RedApt: An Adaptor for WAV2VEC 2 Encoding Faster and Smaller Speech Translation without Quality Compromise

Jinming Zhao **Hao Yang** **Gholamreza Haffari** **Ehsan Shareghi**

Department of Data Science & AI, Monash University

firstname.lastname@monash.edu

Abstract

Pre-trained speech Transformers in speech translation (ST) have facilitated state-of-the-art (SotA) results; yet, using such encoders is computationally expensive. To improve this, we present a novel Reducer Adaptor block, RedApt, that could be seamlessly integrated within any Transformer-based speech encoding architecture. Integrating the pretrained WAV2VEC 2 speech encoder with RedApt brings 41% speedup, 33% memory reduction with 24% fewer FLOPs at inference. To our positive surprise, our ST model with RedApt outperforms the SotA architecture by an average of 0.68 BLEU score on 8 language pairs from Must-C.

1 Introduction

Leveraging pre-trained speech Transformers, such as WAV2VEC 2 (W2V2) (Baevski et al., 2020), in speech-to-text translation (ST) systems have established state-of-the-art (SotA) results across several languages (Li et al., 2021; Ye et al., 2021). Meanwhile, the high computational cost of such encoders is well-documented (Wu et al., 2022) and mainly attributed to the self-attention mechanism inside Transformers (Vaswani et al., 2017).

However, speech modality introduces unique challenges compared to text: representation of raw speech signals are orders of magnitude longer[1], while empirical findings suggest that high-quality ST systems require much deeper encoders compared to text-to-text translation (Wang et al., 2020). Consequently, training and inference with such encoders is expensive, in terms of memory and time, even for reasonably powerful hardware.[2]

As a mitigation, Li et al. (2021) augmented W2V2 at the output by CNN layers to reduce the representation length, and Zhao et al. (2022) proposed a Transformer-based adaptor to shrink a sequence. Yet, the complexity of encoding remains high. Wu et al. (2022) proposed lower feature dimensions in W2V2 which improved efficiency at the expense of performance drop. Earlier studies focused on designing feature selection modules at the input level by using phone labels for merging adjacent vectors (Salesky and Black, 2020), using dynamic sparsification mechanism (Zhang et al., 2020), or injecting Connectionist Temporal Classification (Gaido et al., 2021) to regulate feature passing between layers. These approaches improve speed at inference but are limited as they rely either on hand-crafted features, transcripts, or external modules.[3]

In this work we focus on W2V2, as one of the most widely used pre-trained speech encoder for ST, and propose a novel block, Reducer Adaptor (RedApt), to reduce the computational load of processing speech sequences through W2V2 while improving translation quality. Our approach does not require any additional pretraining or information beyond the audio input, and works similar to adaptor blocks (Houlsby et al., 2019) placed on top of any layers of a pretrained transformer but trained along with the underlying Transformer on ST task.

Through extensive experiments on 8 language pairs from Must-C, we show that integrating RedApt into WAV2VEC yields 41% speedup, 33% memory savings, and 24% fewer FLOPs at inference time. Meanwhile, our ST model outperforms existing SotA by 0.68 BLEU score. To the best of our knowledge, we are the first to target the efficiency of pretrained speech encoders for ST. We hope this to facilitate further improvement across a broader range of speech processing tasks that require pretrained speech encoders.

[1] Speech features, e.g. Mel-Spectrogram, are lower dimensional but result in worse ST quality (Ye et al., 2021).

[2] Training batch size for a modern ST system (Gállego et al., 2021) could not exceed 1 on a V100 16GB GPU.

[3] Pruning (Lai et al., 2021), quantization and knowledge distillation (Peng et al., 2021; Chang et al., 2022) approaches are promising directions but they have yet to be applied to ST.

2 Reducer Adaptor (RedApt)

Our proposed RedApt has some key properties: (i) can be integrated seamlessly with pretrained speech Transformer such as WAV2VEC 2 (W2V2), (ii) flexibly reduces the computational load of encoding (both in terms of memory and time), and (iii) allows to retain the downstream ST performance. While beneficial for training, RedApt is in particular useful in the repetitive nature of inference phase. In this section, we will present RedApt architecture, and show how it can be integrated into WAV2VEC and a full speech-to-text translation (ST) system.[4]

2.1 Architecture of RedApt

The core idea is to pool a temporal speech sequence, to reduce it length while learning local information from the shrunk sequence. Suppose W2V2 feature encoder yields a sequence, $\mathbf{a}$. As shown in Figure 1 (W2V2 is omitted for brevity), RedApt is built on two CNN blocks which are wrapped by layer normalization, residual connection, and GELU nonlinear activation.

The first CNN block is a pooling module to shrink the length of $\mathbf{a}$. It is parameterized with kernel k, stride length s and padding p. The input sequence length thus can be reduced,

$$n' = \left\lfloor \frac{n + 2p - k}{s} \right\rfloor + 1 \qquad (1)$$

where n and n' are the lengths of the original and shortened sequences, respectively. After GELU activation, we can get $\mathbf{a}'$. The second CNN block learns shared position-wise kernels within a given window which can re-capture local information,

$$\mathbf{a}'' = \mathbf{a}' + \text{GELU}\left(\text{Norm}\left(\text{CNN}\left(\mathbf{a}'\right)\right)\right) \qquad (2)$$

The intuition is that certain information (e.g., positional awareness (Dai et al., 2020)) is lost during pooling, and requires restoration.The inclusion of layer normalization, residual connection, and nonlinearity follow the same rational as Transformer blocks (Vaswani et al., 2017). The total number of parameters for a single block of RedApt is 11.5M.

2.2 RedApt Integration into WAV2VEC 2

WAV2VEC 2 (W2V2) architecture starts with a CNN-based feature encoder to extract features from raw speech signals, while performing downsampling. A quantization module is attached on top of the feature encoder to learn discrete latent speech vectors. The output of the feature encoder is masked and forwarded into a context network, consisting of 24 Transformer blocks and 16 self-attention heads (for the LARGE configuration), to learn contextualized speech representations. The entire model is pretrained with a contrastive loss to distinguish a true masked latent vector from those generated by the model. After pretraining, only the feature extractor and context network are fine-tuned in the downstream tasks.

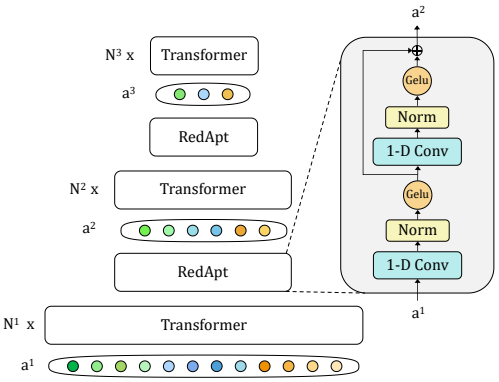

Figure 1: The RedApt architecture and integration.

RedApt progressively shrinks a temporal sequence during the forward propagation in the W2V2 Transformer-based encoder. Assuming an integration of m RedApt blocks into Transformer blocks, RedApt blocks compress a sequence by a factor of s^m. As each Transformer block has a quadratic complexity w.r.t. the input length n_0, pooling tensors has significant benefits on memory and compute requirements of W2V2. Denoting the sequence length at i-th layer of W2V2 as n_i and the stride size as s_i, the complexity of each subsequent Transformer layer (until the next pooling) is $\mathcal{O}((\frac{n_i}{s_i})^2)$ compared to the $\mathcal{O}(n_0^2)$ in vanilla Transformer block.

2.3 RedApt Integration into ST

We are motivated to capitalize on pre-trained modules, W2V2 and mBART (Liu et al., 2020), that were trained on large unlabelled and labelled data. Our ST model consists of the W2V2+RedApt encoder, and an mBART decoder. Since the goal of our work is to enable faster signal encoding, we train the encoder while freezing the decoder in all experiments. Unlike Gállego et al. (2021) that use stage-training, we train W2V2 and RedApt jointly in one step.

[4]Our code is available at https://github.com/mingzi151/w2v2-st

3 Experiments

In this section, we first describe our experimental settings (§3.1). Next, we investigate the effect of the number of RedApt blocks (§3.2) and their positions (§3.3) on speed, memory, and translation quality. Then, we evaluate our ST model on 8 language pairs from Must-C benchmarks (§3.4) and analyze inference time (§3.5). Lastly, we provide an ablation study on RedApt components (§3.6).

3.1 Experimental Settings

Dataset. We use the Must-C dataset (Cattoni et al., 2021), a multilingual ST corpus collected from TED talks. We experimented with 8 language pairs, using English (EN) as source and the followings as target: German (DE), Romanian (RO), Spanish (ES), French (FR), Dutch (NI), Portuguese (PT), Russian (RU), and Italian (IT). The data is pre-processed and filtered following steps outlined in Gállego et al. (2021). The best systems were selected on dev sets, and results are reported on test set (tst-COMMON). We use the EN-DE pair for the detailed analysis and ablation.

Implementation Details. Similar to the modern ST architectures (Gállego et al., 2021; Tang et al., 2020), we use pretrained W2V2 large as our encoder and pretrained mBART50 decoder as the decoder. We randomly initialize the top 3 layers of W2V2 in experiments involving RedApt and find that it enables faster convergence, verifying earlier observations by Sun et al. (2021). We free W2V2 feature extractor. For full details on training configurations and hardware, please refer to *Appendix* A.2.

Baseline. We use the SotA ST model from Gállego et al. (2021) as our baseline. The model uses a similar W2V2 large encoder, along with a CNN-based length adapter on top of the encoder which reduces the sequence length by a factor of 8. The decoder is similar to ours and is frozen during training for fair comparison between the two models. This model is denoted as W2V2+ hereafter. While RedApt offers various degrees of layer-wise reduction, for comparability in the translation experiments on 8 language pairs (§3.4), the configuration with the same reduction factor (i.e., 3 blocks of RedApt) and total encoder parameter size is used.

Metrics. We measure efficiency at inference in terms of *throughput* (the number of speech data that can be processed in a unit of time), *memory* (GPU memory usage); and *FLOPs* (floating-point operations performed given a single process, higher FLOPs means slower inference speed). See *Appendix* A.4 for more details. We use BLEU[5] to evaluate translation quality.

m	0	1	2	3	4
BLEU	26.51	27.69	27.61	27.42	25.86
Throughput ↑	1.00x	1.22x	1.26x	1.31x	1.35x
Memory ↓	1.00x	0.80x	0.77x	0.73x	0.67x
FLOPs ↓	1.00x	0.86x	0.84x	0.81x	0.76x

Table 1: BLEU, throughput, memory consumption, and FLOPs at training and inference for various number of RedApt blocks: $m = \{1, 2, 3, 4\}$, the positions of blocks are $\{[15], [15, 20], [15, 18, 19], [14, 15, 18, 19]\}$.

3.2 Selecting the Number of RedApt Blocks

We examine the impact of RedApt on translation quality and efficiency by injecting various number of RedApt blocks, $m = \{0, 1, 2, 3, 4\}$, where 0 refers to the baseline and the rest indicate our models. As a heuristic for setting the cap on m, we use the maximum representation length reduction that matches that of a corresponding text transcripts to avoid the risk of information loss. The intuition is that given the same content, the length of speech representations (after W2V2) should not be less than that of text representations; the former may vary depending on the degree of compression, whereas the latter is a fixed number. This reflects on the fact that text, unlike speech, carries only the content information that is essential for translation task. Further investigation of text representations optimality compared with speech is beyond our current focus and we leave it to future work.

Table 1 summarizes the results.[6] Overall, we achieve significant throughput speedup, memory footprint saving and FLOPs reduction as m increases, while the trend follows a law of diminishing returns. All models with $m \leq 3$ retained the same level of translation quality as the baseline ($m = 0$). Particularly for $m = 3$, memory consumption, throughput, and FLOPs are 0.73×, 1.27×, and 0.81× of the baseline. The gains from reduced computational cost can be re-invested to increase the batch size at inference and we report the changes of these metrics as the batch size varies in *Appendix* A.3. We observe a trade-off associated

[5] https://github.com/mjpost/sacreBLEU

[6] Different position configurations for each value of m are tried and we report the best results.

Model	Positions	BLEU	T.put↑	Mem↓	FLOPs↓
W2V2+	-	26.51	1.00x	1.00x	1.00x
RedApt	[2,5,6]	0.7$^{\diamond}$	1.47x	0.28x	0.45x
	[7,9,11]	22.02	1.41x	0.45x	0.58x
	[13,15,20]	27.24	1.41x	0.67x	0.76x
	[14,18,20]	26.83	1.33x	0.60x	0.80x
	[15,18,19]	**27.42**	1.31x	0.73x	0.81x
	[16,18,20]	27.17	1.28x	0.76x	0.83x
	[17,19,20]	26.78	1.24x	0.79x	0.85x

Table 2: Comparison of different position configurations in terms of translation quality, throughput (T.put), memory (Mem) and FLOPs, at training and inference time. $^{\diamond}$: models did not converge. **Bold**: Best BLEU score. Underline: Best configuration.

with m in translation quality and efficiency, and we set m to 3 in all our following experiments. For brevity, we use RedApt to refer to our ST models in which RedApt blocks are injected into the encoder, hereafter. In the next section, we will investigate various layers of placing RedApt blocks.

3.3 Selecting the Positions of RedApt Blocks

We investigated various positions of RedApt blocks. Since it is not ideal to experiment all 24^m choices, to determine optimal positions, we apply a backward selection mechanism starting from the configuration of [14,15,18,19] by removing one position, or replacing it with another position. For selection purposes, we segment Transformer networks of WAV2VEC 2 to 2 buckets, i.e., *low-mid* (0-11), *mid-top* (12-23). While we treat positions as hyper-parameter in our work, a more principled approach could frame it as neural architecture search (Elsken et al., 2019). We leave further investigation to future work.

The majority of the models with exceptional performance come from *mid-high* layers. Table 2 presents BLEU scores, memory usage, throughput and FLOPs for EN-DE, with different position configurations. We observe injecting RedApt into lower level of W2V2 leaves a major toll on BLEU. This verifies the earlier findings on text transformers (Goyal et al., 2020) that higher layers typically convey similar overlapping information while disturbing the lower layers could result in a great deal of information loss. Additionally, compressing sequences in lower levels has greater impact on the pre-trained weights in the subsequent layers, which can result in optimization issues.

We choose our best configuration, [13,15,20], which exhibits efficiency improvements of 1.41× in throughput, 0.67× in memory usage and 0.76× with FLOPs, and use it in our next translation experiments on 8 language pairs.

Model	DE	RO	ES	FR	NI	PT	RU	IT
W2V2+	26.51	24.66	30.04	36.26	31.08	32.67	17.19	22.13
RedApt	**27.24**	24.34	**30.49**	**37.59**	29.82	32.65	**18.08**	**25.73**

Table 3: Translation BLEU of the SotA model (W2V2+) and our model on 8 language pairs from Must-C.

3.4 Translation Quality on Must-C

Table 3 reports the results for the 8 Must-C language pairs, using our best configuration (§3.3). Our ST models outperform the baseline models (§3.1) on 5 language pairs by a large margin, while being comparable on the rest. On average, we observe a boost of 0.68 BLEU scores across 8 languages. We speculate that these gains could be attributed to the positive impact of dimensionality reduction on filtering out redundancy and noise from the representations, verifying the earlier observations by Zhang et al. (2020).

3.5 Inference time for ST

In order to measure the inference time for the entire ST model, we partitioned audio of EN-DE test set into 5 buckets based on length (seconds) Compared to the baseline, the decoding speedups are 7%, 7%, 5%, 3%, 3% for these buckets $(0, 4), [4, 8), [8, 13), [13, 20], (20, \infty)$, which have 1024, 896, 384, 128 and 64 examples, respectively. As expected, the efficiency gain in encoding tends to vanish in the full ST setup due to the depth of the mBART decoder (i.e., 12 layers) and the autoregressive decoding.

3.6 Ablation of RedApt components

To study the contribution from each components of RedApt block beyond the first CNN block (§2.1), we conduct an ablation by removing the remaining three components one at a time. The ablation (on EN-DE) for our best configuration (§3.3) indicates that removing the second CNN block leads to 0.44 BLEU decay, while removing either the GELU and LayerNorm leads to convergence issues (neither models converge). We report further details and positions in *Appendix* A.5.

4 Conclusion

We proposed a novel dimensionality reduction block, `RedApt`, to improve the efficiency of pretrained speech encoders, e.g. WAV2VEC 2 (W2V2), in speech translation (ST). We demonstrated that the integration of `RedApt` brings $1.41\times$, $0.67\times$ and $0.76\times$ in speedup, memory usage and FLOPs reduction at inference. Meanwhile, compared with SotA, our ST system on average improves translation quality by 0.68 BLEU scores over 8 MuST-C language pairs. As our future work, we will be investigating the impact of `RedApt` in other speech processing tasks (Yang et al., 2021), as well as learning the optimal positions for injecting `RedApt` blocks via neural architecture search.

5 Limitations

While hardware requirement is a common challenge shared across all modern ST models, it is worth mentioning that our work requires GPUs with 16 GB of memory for inference and 48 GB memory for training.

6 Ethics Statement

Our work is leveraging pretrained models of language (WAV2VEC2 for speech, mBART for text). However, our method is not designed or intended to rectify any of the well-documented issues of such models. Hence, our work inherits similar potential risks that these models pose.

References

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

Roldano Cattoni, Mattia Antonino Di Gangi, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021. Must-c: A multilingual corpus for end-to-end speech translation. *Computer Speech & Language*, 66:101155.

Heng-Jui Chang, Shu-wen Yang, and Hung-yi Lee. 2022. Distilhubert: Speech representation learning by layer-wise distillation of hidden-unit bert. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7087–7091. IEEE.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems*, 33:4271–4282.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017.

Marco Gaido, Mauro Cettolo, Matteo Negri, and Marco Turchi. 2021. Ctc-based compression for direct speech translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 690–696.

Gerard I Gállego, Ioannis Tsiamas, Carlos Escolano, José AR Fonollosa, and Marta R Costa-jussà. 2021. End-to-end speech translation with pre-trained models and adapters: Upc at iwslt 2021. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 110–119.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Cheng-I Jeff Lai, Yang Zhang, Alexander H Liu, Shiyu Chang, Yi-Lun Liao, Yung-Sung Chuang, Kaizhi Qian, Sameer Khurana, David Cox, and Jim Glass. 2021. Parp: Prune, adjust and re-prune for self-supervised speech recognition. *Advances in Neural Information Processing Systems*, 34.

Xian Li, Changhan Wang, Yun Tang, Chau Tran, Yuqing Tang, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. 2021. Multilingual speech translation from efficient finetuning of pretrained models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 827–838.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Zilun Peng, Akshay Budhkar, Ilana Tuil, Jason Levy, Parinaz Sobhani, Raphael Cohen, and Jumana Nassour. 2021. Shrinking bigfoot: Reducing wav2vec 2.0 footprint. *arXiv preprint arXiv:2103.15760*.

Tomasz Potapczyk, Paweł Przybysz, Marcin Chochowski, and Artur Szumaczuk. 2019. Samsung's system for the iwslt 2019 end-to-end speech translation task. In *Proceedings of the 16th International Conference on Spoken Language Translation*.

Elizabeth Salesky and Alan W Black. 2020. Phone features improve speech translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2388–2397.

Zewei Sun, Mingxuan Wang, and Lei Li. 2021. Multilingual translation via grafting pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2735–2747.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq s2t: Fast speech-to-text modeling with fairseq. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 33–39.

Felix Wu, Kwangyoun Kim, Jing Pan, Kyu J Han, Kilian Q Weinberger, and Yoav Artzi. 2022. Performance-efficiency trade-offs in unsupervised pre-training for speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7667–7671. IEEE.

Shu-Wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y. Lin, Andy T. Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, Tzu-Hsien Huang, Wei-Cheng Tseng, Ko-tik Lee, Da-Rong Liu, Zili Huang, Shuyan Dong, Shang-Wen Li, Shinji Watanabe, Abdelrahman Mohamed, and Hung-yi Lee. 2021. SUPERB: speech processing universal performance benchmark. In *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, pages 1194–1198. ISCA.

Rong Ye, Mingxuan Wang, and Lei Li. 2021. End-to-end speech translation via cross-modal progressive training. *arXiv preprint arXiv:2104.10380*.

Biao Zhang, Ivan Titov, Barry Haddow, and Rico Sennrich. 2020. Adaptive feature selection for end-to-end speech translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2533–2544.

Jinming Zhao, Hao Yang, Ehsan Shareghi, and Gholamreza Haffari. 2022. M-adapter: Modality adaptation for end-to-end speech-to-text translation. *arXiv preprint arXiv:2207.00952*.

A Appendix

A.1 Dataset

We use all 8 language pairs from Must-C. Please refer to Cattoni et al. (2021) for details for the size of the datasets and train/dev/test splits. We adopt the filtering techniques proposed in Gállego et al. (2021). We remove instances whose audios are over 25 seconds. We then filter out examples whose transcriptions that 1) have speaker names and non-textual events; 2) start with certain patterns indicating noise. Next, we apply ASR to audios and remove those whose ASR outputs have low WER scores.

To diversify training data, we also apply data augmentation on the audio data on-the-fly, an effective technique used in ST (Potapczyk et al., 2019). We apply "tempo" and "pitch" to make the model become invariant to speaking speeed, and "echo" to simulate echoing. Each instance is augmented with a probability of 0.8 where all effects are applied. We then normalize it to zero mean and unit variance. The parameters of tempo, pitch, echo-delay and echo-decay are to set to (0.85, 1.3), (-300, 300), (20, 200) and (0.05, 0.2).

A.2 Training Details

All models are trained with fairseq (Ott et al., 2019) on 4 RTX 6000 GPUs, using 16 floating point precision, for $25k$ updates. We use WAV2VEC 2[7] and the mBart50[8] decoder. We limit the source and target lengths and to 400k (i.e., 25 seconds) and 1,024 tokens, respectively. We use an Adam optimizer with (Kingma and Ba, 2015) $\beta_1 = 0.99$ and $\beta_2 = 0.98$. We set the dropout to 0.1, clip norm to 20, and the label smoothing value to 0.2. For the baseline models, we use a learning rate of 5e-04 and reduce it when loss stops improving. Depending on speech lengths for each source language, we set the average batch size being either 64, or 128. For our models, we use a learning rate of 5e-04 for DE and NL, 4e-04 for Fr and 3e-04 for the rest, and we also decrease the learning rate at plateau. We use an effective batch size of 64 for all language pairs. We set kernel size, stride and padding for the two CNN blocks in `RedApt` to <3, 2, 1> and <3, 1, 1>. We report BLEU results on single models without checkpoint averaging.

[7] https://dl.fbaipublicfiles.com/fairseq/wav2vec/wav2vec_vox_960h_pl.pt

[8] https://dl.fbaipublicfiles.com/fairseq/models/mbart50/mbart50.ft.1n.tar.gz

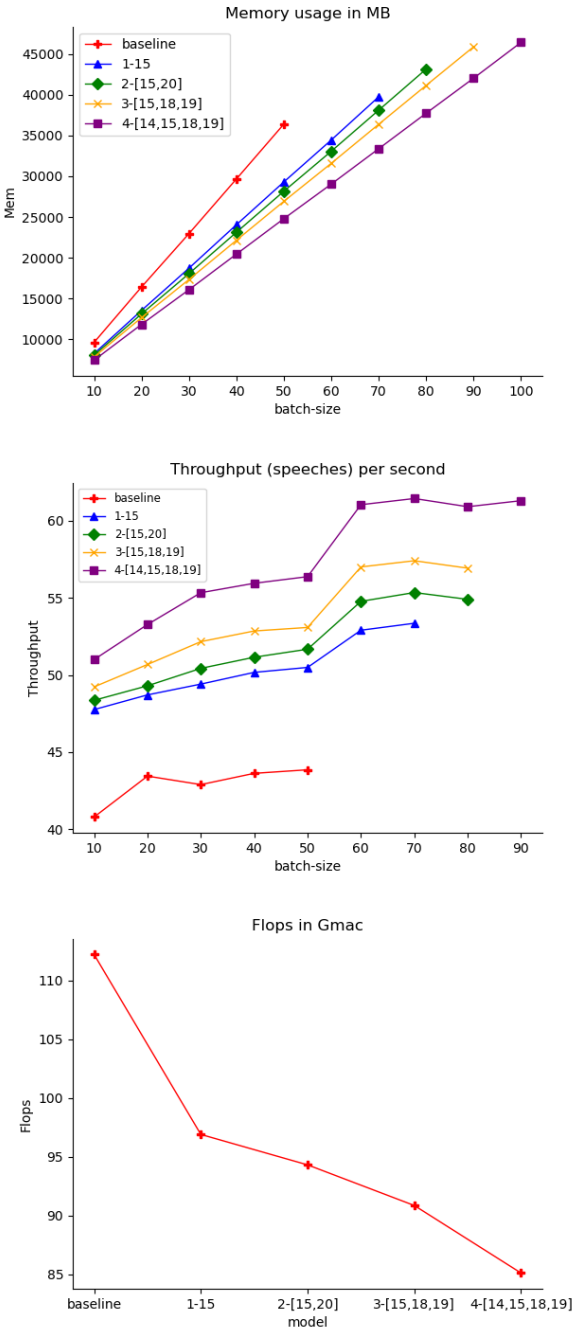

Figure 2: memory usage (Upper) throughput (middle) and FLOPs (bottom)

A.3 Memory and Throughput vs. Batch Size

See Figure 2 (upper, middle) for memory usage and throughput (bottom) for each model when we increase batch size until GPU memory gets full. Our methods have better utilization of GPU memory.

A.4 Measuring Throughput and FLOPs

We set the length of raw signals as 88,000, which is the average signals length in the training set, and the batch size as 64 when computing throughput and memory. We use batch size 1 for calculating FLOPs, whose value is mainly affected by the input length. All testings are performed on one RTX 8000 GPU.

A.5 Ablation of `RedApt` components

Table 4 shows the effect of removing each components in `RedApt`.

# Positions	2nd CNN	LayerNorm	GELU	BLEU
13-15-20	✓	✓	✓	27.24
	-	✓	✓	26.80
	✓	-	✓	◇
	✓	-	-	◇
15-18-19	✓	✓	✓	27.42
	-	✓	✓	27.11
	✓	-	✓	◇
	✓	-	-	◇

Table 4: Ablation on Acoustic Pooler. ◇: models did not converge.