CONTEXT-AWARE DYNAMIC PRUNING FOR SPEECH FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Foundation models, such as large language models, have achieved remarkable success in natural language processing and are evolving into models capable of handling multiple modalities. Listening ability, in particular, is crucial for many applications, leading to research on building speech foundation models. However, the high computational cost of these large models presents a significant challenge for real-world applications. Although substantial efforts have been made to reduce computational costs, such as through pruning techniques, the majority of these approaches are applied primarily during the training phase for specific downstream tasks. In this study, we hypothesize that optimal pruned networks may vary based on contextual factors such as speaker characteristics, languages, and tasks. To address this, we propose a dynamic pruning technique that adapts to these contexts during inference without altering the underlying model. We demonstrated that we could successfully reduce inference time by approximately 30% while maintaining accuracy in multilingual/multi-task scenarios. We also found that the obtained pruned structure offers meaningful interpretations based on the context, e.g., task-related information emerging as the dominant factor for efficient pruning.

1 INTRODUCTION

In recent years, foundation models have achieved remarkable success across various tasks in natural language processing (OpenAI, 2022; 2023; Google, 2023; Anthropic, 2024). These Large Language Models (LLMs) have been particularly effective as multi-modal systems, incorporating modalities such as images and videos (Google, 2024; Anthropic, 2024). The integration of voice as a modality for communication between humans and LLMs has also gained traction, leading to applications that facilitate interactive conversations with LLMs (OpenAI, 2024; Défossez et al., 2024). Many studies have explored features to integrate the hearing ability into LLMs, employing methods such as connecting massive audio encoders to LLMs (Changli et al., 2024; Yuan et al., 2024; Chu et al., 2023; HU et al., 2024; Défossez et al., 2024) and utilizing large speech-to-text foundation models with powerful multilingual and multi-task capabilities (Radford et al., 2023; Peng et al., 2023d; Puvvada et al., 2024).

However, this broad support necessitates the training of large-scale models with billions of parameters, introducing new challenges such as increased inference costs. In speech processing, where input sequences tend to be longer than those in language processing, computationally intensive models can significantly prolong inference times. To address these challenges, various methodologies have been proposed, including pruning (Fu et al., 2022; Lai et al., 2021; Peng et al., 2023a; Wang et al., 2023; Ding et al., 2024), quantization (Ding et al., 2024), distillation (Chang et al., 2022; 2024; Gandhi et al., 2023), and combinations of those methods (Peng et al., 2023c). However, these approaches primarily focus on reducing the model size during training.

While these models handle various tasks, it raises a fundamental question: *is a single model optimal structure for all tasks and languages?* Different languages and tasks may require unique pruning strategies for effective processing. For example, Chen et al. (2022) highlights that different layers contribute differently depending on the task, indicating that the optimal model structure might vary across tasks. They also showed excellent performance in ASR using the WavLM encoder with only a single linear layer as the decoder, which raises questions about the need for a large-scale decoder in ASR systems. Conversely, Peng et al. (2024a) highlight the importance of the decoder network

in ST, suggesting that the decoder may play a more critical role in ST compared to ASR. Therefore, we hypothesize that there might be an optimal model structure depending on each task and language combination so that each subnetwork has the potential to perform with comparable accuracy with less inference complexity.

Given this hypothesis, we propose a method for *dynamically* pruning a pre-trained foundation model based on the context information, including speech features, language, and task characteristics, enabling the construction of an optimal model architecture tailored to the contextual requirements *during inference*. Specifically, we train a model that computes module-level masks for each layer in the encoder and decoder networks based on the provided context while simultaneously fine-tuning the foundation model. The predicted mask is utilized to determine which modules to activate or skip while maintaining accuracy. By analyzing the pruned network, we offer an interpretation of the importance of the optimal subnetwork within the given contexts.

This paper makes the following key contributions:

1. We propose to apply a novel context-aware pruning technique to each module in a speech foundation model dynamically within multilingual and multi-task scenarios.
2. We were able to reduce inference time by 34.3% without degradation in the BLEU score for the ST task and 28.6% with only 2.8% WER degradation on the ASR task.
3. We conducted a detailed comparative analysis and found that the obtained pruned structure offers meaningful interpretations based on the context, e.g., task-related information emerging as the dominant factor for efficient pruning.

2 Related Work

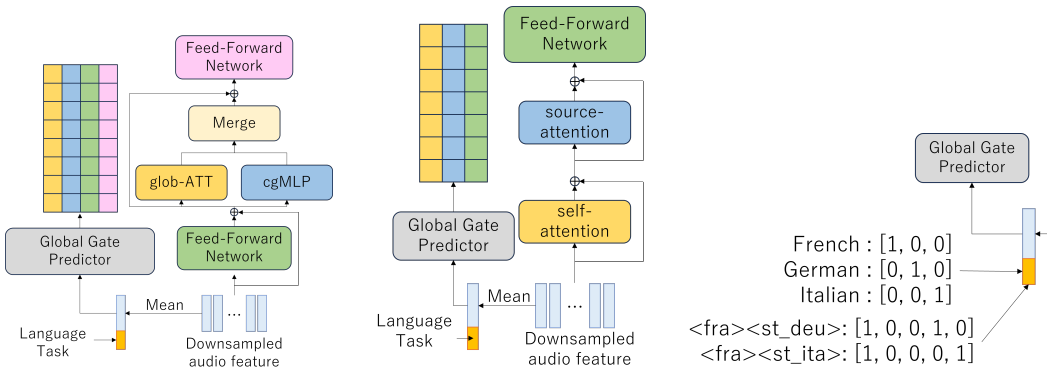
Pruning techniques are mainly classified into unstructured and structured approaches. The former is a technique for deleting individual weights in a network (LeCun et al., 1989; Hassibi et al., 1993; Han et al., 2016b); however, it has a problem of low compatibility with hardware accelerators (Han et al., 2016a; Liu et al., 2024). This method is further investigated in Appendix D. On the other hand, structured pruning has a more direct benefit in reducing the complexity, which performs pruning on a layer or module basis, including filters/layers in CNNs (Wen et al., 2016; Li et al., 2017; Alvarez & Salzmann, 2016; Han et al., 2017) or layer-wise pruning in models (Fan et al., 2020; Lee et al., 2021; Chen & Zhao, 2019). Thus, our paper employs structured pruning.

Methods for determining pruning targets include gradient-based techniques (Guo et al., 2016; He et al., 2020; Fu et al., 2022; Wen et al., 2016) as well as magnitude-based approaches for modules (Li et al., 2017; 2022). However, these methods typically use a fixed architecture during inference. Addressing this limitation, recent research has focused on implementing efficient inference by dynamically adjusting the computational load during the inference process (Bengio et al., 2016; Jernite et al., 2017; Bolukbasi et al., 2017; Graves, 2016). Notably, in the speech domain, numerous studies have explored streaming models to achieve dynamic model structures aimed at speedup (Macoskey et al., 2021a;b; Strimel et al., 2023; Xie et al., 2022; Xu et al., 2023). In this context, Peng et al. (2023b); Bittar et al. (2024) extended this concept to large-scale Transformer-based models, exploring dynamic layer-wise structural changes to enhance efficiency. In our study, we extend the work of Peng et al. (2023b) by utilizing the model structure of a speech foundation model to address multilingual and multi-task scenarios. This extension explores how a large-scale speech foundation model adapts its structure based on context, providing insights into more efficient and context-aware speech processing systems.

3 Methods

3.1 Open Whisper-Style Speech Models

In this study, we utilized Open Whisper-Style Speech Models (OWSM) (Peng et al., 2023d) as the foundation for our speech model experiments. OWSM is an open-source reproduction of OpenAI's Whisper model (Radford et al., 2023). Among the available versions, we selected OWSM-v3.1 (Peng et al., 2024b) as the speech foundation model for our experiments. The key rationale



(a) E-Branchformer (b) TransformerDecoder (c) Language/Task token

Figure 1: The sparse E-Branchformer and Transformer architectures in the experiment, and the method for embedding the language/task information. The audio information and the language/task information are concatenated, and the Gate Predictor calculates the gate probability for each module in each layer.

for selecting OWSM models lies in its fully open training data, processes, and configuration compared to Whisper. We ensure that the data used for validation in our experiments is not part of the pre-training corpus. This transparency is critical. If validation data was included in the pre-training data, it could inflate post-pruning accuracy, hindering a proper evaluation of the pruned network's performance.

Additionally, the E-Branchformer architecture integrated into OWSM-v3.1 offers a more flexible and generalized structure compared to the conformer, due to its parallel design. The model employs a dual-branch structure: one branch extracts global context using a self-attention-based module (glob-ATT), while the other captures local context using a convolution-based module (cgMLP) Sakuma et al. (2022). These branches are merged through a convolution-based merging layer, and are enclosed between two feed-forward networks (FFN1 and FFN2). Through the elimination of particular modules in the E-Branchformer layers, we can create a model resembling a conformer. This will enable us to thoroughly evaluate the Transformer's efficiency and effectiveness as an architectural design.

3.2 Module-level pruning

We implement module-level pruning in our study, targeting essential components within foundation models like self-attention (ATT) and feed-forward networks (FFN). For example, in the Transformer architecture, we prune the self-ATT, source-attention (src-ATT), and FFN as modules. In the case of the E-Branchformer, pruning modules include the FFN1, glob-ATT, cgMLP, and the FFN2.

The motivation for adopting module-wise pruning is our assumption that the model's architecture should remain flexible and adaptable during inference. Pruning techniques that operate at a finer granularity, such as kernel pruning or layer pruning, which removes individual kernels or layers from convolution components, would disrupt the model's structural integrity and limit its ability to dynamically adapt to different audio inputs. While layer-wise pruning aligns with our goal of simplifying the model, it oversimplifies the pruned model and prevents us from observing the importance of individual modules. For these reasons, we decided to employ module-wise pruning techniques to balance between structural flexibility and model interpretability. Further considerations on layer-skip approaches have been included in the Appendix C for reference.

3.3 Pruning

Given the dynamic nature of the input speech, it is necessary to generate a mask for each module based on audio input, language, and task information. To achieve this, we employ a neural network model to estimate a binary mask that determines whether to use a module. We frame the pruning

problem as an L0 regularization task Louizos et al. (2018), optimizing the expected value of the binary mask to achieve the desired sparsity. While Louizos et al. (2018) uses a Sigmoid-based approach, we follow Peng et al. (2023b) for implementation efficiency and treat the mask estimation as a two-class classification problem using Gumbel-Softmax Jang et al. (2017) for implementation efficiency.

In these works, pruning masks were learned using the sigmoid function or Gumbel-Softmax. However, the masks used during training were continuous values between 0 and 1, rather than strict binary values. As a result, modules that should have been completely skipped during inference were still partially utilized during training. During fine-tuning the OWSM model, we observed that even with a very low temperature for the softmax operation for probabilities, the gate probabilities often remained in the range between 0.4 and 0.6. This led to a discrepancy where the output of a module was scaled by a factor of 0.4 during training, while the same module was entirely skipped during inference because the probability fell below the threshold value, such as 0.5. To address this issue, we employed the Straight-through Gumbel-Softmax Estimator (SGST) Jang et al. (2017) to ensure that the output of the gate predictor was strictly binary. With SGST, the forward pass computations are performed using binary values, while the backward pass estimates gradients with continuous values, allowing the model to be trained effectively. The detailed formulation is provided in Appendix A.

In our work, inspired by Peng et al. (2023b) and Wang et al. (2020), we define the sparsity loss function, $\mathcal{L}_{\text{sparsity}}$, as follows:

$$\mathcal{L}_{\text{sparsity}} = \alpha\{|g - s_{\text{target}}| + (g - s_{\text{target}})^2\}, \tag{1}$$

where g is the average of the gate probabilities for all modules, α refers to the weight for $\mathcal{L}_{\text{sparsity}}$, and s_{target} is the desired sparsity ratio for the model. Since we use Gumbel-Softmax to binarize all gate probabilities, g represents the proportion of modules that are activated in the entire model, i.e., the model's sparsity ratio. For a detailed derivation of the loss function, please refer to the Appendix E.

Additionally, when both the Encoder and Decoder are pruned, we calculate the $\mathcal{L}_{\text{sparsity}}$ based on the gate probabilities from both components to bring the overall model sparsity closer to s_{target}. Let g_{enc} be the gate probability for any module in the encoder, and g_{dec} be the gate probability for any module in the decoder. Then, the L_{sparsity} is calculated for three scenarios, from top to bottom: first, when only the encoder is pruned; second, when only the decoder is pruned; and third, when both the encoder and decoder are pruned simultaneously.

$$\mathcal{L}_{\text{sparsity}} = \begin{cases} \alpha\,|\mathbb{E}[g_{\text{enc}}] - s_{\text{target}}| + \alpha\,(\mathbb{E}[g_{\text{enc}}] - s_{\text{target}})^2 & \text{encoder only} \\ \alpha\,|\mathbb{E}[g_{\text{dec}}] - s_{\text{target}}| + \alpha\,(\mathbb{E}[g_{\text{dec}}] - s_{\text{target}})^2 & \text{decoder only} \\ \frac{\alpha}{2}\,|\mathbb{E}[g_{\text{enc}}] + \mathbb{E}[g_{\text{dec}}] - 2s_{\text{target}}| + \frac{\alpha}{4}\,(\mathbb{E}[g_{\text{enc}}] + \mathbb{E}[g_{\text{dec}}] - 2s_{\text{target}})^2 & \text{jointly} \end{cases}$$

As noted by Wang et al. (2020), s_{target} is gradually increased during training. Therefore, let the loss for the downstream task be $\mathcal{L}_{\text{owsm}}$, the overall loss function that we aim to minimize is:

$$\mathcal{L} = \mathcal{L}_{\text{owsm}} + \mathcal{L}_{\text{sparsity}}. \tag{2}$$

3.4 Context-Aware Gate Predictor

As shown in Fig. 1, the gate probability is calculated using Gate Predictors. In Peng et al. (2023b), two types of Gate Predictors are proposed: GlobalGP and LocalGP. GlobalGP calculates the gate probability based on the encoder's input, which is also fed into the first layer of the encoder. In contrast, LocalGP provides a Gate Predictor for each layer, computing the probability based on the input to that specific layer. While both methods have shown promising results, we opted for GlobalGP due to its implementation simplicity. The detailed process of calculating gate probability is in Appendix F.

To handle multiple languages and tasks simultaneously, we created vectors representing the language and task, combined them with the speech features, and used them as input to the Gate Predictors. These vectors are combinations of one-hot vectors representing the language and task. For example, if there are two languages, French and German, and tasks including speech recognition and translation between them, the language conditions are $[0, 1]$ for French and $[1, 0]$ for German, and the task conditions are $[0, 0, 1]$ for speech recognition, $[0, 1, 0]$ for French to German translation, and $[1, 0, 0]$

for German to French translation. Combining these, tasks such as French speech recognition can be expressed as $[0, 1, 0, 0, 1]$, and similarly $[0, 1, 0, 1, 0]$ means translating French to German.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Dataset This study employs the Europarl-ST (Iranzo-Sánchez et al., 2020) dataset to evaluate model performance across multiple languages. The corpus was compiled from debates held in the European Parliament between 2008 and 2012. We utilized version 1.1 of the dataset, which comprises speech data in nine languages. For our experiments, we selected German, French, and Italian, which consist of approximately 20 hours of speech data. As the europarl-ST dataset is not part of the OWSM training data, we deemed it suitable for evaluating the model under multi-lingual and multi-task settings.

Task We fine-tuned the OWSM model with a pruning objective across ASR and ST tasks. The experiments were designed to compare two pruning strategies: one in which the model was trained on ASR and ST tasks independently, and another where both tasks were integrated during training. Additionally, we investigated the effects of sparsifying the model in three configurations: sparsifying only the Encoder, only the Decoder, and both simultaneously.

Evaluation In this experiment, we evaluated the ASR task using Word Error Rate (WER) and the ST task using BLEU scores. For each language, we prepared models with sparsity ratio of 10%, 30%, 50%, 70%, and 90%, and assessed their performance. Additionally, we used a baseline model that was fine-tuned with all modules retained for comparison. Note that the sparsity level refers to the ratio of activated modules to the total number of modules, not the number of parameters in the model. In all experiments, we performed auto-regressive decoding with a beam size of 5.

The model sparsity observed in this experiment is visualized using heat maps, where each cell represents the gate probability for all modules across all layers. During model validation, the activation frequency of each gate is accumlated, and the average is computed to derive the expected activation probability per module.

4.2 RESULTS

4.2.1 MULTI-LINGUAL ASR

Figure 2 shows the WER for German. In Figures 3 and 4, we present the visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ for each module when the encoder and decoder were pruned separately. Figure 5 illustrates the $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when encoder and decoder were jointly pruned. We analyzed the g across varied s_{target} and languages, and found no significant differences. Therefore, we focus on the German ASR results here. For complete heatmaps and a WER table, refer to Appendix G. We considered the possibility that the differences in the amount of data used for pre-training the OWSM model across languages might affect to our results and conducted additional training accordingly. The results are provided in Appendix B.

Inference Performance We found that pruning the decoder side did not harm WER, even with high sparsity ratios, where pruning encoder modules greatly deteriorate the WER in high sparsity ratio. By analyzing the results alongside the module heatmap, we observed a decline in encoder accuracy at $s_{\text{target}} = 0.7$, specifically when cgMLP started to be pruned, underscoring the critical role of cgMLP in ASR tasks. In contrast, observing the decoder side with a $s_{\text{target}} = 0.9$, where a substantial number of FFNs have been pruned, we find that WER does not deteriorate as severely as in the ST case discussed in section 4.2.2. This finding supports our initial question that ASR may not require large-scale decoders to the same extent as ST.

Sparse Encoder Analysis Referring to Figure. 3, the encoder's pruning strategy remained consistent across languages. Additionally, the highly polarized colors in Figure. 3, indicate that the gate probabilities are concentrated at extreme values, suggesting minimal variation in module selection

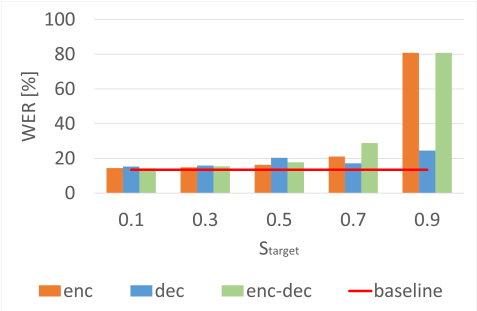


Figure 2: Comparison of sparsity ratio and WER. The label *enc* indicates pruning of the encoder only, *dec* indicates pruning of the decoder only, and *enc-dec* indicates simultaneous pruning of both. The WER is evaluated for German as the s_{target} varies from 0.1 to 0.9. The baseline WER is 13.5. Decoder pruning retains WER even at high sparsity, while encoder pruning significantly degrades it.

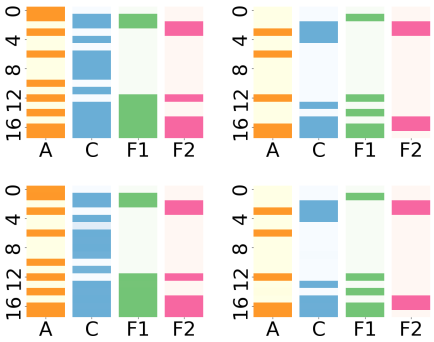


Figure 3: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when pruned separately with German and French ASR. The columns represents the s_{target}, with s_{target} being 0.5 and 0.7 from left to right. The first row represents the result for German ASR, and the second row represents the French ASR. The y-axis within the heatmap represents the depth, where top is the first layer. The label *A* indicates glob-ATT, *C* indicates cgMLP, *F1* indicates FFN1, and *F2* indicates FFN2.

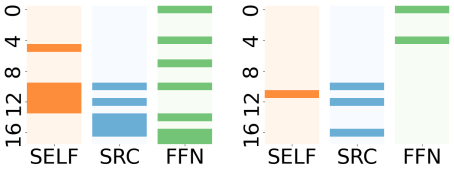


Figure 4: Visualization of $\mathbb{E}[g_{\text{dec}}]$ in German ASR, when decoder was pruned separately. The columns represent the s_{target}, with s_{target} being 0.7 and 0.9 from left to right. The label *SELF* indicates self-ATT, *SRC* indicates src-ATT, and *FFN* indicates FFN. The other settings are consistent with those in Figure 3.

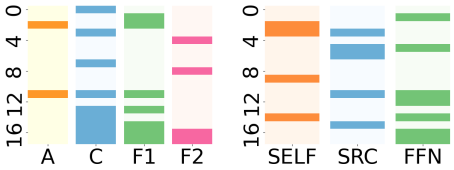


Figure 5: Visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when they were pruned jointly. The s_{targat} for this figure is 0.7. The left image corresponds to the encoder, and the right image corresponds to the decoder. The other settings are consistent with those in Figure 3 and Figure 4.

based on speech characteristics. Interestingly, pruning approximately 50% of the encoder modules led to a more biased module selection, favoring cgMLP activations. This highlights the crucial role of local context captured by cgMLP, challenging the current architectural convention that equally balances the both.

Sparse Decoder Analysis In conditions where 90% of the modules were pruned, src-ATT was prioritized over self-ATT and FFN in Figure 4, indicating its essential role in inference. Given that src-ATT is the module responsible for incorporating audio information, this behavior is understandable. At a 70% sparsity ratio, we observed a typical flow in the TransformerDecoder, where the process moves from self-ATT to src-ATT, and then to FFN. The heatmap indicates that groups of modules can be computed in chunks, with several self-ATT modules processed together, followed by a block of src-ATT modules, and then a group of FFN modules. These findings suggest that incorporating chunk-wise computation could improve the efficiency of conventional decoder architectures.

Combined Encoder-Decoder Sparsity Analysis We found that several portion of the encoder exhibit a similar architecture to that of the Conformer. That is, the processing sequence progresses

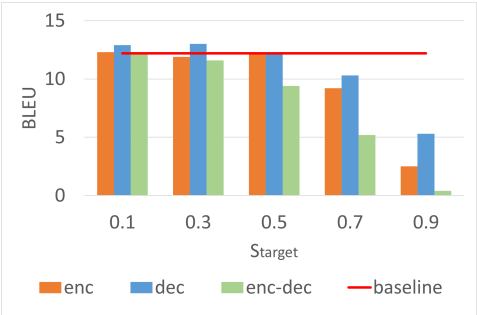


Figure 6: Comparison of sparsity ratio and BLEU scores for French-to-German translation. BLEU is evaluated for encoder-only, decoder-only, and encoder-decoder pruning, with a baseline score of 12.2. Similar to ASR, the decoder retains performance better than the encoder at high sparsity.

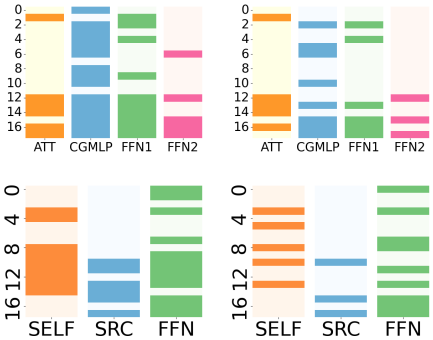


Figure 7: Visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when they were pruned jointly. The columns represents the s_{target}, with s_{target} being 0.5 and 0.7 from left to right. The first row represents the encoder, and the second row represents the decoder. The other settings are consistent with those in Figure 3.

from the FFN to the glob-ATT, then to the cgMLP, and back to the FFN. This finding suggests that the Conformer architecture is effective in speech foundation models, particularly when the model size is constrained.

Compared to Figure 4 and Figure 5, we observed that when pruning is applied only to the decoder, src-attention layers in the early part of the decoder are often skipped. However, when both the encoder and decoder are pruned together, the earlier src-attention layers in the decoder become more active. This difference appears to stem from whether the encoder's full capacity is available. When the encoder is fully utilized, the decoder computes more self-attention and FFN layers before src-attention to incorporate additional contextual information from the output tokens. In contrast, when encoder capacity is limited, the decoder compensates by performing self-attention and FFN computations directly on the audio features to capture details that may have been missed by the encoder. Additionally, by analyzing the number of active modules in the decoder, we found differences in the number of FFN layers processed before src-attention, which further supports this interpretation. These findings are also supported by visualizations provided in the Appendix G.

4.2.2 MULTI-LINGUAL ST

Figure 6 presents the BLEU scores for the French-to-German translation tasks. Figure 7 visualize the $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ for each module when the encoder and decoder were pruned separately. We also analyzed the g across varied s_{target} and tasks, and found no significant differences. Therefore, we focus on the French-to-German ST results here. Refer to the Appendix G for a complete heatmaps and table of BLEU score. The analysis on combined encoder-decoder settings are also in the Appendix G.

Inference Performance Focusing on the encoder side in Figure 6, the BLEU scores remained relatively stable even with a $s_{\text{target}} = 0.5$. On the other hand, from Figure 6 and Figure 7, BLEU scores began to degrade when usage of cgMLP modules dropped when s_{target} becomes 0.7. This highlights the importance of convolution-based models in ST, consistent with ASR tasks. A notable difference from ASR is that pruning the decoder also deteriorates model performance. We found that in the ST decoder, the FFN tends to be retained in computation. As the model removes the FFN, the BLEU score also degrades. This observation supports our initial hypothesis that the decoder plays a more critical role in ST compared to ASR.

Sparse Encoder Analysis In Figure 7, we observed an increased importance of cgMLP, similar to the findings in ASR. However, unlike ASR, the utilization of FFN2 decreased in the earlier layers

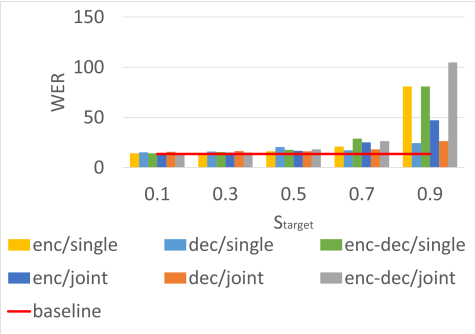


Figure 8: Comparison of s_{target} and WER on German ASR. This figure compares results when using only ASR data versus using both ASR and ST data. *single* refers to the results obtained using only ASR data, and *joint* includes results from pruning that also incorporates ST data. The other settings are consistent with those in Figure 3.

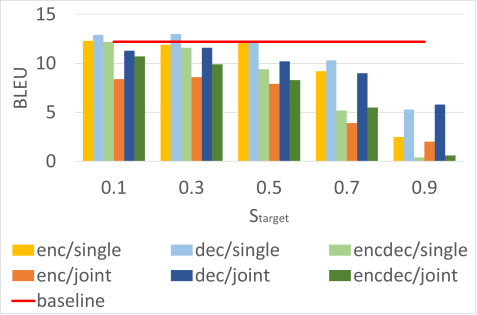


Figure 9: Comparison of s_{target} and BLEU on French-to-German ST. This figure compares results when using only ST data versus using both ASR and ST data. The labels are the same as Figure 8. Combined with Figure 8, this suggests that decoder pruning in multitask settings maintains competitive performance compared to task-specific pruning.

of the ST task. Nevertheless, there were no significant differences in module selection within the encoder structure. This indicates that it may not be necessary to modify the encoder when designing models intended to handle both ASR and ST tasks simultaneously.

Sparse Decoder Analysis From $s_{\text{target}} = 0.5$ to $s_{\text{target}} = 0.7$, the number of activated src-ATT decreased significantly in Figure 7, with this reduction being larger than ASR. Additionally, it became apparent that self-ATT were computed over a broader range of depths compared to ASR. This suggests that while ASR places greater importance on audio information, ST requires self-ATT and FFN more than audio information for translation. This increased reliance can be attributed to the non-monotonic relationship between input audio and output text in ST, necessitating a greater use of self-ATT and FFN to capture complex dependencies.

In Figure 7, we observed that self-ATT layers are more frequently activated before src-ATT in ST compared to ASR. We hypothesize that this difference arises from the distinct priorities of each task. In ASR, the primary focus is on integrating audio features directly, as each computation of src-ATT increases the prominence of audio information as a weighted sum. In contrast, ST seems to place a higher importance on alignment text information through self-ATT before src-ATT to effectively map different languages. As a result, self-ATT layers are activated earlier to better contextualize before src-ATT, reflecting the task-specific demands of aligning cross-modal information. These findings underscore how the allocation of self-ATT and src-ATT computations is influenced by the differing requirements of ASR and ST.

4.2.3 Pruning by Joint ASR and ST

Figure 8 represents the WER for German ASR, and Figure 9 represents the BLEU score for French-to-German translation task. Figure 9 represents the BLEU score for French-to-German translation task. Same as the single-task settings, we also analyzed the g across varied s_{target} and tasks, but could not found differences in module selection across languages and tasks. Therefore, we focus on the French-to-German ST results here. Refer to the Appendix G for a complete heatmaps and tables for other tasks and languages. The analysis on the case when encoder decoder were separately pruned are included in Paragraph 4.2.3.

Inference Performance When ASR and ST tasks were trained simultaneously, a slight degradation in WER and BLEU was observed in Figures 8 and 9, particularly when pruning was applied to both the encoder and decoder. However, we found that when only the decoder was pruned, performance was better maintained compared to other settings. These findings suggest that, for large-scale

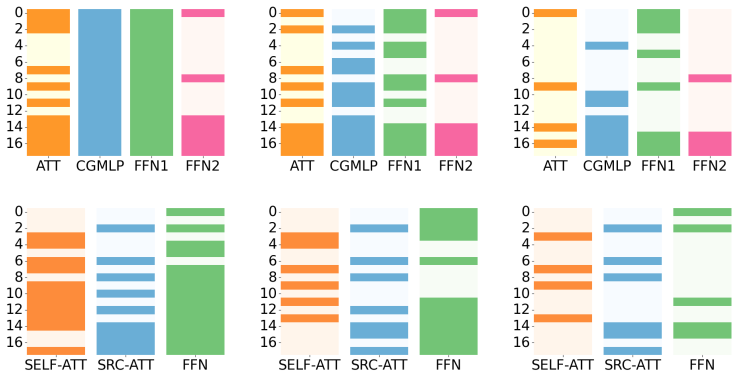


Figure 10: Visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when they were pruned jointly. The columns represent the s_{target}, with s_{target} being 0.3, 0.5, and 0.7 from left to right. The first row represents the encoder, and the second row represents the decoder. The other settings are consistent with those in Figure 3.

speech foundation models trained on multiple tasks, focusing on decoder pruning is a more effective strategy for preserving accuracy across various tasks.

Combined Encoder-Decoder Sparsity Analysis In Figure 10, the processing order observed was FFN, followed by src-ATT, and then self-att, particularly when s_{target} is 0.7. This order contradicts the typical processing sequence of a Transformer decoder and the observations made in 4.2.1. These results suggest that the conventional processing order of Transformer decoders may not be optimal for speech foundation models trained on multi-task data.

Table 1: Metrics and Elapsed time on inference for ASR (German) and ST (French-to-German) across varying s_{target}. The models were trained on task-specific training dataset. The table compares elapsed time, GFLOPs, and metrics (WER for ASR and BLEU for ST) at different s_{target} for the enc, dec, and enc-dec. Baseline results are: ASR - 9.28 seconds and 13.5 WER, and 3781 GFLOPs; ST - 10.54 seconds, 12.2 BLEU, and 3409 GFLOPs. Each row represents a different sparsity target, showing the impact on inference time and output quality as the sparsity increases. *ET* refers to the elapsed time. We use `fvcore` library to estimate the GFLOPs.

ASR (German)									
	Encoder			Decoder			EncDec		
sparsity	ET	GFLOPs	WER	ET	GFLOPs	WER	ET	GFLOPs	WER
10%	9.07	3697	14.4	9.33	3268	15.3	10.39	2843	14.3
30%	9.21	3690	14.8	7.24	2293	15.9	7.09	2249	15.4
50%	8.92	3669	16.3	6.62	1713	20.4	5.99	1698	17.8
70%	9.15	3633	21.0	4.79	1272	17.2	5.49	1139	28.8
90%	8.52	3613	80.8	4.80	625	24.5	5.22	682	80.8

ST (French-to-German)									
	Encoder			Decoder			EncDec		
sparsity	ET	GFLOPs	BLEU	ET	GFLOPs	BLEU	ET	GFLOPs	BLEU
10%	10.42	3369	12.3	10.72	2718	12.9	10.58	2861	12.2
30%	10.08	3357	11.9	8.27	2015	13.0	9.27	2093	11.6
50%	10.27	3335	12.3	6.92	1521	12.2	7.09	2023	9.4
70%	10.38	3311	9.2	5.38	1063	10.3	6.58	936	5.2
90%	10.44	3298	2.5	4.27	504	5.3	4.81	549	0.4

4.3 INFERENCE EFFICIENCY

We measured the actual inference time of the pruned model to analyze the effect of pruning on inference speed. We used one A40 GPU and 16 CPUs per each inference. We employed vectorized beam search (Seki et al., 2019) for decoding, where the beam size is 5. Since previous experiments showed no variation in module selection across languages or tasks, we focused on one language and task, specifically German ASR for measuring inference time in each case. Table 1 presents the trends in metrics, inference time, and FLOPs as a function of module sparsity. It is important to note that we ignored the first run of inference, as it contains initialization processes that make it slower.

The results show that reducing the decoder modules by 50% improves latency while maintaining accuracy for both ASR and ST tasks. Specifically, we achieved a 34.3% reduction in inference time with no degradation in BLEU for ST, and a 28.6% reduction with only a 2.8% WER increase for ASR. Since OWSM uses auto-regressive inference with vectorized beam search, the decoder handles the majority of the computational load. This is reflected in the significant reduction in FLOPs observed during pruning, as the decoder processes each output token individually, treating the beam size as the batch size. In this analysis, we set the beam size to 5, meaning the encoder's batch size is 1, while the decoder's is 5. As a result, pruning the decoder not only reduces FLOPs but also has a more pronounced impact on inference speed compared to pruning the encoder.

5 CONCLUSION

In this work, we proposed a novel context-aware dynamic pruning method for speech foundation models that adapts pruning dynamically during inference. With the pruned model, we successfully accelerated the inference of speech foundation models, particularly without any degradation in the ST task. Through a detailed analysis of the model structures that emerge after pruning, we identified the efficiency of the Transformer decoder and Conformer, while also uncovering an interesting computational flow when the model was pruned in multi-task settings. Although this study focused on the speech domain, our approach can be readily extended to foundation models in other fields, such as NLP and computer vision.

6 REPRODUCIBILITY

You can download the OWSM-v3.1 we employed in this experiment from the huggingface hub [1] All of our experiments are conducted with ESPnet (Watanabe et al., 2018). Based on the training configuration of OWSM-v3.1, we added or modified the following configuration:

```
encoder: e_branchformer_token_condition
decoder: transformer_decoder_token_condition

tau_ini: 1
tau_end: 0.1
tau_cooldown_steps: 15000
sparsity_init: 0.0
sparsity_end: 0.3

optim: adamw
optim_conf:
    lr: 0.00001
    weight_decay: 0.000001
scheduler: warmuplr
scheduler_conf:
    warmup_steps: 6000
```

Several configurations were added to the original ESPnet. Each configuration is as follows:

- tau_ini / tau_end: The initial and final temperatures for Gumbel-Softmax.

[1] https://huggingface.co/espnet/owsm_v3.1_ebf

- tau_cooldown_steps: The iteration number for the temperature of Gumbel-Softmax. Target sparsity gradually increases to sparsity_end.
- sparsity_init / sparsity_end : The initial target sparsity and after the warmup.
- sparsity_warmup_steps: Warmup steps for the sparsity. The target sparsity will be gradually increased to reach sparsity_end.

The class we set for the encoder and decoder is the extended class of E-Branchformer and TransformerDecoder to incorporate pruning in this study. Other configurations are same as the OWSM-v3.1, and you can refer to all settings in huggingface hub [2]

REFERENCES

Jose M. Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2262–2270, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/6e7d2da6d3953058db75714ac400b584-Abstract.html.

Anthropic. Claude 3 haiku: our fastest model yet, 2024. URL https://www.anthropic.com/news/claude-3-haiku.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint*, 1511.06297, 2016.

Alexandre Bittar, Paul Dixon, Mohammad Samragh, Kumari Nishu, and Devang Naik. Improving vision-inspired keyword spotting using dynamic module skipping in streaming conformer encoder. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 10386–10390. IEEE, 2024.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 527–536. PMLR, 2017. URL http://proceedings.mlr.press/v70/bolukbasi17a.html.

Heng-Jui Chang, Shu-Wen Yang, and Hung-yi Lee. Distilhubert: Speech representation learning by layer-wise distillation of hidden-unit bert. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*, pp. 7087–7091. IEEE, 2022. doi: 10.1109/ICASSP43922.2022.9747490. URL https://doi.org/10.1109/ICASSP43922.2022.9747490.

Heng-Jui Chang, Ning Dong, Ruslan Mavlyutov, Sravya Popuri, and Yu-An Chung. Colld: Contrastive layer-to-layer distillation for compressing multilingual pre-trained speech encoders. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 10801–10805, 2024.

Tang Changli, Yu Wenyi, Sun Guangzhi, Chen Xianzhao, Tan Tian, Li Wei, Lu Lu, MA Zejun, and Zhang Chao. SALMONN: Towards generic hearing abilities for large language models. In *The International Conference on Learning Representations (ICLR)*, 2024.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022. doi: 10.1109/JSTSP.2022.3188113.

[2] https://huggingface.co/espnet/owsm_v3.1_ebf/blob/main/exp/s2t_train_s2t_ebf_conv2d_size1024_e18_d18_piecewise_lr2e-4_warmup60k_flashattn_raw_bpe50000/config.yaml

Shi Chen and Qi Zhao. Shallowing deep networks: Layer-wise pruning based on feature representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(12):3048–3056, 2019. doi: 10.1109/TPAMI.2018.2874634.

Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and Jingren Zhou. Qwen-audio: Advancing universal audio understanding via unified large-scale audio-language models. *arXiv preprint arXiv:2311.07919*, 2023.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. Technical report, Kyutai, September 2024. URL `http://kyutai.org/Moshi.pdf`.

Shaojin Ding, David Qiu, David Rim, Yanzhang He, Oleg Rybakov, Bo Li, Rohit Prabhavalkar, Weiran Wang, Tara N. Sainath, Zhonglin Han, Jian Li, Amir Yazdanbakhsh, and Shivani Agrawal. Usm-lite: Quantization and sparsity aware fine-tuning for speech recognition with universal speech models. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 10756–10760, 2024.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *The International Conference on Learning Representations (ICLR)*, 2020.

Yonggan Fu, Yang Zhang, Kaizhi Qian, Zhifan Ye, Zhongzhi Yu, Cheng-I Jeff Lai, and Celine Lin. Losses can be blessings: Routing self-supervised speech representations towards efficient multilingual and multitask speech processing. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20902–20920. Curran Associates, Inc., 2022.

Sanchit Gandhi, Patrick von Platen, and Alexander M Rush. Distil-whisper: Robust knowledge distillation via large-scale pseudo labelling. *arXiv preprint arXiv:2311.00430*, 2023.

Gemini Team Google. et al. gemini: a family of highly capable multimodal models, 2023. URL `https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf`.

Gemini Team Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint*, 1603.08983, 2016.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. Eie: Efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 243–254, 2016a. doi: 10.1109/ISCA.2016.30.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *The International Conference on Learning Representations (ICLR)*, 2016b.

Song Han, Huizi Mao, and William J. Dally. Pruning convolutional neural networks for resource efficient inference. In *The International Conference on Learning Representations (ICLR)*, 2017.

Babak Hassibi, David Stork, and Gregory Wolff. Optimal brain surgeon: Extensions and performance comparisons. In J. Cowan, G. Tesauro, and J. Alspector (eds.), *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993.

Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2006–2015, 2020. doi: 10.1109/CVPR42600.2020.00208.

Shujie HU, Long Zhou, Shujie LIU, Sanyuan Chen, Lingwei Meng, Hongkun Hao, Jing Pan, Xunying Liu, Jinyu Li, Sunit Sivasankaran, Linquan Liu, and Furu Wei. Wavllm: Towards robust and adaptive speech large language model. *arXiv preprint arXiv:2404.00656*, 2024.

J. Iranzo-Sánchez, J. A. Silvestre-Cerdà, J. Jorge, N. Roselló, A. Giménez, A. Sanchis, J. Civera, and A. Juan. Europarl-st: A multilingual corpus for speech translation of parliamentary debates. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8229–8233, 2020.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *The International Conference on Learning Representations (ICLR)*, 2017.

Yacine Jernite, Edouard Grave, Armand Joulin, and Tomas Mikolov. Variable computation in recurrent neural networks. In *The International Conference on Learning Representations (ICLR)*, 2017.

Cheng-I Jeff Lai, Yang Zhang, Alexander H. Liu, Shiyu Chang, Yi-Lun Liao, Yung-Sung Chuang, Kaizhi Qian, Sameer Khurana, David D. Cox, and James R. Glass. PARP: prune, adjust and re-prune for self-supervised speech recognition. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 21256–21272, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/b17c0907e67d868b4e0feb43dbbe6f11-Abstract.html.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

Jaesong Lee, Jingu Kang, and Shinji Watanabe. Layer pruning on demand with intermediate ctc. In *Interspeech 2021*, pp. 3745–3749, 2021. doi: 10.21437/Interspeech.2021-1171.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *The International Conference on Learning Representations (ICLR)*, 2017.

Qi Li, Hengyi Li, and Lin Meng. Feature map analysis-based dynamic cnn pruning and the acceleration on fpgas. *Electronics*, 11(18), 2022. ISSN 2079-9292. doi: 10.3390/electronics11182887. URL https://www.mdpi.com/2079-9292/11/18/2887.

Hou-I Liu, Marco Galindo, Hongxia Xie, Lai-Kuan Wong, Hong-Han Shuai, Yung-Hui Li, and Wen-Huang Cheng. Lightweight deep learning for resource-constrained environments: A survey. *ACM Comput. Surv.*, 56(10), jun 2024. ISSN 0360-0300.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *The International Conference on Learning Representations (ICLR)*, 2018.

Jon Macoskey, Grant P. Strimel, and Ariya Rastrow. Bifocal neural asr: Exploiting keyword spotting for inference optimization. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5999–6003, 2021a. doi: 10.1109/ICASSP39728.2021.9414652.

Jonathan Macoskey, Grant P. Strimel, Jinru Su, and Ariya Rastrow. Amortized neural networks for low-latency speech recognition. In *Interspeech 2021*, pp. 4558–4562, 2021b. doi: 10.21437/Interspeech.2021-712.

OpenAI. Introducing chatgpt, 2022. URL https://openai.com/blog/chatgpt.

OpenAI. Gpt-4 technical report, 2023.

OpenAI. Gpt-4o system car, 2024. URL https://cdn.openai.com/gpt-4o-system-card.pdf.

Yifan Peng, Kwangyoun Kim, Felix Wu, Prashant Sridhar, and Shinji Watanabe. Structured pruning of self-supervised pre-trained models for speech recognition and understanding. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pp. 1–5. IEEE, 2023a. doi: 10.1109/ICASSP49357.2023.10095780. URL https://doi.org/10.1109/ICASSP49357.2023.10095780.

Yifan Peng, Jaesong Lee, and Shinji Watanabe. I3d: Transformer architectures with input-dependent dynamic depth for speech recognition. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023b.

Yifan Peng, Yui Sudo, Muhammad Shakeel, and Shinji Watanabe. Dphubert: Joint distillation and pruning of self-supervised speech models. In Naomi Harte, Julie Carson-Berndsen, and Gareth Jones (eds.), *24th Annual Conference of the International Speech Communication Association, Interspeech 2023, Dublin, Ireland, August 20-24, 2023*, pp. 62–66. ISCA, 2023c. doi: 10.21437/INTERSPEECH.2023-1213. URL https://doi.org/10.21437/Interspeech.2023-1213.

Yifan Peng, Jinchuan Tian, Brian Yan, Dan Berrebbi, Xuankai Chang, Xinjian Li, Jiatong Shi, Siddhant Arora, William Chen, Roshan Sharma, Wangyou Zhang, Yui Sudo, Muhammad Shakeel, Jee-Weon Jung, Soumi Maiti, and Shinji Watanabe. Reproducing whisper-style training using an open-source toolkit and publicly available data. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8, 2023d.

Yifan Peng, Yui Sudo, Muhammad Shakeel, and Shinji Watanabe. OWSM-CTC: An open encoder-only speech foundation model for speech recognition, translation, and language identification. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10192–10209, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.549. URL https://aclanthology.org/2024.acl-long.549.

Yifan Peng, Jinchuan Tian, William Chen, Siddhant Arora, Brian Yan, Yui Sudo, Muhammad Shakeel, Kwanghee Choi, Jiatong Shi, Xuankai Chang, Jee weon Jung, and Shinji Watanabe. Owsm v3.1: Better and faster open whisper-style speech models based on e-branchformer. In *Interspeech 2024*, pp. 352–356, 2024b. doi: 10.21437/Interspeech.2024-1194.

Krishna C. Puvvada, Piotr Żelasko, He Huang, Oleksii Hrinchuk, Nithin Rao Koluguri, Kunal Dhawan, Somshubra Majumdar, Elena Rastorgueva, Zhehuai Chen, Vitaly Lavrukhin, Jagadeesh Balam, and Boris Ginsburg. Less is more: Accurate speech recognition & translation without web-scale data. In *Interspeech 2024*, pp. 3964–3968, 2024. doi: 10.21437/Interspeech.2024-2294.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever (eds.). *Robust Speech Recognition via Large-Scale Weak Supervision*, volume 2023 of *Proceedings of Machine Learning Research*, 2023.

Jin Sakuma, Tatsuya Komatsu, and Robin Scheibler. MLP-based architecture with variable length input for automatic speech recognition, 2022. URL https://openreview.net/forum?id=RA-zVvZLYIy.

Hiroshi Seki, Takaaki Hori, Shinji Watanabe, Niko Moritz, and Jonathan Le Roux. Vectorized beam search for CTC-attention-based speech recognition. pp. 3825–3829, 2019. doi: 10.21437/Interspeech.2019-2860.

Grant P. Strimel, Yi Xie, Brian King, Martin Radfar, Ariya Rastrow, and Athanasios Mouchtaris (eds.). *Lookahead When It Matters: Adaptive Non-causal Transformers for Streaming Neural Transducers*, Proceedings of Machine Learning Research, 2023. PMLR.

Haoyu Wang, Siyuan Wang, Wei-Qiang Zhang, Hongbin Suo, and Yulong Wan. Task-agnostic structured pruning of speech representation models. In Naomi Harte, Julie Carson-Berndsen, and Gareth Jones (eds.), *24th Annual Conference of the International Speech Communication Association, Interspeech 2023, Dublin, Ireland, August 20-24, 2023*, pp. 231–235. ISCA, 2023. doi: 10.21437/INTERSPEECH.2023-1442. URL https://doi.org/10.21437/Interspeech.2023-1442.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6151–6162. Association for Computational Linguistics, November 2020.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. Espnet: End-to-end speech processing toolkit. In *Proc. Interspeech*, pp. 2207–2211, 2018. doi: 10.21437/Interspeech.2018-1456.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yi Xie, Jonathan J. Macoskey, Martin Radfar, Feng-Ju Chang, Brian King, Ariya Rastrow, Athanasios Mouchtaris, and Grant Strimel. Compute cost amortized transformer for streaming asr. In *Interspeech 2022*, pp. 3043–3047, 2022. doi: 10.21437/Interspeech.2022-10465.

Hainan Xu, Fei Jia, Somshubra Majumdar, He Huang, Shinji Watanabe, and Boris Ginsburg. Efficient sequence transduction by jointly predicting tokens and durations. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38462–38484. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/xu23g.html.

Gong Yuan, Luo Hongyin, H. Liu Alexander, Karlinsky Leonid, and R. Glass James. Listen, think, and understand. In *The International Conference on Learning Representations (ICLR)*, 2024.

A PROBLEM FORMULATION WITH STRAIGHT-THROUGH GUMBEL-SOFTMAX ESTIMATOR

The input audio information is denoted as x, the foundation model as θ, and the model as $f(\theta)$. Before pruning, the output token y can be simply represented as:

$$y = f(x|\theta)$$

Let the model parameters be θ, and s_{target} denote a sparsity ratio, where $0 \leq s_{\text{target}} \leq 1$. The parameter set after pruning with this sparsity is denoted as $\tilde{\theta}$, which can be expressed using a pruning mask z as follows:

$$\tilde{\theta} = \theta \odot z, \quad z \in \{0,1\}^{|\theta|},$$

where $|\theta|$ is the dimensionality of the parameters and $\odot$ denotes the element-wise product.

Here, we follow Louizos et al. (2018), which they replace the loss function regularized with L^0-regularization with a surrogate loss function that is continuously optimizable. The objective in Louizos et al. (2018) is expressed as $\mathcal{R}$, where $\mathcal{L}$ is the model's loss function and λ is the weight for the regularization term:

$$\mathcal{R}(\theta) = \mathbb{E}[\mathcal{L}(f(x|\theta), y)] + \lambda \|\theta\|_0$$

In Louizos et al. (2018), z is represented as a Bernoulli distribution parameterized by π_j, i.e., $q(z_j|\pi_j) = Bern(z_j|\pi_j)$. Using this representation of z, the above equation can be rewritten as:

$$\mathcal{R}(\tilde{\theta}, \pi) = \mathbb{E}_{q(z|\pi)}[\mathcal{L}(f(x|\theta \odot z), y)] + \lambda \sum_{j=1}^{|\theta|} \pi_j$$

where $|\theta|$ is the dimensionality of the parameters and $\odot$ denotes the element-wise product.

However, since z is binary (0 or 1), its gradient cannot be directly computed. To address this, Louizos et al. (2018) defines a continuous random variable $s \sim q(s|\phi)$, parameterized by ϕ, to approximate z. Applying a hard sigmoid function $h(\cdot)$ to s, z can be expressed as:

$$z = h(p(\epsilon, \phi))$$

Here, $Q_j = \int_0^\infty s_j$, $\epsilon \sim n(\epsilon)$ is noise sampled from a noise distribution n, and s is determined as a deterministic function of ϕ using a differentiable function $p(\epsilon, \phi)$, i.e., $s = p(\epsilon, \phi)$. Using this representation, the objective becomes:

$$\mathcal{R}(\tilde{\theta}, \phi) = \mathbb{E}_{n(\epsilon)}[\mathcal{L}(f(x|\theta \odot h(p(\epsilon, \phi))), y)] + \lambda \sum_{j=1}^{|\theta|} Q_j$$

The expectation term can then be efficiently estimated using Monte Carlo sampling.

In Louizos et al. (2018), the binary concrete distribution was adopted for s. In our implementation, we utilized PyTorch, which provides a built-in function for Gumbel-Softmax. Considering the simplicity of implementation and reproducibility, we applied Gumbel-softmax to the above formulation in this study. Specifically, s, initially formulated using a sigmoid function, was redefined as a 2-class classification task that determines whether to activate the model parameters. This means the result of $p(\cdot)$ becomes two values. During the forward computation, the hard sigmoid function $h(\cdot)$ was replaced by the straight-through Gumbel-softmax estimator $t(\cdot)$. This allowed z_j to be represented as a binary value, strictly 0 or 1. Consequently, the loss during the forward pass can be expressed as:

$$\mathcal{R}(\tilde{\theta}, \phi) = \mathbb{E}_{n(\epsilon)}[\mathcal{L}(f(x|\theta \odot t(p(\epsilon, \phi))), y)] + \lambda \sum_{j=1}^{|\theta|_M} t(p(\epsilon, \phi_j))$$

where $|\theta|_M$ represents the total number of modules in the model. The summation of probabilities Q becomes unnecessary due to the binary output of Gumbel-softmax, allowing for a simple summation instead. In the straight-through Gumbel-softmax approach, they approximate the gradient of the continuous variable in the backward pass.

A.1 GATE PREDICTOR

To make pruning more dynamic, researchers have replaced the function $p(\epsilon, \phi)$ with a trainable neural network $g(\cdot)$. In Peng et al. (2023b), they designed this gate predictor to incorporate acoustic context information C_{acoustic}, allowing the equation to be reformulated as:

$$\mathcal{R}(\tilde{\theta}, \phi) = \mathbb{E}[\mathcal{L}(f(x|\theta \odot t(g(C_{\text{acoustic}}))), y)] + \lambda \sum_{j=1}^{|\theta|_M} t(g(C_{\text{acoustic}}, j))$$

In this work, we incorporated language context, C_{language}, and task information, C_{task}, reformulating the above equation as:

$$\mathcal{R}(\tilde{\theta}, \phi) = \mathbb{E}[\mathcal{L}(f(x|\theta \odot t(g(C_{\text{acoustic}}, C_{\text{language}}, C_{\text{task}}))), y)] + \lambda \sum_{j=1}^{|\theta|_M} t(g(C_{\text{acoustic}}, C_{\text{language}}, C_{\text{task}}, j))$$

B DATA SIZE

The language-specific data volumes used for training OWSM-v3.1 are shown in Table 2. For the three languages used in this experiment, we selected data from the Voxforge dataset, which includes the same languages as Europarl-ST. We created a joint dataset with Europarl-ST, and the training and evaluation process is aligned with the section 4. To account for the potential impact on languages underrepresented in the pre-training data, We selected Hungarian and Bulgarian from Fleurs dataset to account for the potential impact on languages underrepresented in the pre-training data. All hyper parameters were kept identical to those used in section 4.

Table 3 shows the results of additional training conducted on these languages. It seems increasing the dataset size contributes to learning more effective pruning masks. However, as VoxForge is included in the OWSM training dataset, potential data overlap remains a concern, as discussed in Section 3.1. The figure 12 shows the loss curves during training with three different datasets: the Europarl-ST dataset (left), the joint dataset with Fleurs (middle), and the joint dataset with VoxForge (right). It is evident that the training loss decreases significantly in the latter two cases, supporting the data overlap concerns.

For low-resource languages, Table 4 presents the WER results when the encoder was pruned. While pruning the decoder resulted in accuracy declines similar to higher-resource languages like French and German, pruning the encoder caused more significant degradation at lower sparsity ratios.

Additionally, the sparsity patterns for Hungarian ASR, with the encoder and decoder pruned to 70% sparsity, showed no significant deviations from trends observed in experiments with German or French. These results highlight the effectiveness of module-level pruning for the decoder, even in low-resource language settings.

Table 2: Data size used in OWSM-v3.1 pre-training for each language.

Language	**amaount (h)**
French	2489
German	3704
Italian	707
Hungarian	97
Bulgarian	18

C LAYER-LEVEL PRUNING

To address the concern on comparing the performance on other pruning methods, we conducted experiments with layer-level pruning. The example pruning pattern on layer-level skipping is shown in figure 13. The table 5 shows the WER of German ASR, where only decoder was pruned. The

Table 3: Additional dataset with Voxforge and Fleurs. We show WER for German, Hungarian, and Bulgarian. Decoder was pruned from OWSM-v3.1 model.

target sparsity	**Europarl-ST**	**+ voxforge**	+ Fleurs (Hungarian)	+ Fleurs (Bulgarian)
0.0	13.5	10.1	33.3	23.9
0.1	15.3	11.9	35.1	29.5
0.3	15.9	12.3	34.1	24.2
0.5	20.4	15.7	40.1	29.1
0.7	17.2	14.1	36.3	26.5
0.9	24.5	28.8	42.1	36.2

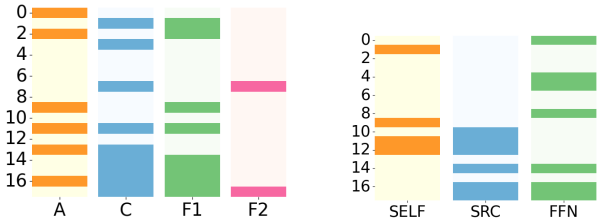
Table 4: Additional results for Hungarian and Bulgarian. We show WER for these two languages when encoders were pruned separately.

Target Sparsity	**Hungarian**	**Bulgarian**
0.1	38.7	38.2
0.3	42.5	37.7
0.5	42.1	38.3
0.7	54.4	44.2
0.9	89.2	78.8

results indicate that up to 70% sparsity, there is no significant difference between skipping at the layer level and skipping at the module level. However, at lower sparsity levels, the layer-level approach appears to perform better, whereas at higher sparsity levels, the module-level approach demonstrates superior performance. Based on these findings, for the decoder side, we believe that at higher sparsity levels, the roles and processing order of individual modules are more effectively optimized compared to the more coarse-grained approach of skipping entire layers, leading to the observed results.

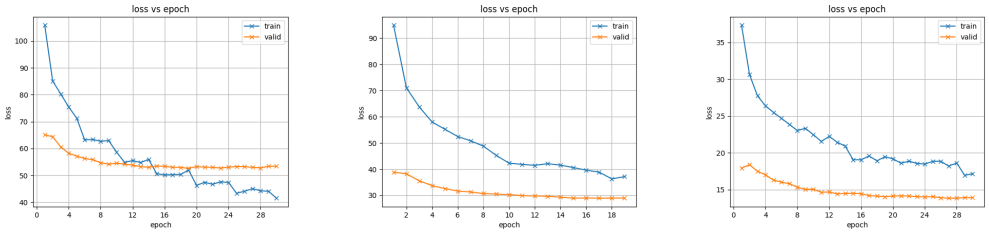
The table 5 shows the WER of German ASR, where encoder was pruned. The degradation in accuracy on the encoder side is significant, leading us to believe that layer-level pruning is not well-suited for speech foundation models. With layer-level pruning, even cgMLP, which is preserved in module-level pruning, is forcibly removed. This results in the loss of parameters that play a critical role on the encoder side, highlighting a key disadvantage of this approach.

We also calculated the FLOPs for these models and presented in table 6. Since different modules are utilized depending on whether module-level or layer-level pruning is applied, we examined the resulting differences. The table below shows the FLOPs when the encoder is pruned. As a result, we found that module-level pruning results in slightly lower FLOPs compared to layer-level pruning. These findings further emphasize the importance of module-level pruning from a performance perspective.



(a) Encoder (b) Decoder

Figure 11: The sparsity plot for Hungarian ASR shows the results when the encoder and decoder were pruned separately, with the target sparsity ratio set to 70%. The pruning patterns exhibit no notable differences compared to those observed in high-resource languages.



(a) Europarl-ST (b) Fleurs (c) Voxforge

Figure 12: Loss curve for pruning the decoder with sparsity ratio of 30%. We show curves from three different datasets; Europarl-ST (left), joint dataset with Flrues (middle), and the joint dataset with VoxForge (right). It is evident that the training loss decreases in the latter two cases, supporting the data overlap concerns.

Table 5: WER on module-level pruning and layer-level pruning. Results from German ASR, with only decoder pruned (left) and only encoder pruned (right).

Target Sparsity	Module Skip	Layer Skip
0.1	15.8	15.4
0.3	16.4	15.2
0.5	16.2	16.5
0.7	18.3	18.7
0.9	26.4	99.9

Decoder pruned.

Target Sparsity	Module Skip	Layer Skip
0.1	14.5	31.3
0.3	15.0	32.0
0.5	16.8	34.2
0.7	25.1	40.2
0.9	47.3	108.6

Encoder pruned.

Table 6: Comparison of GFLOPs between module skip and layer skip at different target sparsity levels. We used the `fvcore` library to calculate the FLOPs. FLOPs were computed over multiple utterances, and the average value was taken. For reference, the FLOPs for the model without pruning were 3781.

Target Sparsity	Module Skip (GFLOPs)	Layer Skip (GFLOPs)
0.1	3697	3697
0.3	3690	3692
0.5	3669	3666
0.7	3633	3640
0.9	3613	3620

D UNSTRUCTURED PRUNING

We further experimented the difference based on pruning strategy: unstractured pruning or structured pruning. Table 7 shows the comparison between magnitude-based unstructured pruning and PARP. Fine-tuning was performed on a training dataset that included both ASR and ST, and WER was compared on the German ASR task.

Even with unstructured pruning, we could see that pruning the decoder keeps the performance. However, compared to module-level results at the same sparsity ratio, module-level pruning, especially on the encoder side, maintained accuracy more effectively.

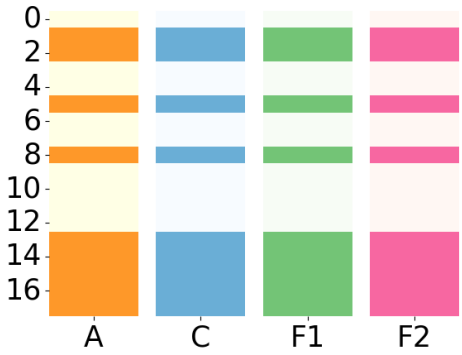


Figure 13: The sparsity plot when layers were skipped from encoder with the target sparsity ratio set to 70%. This model was trained on German-to-Italian ST.

Table 7: WER results for German ASR with different unstructured pruning techniques.

Model	**WER (%)**
Original model (without fine-tuning)	24.6
+ Unstructured pruning (sparsity = 0.1)	24.5
+ Unstructured pruning (sparsity = 0.3)	24.5
+ PARP (sparsity = 0.1, applied to encoder)	31.2
+ PARP (sparsity = 0.3, applied to encoder)	31.8
+ PARP (sparsity = 0.1, applied to decoder)	19.8
+ PARP (sparsity = 0.3, applied to decoder)	16.4

E SPARSITY LOSS

In Peng et al. (2023b), pruning is applied to the Transformer encoder by targeting the self-attention and feed-forward network modules. The sparsity loss function, $\mathcal{L}_{\text{sparsity}}$, is defined as:

$$\mathcal{L}_{\text{sparsity}} = \lambda \left(\frac{1}{2N} \sum_{l=1}^{N} (g_{\text{self-ATT}}^{(l)} + g_{\text{FFN}}^{(l)}) \right),$$

where $g_{\text{self-ATT}}^{(l)}, g_{\text{FFN}}^{(l)}$ are the gate probabilities of each module in the l-th layer, and the N is the number of layers. λ is a constant loss weight for the sparsity and the value is determined heuristically. Peng et al. (2023b) aims to achieve model sparsity by controlling the magnitude of the loss through λ. In Peng et al. (2023b), different values of λ will lead to different inference costs, as the final sparsity ratio of the model is controlled by sparsity loss. The problem here is that we cannot train the model to achieve a specific sparsity ratio.

Unlike Peng et al. (2023b), where a constant regularization factor λ is used, we followed Wang et al. (2020), that introduces a Lagrange multiplier λ_1 and λ_2 to calculate the sparsity loss. Our experiments showed that only using the fixed λ did not allow the model to achieve the desired sparsity, particularly when attempting to prune over 70% of the modules. To address this, Wang et al. (2020) introduces a penalty term, defined as:

$$\mathcal{L}_{penalty} = \lambda_1 (g - s_{\text{target}}) + \lambda_2 (g - s_{\text{target}})^2$$

where λ_1 and λ_2 are Lagrange multipliers updated based on the model's sparsity. In Wang et al. (2020), the initial values of λ_1 and λ_2 are set to 0, and they updated the parameter if $g = s_{\text{target}}$ is not true.

$\mathcal{L}_{penalty}$ can take a negative value when $-\frac{\lambda_2}{\lambda_1} \leq g - s_{\text{target}} \leq 0$ (Details are in E.1). If $\mathcal{L}_{\text{penalty}}$ becomes negative, it complicates solving the minimization problem when combined with the ASR and ST losses. To address this issue, we employed a function to calculate the absolute value of

$g - s_{\text{target}}$, so that the $\mathcal{L}_{penalty}$ remains non-negative. For the sake of simplicity, we set the $\lambda_1 = \lambda_2 = 1$ and employed a constant α on top of it. Starting from 1, we gradually increase the α unless it reaches the desired sparsity. The visualization of each cost function is in the Appendix. Thus, the sparsity loss we used in this experiment becomes Eq. 1. When calculating the $\mathcal{L}_{\text{sparsity}}$, s_{target} was gradually increased following Wang et al. (2020).

E.1 Visualization on sparsity loss

The purpose of Lagrange multipliers in section 3.3 is to make the $\mathcal{L}_{\text{penalty}}$ more aggressive penalty term. So we hypothesized that simply introducing a regularization term, which has similar role, we can make the model prune the desired number of modules. Let's visualize the graphs of different formulation we discussed. Figure 14 illustrates the graphs of various penalty terms. We define $\alpha = g_{l_0} - s_{\text{target}}$, where the x-axis represents α and the y-axis represents $\mathcal{L}_{\text{sparsity}}$.

The Lagrange multiplier approach, can be rewritten as a quadratic function: $\mathcal{L}_{\text{sparsity}} = (x + \frac{\lambda_1}{2\lambda_2})^2 - \frac{\lambda_2^2}{4\lambda_1^2}$ This is represented by the green curve in Figure 14. They move the vertex of this green curve to get the maximum.

In our implementation, we set $\lambda_1 = \lambda_2 = 1$ and introduced an overall coefficient α. Thus, $\mathcal{L}_{\text{sparsity}}$ becomes: $\mathcal{L}_{\text{sparsity}} = \alpha(x + \frac{1}{2})^2 - \frac{\alpha}{4}$ Increasing this coefficient α lowers the vertex of the blue quadratic function. For example, setting $\alpha = 5$ results in the red curve, which has a steeper slope for $x > 0$.

In our approach, we increase the value of this coefficient when the difference between the actual sparsity and the target sparsity exceeds a certain threshold (0.05). This results in a more aggressive penalty, helping to adjust the model towards the desired sparsity level. This quadratic penalty function allows for a more nuanced and effective approach to maintaining target sparsity compared to the constant penalty method.

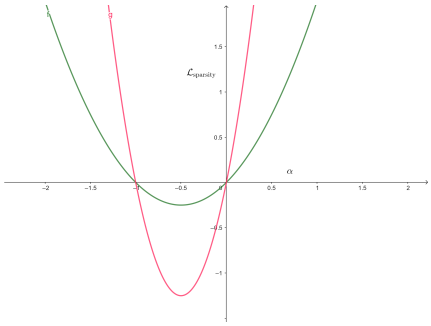


Figure 14: Visualization on different $\mathcal{L}_{\text{sparsity}}$ Wang et al. (2020) moves the green line to maximize the loss. Our method simply shifts the vertex downward. For example, by lowering the vertex of the green line, we transform it into a penalty function resembling the red line.

F Context-Aware Gate Probability

The gate probability g is calculated using a separate predictor for each module. Let the gate predictor be denoted as G(), the speech features input to the Gate Predictor be $y_{\text{GP}} \in \mathbb{R}^{T \times D}$, the condition representing the language and task be $c_{\text{cond}} \in \mathbb{R}^{D_{\text{cond}}}$, and the total number of layers be L. Here, T represents the number of frames in each speech features, D is the number of dimensions of the speech features, and D_{cond} is the number of dimensions of the condition representing the language and task. Then, the gate probability g of a certain module is calculated as follows:

Note that g_{l_0} can be calculated as $g_{l_0} = \mathbb{E}[g]$.

To prevent early learning instability, we avoid initializing the weights of the Gate Predictor randomly. If initialized randomly, approximately 50% of the modules may be removed in the initial

Algorithm 1 Gate Predictor

$x_{\text{pooled}} \leftarrow \text{Average}(y_{\text{GP}})$ ▷ Average over time dimension, $x_{\text{pooled}} \in \mathbb{R}^D$
$x \leftarrow \text{Concat}(c_{\text{cond}}, x_{\text{pooled}})$ ▷ Concatenate conditional info, $x \in \mathbb{R}^{D+D_{\text{conf}}}$
$\text{logit} \leftarrow \text{Reshape}(G(x))$ ▷ Reshape $G(x)$ to logit $\in \mathbb{R}^{L\times 2}$
$g \leftarrow \text{GumbelSoftmax}(\text{logit}, \text{axis} = 1)$ ▷ Compute Gumbel-Softmax for layers
$g \leftarrow g[:, 1]$ ▷ Select second column from Gumbel-Softmax output

learning stages, preventing the model's ability to gradually decrease the number of active modules. To address this, we adjust the bias of the final layer during the initialization. By configuring the g to output values close to 1 for all modules initially, we ensure that the model begins with full activation. This setup encourages a gradual reduction in the number of utilized modules as training progresses.

In our experiment, we followed Peng et al. (2023b) and used a two-layer MLP with an intermediate size of 32. We also examined whether increasing the intermediate layer size of the Gate Predictor to 512 would affect pruning. However, no changes in pruning trend were observed. This experiment confirmed a 50% sparsity ratio when fine-tuning with ASR and ST training data. Based on these results, we opted to use a Gate Predictor model of the same size as Peng et al. (2023b).

G HEATMAPS AND TABLES

Table 8: Comparison of WER (%) for French, German, and Italian using encoder-sparsified model, decoder-sparsified model, and jointly sparsified model.

		Sparse Encoder					Sparse Decoder					Jointly Sparsified Encoder-Decoder				
	Baseline	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
French	10.3	11.4	11.7	12.8	18.3	84.8	11.3	13.4	16.4	13.3	26.5	10.9	11.7	14.1	27.1	84.8
German	13.5	14.4	14.8	16.3	21.0	80.8	15.3	15.9	20.4	17.2	24.5	14.3	15.4	17.8	28.8	80.8
Italian	12.8	14.5	14.4	16.5	22.5	86.0	13.8	13.9	20.1	15.4	26.5	13.5	14.7	17.5	32.5	86.0

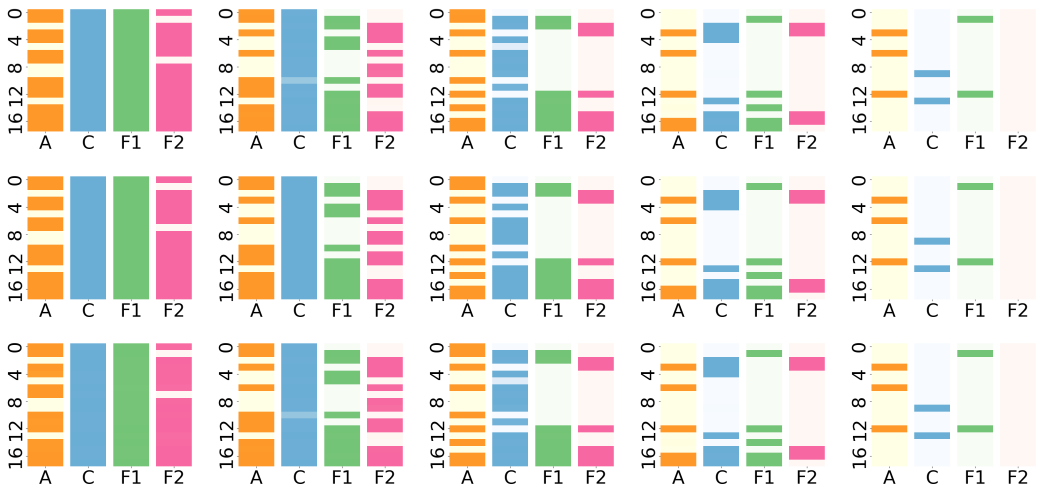


Figure 15: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French, the second row represents German, and the third row represents Italian. The other settings are consistent with those in Figure 3.

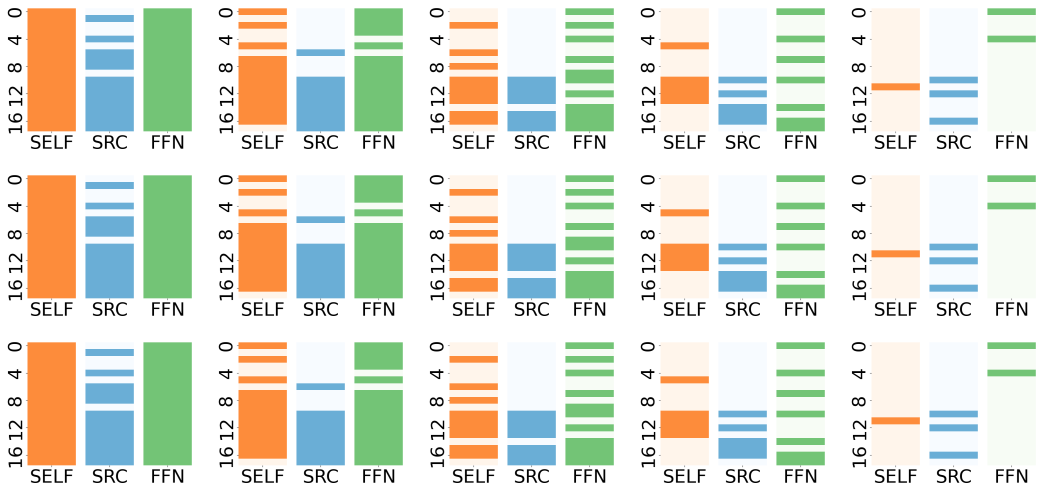


Figure 16: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French, the second row represents German, and the third row represents Italian. The other settings are consistent with those in Figure 3.

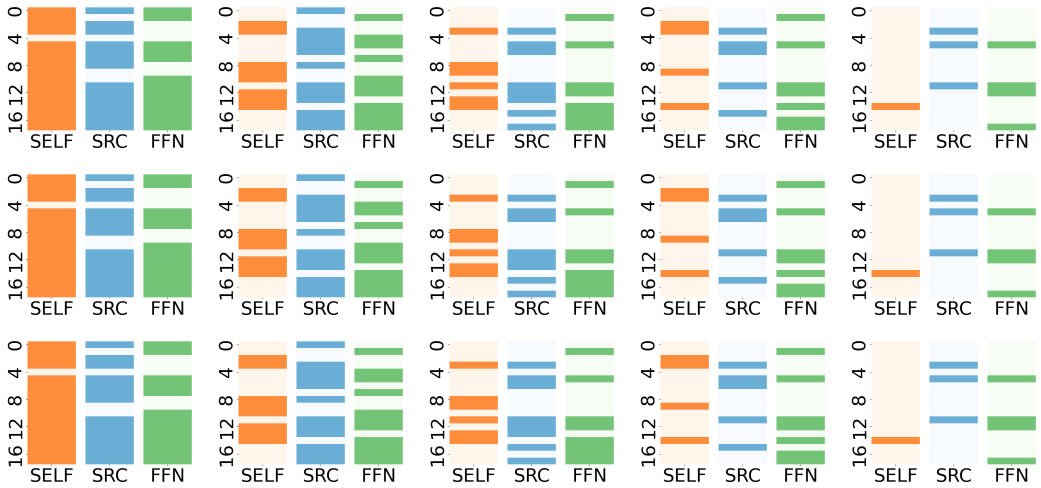


Figure 17: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder and decoder was pruned jointly. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French, the second row represents German, and the third row represents Italian. The other settings are consistent with those in Figure 3.

Table 9: BLEU score on each speech translation direction for Sparse Encoder, Sparse Decoder, and Jointly Sparsified Encoder-Decoder

			Sparse Encoder					Sparse Decoder					Jointly Sparsified Encoder-Decoder				
src	trg	baseline	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
fra	de	12.2	12.3	11.9	12.3	9.2	2.5	12.9	13.0	12.2	10.3	5.3	12.2	11.6	9.4	5.2	0.4
	it	13.5	12.8	12.7	12.3	10.2	1.8	12.5	11.8	12.0	9.3	3.1	12.8	12.0	8.8	4.2	0.0
deu	fr	9.4	9.1	8.7	8.5	6.6	2.2	9.3	9.2	9.0	6.4	3.1	8.9	8.5	6.3	3.2	3.1
	it	7.5	7.0	7.0	6.6	5.1	1.3	8.4	8.4	7.9	6.4	2.8	7.1	6.5	4.7	2.3	0.0
ita	de	11.8	11.1	10.8	10.1	8.2	2.8	12.1	12.0	11.8	9.4	5.5	11.2	10.3	8.5	4.6	0.7
	fr	14.0	12.8	12.6	11.8	9.9	3.1	13.0	12.3	11.8	8.3	3.3	12.8	12.0	8.8	4.2	0.9

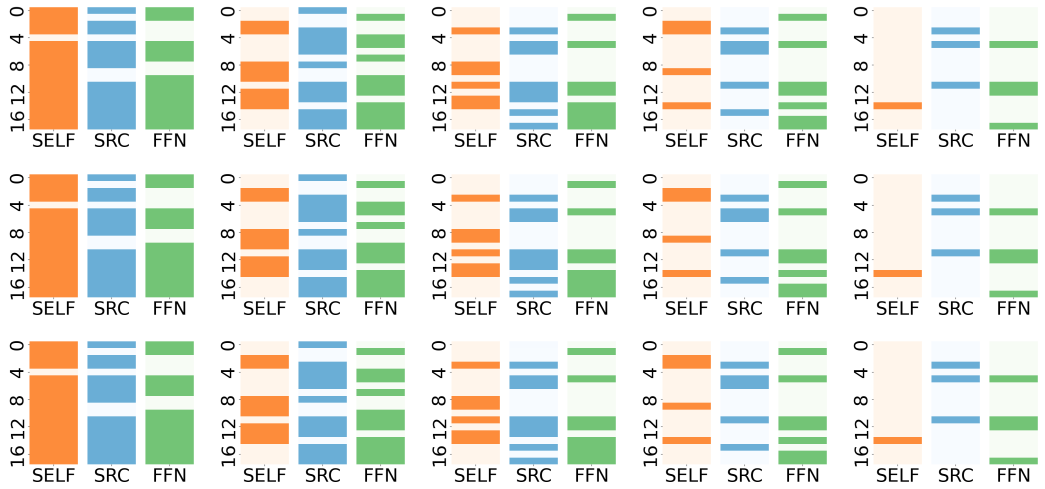


Figure 18: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when encoder and decoder was pruned jointly. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French, the second row represents German, and the third row represents Italian. The other settings are consistent with those in Figure 3.

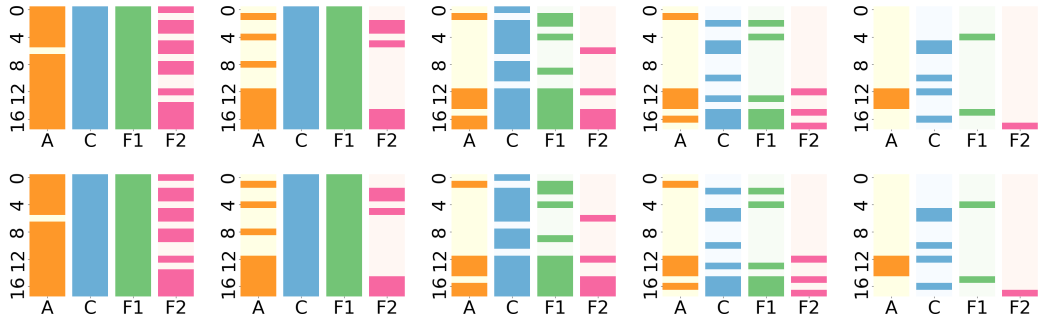


Figure 19: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French-to-German, the second row represents French-to-Italian translation. The other settings are consistent with those in Figure 3.

Figure 20: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French-to-German, the second row represents French-to-Italian translation. The other settings are consistent with those in Figure 3.

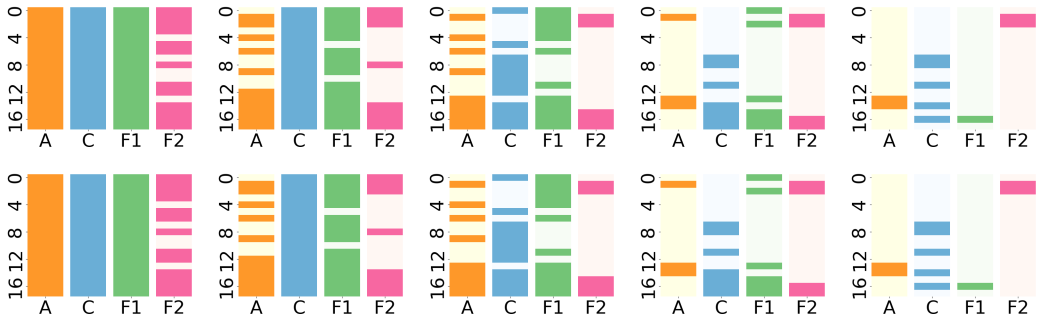


Figure 21: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was jointly pruned with decoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French-to-German, the second row represents French-to-Italian translation. The other settings are consistent with those in Figure 3.

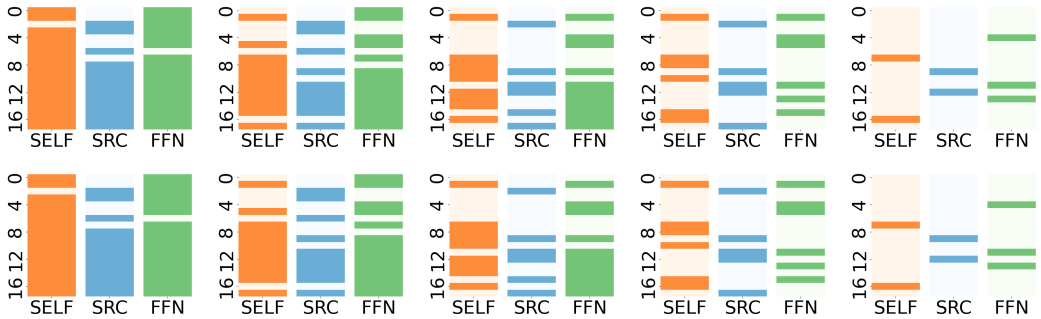


Figure 22: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was jointly pruned with encoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French-to-German, the second row represents French-to-Italian translation. The other settings are consistent with those in Figure 3.

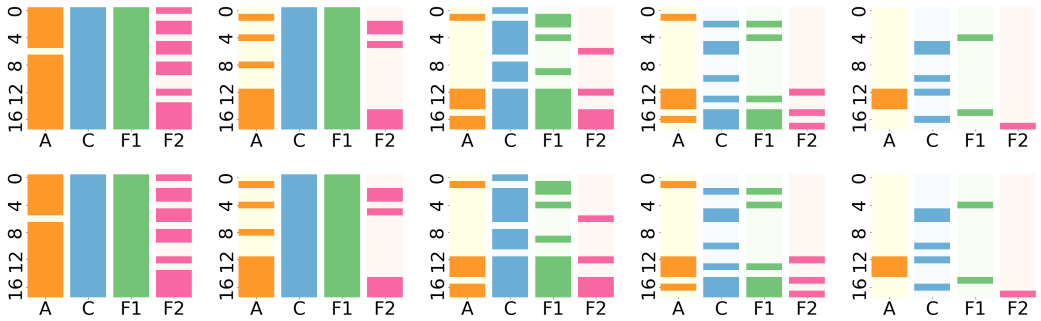


Figure 23: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the German-to-French, the second row represents German-to-Italian translation. The other settings are consistent with those in Figure 3.

Figure 24: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the German-to-French, the second row represents German-to-Italian translation. The other settings are consistent with those in Figure 3.

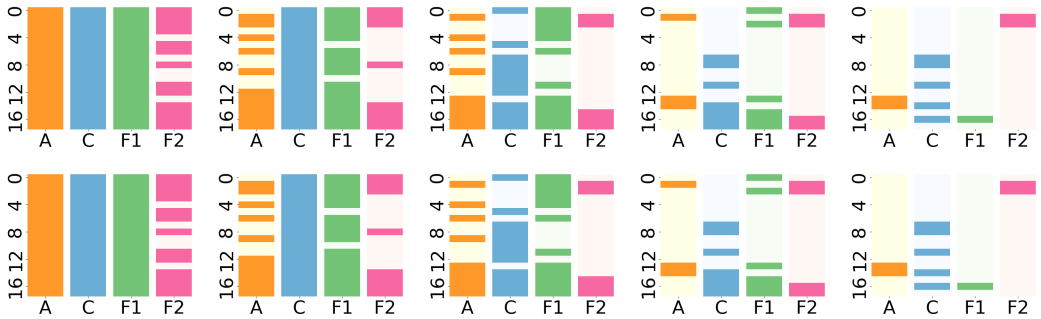


Figure 25: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was jointly pruned with decoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the German-to-French, the second row represents German-to-Italian translation. The other settings are consistent with those in Figure 3.

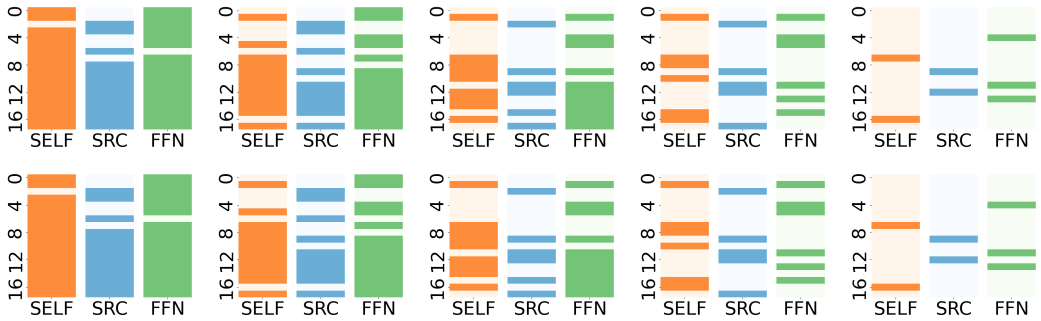


Figure 26: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was jointly pruned with encoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the German-to-French, the second row represents German-to-Italian translation. The other settings are consistent with those in Figure 3.

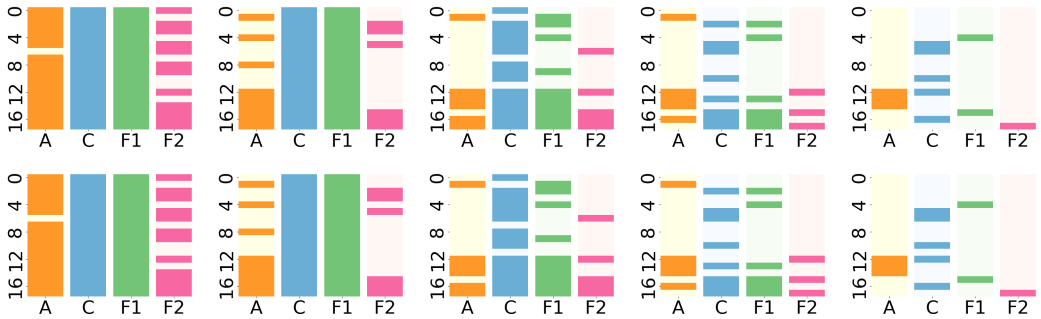


Figure 27: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the Italian-to-French, the second row represents Italian-to-German translation. The other settings are consistent with those in Figure 3.

Figure 28: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the Italian-to-French, the second row represents Italian-to-German translation. The other settings are consistent with those in Figure 3.

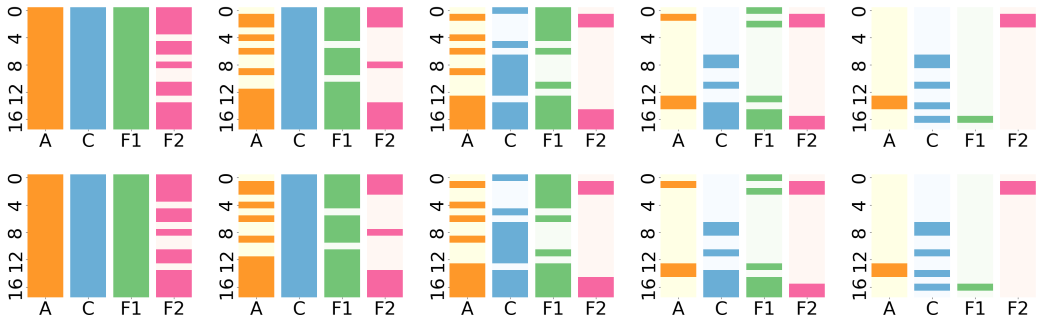


Figure 29: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was jointly pruned with decoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the Italian-to-French, the second row represents Italian-to-German translation. The other settings are consistent with those in Figure 3.

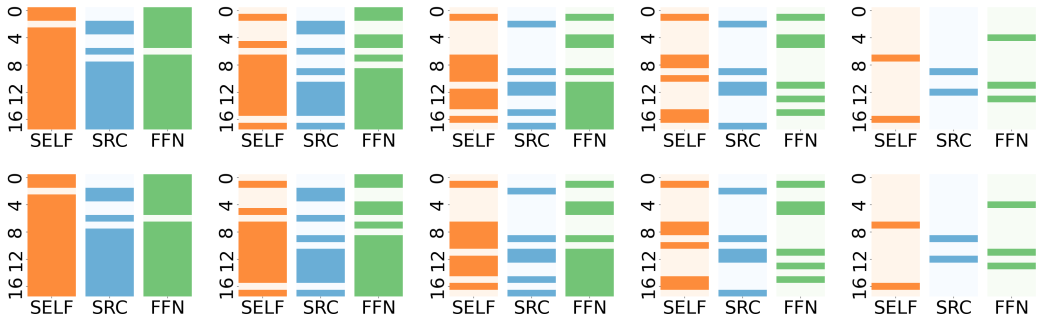


Figure 30: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was jointly pruned with encoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the Italian-to-French, the second row represents Italian-to-German translation. The other settings are consistent with those in Figure 3.

Table 10: BLEU score on each speech translation direction and WER for ASR tasks for Sparse Encoder, Sparse Decoder, and Jointly Sparsified Encoder-Decoder.

			Sparse Encoder					Sparse Decoder					Jointly Sparsified Encoder-Decoder				
src	trg	metric	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
	fr	WER	11.6	11.9	13.1	22.3	46.8	12.6	13.4	13.2	15.0	25.9	11.2	12.4	14.5	24.2	107.0
fr	de	BLEU	8.4	8.6	7.9	3.9	2.0	11.3	11.6	10.2	9.0	5.8	10.7	9.9	8.3	5.5	0.6
	it	BLEU	12.2	12.3	11.6	7.6	2.5	12.3	11.9	11.2	9.6	4.7	10.2	9.5	7.8	2.9	0.0
	de	WER	14.5	15.0	16.8	25.1	47.3	15.8	16.4	16.2	18.3	26.4	14.6	15.0	18.2	26.5	105.0
de	fr	BLEU	6.1	5.8	5.0	3.1	1.5	8.2	8.3	7.7	6.3	4.0	8.0	7.3	6.0	3.8	0.4
	it	BLEU	6.1	6.0	5.7	3.4	1.2	6.9	7.0	6.2	4.9	2.6	4.9	4.7	3.7	1.7	0.0
	it	WER	14.4	15.2	17.3	27.6	86.0	14.5	15.2	15.0	16.7	28.9	13.9	15.1	17.8	28.7	162.6
it	de	BLEU	6.5	6.5	6.2	3.9	2.2	10.6	10.4	9.8	8.4	5.8	7.1	8.7	7.4	3.2	0.1
	fr	BLEU	13.0	12.6	11.9	8.1	2.9	12.6	12.6	11.9	9.5	5.6	12.6	11.5	9.9	5.6	0.6

Figure 31: Visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when encoder and decoder was pruned separately for French ASR. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the encoder, the second row represents the decoder. The other settings are consistent with those in Figure 3.

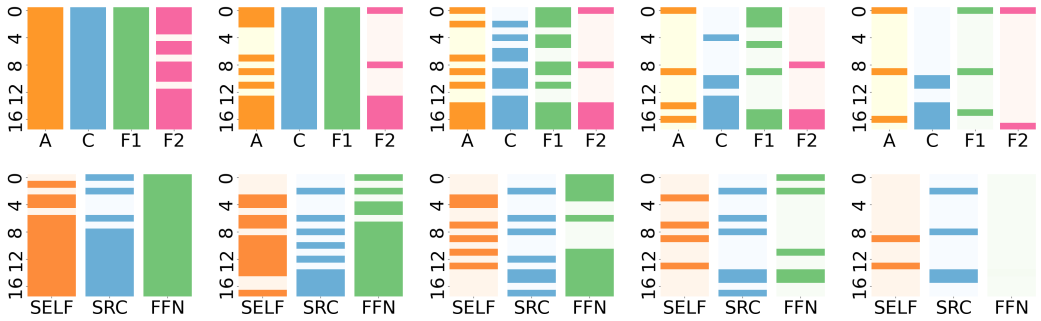


Figure 32: Visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when encoder and decoder was pruned jointly for French ASR. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the encoder, the second row represents the decoder. The other settings are consistent with those in Figure 3.

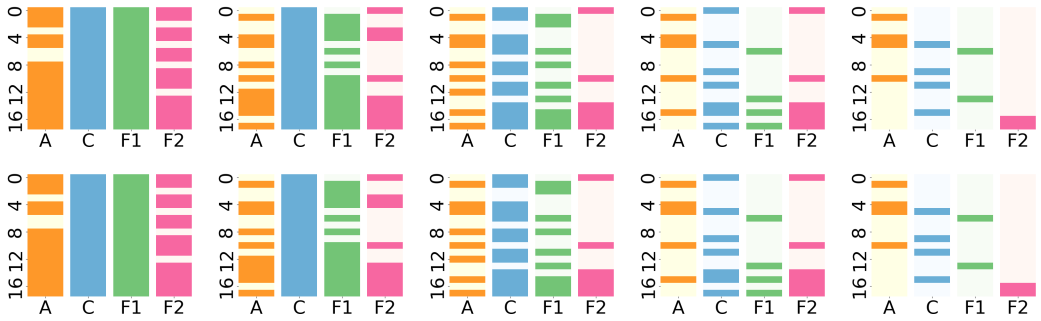


Figure 33: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French-to-German, the second row represents French-to-Italian translation. The other settings are consistent with those in Figure 3.

Figure 34: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French-to-German, the second row represents French-to-Italian translation. The other settings are consistent with those in Figure 3.

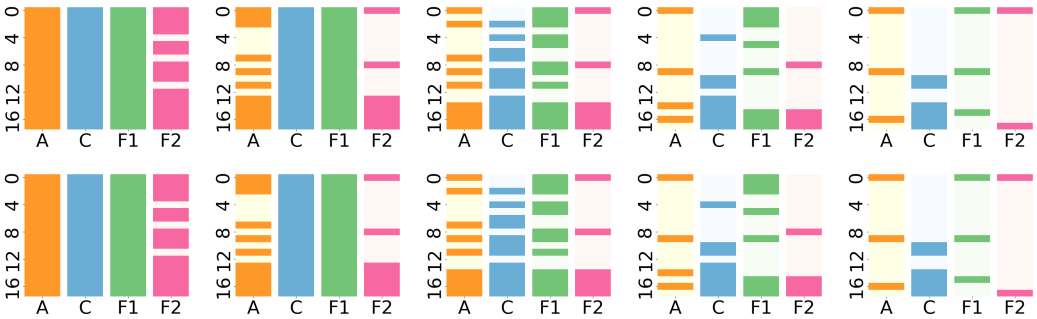


Figure 35: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was jointly pruned with decoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French-to-German, the second row represents French-to-Italian translation. The other settings are consistent with those in Figure 3.

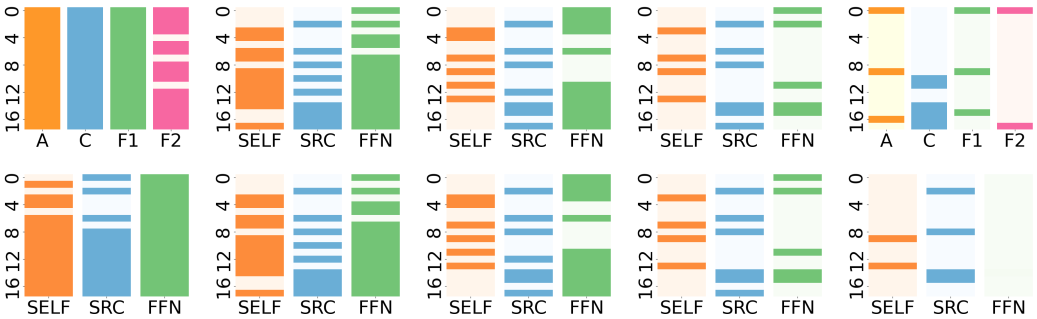


Figure 36: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was jointly pruned with encoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the French-to-German, the second row represents French-to-Italian translation. The other settings are consistent with those in Figure 3.

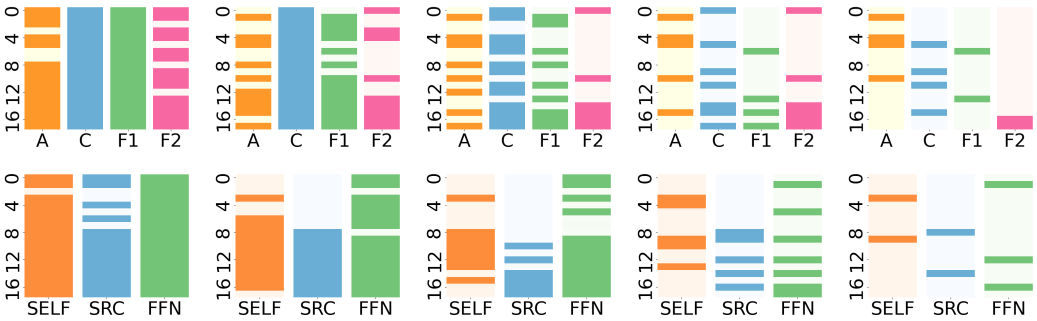


Figure 37: Visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when encoder and decoder was pruned separately for German ASR. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the encoder, the second row represents the decoder. The other settings are consistent with those in Figure 3.

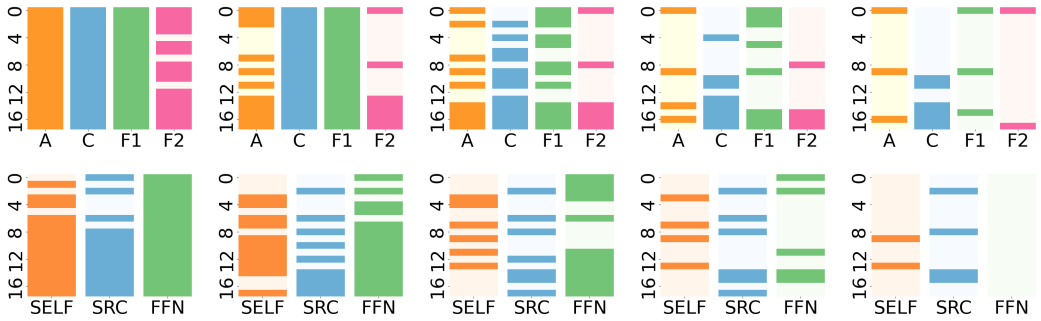


Figure 38: Visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when encoder and decoder was pruned jointly for German ASR. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the encoder, the second row represents the decoder. The other settings are consistent with those in Figure 3.

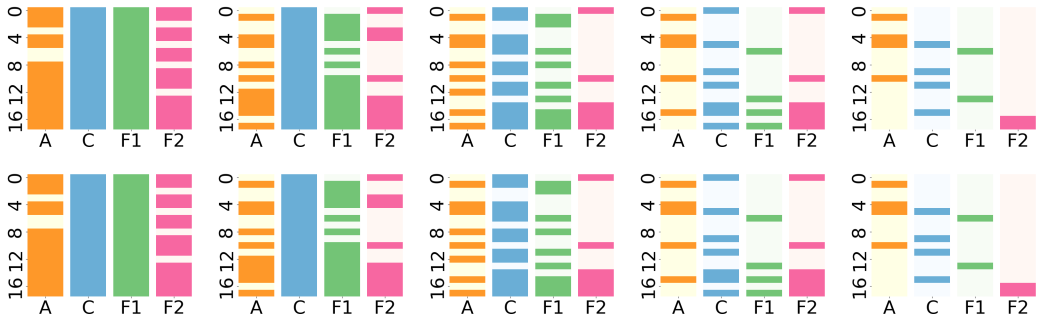


Figure 39: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the German-to-French, the second row represents German-to-Italian translation. The other settings are consistent with those in Figure 3.

Figure 40: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the German-to-French, the second row represents German-to-Italian translation. The other settings are consistent with those in Figure 3.

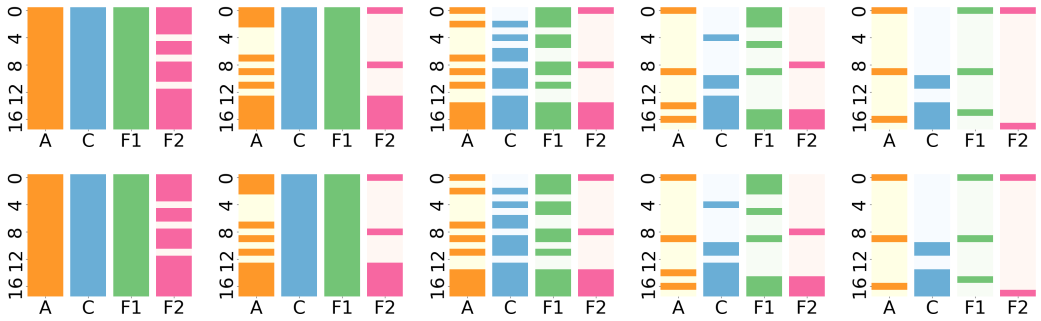


Figure 41: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was jointly pruned with decoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the German-to-French, the second row represents German-to-Italian translation. The other settings are consistent with those in Figure 3.

Figure 42: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was jointly pruned with encoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the German-to-French, the second row represents German-to-Italian translation. The other settings are consistent with those in Figure 3.

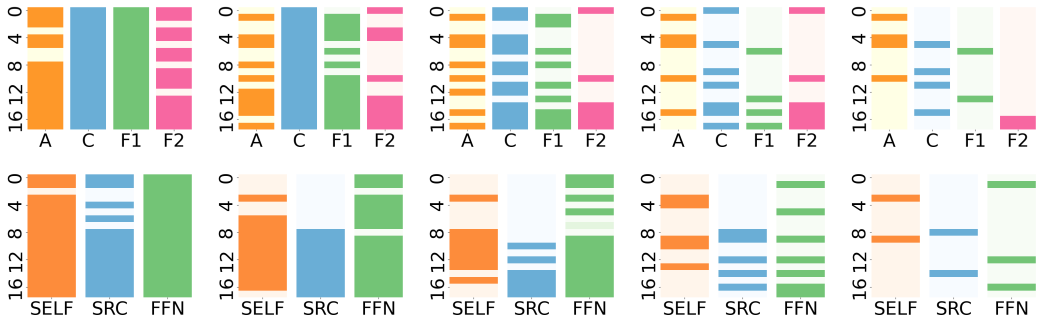


Figure 43: Visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when encoder and decoder was pruned separately for Italian ASR. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the encoder, the second row represents the decoder. The other settings are consistent with those in Figure 3.

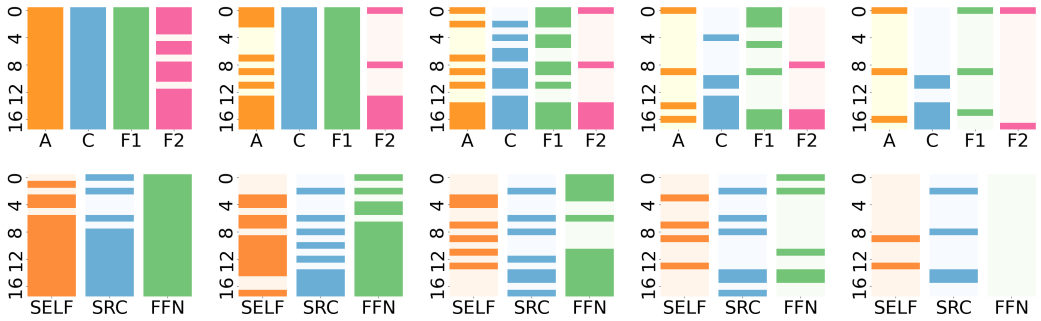


Figure 44: Visualization of $\mathbb{E}[g_{\text{enc}}]$ and $\mathbb{E}[g_{\text{dec}}]$ when encoder and decoder was pruned jointly for Italian ASR. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the encoder, the second row represents the decoder. The other settings are consistent with those in Figure 3.

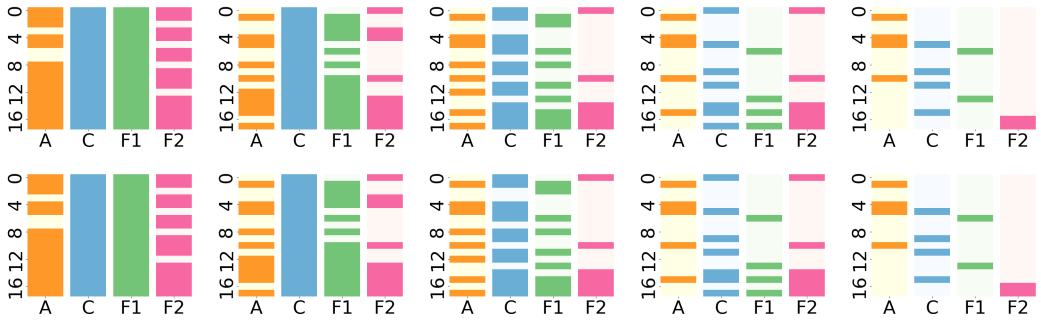


Figure 45: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the Italian-to-French, the second row represents Italian-to-German translation. The other settings are consistent with those in Figure 3.

Figure 46: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was pruned separately. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the Italian-to-French, the second row represents Italian-to-German translation. The other settings are consistent with those in Figure 3.

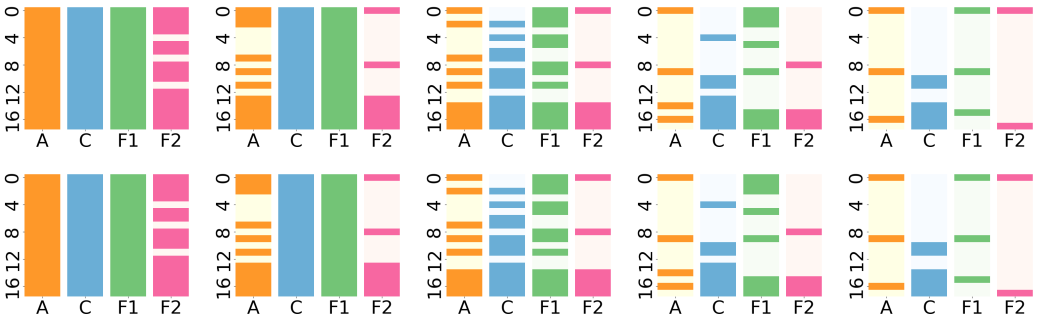


Figure 47: Visualization of $\mathbb{E}[g_{\text{enc}}]$ when encoder was jointly pruned with decoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the Italian-to-French, the second row represents Italian-to-German translation. The other settings are consistent with those in Figure 3.

Figure 48: Visualization of $\mathbb{E}[g_{\text{dec}}]$ when decoder was jointly pruned with encoder. The columns represents the s_{target}, with s_{target} being 0.1, 0.3, 0.5, 0.7, and 0.9 from left to right. The first row represents the Italian-to-French, the second row represents Italian-to-German translation. The other settings are consistent with those in Figure 3.