UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Adaptation of Large Language Models for Streaming Machine Translation

End of Degree Project

Bachelor's Degree in Informatics Engineering

AUTHOR: Vicente Hungerbuhler, Daniel Kyu

Tutor: Juan Císcar, Alfonso

Cotutor: Civera Saiz, Jorge

Experimental director: Iranzo Sánchez, Jorge

ACADEMIC YEAR: 2023/2024

Resum

La traducció automàtica (MT, de l'anglés) és un àmbit fonamental dins de l'aprenentatge automàtic, on les xarxes neuronals han assolit avanços significatius que han impulsat la recerca en MT. Això s'ha vist encoratjat encara més per la proliferació de plataformes que faciliten la transmissió de contingut audiovisual en streaming (per exemple, YouTube, Twitch) i videoconferències (per exemple, Zoom, Webex). Aquestes plataformes han posat de manifest la necessitat d'adaptar els models i tècniques de MT convencionals al context de transmissió en temps real, és a dir, per a una entrada contínua (en streaming) i sota un temps de resposta determinat (latència). Els avanços en l'entrenament de grans xarxes neuronals amb col·leccions massives de dades per part dels principals proveïdors tecnològics, com ara Google, Meta i Microsoft, han conduït a l'aparició de models de MT multilingüe i grans models de llenguatge que poden ser utilitzats com a models fonamentals per abordar tasques posteriors específiques. En aquest context, aquest treball aprofundeix en l'avaluació de l'eficàcia dels models fonamentals quan s'adapten a tasques de MT, particularment per a MT en streaming. Per a això, utilitzarem dades, tecnologia i experiència del grup MLLP de VRAIN, adquirits en el marc de projectes de recerca i transferència de tecnologia desenvolupats en els últims anys.

Paraules clau: Aprenentatge automàtic; Xarxes neuronals; Traducció automàtica; Traducció automàtica en streaming; Models de llenguatge grans; Adaptació de models.

Resumen

La traducción automática (MT, del inglés) es un área fundamental dentro del aprendizaje automático, donde las redes neuronales han alcanzado avances significativos que han impulsado la investigación en MT. Esto se ha visto fomentado aún más por la proliferación de plataformas que facilitan la transmisión de contenido audiovisual en streaming (por ejemplo, YouTube, Twitch) y videoconferencias (por ejemplo, Zoom, Webex). Estas plataformas han puesto de manifiesto la necesidad de adaptar los modelos y técnicas de MT convencionales al contexto de transmisión en tiempo real, esto es, para una entrada continua (en streaming) y bajo un tiempo de respuesta determinado (latencia). Los avances en el entrenamiento de grandes redes neuronales con colecciones masivas de datos por parte de los principales proveedores tecnológicos, como Google, Meta y Microsoft, han conducido a la aparición de modelos de MT multilingüe y grandes modelos de lenguaje que pueden usarse como modelos fundacionales para abordar tareas posteriores específicas. En este contexto, este trabajo profundiza en la evaluación de la eficacia de los modelos fundacionales cuando se adaptan a tareas de MT, particularmente para MT en streaming. Para ello, utilizaremos datos, tecnología y experiencia del grupo MLLP de VRAIN, adquiridos en el marco de proyectos de investigación y transferencia de tecnología desarrollados en los últimos años.

Palabras clave: Aprendizaje automático; Redes neuronales; Traducción automática; Traducción automática en streaming; Modelos de lenguaje grandes; Adaptación de modelos

Abstract

Machine translation (MT) stands as a pivotal domain within machine learning, where the rise of neural networks has sparked significant advancements, propelling MT into a highly researched field. This has been further fostered by the proliferation of platforms facilitating audiovisual content streaming (e.g., YouTube, Twitch) and video conferencing (e.g., Zoom, Webex). These platforms have underscored the necessity of adapting conventional MT models and techniques to accommodate real-time streaming scenarios, which entails a continuous input stream to be translated under a given response time (latency). Progress in training large neural networks on massive collections of data by major technological providers, such as Google, Meta, and Microsoft, has led to the emergence of multilingual MT and large language models, which can be used as foundational models to tackle specific downstream tasks. In this context, this study delves into evaluating the efficacy of foundational models when adapted to MT tasks, particularly for streaming MT. To this end, we will make use of data, technology, and expertise from the MLLP group of VRAIN, acquired within the framework of research and technology transfer projects developed in recent years.

Key words: Machine learning; Neural networks; Machine translation; Streaming machine translation; Large language models; Model adaptation

Contents

Appendix

List of Figures

List of Tables

List of Acronyms

MT Machine Translation

LLM Large Language Model

AI Artificial Intelligence

ML Machine Learning

NLP Natural Language Processing

MAE Mean Absolute Error

MSE Mean Square Error

CE Cross-Entropy

ReLU Rectified Linear Unit

PLM Pre-trained Language Model

DP Data Parallelism

TP Tensor Parallelism

PP Pipeline Parallelism

ZeRO Optimizer Parallelism

PEFT Parameter-Efficient Fine-Tuning

LoRA Low-Rank Adaptation

SMT statistical machine translation

NMT Neural machine translation

BLEU Bilingual Evaluation Understudy

BP brevity penalty

COMET Crosslingual Optimized Metric for Evaluation of Translation

ASR Automatic Speech Recognition

NLLB No Language Left Behind

OOV Out-Of-Vocabulary

VRAIN Valencian Research Institute for Artificial Intelligence

AP Average Proportion

AL Average Lagging

DAL Differentiable Average Lagging

SLT Spoken Language Translation

EOS End-Of-Sentence

CHAPTER 1
Introduction

This study delves into the current state of machine translation (MT), particularly focusing on Streaming MT, and explores the possibilities, to this end, of the massively scaled-up models known as Large Language Models (LLMs) that are emerging in the machine learning ecosystem. Their potential for adaptation to the streaming context is examined, which opens a promising path of cost-effective techniques to obtain adapted, fully-functional models. These are compared to a fully-trained, state-of-the-art streaming bilingual MT baseline model, in order to determine the degree to which LLMs can genuinely produce competitive outcomes in streaming MT.

In this chapter, the motivation and context of this study are explored in greater detail, objectives are delineated, and an overview of the document's structure is presented.

1.1 Motivation

In today's interconnected global landscape, where individuals from different countries and cultural backgrounds must collaborate and communicate, language poses a significant barrier. Machine translation, empowered by neural network advancements, has become an essential tool in bridging these linguistic gaps. The current social and technological landscape has given rise to audiovisual content streaming platforms such as Youtube or Twitch, that allow anyone from anywhere in the world to consume a given content produced in real-time, and video conferencing tools such as Zoom and Webex, facilitating meetings with participants dispersed across the globe.

Existing MT models and methods are generally designed to handle entire sentences, processing them in batches rather than in real-time, incrementally processing each word. As a result, these models and traditional MT methods are not well-suited for streaming scenarios and must be adapted for these environments, where low latency is crucial. Therefore, the predominant strategy to achieve state-of-the-art performance is to train new models from scratch [1]. While this allows for tailored modeling efforts specific to streaming cases, the process can be computationally intensive and hindered by the limited availability of data specifically suited for streaming purposes.

This raises the question of whether using pre-trained models with massive amounts of data — the so-called 'Large Language Models' (LLMs) or foundational models [2] — developed by major tech companies such as Google, Meta, and Microsoft can be made feasible with fine-tuning or similar techniques.

These methods significantly reduce the training process costs and could yield promising results [3]. The development of multilingual MT and LLMs offers a strong foundation for specialized MT applications. Their adaptation allows leveraging the cumulative knowledge embedded in large-scale datasets, that cover various domains and languages providing the adapted model with a rich foundation to start learning new tasks.

1.2 Objectives

In this context, our study aims to assess the effectiveness of foundational models for streaming MT. Our objectives are twofold: firstly, to assess the adaptability of foundational models to streaming environments, which implies the processing of a continuous input stream to be translated under a given response time latency; and secondly, to explore their potential for bespoken solutions tailored to the unique challenges of real-time translation, comparing them with state-of-the-art streaming MT models developed by the MLLP group.

The primary objectives of this study are:

1. To acquire and process data appropriately for the streaming context.
2. To adapt and assess the efficacy of publicly accessible LLMs for streaming MT.
3. To compare and evaluate the effectiveness and results of the adapted models against baseline state-of-the-art streaming MT models.

1.3 Document structure

The document consists of 5 chapters. Chapter 1 has provided a concise introduction of the work, offering context and outlining its objectives. Chapter 2 familiarizes readers with essential terminology and concepts necessary to grasp the objectives of the study, providing a concise introduction to the domain of Machine Learning, exploring the transformer architecture that is the foundation of most MT models, discussing the development of LLMs and their adaptation possibilities, and delving into the field of modern MT and specifically, the particularities of Streaming MT. Chapter 3 explores the outcomes obtained from the adaptation of large multilingual models tailored specifically for offline bilingual MT, examining the nuances of the adaptation process, and detailing the strategies employed and the corresponding performance metrics achieved. Chapter 4 ventures into the realm of streaming MT, adjusting the adaptation of the large multilingual models realized in the previous section to this new task, and evaluates the results obtained. Chapter 5 concludes with a general overview of the conclusions and findings derived from this study, together with a review on potential future paths of research. Appendix A is also added to provide additional figures related to training procedures and experimentation results.

CHAPTER 2
Background

In this chapter, the fundamental concepts, ideas, and terminology related to MT are presented. An overview of machine learning is presented, followed by a comprehensive examination of the transformer architecture, the primary model architecture employed in MT tasks. These parts draw extensively from [4]. Subsequently, the concept of LLMs is delved into, exploring the adaptation techniques available and the methods to make them computationally feasible. Finally, an in-depth explanation of MT is provided, with a specific focus on streaming MT.

2.1 Machine Learning

Artificial intelligence (AI) aims to develop systems that mimic intelligent behaviors. One of its branches, machine learning (ML), focuses on training models to make decisions by analyzing data patterns through probabilistic modeling. These models seek to generalize datasets, predicting outcomes that closely match desired results when presented with new data. Achieving this entails employing algorithms that adaptively learn the most effective parameters or weights, influencing the system's output.

ML problems are typically classified based on the type of output the system produces, into *classification* (predicting discrete classes) or *regression* problems (predicting continuous values), or based on the availability of an output label for given input data during the training process. **Supervised learning**, commonly utilized in *Natural Language Processing (NLP)* tasks, including MT, uses labeled data to train models to predict outputs accurately.

Deep neural networks, a type of ML models that have profoundly impacted the field, excel at representing complex functions capable of capturing intricate data patterns and dependencies, enabling these networks to model relationships that are not easily described by simple linear functions. Deep neural networks have the capability to handle inputs of large size, variable length, and containing diverse internal structures. They can produce single real numbers (*regression*), multiple real numbers (*multivariate regression*), or probabilities across two or more classes (*binary* and *multiclass classification*, respectively).

2.1.1. Supervised Learning

The objective of supervised learning is to construct a model that takes an *input* $\boldsymbol{x}$ and generates a *prediction* $\boldsymbol{y}$. Both $\boldsymbol{x}$ and $\boldsymbol{y}$ are usually vectors of fixed size. The model is defined by *parameters* $\boldsymbol{\phi}$, which determine the specific relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$. The prediction $\boldsymbol{y}$ from the input $\boldsymbol{x}$ is called *inference*, and can formally be represented as:

$$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\phi}) \tag{2.1}$$

The *training* process of a model involves optimizing these parameters using a training dataset of input-output pairs (x_i, y_i) to minimize a chosen loss function $L(\boldsymbol{\phi})$ that quantifies the discrepancy between the model's predictions and the actual target values in the training data. Thus our goal during training is to find parameters $\hat{\boldsymbol{\phi}}$ that minimize this loss function:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\, L(\boldsymbol{\phi}) \tag{2.2}$$

Different loss functions can be used depending on the case. For regression problems, the mean absolute error (**MAE**) or L1

$$L(\boldsymbol{\phi}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(x_i, \boldsymbol{\phi})| \tag{2.3}$$

or the mean square error (**MSE**) or L2

$$L(\boldsymbol{\phi}) = \frac{1}{n}\sum_{i=1}^{n} (y_i - f(x_i, \boldsymbol{\phi}))^2 \tag{2.4}$$

are usually used, where $f(x_i, \boldsymbol{\phi})$ are the predicted values and y_i the real values.

For classification problems, like MT, the **Cross-Entropy (CE)** $H(p, q)$, which quantifies dissimilarity between two distributions, the true probability distribution p and the model's predicted probability distribution q, is commonly employed [5]. It is defined as:

$$H(p, q) = \sum_{x \in \mathcal{X}} p(x) \log(q(x)) \tag{2.5}$$

Gradient descent is typically used to iteratively adjust parameters towards minimizing the loss, which involves initially selecting parameters at random and subsequently improving them by walking down the loss function until its minimum is reached. One method to accomplish this is to measure the gradient of the surface at the current position and adjusting the parameters in the direction of the steepest descent. This iterative process continues until the gradient flattens out, signifying that further improvements are not feasible, as seen in Figure 2.1.

After training the model, we *evaluate* its real-world performance by computing loss on a separate test dataset. The accuracy on test data depends on the adequacy of the training data's representation and completeness, as well as the model's complexity. A simple model, like a straight line, may *underfit* and miss the true input-output relationship. Conversely, an overly complex model can *overfit* by capturing irrelevant statistical nuances from the training data, leading to unusual predictions.

2.1.2. Shallow Neural Networks

The 1D linear regression model, expressed as $\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\phi}) = \phi_0 + \phi_1 \boldsymbol{x}$, defines a simple relationship between input $\boldsymbol{x}$ and output $\boldsymbol{y}$ using parameters $\boldsymbol{\phi} = [\phi_0, \phi_1]^T$. Varying ϕ_0 and ϕ_1 alters this relationship, as seen in Figure 2.2.

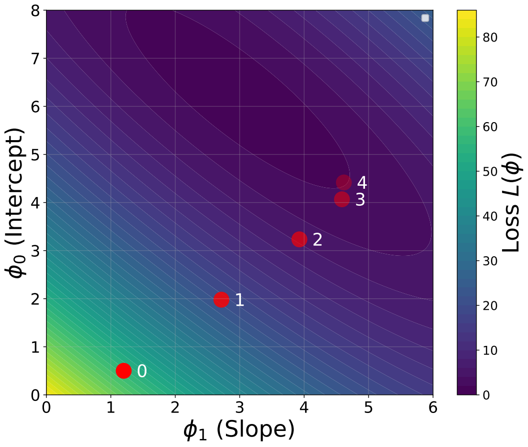

Figure 2.1: Gradient Descent. Iterative training algorithms initialize the parameters randomly and then improve them by moving towards lower loss values (indicated by darker regions) until no further improvement is possible. In this illustration, we start at point 0 and move downhill (perpendicular to the contour lines) to reach point 1. We then reassess the downhill direction and proceed to point 2. This iterative process continues until we reach the minimum of the function.

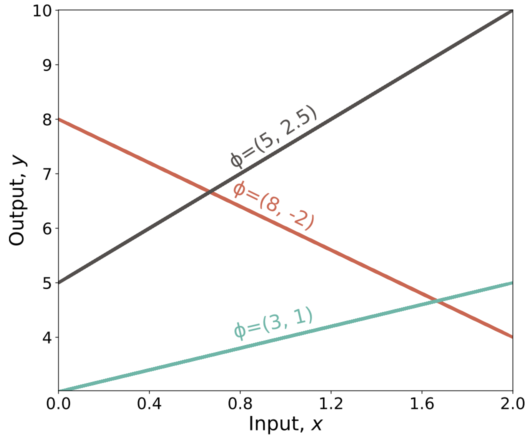

Figure 2.2: Linear Regression Model. With a specific selection of parameters $\boldsymbol{\phi} = [\phi_0, \phi_1]^T$, the model generates output predictions (y-axis) corresponding to input values (x-axis).

This linear model's limitation prompts exploration into **shallow neural networks**, which, with their piecewise linear functions, can model more complex input-output relationships and can approximate any continuous function with arbitrary precision. Unlike linear regression, these networks incorporate hidden units $\boldsymbol{h}$ and nonlinear activation functions like ReLU to handle multi-dimensional inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$. Following this, *deep neural networks* emerge, equally expressive but successful in depicting complex functions with fewer parameters, leading to better performance.

Shallow neural networks are functions $\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\phi})$ with parameters $\boldsymbol{\phi}$ that map multivariate inputs $\boldsymbol{x} \in \mathbb{R}^{D_i}$ to multivariate outputs $\boldsymbol{y} \in \mathbb{R}^{D_o}$ using $\boldsymbol{h} \in \mathbb{R}^{D}$ hidden units. Each hidden unit is calculated as:

$$h_d = a\left(\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i\right) \tag{2.6}$$

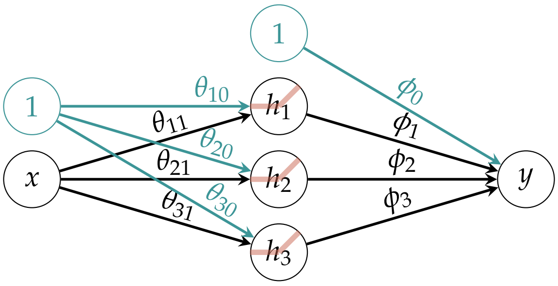

Figure 2.3: Shallow neural network. Input x on the left, hidden units $h1$, $h2$, and $h3$ in the center, and output y on the right. Computation flows left to right, with each of the ten arrows representing a parameter, either an intercept (orange) or a slope (black). Parameters are multiplied by their source and added to their target. ReLU functions are applied at the hidden units.

and these are combined linearly to generate the output:

$$y_j = \phi_{j0} + \sum_{d=1}^{D} \phi_{jd} h_d \tag{2.7}$$

where $a(\cdot)$ is a nonlinear *activation function*, and the model has parameters $\boldsymbol{\phi} = \{\theta_{di}, \phi_{jd}\}$. Neural networks are commonly referred to as having *layers*, and the neural network in Figure 2.3 has one input layer, three hidden layers, and one output layer. The hidden units are usually referred to as *neurons*. Then, what characterizes shallow neural networks is that they have 1 hidden unit.

The most common **activation function** is the *Rectified Linear Unit (ReLU)* [6], as illustrated in Figure 2.4, primarily due to its ease of interpretability. It allows the model to capture nonlinear relationships by partitioning the input space into regions with distinct linear functions.

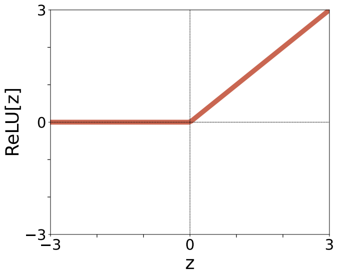

Figure 2.4: Rectified Linear Unit (ReLU) is an activation function that outputs zero if the input is negative, and otherwise returns the input unchanged. Essentially, it truncates negative values to zero.

Hence, the computation flow, seen for the previous network of Figure 2.3 and illustrated for a given function in Figure 2.5, involves each hidden unit containing a linear function $\theta_{d0} + \theta_{d1}x$ of the input, producing a line that is then clipped by the ReLU function $a(\cdot)$ when it falls below zero. The positions where the three lines intersect zero serve as the three "joints" in the final output. Subsequently, the three rectified lines are weighted by ϕ_1, ϕ_2, and ϕ_3, respectively. Finally, the offset ϕ_0 is added, which adjusts the overall height of the final function.

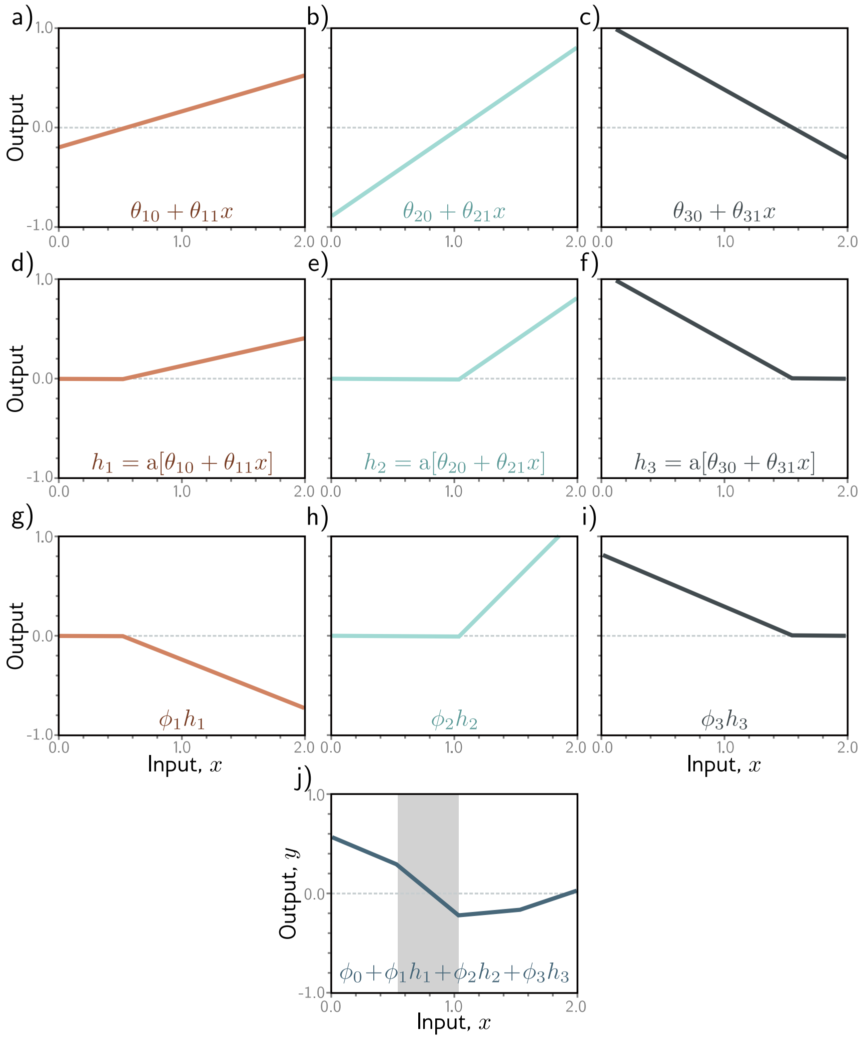

Figure 2.5: Computation flow of a shallow neural network: a–c) Input x is processed through three distinct linear functions, each with a unique y-intercept $\theta_{\bullet 0}$ and slope $\theta_{\bullet 1}$. d–f) Each resulting line is subjected to the ReLU activation function, which truncates negative values to zero. g–i) The three rectified lines are then scaled by the respective weights ϕ_1, ϕ_2, and ϕ_3. j) The scaled and rectified functions are added, and an offset ϕ_0, governing the overall height, is introduced. Each of the four linear regions corresponds to a distinct activation pattern in the hidden units. Within the shaded region, h_2 remains inactive (clipped), while both h_1 and h_3 are active. Obtained from [4].

2.1.3. Deep Neural Networks

As the number of hidden units increases, shallow neural networks enhance their descriptive capacity, allowing them to describe complex functions in high-dimensional spaces. However, the required number of hidden units can become excessively large. **Deep neural networks**, with more than one hidden layer, can generate significantly more linear regions compared to shallow networks with the same number of parameters. Therefore, deep networks can capture a wider range of functions effectively.

Let's consider a deep network comprising two hidden layers, with each layer housing three hidden units as seen in Figure 2.6. The first layer is defined by:

$$h_d = a(\theta_{d0} + \theta_{d1}x), \quad \text{for } d = 1,2,3 \tag{2.8}$$

the second layer by:

$$h'_d = a(\psi_{d0} + \sum_{i=1}^{3} \psi_{di} h_i), \quad \text{for } d = 1, 2, 3 \tag{2.9}$$

and the output by:

$$y' = \phi'_0 + \sum_{i=1}^{3} \phi'_i h'_i. \tag{2.10}$$

Taking into account these equations, a new perspective emerges on how the network constructs a progressively complex function, illustrated in Figure 2.7:

1. The first layer computes three hidden units h_1, h_2, and h_3 by forming linear functions of the input and passing them through ReLU activation functions (see Equation 2.8).

2. The second layer forms new linear functions of these hidden units, passed through activation functions (see Equation 2.9), effectively creating a shallow network with three piecewise linear functions.

3. At the second hidden layer, another ReLU function $a[\cdot]$ is applied to each function (see Equation 2.9), which clips them and adds new "joints" to each.

4. The final output is a linear combination of these hidden units (see Equation 2.10).

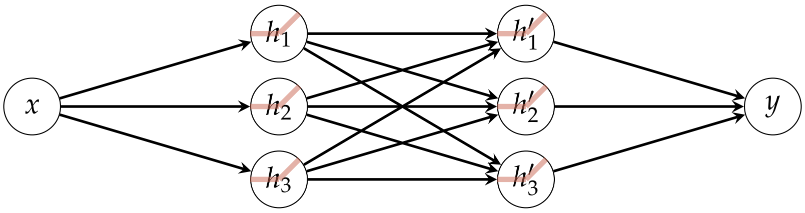

Figure 2.6: Neural network with one input, one output, and two hidden layers, each with three hidden units.

Deep neural networks also introduce the concept of *hyperparameters*, such as the number of hidden units in each layer (network *width*), the number of hidden layers (*depth*), and the total number of hidden units (network *capacity*). They are easier to fit and better at generalizing to new data than shallow networks, which has driven advancements in image recognition and natural language processing. However, challenges like training complexity and data requirements persist. The next chapter explores transformer architectures, which leverage self-attention mechanisms to achieve remarkable results in natural language processing and modern MT.

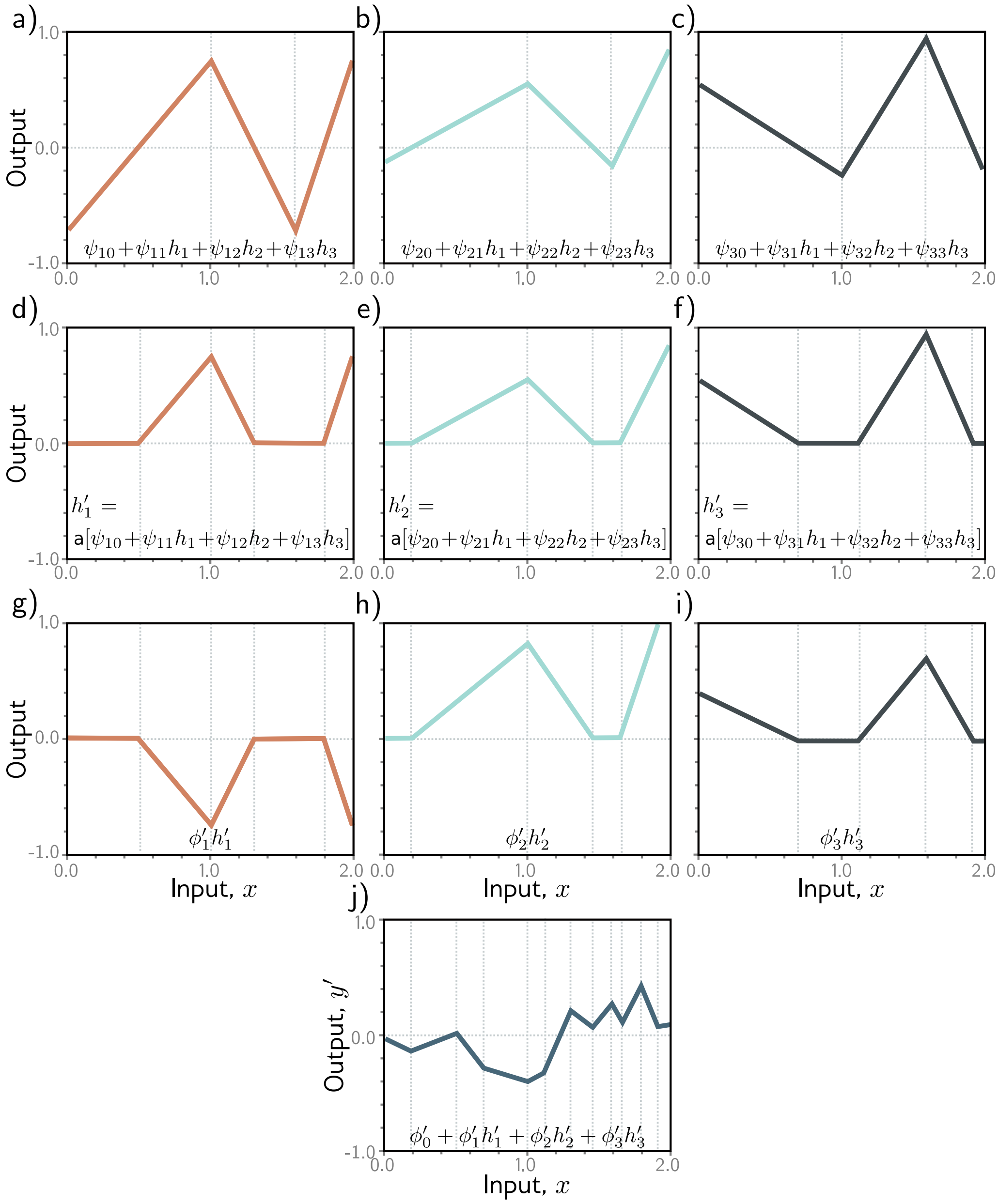

Figure 2.7: Computation flow for the deep neural network in Figure 2.6: a–c) The inputs to the second hidden layer (pre-activations) comprise three piecewise linear functions, with the "joints" between linear regions located at identical positions. d–f) Each piecewise linear function undergoes clipping through the ReLU activation function. g–i) These clipped functions are weighted using parameters ϕ'_1, ϕ'_2, and ϕ'_3, respectively. j) Finally, the clipped and weighted functions are summed, and an offset ϕ'_0, adjusting the overall height, is added. Obtained from [4].

2.2 Transformers

The **transformers** architecture [7] resolved significant limitations found in MT systems predating its invention, which struggled with adapting to varying sequence lengths found in language datasets, establishing connections between various words in a text, and modeling long-distance word relationships [8]. All these issues called for a new model architecture capable of addressing them, and the introduction of the transformer architecture marked a groundbreaking advancement in the field of MT.

2.2.1. Self-attention

The transformer architecture revolves around the concept of the **dot-product self-attention**. A self-attention block $\mathbf{sa}(\bullet)$ computes values for each input word, and then forms weighted sums that are influenced by similarities between words. This mechanism allows for efficient handling of lengthy input sequences of varying lengths, although it's important to note that it scales quadratically with sequence length N due to the pairwise comparisons involved. Nevertheless, transformers use a single parameter set regardless of input size, making them highly scalable.

If we have N inputs $x_1, \ldots, x_N$, each with dimensions $D \times 1$, the self-attention block produces N output vectors of identical size. In the context of NLP and MT, each input represents a word or word fragment. Then, a set of **values** are computed for each input as

$$\mathbf{v}_j = \boldsymbol{\beta}_v + \boldsymbol{\Omega}_v \boldsymbol{x}_j, \tag{2.11}$$

where $\beta_v \in \mathbb{R}^D$ and $\Omega_v \in \mathbb{R}^{D \times D}$ represent biases and weights of the transformer model, respectively.

Then, the i^{th} output $\mathbf{sa}_i(x_1, \ldots, x_N)$ is computed as a weighted sum of all the values $\mathbf{v}_1, \ldots, \mathbf{v}_N$:

$$\mathbf{sa}_i(x_1, \ldots, x_N) = \sum_{j=1}^{N} a(x_j, x_i)\mathbf{v}_j. \tag{2.12}$$

Here, the scalar weight $a(x_j, x_i)$ represents the **attention** that the i^{th} output directs toward input x_j. The N weights $a(x_j, x_i)$ are non-negative and sum to one. Thus, self-attention can be interpreted as routing the values in varying proportions to create each output, illustrated in Figure 2.8.

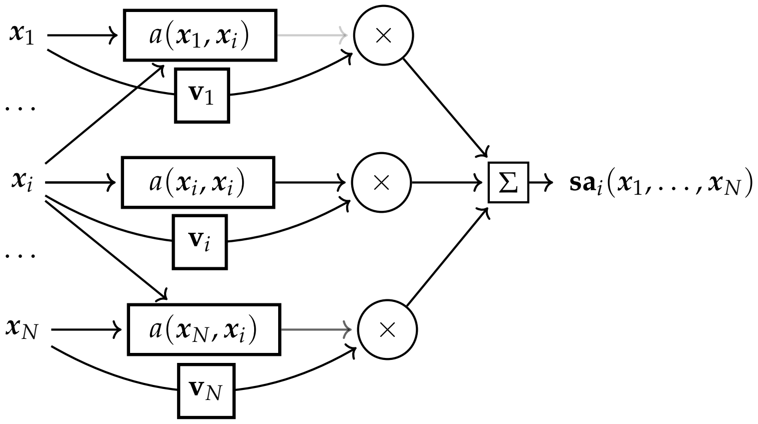

Figure 2.8: Self-attention processes N inputs $x_1, \ldots, x_N \in \mathbb{R}^D$ individually to compute N value vectors. The i^{th} output $\mathbf{sa}_i(x_1, \ldots, x_N)$ is then determined as a weighted sum of these N value vectors, with weights that are positive and sum to one.

To compute attention, the inputs undergo two linear transformations, resulting in $\boldsymbol{q}_n$ (queries) and $\boldsymbol{k}_m$ (keys):

$$\begin{aligned} \boldsymbol{q}_n &= \boldsymbol{\beta}_q + \boldsymbol{\Omega}_q \boldsymbol{x}_n \\ \boldsymbol{k}_m &= \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \boldsymbol{x}_m, \end{aligned} \tag{2.13}$$

Dot products between queries and keys are then computed and passed through a softmax function (which transforms the vector containing N real numbers into a probability distribution across N potential outcomes, such that $\sigma : \mathbb{R}^N \rightarrow (0,1)^N$):

$$a(\boldsymbol{x}_j, \boldsymbol{x}_i) = \sigma_j\left(\boldsymbol{k}^T \boldsymbol{q}_i\right) = \frac{\exp(\boldsymbol{k}_j^T \boldsymbol{q}_i)}{\sum_{n=1}^{N} \exp(\boldsymbol{k}_n^T \boldsymbol{q}_i)}, \tag{2.14}$$

ensuring that for each $\boldsymbol{x}_n$, the attention weights are positive and sum to one. This process, known as **dot-product self-attention**, measures similarity between inputs, crucially affecting how information flows through the network. Large dot products between queries and keys may lead to a dominance of the largest value in the softmax function, resulting in minimal changes in output for small input variations and challenging training due to negligible gradients. To counter this, dot products are scaled by the square root of the dimension D_q of queries and keys (i.e., the number of rows in $\boldsymbol{\Omega}_q$ and $\boldsymbol{\Omega}_k$, which are identical), acting as a normalization step.

The mechanism's nonlinearity arises from the dot-product and softmax operations. This mechanism uses a single set of parameters $\boldsymbol{\phi} = \{\boldsymbol{\beta}_v, \boldsymbol{\Omega}_v, \boldsymbol{\beta}_q, \boldsymbol{\Omega}_q, \boldsymbol{\beta}_k, \boldsymbol{\Omega}_k\}$, independent of input size N, and establishes input connections based on attention weights, solving the challenges mentioned at the beginning of the section.

2.2.2. Transformer model & natural language processing

Self-attention is just a component within a broader **transformer layer**. This layer, seen in Figure 2.9, encompasses a *multi-head self-attention unit* (multiple self-attention mechanisms are usually applied in parallel), facilitating interactions between word representations, followed by a *fully connected network*, which operates independently on each word. Both units are *residual networks*, where their output is combined with the original input. Additionally, a *LayerNorm* [9] operation is commonly incorporated after both the self-attention and fully connected networks. This operation, akin to BatchNorm [10], utilizes statistics across tokens within a single input sequence for normalization.

The standard **NLP pipeline** typically starts with *tokenization*, where text is divided into tokens. These tokens are then mapped to learned *embeddings* stored in a matrix, enabling the representation of words as continuous vectors. These embeddings are processed through multiple transformer layers in the transformer model. The vocabulary for tokenization is constructed using a sub-word tokenizer like *byte-pair encoding*, which merges frequently occurring substrings based on their frequency.

Each token in the vocabulary $\mathcal{V}$ is linked to a unique word embedding, stored in a matrix $\boldsymbol{\Omega}_e \in \mathbb{R}^{D\times|\mathcal{V}|}$. Initially, the input tokens are encoded into a matrix $\mathbf{T} \in \mathbb{R}^{|\mathcal{V}|\times N}$, where the n^{th} column corresponds to the n^{th} token and is encoded as a $|\mathcal{V}| \times 1$ one-hot vector. Input embeddings are then computed as $\boldsymbol{X} = \boldsymbol{\Omega}_e\mathbf{T}$, with $\boldsymbol{\Omega}_e$ learned as a network parameter. Typically, the embedding size D is 1024, with a vocabulary size $|\mathcal{V}| \approx 30,000$, resulting in a considerable number of parameters to learn in $\boldsymbol{\Omega}_e$ even before engaging with the main network.

The embedding matrix $\boldsymbol{X}$ is then processed through a sequence of K transformer layers, forming a **transformer model**. Transformers present themselves in various forms: as encoder-only, decoder-only form, or **encoder-decoder** form, with the latter being the original version of the transformer [7] and the one used in most sequence-to-sequence tasks, including MT. In this last setup, the **encoder** transforms an input sequence of symbol representations $(x_1, \ldots, x_n)$ that can represent our source sentence, into a sequence of continuous representations $\boldsymbol{z} = (z_1, \ldots, z_n)$. Given z, the **decoder** generates an output

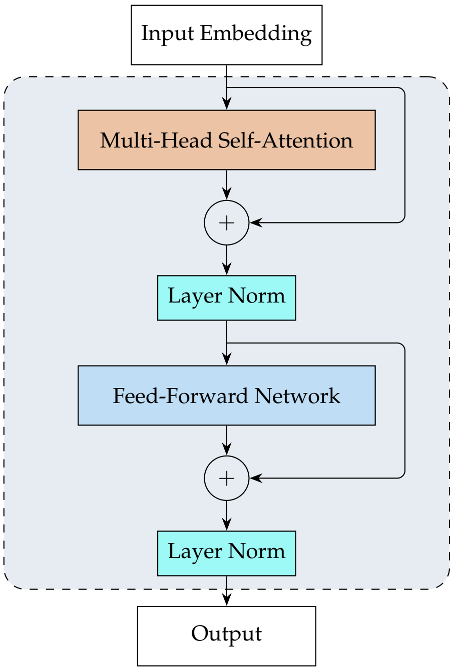

Figure 2.9: The **transformer layer** processes an input represented as a $D \times N$ matrix of D-dimensional word embeddings (continuous vector representations of words) for N tokens. It comprises several operations: multi-head attention to enable interaction among embeddings, followed by a residual block and LayerNorm. Then, a fully connected neural network is applied separately to each word representation in another residual layer, followed by LayerNorm.

sequence $(y_1, \ldots, y_m)$ of symbols one by one, representing our sentence in the target language. Throughout the process, the model operates in an *auto-regressive* manner, utilizing previously generated symbols as additional input during each step of generation, as seen in Figure 2.10.

Considering English-to-French translation, for instance, the encoder processes the English input using transformer layers to create token representations. During training, the decoder predicts subsequent French words using masked self-attention, ensuring predictions depend only on preceding words. The decoder also attends to encoder outputs, incorporating both prior outputs and the English input. This is facilitated by introducing cross-attention, where decoder embeddings query encoder embeddings to enhance translation quality.

Let's consider English-to-French translation. The encoder processes the English input through transformer layers, producing an output representation for each token. In the training phase, the decoder, given the correct French translation, employs transformer layers with **masked self-attention** (which ensures that predictions rely only on preceding words, preventing access to future ones) to predict subsequent words. The decoder also attends to encoder outputs, conditioning each French word on prior outputs and the English input. This is facilitated by introducing **cross-attention**, here queries originate from decoder embeddings, while keys and values are drawn from encoder embeddings.

2.3 Large Language Models

The NLP field has developed from statistical language modeling to neural language modeling, and then from pre-trained language models (PLMs) to LLMs. While traditional models are trained for specific tasks in supervised settings, PLMs train in a self-

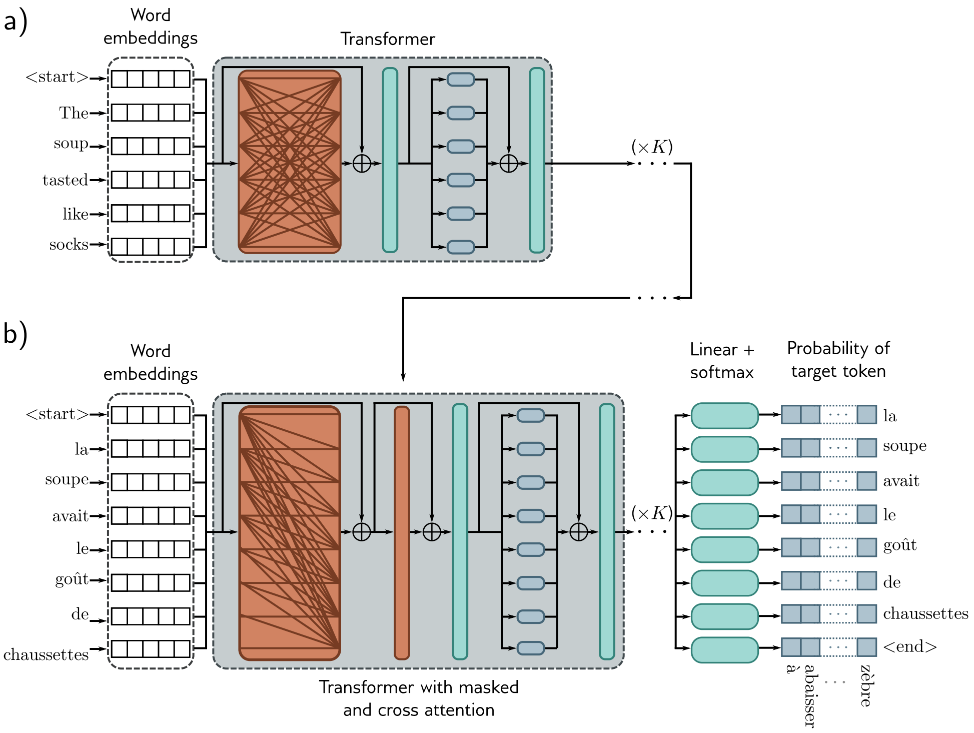

Figure 2.10: Transformer model featuring an encoder-decoder architecture. Two sentences are fed into the system for translation, aiming to convert the first sentence into the second. The initial sentence is processed by an encoder. Then, the second sentence is processed by a decoder with masked self-attention, while also attending to the output embedding through cross-attention (depicted by the orange rectangle). The loss function remains consistent with the decoder model, aiming to maximize the probability of the subsequent word in the output sequence. Obtained from [4].

supervised manner on large text corpora to acquire generic representations, applicable across different NLP tasks after a process of **fine-tuning**, outperforming traditional models. The recent developments in performance improvements have led to the transition from PLMs to LLMs, achieved by an increase in computational power, training dataset sizes, and model parameters [11].

LLMs typically denote transformer-based neural language models with tens to hundreds of billions of parameters (GPT-3 has 175B, BLOOM 176B, or MT-NLG 540B) pre-trained on extensive text data, and represent state-of-the-art systems capable of processing and generating coherent text, approximating human-level performance, and generalizing across multiple tasks. Initially relying on *transfer learning* (fine-tuning on a specific task), newer models like GPT-3 [12] show great results in zero-shot setups, where fine-tuning is not performed on the new downstream task, but improved further in a few-shot setup or with task-specific fine-tuning [11].

LLMs also showcase unexpected emergent capabilities that were not part of their training, such as reasoning, planning, and decision-making. Their huge scale contributes to these abilities, however these impressive capabilities come with a cost, including slow training and inference times, significant hardware requirements, and higher operational costs, limiting their widespread adoption and driving efforts to develop more efficient architectures and training strategies. Methods such as parameter-efficient tuning, pruning, quantization, knowledge distillation, and context length interpolation are among the techniques extensively researched to enhance the efficiency of LLM utilization [11]. We'll dive deeper into distributed training, parameter-efficient tuning methods, particularly on the LoRA method [3], and quantization [13].

2.3.1. Distributed training

A variety of approaches is used to distribute the load of the training process for LLM models, which include:

1. **Data Parallelism (DP)**: The model is replicated across different devices, where each device processes a slice of the data. The weights are then synchronized across all devices after each training iteration.

2. **Tensor Parallelism (TP)**: Also referred to as *horizontal parallelism*, each tensor is divided into multiple chunks, having each shard of the tensor on multiple devices instead of having the whole tensor on a single device. During processing, each shard is processed separately and simultaneously across the different devices, with the results synchronized at the end of the step.

3. **Pipeline Parallelism (PP)**: Also known as *vertical parallelism*, the model layers get distributed across different devices, having each device processing a different stage of the pipeline.

4. **3D Parallelism**: A combination of data, tensor, and pipeline parallelism (see Figure 2.11).

5. **Optimizer Parallelism (ZeRO)** [14]: Memory redundancies among data-parallel processes get eliminated by partitioning model states (gradients, parameters and optimizer states) across devices instead of duplicating them. It allows per-device memory usage to grow linearly with the level of data parallelism and results in a comparable communication volume to traditional data parallelism. ZeRO-powered data parallelism can handle models of any size, given that the combined device memory is adequate for sharing the model states.

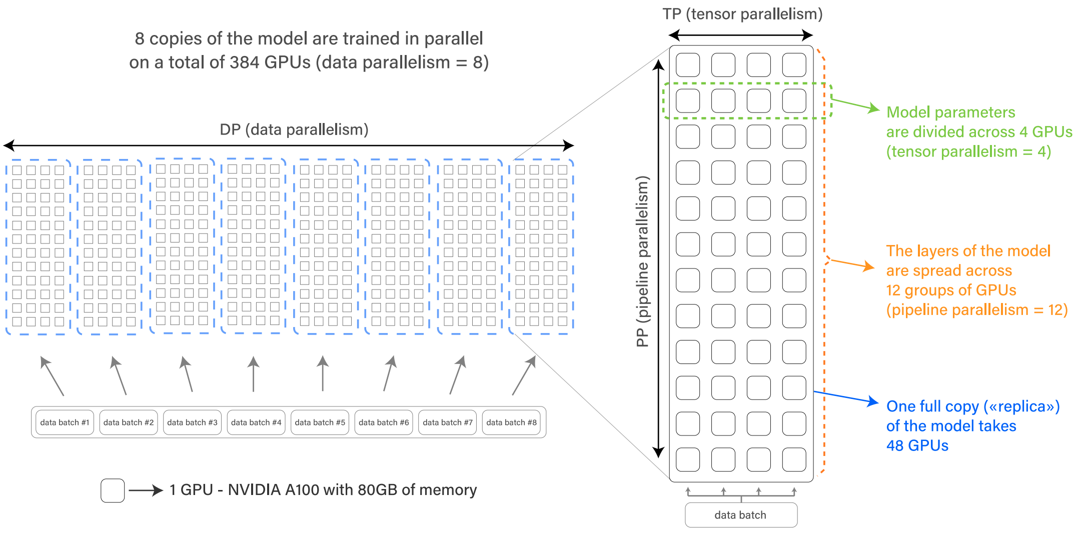

Figure 2.11: Distributed training setup for the BLOOM LLM, a DP+TP+PP combination or **3D Parallelism**. 8 copies of the model are trained in parallel, one copy taking 48 GPUs and one data batch. The model parameters are then divided across 4 GPUs, and the layers spread across 12 GPUs. Obtained from [15].

2.3.2. Parameter Efficient Fine-Tuning & LoRA

The scale of billions of parameters in most LLMs makes fine-tuning a computationally-intensive and time-consuming process. **Parameter-efficient fine-tuning (PEFT)** techniques aim to strike a balance between a good model fine-tuning performance and reduced costs by minimizing the number of parameters and computational resources needed to adapt a model to a new given task. The performance of these techniques compared to a full fine-tuning is better in low-resource settings, comparable in medium-resource scenarios, and slightly worse in high-resource environments in general, although there are cases where full fine-tuning may lead to *catastrophic forgetting* [16] as it modifies all model parameters, and since PEFT only updates a small subset of parameters, it has the potential to reduce this catastrophic forgetting effect.

Low-Rank Adaptation (LoRA) [3] improves efficiency by restricting the updates to the model's weights using a low-rank decomposition. It keeps the initial weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d\times k}$ frozen without gradient updates and trains only the smaller low-rank matrices $\mathbf{A} \in \mathbb{R}^{r\times k}$, $\mathbf{B} \in \mathbb{R}^{d\times r}$, with rank $r \ll \min(d, k)$. This approach constrains the update using the low-rank decomposition $\mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{BA}$.

Both $\mathbf{W}_0$ and $\Delta\mathbf{W} = \mathbf{BA}$ are applied to the same input, and their respective output vectors are summed coordinate-wise, as illustrated in Figure 2.12. For $\mathbf{h} = \mathbf{W}_0\boldsymbol{x}$, the modified forward pass becomes:

$$\mathbf{h} = \mathbf{W}_0\boldsymbol{x} + \Delta\mathbf{W}\boldsymbol{x} = \mathbf{W}_0\boldsymbol{x} + \mathbf{BA}\boldsymbol{x} \tag{2.15}$$

A random Gaussian initialization for matrix $\mathbf{A}$ is used, and matrix $\mathbf{B}$ is set to zero, so $\Delta\mathbf{W} = \mathbf{BA}$ is zero at the start of training. The change $\Delta\mathbf{W}$ is then scaled by $\frac{\alpha}{r}$, where α is a constant dependent on r. This scaling strategy helps stabilize training and avoids the need for frequent hyperparameter adjustments.

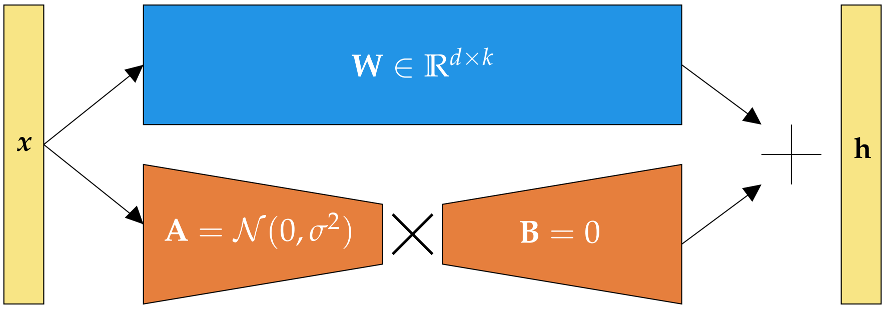
$\boldsymbol{x}$
$\mathbf{W} \in \mathbb{R}^{d\times k}$
$\mathbf{A} = \mathcal{N}(0, \sigma^2)$
$\mathbf{B} = 0$
$\mathbf{h}$

Figure 2.12: LoRA reparametrization, only **A** and **B** are trained.

LoRA can be applied to any subset of weight matrices within a neural network to decrease the count of trainable parameters. In the self-attention module of the transformer, LoRA targets the weight matrices ($\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$, $\mathbf{W}_o$) and freezes the MLP module (the fully connected network), aiming for simplicity and parameter efficiency. By freezing parts of the model and updating only the necessary matrices, LoRA achieves significant gains in efficiency, reducing hardware requirements and allowing for easy adaptation to different tasks without retraining the entire model, allowing for multiple LoRA modules to be trained, with a significant reduction in storage requirement and task-switching overhead.

However, it has its limitations. It is complicated to batch inputs to different tasks within a single forward pass if adjustments to attention matrices (**A** and **B**) are absorbed into weights (**W**) to reduce inference latency. Nonetheless, this problem can be managed by selectively applying LoRA modules based on batch samples, particularly in scenarios where latency isn't critical.

2.3.3. Quantization

LLMs demand significant computational resources and memory not only for training, but for inference tasks too. The GPT-3 model with its 175B parameters necessitates at least five A100 GPUs with a memory capacity of 80GB each, totaling 350GB of memory when stored in FP16 format [17]. This supposes a huge barrier for smaller organizations aiming to utilize LLMs.

To address this, techniques like **mixed precision** [13] are used, where model weights are stored in high precision (FP32) for accuracy but computations during training and inference are performed in lower precision (FP16 or BF16) to enhance speed and reduce memory usage. The gradients computed in lower precision are then utilized to update the FP32 main weights.

In some cases, FP32 precision is necessary, such as when gradients become too small in FP16, potentially rounding to zero and compromising optimizer performance. Similarly, a large weight value to weight update ratio can cause weight updates to be rounded down to zero in FP16 when added to the weight value, as it gets shifted to align with the binary point of the weight. This occurs when a weight value of 1000 is summed with a weight update of 0.001, resulting in a sum of 1000.0, for example. Both issues are addressed by using FP32 copies for gradient updates, ensuring accuracy is maintained.

Furthermore, gradient values often have small magnitudes, leaving much of the positive representable range of FP16 unused. Scaling up gradients shifts them to occupy more of this range, preserving values that might otherwise be lost to rounding. Before updating weights, these scaled gradients are unscaled to maintain update magnitudes as in FP32 training.

2.4 Neural Machine Translation

Machine translation (MT) can be defined as the application of rule-based or probabilistic ML techniques to automatically translate text from one language to another. The inherent complexity of natural languages poses a problem in trying to cover all language particularities with hard-coded manual translation rules. With the increasing amounts of data available nowadays, approaches that can learn linguistic information from data have increased in popularity.

Early efforts in the mid-20th century focused on **rule-based** approaches, where linguists and computer scientists devised sets of linguistic rules to translate text from one language to another. However, these systems struggled with the complexities of natural languages.

In the early 1990s, **statistical machine translation (SMT)** emerged, an approach that learns to translate by analyzing the statistical correlations between original texts and their corresponding human translations, leveraging *parallel corpora* (collections of texts in two or more languages that are translations of each other, aligned at the sentence or phrase level). Despite its advancements, SMT had limitations in handling long-distance word relationships [8].

The breakthrough of deep learning addressed this drawback, giving rise to a new paradigm. **Neural machine translation (NMT)** utilizes continuous representations instead of the discrete symbolic representations used in SMT, and a single large neural network with an end-to-end training that models the entire translation procedure, which avoids over the top feature engineering and separately tuning components as it happens with SMT. Despite this simplicity, NMT has achieved state-of-the-art results in a wide variety of MT tasks and has become the key technology behind a lot of commercial MT systems [8].

Formally, NMT aims to approximate the conditional distribution $P(\boldsymbol{y}|\boldsymbol{x})$ using a dataset $\mathcal{D}$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ represent source and target sentences respectively [8]. Translation is formulated as finding the most likely target sentence $\hat{\boldsymbol{y}}$ given some source sentence $\boldsymbol{x}$.

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, P(\boldsymbol{y} \mid \boldsymbol{x}). \tag{2.16}$$

Translation can be represented at various granularities, including document, paragraph, and sentence levels. Assuming sentence-level translation, and presuming both input and output sentences to be sequences, an NMT model can be viewed as a sequence-to-sequence model. Having a source sentence $\boldsymbol{x} = \{x_1, \ldots, x_S\}$ and a target sentence $\boldsymbol{y} = \{y_1, \ldots, y_T\}$, the conditional distribution can be expressed as:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{T} P(y_t|y_0, \ldots, y_{t-1}, x_1, \ldots, x_S). \tag{2.17}$$

where prediction at step t is taken as an input at step $t + 1$ [8].

NMT models typically employ the **encoder-decoder framework**, and it comprises four fundamental components, which are embodied in different architectures including the transformer, previously discussed and currently the state-of-the-art in MT: embedding layers, encoder and decoder networks, and the classification layer (an example can be seen in Figure 2.13).

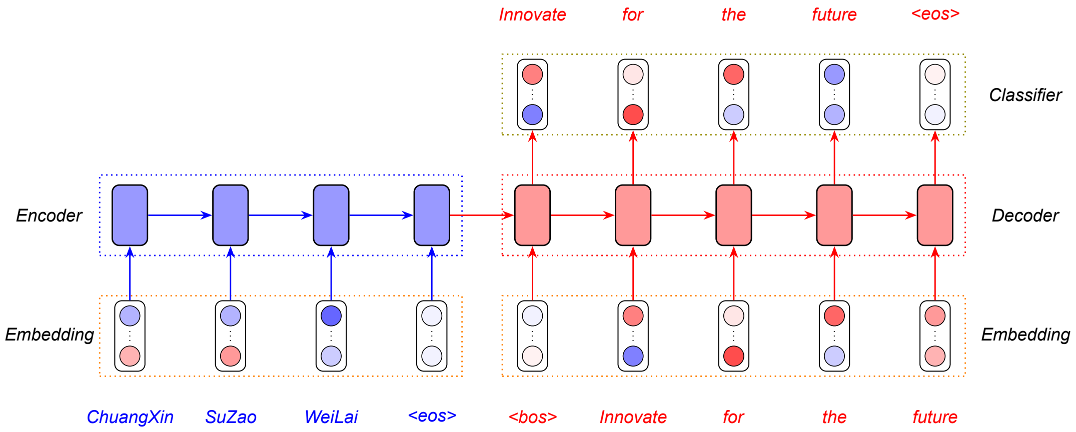

Figure 2.13: A summary of the NMT structure, including embedding layers, an encoder and a decoder network, and a classification layer, with distinct colors indicating different languages. "<bos>" and "<eos>" are special tokens used to denote the start and end of a sentence, respectively. Obtained from [8].

In terms of **evaluation**, human experts are the best option for obvious reasons, since it is them which can assess the quality of a given translation. This is quite expensive and impractical, and a variety of automatic evaluation metrics have been developed, which allow for a fast and easy evaluation of the performance of the models during the experimental phase.

BLEU (Bilingual Evaluation Understudy) [18] is the most commonly used automatic metric in MT. This metric relies on the concepts of an *n-gram*, a contiguous sequence of n

items (usually words), and that of *precision*, a metric that makes a count of the number of candidate translation words (unigrams) found in any reference translation and divides it by the total number of words in the candidate translation.

BLEU is calculated by using a **modified average n-gram precision** p_n, where the precision is adjusted to ensure that an n-gram appears in the candidate c translation no more frequently than it does in the reference r translation. Standard precision measures don't take this into account, and can give a very high precision value to a candidate phrase such as *"The the the the the"*. Thus, our average precision is:

$$\text{Average Precision}(N) = \frac{1}{N} \sum_{n=1}^{N} \log p_n \tag{2.18}$$

A **brevity penalty (BP)** component is introduced, which weights the precision. Let c be the length of the candidate translation, while r denotes the effective length of the reference corpus.

$$\text{Brevity Penalty} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases} \tag{2.19}$$

Then our BLEU metric, typically computed with a value of $N = 4$, is left as:

$$\text{BLEU} = \text{Brevity Penalty} * \text{Average Precision}(4) \tag{2.20}$$

The resulting BLEU score ranges between 0 and 1, usually multiplied by 100 for easier interpretation, with higher values indicating better translation quality.

There have been recent studies that demonstrate that overlap-based metrics like BLEU underperform against the newer **neural-based learned metrics** [19], which primarily utilize the embedding representations of specifically pre-trained neural encoders for MT evaluation. This is attributed to the challenges the traditional metrics face in accurately correlating with human evaluations at segment level and in distinguishing effectively between the top-performing MT systems.

COMET (Crosslingual Optimized Metric for Evaluation of Translation) [20] is one of the most popular neural metrics. It utilizes a pre-trained, multilingual model adapted and trained on high-quality translation pairs to serve as a regressor for translation quality scores. The architecture of the COMET model can be seen in Figure 2.14.

Initially, the source, hypothesis, and reference sentences undergo encoding into multiple word embeddings for each of the intermediate layers of the model, these embeddings are then passed through a trainable layer-wise attention mechanism, are concatenated, and averaged to produce sentence-level embedding vectors $\mathbf{s}$, $\mathbf{h}$, and $\mathbf{r}$. Then, the combined features $\mathbf{h} \odot \mathbf{s}$, $\mathbf{h} \odot \mathbf{r}$, $|\mathbf{h} - \mathbf{s}|$, and $|\mathbf{h} - \mathbf{r}|$ are obtained, which are then concatenated with $\mathbf{r}$ and $\mathbf{h}$ into a single vector $\mathbf{x} = [\mathbf{h}; \mathbf{r}; \mathbf{h} \odot \mathbf{s}; \mathbf{h} \odot \mathbf{r}; |\mathbf{h} - \mathbf{s}|; |\mathbf{h} - \mathbf{r}|]$ that is fed into a trainable forward regressor. This regressor produces scores ranging from 0 to 1, where 0 signifies random input and 1 denotes a flawless translation. Typically, these scores are then rescaled to a range between 0 and 100 for reporting.

2.5 Streaming Machine Translation

Streaming MT is closely related to the concept of simultaneous MT. Both address the challenges of translating continuous input streams, such as those generated by *Automatic Speech Recognition* (ASR) systems or live audiovisual content.

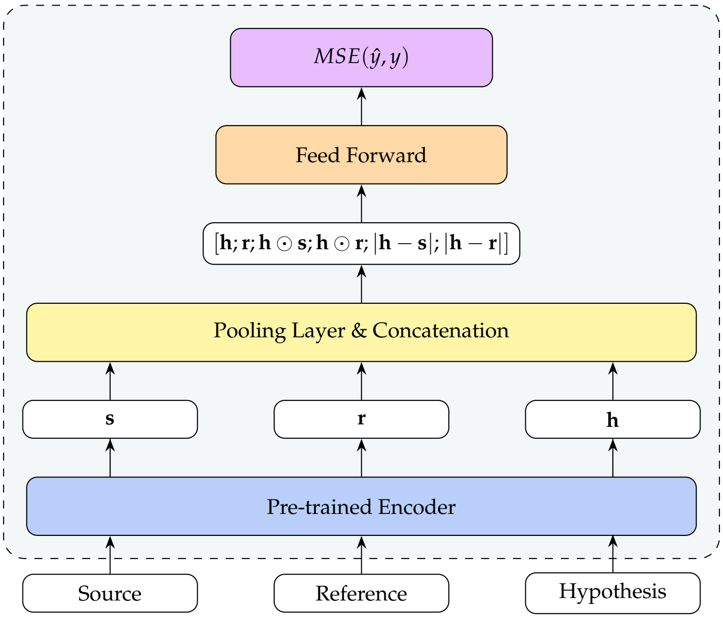
$MSE(\hat{y}, y)$
Feed Forward
$[\mathbf{h}; \mathbf{r}; \mathbf{h} \odot \mathbf{s}; \mathbf{h} \odot \mathbf{r}; |\mathbf{h} - \mathbf{s}|; |\mathbf{h} - \mathbf{r}|]$
Pooling Layer & Concatenation
s
r
h
Pre-trained Encoder
Source
Reference
Hypothesis

Figure 2.14: COMET architecture. Source, hypothesis, and reference sentences undergo individual encoding via a pre-trained cross-lingual encoder to produce word embeddings. Pooling is applied to the word embeddings to generate a sentence embedding for each segment. These sentence embeddings are then combined and concatenated into a single vector, which serves as input to a feed-forward regressor. The model's training process involves minimizing the Mean Squared Error (MSE).

Simultaneous MT consists of translating an input sentence incrementally, before it is fully available. This concept is particularly relevant in scenarios such as translating a continuous text stream generated by ASR, and its uses range from facilitating person-to-person communication to providing subtitles for audiovisual content.

Streaming MT, on the other hand, expands upon Simultaneous MT by focusing on the gradual translation of a continuous input text stream, that is, the simultaneous translation of a potentially unbounded and unsegmented text stream. Unlike traditional batch processing where sentences are processed independently, streaming MT maintains context over time, leveraging a *streaming history* to improve translation quality and coherence throughout the entire stream. This approach introduces challenges such as handling latency constraints and maintaining alignment between source and target texts, especially in the absence of clear sentence boundaries.

Therefore, the concept of *sentence segmentation* becomes crucial in the context of streaming MT. It helps establish a monotonic alignment between the source and target sentences, ensuring that translations remain coherent and accurate despite the continuous nature of the input stream [21]. A comprehensive exploration of streaming MT is elaborated in Section 4.1, which offers both a formal definition and a detailed examination, as well as the details of its practical implementation in this work.

CHAPTER 3

Adaptation for offline MT

In this chapter, the capabilities of LLMs are explored, along with how they can be enhanced through adaptation for MT tasks. Previous studies have determined the No Language Left Behind (NLLB) model as one of the publicly available LLMs with best results on various test sets [22], and is characterized as being a multilingual MT model. Hence, our focus will be on this model, given its potential to deliver superior results. This chapter is dedicated to discussing adaptation within the offline context, which constitutes the standard setup for MT, where the model is provided with complete phrases or sentences for translation.

In our first step towards our primary goal of obtaining a fine-tuned LLM adapted for streaming MT, NLLB will be fine-tuned for a specific language pair and domain in an offline setting in order to gain experience and reproduce the experiments reported in [5]. This will help us grasp the basics of the process and understand the group's experimental setting, utilizing their resources and methodologies. To this purpose, LoRA was applied to different versions of the NLLB model and then evaluated on the INTERACT-EUROPE dataset, which is domain-specific to medicine and oncology. Our goal is to study how the model's performance evolves for a given language pair and a particular domain when LoRAs are trained using data specific to each language pair and domain.

3.1 Multilingual Encoder-Decoder Models

The most general approach for an MT model typically involves an individual model trained for a particular language pair, directly mapping a symbol or a sequence of symbols from a source language with its corresponding symbol or sequence in a target language. However, this approach presents practical challenges. This mapping method is exclusively tailored to a particular language pair, and extending it to accommodate multiple language pairs is not a trivial task. Then, as the number of language directions increases, the cost at training and inference also increases, as more models are required to cover all directions.

However, neural MT systems can be decomposed into two modules, the already mentioned encoder and decoder. This opens the door to the construction of systems capable of mapping a source sentence from any language into a unified continuous representation space. From this representation, the system can decode the sentence into any target language, opening the way to **multilingual MT systems**. This type of models have shown better results when evaluated across various languages compared to single-pair models, particularly excelling in translating low-resource language pairs [23].

The design of multilingual MT models includes important aspects such as determining the number of languages that the model is able to handle. There exists a trade-off between the number of supported languages and translation accuracy when employing a fixed model capacity and training setup, where initially adding more languages enhances performance but seems to be counterproductive at a given point [24]. The accuracy of *zero-shot translation*, which involves translating between languages lacking parallel data, is also affected by the number of languages supported. The zero-shot accuracy can be seen as a measure of model generalization, as by accommodating more languages the model is prompted to create a more generalized representation to better use its capacity. While the trend suggests that incorporating more languages improves this accuracy, at a given point the benefit stops and the accuracy even deteriorates, hinting at the necessity of even larger models for such settings [24].

The promising potential of multilingual MT models has encouraged a multitude of studies aimed at improving their performance. Consequently, the NLLB model, an LLM that is also a multilingual MT model, will be introduced, and examined in our pursuit of achieving streaming MT.

3.1.1. No Language Left Behind (NLLB)

No Language Left Behind (NLLB) [25] is a series of multilingual MT models that can produce translations between 200 languages, including low-resource languages. The NLLB model architecture is based on the Transformer encoder-decoder architecture, and is offered in three versions: with 600M, 1.3B, and 3.3B parameters.

Various strategies were employed in its development. **Curriculum learning**, gradually introducing harder aspects of a problem during training, was used to mitigate overfitting on low-resource language pairs by introducing these in phases during training, with each pair being introduced K updates before training conclusion, being K the number of updates a given language pair needs to empirically overfit. **Self-supervision** was employed to leverage the amounts of monolingual data available for low-resource languages, which often lack bitext data. Self-supervised learning enables the model to learn linguistic patterns from monolingual text, and different self-supervised tasks were tried including a language modeling objective, a denoising autoencoder objective, and the combination of both. **Data augmentation** was also used to leverage monolingual data through back-translation. Back-translation involves generating parallel corpora that are noisy on the source side from monolingual text using MT.

3.2 Datasets

3.2.1. Evaluation datasets

To examine how the NLLB model performance changes, the **evaluation** is done on the INTERACT-EUROPE project evaluation dataset, a recent project undertaken by the MLLP group. This dataset comprises English transcriptions and their corresponding translations in various languages, derived from a series of video conferences held at the European School of Oncology (ESO)[1]. The transcriptions, initially provided by ESO, were sourced from 10 videos, each split into two subgroups summing 3.5 hours and 3.8 hours of speech, respectively. These videos were then professionally translated into Spanish, French, German, and Slovene by a translation agency.

[1] https://www.eso.net/

Through this process, a collection of non-aligned translations were obtained. A curated list of regular expressions was used to preprocess them, and the Moses toolkit script `split-sentences.perl` was used to extract phrases from the evaluation sets, obtaining sentence-level parallel text for each language direction. The Vecalign alignment tool [26] additionally improved the alignments of these extracted phrases. Ultimately, another manual review was performed to identify alignment errors and any other undetected mistakes that may have slipped through the cleanup. Table 3.1 displays the total count of sentences for each language pair in the evaluation sets.

Table 3.1: INTERACT-EUROPE datasets. Total number of sentence-level bitext for the evaluation sets.

Language pair	Dev	Test
$en \rightarrow fr$	1445	1407
$en \rightarrow es$	1450	1405
$en \rightarrow de$	1424	1399
$en \rightarrow sl$	1458	1407

3.2.2. Training datasets

The **training sets** for the fine-tuning included, for the $\boldsymbol{en \rightarrow es, de, fr}$ language directions, data from Medline-WMT22, which includes Medline abstracts provided by the WMT22 Biomedical Translation Task [27] [2], the Europarl-ST dataset [28] which contains paired audio-text samples made from publicly available videos of debates held in the European Parliament, and the MuST-C dataset [29] which is based on TED Talks. This data selection tries to represent both a clean, high quality corpora of data sourced from speech to represent spoken language, and close in-domain medical data.

Given the limited availability of data in comparison to the other language pairs, the $\boldsymbol{en \rightarrow sl}$ language direction used an aleatory subset of medical data from the EMEA corpus, comprised of PDF documents from the European Medicines Agency, and back-translations from a baseline model trained on the reverse direction of data provided by the Institute of Oncology of Ljubljana [3], approximately matching the number of sentences found in the other fine-tuning datasets.

Tables 3.2, 3.3, 3.4 and 3.5 report an overview of the selected corpora for English into French, Spanish, German and Slovene, respectively. In summary, the total volume of data amounts to 497.3K sentences for $en \rightarrow es$, 468.0K sentences for $en \rightarrow fr$, 430.5K sentences for $en \rightarrow de$, and around 147.8K sentences for $en \rightarrow sl$ when factoring in the additional bilingual and monolingual data.

Table 3.2: Training corpora for language pair $en \rightarrow fr$.

Dataset	Sentences	Words	
		English	French
Europarl-ST [28]	96.5 K	2.3 M	2.6 M
MuST-C [29]	275.0 K	5.1 M	5.3 M
Medline-WMT22	96.5 K	2.4 M	3.0 M
Total	468.0 K	9.8 M	10.9 M

[2] https://github.com/biomedical-translation-corpora/corpora

[3] https://www.onko-i.si/eng/sectors/research_and_education/medical_and_other_scientific_publication

[3] https://commoncrawl.org

Table 3.3: Training corpora for language pair *en* → *es*.

Dataset	**Sentences**	**Words**	
		English	**Spanish**
Europarl-ST [28]	77.9 K	1.8 M	1.9 M
MuST-C [29]	265.6 K	5.2 M	5.0 M
Medline-WMT22	153.8 K	3.6 M	4.2 M
Total	497.3 K	10.6 M	11.1 M

Table 3.4: Training corpora for language pair *en* → *de*.

Dataset	**Sentences**	**Words**	
		English	**German**
Europarl-ST [28]	105.3 K	2.3 M	2.2 M
MuST-C [29]	229.7 K	3.9 M	4.2 M
Medline-WMT22	95.5 K	2.0 M	2.2 M
Total	430.5 K	8.2 M	8.6 M

Table 3.5: Training corpora for language pair *en* → *sl*.

Dataset	**Sentences**	**Words**	
		English	**Slovene**
Back-translations	36.1 K	693.7 K	683.9 K
EMEA [30]	111.7 K	2.1 M	2.1 M
Total	147.8 K	2.8 M	2.8 M

3.2.3. Data processing pipeline

Pre-tokenization is applied to the data with the Moses `tokenizer.perl` tokenizer. Any sentences exceeding 250 tokens are eliminated through the Moses `clean-corpus` script. The vocabulary size is then reduced using Truecasing [31] that acts similarly to a normalization technique. Its goal is to transform text into its most appropriate upper or lower case form by analyzing diverse statistics to then construct a prediction model based on them. In Moses, the Truecasing model primarily modifies words at the beginning of sentences to their most common form, along with any words whose current form is unfamiliar.

Next, the data undergoes tokenization using a subword vocabulary derived from the gathered data. This provides a cost-effective method for simulating large vocabularies and addressing out-of-vocabulary (OOV) issues. Subword tokenization can be very handy in cases where, for example, a word-level Spanish vocabulary is available, denoted as $\mathcal{V}$, containing words like $\mathcal{V} = [\texttt{revestido}, \texttt{acabaría}]$. A model trained solely with $\mathcal{V}$ would only recognize these specific words. In contrast, a model trained with a subword vocabulary, such as $\mathcal{V} = [\texttt{re}, \texttt{ves}, \texttt{ti}, \texttt{do}, \texttt{a}, \texttt{ca}, \texttt{ba}, \texttt{ría}]$, could also identify words like `vestido`, `acabado`, or `batido`.

The **SentencePiece** library [32] is used for subword tokenization. Its C++ implementation ensures its subword tokenizer's speed and robustness, and features different partitioning algorithms like Byte-Pair Encoding [33], [34] and unigram-based language models [35]. Additionally, SentencePiece offers automatic normalization of NFKC UTF-8 text, potential regularization of subwords, and language-agnostic representations facilitated by special treatment of whitespaces.[4]

BPE was selected as the partitioning algorithm for this work. BPE breaks down text at character level and identifies text combinations (referred to as *merge operations*) to generate a vocabulary based on character frequency and sequences in the training dataset. This process continues until reaching the vocabulary limit or the maximum specified number of merge operations. Figure 3.1 illustrates an example of a training sentence after Truecasing and SentencePiece have been applied.

Original	`This is a strange day ...`
Truecase and SPM	`th is_ is_ a_ str ange_ day_ ...`

Figure 3.1: Comparison of a training sentence following Truecasing and SentencePiece. Observe the change from `This` to `this` due to Truecasing, and the space representation with _ as well as the segmentation into subwords `strange` by SentencePiece.

3.3 Offline experimentation on NLLB

The performance of NLLB model variants with 600M, 1.3B, and 3.3B parameters was studied across all INTERACT dev sets to establish a baseline of the model's behavior. Then, LoRAs were trained for each of these sizes and language directions to explore the efficacy of this method as a lightweight domain adaptation tool. Our objective is to study and measure the enhancement of NMT models when tailored towards a specific language direction using LoRAs, which could reduce potential biases in the models representation space due to multilingualism. Positive outcomes in this offline setting would also motivate the idea that the streaming adaptation could be achieved through a LoRA adaptation.

In order to conduct these experiments, the **Hugging Face**[5] framework was chosen. It consists of a series of tools to build ML applications, and stands out for its Transformers library[6], which facilitates building natural language processing applications by easily downloading and training pretrained models (including the NLLB model). These models can be found built on different frameworks, and those implementing the PyTorch framework [36] are utilized in this work. It also supports the use of techniques such as multi-GPU, mixed precision training and parameter efficient fine-tuning.

The LoRAs training was done using the fine-tuning sets introduced in Section 3.2.1 for language directions $en \rightarrow es, de, fr, sl$. The LoRA hyperparameters chosen are outlined in Table 3.6 and remain consistent across all language pairs. The LoRAs training was conducted using the Valencian Research Institute for Artificial Intelligence (VRAIN)[7] infrastructure, making use of a computation node composed of 8 NVIDIA A40 GPUs, each with 48GB of memory.

As previously stated, the three versions of the NLLB are composed of 600M, 1.3B and 3.3B parameters. The model size in bytes can be calculated by multiplying the number of parameters by the size of the chosen precision in bytes. In this case, the training was conducted in `float16`. The respective sizes of the models are detailed in Table 3.7. Notably, there is some leeway to increase batch sizes since our GPUs can handle up to 48GB of memory. For the 600M model, a batch size of 16 is feasible, while for the 1.3B and

[4] In SentencePiece, whitespace is escaped by default with the Unicode character ▁ (U+2581).

[5] https://huggingface.co/

[6] https://huggingface.co/docs/transformers/index

[7] https://vrain.upv.es/

3.3B models, a maximum batch size of 8 is possible before encountering memory overflow. This combined with gradient accumulation (a method emulating a larger batch size, which involves accumulating gradients from several small batches prior to executing a weight update) and multi-GPU training (DDP), provided an effective batch size of 32, making the training of one epoch of the 600M parameters model in around 12 minutes, while for the larger 3.3 billion parameter model, it takes approximately 102 minutes.

Table 3.6: LORA hyperparameters for the fine-tuned models.

Hyperparameter	**Value**
Optimizer	AdamW [37]
Warm up Steps	100
LR Schedule	Linear
Initial Learning Rate	2e-5
Epochs	3
Lora Dropout	0.05
LoRA Target Modules	$\boldsymbol{Q}, \boldsymbol{V}, \boldsymbol{O}$
LoRA rank config.	$r_Q = r_V = r_O = 16$
LoRA α	32
Trainable parameters[8]	0.3-0.4%

Table 3.7: Sizes in Gigabytes of the different versions of the NLLB model.

Model	**Size**
NLLB-600M	$600 \times 10^6 \times 2$ Bytes $= 1.2$ GB
NLLB-1.3B	$1.3 \times 10^9 \times 2$ Bytes $= 2.6$ GB
NLLB-3.3B	$3.3 \times 10^9 \times 2$ Bytes $= 6.6$ GB

3.3.1. Evaluation on adapted NLLB

The inference of the predictions is done using *beam search* [38] as a decoding algorithm, with size 5 and early stopping. Then, the inferences were evaluated using BLEU and COMET-22, with Table 3.8 showing the results for each language direction. These are categorized by model version, the utilization of LoRA, and the language pair.

Table 3.8: Results for NLLB models in the INTERACT dev sets

en → fr, es, de, sl

		French		Spanish		German		Slovene	
Model	LORA	BLEU	COMET	BLEU	COMET	BLEU	COMET	BLEU	COMET
NLLB-600M	✗	51.90	83.40	56.13	85.92	37.52	81.81	32.88	84.21
NLLB-1.3B	✗	54.59	84.17	57.32	86.32	40.29	83.27	37.15	86.09
NLLB-3.3B	✗	54.97	84.28	57.31	86.34	41.70	83.69	38.63	86.36
NLLB-600M	✓	52.24	83.89	56.33	86.49	39.09	82.98	33.68	84.65
NLLB-1.3B	✓	55.07	85.01	58.76	87.25	41.71	84.29	37.80	86.82
NLLB-3.3B	✓	**56.18**	**85.16**	**59.76**	**87.51**	**42.97**	**84.80**	**39.17**	**87.52**

It can be observed that the model's scale has a significant impact on performance, where the discrepancy between the largest and smallest NLLB models results in BLEU

[8] The dimension and number of matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{O}$ can differ across models, which varies the amount of trainable parameters.

differences ranging from 1.18 in *en* → *es* up to 5.75 in *en* → *sl*, and COMET differences ranging from 0.42 in *en* → *es* up to 2.87 in *en* → *sl*.

Then, analyzing the LoRA's effect, a positive effect is observed across all models in every language pair combination. Remarkably, it can be seen that in most instances the effect of training a LoRA for a given model size has a bigger improvement in both metrics compared to scaling model size. The most substantial enhancements are noted in the *en* → *es* language pair, with a performance boost of 2.45 in BLEU and 1.17 in COMET.

Then, having determined that LoRA is an improvement in every case and with our training configuration set, the fine-tuning of the NLLB model for streaming MT will be proceeded with in the next section.

CHAPTER 4

Adaptation for streaming MT

In this chapter, the NLLB model adaptation into a streaming MT setting is explored, following on the successful results from the previous chapter. Key theoretical concepts of streaming MT are introduced and explained, including the SegFree architecture that will be employed for our experiments, as well as some important notions of MT, such as document-level MT and prefix training, which are crucial to realize our fine-tuning.

Following this, a series of experiments to study different adaptations of the NLLB model under a variety of conditions will be conducted, evaluating how these adaptations affect the model's capabilities, particularly in terms of translation quality and latency. The Europarl-ST dataset [28] will be used for these evaluations. This dataset is well-suited for assessing streaming MT and reference results for comparison are available.

The results of these experiments will be compared against the NLLB's performance in an offline setting, serving as a benchmark of the model's maximum performance, as well as with a baseline reference model consisting of a Transformer BIG model utilizing the SegFree architecture, which was trained for streaming MT as described in [1]. However, this baseline model is not directly comparable to our NLLB adaptation due to significant differences in training data and setup. Therefore, it serves only as a reference to illustrate the current state-of-the-art performance, rather than for direct comparison.

4.1 Streaming MT

As described briefly in Section 2.5, **Streaming MT** focuses on a gradual translation of a continuous input text stream. In a traditional cascade setup, combining an ASR and an MT system, an intermediate *segmentation* is performed, where the transcription stream is split into sentence-like units to allow the MT system to translate them.

Thus, a source stream $\boldsymbol{X}$ is desired to be translated into $\boldsymbol{Y}$, and in this configuration, the decoding process typically entails making a greedy decision regarding which token will appear next at the i-th position of the ongoing translation being generated:

$$\hat{Y}_i = \operatorname*{argmax}_{y \in \mathcal{Y}} p(y \mid X_1^{G(i)}, Y_1^{i-1}) \tag{4.1}$$

being $G(i)$ a global **delay function**, indicating the last position in the source stream that was accessible when the i-th target token was produced, while $\mathcal{Y}$ denotes the target vocabulary.

As considering the entire source and target streams can be computationally demanding, the next token can be made dependent on the last $H(i)$ tokens of the stream:

$$\hat{Y}_i = \operatorname*{argmax}_{y \in \mathcal{Y}} p(y \mid X_{G(i)-H(i)+1}^{G(i)}, Y_{i-H(i)}^{i-1}) \tag{4.2}$$

4.1.1. Policy

Delay functions are determined based on the policy implemented. This policy dictates the action taken at each time step, whether to read a token from the input or generate a target token. Policies can be fixed, relying solely on the current timestep, or adaptive, considering available input source words as well. Notable fixed policies include the sentence-level **wait-k policy** [39] widely utilized in simultaneous MT. Since source and target segmentations are available in simultaneous MT, instead of the global delay function $G(i)$, a sentence-level local delay function is used:

$$g(i) = k + i - 1 \tag{4.3}$$

This policy begins by reading k source tokens before generating any target tokens. Consequently, higher values of k tend to yield better outcomes since the offline scenario, where the entire source text is visible beforehand, is progressively approached. After the initial read of k tokens, it alternates between generating a target token and reading another source token, see Figure 4.1. This holds when the ratio between the lengths of the source and target sentences is one. However, in the broader scenario, a catch-up factor γ, computed as the inverse of the source-target length ratio, determines the number of target tokens generated for each read token:

$$g(i) = \left\lfloor k + \frac{i-1}{\gamma} \right\rfloor \tag{4.4}$$

The wait-k policy can be reformulated in the context of streaming MT such that the wait-k behavior is implemented for each sentence individually:

$$G(i) = \left\lfloor k + \frac{i-b_n}{\gamma} \right\rfloor + a_n - 1 \tag{4.5}$$

where variables $\boldsymbol{a}$ and $\boldsymbol{b}$ represent the segmentation of the source and target streams, respectively, and can be interpreted as two vectors of the same length, indicating that the n-th source sentence begins at position a_n, while the n-th target sentence begins at position b_n [21].

4.1.2. Document-level MT & Prefix Training

Standard MT models trained with full sentences usually underperform in streaming situations. There are mainly two factors for this occurrence: First, these models fail to recognize that they are dealing with a continuous, interconnected discourse rather than isolated sentences; and second, they are not well prepared to deal with incomplete phrases, which is typical in streaming contexts. To address these issues, document-level MT and prefix training are essential concepts, as they help the model exploit the fact that it is working with a continuous discourse and handle incomplete inputs effectively, respectively, making them crucial for successful streaming MT models.

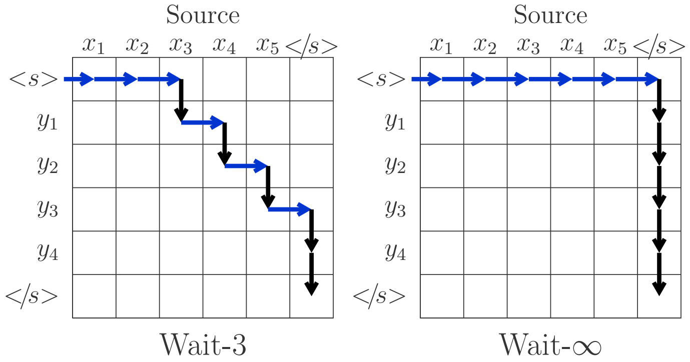

Figure 4.1: Wait-k. Wait-k decoding as a sequence of reads (horizontal) and writes (vertical) over a source-target grid. After first reading k tokens, the decoder alternates between reads and writes. In Wait-∞, or Wait-until-End, the entire source is read first, which just happens to be the offline, traditional MT case. Obtained from [40].

Document-level MT is different from sentence-level MT in that it entails the translation of complete documents while also capturing specific discourse phenomena through modeling. The methods used in document-level MT can be tailored for streaming scenarios by employing the concept of **streaming history** [1], which consists of a sliding window that maintains a limited context of previously encountered source segments along with their corresponding translations generated by the MT model, discarding over time the oldest translated words that are no longer necessary.

Thus, the sliding window has two segments, a portion of *history words*, which has been fully processed and can thus be discarded in the future, and the *active segment* that requires translation. Once the maximum capacity of the streaming history is reached, the oldest source segment and its corresponding translation are discarded.

Prefix training [41] is a data augmentation technique designed to enable a model to handle incomplete phrases. It achieves this by generating prefix pairs from the original available data, thereby preparing the model to work effectively with partial inputs.

Before prefix training, an additional step of document-level data adaptation can be performed, wherein sentences containing document-level information are augmented with their respective streaming histories [21], by concatenating the preceding source and target sentences until a maximum size of 50 tokens is attained.

Then, a partial translation pair is obtained from each sentence pair in the original corpus by randomly selecting a partial prefix from both the source and target sentences. If a sentence pair includes streaming history information, the streaming history remains unchanged, and prefix generation is solely applied to the current sentence pair. The model can then be trained using the concatenation of the original training data and the partial prefix data, doubling the size of the training set [1]. This training data construction process can be seen in Figure 4.2.

4.1.3. Segmentation-Free Architecture

As already mentioned, a canonical use of streaming MT is in the context of speech translation where we translate a continuous text stream generated by an ASR system. The actual MT systems rely on an intermediate **segmentation** step that divides the transcription stream into units resembling sentences. This necessity comes from the fact that standard MT systems training data consist of pairs of sentences aligned with each other (bitexts).

The segmentation model is employed to divide the incoming text stream into units resembling sentences, enabling translation by the MT system. Each of these sentence-like

Original	Prefix-augmented
I'm going to talk today about energy and climate.	I'm going to talk today
Hoy voy a hablar sobre energía y clima.	Hoy voy a hablar
Think about it. [SEP] The PC is a miracle.	Think about it. [SEP] The PC is
Piensa en ello. [SEP] El PC es una maravilla.	Piensa en ello. [SEP] El PC es

Figure 4.2: Streaming MT training data preparation. For each sentence pair, a prefix training variant is generated by removing a section of both the source and target sentences. In the first row, a source-target sample lacking streaming history is randomly prefixed. In contrast, the second row illustrates a source-target sample incorporating streaming history (light gray), where prefix augmentation is only applied to the current sentence being translated, while the history remains unmodified. Adapted from [1].

units, or segments, is usually translated independently, and techniques from document-level MT can be applied to furnish a standard MT model with additional context beyond the current sentence. However, this approach can introduce limitations and errors into the system due to its rigid nature. A notable challenge arises from the difference in length between the training data, typically consisting of a few hundred tokens in aligned sentence pairs, and the conditions during inference, which frequently involve thousands of tokens in live sessions.

Thus, the primary drawback of the segmented setting lies in its heavy reliance on the quality of the segmenter, which significantly influences translation quality. By enforcing a rigid decision-making process independently of the MT system, the resulting translation quality becomes conditioned and potentially compromised.

The **Segmentation-Free (SegFree)** approach addresses this issue and has been shown to improve results [1]. In this method, the MT model receives an unsegmented stream of source text and produces a continuous sequence of translated words. It does not only generate the translation, but also determines its segmentation by incorporating a special token ("[SEP]") into the translation stream. Thus, where in the segmented setting the segmentation is made independently from the MT model on the source side, in the SegFree setting is made by the MT model, which considers both the source and target streams, giving the SegFree model the ability to leverage additional target-side information, a pivotal factor contributing to its improved performance over the segmented approach [1].

Formally, the source stream $\boldsymbol{X} = \{x_1, x_2, \ldots, x_J\}$ and target stream $\boldsymbol{Y} = \{y_1, y_2, \ldots, y_I\}$ are represented by sequences of words. A segment of active source words $\boldsymbol{x}_j^{j'}$ and its corresponding partial translation $\hat{\boldsymbol{y}}_i^{i'}$ are also considered. Whenever the translation model generates an end-of-segment token [SEP], the **memory mechanism** comes into play. A source position $\hat{a} \in [j, j']$ is selected within the active segment based on the ongoing translation. Then, the words $\boldsymbol{x}_j^{\hat{a}}$ at the source segment and their corresponding translations $\hat{\boldsymbol{y}}_i^{i'}$ are moved to the streaming history, while the translation continues with $\boldsymbol{x}_{\hat{a}+1}^{j'}$. This is shown in Figure 4.3.

In this context, the memory mechanism takes the current active source segment $\boldsymbol{x} = \boldsymbol{x}_j^{j'}$ and target segment $\hat{\boldsymbol{y}} = \hat{\boldsymbol{y}}_i^{i'}$, and employs a log-linear model composed of a set of feature functions $h_f(a, \boldsymbol{x}, \hat{\boldsymbol{y}})$ that assign a score to each position a within the active source

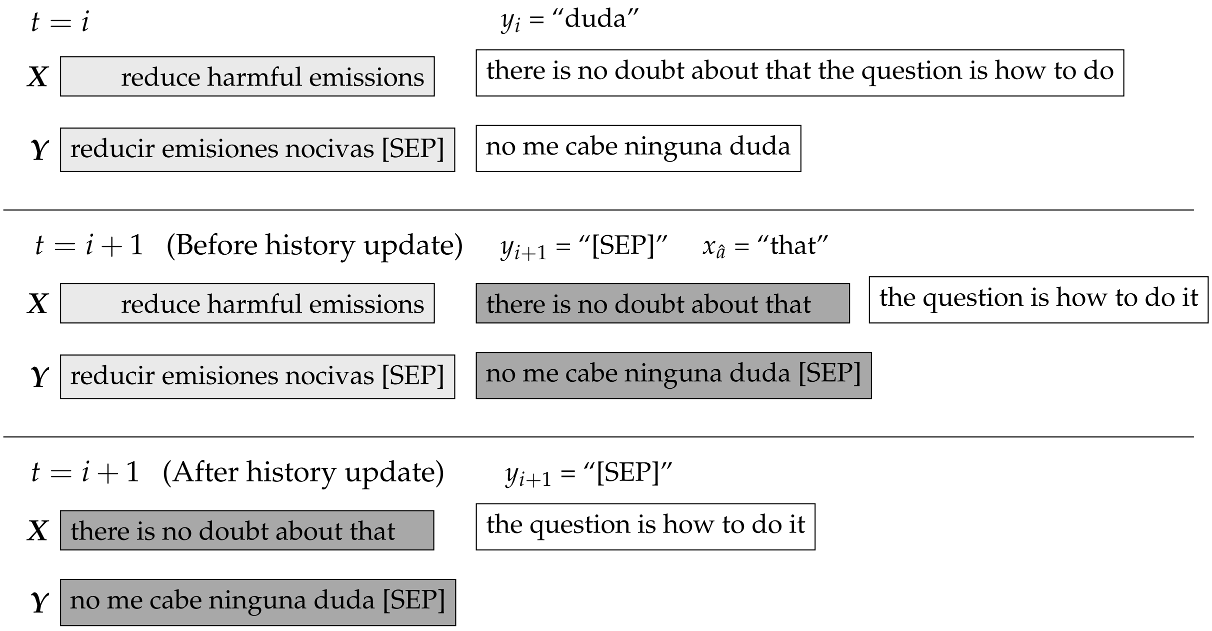
$t = i$
y_i = "duda"
X
reduce harmful emissions
there is no doubt about that the question is how to do
Y
reducir emisiones nocivas [SEP]
no me cabe ninguna duda
$t = i + 1$ (Before history update)
y_{i+1} = "[SEP]"
$x_{\hat{a}}$ = "that"
X
reduce harmful emissions
there is no doubt about that
the question is how to do it
Y
reducir emisiones nocivas [SEP]
no me cabe ninguna duda [SEP]
$t = i + 1$ (After history update)
y_{i+1} = "[SEP]"
X
there is no doubt about that
the question is how to do it
Y
no me cabe ninguna duda [SEP]

Figure 4.3: Memory mechanism. Shaded chunks are part of the streaming history, while unshaded chunks depict the current active source (top) and target (bottom) streams. The first row ($t = i$) shows that the MT system has just generated the final target word y_i = "duda". In the second row ($t = i + 1$), the translation model produces the [SEP] token, signaling the end of a target segment. The memory mechanism triggers and selects $\hat{a} = j + 5$ with $x_{\hat{a}}$ = "that", resulting in the phrase "there is no doubt about that" being transferred to the streaming history together with the ongoing translation. In the third row, the streaming history has grown excessively large but the translation process continues. Consequently, the memory mechanism discards the oldest segment. Obtained from [1].

segment. The position of the last source word $\hat{a}$ to be moved to the streaming history is determined as:

$$\hat{a} = \underset{a}{\operatorname{argmax}} \sum_{f} \lambda_f \log h_f(a, \boldsymbol{x}, \hat{\boldsymbol{y}}) \tag{4.6}$$

The weights of these feature functions, denoted by $\boldsymbol{\lambda} \in \mathbb{R}^F$, are optimized using gradient descent by minimizing the standard cross-entropy loss across a dataset.

In this study, the SegFree model introduces a memory mechanism that relies on two feature functions:

- **Reverse translation model (Reverse MT)**: This model assigns a score to each position in the source by calculating the probability of a partial translation $\boldsymbol{x}_j^a$, a being the relative position from the start of the active chunk ($a \in [1, |x|]$), followed by an end-of-sentence symbol ("</s>"). Formally, it is represented as $h_f(a, \boldsymbol{x}, \hat{\boldsymbol{y}}) = p_{y \to x}([\boldsymbol{x}_j^a, \text{</s>}] | \hat{\boldsymbol{y}})$, which is the product of individual token probabilities including the end marker. The reverse model employs the same architecture and training data as the forward model but reverses the translation direction during training.

- **Linear Regression-based Normal Distribution (LinReg)**: This model defines $h_f(a, \boldsymbol{x}, \hat{\boldsymbol{y}})$ as a normal distribution $\mathcal{N}(a \mid \theta_\mu \cdot |\hat{\boldsymbol{y}}|, \theta_\sigma^2)$, where the mean and variance are estimated using Ordinary Least Squares regression based on the length of the hypothesis translation, $|\hat{\boldsymbol{y}}|$.

4.1.4. Evaluation

Additionally to the traditional MT metrics BLEU and COMET seen, which are also suited for simultaneous MT [42], streaming MT systems must contend with real-time latency constraints. This demands the adoption of specific metrics to measure latency effectively. Currently, translation latency metrics include the Average Proportion (**AP**) [43], Average Lagging (**AL**) [39] and Differentiable Average Lagging (**DAL**) [44]. For our study, the AL metric will be used, which is the standard in conferences[1] given its consistency, and informativeness.

Although these metrics are the standard, they are evaluated on a sentence-level basis, meaning they compute the translation latency for each sentence independently, ignoring potential interactions that can result from accumulated delays in a real-world streaming scenario [45]. Further research efforts should address this limitation to provide a more comprehensive evaluation of streaming MT systems.

These metrics can be described as a normalization of the number of read-write word operations needed to produce a translation $\boldsymbol{y}$ from a source sentence $\boldsymbol{x}$:

$$L(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{y})} \sum_i C_i(\boldsymbol{x},\boldsymbol{y}) \tag{4.7}$$

Here, Z is a normalization function, i is an index over the target positions, and C_i is a **cost function** for target position i. C_i is defined depending on the specific latency measure being used. The cost function for AL is:

$$C_i(\boldsymbol{x},\boldsymbol{y}) = g(i) - \frac{i-1}{\gamma} \tag{4.8}$$

Here, $g(i)$ represents the number of source tokens read when a token is written at position i, and γ is the target-to-source length ratio $\frac{|\boldsymbol{y}|}{|\boldsymbol{x}|}$. The AL cost function takes the number of source words the model falls behind a *wait-0 oracle*. This oracle distributes source words uniformly across target positions based on the ratio $\frac{1}{\gamma}$.

The **normalization function** Z varies depending on the specific measure used. The normalization function for AL is:

$$Z(\boldsymbol{x},\boldsymbol{y}) = \operatorname*{argmin}_{i:g(i)=|\boldsymbol{x}|} i \tag{4.9}$$

The AL normalization term normalizes over the number of target positions, which in AL is limited to those target positions reading new source tokens. These sentence-level latency measures are typically reported as an average value over an evaluation set comprising multiple sentence pairs, with each pair evaluated independently from each other.

All this is, however, not straightforwardly useful for the latency evaluation of a continuous paired stream of sentences, with one of the few evaluations strategies used in this context being the $Concat-1$, which consists of concatenating all sentences into a single source/target pair in order to compute the corresponding latency measure. A study of this strategy compared to the conventional one that considers independent sentences [45] concludes that enhancements to the accuracy of AL could be achieved if there were means to obtain sentence-level estimates for γ in a streaming setting. Then, a streaming adaptation of the cost functions from Equation 4.8 could be done based on a global

[1] https://iwslt.org/2021/simultaneous

function $G(i)$, which outputs the count of source tokens read, encompassing those from preceding sentences, similar to the Concat-1 strategy.

$$C_i(\boldsymbol{x}, \boldsymbol{y}) = g_n(i) - \frac{i-1}{\gamma_n} \tag{4.10}$$

having $g_n(i) = G(i + |\boldsymbol{y}_1^{n-1}|) - |\boldsymbol{x}_1^{n-1}|$, where the global delay is transformed into a local representation, facilitating its comparison with the local sentence oracle.

4.2 Datasets

In order to conduct our experiments, it is essential to examine the data available for use. This assessment will inform the selection of appropriate datasets for training our models and ensure that the data aligns with the objectives of our study.

First, since the NLLB model processes sentences individually, it ignores the possibility of an input being part of a larger body of text, and thus is not able to leverage previous context, an issue that **document-level** MT addresses. Additionally, the model requires complete sentences as input. Therefore, if the input is a continuous stream of words, which may form an unfinished sentence, the model would not be able to process it correctly, a problem that **prefix training** can resolve. Moreover, in most cases the source of this text stream is a lowercased and unpunctuated audio transcription, which involves *spoken language*. This differs from the written language that constitutes the majority of the data the NLLB has been trained on.

Thus, the training of our LoRA will be aimed towards making our model able to work under these conditions. For this end, the *data* used will ideally be formed by large bodies of related text (documents, books...) and from spoken language (speech transcriptions). Then, this data will have to be preprocessed and prepared for the model to be able to learn the new task.

An excellent source of data for our needs can be found in the field of **spoken language translation** (**SLT**), also known as speech-to-text translation. This domain effectively handles long form, spoken language data. However, research in SLT has always presented difficulties due to the scarcity of specific datasets, as existing SLT datasets are limited to a small number of language pairs. Two prominent datasets in this domain are the Europarl-ST and MuST-C datasets.

The **Europarl-ST** dataset [28] is a multilingual SLT corpus featuring paired audio-text samples for speech-to-text translation from and into 9 European languages (English, German, French, Spanish, Italian, Portuguese, Romanian, Polish and Dutch), for a total of 72 different translation directions. This corpus was obtained from publicly available videos of debates held in the European Parliament between 2008 and 2012. A summary of the data can be seen on Table 4.1.

The **MuST-C** dataset [29] is another multilingual SLT corpus for spoken language translation from English into 14 languages (German, Spanish, French, Italian, Dutch, Portuguese, Romanian, Russian, Arabic, Chinese, Czech, Persian, Turkish and Vietnamese). MuST-C comprises hundreds of hours of audio recordings from English TED Talks, automatically aligned at the sentence level with their manual transcriptions and translations. A summary of the data can be seen on Table 4.2.

These two datasets will be utilized as our training datasets since they currently provide the best fit for our task. The dev and test sets from Europarl-ST will be employed to assess the performance of our model, as well as that of the baseline model. As previously

Table 4.1: Europarl-ST dataset. Total number of hours for each language direction.

src/tgt	en	fr	de	it	es	pt	ro	nl
en	-	81	83	80	81	81	72	80
fr	32	-	21	20	21	22	18	22
de	30	18	-	17	18	18	17	18
it	37	21	21	-	21	21	19	20
es	22	14	14	14	-	14	12	13
pt	15	10	10	10	10	-	9	9
ro	24	12	12	12	12	12	-	12
nl	7	5	5	4	5	4	4	-

Table 4.2: MuST-C dataset. Total number of talks, hours, and sentence pairs for each language direction.

src/tgt	Talks	Hours of speech	Sentence pairs
en-de	2093	408	234K
en-es	2564	504	270K
en-fr	2510	492	280K
en-it	2374	465	258K
en-nl	2267	442	253K
en-pt	2050	385	211K
en-ro	2216	432	241K

mentioned, the primary focus will be on one language direction, specifically $en \to es$, which is present in both datasets, allowing us to leverage them effectively. This results in a training dataset comprising 304,634 sentence pairs sourced from Europarl-ST and MuST-C, along with 1,316 sentence pairs for the dev set and 2,502 for the test set, all exclusively from Europarl-ST.

4.2.1. Data processing pipeline

With our chosen datasets, processing is initiated for model training [1]. In order to exploit the **document-level information**, sentences are augmented with their respective streaming history [21]. This involves concatenating previous source and target sentences until a maximum of 50 tokens is reached. Sentence boundaries are defined using special tokens: [DOC] indicates the beginning of a document, [CONT] signifies being in the middle of a document, [END] marks the end of a document, and [SEP] separates the sentences in the document. This part of the processing is crucial for our fine-tuning, as it enables our model not only to use the document-level information. Given that our model was extensively trained without this context before, formatting the input with these special tokens signals to the model that it is engaged in a distinct separate task, which might play a role in preventing interference from the prior training when the model is performing streaming MT.

As previously mentioned in Section 4.1.2, the **prefix-training** data augmentation technique [41] is used to enable simultaneous translation. The model is trained using both the original training data and the augmented partial prefix data. Figure 4.2 previously seen illustrates an example of this data processing.

Lastly, as previously mentioned, the data processing could also make the input resemble the raw output of an ASR system. This entails *lowercasing* the source side of the dataset and eliminating *punctuation marks*. However, this case will be examined separately in Section 4.5, as previously indicated.

Now that our data processing pipeline is prepared, in the next section, our model and its training process will be examined. Different approaches and hypothesis will be explored, and the benefits they provide will be discussed and their results evaluated.

4.3 Experiments on NLLB - Setup

To implement our NLLB adaptation, several factors need to be addressed. The NLLB model operates on a sentence-by-sentence basis, taking a complete sentence in one language and generating its translation. Therefore, using it for real-time usage requires adjustments. This entails modifying the model to handle the new task, and obtaining appropriate data as well as processing it in a way that allows the model to effectively learn from it. Subsequently, a real-time situation will be simulated, and the model's performance will be evaluated.

Furthermore, an additional challenge arises. In streaming MT models, inputs typically consist of lowercase text streams without punctuation coming from ASR systems, although exceptions may exist, such as when a casing module is present. This also implies that the models are able to punctuate their output. However, the NLLB model is predominantly trained on truecased text, therefore it lacks punctuation training, and cased tokens become redundant in a non-cased setting, therefore making much of its size and power becomes ineffective and potentially detrimental to performance in such a setting.

For this reason, our study is separated into two parts: the first one focuses on enabling the NLLB to work with an input including casing, while the next aims to make it work on an input without casing nor punctuation. Our baseline model exclusively handles lowercase inputs, and thus it will be used as a reference on the second section for the results to be comparable, while in the first one, the model's offline performance will serve as our reference. Since the primary objective of our experiments is to assess the plausibility of the streaming adaptation, attention will primarily be directed towards a single language direction.

4.3.1. Model Training

In order for the NLLB model to undergo training using our processed corpora, modifications are necessary. Particularly, the new *special tokens* introduced previously need to be incorporated to capture the document-level information:: [END], [DOC], [SEP] and [CONT]. This requires modifying both the *tokenizer*, to which these tokens need to be added, and the *token embedding matrix* of the model, which now has to include the new token embeddings. The initialization values for these new token embeddings are a subject of debate. In our case, random initialization is opted for, although alternative options exist. These include initializing them to the mean of the other token embeddings, or temporarily freezing the model to conduct an initial training solely on the new token embeddings before subsequently unfreezing and training all of them.

Given that our embedding matrix has changed, our LoRA training now incorporates the embedding matrix as an additional target module alongside the $\boldsymbol{Q}$, $\boldsymbol{V}$, and $\boldsymbol{O}$ matrices from our previous offline experiment. The remaining LoRA hyperparameters and training setup remain consistent with those defined in the offline experiment in Section 3.3, although it was found through experimentation that training for up to 5 epochs produces slightly better results.

4.3.2. SegFree Architecture

With our trained model ready, only the inference and evaluation setups need to be established. As previously said, the **SegFree architecture** will be implemented by employing the Segmentation Free Toolkit from the MLLP group. This toolkit was also used to evaluate the SegFree model described in the SegFree paper [1], which is established as our baseline for state-of-the-art performance as already mentioned. Key differences between this model and the NLLB models include its training exclusively with lowercased text as input, and its use of an additional model for the reverse translation model feature that performs the segmentation.

The SegFree system utilizes a *memory mechanism*, as detailed in Section 4.1.3, to track the translated chunks of the source stream and their corresponding translations. During **inference**, model latency is managed using a *wait-k* policy, outlined in Section 4.1.1. The policy ignores words in the streaming history, focusing only on the active chunk for READ or WRITE decisions. A *beam search* with a beam size of 4 generates hypotheses, from which the best scoring one is selected. The search is initialized with a target prefix of already committed words. Then, according to the wait-k policy, a number of the generated words are committed as a WRITE operation, and the rest are discarded.

Whenever a target sentence is committed (indicated by the "[SEP]" token), the segmentation mechanism is triggered, and the length of the streaming history is checked. If it surpasses the maximum *history size* on either the source or target side, pairs of segments are removed from the streaming history until the word count is below our chosen 50-word limit.

As for the **reverse translation model feature** the segmentation mechanism uses, the SegFree baseline model operates only in the *en* → *es* language direction, and thus an additional *es* → *en* model is used for the reverse translation model feature. Since the NLLB model is a multilingual model, the reversed language direction training is incorporated into the model's training, allowing us to use the same model for both directions, eliminating the need for an additional model.

In the original SegFree paper, the reverse translation feature calculates the score for each position as the probability of a partial translation followed by an end-of-sentence (EOS) symbol, which is the product of individual token probabilities including the EOS marker [1]. It was found experimentally that giving more weight to the probability of the EOS token alone improved results in our case. Since we are interested in cases where the EOS is highly probable, indicating a complete phrase and potential alignment, and since we are working with log probabilities, squaring the EOS probability penalizes scores for partial translations where the model is uncertain about the phrase's completeness, even if the overall sequence score (taking into account the probability of the other tokens) is high.

4.4 Experiments on NLLB - Cased NLLB

In this section, the evaluation focuses on the NLLB-adapted model using cased inputs. As previously mentioned, a comparison with the baseline model from the SegFree paper [1] will be presented in the next section. In that comparison, the model will be trained on lowercase input, which is the expected format for the baseline model.

A performance baseline was established by testing the model in an offline context, both with and without a *streaming history*. This **offline baseline** sets an upper bound of the model's peak performance, as it is tested under ideal conditions (it receives the entire, pre-aligned phrases). By comparing the streaming performance to this offline baseline, the degradation of the model's performance on streaming can be evaluated. Furthermore, comparing the offline results with and without streaming history helps us assess how effectively the model leverages the streaming history, which would indicate the training for streaming scenarios, where streaming history is crucial, was successful.

Then, additionally to the model trained with the hyperparameters and the data previously mentioned in Section 4.3, different hypotheses on how the model could be improved are tested. For this, fine-tune is applied under different conditions, ending up with a variety of fine-tuned models:

- **NLLB-600M LoRA Ranks**: Our base fine-tuning performs a LoRA with a rank of 16. A *higher rank* could allow the model to capture more complex patterns and nuances in the data since more parameters can be fine-tuned, potentially preventing underfitting. This is particularly crucial for our case, as the model needs to learn a completely new task. Conversely, a *lower rank* would result in lower training times and reduced memory consumption, although it could impact translation quality.

- **NLLB-600M Extended**: A general rule of thumb is that more data leads to better model performance. To test this, the training data will be expanded beyond our initial datasets, Europarl-ST and MuST-C, which are concise and relevant to our evaluation set (a subset of Europarl-ST, composed of parliamentary debates). The new data, which needs to adhere to the format described in Section 4.2, includes the *News Commentary* dataset [30] which consists of 49,089 bitext sentences from newspaper articles, the *QED* dataset [46] containing 1,115,444 sentences from subtitles of educational videos and lectures, and the *Biomedical WMT22* dataset [27] with 169,884 sentences from biomedical abstracts. Including reverse direction training for the reverse model, this amounts to a total of 3,238,102 sentences, compared to the baseline's 609,268 sentences, representing a fivefold increase. However, this additional data is less domain-specific, which might not entirely benefit the model.

- **NLLB-600M Multilingual**: A multilingual MT model is being utilized, capable of handling up to 200 different languages. As demonstrated in Section 3, when evaluating for a specific language direction, fine-tuning the model to it enhances performance, as it guides the model to focus on that particular direction. Extending fine-tuning to multiple directions tends to dissipate this benefit. However, since not only adaptation to a particular language direction is involved but also the introduction of a new task, our model may benefit from fine-tuning for streaming MT across various language directions, as the embedding space is shared for all languages, and the adaptation of just one language could distort the representation.

  To explore this, our model will be trained on languages included in both Europarl-ST and MuST-C: English, Spanish, German, French, Italian, Portuguese, Romanian, and Dutch. The training will focus on *English → (Spanish, German, French, Italian, Portuguese, Romanian, Dutch)*, with performance evaluated on English → Spanish. Languages similar to Spanish will progressively be incorporated into the training: first, only Italian; then, only Portuguese; followed by both Italian and Portuguese; next, Italian, Portuguese, French, and Romanian; and finally all seven languages. The goal is to mitigate potential drawbacks of multilingual training by using languages closely related to Spanish.

To enable the trained models to also translate into these newly included language directions, the training data must also incorporate the reverse translation direction for all language pairs. This ensures the reverse translation feature functions effectively for all included directions.

- **NLLB-1.3B & NLLB-3.3B**: As seen in Section 3, the larger 1.3B and 3.3B parameters models deliver better performance. However, in the context of streaming MT, latency becomes a significant issue. For real-time applications, generation times are crucial, and larger models tend to produce inferences more slowly. Furthermore, our AL latency metric is not computationally aware, that is, it does not account for the computation time required for inferences, making it particularly unsuitable for direct comparison between models of different sizes. Consequently, the 1.3B and 3.3B parameter models cannot be fairly compared with the 600M parameter model or the SegFree baseline, as the latter two have faster inference times that are not reflected in the AL metric. The NLLB-600M model, being closer in size to the SegFree baseline (which consists of two 300M parameter transformers) offers a more appropriate comparison.

4.4.1. Results - Cased NLLB

The streaming LoRA-adapted models performance is assessed by varying the k parameter of the wait-k policy to analyze the tradeoff between latency and quality across values from 1 to 10. Performance metrics include BLEU and COMET scores, while latency is measured using AL, which indicates the delay in words compared to an ideal real-time system. Three regimes can be determined for the AL: low latency $(AL \leq 3)$, medium latency $(AL \leq 6)$ and high latency $(AL \leq 15)$ [47].

Figure 4.4 illustrates the outcomes of the *LoRA rank* experiments, depicting adaptations for LoRAs with ranks 4 and 16. Tests with rank 64 yielded performance similar to rank 16, and rank 8 closely mirrored rank 4; hence, these were excluded from Figure 4.4 for clarity. The rank 64 LoRA results suggest that a rank of 16 may be sufficient to capture the necessary amount of parameters. On the other side, reducing the rank to 4 or 8 slightly degrades model performance, although not significantly, with the overall quality remaining comparable. These results indicate that there are no substantial performance differences across the tested ranks.

The best streaming case, with $k = 10$, lags behind the offline baseline by 0.89 BLEU points and 1.67 COMET points. This suggests that our streaming model could potentially improve its results by this margin. However, having such a small difference also implies that the training was quite successful in transferring the model's translation capabilities into the streaming context.

Translations with BLEU scores in the range of 30-40 are considered good translations, and scores between 40-50 indicate high quality. Similarly, a COMET score around 85 indicates high quality [48]. Overall, for $k = 4$, BLEU scores of approximately 40 are achieved with an AL around 4, which is slightly above the low latency threshold. The best results are found around $k = 6$, offering a good balance between quality and latency for medium latency requirements. Beyond this range, the benefits of increasing k become less pronounced, and the high latency domain is entered.

Figure 4.5 shows the results of training with *additional data*. Training with an extended dataset appears to negatively affect performance in our case, probably because the data strays too far from the test set's domain (parliamentary debate and spoken text), the lower-quality of some datasets like QED, the lack of a preprocessing to clean the new data, and the presence of shorter sentences that teach the model to segment more of-

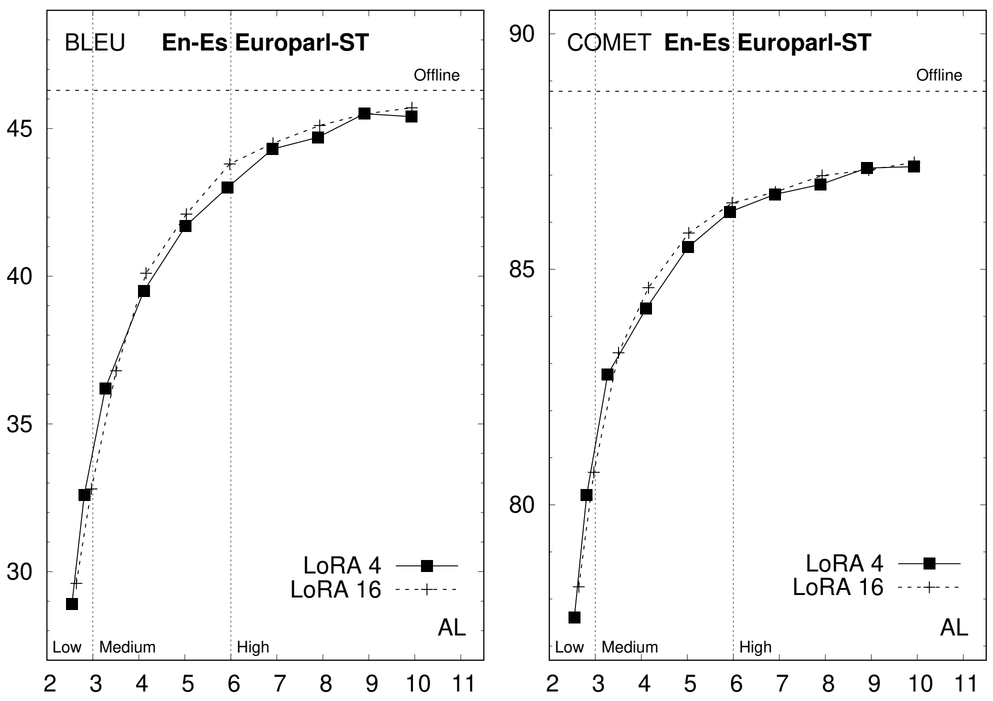

Figure 4.4: Rank experiment: BLEU vs. AL (left) and COMET vs. AL (right) at three regimes, low, medium, and high latency, on the English to Spanish Europarl-ST test sets for the the NLLB-600M LoRA adapted model. LoRAs displayed have ranks 4 and 16, and an offline baseline is established using the rank 16 model.

ten. This last issue would be expected to worsen in the next section, where there is no punctuation to guide the model's segmentation decisions.

Therefore, it appears that the model adapts well to the streaming task, indicating that in-domain and high-quality data are more crucial than providing the model with an enormous amount of data to learn the new task.

Figure 4.6 shows the results of utilizing the *larger* versions of the NLLB model. As expected, the NLLB-1.3B model outperforms the NLLB-600M, showing an average improvement across all values of k by 1.75 BLEU and 0.9 COMET. The most significant variability is seen at $k = 3$ with a BLEU increase of 2.5 and at $k = 4$ with a COMET increase of 1.57. Similarly, the NLLB-3.3B model outperforms the NLLB-1.3B model, displaying comparable gains to those seen when moving from the NLLB-600M to the NLLB-1.3B, and particularly excelling at lower values of k. Interestingly, the performance jump from the NLLB-1.3B to the NLLB-3.3B model is quite significant, when in our offline tests in Section 3.3 the difference from these models was minor. However, as mentioned earlier, a direct comparison between these models is not possible, since the latency metric is not computationally aware.

In Figure 4.7, the outcomes of NLLB-600M models trained with *additional language directions* beyond the initial $en \rightarrow es$ direction can be observed. These results are juxtaposed with the base training, which only includes the $en \rightarrow es$ direction. A range of LoRAs were trained, progressively introducing more languages starting with those most similar to Spanish. Therefore, models were trained with the following language directions: $en \rightarrow \{es, it\}$, $en \rightarrow \{es, pt\}$, $en \rightarrow \{es, it, pt\}$, $en \rightarrow \{es, it, pt, fr, ro\}$, and $en \rightarrow \{es, it, pt, fr, ro, de, nl\}$. The models trained with the added directions $en \rightarrow it$ and $en \rightarrow pt$ separately yielded results nearly identical to the one trained with both added directions, so the first two were removed from the graph for clarity.

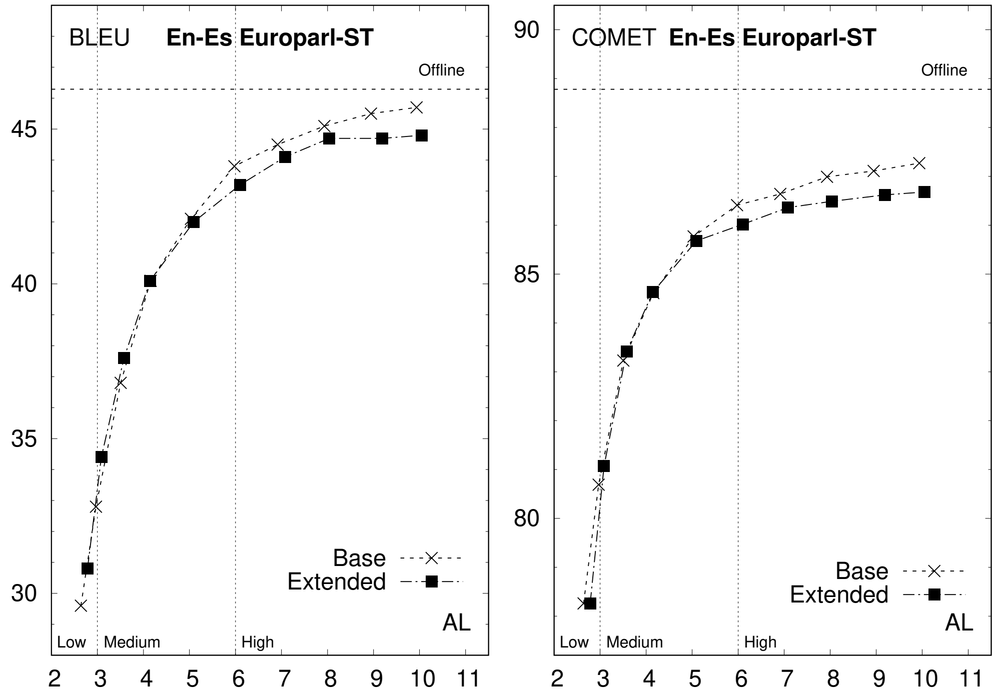

Figure 4.5: Extended data experiment: BLEU vs. AL (left) and COMET vs. AL (right) at three regimes, low, medium, and high latency, on the English to Spanish Europarl-ST test sets for the NLLB-600M LoRA adapted models with the base training and with additional training data. An offline baseline is established using the model with the base training.

Upon initial inspection, a gradual decline in terms of COMET is observed as more languages are incorporated, with a maximum loss of 1.56 COMET, although minimal differences are observed in terms of BLEU between all the models. The largest degradation in performance in the *en* $\rightarrow$ *es* direction is observed in the model trained with seven language directions. Given that no additional improvement has been observed from incorporating additional language directions, the hypothesis that a training including streaming for multiple languages would enhance overall streaming performance does not appear to be substantiated.

However, these differences, particularly on the BLEU metric, are small. A paired t-test conducted for statistical significance even shows no significant difference between the base training and the multilingual training encompassing seven language directions, with a p-value of 0.66 for the BLEU metric. Furthermore, the gap between the model trained solely on the *en* $\rightarrow$ *es* directions and the one trained on seven language directions narrows when reverse translation training is excluded for all languages, as illustrated in Figure A.1 in the Appendix. With our training encompassing seven language directions, a single model capable of translating from English to seven languages has been developed, and for values of k at six and below (the range that would be used in deployed models, with latencies in the low to medium regimes) the model performs comparably to the one trained solely for the *en* $\rightarrow$ *es* direction. These results suggest our model can be effectively used within a **multilingual streaming framework**, where the same model translates in real-time to different languages, with minimal negative impact to translation quality.

This presents a clear advantage over the SegFree baseline *bilingual* model, which operates exclusively within a single language direction. With such model architectures, additional models would have to be trained to work with new language directions. In contrast, our approach only needs to train a single LoRA, which is fast and needs mini-

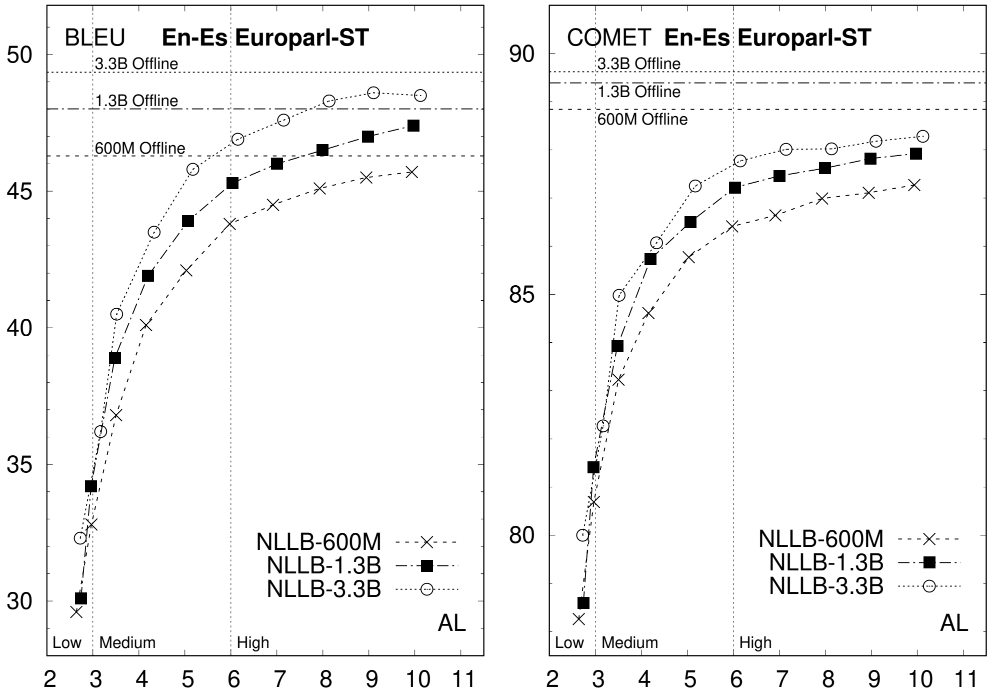

Figure 4.6: Size experiment: BLEU vs. AL (left) and COMET vs. AL (right) at three regimes, low, medium, and high latency, on the English to Spanish Europarl-ST test sets for the LoRA streaming adapted NLLB-600M, NLLB-1.3B and NLLB-3.3B models. Offline baselines are established for each model.

mal memory space, and provides a single streaming model capable of translating across a multitude of different languages. Moreover, for applications prioritizing maximum quality, aiming to compensate the differences in COMET scores observed in Figure 4.7, multiple LoRAs could be trained and seamlessly interchange them to our needs while maintaining the same base NLLB model, which is fast and minimizes storage requirements, given the lightweight nature of LoRAs.

An experiment was conducted comparing the original unmodified NLLB-600M model, our streaming LoRA-adapted model, and a new LoRA adaptation of the model using the same datasets as the streaming adaptation but in an *offline setup*. This experiment aimed to evaluate the offline performance of the streaming-adapted model to determine if it could also function effectively in an offline environment. The results reveal that our LoRA streaming-adapted model outperforms the *original model*, as shown in Table 4.3. Therefore, our streaming model can also serve as an offline model, even surpassing the original NLLB model and offering an advantage over models exclusively trained for streaming MT. Then, the LoRA *offline adaptation* showed a modest improvement of 0.25 BLEU and 0.08 COMET over the streaming adaptation, indicating that the LoRA successfully learns the new task without significantly degrading offline performance in our streaming adaptation.

Table 4.4 shows BLEU and COMET scores for the base NLLB-600M and NLLB-1.3B streaming-adapted models when evaluated in an offline setup with and without the use of *context*. Context is provided by including the source and translation from the previous iteration (either the model's translation or the correct *golden translation*), while also adding [DOC] and [SEP] tokens.

As shown, adding the model's translation did not lead to an improvement on neither of the models. This failure to leverage the context might be due to the model not

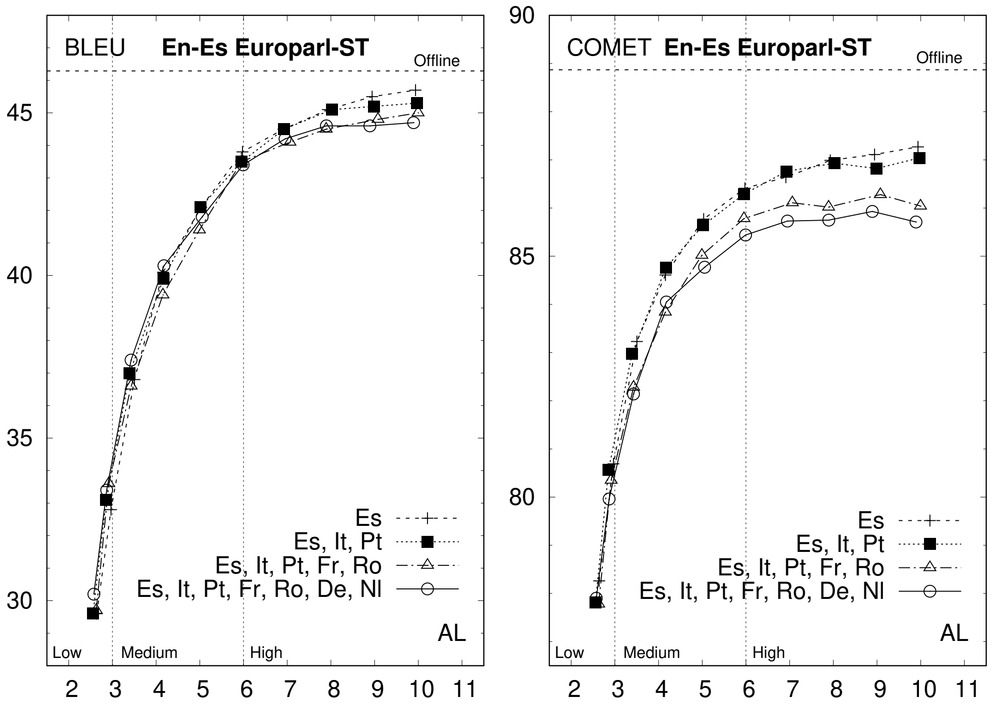

Figure 4.7: Multilingual experiment: BLEU vs. AL (left) and COMET vs. AL (right) at three regimes, low, medium, and high latency, on the English to Spanish Europarl-ST test sets for the NLLB-600M multilingual models: the model fine-tuned with 7 language pairs, and the one with just italian and portuguese added. The *en* → *es* base model is also added, and it also establishes the offline baseline.

Table 4.3: Offline performance of the original NLLB model, the streaming LoRA adaptation of the NLLB model, and an additional LoRA adaptation using the same data for an offline setup.

Model	**BLEU**	**COMET**
Original	44.74	88.51
LoRA Offline	46.54	88.92
LoRA Streaming	46.29	88.84

learning to use it properly with the fine-tuning, not being able to handle long dependencies, relying too much on its previous training and still preferring a sentence-to-sentence processing, or not properly translating the context sentence is using and therefore, propagating the errors to the new translation. The improvement seen with golden translations primarily in BLEU scores, but not COMET scores, suggests that the translations are closer to the references rather than genuinely better in quality. The larger NLLB-1.3B model also fails to utilize context more effectively, indicating that the issue is not related to model size.

In conclusion, the results of our different experiments suggest that deviating from the base training approach minimally impacts model performance. Training a small LoRA with just enough data to learn the new task adapts our model while enhancing its offline performance. Nevertheless, further hyperparameter optimization, such as adjusting the learning rate and the LoRA alpha scaling factor, or the LoRA rank in the extended dataset and multilingual settings, could potentially yield different outcomes.

Furthermore, by incorporating additional language directions into the training process, the way is paved for a single model capable of multilingual streaming translation.

Table 4.4: Offline evaluation with and without context in the test set for the NLLB models LoRA adapted for streaming MT.

Setup	**BLEU**	**COMET**
600M base train w/o context	46.29	88.84
600M base train w/ context	46.29	88.83
+ Golden translations	46.68	88.90
1.3B base train w/o context	47.92	89.39
1.3B base train w/context	48.01	89.38
+ Golden translations	48.24	89.44

Alternatively, various LoRAs can be trained, each fine-tuned for specific language directions. Therefore, unlike current state-of-the-art streaming MT models, such as our SegFree model baseline, our adaptation opens a path for streaming MT models that can also work offline, and with support for multiple language directions.

In the following section, adaptation of our model to a more realistic scenario will be explored, where lowercased input, akin to the type of output an ASR system produces, is handled. Results will be compared with those of the SegFree baseline model in this section.

4.5 Expriments on NLLB - Lowercased NLLB

In this section, NLLB-adapted model using *lowercased inputs* is evaluated. As previously mentioned, this scenario is more representative of real-world applications, where streaming MT models are part of a cascade speech-to-text setup, receiving input from an ASR system that typically outputs lowercased text. Its performance is compared to the reference state-of-the-art SegFree model [1], which is trained specifically for streaming from scratch with lowercased inputs. However, it is important to note that this reference should be regarded as an *upper bound* rather than used for a direct comparison, due to the differing training methods and data used.

The *training setup* of the model remains identical to our previous experiment with cased inputs. However, this time, the source side of the input will additionally be lowercased and have all its punctuation marks removed, while the casing on the target side will be maintained. This adjustment introduces a new challenge for the model: to correctly case and punctuate its translations when the input lacks these features. As previously explained, this significantly increases the complexity of the task at hand.

The *evaluation* process follows the same protocol defined in Section 4.4: the wait-k policy with values of k ranging from 1 to 10 is employed, translation quality is assessed using BLEU and COMET, and latency measured using AL. An offline baseline without using context will also be established for comparison.

Analogously to Section 4.4, in this section the experimental setup explores the LoRA rank, the use of additional data and the model size. However, for the sake of clarity, only those figures drawing significant conclusions are reported in this section, while the rest of figures are available in AppendixA for the interested reader. A larger LoRA rank can address increased complexity, the extended dataset training can provide the model with more opportunities to enhance its casing capabilities, and the larger 1.3B and 3.3B models are used to assess potential translation quality improvements due to increased model capacity.

The results of the LoRA rank and extended dataset experiments can be found in Appendix A. Figure 4.8 shows the results of exploring *larger* versions of the NLLB model, alongside those of the SegFree baseline model. The smaller 600M and 1.3B models can achieve performance comparable to the SegFree baseline at lower latency values, particularly when higher values of k are used, but from medium latency onwards, the SegFree baseline outperforms the fine-tuned models in both quality and latency. The NLLB-3.3B model gets closer to the SegFree baseline, achieving nearly equivalent COMET scores to the SegFree baseline. However, its inference time is considerably longer.

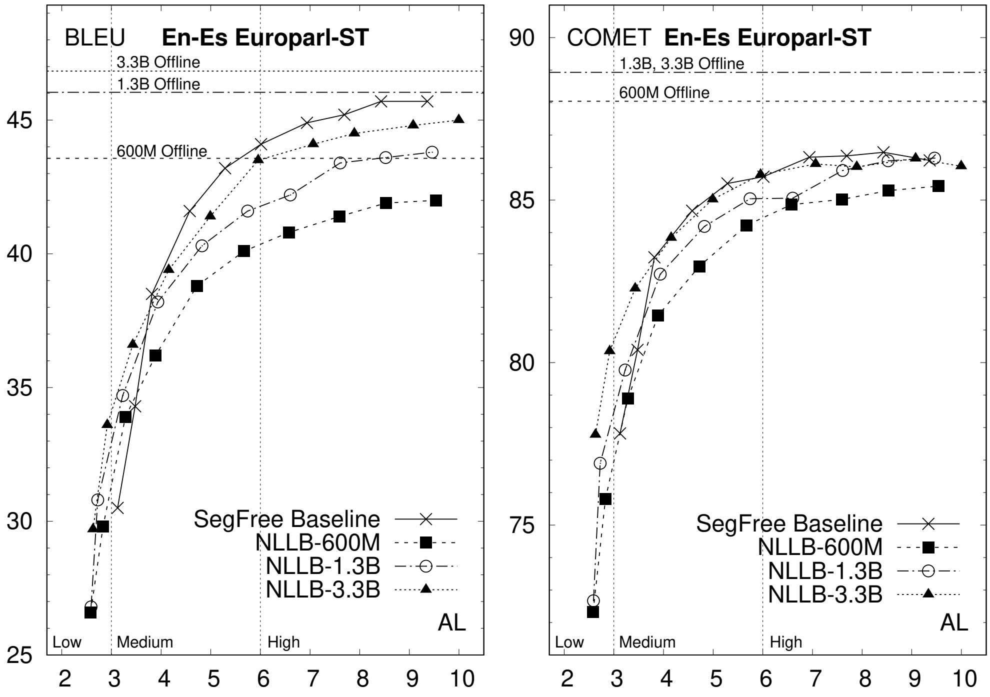

Figure 4.8: Size experiment: BLEU vs. AL (left) and COMET vs. AL (right) at three regimes, low, medium, and high latency, on the English to Spanish Europarl-ST test sets for the NLLB-600M, NLLB-1.3B and NLLB-3.3B models trained with lowercase input, together with the SegFree baseline model.

On average, for a given value of k, the SegFree model outperforms the NLLB-600M LoRA adapted with the base training by 4.22 BLEU points, and 2.65 COMET points. Even disregarding the fact that the latency metric is not computationally aware, the performance of the larger models, including the 3.3B model which is eleven times the size of the SegFree baseline, falls short. This significant degradation due to the additional casing task was expected since the NLLB model is solely a translation model.

The gap between streaming and *offline performance* of the adapted model closely mirrors that observed in the previous section. However, closing this gap alone won't be enough to match the SegFree baseline model's BLEU score, although there is potential for an improvement in COMET score that would. This suggests that surpassing the SegFree baseline model is achievable even with the current state of the training and adaptation methods, paving the way to potentially reach state-of-the-art performance levels.

Figure 4.9 illustrates a comparison between the NLLB-600M model adapted with cased input (as discussed in Section 4.4) and the same model adapted with lowercased input from this section. When lowercased input is used, there is an observed average decrease of 3.22 BLEU points and 2.89 COMET points. This translates to a relative performance decline of 8.6% in BLEU scores and 3.5% in COMET scores compared to the cased input adaptation.

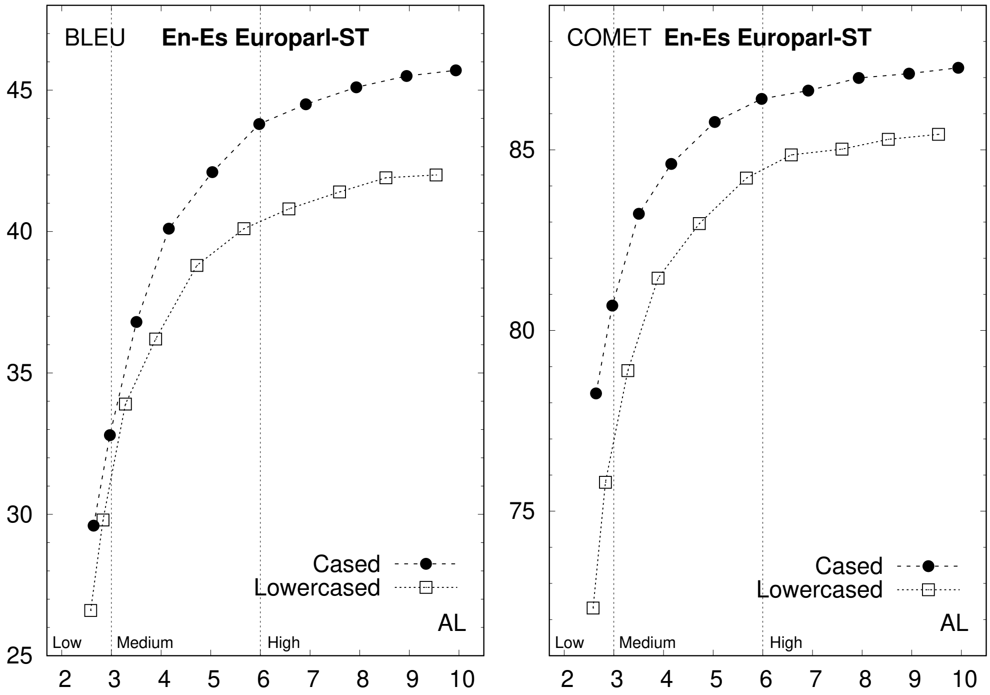

Figure 4.9: Casing comparison: BLEU vs. AL (left) and COMET vs. AL (right) at three regimes, low, medium, and high latency, on the English to Spanish Europarl-ST test sets for the NLLB-600M model trained with cased input and with lowercase input.

Figure 4.10 illustrates the outputs of both models for an example input, showing that the translation quality is quite comparable.

A further study is necessary to determine the full extent of the model's adaptability to the new task. The upcoming section will discuss different points, highlighting potential lines of research to fully leverage the potential of LLMs for streaming MT.

Input

commissioner i would like to ask you once more about our famous ignalina atomic plant recently our prime minister met the president of the european commission mister barroso and in our papers there were some interpretations that there is a gap there is a possibility of prolonging the work of this station what is your opinion and what would you recommend to the lithuanian government in this situation

SegFree	**NLLB**
Señor Comisario, me gustaría pedirle una vez más información sobre nuestra famosa planta atómica de Ignalina. [SEP] Recientemente, nuestro Primer Ministro se reunió con el Presidente de la Comisión Europea, el señor Barroso, y en nuestros documentos había algunas interpretaciones de que hay una brecha. [SEP] Existe la posibilidad de prolongar el trabajo de esta estación. [SEP] ¿Cuál es su opinión y qué le recomendaría al Gobierno lituano en esta situación? [SEP]	Señor Comisario, quisiera preguntarle una vez más sobre nuestra famosa planta atómica de Ignalina. [SEP] Recientemente, nuestro Primer Ministro se reunió con el Presidente de la Comisión Europea, el señor Barroso. [SEP] Y en nuestros artículos había algunas interpretaciones de que hay una brecha. Hay una posibilidad de prolongar el trabajo de esta estación. [SEP] ¿Cuál es su opinión? ¿Y qué recomendaría a la administración lituana en esta situación?

Figure 4.10: Outputs for the NLLB-600M adapted model and the SegFree baseline model for the input in gray.

CHAPTER 5
Conclusions

In this work, the possibility of adapting an LLM for streaming MT was explored. Following an initial examination of the adaptation of LLMs in offline scenarios, the necessary data to develop a streaming MT model was successfully acquired and processed. Subsequently, Meta's NLLB LLM [25] was effectively adapted, achieving excellent results with cased inputs adaptation and competitive, promising results with lowercase inputs adaptation, the case that better reflects real-world scenarios.

The performance of the NLLB adapted model in the new streaming setting closely resembles its performance in the offline setting for which it was originally designed, demonstrating the success of the adaptation. A comparison with a state-of-the-art streaming MT model also revealed that the results of the adapted model are highly competitive, suggesting that there is still potential for further improvement to bring the adaptation to state-of-the-art levels.

5.1 Objectives achieved

Data for streaming MT adaptation was successfully sourced from speech-to-text datasets including Europarl-ST and MuST-C. A successful data processing pipeline was established, leveraging document-level MT techniques to exploit context and employing prefix training to facilitate the translation of incomplete phrases.

The model was successfully modified with new special tokens to allow learning of the new task and subsequently trained using the processed data. The LoRA adaptation was successful, resulting in an NLLB model capable of streaming MT. A variety of training approaches were tested and evaluated to further refine the model's performance.

Excellent translation quality was achieved, particularly for cased inputs as outlined in Section 4.4. The model performs nearly as effectively in the streaming setup as it does offline, showcasing its successful adaptation to the new task and achieving a quality level comparable to state-of-the-art specialized streaming MT models.

5.2 Future work

Throughout this study, various hypotheses have emerged, aimed at enhancing performance for the adapted model, and devising new approaches for streaming MT in production.

A significant finding is that training the LoRA with various language directions did not significantly degrade the translation quality in our evaluation of the *en* → *es* direction. This paves the way for **multilingual streaming MT**, where a single model could be employed to translate to different languages. This can be implemented by using the same model in a single GPU, increasing the batch size by the amount of additional languages desired for inference, or more ideally with a multi-GPU setup, with the model loaded onto each GPU and having each one perform inferences for one language direction at maximum speed. Since LoRAs are easily trainable and lightweight, and LoRA adapters can be interchanged effortlessly, there is the possibility to use either the model trained across all language directions or individual LoRAs for each language direction to maximize performance in the multi-GPU setup. Additional tests should be conducted using the model trained on seven language pairs, also comparing it with LoRAs trained for single language directions to evaluate the quality degradation.

Various approaches have surfaced to **enhance streaming** outcomes too, encompassing improvements in training methodologies, dataset selection and processing, or streaming setup configurations.

Our examination of model performance in offline settings, with and without additional **context**, revealed that the addition of extra context did not provide better results, unless the context included was the golden translation rather than the model-generated one. This suggests that the model is not properly leveraging context, possibly due from inadequate fine-tuning, inability to handle long dependencies, over-reliance on previous training and still preferring a sentence-to-sentence processing, or inaccurate translation of the context sentence, leading to error propagation in new translations.

The model's **generation of [SEP] special tokens** also exhibits inconsistencies, often generating consecutive [SEP] tokens and empty sentences between them. This issue frequently arises when a new sentence, meaning a semantically encapsulated block that would be delimited by alignments, starts to get into the input. This happens when the source is almost empty, either because a new stream is starting or the streaming history has been cleared, or when the input has only a few words appended after a position where the model placed a [SEP]. This behavior suggests that the model struggles with very short prefixes and doesn't know how to proceed in such cases. Additionally, the model tends to oversegmentate, likely due to the presence of very short lines in the training data that teach the model to insert too many [SEP] tokens.

In Section 4.5, the adaptation of the model with **lowercase inputs** was explored, which required the model to also learn how to case the output. The new casing task showed detrimental to performance. Learning to leverage context through prefix and document-level training was already challenging for the model, but still related to translation. The impact on performance from the additional casing task is possibly due to it not typically being associated with translation, and the model likely had little prior exposure to it. To address this issue, solutions proposed include integrating a casing tool into the speech-to-text pipeline to provide cased inputs to the translation model, enhancing the model with additional casing-focused training data, studying bilingual models to assess how the presence of additional language directions influences adaptation to the new tasks, or trying LLMs such as Llama 2 [49].

The **inferences** of the model are also not optimized, and changing elements like the policy could significantly boost its performance. The wait-k policy is too rigid, failing to consider scenarios where waiting for more input is necessary, or having too much is not. Additionally, our inferences generate as many words as possible, even if they are not being written in the final output translation.

Given the various issues identified, several options are proposed to address them and potentially enhance the results. Among these options, the most promising one are:

- **Additional Data**: Adding more data sourced from speech to reflect spoken language, that maintains a document-level format and is high-quality, such as the CoVoST 2 dataset [50], could significantly benefit the model. An important distinction between spoken and written text lies in the nuanced relationship between intended meaning and actual expression in spoken language. Thus, our system needs to exhibit adaptability and creativity in its interpretations, going beyond literal translations. Additionally, the impact of in-domain data could be investigated by incrementally incorporating portions of the Europarl-ST dataset during training to evaluate how this addition enhances streaming capabilities.

- **Phased Training Approach:** Implement a phased training approach where training begins with an extended dataset or various language directions to assist the model in adapting to the new task, and later fine-tune it with domain-specific test data and the target language direction.

- **Revamping Prefix Training:** Revise the prefix training phase, ensuring different prefix sizes are generated, particularly very short prefixes of a couple of words.

- **Adaptive Policies:** Experiment with different policies other than the wait-k policy. Adaptive policies considering inference time, the total available words in a given time, or the probability of a given generated inference being correct, can make low latency inferences match the translation quality of medium to high latency inferences observed in the wait-k results.

- **Inference Efficiency:** Investigate methods to speed up inferences and reduce their number. This implies the use of different policies as previously mentioned, and the use of other different criteria and heuristics, like limiting the amount of tokens generated to only what will be written by the policy for example, or avoiding inference generation at any new word that enters the stream.

- **Streaming History Impact:** Examine how the context size and its content affect translation quality and inference speed. Different criteria to keep or not parts of the streaming history could be considered to reduce unproductive inferences.

- **Reverse Feature:** In the original SegFree paper, various features were used to determine the optimal segmentation of the stream. It was found that, for our case, relying solely on the reverse model feature, which in our case was carried by the same NLLB model, was sufficiently effective. This suggests that, rather than employing the model in reverse mode and making additional inferences, the model's inference could be monitored at the point when the [SEP] token is generated, and observe which tokens does the [SEP] attend to in the source stream, which would pinpoint the segmentation locations. This approach would reduce the LoRA training process to a single direction for each language pair, eliminating the need for reverse training. Consequently, this would halve the amount of training data and focus the training exclusively on translation, potentially enhancing model performance.

- **Different Models:** Accelerating the inference generation sufficiently might make the use of larger models plausible, and different LLMs could also be examined. Alternatives might include bilingual models to assess how the presence of additional language directions influences adaptation to the new tasks, or LLMs such as Llama 2 [49] which are less focused on translation.

Additionally, other potentially beneficial options can be tested. One such option involves using the whitespace meta symbol ”〃 as a suffix instead of a prefix in NLLB's tokenizer to be able to write a full word after its last subword has been generated [1]. Optimizing special tokens by incorporating a whitespace to them, akin to other NLLB tokens, and replacing [DOC] and [CONT] tokens with [SEP] could standardize token representation and simplify it. Enhancing data preprocessing may also be tried, by removing or merging short sentences to reduce [SEP] tokens and replacing uncommon characters converted into <unk> tokens. And lastly, exploring full fine-tuning, varying hyperparameters for LoRA, and exploring alternative PEFT methods like DoRA [51] could lead to better model adaptations.

Bibliography

[1] J. Iranzo-Sánchez, J. Iranzo-Sánchez, A. Giménez, J. Civera *et al.*, "Segmentation-Free Streaming Machine Translation," Sep. 2023, *arXiv preprint arXiv:2309.14823*.

[2] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman *et al.*, "On the Opportunities and Risks of Foundation Models," Jul. 2022, *arXiv preprint arXiv:2108.07258*.

[3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations*, Oct. 2021.

[4] S. J. D. Prince, *Understanding Deep Learning.* MIT Press, 2023.

[5] J. Iranzo Sánchez, "Evaluation of strategies for the adaptation of large neural models to the task of machine translation in constrained scenarios," Trabajo Final de Máster, Universitat Politècnica de València, Valencia, 2023.

[6] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," Feb. 2019, *arXiv preprint arXiv:1803.08375*.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[8] Z. Tan, S. Wang, Z. Yang, G. Chen *et al.*, "Neural machine translation: A review of methods, resources, and tools," *AI Open*, vol. 1, pp. 5–21, Jan. 2020.

[9] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," Jul. 2016, *arXiv preprint arXiv:1607.06450*.

[10] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Jun. 2015, pp. 448–456.

[11] H. Naveed, A. U. Khan, S. Qiu, M. Saqib *et al.*, "A Comprehensive Overview of Large Language Models," Feb. 2024, *arXiv preprint arXiv:2307.06435*.

[12] T. Brown, B. Mann, N. Ryder, M. Subbiah *et al.*, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[13] P. Micikevicius, S. Narang, J. Alben, G. Diamos *et al.*, "Mixed Precision Training," in *International Conference on Learning Representations*, Feb. 2018.

[14] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: Memory optimizations Toward Training Trillion Parameter Models," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2020, pp. 1–16.

[15] B. Workshop, T. L. Scao, A. Fan, C. Akiki *et al.*, "BLOOM: A 176B-Parameter Open-Access Multilingual Language Model," Jun. 2023, *arXiv preprint arXiv:2211.05100*.

[16] D. Kalajdzievski, "Scaling Laws for Forgetting When Fine-Tuning Large Language Models," Jan. 2024, *arXiv preprint arXiv:2401.05605*.

[17] G. Xiao, J. Lin, M. Seznec, H. Wu *et al.*, "SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models," in *Proceedings of the 40th International Conference on Machine Learning*. PMLR, Jul. 2023, pp. 38 087–38 099.

[18] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318.

[19] M. Freitag, R. Rei, N. Mathur, C.-k. Lo *et al.*, "Results of WMT22 Metrics Shared Task: Stop Using BLEU – Neural Metrics Are Better and More Robust," in *Proceedings of the Seventh Conference on Machine Translation (WMT)*, P. Koehn, L. Barrault, O. Bojar, F. Bougares *et al.*, Eds. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 46–68.

[20] R. Rei, C. Stewart, A. C. Farinha, and A. Lavie, "COMET: A Neural Framework for MT Evaluation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 2685–2702.

[21] J. Iranzo-Sánchez, J. Civera, and A. Juan, "From Simultaneous to Streaming Machine Translation by Leveraging Streaming History," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6972–6985.

[22] W. Zhu, H. Liu, Q. Dong, J. Xu *et al.*, "Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis," Oct. 2023, *arXiv preprint arXiv:2304.04675*.

[23] O. Firat, K. Cho, and Y. Bengio, "Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, 2016, pp. 866–875.

[24] R. Aharoni, M. Johnson, and O. Firat, "Massively Multilingual Neural Machine Translation," 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)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 3874–3884.

[25] N. Team, M. R. Costa-jussà, J. Cross, O. Çelebi *et al.*, "No Language Left Behind: Scaling Human-Centered Machine Translation," Aug. 2022, *arXiv preprint arXiv:2207.04672*.

[26] B. Thompson and P. Koehn, "Vecalign: Improved Sentence Alignment in Linear Time and Space," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language*

Processing (EMNLP-IJCNLP), K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1342–1348.

[27] M. Neves, A. Jimeno Yepes, A. Siu, R. Roller *et al.*, "Findings of the WMT 2022 Biomedical Translation Shared Task: Monolingual Clinical Case Reports," in *Proceedings of the Seventh Conference on Machine Translation (WMT)*, P. Koehn, L. Barrault, O. Bojar, F. Bougares *et al.*, Eds. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 694–723.

[28] J. Iranzo-Sánchez, J. A. Silvestre-Cerdà, J. Jorge, N. Roselló *et al.*, "Europarl-ST: A Multilingual Corpus for Speech Translation of Parliamentary Debates," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 8229–8233.

[29] R. Cattoni, M. A. Di Gangi, L. Bentivogli, M. Negri *et al.*, "MuST-C: A multilingual corpus for end-to-end speech translation," *Computer Speech & Language*, vol. 66, p. 101155, Mar. 2021.

[30] J. Tiedemann, "Parallel Data, Tools and Interfaces in OPUS," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, N. Calzolari, K. Choukri, T. Declerck, M. U. Doğan *et al.*, Eds. Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 2214–2218.

[31] L. V. Lita, A. Ittycheriah, S. Roukos, and N. Kambhatla, "tRuEcasIng," in *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo, Japan: Association for Computational Linguistics, Jul. 2003, pp. 152–159.

[32] T. Kudo and J. 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*, E. Blanco and W. Lu, Eds. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 66–71.

[33] P. Gage, "A new algorithm for data compression," *The C Users Journal archive*, Feb. 1994.

[34] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Erk and N. A. Smith, Eds. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725.

[35] T. Kudo, "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 66–75.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[37] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations*, Sep. 2018.

[38] M. Freitag and Y. Al-Onaizan, "Beam Search Strategies for Neural Machine Translation," in *Proceedings of the First Workshop on Neural Machine Translation*, 2017, pp. 56–60.

[39] M. Ma, L. Huang, H. Xiong, R. Zheng *et al.*, "Stacl: Simultaneous Translation with Implicit Anticipation and Controllable Latency Using Prefix-to-Prefix Framework," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3025–3036.

[40] M. Elbayad, L. Besacier, and J. Verbeek, "Efficient Wait-k Models for Simultaneous Machine Translation," in *Interspeech 2020 - Conference of the International Speech Communication Association*, Shangai (Virtual Conf), China, Oct. 2020, pp. 1461–1465.

[41] N. Arivazhagan, C. Cherry, W. Macherey, and G. Foster, "Re-translation versus Streaming for Simultaneous Translation," in *Proceedings of the 17th International Conference on Spoken Language Translation*. Online: Association for Computational Linguistics, 2020, pp. 220–227.

[42] D. Macháček, O. Bojar, and R. Dabre, "MT Metrics Correlate with Human Ratings of Simultaneous Speech Translation," in *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*, E. Salesky, M. Federico, and M. Carpuat, Eds. Toronto, Canada (in-person and online): Association for Computational Linguistics, Jul. 2023, pp. 169–179.

[43] K. Cho and M. Esipova, "Can neural machine translation do simultaneous translation?" Jun. 2016, *arXiv preprint arXiv:1606.02012*.

[44] C. Cherry and G. Foster, "Thinking Slow about Latency Evaluation for Simultaneous Machine Translation," May 2019, *arXiv preprint arXiv:1906.00048*.

[45] J. Iranzo-Sánchez, J. Civera Saiz, and A. Juan, "Stream-level Latency Evaluation for Simultaneous Machine Translation," in *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 664–670.

[46] A. Abdelali, F. Guzman, H. Sajjad, and S. Vogel, "The AMARA Corpus: Building Parallel Language Resources for the Educational Domain," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, N. Calzolari, K. Choukri, T. Declerck, H. Loftsson *et al.*, Eds. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 1856–1862.

[47] A. Anastasopoulos, O. Bojar, J. Bremerman, R. Cattoni *et al.*, "FINDINGS OF THE IWSLT 2021 EVALUATION CAMPAIGN," in *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, M. Federico, A. Waibel, M. R. Costa-jussà, J. Niehues *et al.*, Eds. Bangkok, Thailand (online): Association for Computational Linguistics, Aug. 2021, pp. 1–29.

[48] V. Raunak, A. Sharaf, Y. Wang, H. Awadalla *et al.*, "Leveraging GPT-4 for Automatic Translation Post-Editing," in *Findings of the Association for Computational Linguistics: EMNLP 2023*. Singapore: Association for Computational Linguistics, 2023, pp. 12 009–12 024.

[49] H. Touvron, L. Martin, K. Stone, P. Albert *et al.*, "Llama 2: Open Foundation and Fine-Tuned Chat Models," Jul. 2023, *arXiv preprint arXiv:2307.09288*.

[50] C. Wang, A. Wu, J. Gu, and J. Pino, “CoVoST 2 and Massively Multilingual Speech Translation,” in *Proc. Interspeech 2021*, 2021, pp. 2247–2251.

[51] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov *et al.*, “DoRA: Weight-Decomposed Low-Rank Adaptation,” Jun. 2024, *arXiv preprint arXiv:2402.09353*.

APPENDIX A

Additional figures

A.1 Results on multilingual reverse models

Figure A.1 shows the results of the NLLB-600M model adaptation using 7 different language directions, where the reverse translation direction is added to all of the 7 directions (En–>All Reverse) or only to the Spanish direction (En–>Es Reverse). Figures A.2 and A.3 illustrate the results of the LoRA rank and extended dataset experiments on the NLLB-600M model adaptation using lowercase input, which similarly to the results with cased input case, didn't provide any improvement on the base training.

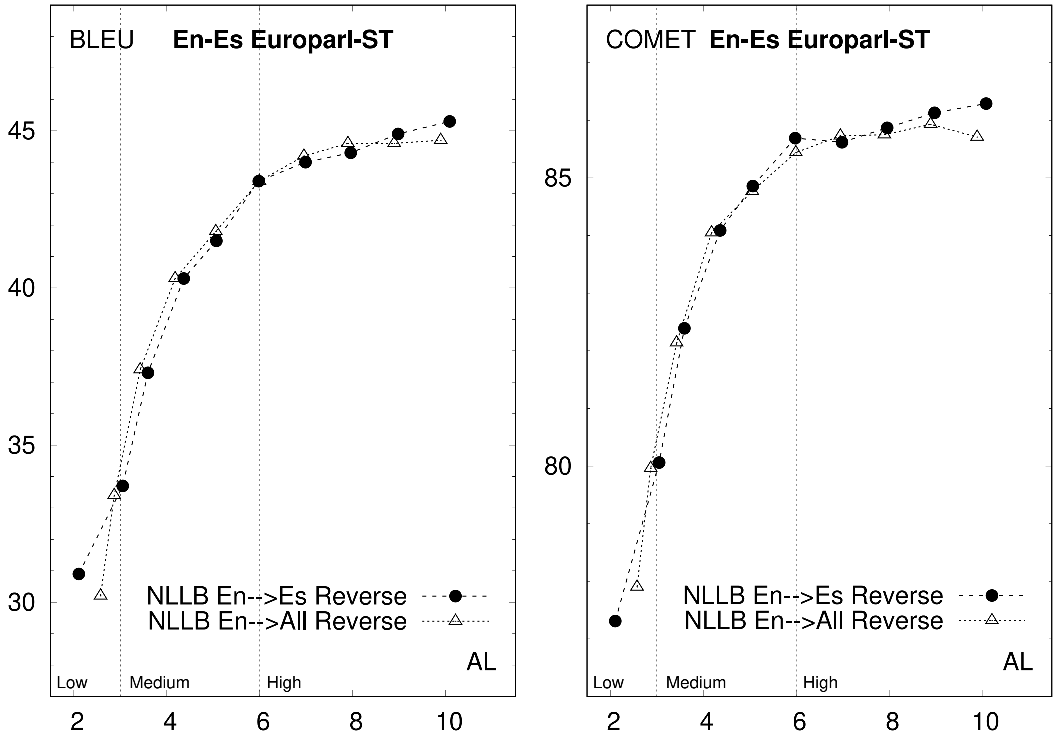

Figure A.1: BLEU vs. AL (left) and COMET vs. AL (right) at three regimes, low, medium, and high latency, on the English to Spanish Europarl-ST test sets for the NLLB-600M models trained using 7 language directions, including reverse training for each of the language directions for the reverse translation model feature (En–>All Reverse), and specifically for the *en* → *es* direction (En–>Es Reverse).

A.2 Results on LoRA rank and additional data experiments for the uncased model

Figure A.2 and Figure A.3 present the outcomes of the LoRA fine-tunings using a higher *rank* and an *extended dataset*. Once again, differences in LoRA rank and additional data fail to yield improved results, likely for similar reasons as the ones exposed in Section 4.4. Although, in this case the extended dataset LoRA fine-tuning did not hurt the model's performance as before, which suggests that the model doesn't learn the new task as easily as the previous one.

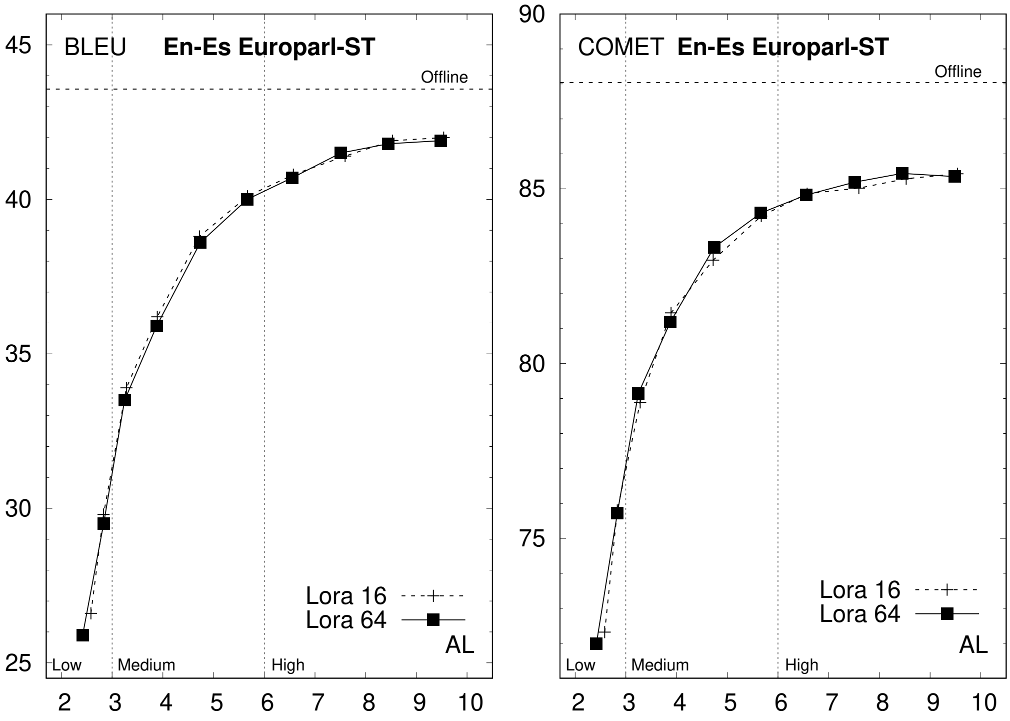

Figure A.2: BLEU vs. AL (left) and COMET vs. AL (right) at three regimes, low, medium, and high latency, on the English to Spanish Europarl-ST test sets for the NLLB-600M model trained with lowercase inputs and LoRA ranks of 16 and 64. An offline baseline is established using the rank 16 model.

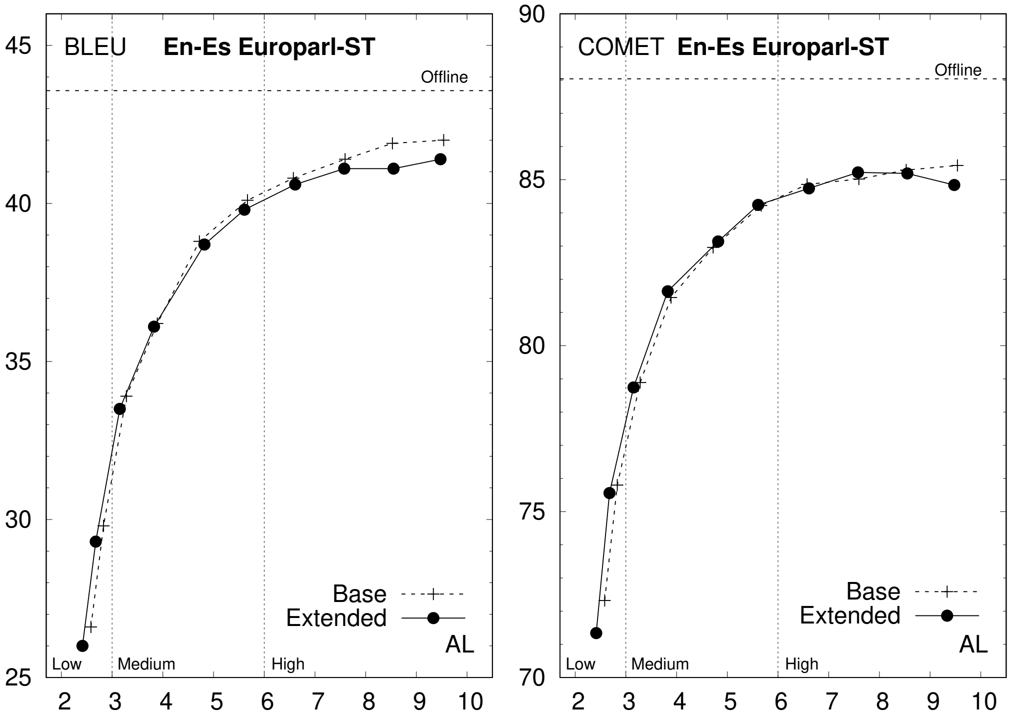

Figure A.3: BLEU vs. AL (left) and COMET vs. AL (right) at three regimes, low, medium, and high latency, on the English to Spanish Europarl-ST test sets for the NLLB-600M LoRA adapted model with lowercase input and the base training setup, and the NLLB-600M LoRA adapted model with additional training data. An offline baseline is established using the model with the base training.

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

etsinf

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles		**Alto**	**Medio**	**Bajo**	**No Procede**
1	**Fin de la pobreza.**				**X**
2	**Hambre cero.**				**X**
3	**Salud y bienestar.**		**X**		
4	**Educación de calidad.**	**X**			
5	**Igualdad de género.**				**X**
6	**Agua limpia y saneamiento.**				**X**
7	**Energía asequible y no contaminante.**				**X**
8	**Trabajo decente y crecimiento económico.**		**X**		
9	**Industria, innovación e infraestructuras.**		**X**		
10	**Reducción de las desigualdades.**			**X**	
11	**Ciudades y comunidades sostenibles.**				**X**
12	**Producción y consumo responsables.**		**X**		
13	**Acción por el clima.**				**X**
14	**Vida submarina.**				**X**
15	**Vida de ecosistemas terrestres.**				**X**
16	**Paz, justicia e instituciones sólidas.**			**X**	
17	**Alianzas para lograr objetivos.**			**X**	

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

La tecnología de traducción automática en streaming tiene el potencial de influir positivamente en la salud y el bienestar, la educación de calidad, el trabajo decente y el crecimiento económico, así como en la industria, la innovación y la infraestructura. Identificamos que el impacto de este trabajo sobre los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas es más relevante en:

- **ODS 4. Educación de Calidad**: La capacidad de traducir contenido educativo en tiempo real puede revolucionar la educación, permitiendo a estudiantes de todo el mundo acceder a recursos y clases en su idioma preferido. Esto fomenta la inclusividad y promueve una educación de calidad accesible para todos.

- **ODS 8. Trabajo Decente y Crecimiento Económico**: La tecnología de traducción en streaming puede abrir nuevos mercados y oportunidades laborales al permitir una mejor comunicación entre empresas y clientes internacionales, así como facilitar la inclusión laboral de personas que hablan diferentes idiomas o la formación de equipos de trabajo internacionales.

- **ODS 3. Salud y Bienestar:** La traducción automática en streaming puede permitir una comunicación fluida entre personas independientemente de los idiomas que hablen, lo que podría facilitar la vida de personas extranjeras o emigrantes, eliminando la barrera del idioma para su inclusión en las sociedades que las acogen.
  Además, permitiría mejorar significativamente la accesibilidad a información médica, permitiendo a los profesionales de la salud y a los pacientes superar barreras lingüísticas cuando las hubiera. Esto podría facilitar una mejor comunicación, y mejorar así la precisión del diagnóstico y el tratamiento.

- **ODS 9. Industria, Innovación e Infraestructura:** La innovación en traducción automática en tiempo real contribuye al desarrollo de infraestructuras tecnológicas avanzadas, como pueden ser plataformas de traducción en tiempo real integradas en sistemas de comunicación, o al desarrollo de nuevas herramientas y aplicaciones basadas en IA para mejorar la traducción automática. Esto impulsa la competitividad industrial y promueve la innovación continua en el campo de la inteligencia artificial y el procesamiento del lenguaje natural, donde el desarrollo de sistemas de IA capaces de funcionar a tiempo real y la investigación de las posibilidades de los LLMs son temas de gran relevancia en la actualidad.

- **ODS 12. Producción y consumo responsables:** El entrenamiento de modelos de aprendizaje automático es un proceso extremadamente costoso, tanto en términos materiales como en energéticos. Se estima que los modelos más grandes de la actualidad pueden llegar a utilizar 25.000 GPUs de manera continua durante 90-100 días, por ello es crucial saber cómo aprovechar estos enormes modelos para evitar volver a entrenar otros.

Adicionalmente, de manera secundaria podemos identificar que también se cumplen los ODS **Paz, Justicia e Instituciones Sólidas (ODS 16)**, **Alianzas para Lograr los Objetivos (ODS 17)** y **Reducción de las Desigualdades (ODS 10)**.

En el caso de las **ODS 16** y **ODS 17**, la traducción automática en streaming puede facilitar el diálogo y la cooperación internacional, promoviendo la paz y el entendimiento entre diferentes culturas y naciones. La capacidad de superar barreras lingüísticas puede fomentar alianzas globales más efectivas y la implementación de proyectos y políticas que requieren la cooperación de múltiples partes interesadas a nivel global.

En cuanto a la **ODS 10**, la traducción automática en streaming puede reducir las desigualdades al proporcionar acceso equitativo a información y servicios para personas que hablan diferentes idiomas. Esto es especialmente relevante en regiones con gran diversidad lingüística y en comunidades marginadas, y permitiría la realización de proyectos de inclusión social y económica basados en la accesibilidad lingüística.

En conclusión, la traducción automática en streaming, y particularmente enfocada a través de la adaptación de LLMs, tiene un impacto significativo en múltiples ODS, promoviendo un mundo más conectado, accesible y energéticamente eficiente. La implementación y el desarrollo continuo de esta tecnología no solo mejoran la comunicación global, sino que también contribuyen al progreso social, económico, tecnológico y educativo a nivel mundial. Para maximizar estos beneficios, es crucial seguir invirtiendo en innovación tecnológica y fomentar colaboraciones internacionales que apoyen estos objetivos.