Exploring Fine-Grained Early Exiting For Simultaneous Translation

Anonymous ACL submission

Abstract

Simultaneous translation is a difficult subset of neural machine translation with incomplete source context during most translation decisions that balances translation quality with latency as equal goals. Given that, a variety of computational reduction methods are popular for simultaneous translation. One underexplored method is early exiting, where tokens in a sequence cease computation before the final layer of a given model, being fed directly to another stack or classifier instead. In this work, we explore applying early exiting techniques popular with transformers to speech-to-text simultaneous translation (SimulST), achieving up to 7.0% savings on encoder FLOPs with a reduction of only 0.42 BLEU.

1 Introduction

Simultaneous translation, especially when rooted in speech modalities, is a difficult subset of neural machine translation that makes translation decisions with incomplete source context. Ideally, simultaneous translation systems achieve both high quality translations and a low lagging factor (low latency) between receiving source context and outputting an aligned translation. In practice, balancing translation quality and latency is a challenging and well-studied problem, but slight variations on transformer models (Vaswani et al., 2023) have provided a promising translation quality baseline from which to iterate on.

Many techniques exist to reduce end-to-end system latency across domains, such as layer and/or attention head pruning (Peer et al., 2022), various model distillation techniques (Sanh et al., 2020), or mixture-of-expert (MoE) approaches to feed-forward networks. Such approaches are useful, but can be limited either by their static nature (e.g. distilled model size set before training, expert network size) or by sometimes limited applicability (e.g. pruning is most effective for truly over-parameterized models). One often dynamic method for lowering end-to-end latency that is underexplored for simultaneous translation is early exiting, where tokens in a given sequence cease computation upon meeting some early exit criteria.

In this paper, we explore applying fine-grained early exit techniques that operate at a per-token level (as opposed to coarse-grained sequence-level early exits), inspired by the Depth Adaptive Transformer (DAT) (Elbayad et al., 2020) with some major adaptations. First and foremost, we apply our proposed approach, inspired by portions of DAT, to the acoustic encoder in a classical simultaneous translation end-to-end architecture (Inaguma et al., 2020). Second, we construct a custom additional training goal for the end-to-end model that encourages early exiting. Given that we are not early exiting directly from a decoder stack to a classifier, as DAT does, this is significantly more difficult to craft such that the model is encouraged to early exit without sacrificing too much translation quality.

Based on our proposed early exit scheme for acoustic encoders, adapted from some of the ideas in DAT with some of our own additions, we explore the hyperparameter space within our approach in a course-grained manner, providing what we believe to be a best set of values, in addition to exploring our method's application to English-to-German (en-de) and English-to-Spanish (en-es). On en-de, we achieve up to 7.0% FLOPs savings with only a 0.42 BLEU reduction, whereas our results for en-es are surprisingly modest.

2 Brief Background and Related Work

In the interest of brevity, we review only the basics of simultaneous translation efforts, focusing specifically on simultaneous speech-to-text translation (SimulST). Additionally, we briefly cover early exiting alongside some coarse-grained alternatives to the approach proposed by Depth Adaptive Trans-

former (DAT) (Elbayad et al., 2020).

2.1 Simultaneous Translation

A subset of full sentence-to-sentence neural machine translation (NMT), simultaneous translation requires that translation steps (or write decisions) are made without complete source context (increased by read decisions). Balancing read-write decisions to maintain low latency between aligned source context and translation output with enough source context to make high quality translation renders simultaneous translation necessarily more difficult. A simple, but popular, read-write scheduler is *wait-k* (Ma et al., 2019), where write decisions lag an aligned read decision by k steps, and we employ it for our explorations in this work.

2.2 Coarse-Grained Early Exits

As opposed to some alternatives (e.g. pruning, model distillation, MoE approaches), early exiting tends to be capable of achieving a more dynamic accuracy-throughput tradeoff during inference. Some coarse-grained approaches have been popularized within transformers, exemplified by those employed by DeeBERT (Xin et al., 2020) and EdgeBERT (Tambe et al., 2021). Both implementations, at a high level, estimate the entropy of a given sequence of tokens and, based on whether or not that entropy is under a certain threshold, early exit the entire sequence. This can result in an intuitive coarse-grained averaging of the computational needs of the entire sequence, as opposed to optimizing for individual token needs in a fine-grained manner, a likely sub-optimal approach to early exiting.

2.3 Fine-grained Early Exits

A comprehensive work on fine-grained early exits for transformers, Depth Adaptive Transformer (DAT) (Elbayad et al., 2020) features a number of early exit methodologies for decoder-only architectures with varying degrees of effectiveness. Several components of their overall proposed approaches are reusable outside of their original scope, with some modifications. Notably, applying early exit mechanisms, especially in a fine-grained, per-token manner, in decoders benefits significantly from access to a classifier that allows for immediate evaluation of the accuracy of an token that early exited computation. As such, applying variations on the techniques proposed by DAT to encoders requires some adaptations.

Algorithm 1 Early Exit Scheme (Input: X, FFN: $f_i(\theta)$)

$X = [x_0, x_1, x_3, ..., x_m]$
$Unexited_{list} = [0, 1, 2, ..., m]$
for i = 0 to N **do**
  $Y_i = [x_k \; for \; k \; in \; Unexited_{list}]$
  $Z_i = f_i(\theta;\; Y_i)$
  $X_i = [x_k \; in \; X_i \; or \; z_p \; in \; Z_i \; if \; p = k]$
  $K_i = h_i(\theta;\; (Z_i - Y_i))$
  $Unexited_{list} = [m \; for \; k_m \; in \; K_i \; \geq \; \eta]$
  if length($Unexited_{list}$) = 0 **then**
    $exit \; inference$
  end if
end for

3 Proposed Approach

Contrasting to the aforementioned coarse-grained early exiting approaches, we take inspiration from the Depth Adaptive Transformer (DAT) (Elbayad et al., 2020) and approach the problem of early exiting in speech-to-text simultaneous translation (SimulST) in a fine-grained manner, enabling individual tokens to early exit computation. Unlike the techniques proposed via DAT, which were only applied to transformer decoders, we focus on applying them to the more challenging half of a single end-to-end speech-to-text translation system: the acoustic encoder. Given the bi-directional nature of acoustic encoders, we cannot simply early exit to a classifier and instead propose a novel approach to the problem that allows early exited tokens to propagate their hidden states where necessary.

3.1 Acoustic Encoder Early Exit Implementation

The approach outlined in Algorithm 1 takes a granular approach to early exiting within the encoder stack, analogous to the high level approach of DAT in a transformer decoder stack. Intuitively, fine-grained early exiting in a per-token manner can allow for maximized performance via specific early exit tradeoffs per-token, as opposed to the aforementioned coarse-grained approaches where computational needs are averaged. For our purposes, we only early exit from feed-forward network (FFN), computation in an encoder stack, as removing tokens from the query during self-attention caused significant translation quality degradation.

For keeping track of which tokens have exited we simply employ a list of unexited token indices,

Algorithm 2 Target Set Generation and Loss Calculation (x = Early Exit Results)

for i=0 to number of tokens **do**
$sim = \frac{x_{ref} \cdot x_i}{max(||x_{ref}|| \cdot ||x_i||,\, 10^{-8})}$
$sim = sim \cdot (1 - \frac{L_{exit}-1}{LE})$
$Simularity[i] = sim$
end for
$target = Simularity > \tau$
$loss = BCE_{loss}(p_{exit}, target)$

$Unexited_{list}$, that is passed through the encoder stack. Each layer uses the $Unexited_{list}$ to filter out exited tokens after self-attention. Afterwards, the remaining tokens enter the current encoder layer's FFN. The output of the FFN is subtracted from the tokens pre-FFN processing, which we feed into the small early exit network which is composed of a single linear layer and a sigmoid activation for every encoder layer (different early exit networks per layer). From that network's output, the resulting values ranging from 0 to 1 dictate which tokens can exit the encoder stack via hyperparameter η, which is treated as an early exit confidence threshold. If the probability associated with a token is above the set η, the token's index is removed from $Unexited_{list}$ and it does not pass through the current encoder layer's or future layers' FFN. The model will continue through the encoder stack until there are no more IDs' on the $Unexited_{list}$ or the last layer of the encoder stack is reached.

3.2 Loss Calculations

Our proposed custom training goal is calculated via binary cross entropy, BCE_{loss}, and added onto the categorical cross entropy loss that's calculated for the entire encoder-decoder architecture's predictions. The input for this loss is generated by tracking the probabilities associated with a token exiting, k_i, in p_{list}. Unlike decoder-based early exiting schemes, such as DAT, the labels associated with a data set cannot be used as the target set for early exit loss calculations. To resolve this problem, we allow a copy of the original sequence being early exited to pass through all layers of the encoder stack and employ those values in generation of a target set for early exited values of tokens.

As outlined in Algorithm 2, target set generation starts with calculating the cosine similarity between early exited tokens and x_{ref} tokens. The resulting similarity is linearly scaled using the layer a token exited from L_{exit} and the total layers in the encoder LE. Such scaling sets a greater penalty for tokens that exited later in the encoder stack than ones that exited earlier in the encoder stack and discourages the model from seeking out maximal similarity by not early exiting at all. The scaled similarities are then compared to hyperparameter τ, if the similarity is less then τ the exit is considered invalid but if the similarity is great then τ then the loss is considered valid. The motivation for such system is to minimize the difference between the tokens when early exited and not early exited while also encouraging the model to make early exits via the linearly scaling penalty.

3.3 Comparing and Contrasting to Other Early Exit Approaches

As mentioned in Section 2, we differ significantly from other popular early exit schemes like those proposed in DeeBERT and EdgeBERT in a few ways. Most notably, we consider the computational needs of individual tokens, but even if the needs of individual tokens were taken into account in these schemes, we differ from their approaches in the level of dynamism in our proposed early exit decisions. Entropy estimations of the sequence in both schemes are only calculated once, whereas we allow a given acoustic encoder to make early exit decisions with a separate, compact network per-layer.

We do borrow intuition from the geometric-like approach outlined in Depth Adaptive Transformer (DAT) (Elbayad et al., 2020), employing a similar architecture to theirs for their early exit network via a single linear layer with a sigmoid activation. Unlike DAT, we deploy this network between every encoder layer (weights unshared between layers) within the acoustic encoder stack. Similar to DAT, a confidence threshold needs to be cleared for a given token to early exit, but unlike DAT's proposed approach (and due to deploying multiple networks), we do not tune the individual layer confidence thresholds on the validation set and instead engage in a simple hyperparameter search. Our intuition for no threshold fine-tuning is that multiple early exit networks can learn to depress or encourage early exits without individual tuning as might be necessary when deploying a single network.

Given the above behavior, our custom loss goals are superficially similar to a combination of the likelihood-based approach and correctness-based proposed by DAT for this geometric-like early exit

Language Pair	Orignal BLEU Score	BLEU Score	Savings
en-de	17.46	17.02	7.0 %
en-es	21.15	19.63	0.7 %

Table 1: BLEU score and FLOPs saving percentage for various τ and η values across language pairs on the tst-COMMON split of MuST-C.

τ	η	BLEU Score	Saving %
0.3	0.3	6.36	38.6%
0.3	0.5	15.58	0.7%
0.3	0.7	17.48	0.00%
0.5	0.3	8.38	39.0%
0.5	0.5	17.02	7.0%
0.5	0.7	17.32	0.0%
0.7	0.3	7.78	38.8%
0.7	0.5	17.73	0.0%
0.7	0.7	17.44	0.0%

Table 2: BLEU scores and saving percentages for various τ and η values on en-de. Configurations with no FLOPs savings failed to learn to early exit.

distribution, with the reference labels used for correctness being the fully calculated hidden states of the acoustic encoder. This represents a fundamentally more difficult version of representing the fitness of fine-grained early exiting, as no real reference labels can be employed.

4 Experiments

For the following experiments, we train all models via Fairseq (Ott et al., 2019), a popular sequence modeling toolkit, and evaluate them on SimulEval (Ma et al., 2020), the premier simultaneous translation evaluation framework. Models employed in these experiments were trained on MuST-C (Cattoni et al., 2021) across English-Spanish (en-es), and English-German (en-de). Our architecture of choice for these explorations is one inspired by an end-to-end model in the ESPnet-ST toolkit (Inaguma et al., 2020) with a modified cross-attention block to enable a *wait-k* translation schedule. Additional details related to model architecture and training hyperparameters can be found in our Appendix.

4.1 Coarse Grained Tuning of Proposed Hyperparameters

τ and η were selected via a coarse grained search seeking an optimal translation quality to FLOPs reduction tradeoff and this search is shown in Table 2. The search only dealt with looking at results generated on en-de via a wait-7 schedule. This analysis demonstrated that τ and η values of 0.5 produced the best reduction in FLOPs with a minimal reduction in BLEU score. While we settled on values of our proposed hyperparameters that minimized BLEU score reductions, one could engage with more aggressive approaches for additional FLOPs savings. For example, τ and η values of 0.5 and 0.3 respectively produced a 39.0% FLOPs savings, although the BLEU score reduction was significant. We assume that best values for τ and η at least partially hold across language pairs, although truly optimal values are likely language pair specific.

4.2 Evaluation Across Language Pairs

Using the values of η and τ derived from our coarse-grained search, our proposed method was tested also tested on en-es via a wait-7 schedule. The results are shown in Table 1. The model seemed performed best on the en-de language pair resulting is a 7.04% reduction in FLOPs, with only a 0.42 reduction in BLEU score. Surprisingly, the proposed method performs poorly for en-es. We posit that this is likely because our initial assumption regarding τ and η is incorrect and that coarse-grained searches are likely required to achieve reasonable results for each language pair.

5 Conclusion and Future Work

This paper explored fine-grained early exiting in the acoustic encoder of a SimulST model. When tested on the en-de language pair using a wait-7 translation schedule, with τ, and η values of 0.5, a FLOPs decrease of 7% in the acoustic encoder was observed and BLEU score only decreased by 0.4 when compared to the baseline implementation. Further development of this model could include refinement of target set generation, as this would help with training the model's early exit networks to better recognize when to exit tokens. In addition, further exploration into model hyperparameters would allow for better performance across language pairs, where performance currently suffers.

6 Limitations

While our proposed implementation demonstrates promising results, it has only been validated on SimulST, which limits the breadth of this work. Additionally, further tests need to be executed on avoiding continued token computation during attention.

References

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.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer.

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Enrique Yalta Soplin, Tomoki Hayashi, and Shinji Watanabe. 2020. Espnet-st: All-in-one speech translation toolkit.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. Stacl: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework.

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020. Simuleval: An evaluation toolkit for simultaneous translation.

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.

Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2022. Over-generation cannot be rewarded: Length-adaptive average lagging for simultaneous speech translation. In *Proceedings of the Third Workshop on Automatic Simultaneous Translation*. Association for Computational Linguistics.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul N. Whatmough, Alexander M. Rush, David Brooks, and Gu-Yeon Wei. 2021. Edge-bert: Sentence-level energy optimizations for latency-aware multi-task nlp inference.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference.

A Appendix

A.1 Licensing Information

Fairseq (Ott et al., 2019) is MIT-licensed and widely available for non-commercial use. SimulEval (Ma et al., 2020) and MuST-C (Cattoni et al., 2021) are licensed via Creative Commons (CC BY-SA 4.0 and CC BY-NC-ND 4.0 respectively).

A.2 Training Hyperparameters and Architectural Details

The model was trained on a convolutional transformer with a downsampling convolutional pre-net (downsampling by a factor of 4), 12 encoder layers, and 6 decoder layers. Encoder and decoder hidden dimensions were 256 and FFN dimensionality was 2048. Acoustic dimensionality was set to an 80 mel-filter bank.

Model training consisted of ASR pre-training and then SimulST training. During both training stages the overall loss of the system was calculated using label-smoothed cross-entropy and the early exit loss was calculated using binary cross-entropy. In both stages the Adam optimizer (Kingma and Ba, 2017) was used. During ASR pre-training a learning rate of 0.0007 was used. During SimulST training a warm up learning rate 0f 0.0001 was used on on the first 4000 updates, after the first 4000 updataes a learning rate of 0.00035 was used.

Both training stages used early stopping to evaluate when to cease training. Early stopping involves monitoring the model to see if it is still improving after a certain amount of epochs. Patience is the amount of epochs the model should compare to the dev set to see if the model is improving. ASR pre-training used a patience of 5 to determine when to stop, once ASR pre-training ends the top 5 checkpoints were averaged and used for SimulST training. For SimulST training a patience of 10 was used, and the best 10 checkpoints where averaged and used for evaluation.

During all runs the end of sequence signal from the decoder was suppressed until all source context is read in to ensure that FLOPs calculations were consistent.

A.3 Hardware and Estimated Computational Cost

Models were trained and evaluated on a single V100 32GB GPU. In total, we estimate roughly 40 GPU days went into experimentation and data collection.

A.4 Lack of Additional Latency Metrics

Length Adaptive Average Lagging (LAAL) (Papi et al., 2022) and similar latency metrics (especially computationally aware ones) are commonly employed for SimulST. Given that we engage in “force finishing," where the encoder must read in the entire source context before an end-of-sequence decision by the decoder, and employ a constant wait-7 schedule, LAAL and computationally aware variants do not quite appropriately reflect the changes to the latency of the system given that LAAL is mostly affected by the read-write scheduler and computationally aware variants are largely affected by the hardware the model is being run on. We posit that characterizing latency changes in FLOPs reductions is a more consistent reflection of changes to the model.