Attention Forcing 2: Addressing Exposure Bias in Attention-based Neural Machine Translation

Anonymous ACL submission

Abstract

Attention-based autoregressive models have achieved state-of-the-art performance in various sequence-to-sequence tasks, including Text-To-Speech (TTS) and Neural Machine Translation (NMT), but can be difficult to train. The standard training approach, teacher forcing, guides a model with the reference back-history. During inference, the generated back-history must be used. This mismatch limits the evaluation performance. Attention forcing has been introduced to address the mismatch, guiding the model with the generated back-history and reference attention. While successful in tasks with continuous outputs like TTS, attention forcing faces additional challenges in tasks with discrete outputs like NMT. This paper introduces the two extensions of attention forcing to tackle these challenges. (1) Scheduled attention forcing automatically turns attention forcing on and off, which is essential for tasks with discrete outputs. (2) Parallel attention forcing makes training parallel, and is applicable to Transformer-based models. The experiments show that the proposed approaches improve the performance of models based on RNNs and Transformers.

1 Introduction

Attention-based models are good at connecting sequences of different length, and have achieved state-of-the-art performance in various sequence-to-sequence (seq2seq) tasks (Vaswani et al., 2017; Tay et al., 2020). Here the term performance refers to the overall quality of the output sequences, e.g. word error rate in Automatic Speech Recognition (ASR). On the other hand, these models can be difficult to train (Bengio et al., 2015). From a probabilistic perspective, seq2seq models estimate the probability of the output sequence conditioned on the input sequence. To achieve more accurate estimation, the models are often autoregressive (Chen et al., 2018). The standard training approach, teacher forcing, guides a model with reference back-history during training. This makes the model unlikely to recover from its mistakes during inference, where the model operates in free running mode, and the reference output is replaced by the generated output. This problem is referred to as exposure bias (Ranzato et al., 2016).

Many approaches have been introduced to address exposure bias, and will be described in section 2.2. Attention forcing is a simple and effective option (Dou et al., 2020). The idea is to guide the model with the generated output history and reference attention. While successful in TTS, attention forcing faces additional challenges when it comes to tasks with discrete outputs (Dou et al., 2019), and models such as Transformers (Vaswani et al., 2017). To tackle these challenges, this paper introduces scheduled attention forcing in section 3.1, and parallel attention forcing in section 3.2. The experiments in section 4 show that these approaches improve strong NMT models based on RNNs and Transformers.[1]

2 Attention-based sequence-to-sequence generation

Sequence-to-sequence generation can be defined as the task of mapping an input sequence $\boldsymbol{x}_{1:L}$ to an output sequence $\boldsymbol{y}_{1:T}$ (Bengio et al., 2015). From a probabilistic perspective, a model $\boldsymbol{\theta}$ estimates the distribution of $\boldsymbol{y}_{1:T}$ given $\boldsymbol{x}_{1:L}$, which can be factorized into token distributions: $p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L};\boldsymbol{\theta}) = \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L};\boldsymbol{\theta})$.

2.1 Encoder-attention-decoder architecture

Attention-based seq2seq models usually have the encoder-attention-decoder architecture (Lewis et al., 2020; Tay et al., 2020). Figure 1 shows the architecture. The distribution of a token is conditioned on the back-history $\boldsymbol{y}_{1:t-1}$, input sequence

[1] Links to the source code for the experiments will be available after the anonymous review.

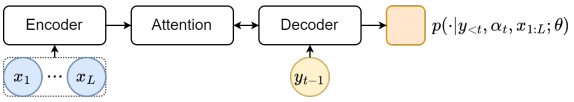

Figure 1: An attention-based encoder-decoder model, in teacher forcing mode, at decoding step t; a circle depicts a token, and a rounded square a distribution.

$\boldsymbol{x}_{1:L}$ and an attention map $\boldsymbol{\alpha}_{1:T}$:

$$p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L}; \boldsymbol{\theta}) \approx p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L}; \boldsymbol{\theta}) \approx p(\boldsymbol{y}_t|\boldsymbol{s}_t, \boldsymbol{c}_t; \boldsymbol{\theta}_y) \quad (1)$$

where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_y, \boldsymbol{\theta}_s, \boldsymbol{\theta}_\alpha, \boldsymbol{\theta}_h\}$; $\boldsymbol{s}_t$ is a state vector representing $\boldsymbol{y}_{1:t-1}$, and $\boldsymbol{c}_t$ is a context vector summarizing $\boldsymbol{x}_{1:L}$ for time step t. The discussions in this paper are agnostic to the form of attention. The following equations give a detailed example about how $\boldsymbol{\alpha}_t$, $\boldsymbol{s}_t$ and $\boldsymbol{c}_t$ can be computed:

$$\boldsymbol{h}_{1:L} = f(\boldsymbol{x}_{1:L}; \boldsymbol{\theta}_h) \quad (2)$$

$$\boldsymbol{s}_t = f(\boldsymbol{y}_{1:t-1}; \boldsymbol{\theta}_s) \quad (3)$$

$$\boldsymbol{\alpha}_t = f(\boldsymbol{s}_t, \boldsymbol{h}_{1:L}; \boldsymbol{\theta}_\alpha) \quad \boldsymbol{c}_t = \textstyle\sum_{l=1}^{L} \alpha_{t,l} \boldsymbol{h}_l \quad (4)$$

$$\hat{\boldsymbol{y}}_t \sim p(\cdot|\boldsymbol{s}_t, \boldsymbol{c}_t; \boldsymbol{\theta}_y) \quad (5)$$

The encoder maps $\boldsymbol{x}_{1:L}$ to $\boldsymbol{h}_{1:L}$, considering the entire input sequence; $\boldsymbol{s}_t$ summarizes $\boldsymbol{y}_{1:t-1}$, considering only the past. With $\boldsymbol{h}_{1:L}$ and $\boldsymbol{s}_t$, the attention mechanism computes $\boldsymbol{\alpha}_t$, and then $\boldsymbol{c}_t$. The decoder estimates a distribution based on $\boldsymbol{s}_t$ and $\boldsymbol{c}_t$, and optionally generates an output token $\hat{\boldsymbol{y}}_t$.

2.2 Inference and training

During inference, given an input $\boldsymbol{x}_{1:L}$, the output $\hat{\boldsymbol{y}}_{1:T}$ can be obtained from the distribution estimated by the model $\boldsymbol{\theta}$: $\hat{\boldsymbol{y}}_{1:T} = \operatorname{argmax}_{\boldsymbol{y}_{1:T}} p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L}; \boldsymbol{\theta})$. The exact search is often too expensive and approximated by greedy search for continuous output, or beam search for discrete output (Bengio et al., 2015).

Conceptually, the model is trained to learn the natural distribution, e.g. through minimizing the KL-divergence between the natural distribution $p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L})$ and the estimated distribution $p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L}; \boldsymbol{\theta})$. In practice, this can be approximated by minimizing the Negative Log-Likelihood (NLL) over some training data $\{\boldsymbol{y}_{1:T}^{(n)}, \boldsymbol{x}_{1:L}^{(n)}\}_1^N$, sampled from the natural distribution:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\boldsymbol{x}_{1:L}} \mathrm{KL}\big(p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L}) || p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L}; \boldsymbol{\theta})\big) \propto -\textstyle\sum_{n=1}^{N} \log p(\boldsymbol{y}_{1:T}^{(n)}|\boldsymbol{x}_{1:L}^{(n)}; \boldsymbol{\theta}) \quad (6)$$

$\mathcal{L}(\boldsymbol{\theta})$ denotes the loss; N denotes the size of the training dataset; n denotes the data index. To simplify the notation, n is omitted for the length of the sequences, although they also vary with n. In the following sections, the sum over the training set will also be omitted.

A key question here is how to compute the token distribution $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L}; \boldsymbol{\theta})$. For the most standard training approach, teacher forcing, the token distribution is computed with the reference output history $\boldsymbol{y}_{1:t-1}$ at each time step t. The loss can be written as:

$$\mathcal{L}_y(\boldsymbol{\theta}) = -\textstyle\sum_{t=1}^{T} \log p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L}; \boldsymbol{\theta}) \quad (7)$$

Despite its advantages such as parallel training (Dou, 2022), teacher forcing suffers from exposure bias: during training, the model is guided by the reference output history; during inference, however, the model runs in free running mode, where the generated output history is used. This mismatch leads to errors that can accumulate along the inference process (Ranzato et al., 2016).

There are mainly two lines of research addressing exposure bias. Scheduled sampling (Bengio et al., 2015) and professor forcing (Lamb et al., 2016) are prominent examples along the first line. These approaches guide a model with both the reference and the generated output history, and the goal is to learn the data distribution. To facilitate convergence, they often depend on a heuristic schedule or an auxiliary classifier, which can be difficult to design and tune (Guo et al., 2019). The second line is a series of sequence-level training approaches, leveraging reinforcement learning (Ranzato et al., 2016), minimum risk training (Shen et al., 2016) or generative adversarial training (Yu et al., 2017). Theses approaches guide a model with the generated output history. During training, the model operates in free running mode, and the goal is not to generate the reference output, but to optimize a sequence-level loss. However, many tasks do not have well established sequence-level objective metrics. Examples include voice conversion, machine translation and text summarization (Tay et al., 2020). Both lines of research require sequentially generating output sequences during training. In recent years, Transformers (Vaswani et al., 2017) have been widely used, and parallel training has been essential. To efficiently generate output sequences from Transformer-based models, an approximation scheme (Duckworth et al., 2019) has been proposed to parallelize scheduled sampling.

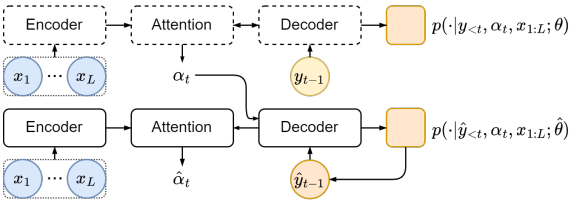

Figure 2: Attention forcing; the solid blocks are the attention forcing model $\hat{\boldsymbol{\theta}}$; the dashed blocks the teacher forcing model $\boldsymbol{\theta}$. During inference, $\hat{\boldsymbol{\theta}}$ runs in free running mode without $\boldsymbol{\theta}$.

3 Attention forcing 2

This section will revisit the general framework of attention forcing, and then analyze its challenges and introduce extensions to tackle them. The basic idea of attention forcing (Dou et al., 2020) is to train the model with the generated output and reference attention. The generated output helps addressing the exposure bias, and reference attention helps with convergence. Let $\boldsymbol{\theta}$ denote a standard attention-based model, trained with teacher forcing. Let $\hat{\boldsymbol{\theta}}$ denote a model with the same structure, but trained with attention forcing, and later used for inference. Figure 2 illustrates attention forcing. In attention forcing mode, equation 1 becomes:

$$\begin{aligned} p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}) &\approx p(\boldsymbol{y}_t|\hat{\boldsymbol{y}}_{1:t-1}, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}) \\ &\approx p(\boldsymbol{y}_t|\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{c}}_t; \hat{\boldsymbol{\theta}}_y) \end{aligned} \quad (8)$$

$\hat{\boldsymbol{s}}_t$ and $\hat{\boldsymbol{c}}_t$ denote the state vector and context vector generated by $\hat{\boldsymbol{\theta}}$. Details of attention forcing are in the following equations:

$$\boldsymbol{h}_{1:L} = f(\boldsymbol{x}_{1:L}; \boldsymbol{\theta}_h) \quad \hat{\boldsymbol{h}}_{1:L} = f(\boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}_h) \quad (9)$$

$$\boldsymbol{s}_t = f(\boldsymbol{y}_{1:t-1}; \boldsymbol{\theta}_s) \quad \hat{\boldsymbol{s}}_t = f(\hat{\boldsymbol{y}}_{1:t-1}; \hat{\boldsymbol{\theta}}_s) \quad (10)$$

$$\boldsymbol{\alpha}_t = f(\boldsymbol{s}_t, \boldsymbol{h}_{1:L}; \boldsymbol{\theta}_\alpha) \quad \hat{\boldsymbol{\alpha}}_t = f(\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{h}}_{1:L}; \hat{\boldsymbol{\theta}}_\alpha) \quad (11)$$

$$\hat{\boldsymbol{c}}_t = \sum_{l=1}^{L} \alpha_{t,l} \hat{\boldsymbol{h}}_l \quad (12)$$

$$\hat{\boldsymbol{y}}_t \sim p(\cdot|\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{c}}_t; \hat{\boldsymbol{\theta}}_y) \quad (13)$$

The right side of the equations 9 to 11, as well as equations 12 and 13, show how the attention forcing model $\hat{\boldsymbol{\theta}}$ operates. The decoder state $\hat{\boldsymbol{s}}_t$ is computed with $\hat{\boldsymbol{y}}_{1:t-1}$. While an alignment $\hat{\boldsymbol{\alpha}}_t$ is generated by $\hat{\boldsymbol{\theta}}$, it is not used by the decoder, because the context $\hat{\boldsymbol{c}}_t$ is computed with the reference alignment $\boldsymbol{\alpha}_t$. One option of obtaining $\boldsymbol{\alpha}_t$ is shown by the left side of equations 9 to 11: to generate $\boldsymbol{\alpha}_t$ from a teacher forcing model $\boldsymbol{\theta}$. $\boldsymbol{\theta}$ is trained in teacher forcing mode, and generates $\boldsymbol{\alpha}_t$ in the same mode. Although the reference output is used to compute the reference attention, it is not directly fed into the model, hence the model cannot rely too much on the back-history.

At the inference stage, the attention forcing model operates in free running mode, and equation 12 becomes $\hat{\boldsymbol{c}}_t = \sum_{l=1}^{L} \hat{\alpha}_{t,l} \hat{\boldsymbol{h}}_l$. The decoder is guided by $\hat{\boldsymbol{\alpha}}_t$, instead of $\boldsymbol{\alpha}_t$.

During training, there are two objectives: to infer the reference output and to imitate the reference alignment. This can be formulated as:

$$\mathcal{L}_{y,\alpha}(\hat{\boldsymbol{\theta}}) = \mathcal{L}_y(\hat{\boldsymbol{\theta}}) + \gamma \mathcal{L}_\alpha(\hat{\boldsymbol{\theta}}) \quad (14)$$

$$\mathcal{L}_y(\hat{\boldsymbol{\theta}}) = -\sum_{t=1}^{T} \log p(\boldsymbol{y}_t|\hat{\boldsymbol{y}}_{1:t-1}, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}})$$

$$\begin{aligned} \mathcal{L}_\alpha(\hat{\boldsymbol{\theta}}) &= \sum_{t=1}^{T} \mathrm{KL}(\boldsymbol{\alpha}_t || \hat{\boldsymbol{\alpha}}_t) \\ &= \sum_{t=1}^{T} \sum_{l=1}^{L} \alpha_{t,l} \log \frac{\alpha_{t,l}}{\hat{\alpha}_{t,l}} \end{aligned}$$

$\mathcal{L}_y$ and $\mathcal{L}_\alpha$ respectively denote the loss over the output and the attention; γ is a scaling factor. As an alignment corresponds to a categorical distribution, KL-divergence is a natural difference metric. By default, the two models are trained separately. $\boldsymbol{\theta}$ is trained in teacher forcing mode, and then fixed to generate the reference attention. $\hat{\boldsymbol{\theta}}$ is trained with the joint loss $\mathcal{L}_{y,\alpha}$.

3.1 Scheduled attention forcing

Motivation When applying attention forcing, it is important to consider the nature of the attention connecting the input and output. For some tasks, the attention is expected to be monotonic, and there is only one type of valid attention maps, where the important positions roughly form a diagonal line. Examples include ASR and TTS. For other tasks, there may be multiple valid modes of attention: the ordering of tokens can be changed while the output sequence remains correct. Examples include NMT and text summarization. If the model takes an ordering that is different from the reference output, the token-level losses will be misleading. Figure 3 illustrates the problem with an NMT example.

Furthermore, there might be other issues such as grave mistakes. For tasks where the output is continuous,[2] such as TTS and voice conversion, a small deviation from the reference output is usually not a serious problem. However, this is more serious for tasks where the output is discrete, such as

[2]The meaning of "continuous" comes in two folds. First, speech is continuous in time, although often sampled as a discrete sequence. For a speech sequence, the correlation between tokens is stronger than that in a text sequence. Second, a speech token follows a continuous distribution. A text token follows a discrete distribution.

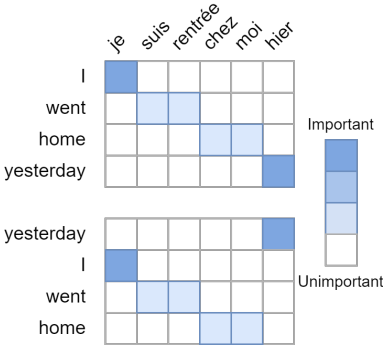

Figure 3: Alignments: up) $\boldsymbol{\alpha}_{1:T}$ between the input and the reference output; down) $\hat{\boldsymbol{\alpha}}_{1:T}$ between the input and the generated output. For the input *"je suis rentrée chez moi hier"*, the reference output is *"I went home yesterday"*. When using attention forcing, the model is guided by the generated back-history and outputs *"yesterday I went home"*. Here the alignment $\boldsymbol{\alpha}_{1:T}$ is not a sensible target for $\hat{\boldsymbol{\alpha}}_{1:T}$.

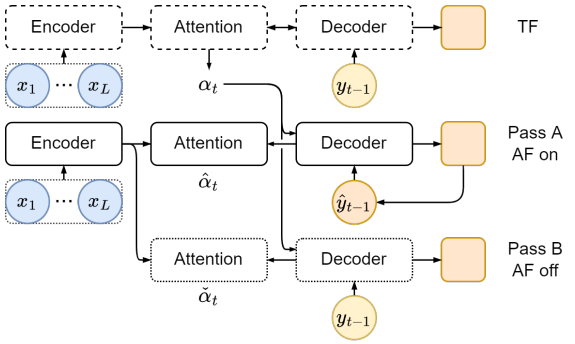

Figure 4: Scheduled attention forcing. Passes A and B share the same model parameters; only one of them will be used in back-propagation, depending on the data.

NMT and text summarization. During training, errors in the output history can be so serious that the token-level target is not appropriate, often due to misalignment between the generated output and the reference output. To illustrate the problem, suppose the reference output is *"thank you for listening"*, and the model predicts *"thanks"* at the first time step. In this case, the next output should not be *"you"*, and *"for"* would be a more sensible target.

Framework Scheduled attention forcing is proposed for applications where attention forcing, also referred to as "vanilla attention forcing", may result in an inappropriate loss. The basic idea is to automatically decide, for each input-output pair in the training data, whether vanilla attention forcing will be used. This is realized by tracking the alignment between the reference and the generated output. If they are relatively well-aligned, vanilla attention forcing will be used, otherwise a more stable training mode will be used.

Figure 4 illustrates scheduled attention forcing. For each input sequence, the attention forcing model $\hat{\boldsymbol{\theta}}$ takes two forward passes. Pass A is guided by the generated output history $\hat{\boldsymbol{y}}_{1:t-1}$, which is the same as vanilla attention forcing. Pass B is guided by the reference output history $\boldsymbol{y}_{1:t-1}$. The reference attention is always used, so the context vector $\hat{\boldsymbol{c}}_t$ is the same in both passes. If memory permits, the two forward passes can be completed in parallel, resulting in no extra time. This can be formulated as follows. For vectors produced in pass A, the notation has the hat ˆ accent; the equivalent for pass B is the check ˇ accent.

$$\boldsymbol{h}_{1:L} = f(\boldsymbol{x}_{1:L}; \boldsymbol{\theta}_h) \quad \hat{\boldsymbol{h}}_{1:L} = f(\boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}_h) \tag{15}$$

$$\boldsymbol{s}_t = f(\boldsymbol{y}_{1:t-1}; \boldsymbol{\theta}_s) \quad \begin{matrix} \hat{\boldsymbol{s}}_t = f(\hat{\boldsymbol{y}}_{1:t-1}; \hat{\boldsymbol{\theta}}_s) \\ \check{\boldsymbol{s}}_t = f(\boldsymbol{y}_{1:t-1}; \hat{\boldsymbol{\theta}}_s) \end{matrix} \tag{16}$$

$$\boldsymbol{\alpha}_t = f(\boldsymbol{s}_t, \boldsymbol{h}_{1:L}; \boldsymbol{\theta}_\alpha) \quad \begin{matrix} \hat{\boldsymbol{\alpha}}_t = f(\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{h}}_{1:L}; \hat{\boldsymbol{\theta}}_\alpha) \\ \check{\boldsymbol{\alpha}}_t = f(\check{\boldsymbol{s}}_t, \hat{\boldsymbol{h}}_{1:L}; \hat{\boldsymbol{\theta}}_\alpha) \end{matrix} \tag{17}$$

$$\hat{\boldsymbol{c}}_t = \textstyle\sum_{l=1}^{L} \alpha_{t,l} \hat{\boldsymbol{h}}_l \tag{18}$$

$$\begin{matrix} \hat{\boldsymbol{y}}_t \sim p(\cdot|\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{c}}_t; \hat{\boldsymbol{\theta}}_y) \\ \check{\boldsymbol{y}}_t \sim p(\cdot|\check{\boldsymbol{s}}_t, \hat{\boldsymbol{c}}_t; \hat{\boldsymbol{\theta}}_y) \end{matrix} \tag{19}$$

Next, the choice of training mode is made at the sequence level. If $\sum_{t=1}^{T} \mathrm{KL}(\boldsymbol{\alpha}_t || \hat{\boldsymbol{\alpha}}_t; \hat{\boldsymbol{\theta}}) < \lambda \sum_{t=1}^{T} \mathrm{KL}(\boldsymbol{\alpha}_t || \check{\boldsymbol{\alpha}}_t; \hat{\boldsymbol{\theta}})$, meaning that $\hat{\boldsymbol{y}}_{1:T}$ is well aligned with $\boldsymbol{y}_{1:T}$, pass A will be used in the back-propagation. The loss is the same as in vanilla attention forcing, shown in equation 14. Otherwise pass B will be used:

$$\begin{aligned} \mathcal{L}_{y,\alpha}(\hat{\boldsymbol{\theta}}) &= \textstyle\sum_{t=1}^{T} \log p(\boldsymbol{y}_t | \boldsymbol{x}_{1:L}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\alpha}_t; \hat{\boldsymbol{\theta}}) \\ &\quad + \gamma \textstyle\sum_{t=1}^{T} \mathrm{KL}(\boldsymbol{\alpha}_t || \check{\boldsymbol{\alpha}}_t; \hat{\boldsymbol{\theta}}) \end{aligned} \tag{20}$$

The KL attention loss is used to determine if the alignment is good enough between the reference output $\boldsymbol{y}_{1:T}$ and the generated output $\hat{\boldsymbol{y}}_{1:T}$. As both $\boldsymbol{\alpha}_t$ and $\check{\boldsymbol{\alpha}}_t$ are computed using $\boldsymbol{y}_{1:t-1}$, they are expected to be similar, yielding a relatively small $\mathrm{KL}(\boldsymbol{\alpha}_t || \check{\boldsymbol{\alpha}}_t; \hat{\boldsymbol{\theta}})$. In contrast, $\hat{\boldsymbol{\alpha}}_t$ is computed using $\hat{\boldsymbol{y}}_{1:t-1}$, and $\mathrm{KL}(\boldsymbol{\alpha}_t || \hat{\boldsymbol{\alpha}}_t; \hat{\boldsymbol{\theta}})$ is expected to be larger. λ is a hyper-parameter controlling how much out-of-alignment $\hat{\boldsymbol{y}}_{1:T}$ and $\boldsymbol{y}_{1:T}$ can be. If $\lambda \to +\infty$, scheduled attention forcing will be the same as vanilla attention forcing.

For each pair of training data, scheduled attention forcing makes a choice whether to guide the model with the reference output history or the generated output history. This approach is named

"scheduled attention forcing", because scheduled sampling also selectively uses the generated output history. For scheduled sampling, the selection is random. For scheduled attention forcing, the selection depends on the data.

3.2 Parallel attention forcing

Motivation Transformer-style models have achieved state-of-the-art performance in various tasks including NMT and TTS. For such models with a large number of parameters, parallel training is essential (Vaswani et al., 2017). When teacher forcing is used, there are no recurrent connections in the model, and training can be done in parallel across the length T of the output $\boldsymbol{y}_{1:T}$. This is more obvious when teacher forcing is rewritten as $\hat{\boldsymbol{y}}_t \sim p(\cdot|\boldsymbol{y}_{1:t-1}, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L}; \boldsymbol{\theta})$, where $\boldsymbol{\alpha}_t = f(\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L}; \boldsymbol{\theta})$. The reference output history $\boldsymbol{y}_{1:t-1}$ is available for any t, so $\hat{\boldsymbol{y}}_{1:T}$ can be computed in parallel.

Attention forcing can be rewritten in a similar fashion: $\hat{\boldsymbol{y}}_t \sim p(\cdot|\hat{\boldsymbol{y}}_{1:t-1}, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}})$, where $\hat{\boldsymbol{\alpha}}_t = f(\hat{\boldsymbol{y}}_{1:t-1}, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}})$. The model is guided with generated output history $\hat{\boldsymbol{y}}_{1:t-1}$. $\hat{\boldsymbol{y}}_{1:T}$ is not available beforehand, and is generated sequentially. So when applying attention forcing to Transformer-based models, training is no longer parallel.[3]

Framework For parallel attention forcing, the core idea is to approximate the sequential generation of $\hat{\boldsymbol{y}}_{1:T}$ with a parallelizable process. Here the output $\hat{\boldsymbol{y}}^K_{1:T}$ is generated iteratively in K forward passes. For each pass, the complete output history is available beforehand, so training can be run in parallel across time t, as illustrated by figure 5.

For the first pass, the output history is the reference $\boldsymbol{y}_{1:T}$. For the following passes, the output history is the output of the previous pass $\hat{\boldsymbol{y}}^{k-1}_{1:T}$.

$$\hat{\boldsymbol{y}}^0_{1:T} = \boldsymbol{y}_{1:T} \tag{21}$$

$$\hat{\boldsymbol{\alpha}}^k_t = f(\hat{\boldsymbol{y}}^{k-1}_{1:t-1}, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}) \tag{22}$$

$$\hat{\boldsymbol{y}}^k_t \begin{cases} = & \hat{\boldsymbol{y}}^{k-1}_t & \text{if } t < k \\ \sim & p(\cdot|\hat{\boldsymbol{y}}^{k-1}_{1:t-1}, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}) & \text{if } t \geq k \end{cases} \tag{23}$$

It can be proved that when $K = T$, $\hat{\boldsymbol{y}}^K_{1:T}$ is independent of the reference back-history, and is equivalent to an output sequentially generated (Duckworth et al., 2019). In appendix A, figure 8 illustrates how the iterative parallel generation approximates sequential generation. Empirically, K could be much smaller than T, while still addressing the exposure bias (Duckworth et al., 2019). So although parallel attention forcing requires more computation than vanilla attention forcing, it is more efficient thanks to parallel training.

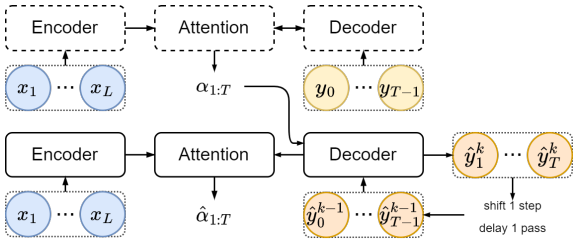

Figure 5: Parallel attention forcing; at pass k, $\hat{\boldsymbol{y}}^{k-1}_{1:T}$ is available, so $\hat{\boldsymbol{y}}^k_{1:T}$ can be computed in parallel.

Attention forcing has a regularizing effect. During training, the attention mechanism(s) of the attention forcing model is encouraged to mimic the teacher forcing model. Hence there is the risk of over regularization, in which case the attention forcing model converges to the teacher forcing model. When applying attention forcing to Transformer-based models, our default option is to force all the cross-attention connecting the encoder and the decoder, while leaving the self-attention alone. However, in a Transformer-based model, there are usually dozens of such attention maps. The exact number is equal to the number of decoder layers times the number of attention heads in each layer. In contrast, in a model based on RNN or CNN, there is only one attention map. Forcing all the attention heads tends to over regularize Transformer-based models. This problem can be addressed by forcing selected attention heads only. For the selected attention heads, the reference attention is given to the following layer, and an alignment loss is computed between the reference and the predicted attention. For the other attention heads, the predicted attention is given to the following layer as usual. In appendix A, figure 6 illustrates the idea of forcing selected attention heads. For Transformer-based models, different attention heads have different functions (Voita et al., 2019; Vig and Belinkov, 2019). For example, attention heads in the deepest layers of the model capture the most distant relationships (Vig and Belinkov, 2019). In this paper, the selection is mainly based on the layer.

[3] Transformer-based models have multiple cross attention mechanisms connecting the encoder and decoder. So when applied to these models, attention forcing involves a group of the reference attention $\boldsymbol{\alpha}^{(1:N,1:H)}_t$ and generated attention $\hat{\boldsymbol{\alpha}}^{(1:N,1:H)}_t$, where N is the number of decoder layers, and H the number of heads in each layer. These superscripts are omitted, to simplify the notation and to facilitate comparison with other forms of attention forcing.

3.3 Related work

Attention forcing follows the first line of approaches addressing exposure bias, descried in section 2.2. Similar to scheduled sampling and professor forcing, it is between teacher forcing and free running. An advantage of attention forcing is that it does not require a heuristic schedule or a discriminator, which can be difficult to tune. (Lamb et al., 2016) reported negative results on TTS. Variations of scheduled sampling were applied to NMT, resulting in both positive (Zhang et al., 2019) and negative (Duckworth et al., 2019) results.

Compared with sequence-level training approaches, attention forcing is more efficient, in the sense that it does not require generating multiple output sequences during training. Reference (Shen et al., 2016) applied Minimum Bayes Risk (MBR) training to NMT, and approximates the expectation of the risk by sampling and renormalizing the probability of the samples. References (Ranzato et al., 2016; Bahdanau et al., 2017) approximate the same loss with Monte Carlo sampling, and optimizes the loss using Reinforcement Learning (RL).

For sequence-level training, another general concern is the choice of the distance metric, i.e. the risk. Many tasks, including NMT and TTS, do not have a gold-standard objective metric. Empirically, models trained with one metric may not perform equally well when assessed using another metric (Ranzato et al., 2016). To tackle this issue, adversarial training (Yu et al., 2017; Wu et al., 2018) can be used: a discriminator learns a loss function, which is potentially better than standard metrics. The difficulty here is that the discriminator itself can be difficult to train (Zhang et al., 2018). While attention forcing does not directly optimize a sequence-level loss, it can indirectly reduce the loss by training the model to recover from errors. This is because sequence-level metrics are usually computed by comparing units of the sequences. Examples include word error rate for ASR, and BLEU (Papineni et al., 2002) and ROUGE (Lin and Hovy, 2003) for NMT. If models can recover from its errors, the sequence-level loss can be reduced.

It is challenging to apply attention forcing to models without an attention mechanism. However, the concept of attention forcing can be applied to such models, where it is essential to find something analogous to attention. For convolutional neural networks, for example, attention maps can be defined based on the activation or gradient (Zagoruyko and Komodakis, 2017).

Table 1: Data used in the experiments.

	Languages	# sentence pairs Training-valid-test
IWSLT'15	En→Fr	208K-1026-1305
	En→Vi	133K-1553-1268
WMT'16	En→De	4.5M-3000-3003

4 Experiments

Data There are two sources of data: IWSLT'15 (Cettolo et al., 2012, 2015) and WMT'16 (Bojar et al., 2016). Table 1 shows the data split. Detailed descriptions are in appendix B. The sentences in IWSLT are from TED talks. Two translation directions are investigated: English-to-French (EnFr) and English-to-Vietnamese (EnVi). The data preprocessing follows (Luong et al., 2015a). The sentences in WMT are from newspaper articles. Here English-to-German (EnDe) translation is investigated. The data preprocessing follows (Ott et al., 2018). For all the translation directions, the Moses tokenizer (Koehn et al., 2007) is adopted, and the translations are detokenized before evaluation. The checkpoints are selected based on the validation set, and the results are compared on the test set.

Performance Metrics The overall translation quality is measured by BLEU (Papineni et al., 2002). The average of 1-to-4 gram BLEU scores are computed and a 0.6 brevity penalty is applied. For IWSLT and WMT data, the BLEU score is computed using the Moses toolkit (Koehn et al., 2007) and the SacreBLEU toolkit (Post, 2018), respectively.

In NMT, there can be multiple valid output sequences for a single input sequence. Given that the overall translation quality is the same, it is desirable for an NMT model output to be diverse translations for the same input. This work measures the diversity of the candidate translations by pairwise BLEU (Shen et al., 2019) and entropy. For a translation model $\boldsymbol{\theta}$, we use sampling search M times with different random seeds, obtaining a group of translations $\{\hat{\boldsymbol{y}}^{(m)}\}_{m=1}^{M}$. $\hat{\boldsymbol{y}}^{(m)}$ denotes all the output sentences in the dev or test set. Then we compute the average BLEU between all pairs: $\frac{1}{M(M-1)}\sum_{n=1}^{M}\sum_{m=1}^{M}\texttt{BLEU}(\hat{\boldsymbol{y}}^{(n)},\hat{\boldsymbol{y}}^{(m)})_{n\neq m}$. In our experiments, M is set to 5. The more diverse the translations, the lower the pairwise BLEU. In addition to pairwise BLEU, we use greedy search

Table 2: BLEU of Teacher Forcing (TF), Attention Forcing (AF) and Scheduled Attention Forcing (SAF) with different values of λ; the higher λ is, the more likely the generated output history is used; the models are based on GNMT, and trained with data from IWSLT'15.

Task	Training	λ	BLEU ↑
EnFr	TF	-	30.70
	AF	-	22.93
	SAF	2.5	31.44
	SAF	3.0	**31.66**
	SAF	3.5	31.34
EnVi	TF	-	25.57
	AF	-	18.27
	SAF	2.5	26.02
	SAF	3.0	25.71
	SAF	3.5	**26.72**

and save the entropy e_t of the output token's distribution at each time step. Let $e_{1:T}$ cover all the model output steps, we compute the average value: $e = \frac{1}{T}\sum_{t=1}^{T} e_t$. Higher entropy means that the model is less certain, and thus more likely to produce diverse outputs. This process is deterministic, and is not repeated with different random seeds.

4.1 Scheduled attention forcing

Setup The experiments are conducted with EnFr and EnVi data in IWSLT'15. The model is based on GNMT (Wu et al., 2016). The details of the model and training are described in appendix B.2. By default, the baseline models are trained with Teacher Forcing (TF) for fewer than 60 epochs. Starting from the baseline, models are finetuned with Attention Forcing (AF) for fewer than 30 epochs. For AF, the scale γ of the attention loss is 10. The default inference approach is greedy search. When investigating diversity, sampling search is also adopted. The checkpoints are selected based on the validation BLEU. For all the training approaches, the effective number of epochs is smaller than the maximum, i.e. training goes on until convergence. The computational budget to train a model is 96 hours on a single Nvidia Tesla P100.

Results First, we compare TF and vanilla AF. The preliminary experiments show that when pretraining with TF is adopted, the BLEU of AF increases from 21.77 to 22.93 for EnFr, and from 13.92 to 18.27 for EnVi. However, it does not outperform TF, as shown by the first two rows in each section of table 2. This result is expected, considering the discrete and multi-modal nature of the NMT output space, analyzed in section 3.1.

Table 3: BLEU, Entropy and pairwise (P.) BLEU of TF and SAF; each approach is run 5 times, and the mean $\pm$ std is shown; the models are based on GNMT, and trained with data from IWSLT'15.

Task	Training	BLEU↑	Entropy↑	P. BLEU↓
EnFr	TF	31.10 $\pm$0.27	1.060 $\pm$0.047	27.43 $\pm$0.75
	SAF	**31.54** $\pm$0.14	1.034 $\pm$0.013	27.82 $\pm$0.67
EnVi	TF	25.86 $\pm$0.44	1.508 $\pm$0.012	22.11 $\pm$0.34
	SAF	**26.41** $\pm$0.33	**1.582** $\pm$0.017	**20.75** $\pm$0.29

Next, TF is compared with Scheduled Attention Forcing (SAF). The hyper parameter λ, introduced in section 3.1, controls the tendency to use generated outputs. As shown in table 2, with very limited tuning, SAF outperforms TF. The performance is robust in a certain range of λ. In the following experiments, λ is set to 3.0 for EnFr and 3.5 for EnVi. To reduce the randomness of the experiments, both TF and SAF are run $R = 5$ times with different random seeds. Let $\{\boldsymbol{\theta}^{(r)}\}_{r=1}^{R}$ denote the group of TF models, and $\{\hat{\boldsymbol{\theta}}^{(r)}\}_{r=1}^{R}$ the SAF models. For both groups, the BLEU's mean $\pm$ standard deviation is computed. Table 3 shows the results. In terms of mean BLEU, SAF yields a 0.44 gain for EnFr, and a 0.55 gain for EnVi.

To measure the diversity of the translations, the entropy and pairwise BLEU are computed for $\{\boldsymbol{\theta}^{(r)}\}_{r=1}^{R}$ and $\{\hat{\boldsymbol{\theta}}^{(r)}\}_{r=1}^{R}$, as shown in the last two columns of table 3. Focusing on the entropy column, SAF leads to higher entropy for EnVi, which indicates higher diversity. For EnFr, SAF and TF lead to similar levels of diversity, especially when the standard deviation is considered. We believe that the difference is due to the nature of the tasks. While English and French have similar syntax and lexicon, English and Vietnamese are more different. When trained with SAF, the EnVi model benefits more from using generated back-history, which is more likely to be different from the reference back-history. The pairwise BLEU shows similar trends. For EnVi, SAF leads to lower pairwise BLEU, i.e. higher diversity. For EnFr, the difference between SAF and TF is negligible. So in the following experiments, only pairwise BLEU will be reported.

4.2 Parallel attention forcing

Setup The experiments in this section are conducted with WMT'16 EnDe data. Compared with IWSLT, WMT is more suitable for Transformer models in terms of the amount of data. The Translation models have the same structure as the "big"

Transformer in (Vaswani et al., 2017), and the training follows (Ott et al., 2018). The details are in appendix B.3. The baseline models are trained with Teacher Forcing (TF). Starting from the baseline, other models are finetuned respectively with sequence-level Scheduled Sampling (SS) and Attention Forcing (AF). To keep the benefit of parallel training, AF and SS are approximated by their parallel version, as described in section 3.2 and reference (Duckworth et al., 2019). The number of iterations is two. For SS, the probability of using the reference output decreases linearly from 1 to 0.7; more aggressive schedules are found to degrade the performance. For AF, the scale γ of the attention loss is 1000. The default inference approach is beam search with beam size 4. The validation BLEU is monitored to select checkpoints and to stop training when no performance gain is observed after 10 epochs. The computational budget to train a model is 144 hours on a single Nvidia Tesla V100.

Results The first two sections of table 4 lists the preliminary results of TF, parallel SS and parallel AF. Here all the attention heads are constantly forced, regardless of the alignment between the reference and generated output history. Compared with TF, parallel SS yields lower BLEU as well as pairwise BLEU. It is difficult to conclude whether the decrease in pairwise BLEU results from higher diversity or lower translation quality. In its parallel version, AF performs similarly to TF. This is probably because the back-history is generated in TF mode. As analyzed in section 3.1, when applying AF to NMT, it is important to turn AF on and off based on the alignment between the reference and the generated outputs. Hence unless otherwise mentioned, a schedule is added to parallel AF in the following experiments. The last section of table 4 lists the performance of parallel AF, where the hyperparameter λ of the schedule is tuned. While the BLEU remains at the same level, the pairwise BLEU decreases when the percentage of AF decreases, signaling that AF regularizes the translation model to operate in a safe zone.

As analyzed in section 3.2, the encoder and decoder are connected by multiple attention mechanisms. It is likely that too much information is passed from the TF baseline to the AF model. To reduce this information, we only force selected attention heads. The first two decoder layers are selected, and the reason is discussed in section B.3 in the appendix. In each layer, the number of heads forced are 8, 12 or 16 out of 16. Table 5 lists the results. When only two layers are forced, the performance of parallel AF surpasses TF in both BLEU and pairwise BLEU. The best performance (first row) is achieved when 8 heads are forced in each layer. To reduce the randomness of hyperparameter tuning, we run another experiment where the other 8 heads are forced, and the result (second row) is comparable to the best performance.

Table 4: BLEU and Pairwise BLEU of Teacher Forcing (TF), Parallel Scheduled Sampling (PSS) and Parallel Attention Forcing (PAF); higher λ means higher tendency to use AF; the models Transformer-based, trained with WMT'16 EnDe, tested on newstest14; the bold numbers correspond to the λ for further experiments.

	λ	BLEU↑	Pairwise BLEU↓
TF	-	28.68	31.12
PSS	-	28.19	30.17
PAF	$+\infty$	28.74	31.90
PAF	1.1	28.75	31.60
PAF	1.2	**28.57**	**30.62**
PAF	1.3	28.48	31.89
PAF	1.4	28.56	31.99
PAF	1.5	28.47	32.06

Table 5: BLEU and Pairwise BLEU of TF, PSS and PAF, where only selected attention heads are forced; the models are based on Transformer, trained with WMT'16 EnDe, tested on newstest14.

	λ	Layers	Heads	BLEU↑	Pairwise BLEU↓
TF	-	-	-	28.68	31.12
PSS	-	-	-	28.19	30.17
PAF	1.2	1-2	1-8	**29.04**	**30.47**
PAF	1.2	1-2	9-16	**28.91**	**30.05**
PAF	1.2	1-2	1-12	28.86	30.87
PAF	1.2	1-2	1-16	28.64	30.79

5 Conclusion

This paper introduced two extensions to attention forcing, a training approach addressing exposure bias in attention-based seq2seq models. We recommend the basic form of attention forcing in tasks with continuous outputs like TTS. For tasks with discrete outputs like NMT, we recommend scheduled attention forcing. For models based on Transformers, it is essential to use parallel attention forcing, and to not force all the attention heads.

6 Broader impacts

The training approaches introduced in this work can be applied to a range of attention-based seq2seq models. NMT is used as a representative task, where the output is discrete. The baselines (Wu et al., 2016; Vaswani et al., 2017) in the experiments are prominent models based on RNNs or Transformers, which are widely used in various tasks (Tay et al., 2020; Dou, 2022). While this work focuses on autoregressive models, it can also benefit non-autoregressive models. For example, autoregressive models can act as teachers in teacher-student training of non-autoregressive models. They can also generate data for semi-supervised training.

The datasets (Cettolo et al., 2012, 2015; Bojar et al., 2016) used in the experiments are public dataset repeatedly used in the NMT community (Luong et al., 2015a; Ott et al., 2018). Sentences in IWSLT are from TED talks, and sentences in WMT are from newspapers. While we did not observe any sensitive information in the data, please contact us if any potential risks are spotted in the data, such as privacy issues. The tools (Koehn et al., 2007; Post, 2018) used to compute BLEU scores are also public. The use of the data and tools is consistent with their intended use. Detailed documentation of the data and code is available in their references.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. *5th International Conference on Learning Representations, ICLR*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of european association for machine translation*, pages 261–268.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The IWSLT 2015 evaluation campaign. In *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86.

Qingyun Dou. 2022. *Improving Attention-based Sequence-to-sequence Models*. Ph.D. thesis, University of Cambridge.

Qingyun Dou, Joshua Efiong, and Mark JF Gales. 2020. Attention forcing for speech synthesis. *Proc. Interspeech 2020*, pages 4014–4018.

Qingyun Dou, Yiting Lu, Joshua Efiong, and Mark JF Gales. 2019. Attention forcing for sequence-to-sequence model training. *arXiv preprint arXiv:1909.12289*.

Daniel Duckworth, Arvind Neelakantan, Ben Goodrich, Lukasz Kaiser, and Samy Bengio. 2019. Parallel scheduled sampling. *arXiv preprint arXiv:1906.04331*.

Haohan Guo, Frank K Soong, Lei He, and Lei Xie. 2019. A new GAN-based end-to-end TTS training algorithm. *Interspeech*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence

statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

Minh-Thang Luong, Christopher D Manning, et al. 2015a. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the international workshop on spoken language translation*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture models for diverse machine translation: Tricks of the trade. In *ICML*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient Transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a Transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

Lijun Wu, Yingce Xia, Fei Tian, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. Adversarial neural machine translation. In *Asian Conference on Machine Learning*, pages 534–549. PMLR.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2852–2858.

Sergey Zagoruyko and Nikos Komodakis. 2017. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *5th International Conference on Learning Representations, ICLR*.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Bidirectional generative adversarial networks for neural machine translation. In *Proceedings of the 22nd conference on computational natural language learning*, pages 190–199.

A Details of parallel attention forcing

Section 4.2 introduced parallel attention forcing, and explained the reasons to not force all the attention heads. Figure 6 illustrates the idea of forcing selected attention heads. Parallel attention forcing is applied to a Transformer with two decoder layers, but only forcing the attention heads in the second layer. Here the Transformer model is simplified; the detailed model structure is shown in figure 7.

A Transformer block (Vaswani et al., 2017), also referred to as a Transformer layer, is a combination of many basic modules. There are two types of

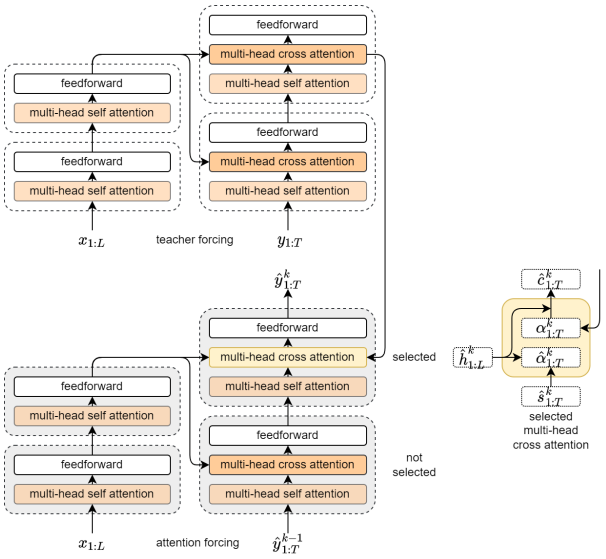

Figure 6: Illustration of parallel attention forcing, applied to a Transformer with two encoder layers and two decoder layers; the attention heads in the second decoder layer are forced.

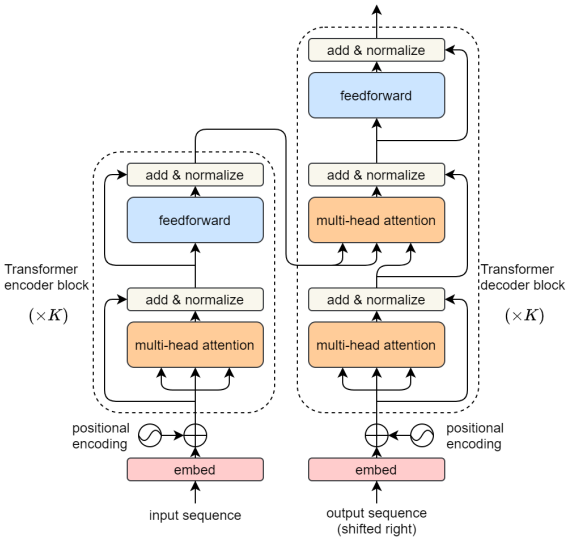

Figure 7: Illustration of Transformer blocks (Vaswani et al., 2017); the dashed rectangle in the left is a Transformer encoder block; the dashed rectangle in the right is a Transformer decoder block.

Transformer blocks: Transformer encoder blocks, which encode a sequence, and Transformer decoder blocks, which connect two sequences. Figure 7 illustrates both types of Transformer blocks, as well as how they are combined to form an encoder-decoder model.

Similar to scheduled attention forcing, multiple forward passes can be taken for each pair of training data in parallel attention forcing. The loss function is selected based on the alignment between the reference and generated output sequences. If $\sum_{t=1}^{T} \mathrm{KL}(\boldsymbol{\alpha}_t || \hat{\boldsymbol{\alpha}}_t^K; \hat{\boldsymbol{\theta}}) < \lambda \sum_{t=1}^{T} \mathrm{KL}(\boldsymbol{\alpha}_t || \hat{\boldsymbol{\alpha}}_t^1; \hat{\boldsymbol{\theta}})$, it will be assumed that $\hat{\boldsymbol{y}}_{1:T}^K$ is well aligned with $\boldsymbol{y}_{1:T}$, and the K-th forward pass will be used in the back-propagation:

$$\mathcal{L}_{y,\alpha}(\hat{\boldsymbol{\theta}}) = -\sum_{t=1}^{T} \log p(\boldsymbol{y}_t | \hat{\boldsymbol{y}}_{1:t-1}^K, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}) + \gamma \sum_{t=1}^{T} \mathrm{KL}(\boldsymbol{\alpha}_t || \hat{\boldsymbol{\alpha}}_t^K; \hat{\boldsymbol{\theta}}) \quad (24)$$

Otherwise the first pass will be used:

$$\mathcal{L}_{y,\alpha}(\hat{\boldsymbol{\theta}}) = -\sum_{t=1}^{T} \log p(\boldsymbol{y}_t | \hat{\boldsymbol{y}}_{1:t-1}^1, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}) + \gamma \sum_{t=1}^{T} \mathrm{KL}(\boldsymbol{\alpha}_t || \hat{\boldsymbol{\alpha}}_t^1; \hat{\boldsymbol{\theta}}) \quad (25)$$

Figure 8 illustrates how the iterative parallel generation approximates sequential generation. It can be proved that when $K = T$, $\hat{\boldsymbol{y}}_{1:T}^K$ is independent of the reference back-history, and is equivalent to an output sequentially generated (Duckworth et al., 2019). Empirically, K could be much smaller than T, while still addressing the exposure bias (Duckworth et al., 2019). So although parallel attention forcing requires more computation than vanilla attention forcing, it is more efficient thanks to parallel training.

B Details of experimental setup

B.1 Data

As shown in table 1, there are two sources of data: IWSLT'15 (Cettolo et al., 2012, 2015) and WMT'16 (Bojar et al., 2016). One reference output is provided for each input. The IWSLT datasets correspond to subtitle translation tasks, where the sentences are from TED talks. Two translation directions are investigated: English-to-French (EnFr) and English-to-Vietnamese (EnVi). These datasets are relatively small, and are used to train RNN-based models. For EnFr, the training set contains 208K sentence pairs. The validation set (tst2013) and test set (tst2014) respectively contain 1026 and 1305 sentence pairs. For EnVi, the training set contains 133K sentence pairs. The validation set (tst2012) and test set (tst2013) respectively contain 1553 and 1268 sentence pairs. The data preprocessing follows (Luong et al., 2015a). The vocabularies are at the word-level, i.e. the units are words. For

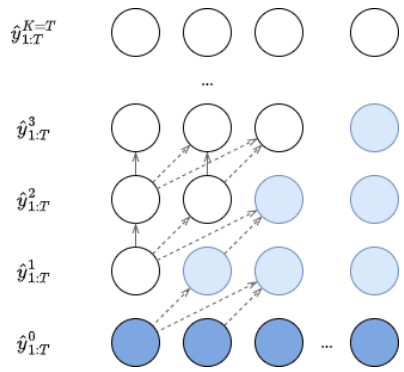

Figure 8: Illustration of iterative parallel generation; the dark blue circles are the reference tokens, and the rest are generated tokens; the light blue circles are influenced by the reference, and the white circles are not; the solid arrows represent copying, the dashed arrows represent dependency.

EnFr, both English and French vocabularies are limited to 50K. For EnVi, the vocabulary sizes are 17K and 7.7K for English and Vietnamese.

The WMT datasets correspond to news translation tasks, where the sentences are from newspaper articles. Here English-to-German (EnDe) translation is investigated. The dataset is considerably bigger, and is used to train Transformer-based models. The training set contains 4.5M sentence pairs. The validation set (newstest13) and test set (newstest14) respectively contain 3000 and 3003 sentence pairs. The data preprocessing follows reference (Ott et al., 2018). A joint source and target sub-word vocabulary is built using byte pair encoding. The vocabulary is 32K BPE tokens. For all the translation directions, the Moses tokenizer (Koehn et al., 2007) is adopted, and the translations are detokenized before evaluation. The checkpoints are selected based on the validation set, and the results are compared on the test set.

B.2 Scheduled attention forcing

B.2.1 Setup

The experiments in section 4.1 are conducted with EnFr and EnVi data in IWSLT'15. The model is based on GNMT (Wu et al., 2016). The differences are as follows. The model is simplified with a smaller number of LSTM layers due to the small scale of data: the encoder has 2 layers of bidirectional LSTM and the decoder has 4 layers of unidirectional LSTM; the attention mechanism is the general form of dot-product attention (Luong et al., 2015b); both English and Vietnamese word embeddings have 200 dimensions and are randomly initialized. Table 6 summarizes the hyperparameters.

Table 6: Hyperparameters of the RNN-based translation model (Wu et al., 2016).

Word embedding	200D
Encoder	2-layer bidirectional LSTM (200D)
Attention	General dot-product (Luong et al., 2015b)
Decoder	4-layer unibidirectional LSTM (200D)

The Adam optimiser is used with a learning rate of 0.002; $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$. The maximum gradient norm is set to be 1. If there is a finetuning phase, the learning rate will be halved. The batch size is 50. Dropout is used with a probability of 0.2. By default, the baseline models are trained with Teacher Forcing (TF) for fewer than 60 epochs. Starting from the baseline, models are finetuned with Attention Forcing (AF) for fewer than 30 epochs. For AF, the scale γ of the attention loss is 10. The default inference approach is greedy search. When investigating diversity, sampling search is also adopted, which replaces the argmax operation by sampling. The checkpoints are selected based on the validation BLEU. For all the training approaches, the effective number of epochs is smaller than the maximum, i.e. training goes on until convergence. The computational budget to train a model is 96 hours on a single Nvidia Tesla P100.

B.3 Parallel attention forcing

B.3.1 Setup

The experiments in section 4.2 are conducted with WMT'16 EnDe data. Compared with IWSLT, WMT is more suitable for Transformer models in terms of the amount of data. The Translation models have the same structure as the "big" Transformer in (Vaswani et al., 2017). Table 7 shows the hyperparameters. The models are optimized with Adam using $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e^{-8}$. Following reference (Ott et al., 2018), large batches are built to have a maximum of 3584 tokens. The learning rate increases linearly for 4,000 steps to $5e^{-4}$, after which it is decayed proportionally to the inverse square root of the number of steps. Label smoothing (Pereyra et al., 2017) is applied with 0.1 weight for the uniform prior distribution over the vocabulary. Dropout is applied with probability 0.3 after each attention or feedforward module. Half precision optimization techniques (Ott et al.,

Table 7: Hyperparameters of the Transformer-based translation model (Vaswani et al., 2017); "FC" stands for "fully connected".

Sub-word embedding	1024D
Encoder	(Multi-head self-attention → Feedforward) × 6
Decoder	(Multi-head self-attention → Multi-head cross attention → Feedforward) × 6
Multi-head attention	16 heads × (64D query / key / value) → 1024D output Scaled dot-product attention (Vaswani et al., 2017)
Feedforward	FC-4096-ReLU → FC-1024-Linear

Table 8: Ablation study on forcing selected attention heads; BLEU and Pairwise BLEU of Teacher Forcing (TF), Parallel Scheduled Sampling (PSS) and Parallel Attention Forcing (PAF) without a schedule; the models are based on Transformer, trained with WMT'16 EnDe, tested on newstest14.

	λ	Layers	Heads	BLEU↑	Pairwise BLEU↓
TF	-	-	-	28.68	31.12
PSS	-	-	-	28.19	30.17
PAF	$+\infty$	1-2	1-4	28.38	31.43
PAF	$+\infty$	1-2	1-8	28.53	31.28
PAF	$+\infty$	1-2	1-12	28.80	31.85
PAF	$+\infty$	1-2	1-16	28.74	31.31
PAF	$+\infty$	3-4	1-16	27.94	31.61
PAF	$+\infty$	5-6	1-16	27.46	32.29

2018), are adopted to speed up training.

The baseline models are trained with Teacher Forcing (TF). Starting from the baseline, other models are finetuned respectively with sequence-level Scheduled Sampling (SS) and Attention Forcing (AF). To keep the benefit of parallel training, AF and SS are approximated by their parallel version, as described in section 3.2 and reference (Duckworth et al., 2019). The number of iterations is two. For SS, the probability of using the reference output decreases linearly from 1 to 0.7; more aggressive schedules are found to degrade the performance. For AF, the scale γ of the attention loss is 1000. The default inference approach is beam search with beam size 4. The validation BLEU is monitored to select checkpoints and to stop training when no performance gain is observed after 10 epochs. The computational budget to train a model is 144 hours on a single Nvidia Tesla V100.

B.3.2 Ablation studies

Table 4 has shown that adding a schedule itself is not enough for parallel AF to surpass TF. Another series of experiments show that limiting the information passed from the TF baseline is also not enough. In other words, the two techniques must be combined. Table 8 shows the results of forcing selected heads, without using a schedule. As analyzed in section 3.2, different layers in a Transformer model perform different roles. The last three rows show that forcing layers 1 and 2 yields the best performance. The first four rows show that once the layers are selected, forcing more than four heads generally leads to better performance in BLEU and pairwise BLEU. This is the motivation behind the setup of the experiments described in section 4.2.