

Parameter-Efficient Fine-Tuning of Llama-Based Models for Multilingual Translation of Short Industrial UI Text

Weronika Górak

*Department of Advanced Computing Sciences
Faculty of Science and Engineering
Maastricht University
Maastricht, The Netherlands*

Abstract—Compact, openly available Llama 2 7B derivatives are evaluated for their ability to translate short, ambiguous user-interface strings on a single 16 GB GPU and with only 2–3k translation examples per language. Two language-specific checkpoints - TowerBase-7B (English-German/Chinese) and SambaLingo-Serbian-7B (English/German-Serbian) - undergo 4-bit QLoRA fine-tuning with low-rank adapters. A standard zero-shot regimen is contrasted with a “hybrid” approach that embeds 0–5 few-shot demonstrations per training prompt.

On Aucos’s in-house user interface (UI) corpus, zero-shot fine-tuning yields the largest domain gains, boosting BLEURT while preserving fluency. The hybrid strategy retains demonstration awareness at inference, yet a smaller effective training set negates its advantage; with only a few thousand lines, additional data outweighs the benefits of incorporated few-shot examples. After adaptation, even 1-2 word labels translate reliably, and remaining errors stem mainly from German compound segmentation and Cyrillic–Latin mixing in Serbian.

Index Terms—UI Text Translation, Neural Machine Translation, Large Language Models, Transformer Architecture, Few-shot Prompting, In-context Learning, QLoRA

I. INTRODUCTION

Accurate translation of user interface (UI) text is a crucial part of the software localization process: a mistranslated warning or button label on a chemical-surface production line can compromise both safety and productivity. Yet UI strings are notoriously mistranslated by the available translators, as they are extremely short (often one or two tokens), arrive without surrounding context, and contain domain-specific jargon [1]. Take as an example the word “Run”: in the context of a UI we might mean running a program on our device, however, how does a translator know that we do not mean run, as in the sports activity? That is why context is crucial for UI text translations.

The transformer architecture established the modern baseline for NMT by modeling long-range dependencies efficiently

[2]. Building on this, mBART-50 employed denoising pre-training over 50 languages and achieves strong zero-shot transfer to low-resource pairs [3]. Meta’s No Language Left Behind (NLLB) scales the idea to 200 languages through a mixture-of-experts Transformer¹, setting new BLEU records on FLORES-200 while remaining fully open source [4].

Despite the recent advancements in NMT, the models trained for machine translation are not suitable for the tasks of UI text translation for a few, main reasons:

- **Data scarcity.** Despite having many translation data for the common language pairs, there are not enough high-quality, domain-specific parallel corpora for UI text, making the training process difficult.
- **Placeholder and tag handling.** Labels often contain variables such as “%1” or HTML-style markup. Generic systems either copy these tokens to the wrong position or drop them entirely, breaking runtime rendering.
- **Case and punctuation sensitive glossaries.** UI style guides reserve title case for menu items and sentence case for tooltips. Standard NMT, trained on long-form prose, ignores such casing conventions.
- **Low tolerance for terminology drift.** In process-control software a single term (Fahrwagen, Transportwagen, Hoist) must be rendered identically across hundreds of screens to avoid operator confusion. NMT’s tendency to paraphrase harms this consistency.

Recent open-source large language models (LLMs) target those gaps to some extent. TowerBase-7B, a Llama 2 7B continuation trained on 20B multilingual tokens, surpasses larger generic models on WMT benchmarks while retaining decoder-only architecture for prompt conditioning [5]. SambaLingo-Serbian-7B shows that language-specialist continuations of Llama 2 7B can gain ≈ 8 chrF on FLORES over generic baselines [6].

This thesis was prepared in partial fulfilment of the requirements for the Degree of Bachelor of Science in Data Science and Artificial Intelligence, Maastricht University. Supervisor(s): Konstantia Zarkogianni

¹Mixture-of-experts (MoE) Transformer is an architecture where, instead of using the same feedforward layer for every token in a layer, the model selectively routes tokens through a subset of specialized sub-networks, called experts.

Such checkpoints are amenable to parameter-efficient adaptation: LoRA inserts low-rank adapters with negligible inference cost [7], and QLoRA combines 4-bit quantisation with LoRA to fine-tune 7-B-parameter models on a single 16 GB GPU without quality loss [8]. Empirical evidence shows domain-tuned LLMs can outperform GPT-3.5/4 on specialised translation tasks when adequate in-domain data are available [9].

This research will be conducted in partnership with Aucos - a company, which works on automating surface treatment plants and delivers process optimization solutions to large industrial clients all around the world. Having many international clients, there is a need to translate their software to multiple languages, such as Chinese, Serbian and German, among others. The focus of this study will be on utilizing state-of-the-art transformer-based models, exploring advanced fine-tuning techniques, and evaluating the impact of incorporating in-context learning examples into the fine-tuning process as proposed by M. Alves et al. [10] in order to include context in the translation prompt, possibly improving the performance of a traditionally fine-tuned LLM. Given the technical nature of the UI text, the research also aims to investigate how effectively pre-trained language models can adapt to domain-specific tasks with limited data and how well they perform depending on the length of the text.

A major consideration in this research is the computational complexity involved in fine-tuning large-scale transformer models, such as Llama 2 7B. These models require substantial computational resources, particularly for tasks that involve adapting them to specific domains. Techniques such as low-rank adaptation (LoRA) or quantisation will be explored to address these challenges and make the fine-tuning process more feasible.

The findings of this thesis aim to contribute insights into optimizing machine translation systems for industrial environments with minimal context, benefiting the broader fields of AI-driven translation and domain-specific NLP applications.

RESEARCH QUESTIONS

- 1) How does incorporating 1–5 in-context UI translation examples during both fine-tuning and evaluation affect the COMETKiwi-22 and BLEURT scores of TowerBase-7B (English-German, English-Chinese) and SambaLingo-Serbian-7B (English/German-Serbian), compared to the same models fine-tuned and evaluated only with zero-shot examples?
- 2) How does UI-string length affect the translation quality (COMETKiwi-22, BLEURT) of the fine-tuned TowerBase-7B and SambaLingo-Serbian 7B models?
- 3) How do translation metrics (COMETKiwi-22, BLEURT) differ between high-resource (German), mid-resource (Chinese), and low-resource (Serbian)

language pairs when using the same LoRA-finetuned LLaMA-2 7B checkpoint versus the language-specialized SambaLingo-Serbian 7B?

- 4) To what extent does fine-tuning a continuously pre-trained language-specific model on UI translation data improve performance (COMETKiwi-22, BLEURT) over the baseline models - TowerBase-7B for English-German and English-Chinese translations, and NLLB for English/German-Serbian translations?

II. METHODS

A. Dataset

The dataset provided by Aucos consists of Qt TS translation files from various projects undertaken by the company. The .ts extension files are XML-based, and they are used in conjunction with the Qt Linguist app to facilitate the translation process. After extracting the translations from the .ts files, they were transferred into an SQL Postgres database. Later, data cleaning was performed: getting rid of the "Context" column, removing duplicates and rows with missing entries. In Table I you can find example translations in a tabular format, before they were cleaned.

TABLE I
EXAMPLE OF TRANSLATIONS IN A TABULAR FORMAT

Context	Source	Target
AddAction	Add	Hinzufügen
AnalogDevice	Set	Sollwert
FaultsView	Warning	Warnung
Devices	Upper warning limit	Obere Warngrenze

The "Context" column (Table 1) represents the location of the text in the UI, however, many source-target pairs repeat over different locations, which creates duplicates in the data, just with different "Context" field. That, on top of the fact that there is a very large number of locations, does not allow the model to get insightful information from this column, thus, when fine-tuning the model, this column will be omitted (before doing the data cleaning, in order to avoid having duplicates).

The translations in the files are mostly English based, meaning that the text will be translated from English, the only exception is the Serbian dataset where some of the source texts were German. The target languages which are available, are German, Chinese and Serbian. The English-German dataset consists of 1,934 entries, English-Chinese - 2,599 entries, and English/German-Serbian consists of 3,054 entries. After the data cleaning, all data was stored in the .json file format.

B. Model Selection Process

While choosing the models to be fine-tuned, the following factors were taken into account:

- 1) **Baseline quality** — zero-shot or minimally tuned translation accuracy on our English-German, Serbian, and Chinese UI benchmark.
- 2) **Language coverage** — confirmed tokenizer support and pre-training exposure to the target languages.
- 3) **Domain or task specialization** — whether the checkpoint was explicitly trained for machine translation.
- 4) **Architecture** — decoder-only (autoregressive) versus encoder-decoder; the former is required for in-context demonstration during fine-tuning and inference.
- 5) **Computational footprint** — parameter count and GPU memory needed under 4-bit weights plus LoRA adapters (QLoRA set-up).

In the end it was decided, that the TowerBase-7B model would be used for two out of three of our languages - German and Chinese. It is a decoder-only model and it achieved the most promising baseline performance on our data for the English-German and English-Chinese translation tasks (TowerBase-7B vs. NLLB). The models compared were the TowerBase-7B and NLLB 3.3B models, as they both support 2 or 3 of our languages, have different architectures (decoder-only vs. encoder-decoder) and are the best performing, open-source, machine translation models. Their baseline performance is shown in Table II. For en-de and en-zh, each model is better in terms of one of the metrics; therefore, the TowerBase-7B was chosen due to its prompting abilities. Autoregressive models are the only ones that allow us to incorporate in-context learning examples into the fine-tuning and provide context to the prompt. TowerBase-7B is a model made specifically for translation-related tasks, so it was possible to get the chosen metrics; BLEURT and COMETKiwi-22, without the model hallucinating. In addition, it comfortably fits on a single, 16 GB GPU under the QLoRA set-up, making large-scale experimentation feasible even on modest hardware.

As for Serbian - only the NLLB model could produce translations without hallucinations; however, seeing the promising performance of the Llama 2 7B based model - TowerBase-7B - on translations, combined with the fact that the SambaLingo-Serbian-Base is specifically made for the Serbian language, compared to a model that is very broad and supports 200 languages (NLLB) on 3.3B parameters, it was decided that this model will be deployed for fine-tuning, and the NLLB model’s baseline performance will be used for comparison.

TABLE II
BASELINE PERFORMANCE OF THE TOWER-BASE 7B AND NLLB 3.3B
MODELS ON AUCOS DATA IN TERMS OF THE BLEURT AND
COMETKIWI-22 METRICS.

Language Pair	COMET		BLEURT	
	Tower	NLLB	Tower	NLLB
en-de	75.9564	77.8987	-0.1685	-0.2560074
en-sr	-	72.7736	-	0.548201
en-zh	72.5524	71.3316	0.70373	0.727788

C. Llama Architecture

Both the models chosen for the experiments are based on the Llama 2 7B model. Llama 2 7B is a state-of-the-art LLM developed by Meta. It belongs to the second generation of the Llama family and has been pre-trained to generate text, making it a highly capable model for various NLP tasks. The 7B parameter version strikes a balance between performance and computational efficiency, making it suitable to be adapted for many practical applications, including translation.

Llama 2 7B is an auto-regressive language model, which means it is trained to predict the next token of a textual sequence. It uses an optimized version of the transformer architecture, and the tokenizer has a vocabulary consisting of 32K tokens.

The model deploys a standard dense transformer architecture [2]. In order to improve inference efficiency, Grouped-Query Attention (GQA) with 8 key-value heads is used. To prevent self-attention between different documents (in this case - source-target pairs) within the same sequence, an attention mask is used [11].

Llama 2 7B was pre-trained on roughly two trillion tokens of text drawn entirely from publicly available sources - a mix of CommonCrawl and other large web corpora, Wikipedia, Project Gutenberg and other books, academic papers (e.g., arXiv), software repositories such as GitHub, and cleaned parallel/multilingual data sets collected up to September 2022 [11]. The corpus was deduplicated, filtered for quality (e.g., language ID, perplexity thresholds, toxicity heuristics) and intentionally excluded any user data, giving the model broad coverage across domains while respecting privacy constraints².

D. TowerBase-7B

TowerBase-7B is a translation-centric continuation of the Llama 2 7B model. The model was optimized for high-quality translations across multiple language pairs by training the base model on a multilingual dataset consisting of 20 billion tokens. The languages for which it extends the model’s capabilities are: German (de), French (fr), Dutch (nl), Italian (it), Spanish (es), Portuguese (pt), Korean (ko), Russian (ru), and Chinese (zh) [5].

The model is first pre-trained on a broad multilingual corpus, enabling it to capture the syntax and semantics of individual languages as well as the relationships among them. It is then further exposed to large parallel-text datasets, which sharpen its cross-lingual alignments and teach nuanced correspondences between source and target sentences. This two-stage regimen - multilingual pre-training followed by parallel fine-tuning - makes TowerBase-7B particularly adept at context-dependent translations and idiomatic expressions

²<https://originality.ai/blog/meta-llama-2-statistics>

[5].

TowerBase-7B reaches COMET scores of 0.828–0.838 and BLEU scores of 25.6–26.4, compared to Llama 2 7B’s COMET scores around 0.79–0.81 and BLEU scores of 20.8–22.8. These improvements are robust across both sentence-level and document-level tasks, and notably, TowerBase-7B even outperforms the larger Llama 2 13B model in several translation benchmarks.

E. SambaLingo-Serbian-Base 7B

SambaLingo-Serbian-Base 7B is a bilingual Serbian and English model, which was built on top of Llama 2 7B in order to provide support for the Serbian language. The base model was additionally trained on 17 billion tokens from the Serbian split of the Cultura-X dataset. Additionally, as suggested by Csaki et al. 25% of the training data was English in order for the model to not lose its English capabilities [12]. The model achieved state-of-the-art evaluation results in perplexity and the FLORES-200 translation task. The authors have managed to extend the vocabulary of the base llama model from 32,000 to 57,000 tokens by adding around 25,000 non-overlapping tokens from Serbian [6].

On FLORES-200, SambaLingo-Serbian 7B exceeds the base Llama-2 7B by +8.1 chrF for en to sr and +7.2 chrF³ for sr to en; perplexity on held-out Serbian Wikipedia falls from 2.7 to 1.4 (48%). Against other open models (XGLM-7.5 B, BLOOM-7 B) it is state-of-the-art across Serbian evaluation suites [6].

F. Adaptation Strategies

Research by Zheng et al. shows that fine-tuning large language models for domain-specific machine translation significantly outperforms general-purpose models, achieving superior performance compared to advanced models such as GPT-3.5 and GPT-4o [9]. Therefore, finetuning techniques had to be explored in order to achieve the highest possible performance. One big constraint was computational resources - the biggest GPU available had a memory of 16B, which cannot handle models which are larger than 8B parameters. Techniques such as quantization can be very effective for inference, however, if one wants to fine-tune an LLM, more memory footprint reduction has to be done in order to perform the task successfully. Here, the methods for finetuning the chosen models will be explained, as well as the evaluation metrics that were picked.

1) *QLoRA fine-tuning*: QLoRA finetuning combines two techniques for reducing the memory footprint - quantizing the model’s parameters to 4-bit precision and the LoRA (low rank adaptation) method. QLoRA can reduce the average memory requirements of finetuning a 65B parameter model from >780GB of GPU memory to <48GB without degrading

the runtime or predictive performance compared to a 16-bit fully finetuned baseline [8].

a) *Quantization*: Quantization is the process of discretizing an input from a representation that holds more information to a representation with less information [8]. In simpler words, it means taking a data type with more bits, for example 32-bit floats, and converting it to fewer bits, for example 8-bit Integers.

b) *LoRA - Low-Rank Adaptation*: LoRA is a parameter-efficient fine-tuning technique that lets us adapt an LLM without touching (or storing gradients for) the billions of original weights. The intrinsic-rank hypothesis behind LoRA argues that the task-specific update for a pre-trained model often has low numerical rank, so it can be expressed in a subspace that is dramatically smaller than the weight matrix itself.

Formulation Let the frozen weight matrix of a linear layer be $W \in \mathbb{R}^{d \times r}$. Instead of learning a dense update $\Delta W \in \mathbb{R}^{d \times r}$, LoRA introduces two thin factors $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{d \times r}$ with rank $r \ll d$. During training the effective weight is

$$\widetilde{W} = W + \alpha \frac{BA}{r},$$

where α is a small scaling factor (typically 8–32) that keeps the update numerically stable at the start of fine-tuning. Because r is tiny, LoRA exposes only $\mathcal{O}(r(d+r))$ trainable parameters per layer, often $\leq 0.5\%$ of a 7B model, slashing GPU memory for gradients, optimizer states, and checkpoints by more than an order of magnitude [7].

Integration into Transformers In practice the low-rank adapters are inserted only in the weight matrices that dominate memory and FLOPs:

- 1) The query, key, and value projections of multi-head attention.
- 2) The up and down projection in the MLP block.

All other parameters (layer-norms, bias terms, position encodings) remain frozen.

Inference and Checkpoint Merging After training, A and B can be merged into the frozen weights once and for all:

$$W = W + \alpha \frac{BA}{r},$$

Inference then proceeds on the usual dense weights with zero runtime overhead, and the LoRA factors themselves can be discarded, yielding a footprint identical to the original base model.

Hyperparameters

- **Rank (r)**: Determines the matrix size. Specifically, it represents the number of independent columns in A and the number of independent rows in B .

³chrF is a metric that uses the F-score statistic for character n-gram matches.

- **Alpha (α):** a scaling factor for updates. Alpha directly impacts the adapters contribution and is often set to 1 or 2 times the rank value.
- **Dropout:** A small dropout (≈ 0.05) on BA is often used to avoid over-fitting when r is large.

Strengths and Limitations LoRA delivers substantial memory and storage savings and, unlike adapter-based methods that add extra layers, imposes no latency penalty at inference once the adapters are merged. However, because the update is constrained to rank r , LoRA can under-perform full fine-tuning on highly specialized tasks that require large departures from the pre-trained representation. Choosing r therefore trades off efficiency and head-room.

G. Data Division

A few-shot prompt is a single input sequence that embeds the task description, the new query, and a small set (usually 1–5) of fully worked examples illustrating the desired input-to-output mapping. During inference, by reading the examples in the prompt, the language model infers the pattern it should follow and generates an answer that parallels those demonstrations without altering its internal weights [13].

For each target language, two fine-tunings were performed: one solely on zero-shot prompts ("normal" fine-tuning) and the other one on 0-5 shot prompts ("hybrid" fine-tuning). The data was divided in the following way for the zero-shot data: 70% - training, 15% - validation and 15% for testing. The following prompt format was used for the "normal" finetuning:

```
<s> You are translating text for
a user interface at a chemical
surface production line. Please
translate the following text
from English to [Target Language].
### English:
[Source Text]
### [Target Language]:
[Target Translation]
</s>
```

For the Serbian dataset, since some of the source text was in German and the libraries available for language detection do not deal well with short UI text, the prompt was adjusted and "English" was replaced with "English/German".

When it comes to the "hybrid" fine-tuning, a script was made, which uniformly selected 0-5 examples from an example pool. Since some of the dataset had to be reserved for the example pools (one for training and one for validation and test), 60% was reserved for training and 15% for both test and validation. Both example pools are 5% of the entire dataset. The 15% part of the dataset for the test dataset and validation dataset was the same for both fine-tunings.

Because the model's weights are never updated during

evaluation, we can reuse a single pool of demonstration sentences for both the validation and test phases without risking data leakage. In other words, the examples that precede each source sentence at validation or test time serve only as context to guide the generation; they are not part of the optimization process. During training, we make this explicit by assigning a target label of $[-100]$ (the "ignore" index in PyTorch's 'CrossEntropyLoss') to every token that belongs to the instruction or to a demonstration example. Those masked tokens therefore contribute zero to the loss, ensuring that only the model's prediction of the true target translation influences gradient updates, while still allowing the preceding examples to act as in-context hints.

The following prompt format was used for the "hybrid" finetuning:

```
<s> Consider the following translations
from English to [Target Language]
for a user interface at a chemical
surface production line.
Example 1:
### English:
[SourceText]
### [Target Language]:
[Target Translation]
...
Example n:
### English:
[Source Text]
### [Target Language]:
[Target Translation]
Now, you are translating text for
a user interface at a chemical
surface production line. Please
translate the following text
from English to [Target Language].
### English:
[Source Text]
### [Target Language]:
[Target Translation]</s>
```

As mentioned above, the number of few-shot prompts was sometimes 0; therefore, some of the prompts in this finetuning had the same format as previously.

The choice for the format of the few-shot prompt was motivated by the results from [10], as it yielded the best performance for their similar experiments. As for the 0-shot prompt, it was designed so that there was minimal text and as much context as possible provided about the task.

H. Evaluation Metrics and Strategy

There is no single metric that evaluates translations accepted by all researchers. Although many papers on machine translation use the BLEU metric, it has significant limitations and is based on overlapping n-grams, which means it fails to

assess fluency and broader contextual understanding. This is a particularly poor choice for evaluating UI text translations, since whenever the text consists of, for example, 1 or 2 words, the score will most likely be 0 or 1 since synonyms give a score of 0. More advanced techniques have emerged with the advent of LLMs. For example, using neural networks

Based on an in-depth analysis by a team at Microsoft and ETH Zurich [14], the decision was made to use BLEURT and COMETKiwi-22 as the primary evaluation metrics for UI text translations in this paper. Both of these metrics have been rigorously validated on large-scale datasets, such as ToShip23 and WMT, demonstrating superior accuracy and reliability compared to traditional metrics like BLEU.

1) *BLEURT*: BLEURT (Bilingual Evaluation Understudy with Representations from Transformers) is a metric which combines deep learning techniques and extensive pre-training on synthetic data in order to align as closely as possible with human judgments. It was built on the BERT (Bidirectional Encoder Representations from Transformers) architecture and it was pre-trained on synthetic data as well as fine-tuned on human annotated datasets to achieve state-of-the-art performance. The metric takes into account the reference translation and the candidate translation when evaluating.

The metric underwent extensive pre-training using millions of synthetic sentence pairs generated from Wikipedia. These pairs are created through various perturbation techniques to introduce diverse linguistic variations [15]. The perturbation techniques are: mask-filling, backtranslation, and word dropping.

After pre-training, BLEURT is fine-tuned on datasets containing human evaluation scores, enabling it to predict translation quality scores that closely align with human judgments [15]. This fine-tuning allows BLEURT to learn linguistic nuances, including fluency, adequacy, and stylistic consistency.

To achieve a noticeable improvement in translation quality that, for instance, 90% of human evaluators would agree upon, there needs to be a 0.0173 increase in the BLEURT score [14].

2) *COMETKiwi-22*: COMETKiwi-22 is an advanced machine translation evaluation metric, which does not take into account the reference translation; it evaluates the translation solely based on the source text and the candidate translation. This feature makes it particularly useful in real-world applications, such as UI text translation, where high-quality reference translations may not always be available or practical to produce.

CometKiwi-22 is built on top of the COMET framework and it utilizes pre-trained multilingual language models to analyze the semantic adequacy and fluency of translations.

To achieve a noticeable change in the translation quality that, for example, 90% of human evaluators would agree upon, there needs to be a 0.0085 increase in the score [14].

3) *How the metrics complement each other*: The combination of the BLEURT score with the COMETKiwi-22 score gives us a very powerful evaluation framework. BLEURT ensures high precision by comparing the candidate with the reference, making it ideal for assessing fluency, adequacy, and nuanced differences in wording. On the other hand, COMETKiwi-22 offers a robust quality estimation in cases where reference translations are unavailable or incomplete.

III. EXPERIMENTS

As mentioned before, three language directions were explored: English-German, English-Chinese, and English-Serbian. The models picked were fine-tuned in two ways:

- 1) On the normal prompt format.
- 2) On a mixture of the 0-5 shot prompt format.

For each language pair and type of fine-tuning a separate jupyter notebook was used. However, the hyperparameters and most of the configurations remained the same among all of them.

First, the quantization configuration was defined using the bitsandbytes Python library, which was later used to load the quantized model directly from hugging face:

```
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16
)
```

The frozen pretrained weights are stored in a 4-bit data type called NormalFloat-4 (NF4). Each linear layer is quantized in small 64-value blocks so that outliers do not wreck precision. NF4 and double quantization minimize the information loss compared with plain int-4 while still letting the matrix multiply run in bfloat16 for speed and stability. All the original weights stay frozen; no gradients flow through them during training.

Then, the LoRA adapters were attached with the following configuration:

```
config = LoraConfig(
    r=128,
    lora_alpha=256,
    target_modules=get_last_layer_
    linears(model),
    lora_dropout=0.05,
    task_type="CAUSAL_LM"
)
```

```
model = get_peft_model(model, config)
```

The rank $r = 128$ and $\alpha = 256$ were chosen to be this high because translation is a high-capacity task: doubling the usual rank - 64, gives the adapters enough expressiveness to learn domain-specific word order and terminology, while $\alpha = 256$ (which yields a desirable scale of 2.0) keeps those updates numerically aligned with the frozen 4-bit backbone, preventing gradient blow-ups. Only the final block’s linear projections (reached by the custom function `get_last_layer_linears(model)`) are patched because the top layer encodes task specific abstractions; adapting a single block reduces memory, which is desired in our set-up: 16GB GPU. A small LoRA dropout of 0.05 regularizes the thin adapters so they do not over-fit the 1,934-3,054 entry corpus. Lastly, the `task_type="CAUSAL_LM"` tells PEFT to insert adapters only in self-attention and MLP (multilayer perceptron) layers, the correct locations for our decoder-only TowerBase/SambaLingo backbone models.

TABLE III
TRAINING PARAMETERS FOR BOTH FINE-TUNING TASKS

Parameter	Value
Batch size	8 sequences x 2048 tokens
Optimizer	AdamW
Learning rate	2×10^{-4}
Scheduler	Linear with 500 warm-up steps
Number of epochs	6-8 (depends when eval_loss converges)

The training configuration (Table III) was deliberately lean so it would run on a single 16 GB GPU while still giving the adapters enough signal to adapt. A global batch of eight 2,048 token prompts fits comfortably in memory with 4-bit-quantized weights, so if needed, every iteration sees a full instruction, up to five demonstrations, and the source text, which are all required for learning in-context behavior. Because only the low-rank adapter matrices are trainable, a relatively high learning rate of 2×10^{-4} can be used. Exactly 500 warm-up steps are applied followed by a linear decay so that the optimizer does not overshoot at the start but still converges within a few hundred updates. AdamW is used, which is a robust choice for transformer fine-tuning. Given our small dataset, running fewer than six epochs underfits, while more than eight epochs leads to overfitting; in practice, validation loss flattened between the sixth and eighth epoch, so we stop training as soon as it levels off. Together, this schedule balances GPU memory constraints, fast convergence, and overfitting control.

As local hardware could not provide the VRAM needed for a 7B parameter model, all fine-tuning experiments were run on Paperspace, the cloud platform to which AUCOS maintains an enterprise account. Each job used a single NVIDIA RTX A4000 (16 GB) instance running Ubuntu 22.04, Python 3.10,

and CUDA 11.8. On this setup, one epoch training took roughly an hour.

A. Evaluation Experiments

All fine-tuned models were first evaluated on the same test set. The "normal" fine-tuning only with 0-shot examples and the "hybrid" fine-tuning with 0-shot examples and 1-5-shot examples. The generation configuration was usually the same, with the exception of evaluating the "hybrid" fine-tuning on few-shot prompts, where a minimum number of new tokens generated had to be defined.

Here is the generation configuration:

```
max_new_tokens = 100
temperature = 0.7
top_p = 0.7
```

If the number of maximum new tokens was too low, it might truncate the translation before all words are generated. Therefore, a safe upper bound was defined at 100, giving the model enough space to reach the end of sequence token ($\langle \text{;}/s_{\text{;}} \rangle$). The temperature controls the randomness in sampling, and it was set to 0.7 to have a balance between safety and diversity (at 1.0 the model would sample directly from the learned distribution). The `top_p` parameter (set to 0.7) implements nucleus sampling, restricting generation to the smallest set of tokens whose cumulative probability reaches 0.7, which helps maintain coherence while still allowing some variety.

Evaluation of the scores based on the length of the source sequence was also performed. The sequences were divided into 3 groups: 1) 1-2 words in the source text, 2) 3-4 words in the source text, or 3) 5 or more words in the source text. Then the metrics scores were calculated for each category.

Another evaluation experiment performed evaluated the performance of the "hybrid" fine-tuned model on a each number of few-shot examples (between 1 and 5). It was done in order to further investigate the influence of in-context examples incorporated into the training prompts on the performance of the "hybrid" fine-tuned model.

Last evaluation experiment performed, was a small-scale human evaluation study. The participants (all fluent in English and one of the target languages) were given translations which achieved a BLEURT score of less than 0. They were told to give an overall impression of the translations and, if possible, find recurring patterns. multirow

IV. RESULTS & DISCUSSION

A. Best Performing Fine-Tuned Model

After running the evaluation experiments, it was clear that the best performing (fine-tuned) models for all language directions were the "normal" ones that were fine-tuned without the incorporated in-context examples (Table IV, V & VI, Figure 1 & 2). All the "normal" fine-tuned models greatly outperformed

the baseline BLEURT scores by $\uparrow 0.22$ - 0.34 , however, the COMETKiwi-22 score remained the highest for the baseline model for all languages and it dropped by $\downarrow 0.02$ - 0.06 . The reason behind it is quite easily explainable - the BLEURT metric mostly determines whether the candidate translation aligns with the reference translation, therefore, evaluates whether the candidate is translated similarly to the reference, testing the domain-adaptation of the model. On the other hand, COMETKiwi-22 metric bases its output on the source and the candidate, omitting the reference translation in the evaluation process. It evaluates the overall linguistic capabilities of the model and does not take into account the domain-specific style that is present in the reference. This is why the improved BLEURT score shows us that our model has improved in the domain-specific translations for Aucos’s UI text, however, it lost some of its generic translation capabilities.

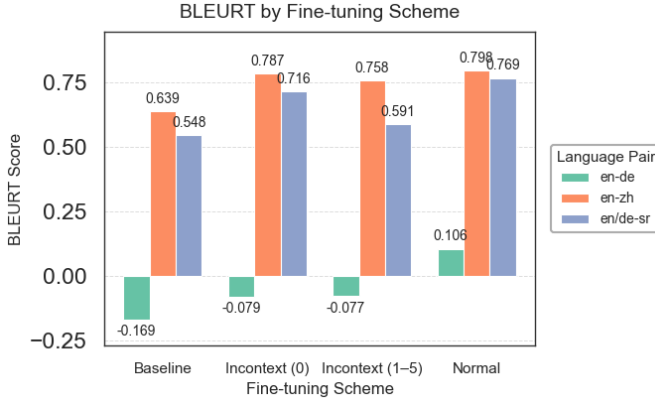


Fig. 1. BLEURT metric score for all fine-tunings and the baseline for all language directions. The numbers in the brackets for the incontext fine-tuning represent the number of few-shot examples in the test dataset.

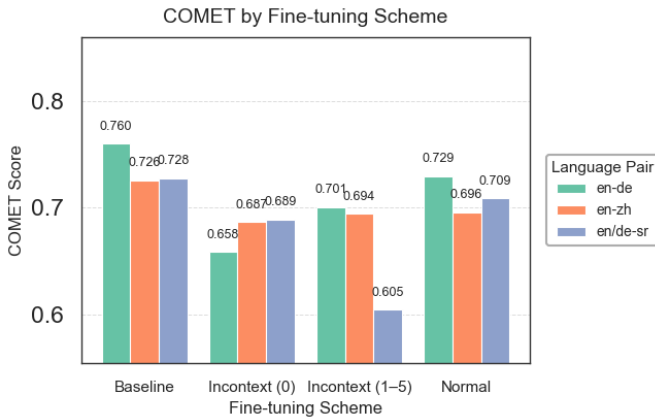


Fig. 2. COMETKiwi-22 metric score for all fine-tunings and the baseline for all language directions. The numbers in the brackets for the incontext fine-tuning represent the number of few-shot examples in the test dataset.

TABLE IV
PERFORMANCE OF THE “EN-DE” MODELS UNDER DIFFERENT FINE-TUNING SCHEMES (95% CONFIDENCE INTERVAL).

Metric	Baseline	Normal FT	Incontext FT – 0 shot	Incontext FT – 1-5 shot
BLEURT	-0.1685	0.1063 $\pm$ 0.0867	-0.0787 $\pm$ 0.0918	-0.0775 $\pm$ 0.0920
COMETKiwi-22	0.7596	0.7294 $\pm$ 0.0172	0.6584 $\pm$ 0.0223	0.7005 $\pm$ 0.0198

TABLE V
PERFORMANCE OF THE “EN-ZH” MODELS UNDER DIFFERENT FINE-TUNING SCHEMES (95% CONFIDENCE INTERVAL).

Metric	Baseline	Normal FT	Incontext FT – 0 shot	Incontext FT – 1-5 shot
BLEURT	0.6394	0.7985 $\pm$ 0.0260	0.7867 $\pm$ 0.0295	0.7576 $\pm$ 0.0351
COMETKiwi-22	0.7255	0.6956 $\pm$ 0.0147	0.6867 $\pm$ 0.0150	0.6944 $\pm$ 0.0140

TABLE VI
PERFORMANCE OF THE “EN/DE-SR” MODELS UNDER DIFFERENT FINE-TUNING SCHEMES (95% CONFIDENCE INTERVAL).

Metric	Baseline	Normal FT	Incontext FT – 0 shot	Incontext FT – 1-5 shot
BLEURT	0.5482	0.7689 $\pm$ 0.0285	0.7163 $\pm$ 0.0335	0.5905 $\pm$ 0.0391
COMETKiwi-22	0.7277	0.7087 $\pm$ 0.0151	0.6891 $\pm$ 0.0156	0.6045 $\pm$ 0.0162

B. Metric Score Value Range for each Language

What is interesting is the BLEURT score distribution for the en-de direction: 28 % of German test strings score < 0.0 , whereas only 4 % (en-zh) and 6 % (en/de-sr) do (see Figure 1). This gap stems not from genuinely poorer German output but from BLEURT’s training bias and German morphology: the metric was tuned on full-sentence WMT data, so it over-penalises any lexical deviation inside long compounds such as “Leistungsschutzschalter” (27 characters), producing a negative tail that is absent in Chinese or Serbian.

The COMETKiwi-22 metric seems to have the highest values for the English-German translations (0.7294 - 0.7596), the English/German-Serbian translations come in second (0.7087 - 0.7277), and the English-Chinese translations have the lowest COMETKiwi-22 scores (0.6956 - 0.7255). As for the BLEURT score - English-Chinese translations perform best (0.6394 - 0.7985), the English/German-Serbian translations come in second (0.5482 - 0.7689), and the English-German translations obtain the worst values (-0.1685 - 0.1063). Since, the order of these is exactly the opposite of the other metric, it is hard to draw a conclusion on how the resource-level affects the values of the metrics.

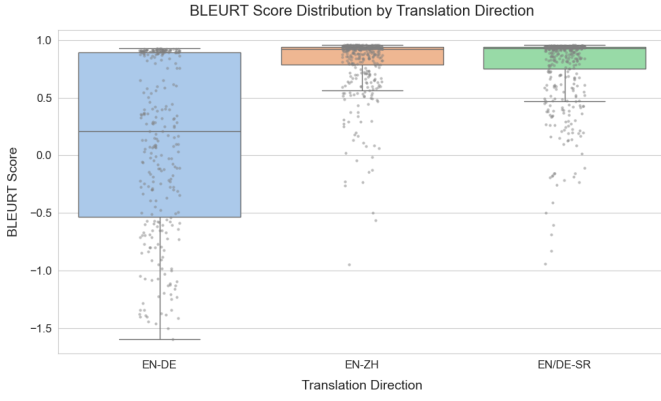


Fig. 3. The distribution of the BLEURT score of the best performing - "normal" fine-tuned models for each language direction.

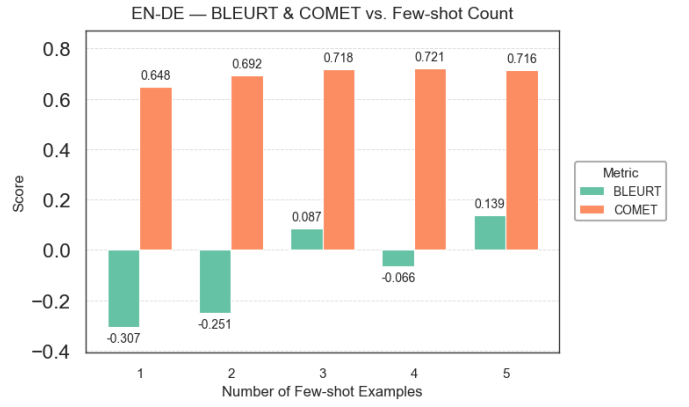


Fig. 4. BLEURT and COMETKiwi-22 metrics score for the hybrid fine-tuning (en-de language direction) depending on the number of few-shot examples in the test dataset.

C. Influence of Incorporating Few-shot Examples into the Dataset

Incorporating 1-5 in-context UI translation examples into the fine-tuning data did not improve the performance of the models (Table IV, V&VI, Figure 2&3). Despite providing few-shot examples in most of the test data for the hybrid fine-tuning, the hybrid FT model still performs better on zero-shot prompts, but slightly worse than the normal FT model on the same zero-shot prompts. The model did not lose its few-shot prompting capabilities, as proven by Alves et al. [10]. However, the model sometimes hallucinated on those few-shot prompts and the generation configurations had to be tweaked multiple times to handle the model not producing any translations (we forced a minimum number of new tokens it has to produce during inference). This disproved the hypothesis that fine-tuning a model on few-shot examples and then performing inference with few-shot examples included in the prompt would yield better results than "normal" fine-tuning. The reasons behind this could be countless, one of them being our dataset being too small, and the size difference in training datasets (10% $\approx$ 200-300 examples) had a great impact on the performance of the fine-tuned models (more training data). Perhaps the trade-off between using few-shot prompts and having slightly less data on such small datasets is not worth it.

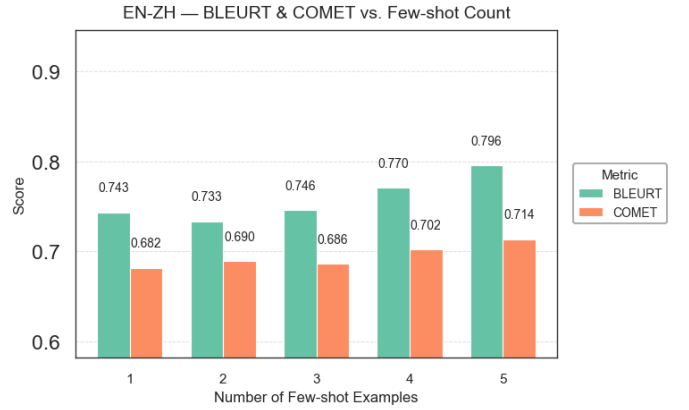


Fig. 5. BLEURT and COMETKiwi-22 metrics score for the hybrid fine-tuning (en-zh language direction) depending on the number of few-shot examples in the test dataset.

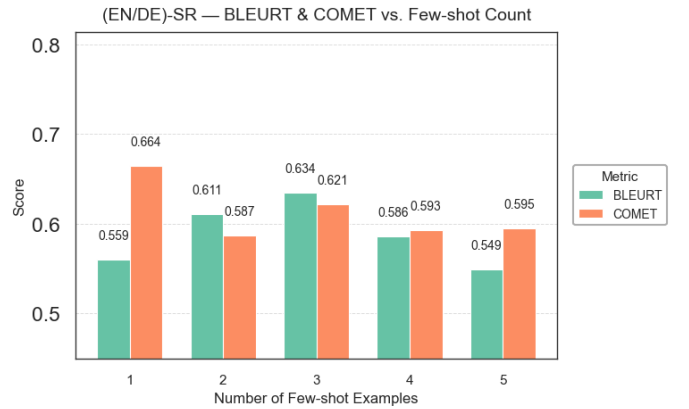


Fig. 6. BLEURT and COMETKiwi-22 metrics score for the hybrid fine-tuning (en/de-sr language direction) depending on the number of few-shot examples in the test dataset.

TABLE VII
PERFORMANCE OF THE “HYBRID FT” MODEL ON EN-DE WITH VARYING FEW-SHOT EXAMPLES.

Number of few-shot examples	BLEURT	COMET
1	-0.3070 ± 0.2113	0.6485 ± 0.0538
2	-0.2511 ± 0.2345	0.6923 ± 0.0521
3	0.0867 ± 0.1951	0.7178 ± 0.0378
4	-0.0662 ± 0.1831	0.7213 ± 0.0380
5	0.1389 ± 0.2116	0.7155 ± 0.0461

TABLE VIII
PERFORMANCE OF THE “HYBRID FT” MODEL ON EN-ZH WITH VARYING FEW-SHOT EXAMPLES.

Number of few-shot examples	BLEURT	COMET
1	0.7434 ± 0.0904	0.6817 ± 0.0358
2	0.7334 ± 0.0949	0.6895 ± 0.0343
3	0.7460 ± 0.0789	0.6858 ± 0.0262
4	0.7705 ± 0.0699	0.7023 ± 0.0323
5	0.7961 ± 0.0564	0.7135 ± 0.0291

TABLE IX
PERFORMANCE OF THE “HYBRID FT” MODEL ON EN-SR WITH VARYING FEW-SHOT EXAMPLES.

Number of few-shot examples	BLEURT	COMET
1	0.5593 ± 0.0948	0.6640 ± 0.0306
2	0.6107 ± 0.0825	0.5867 ± 0.0323
3	0.6344 ± 0.0732	0.6210 ± 0.0298
4	0.5860 ± 0.0971	0.5929 ± 0.0368
5	0.5490 ± 0.0987	0.5947 ± 0.0356

As the “hybrid” fine-tuning did not perform very well, it was decided that it should be further investigated. It was done by seeing whether some number of few-shot examples in the prompt performed better than others. When prompted with few-shot examples, both the en-de and en-zh fine-tuned models performed best when given 5 few-shot examples in terms of the BLEURT score (Table VI & VII). The (en/de)-sr translation direction obtained the highest result with 3 few-shot examples (Table VIII). Similarly, the COMETKiwi-22 metric was the highest for 4/5 few-shot examples for en-de and en-zh, however, it was the highest for 1-shot prompts for en/de-sr. These results could be caused by many factors: German and Chinese demonstrations are, on average, shorter than Serbian ones. Five examples still fit comfortably in the Llama 2 2,048 token window, however, Serbian demos at more than 3 (or 1, if we focus on COMETKiwi-22) examples fill the context, forcing KV-cache⁴ paging and weakening attention to earlier steps - hence the BLEURT/COMETKiwi-22 drop.

⁴The tensors of keys (K) and values (V) that the model stores for every past token so it can compute attention for new tokens without re-encoding the whole prefix.

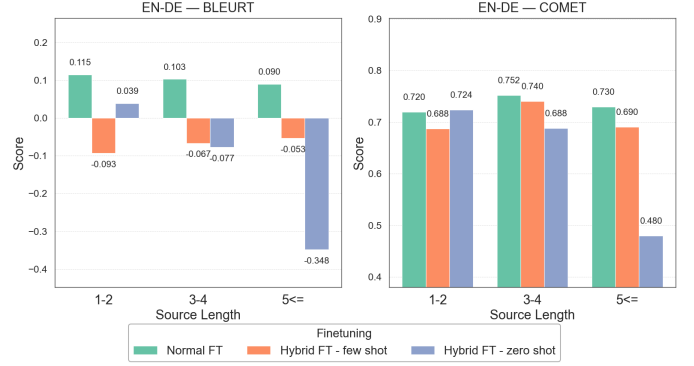


Fig. 7. BLEURT and COMETKiwi-22 scores for en-de translations, based on the length of the source text for each of the performed fine-tunings. The “hybrid” fine-tuning is evaluated twice - once with 0-shot prompts and once with 1-5 few-shot prompts.

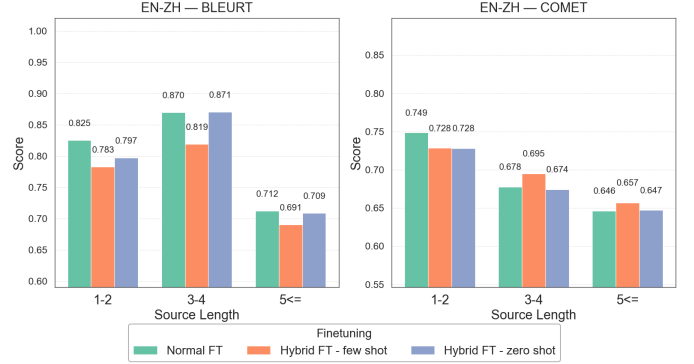


Fig. 8. BLEURT and COMETKiwi-22 scores for en-zh translations, based on the length of the source text for each of the performed fine-tunings. The “hybrid” fine-tuning is evaluated twice - once with 0-shot prompts and once with 1-5 few-shot prompts.

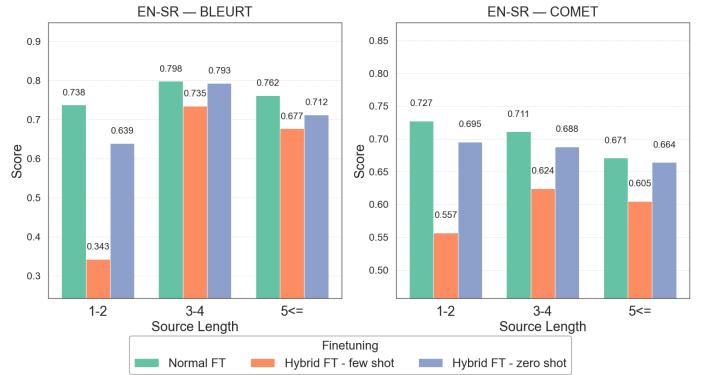


Fig. 9. BLEURT and COMETKiwi-22 scores for en/de-sr translations, based on the length of the source text for each of the performed fine-tunings. The “hybrid” fine-tuning is evaluated twice - once with 0-shot prompts and once with 1-5 few-shot prompts.

D. Influence of the Source Text Length

Despite sequences of length 1 or 2 having less context, the models did not show worse performance on those sequences compared to 3-4 or $5 \leq$ word long sequences (Table X, XI & XII, Figure 7, 8 & 9). For the en-de translation direction, longer sequences had worse BLEURT and COMETKiwi-22 scores for both the "normal" and "hybrid" fine-tuning evaluated on 0-shot prompts. The best performing model - "normal" fine-tuned - performed best on sequences of 1-2 words in terms of BLEURT and 3-4 words in terms of COMETKiwi-22. As for en-zh, it was also visible, where the best performance in terms of BLEURT was achieved by the "normal" fine-tuned model on sequences of 3-4 words and 1-2 words in terms of COMETKiwi-22. (En/de)-sr performed best on sequences which were 3-4 words long (in terms of BLEURT) and 1-2 in terms of COMETKiwi-22 for the models evaluated on 0-shot prompts.

TABLE X
BLEURT AND COMET SCORES FOR EN-DE BY SOURCE-SENTENCE LENGTH UNDER DIFFERENT FINE-TUNING SCHEMES.

NORMAL FT – en-de		
Source length	BLEURT	COMET
1-2	0.1148 ± 0.1352	0.7196 ± 0.0253
3-4	0.1030 ± 0.1703	0.7518 ± 0.0348
$5 \leq$	0.0900 ± 0.1266	0.7295 ± 0.0306
HYBRID FT – few shot (en-de)		
Source length	BLEURT	COMET
1-2	-0.0929 ± 0.1416	0.6878 ± 0.0275
3-4	-0.0665 ± 0.1707	0.7402 ± 0.0360
$5 \leq$	-0.0534 ± 0.1557	0.6904 ± 0.0455
HYBRID FT – zero shot (en-de)		
Source length	BLEURT	COMET
1-2	0.0391 ± 0.1398	0.7236 ± 0.0239
3-4	-0.0775 ± 0.1759	0.6884 ± 0.0424
$5 \leq$	-0.3481 ± 0.1363	0.4802 ± 0.0453

TABLE XI
BLEURT AND COMET SCORES FOR EN-ZH BY SOURCE-SENTENCE LENGTH UNDER DIFFERENT FINE-TUNING SCHEMES.

NORMAL FT – en-zh		
Source length	BLEURT	COMET
1-2	0.8250 ± 0.0490	0.7485 ± 0.0201
3-4	0.8700 ± 0.0315	0.6776 ± 0.0291
$5 \leq$	0.7121 ± 0.0417	0.6464 ± 0.0356
HYBRID FT – few shot (en-zh)		
Source length	BLEURT	COMET
1-2	0.7831 ± 0.0677	0.7284 ± 0.0229
3-4	0.8190 ± 0.0540	0.6949 ± 0.0255
$5 \leq$	0.6906 ± 0.0466	0.6569 ± 0.0226
HYBRID FT – zero shot (en-zh)		
Source length	BLEURT	COMET
1-2	0.7974 ± 0.0584	0.07280 ± 0.0226
3-4	0.8705 ± 0.0327	0.6742 ± 0.0290
$5 \leq$	0.7091 ± 0.0446	0.6471 ± 0.0262

TABLE XII
BLEURT AND COMET SCORES FOR (EN/DE)-SR BY SOURCE-SENTENCE LENGTH UNDER DIFFERENT FINE-TUNING SCHEMES.

NORMAL FT – (en/de-sr)		
Source length	BLEURT	COMET
1-2	0.7377 ± 0.0608	0.7274 ± 0.0261
3-4	0.7983 ± 0.0517	0.7110 ± 0.0293
$5 \leq$	0.7621 ± 0.0416	0.6709 ± 0.0262
HYBRID FT – few shot (en/de-sr)		
Source length	BLEURT	COMET
1-2	0.3432 ± 0.0813	0.5569 ± 0.0288
3-4	0.7351 ± 0.0538	0.6242 ± 0.0306
$5 \leq$	0.6772 ± 0.0515	0.6050 ± 0.0251
HYBRID FT – zero shot (en/de-sr)		
Source length	BLEURT	COMET
1-2	0.6393 ± 0.0726	0.6951 ± 0.0293
3-4	0.7932 ± 0.0503	0.6879 ± 0.0303
$5 \leq$	0.7123 ± 0.0523	0.6642 ± 0.0249

E. Human Evaluation Study

The user evaluation study was performed on the best performing model - "normal" fine-tuning. It revealed there to be some recurring patterns that the models used when mistranslating certain sequences. The participants only saw the worst translations ($\text{BLEURT} < 0$), therefore, the results here are only for a bad and small subset of the translations.

For the en-de model, the dominant failure mode was a part-of-speech slip: one-word English buttons such as "Back", "Reload", or "Print" were treated as noun labels and rendered as unrelated German nouns - e.g. "Back" - "Rückgrat" (spine) instead of the imperative "Zurück". A second, equally penalising pattern was the creation of very long compounds that obscured the intended term, e.g. "Stocked Carrier Location" - "Gestellatenträger-Station" where the UI guideline prefers the shorter "Trägersammelort". For the en-zh direction the most common issue was the survival of untranslated Latin tokens or codes inside otherwise correct Chinese strings - weekday abbreviations (MON, THU), units (g), or date masks (yyyy-MM-dd) which BLEURT flags heavily. The en/de-sr model showed two characteristic errors: (i) mixing Latin and Cyrillic in the same label, and (ii) drifting into closely-related South-Slavic vocabulary—e.g. "Default" - "Uobičajeno" (a Croato-Bosnian everyday word) instead of the Serbian IT term "Podrazumevano". Together these patterns explain nearly all low BLEURT outliers.

V. CONCLUSION

This research set out to determine whether small, open-source models can be pushed to deliver reliable UI translations with limited in-domain data and modest hardware. The evidence collected answers the four research questions as follows.

Including one to five shot demonstrations in the training prompts and again at test time failed to raise either BLEURT or COMETKiwi-22 above the scores obtained with a

pure zero-shot approach. Although the "hybrid" procedure preserved the model's ability to use demonstrations at inference, the cost of diverting 10% of the corpus into example pools outweighed the benefit: on a dataset this small, every additional parallel pair added to the training set is more valuable than an extra few-shot exemplar in the prompt.

Second, the length of the source string is not a major obstacle: labels containing only one or two words are translated as well as, and sometimes better than, longer phrases. Where performance drops for German, the cause is BLEURT's sensitivity to small deviations inside long compounds rather than lack of context.

Third, once language-specialized checkpoints were brought into play, the usual gap between high-resource (German), mid-resource (Chinese) and low-resource (Serbian) directions was not visible. TowerBase-7B already carries strong German and Chinese priors, while SambaLingo-7B inherits 17B Serbian tokens from Cultura-X; after QLoRA adaptation the three directions converge to a narrow BLEURT range (0.70 – 0.80) and a COMETKiwi-22 spread under 0.03. In other words, with a model whose pre-training explicitly covers the target language, resource level ceases to be the dominant factor — the decisive variable becomes how much domain-specific data the model sees during fine-tuning.

Finally, task-specific fine-tuning with 4-bit QLoRA and single-layer LoRA adapters proves to be a cost-effective route: it delivers the largest BLEURT gains, trains in under eight hours per language on a single RTX A4000, and avoids the memory footprint of full-precision or full-model updates.

In short, parameter-efficient adaptation is a practical solution for translating UI text in industrial settings, but including demonstration examples during training is not yet justified on corpora of only a few thousand lines. Options to expand the in-domain corpus should be explored, for example, by mining public UI strings, generating synthetic pairs with terminology constraints, or leveraging parallel leakage from documentation in order to test whether the hybrid few-shot strategy begins to pay off once data scarcity is relaxed.

REFERENCES

- [1] Sowmya Koneru, Matthias Huck, Maximilian Exel, Jan Niehues, et al. Analyzing challenges in neural machine translation for software localization. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2434–2446, May 2023.
- [2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.
- [3] Yiding Tang, Christian Tran, Xinchu Li, Peng-Jen Chen, et al. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint*, arXiv:2008.00401, 2020.
- [4] NLLB Team et al. No language left behind: Scaling human-centered machine translation. *arXiv preprint*, arXiv:2207.04672, 2022.
- [5] Duarte M. Alves, José Pombal, Nuno M. Guerreiro, Pedro H. Martins, Jo ao Alves, et al. Tower: An open multilingual large language model for translation-related tasks. *arXiv preprint*, arXiv:2402.17733, 2024.
- [6] Zoltan Csáki, Bo Li, Jonathan Li, Qiantong Xu, Pian Pawakapan, et al. Sambalingo: Teaching large language models new languages. *arXiv preprint*, arXiv:2404.05829, 2024.
- [7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, et al. Lora: Low-rank adaptation of large language models. *arXiv preprint*, arXiv:2106.09685, 2021.
- [8] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer, et al. Qlora: Efficient finetuning of quantized llms. *arXiv preprint*, arXiv:2305.14314, 2023.
- [9] Jiawei Zheng, Hanghai Hong, Feiyan Liu, Xiaoli Wang, Jingsong Su, Yonggui Liang, et al. Fine-tuning large language models for domain-specific machine translation. *arXiv preprint*, arXiv:2402.15061, 2024.
- [10] Duarte M. Alves, Nuno M. Guerreiro, Jo ao Alves, José Pombal, Ricardo Rei, et al. Steering large language models for machine translation with finetuning and in-context learning. *arXiv preprint*, arXiv:2310.13448, 2023.
- [11] AI @ Meta Llama Team et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint*, arXiv:2307.09288, 2023.
- [12] Zoltan Csáki, Pian Pawakapan, Urmish Thakker, and Qiantong Xu. Efficiently adapting pretrained language models to new languages. *arXiv preprint*, arXiv:2311.05741, 2023.
- [13] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. *arXiv preprint*, arXiv:2005.14165, 2020.
- [14] Tom Kocmi, Vilém Zouhar, Christian Federmann, Matt Post, et al. Navigating the metrics maze: Reconciling score magnitudes and accuracies. *arXiv preprint*, arXiv:2401.06760, 2024.
- [15] Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. Bleurt: Learning robust metrics for text generation. *arXiv preprint*, arXiv:2004.04696, 2020.