Guiding Attention for Self-Supervised Learning with Transformers

Ameet Deshpande **Karthik Narasimhan**
Department of Computer Science
Princeton University
{asd, karthikn}@cs.princeton.edu

Abstract

In this paper, we propose a simple and effective technique to allow for efficient self-supervised learning with bi-directional Transformers. Our approach is motivated by recent studies demonstrating that self-attention patterns in trained models contain a majority of non-linguistic regularities. We propose a computationally efficient auxiliary loss function to guide attention heads to conform to such patterns. Our method is agnostic to the actual pre-training objective and results in faster convergence of models as well as better performance on downstream tasks compared to the baselines, achieving state of the art results in low-resource settings. Surprisingly, we also find that linguistic properties of attention heads are not necessarily correlated with language modeling performance.[1]

1 Introduction

Recent advances in self-supervised pre-training (Radford et al., 2018; Devlin et al., 2018a; Liu et al., 2019) have resulted in impressive downstream performance on several NLP tasks (Wang et al., 2018, 2019). However, this has led to the development of enormous models, which often require days of training on non-commodity hardware (e.g. TPUs) (Kaplan et al., 2020). Furthermore, studies have shown that it is quite challenging to successfully train these large Transformer models (Vaswani et al., 2017), requiring complicated learning schemes and extensive hyperparameter tuning (Xiong et al., 2020; Raffel et al., 2019; Popel and Bojar, 2018).

Despite these expensive training regimes, recent studies have found that once trained, these bi-directional language models exhibit simple patterns of self-attention without much linguistic backing (Voita et al., 2019; Raganato and Tiedemann, 2018). For example, 40% of heads in a pre-trained BERT model (Devlin et al., 2018a) simply pay attention to delimiters added by the tokenizer (e.g. `[CLS]` or `[SEP]`) (Kovaleva et al., 2019). Since these attention patterns are independent of linguistic phenomena, a natural question arises: can Transformer models be guided towards such attention patterns without requiring extensive training?

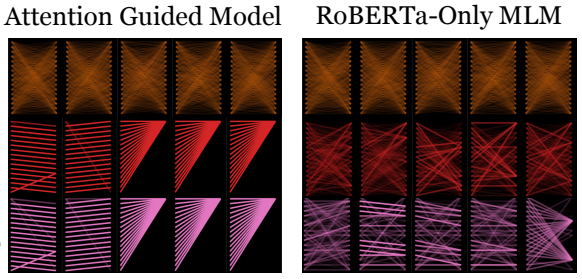


Figure 1: Attention patterns of our model (left) and the default RoBERTa model (right) after 0% (top), 1% (middle) and 100% (bottom) of pre-training. Inducing simple patterns (left) using an auxiliary loss leads to benefits in convergence speed, downstream performance, and robustness to hyperparameters.

In this paper, we propose an *attention guidance* (AG) mechanism for self-attention modules in Transformer architectures to enable faster, more efficient, and robust self-supervised learning. Our approach is simple and agnostic to the training objective. Specifically, we introduce an auxiliary loss function to guide the self-attention heads in each layer towards a set of pre-determined patterns (e.g. Figure 1 (Vig, 2019)). These patterns encourage the formation of both global (e.g. attend to `[CLS]`, `[SEP]` tokens) and local (e.g. attend to `[Next]`, `[Prev]` token) structures in the model.

Through several experiments, we show that our approach enables training large Transformer models considerably faster — for example, we can train a 16-layer RoBERTa model with SOTA performance on a low-resource domain in just two

[1]Code: https://github.com/ameet-1997/AttentionGuidance

days using four GPUs, while excluding our loss leads to slow or no convergence. Our method also achieves competitive performance with BERT (Devlin et al., 2018a) on three English natural language understanding tasks, and outperforms the baseline *masked language modeling* (MLM) models on eleven out of twelve settings considered.

Further, we also show that our initialization is agnostic to the training objective by demonstrating gains on the *replaced token detection* objective proposed by ELECTRA (Clark et al., 2020) and on machine translation with Transformers. Finally, we provide an analysis of the attention heads learned using our method. Surprisingly, contrary to recent studies (Clark et al., 2019; Lin et al., 2019), we find that it is possible to train models that perform well on language modeling without learning a single attention head that models coreferences.

To summarize, our main contributions are:

- We propose a simple auxiliary loss for self-attention heads that enables large models to converge quickly on commodity hardware.
- We demonstrate the effectiveness of our auxiliary loss on different languages, model sizes, and training objectives.
- We provide evidence that the linguistic performance of individual attention heads is not a necessary condition for good language modeling (LM) or downstream task performance.

2 Related Work

Improving efficiency of LMs The high computational costs of BERT-style models have accelerated research on developing efficient contextual language models. Clark et al. (2020) used a GAN-like setup to predict if each word in the input sequence is corrupted by a generator (another pre-trained LM). They show that their method is more sample efficient than the standard MLM objective. Other studies have explicitly focused on making the self-attention modules more efficient. Reformer (Kitaev et al., 2020) and Sparse Transformer (Child et al., 2019) introduce locality-sensitive hashing and sparse factorizations to reduce the quadratic complexity of dot-product attention, while Longformer (Beltagy et al., 2020) uses local-windowed and task motivated global attention to scale the memory usage of self-attention modules linearly.

Analyzing Self-Attention Recent papers have analyzed the attention patterns in trained Transformer-based LMs. Some studies hypothesize that multiple attention heads capture linguistic phenomena like co-reference links and dependency arcs (Clark et al., 2019; Htut et al., 2019). However, other studies show that pruning those heads leads to minimal performance degradation on downstream tasks (Kovaleva et al., 2019; Michel et al., 2019). Others note that there are recurring patterns in attention distributions corresponding to different attention heads (hereon, heads), which are not language or task-dependent (Voita et al., 2019; Raganato and Tiedemann, 2018). While our study also questions the role of heads for language modeling and downstream performance, we focus on making modifications to the LM pre-training and not on analyzing published pre-trained models.

Constraining Self-Attention Qiu et al. (2019) enforce local constraints on the attention patterns to reduce computation and build deeper models with longer contexts. The studies that are perhaps most similar to ours explore fixed attention patterns for machine translation (You et al., 2020; Raganato et al., 2020). You et al. (2020) replace all attention heads in the encoder with hard-coded Gaussian distributions centered around the position of each token while observing a minimal reduction in BLEU scores. Raganato et al. (2020) substitute all but one head with fixed attention patterns in each encoder layer and note little performance degradation. Both these studies enforce hard constraints on the self-attention and try to match baselines in terms of speed and performance. Our approach is complementary – our attention guidance loss is a form of soft regularization and outperforms baseline models both in terms of convergence speed and quantitative metrics.

3 Approach

3.1 Prelude: The surprising effectiveness of non-linguistic attention

Several recent studies (Clark et al., 2019; Kovaleva et al., 2019) have demonstrated that Transformers trained with the masked language modeling (MLM) objective exhibit simple self-attention patterns (e.g., attending to delimiter tokens). These patterns (e.g. Figure 2) are consistent across models pre-trained on different languages, or fine-tuned on various downstream tasks (Kovaleva et al., 2019).

Pre-trained (PT) Model	SST-2	MRPC	QNLI
No PT (Kovaleva et al., 2019)	0.80	0.81/0.68	0.49
Chinese PT (Devlin, 2020)	0.86	0.86/0.81	0.83
French PT (Martin et al., 2019)	0.88	0.88/0.84	0.85
English PT (Devlin et al., 2018a)	0.93	0.89/0.84	0.91

Table 1: Models pre-trained even with French and Chinese data perform significantly better than no pre-training on *English* downstream tasks.

Since these patterns are not linguistically motivated, we hypothesize that pre-training a model serves the dual purpose of lending linguistic and non-linguistic structure. To test the impact of the latter, we finetune CamemBERT (Martin et al., 2019) (a model pre-trained on the French part of OSCAR corpus (Suárez et al., 2019)), and BERT-Base Chinese (Devlin, 2020) (a model pre-trained on Chinese Wikipedia articles), on three English downstream tasks (Socher et al., 2013; Rajpurkar et al., 2016; Dolan and Brockett, 2005). We also compare with a randomly initialized Transformer, which is finetuned on downstream tasks without any pre-training (Kovaleva et al., 2019).

Surprisingly, the results in Table 1 show that despite both models having mismatched tokens and being trained on languages with linguistic constructs that are different from those of English, their performance is significantly better than a model with no pre-training. This corroborates the idea that the non-linguistic structure in attention heads is beneficial for learning, and inducing it explicitly may lead to faster training and better performance.

3.2 Our method: Attention guidance for Transformers

We first formally define the masked language modeling (MLM) setup with Transformers (Vaswani et al., 2017) and then describe our attention guidance mechanism.

MLM with Transformers Transformers used for sequence-to-sequence prediction tasks are trained on a dataset $\mathcal{D}$ of pairs of sequences $\mathbf{x}$ and corresponding labels $\mathbf{y}$. In the case of masked language modeling (MLM), the input sequence $x_1, x_2, \ldots, x_n$ of length n consists of individual tokens and the output labels $y_1, y_2, \ldots, y_n$ are the same as the input sequence, i.e., $y_i = x_i$. A fraction k of the input tokens, chosen randomly, are *masked*, i.e., replaced with a <MASK> token. Assume that these masked indices are collected together in a set $\mathcal{C}$. The MLM objective then is a cross-entropy loss on the predictions y'_j made by the model at the masked locations $j \in \mathcal{C}$, and is used to optimize all the parameters of the model, θ by minimizing:

$$\mathcal{L}_{MLM}(\mathbf{x}, \mathbf{y}) = -\sum_{j \in \mathcal{C}} \log \mathbb{P}\left(y_j | \mathbf{x}; \theta\right)$$

The Transformer architecture for MLM consists of ℓ layers with h self-attention heads per layer. Let the input activations to layer k of this model be $\mathbf{s}_k$, with $|\mathbf{s}_k| = n$. Naturally, $\mathbf{s}_1 = \mathbf{s} = \mathbf{x}$. For every position $p \in [1, n]$ in its output, each attention head in layer k induces a probability distribution over all positions in the input $\mathbf{s}_k$. Let a single head's attention activations (as described in Equation 1 of (Vaswani et al., 2017)), which is a function of $\mathbf{s}$, be denoted by the following:

$$\mathbf{H}(\mathbf{s}) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) \in \mathbb{R}^{n \times n}, \quad (1)$$

where Q and K are the query and key matrices respectively, and d_k is the dimensionality of the queries or keys. Further, let $\mathbf{H}(\mathbf{s})[p, q]$ (a scalar) denote the attention that token p in the head's output layer pays to token q in the head's input layer. We drop the dependence on $\mathbf{s}$ in the following sections for notational convenience.

Guiding attention heads To guide an attention head, we impose a mean squared error (MSE) loss on $\mathbf{H}$ using a pre-defined pattern $\mathbf{P}(\mathbf{s}) \equiv \mathbf{P} \in \mathbb{R}^{n \times n}$, where $||\cdot||_\text{F}$ is the Frobenius norm:

$$\mathcal{L}_{attn}\left(\mathbf{H}, \mathbf{P}\right) = ||\mathbf{H} - \mathbf{P}||_\text{F}^2 \quad (2)$$

Specifically, we consider two types of patterns:

- **Global attention patterns** that focus their attention on specific global positions like the first token of the sequence (`[First]`), punctuations like the *period* token (`[Period]`), or on delimiters like `[CLS]` or `[SEP]` added by the tokenizer (`[Delim]`). As an example,
$$\mathbf{P}_{\texttt{[First]}}[p, q] = \begin{cases} 1 & \text{if } q = 1 \\ 0 & \text{otherwise} \end{cases}$$
- **Local attention patterns** that focus *either* on the next or the previous token (e.g. `[Next]`, `[Prev]`). As an example,
$$\mathbf{P}_{\texttt{[Next]}}[p, q] = \begin{cases} 1 & \text{if } q = p + 1 \\ \frac{1}{n} & \text{if } p = n \\ 0 & \text{otherwise} \end{cases}$$

Figure 2 displays example $\mathbf{P}$ matrices for the different patterns we use. Note that the first and the last rows in `[Prev]` and `[Next]` patterns respectively are set to uniform distributions.

Overall loss function We apply the attention loss in Equation 2 to each head in each layer to obtain the overall **attention guidance (AG) loss**:

$$\mathcal{L}_{AG}(\mathbf{x}) = \sum_{k=1}^{\ell}\sum_{j=1}^{h} \mathcal{L}_{attn}\left(\mathbf{H}_{kj}, \mathbf{P}_{kj}\right) \times \mathbb{1}(k,j), \tag{3}$$

where $\mathbb{1}(k,j)$ denotes an indicator function which is 1 only if the jth head in layer k is being guided.

In general, this loss allows for arbitrary choices of patterns for each $\mathbf{P}_{kj}$. However, to simplify matters in our experiments, we guide a particular head number to the same pattern across all layers, i.e., $\mathbf{P}_{\cdot j}$ is the same for all layers. We utilize the gradients from this loss to update all the parameters of the model (including the feedforward and input embedding layers). It is worth noting that this loss only depends on the input $\mathbf{x}$ and not on labels $\mathbf{y}$.

Finally, we combine our attention guidance (AG) loss with the MLM loss to get our overall optimization objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\left[\mathcal{L}_{MLM}(\mathbf{x},\mathbf{y}) + \alpha_t \cdot \mathcal{L}_{AG}(\mathbf{x})\right] \tag{4}$$

where α_t is a hyperparameter. In practice, we find that $\mathcal{L}_{AG}$ converges faster than $\mathcal{L}_{MLM}$, so we linearly decay α_t from an initial value α_0 to 0 as the training progresses (details in Section 4).

4 Experimental Setup

We demonstrate the effectiveness of our attention guidance loss through several empirical studies. Specifically, we 1) report convergence results on masked language modeling, 2) evaluate trained language models on downstream tasks, and 3) analyze the learned attention representations using probes. For 1) and 2) above, we perform experiments on both high-resource and low-resource settings.

4.1 Datasets

We use the following datasets spanning three different languages (details in Table 2):

1. *English*: To train language models, we use a 2.1 billion token corpus from English Wikipedia. We download and pre-process articles according to Shoeybi et al. (2019). For downstream evaluation, we choose three tasks: QQP[2], MNLI (Williams et al., 2017), and QNLI (Rajpurkar et al., 2016)

2. *Filipino*: We use a 36 million token corpus of Wikipedia text collected by Cruz and Cheng (2020) to train language models, and the accompanying binary sentiment classification task to evaluate downstream performance.

3. *Oromo*: Our smallest corpus contains 4.6 million tokens ((Strassel and Tracey, 2016)). We use the accompanying named entity tags for NER, which is our downstream task.

Lang.	LM training		Downstream task		
	Train	Valid	Task	# instances	Eval. Metric
English	2116M	1%	MNLI	393k	Accuracy
			QNLI	105k	Accuracy
			QQP	364k	F-1
Filipino	36M	10%	SC	10k	Accuracy
Oromo	4.6M	3%	NER	1k	F-1

Table 2: Dataset statistics for LM training and downstream tasks on English, Filipino, and Oromo. SC=Sentiment Classification. Filipino and Oromo are low-resource languages.

These cover a range of dataset sizes — from high-resource (English) to low-resource (Oromo).

4.2 Evaluation

Evaluation metrics for the different tasks:

1. *Language modeling*: We report the training and validation MLM losses. Even though our attention guided models are trained with an auxiliary loss, we report only the MLM loss for direct comparison with the corresponding baseline. We also report the average training loss to compare models' convergence rates.

2. *English downstream tasks*: Accuracy for MNLI and QNLI, and F-1 score for QQP.[3]

3. *Filipino Sentiment Classification*: Since the dataset for Filipino is balanced, we use binary classification accuracy.

4. *Oromo NER*: Following Wang et al. (2020), we perform 10-fold cross-validation and use the F-1 scores aggregated over 9 tag classes.

[2] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

[3] We report the test scores obtained by submitting to https://gluebenchmark.com (Wang et al., 2018)

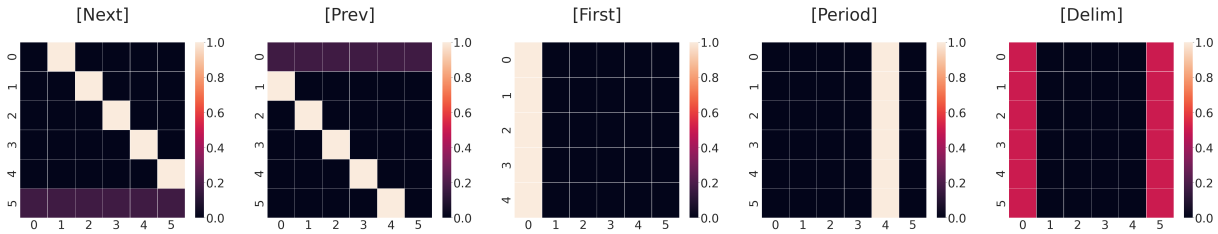


Figure 2: Example attention patterns used in our AG models for the sentence "<s> *Welcome to EMNLP* . </s>". Note that the first three patterns (`[Next]`, `[Prev]`, `[First]`) do not even depend on the input sentence.

4.3 Models and Training

To make comparisons across different settings easy, we choose RoBERTa (Liu et al., 2019) as the base architecture for all our experiments. We train variants with $8, 12$, and 16 layers following the configurations given in the original paper (Liu et al., 2019) on all 3 languages, which gives us a total of 9 settings. Since the current SOTA model for Filipino (Cruz and Cheng, 2019) is a BERT model, we train our Filipino models with both the MLM and next sentence prediction loss. Details of the model hyperparameters are provided in Appendix A.8. For each model, we compare its learning with and without our AG loss. We denote the attention guided models by RoBERTa-AG and the unmodified versions by RoBERTa-MLM. For notational convenience, RoBERTa-X-MLM and RoBERTa-X-AG represent RoBERTa models with X layers.

Comparison with state-of-the-art (SOTA) While we train all variants of our models with and without AG loss, and only these results are strictly comparable, we also compare with SOTA models for reference. These are E-MBERT (Wang et al., 2020), a recent extension of multilingual BERT (Devlin et al., 2018b) which performs well on low-resource languages, for Oromo, BERT (Cruz and Cheng, 2020) for Filipino, and RoBERTa$_{\text{BASE}}$ (Liu et al., 2019) for English[4].

4.4 Attention patterns

We consider the following patterns $\mathbf{P}$ (section 3) for guiding the self-attention heads:

1. `[Next]` attends to the next token.
2. `[Prev]` attends to the previous token.
3. `[First]` attends to the first token in the sequence.
4. `[Delim]` attends to delimiter tokens like `<s>`, `</s>`, `[CLS]` and `[SEP]` added by the model's tokenizer.
5. `[Period]` attends to the period ('.') token.

Only the `[Delim]` and `[Period]` patterns depend on the input because the corresponding tokens vary in position with the input. All other patterns are static and have a low memory footprint. Mathematical specifications of these patterns are provided in appendix A.2, and Figure 2 illustrates them.

4.5 Implementation details

Basic MLM models: We tune the learning rate from the set {1e-5, 5e-5, 1e-4}, the dropout in self-attention from the set $\{0.0, 0.1\}$, and the number of warmup steps from the set $\{0, 1000, 10000\}$.
AG models: For our AG models, we guide a fraction $\lambda \in \{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1\}$ of heads in each layer. We choose α_0 (equation 4) from the set $\{1, 10, 100\}$ such that the scales of the MLM loss and auxiliary loss are comparable at the beginning of training.

Best performing hyperparameters: **RoBERTa-MLM** is very sensitive to the learning rate and the number of warmup steps, and the best performing hyperparameters are reported in appendix A.8. On the other hand, we find that **RoBERTa-AG** is very robust and does not need much tuning. A learning rate of 1e-4, $\lambda = 0.5$, and 0 warmup steps work well for all the experiments. $\alpha = 10$ is used for our 12,16 layer models, and $\alpha = 100$ for smaller models. We fit the largest batch size possible for each model. We perform an ablation study and find that the `[Next]` and `[Prev]` patterns are most important, followed by `[First]`(section 5.3). Hence, one head each is modified with the `[Next]` and `[Prev]` patterns, and $(\lambda h - 2)$ heads are modified with `[First]`.

Compute Time and Hardware Unlike state-of-the-art models, we emphasize that our studies are

[4]MNLI-m and MNLI-mm scores are reported as the same in table 4 because they are not reported separately in (Liu et al., 2019). QQP scores reported are for RoBERTa$_{\text{Large}}$ because the F-1 score is not reported for RoBERTa$_{\text{Base}}$

performed on a smaller computational budget, both with respect to wall clock time and hardware. Our English models are trained for 10 epochs, with a cap of 4 days, on 8 NVIDIA Tesla P40 GPUs, and Filipino and Oromo models for 40 epochs with a cap of 2 days on 4 NVIDIA Tesla P40 GPUs. We emphasize that the RoBERTa-MLM and RoBERTa-AG variants in an experiment are trained on the same number of epochs. We also pre-train both RoBERTa-12-MLM and RoBERTa-12-AG for longer and on TPUs to show that the trends hold even when using specialized hardware and more compute time (appendix A.4).

5 Results

5.1 Language Modeling

Faster convergence Table 3 provides an overview of our results on language modeling. As seen from the average loss, we observe that the AG loss greatly helps improve the speed of convergence on all model sizes and domains. Figure 3 shows the train loss curves for two model sizes trained on English, where the losses for AG models almost instantaneously drop, whereas the MLM models have an extended period where the losses don't reduce. The gains are particularly notable for larger models like RoBERTa-12 and RoBERTa-16, where careful hyperparameter tuning is required for guaranteeing convergence if AG loss is not used. In contrast, using our auxiliary loss allows for fast convergence with standard out-of-the-box hyperparameters. For example, after just a day's training, the MLM loss for RoBERTa-16-AG has decreased from **11** to **2.5**, whereas RoBERTa-16-MLM's is still at **6.5**.

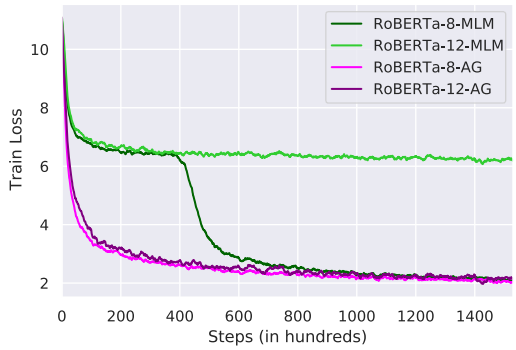


Figure 3: MLM Loss Curves for first 150k steps when training on English Wikipedia. Our AG models begin to converge instantly, while the MLM models have an extended plateau

Final loss values Not only do the AG models converge faster, but their final train and validation losses are also lower than their MLM counterparts on 8 out of 9 settings (table 3). This is facilitated by AG models' fast initial convergence coupled with robustness to hyperparameters, allowing us to use larger learning rates and no warmup period. On 5 of the 9 settings, namely 12,16 layer models on Filipino and Oromo, and the 16 layer model on English, only our AG model can converge. We also provide a hypothesis about the usefulness of AG loss in appendix A.3.

5.2 Downstream performance

We evaluate all the models' downstream performance to verify if better language modeling corresponds to better language understanding.

English Our AG models outperform their MLM counterparts on 11 out of the 12 settings (Table 4), with 7 comparisons being statistically significant ($p < .05$, paired t-test). We emphasize that the scores are not directly comparable to $\text{RoBERTa}_{\text{BASE}}$, which is trained on 10 times more data, up to 8 times more epochs, and on several GPUs. Having experimentally shown the usefulness of AG loss in optimizing the MLM objective, we believe that training our models on more data and compute is bound to match or outperform the MLM counterparts.

Filipino As shown in table 4, our AG models outperform the MLM variants on all model sizes. Additionally, our best performing model beats the current SOTA (Cruz and Cheng, 2019) by almost 1 point, even though the latter was trained on a TPU and for longer wall-clock time.

Oromo Our AG models continue to have an edge even in sparse data domains. Though Oromo has only 0.2% of the pre-training data when compared to English, which makes it prone to overfitting, it is interesting to note that larger models continue to outperform smaller ones on downstream tasks. Our models are competitive with E-MBERT (Wang et al., 2020), which is a BERT model leveraging resources from over 104 other languages. We hope that our competitive results on both Filipino and Oromo when using as little as 4 GPUs encourages more NLP research in low-resource languages.

Language	Loss	RoBERTa-8		RoBERTa-12		RoBERTa-16	
		MLM	AG	MLM	AG	MLM	AG
English	Train	1.75	**1.74**	1.86	**1.73**	6.40	**1.81**
	Validation	**2.41**	2.43	2.47	**2.29**	7.28	**2.46**
	Average	2.48	**2.09**	2.56	**2.07**	6.67	**2.24**
Filipino	Train	3.10	**0.74**	5.04	**0.66**	5.18	**0.63**
	Validation	3.22	**0.99**	5.06	**0.97**	5.05	**0.82**
	Average	4.95	**1.20**	5.24	**1.16**	5.28	**1.11**
Oromo	Train	3.44	**3.31**	6.24	**3.23**	6.51	**3.34**
	Validation	4.06	**3.93**	6.74	**3.88**	6.93	**3.92**
	Average	5.15	**4.52**	6.75	**4.52**	6.95	**4.62**

Table 3: Train, validation, and average train MLM loss on all three languages. Even after 4 days of training (section 4.5), our AG models outperform MLM models on all but one settings. Comparisons are columnwise.

Language	Task	RoBERTa-8		RoBERTa-12		RoBERTa-16		SOTA	
		MLM	AG	MLM	AG	MLM	AG	Model	Score
English	MNLI-m	78.8	**79.1**	78.9	**79.0**	69.7	**79.6**$^{+}$	RoBERTa$_{\text{BASE}}$ (Liu et al., 2019)	87.6*
	MNLI-mm	77.6	**77.7**	77.6	**78.9**$^{+}$	68.8	**78.7**$^{+}$		87.6*
	QNLI	84.3	**84.6**	86.1	**86.8**	72.0	**84.4**$^{+}$		92.8*
	QQP	**69.1**	68.3^{+}	68.4	**68.9**$^{+}$	58.2	**68.5**$^{+}$		74.3*
Filipono	Sentiment	74.1	**75.6**$^{\star+}$	74.1	**74.6**	74.1	**75.5**$^{+}$	BERT (Cruz and Cheng, 2020)	74.8
Oromo	NER	64.6	**66.7**$^{+}$	51.5	**67.9**$^{+}$	53.5	**67.2**$^{+}$	E-MBERT (Wang et al., 2020)	72.8*

Table 4: Evaluation on downstream tasks. Our AG models outperform their MLM counterparts on all but one settings (entries marked with '+' are significant with $p < .05$, paired t-test). Comparisons are column-wise. SOTA=state-of-the-art published numbers (marked *) with similar model types on each task. The SOTA models are trained on more compute and data and are not directly comparable to our models.

5.3 Ablation study with attention patterns

As mentioned in section 3.2, we introduce five different attention patterns for guiding our models using the AG loss. To select the best performing patterns, we use the leave-one-out strategy, in which we omit patterns and record the increases in loss (after 100,000 steps) when compared to a model with all patterns included. The patterns which cause a large increase in loss when omitted are naturally more important. The increases in loss are recorded in Table 5, which shows that `[Next,Prev]` patterns are most important, followed by `[First]` and `[Period]`, while `[Delim]` isn't very useful. Furthermore, unlike `[Period]`, the `[First]` pattern's guidance matrix $\mathbf{P}$ (section 3.2) is fixed, making it more computationally efficient to use. Hence, we guide one head each with `[Next,Prev]` patterns, and $(\lambda h - 2)$ heads with the `[First]` pattern.

Pattern(s) omitted	Change in loss
`[Next,Prev]`	$2.99 \rightarrow 7.74$
`[First]`	$2.99 \rightarrow 3.02$
`[Period]`	$2.99 \rightarrow 3.03$
`[Delim]`	$2.99 \rightarrow 2.99$

Table 5: Ablation study for choosing the best performing attention patterns to use for guidance (AG loss). The entry $x \rightarrow y$ means that the loss after omitting the respective pattern increased from x to y. We see that `[Next],[Prev]` are most important followed by `[First]` and `[Period]`.

5.4 Attention Guidance for ELECTRA

ELECTRA (Clark et al., 2020) is an efficient model which uses *replaced token detection* as the pretraining task. It comprises training a discriminator and a generator, in which the generator randomly changes $k\%$ of tokens in an input sequence to plausible alternatives, and the discriminator has

Loss	ELECTRA	ELECTRA-AG
Final	0.27	**0.10**
Average	0.32	**0.15**

Table 6: Training loss and average training loss over two epochs for ELECTRA discriminator. Adding AG loss reduces both the final and average training loss

to identify if a token was modified or not. The generator learns using the MLM objective, and the discriminator, which is used for downstream tasks, uses the logistic loss. We use an ELECTRA variant in which the generator is a unigram LM, and compare the performance when AG loss is added. The results after training ELECTRA-12 and ELECTRA-12-AG for 2 epochs on BooksCorpus (Zhu et al., 2015) are presented in Table 6. Like with RoBERTa, we report only the discriminator's logistic loss even though our model is trained on an auxiliary loss. The AG model shows gains in convergence without any ELECTRA specific hyperparameter tuning.

5.5 Attention Guidance for Machine Translation

Models We also experiment with adding our AG loss to Machine Translation (MT) models that use Transformers for both the encoder and decoder. We compare with the BASE Transformer (Vaswani et al., 2017) and a recently proposed hard-coded Gaussian model (You et al., 2020), which fixes all the attention heads in the encoder and decoder to pre-determined Gaussian distributions centered around nearby tokens. While the latter's attention patterns are similar to our *local* attention patterns, they are hard-coded and not an auxiliary loss. Following (You et al., 2020), the cross-attention in our MT model is not guided. Using a held-out set, we search for the best combination of AG patterns (Figure 2) for both the encoder and decoder. We find this to be one head each guided with the `[Next,Prev]` pattern in the encoder, and no heads being guided in the decoder. **Global** patterns (like attending to `[First]`) seem to be detrimental to performance in MT.

Results We perform experiments on IWSLT16 En-De (Cettolo et al., 2016) and WMT14 En-De datasets, and report train negative log-likelihood (NLL), validation NLL, average train NLL (to compare convergence speed), and the BLEU score on the test set. All models are trained for $100,000$ steps. Similar to LM pre-training, we observe that our model has the lowest train, validation, and average NLL for both the datasets, showing that guiding attention heads helps even with MT. Furthermore, the AG model's BLEU scores are comparable to the scores of BASE and hard-coded Gaussian. We note that our AG patterns are tailored for language-modeling, and MT models could benefit from a more extensive search over possible patterns.

Dataset	Loss/Metric	BASE	Hard-coded	AG
IWSLT	Train NLL	**1.18**	1.30	**1.18**
	Average NLL	1.88	1.92	**1.84**
	Validation NLL	2.25	2.26	**2.22**
	BLEU	**24.52**	24.42	24.42
WMT	Train NLL	1.77	1.94	**1.75**
	Average NLL	2.07	2.23	**2.05**
	Validation NLL	1.61	1.73	**1.60**
	BLEU	**26.24**	25.50	23.34

Table 7: Comparing the train, average train, and validation negative log-likelihood (NLL) loss, and also the BLEU scores for BASE (standard Transformer), Hard-Coded (You et al., 2020), and AG (our model). AG model has the lowest NLL losses and its BLEU scores are comparable to the other models.

5.6 Probing analysis

Motivated by recent studies (Clark et al., 2019; Lin et al., 2019; Manning et al., 2020) which posit that individual attention heads can encode linguistic information, we analyze attention patterns in the self-attention heads of our models. Specifically, we search for heads that can individually perform coreference resolution.

Method We use the probe described in (Clark et al., 2019), which evaluates attention heads on antecedent selection accuracy. A sentence (e.g. "***The CEO*** *led* ***her*** *company to success*") is input to the model, and each head is scored on its ability to identify antecedents, e.g. a score of 1 if the token '*her*' attends most to a token in '*The CEO*'. We aggregate the scores over all the coreferent mention-antecedent pairs in the dataset and report the accuracy of each model's best performing head. We also include the scores of a *randomly initialized* RoBERTa model for comparison. We leave further details to Clark et al. (2019).

Datasets Following Clark et al. (2019), we evaluate our models on the CoNLL-2012 dataset (Pradhan et al., 2012). We also evaluate on a synthetic

Model	MLM loss (valid)	CoNLL-2012				Synthetic
		ALL	NOMINAL	PRONOMINAL	PROPER	ALL
Rule-Based	-	0.66	0.48	0.72	0.73	-
Randomly Initialized	11.0	0.50	0.41	0.47	0.60	26.6
$\text{BERT}_{\text{BASE}}$ (Devlin et al., 2018a)	-	0.70	0.64	0.68	0.76	0.97
$\text{RoBERTa}_{\text{BASE}}$ (Liu et al., 2019)	-	0.74	0.71	0.74	0.76	0.99
RoBERTa-MLM	2.47	0.68	0.58	0.69	0.73	0.87
RoBERTa-AG ($\lambda = 1/2$)	2.29	0.28	0.21	0.32	0.29	0.84
RoBERTa-AG ($\lambda = 1$)	2.31	**0.21**	**0.13**	**0.21**	**0.28**	**0.00**

Table 8: Probing analysis to measure coreference resolution accuracies of the best performing attention head from each model on CoNLL-2012 (Clark et al., 2019) and synthetic (Lin et al., 2019) datasets. Interestingly, our AG models (last two rows) can be better at language modeling (lower MLM loss) without having a single head that is good at coreference. (**ALL**=overall mean scores, **Bold=lowest**)

dataset of 10000 samples from Lin et al. (2019) and follow their method of adding a distractor sentence (e.g. adding "*The **people** were happy*" after "***The CEO** led **her** company to success*") which serves to introduce spurious entities. We ensure that the antecedent is not the word directly before the coreferent mention so that a trivial baseline which always chooses the previous word gets a score of 0.

Discussion We discuss results reported in Table 8. We observe the same trends on both the CoNLL-2012 dataset and the synthetic dataset and discuss the former in detail. In line with Clark et al. (2019)'s observation, BERT and RoBERTa have heads which achieve the highest accuracies. Even though RoBERTa-MLM (section 4.3) is trained on significantly lesser compute and data, its performance is comparable to BERT and better than the Rule-based baseline. But interestingly, both RoBERTa-AG ($\lambda = 1/2$) and RoBERTa-AG ($\lambda = 1$), which have half and all their heads guided respectively, perform significantly worse than both the baseline and a randomly initialized (untrained) model. Surprisingly, this is true even though the validation loss for both RoBERTa-AG ($\lambda = 1/2$) and RoBERTa-AG ($\lambda = 1$) is lower (better) than RoBERTa-MLM's. The performance degradation in RoBERTa-AG models is because half/all the heads pay most of their attention to a predefined pattern, thus rendering them unable to pay attention to the antecedent. This provides evidence that language modeling performance is not necessarily correlated with the performance of individual heads on linguistic tasks, and that attention patterns of the heads are not necessarily directly interpretable. This observation is in line with a recent study (Brunner et al., 2020) that questions the interpretability of attention distributions.

The trends on the synthetic dataset (Table 8) are similar where BERT and RoBERTa have a head that achieves close to perfect accuracy, and RoBERTa-MLM has a head whose accuracy is significantly better than that of a randomly initialized model. However, RoBERTa-AG ($\lambda = 1$) performs poorly (an accuracy of 0) even though its validation MLM loss is lower (better) than RoBERTa-MLM's.

6 Conclusion

In this study, we introduce the simple yet effective Attention Guidance (AG) loss, which speeds up convergence and improves performance on various domains and model sizes. Adding this loss also makes Transformers robust to hyperparameters like learning rate, warmup steps, and dropout. Our experiments also show its usefulness in multiple pre-training objectives. The gains are particularly strong on larger models, enabling their usage in low-compute scenarios and low-resource domains. Our analysis of the relation of AG loss and MLM loss shows the usefulness of our method, and we hope that this paper can serve as a starting point for future works aiming to exploit and question self-attention in Transformers.

Acknowledgement

This research was partially funded by the Center for Statistics and Machine Learning at Princeton University through support from Microsoft. This work was also supported with Cloud TPUs from Google's TensorFlow Research Cloud (TFRC). We thank Austin Wang, Jens Tuyls, Zexuan Zhong, Michael Hu, and Danqi Chen for providing valuable comments and feedback.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Gino Brunner, Yang Liu, Damian Pascual Ortiz, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On identifiability in transformers.

Mauro Cettolo, Niehues Jan, Stüker Sebastian, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2016. The iwslt 2016 evaluation campaign. In *International Workshop on Spoken Language Translation*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jan Christian Blaise Cruz and Charibeth Cheng. 2019. Evaluating language model finetuning techniques for low-resource languages. *arXiv preprint arXiv:1907.00409*.

Jan Christian Blaise Cruz and Charibeth Cheng. 2020. Establishing baselines for text classification in low-resource languages. *arXiv preprint arXiv:2005.02068*.

Jacob Devlin. 2020. *BERT-Base Chinese*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. Multilingual bert - r.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R Bowman. 2019. Do attention heads in bert track syntactic dependencies? *arXiv preprint arXiv:1911.12246*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. *arXiv preprint arXiv:1908.08593*.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside bert's linguistic knowledge. *arXiv preprint arXiv:1906.01698*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2019. Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pages 14014–14024.

Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40.

Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. 2019. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. *arXiv preprint arXiv:2002.10260*.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Stephanie Strassel and Jennifer Tracey. 2016. Lorelei language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3273–3280.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. *Challenges in the Management of Large Corpora (CMLC-7) 2019*, page 9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

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. *arXiv preprint arXiv:1905.09418*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Zihan Wang, Stephen Mayhew, Dan Roth, et al. 2020. Extending multilingual bert to low-resource languages. *arXiv preprint arXiv:2004.13640*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. *arXiv preprint arXiv:2002.04745*.

Weiqiu You, Simeng Sun, and Mohit Iyyer. 2020. Hard-coded gaussian attention for neural machine translation. *arXiv preprint arXiv:2005.00742*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Appendices

A.1 Glossary

- AG: Attention Guidance
- AG Loss: Attention Guidance Loss
- AG Model: Attention Guided Model
- RoBERTa-X-AG: An X layer RoBERTa model trained with MLM and AG loss
- MLM: Masked Language Modeling
- RoBERTa-X-MLM: An X layer RoBERTa model trained with only MLM loss
- SOTA: State-of-the-art
- Head: Self-Attention Heads

A.2 Mathematical Specification of Patterns

Please refer to section 3 for definitions.

Let $CNT(\text{`token'})$ be the total number of occurences of `token` in an input $I$ of length $n$. $I[j]$ represents the $j^{th}$ token in $I$. Let `DELIM` represent the set of all delimiters added by the tokenizer.

$$\mathbf{P}_{\texttt{[Next]}}[p,q] = \begin{cases} 1 & \text{if } q = p+1 \\ \frac{1}{n} & \text{if } p = n \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{P}_{\texttt{[Prev]}}[p,q] = \begin{cases} 1 & \text{if } q = p-1 \\ \frac{1}{n} & \text{if } p = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{P}_{\texttt{[First]}}[p,q] = \begin{cases} 1 & \text{if } q = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{P}_{\texttt{[Period]}}[p,q] = \begin{cases} \frac{1}{CNT(\text{`.'})} & \text{if } I[q] = \text{`.'} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{P}_{\texttt{[Delim]}}[p,q] = \begin{cases} \frac{1}{CNT(\text{`DELIM'})} & \text{if } I[q] \in \texttt{DELIM} \\ 0 & \text{otherwise} \end{cases}$$

We add the patterns in figure 4 for reference.

A.3 Why is AG Loss Useful?

The AG loss converges within 0.2% of pre-training time. This fast convergence is because it is simple to attend to our patterns, which only require propagation of the positional embedding (for `[Next]`, `[Prev]`), or the non-contextual word embeddings in the input layer (for `[Delim]`, `[Period]`). In theory, this is particularly easy for Transformers because of the presence of residual connections (He et al., 2016). We observe from Figure 5 that as soon as AG loss converges, the MLM loss starts decreasing, and we hypothesize that the quick convergence of AG loss because of the reasons explained above is responsible for our method's advantages.

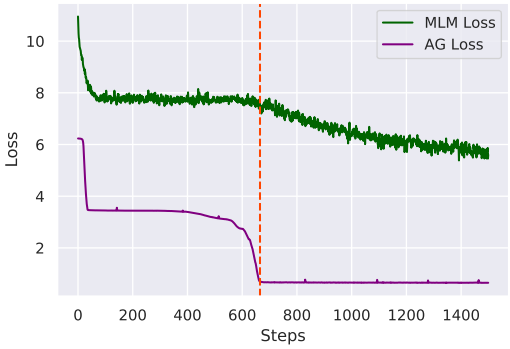


Figure 5: MLM loss ($\mathcal{L}_{MLM}$) and AG loss ($\mathcal{L}_{AG}$) for RoBERTa-12-AG. The MLM loss starts dropping as soon as AG loss converges.

A.4 Running English RoBERTa models on TPUs

To show that the trends mentioned in Table 4 hold even when specialized hardware is used, and models are run for longer, we run the RoBERTa-12-AG and RoBERTa-12-MLM models for 8 epochs (as opposed to 7) on TPUs. The results in Table 9 show that AG models to continue to have the advantage over MLM models.

A.5 Train Loss Curves for ELECTRA

Our method shows faster convergence with ELECTRA, as shown in Figure 6.

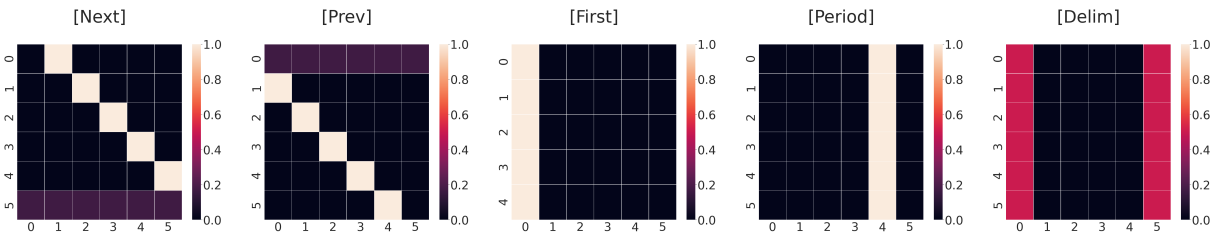


Figure 4: Example attention patterns for the sentence "<s> *Welcome to EMNLP* . </s>". Note that the first three patterns don't depend on the sentence, and can be considered fixed patterns, and the last two depend on the position of the period and delimiters respectively.

Task	RoBERTa-12-GPU		RoBERTa-12-TPU		SOTA	
	MLM	AG	MLM	AG	Model	Score
MNLI-m	78.9	**79.0**	80.3	**81.2**	$\text{BERT}_{\text{BASE}}$ (Devlin et al., 2018a)	84.6⋆
MNLI-mm	77.6	**78.9**	80.8	**81.4**		83.4⋆
QNLI	86.1	**86.8**	88.7	**89.0**		90.5⋆
QQP	68.4	**68.9**	69.3	**69.7**		71.2⋆

Table 9: RoBERTa-12-AG continues to outperform RoBERTa-MLM-AG even when trained on TPUs. This shows that using AG loss provides performance improvements even when using specialized hardware. We also report BERT-Base's scores for reference. Note BERT's scores are not directly comparable because it is trained for 40 epochs and our models are trained for 8.

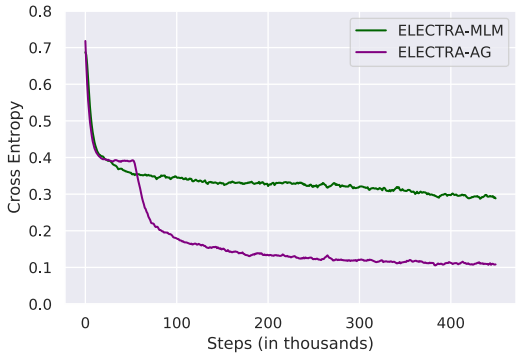


Figure 6: Loss Curves for ELECTRA. The MLM model has an extended plateau, whereas our AG model starts converging almost instantly.

A.6 Train Negative Log-Likelihood Curves for Machine Translation

We report the train loss curves for experiments on machine-translation (section 5.5) in Figure 7. Our AG model converges to the same loss as the BASE model, but the Hard-coded Gaussian model (You et al., 2020) converges to a slightly higher loss.

A.7 Model Configurations

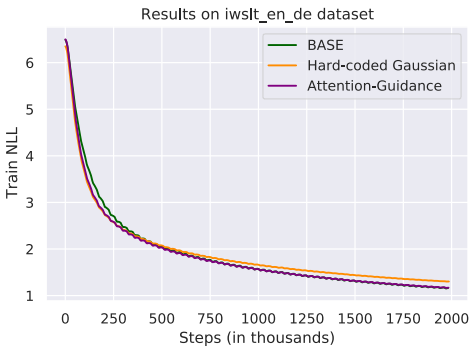
We follow Liu et al. (2019) for all design choices not mentioned in Table 10. The size of feed-forward layers is always $4 \times d_{model}$.

Model	Layers	Heads	Hidden Size
RoBERTa-8	8	12	768
RoBERTa-12	12	12	768
RoBERTa-16	16	16	768

Table 10: Model design choices. Hidden Size is d_{model} in Vaswani et al. (2017). Heads is the number of heads per layer.

A.8 Best performing hyperparameters

The best performing hyperparameters for each model are mentioned in Table 11. All design choices that are not mentioned (like dropout in the feed-forward layer) follow Liu et al. (2019).



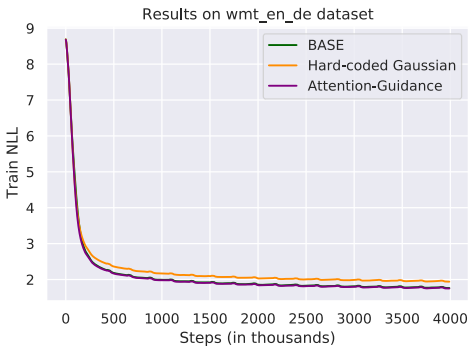


Figure 7: Train loss curves on IWSLT-en-de dataset (left) and WMT-en-de dataset (right). Our AG model converges to the same loss as the BASE model, but the Hard-coded Gaussian model (You et al., 2020) converges to a higher loss.

Language	Model	Learning Rate	Warmup Steps	α_0/λ	Batch Size
English	RoBERTa-8-MLM	1e-4	10000	-	120
	RoBERTa-12-MLM	5e-5	10000	-	84
	RoBERTa-16-MLM	1e-5	10000	-	48
	RoBERTa-8-AG	1e-4	0	100/0.5	120
	RoBERTa-12-AG	1e-4	0	10/0.5	84
	RoBERTa-16-AG	1e-4	0	10/0.5	48
Filipino	RoBERTa-8-MLM	1e-4	10000	-	40
	RoBERTa-12-MLM	5e-5	10000	-	28
	RoBERTa-16-MLM	1e-5	10000	-	16
	RoBERTa-8-AG	1e-4	0	100/0.5	40
	RoBERTa-12-AG	1e-4	0	10/0.5	28
	RoBERTa-16-AG	1e-4	0	10/0.5	16
Oromo	RoBERTa-8-MLM	1e-4	1000	-	40
	RoBERTa-12-MLM	5e-5	1000	-	40
	RoBERTa-16-MLM	1e-5	1000	-	32
	RoBERTa-8-AG	1e-4	0	100/0.5	40
	RoBERTa-12-AG	1e-4	0	10/0.5	40
	RoBERTa-16-AG	1e-4	0	10/0.5	32

Table 11: Best performing hyperparameters. α_0 is the relative weight placed on AG loss (equation 4) and λ is the fraction of heads being guided in each layer.