NOT ALL ATTENTION IS ALL YOU NEED

Anonymous authors
Paper under double-blind review

ABSTRACT

Dropout has shown an effective mediation to alleviate the over-fitting of neural models by forcedly blocking less helpful connections. However, common dropout has to be done crudely over all neural structures with the same dropout pattern once for all to dodge huge search space of tuning every individual structures. Thus in terms of meta-learning, we propose *AttendOut* which is capable of performing smart unit-specific dropout for attention models. The proposed smart dropout is nearly parameter-free and makes it possible to achieve even stronger performances with a faster tuning circle even though we evaluate our proposed method on state-of-the-art pre-trained language models. Eventually, we verify the universality of our approach on extensive downstream tasks in both pre-training and fine-tuning stages.

1 INTRODUCTION

Self-attention network (SAN) empowered models like Transformer (Vaswani et al., 2017) have achieved remarkable achievements in recent natural language processing, which have been broadly chosen as the basic architecture in a series successful pre-trained language models (PrLMs) such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), ALBERT (Lan et al., 2020), ELECTRA (Clark et al., 2020), DeBERTa (He et al., 2021) and GPT (Radford et al., 2018).

In this paper, we specialize in the attention dropout module which selectively shuts down unhelpful units in the attention matrix. This purpose serves the following two motivations. Numbers of existing studies (Wu et al., 2021; Zhang et al., 2020; Pham & Le, 2021) have shown that attention dropout can do a great job, including manually masking unimportant positions based on prior knowledge and searching an optimized pattern through neural architecture search. All these methods are effective but static that is one dropout pattern for one training. A smart substitute can be a self-adjusting one over time. On the other hand, PrLMs are always with large sizes, which leads to a huge search space for parameter sweep. For simplicity, researchers always choose to apply Vanilla Dropout (Srivastava et al., 2014), in which a dropout probability should be predefined for all hidden layers. However, it can be disaster if we meticulously tune one for each layer. Indeed, a parameter-free one that can be end-to-end trained serves the most convenience of reducing our experiment circles.

We propose *AttendOut* to bring self-attention empowered PrLMs into full play. Our work is based on Meta-Learning (Finn et al., 2017). The idea of meta-learning is to formulate the learning problem into two stages, meta-train and meta-test, where accumulating knowledge in the former and learning to control the training process in the latter. In our work, meta-train simply refers to training the model on training set. In meta-test, the "smart dropout maker" adjusts its dropout probabilities based on the feedback from validation set. For the concerned dropout maker, we present a novel LSTM-based modeling to hit the goal, which is efficiently optimized through a non-differentiable manner. The smart self-adjusting dropout is nearly parameter-free and makes it possible to achieve even stronger performances with a faster tuning circle. We verify the universality of our approach on extensive downstream tasks in both pre-training and fine-tuning stages.

2 RELATED WORK

2.1 DROPOUT

Dropout is a group of algorithms which alleviate over-fitting of deep neural networks by training with sub-networks (Srivastava et al., 2014; Wan et al., 2013; Klambauer et al., 2017). It is possible to

learn with Bayesian inference when interpreted as an approximate variational distribution of network weights (Gal & Ghahramani, 2016; Fan et al., 2021). It is also doable to design automatically (Pham & Le, 2021) with the help of Neural Architecture Search (Zoph & Le, 2017; Liu et al., 2019a). A parallel line of our work is to consider dropout as parameters jointly trained with the model (Ba & Frey, 2013; Boluki et al., 2020; Lee et al., 2020).

With self-attention network continuously stands out, attention-related dropout is being explored, like randomly skipping self-attention blocks (Fan et al., 2020), randomly removing attention heads (Michel et al., 2019; Voita et al., 2019; Zhou et al., 2020). Additionally, prior knowledge is shown highly effective for guiding attention dropout (Zhang et al., 2020; Wu et al., 2021).

2.2 META-LEARNING

While Meta-Learning, or Learning to Learn (Thrun & Pratt, 1998) has a long history with vast contributed literature, we could only discuss several relevant works here. Ravi & Larochelle (2017) designs an LSTM-based optimizer which learns the update rule for few shot learning. Finn et al. (2017) proposes Model-Agnostic Meta-Learning (MAML) to learn an optimized initialization ready to fast adapt to new tasks within few steps. More than that, the idea of learning at two stages sheds light to a variety of scenarios. Lee et al. (2020) learns to perturb samples of unseen tasks, Khetan & Karnin (2020) makes model compression by wiping the insignificant weights off during meta-test, Lv et al. (2020); Ke et al. (2021) use MAML-inspired pre-training to find a shared representation of both language modeling and downstream tasks, Kang et al. (2020) designs a mask generator for masked language model (Devlin et al., 2019).

3 PRELIMINARIES

In this section, we first review the details of self-attention network (SAN) (Vaswani et al., 2017) and then present the concerned attention dropout.

3.1 SELF-ATTENTION

Generally, a standard SAN block is composed of an attention layer and several feed-forward layers (actually there are residual connection, layer normalization, etc. as well). The input of it is a sentence or batch of sentences of length n, which is first embedded through an embedding layer. The embedded input E may go through three linear projections W_Q, W_K and W_V, and then obtain three matrices of self-attention, Q, K and V, which refer to query, key and value components respectively. Subsequently, a dot-product of Q and K is taken and then normalized using the Softmax function to obtain the attention matrix A. Then another dot-product of A and V follows. The mentioned calculation can be formalized as follow:

$$Attention(Q, K, V) = Softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \tag{1}$$

where $\sqrt{d_k}$ is a scaling factor. Finally, the attention layer ends up with an output projection W_O.

3.2 DROPOUT ON SELF-ATTENTION

Our proposed dropout strategy will be applied to the attention matrix of the concerned attention layer. We now define two specific dropouts onto Eq. 1, where both implementations are just as simple as in standard dropout via a mask matrix M.

Weights Dropout Weights dropout is applied to the attention matrix after the Softmax by default, which is formulated as:

$$Attention(Q, K, V) = \frac{1}{p}\left(Softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \odot M\right) \cdot V \tag{2}$$

where M is a binary matrix with elements in $\{0, 1\}$ and $\odot$ refers to element-wise multiplication. Especially for Vanilla Dropout, M is subject to a Bernoulli distribution of probability $1 - p$.

Scores Dropout Different from weights dropout, scores dropout is applied before Softmax, which is formulated as:

$$Attention(Q, K, V) = Softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}} + M\right) \cdot V \tag{3}$$

Since the outer Softmax, we conduct an addition instead of multiplication. The elements in M are set to 0 for kept units and $-inf$ for removed ones. Note that the Softmax takes a similar function as $1/p$, which balances the expectation of the network. In weights dropout, however, the excepted output can not be the same after scaled by $1/p$ if the output of Softmax is nonuniform distributed. In this paper, we apply scores dropout in our study, even if we empirically find similar performances on downstream tasks under weights and scores dropout.

4 METHODOLOGY

4.1 MODELING

As discussed above, dropout can be quantified as a mask matrix M. For dynamic dropout, where the distribution of M changes over time, we denote it as D_t. To model D_t, we shall establish the relationship between dropout and time step t. In this paper, our key idea is to model dropout as the update of an LSTM cell (Hochreiter & Schmidhuber, 1997):

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{4}$$

where $c_t = D_t$, f_t and i_t refer to the forget gate and input gate of each cell state respectively, $\tilde{c}_t$ refers to the cell input. Previously, Ravi & Larochelle (2017) utilized LSTM to model the learning rule. Here, we assume that dropout acts as another connection between two training moments.

Based on the update rule, we need to consider the parametric forms of f_t, i_t and $\tilde{c}_t$. We start with two gates f_t and i_t, which control the state transfer between two time steps. Basically, dropout is utilized to prevent the situation when the model fits well on training points but behaves worse on unseen ones. These two can be featured with training and evaluation losses. Hence, we let:

$$\begin{aligned} f_t &= \sigma(W_f \cdot [c_{t-1}, f_{t-1}, l_{t-1}] + b_f) \\ i_t &= \sigma(W_i \cdot [c_{t-1}, i_{t-1}, l_{t-1}] + b_i) \end{aligned} \tag{5}$$

where $[\cdot]$ is a one-dimensional feature vector and l_{t-1} refers to the previous training and evaluation losses. Both input and forget gates are dense layers with Sigmoid activation function σ.

Then we consider the cell input $\tilde{c}_t$. Intuitively, it is related to the current attention matrix a_t, which has already learned useful features from complex inputs of languages. However, it will be cumbersome to learn when sequence is long. Remember that a_t derives from the dot-product of query and key components, q_t and k_t. Hence, we factorize $\tilde{c}_t$ into two matrices:

$$\tilde{c}_t = (W_q \cdot q_t) \cdot (W_k \cdot k_t)/\sqrt{h} \tag{6}$$

where W_q and W_k project the two components into h dimensions respectively.

Additionally, we try non-temporal modeling based on MLP. Please refer to Appendix A.4 for details.

4.2 OPTIMIZATION

The challenging part of learning AttendOut is to find the update rule of cell states. By definition, however, dropout is only activated during training but is removed during evaluation. In other words, the feedback signal is supposed to be the evaluation performance instead of the training one. We denote the model parameters from step 0 to step K as $\theta_{0\sim K}$, and additional dropout module as θ_G. According to the update rule, we may easily derive the general formula of θ_K:

$$\theta_K = \theta_0 - \alpha \nabla_{\theta_0} \mathscr{L}(\theta_0, \theta_G) - \cdots - \alpha \nabla_{\theta_{K-1}} \mathscr{L}(\theta_{K-1}, \theta_G)$$

where α refers to the learning rate and $\mathscr{L}$ refers to the loss function. Now we take another gradient step of θ_G. The new θ_G comes to:

$$\theta_G - \beta \nabla_{\theta_K} \mathscr{L}(\theta_K) \nabla_{\theta_G} \theta_K$$

Note that it comes to a second-order optimization problem, which requires double back propagation. However, repetitive back propagation tends to be a memory killer when it comes to PrLM like BERT, which heavily slows down the training process. By the light of Reinforcement Learning (Williams, 1992), we choose to optimize through a non-differentiable manner.

4.2.1 SINGLE-UNIT CASE

Let us start with the simplest case, in which the attention matrix $[a_t]$ is in size of 1×1. The corresponding dropout distribution is denoted as d_t. Supposing there is a series of dropout actions:

$$d_{1:K} = \{d_1, d_2, d_3, \cdots, d_K\}$$

where K refers to the number of time steps, we train our model for K steps with $d_{1:K}$. Then we evaluate it on validation set and parameterize the evaluation performance as ϵ_t. We assume that the difference between two evaluations $\epsilon_t - \epsilon_{t-K}$ is the accumulated gain R within K steps. Thus, the optimization objective is to maximize:

$$J(\theta_G) = E_{P(a_{1:K};\theta_G)}[R]$$

where R can be discounted to each single step $\sum_{t=1}^{K} r_t$. Since R is non-differentiable, we apply Policy Gradient (Sutton et al., 1999):

$$\nabla_{\theta_G} J(\theta_G) = \sum_{t=1}^{K} E_{P(d_{1:K};\theta_G)}[\nabla_{\theta_G} \log P(d_t | d_{(t-1):1}; \theta_G) r_t]$$

The above equation could be approximated as:

$$\frac{1}{m} \sum_{s=1}^{m} \sum_{t=1}^{K} \nabla_{\theta_G} \log P(d_t^s | d_{(t-1):1}^s; \theta_G) r_t$$

where m refers to the number of samples. We omit s for conciseness in the following formulas.

4.2.2 MULTI-UNIT CASE

The question is each attention dropout comes to $n \times n$ attention units $[a_t^{ij}]_{n \times n}$ according to the size of attention matrix. Generally, $n \times n$ leads to a huge space when n is moderately large, which makes it unaffordable to calculate the joint probability. To facilitate calculation, we assume that each unit dropout d_t^{ij} stands for its own probability environment which is independent with each other. Hence, we are able to obtain the following probability likelihood:

$$\log P(D_t | D_{(t-1):1}; \theta_G) = \frac{1}{n^2} \sum_{i,j} \log P(d_t^{ij} | d_{(t-1):1}^{ij}; \theta_G)$$

Thus, the final gradient could be formalized as:

$$\nabla_{\theta_G} J(\theta_G) = \frac{1}{m} \frac{1}{n^2} \sum_{s=1}^{m} \sum_{t=1}^{K} \sum_{i,j} \nabla_{\theta_G} \log P(d_t^{ij} | d_{(t-1):1}^{ij}; \theta_G)(r_t - b) \tag{7}$$

where b is a baseline function of moving average (Williams, 1992) to reduce the variance. Note that we omit the subscript s in d_t^{ij}, which actually refers to the unit dropout for all in-batch samples.

4.3 TRAINING

Algorithm 1 summarizes the overall procedure of training models with AttendOut. We first initialize both two networks θ and θ_G. In meta-train, model is fed with K batches of samples and is trained under AttendOut. The dropout action is calculated using the Gumbel-Sigmoid estimator (Maddison et al., 2014), which is widely used in discrete RL setting (Jang et al., 2017; Lowe et al., 2017; Geng et al., 2020; Liu et al., 2021). Then in meta-test, model is evaluated without any dropout. Based on the evaluation scores, AttendOut is rewarded. When the score keeps going high, it is always rewarded positively. When the score goes down, it will be punished with a negative reward.

Algorithm 1 AttendOut

Input: Meta-train set $\mathscr{D}_{train}$, Meta-test set $\mathscr{D}_{test}$, Model θ, AttendOut θ_G, Meta step K

$\theta_G \leftarrow c_0$ ▷ Initialize dropout parameters
while not converged **do**
  Sample K batches from $\mathscr{D}_{train}$
  for $t = 1$ to K **do**
    $\mathscr{L}_t \leftarrow \mathscr{L}(\theta_t, \theta_G)$ ▷ Get loss of model with dropout
    $\theta_{t+1} = \theta_t - \alpha \nabla_{\theta_t} \mathscr{L}_t$ ▷ Update model parameters
  end for
  Sample from $\mathscr{D}_{test}$
  $R \leftarrow \epsilon_t - \epsilon_{t-K}$ ▷ Get reward of dropout
  Update θ_G via Eq. 7 ▷ Update dropout parameters
end while

4.3.1 INITIALIZATION

Our preliminary empirical study shows that normal initialization is not an effective choice for AttendOut, which destabilizes and hurts the performance with very small values like 0.5, 0.6 at the beginning of training. For simplicity, we choose the initial probabilities of all attention layers to be around 0.1 by setting c_0 to 0.9 as well as adjusting the input gate bias.

4.3.2 MEMORY USAGE AND META STEP

We notice that training PrLMs with AttendOut may sacrifice GPU memory. The main pressure comes from the cached states of AttendOut. We demonstrate the detail of memory usage in Appendix C.2. Generally, it restrains us from using very large K, where the meta-learning framework concentrates more on long-term benefits. However, small values like 4 or 8 already work well in our experiments, which merely results in slight memory consumption.

Now we return to an early question. Our approach can be applied to other layers with dropout in Transformer where the job is to redesign the training objective. However, applying unit-specific dropout to each layer on a hundred-million model is impossible on most current devices.

5 EXPERIMENTAL SETUP

We demonstrate the universal effectiveness of AttendOut on extensive tasks. Our implementations are based on PyTorch using *transformers*[1]. For further training details, please refer to Appendix B.

5.1 DATASET

Experiment datasets include: (1) **natural language understanding:** General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019), a collection of nine natural language understanding tasks (we exclude problematic WNLI); (2) **document classification:** IMDB (Maas et al., 2011), a long-sentence sentiment analysis dataset; (3) **named entity recognition:** CoNLL2003 (Sang & Meulder, 2003); (4) **part-of-speech tagging:** English Penn Treebank (PTB) (Marcus et al., 1993); (5) **multi-choice machine reading comprehension:** DREAM (Sun et al., 2019).

[1] https://github.com/huggingface/transformers/

Table 1: GLUE test results, where † refers to continual pre-training. Due to space limitation, we only demonstrate the overall confidence level as in Table 2.

Model	CoLA Mcc	SST-2 Acc	MRPC F1	QNLI Acc	MNLI-m/mm Acc	QQP Acc	RTE Acc	STS-B Spc	Avg
BERT	51.5	92.9	87.6	90.2	84.3/83.6	89.0	67.6	84.6	81.3
BERT†	51.1	93.1	87.5	90.2	84.2/83.4	89.0	66.9	84.8	81.1
+ *AttendOut*†	54.3	**93.9**	**88.3**	**90.6**	84.2/83.5	89.1	66.5	85.6	81.7
+ *AttendOut*	**56.8**	**93.9**	88.1	**90.6**	**84.6/83.7**	**89.4**	**70.6**	**85.8**	**82.6**
RoBERTa	57.0	95.4	90.4	92.9	87.3/86.1	89.4	71.8	89.2	84.4
+ *AttendOut*	**61.3**	**96.2**	**91.2**	**93.2**	**87.5/86.9**	**89.4**	**72.3**	**89.9**	**85.3**

Table 2: Fine-tuning results of all tasks.

Model	GLUE Avg	IMDB Acc	CoNLL03 F1	PTB F1	DREAM Acc
BERT	$81.3_{\pm 0.3}$	$92.0_{\pm 0.3}$	$90.0_{\pm 0.2}$	$95.6_{\pm 0.1}$	$63.4_{\pm 1.2}$
+ *AttendOut*	$\mathbf{82.6}_{\pm 0.2}$	$\mathbf{92.9}_{\pm 0.1}$	$\mathbf{90.6}_{\pm 0.2}$	$\mathbf{96.5}_{\pm 0.1}$	$\mathbf{64.5}_{\pm 1.0}$
RoBERTa	$84.6_{\pm 0.3}$	$93.4_{\pm 0.3}$	$90.9_{\pm 0.1}$	$96.3_{\pm 0.1}$	$68.8_{\pm 0.8}$
+ *AttendOut*	$\mathbf{85.4}_{\pm 0.3}$	$\mathbf{94.1}_{\pm 0.2}$	$\mathbf{91.5}_{\pm 0.2}$	$\mathbf{97.1}_{\pm 0.0}$	$\mathbf{69.3}_{\pm 0.5}$

5.2 Fine-tuning

We fine-tune models with AttendOut based on Huggingface (Wolf et al., 2020) checkpoints of BERT (Devlin et al., 2019) and its stronger variant RoBERTa (Liu et al., 2019b)[2]. For all downstream tasks, we average at least three random seeds and report the confidence intervals. That is why our reported results tend to be lower compared to those on the leaderboard. We also find highly unstable results on small datasets (CoLA, MRPC, RTE, STS-B, DREAM), so we run six random seeds for them.

5.3 Pre-training

We also verify the gain of AttendOut on pre-training. Specifically, we train MLM (Devlin et al., 2019) with AttendOut based on BERT checkpoint for another one epoch on the corpus of English Wikipedia and directly fine-tune on it without AttendOut on GLUE tasks. For fair enough comparison, we train another BERT model based on the same checkpoint in the same manner.

Note that for all reported baseline results, we select the best dropout probability in $\{0.1, 0.2\}$. For training with AttendOut, we substitute it for Vanilla Dropout on all attention layers and utilize development set for meta-test. Our final results are all evaluated on test sets. Beyond that, we only adjust the meta steps and keep all other parameters the same for strict fair comparison. For example, the parameters we use in RoBERTa are identical with what we use in training it with AttendOut.

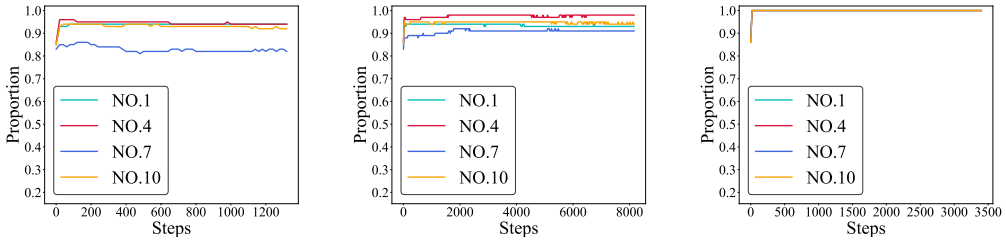
6 Results

6.1 Significance Analysis

Pictorially in Table 1, RoBERTa is strong enough as it outperforms BERT by a big margin, while AttendOut empowered one is much stronger on all tasks. Especially on small datasets, which are more likely to over-fit, AttendOut helps unfold remarkable performance gain[3] (**10.3%** over BERT on CoLA, **4.4%** over BERT on RTE). However, for larger ones which tend to be more stable, we still see considerable boost, (**1.1%** over BERT on SST-2, **0.9%** over RoBERTa on MNLI-mm).

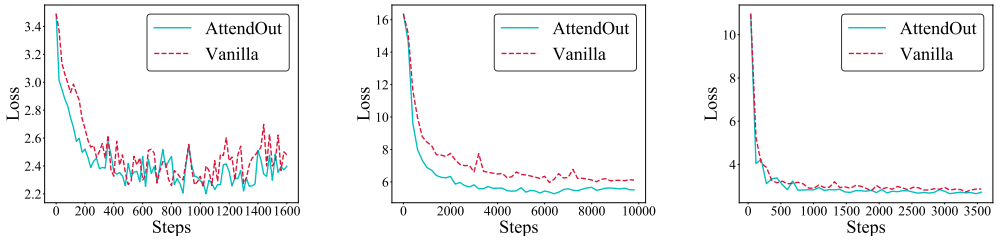
[2]We adopt `bert-base-uncased` and `roberta-base` checkpoints in our experiments.

[3]We calculate the relative growth rates here. Take RTE as an example: $(70.6 - 67.6)/67.6 = 4.4\%$.



(a) CoLA (b) QNLI (c) STS-B

Figure 1: Dropout patterns with AttendOut, where we show the proportion of the kept units (0 means completely dropout). For conciseness, we uniformly take four attention layers.



(a) CoLA (b) QNLI (c) STS-B

Figure 2: Total evaluation loss with AttendOut and Vanilla Dropout.

For continue pre-training, AttendOut is not that strong compared to task-specific tuning but still competitive (**5.4%** over BERT on CoLA, **0.8%** over BERT on MRPC). It is pointed out that dropout gives no gain in training MLM with Transformer-based models (Lan et al., 2020). However, we show the effectiveness of attention dropout under a more sophisticated pattern.

Similar results can be seen in Table 2, (**0.7%** over BERT for NER, **0.9%** over BERT for POS Tagging, **1.7%** over BERT for multi-choice MRC), which demonstrates the universal effectiveness of AttendOut. We also discuss the training variance in Appendix A.1.

6.2 Visual Analysis

Dropout Patterns Figure 1 depicts the patterns learned by AttendOut on three kinds of tasks (sentence classification, inference, semantic similarity). For further analysis, please refer to Appendix A.2. Intuitively, the learned patterns differ largely, which is fair since AttendOut is sample-dependent. For the first two tasks, the overall dropout proportions are around 0.1. Interestingly, the 7^{th} layer is the highest one in both cases. The difference is on semantic similarity, however, we can see very low dropout proportions, actually all 0 since the beginning of training (the four curves are overlapped). Another interesting fact is that we may see a common linear "warm-up" stage for dropout, nearly the first 100 training steps. Since then, all the layers tend to be stabilized.

Convergence We also track the evaluation loss during the training process. As in Figure 2, two curves are struggling at the beginning of training. As training continues, lower evaluation loss can be seen in AttendOut case on QNLI and STS-B. On CoLA, more fluctuate situation on both sides, we can still find the advantage in the later stage. The red line is up to rise, while the blue one is not.

7 Ablation Study

We conduct ablation experiments to demonstrate how AttendOut works. All the results here are on development sets, based on which we utilize early stop on the baseline models for fair comparison.

Effect of Regularization A specific experiment is made to show the impact of AttendOut as it supposed to a regularizer. First, we design a dropout scheduler, in which we follow Vanilla Dropout utilizing Bernoulli distribution to approximate the predicted trends as depicted in Figure 1. Specifically, we record all the instantaneous dropout probabilities learned by AttendOut and schedule Vanilla Dropout with them. Besides, the weight decay rate is set to 0 to exclude other contributed regularization factors. Experiments are conducted on the three small GLUE sub-tasks (MRPC, RTE and STS-B), since they are more vulnerable from over-fitting.

As shown in Table 3, the inspired scheduled Bernoulli dropout acts as an effective regularizer on BERT without any other regularization, quite closer to AttendOut and even slightly better on RTE, even if the strategy here is much looser. Indeed, it makes sense since the scheduler pattern, which turns out to be correct, is inverse-engineered from AttendOut, which provides us with the idea to carefully schedule the attention dropout pattern. On the other hand, we can see much lower variance on MRPC and STS-B in AttendOut case. Please refer to Appendix A.1 for details.

Table 3: Effect of regularization. The models are trained without attention dropout and weight decay.

Model	MRPC F1	RTE Acc	STS-B Spc
BERT	$84.9_{\pm 1.2}$	$69.3_{\pm 1.5}$	$88.7_{\pm 0.5}$
+ *Scheduled Bernoulli*	$85.4_{\pm 0.3}$	$\mathbf{70.6}_{\pm 1.8}$	$89.1_{\pm 0.4}$
+ *AttendOut*	$\mathbf{86.1}_{\pm 0.4}$	$70.2_{\pm 0.8}$	$\mathbf{89.7}_{\pm 0.4}$

Effect of Loss Features In the LSTM cell of AttendOut, the forget and input gates are featured with both training and evaluation losses with which we assume the expected dropout patterns are in strong relationship. To validate our design, we make experiments in which we remove the loss features in LSTM, that is the current cell state is merely related to the previous one. We report such results in Table 4, where considerable improvement can be seen on CoLA and STS-B when excluding loss features from AttendOut. However, we may find apparent performance gain when taking the advantage of training and evaluation loss to control the dynamic dropout.

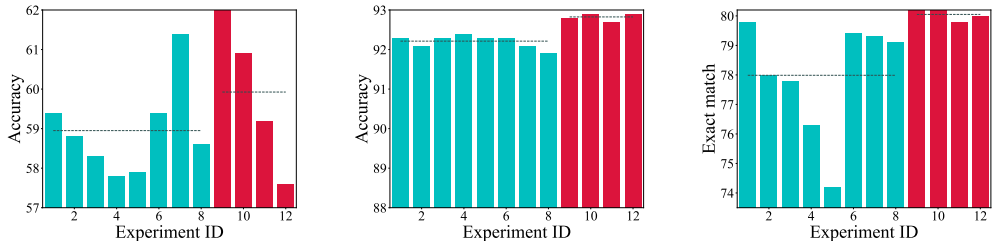
Table 4: Effect of Loss features in AttendOut.

Model	CoLA Mcc	MNLI-m/mm Acc	STS-B Spc
BERT	$57.4_{\pm 1.0}$	$84.4/84.0_{\pm 0.1}$	$88.7_{\pm 0.6}$
+ *AttendOut w/o loss*	$57.8_{\pm 0.7}$	$84.5/84.2_{\pm 0.0}$	$89.2_{\pm 0.3}$
+ *AttendOut*	$\mathbf{58.3}_{\pm 0.8}$	$\mathbf{84.7/84.6}_{\pm 0.1}$	$\mathbf{89.6}_{\pm 0.3}$

8 FURTHER STUDY

Comparison with Vanilla Dropout AttendOut introduces a crucial parameter K. Here comes a potential risk that tuning with K is still costly. To dispel this concern, we conduct parameter sweeps on three tasks, in which we go through eight kinds of Vanilla Dropout settings as well as four different meta steps chosen in {4, 8, 12, 15} for AttendOut. For Vanilla Dropout, we first choose 5 probabilities in {0, 0.1, 0.2, 0.3, 0.4} across all layers. Additionally, we design three more complicated settings: 0.1 for the first layer and 0 for the rest, 0.2 for the first and 0 for the rest, 0.1 for the first 0.2 for the second and 0 for the rest.

As we can see pictorially in Figure 3, AttendOut (red bars) takes a higher tolerance rate since the average performances (grey lines) are always higher on all three tasks. Intuitively, different meta steps result in little differences on SST-2 and SQuAD, while different dropout probabilities make large differences on CoLA and SQuAD. We can see tuning AttendOut with K serves as a more efficient way compared to Vanilla Dropout. A nice example is on CoLA where "0.2 first" setting



(a) CoLA (b) SST-2 (c) SQuAD

Figure 3: Parameter sweeps between AttendOut (red bars) and Vanilla Dropout. The experiment IDs from 1-5 refer to the dropout probabilities from 0 to 0.4, while the IDs 6, 7, 8 refer to "0.1 first", "0.2 first", "0.1 first 0.2 second" settings respectively.

largely outperforms other settings. The question is we can only go through a large search space to pick that out. For AttendOut, only four experiments are what we need.

Comparison with Structured Dropout We further compare AttendOut with LayerDrop (Fan et al., 2020), which focuses on skipping the entire attention blocks. Similarly, we choose the best dropout probability in $\{0.1, 0.2\}$. The results are shown in Table 5. Intuitively, both regularizers are effective on RoBERTa, while AttendOut performs stronger and more stable.

Table 5: Comparison of AttendOut and LayerDrop.

Model	CoLA Mcc	MNLI-m/mm Acc	STS-B Spc
RoBERTa	$60.0_{\pm 1.2}$	$87.6/86.6_{\pm 0.1}$	$90.4_{\pm 0.2}$
+ *LayerDrop*	$61.8_{\pm 0.7}$	$87.8/87.1_{\pm 0.2}$	$90.1_{\pm 0.3}$
+ *AttendOut*	$\mathbf{63.0}_{\pm 1.0}$	$\mathbf{87.8/87.4}_{\pm 0.1}$	$\mathbf{90.8}_{\pm 0.3}$

Combination with Vanilla Dropout We try to combine AttendOut and Vanilla Dropout together. Specifically, we halve each meta-train stage into two, in which we first apply Vanilla Dropout and then apply AttendOut. The following reward is made by the difference between two evaluations. In this sense, AttendOut is learning to compete with Vanilla Dropout, in which it looks like an ensemble of two regularizers. However, only the same amount of training time is needed. The results are reported in Appendix A.3 due to space limitation. As shown in Table 6, we can see the effectiveness when intermittently using two kinds of dropout approaches.

Interestingly, we find very different dropout patterns under this strategy, which we depict in Appendix A.3 due to space limitation. As depicted in Figure 5, the overall dropout proportion is unexpectedly low. Especially for CoLA, the 4^{th} layer is even close to be blocked. We notice that CoLA is a small set with 8500 training samples, on which SAN model is more inclined to suffer from over-fitting. It seems that intermittently using AttendOut makes it possible for much more intensive regularization. However, the situation on STS-B remains almost the same.

9 Conclusion

This paper presents a self-adjusting attention dropout based on meta-learning onto self-attention empowered PrLMs. Extensive experiments on multiple natural language processing and understanding tasks demonstrate the proposed Attendout is a stronger regularizer for further PrLM enhancement on both pre-training and fine-tuning stages. We probe into the learned dropout patterns on different tasks, which empirically guide us for the needed dynamic attention dropout design.

REFERENCES

Lei Jimmy Ba and Brendan J. Frey. Adaptive dropout for training deep neural networks. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 3084–3092, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/7b5b23f4aadf9513306bcd59afb6e4c9-Abstract.html.

Shahin Boluki, Randy Ardywibowo, Siamak Zamani Dadaneh, Mingyuan Zhou, and Xiaoning Qian. Learnable bernoulli dropout for bayesian deep learning. In Silvia Chiappa and Roberto Calandra (eds.), *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3905–3916. PMLR, 2020. URL http://proceedings.mlr.press/v108/boluki20a.html.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1xMH1BtvB.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SylO2yStDr.

Xinjie Fan, Shujian Zhang, Korawat Tanwisuth, Xiaoning Qian, and Mingyuan Zhou. Contextual dropout: An efficient sample-dependent dropout module. *CoRR*, abs/2103.04181, 2021. URL https://arxiv.org/abs/2103.04181.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 2017. URL http://proceedings.mlr.press/v70/finn17a.html.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1050–1059. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/gal16.html.

Xinwei Geng, Longyue Wang, Xing Wang, Bing Qin, Ting Liu, and Zhaopeng Tu. How does selective mechanism improve self-attention networks? In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 2986–2995. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.269. URL https://doi.org/10.18653/v1/2020.acl-main.269.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. {DEBERTA}: {DECODING}-{enhanced} {bert} {with} {disentangled} {attention}. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XPZIaotutsD.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Minki Kang, Moonsu Han, and Sung Ju Hwang. Neural mask generator: Learning to generate adaptive word maskings for language model adaptation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6102–6120. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.493. URL https://doi.org/10.18653/v1/2020.emnlp-main.493.

Zhen Ke, Liang Shi, Songtao Sun, Erli Meng, Bin Wang, and Xipeng Qiu. Pre-training with meta learning for chinese word segmentation. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 5514–5523. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.436. URL https://doi.org/10.18653/v1/2021.naacl-main.436.

Ashish Khetan and Zohar S. Karnin. schubert: Optimizing elements of BERT. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 2807–2818. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.250. URL https://doi.org/10.18653/v1/2020.acl-main.250.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 971–980, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5d44ee6f2c3f71b73125876103c8f6c4-Abstract.html.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Haebeom Lee, Taewook Nam, Eunho Yang, and Sung Ju Hwang. Meta dropout: Learning to perturb latent features for generalization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BJgd81SYwr.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019a. URL https://openreview.net/forum?id=S1eYHoC5FX.

Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. In Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé (eds.), *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pp. 809–817. ACM, 2021. URL https://dl.acm.org/doi/10.5555/3463952.3464049.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019b. URL http://arxiv.org/abs/1907.11692.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6379–6390, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html.

Shangwen Lv, Yuechen Wang, Daya Guo, Duyu Tang, Nan Duan, Fuqing Zhu, Ming Gong, Linjun Shou, Ryan Ma, Daxin Jiang, Guihong Cao, Ming Zhou, and Songlin Hu. Pre-training text representations as meta learning. *CoRR*, abs/2004.05568, 2020. URL https://arxiv.org/abs/2004.05568.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (eds.), *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pp. 142–150. The Association for Computer Linguistics, 2011. URL https://www.aclweb.org/anthology/P11-1015/.

Chris J. Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3086–3094, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/309fee4e541e51de2e41f21bebb342aa-Abstract.html.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330, 1993.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14014–14024, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/2c601ad9d2ff9bc8b282670cdd54f69f-Abstract.html.

Hieu Pham and Quoc V. Le. Autodropout: Learning dropout patterns to regularize deep networks. *CoRR*, abs/2101.01761, 2021. URL https://arxiv.org/abs/2101.01761.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rJY0-Kcll.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne (eds.), *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pp. 142–147. ACL, 2003. URL https://www.aclweb.org/anthology/W03-0419/.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, 2014. URL http://dl.acm.org/citation.cfm?id=2670313.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. DREAM: A challenge dataset and models for dialogue-based reading comprehension. *Trans. Assoc. Comput. Linguistics*, 7:217–231, 2019. URL https://transacl.org/ojs/index.php/tacl/article/view/1534.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller (eds.), *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pp. 1057–1063. The MIT Press, 1999. URL https://dl.acm.org/doi/10.5555/3009657.3009806.

Sebastian Thrun and Lorien Y. Pratt. Learning to learn: Introduction and overview. In Sebastian Thrun and Lorien Y. Pratt (eds.), *Learning to Learn*, pp. 3–17. Springer, 1998. doi: 10.1007/978-1-4615-5529-2_1. URL https://doi.org/10.1007/978-1-4615-5529-2_1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 5797–5808. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1580. URL https://doi.org/10.18653/v1/p19-1580.

Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 1058–1066. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/wan13.html.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Hongqiu Wu, Hai Zhao, and Min Zhang. Code summarization with structure-induced transformer. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 1078–1090. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.93. URL https://doi.org/10.18653/v1/2021.findings-acl.93.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. Sg-net: Syntax-guided machine reading comprehension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 9636–9643. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6511.

Wangchunshu Zhou, Tao Ge, Furu Wei, Ming Zhou, and Ke Xu. Scheduled drophead: A regularization method for transformer models. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pp. 1971–1980. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.178. URL https://doi.org/10.18653/v1/2020.findings-emnlp.178.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

A COMPLEMENTARY RESULTS

A.1 CONFIDENCE INTERVAL

We depict the confidence intervals on a variety of tasks in Figure 4, where we can see very wide ranges on small datasets like CoLA, MRPC, RTE and DREAM. However, the overall performance variance when training with AttendOut looks lower. Especially on RTE, the confidence interval for AttendOut is 0.6, much lower than that for Vanilla Dropout. In most cases, the dropout proportion brought by AttendOut is higher. We speculate that a more intensive regularizer might help reduce the variance.

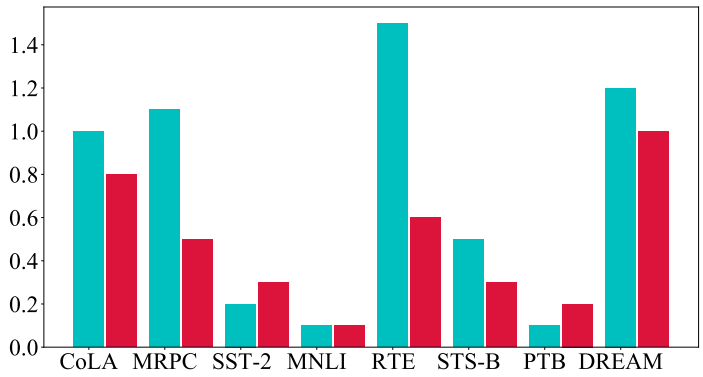
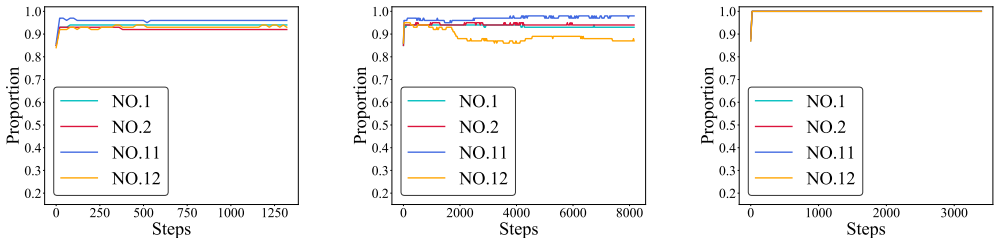


Figure 4: Confidence intervals between AttendOut (red bars) and Vanilla Dropout on a variety of tasks. The results here are on development sets.

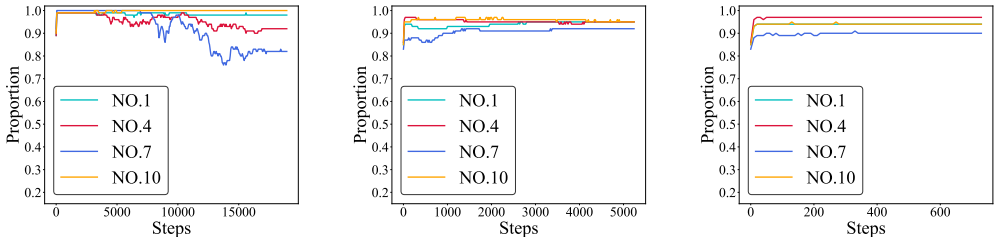
A.2 DROPOUT PATTERNS

We depict the dropout patterns on other layers in Figure 5, the lowest and highest two layers respectively. Previously, it is found that the 7^{th} layer takes the highest dropout proportion on CoLA and QNLI (the situation is similar in Figure 8). Now we can see that the dropout proportion of the 11^{th} layer is always much higher than that of the last layer, while the first two layers take similar dropout proportions. Note that CoLA is for single-sentence classification while QNLI is for sentence-pair classification. Interestingly, we find similarities between the two. However, it is different on STS-B, where all the layers take very low dropout proportions all the time. We assume that there is a specific pattern for each attention layer, which is probably task-dependent. Further study for this part will be reserved for future work.



(a) CoLA (b) QNLI (c) STS-B

Figure 5: Dropout patterns of both the lowest and highest two layers. The tasks we choose here are the same with those in Figure 1.



(a) Pre-training (b) SST-2 (c) RTE

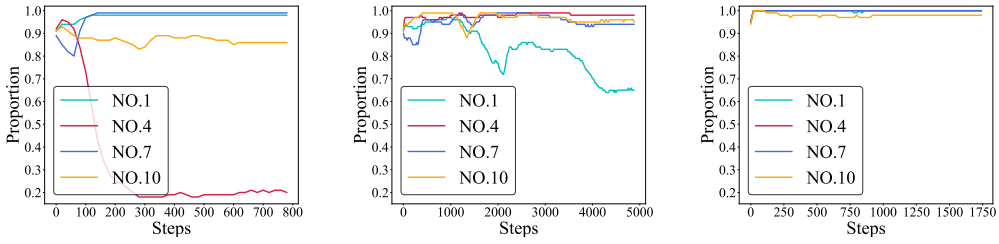
Figure 6: Dropout patterns on more tasks. For conciseness, we uniformly take four attention layers.

A.3 Combination with AttendOut and Vanilla Dropout

Interestingly, we find very different dropout patterns under this strategy. As depicted in Figure 5, the overall dropout proportion is unexpectedly low. Especially for CoLA, the 4^{th} layer is even close to be blocked. We notice that CoLA is a small set with 8500 training samples, on which SAN model is more inclined to suffer from over-fitting. It seems that intermittently using AttendOut makes it possible for much more intensive regularization. However, the situation on STS-B remains almost the same.

Table 6: GLUE development results when intermittently using AttendOut and Vanilla Dropout.

Model	CoLA Mcc	SST-2 Acc	MRPC F1	QNLI Acc	MNLI-m/mm Acc	QQP Acc	RTE Acc	STS-B Spc
BERT	57.3	92.1	84.2	**91.3**	84.4/84.0	**91.2**	70.2	88.9
+ *Atd.O. w Van.*	**59.8**	**92.5**	**85.2**	91.0	**84.7/84.7**	91.1	**72.0**	**89.7**



(a) CoLA (b) QNLI (c) STS-B

Figure 7: Dropout patterns when intermittently using AttendOut and Vanilla Dropout.

A.4 Temporal and Non-temporal Modeling

We have tried MLP-based modeling, which also works and achieves competitive results as in Table 7. It turns out that our proposed optimization is general under different modeling manners, while we see LSTM-base one performs better.

There is another contributed factor of choosing LSTM. We assume that the dropout proportion is related to the training and evaluation losses, to prevent the situation when model fits well on training points but behaves worse on unseen ones. For example, the training loss is continually down, while the evaluation loss is up. LSTM is able to model such trend in a few steps but MLP cannot which merely cares about the moment.

Table 7: Comparison of LSTM-based and MLP-based modeling.

Model	CoLA Mcc	SST-2 Acc	MRPC F1	MNLI-m/mm Acc	STS-B Spc
BERT	51.5	92.9	87.6	84.3/83.6	84.6
+ *AttendOut-MLP*	**56.4**	93.6	88.0	84.6/**83.8**	85.2
+ *AttendOut-LSTM*	**56.8**	**93.9**	**88.1**	**84.6**/83.7	**85.8**

B TRAINING DETAILS

Table 8: Hyperparameters for BERT and RoBERTa in all downstream tasks. LR: learning rate; BSZ: batch size; EP: training epochs; WP: warmup proportion; MSL: maximum sequence length; MLR: meta learning rate.

	GLUE	IMDB	CoNLL03	PTB	DREAM
LR	{1e-5, 2e-5, 3e-5}	3e-5	5e-5	5e-5	1e-5
BSZ	{16, 32}	32	32	32	8
EP	3	5	5	5	8
WP	0.06	0.1	0.1	0.1	0.06
MSL	128	256	128	128	128
MLR	1e-4	1e-4	1e-4	1e-4	1e-4
K	{4, 8, 12, 15}	12	8	8	8

C TRAINING EFFICIENCY

C.1 TIME CONSUMPTION

As shown, the computational cost is $0.2\% \sim 0.7\%$ increased, which is less than two tries of dropout probability using Vanilla Dropout. However, if we target to tune the probabilities for all attention layers, twice is not enough.

Table 9: Time consumption using AttendOut on selected tasks compared to using Vanilla Dropout. We conduct random sampling in the situation of abundant validation data to guarantee the efficiency.

	CoLA	SST-2	MNLI-m	RTE	STS-B
Performance up	5.5	1.0	0.3	3.0	0.7
Consumption up	x1.4	x1.5	x1.7	x1.2	x1.5

C.2 RESOURCE USAGE

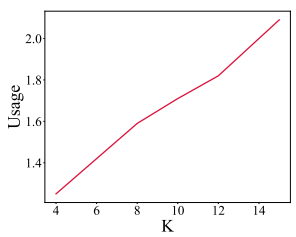


Figure 8: Memory usage using AttendOut.