CULL-MT: Compression Using Language and Layer pruning for Machine Translation

Anonymous ACL submission

Abstract

Multilingual machine translation models often outperform traditional bilingual models by leveraging translation knowledge transfer. Recent advancements have led to these models supporting hundreds of languages and achieving state-of-the-art results across various translation directions. However, as these models grow larger, their inference operations become increasingly costly. In many use cases, there is no need to support such a wide range of language pairs, as translation is typically needed in only a few selected directions. In this paper, we present CULL-MT, a compression method for machine translation models based on structural layer pruning and selected language directions. Our approach identifies and prunes unimportant layers using a greedy strategy, then mitigates the impact by applying knowledge distillation from the original model along with parameter-efficient fine-tuning. We apply CULL-MT to the NLLB-3.3B and LLaMA3.1-8B-Instruct models. In a multi-way translation scenario (Persian, French, and German to English), we find the NLLB-3.3B model to be robust, allowing 25% of layers to be pruned with only a 0.9 spBLEU drop. However, LLaMA3.1-8B-Instruct is more sensitive, with a 2.0 spBLEU drop after pruning 5 layers.

1 Introduction

In recent years, multilingual neural machine translation (NMT) models have become an efficient solution for translating multiple language pairs, compared to using separate models for each translation direction (Aharoni et al., 2019). Additionally, multilingual NMT models (Firat et al., 2016) outperform bilingual NMT models (Bahdanau et al., 2015), particularly in low-resource language pairs, due to the transfer of knowledge from high-resource language pairs (Dabre et al., 2020) . Therefore, recent multilingual NMT models have increasingly focused on expanding the number of supported languages. Notable models like NLLB (Costa-jussà et al., 2022) and MADLAD (Kudugunta et al., 2024) now support 200 and 400 languages, respectively, with many of these being low-resource languages.

Despite their impressive performance, multilingual NMT models have grown significantly in size, often containing billions of parameters. This increase has made their deployment and inference computationally expensive, requiring substantial hardware resources. The parameter count of multilingual NMT models, such as NLLB (Costa-jussà et al., 2022) and MADLAD (Kudugunta et al., 2024), has surged in recent years, mirroring the trend in language models, which gave rise to the term *Large Language Models* (LLMs). This rapid expansion raises concerns about the cost and practicality of using these models in real-world applications, particularly in resource-constrained environments. As a result, compressing large models without significantly impacting performance has become a critical research challenge.

While recent multilingual NMT models support hundreds of languages, many medium- and small-sized translation systems require only a few translation directions. Compressing massive translation models to focus on the relevant language pairs can result in significant computational savings. Our approach addresses this need by compressing the model with minimal performance sacrifice in the selected translation tasks.

Both multilingual NMT models and large language models are designed to handle a wide range of tasks. While multilingual NMT models can translate hundreds of language directions, LLMs are versatile enough for various natural language processing (NLP) tasks, including multilingual translation (Zhao et al., 2023). Our approach focuses on compressing these models for specific tasks. For multilingual NMT, we reduce model size while maintaining performance on key translation

directions. Similarly, for LLMs, we compress the model specifically for multilingual translation of selected language pairs, showcasing the versatility and architecture-agnostic nature of our method.

For addressing this issue, we introduce CULL-MT, a method that compresses models by focusing on the key translation directions relevant to specific use cases, achieving notable computational savings with minimal performance sacrifice. Our method employs a greedy structural pruning technique, wherein we iteratively assess the importance of each layer by comparing the model's performance with and without that layer. If removing a layer results in only a minimal performance drop, it is considered less critical and is subsequently pruned. This iterative process continues until the performance falls below a predefined threshold. After the pruning phase, the model is fine-tuned to recover any performance loss, ensuring effectiveness for the targeted translations.

The following key contributions highlight the effectiveness of our CULL-MT approach:

We applied our approach to the NLLB-3.3B model, demonstrating that it is considerably robust against layer pruning. In a multi-way scenario (translating from Persian, French, and German to English), we were able to prune 12 out of 48 layers, with the pruned model experiencing only a 0.9 drop in spBLEU score. In a single-way scenario (translating from English to Persian), CULL-MT allowed us to prune 15 layers while resulting in a 1.2 drop in spBLEU score.

Our approach was also tested on the LLaMA3.1-8B-Instruct model, revealing its sensitivity to layer pruning in translation tasks. In the multi-way scenario, CULL-MT pruned 5 layers, resulting in a 2.0 drop in spBLEU score.

Analyzing the importance of layers in the multi-way translation task for both NLLB-3.3B and LLaMA3.1-8B-Instruct revealed critical insights. Our findings indicate that the first layer in the encoder and the decoder of the NLLB-3.3B model are essential, while other layers have a considerably lesser impact on performance. In contrast, the LLaMA3.1-8B-Instruct model exhibits more significant important regions, with the first two and last five layers being crucial, alongside several important layers in the middle of the model.

The remainder of this paper is organized as follows: Section 2 reviews common compression methods and layer-wise pruning techniques. Section 3 presents the CULL-MT methodology, while Section 4 outlines the experimental setup. Section 5 reports the results of CULL-MT and compares them with other approaches. Section 6 provides an analysis of layer importance, and Section 7 concludes the paper. Finally, Section 8 discusses the limitations.

2 Related Works

2.1 Compressing Massive Models

The rapid increase in the size of pre-trained language models (Devlin et al., 2019), such as GPT-3 (Brown et al., 2020) and LLaMA (Touvron et al., 2023), has led to substantial interest in model compression techniques to reduce inference costs and memory footprints. Common methods include quantization (Jacob et al., 2018), which reduces precision in model weights, and knowledge distillation (Hinton, 2015), which transfers knowledge from a larger model to a smaller one while retaining performance on downstream tasks. Pruning, another effective method, involves removing unimportant weights or neurons from a model (Han et al., 2016) to achieve a more compact version without significant performance degradation.

In addition to advancements in model compression, significant progress has been made in parameter-efficient fine-tuning (PEFT) methods, which facilitate the adaptation of LLMs for consumer-level devices. These techniques allow for effective fine-tuning while minimizing the computational burden (Lialin et al., 2023). One prominent example is Low-Rank Adaptation (LoRA), which utilizes low-rank matrices to enhance fine-tuning efficiency (Hu et al., 2022). Furthermore, some approaches have combined various compression and PEFT methods to maximize efficiency. For example, QLoRA employs 4-bit double quantization of the original model alongside LoRA weights, facilitating low-resource fine-tuning (Dettmers et al., 2023). Moreover, Minitron utilizes a combination of pruning and knowledge distillation to create smaller, efficient language models from larger pre-trained architectures (Muralidharan et al., 2024).

2.2 Structural Pruning

Pruning (LeCun et al., 1989) is a key approach for compressing neural networks, aiming to identify and remove less important weights using various methods. While unstructured pruning allows for the sparsification of specific weights, it often necessitates specialized hardware for speed improve-

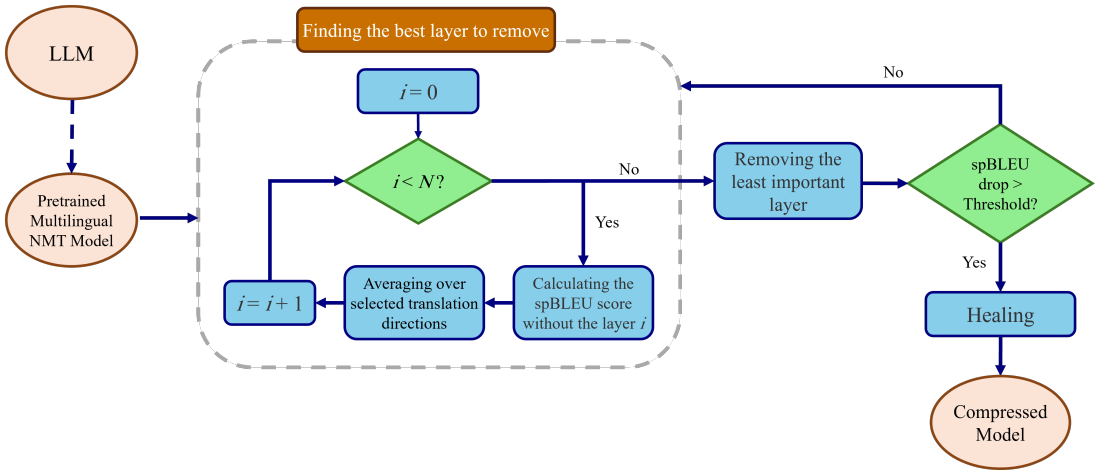

Figure 1: Overview of the CULL-MT approach. It iteratively prunes a multilingual translation model by identifying and removing the least important layers while preserving performance in key translation directions.

ments (Cheng et al., 2024). In contrast, structural pruning (Molchanov et al., 2017) focuses on removing entire filters, blocks, or layers, providing efficiency that is not dependent on specific devices (Cheng et al., 2024).

Voita et al. (2019) indicated that the top layers of pre-trained transformer (Vaswani et al., 2017) models are often specialized for the original training objectives. Therefore, Sajjad et al. (2023) showed that the higher layers are not essential for fine-tuning on downstream tasks and they can be pruned. Peer et al. (2022) introduce a Greedy Layer Pruning (GLP) method that iteratively selects layers to remove, optimizing performance on GLUE tasks with encoder-only models. While their focus is on GLUE tasks, we concentrate on translation tasks involving encoder-decoder and decoder-only architectures.

Recent works concentrate on pruning LLMs. Ma et al. (2023) developed LLM-Pruner, which uses structural pruning to remove non-critical coupled structures based on gradient information, effectively compressing large language models while maintaining their core functionality. This method allows for performance recovery with tuning techniques like LoRA and demonstrates satisfactory zero-shot task capabilities across various models. Furthermore, Gromov et al. (2024) highlighted that deeper layers often exhibit highly similar representations. This similarity allows for significant model compression, particularly by pruning deeper layers starting from the penultimate layer, which can be achieved without major performance loss when combined with light fine-tuning.

3 CULL-MT

Our approach focuses on structurally pruning a multilingual translation model by iteratively identifying and removing the least important layers, while preserving performance on key translation directions. The procedure involves two nested iterations: an inner loop for identifying the best layer to remove, and an outer loop that repeats this procedure until the model's performance drop exceeds a predefined threshold. The process is visualized in Figure 1, which gives an overall view of our approach.

Layer Importance Evaluation. The core of our methodology is evaluating the importance of each layer. For this, we measure the model's performance with and without each layer using a development translation dataset. Specifically, we assess the spBLEU (Costa-jussà et al., 2022; Papineni et al., 2002) score for the selected translation directions. After testing each layer, we average the spBLEU scores across all directions of interest to determine a layer's overall importance. The layer whose removal results in the smallest performance drop is considered the least important and is pruned.

Iterative Layer Pruning. The inner loop of the algorithm — referred to as "Finding the best layer to remove" — iterates over the model's layers and evaluates the importance of each layer as described above. Once all layers have been evaluated, we identify the best layer to prune, i.e., the one which minimally impacts the model's performance.

After each pruning step, the outer loop repeats the inner procedure, re-evaluating the remaining layers. By doing so, our approach continually as-

sesses how the removal of one layer affects the importance of other layers, ensuring that we prune in a way which maintains maximum performance for the selected directions.

By stopping the pruning process at a specific threshold and utilizing efficient fine-tuning techniques, running the CULL-MT approach on the NLLB-3.3B and LLaMA3.1-8B-Instruct models takes less than a day. This is not a significant concern, as the process is intended to run only once to create the compressed model.

Our greedy approach identifies local solutions for determining the optimal set of layers to prune. Although we explored other methods, such as dynamic programming, they did not yield better results. This is because the effect of removing a specific layer depends on the layers that have already been removed, making it difficult to generalize. Therefore, finding the truly optimal solution requires evaluating all possible combinations of layer removals, which is computationally infeasible due to the sheer number of configurations.

Focus on Special Translation Directions. Our approach focuses on preserving performance for specific translation directions by evaluating layer importance based on the averaged performance across these directions. In real-world scenarios, where certain translation directions may hold greater importance, we can assign higher weights to those directions when calculating the spBLEU score drop during averaging. This allows the pruning process to prioritize key translation directions while still considering others. More generally, each direction can be assigned a custom weight based on its priority, providing flexibility to adapt the pruning process to real-world needs and focus on critical translation tasks.

Healing. To restore any performance loss after pruning, we perform fine-tuning using sequence-level knowledge distillation (Kim and Rush, 2016) from the original model to the pruned model. Fine-tuning the pruned model with new datasets can result in catastrophic forgetting (De Lange et al., 2022). To address this, we utilize knowledge distillation from the original model, which helps maintain performance consistency and mitigate the risk of forgetting (Li and Hoiem, 2018).

To leverage sequence-level knowledge distillation, we use a generated dataset. The source sentences are drawn from the training datasets, while the target sentences are the translations produced by the original model. This generated dataset encompasses all the selected translation directions.

As recent multilingual NMT models and LLMs continue to grow in size, fine-tuning all their parameters has become increasingly challenging. Utilizing PEFT methods presents a viable solution (Lialin et al., 2023). Specifically, we employ the LoRA approach, which freezes the original model and injects trainable low-rank matrices into specific layers (Hu et al., 2022). Additionally, LoRA is particularly effective with smaller training datasets, making it an ideal option for efficient fine-tuning in our context (Zhang et al., 2024).

4 Experiments Setup

4.1 Models

In our experiments, we utilized two models. The first is the NLLB-3.3B model (Costa-jussà et al., 2022), a multilingual NMT model supporting 200 languages, which follows an encoder-decoder architecture with 24 layers in both the encoder and decoder. While NLLB is designed to handle a wide range of language directions, our compression approach focuses specifically on a subset of selected translation directions. We used the NLLB model as a fully pre-trained model without further fine-tuning.

The second model is the LLaMA3.1-8B model (Meta, 2024; Dubey et al., 2024), an LLM known for its openness and strong performance across various NLP tasks. This model follows a decoder-only architecture and contains 32 layers. Although LLaMA3.1-8B-Instruct is versatile across various NLP tasks, we focused on compressing it specifically for the task of multilingual translation in certain language directions. Before applying the compression, we performed instruction-tuning on it to enable zero-shot translation and improve the model's translation performance for this specific task across translation directions of interest.

4.2 Datasets

The training dataset is essential for enhancing the performance of both models in specific translation tasks, and we use the NLLB dataset (Costa-jussà et al., 2022) as the source for our training data. For the NLLB-3.3B model, the healing procedure utilizes a training dataset consisting of 160,000 parallel sentences for each selected direction, which is critical for effectively refining the model. We conducted experiments on various training data sizes to determine that 160,000 sentences is op-

timal for our needs, as detailed in Appendix A. In contrast, recent studies have shown that large language models do not require extensive datasets for fine-tuning in machine translation tasks (Zhang et al., 2023). Consequently, we utilize the same 15,000 parallel sentences for each selected direction in the LLaMA3.1-8B-Instruct model for both the instruction-tuning and healing processes.

The `dev` dataset plays a critical role in the evaluation process, where it is employed to calculate the importance of each layer during the pruning procedure. For this purpose, we used the development subset of FLORES-200 dataset (Costa-jussà et al., 2022), which provides high-quality and reliable multilingual translation data.

To evaluate the final pruned model and compare it against the original, we utilized the `test` subsets from Flores-200. Additionally, for some developmental experiments, we used the `test` subset of the NTREX dataset (Federmann et al., 2022).

4.3 Data Preprocessing and Prompt Format

We did not perform any special data preprocessing for the development and test datasets, because they are already have high quality. The training data was sourced from the NLLB dataset, which has been preprocessed by the NLLB team and sorted based on cosine similarity scores between source and target sentences using the LASER3 model (Heffernan et al., 2022). However, we observed that very short phrases or sentences tended to be located at the beginning of the dataset. To address this, we selected sentences which contained more words than the average for each specific language within the dataset. This approach allowed us to maintain a focus on more meaningful and informative translations during the training process.

For instruction tuning and healing of the LLaMA3.1-8B-Instruct model, we used the default template of the model to ensure compatibility with the original model. Additionally, we adopted the prompt format described by Zhang et al. (2023) to establish a clear boundary around translated sentences. For the system prompt, we utilized the following message: `You are a translator who translates messages from {source language} to {target language}` (Giray, 2023).

4.4 Efficient Fine-tuning

We fine-tune the models at different stages to optimize their performance. For LLMs, we initially fine-tune them to improve their zero-shot performance on the translation task. After pruning layers from the models, we fine-tune them again to *heal* the damage caused by removing layers. However, if the model already exhibits excellent performance in translation, there is no need for initial fine-tuning. Therefore, for multilingual NMT models such as NLLB-3.3B, we only perform fine-tuning in the healing phase.

For our experiments, we used an NVIDIA RTX 3090 with 24 GB of memory. To fit a relatively model such as LLaMA3.1 (with 8 billion parameters) inside the GPU's memory, we utilized the `bfloat16` data type, which uses two bytes per parameter instead of four, thereby halving the memory requirement (Wang and Kanwar, 2019). Additionally, *gradient checkpointing* was applied to save memory by recomputing intermediate activations during the backward pass instead of storing them in memory (Chen et al., 2016).

We also employed ZeRO-Offload, which offloads optimizer states from the GPU to the CPU, allowing us to free up valuable GPU memory for other tasks (Ren et al., 2021). While offloading increased training time, it enabled us to increase the batch size, which positively impacted the overall fine-tuning process. To further handle larger batch sizes, we employed *gradient accumulation*, which splits the gradient calculation over multiple forward and backward passes, reducing the memory requirements for each pass.

Using these techniques, the memory consumption during fine-tuning is nearly equivalent to that of inference, ensuring efficient use of GPU resources. While these methods allowed us to fine-tune all parameters of both models, we opted to use the LoRA technique. As will be shown in Appendix B, when the training data is limited, LoRA can outperform full parameter fine-tuning in terms of both efficiency and translation performance.

For the initial fine-tuning of the LLaMA3.1-8B-Instruct model, we utilized the optimal LoRA hyperparameters identified by Zhang et al. (2023). However, during the healing process, we adjusted the LoRA rank to 128 and the LoRA α to 256 to enhance the model's recovery by increasing the number of trainable parameters. In addition, we applied the same LoRA rank and alpha settings (128 and 256, respectively) to all linear weights in the pruned NLLB model. To further enhance the robustness of our experiments, we implemented a LoRA dropout rate of 0.1 across all trials.

4.5 Evaluation Metrics

To evaluate the performance of our models, we employed three key metrics: spBLEU score, number of removing layers, and inference speed. The spBLEU score was calculated using the `SacreBLEU` package with the tokenization of the Flores-200 dataset (Costa-jussà et al., 2022; Papineni et al., 2002; Post, 2018). Additionally, we measured inference speed by counting the number of tokens generated per second for batch size 1, providing insights into the models' efficiency during deployment. Together, these metrics offer a comprehensive view of our models' translation capabilities and operational efficiency.

4.6 Evaluation Scenarios

We tested our approach in two distinct scenarios: multi-way translation and single-way translation. In the multi-way setting, we aimed to compress the model for translating from German, French, and Persian to English. While German and French share some similarities, Persian is notably different, providing a diverse challenge for our method. Conversely, in the single-way translation scenario, we focused on compressing the model specifically for translating from English to Persian. This approach allowed us to evaluate the model's performance in generating translations for a target language distinct from English.

5 Results

In this section, we present a comparative analysis of our results for LLaMA3.1-8B-Instruct with other approaches for identifying unimportant layers, focusing on the spBLEU score of pruned models. We then compare our final compressed NLLB-3.3B and LLaMA3.1-8B-Instruct models with their original versions in both multi-way and single-way translation tasks, evaluating them in terms of number of pruned layers, token generation speed, and spBLEU score.

5.1 Comparing Layer Pruning Approaches

We compared our approach with Top-N and Blockwise pruning methods. In the Top-N approach, the top N layers of the model are simply removed. This method has shown good results in encoder-only language models (Voita et al., 2019; Sajjad et al., 2023) and was employed by LLM-Pruner (Ma et al., 2023) in decoder-only layer-wise scenarios. In contrast, the Blockwise approach removes a block of layers, typically starting from the penultimate layers (Gromov et al., 2024). We evaluated these methods in a multi-way translation scenario using the LLaMA3.1-8B-Instruct model, fine-tuned for multilingual translation and pruned by 4 layers. Although all approaches include a healing phase, no healing was performed in this comparison to focus solely on the identification of unimportant layers. We specifically compared the number of pruned layers and spBLEU scores across three translation directions: French, German, and Persian to English.

Approach	Fr→En	De→En	Fa→En
Original-FT	48.8	47.1	38.9
CULL-MT	**45.8**	**43.2**	**33.3**
Top-N	1.4	2.0	2.0
Blockwise	6.3	7.7	8.0

Table 1: Comparison of spBLEU scores after pruning 4 layers using different methods on the LLaMA3.1 model across three translation directions. The first row shows the performance of the original fine-tuned model. Note that no healing is performed after pruning.

The results, presented in Table 1, show that the last layers of the model are crucial, and removing them—or even a block of layers near them—leads to a significant drop in performance. In the Top-N approach, layers 28, 29, 30, and 31 were removed, while in the Blockwise approach, layers 25, 26, 27, and 28 were pruned. However, our approach successfully identified unimportant layers, allowing us to remove layers 10, 11, 19, and 26 while maintaining a spBLEU score close to that of the fine-tuned model. Note that no healing is performed after pruning.

5.2 Pruning Stopping Threshold

We ran the layer pruning algorithm without any stopping threshold to compare the results of pruning NLLB-3.3B and LLaMA3.1-8B-Instruct models in the multi-way scenario. We calculated the spBLEU score across all directions after pruning each layer and averaged the spBLEU scores.

The results, presented in Figure 2, demonstrate that the NLLB-3.3B model, being a natural multilingual NMT model, is more robust against layer pruning. In contrast, the fine-tuned LLaMA3.1-8B-Instruct model showed significant sensitivity to the pruning. After removing 40% of its layers, the model performance nearly dropped to zero, while

the NLLB-3.3B model maintained an average spBLEU score of 40 after losing the same percentage of layers. As such, we empirically chose the stopping threshold for the NLLB-3.3B model to be a drop of 3.0 in the spBLEU score, while for the LLaMA3.1-8B-Instruct model, it was chosen as 5.0 spBLEU score in both multi-way and single-way scenarios.

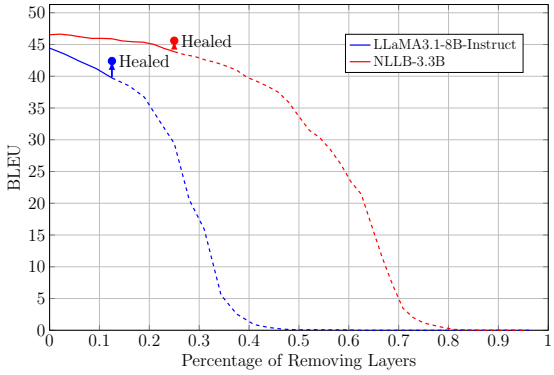

Figure 2: Comparison of layer pruning effects on NLLB-3.3B and LLaMA3.1-8B-Instruct models in a multi-way translation scenario. The NLLB-3.3B model maintains more consistent performance after pruning, whereas the LLaMA3.1-8B-Instruct model shows a steeper performance decline as layers are removed.

5.3 CULL-MT Evaluation

We applied our approach, which includes instruction tuning (for the LLaMA3.1-8B-Instruct model), identifying unimportant layers, pruning them, and healing the model through knowledge distillation.

The results are presented in Table 2. For NLLB-3.3B, the model demonstrated considerable robustness against layer removal; despite the 3.0 BLEU drop threshold, 12 layers could be pruned in the multi-way scenario and 15 layers in the single-way scenario, representing 25% and 31% of the total layers, respectively.

Conversely, the LLaMA3.1-8B-Instruct model was more sensitive to layer pruning, allowing only 5 and 4 layers to be pruned from the overall 32 layers in the multi-way and single-way scenarios, respectively. Although the performance gap between the pruned and the original (or instruction-tuned) LLaMA3.1-8B-Instruct model was significantly greater than that of the NLLB-3.3B model, the healing process proved to be more effective for LLaMA3.1-8B-Instruct. It successfully mitigated much of the spBLEU drop, and in the single-way scenario, the healed model even outperformed the instruction-tuned version.

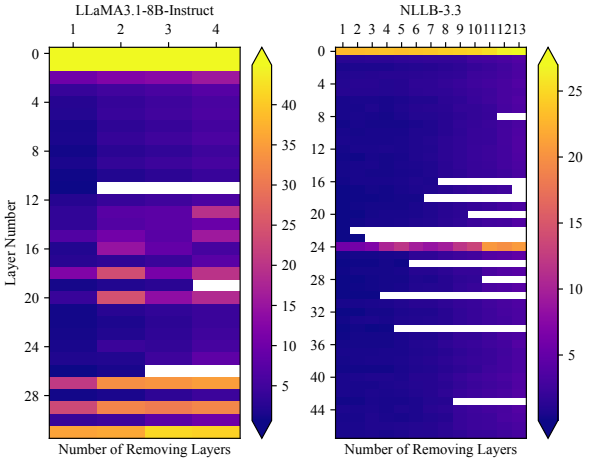

Figure 3: Layer importance during the CULL-MT pruning process, represented by the spBLEU score drop.

Additionally, layer-wise pruning had a more noticeable impact on the generation speed (in terms of tokens per second) for the LLaMA3.1-8B-Instruct model. As it is a decoder-only architecture, which tends to have slower generation speeds, removing layers noticeably improved its generation speed. In contrast, removed layers in the NLLB-3.3B model belong to its encoder, which is a faster module compared to a decoder (Vaswani et al., 2017). Thus, pruning these layers did not significantly affect the model's generation speed.

6 Layer Importance Analysis

In this section, we compared the important layers of both models in the multi-way scenario during the process of identifying and removing unimportant layers. The results, illustrated in Figure 3, indicate that the color of each layer corresponds to the average spBLEU score drop. Brighter colors indicate greater spBLEU score drops, meaning the layer is more important, while darker colors signify smaller drops, indicating less importance. The white color denotes layers that were pruned.

In the NLLB-3.3B model, the first layer of the encoder and the first layer of the decoder were identified as the most important layers. The model demonstrated considerable robustness against layer removal, with other layers not becoming crucial when some were pruned. Conversely, the LLaMA3.1-8B-Instruct model was highly sensitive to layer pruning. The first two layers were critical; removing them caused a significant collapse in performance. Additionally, the last layers (especially 27, 29, and 31) were also very important, and their removal led to substantial drops in performance.

Model	Step	Multi-way				Single-way	
		De→En	Fa→En	Fr→En	Tok/s	En→Fa	Tok/s
NLLB-3.3B	Original	48.4	41.5	49.7	60.9	31.8	62.7
	Pruned	46.1	37.8	47.7	68.7	28.7	66.6
	Healed	47.7	39.7	49.5	68.7	30.6	66.6
LLaMA3.1-8B-Instruct	Instruction-Tuned	46.9	38.3	48.0	35.6	26.0	40.9
	Pruned	41.6	30.7	43.2	43.6	17.2	46.3
	Healed	45.2	36.0	46.0	43.6	27.3	46.3

Table 2: spBLEU scores for the NLLB-3.3B and LLaMA3.1-8B-Instruct models across multi-way and single-way translation scenarios. The table includes results for each of the following steps: the original (or instruction-tuned), pruned, and healed models to demonstrate the effectiveness of the healing process. NLLB-3.3B pruned 12 layers for multi-way and 15 for single-way translation, while LLaMA3.1-8B-Instruct pruned 5 and 4 layers, respectively.

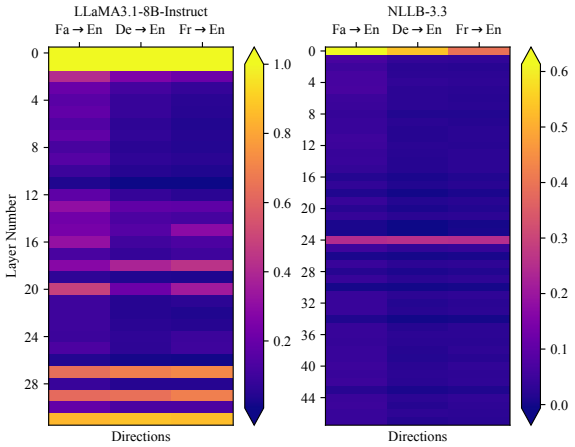

Figure 4: Averate layer importance of NLLB-3.3B and LLaMA3.1-8B-Instruct across Fa→En, Fr→En, and De→En translation directions during pruning.

This sensitivity explains why approaches that remove the top N layers or a block of penultimate layers resulted in large spBLEU score losses. Notably, while the first and last layers were particularly vital, the middle layers of the LLaMA3.1-8B-Instruct model were important as well; their significance increased with layer removal, suggesting that it is advisable to avoid pruning layers from this region.

We also analyzed the importance of layers across different translation directions for both models in the multi-way scenario. To accomplish this, we executed our algorithm to prune 4 layers from the LLaMA3.1-8B-Instruct model and 12 layers from the NLLB-3.3B model, calculating the spBLEU score drop for each layer across all directions for each model. Since the original models have different baseline scores for each direction, we normalized the dropped spBLEU scores by dividing them by the corresponding baseline values. This normalization enables comparability of spBLEU scores across different directions.

The results, depicted in Figure 4, indicate that certain layers are generally important across all directions. However, we observed that removing the unimportant layers led to a more significant decrease in the spBLEU score for the Persian→English (Fa→En) direction compared to the other two directions. This demonstrates that the both models are more robust in high-resource directions, while lower-resource directions are more sensitive to layer pruning.

7 Conclusion

In this paper, we introduced the CULL-MT approach for compressing multilingual NMT models for specific translation directions. Our method begins by identifying unimportant layers in a greedy manner and subsequently pruning them. To mitigate the damage caused by this pruning, we employed sequence-level knowledge distillation from the original model, utilizing the LoRA fine-tuning technique. Our findings revealed that the NLLB-3.3B model demonstrated considerable robustness against layer pruning, while the LLaMA3.1-8B-Instruct model was notably sensitive. Nonetheless, our approach effectively pruned both models with minimal performance.

8 Limitations

The primary limitation of our work is that we tested CULL-MT only on models with fewer than 10 billion parameters due to hardware constraints. While quantization could enable the application of CULL-MT to larger models, we opted not to use it to avoid introducing additional complexity. Consequently, our experiments were conducted on models that fit within our GPU's 24 GB memory.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively Multilingual Neural Machine Translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A Survey of Multilingual Neural Machine Translation. *ACM Comput. Surv.*, 53(5).

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2022. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Christian Federmann, Tom Kocmi, and Ying Xin. 2022. NTREX-128 – news test references for MT evaluation of 128 languages. In *Proceedings of the First Workshop on Scaling Up Multilingual Evaluation*, pages 21–24, Online. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Louie Giray. 2023. Prompt engineering with chatgpt: a guide for academic writers. *Annals of biomedical engineering*, 51(12):2629–2633.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Kevin Heffernan, Onur Çelebi, and Holger Schwenk. 2022. Bitext mining using distilled sentence representations for low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2101–2112, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Geoffrey Hinton. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Sneha Kudugunta, Isaac Caswell, Biao Zhang, Xavier Garcia, Derrick Xin, Aditya Kusupati, Romi Stella, Ankur Bapna, and Orhan Firat. 2024. MADLAD-400: a multilingual and document-level large audited dataset. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Zhizhong Li and Derek Hoiem. 2018. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Meta. 2024. Introducing llama 3.1: Our most capable models to date. `https://ai.meta.com/blog/meta-llama-3-1/`. Accessed: 2024-10-16.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. *arXiv preprint arXiv:2407.14679*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

David Peer, Sebastian Stabinger, Stefan Engl, and Antonio Rodríguez-Sánchez. 2022. Greedy-layer pruning: Speeding up transformer models for natural language processing. *Pattern Recognition Letters*, 157:76–82.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. {Zero-offload}: Democratizing {billion-scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China. Association for Computational Linguistics.

Shibo Wang and Pankaj Kanwar. 2019. Bfloat16: The secret to high performance on cloud TPUs. *Google Cloud Blog*, 4(1).

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024. When scaling meets LLM finetuning: The effect of data, model and finetuning method. In *The Twelfth International Conference on Learning Representations*.

Xuan Zhang, Navid Rajabi, Kevin Duh, and Philipp Koehn. 2023. Machine translation with large language models: Prompting, few-shot learning, and fine-tuning with QLoRA. In *Proceedings of the Eighth Conference on Machine Translation*, pages 468–481, Singapore. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen

Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Dataset size for Healing

To evaluate the impact of dataset size on the healing process for the NLLB-3.3B model, we conducted experiments focusing on healing the model specifically for the Persian-English translation direction. The pruned multiway model was fine-tuned with various dataset sizes generated through sequence-level knowledge distillation from the original model, and evaluated on the NTREX dataset. We fine-tuned all parameters of the model for each dataset size and present the results in Table 3.

Our findings show that using a very small dataset (10,000 sentences) was insufficient to effectively heal the model's performance drop. However, a dataset size of 40,000 sentences resulted in recovery of the model's performance. Moreover, By doubling the dataset size, healing performance showed a slight improvement. Therefore, to maximize the healed model's performance, we used 160,000 sentences.

Original	Pruned	No. of Sentences	Healed
		10,000	32.0
36.2	32.2	40,000	32.8
		80,000	33.0

Table 3: Evaluating the healing process of the multi-way pruned NLLB-3.3B model with various dataset sizes on the NTREX dataset using the spBLEU score.

B Comparing LoRA with Full-Parameter Fine-tuning

We conducted an experiment comparing LoRA fine-tuning with full parameter fine-tuning to assess their effectiveness in healing the pruned NLLB-3.3B model. For this experiment, we applied sequence-level knowledge distillation using an 80,000 sentence dataset for each translation direction in the multi-way scenario and evaluated our models using the NTREX dataset.

The results, presented in Table Table 4, indicate that healing the pruned model with the LoRA approach outperforms full parameter fine-tuning. Therefore, this technique is more suitable for the healing process. Additionally, healing the NLLB-3.3B model with LoRA eliminates the need for optimizer state offloading, resulting in faster training times compared to full parameter fine-tuning, which requires optimizer state offloading to function effectively.

Direction	Full-Parameter FT	LoRA
Fa→En	33.2	34.0
Fr→En	41.2	41.6
De→En	42.4	43.1
Average	38.9	39.5

Table 4: Comparison of spBLEU scores for healing the multi-way pruned NLLB-3.3B model using the LoRA approach and full parameter fine-tuning.