Computer Science
&
Data Science

Capstone Report - Fall 2022

Simultaneous Machine Translation with Deep Reinforcement Learning

Zecheng Wang

supervised by
Keith Ross

Preface

This project is on deep learning technology applied to real-life problems. At the same time, it is also an opportunity for me to test my ability to do original research after nearly two years of independent research as an undergraduate. After quite some time as a research assistant with Professor Yik-Cheung Tam, I finally had the courage to make one step forward: to come up with an idea to try and solve a problem on my own. With the constant support from Professor Keith Ross, I eventually developed the knowledge necessary to conduct every research step including identifying and defining the problem, doing a literature review, proposing an approach, implementing the idea, and reporting the results.

The problem of simultaneous machine translation has long been a challenging problem in the signal processing and natural language processing (NLP) community due to its "online" nature. This project aims to provide novel insight to this problem through the application of deep reinforcement learning for audiences within not only the NLP community but also those who are interested in tackling natural language tasks with reinforcement learning.

Acknowledgements

I'd like to thank Professor Keith Ross for supervising me in this project with great kindness and support during these trying times. I want to thank Professor Yik-Cheung Tam for taking me into undergraduate research.

I'd also like to thank my parents for their dedicated support throughout my academic career, and also my cat for the emotional support.

Abstract

Simultaneous machine translation (SiMT) remains a challenging task in that translation needs to take place before a source sentence ends. A SiMT system needs to determine when to output given the current source sequence to achieve a good trade-off between translation quality and latency. Ideally, the translation module and the latency control module should be jointly, directly optimized, which is challenging due to instability and lack of timestamped expert data. Recent works on online Transformer have provided insights on a joint, direct optimization, however, it was unclear whether such methods are sufficient in optimizing the trade-off. In this paper, we propose a reinforcement learning (RL) framework (MMA-PPO) to address the aspect of direct optimization by training an independent RL agent to traverse through states of the translation modeled by a pretrained online translation model. Our approach aims to enhance latency control of standard online transformers by optimizing the episodic return consisting of quality and latency metrics that can be non-differentiable.

Keywords

Here your keywords: eg. **Machine translation; Natural language processing; Reinforcement learning; Signal processing**

Contents

1 Introduction

Due to the recent advances in sequence-to-sequence modeling made possible by enhancing the attention mechanism [1] with matrix multiplication [2], there have been many works addressing the problem of simultaneous machine translation (SiMT), which has been a challenging task in machine translation and sequence modeling. Simultaneous machine translation, also known as real-time machine translation, is different from traditional machine translation tasks in that the translation needs to take place before a source sentence ends. This means that a SiMT system needs to determine when to output target words given the current, partial sentences with the goal of achieving low latency while maintaining translation quality. Naturally, an ideal SiMT agent should behave like a human expert real-time interpreter who can dynamically determine at which point during a stream of source language words they have obtained enough information to provide partial translations with certainty. In order to achieve or even surpass human-level performance, the main focus has been to mimic such human behaviors.

The core issue with simultaneous translation in machine learning is that the translation agent only has access to partial sentences during online decoding or inference. However, large corpus [3, 4] used to train machine translation models are mostly full-sentence translations pairs, meaning that it is difficult to train a SiMT model using supervised learning as a model trained with offline data will likely suffer from exposure bias [5, 6] due to the discrepancy of the data distribution between training and testing.

To tackle this, existing approaches on SiMT mainly fall into 4 categories. Heuristic or fix policy approaches [7, 8, 9, 10, 11, 12] were the first to be proposed. These methods enable training on a prefix-to-prefix basis [9] at the cost of the latency control policy being not trainable as they impose a predefined policy or a combination of predefined policies. There are also early works [13, 14, 15, 16] utilizing reinforcement learning [17] agents to learn adaptive policies by directly chasing the quality-latency objective as the rewards. However, these methods need to use translation models pretrained on full sentences to prevent instability issues when jointly training both modules. To simulate a supervised learning setting for simplicity and scalability during training, imitation learning [18, 19, 20, 21, 22, 22] has been an effective go-to framework which evolved from simple expert sequence imitation to meaningful unit prediction. Due to the human-imposed assumptions and heuristics involved when generating the "oracle" sequence, we speculate that the learned policy should be biased. More recently, the monotonic attention

mechanism for recurrent neural networks [23, 24] as well as monotonic multihead attention (MMA) for Transformer [25] enables online translation with implicit, a built-in policy which controls the latency, which satisfies joint optimization over quality and latency, as well as direct optimization towards minimizing latency.

Based on the recent development of the online approaches, we propose a novel reinforcement learning framework [1] which addresses the duality of joint training and direct optimization. We first train an online Transformer using MMA on a large corpus to achieve a SiMT model with reasonable performance. As the translation model has been trained to process prefixes, we freeze the translation module and incorporate it as part of the environment for a reinforcement learning agent to train in. Using Proximal Policy Optimization [26], we optimize the policy using a direct combination of the quality and latency metric: BLEU score [27] and Average Lagging [**?**]. Our contributions are three-folded:

1. We provided a comprehensive analysis of related works to provide a clear picture of the challenges SiMT poses.

2. We reran the experiments for MMA and discovered that the performance for MMA varied drastically over different checkpoints and types of attention and was very easy to overfit thus it was difficult to find a good trade-off.

3. We proposed a novel latency control agent using PPO. We incorporated the MMA model in the environment to provide representations of the source and current prefixes as well as a "suggested" action given by the MMA model. We call this method MMA-PPO.

Currently, MMA-PPO has not proven to bring good results due to the complexity of the problem as well as the limitation of resources. In the future, we plan to further refine the idea as well as the implementation to prove the feasibility of the concept.

2 Related Work

2.1 Fix and heuristic approaches

Early approaches to SiMT mostly propose fixed or heuristic policies. Cho et al [7] first introduce a simultaneous greedy decoding framework with a frozen translation system and heuristic waiting criteria like wait-if-worse, which is later improved by Dalvi et al [8]. Simultaneous greedy decoding

[1] Code will be released at https://github.com/Victor-wang-902/SiMT-PPO.git

only focuses on the decoding process with a frozen network, which implies the assumption that a network trained on full-sentence neural machine translation (NMT) [28] can perform just as well on sentence prefixes, which is often argued otherwise. The state-of-the-art at the time, STACL [9], features a wait-k policy (translation is always k tokens behind source) that enables fine tuning of a SiMT model from a full-sentence NMT model. Training of STACL is later improved by an ensemble of multiple latency paths [10][11], and steaming history [12]. STACL features anticipation as a by-product meaning that the model tends to predict words that should have been aligned with source tokens not yet available. Chang et al[29] argue that this long-distance reordering is in fact not desirable in the simultaneous context. They propose a gumbel-sinkhorn network architecture with CTC loss that reranks the encoder representations aiming to prevent the model from anticipating.

2.2 Reinforcement learning approaches

There are also early attempts using reinforcement learning (RL) [17]. Grissom II et al [13] first apply reinforcement learning on SVO-SOV translation (EN-DE). However, the system only focuses on policy assuming a perfect translation model, meaning that it is a theoretical investigation on the possibility of using RL on SiMT problems. Satija et al [14] use Deep Q-network (DQN) [30] with a fixed translation model and [15] improves on the previous work and develops a simultaneous greedy decoding strategy with policy gradient algorithm. [16] adds an additional prediction action as in [13]. The objective for most of these approaches, which is the reward function, is a combination of translation quality and latency. BLEU score (or its variations) [31] is used as a measurement of translation quality, which has also been an appeal to many in NMT generally as it eliminates the discrepancy between train objective (MLE mostly for supervised learning) and evaluation metric (predominantly, the BLEU score)[27].

However, these works argue that the RL agent under such frameworks could not be jointly optimized with the translation model. The reason is as follows. Suppose that we do not stop the gradient flow between the two modules. Technically, both modules will improve as training goes on. When an agent receives a high reward at the end of an episode, the agent won't be able to distinguish whether the increase in reward comes from the improvement of the translation model, or better latency control policy by the RL agent.

2.3 Imitation learning approaches

To address the issue with RL while still allowing an adaptive policy, there have also been methods that use supervised or imitation learning which makes the agent model learn a policy through generated "oracle" data [18] [19] [20] [21] [22]. The recent state-of-the-art method by Zhang et al [22] proposes training a Meaningful Unit Detector (MU) for segmentation. Once the input stream is segmented, each isolated segment is translated independently. Generation of the "oracle" segmentation reference is done by choosing segments that achieve the highest likelihood from an NMT. As I have mentioned in Section 1, this generation process does not guarantee that the "oracle" action sequence stems from an optimal policy.

2.4 Implicit policy approaches

The policy can also be implicit, meaning that in such frameworks, a single network plays the role of both the NMT model and the agent. Transducer networks [32] [33] are networks that compute the expected likelihood of the output by marginalizing over all possible alignments (paths), which is ideal for simultaneous translation. [32] enhanced this process by introducing attention between target representations and encoder states. [33] computes a set of probability (translation, action, buffer size) to "interpret" each state and a Poisson prior to regularization. During testing, translation probability is used for word prediction and action probability for whether to decode. While transducer networks are computationally expensive, one revolutionary breakthrough which enables joint optimization on the policy and translation is the monotonic attention mechanism [23] which modifies sequence-to-sequence attention heads such that they can only attend to one encoder state for each decoder timestep from left-to-right (uni-directional). This hard monotonic attention fits the online translation setting as only the prefix of a streaming input sentence is visible until the speaker finishes a sentence, and the output attention weights can also be used as the indicator of whether the network is ready to decode a word. Arivazhagan et al [24] propose a method that enables infinite lookback by applying soft attention overall visible encoder states so that translation can leverage history contexts. Ma et al [25] propose Monotonic Multihead Attention (MMA) which modifies the Transformer [34] attention heads such that each head produces an alignment resulting in more robust latency control.

2.5 Implicit policy and reinforcement learning

Only until very recently have monotonic attention and RL got integrated to enable joint optimization [35, 36]. Based on MMA, Li et al [36] introduce an RL agent to help MMA control latency. However, unlike RL methods mentioned in section 2.2, they utilize a heuristic reward scheme that gives sparse reward based on the performance change of both the MMA and the agent across epochs.

3 Solution

3.1 Problem definition

We now give a formal definition of the problem. Given a streaming input sequence of language $\mathcal{S}$: $X = \{x_0, x_1, \dots\}$ which progressively updates itself until a terminal timestep K resulting in a complete sequence $X^+ = \{x_0, x_1, \dots, x_K\}$, the objective is to generate an output sequence of language $\mathcal{T}$: $Y = \{y_0, y_1, \dots, y_U\}$ and a non-decreasing assignment: $T = \{t_0, t_1, \dots, t_U\}$ where $t_i \in \{0, 1, \dots, K\}$. The generation also satisfies the following conditions:

1. Uni-directionality: At output timestep i, only information from $x_j \in X^+, 0 \leq j \leq t_i$ can be utilized to output y_i, i.e.
$$P(Y|X) = \prod_{i=0}^{U} P(y_i|x_{<t_i}, y_{<i}); \tag{1}$$

2. Objective: Given golden (target) reference sequence Y^*, source sequence X, NMT model F_θ and Agent A_ϕ with weights θ and ϕ, the quality of the translation $\mathcal{C}_q(F_\theta(X), Y^*)$ is maximized while the latency $\mathcal{C}_l(A_\phi(X))$ is minimized. An example of a cost functional:

$$C(F_\theta, A_\phi) = -\Sigma_{(X,Y*)}(\mathcal{C}_q(F_\theta(X), Y^*) + \lambda \mathcal{C}_l(A_\phi(X))) \tag{2}$$

where $\mathcal{C}_q$ the quality metric and $\mathcal{C}_l$ the latency metric.

3.2 Method

Inspired by MMA and early reinforcement learning approaches, we propose MMA-PPO, a novel framework combining MMA with RL agent trained with PPO. While we don't perform joint optimization between MMA and PPO, we argue that training the two modules separately also mitigates the problem of exposure bias mentioned in section 1. Unlike ordinary sequence-to-

sequence models, MMA Transformer is able to perform monotonic attention which simulates partially visible sentences during the online setting. However, we speculate that the monotonic attention of MMA is not enough to approximate the policy well. Therefore, we design the RL framework to learn a separate policy externally while conditioning on the action choice the MMA model has made at each timestep. Intuitively, the RL agent acts as a "supervisor" making the final decision based on the current translation state while assisted by the MMA model. A detailed explanation of each component will be discussed below.

3.2.1 MMA

Monotonic Multihead Attention [25] adapts monotonic attention from RNNs [23, 24] to transformer architectures. In sequence modeling problems, we use an encoder-decoder model to project input X and target Y into encoder states $H = \{h_0, h_1, \ldots, h_K\}$ and decoder states $S = \{s_0, s_1, \ldots, s_U\}$. For RNNs, at the i-th decoding step during inference, the monotonic attention process can be expressed as

$$e_{i,j} = MonotonicEnergy(s_{i-1}, h_j) \tag{3}$$

$$p_{i,j} = Sigmoid(e_{i,j}) \tag{4}$$

$$z_{i,j} \sim Bernoulli(p_{i,j}). \tag{5}$$

This means that at every timestep, we take a sample from the attention distribution, which is the compatibility between the last target token s_{i-1} and the j-th encoder state. We traverse through h_j's until we get $z_{i,j} = 1$, meaning that we'd like to attend to the current encoder state, i.e. $t_i = j$ which satisfies Equation 1. Moving on to Transformer, we instead have the attention product as:

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{6}$$

where $Q = S; K, V = H$. Similarly, one can impose monotonicity:

$$e_{i,j}^{l,h} = (\frac{h_j W_{l,h}^K (s_{i-1} W_{l,h}^Q)^T}{\sqrt{d_k}})_{i.j} \tag{7}$$

$$p_{i,j}^{l,h} = Sigmoid(e_{i,j}) \tag{8}$$

$$z_{i,j}^{l,h} = Bernoulli(p_{i,j}) \tag{9}$$

where l, h denotes the l-th layer and the h-th head of a transformer and W's the projection matrices. The selection of heads is eventually decided by the "slowest" one. Note that such a process involving sampling prevents backpropagation. During training, the expected attention is computed instead.

3.2.2 SiMT as Markov Decision Process

According to our problem formulation in Section 3.1, the latency control can be modeled as a Markov Decision Process (MDP). That is, we view each configuration of a partial source sequence X and the current target sequence Y as a state s, and the choice of whether to "read" a source token or to "write" a target token as binary actions $a \in \{READ, WRITE\}$. And at the end of the translation for a sentence, the agent receives a reward r in terms of how well it did, which is a combination of quality and latency metrics. Note that in this case, the agent is only responsible for deciding whether to read or write while exactly which token is decided by the environment, namely, the MMA translation model. We can see that the sequences the agent gets as the next state is conditioned only on the current state and the action it has taken, but not on the previous states.

3.2.3 PPO

We use Proximal Policy Optimization [26] as the training algorithm for our RL agent. PPO is an on-policy algorithm whose core idea is to prevent the new updated policy from deviating too much from the old policy. In our implementation, we use PPO-Clip from `stable-baselines3` [37], which optimizes the policy as follows:

$$\theta_{k+1} = argmax_\theta E_{s,a\sim\pi_{\theta_k}}(L(s, a, \theta_k, \theta)) \tag{10}$$

$$\text{where } L(s, a, \theta_k, \theta) = min(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), clip(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon) A^{\pi_{\theta_k}(a|s)}(s, a)). \tag{11}$$

Here θ are the policy parameters, π_θ is the policy under parameters θ, ϵ a hyper-parameter controlling the clipping range, and A is the advantage function derived from a value network V_ϕ. After the update of the policy network, V_ϕ is updated via mean squared error with the collected rewards-to-go. Note that the choice of RL algorithms does not matter as long as the algorithm can process discrete action spaces, which is binary in our case.

3.2.4 Reward

For the RL agent, we need to design a reward scheme that can correctly reflect how well the agent is doing. While it is unclear how to design rewards for intermediate terms, we define the reward at terminal states with BLEU [31] as the quality metric and Average Lagging[9] as the latency metric. Following the definition from Equation 2, we have:

$$C(F_\theta, A_\phi) = -\Sigma_{(X,Y*)}(BLEU(F_\theta(X), Y^*) + \lambda AL(A_\phi(X))). \tag{12}$$

Average Lagging is a metric developed and used in STACL [9] with the following definition:

$$AL(x, y) = \frac{1}{\tau(|x|)}\Sigma_t^{\tau(|x|)}(g(t) - \frac{t-1}{r}). \tag{13}$$

Here, $\tau(|x|)$ denotes the cut-off step when X stops updating while the translation is still going; $g(t)$ denotes the number of source tokens read at decoding step t; and $r = \frac{|y|}{|x|}$. This design prevents the agent from getting penalized for longer target sentences, and it has great interpretability, as a wait-k model which follows the source stream after reading k tokens has an Average Lagging of k.

We choose BLEU and AL to be our reward functions for a reason. Since the reward signal doesn't have to be differentiable, we think that it would be more beneficial if we eliminate the discrepancy of the objective during training and inference, which has been mentioned in Section 2.2, as both BLEU and AL are non-differentiable and widely used during evaluation.

3.2.5 Overall Architecture

Finally, we present the overall architecture of our framework. The workflow for MMA-PPO during inference is illustrated in Figure 1. During training, we first train our online Transformer (MMA) using offline datasets. Once we have a trained model, we train the RL agent. The training procedure is very similar to the inference procedure as the main difference should come from different algorithms. As is shown in Figure 1, the environment for training the agent consists of a transcript emitter and the MMA model. Suppose an episode starts with the transcript emitter giving the first token of a random sentence. The token is first passed into the MMA model, outputting three elements: the suggested action (sampled from a Bernoulli distribution as is mentioned in Section 3.2.1), encoder states, and decoder states.

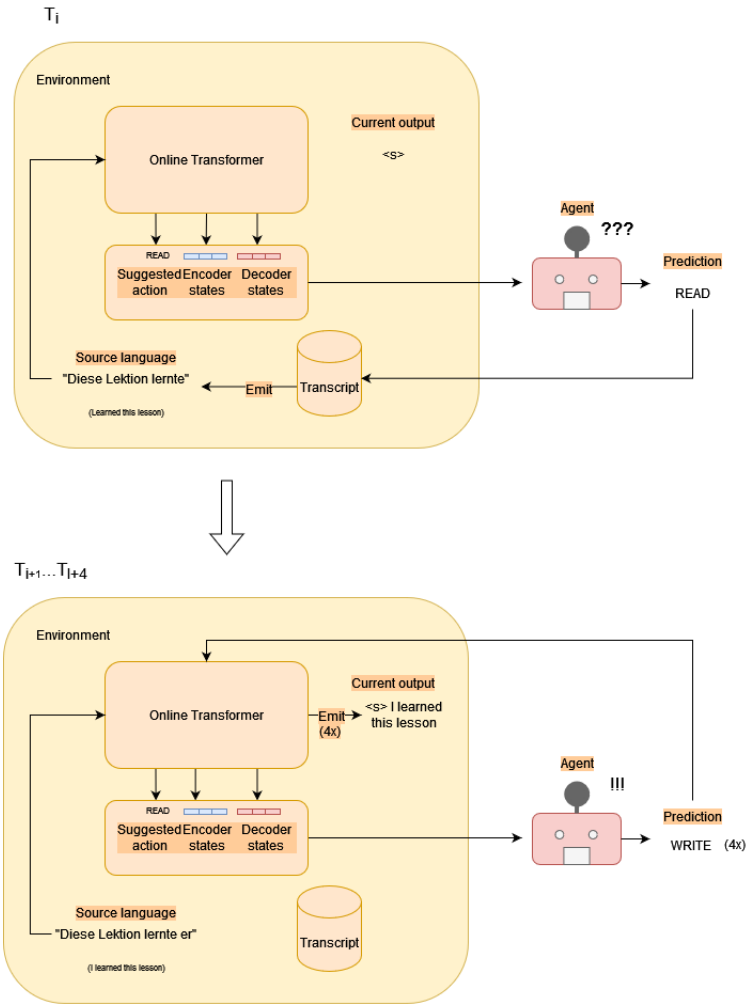

Figure 1: MMA-PPO workflow during inference. This figure showcases the agent's desired behavior when translating German to English. At timestep T_i (upper), the agent is not confident given the transformed observation of the state as the source sentence is likely to be missing a subject. Note that at this timestep, the suggested action agrees with the agent's final decision. At timestep, T_{i+1} to T_{i+4} (bottom), the agent perceives enough information in the source sentence and decides to output target words despite the suggested action still being READ.

As is also mentioned in Section 3.2.1, the encoder states and decoder states are representations of the source tokens as well as the target tokens (which start with a special signal `<s>`). These three elements are then passed as observations to the agent to receive a predicted action given the current policy. The predicted action is then passed back to the environment to make a step. If the action is READ, then a source token will be emitted changing the state of the environment; if the action is WRITE, the MMA model will project the decoder states with an output layer to get a target prediction and append it to the output, resulting in state change as well. This process terminates when the MMA model outputs an end signal, or the maximum steps are reached. For

the former case, a reward consisting of BLEU and AL will be given to the agent which is defined in Section 3.2.4. Naively, we set $\lambda = -1$. Given the rollout process of the environment by a policy, any RL algorithms that deal with discrete action spaces should be able to train the agent.

In order for the agent to capture the relations between the encoder states and the decoder states better, we use a 2-layer Transformer decoder as a feature extractor attending decoder states to encoder states. Mean pooling is then applied on the sequence length dimension as in SBERT [38] to reduce the features to a fixed observation space. To incorporate the suggested action, we apply linear regression to this value and append it to the end of the features extracted from the tiny Transformer.

4 Results and Discussion

We first trained two types of MMA model[25], MMA-IL [24] and MMA-H [23], on WMT15 [39] DE-EN dataset. We then compare the performance between different checkpoints of the two variants in terms of quality-latency trade-off.

Then, we used the trained MMA to train an RL agent using PPO. Unfortunately, due to limitations in time and resources, We didn't manage to train an effective PPO agent. Therefore, we will analyze the reason for failure for this section.

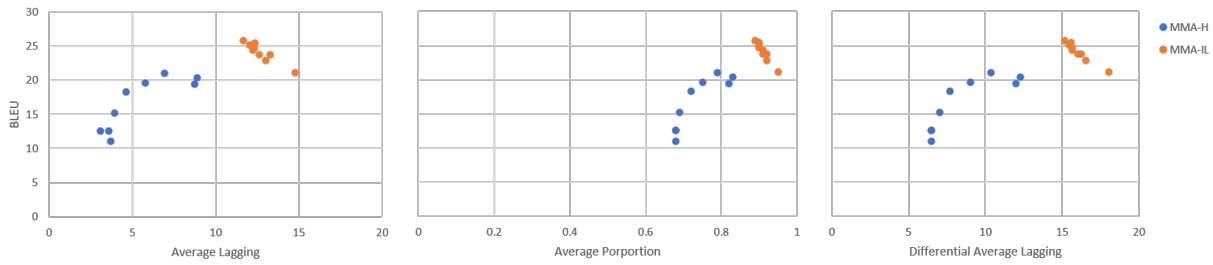

Figure 2: BLEU versus latency trade-off. Latency metrics include: Average Lagging, Average Proportion, and Differentiable Average Lagging.

4.1 Experiment details

For the first part of the experiment, we used the `fairseq` [40] toolkit. We followed the training procedure from the original paper, except that we used a smaller model architecture with 512 dimensions and 6 layers for encoder and decoder each. We kept all the other hyper-parameters the same.

To evaluate MMA models, we used the `SimulEval` library for an online translation environment.

We evaluated the models' translation quality with BLEU, and latency with three different metrics: Average Lagging, Average Proportion [7], and Differentiable Average Lagging [24].

For the second part of the experiment, we modified the `SimulEval` library to create a custom online latency control environment compatible with `gym` [41] API and incorporated the MMA model. Upon each reset of the environment, the first token of a randomly sampled sentence gets encoded. The observation space was implemented as a `gym.spaces.Box` of size (3, 1024, 512) with the first channel filled with the suggested action, the second and third being the encoder and decoder states padded to have a sequence length of 1024. The Transformer feature extractor of the agent took the encoder and decoder states as input while the suggested action was received as a scalar appended to the extracted features after passing through a `Linear(1,1)` layer. Therefore, the core module for the agent has dimension 512+1.

4.2 Performance trade-off of MMA models

As is shown in Figure 3.2.1, different attention type achieves trade-offs of varied scale. With the recommended hyper-parameters, MMA-H shows a clear sign of overfitting toward lower latency as training goes on. However, While MMA-IL has a much smaller variance in terms of trade-off, it is still gradually improving towards a higher BLEU score and lower latency despite being trained for the same number of iterations. From the experiments, we argue that it's not very straightforward to directly control this trade-off. Potentially, we can address this issue using MMA-PPO, which may control or optimize the latency without sacrificing quality.

End of training log	
Mean rollout episode length	106
Mean rollout episodic reward	-58.3
KL divergence	0.0
Learning rate	4e-5
Loss	4.3e+10
Policy gradient loss	7.24e-06
Value loss	9.03e+10

Table 1: Training log of typical failed attempts

4.3 Discussion on training of MMA-PPO

Using the MMA-IL with the highest quality score, we trained a PPO agent with various hyper-parameter settings and environment reward schemes. However, none of the experiments converged to a non-trivial policy. Table 1 is an example of the training log at the end of training. We

observed from our experiments that:

1. The agent was easy to converge to a locally optimal policy. We did a lot of modifications to the reward scheme, such as giving negative rewards at each timestep or giving penalties for invalid actions (READing when source was already empty). The agent would then adapt to such modifications to mitigate the penalties received. For example, for the former, the agent would learn to get to a terminal state as soon as possible no matter if the translation was good. For the latter, the agent simply learned to only predict WRITE. If we combined them both, the agent would intertwine between two actions trying to avoid penalties while getting to a terminal state as fast as possible. None of these behaviors helped with finding the actual path toward the quality-latency rewards.

2. The episodic reward would steadily increase to a point where it stopped, justifying the behavior we described above.

3. The agent also had a huge value loss but a very small policy gradient loss. Considering the nature of our environment which had a very large state space, a high-value loss would mean the value network was constantly changing its parameters which were incompatible with states observed later during training.

5 Discussion

In this project, the main challenge is the novelty of the project itself. Few recent works have tried to take the reinforcement learning approach to tackle this problem. Therefore, I didn't have any similar prior works that I could refer to detailedly and I had to research, design, and implement everything either through experience from other projects or just by thinking over and over. Obviously, the framework I have proposed is very crude with a lot of improvement to make and hyper-parameters to tune in order to really work properly. And it's nowhere near being able to "outperform" other approaches as the state-of-the-art approaches using segmentation and imitation learning are very effective empirically. However, I still believe in the potential of reinforcement learning applied in SiMT. Therefore, in future work, I'd like to further investigate the issues mentioned in Section 4.3. Specifically, I'd like to tune the reward scheme and the hyper-parameters more carefully. For example, I need to think of a way to normalize the terminal rewards regardless of the length of the sentences. The issue with a sparse reward scheme like this

may also be responsible for its ineffectiveness. In addition, I'd also like to investigate more on the uniqueness of this translation environment, which may have special mathematical properties that complicate the training process.

6 Conclusion

In this paper, we have done an extensive analysis of existing SiMT works, pointing out the limitations while justifying the use of Reinforcement learning. We have replicated the experiments on MMA models, pointing out the challenge of achieving a quality-latency trade-off. We have proposed and implemented a novel framework integrating online transformers with deep reinforcement learning for better policy control. Our experiments on the new framework have yet to be successful due to the difficulty in hyper-parameter-tuning and reward design for reinforcement learning. As future work, we will continue investigating the reason behind the ineffective training we are seeing now, and hopefully fix it and enable reinforcement learning to be competitive among the other SiMT approaches.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[3] L. Barrault, O. Bojar, M. R. Costa-jussà, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi, C. Monz, M. Müller, S. Pal, M. Post, and M. Zampieri, "Findings of the 2019 conference on machine translation (wmt19)," in *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*. Florence, Italy: Association for Computational Linguistics, August 2019, pp. 1–61. [Online]. Available: http://www.aclweb.org/anthology/W19-5301

[4] M. Cettolo, M. Federico, L. Bentivogli, J. Niehues, S. Stüker, K. Sudoh, K. Yoshino, and C. Federmann, "Overview of the IWSLT 2017 evaluation campaign," in *Proceedings of the 14th International Conference on Spoken Language Translation*. Tokyo, Japan: International Workshop on Spoken Language Translation, Dec. 14-15 2017, pp. 2–14. [Online]. Available: https://aclanthology.org/2017.iwslt-1.1

[5] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.06732

[6] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *Advances in neural information processing systems*, vol. 28, 2015.

[7] K. Cho and M. Esipova, "Can neural machine translation do simultaneous translation?" *CoRR*, vol. abs/1606.02012, 2016. [Online]. Available: http://arxiv.org/abs/1606.02012

[8] F. Dalvi, N. Durrani, H. Sajjad, and S. Vogel, "Incremental decoding and training methods for simultaneous translation in neural machine translation," *CoRR*, vol. abs/1806.03661, 2018. [Online]. Available: http://arxiv.org/abs/1806.03661

[9] M. Ma, L. Huang, H. Xiong, R. Zheng, K. Liu, B. Zheng, C. Zhang, Z. He, H. Liu, X. Li, H. Wu, and H. Wang, "STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3025–3036. [Online]. Available: https://aclanthology.org/P19-1289

[10] M. Elbayad, L. Besacier, and J. Verbeek, "Efficient wait-k models for simultaneous machine translation," *CoRR*, vol. abs/2005.08595, 2020. [Online]. Available: https://arxiv.org/abs/2005.08595

[11] S. Zhang and Y. Feng, "Universal simultaneous machine translation with mixture-of-experts wait-k policy," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7306–7317. [Online]. Available: https://aclanthology.org/2021.emnlp-main.581

[12] B. Zheng, K. Liu, R. Zheng, M. Ma, H. Liu, and L. Huang, "Simultaneous translation policies: From fixed to adaptive," *CoRR*, vol. abs/2004.13169, 2020. [Online]. Available: https://arxiv.org/abs/2004.13169

[13] A. Grissom II, H. He, J. Boyd-Graber, J. Morgan, and H. Daumé III, "Don't until the final verb wait: Reinforcement learning for simultaneous machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1342–1352. [Online]. Available: https://aclanthology.org/D14-1140

[14] H. Satija and J. Pineau, "Simultaneous machine translation using deep reinforcement learning," 2016.

[15] J. Gu, G. Neubig, K. Cho, and V. O. K. Li, "Learning to translate in real-time with neural machine translation," *CoRR*, vol. abs/1610.00388, 2016. [Online]. Available: http://arxiv.org/abs/1610.00388

[16] A. Alinejad, M. Siahbani, and A. Sarkar, "Prediction improves simultaneous neural machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3022–3027. [Online]. Available: https://aclanthology.org/D18-1337

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[18] B. Zheng, R. Zheng, M. Ma, and L. Huang, "Simpler and faster learning of adaptive policies for simultaneous translation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1349–1354. [Online]. Available: https://aclanthology.org/D19-1137

[19] ——, "Simultaneous translation with flexible policy via restricted imitation learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5816–5822. [Online]. Available: https://aclanthology.org/P19-1582

[20] P. Arthur, T. Cohn, and G. Haffari, "Learning coupled policies for simultaneous machine translation," *CoRR*, vol. abs/2002.04306, 2020. [Online]. Available: https://arxiv.org/abs/2002.04306

[21] A. Alinejad, H. S. Shavarani, and A. Sarkar, "Translation-based supervision for policy generation in simultaneous neural machine translation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1734–1744. [Online]. Available: https://aclanthology.org/2021.emnlp-main.130

[22] R. Zhang, C. Zhang, Z. He, H. Wu, and H. Wang, "Learning adaptive segmentation policy for simultaneous translation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 2280–2289. [Online]. Available: https://aclanthology.org/2020.emnlp-main.178

[23] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," in *Proceedings of the 34th International Conference*

on Machine Learning, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 2837–2846. [Online]. Available: https://proceedings.mlr.press/v70/raffel17a.html

[24] N. Arivazhagan, C. Cherry, W. Macherey, C. Chiu, S. Yavuz, R. Pang, W. Li, and C. Raffel, "Monotonic infinite lookback attention for simultaneous machine translation," *CoRR*, vol. abs/1906.05218, 2019. [Online]. Available: http://arxiv.org/abs/1906.05218

[25] X. Ma, J. M. Pino, J. Cross, L. Puzon, and J. Gu, "Monotonic multihead attention," *CoRR*, vol. abs/1909.12406, 2019. [Online]. Available: http://arxiv.org/abs/1909.12406

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[27] L. Wu, F. Tian, T. Qin, J. Lai, and T.-Y. Liu, "A study of reinforcement learning for neural machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3612–3621. [Online]. Available: https://aclanthology.org/D18-1397

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[29] C.-C. Chang, S.-P. Chuang, and H.-y. Lee, "Anticipation-free training for simultaneous machine translation," in *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*. Dublin, Ireland (in-person and online): Association for Computational Linguistics, May 2022, pp. 43–61. [Online]. Available: https://aclanthology.org/2022.iwslt-1.5

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[31] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, 2002, p. 311–318. [Online]. Available: https://doi.org/10.3115/1073083.1073135

[32] D. Liu, M. Du, X. Li, Y. Li, and E. Chen, "Cross attention augmented transducer networks for simultaneous translation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 39–55. [Online]. Available: https://aclanthology.org/2021.emnlp-main.4

[33] Y. Miao, P. Blunsom, and L. Specia, "A generative framework for simultaneous machine translation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6697–6706. [Online]. Available: https://aclanthology.org/2021.emnlp-main.536

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[35] Y. Lee, J. Shin, and Y. Kim, "Simultaneous neural machine translation with a reinforced attention mechanism," *ETRI Journal*, vol. 43, no. 5, pp. 775–786, 2021.

[36] M. Li and R. Doddipatla, "Improving hs-dacs based streaming transformer asr with deep reinforcement learning," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 154–161.

[37] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[38] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *CoRR*, vol. abs/1908.10084, 2019. [Online]. Available: http://arxiv.org/abs/1908.10084

[39] O. r. Bojar, R. Chatterjee, C. Federmann, B. Haddow, M. Huck, C. Hokamp, P. Koehn, V. Logacheva, C. Monz, M. Negri, M. Post, C. Scarton, L. Specia, and M. Turchi, "Findings of the 2015 workshop on statistical machine translation," in *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, pp. 1–46. [Online]. Available: http://aclweb.org/anthology/W15-3001

[40] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[41] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.