BERMo: What can BERT learn from ELMo?

Anonymous authors
Paper under double-blind review

Abstract

We propose BERMo, an architectural modification to BERT, which makes predictions based on a hierarchy of surface, syntactic and semantic language features. We use linear combination scheme proposed in Embeddings from Language Models (ELMo) to combine the scaled internal representations from different network depths. Our approach has two-fold benefits: (1) improved gradient flow for the downstream task as every layer has a direct connection to the gradients of the loss function and (2) increased representative power as the model no longer needs to copy the features learned in the shallower layer which are necessary for the downstream task. Further, our model has a negligible parameter overhead as there is a single scalar parameter associated with each layer in the network. Experiments on the probing task from SentEval dataset show that our model performs up to 4.65% better in accuracy than the baseline with an average improvement of 2.67% on the semantic tasks. When subject to compression techniques, we find that our model enables stable pruning for compressing small datasets like SST-2, where the BERT model commonly diverges. We observe that our approach converges $1.67\times$ and $1.15\times$ faster than the baseline on MNLI and QQP tasks from GLUE dataset. Moreover, our results show that our approach can obtain better parameter efficiency for penalty based pruning approaches on QQP task.

1 Introduction

The invention of Transformer (Vaswani et al. (2017)) architecture has paved new research directions in the deep learning community. Descendants of this architecture, namely BERT (Devlin et al. (2019)) and GPT (Brown et al. (2020)), attain State-of-The-Art (SoTA) performance for a broad range of NLP applications. The success of these networks is primarily attributed to the two-stage training process (self-supervised pre-training and task-based fine-tuning), and the attention mechanism introduced in Transformers. Many of the top models on various leader boards use models from the BERT family. All of the fifteen systems that surpass the human baseline on the General Language Understanding Evaluation (GLUE) (Wang et al. (2018)) benchmark use variants of BERT or have it as one of the constituents in an ensemble, except for T5 (Raffel et al. (2019)), which uses the Transformer architecture. Further, the best-performing systems for each task in the Ontonotes (Weischedel et al. (2011)) benchmark belong to the BERT family, with the exception of Entity Typing task where Embeddings from Language Models (ELMo) (Peters et al. (2018)) tops the leaderboard. These promising results make the BERT family of models increasingly ubiquitous in solving several tasks in the various domains of machine learning like NLP, Image Recognition (Dosovitskiy et al. (2021); Jaegle et al. (2021)) and Object detection (Carion et al. (2020)).

Motivation: In (Jawahar et al. (2019)), the authors manifest the lower layers of BERT to capture phrase-level information, which gets diluted with the depth. Moreover, they also demonstrate that the initial layers capture surface-level features, the middle layers deal with syntactic features, and the last few layers are responsible for semantic features. These findings indicate that BERT captures a rich hierarchy of linguistic features at different depths of the network. Intrigued by this discovery, we aim at combining the activations from different depths to obtain a richer feature representation. We find our problem formulation similar to the one presented in ELMo, where the authors illustrate higher-level Long Short Term Memory (LSTM) states capture context-dependent aspects of word meaning or semantic features, and the lower-level LSTM states model the syntax. Inspired by ELMo, we propose BERMo by modifying the BERT architecture to increase the dependence of features from different depths to generate a rich context-dependent embedding. This approach improves the gradient flow during the backward pass and increases the representative power of the network (He et al. (2016); Huang et al. (2016)). Further, the linear combination of features from intermediate layers, proposed in ELMo, is simpler form of skip connection introduced in ResNets (He et al. (2016)). The skip connections in ResNets enable aggressive pooling in initial layers without affecting the gradient flow. This in turn leads to low parameters allowing these networks to have orders of magnitude lower parameters compared to the architectures without skip connections such as VGG (Simonyan & Zisserman (2014)) while achieving competitive accuracy. Since the performance improvements associated with the BERT architecture come at the cost of a large memory footprint and enormous computational resources, compression becomes necessary to deploy these networks in resource-constrained environments like drones, mobile computers and IoT devices. As we introduce skip connections in the architecture to obtain a complex feature representation, the gradient flow during the backward pass improves. Further, our model has a negligible parameter overhead as there is a single scalar parameter associated with each layer in the network.Therefore, we expect BERMo architecture to be better candidate for compression and hence, we believe the proposed model could be ideal for resource-constrained settings.

Contributions: The contributions of this work can be summarized as follows:

i We propose BERMo, which generates complex feature maps using linear combination of features from different depths. We evaluate the proposed model on the probing task from SentEval dataset (Conneau et al. (2018)) and find our model performs 2.67% better than the baseline on the semantic tasks (Tense, Subjnum, Objnum, Somo, Coordinv) on average.

ii We observe our approach is stable when pruning with smaller datasets like SST-2 (Wang et al. (2018)), where BERT commonly diverges.

iii We show our model supports higher pruning rates when compressing and converges $1.67\times$ and $1.15\times$ faster than BERT on MNLI and QQP (Wang et al. (2018)), respectively.

iv For loss penalty based pruning method our approach can obtain better parameter efficiency, $1.35\times$ for QQP, than BERT model for comparable performance.

v Our approach produces comparable results to the baseline for Knowledge Distillation with marginal improvements on SQuAD dataset.

Outline: The rest of the paper is organised as follows: Section 2 describes ELMo (Peters et al. (2018)), BERT (Devlin et al. (2019)) and Pruning methods. Section 3 elaborates the proposed model. The experimental setup and the results are presented in Section 4. In Section 5 we summarize our work and discuss the future possibilities. Section 6 reports the related work and we conclude our paper with Broader Impact in Section 7.

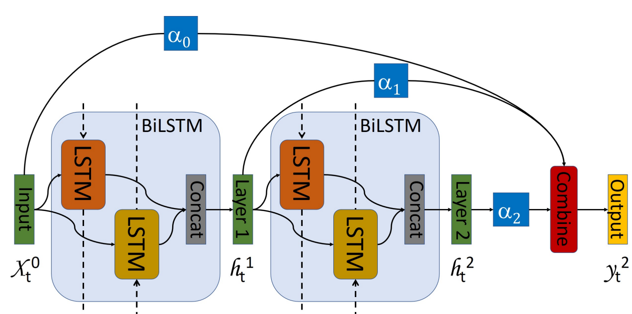

Figure 1: ELMo model Architecture Peters et al. (2018)

2 Background

2.1 ELMo

$$\textbf{Combine} = \gamma \sum_{j=0}^{L} \alpha_j \times \mathbf{h}_j \quad (1)$$

where, h_j is the activation, α_j is a learnt parameter associated with layer j and γ is the scaling factor learnt during training.

ELMo (Peters et al. (2018)) studies the features at different depths of a Bi-LSTM architecture. The authors analyze a two-layered Bi-LSTM network, where they illustrate that the first layer is good at capturing the syntactic aspects and the second layer captures the semantic information. Further, the authors of this work propose an architecture shown in Figure 1, to combine the features from the different layers to obtain complex features representations. Equation (1) presents the mathematical details for mixing the activations from different depths in this architecture. The authors present a two-stage training process. The first stage, called the pre-training phase, involves training the network on a large unlabelled corpus in an unsupervised fashion. The second stage, popularly known as the fine-tuning phase, deals with supervised training on a downstream task. Due to the tremendous generalization improvements, practitioners have widely adopted this two-stage training methodology.

2.2 BERT

Similar to ELMo, BERT (see Figure 2) uses the two-stage training process on the encoder network from the Transformer model. The pre-training stage performs self-supervised learning on a large corpus of training data. The model is trained on Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) objectives in the semi-supervised training phase. MLM objective trains the model to predict the masked, replaced, or retained words. Such a scheme enables bidirectional information flow for the prediction of a particular word. The NSP task is oriented for the model to learn the relationships between the sentences. The authors used two additional embeddings along with the token embedding: the segment embedding to distinguishing the different sentences, and the position embedding that marks the position of the words in the input. We refer the readers to (Devlin et al. (2019)) for implementation details.

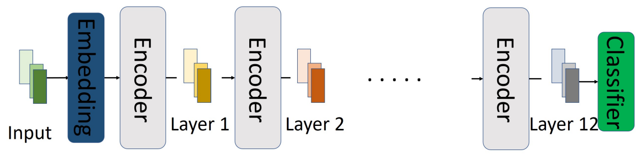

Figure 2: BERT model architecture Devlin et al. (2019)

2.3 PRUNING

The general form of pruning method can be expressed as $a = (W \odot M)x$, where x is the input to the current layer, a is the activation of the output layer, W is the layer weight, M is the pruning mask, and $\odot$ is the Hadamard product (element-wise product). The pruning mask M is the same size as the weights, with each element commonly restricted to 0 or 1. M is a function of the importance scores, S, of the weights, mathematically represented by $M = f_p(S)$, where $f_p(.)$ is some nonlinear function. Any pruning approach can be defined by the algorithm used to compute the importance score (S) and the mapping function ($f_p(.)$) used to compute the mask M.

2.3.1 MAGNITUDE PRUNING

One of the well known approaches, Magnitude Pruning (Han et al. (2015)), defines $S_{i,j} = (|W_{i,j}|)$ or the magnitude of the weights and the function $f_p(.) = \text{Top}_\text{v}(.)$ given by equation (2). This algorithm is 0^{th} order as it does not rely on the gradients of the weights.

$$\text{Top}_\text{v}(S_{i,j}) = \begin{cases} 1 & S_{i,j} \text{ in top v \%} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2.3.2 L_0 PRUNING

This approach uses gradient information to compute S. This is a stochastic pruning approach as it samples a parameter from uniform distribution. The mask function $f_p(.)$ is given by equation (3, 4) and the Scores (S) are computed using equation (5). At the time of inference, the non stochastic version, given by equation (6) is used. In this approach L_0 penalty is added to the optimization which indirectly controls the levels of sparsity in the model.

$$u \sim \mathcal{U}(0,1), \qquad \overline{S}_{i,j} = \sigma((log(u) - log(1-u) + log(S_{i,j}))/\beta) \tag{3}$$

$$Z_{i,j} = \overline{S}_{i,j}(\epsilon - \gamma) + \gamma \quad M_{i,j} = \min(1, \max(0, Z_{i,j})) \tag{4}$$

where, β, γ and ϵ are hyperparameters.

$$S_{i,j} = -\sum_t \frac{\partial \mathcal{L}}{\partial W_{i,j}} \times W_{i,j} \times g(\overline{S}_{i,j}), \quad g(\overline{S}_{i,j}) = \frac{(\epsilon - \gamma)}{\beta} \times \overline{S}_{i,j} \times (1 - \overline{S}_{i,j}) \times 1_{\{1 \leq Z_{i,j} \leq 1\}} \tag{5}$$

$$M = \min(1, \max(0, \frac{(\epsilon - \gamma)}{\beta}\sigma(S) + \gamma)) \tag{6}$$

2.3.3 Movement Pruning

Movement pruning (Sanh et al. (2020)) gives importance to the changes in the weights during the fine-tuning task instead of just relying on the magnitude of the weight. The core idea is to prune the weights that shrink during the fine-tuning process. Such a scheme retains lower values of weights with an increasing magnitude over the higher weights with shrinking magnitudes. Mathematically, the masking function $f_p(.)$ is the same as the Magnitude pruning (equation (2)) and the Scores S is given by equation (7). As the gradient of Top_v is 0 wherever it is defined, we use the straight-through estimator (Bengio et al. (2013)). In order to gradually prune the weights, cubic sparsity scheduler (Zhu & Gupta (2017)) is introduced. We follow the implementation from (Sanh et al. (2020))

$$S_{i,j} = -\sum_t \frac{\partial \mathcal{L}}{\partial W_{i,j}} \times W_{i,j} \tag{7}$$

2.4 Knowledge Distillation

Knowledge Distillation (KD) (Hinton et al. (2015)) is a technique where a smaller model, called the student model, is trained to mimic the logits of a bigger network, called the teacher model. This technique is different from transfer learning as the student model is different from the teacher model, and the weights of the students are not initialized based on the teacher model. KD improves the generalization capacity of the smaller network (Jiao et al. (2019); Sanh et al. (2019)). In order to perform KD, its not necessary to have access to the training dataset. One can train the student network to mimic the teacher network on random inputs.

3 Methodology

3.1 BERMo

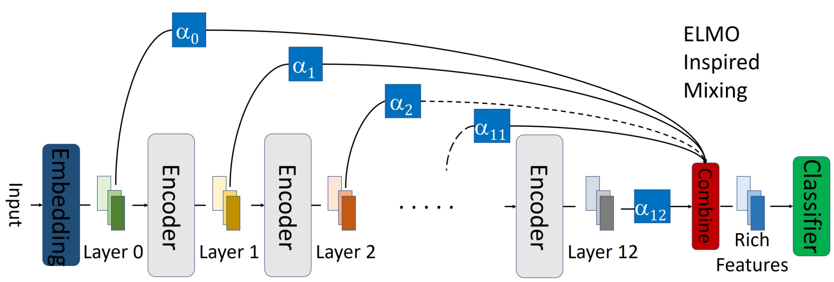

Figure 3: Proposed model architecture BERMo

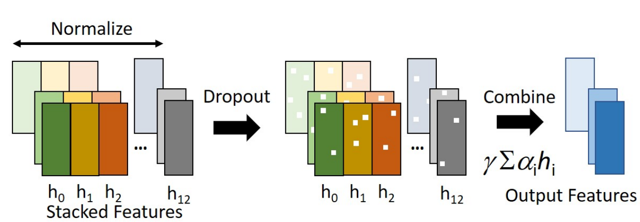

Figure 4: Combine Block

BERMo, shown in Figure 3, modifies the BERT architecture by linearly combining the features using the scheme proposed in ELMo in order to obtain a rich feature representation. Learnable scalar parameters α_i is associated with the activations of i^{th} layer with α_0 being the scalar parameter associated with embedding of the input. These scalar parameters are softmax normalized to ensure $\sum_{i=0}^{12} \alpha_i = 1$. The **combine block** is used to obtain the weighted-average of the features from different layers and scales the result with a learnable parameter γ following the equation (1). The parameters α's are initialized using Xavier initialization (Glorot & Bengio (2010)) and γ is initialized to 1.

To ensure the inputs are on similar scale, we normalize the stacked features across the layer as shown in Figure 4. This is conceptually similar to Layer Normalization (Ba et al. (2016)) where channels are replaced by the features of the layers. Dropout (Srivastava et al. (2014)) is used as a regularizer to avoid the dependence of neural network on specific neurons. Finally, the features maps from each layer are combined with the scalar weights as per equation (1) to obtain the weighted average of features from different depths.

4 RESULTS

In our experiments we consider BERT-based-uncased[1] as our baseline model which roughly contains 84M parameters. Our experiments are built on top of Movement Pruning Repository by Huggingface (Sanh et al. (2020))[2]. We call this model as $\text{BERT}_{\text{BASE}}$ throughout the paper. We refer to the model with our approach applied to $\text{BERT}_{\text{BASE}}$ as $\text{BERMo}_{\text{BASE}}$. We run all our experiments on RTX 2080 Ti having Intel Xeon Silver 4114 CPU (2.20 GHz). For all the tasks we use a single card, except for SQuAD simulations where we use 2 cards to support longer sequence length. Detailed list of hyperparameters for all the experiments can be found in Appendix A.1. We found that higher learning rates for the skip connections improved the performance of the model and hence the learning rate for the skip connections was kept same as the mask learning rate (10^{-2}). The code for the experiments in this work can be found on `https://anonymous.4open.science/r/BERMo-112D/README.md`

In this section, we present the results for various experiment. We assess the performance of our model on different linguistic tasks (4.1). In Subsection 4.2, we test the fine-pruning performance of the models for stability (4.2.1), convergence (4.2.2), different pruning approaches (4.2.3) and knowledge distillation (4.2.4).

4.1 STANDALONE PERFORMANCE OF BERMO

Model	Surface		Syntactic			Semantic				
	Sentlen	WC	TreeDepth	TopConst	Bshift	Tense	SubjNum	ObjNum	Somo	CoordInv
$\text{BERT}_{\text{BASE}}$	99.12	**84.16**	**72.77**	**77.07**	96.22	82.57	87.43	90.71	61.51	82.95
$\text{BERMo}_{\text{BASE}}$	**99.24**	83.39	72.36	76.85	**96.30**	**87.22**	**90.21**	**93.68**	**64.27**	**83.16**
Improvements	**0.036**	-0.771	-0.41	-0.214	**0.082**	**4.646**	**2.78**	**2.968**	**2.756**	**0.212**

Table 1: Accuracy comparison between $\text{BERT}_{\text{BASE}}$ and $\text{BERMo}_{\text{BASE}}$ on the probing task from the SentEval dataset. The results in the table correspond to the average accuracy over 5 runs. Two of the runs diverged for the WC tasks on the $\text{BERT}_{\text{BASE}}$ model and hence the results for WC are averaged over three runs.

As we combine the features from different depths to obtain a rich representation, we follow Jawahar et al. (2019) and evaluate our model on the probing tasks from the SentEval dataset (Conneau et al. (2018)). Tasks of this dataset are categorised into Surface (Sentence Length-Sentlen, Word content-WC), Sytactic (TreeDepth, TopConst, and Bshift) and Semantic tasks (Tense, SubjNum, ObjNum, somo and coordinv), helping us examine what linguistic tasks benefit from the BERMo architecture. Further, all the tasks of this dataset have 100k training, 10k validation and 10k testing samples with each class having identical number of samples. These properties of the dataset allow us to compare the raw accuracies of the model without worrying about the data imbalance issue. In this subsection, we evaluate $\text{BERMo}_{\text{BASE}}$ against $\text{BERT}_{\text{BASE}}$ on these probing task from the SentEval dataset to understand what linguistic feature benefit from our approach. All the models in these experiments are run for 3 epochs with the batch size of 32 and a sequence length of 128 following (Sanh et al. (2020)). Further, to ensure the results are statistically significant we run our experiment with 5 randomly chosen initialization. The detailed results for each initialization are presented in Appendix A.2. Table 1 presents the average over these 5 trails. We find our approach outperform $\text{BERT}_{\text{BASE}}$ on 7 of these 10 tasks. Semantic tasks benefit the most from complex feature representation enabled

[1]`https://huggingface.co/bert-base-uncased`
[2]`https://github.com/huggingface/block_movement_pruning`

Model	SST-2 Acc	MNLI ACC-m / ACC-mm	QQP Acc./F1	SQuAD F1
BERT$_{\text{BASE}}$	91.74	**80.58/81.2**	**88.62/84.53**	76.95
BERMo$_{\text{BASE}}$	**92.43**	80.04/80.84	88.27/84.29	**77.43**

Table 2: Accuracy/F1-Score comparison results for SST-2, MNLI, QQP and SQuAD for BERT$_{\text{BASE}}$ and BERMo$_{\text{BASE}}$.

by our approach. All the five semantic tasks show improvements in accuracy with the maximum improvement of $\sim$ 4.646% obtained on Tense task and an average improvement of $\sim$ 2.673% (Contribution (i)). We find our approach improves over the baseline by $\sim$ 1.21% across all the 10 tasks on average.

4.2 Pruning

As we introduce skip connections in our model, we expect our model to train in a stable and efficient manner when subject to pruning techniques. In our experiments, we find pruning approach diverges on some of the probing tasks for the baseline model. We believe this unstable pruning behaviour is due to the small size of the dataset. We test this hypothesis with Stanford Sentiment Treebank-2 (SST-2) task from the General Language Evaluation and Understanding (GLUE) dataset (Wang et al. (2018)) and find the baseline model diverges most of the times whereas our approach is compatible with smaller datasets. To fairly compare our approach with BERT$_{\text{BASE}}$, for the pruning task we evaluate our approach on larger datasets like Multi-genre Natural Language Inference (MNLI) and Quora Questions Pair (QQP) tasks from the GLUE dataset and Stanford Question Answering (SQuAD) dataset (Rajpurkar et al. (2016)). For comparing the model performance we use the F1 score for SQuAD, accuracy and F1 score for QQP, accuracy for MNLI and accuracy for SST-2. These datasets are selected in order to be comparable to Sanh et al. (2020). The performance of our model on these tasks, without the compression, are reported in Table 2. The hyperparameters for this experiment are kept same as Sanh et al. (2020) and are reported in Appendix A.1.

For our analysis in Subsection (4.2.1, 4.2.2, 4.2.4), we chose Movement Pruning (2.3.3) (Sanh et al. (2020)) for benchmarking our approach against the baseline. This approach was chosen as it allows us to control the exact pruning fraction in the experiments, which allows us to fairly compare different models. We compare the two approaches for nearly 90% compression in these sections. Further in Subsection (4.2.3), we maintain $\sim$ 10% weight retention for magnitude and movement pruning approaches, however, for penalty based approaches we select the hyperparameter to be close to $\sim$ 90% pruning from Sanh et al. (2020) for BERT$_{\text{BASE}}$. The hyperparameters for BERMo$_{\text{BASE}}$ for loss penalty based methods were selected to match the performance of the baseline.

4.2.1 Stability under Pruning

Model	Seed 9	Seed 25	Seed 39	Seed 52	Seed 59	Seed 63	Seed 77	Seed 87	Seed 91	Seed 96
BERT$_{\text{BASE}}$	79.85	Diverges	Diverges	82.22	Diverges	Diverges	Diverges	Diverges	Diverges	Diverges
BERMo$_{\text{BASE}}$	87.39	87.61	85.88	87.04	87.39	86.47	87.39	86.70	86.58	86.24

Table 3: Stability comparison on SST-2 dataset for different initialisation.

During our experiment with the probing task we found the baseline approach to diverge when we used movement pruning to retain 10% of the weights. Such divergence was not seen when pruning was applied to our model. These divergence issues were significant when the training dataset size is small. The skip connection in our model leads to better gradient flow and we believe that improved gradient flow is important for the stability of the pruning. To verify this claim, we evaluate the proposed model for pruning on SST-2 which is a small dataset having 67k training samples. The results in Table 3 show that the BERT$_{\text{BASE}}$ model in most cases diverges with this dataset. We also found that the BERT$_{\text{BASE}}$ model diverged for two of the seeds for Word Content task, showing that our approach can also stabilize the fine-tuning phase. Our proposed model did not diverge during the pruning or the training process in any of our experiments (Contribution (ii)).

4.2.2 CONVERGENCE

We analyze the convergence by monitoring how the performance of the models varies with the change in epoch. As fine-pruning with smaller dataset was challenging for the baseline approach, we chose to evaluate the convergence for MNLI, QQP and SQuAD tasks. For this study we employ movement pruning to retain $\sim$ 10% of the model weights. We use the cubic sparsity scheduler (Zhu & Gupta (2017)), where the model gradually reduces the network weights over the epochs, pruning certain amount of weights as dictated by the sparsity scheduler. Hence, varying the number of epochs also controls the speed of pruning i.e. lower epochs means more parameters being pruned per epoch to attain the same desired sparsity level ($\sim$ 10%) at the end the fine-pruning.

Figure 5 shows the results for MNLI, QQP and SQuAD. We see that our approach converges significantly faster on MNLI (1.67×) and QQP (1.15×) while the convergence on SQuAD is similar to the baseline (Contribution (iii)). On our setup, each epoch for MNLI and QQP took roughly 5hrs. The faster convergence obtained by our method hence saves 10hrs and 5hrs on MNLI and QQP respectively. Moreover, less epochs also means that the model supports higher pruning rates as the model is able to prune 90% of weights is lesser epochs.

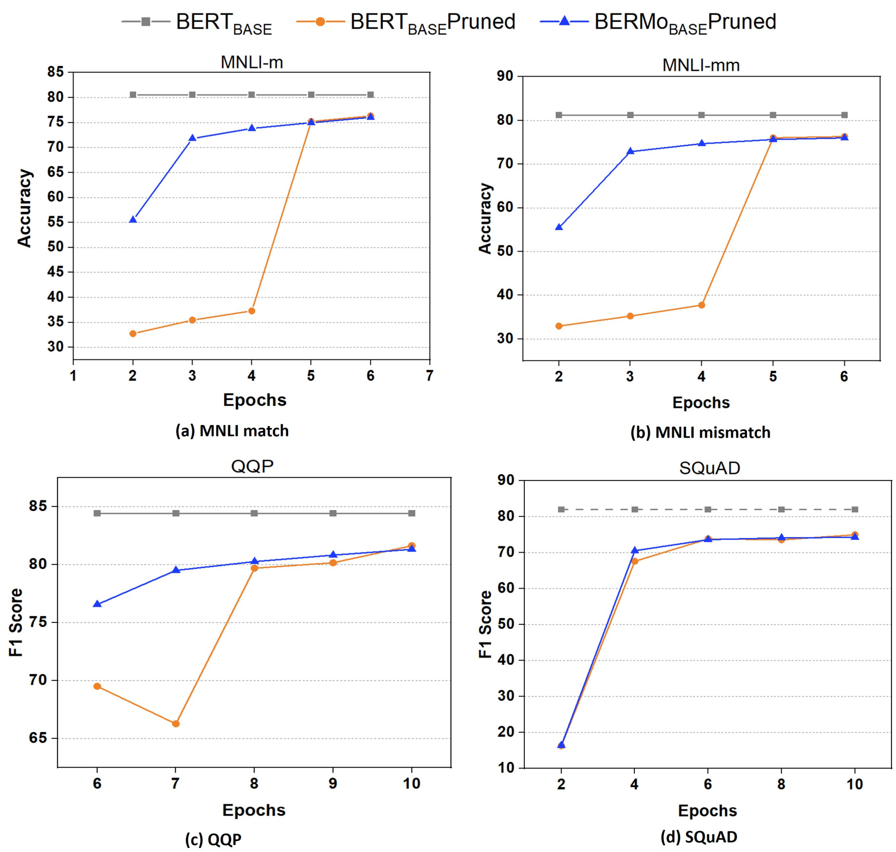

Figure 5: Accuracy/F1-Score comparison with varying epochs for pruning BERT$_{\text{BASE}}$ and BERMo$_{\text{BASE}}$ to retain 10% of the weights. BERT Baseline represents the performance of the model trained for 3 epochs without any pruning (Table 2).

4.2.3 DIFFERENT PRUNING

We compare our model with Magnitude Pruning (2.3.1), L_0 Pruning (2.3.2), Movement Pruning and Soft Movement Pruning (2.3.3). We train the both the models with the best hyperparameters (Sanh et al. (2020)) for the baseline to ensure that the baseline converges. We find our model performance is similar to the baseline. The pruning threshold for Magnitude and Movement pruning approaches are set to retain $\sim$ 10% of the weights. The pruning percentage for the loss penalty based approaches like the L_0 and Soft Movement Pruning can be tuned by varying the Lagrange multiplier (λ) for the pruning loss term. However, it is difficult to exactly control the percentage of weights retain with these approaches. Hence, we train the $\text{BERT}_{\text{BASE}}$ model with the hyperparameters suggested by (Sanh et al. (2020)) and for experiment on our model we select lambda such that the performance of our model matches the baseline model performance.

The result of these experiments is presented in Table 4. Although the model is trained with the optimal hyperparameters for the baseline our model achieves comparable results to $\text{BERT}_{\text{BASE}}$ for the Magnitude and Movement pruning approach. Surprisingly, for penalty based methods we are able to prune more aggressively for comparable accuracies on QQP task (Contribution (iv)). In the case of L_0 regularization, our model has $1.4\times$ less weights as compared to $\text{BERT}_{\text{BASE}}$ for QQP. Similarly, for soft movement pruning approach our model has $1.29\times$ less weight as compared to $\text{BERT}_{\text{BASE}}$ for QQP. Soft movement pruning is the State-of-The-Art approach in literature (Sanh et al. (2020)) and the results in Table 4 show our model can potentially prune more weights for comparable accuracy for some tasks.

Model	Pruning Approach	MNLI Remaining Weights	MNLI ACC-m / ACC-mm	QQP Remaining Weights	QQP Acc./F1	SQuAD Remaining Weights	SQuAD F1
$\text{BERT}_{\text{BASE}}$	Magnitude	10.1%	74.9/76.05	10.1%	86.37/81.41	10.1%	71.05
$\text{BERMo}_{\text{BASE}}$			74.2/75.19		85.81/80.97		70.93
$\text{BERT}_{\text{BASE}}$	L0	7.6%	74.42/74.96	7.7%	85.85/79.44	9.4%	72.72
$\text{BERMo}_{\text{BASE}}$		7.5%	74.27/74.91	5.5%	85.78/80.67	9.5%	72.46
$\text{BERT}_{\text{BASE}}$	Movement	10.1%	76.32/76.67	10.1%	86.28/81.64	10.1%	74.99
$\text{BERMo}_{\text{BASE}}$			76.03/76.32		86.02/81.33		74.28
$\text{BERT}_{\text{BASE}}$	Soft Movement	12.2%	77.83/78.39	14.9%	87.6/83.23	8.4%	74.57
$\text{BERMo}_{\text{BASE}}$		12%	77.92/78.24	11.5%	87.49/83.21	8.5%	74.87

Table 4: Comparing accuracy/F1-score of $\text{BERT}_{\text{BASE}}$ and $\text{BERMo}_{\text{BASE}}$ for different pruning approaches

4.2.4 Knowledge Distillation

Many approaches complement the pruning strategies with the knowledge distillation to recover the loss in performance caused by pruning. We also show improvement in the performance by Knowledge Distillation, see Table 5. Our approach gives better results on SQuAD and comparable results for MNLI and QQP as compared to baseline with the optimal hyperparameters for the baseline (Contribution (v)).

Model	Remaining Weights	MNLI ACC-m / ACC-mm	QQP Acc./F1	SQuAD F1
$\text{BERT}_{\text{BASE}}$	10.1%	76.63/**77.46**	81.79/**86.58**	76.69
$\text{BERMo}_{\text{BASE}}$	10.1%	**76.82**/77.31	**81.86**/86.53	**76.85**

Table 5: Knowledge Distillation performance of $\text{BERT}_{\text{BASE}}$ and $\text{BERMo}_{\text{BASE}}$

5 Discussion

Summary: This work presents BERMo, a model architecture which uses the features from different layers to obtain a complex feature representation. Our work shows that the proposed model performs better than the baseline on the semantic tasks. Further, we find that the proposed model is more stable and converges faster (supports higher pruning rates) when subject to the compression techniques. We find our approach gives better parameter efficiency than the baseline on the QQP task for penalty based pruning approaches. Moreover, our approach is comparable to $\text{BERT}_{\text{BASE}}$ when tested with Knowledge Distillation.

Limitations: The skip connections introduced in the network require the activation maps from different layers to be stored which marginally increases the memory requirement at the time of inference. The reduction in the training time and stability of the training compensate for this limitation.

Future Directions: We choose a simple weighted linear combination scheme, as the premise was to test the applicability of our model in the resource constrained setup. Our work paves a way for exploring more complex skip connection having more parameters in the quest to improve the baseline performance. Further, we mix the features only in the last layer. We believe that such a summarization of features at each layer would benefit the model even further. Testing this approach would require pretraining the model from scratch as this scheme completely changes the activation maps at the input of each layer. Although, we evaluate our model against $\text{BERT}_{\text{BASE}}$, our approach could be extended to any Transformer based model as our methodology is agnostic to the choice of the backbone network. Moreover, as our method was able to converge in a fast and stable manner for pruning, we believe these trends should also hold during the pretraining phase. However, this needs to be extensively tested.

6 Related Work

6.1 Linear combination of features from different layers in BERT

Linear combination scheme of ELMo has been used to quantify where the linguistic information is captured within the network (Tenney et al. (2019)). The authors of this work call this approach scalar mixing weights and analyze these scalar weight to interpret the linguistic contribution of a particular layer. Concurrently, author's of (Kondratyuk & Straka (2019)) call these skip connections as Layer Attention and propose a model capable of accurately predicting universal part of speech, morphological features, lemmas, and dependency trees across 75 languages. The architectural modification presented in this work is similar to (Tenney et al. (2019); Kondratyuk & Straka (2019)), however, we intend to improve the baseline accuracies and enable faster and stable compression. In (Su & Cheng (2020)), the authors use global average pooling followed by two fully connected layers to obtain the scalar parameter associated with the network. The scale value is then used to compute the weighted average of the features from different layers. Although, we target improvement in the baseline performance in our work our approach is computationally simpler and requires fewer parameters.

6.2 Compression

Weight Pruning (Reed (1993); Liang et al. (2021)), knowledge distillation (Gou et al. (2021)), weight quantization (Gholami et al. (2021)), low rank matrix factorization (Nguyen et al. (2019)) are some approaches used to obtain the required level of compression. Several research (Sajjad et al. (2020); Gordon et al. (2020)) works tackle the issue by pruning the parameters of the network. In (Sanh et al. (2020)) the authors show using the gradient momentum information to select the pruning weights is superior to its counterpart which rely on the magnitude of weights. Certain works use quantization (Zafrir et al. (2019); Shen et al. (2020); Bai et al. (2020)) as a resort to compress the network. Knowledge Distillation (KD) Hinton et al. (2015) is leveraged in some works (Jiao et al. (2019); Sanh et al. (2019)) to bridge the gap between a compact model and BERT. Further, many works Mao et al. (2020); Tukan et al. (2020) use Low rank matrix factorization to deal with the issue. The authors of ROSITA (Liu et al. (2021)) outline a methodology to combine weight pruning, KD and low rank factorization. Our work is orthogonal to these as it makes architectural modification to the standard architecture to support high pruning rates and improve the training stability for the network.

7 Broader Impact

Research presented in this work improves the baseline performance on semantic tasks. This work highlights the importance of adding skip connections to the network in improving the training convergence and stability. We believe this work would act as a stepping stone and motivate further research in this direction, reducing the training time for these models. These improved training speeds also make room for enlarging the dataset size generally correlated with improvements in generalization performance. Further, as this work deals with reducing the training time for pruning, a possible application would be online pruning on resource constrained setup. Moreover, from an environmental perspective reducing training time will reduce the carbon footprint of these large language models.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*, 2020.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision – ECCV 2020*, Cham, 2020. Springer International Publishing.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention, 2021.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1356. URL `https://www.aclweb.org/anthology/P19-1356`.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Dan Kondratyuk and Milan Straka. 75 languages, 1 model: Parsing universal dependencies universally. *arXiv preprint arXiv:1904.02099*, 2019.

Tailin Liang, John Glossner, Lei Wang, and Shaobo Shi. Pruning and quantization for deep neural network acceleration: A survey. *arXiv preprint arXiv:2101.09671*, 2021.

Yuanxin Liu, Zheng Lin, and Fengcheng Yuan. Rosita: Refined bert compression with integrated techniques. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8715–8722, 2021.

Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Yaming Yang, Quanlu Zhang, Yunhai Tong, and Jing Bai. Ladabert: Lightweight adaptation of bert through hybrid model compression. *arXiv preprint arXiv:2004.04124*, 2020.

Luong Trung Nguyen, Junhan Kim, and Byonghyo Shim. Low-rank matrix completion: A contemporary survey. *IEEE Access*, 7:94215–94237, 2019.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://www.aclweb.org/anthology/N18-1202`.

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

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

Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. Poor man's bert: Smaller and faster transformer models. *arXiv e-prints*, pp. arXiv–2004, 2020.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20378–20389. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/eae15aabaa768ae4a5993a8a4f4fa6e4-Paper.pdf`.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8815–8821, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Ta-Chun Su and Hsiang-Chih Cheng. Sesamebert: Attention for anywhere. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 363–369. IEEE, 2020.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

Murad Tukan, Alaa Maalouf, Matan Weksler, and Dan Feldman. Compressed deep networks: Goodbye svd, hello robust low-rank approximation. *arXiv preprint arXiv:2009.05647*, 2020.

A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, AN Gomez, L Kaiser, and I Polosukhin. Attention is all you need. In *NIPS*, 2017.

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. *arXiv preprint arXiv:1804.07461*, 2018.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*, 2011.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*, 2019.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A Appendix

A.1 Hyperparameter

A.1.1 Standalone

For experiments on the probing task we chose the sequence length of 128 and train the models for 3 epochs with the batch size of 32. The learning rate of 5e-5. We use the same hyperparameters for SST-2, MNLI, and QQP. For SQuAD dataset we use 2 gpus cards and train the model with the sequence length of 384 for 3 epochs with a batch size of 16 (8 per gpu card). These hyperparameters were selected considering the memory limits of the GPU.

A.1.2 Stability for fine Pruning

Hyperparameter	Value
General Parameters	
number of gpus	1
per_gpu_train_batch_size	32
per_gpu_eval_batch_size	32
num_train_epochs	25
max_seq_length	128
learning_rate	3×10^{-5}
Pruning Parameters	
model_type	masked_bert masked_elbert
model_name	masked_bert_hot_25_0.1_trail_k masked_elbert_hot_25_0.1_trail_k
Pruning Parameters	
warmup_steps	2100
mask_scores_learning_rate	1×10^{-2}
initial_threshold	1
final_threshold	0.10
initial_warmup	1
final_warmup	1
pruning_method	topK
mask_init	constant
mask_scale	0

Table 6: The hyperparameter for experiments in Section 4.2.1. For hyperparameter description refer to `https://github.com/huggingface/block_movement_pruning`. The hyperparameters not mentioned in the table are kept the same as there default.

For our simulations on SST-2 for stability of the fine pruning stage we use the hyperparameters shown in Table 6.

A.1.3 Convergences of fine pruning

For our simulations on convergence for MNLI, QQP and SQuAD of the fine pruning stage we use the hyperparameters shown in Table 7.

A.1.4 Different Pruning Approaches

For movement purning we adopt the hyperparameter presented in Table 7. The experiment on Magnitude, L_0 and soft movement pruning use the hyperparameters shown in Table 8, 9 and 8 respectively.

Hyperparameter	Value		
General Parameters			
	MNLI	QQP	SQuAD
number of gpus	1	1	2
per_gpu_train_batch_size	32	32	8
per_gpu_eval_batch_size	32	32	8
max_seq_length	128	128	384
learning_rate	3×10^{-5}	3×10^{-6}	3×10^{-5}
Pruning Parameters			
model_type	masked_bert masked_elbert		
model_name	masked_bert_hot_epochs_k_0.1 masked_elbert_hot_epochs_k_0.1		
Pruning Parameters			
warmup_steps	12000	11000	5400
mask_scores_learning_rate	1×10^{-2}		
initial_threshold	1		
final_threshold	0.10		
initial_warmup	1	2	1
final_warmup	1	3	2
pruning_method	topK		
mask_init	constant		
mask_scale	0		

Table 7: The hyperparameter for experiments in Section 4.2.2.

Hyperparameter	Value		
General Parameters			
	MNLI	QQP	SQuAD
number of gpus	1	1	2
per_gpu_train_batch_size	32	32	8
per_gpu_eval_batch_size	32	32	8
num_train_epochs	6	10	10
max_seq_length	128	128	384
learning_rate	3×10^{-5}		
Pruning Parameters			
model_type	masked_bert masked_elbert		
model_name	masked_bert_hot_epochs_k_0.1 masked_elbert_hot_epochs_k_0.1		
Pruning Parameters			
warmup_steps	12000	11000	5400
mask_scores_learning_rate	1×10^{-2}		
initial_threshold	1		
final_threshold	0.10		
initial_warmup	1	2	1
final_warmup	1	3	2
pruning_method	Magnitude		

Table 8: The hyperparameter for Magnitude pruning experiments in Section 4.2.3.

Hyperparameter	Value		
General Parameters			
	MNLI	QQP	SQuAD
number of gpus	1	1	2
per_gpu_train_batch_size	32	32	8
per_gpu_eval_batch_size	32	32	8
num_train_epochs	6	10	10
max_seq_length	128	128	384
learning_rate	3×10^{-5}		
Pruning Parameters			
model_type	masked_bert masked_elbert		
model_name	masked_bert_hot_epochs_k_0.1 masked_elbert_hot_epochs_k_0.1		
Pruning Parameters			
warmup_steps	12000	11000	5400
mask_scores_learning_rate	1×10^{-1}	1×10^{-2}	1×10^{-1}
initial_threshold	1		
final_threshold	1		
initial_warmup	1		
final_warmup	1		
pruning_method	l0		
mask_init	constant		
mask_scale	2.197		
regularization	l0		
final_lambda (BERT$_{\text{BASE}}$)	50	50	175
final_lambda (BERMo$_{\text{BASE}}$)	45	50	175

Table 9: The hyperparameter for L_0 pruning experiments in Section 4.2.3.

Hyperparameter	Value		
General Parameters			
	MNLI	QQP	SQuAD
number of gpus	1	1	2
per_gpu_train_batch_size	32	32	8
per_gpu_eval_batch_size	32	32	8
num_train_epochs	6	10	10
max_seq_length	128	128	384
learning_rate	3×10^{-5}	3×10^{-6}	3×10^{-5}
Pruning Parameters			
model_type	masked_bert masked_elbert		
model_name	masked_bert_hot_epochs_k_0.1 masked_elbert_hot_epochs_k_0.1		
Pruning Parameters			
warmup_steps	12000	11000	5400
mask_scores_learning_rate	1×10^{-2}		
initial_threshold	0		
final_threshold	0.10		
initial_warmup	1	2	1
final_warmup	1	3	2
pruning_method	sigmoied_threshold		
mask_init	constant		
mask_scale	0		
regularization	l1		
final_lambda (BERT$_{\text{BASE}}$)	200	150	500
final_lambda (BERT$_{\text{BASE}}$)	200	150	500

Table 10: The hyperparameter for soft movement pruning experiments in Section 4.2.3.

Hyperparameter	Value		
General Parameters			
	MNLI	QQP	SQuAD
number of gpus	1	1	2
per_gpu_train_batch_size	32	32	8
per_gpu_eval_batch_size	32	32	8
max_seq_length	128	128	384
learning_rate	3×10^{-5}	3×10^{-6}	3×10^{-5}
Pruning Parameters			
model_type	masked_bert masked_elbert		
model_name	masked_bert_hot_epochs_k_0.1 masked_elbert_hot_epochs_k_0.1		
Pruning Parameters			
warmup_steps	12000	11000	5400
mask_scores_learning_rate	1×10^{-2}		
initial_threshold	1		
final_threshold	0.10		
initial_warmup	1	2	1
final_warmup	1	3	2
pruning_method	topK		
mask_init	constant		
mask_scale	0		
Distillation parameters			
teacher_type	bert		
teacher_name_or_path	fine tuned bert directory for respective task		
alpha_ce	0.1		
alpha_distill	0.9		

Table 11: The hyperparameter for experiments in Section 4.2.4.

A.1.5 KNOWLEDGE DISTILLATION

For our simulations on Knowledge distillation on MNLI, QQP and SQuAD in the fine pruning stage we use the hyperparameters shown in Table 11.

A.2 DIFFERENT TRIALS

The results for 5 different runs for the probing task on $\text{BERT}_{\text{BASE}}$ and $\text{BERMo}_{\text{BASE}}$ can be found in Table 12.

Model	Surface		Syntactic			Semantic				
	Sentlen	WC	TreeDepth	TopConst	Bshift	Tense	Subjnum	Objnum	Somo	CoordInv
Trial 1 (Seed 63)										
$\text{BERT}_{\text{BASE}}$	99.15	84.14	72.56	77.09	96.29	84.33	86.38	90.51	61.79	82.83
$\text{BERMo}_{\text{BASE}}$	99.23	83.39	73.14	76.74	96.35	87.28	90.55	94.53	64.46	83.60
Trial 2 (Seed 77)										
$\text{BERT}_{\text{BASE}}$	99.22	84.24	72.9	77.38	96.35	80.78	86.89	90.51	61.72	83.25
$\text{BERMo}_{\text{BASE}}$	99.24	83.38	72.34	76.92	96.32	88.17	90.13	93.57	63.88	83.15
Trial 3 (Seed 9)										
$\text{BERT}_{\text{BASE}}$	99.23	Diverges	73.12	77.33	96.13	83.54	88.94	91.32	61.03	82.86
$\text{BERMo}_{\text{BASE}}$	92.23	83.51	71.21	76.76	96.21	87.07	90.23	92.54	62.74	83.47
Trial 4 (Seed 96)										
$\text{BERT}_{\text{BASE}}$	99.12	84.11	72.72	76.81	96.27	81.98	87.19	92.01	62.35	83.21
$\text{BERMo}_{\text{BASE}}$	99.22	83.55	72.16	76.79	96.27	87.18	90.43	94.29	64.69	82.49
Trial 5 (Seed 39)										
$\text{BERT}_{\text{BASE}}$	99.27	Diverges	72.55	76.72	96.06	82.24	87.76	89.21	60.68	82.60
$\text{BERMo}_{\text{BASE}}$	99.25	83.13	72.95	77.05	96.36	86.40	89.72	93.47	64.58	83.10
Mean over 5 trails										
$\text{BERT}_{\text{BASE}}$	99.20	84.16	**72.77**	**77.07**	96.22	82.57	87.43	90.71	61.51	82.95
$\text{BERMo}_{\text{BASE}}$	**99.24**	83.39	72.36	76.85	**96.30**	**87.22**	**90.21**	**93.68**	**64.27**	**83.16**
Standard Deviation										
$\text{BERT}_{\text{BASE}}$	0.0596	0.0681	0.2421	0.2975	0.1204	1.3874	0.9799	1.0479	0.6609	0.2750
$\text{BERMo}_{\text{BASE}}$	0.0114	0.1641	0.7615	0.1310	0.0622	0.6330	0.3205	0.7827	0.4306	0.4307

Table 12: Detailed accuracy comparison between $\text{BERT}_{\text{BASE}}$ and $\text{BERMo}_{\text{BASE}}$ on the probing task from the SentEval dataset.