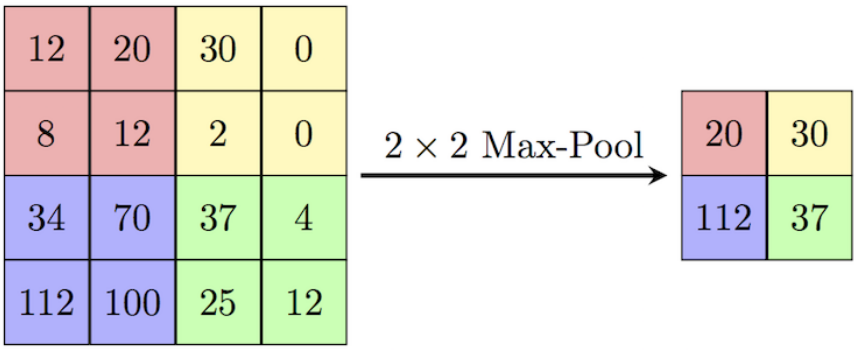

Figure 2.12: Max pooling example with a 2x2 grid and stride of 2 - Source : [17]

- **Average Pooling**: the function takes the average value in the pooling window, providing a smoother representation of the features, and capturing the overall characteristics.

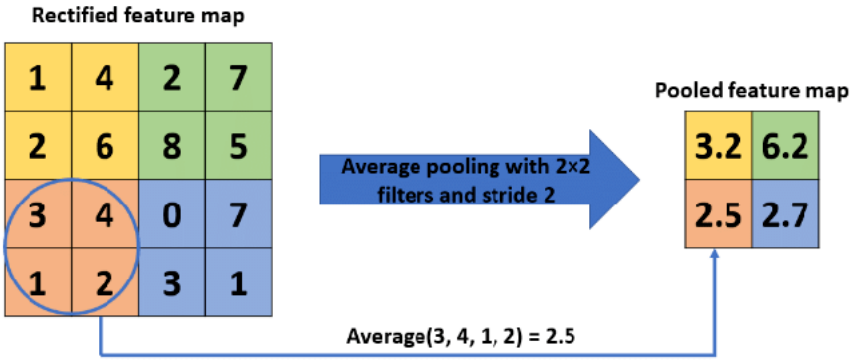

Figure 2.13: Average pooling example with a 2x2 grid and stride of 2 - Source : [18]

2.4 Natural Language Processing

The human language is one of the most complex and challenging research areas of the entire spectrum of knowledge. The field of NLP was born from the need to **combine** the automatic processing of a computer with the complexity and

articulation of human language (verbal and otherwise), making use in the current era of innovations in the field of AIs. In recent years, the increase in available computational power has revolutionized this field of application, making it possible to apply Neural Networks, and very large data sets, where previously only statistical models with small to no amount of data could be used.

2.4.1 Text Encoding

As previously mentioned in **Section 2.1.1**, NN's can accept as input only numerical values, for its nature of a non-linear model. Crucially, the network's ability to differentiate between words within large datasets (often containing thousands of words) necessitates the application of *text-to-number conversion techniques*. The objective is to obtain *multidimensional vector representations* of text, also known as **Embeddings**, suitable to extract semantic and syntactical relationships between words that can be further processed in down-stream tasks such as *Sentiment Analysis*, *Text Classification* and *Neural Machine Translation*.

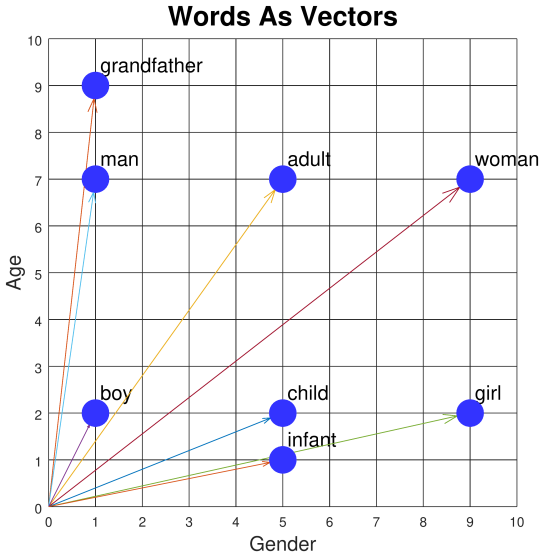

Figure 2.14: An Example of vectorized words in 2D Vectors with two features, Age and Gender - Source : [19]

Tokenization

Tokenization is the fundamental step to process text to be suitable for an NN, it consists of subdividing the text into smaller *sub-units* called **Tokens** (tokens can be words, subwords and also characters) and assigning to each of them a number,

each unique couple (text-unit, number) is stored inside a **Vocabulary**. There are plenty of tokenization strategies, these are the ones applied in this research:

- **Word-Based Tokenization**: The most simple approach, the text is split into individual words based on *whitespace* (spaces, tabs and other separators. One of the disadvantages is that it struggles with *Out-of-Vocabulary (OOV)* words.
- **Byte Pair Encoding (BPE)** [**20**] : This technique splits the text into the most frequent pairs of characters or subwords. It creates a dynamic vocabulary that can adapt to unseen words during the training.
- **SentencePiece (SPM)** [**21**] : Based on BPE, offers additional features like *multilingual support* and *vocabulary* control.

Chapter 3

Transformers

This chapter will develop an explanation of the family of architectures that has revolutionized the entire field of Artificial Intelligence but was primarily born to exploit the sequential data in NLP tasks: the **Transformer**; leading to the development of all the technologies most used nowadays.

3.1 Transformers

To initiate a discussion on the current best practices within the *Machine Translation* task (a prominent area of NLP) it is essential to first establish the fundamental architecture underpinning this field: **The Transformer**, published by Google in 2017 in the paper *"Attention is all you need"* [1]. This architecture has demonstrably achieved *state-of-the-art performance* across a wide range of NLP tasks, and, as we will explore further, its influence extends beyond NLP, impacting other domains within Deep Learning (DL) such as computer vision and audio processing.

Predating the Transformer's introduction in 2017, a significant body of research focused on a specific type of neural network architecture known as *Recurrent Neural Networks (RNNs)*, with a particular emphasis on *Long Short-Term Memory (LSTM)* networks. LSTMs are characterized by their gating mechanisms, which empower them to learn **long-range dependencies within sequential data**. This capability makes them particularly well-suited for tasks that involve temporal information, such as speech recognition and machine translation.

While a detailed examination of the Long Short-Term Memory (LSTM) architecture falls outside the scope of this present investigation, a comprehensive exploration of all Transformer architecture components is essential. This in-depth understanding is necessary to facilitate the comprehension of the architectures proposed in this

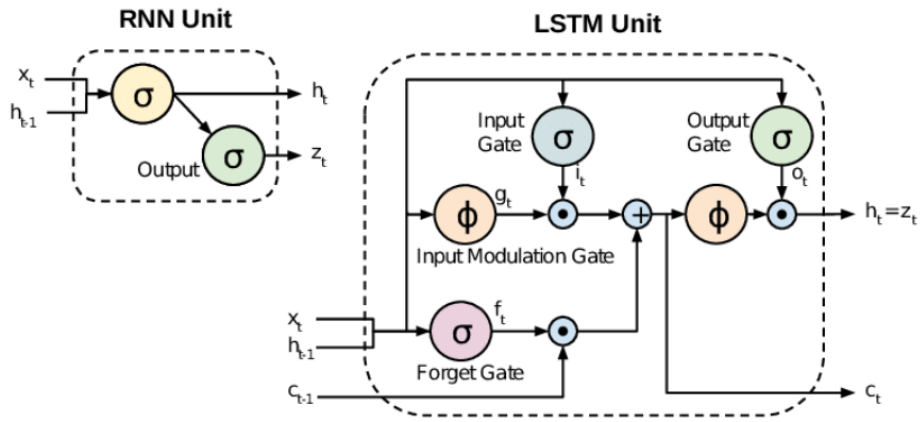

Figure 3.1: A quick overview of RNN and LSTM architectures - Source : [22]

research.

3.1.1 Architecture

The paper's title directly identifies the **attention mechanism** as the most critical component of the proposed model. Notably, the underlying architecture leverages the encoder-decoder strategy, a concept introduced by Google's 2014 LSTM **Seq2Seq model** [23]. To fully grasp the functionality of this architecture, we will delve into the various steps that follow the tokenization process.

Word Embeddings and Positional Encoding

The initial processing stage, preceding those depicted in **(Figure 3.3)**, involves **Input and Output Embedding** (respectively, the first for the *Encoder* and the second for the *Decoder*. This step maps individual words within the input sequence to dense, low-dimensional vectors. This process, known as **Word Embedding**, aims to capture the semantic meaning and relationships between words **(Section 3.1.1)**. However, unlike traditional sequential models, the Transformer architecture lacks inherent mechanisms to capture the order of words within a sequence, which is fundamental to capturing relationships within a sentence. To address this limitation, the Transformer employs an additional technique called **Positional Encoding**. This process injects information about the relative position of each word within the sequence into the embedded vectors. The combination of word embeddings and positional encodings provides the Transformer with the necessary information to understand the context and meaning of the input sentence.

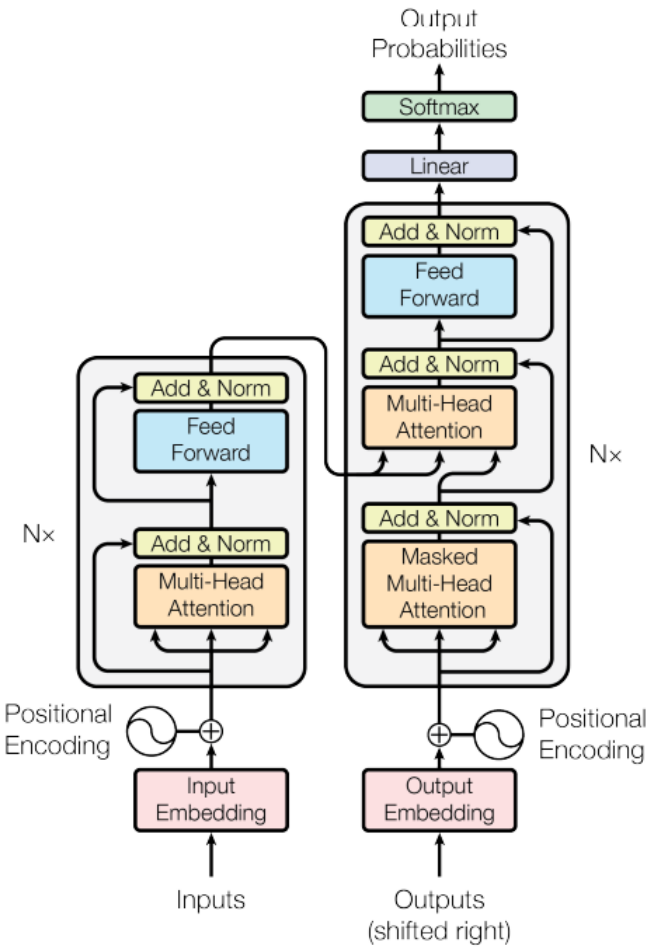

Figure 3.2: Transformer architecture from "Attention is all you need" [1]

Positional Encoding:

$$PE_{(pos,i)} = \begin{cases} \sin\left(\frac{pos}{10000^{i/d}}\right) & \text{if i is even} \\ \cos\left(\frac{pos}{10000^{(i-1)/d}}\right) & \text{if i is odd} \end{cases} \tag{3.1}$$

Where:

- $PE_{(pos,i)}$ is the positional encoding for position *pos* and dimension i of the embedding vector.
- *pos* is the position of the word within the sequence (sequence starts from 0).
- i is the index of specific dimension within the embedding vector (e.g : for a 256-dimensional vector, i would range from 0 t 255)

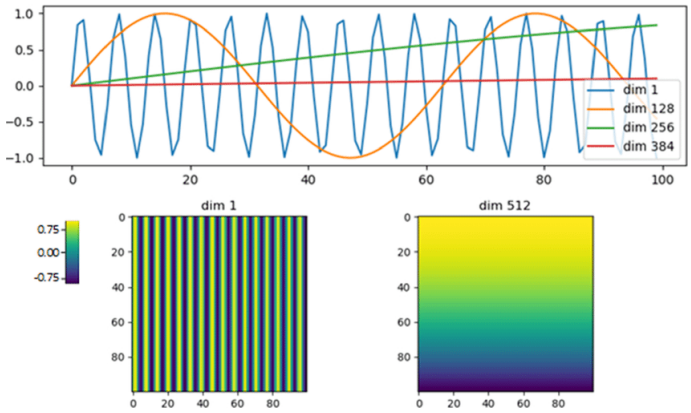

Figure 3.3: Positional Encoding of sentences of dim 1,128, 256 and 384 - Source : [24]

- d is the **Dimension of the word embedding vector** (i.e fixed number of features extracted from a word).
- 10.000 is a scaling factor, in the paper is suggested as this but can be hyper-parameter tuned.

Positional encoding's usage of *sin* and *cos* allows the positional information to have different frequencies across different dimensions, aiding the model in distinguishing between close and distant positions.

Attention Mechanism

The seminal work "Attention Is All You Need" by Vaswani et al. (2017) introduced a revolutionary concept to the Transformer architecture: the self-attention mechanism. This mechanism departs from traditional methods and empowers the model to discern the most critical components within a sentence. By doing so, the Transformer gains the ability to comprehend the diverse contextual nuances embedded within the sentence.

Self-Attention

The core principle underlying **Self-Attention** resides in its ability to compare each element (word) within a sentence to every other element. This comprehensive comparison process culminates in the creation of a matrix that encapsulates the relevance scores between all elements.
To facilitate this intricate comparison, each element in the vector undergoes a linear transformation via a *fully connected layer (FCL)*. This transformation results in

the creation of three distinct vectors: *query (Q), key (K), and value (V)*. Each of these vectors plays a specific and crucial role:

- **Query (Q) Vectors**: These vectors represent the focal point of attention for a particular element. They essentially embody the model's specific "question" regarding the element's significance within the context of the sentence.
- **Key (K) Vectors**: These vectors embody the information content held by each element. They function as a response to the query vectors, conveying the inherent meaning and relevance of each element.
- **Value (V) Vectors**: These vectors represent the current content associated with each element. They essentially capture the raw information that the model can potentially leverage based on the attention scores determined through the query-key interactions.

The attention scores are then computed using the formula of **Scaled do**:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.2}$$

To ensure the stability of gradients during the training process, the query (Q) and key (K) vectors undergo normalization. This normalization is typically achieved by dividing them by the square root of the embedding size. This scaling technique mitigates the vanishing or exploding gradient problem, which can hinder the training process in deep neural networks.
Following the normalization step, a **softmax function** is applied. The softmax function transforms the resulting vectors into a probability distribution. This distribution ranges from 0 to 1, where higher scores correspond to elements within the sequence that are deemed more relevant by the model based on their interaction with the query vector. In essence, the softmax function assigns a weight (*attention score*) to each element, indicating the degree to which it contributes to the final attention output.

Multi-Head Attention

In **Figure 3.3** is proposed a **Multi-Head Attention** Block which is a direct evolution of the *Self-Attention mechanism*, instead of performing the scaled-dot product only one time, it is performed h (heads) times in parallel using different learned linear projection. The heads are then concatenated and projected once again shown in the following figure taken from [1] :

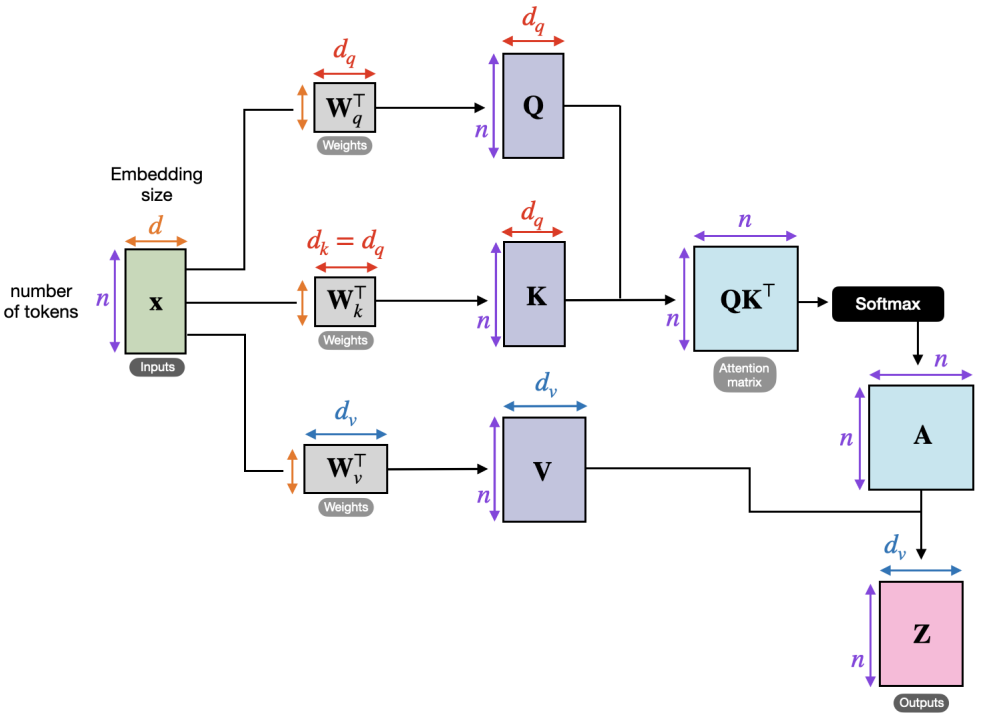

Figure 3.4: Self-Attention Mechanism Schematics - Source : [25]

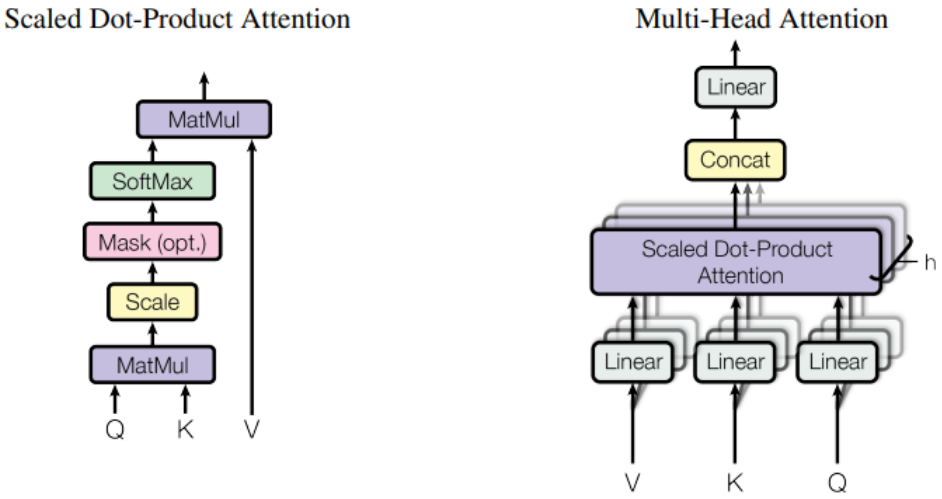

Figure 3.5: Multi-Head Attention Mechanism Schematics - Source : [26]

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^o$$
$$Where : \text{head}_i = \text{Attention}(QW_Q^i, KW_K^i, VW_V^i) \quad (3.3)$$

This concurrent processing architecture facilitates the acquisition of *heterogeneous representations* of the input sequence. This, in turn, yields a more intricate and multifaceted comprehension of the data. Additionally, it fosters enhanced generalizability and potentially augmented efficiency.

Position-wise FFN

Within both the encoder and decoder layers of a transformer architecture, a *Position-Wise FFN* block is incorporated. This FFN block functions similarly to a standard FFN, applying identical non-linear transformations – typically two with a ReLU activation function interposed – to each element within a sequence. However, these transformations are applied **independently**, preserving positional information.
Importantly, while the general structure of the transformations remains consistent across positions, the underlying parameters differ between layers. This architecture leverages a hidden layer dimension significantly exceeding the embedding dimension, enabling the network to capture intricate interactions between elements in the sequence.

$$\text{FFN(x)} = max(0, xW_1 + b_1)W_2 + b_2 \tag{3.4}$$

Encoder and Decoder

Having examined the innovative self-attention mechanism introduced in the "Attention is all you need" paper [21], we now turn our attention to the transformer's **encoder-decoder architecture**. This structure was fundamentally designed to facilitate the transformation of an input sequence into a compressed representation. Subsequently, the decoder leverages this representation, through the employment of a *cross-attention block*, to generate the corresponding output sequence. This architectural design empowers transformers with the remarkable capability of effectively modelling long-range dependencies within sequential data.

The encoder is comprised of a sequential stack of identical encoder layers. This implies that the input data undergoes processing by each layer in succession, progressing from $encoder_0$ to $encoder_n$, where n represents the total number of layers. Internally, each encoder layer incorporates two sub-modules: a *self-attention mechanism* and a *position-wise feed-forward network (FFN)*, as previously described.

Following the application of each sub-module, an **element-wise addition operation** termed a **residual connection**, is employed. This ensures that the output of each layer retains the information from the previous layer. Finally, a normalization layer is applied to rescale the activations within the network. This step contributes

to stabilizing the training process and mitigating overfitting.

The decoder leverages the same fundamental components as the encoder layer but arranges them in a distinct configuration. Unlike the encoder's initial Multi-Head Attention mechanism, the decoder employs a variant termed **Masked Multi-Head Attention**. As the name implies, this mechanism masks each subsequent token within the output sequence. This masking strategy compels the transformer to generate the output sequence one token at a time, ensuring an *autoregressive* fashion. Following the Masked Multi-Head Attention, a *Cross-Attention module* is incorporated. This module utilizes the encoded representation generated by the encoder's stacked layers as both its key and value vectors. This design element fosters alignment between the generated output sequence and the semantic understanding captured by the encoder.

3.2 Vision Transformers

Our exploration has thus far focused on understanding how the transformer architecture processes textual features, as this was its original design intent. However, the scope of this research extends into the realm of **Multimodality**. Therefore, it becomes necessary to introduce a distinct architecture derived from transformers: **the Vision Transformer (ViT)**, introduced in 2020 , by Google researcher in the paper *"An Image is worth 16x16 words"* [27]. This architecture prioritizes the processing of image features. Notably, it has demonstrated superior performance compared to traditional *CNNs* commonly employed in the field of Computer Vision.

While transformers typically operate on inherently sequential data like text, vision transformers (ViTs) adapt this architecture to process images, which lack a natural sequential order. To address this challenge, ViTs employ a preprocessing step that subdivides the input image into fixed-size square patches. These patches are then flattened into vectors and subsequently projected using a linear transformation. Additionally, a special token, denoted as "[CLS]", is prepended to the sequence at position zero. This token serves a crucial role, as it embodies the class of the image that the network aims to predict. It's important to note that ViTs do not innately capture positional information from the processed image patches. To compensate for this limitation, the model incorporates a positional encoding mechanism before feeding the data into the encoder layers. This encoding process injects information about the relative or absolute positions of the patches within the original image, enabling the ViT to learn relationships between these elements.

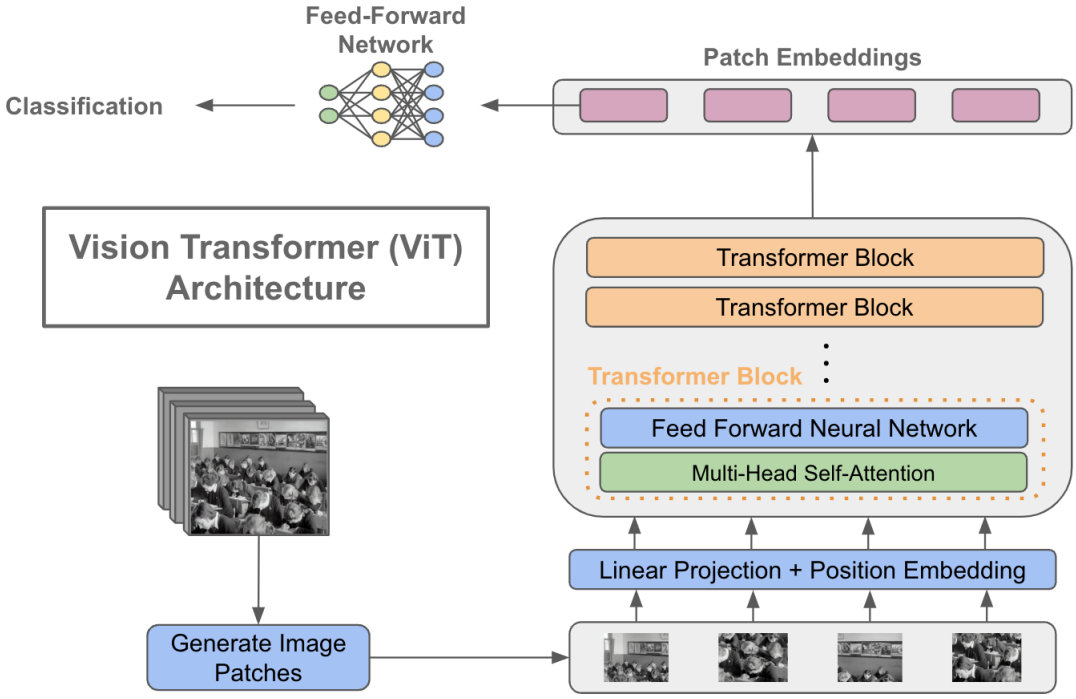

Figure 3.6: Overall ViT Architecture - Source : [28]

Within the context of this research, our primary interest lies not in image classification but rather in leveraging the ViT's capability to generate informative encodings of the image data. These encodings, ideally, should exhibit compatibility with text encodings, facilitating their integration within various multimodal settings.

3.2.1 Object Detector

The emergence of Vision Transformers has spurred a paradigm shift in computer vision tasks, particularly in the domain of Object Detection (OD). Traditionally dominated by Convolutional Neural Networks , OD now sees transformer-based architectures achieving competitive, and often surpassing, performance.

Object Detection: Core Functionality and Challenges

Object Detection is a fundamental computer vision task that aims to analyze a digital image and identify two key aspects of objects present within it:

- **Classification:** Recognizing the type of object in the image. This could involve classifying objects into pre-defined categories like pedestrians, vehicles, animals, or specific objects like furniture or tools.

- **Localization:** Accurately pinpointing the location of each detected object within the image. This is typically achieved by drawing a bounding box around the object's extent in the image frame.

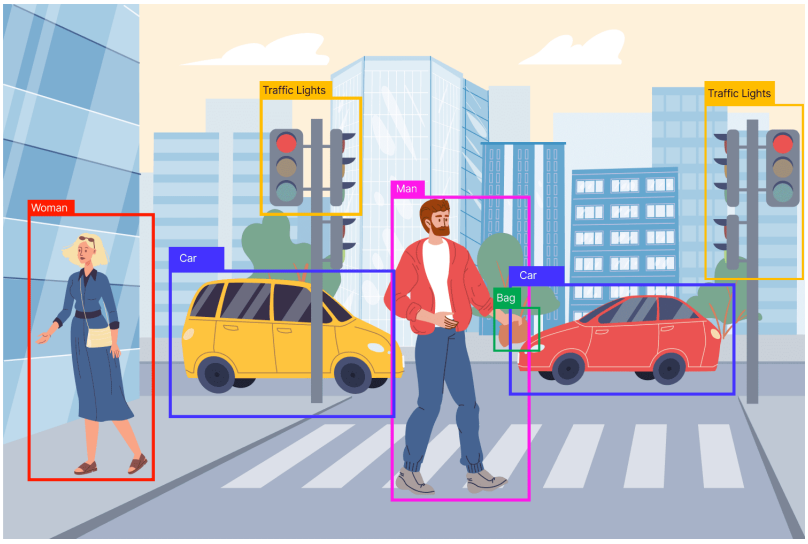

Figure 3.7: Example of Object Detection - Source : [29]

DETR

The year 2020 witnessed a significant breakthrough in object detection with the introduction of the DETR (DEtection TRansformer) architecture by researchers at Facebook AI [30]. DETR deviates from traditional CNN-only object detection frameworks by incorporating a transformer-based approach, leading to several key distinctions:

- **Feature Extraction Backbone:** While DETR retains a CNN as its foundation, its role is relegated to feature extraction. This extracted feature map serves as the input for the subsequent transformer architecture.
- **Transformer for Contextual Analysis:** The core of DETR lies in its encoder-decoder structure. The encoder leverages the self-attention mechanism to analyze relationships between different image regions. This enables the model to capture the global context within the image, crucial for accurate object detection.
- **Object Query-based Decoding:** Unlike classical Vision Transformers (ViT), DETR introduces a novel approach in the decoder stage. It employs a predefined set of learnable object queries. These queries act as placeholders for

the number of objects the model is expected to predict. The decoder then utilizes the self-attention mechanism to associate these queries with specific image regions containing the detected objects.

- **Prediction Head and Directed Set Prediction:** After exiting the decoder, the object queries are fed into the prediction head. This head extracts the confidence scores, bounding boxes, and class labels for each detected object. Notably, DETR formulates the task as a directed set prediction problem. This strategy allows the model to predict all objects in an image simultaneously, leveraging the previously captured global context.

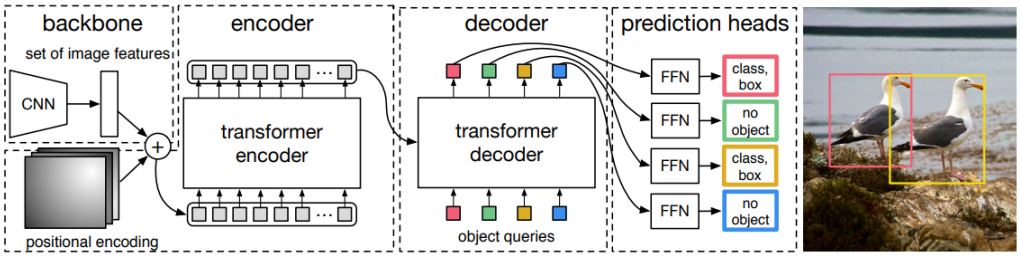

Figure 3.8: DETR Schematics - Source [30]

3.3 Multimodal Architectures for Combining Visual and Textual Features

Following the exploration of the core principles and visual applications of Transformer architectures, this section delves into two specific models employed within this research. These models exemplify the expanding field of text-image multimodality by addressing distinct task types.

The first model under consideration is **CLIP** [31], which stands for *Contrastive Language-Image Pre-Training*. The second model, **MDETR** [32], refers to *Modulated DEtection TRansformer*.

Contrastive Language-Image Pre-Training (CLIP)

Authored by *OpenAI* in their 2021 publication *"Learning Transferable Visual Models From Natural Language Supervision"* [31], the CLIP model presents a novel multimodal architecture. Its core objective lies in establishing a **unified embedding space** for both textual and visual data. This shared space empowers CLIP to address two key multimodal tasks, ultimately enriching the knowledge base for further advancements in multimodal learning.

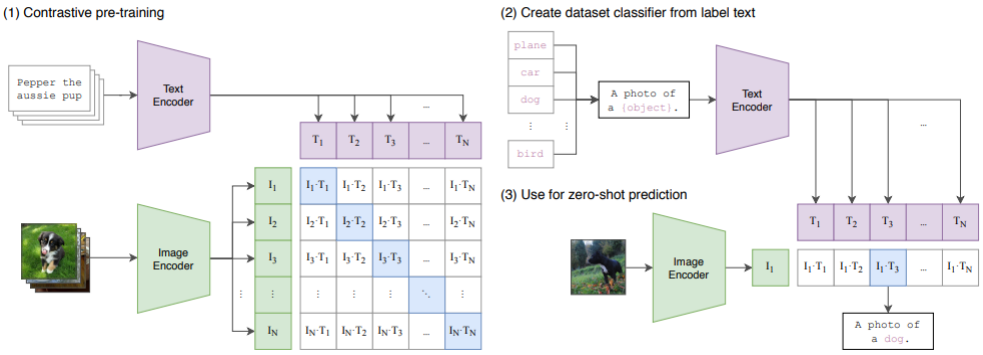

Figure 3.9: Training and zero-shot prediction done by CLIP model - Source: [31]

The CLIP model architecture is characterized by a dual-encoder structure (as shown in **Figure 3.9**, with both encoders leveraging the Transformer architecture. This choice aligns well with the inherent sequential nature of textual data. The:

- **Text Encoder:** employs a classic Transformer architecture, effectively processing sequential text data.
- **Image Encoder:** In contrast, utilizes a Vision Transformer (ViT) to extract pertinent visual features from the input images.

Contrastive Learning serves as the core training paradigm for CLIP. The model is presented with image-text pairs, where it progressively learns to align the genuinely corresponding pairs while distinguishing them from unrelated pairings. This process facilitates the creation of the aforementioned *shared embedding space*, which serves as the foundation for subsequent tasks.

The model's strength lies in its remarkable **generalization capabilities** and **adaptability** to analogous tasks. This characteristic positions CLIP as a cornerstone for the development of more intricate multimodal systems.

Multimodal DEtection TRansformer (MDETR)

MDETR, is a multimodal model introduced in 2021 by the paper *"MDETR - Modulated Detection for End-to-End Multi-Modal Understanding"* [32] that tackles a specific task within text-image multimodality: object detection conditioned on text queries. Unlike CLIP, which focuses on general image-text relationships, MDETR delves deeper, pinpointing specific objects in an image based on textual descriptions.

Its core functioning is taking an image and a corresponding text input (typically a caption or a question) as input, and jointly analyzes them to identify objects relevant to the textual information. This approach shares similarities with the End-to-End architecture of DETR, where both image and text data are processed simultaneously.
It operates within a domain closely related to both **Visual Question Answering (VQA)** and **Phrase Grounding**, albeit with distinct nuances:

- **Visual Question Answering (VQA):** VQA focuses on generating natural language answers to open-ended questions posed about an image. In contrast, MDETR prioritizes object detection based on textual descriptions. While it might not provide full sentence answers, it effectively highlights the relevant objects within the image.

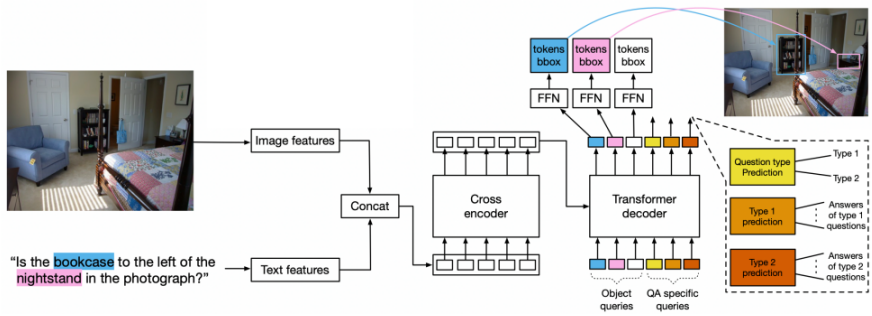

Figure 3.10: MDETR VQA Pre-training - Source : [32]

- **Phrase Grounding:** Phrase grounding aims to localize specific image regions corresponding to a given phrase. This aligns with MDETR's functionality; however, the provided phrase might not necessarily be a complete question. For example, grounding the phrase "red car" would simply locate the red car, not answer a question about its characteristics.

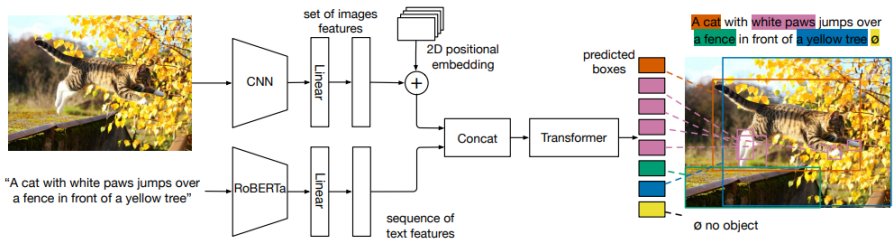

Figure 3.11: MDETR Phrase Grounding - Source : [32]

This model lies at an intersection between these two tasks. It leverages textual descriptions, which can be either questions or phrases, to guide the object detection process within an image. The model's output is not a comprehensive answer in natural language, but rather bounding boxes and class labels for the identified objects of interest.

Chapter 4

Method and Contribution

This chapter delves into the domain from which this research emerges – **Multimodal Machine Translation (MMT)**. We embark on a journey to explore the core concepts and the captivating evolution of MMT research, tracing its trajectory from 2016 to the present day. Over the past eight years, this field has witnessed a remarkable surge in advancements, punctuated by significant paradigm shifts due to the inherent complexity and the multitude of approaches envisioned to tackle *multimodal settings*.

Following this exploration, we will turn our attention to the *Multi30k dataset*, which currently stands as the preeminent (and arguably the sole) resource specifically designed for MMT research. While other multimodal datasets exist, the Multi30k remains uniquely distinguished by its inclusion of multiple target languages alongside images that are described by the corresponding sentences within the dataset.

However, the focal point of this chapter lies in a thorough examination of the **architectural innovation proposed in this research**. We will detail how this architecture leverages **the Transformer as its foundational element**, while ingeniously recombining elements from various tasks within the MMT domain. This detailed analysis will elucidate the rationale behind these design choices and their significance in furthering the capabilities of MMT models.

4.1 State of the art and Related Works

The year 2016 marked a pivotal moment in the evolution of **Machine Translation (MT)**. While significant progress had been made in various related tasks like visual question answering, image captioning, and text-based image retrieval, a crucial limitation emerged: these tasks were predominantly monolingual and centered

largely on the English language. Recognizing this gap, the research community embarked on a new frontier – **Multimodal Machine Translation (MMT)**.

A cornerstone in this pursuit was the introduction of a *"Shared Task on Multimodal Machine Translation" by Specia et al. (2016)* [33] during the *Machine Translation conference (WMT16)*. This initiative provided a critical benchmark for MMT research – the Multi30k dataset [34]. This dataset, an extension of existing resources, incorporated three additional target languages: *German, French, and Czech* (further details on the Multi30k will be explored in the next section). The *"Shared Task"* itself comprised two key challenges: **Multimodal Machine Translation** and **Crosslingual Image Description**. By offering a standardized benchmark and fostering competition among researchers, this initiative significantly spurred advancements in the field of MMT. One of the early approaches to MMT focused on the seemingly straightforward idea of merging text and visual features. Pioneering works like *"Multimodal Attention for Neural Machine Translation" by Caglayan et al., 2016* [4] explored architectures that combined a standard encoder-decoder model for text processing with a visual feature extractor. These visual features were often derived from pre-trained image recognition models, capturing essential information about the content of the accompanying image. The combined features were then fed into the decoder to generate the translated text. While these initial attempts were relatively simple, they demonstrated the potential of leveraging visual information to enhance translation accuracy, particularly for sentences that relied heavily on the context provided by the image.

Next, research such as [35], [36] showed that the architecture can be augmented with an additional attention mechanism. This mechanism is specifically designed to *extract visual context from the input image*. The extracted visual context is then strategically recombined with the attention produced by the text decoder. While CNNs (like ResNet) were initially popular for visual feature extraction in MMT, recent research suggests a shift towards *Visual transformers*. These transformers, often paired with gating mechanisms (see Chapter 3), offer a more nuanced approach to extracting relevant visual context, as explored in works like *"On Vision Features in Multimodal Machine Translation"* [37] and *"Good for Misconceived Reasons: An Empirical Revisiting on the Need for Visual Context in Multimodal Machine Translation"* [38]. As visual transformers have solidified their position as state-of-the-art in computer vision and multimodal tasks, we reaffirmed their use as our visual backbone. Furthermore, these two studies demonstrate the benefits of global feature extraction in terms of regularization, while maintaining the quality of the generated translation. This observation is supported by [37], which decides to also study the effect of an *Object Detector backbone along with an Image Captioner one*. Extracting the regional feature approach is gaining traction,

with several works adapting these models for their specific needs. For instance, [37] utilizes the model for regional feature extraction only, while [39] employs it to investigate the effectiveness of *asymmetric contrastive training*. In order to investigate architectural approaches at a reduced scale, we opted to focus on the object detection pathway, given its documented success in the relevant literature.

The current state-of-the-art in this field primarily explores two avenues: *integration of Pre-trained Language Models (LMs)* as in [2], [3]; *Text or Visual Masking Incorporation* as in [40],[41] and the same [37]. However, since this last cited paper utilizes a very extensive pre-training approach to investigate the capabilities of these architectures at a reduced scale, we opted to forgo these established approaches.

4.2 Multi30k

The *Multi30K* [34] dataset stands as a pivotal resource in the field of **Multimodal Machine Translation (MMT)**. It builds upon the foundation laid by the *Flickr30k dataset* [41], which offers a collection of 30,000 images, each accompanied by five distinct English captions. The Multi30K dataset significantly expands upon this concept by introducing multilingual elements. For each image, it provides not only an English description, but also three additional human-translated captions in German, French, and Czech. This expansion has established the Multi30K dataset as the preeminent benchmark for MMT research.

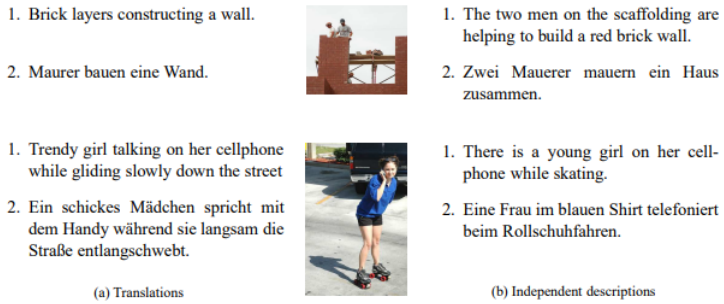

Figure 4.1: Multi30k example - Source : [34]

However, it is essential to acknowledge certain limitations inherent to the dataset. One key limitation is its relatively small scale compared to some more recent datasets. Despite this limitation, the size aligns well with the specific research methodologies employed in our investigation. Another limitation, as highlighted by [3], pertains to the representation of disambiguation, which is a central challenge addressed by MMT. The Multi30K dataset may not fully capture this complexity, as a relatively small percentage of the sentences require visual cues to fully comprehend the context.

This characteristic suggests that further exploration with datasets that emphasize disambiguation may be necessary for a more comprehensive understanding of MMT capabilities.

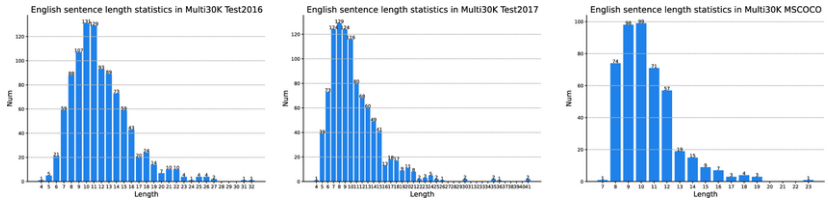

Figure 4.2: Multi30K 2016, 2017, MSCOCO test dataset English sentence length statistical analysis - Source: [42]

4.3 Our Contribution and Main Inspiration

Our methodology leverages various techniques established in the literature, mainly [2] and [3]. Crucially, we aimed to minimize model size while maintaining trainability. Considering the reference benchmark's limitations, we experimented with Transformer Tiny and Small architectures. **Figure 4.1 and 4.2** shows the architecture components and their interconnections.

4.3.1 Overall architecture

The core of our architecture is the traditional **sequence-to-sequence model** introduced by [1]. Building upon this foundation, we incorporate **two visual information streams** complementing the textual input:

- **Global Visual Features**: These capture general scene information not readily identifiable locally. We extract these features using the **CLIP model** [31] (with a *ViT backbone*) described in respectively described in **Section 3.2 and 3.3**, inspired by the success of transfer learning in textual modalities ([40],[41]). However, we apply this strategy to the visual domain.
- **Regional Visual Features**: We extract these features using **MDETR**, a multimodal variant of DETR ([32]) described in **Section 3.3**, which excels in tasks like *phrase grounding* (identifying labels based on textual descriptions) and *visual question-answering* (identifying objects or characteristics based on textual questions).

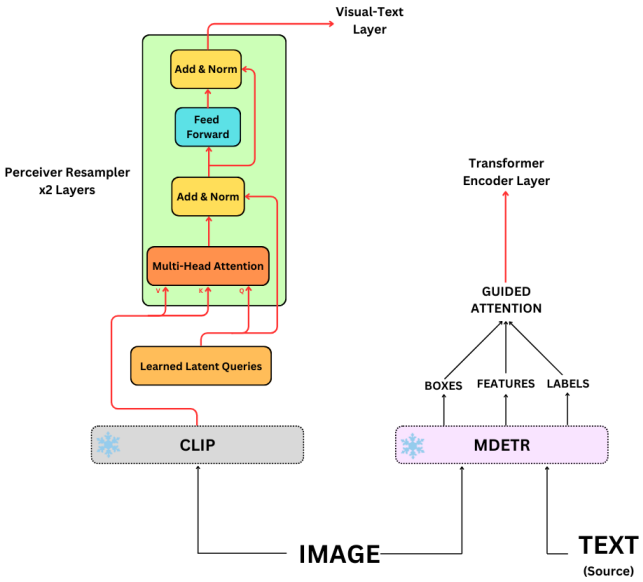

Figure 4.3: Image-side of the model showing Perceiver Resampler and Guided Attention, to notice that CLIP and MDETR are kept freeze for the whole training process

Feature Processing:

The two feature types undergo separate processing:

- **Global Visual Features**: Inspired by video processing techniques, we leverage the **Perceiver Resampler** [43] to reduce the dimensionality of the global features. This is particularly useful for *managing the computational cost* and focusing on the most informative aspects of the scene. The Perceiver Resampler operates by employing a **learnable number of queries** to extract the most important features from the global feature set. These queries essentially act as a **filter**, identifying the most relevant information within the global features for the task at hand.

- **Regional Visual Features**: We construct a **Guided Attention Mask** using the regional features and their corresponding labels, drawing inspiration from [3]. This mask plays a *crucial role in directing the model's attention during the translation process*. It assigns higher attention scores to words in the reference text that directly correspond to objects identified in the image through regional feature extraction. This mechanism essentially guides the model to prioritize translating words that describe the visual content present in the image.

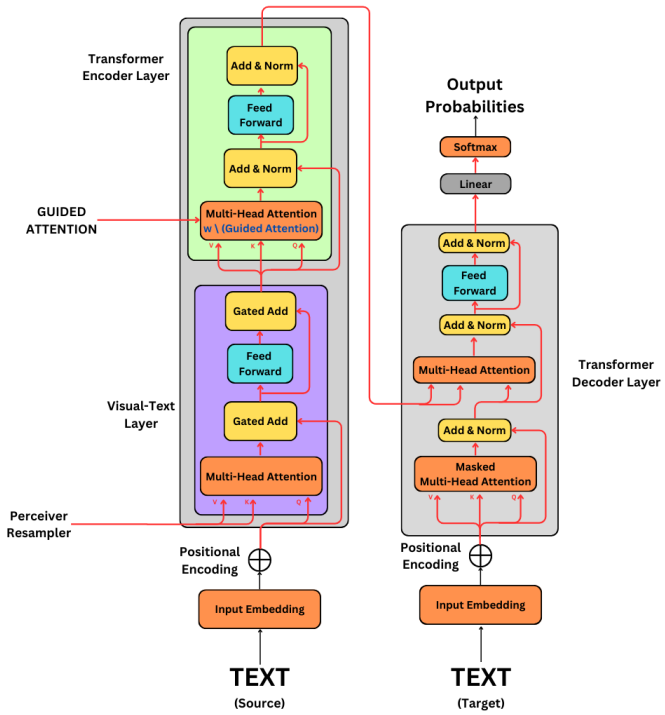

Figure 4.4: Text-side of the model showing the Vision-Text Layer, Transformer Encoder and Transformer Decoder components

By processing the visual features in these ways, we ensure that the model effectively leverages both *global scene information* and *specific object detections* to create a more accurate and informative translation based on the visual context.
The **Figure 4.3** illustrates the pipeline for visual feature extraction, highlighting the specific modules these features interact with before entering the encoder.

Encoder-Decoder Interaction:

The core of our architecture lies in the **interaction between the visual and textual representations within the encoder-decoder framework**. Here's a breakdown of this interplay:

- **Cross Attention**: At each layer of the encoder, the **Vision-Text Encoder** performs a "cross-attention" operation with the reference text. This mechanism allows the encoder to attend to *specific parts of the text based on the visual information it has processed*. In simpler terms, the encoder uses visual features to determine which parts of the reference text are most relevant for generating the target language translation.

- **Gating Mechanism for Visual Feature Integration**: A critical inspiration in our model is the incorporation of a *gating mechanism* within the encoder. This mechanism controls how the global visual features are gradually introduced during the training process. It utilizes **two trainable parameters** to determine the importance of each visual feature based on its contribution to the translation task. By introducing global visual features selectively, the model can focus on the most relevant information at each stage of the training process, leading to more efficient learning.

- **Transformer Decoder**: The decoder component of the model remains *unaltered*. Since its primary function is textual generation, it solely operates on the processed information received from the encoder, which has been already influenced by both the visual contributions.

Figure 4.4 illustrates the key enhancements made to the encoder component of our architecture through the incorporation of visual features. These figures will visually represent the integration of global and regional features, along with the mechanisms employed for their effective utilization

In the next sections will be provided a *comprehensive exploration* of the individual modules that constitute our proposed architecture. Each section will delve into the specific functionality and design choices behind each module, offering a deeper understanding of their contributions to the overall translation process.

This in-depth examination will not only elucidate the inner workings of each module but also shed light on the rationale behind our design decisions. By dissecting each component, we aim to provide a clear picture of how these modules interact to achieve the task of generating accurate and visually grounded translations.

4.3.2 Perceiver Resampler and Learned Latent Queries

A significant challenge in multimodal architectures arises from the inherent disparity between the *dimensionality of visual features* and *textual tokens*. For instance, a *ViT model trained on 224x224 pixel* images generates 196 patches and a single class token, while the average length of a textual sequence can range from 20 to 50 tokens. This vast discrepancy in feature count presents two key drawbacks:

- **Computational Bottleneck:** The sheer volume of visual patches significantly increases the model's computational and memory requirements.

- **Information Dilution:** The abundance of patches, many of which may contain irrelevant background details or corner/edge artifacts, can negatively

impact model performance. These extraneous features dilute the model's focus on the essential visual information crucial for translation.

To address this challenge, [43] introduced the **Perceiver Resampler** module within their *Flamingo model* (**Figure 4.3**). This concept was further adapted by [2] to function effectively with static images rather than video frames.

The **Perceiver Resampler** leverages a structure akin to an encoder with multi-head attention and feed-forward networks (FFNs). However, instead of incorporating visual features directly into the queries, it *employs a set of pre-defined, learned latent queries*. These queries have a *fixed size*, making them suitable for reducing the dimensionality of visual features. Crucially, the model is trained to select the *most critical features* for the translation task by focusing on the most relevant latent queries. This mechanism effectively reduces the **computational burden** and mitigates the negative effects of information dilution caused by irrelevant visual details.

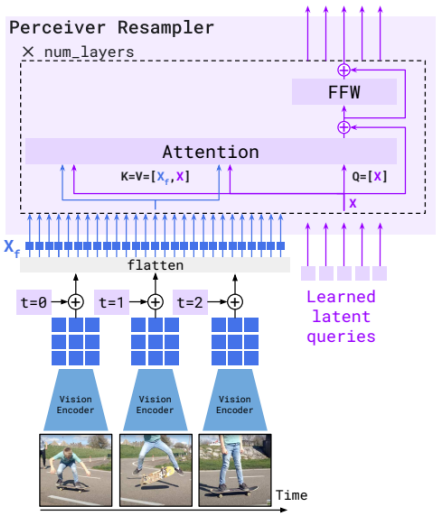

Figure 4.5: Perceiver Resampler introduced by [43], it differs from [2] for the usage of single-frame media instead of multiple-frame

Let w denote the input visual embeddings. Learned latent queries, are represented by vector $\lambda = (\lambda_1, ..., \lambda_r)$, these latent queries have a fixed dimensionality, significantly lower than that of w.
The resampling process iterates through multiple layers (R layers). Each layer refines the latent queries based on their interaction with the visual features. Here's the breakdown of a single layer:

- **Query Update:** The latent queries are projected and concatenated with the visual embeddings: $[w, \lambda] = \text{concat}(w, \lambda)$ This combined representation is then fed into a multi-head attention (MHA) layer, allowing the queries to attend to relevant parts of the visual features:

$$\lambda' = \text{MHA}(K = [w, \lambda], V = [w, \lambda], Q = \lambda) \tag{4.1}$$

- **Feed-Forward Refinement:** The output of the MHA layer is further processed by a feed-forward network (FF) to refine the queries:

$$\lambda' = \text{FF}(\lambda') \tag{4.2}$$

- **Iterative Update:** The updated queries ($\lambda^{'}$) are then combined with the original latent queries to form the input for the next layer:

$$\lambda \leftarrow \lambda + \lambda' \tag{4.3}$$

  This iterative process continues for all R layers.

Finally, the output of the last Perceiver Resampler layer, denoted by p, represents the reduced-dimensionality visual tokens suitable for further processing within the multimodal architecture:

$$p \leftarrow \lambda \tag{4.4}$$

Our implementation builds upon the work presented in [2]. This decision is motivated by the computational efficiency that this approach reached concerning a classical transformer encoder or an MLP

4.3.3 Vision-Text Layer and Gated Cross Attention

Inspired by the work in [43] and [2], we introduce a key element within our encoder architecture: the **Vision-Text Layer**. This layer plays a crucial role in gradually integrating visual context with textual information during the translation process.

Structure and Functionality:

The **Vision-Text Layer** is strategically positioned within the encoder, sharing the same number of layers. This placement fosters a close interaction between visual and textual representations throughout the encoding process. The core mechanism employed within the Vision-Text Layer is *cross-attention*. However, in this specific context, the text acts as the *query*, while the visual features extracted by **CLIP** serve as both the *key and value*. This cross-attention operation allows the model to focus on specific parts of the visual features that are most relevant to the textual information.

Gated Integration for Controlled Information Flow:

To ensure a controlled and gradual introduction of visual context, we incorporate a *gating mechanism* within the Vision-Text Layer. This mechanism leverages *two trainable parameters*, denoted by γ_a *and* γ_f, which are processed using the **Hyperbolic Tangent Function (TANH)** for normalization. The resulting values range between *-1 and 1*, effectively acting as gating factors.

- **Gating Parameter γ_a:** This parameter controls the *portion of the calculated attention that is ultimately incorporated into the encoder output*. By adjusting γ_a, the model can regulate the influence of visual context on the evolving textual representation.
- **Gating Parameter γ_f:** This parameter acts as a filter on the output of the cross-attention mechanism before it is passed through a feed-forward network (FFN). Similar to γ_a, γ_f allows for a controlled flow of information, ensuring that only the most relevant aspects of the visual features are integrated into the encoding process.

The key benefits of the gating mechanism introduced by [2] compared to previous approaches like those presented in [38] and [37] lie in its simplicity and efficiency. Our *design requires only two trainable parameters* (γ_a and γ_f), significantly **reducing the overall model complexity**. Additionally, by analyzing the learned values of these parameters, we can gain valuable insights into the model's reliance on visual information for translation tasks.

The use of a gating mechanism within the encoder is motivated by two primary reasons:

- **Gradual Integration:** This approach allows the model to *progressively incorporate visual context* into the encoding process, potentially leading to a more stable and controlled learning of the multimodal relationships.

- **Targeted Filtering:** By leveraging the gating mechanism, we can *selectively filter the global features extracted by CLIP and reduced by the Perceiver Resampler*, ensuring that only the most relevant information is utilized for translation. This complements the role of the regional features, which are directly integrated without gating to capture finer details within the image.

In contrast to previous work like [2] and [43], which employed gating mechanisms *to gradually make pre-trained language models multimodal*, our focus here is on using **gating as a filter to process only the necessary global features**. This distinction allows for more efficient utilization of visual information specifically tailored to the task of image-to-text translation.

4.3.4 Regional Features and Guided Attention

This concluding section delves into the innovation driving our architecture's effectiveness: the **strategic exploitation of regional features** alongside a meticulously designed attention mechanism focused on objects identified within the image. This approach has been introduced from cutting-edge research presented in [3].

Guided Self-Attention Mask

By constructing a **correlation matrix** that meticulously maps visual features extracted by **MDETR** to their corresponding labels identified in the text (meeting a predefined probability threshold), a powerful tool has been created: the **Guided Self-Attention Mask**. This mask serves as a *guiding light within the transformer encoder*, effectively **highlighting** the specific words in the reference text that directly correspond to objects identified within the image. By providing this explicit guidance, the model is **relieved of the burden of learning to ignore irrelevant visual features**. As the name "Guided" suggests, the model is directly instructed on which visual elements to attend to, leading to more efficient learning and improved translation accuracy.

Alignment Matrix

The process of generating the **alignment matrix** is meticulously crafted. **MDETR** extracts visual features corresponding to bounding boxes and labels found in the text (surpassing a specific confidence threshold). Each element within the alignment matrix is assigned a value of 1 if the visual features directly correspond to the assigned label, and 0 otherwise (**Figure 4.4**).
Through this intricate process, the model **learns to establish strong correlations between specific textual elements describing objects present in the image**, while simultaneously **learning to disregard irrelevant visual information**.

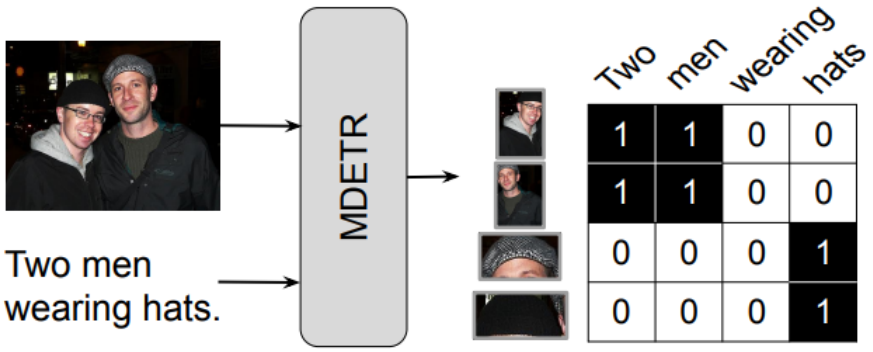

Figure 4.6: MDETR alignment matrix from [3]

Since the attention mechanism is not even calculated for these uncorrelated pairs, the model avoids wasting computational resources and focuses solely on the most relevant features for accurate translation.
In essence, the **Guided Self-Attention Mask** acts as a bridge between the *textual and visual domains*, enabling the model to *precisely align words with their corresponding objects within the image*. This targeted focus on relevant information significantly improves the model's ability to generate accurate and visually-grounded translations.

Chapter 5

Results

5.1 Experimental Settings

Model Selection and Configuration:

Our **Seq2Seq transformer architecture** leverages pre-trained models from the **Hugging Face library** [44] to benefit from optimized implementations and readily available generation code. The **MarianMT model** [45], specifically designed for machine translation and based on the *Marian transformer framework*, was chosen for its efficiency (originally written in C++ and later ported to Python for research purposes). Similarly, the **CLIP** model was obtained from Hugging Face, utilizing the version featuring a *32-patch ViT-Base* backbone, which generates 768-dimensional embeddings. In contrast, the **MDETR** model was sourced directly from its original TorchHub repository and has *Efficientnet B5* [46] as visual backbone. All the code was written using the *Pytorch Library with Cuda up to the latest version.*

HuggingFace Library: a platform for data science containing pre-trained models, libraries and resources for research - Source of logo : *https://huggingface.co/brand*

In **Table 5.1** are shown the configurations employed: N Layers indicates the number of encoder and decoder layers (equating to those in the Vision-Text module), d_{model} specifies the feature dimension utilized by the model, d_{ff} denotes the dimension of the Feed Forward Network (FFN), and h represents the number of heads used in multi-head attention. Both configurations incorporate parameters for the Perceiver Resampler, where L signifies the number of layers (set to 2 in this case) and Num Learnable refers to the learnable latent queries (both values maintained as in [2]). Additionally, h denotes the number of multi-attention heads, and d_h represents the head size.

Table 5.1: Configuration used for experiments "Tiny" and "Small", both for Transformer and Perceiver Resampler

	Seq2Seq Configuration				
	N Layers	**d_model**	**d_ff**	**h**	**Trainable Parameters**
Tiny	4	128	256	4	2,975,360
Small	6	512	1024	8	38,144,512
	Perceiver Resampler Configuration				
	L	**Num Learnable**	**h**	**d_h**	
Tiny	2	64	4	1024	
Small	2	64	8	2048	

Training Parameters:

The **Adam optimizer** was selected, employing an initial *learning rate of 1e-7*, *betas of (0.9, 0.98)* – consistent with prevailing MMT research – and an *epsilon of 1e-9*. The learning rate scheduling follows the original implementation of the transformer [1], incorporating a *linear increase during a warmup period of 4000 steps*, followed by an inverse square root decay thereafter. **Cross-entropy Loss** was chosen due to its suitability for the research domain.

Evaluation and Generation Strategies:

All experiments were trained for 30 epochs, with the checkpoint exhibiting the highest **BLEU** score (calculated via **Greedy Search** at the conclusion of each epoch) designated as the optimal model. During the final generation phase, a **Beam Search** algorithm was employed with a *beam size of 5*. Both generative algorithms were sourced directly from the *Hugging Face libraries*.

Evaluation Metrics:

We used to evaluate the quality of translation, the two most common metrics in MMT: **BLEU** (*BiLingual Evaluation Understudy*) [47] and **METEOR Score** [48].

Both aim to assess how well a generated translation aligns with a human-created reference translation.

- **BLEU:** focuses on *n-gram precision*, meaning it calculates the *percentage of n-word sequences* (e.g., unigrams, bigrams) in the generated text that also appears in the reference. It penalizes translations that are too short compared to the reference.
- **METEOR:** on the other hand, takes a more comprehensive approach. It considers not only exact word matches but also synonyms and paraphrases. This allows for a more nuanced evaluation, capturing the meaning conveyed even if the wording differs slightly.

Hardware settings:

Experiments were conducted on a computer equipped with an *Intel Core i7-13700K processor* (16 cores, maximum frequency 5.40 GHz), an *NVIDIA RTX 4070* graphics card with 12GB of video memory, and *5,888 CUDA cores*.

5.2 Results for EN→DE , EN→FR, EN→CS

This section details the experiments conducted to evaluate the effectiveness of incorporating visual features into a transformer-based machine translation model. The settings used for these experiments were established in **Section 5.1**. A comprehensive evaluation was performed, encompassing a total of 24 experiments. These experiments were divided equally between two model configurations: Tiny and Small, with 12 experiments conducted for each configuration.

The upcoming tables present the experiment design categorized by target language (German, French, and Czech). For each language, a systematic approach was employed to assess the impact of visual features on translation quality. This approach involved three stages:

- **Baseline Experiment:** To establish a baseline for comparison, an initial experiment was conducted utilizing the base transformer architecture without incorporating any visual features.
- **Individual Feature Analysis:** Subsequently, two separate experiments were performed to analyze the influence of each individual visual feature type: global and regional features. These experiments aimed to isolate the effect of each feature type on the translation process.
- **Combined Feature Exploration:** Finally, experiments were conducted that utilized both global and regional visual features simultaneously. This stage investigated the potential for synergistic effects when combining these feature types. The extracted values of $|tanh(ga)|$ and $|tanh(gf)|$ for each layer within the Visual-Text module, which are crucial for understanding the model's internal behaviour, will be presented and analyzed in subsequent sections.

The tables report BLEU and METEOR scores calculated on all three available test sets (2016, 2017, and 2018) for each language. It is important to note that the Czech language dataset was not included in the 2018 test set. Additionally, each experiment displays the absolute and relative percentage improvements compared to the baseline text-only model, allowing for a clear evaluation of the impact of visual features on translation quality.

English to German

Table 5.2: Global and Regional Visual Features - Transformer Tiny - EN → DE

	Test 2016		Test 2017		Test 2018	
	BLEU	**METEOR**	**BLEU**	**METEOR**	**BLEU**	**METEOR**
Text-Only	34,473	62,244	26,423	53,927	24,269	50,722
w\Global Feat	34,568	62,350	27,364	54,520	25,028	51,334
Increment	*0,095*	*0,106*	*0,941*	*0,593*	*0,759*	*0,612*
w.r.t. text-only	*(+0,28%)*	*(+0,17%)*	*(+3.56%)*	*(+1,10%)*	*(+3,13%)*	*(+1,21%)*
w\Regional Feat	33,835	62,054	25,913	53,888	24,057	50,492
Increment	*-0,638*	*-0,190*	*-0,510*	*-0,039*	*-0,212*	*-0,230*
w.r.t. text-only	*(-1.85%)*	*(-0,31%)*	*(-1,93%)*	*(-0,07%)*	*(-0,87%)*	*(-0,45%)*
w\Glob, Reg Feat	36,164	63,905	28,335	56,583	26,205	53,302
Increment	*1,691*	*1,661*	*1,912*	*2,656*	*1,936*	*2,580*
w.r.t. text-only	*(+4,91%)*	*(+2,67%)*	*(+7,24%)*	*(+4,92%)*	*(7,98%)*	*(+5,09%)*

Table 5.3: Global and Regional Visual Features - Transformer Small - EN → DE

	Test 2016		Test 2017		Test 2018	
	BLEU	**METEOR**	**BLEU**	**METEOR**	**BLEU**	**METEOR**
Text-Only	32,787	59,940	23,654	51,985	23,458	48,987
w\Global Feat	33,970	61,983	26,481	53,278	24,929	50,834
Increment	*1,183*	*2,043*	*2,828*	*1,293*	*1,471*	*1,847*
w.r.t. text-only	*(+3,61%)*	*(+3,41%)*	*(+11,95%)*	*(+2,49%)*	*(+6,27%)*	*(+3,77%)*
w\Regional Feat	31,281	58,52	21,827	48,83	21,147	46,377
Increment	*-1,506*	*-1,420*	*-1,827*	*-3,155*	*-2,311*	*-2,610*
w.r.t. text-only	*(-4,59%)*	*(-2,37%)*	*(-7,72%)*	*(-6,07%)*	*(-9,85%)*	*(-5,33%)*
w\Glob, Reg Feat	31,009	58,640	23,523	50,330	21,672	47,120
Increment	*-1,778*	*-1,300*	*-0,131*	*-1,655*	*-1,786*	*-1,867*
w.r.t. text-only	*(-5,42%)*	*(-2,17%)*	*(-0,55%)*	*(-3,18%)*	*(-7,61%)*	*(-3,81%)*

English to French

Table 5.4: Global and Regional Visual Features - Transformer Tiny - EN → FR

	Test 2016		Test 2017		Test 2018	
	BLEU	**METEOR**	**BLEU**	**METEOR**	**BLEU**	**METEOR**
Text-Only	51,559	73,486	42,537	66,086	30,045	56,108
w\Global Feat	52,117	74,227	43,820	67,251	30,856	57,274
Increment	*0,558*	*0,741*	*1,283*	*1,165*	*0,811*	*1,166*
w.r.t. text-only	*(+1,08%)*	*(+1,01%)*	*(+3,02%)*	*(+1,76%)*	*(+2,70%)*	*(+2,08%)*
w\Regional Feat	56,963	77,626	48,425	70,85	33,398	59,723
Increment	*5,404*	*4,140*	*5,888*	*4,764*	*3,353*	*3,615*
w.r.t. text-only	*(+10,48%)*	*(+5,63%)*	*(+13,84%)*	*(+7,21%)*	*(+11,16%)*	*(+6,44%)*
w\Glob, Reg Feat	57,457	77,910	47,603	70,492	33,860	60,498
Increment	*5,898*	*4,424*	*5,066*	*4,406*	*3,815*	*4,390*
w.r.t. text-only	*(+11,44%)*	*(+6,02%)*	*(+11,91%)*	*(+6,67%)*	*(+12,70%)*	*(+7,82%)*

Table 5.5: Global and Regional Visual Features - Transformer Small - EN → FR

	Test 2016		Test 2017		Test 2018	
	BLEU	**METEOR**	**BLEU**	**METEOR**	**BLEU**	**METEOR**
Text-Only	51,731	73,657	42,119	65,395	30,308	56,526
w\Global Feat	53,647	76,026	45,136	68,673	32,723	58,576
Increment	*1,916*	*2,369*	*3,017*	*3,278*	*2,415*	*2,050*
w.r.t. text-only	*(+3,70%)*	*(+3,22%)*	*(+7,16%)*	*(+5,01%)*	*(+7,97%)*	*(+3,63%)*
w\Regional Feat	49,272	72,076	40,878	64,553	28,746	54,85
Increment	*-2,459*	*-1,581*	*-1,241*	*-0,842*	*-1,562*	*-1,676*
w.r.t. text-only	*(-4,75%)*	*(-2,15%)*	*(-2,95%)*	*(-1,29%)*	*(-5,15%)*	*(-2,97%)*
w\Glob, Reg Feat	50,303	72,973	41,138	65,038	29,881	56,078
Increment	*-1,428*	*-0,684*	*-0,981*	*-0,357*	*-0,427*	*-0,448*
w.r.t. text-only	*(-2,76%)*	*(-0,93%)*	*(-2,33%)*	*(-0,55%)*	*(-1,41%)*	*(-0,79%)*

English to Czech

Table 5.6: Global and Regional Visual Features - Transformer Tiny - EN → CS

	Test 2016		Test 2017	
	BLEU	**METEOR**	**BLEU**	**METEOR**
Text-Only	27,235	53,629	19,861	44,497
w\Global Feat	30,807	55,446	24,261	48,537
Increment	*3,572*	*1,817*	*4,400*	*4,040*
w.r.t. text-only	*(+13,12%)*	*(+3,39%)*	*(+22,15%)*	*(+9,08%)*
w\Regional Feat	28,655	55,669	23,263	47,735
Increment	*1,420*	*2,040*	*3,402*	*3,238*
w.r.t. text-only	*(+5,21%)*	*(+3,80%)*	*(+17,13%)*	*(+7,28%)*
w\Glob, Reg Feat	29,031	55,565	22,991	46,773
Increment	*1,796*	*1,936*	*3,130*	*2,276*
w.r.t. text-only	*(+6,59%)*	*(+3,61%)*	*(+15,76%)*	*(+5,11%)*

Table 5.7: Global and Regional Visual Features - Transformer Small - EN → CS

	Test 2016		Test 2017	
	BLEU	**METEOR**	**BLEU**	**METEOR**
Text-Only	25,279	51,204	18,079	40,813
w\Global Feat	26,727	52,756	20,392	43,809
Increment	*1,448*	*1,552*	*2,313*	*2,996*
w.r.t. text-only	*(+5,73%)*	*(+3,03%)*	*(+12,79%)*	*(+7,34%)*
w\Regional Feat	23,819	49,895	16,8	38,91
Increment	*-1,460*	*-1,309*	*-1,279*	*-1,903*
w.r.t. text-only	*(-5,78%)*	*(-2,56%)*	*(-7,07%)*	*(-4,66%)*
w\Glob, Reg Feat	25,051	50,958	17,026	39,482
Increment	*-0,228*	*-0,246*	*-1,053*	*-1,331*
w.r.t. text-only	*(-0,90%)*	*(-0,48%)*	*(-5,82%)*	*(-3,26%)*

5.3 Impact of Regional Features

5.3.1 English to German Translation

Table 5.2 analysis reveals that incorporating solely regional features yields performance metrics nearly equivalent to those achieved by the text-only model (with a maximum decrease of **1.93%**). This suggests that the model fails to leverage the visual information provided by the object detector within these specific inference sets. As will be demonstrated later, the inclusion of global visual features significantly alters this dynamic. It is noteworthy that the reduction in METEOR scores remains minimal. This can be attributed to the model's ability to effectively reconstruct the syntactic structure, which remains unaffected by incorporating regional features.

Table 5.3 replicates the experiments above, employing the transformer architecture in its small size. This iteration reveals a more pronounced performance decline, reaching a maximum decrease of **10%** in the *2018 test set* (where the corresponding Tiny model exhibited the least degradation). These findings suggest that the small architecture is inherently unsuited to the limited size of the dataset. Consequently, incorporating regional features without a corresponding data volume expansion compromises the model's stability, rendering it more susceptible to overfitting.

5.3.2 English to French Translation

Intriguingly, when the identical experiment is conducted on the Tiny configuration for the French language (results presented in **Table 5.4**), highly encouraging outcomes are observed. The model demonstrates significant improvements across both metrics (BLEU and METEOR). Notably, the French language achieves the highest performance due to its numerous syntactic similarities to English. Enhancements surpass **5%** on all datasets, reaching a peak of **13%** in the *2017 test set*. These findings strongly suggest the promise of incorporating regional features, especially considering the dataset's richness in extractable visual information that demonstrably aids translation in the French language.

Unlike the Tiny version, **Table 5.5** shows us a rather clear decrease across all performances. The **Small version** is unable to take advantage of the regional features, which we have proven to have a great impact on the quality of the translation, losing up to **5%** in the 2018 test while in the Tiny version (**Table 5.4**) it is the one that achieves the greatest improvements. Considering the results obtained from the German language together we can see how the size of the dataset on the chosen architecture has a large negative influence, making it not suitable for the processing of small quantities of data.

5.3.3 English to Czech Translation

Compared to French, Czech exhibits the greatest divergence from English in *morphosyntactic structure*. This disparity significantly hinders its performance on BLEU scores, resulting in the lowest scores among all languages tested. However, **Table 5.6** reveals a substantial positive impact from incorporating visual features, particularly in accurate term identification. These features yield improvements of up to **15%**. Similar to French, the Czech version of the multi30k dataset evidently possesses sufficient visual detail and object information to enhance translation performance.

In contrast to the **Tiny** transformer model, the small transformer architecture, whose results are represented in **Table 5.7**, demonstrates a clear inability to surpass the limitations imposed by overfitting. This results in a substantial decline (reaching a maximum decrease of **7%**) in its overall generalization capabilities. The model fails to effectively leverage the incorporated regional features, leading to a significant deterioration in performance across all evaluated metrics.

5.4 Impact of Global Features

5.4.1 English to German Translation

This subsequent analysis focuses solely on global features extracted by CLIP and filtered through the **Perceiver Resampler** (**Table 5.2**). Similar to the observations with regional features, the dataset appears to lack visually informative details that could enrich and enhance translation precision. Consequently, the results exhibit negligible improvements in the 2017 and 2018 datasets, while the 2016 dataset shows virtually no change. These findings corroborate the observations from the previous experiment employing only regional features (**Table 5.2**). Notably, the METEOR metric remains the least affected metric once again, further emphasizing the significant morphosyntactic similarities between the two languages.

As illustrated in **Table 5.3**, the small transformer model exhibits notable improvements, reaching a peak of **12%** in the 2017 test set, despite its lower initial performance baseline. While the larger architecture's limitations for smaller datasets allow it to benefit from visual features, it is unable to surpass the performance of its smaller counterpart. This disparity can be attributed to the small model's more efficient utilization of visual features to mitigate performance degradation, enabling it to achieve superior generalization capabilities.

5.4.2 English to French Translation

In contrast to the positive results observed with regional features (**Table 5.4**), this experiment presents a distinct trend for global features. While the results exhibit a modest improvement, the magnitude of this enhancement suggests a potential limitation in the dataset's richness of global information compared to its wealth of object-specific details. Despite this limitation, the Tiny model demonstrates continued effectiveness, albeit with minimal performance gains, reaching a maximum increase of **3%** and **2%** for BLEU and METEOR scores, respectively.

A dramatic shift is observed in the results for the small transformer model presented in **Table 5.5**. This iteration achieves peak improvements of **8%** on the 2018 test set, further solidifying the notion that the latest dataset possesses a significantly greater wealth of information exploitable for translation enhancement. It is important to acknowledge that the initial baseline performance for the text-only model in this scenario is marginally higher. This factor may partially contribute to the observed discrepancy in results.

5.4.3 English to Czech Translation

The Czech language exhibits a trend that stands in stark contrast to the observations made for French (**Table 5.6**). The demonstrably substantial performance improvements suggest that global features provide further enrichment for specific Czech language terminology, yielding peak enhancements of up to **22%**. When compared to the results obtained with regional features, it becomes evident that the utilization of any kind of visual feature types significantly enhances Czech terminology.

Our investigation into the Czech language using the minimal model configuration (**Tiny**) reveals a significant performance increase solely through the utilization of global features. This trend persists in the Small model configuration (**Table 5.7**), where, in contrast to the performance with regional features, we observe substantial improvements as well. However, the evidence continues to support the superior performance of Tiny models for these types of datasets. While the observed improvements in the Small model are substantial, reaching as high as **13.7%**, they are nevertheless insufficient to surpass the textual baseline established by the Tiny model.

5.5 Overall Impact of both Visual Features

5.5.1 English to German Translation

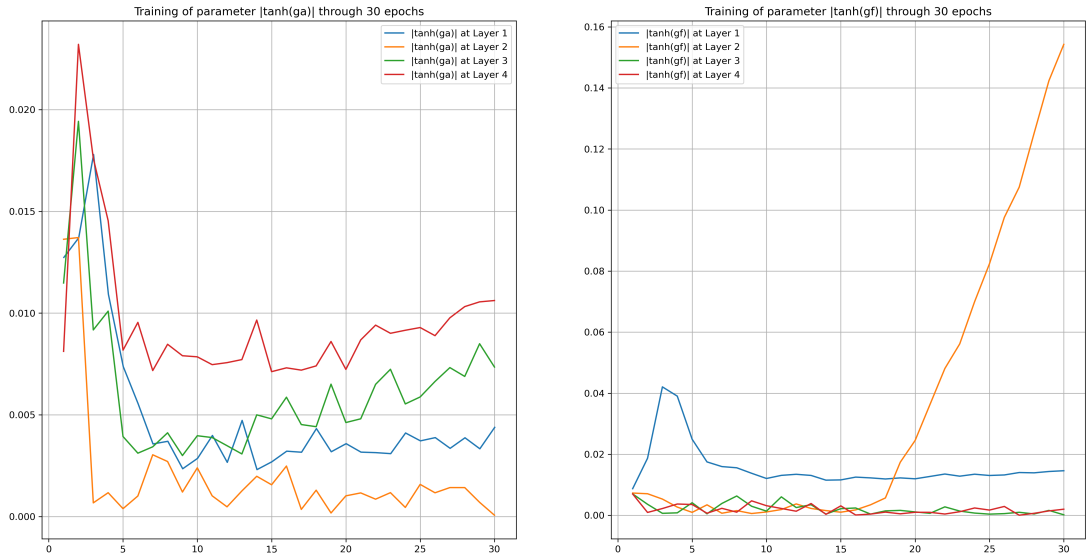

Figure 5.1: Evolution of |tanh(ga)| and |tanh(gf)| values in each layer for configuration tiny for EN → DE

This section delves into the impact of visual features on translation precision, analyzing not only the final experimental results concerning both regional and global features but more importantly, the broader implications for leveraging visual information in translation tasks. We begin our examination with the German language (**Table 5.2**). Here, the scalar and percentage data represent the improvements achieved by the model variant incorporating both visual features. The results presented offer a promising outlook. Notably, the Tiny model exhibits a significant improvement by utilizing both feature types, achieving gains ranging from nearly **5%** to a record-breaking **7%** within this language. This compelling evidence suggests that the German language may be particularly receptive to the combined application of visual features, potentially influencing both the selection of appropriate terminology and the construction of syntactically sound sentences. In essence, this finding underscores the potential of visual information to enhance the accuracy of translations, particularly for languages like German that demonstrably benefit from such multimodal inputs.

Figure 5.1 suggests a diminishing impact of global features throughout training, eventually reaching a stable state. This observation aligns with the final values of

the parameter $|tanh(gf)|$, which indicate a utilization rate of less than **2%** by the model. Consequently, the modest performance improvement observed in **Table 5.2** due to global features is unsurprising. However, the combined application of global and regional features, as evidenced by the results, appears to yield a synergistic effect, leading to superior translation quality.

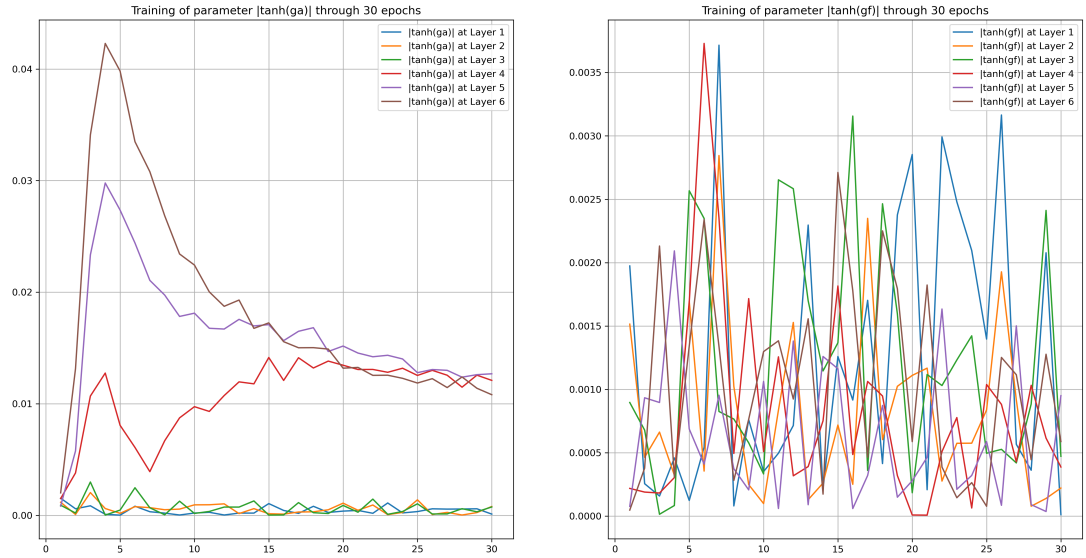

Figure 5.2: Evolution of |tanh(ga)| and |tanh(gf)| values in each layer for configuration small for EN → DE

A stark contrast emerges when examining the Small model (**Table 5.3**). Here, the anticipated benefits of combining both feature types are entirely absent. In fact, the results plummet, with losses reaching nearly **10%**. This detrimental effect extends to the METEOR metric as well, although thankfully, METEOR scores remain marginally higher than those achieved with the model utilizing only regional features. This unexpected outcome suggests a potential explanation. By incorporating global features, the model may have partially recovered the intended syntactic structures through the extraction of more accurate information. However, this inclusion appears to have also introduced additional confusion, leading to a decline in BLEU score despite the positive impact on METEOR. In light of these findings, it becomes clear that the Tiny architecture demonstrably exhibits greater suitability for this task. It appears to possess a superior ability to leverage the information gleaned from the dataset images. Conversely, the Small architecture in German lacks the same level of stability and performs considerably worse when presented with both regional and global features. Further investigation is warranted to fully elucidate the underlying mechanisms at play and to determine how to

optimize the utilization of visual features within the Small model for the German language.

An analysis of the two gating parameters depicted in **Figure 5.3** corroborates these findings. The figure suggests that the model experiences significant instability during the initial incorporation of global visual information. This instability manifests as a concerning pattern of performance degradation, ultimately hindering the model's ability to generalize effectively.

5.5.2 English to French Translation

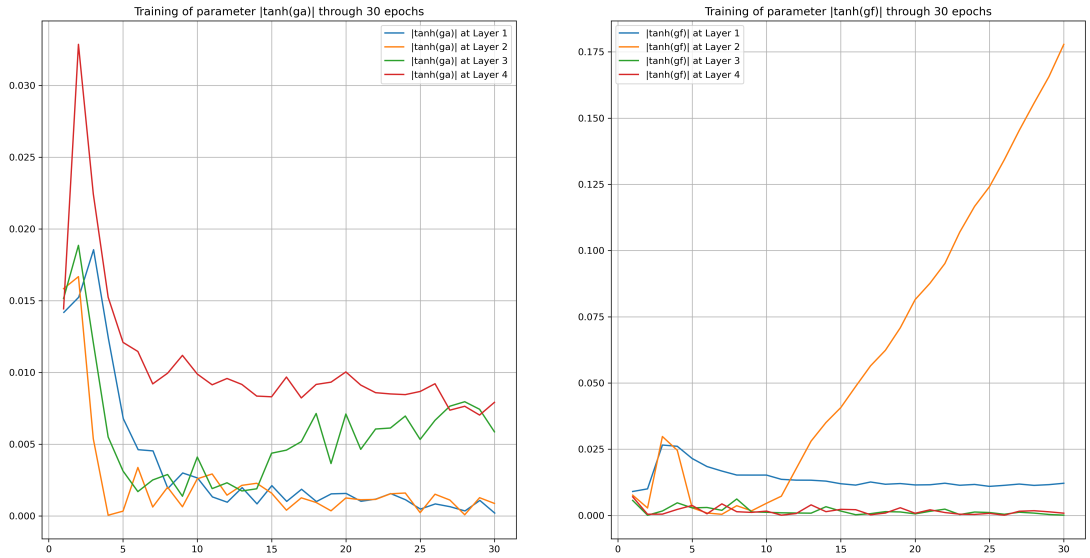

Figure 5.3: Evolution of |tanh(ga)| and |tanh(gf)| values in each layer for configuration tiny for EN → FR

The French language presents a rather encouraging scenario, as illustrated in **Table 5.4**. Here, we observe a positive synergy between regional and global features. When combined, they yield further performance improvements compared to the text-only baseline and the model utilizing only regional features. Notably, these gains represent the most promising results obtained thus far for the French language. While it appears that global features, on their own, may not offer a significant amount of additional information, our proposed approach within the Tiny model successfully harnesses their potential in conjunction with regional features. This finding reiterates the primacy of regional visual information in influencing translation precision, likely due, in part, to the effectiveness of the resampler in selecting the most relevant visual elements. However, importantly, the combination of both feature types does not appear to compromise the stability of the model, suggesting a complementary relationship between the two visual information sources.

In the case of the Small model, as evidenced in **Table 5.5**, the attempt to balance the gains achieved through the incorporation of global features with the substantial losses incurred due to regional features proves unsuccessful in maintaining model stability. Consequently, the model exhibits minimal, albeit negative, deviations from the baseline performance, registering a decline of approximately **2.7%**. This outcome aligns with the observations made for the German and Czech languages,

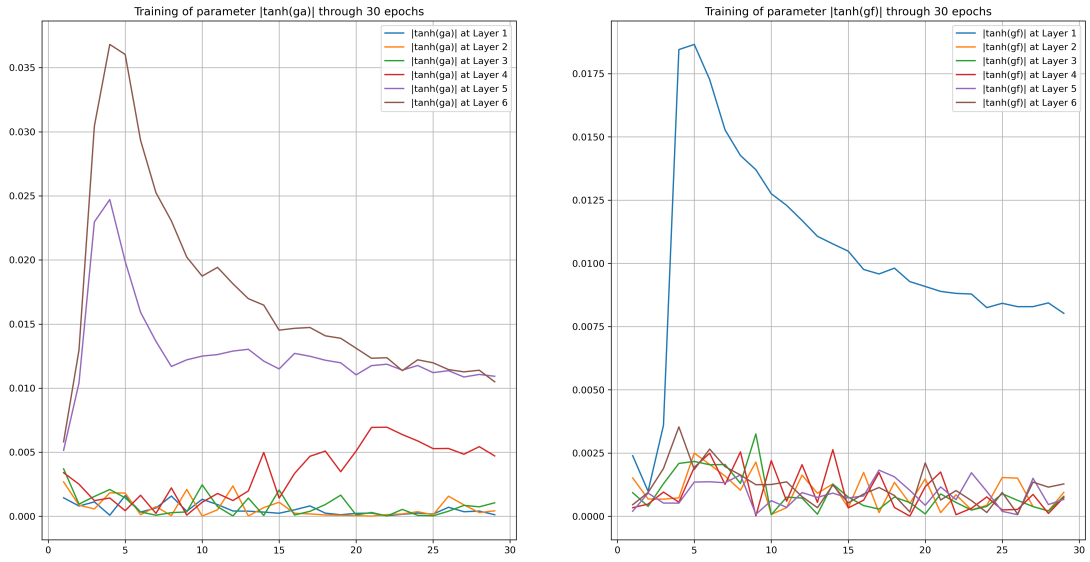

Figure 5.4: Evolution of |tanh(ga)| and |tanh(gf)| values in each layer for configuration tiny for EN → FR

suggesting that the limitations inherent to the Small architecture are a key contributing factor. The reduced capacity of this model variant appears to hinder its ability to effectively leverage the combined influence of both regional and global visual information. Further investigation is required to elucidate the underlying mechanisms at play and to explore potential strategies for mitigating these limitations in the Small model, particularly in the context of exploiting small-scale multimodal data for translation tasks.

Figure 5.4 and 5.5 provide direct evidence for the limited utility of global visual information. In the Tiny configuration, the final layers of the Vision-Text module exhibit values close to zero. This suggests that the model, during training, failed to identify any significant visual details within the images that could be meaningfully incorporated into the translation process. The model appears to rely primarily on the class token used in conjunction with regional features.

This trend persists in the Small configuration, where only the initial layer indicates some integration of image data. However, the final layer encodings remain below **1%**, signifying minimal influence of global visual information on the final translation quality. It should be noted that the small model reports the same degree of instability already seen in **Figure 5.2** for the German language.

5.5.3 English to Czech Translation

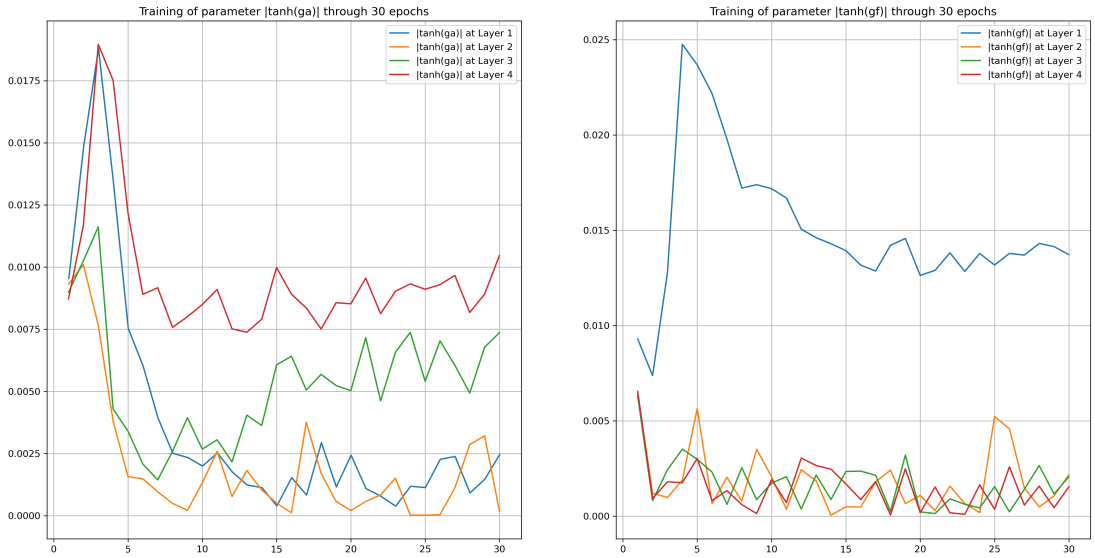

Figure 5.5: Evolution of |tanh(ga)| and |tanh(gf)| values in each layer for configuration tiny for EN → FR

In contrast to the positive outcomes observed for German and French, the Czech language dataset exhibits a diminished response to the combined application of both regional and global visual features. While improvements relative to the baseline model are still evident, the utilization of regional features appears to counteract the positive influence of global features. This phenomenon is also observed within the Tiny model configuration (**Table 5.6**). These findings suggest that further refinement of the training configuration might be necessary for the Czech language. The current configuration may not be fully optimized for the effective integration of both regional and global visual information within the chosen architecture. A deeper investigation into the training process and potential adjustments to the configuration could yield valuable insights into optimizing the model's ability to exploit the complementary nature of these visual information sources in the context of Czech language translation tasks.

This last result (**Table 5.7**) serves as a further confirmation that the Small architecture struggles to effectively process regional features. Consequently, its combination with global features yields suboptimal results, mirroring the trend observed in the Czech language dataset. This suggests that the dataset itself might not contain a sufficient number or quality of regional information elements to be optimally utilized by the model. While this configuration does manage to partially mitigate some of

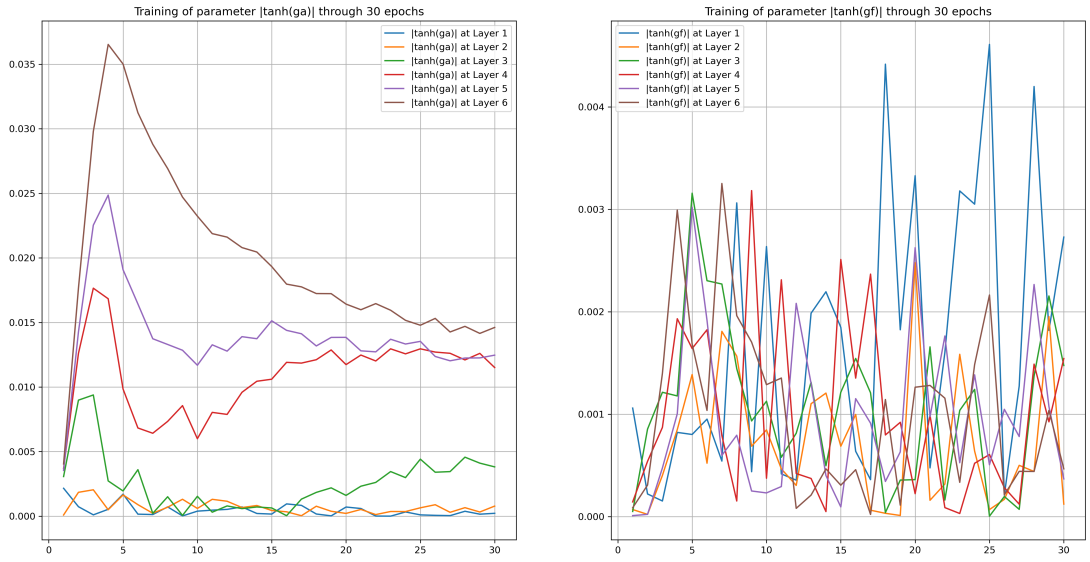

Figure 5.6: Evolution of |tanh(ga)| and |tanh(gf)| values in each layer for configuration tiny for EN → FR

the losses incurred with regional features alone, it falls short of fully exploiting its potential and achieving significant improvements over the baseline model. Further investigation into alternative training strategies or model architectures specifically tailored for datasets with limited regional information content may be necessary to unlock the full potential of this approach in the context of the Czech language.

The instability observed in the Tiny and Small configurations (**Tables 5.3 and 5.5**) is also evident for the Czech language, as illustrated in **Figures 5.5 and 5.6**. This instability likely contributes to the performance degradations observed in Czech translation. Further investigation is warranted to definitively assess the impact of global features on Czech translation quality. It is evident that global features may have a distinct influence on Czech translation compared to other languages.

Chapter 6

Conclusion and Future works

6.1 Final Consideration

This research has yielded significant findings that contribute to the ongoing advancements in **Multimodal Machine Translation (MMT)** and shed light on the complex interplay between visual information, language translation, and model architecture. The proposed architecture, inspired by [2] and [3], integrates a *classic sequence-to-sequence transformer with a visual encoder and an object detector*. This combination has achieved exceptional results, demonstrably improving performance across nearly the entire Multi30k [34] dataset.

The most substantial improvements were observed in the translation of English and French texts. By leveraging both global and object-based visual features, the model achieved an impressive increase of **8%** for German and a remarkable **12%** for French translations. Interestingly, the Czech language translations exhibited a different pattern. Improvements were solely observed when using global features, but the magnitude of improvement was even more pronounced, reaching a significant **22%**.

Our studies suggest that for expansive datasets like *Multi30k*, **smaller transformer architectures** are more suitable. These models achieve stable training without succumbing to *overfitting* within a reasonable number of epochs. Conversely, the *Small version* of the architecture we tested resulted in performance degradation compared to the text-only baseline model in most cases. This highlights the importance of tailoring model complexity to the specific characteristics of the training data.

We further investigated the influence of the **Vision-Text Layer** when integrated with regional visual features. This analysis revealed a surprising finding: the proportion of global visual information utilized by the model never surpassed **2/3%**. This suggests that a significant portion of sentences within the Multi30k dataset can be *accurately translated without the aid of a multimodal model*. Furthermore, the associated images often lack supplementary information pertaining to general characteristics. In contrast, **the impact of regional features was demonstrably more pronounced**, particularly when translating into *French*. This disparity highlights the **potential importance of focusing on specific objects and their relationships within images**, as opposed to relying solely on global scene information, for certain language pairs within MMT tasks.

6.2 Future Works

The implementation choices for our contribution made us discard many ideas that were certainly impactful, such as the use of *LMs already fully trained on larger datasets* or the usage of *masking strategies* (visual and textual) to improve and stabilize the phase of training. Since *many of these ideas have not been implemented for computational reasons*, it would certainly be interesting to see how our proposal interacts with these techniques. However, what has certainly come to light is that there is a **great need for further benchmarks for this specific task**, which to date can only be traced back to Multi30k, whose small scale we have exploited to study the results on restricted architectures but which nevertheless suggest a serious lack within the community. The further benchmark (**CoMMuTe**) also introduced by [3] is certainly interesting for having a further evaluation of our model, but since it is only **less than 200 sentences wide** it certainly cannot be used for training.
To conclude, our results show us a certainly **profitable scenario for small architectures**, possibly finding space within environments in which the **scarcity of resources is the main characteristic** such as *embedded systems* (which however use much smaller scales than these) and *real-time translation applications* that can be used in *video conferences* or to *quickly search for products in different languages*.

List of Tables

List of Figures

Acronyms

ADAM
Adaptive Moment Estimation

AI
Artificial Intelligence

ANN
Artificial Neural Network

BPE
Byte Pair Encoding

BLEU
BiLingual Evaluation Understudy

CNN
Convolutional Neural Networks

CLIP
Contrastive Language-Image Pre-Training

DL
Deep Learning

DNN
Deep Neural Networks

DETR
DEtection TRansformer

FCN

Fully Connected Networks

FCL

Fully Connected Layer

FFN

Feed Forward Network

GD

Gradient Descend

ReLU

Gaussian Error Linear Unit

KLDiv

Kullback-Leibler Divergence Loss

LSTM

Long Short-Term Memory

MAE

Mean Absolute Error

MDETR

Modulated DEtection TRansformer

METEOR

Metric for Evaluation of Translation with Explicit Ordering

MHA

Multi Head Attention

MSE

Mean Squared Error

ML

Machine Learning

MMT
Multimodal Machine Translation

MT
Machine Translation

NN
Neural Networks

NLP
Natural Language Processing

OD
Object Detection

ReLU
Rectified Linear Unit

RMSprop
Root Mean Squared Propagation

RNN
Recurrent Neural Network

SGD
Stochastic Gradient Descend

SPM
SentencePiece

VQA
Visual Question Answering

ViT
Vision Transformer

Bibliography

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964 (cit. on pp. 1, 20, 22, 24, 37, 47).

[2] Vipin Vijayan, Braeden Bowen, Scott Grigsby, Timothy Anderson, and Jeremy Gwinnup. *Adding Multimodal Capabilities to a Text-only Translation Model*. 2024. arXiv: 2403.03045 (cit. on pp. 2, 36, 37, 41–44, 47, 64).

[3] Matthieu Futeral, Cordelia Schmid, Ivan Laptev, Benoıt Sagot, and Rachel Bawden. «Tackling Ambiguity with Images: Improved Multimodal Machine Translation and Contrastive Evaluation». In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, 2023, pp. 5394–5413. DOI: 10.18653/v1/2023.acl-long.295. URL: https://aclanthology.org/2023.acl-long.295 (cit. on pp. 2, 36–38, 44, 45, 64, 65).

[4] Ozan Caglayan, Loïc Barrault, and Fethi Bougares. *Multimodal Attention for Neural Machine Translation*. 2016. arXiv: 1609.03976 (cit. on pp. 2, 35).

[5] Walter Pitts and Warren McCulloch. «A logical calculus of the ideas immanent in nervous activity,» in: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 115–133. URL: https://doi.org/10.1007/BF02478259 (cit. on p. 3).

[6] F. Rosenblatt. «The perceptron: A probabilistic model for information storage and organization in the brain». In: *Psychological Review* 65(6) (1958), pp. 386–408. URL: https://doi.org/10.1037/h0042519 (cit. on p. 3).

[7] Fardin Ghorbani, Javad Shabanpour, Sina Beyraghi, Hossein Soleimani, Homayoon Oraizi, and Mohammad Soleimani. «A deep learning approach for inverse design of the metasurface for dual-polarized waves». In: *Applied Physics A* 127.11 (2021). ISSN: 1432-0630. DOI: 10.1007/s00339-021-05030-6. URL: http://dx.doi.org/10.1007/s00339-021-05030-6 (cit. on p. 5).

[8] "EphraimX". «An Overview on Perceptron and Multilayer Perceptron Neural Network.» In: (). URL: https://dev.to/ephraimx/an-overview-of-the-perceptron-and-multilayer-perceptron-neural-network-498h (cit. on p. 6).

[9] *Cos'è la regressione logistica?* URL: https://aws.amazon.com/it/what-is/logistic-regression/ (cit. on p. 7).

[10] Matthias Ihl, Marcus Torres, Henrique Boschi-Filho, and Alfonso Ballon-Bayona. «Scalar and vector mesons of flavor chiral symmetry breaking in the Klebanov-Strassler background». In: *Journal of High Energy Physics - J HIGH ENERGY PHYS* 9 (Oct. 2010). DOI: 10.1007/JHEP09(2011)026 (cit. on p. 8).

[11] "Lakshya Khandelwal". *"A Short Introduction to Activation Functions"*. 2018. URL: https://medium.com/@lakshaya.khandelwal/a-short-introduction-to-activation-functions-4394e6538bb3 (cit. on p. 8).

[12] Ning Chen, Xinkai Ma, Haixia Luo, Jun Peng, Shangzhu Jin, Xiao Wu, and Yongsheng Zhou. «Stone segmentation based on improved U-Net network». In: *Signal, Image and Video Processing* (May 2024), pp. 1–14. DOI: 10.1007/s11760-024-03201-5 (cit. on p. 9).

[13] Liu Jia-Qi, Feng Yun-Wen, Teng Da, Chen Jun-Yu, and Lu Cheng. «Operational reliability evaluation and analysis framework of civil aircraft complex system based on intelligent extremum machine learning model». In: *Reliability Engineering System Safety* 235 (2023), p. 109218. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2023.109218. URL: https://www.sciencedirect.com/science/article/pii/S0951832023001333 (cit. on p. 10).

[14] *Optimizers in Deep Leaning*. 2021. URL: https://musstafa0804.medium.com/optimizers-in-deep-learning-7bf81fed78a0 (cit. on p. 13).

[15] Konstantinos Demertzis, Stavros Demertzis, and Lazaros Iliadis. «A Selective Survey Review of Computational Intelligence Applications in the Primary Subdomains of Civil Engineering Specializations». In: *Applied Sciences* 13.6 (2023). ISSN: 2076-3417. DOI: 10.3390/app13063380. URL: https://www.mdpi.com/2076-3417/13/6/3380 (cit. on p. 15).

[16] Meng Li, Tao Chen, Zhihua Li, and Hezi Liu. «An Efficient Crowd Density Estimation Algorithm Through Network Compression». In: *Traffic and Granular Flow 2019*. Ed. by Iker Zuriguel, Angel Garcimartin, and Raul Cruz. Cham: Springer International Publishing, 2020, pp. 165–173 (cit. on p. 16).

[17] Elizabeth Endri, Alaa Sheta, and Hamza Turabieh. «Road Damage Detection Utilizing Convolution Neural Network and Principal Component Analysis». In: *International Journal of Advanced Computer Science and Applications* 11 (Jan. 2020). DOI: 10.14569/IJACSA.2020.0110682 (cit. on p. 17).

[18] Hossein Gholamalinejad and Hossein Khosravi. *Pooling Methods in Deep Neural Networks, a Review*. Sept. 2020 (cit. on p. 17).

[19] Mohamed K. Hassanin. *Mastering Search and Retrieval in LLMs: Unveiling the Power of the RAG Pattern: Part 1*. 2024. URL: https://www.linkedin.com/pulse/mastering-search-retrieval-llms-unveiling-power-rag-pattern-hassanin-qonnc/ (cit. on p. 18).

[20] Philip Gage. «A new algorithm for data compression». In: *The C Users Journal* 12(2) (1994), pp. 23–38. URL: https://dl.acm.org/doi/10.5555/177910.177914 (cit. on p. 19).

[21] Taku Kudo and John Richardson. «SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing». In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Ed. by Eduardo Blanco and Wei Lu. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 66–71. DOI: 10.18653/v1/D18-2012. URL: https://aclanthology.org/D18-2012 (cit. on pp. 19, 26).

[22] Daniel Szelogowski. *Deep Learning for Musical Form: Recognition and Analysis*. June 2022. DOI: 10.31237/osf.io/ts27q (cit. on p. 21).

[23] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. «Sequence to sequence learning with neural networks». In: NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 3104–3112. URL: http://arxiv.org/abs/1409.3215 (cit. on p. 21).

[24] Zelun Wang and Jyh-Charn Liu. «Translating math formula images to LaTeX sequences using deep neural networks with sequence-level training». In: *International Journal on Document Analysis and Recognition (IJDAR)* 24 (June 2021), pp. 1–13. DOI: 10.1007/s10032-020-00360-2 (cit. on p. 23).

[25] Tommy Adeliyi. *The ABC of Self-Attention Mechanism: Simplifying AI's Game-Changer*. 2024. URL: https://medium.com/@tommyadeliyi/the-abc-of-self-attention-mechanism-simplifying-ais-game-changer-e2346a8a1f79 (cit. on p. 25).

[26] Iveta Luptáková, Martin Kubovčík, and Jiri Pospichal. *Wearable Sensor Based Human Activity Recognition with Transformer*. Feb. 2022. DOI: 10.20944/preprints202202.0111.v1 (cit. on p. 25).

[27] Alexey Dosovitskiy et al. «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale». In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929 (cit. on p. 27).

[28] Ananya Jain, Aviral Bhardwaj, Kaushik Murali, and Isha Surani. *A Comparative Study of CNN, ResNet, and Vision Transformers for Multi-Classification of Chest Diseases*. 2024. arXiv: 2406.00237 (cit. on p. 28).

[29] DeepLobe.ai. *Exploring Object Detection Applications and Benefits*. URL: https://deeplobe.ai/exploring-object-detection-applications-and-benefits/ (cit. on p. 29).

[30] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. «End-to-End Object Detection with Transformers». In: *CoRR* abs/2005.12872 (2020). arXiv: 2005.12872. URL: https://arxiv.org/abs/2005.12872 (cit. on pp. 29, 30).

[31] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 (cit. on pp. 31, 37).

[32] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. *MDETR – Modulated Detection for End-to-End Multi-Modal Understanding*. 2021. arXiv: 2104.12763 (cit. on pp. 31–33, 37).

[33] Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. «A Shared Task on Multimodal Machine Translation and Crosslingual Image Description». In: *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. Ed. by Ondřej Bojar et al. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 543–553. DOI: 10.18653/v1/W16-2346. URL: https://aclanthology.org/W16-2346 (cit. on p. 35).

[34] Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. *Multi30K: Multilingual English-German Image Descriptions*. 2016. arXiv: 1605.00459 (cit. on pp. 35, 36, 64).

[35] Iacer Calixto, Qun Liu, and Nick Campbell. «Doubly-Attentive Decoder for Multi-modal Neural Machine Translation». In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Regina Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1913–1924. DOI: 10.18653/v1/P17-1175. URL: https://aclanthology.org/P17-1175 (cit. on p. 35).

[36] Hasan Sait Arslan, Mark Fishel, and Gholamreza Anbarjafari. *Doubly Attentive Transformer Machine Translation*. 2018. arXiv: 1807.11605 (cit. on p. 35).

[37] Bei Li, Chuanhao Lv, Zefan Zhou, Tao Zhou, Tong Xiao, Anxiang Ma, and JingBo Zhu. «On Vision Features in Multimodal Machine Translation». In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6327–6337. DOI: 10.18653/v1/2022.acl-long.438. URL: https://aclanthology.org/2022.acl-long.438 (cit. on pp. 35, 36, 43).

[38] Zhiyong Wu, Lingpeng Kong, Wei Bi, Xiang Li, and Ben Kao. «Good for Misconceived Reasons: An Empirical Revisiting on the Need for Visual Context in Multimodal Machine Translation». In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli. Online: Association for Computational Linguistics, Aug. 2021, pp. 6153–6166. DOI: 10.18653/v1/2021.acl-long.480. URL: https://aclanthology.org/2021.acl-long.480 (cit. on pp. 35, 43).

[39] Xuxin Cheng, Zhihong Zhu, Yaowei Li, Hongxiang Li, and Yuexian Zou. «DAS-CL: Towards Multimodal Machine Translation via Dual-Level Asymmetric Contrastive Learning». In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. CIKM '23. Birmingham, United Kingdom: Association for Computing Machinery, 2023, pp. 337–347. ISBN: 9798400701245. DOI: 10.1145/3583780.3614832. URL: https://doi.org/10.1145/3583780.3614832 (cit. on p. 36).

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423 (cit. on pp. 36, 37).

[41] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. *VisualBERT: A Simple and Performant Baseline for Vision and Language*. 2019. arXiv: 1908.03557 (cit. on pp. 36, 37).

[42] ShaoDong Cui, Kaibo Duan, Wen Ma, and Hiroyuki Shinnou. «Dose multimodal machine translation can improve translation performance?» In: *Neural Computing and Applications* (Apr. 2024), pp. 1–12. DOI: 10.1007/s00521-024-09705-y (cit. on p. 37).

[43] Jean-Baptiste Alayrac et al. *Flamingo: a Visual Language Model for Few-Shot Learning*. 2022. arXiv: 2204.14198 (cit. on pp. 38, 41, 42, 44).

[44] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 (cit. on p. 46).

[45] Marcin Junczys-Dowmunt et al. *Marian: Fast Neural Machine Translation in C++*. 2018. arXiv: 1804.00344 (cit. on p. 46).

[46] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 (cit. on p. 46).

[47] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. «BLEU: a method for automatic evaluation of machine translation». In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135 (cit. on p. 47).

[48] Alon Lavie and Abhaya Agarwal. «Meteor: an automatic metric for MT evaluation with high levels of correlation with human judgments». In: *Proceedings of the Second Workshop on Statistical Machine Translation*. StatMT '07. Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 228–231 (cit. on p. 47).