A Generative Framework for Simultaneous Machine Translation

Yishu Miao
Imperial College London
ByteDance
ym713@ic.ac.uk

Phil Blunsom
Universtiy of Oxford
DeepMind
phil.blunsom@cs.ox.ac.uk

Lucia Specia
Imperial College London
University of Sheffield
l.specia@ic.ac.uk

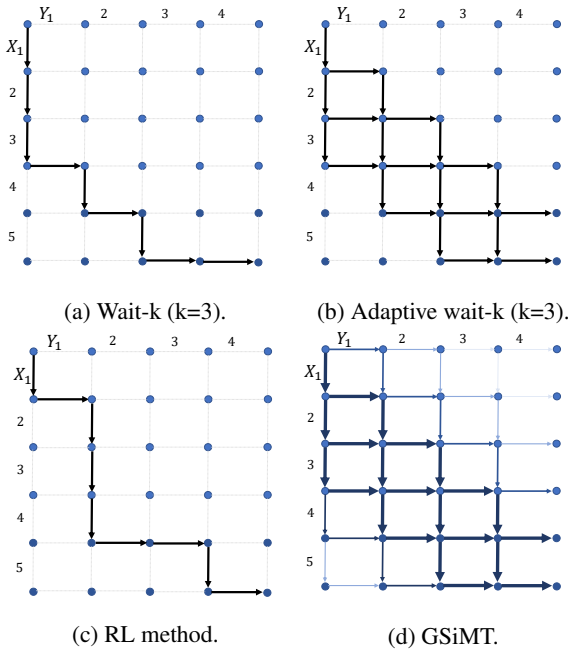
Abstract

We propose a generative framework for simultaneous machine translation. Conventional approaches use a fixed number of source words to translate or learn dynamic policies for the number of source words by reinforcement learning. Here we formulate simultaneous translation as a structural sequence-to-sequence learning problem. A latent variable is introduced to model *read* or *translate* actions at every time step, which is then integrated out to consider all the possible translation policies. A re-parameterised Poisson prior is used to regularise the policies which allows the model to explicitly balance translation quality and latency. The experiments demonstrate the effectiveness and robustness of the generative framework, which achieves the best BLEU scores given different average translation latencies on benchmark datasets.

1 Introduction

The fundamental challenge of simultaneous machine translation (SiMT) is the balance between the translation quality and the latency. It is non-trivial to find an optimal translation strategy, as there is generally a rivalry between the two objectives, i.e. reading more source words before translating leads to better translation quality, but it in turn results in higher latency due to the longer time for reading.

Conventional Wait-k policies (Ma et al., 2019) put a hard limitation over the buffer size k[1], which guarantees low latency but weakens flexibility and scalability when handling long and complicated language pairs. Alternatively, reinforcement learning (RL) approaches (Gu et al., 2017; Satija and Pineau, 2016; Arthur et al., 2021) learn a dynamic policy using a combined reward of a quality metric like the BLEU score and AL (average lagging)[2].



(a) Wait-k (k=3). (b) Adaptive wait-k (k=3). (c) RL method. (d) GSiMT.

Figure 1: Example of translation paths of different simultaneous translation models.

However, the poor sample efficiency make it very difficult to learn a robust SiMT model with RL.

In this paper we propose a generative framework with a latent variable that dynamically decides between the actions of *read* or *translate* at every time step, enabling the formulation of SiMT as a structural sequence-to-sequence learning task. Figure 1 depicts the examples of possible translation paths of different models. Wait-k only explores one hypothesis, while adaptive wait-k ensembles the other hypothesises with lower k. However, the hypothesises of reading more than k words before translating are not considered (e.g. inversion and reordering in long sequence translations). The RL models apply dynamic policies which can explore all the possible hypothesises, but the gradient estimator conditioned on discrete samples has large variance and the variance issue gets worse for long sequences. Instead, Our proposed generative simultaneous machine translation model (GSiMT)

[1]The number of read source words minus the number of translated target words.

[2]A metric for evaluating translation latency by how many words have been read on average before translating a word.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6697–6706
November 7–11, 2021.

integrates out all the hypothesises by a dynamic programming algorithm (Algorithm 1) with the help of the introduced latent variable. It does not suffer from such large variance issue, and can be easily and efficiently learned by gradient back-propagation on GPU hardware.

The generative model can be modelled as a neural transducer (Graves, 2012; Yu et al., 2016). However the vanilla neural transducer is not designed for SiMT. Because it is optimised by the cross-entropy of target words, it naturally prefers *read* actions over *translate* actions in order to see more contexts before translation, which intuitively can result in better translation quality but high latency.

Here, we propose to extend the neural transducer framework to modern Transformer-based translation models (Vaswani et al., 2017), and introduce a re-parameterised Poisson distribution to regularise the latency (i.e. how many source words are read before translating a target word). Inspired by the fast-alignment work by Dyer et al. (2013), the translation model generally favors word alignments distributed close to the diagonal. We hypothesise that the optimal sequence of *translate* actions in SiMT is also located close to the diagonal. Thus the Poisson prior acts as context-independent regularisation on the buffer size proportional to the distance between the current position and the diagonal. This ensures that the number of read source words will not grow indefinitely without translating any target words, while the soft boundary, due to the regularisation, still allows the model to consider complicated/long simultaneous translation cases.

To demonstrate the effectiveness of the proposed framework, we evaluate our generative models on two benchmark datasets: WMT15 (Bojar et al., 2015) for text-only SiMT and Multi30K (Elliott et al., 2016) for multimodal SiMT. Compared to a number of strong baseline models, Wait-k, Adaptive Wait-k and an RL-trained policy, our proposed model achieves the best performance on both BLEU scores and average lagging (AL). Our contributions can be summarised:

- A Transformer-based neural transducer model for simultaneous machine translation.
- Poisson prior for effectively balancing the translation quality and latency.
- State-of-the-art SiMT results (BLEU & AL) on benchmark datasets, and the BLEU scores are on-par-with consecutive MT models.

2 Related Work

Conventional SiMT methods are based on heuristic waiting criteria (Cho and Esipova, 2016) or fixed buffering strategy (Ma et al., 2019) to trade off the translation quality for lower latency. Although the heuristic approaches are simple and straightforward, they lack of scalability and cannot generalise well on longer sequences. There is also a bulk of work attempting to improve the attention mechanism (Arivazhagan et al., 2019) and re-translation strategies (Niehues et al., 2018) for better translation quality. Recently, Zheng et al. (2020) extends the fixed Wait-k policies into adaptive version and ensembles multiple models with lower latency to improve the performance, but one still needs to choose a hard boundary on the maximum value of k. By contrast, our GSiMT model considers all the possible paths with a soft boundary modelled by Poisson distribution, which leads to a more flexible balance between quality and latency.

RL has been explored (Gu et al., 2017) to learn an agent that dynamically decides to *read* or *translate* conditioned on different translation contexts. Arthur et al. (2021) further applies extra knowledge on word alignments as the oracle to improve the learning. However, the high variance of the estimator is still a bottleneck that hinders the applicability of RL in structural sequence-to-sequence learning. The proposed GSiMT model combines the merits of both the Wait-k policies and RL.

Deep learning with structures has been explored in many NLP tasks, especially for sequence-to-sequence learning. Kim et al. (2017) implements structural dependencies on attention networks, which gives the ability to attend to partial segmentations or subtrees without changing the sequence-to-sequence structure. Tran et al. (2016) parameterises the transition and emission probabilities of an HMM with explicit neural components, and Jiang et al. (2016) applies deep structural latent variables to implement the dependency model with valence (Klein and Manning, 2004) and integrates out all the structures in end-to-end learning. Our GSiMT model is based on neural transducer model. Previously, Graves (2012) presents an RNN-based neural transducer for phoneme recognition, and Yu et al. (2016) explores an LSTM-based neural transducer for MT. The uni-directional variant model of Yu et al. (2016) is similar to our proposed GSiMT model, however it is implemented as a vanilla neural transducer, which is not optimised for low la-

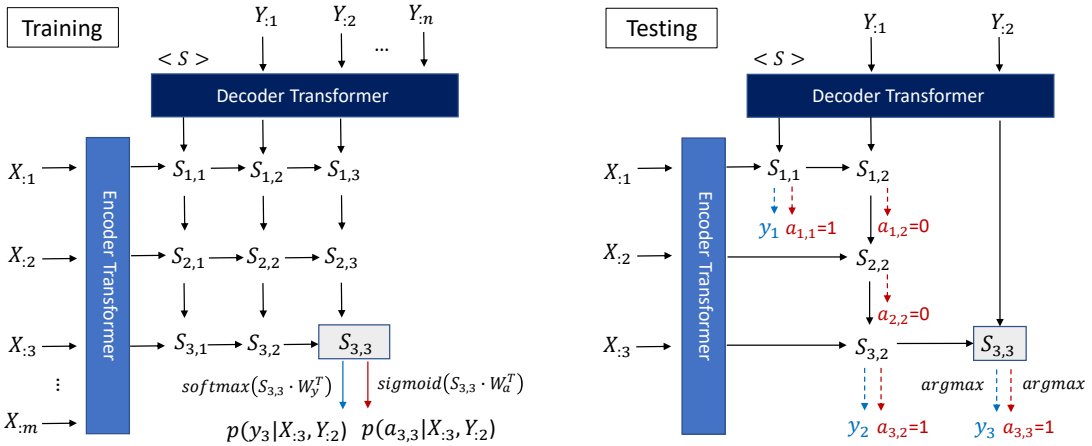


Figure 2: During training, all the contextualised representations $\mathbf{S}_{i,j}$ will be used to compute the translation distribution $p(y_j|X_{:i}, Y_{:j-1})$ and action distribution $p(a_{i,j}|X_{:i}, Y_{:j-1})$, while in testing the model takes the inputs X in real-time and dynamically produces y_j and $a_{i,j}$ until all the inputs have been read.

tency and hence performs poorly on SiMT. Therefore, the Poisson prior for regularising the latency is the key component to enable neural transducer models work on SiMT.

3 Model

3.1 Generative Model

We use $X_{:m}$ and $Y_{:n}$ to represent the source language sequence and target language sequence with lengths m and n. $X_{:i}$ represents the sub-sequence $\{x_1, x_2, ..., x_i\}$. The structural latent variable $a_{i,j}$ (0 or 1) represents the action (*read* or *translate*). Specifically $a_{i,j} = 0$ means reading an extra source word and $a_{i,j} = 1$ means translating the target word y_j. The translation position Z is introduced as an auxiliary variable to simplify the equations, where $z_j = i$ denotes that there i source words have been read when decoding the jth word y_j. Similar to neural transducer (Graves, 2012; Yu et al., 2016), the generative model can be formulated as[3]:

$$p(Y_{:j}|X) = \sum_{i=1}^{|X|} p(y_j|X_{:i}, Y_{:j-1}) p(z_j = i, Y_{:j-1}|X_{:i})$$

Translation distribution. Given the contextualised representation $\mathbf{S}_{i,j}$, the translation distribution of y_j:

$$p(y_j|X_{:i}, Y_{:j-1}) = \text{softmax}(\mathbf{S}_{i,j} \cdot \mathbf{W}_y^T) \quad (1)$$

Specifically, $\mathbf{W}_y^T$ is the projection matrix for word prediction and we leave out the bias terms for simplicity. $\mathbf{S}_{i,j}$ is the state output conditioned on source words $X_{:i}$ and target words $Y_{:j-1}$:

$$\mathbf{S}_{i,j} = g(\text{Enc}(X_{:i}), \text{Dec}(Y_{:j-1})) \quad (2)$$

where Enc and Dec are uni-directional Transformers based encoder and decoder. Different from conventional consecutive NMT model e.g. T5 (Raffel et al., 2020) where encoder is a bi-directional Transformer, the model has no access to the full input stream when translating. Figure 2 shows the training process where $S_{i,j}$ is computed for all the sub-sequences at positions i,j.

Position distribution. The position distribution jointly models the translation position z_j, and the subsequence $Y_{:j-1}$:

$$p(z_j = i, Y_{:j-1}|X_{:i}) \quad (3)$$
$$= \sum_{i'=1}^{i} p(z_j{=}i|z_{j-1}{=}i', X_{:i}, Y_{:j-1}) \cdot p(Y_{:j-1}|X_{:i'})$$

Here, we can recurrently decompose the position distribution into a sum of products for all the possible sub-sequence $Y_{:j-1}$ given read source sequence $X_{:i'}$ and the transitions from $z_{j-1} = i'$ to $z_j = i$, i.e. there are $i - i'$ source words newly read before translating y_j.

Switch distribution. To model all the possible transitions from $z_{j-1} = i'$ to $z_j = i$, we employ the switch distribution:

$$p(z_j = i|z_{j-1} = i', X_{:i}, Y_{:j-1}) \quad (4)$$
$$= \begin{cases} 0 & \text{if } i < i' \\ \alpha_{i,j} & \text{if } i = i' \\ \alpha_{i,j} \cdot \prod_{k=i'}^{i-1}(1 - \alpha_{k,j}) & \text{if } i > i' \end{cases}$$

and

$$\alpha_{i,j} = p(a_{i,j} = 1|X_{:i}, Y_{:j-1}) = \text{sigmoid}(\mathbf{S}_{i,j} \cdot \mathbf{W}_a^T)$$

[3]Slightly different from Yu et al. (2016), where the distribution is modelled by alignment probability and word probability. Here we apply Translation distribution and Position distribution instead.

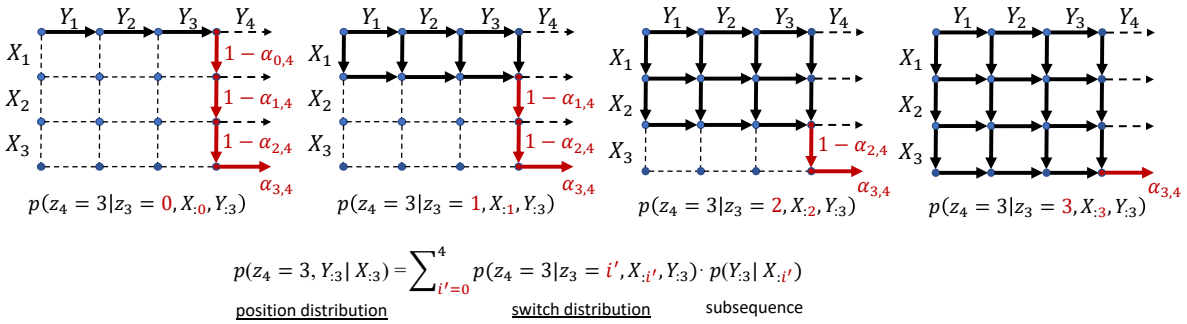


Figure 3: An example of decomposing a position distribution into a sum of products of switch distributions and subsequence generation probabilities.

where $\mathbf{W}_a^T$ is the linear projection to the action space, and Z is a monotonic sequence ($z_j \geq z_{j-1}$), hence for the transitions $i < i'$, the switch probability is zero.

Figure 3 shows a simple example for decomposing $p(Y_{:4}|X_{:3})$ into switch distributions and sub-sequence translations. For the transitions $i > i'$, it accumulates $i - i'$ *read* actions plus one *translate* action, so the switch probability is $\alpha_{i,j} \cdot \prod_{k=i'}^{i-1}(1 - \alpha_{k,j})$. Here, the *read* or *translate* actions are conditionally independent given the translation history.

Objective. In SiMT, we explicitly assume the last target word y_n is translated after reading all source words, hence the final objective can be simplified as:

$$\begin{aligned} p(Y|X) &= p(y_n|X_{:m}, Y_{:n-1}) \cdot p(z_n = m, Y_{:n-1}|X_{:m}) \\ &= p(y_n|X_{:m}, Y_{:n-1}) \cdot \\ &\quad \sum_{i=1}^{m} p(z_n = m|z_{n-1} = i, X_{:m}, Y_{:n-1}) \cdot p(Y_{:n-1}|X_{:i}) \end{aligned}$$

One caveat is that this objective does not encourage low latency translations when optimised by maximum log-likelihood, since the model can read as many source words as possible in order to have the best translation quality. Ideally, the lowest latency means that for all the target words y_j, the model reads one source word at every time step after translating a target word (i.e. the translation positions $z_j = i$ are close to the diagonal of a $m * n$ matrix as much as possible). Therefore, we need an extra regularisation to focus the probability mass of the translation positions along the diagonal.

3.2 Poisson Prior

Dyer et al. (2013) proposes a log-linear diagonal reparameterisation for fast word alignments, which helps the IBM 2 model by encouraging the probability mass to be around the diagonal. This in turn also notably improves efficiency over the vanilla IBM 2 model. Although SiMT is more complex than word alignment, the diagonal reparameterisation can act as a strong regularisation to favor the *translate* actions happening around the diagonal, which can yield balanced actions resulting in high quality and low latency.

Therefore, we introduce a prior distribution to regularise the maximum number of source words that can be stored (b_j) when decoding the jth word (y_j). To that end, we apply Poisson distribution as it is generally used for modelling the number of events in other specified intervals such as distance, area or volume. The distance between the absolute positions (i and j) and the diagonal can be easily modelled as discrete values to be regularised by Poisson, where the probability decreases when the distance grows. Here we re-parameterise a Poisson distribution:

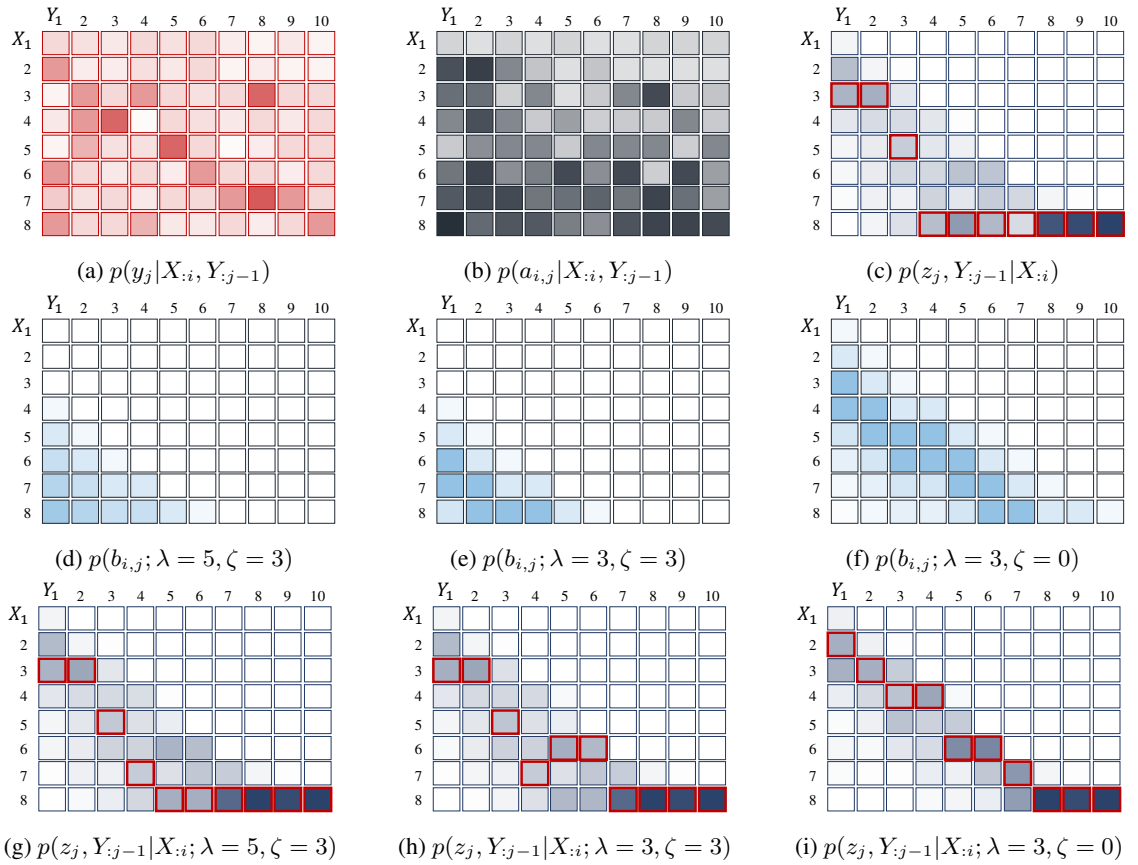
$$p(b_j = i; m, n) = \begin{cases} 0 & \text{if } d(i,j) < 0 \\ \frac{e^{-\lambda}\lambda^{d(i,j)}}{d(i,j)!} & \text{if } d(i,j) \geq 0 \end{cases}$$

and

$$d(i,j) = \lfloor i - j \cdot \frac{m}{n} - \zeta \rceil \tag{5}$$

where $d(i,j)$ is the distance of current position to the diagonal, which is rounded for simplicity. The free parameter λ is the mean of Poisson distribution, and ζ is the free parameter denoting the default offset of the current position to the diagonal. Different from *translate* positions $z_j = i$ which depend on the inputs $X_{:i}$ and $Y_{:j-1}$, $b_j = i$ is independent to the translation context, and is only conditioned on the absolute positions i, j. Therefore, we modify the **position distribution**:

$$\begin{aligned} &p(z_j = i, Y_{:j-1}|X_{:i}) \qquad (6) \\ &= \sum_{i''=1}^{m} p(z_j = i, Y_{:j-1}|X_{:i}, b_j = i'') \cdot p(b_j = i''; m, n) \end{aligned}$$



(a) $p(y_j|X_{:i}, Y_{:j-1})$ (b) $p(a_{i,j}|X_{:i}, Y_{:j-1})$ (c) $p(z_j, Y_{:j-1}|X_{:i})$

(d) $p(b_{i,j}; \lambda = 5, \zeta = 3)$ (e) $p(b_{i,j}; \lambda = 3, \zeta = 3)$ (f) $p(b_{i,j}; \lambda = 3, \zeta = 0)$

(g) $p(z_j, Y_{:j-1}|X_{:i}; \lambda = 5, \zeta = 3)$ (h) $p(z_j, Y_{:j-1}|X_{:i}; \lambda = 3, \zeta = 3)$ (i) $p(z_j, Y_{:j-1}|X_{:i}; \lambda = 3, \zeta = 0)$

Figure 4: Visualisations of the distributions in a generative SiMT model example. The depth of color represents the probability mass in each slot. In the sub-figures (c) (g) (h) and (i), the red rectangles highlight the $argmax$ along the 1-st dimension. **In the first row**, (a), (b) and (c) show the translation distribution, the *translate* action probability and the position distribution respectively. The indices of each slot correspond to the actual positions of i and j. Specifically, the position distribution (c) is generated by the vanilla GSiMT without Poisson prior distribution. **In the second row**, (d), (e) and (f) are the re-parameterised Poisson distribution under different λ and ζ. **In the third row**, (g), (h) and (i) are the position distribution after integrating out the Poisson prior (d), (e) and (f). Compared to the original position distribution (c), the generative models notably put more emphasis on the positions along the diagonal, which acts as a flexible regularisation to balance translation quality and latency.

Here, we make the assumption that the number of source words that have been read i cannot exceed the maximum size b_j. Hence, for all the cases $b_j < i$, the probability $p(z_j = i, Y_{:j-1}|X_{:i}, b_{j<i})$ equals to 0. Then, the **position distribution** can be further simplified as:

$$\begin{aligned} &p(z_j = i, Y_{:j-1}|X_{:i}) \\ &= p(z_j = i, Y_{:j-1}|X_{:i}, b_j \geq i) \cdot p(b_j \geq i; m, n) \end{aligned} \tag{7}$$

Having the Poisson prior, we can directly replace the Eq. 3 with Eq. 7 for computing the position distributions at different positions i and j. Figure 4 shows examples of how different values of λ and ζ affect the position distribution with the help of Poisson. For example, (i) has the lowest values of λ and ζ, so the Poisson prior distribution puts the strongest regularisation on the diagonal. Different from the wait-k models that apply hard limitations or the vanilla neural transducer model without regularisation, GSiMT with Poisson prior combines both advantages, which in turn yields a robust generative SiMT model.

3.3 Training

The training of the proposed GSiMT model follows the standard maximum log-likelihood optimisation. As the generalisation probability of the target sentence Y depends on the sum of the probabilities of its sub-sequence, we employ dynamic programming to construct the computation graph as illustrated in Algorithm 1. With the help of *autograd* computation of deep learning platforms, gradients can be automatically computed and efficiently back-propagated for optimisation.

Compared to the vanilla consecutive NMT, the overhead of the dynamic programming is actually very small, since most of the computations are sum and product in the low dimensional space. Gener-

Algorithm 1 Generative Simultaneous MT

Input: source sequence X, target sequence Y
Initialisation: model parameters Θ
repeat
  for batch $\in$ minibatches **do**
    for $j \in [1, n]$ **do**
      for $i \in [1, m]$ **do**
        Compute Subsequence State: $S_{i,j}$
        Compute Translation Prob: $p(y_j|X_{:i}; Y_{:j-1})$
        Compute Action Prob: $p(a_{i,j}|X_{:i}; Y_{:j-1})$
        Compute Buffersize Prob: $p(b_j = i; m, n)$
      end for
    end for
    for $j \in [1, n]$ **do**
      for $i \in [1, m]$ **do**
        for $i' \in [1, i]$ **do**
          Compute Switch Probability: $p(z_j = i|z_{j-1} = i', X_{:i}, Y_{:j-1})$
        end for
        Compute Position Probability: $p(z_j = i, Y_{:j-1}|X_{:i})$
        Compute Subsequence Probability: $p(Y_{:j}|X_{:i})$
      end for
    end for
    Compute log-likelihood $\log p(Y_{:n}|X_{:m})$
    Compute gradients and update
  end for
until Convergence

ally, the computations of the sub-sequence states ($\mathbf{S}_{i,j}$) consume most of the resources ($O(mn)$), which generally higher than the Adaptive Wait-k approach ($O(k*(m+n))$) (Zheng et al., 2020) and conventional Wait-k ($O(m+n)$) (Ma et al., 2019). However, as shown in Figure 2, in the testing process the computation cost is reduced to the same as Wait-k policies which is $O(m+n)$. Overall, it is a fair compromise in training time to achieve high quality SiMT decoding. More importantly, the dynamic policy grants the ability to process long and complicated translation pairs.

It is worth mentioning that the Poisson prior distribution is only employed for regularising the training, but it is not required at testing time, as the *translate* action distributions have implicitly learned to translate the target words with low latency. Hence, the lengths of the sequences m and n are known during training, but they are not used at test time. During test, we simply use the average length ratio of the whole dataset.

4 Experiments

4.1 Datasets & Settings

We experiment with the proposed models on two commonly used datasets: WMT15 DE→EN (text-only SiMT), and Multi30K (multimodal SiMT).

For WMT15 DE→EN, we follow the exactly the same preprocessing procedure as in (Ma et al., 2019; Zheng et al., 2020). BPE (Sennrich et al., 2016) is applied to achieve 35K vocabulary and we process 4.5M parallel corpus for training, 3K sentences of newstest-2013 for validation and 2,169 sentences of newstest-2015 for testing.

Following (Ma et al., 2019; Zheng et al., 2020), we apply the base version of Transformers with the same parameters in Vaswani et al. (2017) as the backbone. Instead of updating all the parameters from scratch, we pretrain the encoder and decoder (both are uni-directional Transformers) as consecutive NMT model for 10 epochs. Then we freeze the Transformers parameters, and apply 256 batch size and 1e-4 learning rate for training the generative models. On PyTorch (Paszke et al., 2019) platform, each epoch takes around 40 minutes with Adam (Kingma and Ba, 2014) on single V100 GPU[4].

The checkpoints with best performance in 5 runs on development datasets are chosen for testing BLEU (Papineni et al., 2002) and AL (average lagging) (Ma et al., 2019). For GSiMT models, we empirically fix $\lambda = 3$ for all the experiments, and use ζ as the free parameter to achieve different AL.

For Multi30K (Elliott et al., 2016), we use all three language pairs EN→FR, EN→DE and EN→CZ with the image data from Flickr30k as extra modality and flickr2016 as test dataset. We build multimodal models with the goal of testing the generalisation ability of the generative models with extra modalities. To that end, we concatenate the **object detection features** applied in Caglayan et al. (2020) into the state representation $\mathbf{S}_{i,j}$ and maintain the rest of the neural network the same as the unimodal SiMT. The other models (RL, Wait-k and Adpative Wait-k) incorporate the same features as well. Here, as the size of data is small, we apply a smaller Transformers with 4 layers, 4 heads, 512 model dimension and 1024 for linear connection.

4.2 Translation Quality & Latency

Table 1 shows the SiMT performance for the benchmark models and our proposed generative models on the WMT15 DE→EN dataset. **RL** is our implementation of Gu et al. (2017) with policy gradient method. All the numbers for **Wait-k** and **Adaptive-Wait-k** are quoted from Zheng et al. (2020).

[4]If the full parameters are trained, 1 epoch takes around 3 hours with 512 batch size on 8 V100 GPUs. However only marginal improvement is observed compared to the pretraining strategy.

Model	German-English (WMT15)							
	BLEU ↑	AL ↓	BLEU ↑	AL ↓	BLEU ↑	AL ↓	BLEU ↑	AL ↓
RL (Gu et al., 2017)	22.12	3.16	23.81	4.66	24.31	5.52	25.22	6.71
Wait-k (Ma et al., 2019)	25.22	3.76	26.29	4.70	27.42	5.77	27.73	6.66
Adaptive-Wait-k (Zheng et al., 2020)	26.73	3.63	27.84	4.79	28.41	5.33	29.20	6.60
GSiMT-Possion-T5	28.31	3.79	29.18	4.61	29.59	5.41	29.30	6.25
GSiMT-Poisson	**28.82**	**3.64**	**29.50**	**4.45**	**29.78**	**5.13**	**29.63**	**6.24**
GSiMT-NT	29.79	9.75	-	-	-	-	-	-
Consecutive NMT	30.24	28.58	-	-	-	-	-	-

Table 1: SiMT performance on WMT15 DE→EN. The models in the first group are the benchmark models for simultaneous machine translation. The second group is the variants of our proposed GSiMT. The third group is the consecutive NMT model, which provides the upper bound on BLEU score as it has access to the entire source stream. To fairly compare the BLEU under different AL, we apply 4 columns to limit the AL in the similar range but compare the BLEU score. The numbers of Wait-k and Adaptive-Wait-k models are achieved by training different models with k from 1 to 10 and $k_{min} = 1$, $k_{max} = 10$ (Zheng et al., 2020). For both GSiMT-Possion-T5 and GSiMT-Poisson, we apply $\zeta = 4, 5, 6, 7$ respectively to achieve the corresponding AL scores in each block. We highlight the best performance by BLEU score with bold numbers in each block. The underlined results are from the models that are not optimised for translation latency, which are used for reference only.

Model	En-Fr (Multi30k)		En-Cz (Multi30k)		En-Ge (Multi30k)	
	BLEU ↑	AL ↓	BLEU ↑	AL ↓	BLEU ↑	AL ↓
RL (Gu et al., 2017)	54.39	4.01	23.30	2.24	31.23	3.08
Wait-k (Ma et al., 2019)	56.20	3.38	23.31	3.54	33.75	3.47
Adaptive-Wait-k (Zheng et al., 2020)	57.16	3.32	26.9	3.11	33.68	2.99
DEC-OD (Caglayan et al., 2020)	57.90	3.65	28.13	2.83	34.40	2.37
GSiMT-Possion-T5	58.45	3.28	28.92	3.06	**36.23**	**2.58**
GSiMT-Poisson	**58.89**	**3.17**	**29.93**	**2.71**	36.11	2.65
GSiMT-NT	58.81	7.32	29.22	5.21	35.78	6.55
Consecutive NMT	59.29	13.10	30.65	13.10	36.84	13.10

Table 2: SiMT performance on Multi30K dataset. The models in the first group are the benchmark models for **multimodal** simultaneous machine translation. In addition to the models in Table 1, DEC-OD (Caglayan et al., 2020) is an RNN based model with an extra attention layer to attend to object detection features while carrying out translation. The numbers of other models in the first group are from our implementations, of which the state outputs are concatenated with the same visual features from Caglayan et al. (2020) for multimodal SiMT. For better comparison, we only report the BLEU scores with AL around 3. Similarly, the underlined results are from the models that are not optimised for translation latency, which are used for reference only. For both GSiMT-Possion-T5 and GSiMT-Poisson, we apply $\zeta = 3$ for all of the language pairs.

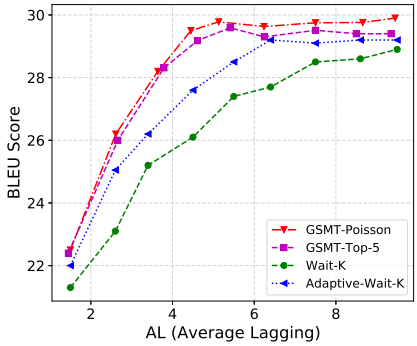


Figure 5: Overview of the performance of different models: BLEU scores versus average lagging (AL).

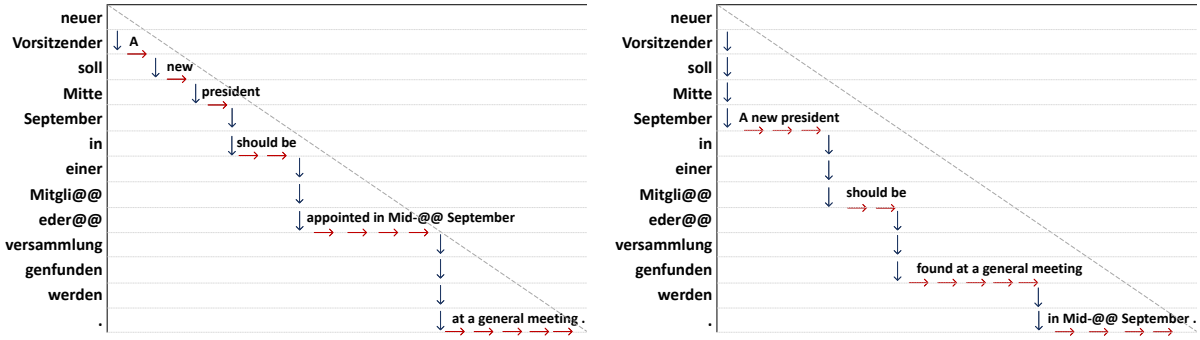
GSiMT-Possion is our proposed generative model with Possion prior. **GSiMT-Possion-T5**[5] is a variant of GSiMT-Poisson which takes the top 5 history paths during dynamic programming when decoding a new target word. It it similar to having a sparse 'attention' over the previous histories, which in turn highlights the simultaneous translation paths with higher confidence. **GSiMT-NT** is the vanilla neural transducer model without Poisson prior.

According to the experimental results in Table 1, the GSiMT-Poisson obtains a good balance between the translation quality and latency. More importantly, it achieves the best BLEU given different AL scores in the same range. Especially when the AL is very low, the GSiMT-Poisson model maintains its high performance on BLEU scores. Interestingly, the performance of GSiMT-Possion-T5 is very similar to the GSiMT-Poisson model that updates all the possible translation paths instead of the top 5. It shows that the model can be further

[5]More details can be found in Appendix A

Model	BLEU ↑	AL ↓	BLEU ↑	AL ↓	BLEU ↑	AL ↓	BLEU ↑	AL ↓
	$k=2$		$k=3$		$k=4$		$k=5$	
Wait-k (Ma et al., 2019)	22.64	3.95	22.96	4.73	23.60	5.56	24.48	6.41
	$\zeta=0$		$\zeta=1$		$\zeta=2$		$\zeta=3$	
GSiMT-Possion-T5	27.14	3.88	**27.88**	**4.36**	28.00	5.43	27.83	6.15
GSiMT-Poisson	**27.20**	**4.01**	27.75	4.69	**28.05**	**5.51**	**28.20**	**6.37**

Table 3: Test-only performance on the SiMT. For both GSiMT-Possion-T5 and GSiMT-Poisson models, we apply different offset ζ to parameterise the prior distribution.



(a) A translation example of GSiMT with $\zeta = 0$

(b) A translation example of GSiMT with prior $\zeta = 3$

Figure 6: Visualisation of decoded sentences with different Poisson parameters. ↓ represents the state was sampled with the *read* action, while → represents the *translate* action. The decoding is carried out by the pretrained GSiMT-Poisson ($\zeta = 8$) and apply $\zeta = 0$, $\zeta = 3$ to generate the decoded sentences in the Test-only setup.

optimised in terms of efficiency without much loss on the performance. As expected, GSiMT-NT is able to achieve high performance on BLEU scores (close to the upper bound BLEU score obtained by the consecutive NMT) but suboptimal AL, because it is able to *read* as many source words as possible. Figure 5 further compares the overall performance on BLEU and AL on the test dataset.

Table 2 further demonstrates the good performance of the proposed generative models on multimodal SiMT. For all three language pairs, the GSiMT-Poisson model maintains the best performance. More importantly, by simply concatenating the visual features, the GSiMT models perform better than state-of-the-art multimodal SiMT model DEC-OD (Caglayan et al., 2020).

4.3 Generalisation Ability

To further verify the effectiveness of the soft boundary modelled by the Poisson distribution, we also test the performance for test-only setup. In this case, we first pretrain a GSiMT-Poisson and a GSiMT-Possion-T5 with $\zeta = 8$ as the base models. Then, we directly set up different free parameters ζ to dynamically adjust the translation latency during testing. The test-only model of Wait-k (Zheng et al., 2020) pretrain a consecutive NMT as the base model and apply the Wait-k policies during testing. Table 3 shows the results on the WMT15 dataset. Compared to Wait-k, both the GSiMT-Possion-T5 and GSiMT-Poisson have stronger generalisation ability in test-only setup. It demonstrates the great potential of adjusting the translation latency on-the-fly without much loss on translation quality given a pretrained GSiMT model.

Figure 6 shows decoded sentences under different set of parameters. As we can see, even in the test-only setup, the generative model can effectively adjust the translation latency to decode the target sentences. Interestingly, the translation quality is not affected much when pursuing lower latency, and with less restrictive latency ($\zeta = 3$ compared to $\zeta = 0$), the generative model is able re-arrange the sub-sequences and produce the word order that is more natural in the target language.

5 Conclusions

This paper proposes a generative framework for simultaneous MT, which we demonstrated achieves the best translation quality and latency to date on common datasets. The introduction of Poisson prior over the buffer size fills in the gap between simultaneous MT and structural sequence-to-sequence learning. More importantly, the overall algorithm is simple and easy to implement, which grants the ability to be massively applied for various real-world tasks. It has the potential to become the standard framework for SiMT and we will release the code to the public for future research.

References

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Philip Arthur, Trevor Cohn, and Gholamreza Haffari. 2021. Learning coupled policies for simultaneous machine translation using imitation learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2709–2719, Online. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Ozan Caglayan, Julia Ive, Veneta Haralampieva, Pranava Madhyastha, Loïc Barrault, and Lucia Specia. 2020. Simultaneous machine translation with visual context. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2350–2361, Online. Association for Computational Linguistics.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74, Berlin, Germany. Association for Computational Linguistics.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas. Association for Computational Linguistics.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 478–485, Barcelona, Spain.

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. In *Proc. Interspeech 2018*, pages 1293–1297.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning. In *ICML 2016 Workshop on Abstraction in Reinforcement Learning*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden Markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71, Austin, TX. Association for Computational Linguistics.

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

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2016. The neural noisy channel. *arXiv preprint arXiv:1611.02554*.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2847–2853, Online. Association for Computational Linguistics.

A Top-k Strategy

Alternative to the full paths dynamic programming computation, each step of the generative simultaneous machine translation can also be formulated as sum of top-k sub-sequence generation probabilities. The original position distribution Equation 3 can be reformed as:

$$\begin{aligned} &p(z_j = i, Y_{:j-1}|X_{:i}) \\ &= \max_{|K|=k} \sum_{i' \in K} p(z_j = i|z_{j-1} = i', X_{:i}, Y_{:j-1}) \cdot \\ &\qquad p(Y_{:j-1}|X_{:i'}) \end{aligned} \tag{8}$$

where K is the set of position indices that equal or lower than i. This yields to a biased estimator for the log-likelihood estimation of the target sequences. The difference is that it acts as an inductive bias to have a sparse 'attention' over the previous sub-sequence generation probabilities. According to the experiments, the top-k strategy can achieve adequate performance as the full paths strategy.