Knowledge distillation in multilingual NMT models

1. Introduction
   1.1. Motivation
   1.2. Research questions
2. Background
   2.1. Machine translation
      2.1.1. Multilingual machine translation
   2.2. Knowledge distillation
      2.2.1. Word-level distillation
      2.2.2. Sequence-level distillation
3. Methods
   3.1. Data sources
      3.1.1. Training data
      3.1.2. Evaluation data
      3.1.3. Selection of language pairs
   3.2. Tools
      3.2.1. Data processing (Opusfilter)
      3.2.2. Model training (Fairseq)
      3.2.3. Computing Environment (Puhti)
      3.2.4. Evaluation metrics
   3.3. Experiments
      3.3.1. TIE-KD (Zhang, et al., 2023)
4. Results and analysis
   4.1. Experiment with method 1 (TIE-KD?)
   4.2. Experiment 2 etc.
5. Discussion
6. Conclusion

Abbreviations

NLP – Natural Language Processing

MT – Machine Translation

NMT – Neural Machine Translation

KD – Knowledge Distillation

ML – Machine Learning

BPE – Byte-pair Encoding

BLEU – Bilingual Evaluation Understudy

GPU – Graphics Processing Unit

<unk> - Unknown token

Language abbreviations

De – German

Sv – Swedish

Lb – Luxembourgish

Af – Afrikaans

En - English

1. Introduction
    1.1. Motivation

In recent years, NMT has proven to be a practical and valuable field of research with many real-world benefits (cite). Many of these NMT models focus on a specific language pair, but researchers have shown that a multilingual approach to machine translation can improve the accuracy of a model by leveraging similarities between different languages (cite), especially within language families (cite). Additionally, a multilingual approach might be beneficial for low-resource languages (cite), which do not have an abundance of training data available. Another consideration is that creating high quality training data is expensive and laborious (cite). A multilingual approach may be beneficial when such resources are not available (cite).

In pursuit of better accuracy, researchers have utilized ever increasingly larger training datasets (cite), which generally tends to improve the accuracy of a model (Koehn and Knowles, 2017). However, having massive amounts of training data comes at the cost of computational efficiency (cite), as larger and more complex NMT models require more energy and increasingly powerful hardware to run. To alleviate this efficiency issue, researchers have proposed various methods to distill the knowledge of large NMT models into smaller, more compact models and make them more computationally efficient (cite). Furthermore, these large models seem to learn a large amount of information from the training data that is underutilized or otherwise less helpful to the end result (cite). As a result, the process of knowledge distillation may not hinder a model’s accuracy while providing a large computational performance uplift. This paper will explore how different types of knowledge distillation (KD) scale with multilingual NMT models.

The process of knowledge distillation has many benefits, the most obvious being the reduced computational requirements. Typically, large NMT models require expensive and power-hungry hardware, but the models can be scaled down to run on smaller devices, such as consumer-grade computer hardware or even smartphones without sacrificing accuracy (cite). Technology companies and consumers are becoming more conscious of environmental factors and a smaller carbon footprint may become a crucial consideration as recent large language models (LLMs) are known to consume massive amounts of energy (cite), evidenced by the fact that companies such as Google have opted to purchase their own nuclear power plants to satisfy the power requirements of their AI hardware. (cite the Guardian?)

Running the models locally, instead of on large server farms, has additional benefits related to privacy (cite). Typically, the text that needs to be translated must be sent via the internet to the provider of the translation service, which can be a problem for privacy-informed individuals or companies translating sensitive information, who do not wish to share their information with the translation service provider. It may not always be clear how or if the submitted text is stored by the provider (cite), and additionally the provider may use the data to further develop their translation model, potentially resulting in the model learning sensitive information which can cause issues.

Personal motivation?

The exact mechanism of how KD functions is still unclear (cite), which is why more research needs to be done in order to understand how and when to use different KD methods. There has been limited prior work on multilingual KD (Gumma, V., et al., 2023), making this study a novel contribution to the field.

1.2. Research questions

The purpose of this thesis is to examine how different knowledge distillation methods scale in a multilingual setting. As such, the research questions can be defined as:

- How do knowledge distillation methods scale in a multilingual setting?
- How do KD methods perform with student models of varying sizes?

- Expectation that KD helps multilingual models
- Measure the overall effectiveness of KD - Smaller model needs more guidance, leads to bigger gain?

2. Background

2.1. Machine translation

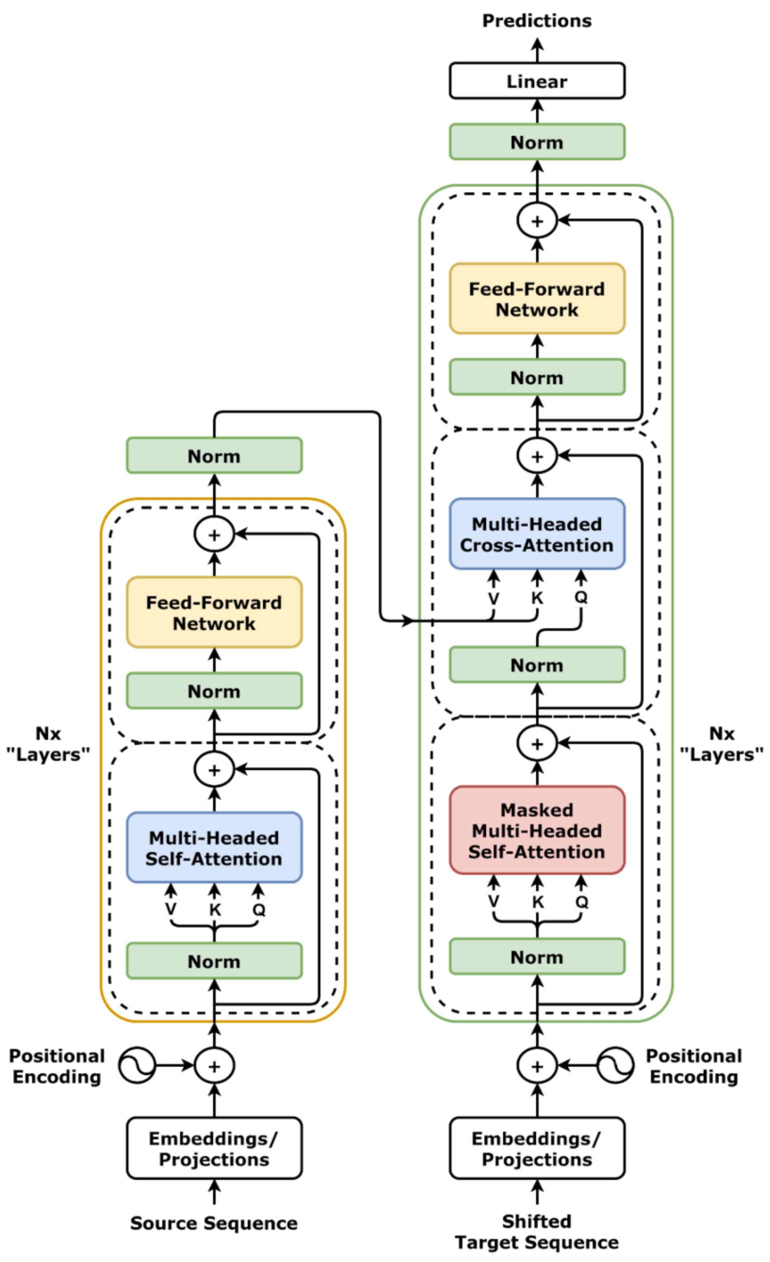
Machine translation (MT) has become ubiquitous in the last few decades, although it is not a recent invention by any means and has undergone multiple paradigm shifts (cite). In fact, the earliest attempts at MT originate from the 1950s in the Georgetown-IBM experiments (cite). These early attempts, while promising, were limited by the available computational power of the time and the inflexible and laborious rule-based methods employed by the researchers, which required the manual hard coding of every linguistic feature from the source language into the target language (cite). These rule-based methods could only produce crude word-for-word translations and were not able to capture the chaotic essence of natural language (cite).

Progress on machine translation was slow as interest in the field had decreased, but the increasingly powerful computer hardware of the 1990s gave rise to a new paradigm, statistical machine translation (SMT) (cite).

SMT offered several advantages over rule-based translation systems. Unlike rule-based approaches, which relied on predefined linguistic rules and extensive lexicons, SMT uses large bilingual corpora to automatically learn translation patterns, making it more adaptable to a wide range of languages and domains (cite). SMT was more effective in producing accurate translations (cite), as it bases translations on probability models derived from real-world linguistic data (cite). Additionally, SMT requires less manual effort in developing linguistic rules (cite), enabling faster deployment and easier updates when new language data becomes available.

Recent years have seen another large shift in methodology, as SMT were largely replaced by neural sequence-to-sequence (Seq2Seq) models (cite). Seq2Seq models, a type of neural network architecture, consist of an encoder-decoder framework where the input sentence is encoded into a fixed-length vector, and the decoder generates the output sequence based on this representation (cite). This architecture surpasses SMT's phrase-based methods by capturing the entire sentence context, including long-range dependencies between words. Seq2Seq models, often enhanced with attention mechanisms (cite), allow for dynamic focus on different parts of the input, leading to more accurate and fluent translations. This transition has resulted in significant improvements in translation quality, especially for languages with complex grammatical structures or sparse training data (cite).

Transformers (cite) revolutionized Seq2Seq models by addressing key limitations of earlier architectures, such as recurrent neural networks (RNNs) and long short-term memory (LSTM) models. Traditional Seq2Seq models struggled with capturing long-range dependencies due to their sequential nature (cite), which often led to information loss due to vanishing gradients in longer sentences. Transformers, on the other hand, utilize a self-attention mechanism (cite) that allows the model to process all words in a sentence simultaneously, rather than sequentially.



The self-attention mechanism operates by computing a set of attention scores for each word in the sequence, determining how much focus should be placed on other words when generating a translation. This process is implemented using query, key, and value (QKV) matrices, which help the model dynamically adjust its attention distribution across the input sequence. Multi-head attention further enhances this process by allowing the model to capture multiple linguistic relationships at once, improving translation fluency and accuracy.

Since transformers do not process text sequentially, they lack an inherent understanding of word order (cite). To address this, positional encodings are added to word embeddings, providing the model with information about the relative positions of words in a sentence. These encodings are generated using sinusoidal functions and enable the model to maintain sentence structure while processing text in parallel.

The advantages of transformer models in machine translation are significant. Their parallelization dramatically reduces training and inference time, making them highly efficient for massive datasets (cite). By capturing global context rather than relying on limited memory like RNNs, transformers produce more fluent and coherent translations (cite).This enables the model to weigh the importance of different words at each position, regardless of their distance from one another. By replacing recurrence with parallel processing, transformers significantly improve computational efficiency and allow for better handling of complex linguistic patterns (cite). This architecture, coupled with the encoder-decoder framework, makes transformers the backbone of state-of-the-art NMT models, yielding more accurate and contextually aware translations.

2.1.1 Word Embeddings

A key improvement to NMT was made through distributional semantics (Hinton, G.E., et al., 1986) where semantic similarity could be modeled computationally in vector space. Word embeddings (Mikolov, T., et al., 2013) are a fundamental component of NMT models, providing a machine-readable vector representation of words that capture semantic and syntactic relationships. The training process for word embeddings typically occurs jointly with the overall model training, allowing the embeddings to adapt to the specific translation task (cite) or by using pretrained embeddings (cite GlovE, Bert), although this practice seems to have fallen out of favor in recent years.

Initially, word embeddings are either randomly initialized or derived from pretrained sources. During training, the model refines these embeddings through backpropagation (cite), where gradient updates adjust the vectors based on a loss function such as cross-entropy (cite). The embeddings are optimized to minimize the difference between the model’s predicted translations and the ground-truth references.

To handle rare or unseen words effectively, subword tokenization techniques such as Byte-Pair Encoding (BPE) (cite) or SentencePiece (cite) are often applied. These methods break words into smaller subunits (cite), reducing vocabulary size and allowing embeddings to generalize across morphologically rich languages. The tokenized input is then mapped to a high-dimensional embedding space, where similar words or subwords should appear closer together based on their contextual usage.

Since transformer-based models do not inherently capture word order, positional encodings are added to the embeddings to preserve sequence information. This ensures that the model can distinguish the relative positions of words in a sentence. The embeddings, enriched with positional information, pass through multiple layers of attention and feed-forward networks, allowing the model to learn contextual word representations that improve translation accuracy.

2.1.1. Multilingual machine translation

Multilingual machine translation is inherently more difficult due to the added complexity of multiple different languages (cite). Languages have different lexical, syntactical and morphological features which the model will not only have to learn but to distinguish between and correctly identify which unique features belong to a specific language in order to provide accurate translations.

Even with the added complexity, multilinguality may not be a complete detriment to the model, as individual languages may share some features, especially when they are genealogically related (cite). Languages that are closely related will most likely share many features, such as similar word conjugation patterns or word stems. The model can leverage these similarities to its advantage, which can be useful especially in situations where one of the languages is considered low-resource (cite), such as Luxembourgish and another is high-resource, such as German. As the two languages are somewhat similar, the model can use the German training data to supplement its knowledge on Luxembourgish.

Having more and more languages will benefit the cross-lingual performance of especially the low-resource languages, but it may come with the cost of degrading the monolingual performance of the model. Researchers have dubbed this phenomenon as the curse of multilinguality (Conneau et al., 2019)

Multilingual machine translation can be achieved with many different configurations with regards to the translation direction. This paper focuses on the many-to-one configuration, where the model learns to translate multiple languages into one. There are many other configurations, such as translating one language into multiple languages (one-to-many), or multiple languages into multiple languages many-to-many). Although all of the configurations are suitable for similar model architectures, many-to-one was chosen for the experiments due to its implementation which allows us to treat all of the languages as one and forgo the need for language differentiation.

2.2 Knowledge distillation

In machine learning, training with more data typically results in better accuracy and generalizability (cite). In pursuit of better accuracy, researchers have utilized ever increasingly larger training datasets, which generally tends to have a positive impact on a model. Training with more data also has added benefits with regards to minimizing bias and may help the model generalize better.

However, having massive amounts of training data can be difficult or impossible to acquire for some languages and comes at the cost of computational efficiency, as larger and more complex NMT models require more energy and increasingly powerful hardware to run. To alleviate this efficiency issue, researchers have proposed various methods to distill the knowledge (Hinton et al., 2015) of large NMT models into smaller, more computationally efficient student models. Being more efficient, these student models can even run on mobile phones.

Furthermore, the knowledge distillation process may not hinder the accuracy of the model, resulting in a student model that is comparable to the accuracy of the teacher model. In some cases, knowledge distillation may even provide a slight increase in accuracy (Jooste et al., 2022). Large models seem to learn a vast amount of information from the training data that is never utilized.

At its core, knowledge distillation is comprised of two objectives. Firstly, useful information must be extracted from the training data by a teacher model and secondly that information must be transferred to the student model (Gou et al., 2020). This approach is convenient, as the teacher model will intrinsically be able to favor beneficial training data with no direct human input. Other researchers have argued that the process of knowledge distillation can be understood as a function matching task, where the primary goal is to align the output function of a smaller, student model with that of a larger, more accurate teacher model (Beyer et al., 2021). In this perspective, the teacher model serves as a proxy for the ideal function that maps input data to desired outputs, encapsulating both the explicit knowledge from labeled data and the implicit knowledge it has learned from its training. Simply put, the objective of KD is to teach a student model to imitate the teacher, whereas with regular machine learning the objective is to imitate the data.However, the exact mechanisms of how knowledge distillation works is still under debate, which is why it is necessary to explore how different types of knowledge distillation methods perform in different scenarios to gain a better understanding on how to utilize these methods to their fullest extent and what types of transfer methods are most beneficial for student training. (Zhang et al., 2018)

One of the hurdles in KD is the capacity mismatch problem, where teacher models are more certain in their predictions, allocating a larger proportion of their probability distribution to certain targets, whereas smaller student models may distribute the probabilities to a wider range of targets. (Chen et al., 2021). Researchers have pointed out that a large teacher-student capacity gap may harm the quality of KD. (Zhang et al., 2023)

KD also seems to help with noisy data, as the teacher model will be able to regularize noisy data points in the data via averaging. We can assume that there will remain some noise even in well processed, high-quality datasets, which the teacher model learns to disregard. This behavior should extend to the student as it learns to mimic the teacher.

Well-trained models will have learned to generalize (cite), which may transfer to the student model. Distilling knowledge is often easier and more efficient than training a smaller model from scratch. This is likely because the teacher model, having been trained on a large dataset, has already learned to identify and prioritize the most useful data points while effectively regularizing noisy or less relevant information. In contrast, training a smaller model is more challenging as there are fewer model parameters to train. The teacher's ability to filter and refine the information simplifies the training process for the student model, reducing the reliance on having large parameter sizes or exceptionally clean data.

2.1.1 Word-level distillation

Word-level distillation (Kim & Rush, 2016) is a method of knowledge distillation in which the student model’s predictions are corrected or guided on a token-to-token basis. In the context of NMT, where translation is a multi-class classification task at each time step, this approach leverages the teacher model's predictions to improve the student model. Specifically, the teacher provides probability distributions at each time step in a sequence, and these predictions are interpolated with those of the student model using a chosen metric, such as Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951). Word-level distillation focuses on aligning the student model's predictions with those of the teacher at the level of individual tokens, helping the student learn more precise word-level translations.

Exposure bias (cite) is a common problem in sequence-to-sequence learning tasks like NMT. Word-level distillation can help mitigate exposure bias by enabling the student to better handle individual token predictions, making it less sensitive to errors that propagate across the sequence in cases where the student makes a bad prediction early on. This can be especially during the early stages of training where the student’s predictions are essentially random.

One major strength of word-level distillation is its efficiency. The student model learns the correct translations of individual words directly from the teacher model, which provides a simple way to improve token-level accuracy. Additionally, the simplicity of this approach is another advantage. Each token is considered independently, without requiring the student model to account for relationships between tokens within a sequence. This reduces computational complexity and makes the method relatively easy to implement.

Despite its strengths, word-level distillation has notable limitations. It is inherently shallow, as it focuses exclusively on individual tokens without accounting for the broader context of a sentence. In natural language, meaning often depends on fixed chunks or collocations of words and on syntactic and semantic relationships within a sentence. By not modeling these dependencies, word-level distillation may fail to capture the nuances of sentence structuring and idiomatic expressions, limiting its ability to distill more complex linguistic phenomena.

Word-level KD works by minimizing the Kullback-Leibler divergence (Kullback & Leibler, 1951) between the predictions of the student and teacher models. This method can be explored with the following example:

Suppose we have a translation task where both a teacher model and a student model are predicting the next word in a sentence. The input sentence is:

> "She decided to bake a..."

The goal is to predict the next word, which in this context should ideally be "cake." Both the teacher and student models output probability distributions for possible next words. These distributions represent how likely each word is to be chosen as the next token.

The teacher model might assign the following probabilities for the next word:

- "cake": 0.80
- "bread": 0.10
- "pie": 0.05
- "cookie": 0.05

The student model might produce a slightly different distribution:

- "cake": 0.60
- "bread": 0.20
- "pie": 0.10
- "cookie": 0.10

KL Divergence measures the difference between the teacher's probability distribution P and the student's probability distribution Q. The formula is:

FIX the formulas!!!!

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x)\ \log\left(\frac{P(x)}{Q(x)}\right)$$

Now, we can DKL for each word:

1. **For "cake":**
   xxxx ≈ 0.222
2. **For "bread":**
   xxxx ≈ −0.070
3. **For "pie":**
   xxxx ≈ −0.035
4. **For "cookie":**
   xxxx ≈ −0.035

Summing these values gives:

DKL≈0.222−0.070−0.035−0.035=0.082

This KL Divergence score indicates a moderate difference between the teacher and student distributions. The student's probabilities are somewhat close but could be improved. For example, the student could increase the probability of "cake" and decrease probabilities for less likely options, like "bread" or "pie." By minimizing KL Divergence during training, the student model gradually aligns its predictions with the teacher's, improving translation quality.

2.1.2 Sequence-level knowledge distillation

Whereas word-level knowledge distillation only deals with the probability distributions of individual tokens at each time step, sequence-level knowledge distillation or SLKD (Kim & Rush, 2016) aims to take into account the complete output of tokens predicted by the student and teacher models. In essence, this method allows the teacher to transfer knowledge about complete sentences instead of individual translations of words. Kim and Rush argued that SLKD addresses a capacity deficiency in student models which are too small to fit noisy data by simplifying these noisy data points. This means that a well-trained teacher model with a larger capacity is able to identify which parts of the training data are useful, thus reducing noise and will be able to transfer useful knowledge about the data to the student.

Other researchers have argued that one of the most important aspects of SLKD is that it does regularize the data and reduction of multimodality (Gu, et al., 2017), which can be thought of as reducing the multitude of ways a sentence from the source language could be expressed in the target language. Essentially, the teacher is able to simplify linguistic features of the original data or reduce alternate, valid translations of a sentence (Zhou et al., 2019) and it is usually assumed that knowledge distillation's reduction of this multimodality in the training data is the key reason why distillation benefits training (Gu et al., 2018). In this sense, regularizing with SLKD instead of using penalty terms such as dropout or L2 regularization helps the student model generalize without restricting model capacity. (Gordon, et al., 2019)

2.1.3 Top-1 Information Enhanced Knowledge Distillation

Top-1 Information Enhanced Knowledge Distillation (TIE-KD) (Zhang et al., 2023) is a word-level knowledge distillation method that incorporates a hierarchical ranking loss to help the student model more effectively learn the most relevant information from the teacher. This approach builds upon the foundational word-level KD method of minimizing the Kullback-Leibler (KL) divergence between the teacher and student model predictions, first proposed by Kim and Rush (2016). By introducing the concept of hierarchical ranking, TIE-KD enhances the learning process by emphasizing the top-1 prediction, i.e., the most likely token in the teacher's probability distribution, while still considering secondary predictions to maintain richer supervision.

TIE-KD shares similarities with the framework of learning with privileged information (Lopez-Paz et al., 2016), where additional information provided by the teacher during training serves as a guiding signal for the student. Specifically, the teacher model not only offers probability distributions for the next word but also supplies auxiliary information about the ranking of candidate tokens, enabling the student to focus on the most important predictions while still learning from less probable tokens.

In TIE-KD, a hierarchical ranking loss function is added to the traditional KD framework to ensure the student model places greater emphasis on correctly predicting the most likely token according to the teacher. This hierarchical approach reduces the noise associated with less relevant tokens and accelerates the student's learning by prioritizing the most informative examples. The ranking loss is typically computed over the teacher's top-k predictions, with the top-1 prediction weighted most heavily.

[formula here]

Here:

- LKL is the Kullback-Leibler divergence between the predictions of the student and teacher models
- L Rank enforces the hierarchical ranking by penalizing the student for incorrectly ordering candidate tokens.
- Alpha is a hyperparameter that balances the two loss components.

By focusing on the most relevant tokens (e.g., top-1 or top-k predictions), TIE-KD reduces the computational overhead compared to methods that equally weigh all vocabulary tokens.

The hierarchical ranking loss ensures that the student model captures not only the correct token but also the relative importance of alternative predictions, improving generalization.

TIE-KD has shown particular promise in settings where the student model is limited in capacity or where the training data is sparse. The privileged information provided by the teacher compensates for these limitations.

While TIE-KD significantly improves the efficiency of word-level KD, its reliance on hierarchical ranking can make it sensitive to the quality of the teacher model. If the teacher's top-1 predictions are incorrect or overly confident, the student may inherit these errors. Additionally, tuning the hyperparameter alpha requires slight experimentation to ensure a balance between the KL divergence and ranking loss.

TIE-KD has been applied successfully in low-resource NMT tasks and domain adaptation scenarios. By leveraging the hierarchical ranking mechanism, student models achieve better translation quality with fewer resources, making TIE-KD a valuable method in multilingual and low-resource NLP settings.

3.1.1 Methods

In this section, we will introduce the model training pipeline. This includes our data sets for training and evaluation, data processing pipeline and the tools used to train and evaluate the models.

3.1.2 Training Data

The training data for the models comes from the NLLB dataset, which is a massive collection of processed parallel data originally released by AllenAI and further processed by Meta AI (AllenAI, NLLB Team, 2022). The project aims to provide training data for a vast amount of languages, including languages that have been previously overlooked in NMT or that are classified as being low-resource. The dataset is composed of various parallel corpora with additional data from bitext mining, a method where parallel sentences are computationally extracted from monolingual data.

For the experiments, a subset of 12 million samples from each language pair {de,lb,sv,af}-en was extracted from the full datasets. This was done in order to create a size-balanced dataset without overrepresenting any individual language pair. Overrepresenting any individual language pair may result in the model developing a preference for a specific language. However, this may not be an issue in the current setup, where all languages are translated into English.

The relevant NLLB language bitexts (from swe, ltz, deu, afr to eng) were acquired through the OPUS database (cite Tiedemann, OPUS, CCMatrix).

1.1.3 OpusFilter

The datasets were processed with OpusFilter (cite Aulamo, M., et al., 2020), which is a parallel corpus processing toolkit. It has many functions, including automatic corpus downloading, processing and filtering, in addition to supporting custom filters written in Python. Simply put, Opusfilter iterates through all of the samples in the dataset and compares each sample in the dataset to a collection of rules defined in a configuration file. (attach to appendix)

	Num. samples
Raw	47960000
Deduplicated	47948299
Filtered	40269225

The objective of the preprocessing pipeline with OpusFilter is to regularize the data and attempt to clean unwanted noise by applying a series of filtering steps. First, the **WhiteSpaceNormalizer** standardizes spacing, removing inconsistencies that could affect tokenization. Next, duplicate entries are eliminated to prevent redundancy and bias in the training data. The **LengthFilter** removes sentences that are too short or too long, ensuring that only reasonable-length sentences are retained. The **LengthRatioFilter** enforces a proportional relationship between source and target sentence lengths, filtering out misaligned or incomplete translations. Finally, the **AlphabetRatioFilter** helps remove noisy data by checking the proportion of valid alphabetic characters, reducing the presence of non-linguistic content such as corrupted or improperly encoded text. These steps collectively enhance the quality and reliability of the training corpus, improving the efficiency and performance of the translation model.

1.1.4 SentencePiece

It is essential to process the data into a format that is better suited for machines. One such way is to tokenize the text, which was done with SentencePiece, an unsupervised text tokenizer and detokenizer developed by Google (cite google/Sentencepiece). SentencePiece operates by learning a fixed-size vocabulary of subword units from any given training data, allowing the model to handle rare and compound words without the need for a vast vocabulary. This is especially important for multilingual machine translation, where maintaining a manageable vocabulary across multiple languages with diverse linguistic structures is particularly challenging. If the data were not tokenized in this way, the model would encounter hundreds of thousands of unique word forms or tokens, which would be inefficient and likely to hinder model performance due to data sparsity. SentencePiece allows us to calculate which subword units can most effectively represent the natural word forms present in the training data.

For instance, take the following Luxembourgish example from our training data:

Original Text:
Wat ass dann elo den déiwere Sënn heivunner?

Tokenized Output:
_Wat _ass _dann _elo _den _déi w ere _Sënn _he iv unner ?

Token IDs:
[1536, 321, 1548, 3657, 178, 851, 14, 10907, 23352, 120, 69, 28806, 214, 444, 3086]

In this example, SentencePiece has segmented the sentence into subword units, with each subword prefixed by the special character "_" to indicate the start of a new word. SentencePiece splits complex words like "déiwere" into smaller parts ("déi", “w” and "ere") which is the most efficient combination from the vocabulary built with SentencePiece. This approach ensures that even words or forms that are rare in the training data can still be represented as combinations of common subword units, reducing the vocabulary size and allowing for better generalization. However, with a large vocabulary size, many words can appear in their entirety in a single segment. SentencePiece thus enables multilingual models to process diverse data more effectively, as it bridges linguistic differences by using subwords that can apply to multiple languages. Furthermore, these character units are replaced with unique integer IDs to make them machine-friendly. This approach is important to improving the efficiency and performance of machine translation models.

3.1.2 Fairseq

Fairseq (Ott, M., et al., 2019) is a versatile, open-source toolkit developed by Meta AI (cite Meta) that provides powerful tools for a variety of sequence modeling tasks, including machine translation, text generation, and summarization. Built on PyTorch, a popular deep learning library, Fairseq enhances PyTorch’s capabilities by offering several advanced features that improve training speed, efficiency, and flexibility. While PyTorch itself is known for its high performance and ease of use in creating neural networks, Fairseq extends these functionalities, cutting down on the manual labor needed to prototype models.

One of the standout features of Fairseq is its optimized support for half-precision (FP16) training, which enables faster computations and reduces memory usage, allowing models to be trained more efficiently. This is especially beneficial for large-scale machine translation models, which typically require extensive training time. Additionally, Fairseq includes fast data loaders and efficient batching mechanisms allowing users to take full advantage of their hardware.

Fairseq also supports training from command-line scripts, a feature that is particularly useful in research settings where multiple experiments may need to be run with varied configurations. This flexibility allows researchers to efficiently train and evaluate multiple models, often required for hyperparameter tuning and testing different architectures. The command-line interface, combined with a wide array of preconfigured model architectures such as the large transformer architecture used in Vaswani, 2017. These preconfigured settings allow users to quickly experiment with model settings. Two of these preconfigured architectures will be used in this study (*transformer* and *transformer_vaswani_wmt_en_de_big)* along with a third derived from the Bergamot project.

1.1.2

Evaluation of the trained models was done with two different test sets. Firstly, the Tatoeba (Tiedemann, J., 2020) test sets from 2023, which is a multilingual parallel corpus created from collections of example translations meant for language learners.

Secondly, the Flores-200 test set (NLLB Team, 2022) was used, which is an evaluation benchmark comprised of samples from hundreds of different articles from Wikipedia making it a very robust benchmark for NMT models. It also serves as a much more challenging testing method than the Tatoeba sets, as it typically contains lengthier and more complex examples dealing with a variety of sub-topics, such as nature, politics, science and travel. Examples from the test sets will not be shown in order to prevent unintended data leakage.

	Num sents (sv)	Num sents (af)	Num sents (de)	Num sents (lb)
Flores	992	992	992	992
Tatoeba	10362	1374	32565	293

1.2.1

The training data was first filtered with OpusFilter (cite), which was used to remove duplicates and samples that contained over 250 tokens.

Lengthy samples can be detrimental for many reasons. The first obvious problem is that these kinds of sentences contain a large amount of information, which can be difficult for models (and humans) to keep track of.

The four source languages are from the same language family and share many features with each other, making it beneficial to treat them as one large language. These languages naturally share many words and word stems, which reduces the complexity of representing the vocabulary. BPE, a subword tokenization method, takes advantage of these shared linguistic structures by segmenting words into frequently occurring subword units. This allows the model to represent a diverse vocabulary with fewer unique tokens, improving efficiency without sacrificing accuracy. Additionally, the entropy of the vocabulary is lower when languages share common words and structures. A lower entropy means that fewer bits are required to encode the vocabulary, leading to reduced memory usage and faster computations. As a result, the model benefits from a smaller vocabulary that is computationally more efficient, while still maintaining translation accuracy.

A single Sentencepiece model was trained on the combined training data with a BPE size of 32k. The validation and testing samples were not included in this step to ensure that they stay isolated from the training process. The Sentencepiece model was used to encode the training, validation and testing sets.

Furthermore, the datasets were binarized using fairseq's built-in fairseq-preprocess function. To maintain consistency across all four languages, the previously generated SentencePiece dictionary was reused, allowing for a shared vocabulary. As a result, only about 0.002% of tokens were replaced with the <unk> token, even though the SentencePiece vocabulary covers 100% of the tokens. This occurrence of <unk> tokens is due to a known quirk in Fairseq's preprocessing mechanism, where certain subword units may not align perfectly with the predefined vocabulary.

1.2.2

The models were trained using the Fairseq (Ott, M., et al., 2019) toolkit. Three different transformer architectures were used: transformer_vaswani_wmt_en_de_big for the teacher model, with transformer and transformer_tiny for the different student models. The transformer_tiny architecture was inspired by the student model proposed in (cite Bergamot project, firefox student.yml):

	transformer_vaswani_wmt_en_de_big	transformer	transformer_tiny
Encoder_embed_dim	1024	512	256
Encoder_ffn_embed_dim	4096	2048	1536
Encoder_attention_heads	16	8	8
Encoder_layers	6	6	6
Decoder_embed_dim	1024	512	256
Decoder_ffn_embed_dim	4096	2048	1536
Decoder_attention_heads	16	8	8
Decoder_layers	6	6	2
Training steps	300k	100k	100k
Methods used	Large teacher	Baseline, sequence-level kd, vanilla kd, TIE-KD	Baseline, sequence-level kd, vanilla kd, TIE-KD

The transformer-based neural machine translation (NMT) models used in this study are defined by several key hyperparameters, each influencing model performance and efficiency. The models examined include *transformer_vaswani_wmt_en_de_big*, *transformer*, and *transformer_tiny*. The *transformer_vaswani_wmt_en_de_big* model is a large-scale transformer architecture based on the work of Vaswani et al. 2017, optimized for high-resource language pairs. The *transformer* model serves as a standard baseline with moderate parameter sizes, while the *transformer_tiny* model is a significantly smaller variant designed for computational efficiency and low-resource deployment.

The encoder is characterized by several parameters that impact the model's ability to process input sequences. The encoder embedding dimension (Encoder_embed_dim) determines the size of the embedding vectors used to represent input tokens. Larger embedding dimensions enable richer representations but increase computational requirements. The encoder feed-forward network dimension (Encoder_ffn_embed_dim) dictates the size of the intermediate layers within the encoder's feed-forward network, with higher values increasing model capacity. The encoder attention heads (Encoder_attention_heads) parameter specifies the number of heads in the multi-head self-attention mechanism, allowing the model to capture multiple aspects of the input sequence simultaneously. The number of encoder layers (Encoder_layers) defines the depth of the encoder stack, with a greater number of layers enabling the model to capture more complex linguistic patterns.

Similarly, the decoder contains similar parameters that affect output generation. The decoder embedding dimension (Decoder_embed_dim) determines the size of the decoder's token embeddings, mirroring the function of the encoder embedding dimension. The decoder feed-forward network dimension (Decoder_ffn_embed_dim) controls the capacity of the decoder's feed-forward layers, influencing the expressiveness of the model. The decoder attention heads (Decoder_attention_heads) parameter regulates the number of attention heads within the decoder's self-attention and encoder-decoder attention mechanisms, facilitating better alignment between input and output sequences. The number of decoder layers (Decoder_layers) specifies the depth of the decoder stack, with smaller values resulting in more lightweight models, such as with the transformer_tiny.

The training process is further defined by the number of training steps (Training steps), which refers to the total optimization steps taken during training. A higher number of training steps allows for more extensive learning but incurs greater computational costs due to a longer training process. Several training methodologies were employed to enhance the efficiency of student models. The large teacher model serves as a high-capacity reference model used for guiding student models through knowledge distillation.

For each of the two student architectures, four different variations were trained with different methods: A baseline version with no knowledge distillation, a model using sequence-level KD (Kim, Y., et al., 2016), a vanilla word-level KD method (Kim, Y., et al., 2016) and finally the word-level KD method described in Zhang, 2023, resulting in a total of 9 different NMT models.

1.2.3

The training process was conducted on the Puhti GPU clusters provided by CSC, leveraging 4 NVIDIA Tesla V100 GPUs. A limit of 300k training steps was set for the teacher model, whereas the students were trained for a maximum of 100k steps with checkpoints every 5k steps. The best performing checkpoint was chosen for the experiments, measured via BLEU. The teacher model required approximately 40 hours to complete training as a result of reaching 300k training steps, reflecting its larger size and complexity. In contrast, the baseline student models were trained in around 20 hours and the smallest student in roughly 8 hours.

1.2.4

Two different datasets were used for testing: The flores200 (NLLB Team, 2022) and the Tatoeba (cite) test set from 2023. Evaluation of the models was carried out by running the completed models in inference mode with fairseq's interactive command, which translates the test set and removes the encoding from the predicted output. The predicted outputs were then compared to the reference translations with sacrebleu (Post, 2018), which scores the predictions using BLEU (Papineni, K., et al., 2002).

4.1 Results and analysis

The results of the experiments are presented in the tables below, with separate sections for the flores200 and Tatoeba test sets. The performance of each model is grouped by architecture and KD method to facilitate comparisons. Additionally, benchmark scores from the best-performing OPUS-MT model according to the Dashboard (cite) and NLLB (cite) models (accessed on 30.04.2024) are included to provide context.

The data reveals that the underlying architecture plays a critical role in model performance. Larger models with more parameters consistently outperform smaller models due to their ability to capture more complex patterns and learn richer representations from the training data. Unsurprisingly, the larger models consistently achieve higher scores across all languages and methods. However, KD proves effective in narrowing the performance gap between the teacher and student models. Across all configurations, student models trained with KD outperform their baseline counterparts. This improvement is consistent across all languages and model sizes, highlighting KD's ability to transfer valuable information from the teacher model. Notably, the extent of improvement varies. In some cases, such as German in the flores200 test set, the increase is modest (+0.3 BLEU with TIE-KD). In others, such as Luxembourgish, the improvement is substantial, with TIE-KD boosting the student model's score by +2.2 BLEU.

The beneficial effect of KD seems to be more pronounced in tiny models, which are significantly constrained in capacity. These models, while inherently less accurate, achieve a proportionally higher boost in BLEU scores when trained with KD compared to their baseline counterparts. For example, in the Tatoeba test set, the TIE-KD tiny student model consistently outperforms the baseline tiny student across all tested languages. This demonstrates that KD remains a valuable approach even in highly parameter-constrained scenarios, offering a practical way to enhance translation quality for lightweight models.

The following tables summarize the results for each test set:

Flores200	Language			
Model config	swe	ltz	deu	afr
best OPUS-MT model	49,8	34,8	41,8	55,1
best NLLB	47,6	46,1	40,7	57,2
teacher	45,8	41,2	39,7	55,4
student_baseline	42,8	36,8	37,6	52,8
student_baseline_tiny	37,6	29,1	31,6	45,8
student_tiekd	43,7	39,0	37,9	53,3
student_vanillakd	43,4	38,3	37,3	52,7
student_seq_kd	42,5	38,0	36,9	52,2
student_tiekd_tiny	38,3	30,5	32,9	46,8
student_vanillakd_tiny	37,7	29,2	32,0	45,9
student_seq_tiny	37,1	28,9	31,2	44,2
Tatoeba	Language			
Model config	swe	ltz	deu	afr
best OPUS-MT model				
best NLLB				
Teacher	59,4	51,9	44,1	59,3
student_baseline	56,0	49,0	41,2	59,0
student_baseline_tiny	50,0	46,2	35,3	55,6

student_tiekd	56,5	53,0	41,9	60,9
student_vanillakd	56,9	51,4	41,5	60,9
student_seq_kd	55,8	51,1	40,3	59,3
student_tiekd_tiny	51,4	49,6	36,2	57,6
student_vanillakd_tiny	51,4	47,8	35,7	56,2
student_seq_tiny	50,2	47,0	35,1	55,4

The table below highlights the translation times (in seconds) for different model configurations on the Flores test set, which was identical in size across all languages.The results seen in the table are the averaged translation time for all languages. Importantly, the results show that the differences in translation time are primarily dictated by the underlying model architecture rather than the specific KD method employed. All methods produced almost identical translation times for a given model size, with any observed variation likely being a random fluctuation rather than a meaningful distinction. It is also important to note that the translation times are highly hardware-dependent, and as such, only the relationship between the times should be explored.

Translation time:

	Baseline	Vanilla KD	TIE-KD	Seq-KD
Teacher	20,2 s	-	-	-
Student	10,3 s	10,2 s	11,6 s	10,8 s
Tiny	6,8 s	6,6 s	6,2 s	6,3 s

As expected, the teacher model, with its larger parameter count and more complex architecture, required the longest time to translate the test set, clocking in at 20.2 seconds. In contrast, the student models achieved significantly faster translation times, with the baseline student configuration translating the same set in 10.3 seconds, nearly cutting the time in half.

The tiny student models were even more efficient, with the baseline tiny configuration completing the translation in just 6.8 seconds, making them over three times faster than the teacher model. This underscores the dramatic efficiency gains of using smaller architectures, particularly in scenarios where computational resources are constrained or low latency is a priority.

It is evident that the KD method itself had little to no impact on translation speed. For example, the TIE-KD student model translated the test set in 11.6 seconds, compared to 10.3 seconds for the baseline student model, but this minor difference is better attributed to random fluctuations rather than any KD approach. Similarly, the tiny TIE-KD model required 6.2 seconds, marginally faster than the baseline tiny model at 6.8 seconds, again reflecting architectural efficiency rather than methodological differences.

The analysis demonstrates that translation efficiency is largely a function of model architecture, with smaller models offering substantial speed gains over the teacher model. The choice of KD method has negligible impact on translation time, reinforcing the idea that KD methods primarily influence accuracy rather than efficiency. These findings further support the viability of smaller student models for use cases requiring rapid translations on resource-constrained devices or real-time applications.

Larger models require more storage space and memory when in use. The following table of model sizes highlights the substantial differences in storage requirements between the teacher (2706 MB), student (837 MB), and tiny (264 MB) configurations. The teacher model, with its large architecture, excels in accuracy and handling complex tasks but demands significant storage, memory, and computational power. The models must be fully loaded into high-speed memory during inference, which makes it impractical to run large models in resource-constrained environments like on mobile devices.

Model sizes

	Size	Parameters
Teacher	2706 MB	209125376
Student	837 MB	60522496
Tiny	264 MB	17138688

As seen in the chart, the student model is over three times smaller than the teacher at 837 MB. Despite its reduced size, it achieves competitive accuracy when enhanced with knowledge distillation methods. Its smaller footprint may make it more suitable for mid-range hardware and applications that prioritize computational efficiency.

The tiny model, at just 264 MB, is highly optimized for environments with strict resource limitations. While it trades some accuracy for size, it is ideal for use on low-power devices or in scenarios requiring real-time processing with minimal hardware.

Smaller models offer several benefits, including reduced storage and memory requirements, faster inference, lower energy consumption, and broader accessibility for deployment in diverse hardware environments. By leveraging KD, these smaller configurations preserve much of the teacher model's performance, making them highly practical for real-world applications where efficiency and scalability are essential.

The parameter count plays a crucial role in determining the trade-off between model capacity and computational cost. The teacher model, with over 209 million parameters, has the highest expressiveness, enabling it to learn and generalize complex linguistic patterns effectively. However, this increased capacity also leads to higher memory usage and computational requirements, making it impractical for certain scenarios. In contrast, the student model, with approximately 60 million parameters, retains much of the teacher's performance while significantly reducing the resource burden. The tiny model, with just 17 million parameters, achieves even greater efficiency, allowing it to run on low-power devices, though at the cost of some translation accuracy.

5. Discussion

In this section, we will discuss the results and experiments in more detail.

The results provide insights into the performance of different models and KD methods across various languages. One of the most notable findings is the consistent improvement achieved by TIE-KD over baseline models, particularly for smaller architectures. This highlights the effectiveness of TIE-KD in extracting and transferring useful information from the teacher model to the student model. For instance, the significant BLEU score improvements underscore the ability of TIE-KD to address challenges associated with small model capacities.

The vocabulary size of 32,000 tokens used in this experiment may be slightly too large for the given language configuration of four very similar languages, which could affect the efficiency and performance of the models. A larger vocabulary size allows for better representation of diverse languages and reduces the need for subword segmentation. However, it also increases the model's complexity, requiring more parameters and computational resources to process. For languages with smaller corpora or less linguistic diversity, such as those in this study, a smaller vocabulary size might be more appropriate. A reduced vocabulary could improve model efficiency by decreasing memory requirements and speeding up training and inference times, without significantly compromising accuracy. This suggests that further optimization of the vocabulary size, tailored to the specific language pair, could yield better results in terms of both computational efficiency and model performance.

It is clear that both of the tested word-level methods provided a small to moderate increase in accuracy compared to the baseline models. TIE-KD slightly outperforms vanilla KD, which suggests that it is a more optimized method of learning. Seq-KD was by far the weakest method of the three, often falling slightly below the baseline score. TIE-KD seems to perform better when the model architecture is smaller, suggesting that the method can be used to extract additional useful information from the data. This effect seems to be more pronounced the smaller the model is: the medium-sized models were slightly better than the baseline but the very small models were significantly more accurate compared to the baseline model.

Training costs should also be taken into consideration, as both TIE-KD and Word-KD are much slower to train. This is due to differences in the implementation, as both methods utilize an approach where the teacher model is kept in memory during training and its predictions and loss calculations are computed in parallel. It should also be mentioned that Seq-KD requires additional work in the form of retranslating the original dataset. This newly translated data must also be encoded and binarized in a similar fashion to the original model, indirectly adding to the training time.

- Compare output of models from different languages into English?

- add more comparisons between models and analyze

Teacher model, Flores sample	This offers a good opportunity to see the Aurora borealis, as the sky will be dark more or less around the clock.
[LB] Dat bitt eng gutt Geleeënheet, d'Aurora borealis ze gesinn, well den Himmel méi oder manner ronderëm d'Auer däischter ass.	This provides a good opportunity to see the aurora borealis because the sky is more or less around the ear dark.
[SV] Detta är ett bra tillfälle att se norrskenet, eftersom himlen är mörk mer eller mindre dygnet runt.	This is a good time to see the northern lights, as the sky is dark more or less around the clock.
[DE] Dies bietet eine gute Gelegenheit, das Nordlicht zu sehen, da der Himmel mehr oder weniger rund um die Uhr dunkel ist.	This provides a good opportunity to see the Northern Lights as the sky is more or less dark around the clock.
[AF] Dit bied 'n goeie geleentheid om die Aurora borealis te sien, aangesien	It provides a good opportunity to see the Aurora borealis, as the air is dark almost

die lug byna die heeltyd donker is.	all the time.

The scores also suggest that having multilingual data can be helpful in many cases. For example, the model performs much better when translating Luxembourgish compared to the best-performing OPUS-MT model, which was trained only on ltz-eng data.

The performance of the big transformer model suggests that it may have untapped potential, limited primarily by the quality and quantity of the training data rather than its architecture. Larger models, such as the big transformer, are able to leverage vast amounts of information due to their larger parameter space. However, if the training data is insufficient in volume or lacks diversity and richness, the parameter space may not be adequately saturated, resulting in wasted potential. This could explain why the performance gap between the big transformer and the baseline model is not as pronounced as expected. The big transformer could likely demonstrate significantly better results with access to better data through larger datasets, more diverse samples, or higher-quality translations. This highlights the importance of high-quality, comprehensive datasets in maximizing the efficacy of larger models.

Model	Type-token ratio	Length ratio
Teacher	0.2557	4.9040
Student_baseline	0.2506	4.8967
Student_TIE-KD	0.2517	4.9105
Student_Vanilla KD	**0.2521**	4.9083
Student_Seq-KD	0.2509	4.9002
Tiny_student_baseline	0.2416	4.8607

Tiny_student_TIE-KD	0.2428	4.8834
Tiny_student_Vanilla KD	**0.2442**	4.8727
Tiny_student_Seq-KD	0.2423	4.8682

The type-token ratio and length ratio provide insights into vocabulary diversity and sentence length across models. The teacher model exhibits the highest TTR (0.2557), indicating a broader vocabulary usage, while the tiny student models have the lowest TTR, suggesting a more repetitive vocabulary. This reduction in lexical diversity in smaller models could be a trade-off for efficiency and compression. However, a clear trend in the data shows that all KD methods promote a measurable improvement in the vocabulary of the models.

- Explain length ratio
- Where do the student models struggle?
- Student models -> More consistent?
- Does word kd produce better word results?

The accuracy of the student usually doesn't exceed the teacher's capabilities, but is usually quite similar in medium-sized student models, which might mean that a more capable teacher could result in better performing students. Additionally, the student model learns to mimic the teacher, adopting its ability to handle noise in the data. All of the models were trained on the same training data, but it would also be possible to train the teacher with more training data or use an existing state-of-the-art NMT model as a teacher.

The results could further be improved by combining Seq-KD and Word-KD to achieve better overall results by leveraging the strengths of both approaches. Seq-KD simplifies training data by reducing multimodality, making it easier for the student model to learn consistent patterns, while Word-KD refines token-level predictions by aligning them with the teacher's probability distribution. When used together, Seq-KD provides a strong foundation by structuring sentence-level knowledge, and Word-KD further enhances accuracy by fine-tuning individual token choices. This complementary combination leads to improved translation quality and more effective knowledge transfer.

One potential issue of this study is whether it can truly assess the multilingual scalability of KD, given that the selected languages are structurally and lexically similar. Since all source languages belong to the same language family and share significant vocabulary, grammar, and syntax, it is not clear whether the improvements observed through KD can be generalized to more diverse multilingual settings. The shared linguistic features may lead to knowledge transfer that is inherently easier, meaning the observed benefits could be amplified when compared against a scenario involving more distant language pairs. This raises the question of whether KD would perform as effectively in a model trained across unrelated languages with vastly different morphological and syntactic structures. Future studies should explore a broader language typology to determine whether the advantages of KD in multilingual NMT extend beyond closely related languages.

The student models in this study were trained using fp16 (half-precision floating point format) arithmetic, which reduces memory usage and speeds up training compared to full fp32 precision. However, further compression could be achieved through quantization, which lowers the precision of numerical representations, allowing models to be stored and executed more efficiently. Recent advancements in fp8 precision and mixed-precision training suggest that models could be made even smaller without significant loss in accuracy. Quantization techniques, such as post-training quantization (cite) or quantization-aware training (cite), could further reduce computational overhead and enable deployment on lower-power devices. Future work should explore these techniques to assess whether they maintain translation quality while improving efficiency, particularly in resource-constrained environments.

Additional methods could be used to improve the accuracy of the models or compress them further. For example, by using model folding, an advanced compression technique that aims to reduce the size and computational complexity of neural networks by merging structurally similar neurons while preserving the model’s overall functionality (Wang et al., 2025). In the context of NMT, this approach can be particularly beneficial for knowledge-distilled models, where redundant or highly correlated neurons often emerge due to the transfer of knowledge from a larger teacher model (cite). By identifying and combining similar neurons, model folding reduces the number of parameters and memory footprint, making the model more efficient for deployment on resource-constrained devices. Unlike traditional pruning, which removes neurons entirely, model folding preserves key learned representations while optimizing resource usage. Future research could explore how model folding interacts with knowledge distillation.

Another way to improve accuracy could be to utilize more advanced KD methods.

6. Conclusion

Recap the whole paper