Task-specific Compression for Multi-task Language Models using Attribution-based Pruning

Nakyeong Yang[1], **Yunah Jang**[1], **Hwanhee Lee**[2], **Seohyeong Jung**[3] **and Kyomin Jung**[1]
[1]Seoul National University, [2]Chung-Ang University, [3]Hyundai Motor Group and 42dot Inc
{yny0506, vn2209, kjung}@snu.ac.kr
hwanheelee@cau.ac.kr, seohyeong.jeong@42dot.ai

Abstract

Multi-task language models show outstanding performance for various natural language understanding tasks with only a single model. However, these language models utilize an unnecessarily large number of model parameters, even when used only for a specific task. This paper proposes a novel training-free compression method for multi-task language models using a pruning method. Specifically, we use an attribution method to determine which neurons are essential for performing a specific task. We task-specifically prune unimportant neurons and leave only task-specific parameters. Furthermore, we extend our method to be applicable in low-resource and unsupervised settings. Since our compression method is training-free, it uses few computing resources and does not destroy the pre-trained knowledge of language models. Experimental results on the six widely-used datasets show that our proposed pruning method significantly outperforms baseline pruning methods. In addition, we demonstrate that our method preserves performance even in an unseen domain setting.

1 Introduction

Various pre-trained language models with large-scale data and parameters have emerged (Devlin et al., 2018; Lewis et al., 2019; Raffel et al., 2019; Brown et al., 2020). Specifically, pre-trained language models like T5 (Raffel et al., 2019) and GPT-3 (Brown et al., 2020) have shown outstanding performance on many natural language understanding tasks. These language models can perform various tasks with a single model by treating every text processing problem as a text generation problem. However, these language models may utilize unnecessary large-scale model parameters even when performing only a specific task. Previous works have introduced various compression methods for language models such as pruning (Chen et al., 2020; Goyal et al., 2020; He et al., 2021), knowledge distillation (Sanh et al., 2019; Hou et al., 2020; Mao et al., 2020; Sun et al., 2020), quantization (Shen et al., 2020), and low-rank factorization (Liu et al., 2021). However, these studies have (1) not compressed the language models task-specifically or (2) demanded an additional training process like the case of knowledge distillation. This additional training process requires excessive computing resources and a massive training dataset. Furthermore, this training process can destroy inherent pre-trained knowledge in language models since it updates the model's pre-trained parameters (Toneva et al., 2018). Due to the catastrophic forgetting (McCloskey and Cohen, 1989) caused by pre-trained knowledge destruction, models which are compressed and trained for a specific task, tend to show degraded performance on solving other pre-trained tasks (Kirkpatrick et al., 2017; Ritter et al., 2018). Also, additional memory space is required to store the trained parameters separately.

In this paper, we propose a novel training-free attribution-based task-specific pruning method that enables more efficient compression and inference by extracting only task-specific parameters from multi-task language models. We can determine which neurons are essential to derive a specific output for each neural network layer by using attribution so that we can extract only task-specific parameters from the entire model, as shown in Figure 1. We can efficiently process input data while preserving the model's task performance by selecting only the important neurons determined by the attribution method. Furthermore, we extend our method to be applicable in two challenging scenarios: low-resource and unsupervised scenarios. The former alleviates insufficient labeled data situations, and the latter handles settings when labels are unavailable. Both methods can relieve the cost of obtaining labeled datasets, which requires excessive human resources and is time-consuming. Especially under the low-resource setting, our attribution-based task-

Findings of the Association for Computational Linguistics: EACL 2023, pages 594–604

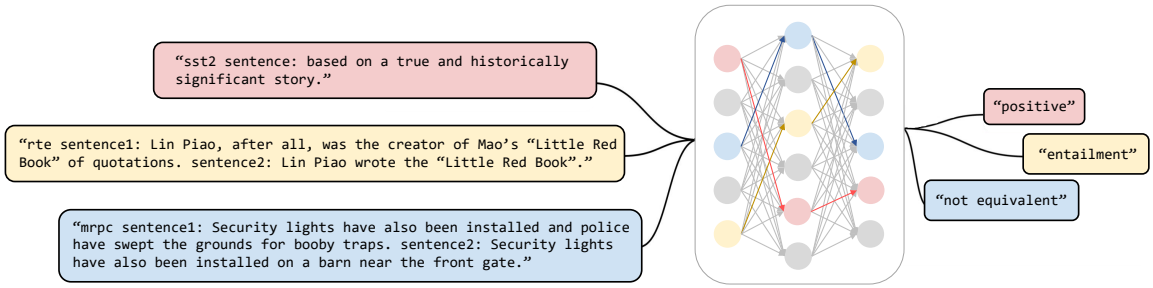


Figure 1: Task-specific Knowledge of Multi-task Language Models. Not all parameters in a language model behave as important parameters when performing a single task. For example, in this figure, when a language model receives SST-2 data, sentiment analysis data, only the parameters expressed in red color behave as essential parameters.

specific pruning requires only a single forward and backward propagation computation for few-shot data samples (e.g., only ten samples) to derive attribution of each neuron. Since this pruning process does not update the model's parameters, it does not destroy the pre-trained knowledge of the language models. Therefore, it is irrelevant to the various disadvantages that arise during an additional training process. Since our method is model-agnostic, it can be applied to any neural network model broadly and generally. Even we can use it to extract only task-specific knowledge after other compression methods are applied.

Experimental results on the six widely-used natural language understanding tasks show that our proposed method significantly outperforms baseline training-free pruning methods. Furthermore, we demonstrate that our method shows robust performance in both low-resource and unsupervised settings. Also, we reveal that our proposed method shows outstanding knowledge preservation even for an unseen related domain, which suggests that our method can preserve task-specific knowledge effectively. We additionally investigate to offer a guideline for our task-specific compression method by analyzing which types of layers are significant for processing task-specific knowledge.

2 Related Works

2.1 Efficient Language Models

As transformer-based (Vaswani et al., 2017) language models (Devlin et al., 2018; Radford et al., 2018; Raffel et al., 2019; Liu et al., 2019; Yang et al., 2019) have become state-of-the-arts on many NLP tasks in the last few years, deep neural network model compression methods have been vastly applied to large-scale language models. Fan et al. (2019) randomly drops layers at training time, which enables structured pruning on transformer layers at inference time. Michel et al. (2019) prunes less important attention heads at inference time. Other works (Goyal et al., 2020; Kim et al., 2021) focus on pruning less important tokens and progressively remove them during inference. However, many of the pruning methods (Goyal et al., 2020; Kim et al., 2021; Chen et al., 2020) require a following fine-tuning step of the model parameters after fixing the configuration of a pruned network, which makes such methods undesirable for efficient task-specific compression.

On the knowledge distillation side, Sun et al. (2019); Jiao et al. (2019); Sanh et al. (2019) employ teacher-student framework (Hinton et al., 2015) to transfer knowledge from an original large model (teacher), to a lightweight shallow model (student). They differ in how the student network is initialized and to which components knowledge distillation is applied. On the other hand, Shen et al. (2020) uses the mixed precision group-wise quantization based on Hessian information to compress BERT.

There are other streams of works that explore efficient language models by solving the bottleneck of the Transformer-based model computation. Beltagy et al. (2020) and Zaheer et al. (2020) sparsify the attention matrix to make transformer-based language models more efficient and Wang et al. (2020) applies low-rank approximation to increase inference speed. However, such works sparsify the full self-attention matrix according to attention score, which does not directly reduce the dimension of the matrices in the model such as query, key, value, and feed-forward matrices.

2.2 Network Pruning

One of the ways to categorize network pruning is to compare structured pruning to unstructured pruning. For structured pruning (Li et al., 2016; Hu et al., 2016; Wen et al., 2016), groups of weight con-

nections are removed from a network together, such as entire channels or filters in CNN-based networks and layers or attention heads in transformer-based networks. For unstructured pruning (Han et al., 2015a,b), weight connections are removed from a network individually. However, unstructured pruning methods produce large sparse weight matrices which are computationally inefficient unless equipped with a specifically designed hardware. In this paper, we utilize the structured pruning method to propose a compression method that enables efficient weight matrix multiplication computation.

2.3 Attribution Method

We utilize an attribution method (Shrikumar et al., 2016) to extract the importance of neurons from the pre-trained language models. Attribution methods are mostly used to derive important features *(i.g., pixel, token)* to extract interpretability from deep neural networks (Baehrens et al., 2010; Springenberg et al., 2014; Shrikumar et al., 2016). Specifically, attribution methods are used to compute the importance of each feature for performing a specific task. Formally, suppose we have a function $\mathcal{P} : \mathbb{R}^d \rightarrow [0, 1]^m$ that represents deep neural networks for multi-class classification. The contribution of the i-th feature in x to the prediction of c-th class using $\mathcal{P}$ is defined as follows:

$$A_i^{(x,c)}(x) = x_i \times \frac{\partial \mathcal{P}(c|x)}{\partial x_i} \quad (1)$$

where $\partial \mathcal{P}(c|x)/\partial x_i$ is the gradient of $\mathcal{P}(c|x)$ with respect to the i-th feature.

3 Methodologies

In this section, we describe our attribution-based pruning method for extracting only the task-specific knowledge from a multi-task language model T5 (Raffel et al., 2019), where attribution is obtained using gradient information. Furthermore, we extend our method to low-resource and unsupervised settings to alleviate insufficient labeled data situations. We select T5 because it is a multi-task solving model and can be used in any natural language understanding setting by treating every text processing problem as a text generation problem. For our problem setting, suppose we have input text $x = \{x_1, ..., x_n\}$ and output text $y = \{y_1, ..., y_m\}$ mapped as $(x, y) \in \mathcal{D}$, where each text corresponds to a sequence of tokens, and an input text contains a prefix task description. We can represent a standard conditional language modeling objective to maximize the following likelihood:

$$\mathcal{L}(x, y) = \sum_i \log \mathcal{P}(y_i|x, y_1, ..., y_{i-1}; \Theta) \quad (2)$$

where the conditional probability $\mathcal{P}$ is modeled using a neural network with parameters Θ.

3.1 Task-specific Knowledge Extraction

Applying Pruning for Transformer variants Deep neural networks can be compressed by pruning unimportant i-th neurons of the layer representation h (Han et al., 2015a,b). The architecture of Transformer-based models mainly consists of multi-head attentions and fully connected feed-forward networks as follows.

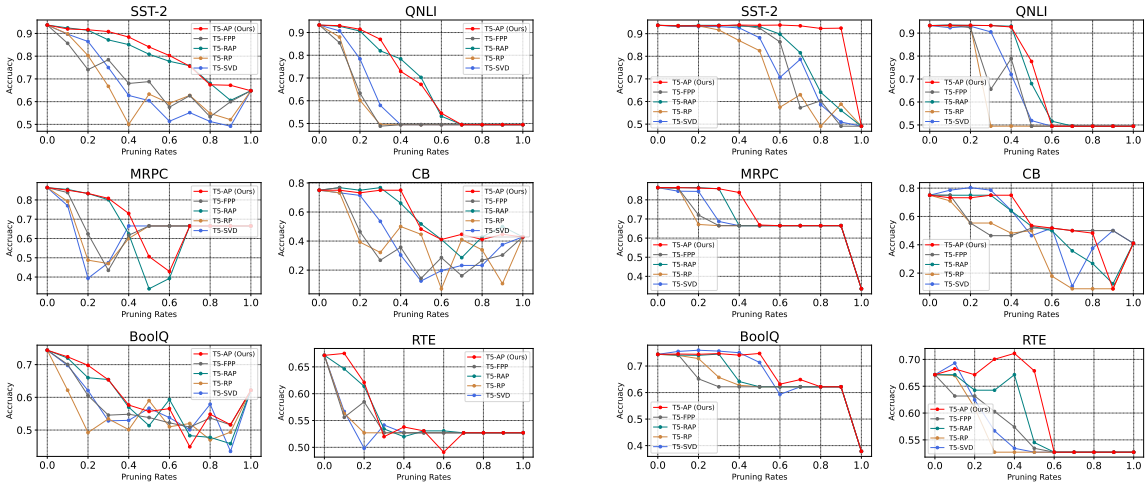
$$\begin{gathered} MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \\ FFN(x) = \sigma(xW_1 + b_1)W_2 + b_2 \end{gathered} \quad (3)$$

where $W_i^{Q,K,V} \in \mathbb{R}^{d_{model} \times d_{q,k,v}}$ and $W_i^O \in \mathbb{R}^{d_v \times d_{model}}$ are the projection matrix parameters for multi-head attentions. For the fully connected feed-forward network (FFN), two linear transformations, denoted with the projection matrix parameters W_1 and W_2 and biases b_1 and b_2, with an activation function are used. Transformer (Vaswani et al., 2017) variants can be compressed by pruning $W^{Q,K,V,O}$, $W_{1,2}$, and $b_{1,2}$ for each transformer block.

Deriving Attribution for Language Models Language models generate text outputs by iteratively selecting a word-piece from the vocabulary dictionary. Therefore, the text generation process can be seen as a classification task dealt with in the attribution methods, and we can apply the attribution methods to compute the importance of features for language models. However, the purpose of this study is to derive the importance of each neuron h_i in the layer representation $h \in \mathbb{R}^d$, rather than deriving the importance for the input feature x_i. Hence, the attribution formula is adapted to compute a neuron attribution $A_i^{(x,y_j)} \in \mathbb{R}$ as follows:

$$A_i^{(x,y_j)}(h) = h_i \times \frac{\partial \mathcal{P}(y_j|x, y_{1:j-1})}{\partial h_i} \quad (4)$$

If the target output text consists of multiple word-pieces rather than a single word-piece, language



(a) Encoder Pruning Results

(b) Decoder Pruning Results

Figure 2: Module-specific Pruning Results. Our proposed attribution-based pruning significantly outperforms the other pruning methods in most cases. Especially, our task-specific pruning is more effective on decoder compression; these results suggest that most task-specific knowledge exists in the decoder of language models. The standard deviations of T5-RP and T5-RAP are shown in appendix B.

models must derive the multiple word-piece output distributions. Therefore, we change the attribution formula to handle multiple word-piece outputs as follows:

$$A_i^{(x,y)}(h) = h_i \times \sum_{j=1}^{|y|} \frac{\partial \mathcal{P}(y_j|x, y_{1:j-1})}{\partial h_i} \quad (5)$$

Since $A_i^{(x,y)}$ is attribution for one sample data x, we obtain the final neuron attribution by summing attributions for multiple sample data as shown in the following formula:

$$A_i^{(\mathcal{D})}(h) = \sum_{(x,y)\in\mathcal{D}} A_i^{(x,y)}(h) \quad (6)$$

where $\mathcal{D}$ means the entire task-specific dataset. In low-resource environments, few-shot samples can be used for $\mathcal{D}$ (e.g., only ten samples), which are sufficient to derive a precise importance score for each neuron. Experimental results for low-resource setting are described in section 4.3.

Attribution-based Layer Pruning We focus on applying attribution-based pruning on the Transformer encoder and decoder, more specifically on multi-head attention and fully connected feed-forward networks. We use neuron attribution $A_i^{(\mathcal{D})}$ as the importance for each neuron of a specific layer. We sort the importance of each neuron in order of magnitude at each layer, and we can compress the model by pruning neurons with lower importance.

$$argsort_i(A) = |\{j|(A_i < A_j) \cup (A_i = A_j, j < i)\}| \quad (7)$$
$$\textbf{where } i, j \in \{1, ..., k\}$$

Once neurons are sorted according to the importance score, we prune neurons from each layer with the pruning rate p by constructing a set $\mathcal{M}$ of neuron indices to be secured.

$$\mathcal{M} = \{i|argsort_i(A) < \lfloor k \times p \rfloor\} \quad (8)$$
$$\textbf{where } i \in \{1, ..., k\}$$

The algorithm for deriving a set $\mathcal{M}$ is shown in appendix A. Suppose $W \in \mathbb{R}^{d\times k}$ is a linear matrix multiplication parameter we want to prune, the matrix after pruning is denoted as $\tilde{W} = (W_{ij})_{\substack{1\le i\le d \\ j\in\mathcal{M}}}$. If the bias term $b \in \mathbb{R}^k$ is added to the operation for an affine transformation, the bias term can also be compressed by performing the $\tilde{b} = (b_i)_{i\in\mathcal{M}}$ operation similarly. The compressed parameters are used to compute the new representation by performing the transformation operation $h\tilde{W}$ or $h\tilde{W} + \tilde{b}$.

More specifically, for W_i^Q, W_i^K, and W_i^V from eq. (3), second dimension (the number of columns) of the matrix is pruned and for W_i^O, W_1, and W_2, the first dimension (the number of rows) is pruned to preserve the original architecture by matching shape with input processed from the previous layer. After pruning, multi-head attention and fully connected feed-forward network computations are precisely the same as before but with the pruned weight matrices:

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)\tilde{W}^O$$
$$head_i = Attention(Q\tilde{W}_i^Q, K\tilde{W}_i^K, V\tilde{W}_i^V)$$
$$FFN(x) = \sigma(x\tilde{W}_1 + b_1)\tilde{W}_2 + \tilde{b}_2 \quad (9)$$

Note that attribution scores are sorted locally within each layer, and the pruning rate p is applied to each prunable layer uniformly.

Our proposed compression process utilizes a structured pruning without any training process. Therefore, our method can conduct on-demand real-time task-specific compression and inference for each task while preserving pre-trained parameters. The detailed algorithm for on-demand real-time task-specific compression and inference is shown in appendix A.

3.2 Unsupervised Pruning

Obtaining labeled data usually requires excessive human resources and is time-consuming. Therefore, we propose an additional method to derive attributions in an unsupervised setting to mitigate this problem. If the label for the dataset is given, we can simply compute attribution by summing the gradients values for the word-piece set composing the label. However, when the label is not given, the target word-piece set is ambiguous. To resolve this problem, we compute task-specific importance by summing the absolute values of attributions for all candidate labels as follows:

$$A_i^{(x,\mathcal{Y})}(h) = \sum_{y\in\mathcal{Y}} |h_i \times \sum_{j=1}^{|y|} \frac{\partial \mathcal{P}(y_j|x, y_{1:j-1})}{\partial h_i}| \quad (10)$$

where $\mathcal{Y}$ is the candidate label set. The above importance computation formula does not require supervision for any data. Hence, we may not reflect definite label information when computing each neuron's importance under our unsupervised compression setting. However, this setting is helpful for a resource-constrained environment, where obtaining labeled data is challenging.

4 Experiments

4.1 Experimental Setup

Datasets We conduct experiments on six downstream tasks (Wang et al., 2018, 2019). Specifically, we utilize SST-2 (sentiment analysis); MRPC (semantic textual similarity); BoolQ (question answering); and QNLI, CB, RTE (natural language inference).

Implementation Details We select pre-trained *T5-base*[1] as a backbone for the following experiments. *T5-base* consists of 12 encoder and 12 decoder layers. Each encoder layer contains 6 prunable matrices: 4 for the multi-head self-attention networks and 2 for the feed-forward networks. Each decoder layer contains 10 prunable matrices: 4 for the multi-head self-attention networks and 2 for the feed-forward networks, and 4 for the cross-attention networks. *T5-base* used in our experiments has been fine-tuned by multi-task learning using the six datasets above. We experiment with pruning rates ranging from 0.1 to 1.0, and a pruning rate is applied to each prunable layer uniformly.

4.2 Task-specific Pruning Efficiency

In this section, we validate the effectiveness of our task-specific attribution-based pruning by comparing the performance with other pruning methods. We collect compressed models using various pruning methods and evaluate the model's performance on testset for all six datasets.

Baselines We select four other training-free pruning methods to compare with our task-specific **T5 Attribution Pruning (T5-AP)**.

- **T5 Forward Propagation Pruning (T5-FPP)** derives the importance of each neuron with the absolute value of the forward propagation value of each neuron. This method is widely used to compress model in various studies (Han et al., 2015b; Hu et al., 2016; Li et al., 2016). Previous studies using FPP generally fine-tune the compressed model to increase the model's performance. However, we eliminate the fine-tuning process to maintain a fair evaluation scenario since we focus on studying training-free compression.

- **T5 Low Rank Factorization (T5-SVD)** prunes weight matrices of neural networks using Singular Value Decomposition (SVD). SVD is commonly used as a main matrix compression idea in various researches (Wang et al., 2020; Noach and Goldberg, 2020). Specifically, SVD is used to compress a ma-

[1] https://huggingface.co/t5-base

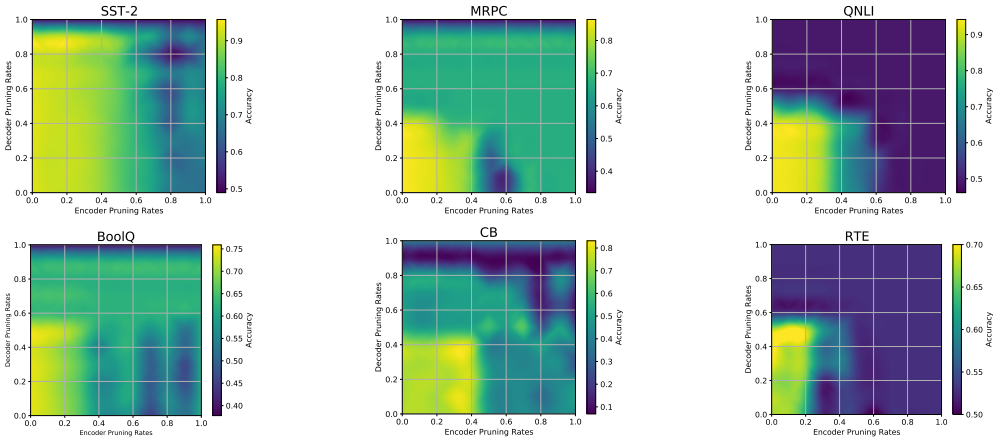


Figure 3: Module-integrated Pruning Results. These results reveal that compressing the whole architecture of the model does not additionally degrade the model's performance compared to module-specific pruning. We experiment with the combinations of ten pruning rates for the encoder and decoder, and plot the interpolated results.

trix based on low rank factorization formula as follows:

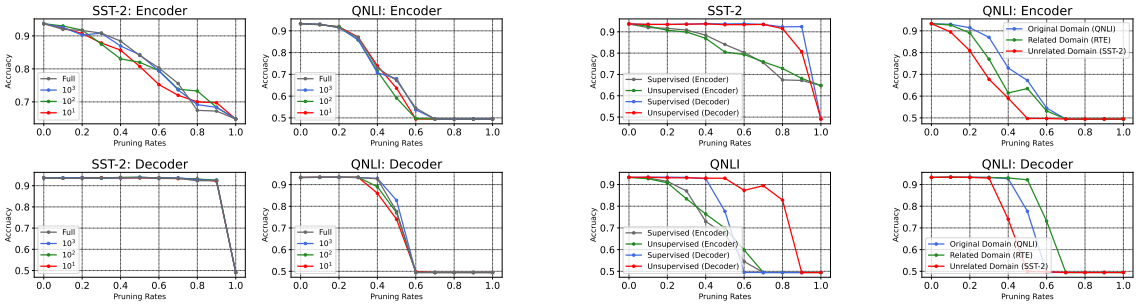
$$W = U\Sigma V \approx \sum_{j=1}^{r} \sigma_j \times (U_j \times V_j) \quad (11)$$

where $W \in \mathbb{R}^{d\times k}$ is a matrix to compress, and $U \in \mathbb{R}^{d\times r}$ and $V \in \mathbb{R}^{r\times k}$ are the decomposed matrices. $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_r)$ is a diagonal matrix consisting of the singular values σ_i, where $r \leq min(d, k)$ is the matrix rank. U_j is the j-th column of U and V_j is the j-th row of V. We can compress the matrices of T5 by determining the rank $r = \lfloor \frac{d\times k\times p}{d+k+1} \rfloor$ to have the same number of parameters as T5-AP, where p is the pruning rate defined in formula 8.

- **T5 Random Attribution Pruning (T5-RAP)** randomly selects word-pieces that are not label, and uses them to compute attribution. RAP does not derive appropriate task-specific importance for each neuron since this method randomly selects word-pieces output. We calculate the final performance of T5-RAP by averaging the accuracy derived from five trials of random word-pieces selection.
- **T5 Random Pruning (T5-RP)** randomly selects which neuron to prune. This method can achieve the lower-bound performance of overall training-free pruning methods since it randomly selects which neuron to prune without any knowledge. We calculate the final performance of T5-RP by averaging the accuracy derived from five trials of random pruning.

Module-specific Pruning For each dataset, we separately compressed the encoder and decoder at varying pruning rates to reveal the effect of our method on the encoder and decoder, respectively. Figure 2 shows the experimental results for five compression methods, including our proposed method. Experimental results show that our method outperforms other compression methods in most cases. Specifically, there is almost no performance difference between the T5-RP and T5-FPP. These results suggest that the T5-FPP does not extract task-specific knowledge. In addition, T5-SVD performs not badly in some cases, but generally performs similarly to T5-RP. It is because the low-rank approximation of T5-SVD does not work task-specifically. Surprisingly, T5-RAP sometimes performs similarly to T5-AP, probably due to the use of partial gradients information calculated from model parameters. Our experiments show that the decoder part of T5 has the robustness for task-specific compression than the encoder part of T5. These results demonstrate that T5 decoder processes more task-specific information than T5 encoder.

Module-integrated Pruning To maximize the compression efficiency of a language model, we should compress the whole model instead of compressing the encoder or decoder, respectively. Therefore, we also validate our method by compressing the whole architecture of T5. Figure 3 shows the experimental results of simultaneously compressing both the encoder and decoder using our method. These experimental results reveal that compressing the whole architecture of the model, not compressing each encoder or decoder sepa-



(a) Low Resource Setting (b) Unsupervised Setting (c) Unseen Domain Setting

Figure 4: Experimental results extending our pruning method to challenging settings: (a) Low-resource setting experiment results. (b) Unsupervised setting experiment results. (c) Unseen domain setting experiment results. These extensions make our method more practical for use in a real-world setting.

rately, does not degrade the model's performance additionally.

Our method focuses on compressing a multi-task language model without any additional training process in a model-agnostic way. Therefore, it is difficult to compare our method with previous compression research due to the inconsistent experimental setting since previous studies have treated training-based and model-specific compression methods. Since our method is model-agnostic, it can be utilized broadly and generally to prune multi-task language models containing only task-specific knowledge after applying other compression methods.

4.3 Low-resource Setting

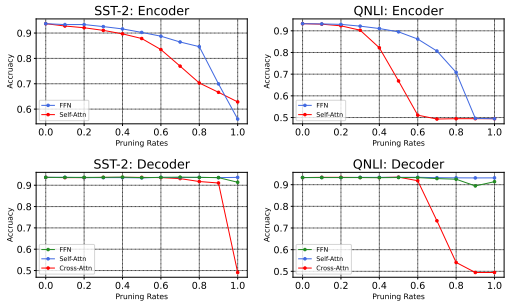
In this section, we demonstrate the results for compressing language models based on the attribution computed from only few-shot. Specifically, we compute neuron importance using only 10^3 and 10^2, and 10^1 samples of SST-2 and QNLI datasets and prune the T5 model with the computed importance, where we balance the number of samples for each class when sampling a subset of the whole dataset. All results are reported by averaging five trials of random sampling. Figure 4-(a) represents the pruning results in low-resource setting. For SST-2 dataset, we find that compression using only 10^1 data samples yields comparable performance to the results of using the entire training dataset. The total number of data samples of SST-2 is 67k, and 10^1 of data samples corresponds to about only 10^{-4} of the whole dataset. For the QNLI dataset, we demonstrate that compression using only 10^3 data samples of the labeled training dataset yields comparable performance to the results of using the entire training dataset. Furthermore, the performance degradation is also insignificant when using only 10^1 samples of the labeled QNLI training dataset. The total number of data samples of QNLI is 105k, and 10^3 and 10^1 data samples correspond to about only 10^{-2}, 10^{-4} of the whole dataset, respectively. These results suggest that most of the task-specific knowledge is derived from computing gradients for only the candidate outputs. We can effectively reduce the time consumption in this low-resource setting by using a few labeled instances to compute the attribution, and it is the most significant advantage over other training-based compression methods.

4.4 Unsupervised Setting

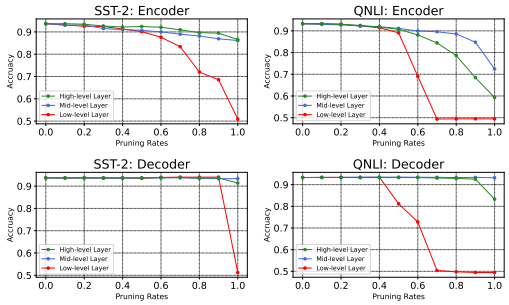
We suggest an additional method to compute attributions using an unlabeled text dataset in section 3.2. We present the pruning results by computing attributions for an unsupervised setting in Figure 4-(b). Results of encoder compression with the unsupervised setting for both SST-2 and QNLI datasets show competitive scores to that of labeled data. For the decoder, the performance of SST-2 decreases slightly, but the performance of QNLI rather increases. The experimental result on SST-2 reveals that the compression in an unsupervised setting shows robust performance maintenance. In the QNLI result, we observe that computing attributions using information from all output candidates enhances the model's performance.

4.5 Unseen Domain Setting

In this section, we validate the effect of our task-specific compression on unseen domains. We compress the T5 using related and unrelated datasets, and then compare the performance preservation for the original dataset. Specifically, we compress the T5 using attribution computed with SST-2 and RTE, respectively. And then, we evaluate the compressed models with the QNLI dataset. QNLI and RTE are



(a) Layer Type-specific Pruning



(b) Layer Depth-specific Pruning

Figure 5: Layer types analysis: (a) Layer architecture experiment results. (b) Layer depth experiment results. The higher the degradation, the more essential layers are.

related domains since both are natural language inference datasets, and SST-2 is an unrelated domain built for sentiment analysis. Figure 4-(c) shows the evaluation results of the compressed model for related and unrelated domains. Experimental results reveal our method's robust performance maintenance for the related domain. Surprisingly the case of decoder compression shows even better performance maintenance in the related domain than in the original domain.

4.6 Layer-specific Pruning Analysis

This section further investigates the pruning effect per layer type. We select two pruning settings: (1) Layer type-specific and (2) Layer depth-specific.

Layer Type-specific Pruning Analysis Layer type-specific pruning analysis focuses on understanding how the performance of the model varies depending on the type of compressed layers. The encoder investigates pruning results for feed-forward neural networks and self-attention networks, and the decoder focuses on feed-forward neural networks, self-attention networks, and cross-attention networks.

Layer Depth-specific Pruning Analysis Layer depth-specific pruning analysis investigates how the performance of the model changes depending on the depth of the compressed layers. We select SST-2 for experiments and separate each encoder and decoder into three parts: (1) Low-level layer, (2) Mid-level layer, and (3) High-level layer. Since *T5-base* consists of 12 layers for each encoder and decoder, each depth consists of 4 layers.

Layer-specific pruning results are shown in Figure 5. For the encoder, self-attention networks are more critical for preserving the performance than feed-forward neural networks. For the decoder, cross-attention networks are more important than feed-forward neural networks and self-attention networks. For each layer-depth, we can conclude that the low-level features are more crucial to preserving the model's performance. Especially, experimental results reveal that the model's performance is preserved even if the pruning rate of a specific layer is 1.0. These results demonstrate that there is redundant information processing between layers for performing a specific task. Note that although the pruning rate is 1.0 for a layer, the representation propagated through the pruned layer does not lose every knowledge completely. It is because transformer variants have residual connections to preserve the knowledge of previous layers.

5 Conclusion

This paper proposes a novel training-free attribution-based task-specific knowledge extraction method for multi-task language models. Specifically, we use attribution to determine which neurons are important to derive a specific output for each task. Then, we prune task-specifically unimportant neurons to extract only task-specific knowledge from the entire model. We further propose a method for computing attributions in low-resource and unsupervised settings. We demonstrate that our method outperforms the other pruning methods on the widely used text datasets. In addition, we examine that our task-specific language model pruning method shows outstanding performance in the unseen domain, especially when the unseen domain is related to the dataset used to configure the compressed version. Our compression method does not update the pre-trained parameters of the language models, which enables efficient on-demand compression and inference. Also, our proposed method is valuable because it can be universally applied to any neural network-based model architecture.

Limitations

To the best our knowledge, this is the first work to compress a multi-task language model without extra training on the target task. Due to insufficient prior work on these training-free compression methods, we couldn't include a thorough comparison with other baseline algorithms. Also, our work focused on analyzing the results of six widely-used natural language understanding datasets among GLUE benchmark. We believe that extra experiments on various challenging natural language understanding tasks will show our work's generalization performance. We have conducted experiments on various settings; varying layer types, layer depth, low resource, unsupervised, and unseen domain. However, there are still extra room for improving this work, such as exploring and applying layer-specific pruning rates, which we leave for future work.

Acknowledgements

We thank anonymous reviewers for their constructive and insightful comments. K. Jung is with ASRI, Seoul National University, Korea. This work was supported by AIRS Company in Hyundai Motor Company & Kia Motors Corporation through HMC/KIA-SNU AI Consortium Fund. This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [NO.2022-0-00184, Development and Study of AI Technologies to Inexpensively Conform to Evolving Policy on Ethics & NO.2021-0-02068, Artificial Intelligence Innovation Hub (Artificial Intelligence Institute, Seoul National University) & NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)]

References

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. 2020. Earlybert: Efficient bert training via early-bird lottery tickets. *arXiv preprint arXiv:2101.00063*.

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

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.

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

Song Han, Jeff Pool, John Tran, and William Dally. 2015b. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Xuanli He, Iman Keivanloo, Yi Xu, Xiang He, Belinda Zeng, Santosh Rajagopalan, and Trishul Chilimbi. 2021. Magic pyramid: Accelerating inference with early exiting and token pruning. *arXiv preprint arXiv:2111.00230*.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793.

Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*.

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

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2021. Learned token pruning for transformers. *arXiv preprint arXiv:2107.00910*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

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

Yuanxin Liu, Zheng Lin, and Fengcheng Yuan. 2021. ROSITA: refined BERT compression with integrated techniques. *CoRR*, abs/2103.11367.

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

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Matan Ben Noach and Yoav Goldberg. 2020. Compressing pre-trained language models by matrix decomposition. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 884–889.

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

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

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. Online structured laplace approximations for overcoming catastrophic forgetting. *Advances in Neural Information Processing Systems*, 31.

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

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

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2016. Not just a black box: Interpretable deep learning by propagating activation differences. *arXiv preprint arXiv:1605.01713*, 4.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2018. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

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

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

A Algorithms

Our pruning method consists of two stages: (1) Derivation of task-specific neuron indices per layer for a specific task (2) Real-time task-specific inference with previously pruned layers.

Algorithm 1 Deriving task-specific neuron indices per layer for a task t

Input: task-specific dataset $\mathcal{D}^t$; model $\mathcal{P}$; pruning rate p
Output: list $\mathcal{M}^t$ with task-specific neuron indices per layer

Initialize all $\mathcal{M}_l^t$ as an empty set and all $A_i^{(\mathcal{D}^t)}$ to zero
$\mathcal{B} \leftarrow$ split $\mathcal{D}^t$ into mini-batches of size β
for each batch $b \in \mathcal{B}$ **do**
  for each layer $l \in \mathcal{P}$ **do**
    for $i = 1$ to k^l **do**
      compute neuron importance $A_i^{(b)}(h^l)$
      $A_i^{(\mathcal{D}^t)}(h^l) \leftarrow A_i^{(\mathcal{D}^t)}(h^l) + A_i^{(b)}(h^l)$
for each layer $l \in \mathcal{P}$ **do**
  for $i = 1$ to k^l **do**
    if $argsort_i(A^{(\mathcal{D}^t)}(h^l)) < \lfloor k^l \times p \rfloor$ **then**
      $\mathcal{M}_l^t \leftarrow \mathcal{M}_l^t \cup \{i\}$
return $\mathcal{M}^t$

In the first stage, we sort neuron indices in descending order by computed attribution scores, leaving high-importance neurons by $(1 - p)$ ratio.

Algorithm 2 Real-time task-specific inference with pruned layers

Input: task t; text inputs x; indices container $\mathcal{M}$; model $\mathcal{P}$
Output: text outputs y

For task t, load corresponding $\mathcal{M}^t$
for each layer $l \in \mathcal{P}$ **do**
  $W^l \leftarrow (W_{ij}^l)_{\substack{i \in \mathcal{M}_{l'}^t \\ j \in \mathcal{M}_l^t}}$ ▷ match rows with a previous layer l'
  if bias b^l exists in layer l **then**
    $b^l \leftarrow (b_i^l)_{i \in \mathcal{M}_l^t}$
compute outputs y with x using the pruned model $\tilde{\mathcal{P}}$
return y

In the second stage, we prune task-specifically unimportant neurons when given a user request for a specific task. We task-specifically compress a model in real-time and conduct an inference with the pruned model.

B Statistic of Pruning Results

We compute the pruning results of the baselines of T5-RP and T5-RAP through five random trials. The standard deviations of the accuracy for the two baselines are shown in Table 1.

		SST-2	MRPC	QNLI	RTE	CB	BoolQ
T5-RP	**Encoder**	0.0202	0.0046	0.0143	0.0426	0.0168	0.0020
	Decoder	0.0241	0.0108	0.0003	0.0580	0.0010	0.0060
T5-RAP	**Encoder**	0.0082	0.0080	0.0119	0.0165	0.0099	0.0061
	Decoder	0.0159	0.0089	0.0029	0.0123	0.0027	0.0063

Table 1: Standard deviations of Pruning results.

We calculate the standard deviations by averaging the values derived by all pruning rates. These results reveal that the variances of T5-RP and T5-RAP are not significant.