**BRNO UNIVERSITY OF TECHNOLOGY**
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**FACULTY OF INFORMATION TECHNOLOGY**
**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**
**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**

# ALIGNING PRE-TRAINED MODELS FOR SPOKEN LANGUAGE TRANSLATION

**SLADĚNÍ PŘEDTRÉNOVANÝCH MODELŮ PRO PŘEKLAD MLUVENÉHO JAZYKA**

**MASTER'S THESIS**
**DIPLOMOVÁ PRÁCE**

**AUTHOR** **Bc. ŠIMON SEDLÁČEK**
**AUTOR PRÁCE**

**SUPERVISOR** **SANTOSH KESIRAJU, Ph.D.**
**VEDOUCÍ PRÁCE**

**BRNO 2024**

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

# Master's Thesis Assignment

157031

| | |
|---|---|
| Institut: | Department of Computer Graphics and Multimedia (DCGM) |
| Student: | **Sedláček Šimon, Bc.** |
| Programme: | Information Technology and Artificial Intelligence |
| Specialization: | Machine Learning |
| Title: | **Aligning pre-trained models for spoken language translation** |
| Category: | Speech and Natural Language Processing |
| Academic year: | 2023/24 |

Assignment:

1. Get familiar with sequence-to-sequence models for spoken language translation (SLT) task.
2. Using existing toolkits (e.g.: Huggingface transformers) implement a baseline SLT system on a standard dataset (e.g.: HOW2: English -> Portuguese)
3. Implement the Q-transformer baseline for aligning pre-trained speech encoder (ASR) and a language / MT model.
4. Implement the alignment model as sequence-to-sequence in the same framework as point 3.
5. For a fixed choice of pre-trained models, experiment with the developed alignment models (points 3, 4) for SLT on two standard datasets and analyse the results.
6. For a fixed choice of alignment model, experiment with various choices of pre-trained models (e.g.: XLS-R, Whisper, LLaMA, GPT-2) for SLT, and summarize your findings.

Literature:

- Yu et al, "Connecting Speech Encoder and Large Language Model for ASR". arXiv:2309.13963, 2023.
- Li et al "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models". ICML, 2023.
- Kesiraju et al "Strategies for improving low resource speech to text translation relying on pre-trained ASR models". Interspeech 2023.

Requirements for the semestral defence:
Points 1-3

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

| | |
|---|---|
| Supervisor: | **Kesiraju Santosh, Ph.D.** |
| Head of Department: | Černocký Jan, prof. Dr. Ing. |
| Beginning of work: | 1.11.2023 |
| Submission deadline: | 17.5.2024 |
| Approval date: | 14.5.2024 |

## Abstract

In this work, we investigate a novel approach to end-to-end speech translation (ST) by leveraging pre-trained models for automatic speech recognition (ASR) and machine translation (MT) and connecting them with a small connector module (Q-Former, STE). The connector bridges the gap between the speech and text modalities, transforming the ASR encoder embeddings into the latent representation space of the MT encoder. During training, the foundation ASR and MT models are frozen, and only the connector parameters are tuned, optimizing for the ST objective. We train and evaluate our models on the How2 English to Portuguese ST dataset. In our experiments, aligned systems outperform our cascade ST baseline while utilizing the same foundation models. Additionally, while keeping the size of the connector module constant and small in comparison (10M parameters), increasing the size and capability of the ASR encoder and MT decoder universally improves translation results. We find that the connectors can also serve as domain adapters for the foundation models, significantly improving translation performance in the aligned ST setting, compared even to the base MT scenario. Lastly, we propose a pre-training procedure for the connector, with the potential for reducing the amount of ST data required for training similar aligned systems.

## Abstrakt

Tato práce zkoumá nový end-to-end přístup k překladu mluveného jazyka (ST) využívající předtrénovaných modelů pro přepis řeči (ASR) a strojový překlad (MT), propojené malým spojovacím modulem (Q-Former, STE). Ten má za úkol překlenout mezeru mezi modalitami řeči a textu mapováním embedding reprezentací ASR enkodéru do latentního prostoru reprezentací MT modelu. Během trénování jsou zvolené ASR a MT model zmrazeny, laděny jsou pouze parametry spojovacího modulu. Trénování a evaluace jsou prováděny na datasetu How2, obsahujícím ST data z Angličtiny do Portugalštiny. V našich experimentech zjišťujeme, že většina sladěných systémů překonává referenční kaskádový ST systém, přičemž využívají stejné základní modely. Navíc, při zachování konstantní a ve srovnání malé (10M parametrů) velikosti spojovacího modulu, větší a silnější ASR a MT modely univerzálně zlepšují výsledky překladu. Zjišťujeme, že spojovací moduly mohou také sloužit jako doménové adaptéry pro zvolené základní systémy, kdy významně zlepšují výsledky překladu ve sladěném ST prostředí, a to i oproti holému MT výkonu daného MT modelu. Nakonec navrhujeme proceduru pro předtrénování spojovacího modulu s potenciálem snížit množství ST dat potřebných pro trénink obdobných sladěných systémů.

## Keywords

spoken language translation, speech translation, model alignment, automatic speech recognition, machine translation, transfer learning, transformers, Q-Former, domain adaptation

## Klíčová slova

překlad mluveného jazyka, překlad řeči, sladění modelů, automatické rozpoznávání řeči, strojový překlad, transfer learning, transformery, Q-Former, doménová adaptace

## Reference

SEDLÁČEK, Šimon. *Aligning pre-trained models for spoken language translation.* Brno, 2024. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Santosh Kesiraju, Ph.D.

# Rozšířený abstrakt

*Překlad mluveného jazyka* (SLT, ST) je proces přepisu audio nahrávky obsahující řeč v určitém zdrojovém jazyce (např. Angličtina) do podoby textového překladu v určitém cílovém jazyce (např. Portugalština). Tento problém je strojově typicky řešen dvěma základními způsoby. Prvním je vytvoření tzv. *kaskádového* kompozitního systému s využitím již předtrénovaných modelů pro *automatický přepis řeči* (ASR) a *strojový překlad* (MT). Tyto modely jsou zapojeny za sebe do série, přičemž je nejdříve vygenerován přepis zdrojové nahrávky a poté je tento textový přepis přeložen MT modelem. Druhým způsobem je natrénování *end-to-end* enkodér-dekodér modelu, založeném například na architektuře *transformer* [46].

Oba přístupy však mají své nevýhody. Kaskádové systémy mohou trpět fenoménem akumulace chyb právě z toho důvodu, že nejdříve nahrávku přepíšou do textu, a ta je až následně přeložena. Vzniká ze takto prostor pro zesílení vlivu odchylek způsobených například rozdíly mezi slovníky, které dané ASR a MT modely používají, nebo doménami dat, na kterých byly systémy trénovány. Dále mají tyto systémy větší latenci, protože k vygenerování překladu je nutné projít dvěma fázemi dekódování.

Na druhé straně, vytváření a trénování end-to-end systémů od začátku není jednoduchý proces, protože daný systém se musí naučit mapovat komplexní a proměnlivé vzory ve vstupních řečových nahrávkách na sekvenci diskrétních symbolů (tokenů) v daném cílovém jazyce. Navíc při překladu je nutné, aby se model naučil reorganizovat a měnit gramatické struktury zdrojového jazyka tak, aby na výstupu vyprodukoval validní překlad v cílovém jazyce.

Je proto běžné použít různé trénovací techniky, které výsledky trénovacího procesu urychlují a zlepšují, jako například inicializace enkodéru váhami ASR modelu pro zdrojový jazyk, dodatečná supervize modelu na výstupu enkodéru, a podobně.

S aktuálním rozvojem v oblasti *velkých jazykových modelů* (LLMs) se objevilo několik přístupů k využití těchto modelů jako bází pro generování textu a usuzování při řešení náročných cross-modálních úloh jako například anotace obrázků. Jeden z těchto přístupů s názvem BLIP-2 [25] uvedl jednoduchý transformer spojovací modul zvaný *Q-Former*. Tento modul je zodpovědný za mapování abstraktních reprezentací extrahovaných z daného zdrojového obrázku zmrazeným předtrénovaným obrázkovým enkodérem do prostoru word-embedding reprezentací daného zmrazeného jazykového modelu, kde slouží jako soft-prompt. Daný jazykový model pak vygeneruje anotaci ke zvolenému obrázku.

Podobné přístupy se následně začaly objevovat i v doménách automatického přepisu řeči a jiných řečově orientovaných úlohách [54, 50, 6]. V této práci se podobný přístup snažíme aplikovat na úlohu překladu mluveného jazyka.

Použití malé spojovací sítě jako Q-Former pro překlenutí mezery mezi řečovou a textovou modalitou zvolených zmrazených předtrénovaných ASR enkodérů a MT modelů je atraktivní z několika důvodů. Za prvé, oproti trénování klasického end-to-end systému pro překlad řeči, tento propojovací modul může být mnohem menší s pouze zlomkem parametrů. Zvolené základní ASR a MT modely mohou zůstat zmrazeny a optimalizovány jsou pouze váhy konektoru. Další výhodou je, že daný konektor může v porovnání se zvolenými základními modely zůstat velice malý, protože stačí, aby se naučil správné mapování z jednoho prostoru reprezentací do druhého. Toto následně vede ke kratším trénovacím časům a obecně snížení výpočetních nákladů na trénování takto sladěného systému oproti end-to-end systému stejné velikosti.

V provedených experimentech používáme dvě základní architektury pro sladění ASR a MT modelů – ECD architektura, kde výstup ASR enkodéru je konektorem mapován na cross-attention vstup daného MT dekodéru, a ECED, kdy konektor provádí stejné mapování do prostoru vstupních textových reprezentací zvoleného MT enkodéru. Experimentujeme se dvěma typy spojovacích model: Q-Former a STE (jednoduchý transformer enkodér s konvolučním subsampling frontendem).

Z výsledků našich experimentů usuzujeme, že metoda slaďování ASR a MT modelů pro ST je vhodný obecný framework pro využití off-the-shelf předtrénovaných modelů pro řešení této úlohy. Zjišťujeme, že náš STE konektor poráží výkon Q-Former, primárně díky flexibilitě kvůli schopnosti STE konektoru mapovat variabilně dlouhé vstupní sekvence na variabilně dlouhý výstup, přičemž tuto schopnost Q-Former nemá.

Dále zjišťujeme, že použití větších a silnějších ASR a MT modelů vede univerzálně ke zlepšení výsledků překladu, a to i v případě, že velikost spojovacího modul zůstává konstantní a malá v porovnání se sladěnými modely. Konektory také můžou sloužit jako doménové adaptéry pro zvolené základní modely. Pro jeden z MT modelů, který byl trénován mimo doménu `How2` datasetu, zlepšuje sladění v ST (řeč-text) architektuře u nejlepšího systému překlad o více než 9 BLEU bodů oproti základnímu případu vyhodnocení strojového překladu na `How2` (text-text).

V posledním setu experimentů prozkoumáváme způsoby předtrénování spojovacího modulu pouze s využitím ASR dat a zjišťujeme, že více 'vanilla' přístupy založené na *knowledge-distillation* mezi prostory reprezentací na výstupu konektoru a MT enkodéru nepřináší zdaleka takové výsledky a potenciál jako více robustní end-to-end metoda, kterou navrhujeme. Tato end-to-end metoda zahrnuje přetrénování zvoleného MT dekodéru na identitu ve zdrojovém jazyce, aby bylo následně sladěný systém možno předtrénovat pouze s využitím ASR dat. Po předtrénování lze pak dosáhnout lepších překladových výsledků při dotrénování na menších objemech ST dat. Přestože tato předtrénovací metoda prozatím nedosahuje nijak zázračných výsledků, domníváme se, že má potenciál pro budoucí experimenty, úpravy a zlepšení.

# Aligning pre-trained models for spoken language translation

## Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Mr. Santosh Kesiraju, Ph.D. I have listed all the literary sources, publications, and other sources, that were used during the preparation of this thesis.

. . . . . . . . . . . . . . . . . . . . . . .
Šimon Sedláček
May 16, 2024

## Acknowledgements

I would hereby like to sincerely thank my supervisor, Santosh Kesiraju, Ph.D. for his support, advice, patience, and most importantly – inspiring discussions while working on this thesis. Also, I would like to thank my friend, Ing. Alexander Polok for giving me access to his excellent repository for my experiments, and always being there to help.

# Contents

# List of Figures

# Chapter 1

# Introduction

*Spoken language translation* (SLT) is a task of *transducing* a speech utterance in a given source language into its corresponding text translation in a given target language. To solve this task, it is typical to either build a composite *cascade* speech translation system from pre-trained *automatic speech recognition* (ASR) and *machine translation* (MT) systems, or to train an encoder-decoder-like *end-to-end* deep learning model, based for example on the *transformer* [46] architecture.

However, both approaches have their caveats. Cascade systems can sometimes suffer from error accumulation due to first transcribing the input utterance and only subsequently translating it.

On the other hand, training end-to-end speech translation systems from scratch is not a simple task, as the system has to learn to map complex and variable speech audio patterns into sequences of discreet symbols (tokens) of the target language. Additionally – unlike in the ASR scenario – there is no consistent monotonic alignment between the input speech audio and the output token sequences. The model has to therefore correctly learn to rearrange the grammatical structure of the text encoded in the input speech utterance to produce a valid translation in the target language.

It is therefore common to utilize several training techniques including – but not limited to – transfer learning and additional model supervision, to obtain better results.

With the recent advancements in the domain of *large language models* (LLMs), approaches have emerged that try to leverage these powerful pre-trained models as language-modeling bases for different cross-modal tasks, such as image-captioning. One such approach is called BLIP-2 [25], introducing a simple connector transformer network called the *Q-Former*. In BLIP-2, the Q-Former is responsible for mapping abstract vision features extracted from the input image by a *frozen image encoder* into the word embedding space of a *frozen large language model*, prompting it to generate an annotation to the input image.

Similar approaches have subsequently been adopted and applied to the domain of automatic speech recognition and other speech-language oriented tasks [54, 50, 6]. In this work, we attempt to do the same with a focus on spoken language translation.

Utilizing a Q-Former-like connector network to bridge the gap between the speech and text modalities of a frozen speech encoder and a frozen machine translation/language model is attractive for a few reasons. First, as opposed to constructing a conventional encoder-decoder speech translation model and training it on large amounts of data, the connector network can be much smaller with much fewer parameters. The pre-trained speech encoder and the chosen language decoder can remain frozen and only the connector network is trained. Even if the chosen pre-trained models are very powerful, the bridge network only has to learn the mapping between their hidden representation spaces, resulting in shorter training times and fewer computational resources required, in contrast to conventional end-to-end models of a similar size.

Chapter 2 first provides an introduction to the topic of transformer-based deep neural networks, describing their architectures and typical use cases for this thesis. Chapter 3 then provides a brief introduction to the topic of spoken language translation (or speech translation).

Chapter 4 further introduces the concept of aligning pre-trained models to solve difficult cross-modal tasks and problems. The chapter gives focus to the Q-former connector model, and proposes an alternative STE connector. Additionally, the two main model alignment frameworks (ECD, ECED) used in the experiments conducted in this work are introduced.

Chapter 5 discusses most of the experiments and findings with regard to the alignment architectures and connector modules, discussing the pros and cons of each, and evaluating the model alignment approach to solving speech translation as a whole.

Finally, Chapter 6 describes two connector network pre-training approaches, focusing on reducing the need for speech translation data for the alignment process, and poses further questions and ideas for potential future work in this domain.

## Findings and contributions

- We rule that our Subsampler-Transformer Encoder (STE) connector is superior to the Q-Former for the ST alignment scenario both performance and flexibility-wise. We find that determining an appropriate number of Q-Former queries is difficult in comparison to leveraging the variable-length sequence mapping ability of the STE connector.

- Scaling up the aligned ASR and MT models leads to universally better speech translation results, while the size of the connector network can remain constant, and relatively small.

- The connector networks can serve as domain adapters, significantly improving translation performance for scenarios, where the aligned MT models are out-of-domain.

- We find that pre-training the connector network using vanilla knowledge-distillation approaches aimed at matching the connector and MT encoder output embedding spaces is not as useful when compared to more principled end-to-end approaches, following the cross-entropy objective of the ASR task. For this purpose, we devise a pre-training procedure that allows us to pre-train the connector using only ASR data.

# Chapter 2

# Transformer-based neural networks

In this chapter, the topic of deep neural networks based on the *transformer* [46] architecture is discussed.

Section 2.1 describes and explains the core concepts that build up the transformer, focusing on the ones that differentiate it from other neural network architectures. Section 2.2 then gives a brief overview of how the transformer and its many architecture variants are used for natural language processing and language modeling in general, as it is relevant to the topic of speech translation. Section 2.3 does the similar, only for automatic speech recognition. Lastly, Section 2.4 briefly touches on the concept of transfer-learning, which is key in the context of this work.

## 2.1 The Transformer

The *transformer* is a deep learning architecture first introduced in [46], which could potentially be regarded as one of the overall most critical and influential advancements in the deep learning domain in recent years. It has achieved state-of-the-art performance in most sequential modeling tasks such as natural language generation [31], summarization [10, 40], automatic speech recognition [38], machine translation [40, 19, 52, 29, 44], and many more. Though originally developed specifically for *sequence-to-sequence* modeling tasks, the transformer can be used to solve a wide range of problems, where it is important to model either temporal or structural relationships in the input data (e.g. even for computer vision [12]).

What makes the transformer stand out in comparison to other sequence modeling approaches such as recurrent neural networks (RNN), is mainly its effectiveness in terms of computational cost and parallelism, leading to significantly better scaling, training, and inference times. This is due to the fact that the transformer disposes of all recurrent connections in the model architecture (minimizing the amount of sequential computation), and uses the so-called *attention mechanism* to bidirectionally model temporal relationships within the input sequences instead. The attention mechanism along with its innovations as presented in [46] is described in the next Section 2.1.1.

### 2.1.1 Attention mechanism

At the core of the transformer architecture is the *attention mechanism*. This concept was first introduced in [3], where it was applied in an RNN-based encoder-decoder model for machine translation. In [46], the attention mechanism was expanded upon, resulting in the introduction of what the authors called *Scaled Dot-Product Attention*.

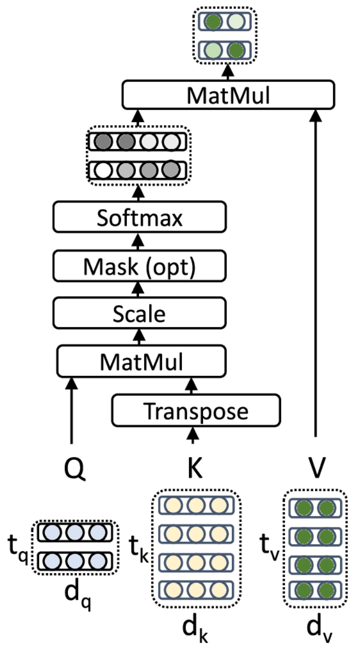


Figure 2.1: Scaled dot-product attention. Typically, $t_k = t_v$ and $d_q = d_k$, however, in the self-attention case used in the transformer model the shapes of all three matrices are the same. Obtained from [11].

The attention mechanism in general can be understood as a function of three input vectors: a *query*, a *key* and a *value*. The names correspond to the respective roles of the vectors in the computation of the attention output. In a real scenario, the output is computed over a number of input query, key and value vectors, organized into matrices: $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, denoting the query, key and value matrix, respectively. The scaled dot-product attention used in transformer models would then be computed as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \tag{2.1}$$

where $d_k$ is the dimensionality of the keys.

Intuitively, the output of the attention mechanism can be thought of as the values scaled and selected by the attention map obtained by multiplying $\mathbf{Q}$ and $\mathbf{K}$. This essentially corresponds to computing a similarity (or a correspondence) measure for each query-key pair and then converting it to probabilities or weights for $V$ via the softmax function. a computation diagram of the scaled dot-product attention can be seen in Figure 2.1.

Another attention mechanism innovation introduced in [46] is the so-called *Multi-Head Attention* depicted in Figure 2.2. Multi-head attention splits the processing of the queries, keys and values into $H$ paths, where $H$ is the number of *attention heads* used. Each head $h$ has its own $\mathbf{W}_h^Q$ (query), $\mathbf{W}_h^K$ (key) and $\mathbf{W}_h^V$ (value) projection matrices, which are used to project the attention inputs $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ into different sub-spaces for each particular head $h$. These projections are, of course, learnable.

Splitting the attention mechanism into multiple heads alleviates some issues with averaging attention-weighted positions when operating in a higher-dimensional attention map space [46]. Additionally, it allows the model to attend to the input information in different sub-spaces at different time steps at once:

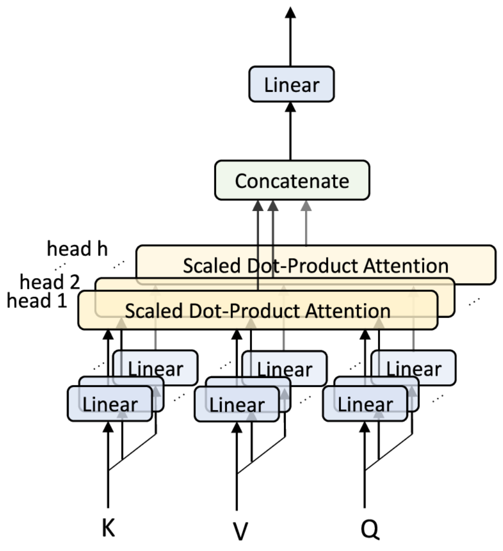


Figure 2.2: Multi-head attention block. $V$, $K$, $Q$ denote the value, key and query attention inputs, respectively. Diagram obtained from [11].

$$\mathbf{head}_h = \text{Attention}(\mathbf{Q}\mathbf{W}_h^Q, \mathbf{K}\mathbf{W}_h^K, \mathbf{V}\mathbf{W}_h^V), \tag{2.2}$$

where the projection dimension of $\mathbf{W}_h^Q$, $\mathbf{W}_h^K$, $\mathbf{W}_h^V$ typically corresponds to $d_q = d_k = d_v = d_{\text{model}}/H_{\text{heads}}$.

After computing the attention outputs for each head, these outputs are then concatenated and finally projected back into the original model dimension $d_{\text{model}}$ with the final output linear layer $\mathbf{W}^O$:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\mathbf{head}_1, \ldots, \mathbf{head}_H)\mathbf{W}^O. \tag{2.3}$$

### 2.1.2 Word embeddings and positional encoding

When using the transformer in a language modeling scenario, the input text string has to first be converted into a form suitable for processing with the transformer encoder or decoder. The input sentence is first split into discrete symbols, called *tokens* using a tokenizer. The tokenization process is typically performed on a sub-word level, where the tokenizer is trained to learn to optimally split different words to separate sub-word units, creating a compromise between efficiency and being able to represent even previously unseen words. Among the commonly used sub-word tokenization algorithms are the *Byte-Pair-Encoding* (BPE) [15], and *Unigram* [23] models. Other word-level or character-level tokenization algorithms can alternatively be used, depending on the task.

After tokenization, the input text now takes the form of a list of integer token IDs, based on the vocabulary of the tokenizer. These integer IDs are then converted into a one-hot-encoding form, where the number of elements in each of the vectors is determined by the size of the vocabulary. Let us denote this one-hot-encoded sequence of tokens as $\mathbf{X}$.

Before being processed by the transformer, each one-hot-encoded token $\mathbf{x}_i \in \mathbf{X}$ ($i$ is the position of the token in the sequence) has to first be *embedded* into the hidden representation space of the transformer. This is done using a trainable *embedding* layer, which for each possible one-hot-encoded position of the source vocabulary keeps a corresponding embedding vector, which represents the particular token $\mathbf{x}_i$ in the transformer embedding space. These word embedding vectors are optimized during training. The embedding space also has a much lower dimensionality than the size of the vocabulary (for typical transformer models ranging from 256 to 1024). We denote the embedding dimensionality $d_{model}$ (this dimensionality is also often referred to as the hidden representation size of the transformer).

Embedding each input token $\mathbf{x}_i$, we obtain a its corresponding word embedding representation $\mathbf{e}_i$:

$$\mathbf{e}_i = \mathrm{EMB}(\mathbf{x}_i). \tag{2.4}$$

Afterward, because the attention mechanism is actually position-agnostic, a positional embedding is added to each of the embedded tokens to preserve the information about the position of the token in the input sequence:

$$\mathbf{e}'_i = \mathrm{pos}(i) + \mathbf{e}_i. \tag{2.5}$$

In [46], a simple sinusoidal positional encoding algorithm is proposed, though often transformer models use different, even trainable positional encoding modules [38]. The embeddings are subsequently passed to the actual transformer encoder.

### 2.1.3 Encoder

The role of the encoder is then to project and encode the input embedding sequence $\mathbf{E}$ into a series of hidden states $\mathbf{E}'$. The length of the input sequence is the same as the length of the output sequence – each embedding vector is gradually enriched with additional contextual abstract information.

The encoder itself consists of identical $N_{\text{enc}}$ layers or *blocks* (refer to Figure 2.3 for a visual reference). In each encoder block $n$, the inputs $\mathbf{E}_n$ first pass through a *self-attention* layer. The self-attention layer is simply a multi-head attention layer for which the queries, keys, and values are one and the same – the inputs attend to themselves. Additionally, there is also a residual connection, which bypasses the self-attention and adds the original inputs to the attention output. Layer normalization is then applied to the result[1]:

$$\mathbf{E}'_n = \mathrm{LayerNorm}\big(\text{Self-Attention}_n(\mathbf{E}_n) + \mathbf{E}_n\big). \tag{2.6}$$

Then, the attention output is then passed to an intermediate feed-forward layer, again employing the same type of residual connection and layer normalization afterward. This gives us the final output of the encoder block $\mathbf{E}''_n$:

$$\mathbf{E}''_n = \mathrm{LayerNorm}\big(\text{Feed-Forward}_n(\mathbf{E}'_n) + \mathbf{E}'_n\big). \tag{2.7}$$

It should be noted that the output of each block has exactly the same tensor shape as the input.

### 2.1.4 Decoder

The decoder has a similar structure to the encoder – it once again consists of several decoder blocks (layers), and the input of the decoder has to be first embedded via an embedding layer, after which positional encoding information is added to the embeddings.

[1]Contrary to the original paper, current transformer architectures mostly use what is called the *pre-norm* architecture [51], where layer normalization is applied before the attention and feed-forward operations. This has been shown to improve gradient behavior, making transformer training more stable.

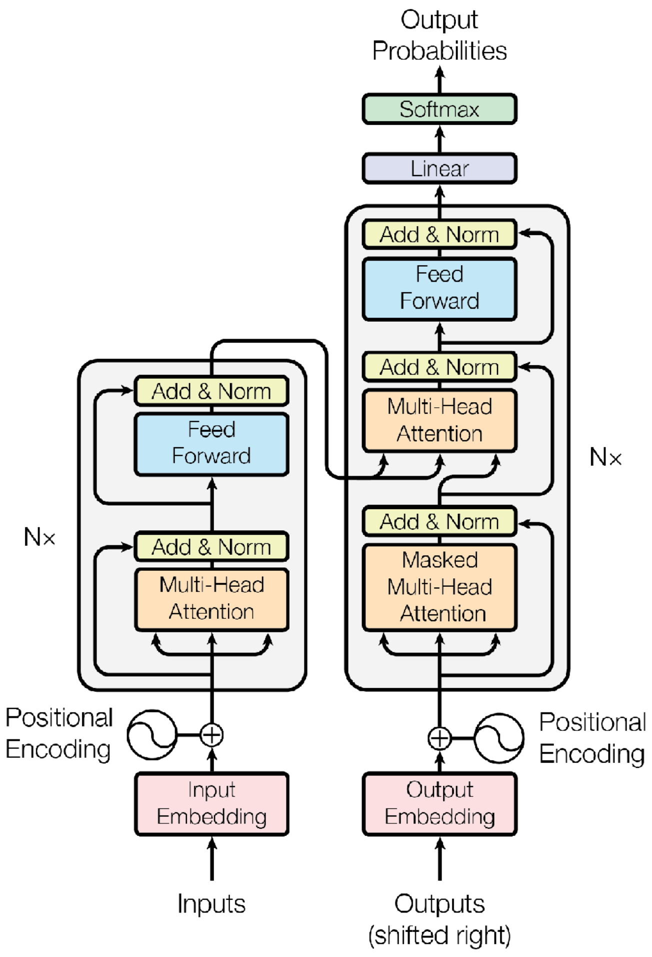


Figure 2.3: Diagram of the transformer model obtained from [46]. The transformer encoder is depicted on the left, the decoder on the right. The encoder feeds its output hidden states to each decoder block via cross-attention connections. The output of the last decoder block is passed to the unembedding linear layer followed by softmax, producing an output probability distribution over all the next likely tokens in the sequence.

In each decoder block $n$, the input embeddings $\mathbf{D}_n$[2] are first passed once again through a self-attention layer, followed by adding the residual connection and applying layer normalization:

$$\mathbf{D}'_n = \text{LayerNorm}\big(\text{Self-Attention}_n(\mathbf{D}_n) + \mathbf{D}_n\big). \tag{2.8}$$

Then, a second multi-head attention layer often referred to as *encoder attention* or *cross-attention* is utilized to allow the decoder to condition its outputs on the output hidden states produced by the encoder. For this purpose, the decoder hidden states $\mathbf{D}'_n$ are used as the *queries* and the encoder output hidden states $\mathbf{E}_{\text{enc.}}$ as the *keys* and the *values*:

$$\mathbf{D}''_n = \text{LayerNorm}\big(\text{Cross-Attention}_n(\mathbf{D}'_n, \mathbf{E}_{\text{enc.}}, \mathbf{E}_{\text{enc.}}) + \mathbf{D}'_n\big). \tag{2.9}$$

This cross-attention connection runs from the encoder output to each of the decoder blocks, further influencing the hidden states in each decoder layer. After the cross-attention, the decoder outputs are once again processed by a fully-connected layer followed by layer norm, producing the final output of the decoder block $\mathbf{D}'''_n$:

$$\mathbf{D}'''_n = \text{LayerNorm}\big(\text{Feed-Forward}_n(\mathbf{D}''_n) + \mathbf{D}''_n\big). \tag{2.10}$$

In contrast to the encoder, which operates bidirectionally on the input sequence, the decoder operates *auto-regressively*. The decoder is first prompted with a *start-of-sentence* token. The token is embedded and passed through all the decoder layers. It interacts with the encoder outputs via cross-attention and finally, at the end of the last decoder block, it is passed through a linear layer, known as the *unembedding* layer, which projects the output embedding into the dimension of the model vocabulary. This operation produces raw *logit* scores for each word unit in the output vocabulary, which can be converted into probabilities using the softmax function, giving us the probability distribution of the next likely words in the resulting sequence. The next word can then be obtained by simply taking the word, whose probability value is the highest.

The auto-regressive behavior of the decoder stems from the fact, that this new output word is then used a new input of the decoder along with all the other inputs from previous time-steps. Then, the decoding process continues, until the model produces an *end-of-sentence* token (or the generation is cut off by crossing a certain token count limit)[3].

Because this auto-regressive process is strictly sequential and time-consuming, during training, the whole desired output sequence is used as the decoder input. Instead of generating each token at each time step one by one, a triangular causal *attention mask* is used in the self and cross-attention layers. This mask is used to essentially simulate performing all of the auto-regressive generation steps in one forward pass only, as the decoder simultaneously predicts the next token for each position in the input sequence. The causal mask is used, so when predicting the next token for a position $i$ in the input sequence, none of the future tokens at any position $k > i$ influence this prediction. During inference, these future tokens would only become available at future time steps of the auto-regressive generation process.

The employment of causal attention masking is one of the things that make the transformer much more efficient during training in contrast to other auto-regressive models, based for example on recurrent neural networks.

[2]Here, the letter ‘D‘ is used to better differentiate the decoder embeddings from the encoder hidden states.

[3]An illustration of the generation process (and the whole transformer model in general) can be found at https://jalammar.github.io/illustrated-transformer/

## 2.2 Transformer architectures in language modeling

Language modeling is perhaps the domain where the transformer is arguably used most widely. Upon its introduction in [46], its capability is demonstrated on a machine translation task, where the transformer was used in its default **encoder-decoder** configuration. Intuitively, this is natural for tasks such as translation [40, 52, 29, 44, 19] as the model is trying to learn a sequence-to-sequence mapping (also referred to as *sequence transduction*) from one language to another – without any monotony guarantees about the length or symbol position relationships between the source input and the translation, for example.

Apart from machine translation, encoder-decoder transformer architectures can also be used for other general language modeling tasks, as is demonstrated by models such as BART [24], or different models from the T5 model family [40, 52, 29]. The T5 family is especially interesting due to its multi-task training approach, resulting in models able to handle multiple general language tasks specified by prepending an instruction in natural language to the encoder input.

However, not all tasks require the full encoder-decoder architecture for solving the given problem. In language modeling, it is also common to only use the transformer encoder or, alternatively, the decoder, depending on the task at hand.

**Encoder-only** transformer models do away with the decoder completely and leverage the ability of the transformer decoder to process the input bidirectionally. Encoder-only language models usually rely on two training stages – *pre-training* and subsequent *fine-tuning*. In the pre-training stage, the model collects information about the source language, learning abstract language representations in the process. Then, after pre-training, new layers are typically added on top of the encoder, and subsequently fine-tuned to interpret and transform the last encoder-hidden states to solve the downstream task, e.g. text classification or named entity recognition.

Perhaps the best-known model of this type is BERT [10]. BERT models are pre-trained using *masked language modeling* (MLM), where some of the tokens in the input sequence are masked and the model is tasked with recovering them, which is combined with the *next sentence prediction* objective, where the model receives two input sentences and is tasked to predict which of them precedes the other.

On the other hand, **decoder-only** transformer models consist of stacked decoder blocks with the cross-attention layers removed. The decoder-only architecture was first introduced in [26], demonstrating its text generation capability to be superior to encoder-decoder architectures, as the models were able to scale better in terms of attending to very long sequences and contexts. Decoder-only models are typically trained for the causal language modeling task and can further be fine-tuned for more specific applications (with some overlap with the encoder-only models).

Decoder-only models have also demonstrated good scaling properties [39, 4, 31, 8, 45], earning a spotlight at the forefront of generative language modeling and understanding in virtually all of their domains and flavors, known as the *Large Language Models* (LLMs).

## 2.3 Transformers for automatic speech recognition

Subsequently, transformers began to be used for processing other modalities apart from text. Notably, in [11], a general method of adapting the transformer to process speech input data was proposed. Transformer architectures for end-to-end (E2E) automatic speech recognition (ASR) began quickly outperforming previous state-of-the-art RNN-based and

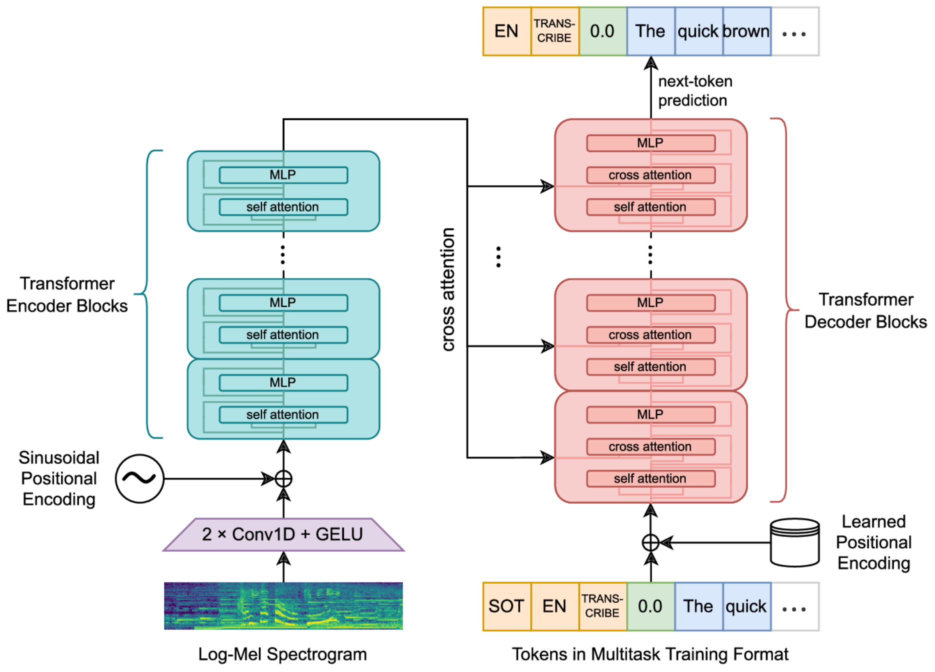

EN
TRANS-CRIBE
0.0
The
quick
brown
...
next-token prediction
MLP
cross attention
self attention
Transformer Encoder Blocks
Transformer Decoder Blocks
cross attention
Sinusoidal Positional Encoding
2 × Conv1D + GELU
Learned Positional Encoding
SOT
Log-Mel Spectrogram
Tokens in Multitask Training Format

Figure 2.4: Diagram of the OpenAI Whisper model [38].

CNN-based models, given enough data. The most important advantages are better scaling ability, computational effectiveness in terms of parallelization and training times (especially in contrast to RNN-based encoder-decoder architectures), and the attention mechanism providing better context windows to that of CNNs.

The typical speech-processing transformer operates on pre-computed spectral features[4] such as spectrograms. For ASR purposes, such an input is needlessly detailed along the time axis. This is why it is common to prepend a downsampling module to the transformer encoder.

The downsampling layer typically consists of several 2-D or 1-D (as shown in Figure 2.4) convolutional layers, processing the spectrogram and producing a sequence of downsampled feature vectors, shortened to a fraction (e.g. a fourth) of its original length. Often, the subsampling module simultaneously projects the original spectrogram features (typically 40 or 80-dimensional) to the operating embedding dimensionality $d_{model}$ of the transformer model, acting similarly to the text embedding layer in the language scenario. Reducing the time granularity of the input features ensures reasonable information density and distribution among the individual feature vectors in the sequence, compared to the original much more fine-grained spectrogram. This or a similar downsampling/intermediate feature extraction approach is universally used by most transformer and transformer-based model architectures [11, 20, 16, 38].

Recently, several new encoder architectures have been introduced, designed from the ground up to specifically accommodate speech processing tasks. Though these approaches are based on the original transformer encoder structure, their specific implementations of the respective encoder blocks differ. Perhaps best known among these is the Conformer [16], which generally outperforms the standard transformer on several ASR and speech processing

[4]Of course some models operate completely end-to-end, that is on raw audio files, learning to extract feature representations completely from scratch [2].

benchmarks. Another such encoder architecture is the Branchformer [34], followed by its successor, the E-Branchformer [22], which has been reported to offer a performance edge among all of these encoder architectures [35]. The E-Branchformer is important in the context of this thesis, as two of the baseline ASR models used in experiments conducted in Chapters 5 and 6, are based on this encoder architecture.

## 2.4 Transfer learning

The introduction of the transformer model was also quite impactful in the domain of transfer learning. Transfer learning is the previously mentioned deep-learning technique where one takes a model that has been pre-trained for a given task – such as the BERT model for masked language modeling – and utilizes the general capabilities and inner abstract models the model has built up during pre-training (with respect to the particular domain) to train a new model for a different objective/task. Typically, this new task is more specific, and the fine-tuning requires training data specific to this task.

Transfer learning alleviates some training costs for training new models. For some applications, this can even be the vast majority of the cost, for example, fine-tuning a pre-trained BERT model for the task of named-entity recognition only requires training a small linear layer with a few thousand parameters instead of the whole multi-million parameter transformer model. Transfer learning can, however, be exploited on much larger scales, as is demonstrated by models from e.g. the T5 family [40, 52, 29].

Overall, transfer learning is one of the most crucial concepts in modern deep learning, especially given the cost of training new powerful models from scratch (mainly referring to the costs of pre-training large language models [4, 31, 45]).

It is also well understood that generally, exploiting transfer learning leads to better results than training a model from scratch with only the final specific task in mind. This can be demonstrated by even a simple example of training a speech translation system, where initializing the model weights with a model trained for automatic speech recognition greatly improves the results, as discussed in Chapter 3. Therefore, in the context of this work, the topic of transfer learning is at the core of what this project is trying to explore. The specifics are discussed beginning from Chapter 4.

# Chapter 3

# Spoken language translation

*Spoken language translation* (SLT), often also referred to as *speech translation* (ST), is the task of transcribing the contents of a spoken audio utterance in a given source language (e.g. English) into text in a given target language (e.g. Portuguese).

This chapter provides an introduction and a brief overview of the methods that are typically used for training and establishing SLT systems.

## 3.1 SLT system architecture

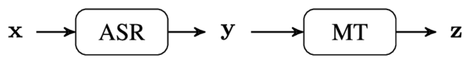
There are two main model architecture classes for spoken language translation. The first class is the so-called *cascade* speech translation system. Cascade ST systems in general consist of two parts:

1. An ASR system in the source language,
2. and an MT (machine translation) system from the source language to the target language.

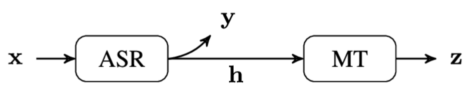
Now, assume a hypothetical scenario with the source audio in English and the target language being German. The ASR system first generates English transcriptions from the source audio. Then, the English-German MT system translates the generated transcriptions to the target German language.

The cascade approach is perhaps the most intuitive way of implementing speech translation, especially given the fact that training reasonably reliable ASR and MT systems is generally not that challenging, provided enough data. Moreover, the abundance of freely available pre-trained models for both ASR and MT means that constructing a cascade system is very simple. In terms of translation performance, models of this kind have always occupied the top positions with other state-of-the-art methods for SLT [36, 30]. In fact, given the reliability and domain robustness of modern ASR and MT systems, they still do nowadays. However, there are some potential drawbacks to these kinds of architectures.

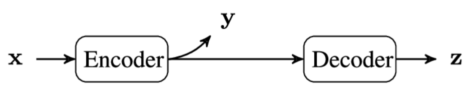
First, the models have to be used whole. This means that there are two encoders and two decoders, and therefore each forward pass through the system has two auto-regressive generation stages, resulting in higher latency. What is more, if the vocabularies between the ASR and MT models do not match (ASR models often generate lower-cased text without punctuation), a third model (or tool), which restores casing and punctuation information has to be added to the stack. In the end, a cascade system can easily have three generation stages and three potential sources of error accumulation.



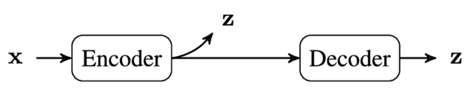
*a. Cascade system*



*b. Joint training with end-to-end differentiability*



*c. End-to-end model with transcriptions as auxiliary objective.*



*d. End-to-end model with translations as auxiliary objective.*

Figure 3.1: Diagram of different *cascade* and *end-to-end* SLT system architectures obtained form [21]. In the diagram, $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ denote the input audio features, source transcriptions, and target translations, respectively. $\mathbf{h}$ in this case represents the hidden states of the ASR system, which establish the differentiable connection to the MT model.

Secondly, in the base scenario, there is no differentiable path from the ASR system all the way to the translation system objective. The translation model therefore has no way of interacting with the speech encoder during training, no means to interpret its uncertainty during inference, meaning that the translations are perhaps more affected by any errors made during the transcription process. Alluding to the previous point with the truecaser model, committing the output of either model in the cascade scenario into text creates room for potential error accumulation and domain adaptation problems.

This second problem can be alleviated by establishing a differentiable path between the two models by, for example, feeding the hidden representations of the ASR system straight into the MT model input embedding space (Figure 3.1, variant *b*). The system can then be fine-tuned using different auxiliary objectives, as discussed in [47, 21].

### 3.1.1 End-to-end SLT systems

On the side of the spectrum, there are *end-to-end* (E2E)[1] speech translation systems. Such systems are typically implemented and trained as a single sequence-to-sequence encoder-decoder model, taking the source speech audio as input and producing the target language translation text as output.

Implementing an ST system with an end-to-end architecture can potentially mitigate some of the cascade system drawbacks; now the speech encoder and the translation decoder have a differentiable pathway all the way from the inputs to the objective. The encoder and

[1]For clarification: E2E is often used to represent a class of speech recognition models such as Wav2Vec2 [2] which operate on raw audio waveforms, having to learn their own speech representations during training. However, this label is as often used for systems that operate end-to-end on top of some low-level extracted features, such as spectrograms.

decoder can interact with each other, meaning that the passing of the speech information to the translation decoder happens in a more abstract and continuous manner, rather than committing it into discrete symbols. This helps to alleviate problems stemming from domain mismatch and error accumulation between the models in a cascade scenario.

However, training an end-to-end ST system is also generally more challenging than training ASR or MT systems alone, as the task is simply more complex. ASR systems *only* have to learn a monotonous mapping between the speech audio input and the corresponding sequence of tokens in the transcription. On the other hand, MT systems do have to learn to alter and rearrange the grammatical structures encoded in the source language input token sequence in order to produce a correct translation. However, they do not have to contend with the input modality being different from the output one, as in the ASR scenario. End-to-end ST systems have to learn to solve both of these problems at once.

In order to obtain better results, it is therefore common to implement several training techniques that decrease the training difficulty and improve performance. Those include mainly several transfer learning techniques, where the speech encoder and translation decoders would be initialized using weights from pre-trained ASR and MT models, respectively [18, 48, 21]. With further ST training after this initialization, the models quickly overcome the language and representation mismatch at the cross-attention connection, leading to generally better results than when training from scratch.

The training could also include additional model supervision [53, 55], self-supervised learning [49], or further regularization and conditioning using multi-task learning [18, 21]. For example, jointly enforcing the transcription objective at the output of the ASR encoder using CTC, and the translation objective at the output of the decoder, similar to what is shown in Figure 3.1.

Chapter 4

# Pre-trained model alignment

This chapter discusses the topic of pre-trained model alignment, considered mainly from the cross-modal viewpoint. This domain has been gaining traction recently, especially due to the introduction and success of large language models [31, 4, 45]. The model alignment research has mostly been trying to leverage the power that is offered by such systems by connecting them to pre-trained source modality encoders using a smaller module, which would facilitate the cross-modal alignment. That is, ideally without needing to fine-tune the large models[1]. Perhaps one of the more critical of these alignment connectors, the Q-former [25] (further discussed in Section 4.1) has recently been introduced and used a number of vision-language tasks, which previously required a more bottom-up approach. This method has subsequently sparked the inspiration for many cross-modal alignment projects [6, 9, 54, 56].

Aside from the Q-former, other alignment methods have been explored in [50, 17, 28, 1, 42, 7]. For most methods, however, the principle remains roughly the same:

1. Use a pre-trained *source modality encoder* (ASR encoder, vision model, etc.) and use it to extract abstract embedding representations from the source data.

2. Choose a powerful *language model*, which will provide the text generation – and perhaps a task-related interactive reasoning basis for the final task.

3. Train a small *modality connector* module, responsible for converting as well as projecting the source modality embedding representations into either the cross-attention connection space of the chosen language decoder or directly to its input embedding space, serving as a quasi-language soft prompt.

In this work, we attempt to apply similar approaches to the task of spoken language translation, trying to evaluate the merit of leveraging pre-trained ASR and MT models to create new ST systems and compare such systems to the more traditional methods of solving speech translation. The alignment experiments are conducted mostly in a smaller-scale, restricted scenario, utilizing our proprietary, smaller, pre-trained models, to allow for a more agile evaluation of the utilized alignment modules and architectures. However, experiments are also conducted to show the extensibility of the proposed aligned ST framework and its viability to be used with arbitrary larger-scale speech encoders and MT models. To the best of our knowledge, this work is the first that focuses solely on speech translation in this

[1]In most cases, the chosen encoder and decoder models are simply frozen and only the connector module is trained.

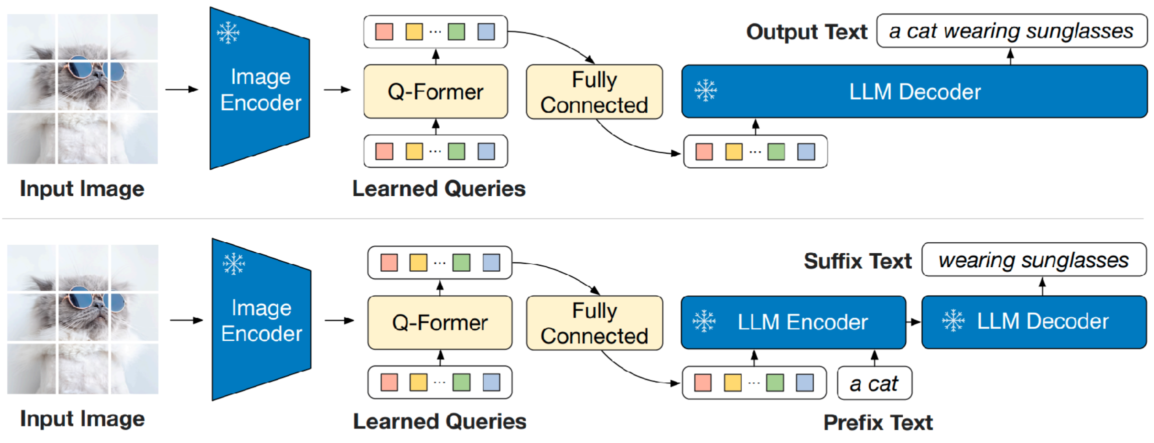


Figure 4.1: BLIP-2 Q-former alignment framework with both the decoder-only and encoder-decoder language model variants. The output Q-former queries bypass the LLM word embedding layer and serve as soft prompts for the language model to generate the image annotation. Diagram obtained from [25].

alignment context, and believe that our experiments show great prospects of utilizing and building on these methods in several ST task domains, including – but not limited to – low resource ST scenarios.

The next Section 4.1 introduces the Q-Former – the connector module that inspired this work as a whole – and its architecture, its possible uses for ASR, and some of its shortcomings with regard to sequence-to-sequence modeling tasks. Section 4.2 then discusses the proposed approach of aligning pre-trained ASR and MT models for solving speech translation, the two main alignment architectures and the modality connectors used within them.

## 4.1 The Q-Former

A novel pre-trained model alignment approach was recently introduced in BLIP-2 [25]. BLIP-2 presents a new method of fusing and aligning already pre-trained off-the-shelf large models for computer vision and language modeling to solve a joint vision-language task of describing the contents of a picture with natural language. Such an approach is quite attractive, as training capable vision-language models from scratch is generally a challenging task – especially resource-wise [37].

The paper proposes using a small *querying* transformer – the *Q-Former* – to connect a frozen image encoder and a frozen language model. The *Q-former* is used to extract information from the image features produced by the image encoder and present this information to the frozen language model in its input text embedding space as a quasi-language *soft-prompt*. To extract the desired information from the vision features, the Q-former uses a fixed-length sequence of trainable *queries*. These query vectors are used as the input of the Q-former and interact with the vision features via cross-attention. While the Q-Former has the architecture of a conventional transformer decoder, it processes the queries bidirectionally. The number of queries used is a hyper-parameter, and their dimensionality is defined by the chosen hidden representation size of the Q-Former.

Upon leaving the Q-former, the queries are projected into the input text embedding space of the frozen large language model with a fully-connected layer, acting as soft prompts to the LLM. Finally, the LLM auto-regressively generates the image annotations. The language model can be decoder-based (as shown in Figure 4.1), or encoder-decoder-based, which also allows using additional text prompts to potentially ask the LLM about more specific parts and aspects of the image.

In BLIP-2, overcoming the wide modality gap between the image encoder and the language model is a challenge, as there is no direct correspondence between the image embedding features and the text that describes it. For this reason, the Q-Former needs to be trained in two phases:

1. **The representation learning phase**, where the Q-Former first learns to extract representations from the image features, that are useful for generating the text annotation.

2. **The generative learning phase**, where the Q-Former is trained to use the learned representation extraction capabilities to supply the frozen LLM with a suitable soft-prompt, training with the regular causal language modeling objective based on the supplied image annotations.

In the representation learning phase, the self-attention and intermediate layers of the Q-Former are first initialized with the pre-trained weights of BERT-base [10], while the cross-attention layer weights are randomly initialized. Then, three pre-training tasks are employed to condition the Q-Former to extract useful representations from the image features:

- **Image-grounded text generation**, where the Q-Former is trained to use the trainable queries to extract useful information from the image features in the cross-attention layers, and then subsequently predict a textual annotation to the image using the self-attention layers, where the transformed queries can interact with the input text embeddings. a special causal self-attention mask is applied so that the text embeddings can interact with the queries, but not vice-versa.

- **Image-text matching**, where the Q-Former predicts whether the supplied text annotation matches the image features supplied in the cross-attention connection.

- **Image-text contrastive learning**, where the Q-Former is trained to maximize the mutual information (similarity) between the queries and the transformed output embedding of the `[CLS]` token of the supplied annotation (or minimize it in the contrastive case).

The pre-training methods are described in more detail in the original paper. The important observation here is that this representation learning stage with multiple tasks and contrastive learning is important in the vision-language case, as the modality gap needs to be shrunk first before employing the LLM for the second training stage. It should also be noted that here contrastive learning is necessary to prevent the representation learning process from collapsing.

For speech-text alignment, such pre-training is not necessary, as the modality gap is already considerably narrower [54, 50]. However, it can be argued that there is potential for similar types of pre-training or auxiliary supervision objectives and that such approaches could potentially decrease the need for large amounts of speech translation data, especially when aligning ASR and MT systems.

Following the introduction of the Q-former model, several new alignment approaches have been introduced, either attempting to improve and expand upon the results of BLIP-2 (such as InstructBLIP [9]) or use the Q-former as an alignment basis for different modalities entirely, e.g. video [6, 56] and audio [6, 54].

### 4.1.1 Q-former for automatic speech recognition

Using the Q-former for automatic speech recognition has recently been explored in [54]. The methodology remains similar to [25]: a large pre-trained speech encoder like Whisper [38] is frozen, and used to encode the input audio into a sequence of intermediate speech features. The features are passed to the Q-former via cross-attention, where they interact with the Q-former queries (see Figure 4.3 for reference). This means that the Q-former turns the variable-sized input audio sequences into a *fixed length sequence* – the length of the Q-former output sequence is the same as the input query vector sequence. The transformed output queries are subsequently projected into the embedding dimension of the connecting LLM, prompting it to generate the transcription. Once again the training objective is the original cross-entropy loss of the foundation LLM, however, only the Q-Former is optimized during the training process.

In comparison to BLIP-2, the Q-former used here is much smaller – only two transformer decoder layers – and does not require any pre-training, as it is argued that the modality gap between the ASR encoder outputs and the LLM embedding space is relatively narrow. The number of queries that were used for ASR purposes is 80 (in contrast to the 32 used in [25]), which is, reportedly, a sufficient amount to retain the information in the encoded input audio slices with lengths of up to 30 seconds with an acceptable performance-to-length ratio.

The performance of the Q-former is also compared to two other connector modules – a stack of linear layers and a multi-head cross-attention module [28] – outperforming them all. It should be noted that the performance of the linear connector ends up behind the Q-former with only a tenth-of-a-percent absolute WER margin, however, the study argues that the Q-former represents a superior approach due to the higher computational complexity of the fully-connected layer module in comparison to the Q-former. On the other hand, the fact that a simple MLP was able to match the results of the Q-Former further reinforces the notion, that the modality gap between the speech encoder output representations and the LLM embedding space is perhaps narrower than expected.

### 4.1.2 Handling variable-length audio inputs

It is common for ASR models to be optimized to operate on shorter audio inputs. In [54], Whisper [38], which is trained and designed to operate on 30s chunks of input speech segments, is used as the speech encoder. On the other hand, the context windows of the chosen LLM allow for processing longer input sequences and prompts.

To counteract the source audio length limitation, the paper proposes a segment-level Q-Former modification. This Q-former variant (shown in Figure 4.2) would receive batches of the encoded source audio from the speech encoder and add positional embedding information to the whole batch. Then, each segment is processed by the Q-former separately and the output query sequences are concatenated. Once the whole source audio is processed,

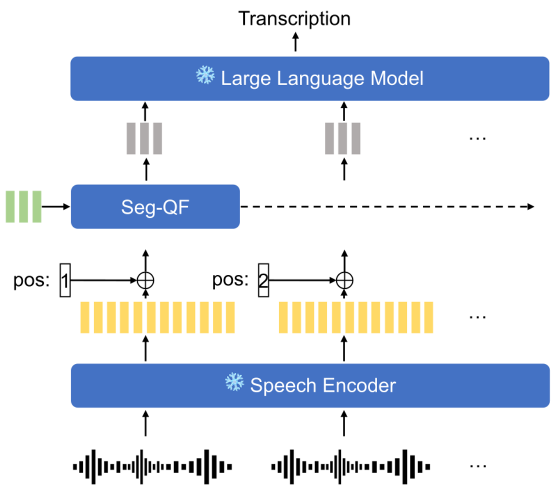


Figure 4.2: Segment-level Q-former structure. The Q-former queries are depicted in green, outputs of the speech encoder in yellow. The rectangles with numbers represent the added segment positional encoding. Diagram obtained from [54].

the concatenated queries are presented to the LLM as one long prompt, and transcriptions are generated.

However, it could be argued that this segment-level approach is a symptom of a deeper mapping problem with the Q-Former – due to the cross-attention interaction, the Q-Former always maps any variable-length inputs to a fixed-length sequence of output queries. Such an approach makes sense in the BLIP-2 scenario as in that case, the system is dealing with abstract image feature representations. It is the actual meaning and content stored in these image features that determine the length of the output text annotation, not the dimensions of the features. Extracting this information into a fixed number of query tensors is therefore adequate, as there is no direct basis for determining the length of the connector module output at the Q-Former stage without perhaps introducing an auto-regressive decoding step.

For the ASR case (and the speech translation case for that matter), the relationship between the encoder output embeddings and the generated text transcription (or translation) is much more linear and monotonous – if the input audio utterance is long, it is probable that so will be the text. This immediately poses the question if it perhaps would be more productive to require a variable-length to variable-length mapping from the connector module.

### 4.1.3 The Subsampler-Transformer Encoder connector (STE)

In addition to the Q-Former, we, therefore, experiment with another connector module variant inspired by the approaches presented in [50, 17]. This variant, though still rather simple, solves the fixed-length mapping of the Q-Former by introducing a convolutional subsampling layer in replacement of the cross-attention connection.

ASR encoder output embedding sequences typically have a higher granularity in comparison to language model text representations. This granularity mismatch could potentially cause problems for leveraging any zero-shot capabilities the language-model might offer when prompted with these raw audio representations, therefore it is natural to downsample

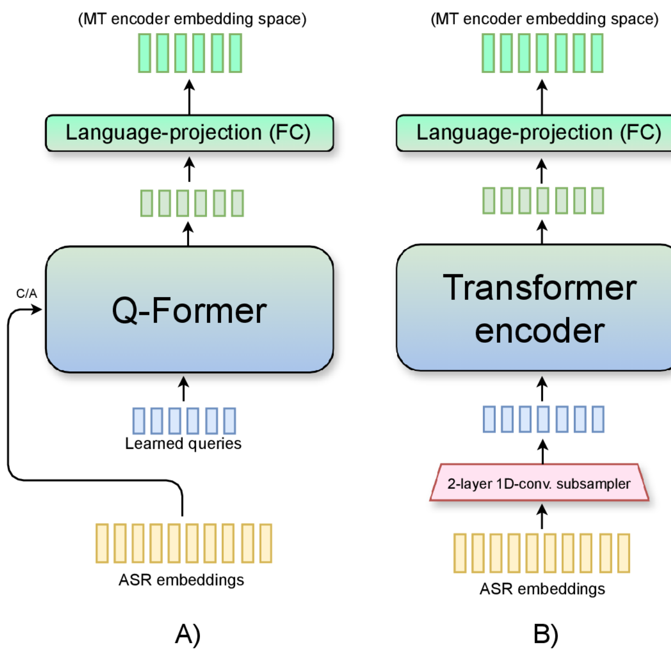


Figure 4.3: Diagram of the two alignment modules. Variant A) is the Q connector (or Q-Former), variant B) is the STE connector. Note the different embedding vector 'height' used by the connectors, which illustrates that the hidden size of the connector network can actually be different to that of the ASR encoder or the MT model.

these embeddings along the time dimension. In [17], two main subsampling approaches are explored – CTC compression and convolutional downsampling (evaluating the convolutional approach as superior). The hidden audio representations are subsampled to a fraction of their original length, and only after that, they are presented to the connector module, which, at this point, can be a simple transformer encoder. On the other hand, in [50], the subsampling is done by simply stochastically discarding a fourth of the output ASR embeddings, still yielding great results.

While the convolutional subsampler adds a number of parameters to tune (in our case directly correlated to the ASR embedding dimensionality), an approach like this intuitively seems superior, because of the fixed-length mapping constraint of the Q-Former. An additional small perk of this second variant is also that no parameters in the model come to waste – in contrast to the Q-Former, which cannot use the first self-attention block in any meaningful way, as it only operates on the raw pre-trained query vectors when used in the final alignment encoder-connector-LLM context.

Going further, this type of connector network will be referred to as the 'STE' connector, which stands for *Subsampler-Transformer Encoder*. The STE connector is depicted in Figure 4.3.

## 4.2 Proposed alignment architectures

In this work, we experiment with two general alignment architectures for speech translation, differing in the configuration of the frozen MT model. Both architectures are trained primarily using the standard cross-entropy loss objective at the output of the MT decoder.

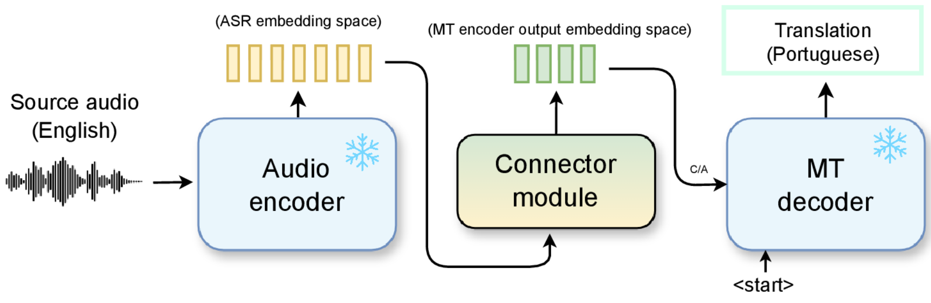


Figure 4.4: Diagram of the ECD alignment configuration. The original MT encoder is stripped away, with the connector module taking its place. The connector module output embeddings are used as the cross-attention (C/A) input to the MT decoder. Both the audio encoder and the MT decoder are frozen.

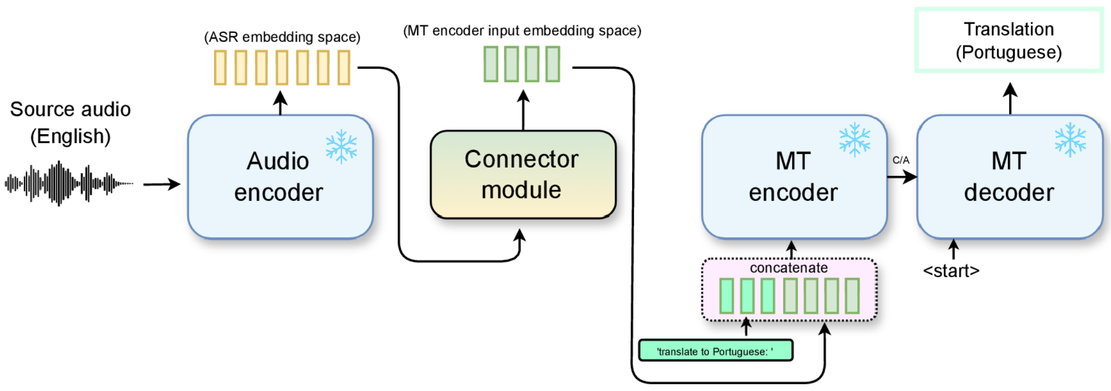


Figure 4.5: Diagram of the ECED alignment configuration. The connector module output embeddings are now directly injected past the input embedding layer of the MT encoder. Additional task prompt embedding sequence (as is typical for instruction-tuned models such as T5 [40]) can be pre-pended to the connector outputs before entering the MT encoder.

The first alignment architecture (shown in Figure 4.4) consists of a *frozen* ASR encoder (the decoder is stripped away to avoid an extra decoding step), which extracts hidden audio representations from the source speech audio. These representations are then passed to the connector module, and its outputs are further projected to the dimension of the *frozen* MT model via a fully-connected layer. In this architecture variant, the connector is used to align the ASR encoder output embedding space with the output embedding space of the MT encoder, as shown in the diagram. Additionally, the original pre-trained MT encoder can be used to initialize the weights of the connector network, or additionally serve as a basis for some auxiliary *modality matching* objectives useful for stabilizing the training procedure. This architecture will further be referred to as *Encoder-Connector-Decoder* (ECD).

The second architecture (shown in Figure 4.5) differs from the first one in the text decoder part. The connector module now projects the speech embeddings into the input text embedding space of the foundation MT encoder. Intuitively, it would seem that in this case, the modality gap between the output speech embeddings and the input text embeddings of the MT encoder is narrower, than in the previous case. Whereas previously the connector had to perform feature space alignment and simultaneously replace and overtake the responsibilities of the original MT encoder, now it is enough for the connector to meaningfully bring up the textual information present in the speech embeddings so that the MT encoder can make use of them. This kind of approach was, for example, explored in [50], where aligning the speech encoder with a language model from the T5 [40] family allowed the authors to leverage the original multi-task and reasoning capabilities of the foundation multi-lingual language model, only enhanced in that the final model could also process speech inputs. In general, this encompasses first taking the text instruction prompt such as: ‘translate to Portuguese: ‘, and subsequently embedding it with the MT encoder embedding layer and prepending these instruction embeddings to the connector module outputs. This architecture will further be referred to as *Encoder-Connector-Encoder-Decoder* (ECED).

For both architecture variants, we conduct experiments with two different types of connector networks: the *Q-Former* (which will also further be referred to as simply ‘Q‘) and the STE connector (a transformer encoder with a convolutional subsampler frontend), as it was introduced in Section 4.1.3. The subsampling module used in our STE implementation is a 2-layer stack of 1D convolutions, which reduces the length of the input speech embedding sequence by a factor of 4, and was inspired by the frontend used for ASR systems in [48], and simultaneously projects the subsampled embeddings from the dimensionality of the ASR encoder to the operating dimensionality of the connector.

# Chapter 5

# Experiments

In this chapter, experiments are conducted to evaluate the pre-trained model alignment approaches and architectures as they are described in Chapter 4. The main aim of these experiments is to determine the viability of these alignment approaches for solving the speech translation task, comparing these methods to our baseline systems constructed in more conventional ways. Subsequently, we further explore the possibilities offered by off-the-shelf pre-trained ASR and MT models in the ST alignment context.

All experiments carried out and models used throughout this work were implemented using the Hugging Face[1] Transformers[2] framework. Hugging Face Transformers is an open-source deep learning library based mainly on PyTorch[3], which mainly aims to centralize and provide simple, effective, and extensible tools to train and fine-tune modern transformer-based models. It also provides a platform for hosting already pre-trained models that are free to be used as off-the-shelf bases for further experiments and projects.

A currently access-restricted Hugging Face Transformers extension repository for ASR training called ‘`huggingface_asr`‘[4] was used as the platform for implementing all models, and training/evaluation scripts. This repository is developed and maintained mainly by Ing. Alexander Polok from the BUT Speech@FIT research group from our faculty. All recipes and code written and used for the experiments conducted throughout this work will ultimately become available as a part of the toolkit, once the repository becomes public. For now, the enclosed storage unit for this work contains the current image of the repository, where all source files created and used for the purposes of this project are appropriately annotated.

Further sections discuss the important topics with regard to the conducted experiments. Section 5.1 introduces the `How2` dataset used in the experiments, along with the WER and BLEU metrics used for evaluation. Section 5.2 goes over the training and selection of the foundation ASR and MT models used for the alignment experiments. Section 5.3 then describes the building and establishing of our reference baseline English to Portuguese ST systems trained on the `How2` dataset. Section 5.4 evaluates the merits of both alignment architectures and connector modules. Section 5.5 explores the behavior of the alignment approach when used with off-the-shelf pre-trained ASR and MT models. Lastly, Sections 5.6 and 5.7 attempt to analyze and compare the behaviors of the Q and STE connectors in terms of input sequence lengths.

[1]https://huggingface.co/

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

[3]https://pytorch.org/

[4]https://github.com/BUTSpeechFIT/huggingface_asr

Table 5.1: Split statistics of the 300h portion of the How2 dataset.

| Split | Videos | Hours | Clips/utterances | Per clip statistics |
|---|---|---|---|---|
| train | 13,168 | 298.2 | 184,949 | |
| val | 150 | 3.2 | 2,022 | 5.8 seconds / 20 words |
| dev5 | 175 | 3.7 | 2,305 | |

## 5.1 The How2 dataset

The How2 dataset [41] is a multi-modal corpus of English instructional videos and their respective transcriptions. The dataset consists of approximately 2000 hours of video data and contains a smaller 300-hour audio subset, for which there are also crowd-sourced translations to the Portuguese language.

The 300-hour subset is divided into three partitions: train, val, and dev5. As shown in Table 5.1, the train partition consists of 298 hours of audio data, totaling 184949 spoken English utterances. Both the dev5 and val partitions contain about 3 hours of speech data, both totaling just above 2000 utterances. The lengths of most utterances in the 300-hour subset range from one to up to twenty seconds, with most recordings being concentrated in the region around the 4 seconds – the overall utterance length distribution is shown in Figure 5.1.

How2 is a commonly used standard benchmark dataset for machine translation and speech translation systems [30, 18, 47, 21] – due to the data being relatively clean, it is considered to be appropriate for studying and evaluating various approaches and models for speech translation. The How2 dataset is freely available for download, however, access to it is restricted behind a license gateway. This is useful, as it essentially restricts any web crawlers from accessing the data directly, minimizing the risk of data contamination in our experiments with any off-the-shelf pre-trained models.

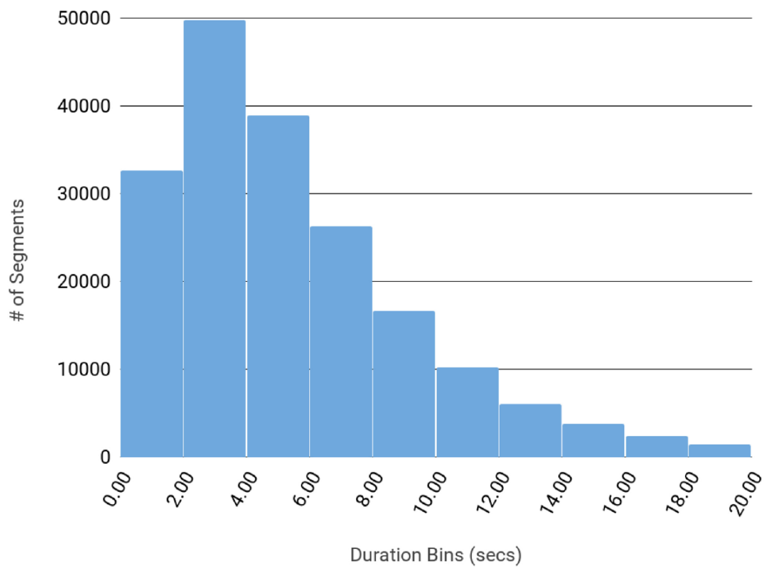


Figure 5.1: Utterance length distribution of the 300-hour How2 subset. Figure obtained from [41].

### 5.1.1 Evaluation metrics

For evaluation, we use the Word Error Rate (WER) metric for ASR systems, and for translation systems, the BLEU metric is used.

#### Word Error Rate

Word Error Rate is a standard metric used to evaluate speech recognition systems, aggregating several types of errors the model can make when generating the transcription hypothesis in relation to the reference text:

$$\text{WER} = \frac{S + D + I}{N}, \tag{5.1}$$

where $S$ is the number of substitutions (words that were changed in the hypothesis), $D$ is the number of deletions (words that are missing from the hypothesis altogether), $I$ denotes the number of insertions (words that cannot be found in the reference but are present in the hypothesis), and $N$ is the number of words in the reference.

WER is typically computed using normalized, lower-cased text without punctuation, as casing and punctuation information can be more open to interpretation. Normalizing the hypotheses and references allows for more accurate evaluation of the actual ASR performance in terms of purely transcribing the correct words.

#### The BLEU metric

BLEU (Bilingual Evaluation Understudy) [32] is a metric used for evaluating the performance of machine translation (and speech translation) systems. Roughly speaking, the metric compares a translation hypothesis to a set of reference translations. These references should be of good quality, representing the ideal translation performance, presumably originating from a top-level human translator.

The value domain of the scores ranges from 0 (worst) to 100 (best). However, a perfect BLEU score represents a virtually unattainable ideal, where the generated translations would have to match the references perfectly. In a translation scenario, this is not realistic for the simple reason that multiple translation hypotheses can be valid for one input source language sentence.

BLEU is computed over N-gram sequences of words, with 4-grams being quite common. To obtain the score reading, first, it is necessary to compute 1 to 4-gram precision scores between the hypothesis and the reference. Then, to penalize generating short hypotheses if the references are long, a brevity penalty is computed to offset the potential high precision scores for these scenarios:

$$\text{Brevity penalty} = \begin{cases} 1, & \text{if } c > r, \\ e^{1-r/c}, & \text{else} \end{cases} \tag{5.2}$$

where $c$ is the hypothesis length and $r$ is the reference length.

The final score is then obtained by computing a geometric mean precision from all N-gram precision scores and multiplying it by the brevity penalty:

$$\text{BLEU(N-gram)} = \text{Brevity penalty} \cdot \prod_{i=1}^{N} p_i^{w_i}, \tag{5.3}$$

where $p_i$ is the $i$-gram precision score, and $w_i$ is the weight of the $i$-gram (the weights are often chosen to be uniform).

Contrary to WER, BLEU scores are computed using true-cased text with punctuation symbols, as machine translation systems make use of this information to produce better translations. The actual meaning of the produced translation can be affected by both casing and punctuation information, and therefore it is natural to include it when comparing to the references.

## 5.2 Baseline ASR and MT systems

First, it is necessary to train and select reliable ASR and MT system baselines. These systems would subsequently be used to establish our end-to-end and cascade ST baselines, as well as serve as the foundation models for a large portion of the alignment experiments in the context of the architectures described in Section 4.2. Our goals for these baseline models are the following:

1. Train in-domain ASR and MT systems on `How2` with close to state-of-the-art performance used to build reference baseline ST systems and subsequently serve as foundation models for evaluating the alignment architectures in a domain-restricted scenario.
2. Select different off-the-shelf pre-trained (possibly out-of-domain) ASR and MT models, used to evaluate the viability of the alignment architectures as generic frameworks for solving speech translation.

For our proprietary in-domain models, we take inspiration from ESPnet [18] and build our ASR and MT systems to match the sizes and results of ESPnet systems on the `How2` dataset[5]. Doing so allows us to better ground our experiment results.

### 5.2.1 Training the ASR model

The reference ESPnet ASR system trained on `How2` is a hybrid CTC/attention [20] transformer encoder-decoder model of 30 million parameters, with a 12-layer encoder and a 6-layer decoder, achieving 13.0% WER on the `dev5` set. During our preliminary experiments we tried to replicate ESPnet ASR results on `How2` using the same transformer-based architecture implemented in Hugging Face under the name Speech2Text[6] [48] (S2T). However, we were unable to do so, partly due to some training stabilization problems when implementing the encoder CTC objective.

The best S2T ASR system trained during this preliminary phase achieved 17.32% WER on the `dev5` `How2` subset. Because this system was not able to match the reference ESPnet result, there was a concern that any cascade/end-to-end ST systems based on this ASR model, or any subsequent alignment experiments would give unsatisfactory results. In both cases, these concerns ended up being substantiated, as is further discussed in Sections 5.3 and 5.4.1.

Ultimately, we decide to pivot and train a baseline ASR system with the same basic dimensions (256 hidden size, 12-layer encoder, 6-layer decoder), but whose encoder is based on the novel E-Branchformer [22] architecture. Just as for ESPnet models, the CTC objective is used at the output of the encoder with a weight of 0.3 alongside the standard cross-entropy loss at the output of the decoder.

[5]`https://github.com/espnet/espnet/tree/master/egs/how2`
[6]`https://huggingface.co/docs/transformers/model_doc/speech_to_text`

This ASR system – we refer to it as *E-Branchformer small* – was trained for 70 epochs, with early stopping enabled to avoid overfitting, batch size of 128, a learning rate of $1e^{-3}$, and 20000 warm-up steps. The model utilizes a lower-cased vocabulary with all punctuation except for the apostrophe removed. The vocabulary of the model is based on the unigram tokenization method and has a size of 5000. For audio features, 80-dimensional log-mel-spectrograms are used. To further improve the performance and robustness of the ASR model, SpecAugment [33] is used to apply random time and frequency masks to the spectrogram, specifically in the 'LD' strategy, as described in the original paper. Before extracting the mel-filterbank features, speed perturbation is applied to the source audio in factors of [0.9, 1.0, 1.1] to further augment the dataset.

While this model has about 7 million more parameters than the ESPnet `How2` baseline, and the E-Branchformer architecture is generally considered to be more powerful for speech recognition tasks than the conventional transformer [22, 35], it allows us to easily match ESPnet ASR results on `How2` , as it achieves 12.6 and 12.2% WER on the `val` and `dev5` `How2` subsets.

For more details about the used ASR systems, refer to Table 5.3.

### 5.2.2 Training the MT model

The baseline machine translation system adopts the MarianMT [19] transformer implementation available in Hugging Face Transformers and consists of a 6-layer encoder and a 6-layer decoder, 4 attention heads, embedding size of 256, and intermediate feed-forward layer dimension of 2048. The source and target word embedding layers were untied in accordance with the experiments in [18]. The system was trained in a true-cased to true-cased manner with both the source and target BPE-based vocabularies containing 8000 word units.

The model was trained for a maximum of 50 epochs with 10k warm-up steps and a peak learning rate of $1e^{-3}$, additionally adopting early stopping. The model was evaluated using the BLEU metric and achieved 57.90 and 56.95 BLEU on the `val` `dev5` `How2` subsets, respectively, which is on par with the results of [18] on `How2` with the same model architecture.

In further experiments, this model will be referred to as either *MarianMT small*, or just *MarianMT*.

Table 5.2: Performance comparison of the foundation MT systems used in the alignment experiments, evaluated on the `How2` dataset. The MarianMT model was trained in-domain on the `How2` corpus, the T5 model is out-of-domain.

| **Model** | **In domain** | `How2` **BLEU** ↑ | | **# of parameters** |
|---|---|---|---|---|
| | | `val` | `dev5` | |
| MarianMT small | Yes | 57.9 | 57.0 | 21.6M |
| T5-en-pt | No | 40.0 | 38.8 | 223M |

### 5.2.3 Choosing off-the-shelf ASR and MT models

On top of our two proprietary systems trained on `How2` , we additionally choose some off-the-shelf pre-trained ASR and MT models available on Hugging Face to conduct further experiments with. Since our two models are both in-domain on `How2` , it is not necessary

Table 5.3: Performance comparison of the foundation ASR systems used in the alignment experiments, evaluated on the How2 dataset. The S2T, E-Branchformer small, and ESPnet reference models were trained solely using How2 data, the E-Branchformer medium and Whisper models were not trained on How2 . The ESPnet CTC/attn. transformer result is included for reference.

| **Model** | **Trained on How2** | **How2 WER ↓** | | **# of parameters** |
|---|---|---|---|---|
| | | val | dev5 | |
| S2T transformer baseline | Yes | 17.6 | 17.3 | 29M |
| CTC/attn. E-Branchformer small | Yes | 12.6 | 12.2 | 38.5M |
| CTC/attn. E-Branchformer medium | No | 12.1 | 11.7 | 174M |
| Whisper-small.en | No | 9.7 | 7.9 | 242M |
| ESPnet CTC/attn. transformer | Yes | - | 13.0 | 30M |

for these additional off-the-shelf models to perform that well on How2 without fine-tuning. Rather than that, the main requirement is for the models to be generally more capable than our proprietary ones. This allows for evaluating the performance scaling behavior of the two alignment architectures, the domain and dimensionality adaptation capabilities of the connector network, etc. In fact, it can be argued that if the foundation models are out-of-domain on How2 , it would be more akin to real-life scenarios, where one would probably choose ASR and MT models that might both be powerful, but also mismatched – both architecturally and domain-wise. For a comparison of all baseline MT models, refer to Table 5.2, and for a comparison of baseline ASR systems, refer to Table 5.3.

Ultimately, we choose and conduct experiments with two additional pre-trained ASR models and one MT model. The chosen MT model is an English-to-Portuguese MT model based on the T5 [40] architecture, which was developed for experiments conducted in [27]. The model was based on an English T5-base checkpoint, which was first pre-trained on Portuguese language [5], and then fine-tuned for English to Portuguese translation on a 5M English-Portuguese sentence subset of the ParaCrawl dataset [14], as well as some domain-specific data in preparation for the WMT19 and WMT20 biomedical translation tasks. Both the encoder and decoder consist of 12 layers with a hidden size of 768, resulting in a model of 223M parameters.

This MT model is out-of-domain on How2 , achieving ‘only‘ 40.0 and 38.8 BLEU on the **val** and **dev5** sets, respectively. However, we chose this model primarily due to it being pre-trained on Portuguese language and thus hope to leverage its Portuguese text-generation capabilities in our alignment experiments. The model is freely available on Hugging Face[7], and will further be referred to as simply the T5 model.

Additionally, we choose two other ASR models. The first one is essentially a scaled-up version of our E-Branchformer small model, trained with additional *decoder-centric regularization*, a novel ASR training method developed by Ing. Alexander Polok at BUT@Speech. The method has not been published yet, however, the model is freely available on Hugging Face[8]. The model was once again trained with an additional CTC objective on top of the 16-layer E-Branchformer encoder. The decoder has 8 layers, and the whole model has a hidden size of 512, therefore totaling around 174 million parameters. Despite not being

[7]https://huggingface.co/unicamp-dl/translation-en-pt-t5
[8]https://huggingface.co/BUT-FIT/EBranchRegulaformermedium

trained on How2 , it achieves reasonable 12.1 and 11.7 WER on the val and dev5 sets, respectively. Moreover, it was trained on 6000 hours of English speech data from various datasets, providing an excellent baseline for an off-the-shelf ASR model with, hopefully, good general audio embedding representation extraction capabilities. In our experiments, it is referred to as E-Branchformer medium.

The second of the two additional ASR models is the English-only Whisper-small.en[9] model by OpenAI [38]. The encoder and decoder of this model both consist of 12 layers with a hidden size of 768. Whisper-small.en achieves 9.7% and 7.9% WER on val and dev5 , respectively. As with the other ASR models, to compute this WER value, both the transcriptions and Whisper output were normalized using the Whisper normalizer, then lower-cased and all punctuation symbols were removed. Without normalization, the WER values are 18.4% and 16.5% WER on the val and dev5 sets, respectively. We use this model in only a few experiments to further demonstrate the ability of our alignment framework to leverage good hidden representation-building capabilities of our foundation models (as this particular model was pre-trained on 680 thousand hours of audio data).

## 5.3 Baseline ST systems

To establish a credible speech translation performance reference, two baseline ST systems are created (plus the preliminary S2T-based system), each representing one side of the cascade/end-to-end architecture spectrum. Both baseline systems are created using the E-Branchformer small ASR and MarianMT small MT systems described in Sections 5.2.1 and 5.2.2. These systems are in-domain on the How2 dataset and were trained only using this data.

First, we train an end-to-end ST system with the same architecture as the foundation E-Branchformer small model. Adopting the How2 approach from [18], we reuse the 12-layer encoder from the pre-trained E-Branchformer small ASR model and initialize the 6-layer decoder with the weights from the MarianMT small decoder. The vocabulary of the decoder is the same true-cased 8000-word unit BPE vocabulary used by the original MT decoder. The system is then trained for a maximum of 40 epochs with a learning rate of $1e^{-3}$, 10000 warm-up steps, and a batch size of 128. Early stopping is used to avoid overfitting. For this ST training phase, SpecAugment is no longer used, however, we keep the speed perturbation augmentation with the same factors of [0.9, 1.0, 1.1]. Lastly, the encoder is frozen for the first 8 epochs of training. This baseline system achieves 45.6 and 45.2 BLEU on the val and dev5 How2 subsets, respectively, which is on par with the results of the reference ESPnet system[10], as shown in Table 5.4.

This end-to-end system represents the more robust but – at the same time – more costly solution to the ST task. We obtain good performance using a small model, which employs only one decoding step and does not suffer from domain mismatch. At the same time, we still have to optimize all the parameters of the model to get the best results, and the training would only consume more and more resources if the model size were to scale up. Experiments presented in the following sections will attempt to argue that fine-tuning the whole ST system is not necessary to obtain performance comparable to end-to-end systems trained from scratch. That is if an appropriate alignment method is used.

[9] https://huggingface.co/openai/whisper-small.en
[10] https://github.com/espnet/espnet/tree/master/egs/how2/st1

Table 5.4: Comparison of both trained and reference ST systems. The ESPnet reference is an end-to-end transformer system, utilizing the same ASR and MT initialization approach as our E-Branchformer end-to-end baseline. The S2T baseline is based on our preliminary ASR transformer system from Section 5.2.1.

| **Baseline ST system** | `How2` **BLEU** ↑ | |
|---|---|---|
| | `val` | `dev5` |
| E-Branchformer end-to-end (ASR + MT init.) | 45.6 | 45.2 |
| Cascade (Ebr. ASR → truecaser → MarianMT) | 40.9 | 40.4 |
| ESPnet transformer reference | - | 45.7 |
| Preliminary S2T transformer baseline | 40.6 | 39.6 |

For completeness, the last line of Table 5.4 also shows the preliminary end-to-end ST system based on the S2T transformer ASR from Section 5.2.1. This system was trained in the same way as the E-Branchformer system, utilizing the S2T ASR encoder and the MarianMT decoder as weight initialization points for the final system. Due to the worse ASR performance of the original S2T model, this end-to-end preliminary ST baseline only achieves 40.6 BLEU on the `val` set.

The other baseline ST system is a cascade system, connecting the E-Branchformer small and MarianMT small models. For this system, no component is trained, however, a minor issue arises, as the output vocabulary of the ASR model only consists of lower-cased text with punctuation removed. The MT system on the other hand expects true-cased English text with punctuation as its input. This type of domain mismatch can be quite common among off-the-shelf ASR and MT systems as ASR systems are often trained to only produce lower-cased, normalized text, contrary to MT systems, which utilize the casing and punctuation information in the source text to produce better translations.

To overcome the vocabulary mismatch, we take a pre-trained T5-based English casing and punctuation restoration model from Hugging Face[11], and use it to process the ASR transcriptions before translating them with the MT model.

Our cascade system represents the *naive* but cheap solution to the ST task, as no parameters have to be tuned. However, the overall performance suffers from error accumulation, resulting in noticeable performance degradation, as the model only achieves 40.9 and 40.4 BLEU on the `val` and `dev5` `How2` subsets, respectively. What is more, three decoding steps have to be performed before obtaining the final translation.

## 5.4 Alignment architecture evaluation

The first set of alignment experiments mainly encompasses evaluating the two alignment architecture variants (ECD, ECED) in conjunction with both of the connector network types (Q and STE), as described in Section 4.2. These experiments aim to primarily determine the viability of the aligning approach to training ST systems as a whole, juxtaposing them against the baseline methods. On top of that, experiments are conducted with different connector model compositions, namely in terms of the number of transformer layers, the number of queries used by the Q-Former, etc.

[11] `https://huggingface.co/SJ-Ray/Re-Punctuate`

Unless specified otherwise, all aligned models are trained for a maximum of 70 epochs with 15000 warm-up steps, a batch size of 128, and a peak learning rate of $2e^{-4}$. We adopt early stopping to avoid overfitting and apply speed perturbation with the same factors as in Section 5.3.

### 5.4.1 Architecture 1 – ECD

First, we evaluate the ECD architecture, where the frozen ASR encoder is connected to the frozen MT decoder using the trained connector network. Starting off, the E-Branchformer small ASR and the MarianMT small MT models (both trained in-domain on `How2` ) are chosen as the foundation models for the first part of alignment architecture evaluations. Though these models are not the most powerful in comparison to other off-the-shelf models pre-trained on thousands of hours of speech data, keeping both the ASR encoder and MT decoder in-domain on the training set allows us to better isolate and identify potential shortcomings of the aligned models. Furthermore, the final results will be directly comparable to the ST baselines from Section 5.3.

Since both foundation models have a hidden size of 256, we start off by defining both the Q and the STE connectors as 6-layer transformer models with 4 attention heads, a hidden size of 256, and an intermediate feed-forward layer with a size of 2048. The Q-Former uses 100 trainable queries as input, the STE connector is prepended with the 2-layer 1-D convolutional subsampler frontend, introduced in [43] and used in Fairseq [48]. Decreasing the number of connector layers to 4 or 2 is also experimented with.

For the Q-Former, an additional modality-matching pooling loss is employed, inspired by [13]. While training, we keep the original pre-trained MT decoder and use it to obtain hidden representations $\mathbf{T}$ from the English transcription of the input speech utterance. Both the transformed Q-Former output queries $\mathbf{Q}$ and the MT encoder output embeddings $\mathbf{T}$ are then averaged along the time dimension to obtain single mean embedding vectors $\mathbf{q}$ and $\mathbf{t}$. The modality-matching loss is computed as mean-squared-error between the elements of the two vectors:

$$L_{\text{MM}} = \text{MSE}(\mathbf{q}, \mathbf{t}), \tag{5.4}$$

essentially trying to enforce that the Q-Former output embeddings occupy the same space as the output embeddings of the transcription, as it is encoded by the original MT encoder. We find that while the modality-matching loss does not bring a clearly attributable increase in performance, it helps stabilize the training.

Additionally, both the Q and STE connectors can be initialized from the original MT encoder weights (the cross-attention layers of the Q-Former would be initialized randomly). However, since this can only be meaningfully done when the connector dimensions match the dimensions of the MT encoder, we ultimately refrain from doing so for most experiments (especially down the line when bigger foundation ASR and MT models are used). For the experiments, where the initialization was performed, we find no difference in performance, however, for the STE architecture the initialization makes the model converge slightly quicker.

From the results shown in Table 5.5 the STE connector outperforms the Q-Former by a significant margin, almost matching the performance of the baseline end-to-end system, achieving 45.0 and 44.8 BLEU on the `val` and `dev5` sets, respectively. It could be argued, that the STE connector perhaps benefits too much from the parameters added by the convolutional subsampler, however, even the 4-layer STE configuration still outperforms the 6-layer Q-Former, suggesting that the problem lies probably in the way the Q connector

Table 5.5: Connector performance comparison for the ECD alignment architecture. The S2T model is our ESPnet-inspired preliminary transformer baseline from Section 5.2.1. Baseline ST model results from Section 5.3 are added for reference.

| ASR enc. | Connector | Connector layers | Queries | MT enc. | MT dec. | Trainable parameters | `How2` BLEU ↑ | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | `val` | `dev5` |
| Ebr. small | Q | 6 | 100 | - | Marian 6l | 9.6M | 44.0 | 43.9 |
| Ebr. small | Q | 4 | 100 | - | Marian 6l | 6.4M | 42.5 | 42.6 |
| Ebr. small | Q | 2 | 100 | - | Marian 6l | 3.2M | 40.6 | 40.2 |
| Ebr. small | STE | 6 | - | - | Marian 6l | 10.7M | **45.0** | **44.8** |
| Ebr. small | STE | 4 | - | - | Marian 6l | 7.9M | 44.1 | 44.4 |
| Ebr. small | STE | 2 | - | - | Marian 6l | 5.3M | 42.8 | 42.9 |
| S2T | Q | 6 | 128 | - | Marian 6l | 9.6M | 36.8 | 36.1 |
| E-Branchformer E2E baseline | | | | | | 38.5M | 45.6 | 45.2 |
| Cascade baseline | | | | | | 0 | 40.9 | 40.4 |
| S2T E2E transformer preliminary baseline | | | | | | 29M | 40.6 | 39.6 |

represents and extracts information from the input speech embeddings. This is further analyzed in Sections 5.6 and 5.7.

What is more, we also find that the training of the STE connector is more stable and ‘well-behaved‘ than the one of the Q-Former – it converges faster and spikes in evaluation metrics while training are less common.

To further stress the importance of utilizing a quality ASR encoder for the alignment experiments, Table 5.5 also shows the performance of one of the preliminary Q-Former aligned models with the S2T transformer encoder (see Table 5.3). The performance gap between the aligned model and the reference preliminary S2T end-to-end baseline is almost 4 BLEU points, which is considerably more than when a better ASR encoder (like our E-Branchformer small) is used.

### 5.4.2 Architecture 2 – ECED

The second ECED architecture, where the connector maps the ASR encoder output embeddings into the input embedding space of the foundation MT encoder, should potentially represent the less challenging alignment scenario between the two. The connector network here does not have to overtake the responsibilities of the MT encoder completely (contrary to the ECD architecture). Now, the main purpose of the connector network is to take the ASR encoder output embeddings containing latent textual information and just transform them in such a way that they roughly match their counterparts in the MT encoder input text embedding space.

For these experiments, we keep the same training procedure, foundation models, and connector configurations as for the ECD architecture.

In contrast to the ECD architecture, the performance falloff is not as stark when decreasing the number of connector layers. The best model is once again utilizing the 6-layer STE connector, dipping slightly below the performance of the ECD configuration (see Section 5.4.1) with 44.7 and 44.8 BLEU on the `val` and `dev5` sets, respectively. The Q-Former maintains better results in contrast to the ECD architecture, further reinforcing the hypothesis that the mapping problem in this second architecture variant is a simpler one to that of ECD.

Table 5.6: Connector performance comparison for the ECED alignment architecture. Baseline ST model results from Section 5.3 are added for reference.

| ASR enc. | Connector | Connector layers | Queries | MT enc. | MT dec. | Trainable parameters | How2 BLEU ↑ | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | val | dev5 |
| Ebr. small | Q | 6 | 100 | Marian 6l | Marian 6l | 9.6M | 44.1 | 44.2 |
| Ebr. small | Q | 4 | 100 | Marian 6l | Marian 6l | 6.4M | 43.8 | 43.7 |
| Ebr. small | Q | 2 | 100 | Marian 6l | Marian 6l | 3.2M | 42.3 | 41.5 |
| Ebr. small | STE | 6 | - | Marian 6l | Marian 6l | 10.7M | **44.7** | **44.8** |
| Ebr. small | STE | 4 | - | Marian 6l | Marian 6l | 8.1M | 44.0 | 44.1 |
| Ebr. small | STE | 2 | - | Marian 6l | Marian 6l | 5.5M | 43.0 | 43.1 |
| E-Branchformer E2E baseline | | | | | | 38.5M | 45.6 | 45.2 |
| Cascade baseline | | | | | | 0 | 40.9 | 40.4 |

Here, all of the aligned models (both using the Q and STE connectors) outperform our cascade baseline. Though none of them surpass the E2E baseline system, it needs to be stated that we are obtaining almost comparable performance while only tuning a small module with less than a fourth of the parameter count of the E2E system. This further indicates that there is potential in the alignment approach to ST, and it remains to be seen as to what can be achieved by scaling up the foundation ASR and MT models.

## 5.5 ASR and MT model scaling effects

Having established that both the ECD and ECED alignment architectures are viable approaches for training speech translation systems, it is necessary to investigate the prospects of aligning also off-the-shelf (and possibly out-of-domain) pre-trained ASR and MT models in the same way. The hope is that increasing the size and capability of the foundation models will yield an ST performance increase. Ideally, the alignment process would also overcome any domain mismatch that would otherwise occur if one were to use the same foundation systems as parts of a cascade ST system[12].

A very important thing to note is that the size of the connector network is kept constant across all experiments. This is especially crucial in terms of the hidden size of the connector (which is kept at 256) because the off-the-shelf adapted models use hidden sizes of 512 or 768. The connectors are intentionally kept this small for a few reasons, mainly because it allows for shorter and more stable training runs (one training run is already around 13-15 hours). Also, the bottleneck nature of the connector does not seem to pose any problems concerning ST performance, as is shown by the experiments that follow in this section – even when the connector projects its output into an embedding space with more than twice the dimensions. It can, of course, be argued that a connector with a hidden size closer to that of the aligned MT model would yield better performance (and some side experiments we conducted indeed support this), however, we argue that such behavior is expected and the more interesting finding is that the connector can, in fact, be much smaller than either of the foundation models, without impeding ST performance.

Once again, we train all the systems with the same training hyper-parameters as in previous alignment experiments. The only exceptions are the two Whisper-small runs,

[12]This is also one of the reasons we do not build a cascade reference system using the chosen En-Pt T5 model, as its performance on How2 is already so impaired by domain mismatch that there is virtually no hope for the cascade system to yield good results.

where we lower the number of warm-up steps to 12000 due to training instability, and change the batch size to 48 with three gradient accumulation steps (resulting in an effective batch size of 144), because of memory constraints.

We train most models in this section in the ECD alignment configuration except for two instances, which are described later in this section. Experiments are conducted with both the Q and STE connectors, trying out all possible combinations between both E-Branchformer ASR encoders and MarianMT and T5 decoders. The results are shown in the first half of Table 5.7.

Table 5.7: Aligned ST system performance comparison for different combinations of scaled-up foundation ASR and MT models.

| ASR enc. | Connector | Connector layers | Queries | MT enc. | MT dec. | Trainable parameters | `How2` BLEU ↑ | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | `val` | `dev5` |
| Ebr. medium | Q | 6 | 128 | - | Marian 6l | 10.4M | 45.6 | 45.7 |
| Ebr. medium | Q | 6 | 128 | - | T5 12l | 10.5M | 46.8 | 47.5 |
| Ebr. small | Q | 6 | 128 | - | T5 12l | 9.7M | 44.5 | 44.4 |
| Ebr. medium | STE | 6 | - | - | Marian 6l | 12.0M | 46.0 | 46.3 |
| Ebr. medium | STE | 6 | - | - | T5 12l | 12.0M | 47.8 | 48.5 |
| Ebr. small | STE | 6 | - | - | T5 12l | 10.7M | 45.4 | 45.6 |
| Ebr. small | STE | 6 | - | T5 12l | T5 12l | 10.7M | 45.2 | 45.7 |
| Ebr. medium | STE | 6 | - | T5 12l | T5 12l | 12.0M | 47.5 | 48.0 |
| Whisper small | Q | 6 | 100 | - | T5 12l | 11.3M | 47.4 | 47.6 |
| Whisper small | STE | 6 | - | - | T5 12l | 13.3M | **48.2** | **48.9** |
| E-Branchformer E2E baseline | | | | | | 38.5M | 45.6 | 45.2 |
| Cascade baseline | | | | | | 0 | 40.9 | 40.4 |

Once again, the STE connector performs slightly better than the Q-Former, improving the BLEU score by one point on average when aligning the E-Branchformer small and the T5 decoder. In the E-Branchformer medium/MarianMT configuration, STE still offers a performance edge, only not as prevalent anymore. The results also suggest that increasing the size and capability of the ASR encoder yields a bigger performance improvement on average than when using a bigger MT decoder. However, more experiments would need to be done in this regard to confirm this hypothesis, as the chosen T5 model is perhaps not the most fair reference point to draw such conclusions. Of course, utilizing both a bigger ASR encoder and a bigger MT decoder yields the best translation results.

What is interesting about the T5 decoder in these experiments, is that the BLEU scores achieved by using this system in the aligned scenario far supersede the raw text translation scores on `How2` , as presented in Section 5.2. We argue that this is due to the connector network being able to serve as a domain adapter for the MT decoder, reinforcing the notion that the selected foundation ASR and MT models might not need to first be fine-tuned for the specific domain of the available ST data.

We repeat this experiment also with the Whisper-small.en model, which achieves the best WER on `How2` of all the ASR systems used in this work. As shown in the last two lines of Table 5.7, this model in combination with the same STE connector[13] yields the best performance among all trained models, achieving 48.2 and 48.9 BLEU points on the

[13]Here both the encoder from Whisper and the T5 decoder have a hidden size of 768, while the connector network still operates with 256-dimensional embeddings.

`val` and `dev5` sets, respectively, which is an improvement of over 9 BLEU points for the T5 model on the `dev5` set.

To further test the domain adaptation capabilities of the connector networks, we also train two models with the whole T5 model in the ECED configuration, as one might argue that a big part of the domain adaptation process is actually cutting off the T5 encoder. The T5 model was trained for translation utilizing a specific task prompt that has to be prepended to the encoder input, signaling the model to translate the input sentence, as is typical for the T5 model family[14]. Therefore when processing each batch, the tokenized version of the prompt ('translate English to Portuguese: ') is first embedded via the T5 encoder input embedding layer and then prepended to the output embeddings produced by the connector network, similarly to [50], and as shown earlier in Figure 4.5. This is done to more closely match the original operational context of the T5 model.

The expectation here would be that if the connector network is actually only trying to match the text embedding representation of the to-be-translated sentence at the input of the MT encoder, the domain adaptation would fail, as there would be virtually no distinction between the aligned scenario and the basic machine translation scenario in the same domain. However (refer to lines 7-8 in Table 5.7), both models with either of the two ASR encoders almost match the performances of their ECD counterparts. This suggests that the connector network is actually doing something more abstract, perhaps learning to steer the behavior of the translation model in a more nuanced way than just passing it transformed ASR embeddings with textual information, resulting in better, more in-domain results when decoding the final translated sentence. In other words, the connector network seems to be adding some additional abstract information to the text information encoded in the transformed embeddings, and this *fine steering* information allows the MT encoder to properly translate the original sentence, without the out-of-domain effect becoming an issue.

This same theme comes up again in Chapter 6, where it becomes apparent that this behavior of the connector networks is not always desirable and can probably even be regarded as a case of domain or task overfitting.

In conclusion, these experiments show that both alignment architectures represent general frameworks for aligning pre-trained ASR encoders with MT models, yielding better results as one scales up the selected foundation models, while the connector network can be kept small in size, allowing for quick and efficient training. Furthermore, we find that the connector networks are able successfully to serve as domain adapters, drastically improving the translation results even for out-of-domain MT systems.

## 5.6 Q-Former query count

Already in some preliminary experiments (using the initial subpar S2T ASR model), it became apparent that the number of queries used by the Q-Former connector plays a crucial role in relation to the aligned system performance. At first, those experiments suggested that around 100 or 128 queries should offer the best results for our experiments. Once the final foundation ASR and MT models were established, we once again tested the Q-Former performance in relation to the number of input queries.

We train the same baseline Q-Former alignment system with our E-Branchformer small and MarianMT small models, specifically in the ECD configuration. The same training

[14]Example encoder prompt: 'translate English to Portuguese: So long, and thanks for all the fish¡

setup and hyper-parameters as described at the beginning of Section 5.4 are used. The query numbers we test are 40, 60, 80, 100, 128 (this number was used in many of the preliminary experiments), and 150. See Table 5.8 for results.

Table 5.8: Q-Former performance comparison with regard to the number of queries used.

| ASR enc. | Connector | Connector layers | Queries | MT enc. | MT dec. | Trainable parameters | How2 BLEU ↑ | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | val | dev5 |
| Ebr. small | Q | 6 | 150 | - | Marian 6l | 9.6M | 43.2 | 42.9 |
| Ebr. small | Q | 6 | 128 | - | Marian 6l | 9.6M | 43.7 | 43.5 |
| Ebr. small | Q | 6 | 100 | - | Marian 6l | 9.6M | 44.0 | 43.9 |
| Ebr. small | Q | 6 | 80 | - | Marian 6l | 9.6M | 43.8 | 43.8 |
| Ebr. small | Q | 6 | 60 | - | Marian 6l | 9.6M | 43.3 | 43.1 |
| Ebr. small | Q | 6 | 40 | - | Marian 6l | 9.5M | 43.3 | 43.1 |
| Ebr. medium | Q | 6 | 128 | - | Marian 6l | 10.4M | 45.6 | 45.7 |
| Ebr. medium | Q | 6 | 128 | - | T5 12l | 10.5M | **46.8** | **47.5** |
| Ebr. small | Q | 6 | 128 | - | T5 12l | 9.7M | 44.5 | 44.4 |
| Ebr. medium | Q | 6 | 100 | - | Marian 6l | 10.4M | 45.8 | 45.4 |
| Ebr. medium | Q | 6 | 100 | - | T5 12l | 10.5M | 46.6 | 47.4 |
| Ebr. small | Q | 6 | 100 | - | T5 12l | 9.7M | 44.3 | 44.0 |

The results show that there is a clear sweet spot of around 100 queries that works best for our experiments when training on How2 using our E-Branchformer small and MarianMT models. Though it might seem that lowering the number of queries to 40 or 60 does not impact the ST performance as much as one would perhaps expect, later in Section 5.7, it is shown that lowering the number of queries to 60 and below comes with a significant performance hit when measured on utterances longer than 15 seconds.

In contrast to [54], we find that increasing the number of queries past a certain threshold does not yield better performance either. This is most likely caused by the fact that the How2 dataset skews significantly towards shorter utterances of around 5 seconds, and the higher number of queries might simply provide a needless amount of headroom that degrades the performance of the MT decoder. Intuitively, this could be thought of as the Q-Former not being able to learn to properly *mask* the unused queries, in a certain sense. Ultimately, more experiments should be conducted to better pinpoint the cause of this problem.

The number of queries used by the Q-Former should generally be informed by the average number of text tokens needed to represent the input utterance in the MT encoder embedding space. Models with larger and finer-grained vocabularies might benefit from a higher query count. This hypothesis is supported by a few experiments conducted with the T5 MT model, where 128 queries marginally outperform the 100 queries used in the experiments with the MarianMT small model.

## 5.7 Evaluating the connectors on different input lengths

As was discussed in Section 4.1.2, the Q-Former has a fundamental issue of not being able to handle audio embedding representation sequences of arbitrary lengths. Instead, it always maps them to a fixed-length sequence, whose length is determined by the number of queries used. This might not be that much of an issue if the foundation ASR model only operates on audio slices of fixed length as well. For example, Whisper models [38] always operate on audio inputs of exactly 30 seconds (if the audio is shorter, it gets padded

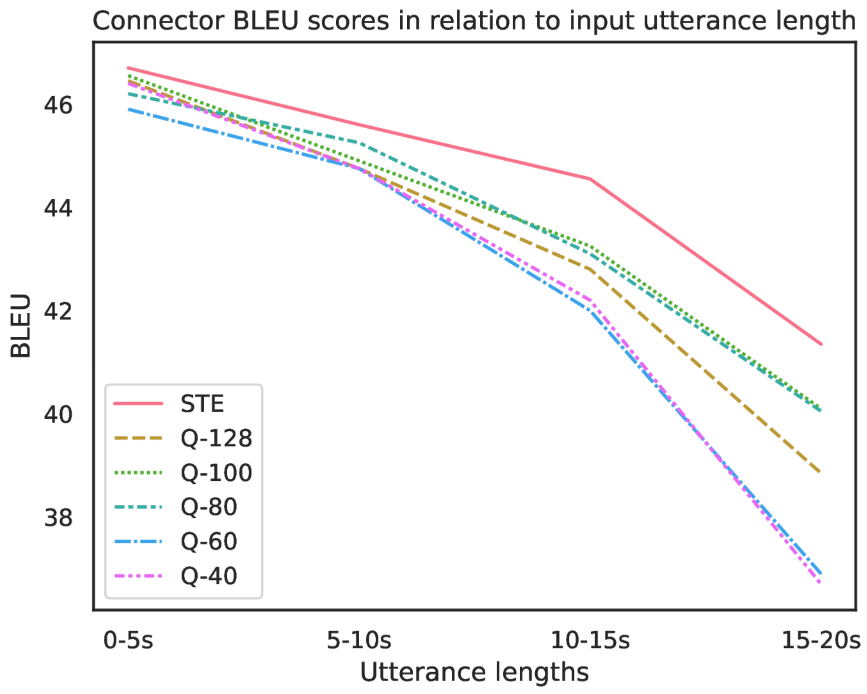


Figure 5.2: Performance comparison between the STE and Q connector networks with varying numbers of queries in relation to input utterance length. The final BLEU score is obtained by averaging BLEUs computed for both the `val` and `dev5` `How2` sets.

up to 30 seconds). In this scenario, the Q connector can potentially better learn to mask unused unused queries. However, for virtually all other cases it makes more sense for the connector network to produce variable-length output – that way, there is always headroom if the input audio sequence is longer, and, on the flip side, the model can be more efficient if the input utterance is short. On top of that, it could be argued that having the connector output length correspond monotonously to its input length, provides less room for confusion in the downstream language/MT model. Such errors could stem from some unexpected residual information being carried by queries that are not necessary to encode the whole utterance, or, on the contrary, simply from not having enough queries to represent a long input utterance properly.

From the experiments conducted in the previous sections, it is already quite apparent that the STE connector generally outperforms the Q connector. However, since the evaluation is always computed over the whole `val` or `dev5` set – and therefore aggregated over all possible input utterance lengths – these results cannot confirm or disprove our hypothesis that the STE connector is perhaps the superior architecture in terms of robustness to input utterance length.

We therefore also evaluate both the Q and STE connectors on four sub-splits of the test `How2` subsets. These sub-splits are simply filtered by lengths: 0 to 5 seconds, 5 to 10, 10 to 15, and 15 to 20 seconds and contain approximately 1100, 700, 220, and 60 utterances each, respectively. Because the majority of `How2` utterances is concentrated in the 0 to 10-second range, the last two splits are considerably smaller than the two made up of shorter utterances of under 10 seconds. However, we think that the results of this experiment show a clear trend that goes beyond any potential errors that could stem from the smaller amount of test data (or training data for that matter). To obtain the final BLEU value for each sub-split, the scores obtained for the `val` and `dev5` sets are averaged to better illustrate the overall performance decline trend of each evaluated system.

All results are obtained using the ECD models constructed using the E-Branchformer small and MarianMT small foundation models, as presented in previous sections.

As shown in Figure 5.2, the STE connector clearly maintains good performance in a much more consistent manner and falls off much slower than the Q-Former with longer utterances. The Q-Former on the other hand greatly suffers from not being able to properly represent longer utterances when only operating over 40 or 60 queries, dipping below 37 BLEU. This experiment also reinforces our previous finding from Section 5.6 that the sweet spot for the Q connector lies around 80 to 100 queries (at least for the MarianMT model). Slightly surprisingly, 128 queries also show degrading performance for longer utterances, perhaps suffering from the skewing of the `How2` dataset towards a greater amount of shorter training utterances. Alternatively, the explanation could stem from the fact that the MT model has – due to its 8000 sub-word unit vocabulary having relatively low granularity – simply not encountered enough training sequences of lengths greater than 128, further contributing to the performance degradation. This would also be supported by our other findings, that when used with the T5 model, 128 queries give better overall performance than 100.

Ultimately, it can be concluded that the STE architecture is a more robust and better-performing connector network among the two, both in terms of raw performance and handling variable-length sequences.

# Chapter 6

# Connector network pre-training

Inspired by BLIP-2 [25], the question arises, if there is a possibility to devise a pre-training procedure for the aligned system, specifically the connector module. Ideally, such an approach would either improve the performance of the final system or help speed up the training process, perhaps resulting in a reduced need for speech translation data.

The form and utility of the chosen pre-training approach depends not only on the entire aligned model architecture, but also on the properties of the chosen pre-trained models, especially the foundation language/MT model – it depends on whether the models are mono or multilingual, and whether they support different kinds of diverse language understanding tasks. For example, in [50] the aligned models are trained using a multitude of tasks that the selected language model is already able to perform, utilizing both ASR and ST objectives and data, which are combined with different ways of instructing the language model.

In this chapter, we explore two possible approaches to pre-training the connector network: one is loosely based on the concept of knowledge distillation and mainly inspired by the ideas presented in [13, 25]. The other one takes inspiration from [50] and involves retraining the MT decoder in a way that allows us to use it as a criterion of the alignment progress. The knowledge distillation-based approach (and mostly its shortcomings) is described in Section 6.1, and the ultimately much more successful end-to-end approach is then discussed in Section 6.2.

## 6.1 Knowledge distillation approach

The first set of pre-training experiments was based on knowledge distillation (KD) in the output hidden representation spaces of the connector network and the frozen MT encoder. These experiments can currently be regarded more as a study of an apparent failure, though with some positive outlooks and prospects for future work.

The goal of this knowledge distillation approach was to train the connector network in the ECD configuration in such a way, that its output embedding sequence (encoding the speech input) roughly matches the output embedding sequence of the MT encoder (encoding the reference text transcription), which is replaced by the connector. If the embeddings occupy similar representation spaces and there is high similarity between the connector outputs and MT encoder output sequences in general, the MT decoder should, in theory, behave similarly when prompted with either of them, provided that these output embeddings encode the same sentence.

For pre-training, we only use the English ASR data from `How2` (the translations are not needed), and alter the training in the following manner. First, the connector network transforms the output embeddings from the ASR encoder, giving us a matrix of hidden representations $\mathbf{C}_K$ of length $K$. Then, the original MT encoder is used to encode the ground truth text transcription for the input speech utterance, producing a sequence of MT encoder output embeddings $\mathbf{T}_L$ of length $L$. We then compute the knowledge distillation loss $L_{\text{KD}}$ between the two sequences in the following way:

$$L_{\text{KD}}(\mathbf{C}_K, \mathbf{T}_L) = MSE(\mathbf{c}, \mathbf{t}) + L_{\text{sim.}}(\mathbf{C}_K, \mathbf{T}_L), \tag{6.1}$$

where $\mathbf{c}$, $\mathbf{t}$ denote mean embedding vectors computed from $\mathbf{C}_K$ and $\mathbf{T}_L$, respectively, and $L_{\text{sim.}}$ is a custom sequence similarity loss function, designed to maximize the similarity between the connector outputs and MT encoder outputs. Note that no contrastive task is employed here, in contrast to [25], as using the original MT encoder embedding space as our target domain keeps the training from collapsing.

The $L_{\text{sim.}}$ function can be defined in several ways, depending on what assumptions are made about the nature of the desired similarity between the connector and MT encoder output embedding spaces. We conduct experiments with two $L_{\text{sim.}}$ variants, further denoted as $L_{\max}$ and $L_{\text{diag.}}$ :

$$L_{\max}(\mathbf{C}_K, \mathbf{T}_L) = -\sum_{k \in K} \max_{l \in L} S_c(\mathbf{C}_K[k], \mathbf{T}_L[l]), \tag{6.2}$$

$$L_{\text{diag.}}(\mathbf{C}_K, \mathbf{T}_L) = -\sum_{k=0}^{m} \sum_{l=0}^{d} S_c(\mathbf{C}_K[k], \mathbf{T}_L[k+l]) \frac{1}{d}, \tag{6.3}$$

where $S_c()$ is cosine similarity, $m$ denotes the diagonal limit computed as $\min(k-d, l-d)$, and $d$ is a hyper-parameter denoting the width of the diagonal.

Putting it into words, $L_{\max}$ simply tries to maximize the mutual information between the two embedding sequences, selecting the maximum similarity between each connector embedding and any of the MT encoder output embeddings. The $L_{\text{diag.}}$ loss function assumes that in an ideal alignment scenario, there should be a monotonous similarity relationship between the connector and MT encoder outputs along the main diagonal and attempts to loosely enforce it by widening the diagonal region, where the similarities are computed. Of course, there is room to alter the shape of the diagonal, even for enforcing a more specific contrastive objective at the same time by requiring the off-diagonal elements to be dissimilar, however, we leave such experiments for future work.

Unfortunately – though perhaps unsurprisingly – the experiments conducted using these knowledge distillation losses did not bring any positive results. The KD losses would, on one side, decrease during training, however, when switching the objective back to speech translation and attempting to fine-tune the models, the training either diverges or severe overfitting is observed every time.

Reflecting, because the MT encoder output embedding space is much more abstract than perhaps the input text embedding space, trying to only optimize such relatively ad-hoc similarity losses is perhaps not the best approach to pre-training the model. The structural and similarity assumptions made about this space might be inaccurate and too specific to provide a useful training objective. This is further supported by the fact that when taking an end-to-end-trained ECD architecture from the previous experiments and plotting a similarity matrix between the output embeddings of the connector and the outputs of the original

MT encoder, there seems to be no discernible diagonal similarity relationship between the two embedding sequences. In other words, when trained end-to-end, the connector seems to find a completely different solution to encoding the speech embeddings than the MT encoder does when encoding the text transcription.

An additional problem is that it might be necessary to involve the *response* feedback from the actual aligned MT decoder (and therefore the causal language modeling objective) in the pre-training as well. This alludes to the fact that the goal of the connector network is, in the end, to align the two feature spaces of the ASR and MT models, with an emphasis on the decoder behaving in the desired way. Perhaps some potential therefore lies in combining similar MT encoder output knowledge distillation objectives with some more specific end-to-end objectives as well. However, we leave that for future work.

For now, we refrain from attempting to pre-train the connector by only optimizing for similarity in the MT encoder output embedding space and switch to a different method, described in the next section.

## 6.2 End-to-end pre-training with MT decoder retraining

Since the knowledge distillation-oriented pre-training approach failed, we devise a different method, partly inspired by the training approach presented in [50]. Instead of trying to design an ad-hoc knowledge distillation loss function, operating in the abstract embedding space, we propose to push the pre-training towards a more end-to-end-oriented goal. This is done by *retraining* the MT decoder of the chosen MT model to serve as somewhat of a *loss-adapter*, which further processes the MT encoder outputs and *interprets* them. We design this pre-training approach to mainly accommodate simple MT models with perhaps only one source and target language and demonstrate the approach using our MarianMT model trained on `How2` .

Since one of the pre-training goals is to reduce the need for speech translation data when aligning the models, the pre-training approach becomes the following:

1. Select the target MT model, freeze the encoder, and reset the decoder weights. Replace the embedding layers and vocabulary of the decoder with that of the encoder.

2. Keeping the encoder frozen, train the decoder for identity on the source language (English), teaching it to repeat the encoder input.

3. Construct the ECED alignment model with this newly trained (English-English) identity MT model. Freeze the ASR encoder and the whole MT model.

4. Train the system for ASR in the source language (English), only optimizing the parameters of the connector.

5. Replace the MT decoder with the original target language (Portuguese) decoder and freeze it.

6. Fine-tune the connector network, now optimizing for the translation objective using speech translation data.

The main idea behind this pre-training method is to delegate the responsibility of the ad-hoc knowledge distillation loss to the much more general ASR objective. For the newly trained English MT decoder to produce the correct transcript of the input utterance, it has to receive the correct set of output MT encoder embeddings. The assumption is that similar decoder cross-attention inputs should produce similar outputs. Therefore, if the connector network supplies the MT encoder with a set of embeddings, one should be able to conclude that the ASR and MT models are aligned *if* the retrained MT decoder produces the correct English transcription. Finally, when switching back to the original Portuguese MT decoder – assuming that the alignment process had been successful – the decoder should be able to infer the correct translations from the aligned MT encoder outputs.

What is more, retraining the MT decoder to reproduce the MT encoder input is very simple, as the model has to only learn an identity mapping with virtually no information bottlenecks. For such tasks, there is no data shortage and the training converges very quickly, achieving near-perfect accuracy in a small number of training steps.

It is possible that this pre-training approach will not yield great ST results without any fine-tuning on ST data, after we switch to the original MT decoder, most likely due to over-fitting to the ASR pre-training task in step four of the procedure. However, we argue that even if some fine-tuning is required afterward, it would be a success if this pre-training approach helped alleviate the need for large amounts of speech translation data to produce well-performing end-to-end ST systems.

### 6.2.1 Pre-training experiments

First, we retrain the decoder of our in-domain MarianMT model to recreate the given English input. The training converges very quickly, achieving near-perfect BLEU in about 20 epochs, though we let the training run until convergence. Then, using this model, we construct the ECED alignment model with the STE connector and E-Branchformer small ASR encoder, and train the connector normally, only using the English ASR data. The dimensions of the STE connector are kept as in previous experiments – 6 layers, 4 attention heads, hidden size of 256.

Then, the ASR pre-training objective (step four) begins. The model is once again trained for a max of 70 epochs with, a batch size of 128, 15000 warm-up steps, a learning rate of $2e^{-4}$, and early stopping. The final pre-trained system achieves 22.5 and 21.7% WER on the `val` and `dev5` `How2` sets.

After the ASR training phase is finished, the English MT decoder is replaced with the original Portuguese one. At this point, the system is evaluated *without* any fine-tuning on speech translation data, achieving 12.5 and 12.9 BLEU on the `val` and `dev5` sets. While this result is not what we initially hoped for, it is obvious that the ASR pre-training procedure is much more productive in terms of aligning the ASR encoder with the MT encoder, contrary to the knowledge distillation approach.

It seems plausible, that the embedding distribution the STE connector learns to produce is reasonably useful for the ASR objective during step four. However, it does not completely match the distribution that would correctly prompt the original MT decoder to produce correct translations. As it was mentioned previously, this result supports the fact that the connector network can, in fact, be susceptible to task and domain overfitting, explaining both the result of this particular experiment and why it can perform domain adaptation for out-of-domain foundation models. Perhaps this is the place where some additional supervision or regularization in the embedding space would aid the final result while keeping the end-to-end ASR objective as the main criterion for the pre-training procedure.

Nevertheless, we proceed with the fine-tuning. The fine-tuning strategy utilizes three low-resource simulation `How2` splits as used in [21], cut from the 300-hour set: 17 hours, 51

Table 6.1: Comparison of BLEU scores achieved by the pre-trained and non-pre-trained E-Branchformer small + MarianMT small ECED/STE systems when fine-tuned using different amounts of ST data.

| Pre-trained | 0 hours | | 17 hours | | 51 hours | | 153 hours | | 300 hours | |
|---|---|---|---|---|---|---|---|---|---|---|
| | `val` | `dev5` | `val` | `dev5` | `val` | `dev5` | `val` | `dev5` | `val` | `dev5` |
| NO | - | - | 25.4 | 25.0 | 37.8 | 38.3 | 42.7 | 43.3 | 44.7 | 44.8 |
| YES | 12.5 | 12.9 | 38.8 | 39.5 | 41.5 | 41.4 | 43.4 | 44.0 | 44.4 | 44.5 |

hours, and 153 hours. The pre-trained connector is fine-tuned using all of these splits to test if the pre-training actually provides any further improvements when training the final aligned ST system. Additionally, the results are compared to the same exact system, only with a randomly initialized connector, that is, with no pre-training. Refer to Table 6.1 for the results.

Quite obviously, the pre-training procedure helps achieve considerably higher BLEU scores when fine-tuned using smaller amounts of data. Even for the 17-hour split, the pre-trained system achieves 38.8 and 39.5 BLEU on the `val` and `dev5` sets after fine-tuning, which is an improvement of more than 13 BLEU points across both sets when compared to the non-pre-trained system. While this result still does not beat our cascade baseline from Section 5.3, we think this result is only a starting point, given the fact that the pre-training procedure has a lot of potential for change and improvement. Additionally, fine-tuning the pre-trained system on the 51-hour set already yields results that surpass the cascade system, contrary to no pre-training.

As the fine-tuning split size grows to 153 and full 300 hours, the pre-training performance advantage slowly diminishes, finally offering no actual performance edge for the 300-hour subset, though it needs to be said that the pre-trained systems converge significantly faster. This particular model configuration and architecture may be reaching saturation on the full 300-hour `How2` dataset, which could explain the lack of performance improvement when utilizing pre-training.

### Pre-training with off-the-shelf models

We repeat the previous pre-training experiment with the E-Branchformer medium and T5 models. Contrary to the previous experiment, these models have not seen any of the `How2` data during training, making them ideal candidates for testing this pre-trained method in a more *real-life* scenario.

First, the decoder of the T5 model is stripped away and replaced with a small, 6-layer decoder with 4 attention heads and a hidden size of 256 (same architecture as the MarianMT small model). The tokenizer used is BPE-based and has an 8000 sub-word true-cased vocabulary. This new decoder is then once again retrained to repeat the input sentence to the T5 encoder – each input English sentence in this scenario is prepended with the T5 instruction prefix 'translate English to Portuguese: '. Therefore, the decoder has to only learn to repeat the sentence, that follows the instruction prefix.

After the decoder re-training is done, the same ECED/STE alignment architecture is constructed, using the T5 with the new decoder and the E-Branchformer medium models. The STE connector size is kept the same as in all previous experiments. The model is then trained for ASR using `How2` data, achieving an impressive 17.1 and 16.2% un-normalized

WER on `How2` . The re-trained decoder is then once again switched for the original T5 decoder.

When evaluating this system for ST after only doing the ASR pre-training, the previously mentioned task overfitting tendency of the connector network manifests clearly, as the BLEU scores without fine-tuning on ST data near absolute zero. Nonetheless, we proceed with the fine-tuning using the same low-resource simulation `How2` splits and report the results in Table 6.2. For reference, we also report BLEU scores for the same architecture when only trained using the low-resource splits, without any ASR pre-training.

Table 6.2: Comparison of BLEU scores achieved by E-Branchformer medium + T5 ECED/STE systems with and without pre-training, when fine-tuned using different amounts of ST data.

| Pre-trained | 0 hours | | 17 hours | | 51 hours | | 153 hours | | 300 hours | |
|---|---|---|---|---|---|---|---|---|---|---|
| | `val` | `dev5` | `val` | `dev5` | `val` | `dev5` | `val` | `dev5` | `val` | `dev5` |
| NO | - | - | 33.1 | 33.5 | 39.7 | 40.6 | 44.2 | 45.4 | 47.5 | 48.0 |
| YES | 0.0 | 0.0 | 37.5 | 37.2 | 41.1 | 41.8 | 44.5 | 45.7 | 47.3 | 47.9 |

Similarly to the previous experiment, the translation performance between the pre-trained and non-pre-trained models narrows with the size of the fine-tuning ST set. Surprisingly, the system achieves better scores for the same fine-tuning splits without pre-training, than in the previous case. This could be explained by the fact that the foundation ASR and MT systems are more powerful. On the other hand, the pre-trained system reaches worse performance for the 17-hour split than before (37.5 and 37.2 BLEU on the `val` and `dev5` splits), possibly stemming from the fact there is no overlap between the MT data used to train the T5 model and the ST data used to fine-tune it. However, there is still a clear performance gap between the models going in favor of the pre-trained system, suggesting that the pre-training approach is generalizable to arbitrary combinations of off-the-shelf pre-trained foundation models.

While this result is perhaps not what we hoped for, it still reinforces the merit of our proposed pre-training method and we argue that it should be possible to further improve and refine the approach in future work, for example by leveraging multiple pre-training objectives and using multi-lingual models, similarly to [50].

# Chapter 7

# Conclusions

In this work, we explored the possibilities of aligning pre-trained ASR and MT models to solve the task of spoken language translation. We did this by conducting experiments in two alignment scenarios, *Encoder-Connector-Decoder* and *Encoder-Connector-Encoder-Decoder*. In the ECD scenario, the connector maps the output embeddings of the foundation ASR encoder into the cross-attention input space of the selected MT decoder, overtaking the responsibilities of the MT encoder. In the ECED scenario, the connector instead injects its output into the input text embedding space of the MT encoder, leaving room for potential leveraging of any multi-task capabilities the model might offer. In both alignment architectures, the foundation ASR and MT models are frozen and only the connector network weights are optimized using a standard cross-entropy loss objective at the output of the MT decoder in the context of the ST task.

We additionally compared two connector network types: the Q-Former and the STE connector, which we used as an alternative. Our experiments find, that the STE connector is superior to the Q-Former in both performance and flexibility, stemming primarily from the variable-to-fixed-length mapping limitations of the Q-Former, caused by its utilization of a fixed set of trainable *queries* to represent the extracted information. This results in the Q-Former not being able to properly represent longer input sequences, impeding translation performance. The STE connector alleviates this by utilizing a subsampler frontend and adopting a regular transformer encoder architecture.

Our experiments show that the alignment approach is a viable framework for establishing new end-to-end speech translation systems, as most of our alignment systems outperform the baseline cascade system constructed using the same foundation models, almost matching the performance of our other end-to-end baseline ST system. We also find that scaling up the aligned ASR and MT models leads to universally better speech translation results, while the size of the connector network can remain constant, and relatively small, in contrast to the aligned foundation models.

The connectors also demonstrate good domain adaptation capabilities when used in conjunction with foundation models that are out-of-domain on the training data, as was the case for the T5 translation model used in our experiments. In that case, the best-aligned ST model improves the translation performance by more than 9 BLEU points on the `dev5` `How2` set, when compared to the `How2` evaluation result of the T5 system in the base MT scenario.

Lastly, we propose a connector pre-training approach aimed at reducing the need for ST data when aligning the systems. This approach involves re-training the original MT decoder to replicate the encoder input, subsequently allowing us to train the entire aligned system in the ECED configuration using the ASR objective and ASR data only. We find that his kind of pre-training yields better results than other more ‘vanilla‘ knowledge-distillation-based approaches we experiment with. The pre-training procedure improves the aligned ST performance when fine-tuned on low-resource simulation ST `How2` splits, allowing the E-Branchformer medium + STE + T5 pre-trained system to achieve 37.5 BLEU on the `val` set when fine-tuned using 17 hours of ST data, compared to the score of 33.1 of the same system without the pre-training. While this result does not outperform our cascade ST baseline, we argue that this result shows that there is potential in the pre-training method if further developed and refined.

# Bibliography

[1] Alayrac, J.; Donahue, J.; Luc, P.; Miech, A.; Barr, I. et al. Flamingo: a Visual Language Model for Few-Shot Learning. In: Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K. et al., ed. *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. 2022. Available at: http://papers.nips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177ccccbb411a7d800-Abstract-Conference.html.

[2] Baevski, A.; Zhou, Y.; Mohamed, A. and Auli, M. Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In: Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. and Lin, H., ed. *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. 2020. Available at: https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html.

[3] Bahdanau, D.; Cho, K. and Bengio, Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016.

[4] Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. et al. Language Models are Few-Shot Learners. In: Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. and Lin, H., ed. *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. 2020. Available at: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[5] Carmo, D.; Piau, M.; Campiotti, I.; Nogueira, R. and Lotufo, R. PTT5: Pretraining and validating the T5 model on Brazilian Portuguese data. *ArXiv preprint arXiv:2008.09144*, 2020.

[6] Chen, F.; Han, M.; Zhao, H.; Zhang, Q.; Shi, J. et al. *X-LLM: Bootstrapping Advanced Large Language Models by Treating Multi-Modalities as Foreign Languages*. 2023.

[7] Chen, G.; Zheng, Y.-D.; Wang, J.; Xu, J.; Huang, Y. et al. *VideoLLM: Modeling Video Sequence with Large Language Models*. 2023.

[8] Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G. et al. PaLM: Scaling Language Modeling with Pathways. *J. Mach. Learn. Res.*, 2023, vol. 24, p. 240:1–240:113. Available at: http://jmlr.org/papers/v24/22-1144.html.

[9] Dai, W.; Li, J.; Li, D.; Tiong, A. M. H.; Zhao, J. et al. InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning. In: Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M. et al., ed. *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. 2023. Available at: http://papers.nips.cc/paper_files/paper/2023/hash/9a6a435e75419a836fe47ab6793623e6-Abstract-Conference.html.

[10] Devlin, J.; Chang, M.-W.; Lee, K. and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Burstein, J.; Doran, C. and Solorio, T., ed. *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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, p. 4171–4186. Available at: https://aclanthology.org/N19-1423.

[11] Dong, L.; Xu, S. and Xu, B. Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, p. 5884–5888.

[12] Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X. et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. Available at: https://openreview.net/forum?id=YicbFdNTTy.

[13] Duquenne, P.-A.; Schwenk, H. and Sagot, B. *SONAR: Sentence-Level Multimodal and Language-Agnostic Representations*. arXiv, 2023. Available at: https://arxiv.org/abs/2308.11466.

[14] Esplà, M.; Forcada, M.; Ramírez Sánchez, G. and Hoang, H. ParaCrawl: Web-scale parallel corpora for the languages of the EU. In: Forcada, M.; Way, A.; Tinsley, J.; Shterionov, D.; Rico, C. et al., ed. *Proceedings of Machine Translation Summit XVII: Translator, Project and User Tracks*. Dublin, Ireland: European Association for Machine Translation, August 2019, p. 118–119. Available at: https://aclanthology.org/W19-6721.

[15] Gage, P. A new algorithm for data compression. *C Users J.* USA: R & D Publications, Inc., feb 1994, vol. 12, no. 2, p. 23–38. ISSN 0898-9788.

[16] Gulati, A.; Qin, J.; Chiu, C.; Parmar, N.; Zhang, Y. et al. Conformer: Convolution-augmented Transformer for Speech Recognition. In: Meng, H.; Xu, B. and Zheng, T. F., ed. *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*. ISCA, 2020, p. 5036–5040. Available at: https://doi.org/10.21437/Interspeech.2020-3015.

[17] Hono, Y.; Mitsuda, K.; Zhao, T.; Mitsui, K.; Wakatsuki, T. et al. *An Integration of Pre-Trained Speech and Language Models for End-to-End Speech Recognition*. 2023.

[18] Inaguma, H.; Kiyono, S.; Duh, K.; Karita, S.; Yalta, N. et al. ESPnet-ST: All-in-One Speech Translation Toolkit. In: Celikyilmaz, A. and Wen, T.-H., ed. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, July 2020, p. 302–311. Available at: https://aclanthology.org/2020.acl-demos.34.

[19] Junczys Dowmunt, M.; Grundkiewicz, R.; Dwojak, T.; Hoang, H.; Heafield, K. et al. Marian: Fast Neural Machine Translation in C++. In: Liu, F. and Solorio, T., ed. *Proceedings of ACL 2018, System Demonstrations*. Melbourne, Australia: Association for Computational Linguistics, July 2018, p. 116–121. Available at: https://aclanthology.org/P18-4020.

[20] Karita, S.; Soplin, N. E. Y.; Watanabe, S.; Delcroix, M.; Ogawa, A. et al. Improving Transformer-Based End-to-End Speech Recognition with Connectionist Temporal Classification and Language Model Integration. In: *Proc. Interspeech 2019*. 2019, p. 1408–1412.

[21] Kesiraju, S.; Sarvaš, M.; Pavlíček, T.; Macaire, C. and Ciuba, A. Strategies for Improving Low Resource Speech to Text Translation Relying on Pre-trained ASR Models. In: *INTERSPEECH 2023*. ISCA, August 2023. Available at: http://dx.doi.org/10.21437/Interspeech.2023-2506.

[22] Kim, K.; Wu, F.; Peng, Y.; Pan, J.; Sridhar, P. et al. E-Branchformer: Branchformer with Enhanced Merging for Speech Recognition. In: *IEEE Spoken Language Technology Workshop, SLT 2022, Doha, Qatar, January 9-12, 2023*. IEEE, 2022, p. 84–91. Available at: https://doi.org/10.1109/SLT54892.2023.10022656.

[23] Kudo, T. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In: Gurevych, I. and Miyao, Y., ed. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, p. 66–75. Available at: https://aclanthology.org/P18-1007.

[24] Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A. et al. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In: Jurafsky, D.; Chai, J.; Schluter, N. and Tetreault, J., ed. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, p. 7871–7880. Available at: https://aclanthology.org/2020.acl-main.703.

[25] Li, J.; Li, D.; Savarese, S. and Hoi, S. C. H. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In: Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S. et al., ed. *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. PMLR, 2023, vol. 202, p. 19730–19742. Proceedings of Machine Learning Research. Available at: https://proceedings.mlr.press/v202/li23q.html.

[26] Liu, P. J.; Saleh, M.; Pot, E.; Goodrich, B.; Sepassi, R. et al. Generating Wikipedia by Summarizing Long Sequences. *CoRR*, 2018, abs/1801.10198. Available at: http://arxiv.org/abs/1801.10198.

[27] Lopes, A.; Nogueira, R.; Lotufo, R. and Pedrini, H. Lite Training Strategies for Portuguese-English and English-Portuguese Translation. In: *Proceedings of the Fifth Conference on Machine Translation*. Online: Association for Computational Linguistics, November 2020, p. 833–840. Available at: https://www.aclweb.org/anthology/2020.wmt-1.90.

[28] Lyu, C.; Wu, M.; Wang, L.; Huang, X.; Liu, B. et al. *Macaw-LLM: Multi-Modal Language Modeling with Image, Audio, Video, and Text Integration*. 2023.

[29] Muennighoff, N.; Wang, T.; Sutawika, L.; Roberts, A.; Biderman, S. et al. Crosslingual Generalization through Multitask Finetuning. In: Rogers, A.; Boyd Graber, J. and Okazaki, N., ed. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, July 2023, p. 15991–16111. Available at: https://aclanthology.org/2023.acl-long.891.

[30] Niehues, J.; Cattoni, R.; Stüker, S.; Negri, M.; Turchi, M. et al. The IWSLT 2019 Evaluation Campaign. In: Niehues, J.; Cattoni, R.; Stüker, S.; Negri, M.; Turchi, M. et al., ed. *Proceedings of the 16th International Conference on Spoken Language Translation*. Hong Kong: Association for Computational Linguistics, November 2-3 2019. Available at: https://aclanthology.org/2019.iwslt-1.1.

[31] OpenAI et al. *GPT-4 Technical Report*. 2023.

[32] Papineni, K.; Roukos, S.; Ward, T. and Zhu, W.-J. Bleu: a Method for Automatic Evaluation of Machine Translation. In: Isabelle, P.; Charniak, E. and Lin, D., ed. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, p. 311–318. Available at: https://aclanthology.org/P02-1040.

[33] Park, D. S.; Chan, W.; Zhang, Y.; Chiu, C.-C.; Zoph, B. et al. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In: *Interspeech 2019*. ISCA, September 2019. Available at: http://dx.doi.org/10.21437/Interspeech.2019-2680.

[34] Peng, Y.; Dalmia, S.; Lane, I. R. and Watanabe, S. Branchformer: Parallel MLP-Attention Architectures to Capture Local and Global Context for Speech Recognition and Understanding. In: Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G. et al., ed. *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. PMLR, 2022, vol. 162, p. 17627–17643. Proceedings of Machine Learning Research. Available at: https://proceedings.mlr.press/v162/peng22a.html.

[35] Peng, Y.; Kim, K.; Wu, F.; Yan, B.; Arora, S. et al. A Comparative Study on E-Branchformer vs Conformer in Speech Recognition, Translation, and Understanding Tasks. In:. August 2023, p. 2208–2212.

[36] Pham, N.-Q.; Nguyen, T.-S.; Ha, T.-L.; Hussain, J.; Schneider, F. et al. The IWSLT 2019 KIT Speech Translation System. In: Niehues, J.; Cattoni, R.; Stüker, S.; Negri, M.; Turchi, M. et al., ed. *Proceedings of the 16th International Conference on Spoken Language Translation*. Hong Kong: Association for Computational Linguistics, November 2-3 2019. Available at: https://aclanthology.org/2019.iwslt-1.3.

[37] Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G. et al. Learning Transferable Visual Models From Natural Language Supervision. In: Meila, M. and Zhang, T., ed. *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. PMLR, 2021, vol. 139, p. 8748–8763. Proceedings of Machine Learning Research. Available at: http://proceedings.mlr.press/v139/radford21a.html.

[38] Radford, A.; Kim, J. W.; Xu, T.; Brockman, G.; McLeavey, C. et al. Robust Speech Recognition via Large-Scale Weak Supervision. In: Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S. et al., ed. *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. PMLR, 2023, vol. 202, p. 28492–28518. Proceedings of Machine Learning Research. Available at: https://proceedings.mlr.press/v202/radford23a.html.

[39] Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D. et al. Language Models are Unsupervised Multitask Learners, 2019.

[40] Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S. et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 2020, vol. 21, p. 140:1–140:67. Available at: http://jmlr.org/papers/v21/20-074.html.

[41] Sanabria, R.; Caglayan, O.; Palaskar, S.; Elliott, D.; Barrault, L. et al. How2: A Large-scale Dataset For Multimodal Language Understanding. In: NeurIPS. *Proceedings of the Workshop on Visually Grounded Interaction and Language (ViGIL)*. 2018. Available at: http://arxiv.org/abs/1811.00347.

[42] Su, Y.; Lan, T.; Li, H.; Xu, J.; Wang, Y. et al. PandaGPT: One Model To Instruction-Follow Them All. In: Hazarika, D.; Tang, X. R. and Jin, D., ed. *Proceedings of the 1st Workshop on Taming Large Language Models: Controllability in the era of Interactive Assistants!* Prague, Czech Republic: Association for Computational Linguistics, September 2023, p. 11–23. Available at: https://aclanthology.org/2023.tllm-1.2.

[43] Synnaeve, G.; Xu, Q.; Kahn, J.; Likhomanenko, T.; Grave, E. et al. End-to-End ASR: from Supervised to Semi-Supervised Learning with Modern Architectures. In: *ICML 2020 Workshop on Self-supervision in Audio and Speech*. 2020. Available at: https://openreview.net/forum?id=OSVxDDc360z.

[44] Team, N.; Costa jussà, M. R.; Cross, J.; Çelebi, O.; Elbayad, M. et al. *No Language Left Behind: Scaling Human-Centered Machine Translation*. 2022.

[45] Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A. et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023.

[46] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L. et al. Attention is All you Need. In: Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, vol. 30. Available at: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[47] Vydana, K. H.; Karafiát, M.; Žmolíková, K.; Burget, L. and Černocký, J. Jointly Trained Transformers Models for Spoken Language Translation. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE Signal Processing Society, 2021, p. 7513–7517. ISBN 978-1-7281-7605-5. Available at: https://www.fit.vut.cz/research/publication/12522.

[48] Wang, C.; Tang, Y.; Ma, X.; Wu, A.; Okhonko, D. et al. Fairseq S2T: Fast Speech-to-Text Modeling with Fairseq. In: Wong, D. and Kiela, D., ed. *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*. Suzhou, China: Association for Computational Linguistics, December 2020, p. 33–39. Available at: https://aclanthology.org/2020.aacl-demo.6.

[49] Wang, C.; Wu, A.; Pino, J.; Baevski, A.; Auli, M. et al. Large-Scale Self- and Semi-Supervised Learning for Speech Translation. In: Hermansky, H.; Cernocký, H.; Burget, L.; Lamel, L.; Scharenborg, O. et al., ed. *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*. ISCA, 2021, p. 2242–2246. Available at: https://doi.org/10.21437/Interspeech.2021-1912.

[50] Wang, M.; Han, W.; Shafran, I.; Wu, Z.; Chiu, C. et al. SLM: Bridge the Thin Gap Between Speech and Text Foundation Models. In: *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2023, Taipei, Taiwan, December 16-20, 2023*. IEEE, 2023, p. 1–8. Available at: https://doi.org/10.1109/ASRU57964.2023.10389703.

[51] Xiong, R.; Yang, Y.; He, D.; Zheng, K.; Zheng, S. et al. On Layer Normalization in the Transformer Architecture. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. PMLR, 2020, vol. 119, p. 10524–10533. Proceedings of Machine Learning Research. Available at: http://proceedings.mlr.press/v119/xiong20b.html.

[52] Xue, L.; Constant, N.; Roberts, A.; Kale, M.; Al Rfou, R. et al. MT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In: Toutanova, K.; Rumshisky, A.; Zettlemoyer, L.; Hakkani Tur, D.; Beltagy, I. et al., ed. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, p. 483–498. Available at: https://aclanthology.org/2021.naacl-main.41.

[53] Yan, B.; Dalmia, S.; Higuchi, Y.; Neubig, G.; Metze, F. et al. CTC Alignments Improve Autoregressive Translation. In: Vlachos, A. and Augenstein, I., ed. *Proceedings of the 17th Conference of the European Chapter of the Association for*

*Computational Linguistics*. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, p. 1623–1639. Available at: https://aclanthology.org/2023.eacl-main.119.

[54] Yu, W.; Tang, C.; Sun, G.; Chen, X.; Tan, T. et al. *Connecting Speech Encoder and Large Language Model for ASR*. 2023.

[55] Zhang, B.; Haddow, B. and Sennrich, R. Revisiting End-to-End Speech-to-Text Translation From Scratch. In: Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G. et al., ed. *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. PMLR, 2022, vol. 162, p. 26193–26205. Proceedings of Machine Learning Research. Available at: https://proceedings.mlr.press/v162/zhang22i.html.

[56] Zhang, H.; Li, X. and Bing, L. Video-LLaMA: An Instruction-tuned Audio-Visual Language Model for Video Understanding. In: Feng, Y. and Lefever, E., ed. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Singapore: Association for Computational Linguistics, December 2023, p. 543–553. Available at: https://aclanthology.org/2023.emnlp-demo.49.

# Appendix A

# Contents of the enclosed storage unit

- `xsedla1h_thesis.pdf` – The final `.pdf` version of this thesis.
- `xsedla1h_thesis.zip` – Contains the LaTeX source code files for this thesis.
- `huggingface_asr` – A folder containing the Hugging Face extension repository with all the source code files.
- `README.md` – A file documenting the code and containing further instructions for using the scripts.