Gradient-Free Structured Pruning with Unlabeled Data

Azade Nova [1] **Hanjun Dai** [1] **Dale Schuurmans** [1 2]

Abstract

Large Language Models (LLMs) have achieved great success in solving difficult tasks across many domains, but such success comes with a high computation cost, and inference latency. As developers and third parties customize these models, the need to provide efficient inference has increased. Many efforts have attempted to reduce inference cost through model compression techniques such as pruning and distillation. However, these techniques either require labeled data, or are time-consuming as they require the compressed model to be retrained to regain accuracy. In this paper, we propose a gradient-free structured pruning framework that uses only unlabeled data. An evaluation on the GLUE and SQuAD benchmarks using BERT$_{BASE}$ and DistilBERT illustrates the effectiveness of the proposed approach. By only using the weights of the pre-trained model and unlabeled data, in a matter of a few minutes on a single GPU, up to 40% of the original FLOP count can be reduced with less than a 4% accuracy loss across all tasks considered.

1. Introduction

Large Language Models (LLMs) have made great strides in solving difficult tasks across many domains, but this has come at the cost of high parameter counts and significant computational overhead. Developers and third parties can now employ these trained models and create custom versions tailored to their particular applications. Customization makes these models applicable to a wider variety of use cases, but this, even more, highlights the need for efficient inference models.

Many efforts have been being made to reduce computational cost through model compression techniques specialized for Transformers, including structured pruning (Xia et al., 2022; Hou et al., 2020; Sajjad et al., 2023; Liu et al., 2021a; Xia et al., 2022), efficient architecture design (Kitaev et al., 2020; Iandola et al., 2020; Sun et al., 2020; Wang et al., 2020b), neural architecture search (So et al., 2021; Xu et al., 2021; Yin et al., 2021), knowledge distillation (Sun et al., 2020; Jiao et al., 2019; Sanh et al., 2019), quantization (Kim et al., 2021; Shen et al., 2020; Zadeh et al., 2020; Zafrir et al., 2019), and hardware-software co-design (Gu et al., 2022; Ham et al., 2021).

Among these techniques, structured pruning shows promising results in reducing model size, while also improving inference time because the resulting model remains compatible with the underlying hardware. However, most existing approaches are quite complex and require significant engineering effort to implement. Moreover, the process of compression is time-consuming and requires retraining the compressed model to regain accuracy. These limitations make effective compression difficult to realize in practice. Recently, Kwon et al. (2022) proposed a post-training pruning for Transformers that does not require any retraining of the model. Even though this approach avoids expensive retraining, it requires labeled data in the pruning pipeline.

LLMs mainly utilize unlabeled data for training and with increased use of pre-trained LLMs by developers and third parties, access to the labeled data is questionable. Especially with the popularity of in-context learning, where the user only provides prompts, the purpose of the task is not necessarily known at compression time. In this scenario, none of the existing pruning techniques can be applied for model compression since they all require labeled data. Even though knowledge distillation (Sun et al., 2020; Jiao et al., 2019; Sanh et al., 2019) trains a student model with unlabeled data, it still requires a large amount of unlabeled data and is expensive to train. This motivates us to investigate whether one can design a structured pruning method that does not require retraining nor labeled data, while avoiding adverse effects on performance.

In this work, we propose Kernelized Convex Masking (KCM), a gradient-free framework (Figure 1) that only requires the trained model and sampled raw data to compress the model. We introduce *R2D2* as the core of this framework that combines two ranking techniques *Representative Ranking* (*R2*) and *Data-Driven* (D2) to estimate the impor-

[1]Google DeepMind [2]University of Alberta. Correspondence to: Azade Nova <azade@google.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

tance of individual neurons. *R2* maps our structured pruning goals into a representative selection problem (Huang et al., 2018), where the goal is to find a small subset of data points that can well represent a large dataset. Specifically, *R2* considers the filters of a Feed-Forward Network (FFN) in the trained model as data points in a high-dimensional space, and ranks these by how well a filter can be represented by others. *D2*, on the other hand, ranks the filters based on statistics gathered from layer-wise model outputs using the raw sampled data. KCM decides which filter to remove by merging the *R2D2* rankings across all layers. Since removing filters may still affect accuracy, we apply an existing scaling transformation method in Kwon et al. (2022) to mitigate the effect of their removal.

Our main contributions are as follows:

- We Propose Kernelized Convex Masking (KCM) pruning framework, a gradient-free structured pruning approach that neither requires labeled data nor retraining.
- As the core of KCM, we propose *R2D2* that combines two ranking techniques *Representative Ranking* (*R2*) and *Data-Driven* (D2). *R2D2* only requires the weights of the trained model and sampled raw data to rank the neurons. An ablation study confirms the importance of combining the two proposed ranking techniques.
- Our evaluation on GLUE and SQuAD benchmarks using BERT$_{BASE}$ and DistilBERT confirms the effectiveness of the proposed approach. Compared to when the labeled data is available, KCM is able to reduce up to 40% of the original FLOPs with less than 4% accuracy loss, in a matter of a few minutes on a single GPU.

2. Problem Definition

2.1. Preliminary

In this paper, we focus on pruning the BERT (Devlin et al., 2018) architecture. BERT is a stack of L homogeneous Transformer encoder blocks (Vaswani et al., 2017), each of which consists of a multi-head attention (MHA) layer followed by a Feed-Forward Network (FFN) layer. Due to the fact that FFN layers have a huge impact on model size and inference latency (Ganesh et al., 2021), we focus on the pruning the filters of the FFN layers. In every transformer encoder layer ℓ, the $FFN^{\ell}(x)$ with N filters is parameterized with $W_{\ell}^{(1)} \in \mathbb{R}^{d\times N}$, $W_{\ell}^{(2)} \in \mathbb{R}^{N\times d}$, $b_{\ell}^{(1)} \in \mathbb{R}^{N}$, $b_{\ell}^{(2)} \in \mathbb{R}^{d}$, and activation function σ:

$$FFN_{\ell}(x) = \sum_{i=1}^{N} (\sigma(xW_{\ell}^{(1)}[:,i] + b_{\ell}^{(1)})W_{\ell}^{(2)}[i,:]) + b_{\ell}^{(2)} \tag{1}$$

For example, $BERT_{BASE}$ has 12 transformer encoder blocks ($L = 12$), where the number of filters (N) is 3072,

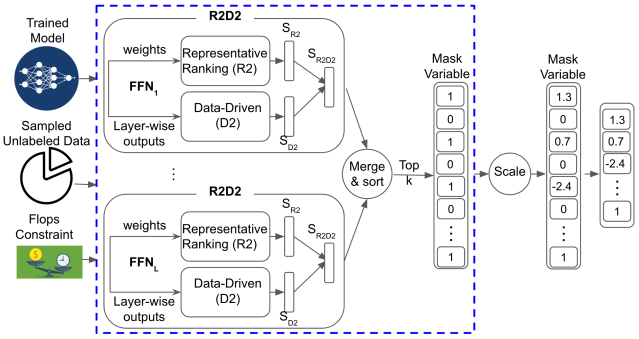

Figure 1. Kernelized Convex Masking (KCM): A gradient-free structured pruning framework with *R2D2* as a core component. *R2D2* combines the ranks of the *Representative Ranking* (*R2*) and *Data-Driven* (*D2*) rank.

$W_{\ell}^{(1)} \in \mathbb{R}^{768\times 3072}$ with the GELU activation (Hendrycks & Gimpel, 2016), and $W_{\ell}^{(2)} \in \mathbb{R}^{3072\times 768}$.

2.2. Structured Pruning by Masking

Masking: Given an integer $n < N$, reducing the number of filters from N to n can be considered as introducing a mask variable $m \in \mathbb{R}^N$ (with n non-zero elements) associated with the outputs of the filters.

$$\widehat{FFN}_{\ell}(x) = \sum_{i=1}^{N} (\sigma(xW_{\ell}^{(1)}[:,i] + b_{\ell}^{(1)})W_{\ell}^{(2)}[i,:] \circ m_i) + b_{\ell}^{(2)} \tag{2}$$

where $\circ$ is Hadamard product.

Objective: Transformer pruning can be formulated as a constrained optimization problem on the mask $\mathcal{M} \in \mathbb{R}^{L\times N}$ that represents the mask m of all layers L. There are LN filter mask variables which is much less than the total number of parameters in the model. For example BERT$_{BASE}$ with 110M parameters needs only 36k mask variables (0.03%).

Optimal structural pruning is usually defined (Kwon et al., 2022) in the supervised setting with respect to minimizing the accuracy loss of the original model:

$$\underset{\mathcal{M}}{\operatorname{argmin}}\, \mathcal{L}(\mathcal{M}) \quad s.t. \quad Cost(\mathcal{M}) \leq \mathcal{C} \tag{3}$$

$Cost(\mathcal{M})$ is the floating point operations (FLOPs) of the pruned model determined by the mask $\mathcal{M}$. In this work, since only unlabeled data is available, the supervised loss $\mathcal{L}(\mathcal{M})$ can not be evaluated. Similar to distillation, we consider minimizing the Feature Map Loss (Sun et al., 2020) $\mathcal{L}_{FMT}$ for each FFN in layer ℓ.

$$\mathcal{L}_{(\ell)FMT}(m) = \|FFN_{(\ell)}(x) - \widehat{FFN}_{(\ell)}(x)\|_2 \tag{4}$$

Given a trained model Model, unlabeled dataset $\mathcal{D}$, and a cost constraint $\mathcal{C}$, find the mask $\mathcal{M} \in \mathbb{R}^{L\times N}$ for every N

Algorithm 1 Kernelized Convex Masking (KCM)

Input: Trained model: Model, FLOPs constraint $\mathcal{C}$, Gaussian Kernel K, convergence rate α
Output: Mask $\mathcal{M}$
Initialize mask $\mathcal{M}$ as $\mathbf{0}$
*//Call Representative Ranking (*R2*) Algorithm 2*
S_{R2} = *R2*(Model, K, α)
// Data-Driven (*D2*) Ranking
for batch in sample-data **do**
  for each layer ℓ in Model collect $H_\ell^{(1)}$
  $S_{D2}[\ell]$ = average over $H_\ell^{(1)}$ for each filter
end for
$S_{R2D2}[\ell] = S_{R2}[\ell] * normalized(S_{D2}[\ell])$
k = Number of neurons to satisfy FLOPs constraint $\mathcal{C}$
Candidates = top-k filters of the sorted S_{R2D2}
$\mathcal{M}[Candidates] = 1.0$
return $\mathcal{M}$

Algorithm 2 Representative Ranking (R2)

Input: Trained model: Model, Gaussian Kernel K with width σ, convergence rate α
Output: S_{R2} that represents importance of Filters in all layers.
for layer ℓ in layers of the Model **do**
  $W_\ell^{(2)} \in \mathbb{R}^{N\times d}$ of FFN_ℓ
  Initialize coefficient matrix: $C_0 \in \mathbb{R}^{N\times N} = \frac{1}{N}$
  repeat
    $C_{i+1} = C_i \circ \sqrt{\frac{K(W_\ell^{(2)}, W_\ell^{(2)})}{K(W_\ell^{(2)}, W_\ell^{(2)})C_i}}$
    $\delta = \frac{(C_{i+1}-C_i).sum()}{C_i.sum}$
    $C_i = C_{i+1}$
  until convergence i.e. $\delta \leq \alpha$
  $S_{R2}[\ell]$= diagonal(C_i)
end for
return S_{R2}

filters of all L transformer layers such that $Cost(\mathcal{M})$ be less than $\mathcal{C}$ and the loss $\mathcal{L}_{FMT}(\mathcal{M}) = \sum_{\ell=1}^{L} \mathcal{L}^{(\ell)}{}_{FMT}(\mathcal{M}_{\ell,:})$ is minimized.

$$\underset{\mathcal{M}}{\operatorname{argmin}}\, \mathcal{L}_{FMT}(\mathcal{M}) \quad s.t. \quad Cost(\mathcal{M}) \leq \mathcal{C} \tag{5}$$

One way to tackle this problem would be to consider it as a version of the distillation problem, where the goal is to find the optimal mask under the sparsity constraint. However, distillation methods require large amounts of unlabeled data and are very expensive to train (Xia et al., 2022).

3. Proposed Approach

Instead, in this work, we propose a gradient-free approach that only uses the weights of the trained model and statistics on layer-wise outputs using the unlabeled data to implicitly minimizes the feature map loss in each layer.

3.1. Framework Overview

Figure 1 shows the overview of our framework, called Kernelized Convex Masking (KCM). KCM takes the trained model Model, sampled unlabeled dataset $\mathcal{D}$ and a cost constraint $\mathcal{C}$, and returns a mask $\mathcal{M} \in \mathbb{R}^{L\times N}$ that represents the mask of the N filters of all layers L.

We introduce *R2D2* that combines ranking techniques *Representative Ranking* (*R2*) and *Data-Driven* (D2) to estimate the importance of the filters. As shown in Figure 1, these two approaches independently rank N filters based on the weights and output of the activation function of the FFNs in all layers L. Then KCM merges the results of *R2D2* across all layers. The top k filters are selected and the rest will be masked to zero. Note that given a FLOPs constraint $\mathcal{C}$, k is the total number of filters that satisfies constraint $\mathcal{C}$. Finally, we apply a scaling transformation in Kwon et al. (2022) over the selected filters to recover the accuracy drop and reduce the feature map loss in Equation 4. Next, we discuss our framework in more detail.

3.2. Kernelized Convex Masking(KCM)

Algorithm 1 illustrates the end-to-end approach. We first present the details of the proposed *R2D2*. Then, we discuss how we use these rankings for the final masking.

3.2.1. Representative Ranking (*R2*)

By considering $H_\ell^{(1)} = \sigma(xW_\ell^{(1)} + b_\ell^{(1)})$, Equation 1 can be written as $FFN_\ell(H_\ell^{(1)}) = H_\ell^{(1)}W_\ell^{(2)} + b_\ell^{(2)}$. Our filter *Representative Ranking* assumes $H_\ell^{(1)}$ is unknown and only uses the weights $W_\ell^{(2)}$ to rank the N filters.

From the computational geometry perspective the filters in $W_\ell^{(2)} \in \mathbb{R}^{N\times d}$ can be considered as N data points in a d dimensional space. The structured pruning goal can be translated as selecting a subset of data points (filters) to be used as representatives that can describe any data point (filter) in the dataset. There has been a lot of work on finding such a representative set (Kazemi et al., 2022; Killamsetty et al., 2021; You et al., 2020). However, for linear functions, this problem can be reduced to finding a convex hull. *The convex hull is a subset of data points that can be used to find the maxima of any linear function.* Since $FFN_\ell(H_\ell^{(1)})$ is, in fact, a linear function, the convex hull of $W_\ell^{(2)}$ can be considered as a representative of the filters that produce the maxima of FFN_ℓ regardless of the input $H_\ell^{(1)}$.

The challenge is that in a d dimensional space finding the exact convex hull is in order of $\mathcal{O}(N^{d/2})$ time, which can be very expensive (e.g. in $BERT_{BASE}$, d=768). Moreover, the number of convex hull data points radically increases with the number of dimensions. To address these limitations,

Table 1. Comparison of the different structured pruning methods studied in this work. ✗, and ✓ show if a method has the specific feature or not. N/A means not applicable. To simplify notation, we show gradient-free with (!∇). Supervision-free indicates not using labeled data.

Method	Gradient-free (!∇)	Retrain/Finetune-free	Supervision-free	Pruning time $\leq 7min$
FLOP (Wang et al., 2019)	✗	✗	✗	✗
SLIP (Lin et al., 2020)	✗	✗	✗	✗
Sajjad et al. (Sajjad et al., 2023)	✗	✗	✗	✗
DynaBERT (Hou et al., 2020)	✗	✗	✗	✗
EBERT (Liu et al., 2021b)	✗	✗	✗	✗
Mask-Tuning (Kwon et al., 2022)	✗	✓	✗	✓
Weight-Magnitude (Li et al., 2016)	✓	✓	N/A	✓
Weight-Magnitude-Scale	✓	✓	✓	✓
KCM (ours)	✓	✓	✓	✓

rather than finding the exact solution, we propose to assign a ranking over the filters that represents how well a filter is representing others.

Convex hull approximation is well-studied area. Among existing methods *Kernelized Convex Hull Approximation (KCHA)* (Huang et al., 2018) is one of the approaches that can be applied to our problem. Algorithm 2 shows the proposed *Representative Ranking* based on the KCHA. Specifically for each layer ℓ, we seek a positive coefficient matrix $C \in \mathbb{R}^{N\times N}$ that minimizes $\|W_\ell^{(2)} - W_\ell^{(2)}C\|_2$. The diagonal elements of C indicate whether the corresponding data instances are extreme points. Huang et al. (2018) solves this problem as a Semi-NMF problem (Ding et al., 2008), rather than a Non-negative Least Square problem, and adopts a multiplicative updating rule as the solver:

$$C^{i+1} = C^i \circ \sqrt{\frac{[W_\ell^{(2)^T} W_\ell^{(2)}]_+ + [W_\ell^{(2)^T} W_\ell^{(2)}]_- C^i}{[W_\ell^{(2)^T} W_\ell^{(2)}]_- + [W_\ell^{(2)^T} W_\ell^{(2)}]_+ C^i}} \tag{6}$$

where $[A]_+ = \frac{A+|A|}{2}$, $[A]_- = \frac{A-|A|}{2}$, and $|A|$ is the absolute values of A. Please refer to Huang et al. (2018) for more detail.

$W_\ell^{(2)^T} W_\ell^{(2)}$ can be considered as $K(W_\ell^{(2)}, W_\ell^{(2)})$. In this paper, we use a Gaussian kernel. Since the kernel value is positive and $K(W_\ell^{(2)}, W_\ell^{(2)}) = 1$, the updating rule of Semi-NMF algorithm can be modified as:

$$C^{i+1} = C^i \circ \sqrt{\frac{K(W_\ell^{(2)}, W_\ell^{(2)})}{K(W_\ell^{(2)}, W_\ell^{(2)}) C^i}} \tag{7}$$

Algorithm 2 illustrates the steps of the *Representative Ranking*, where for each layer the coefficient matrix C is independently calculated using Equation 7. The algorithm then returns the diagonal of C as the ranking score of the filters. The width of the Gaussian kernel σ, and the convergence rate α are hyperparameters. In our experiments, we observe that setting $\sigma = 1.0$ and $\alpha = 0.01$ works for all tasks considered. Moreover, on average it takes less than 20 iterations to converge.

3.2.2. Data-Driven Ranking (*D2*)

Representative Ranking (*R2*) assumes $H_\ell^{(1)}$ is unknown and ranks the filters solely based on the weights $W_\ell^{(2)}$. One could imagine using a similar ranking approach over $W_\ell^{(1)^T}$ to rank the N filters. However, as mentioned, the convex hull is only a good representative for finding the maxima of any *linear function*, and the activation function σ makes $H_\ell^{(1)}$ nonlinear.

Therefore, to incorporate the nonlinearity introduced by the activation function, *Data-Driven* (*D2*) performs a forward pass using sampled unlabeled data, and gathers statistics on the output results of each layer $H_\ell^{(1)}$. It then uses the normalized average of these outputs to rank filters in each layer (Algorithm 1 lines (5 to 7)).

3.2.3. Merge and Scale

Thus far, KCM ranks N filters of each layer independently by the filter Representative Ranking (*R2*) and Data-Driven Ranking (*D2*). In every layer, *R2D2* combines the scores of *R2* and *D2* to capture the importance of filters based on the model weights and the layer outputs of the raw data (Algorithm 1 line 9). In our experiments, we run an ablation study to present the importance of these rankings.

Given a FLOPs constraint $\mathcal{C}$, let k be the total number of filters that satisfy $\mathcal{C}$. In other words, the pruned model only should have k active filters across all layers and the rest should be removed. As shown in Algorithm 1, KCM merges the *R2D2* scores (S_{R2D2}) across layers, and the top k filters are selected to be active in the pruned model (Algorithm 1 lines 10-12).

Since after masking some accuracy drop is inevitable, existing structured pruning methods have shown that scaling can be helpful. Thus, similar to Kwon et al. (2022) as shown in Figure 1, we apply a scaling transformation to the selected filters. Such scaling uses only the unlabeled data and, based on the generated mask, aims to reconstruct the layer-wise outputs by scaling the outputs of the active filters. This, in fact, reduces the feature map loss in Equation 4.

Table 2. Accuracy degradation of pruning BERT$_{BASE}$ using our method and the prior structured pruning methods with different relative FLOPs. Note that our method is gradient-free (!∇), does not use label of the data and not require retraining (more detail in Table 1)

!∇	Method	QQP			QNLI			SST-2			MRPC		
		∼60%	∼65%	75%	∼60%	∼65%	75%	∼60%	∼65%	75%	∼60%	∼65%	75%
✗	FLOP	–	–	–	–	-2.6	–	–	-0.6	–	–	-2.3	–
✗	SLIP	-1.7	-0.9	–	-2.1	-0.9	–	-1.0	-0.9	–	-1.0	-2.8	–
✗	Sajjad	–	-0.4	–	–	-1.4	–	–	-1.8	–	–	-8.6	–
✗	DynaBERT	–	–	–	–	–	–	–	–	-0.6	–	–	-1.7
✗	EBERT	-0.4	–	–	-1.3	–	-1.0	–	–	–	–	–	–
✗	Mask-Tuning	-0.65	-0.42	-0.14	-1.35	-0.89	-0.18	-1.08	-0.91	-0.68	-1.72	-0.24	-0.14
✓	**KCM (Ours)**	-1.85	-.99	-0.35	-3.62	-1.72	-0.78	-2.46	-1.71	-0.68	-1.96	-1.47	-0.71

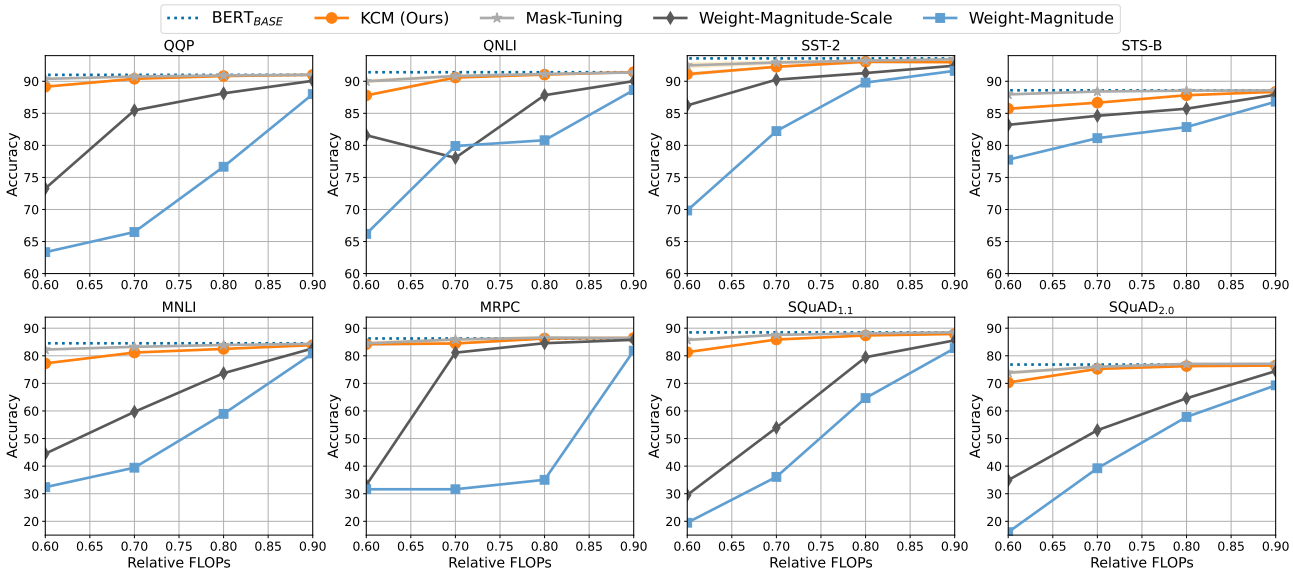

Figure 2. Performance of our pruning framework KCM against Mask-Tuning (Kwon et al., 2022), Weight-Magnitude (Li et al., 2016), and Weight-Magnitude-Scale on BERT$_{BASE}$. Weight-Magnitude-Scale combines (Li et al., 2016) with the scaling approach from (Li et al., 2016). Mask-Tuning uses labeled data but KCM and Weight-Magnitude-Scale are gradient-free with unlabeled data (Table 1). KCM outperforms Weight-Magnitude and Weight-Magnitude-Scale which highlights the effectiveness of our approach in the absence of labeled data. For 70% and 60% FLOPs constraints, Mask-Tuning that uses labeled data performs slightly better than KCM. Table 3 shows this gap more clearly.

4. Evaluation

4.1. Experimental Setup

We implemented our framework with PyTorch (Paszke et al., 2019) using the HuggingFace Transformers (Wolf et al., 2020) library. We evaluate the effectiveness of the proposed approach using BERT$_{BASE}$ (Devlin et al., 2018) and DistilBERT (Sanh et al., 2019) on GLUE (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2018; 2016) benchmarks.

For the Data-Driven ranking we use 2K raw data from the training sets. Note that we only use raw input and no label is used. Figure 6, in Appendix B, shows how sample size affects our performance. For the Representative Ranking calculation in Algorithm 2, the width of the Gaussian kernel, σ, and the convergence rate, α, are the hyperparameters. In our experiments, we set $\sigma = 1.0$ and $\alpha = 0.01$. Moreover, on average it takes less than 20 iterations to converge. All results are averaged over the runs with 10 different seeds. Please refer to Appendix A for more detail on experimental setup.

Baselines from structured pruning methods: Table 1, shows the comparison of the different structured pruning methods specialized for Transformers studied in this work. We compare these methods by 4 important features, including gradient-free (no backward pass), retrain/finetune-free (no retrain/finetune), supervision-free (no use of labeled data), and fast pruning-time. We compare our proposed

Table 3. Performance of our pruning framework KCM against Mask-Tuning (Kwon et al., 2022) on BERT$_{BASE}$, for 70% and 60% FLOPs constraints. Mask-Tuning that uses labeled data performs slightly better than KCM. Unlike Mask-Tuning, KCM is gradient-free(!∇) with unlabeled data.

!∇	Method	QQP		MNLI		MRPC		QNLI	
		60%	70%	60%	70%	60%	70%	60%	70%
	baseline	91.00		84.53		86.27		91.41	
✗	Mask-Tuning	90.38 ± 0.07	90.74 ± 0.07	82.26 ± 0.21	83.24 ± 0.16	84.51 ± 0.63	85.91 ± 0.40	90.00 ± 0.26	90.83 ± 0.16
✓	**KCM (Ours)**	89.15 ± 0.04	90.39 ± 0.04	77.24 ± 0.10	81.18 ± 0.10	84.19 ± 0.44	84.46 ± 0.29	87.79 ± 0.15	90.58 ± 0.08
!∇	Method	SST-2		STS-B		SQuAD$_{1.1}$		SQuAD$_{2.0}$	
		60%	70%	60%	70%	60%	70%	60%	70%
	baseline	93.57		88.59		88.48		76.82	
✗	Mask-Tuning	92.47 ± 0.41	92.92 ± 0.26	87.95 ± 0.12	88.40 ± 0.05	85.77 ± 0.41	87.57 ± 0.11	73.86 ± 0.55	76.00 ± 0.29
✓	**KCM (Ours)**	91.11 ± 0.23	92.26 ± 0.09	85.72 ± 0.12	86.66 ± 0.05	81.29 ± 0.06	85.89 ± 0.04	70.30 ± 0.13	75.24 ± 0.10

Table 4. Performance of our pruning framework KCM against Mask-Tuning (Kwon et al., 2022) on DistilBERT, for 70% and 60% FLOPs constraints. Even though Mask-Tuning uses labeled data, for SQUAD$_{2.0}$ task, KCM performs better than Mask-Tuning, and the results of both approaches on QQP, and STS-B are comparable.

!∇	Method	QQP		MNLI		MRPC		QNLI	
		60%	70%	60%	70%	60%	70%	60%	70%
	baseline	89.99		82.11		84.80		88.56	
✗	Mask-Tuning	88.71 ± 0.22	89.66 ± 0.06	80.51 ± 0.19	81.65 ± 0.09	84.73 ± 0.71	84.83 ± 0.35	87.72 ± 0.38	88.43 ± 0.07
✓	**KCM (Ours)**	88.16 ± 0.03	89.28 ± 0.03	78.05 ± 0.08	80.60 ± 0.05	79.66 ± 0.27	83.01 ± 0.16	85.93 ± 0.09	86.93 ± 0.13
!∇	Method	SST-2		STS-B		SQuAD$_{1.1}$		SQuAD$_{2.0}$	
		60%	70%	60%	70%	60%	70%	60%	70%
	baseline	91.40		86.12		85.73		68.84	
✗	Mask-Tuning	90.44 ± 0.41	90.93 ± 0.24	85.73 ± 0.07	85.96 ± 0.10	83.20 ± 0.16	84.64 ± 0.09	62.36 ± 1.40	65.32 ± 0.48
✓	**KCM (Ours)**	88.38 ± 0.25	90.61 ± 0.25	85.26 ± 0.02	85.55 ± 0.03	76.92 ± 0.11	82.65 ± 0.06	**64.56 ± 0.11**	**68.19 ± 0.06**

method with Flop (Wang et al., 2019), SLIP (Lin et al., 2020), Sajjad et al. (Sajjad et al., 2023), DynaBERT (Hou et al., 2020), and EBERT (Liu et al., 2021b). All of these techniques require retraining of the pruned model and/or jointly learning the pruning configurations during training, which leads to high training time, and they are not gradient-free. Specifically, as shown in Kwon et al. (2022) these methods require 5 to 33 hours of retraining.

Mask-Tuning (Kwon et al., 2022) is a recent work that does not need retraining but still relies on the labeled data and uses gradient computation to evaluate the importance of each filter. We also compare our method with Weight-Magnitude (Li et al., 2016), which is a light-structured pruning method that is gradient-free, does not retrain the pruned model, and does not use the data at all. We introduce Weight-Magnitude-scale that combines Li et al. (2016) with the scaling approach from Li et al. (2016). Note that the scaling step in Li et al. (2016) only needs unlabeled data, so Weight-Magnitude-scale will have the exact problem setup as our method 1. We would like to highlight that our method KCM, Mask-Tuning, Weight-Magnitude, and Weight-Magnitude-scale finish in less than 7 minutes across all tasks, which is 2 to 3 orders of magnitude faster than the other baselines. We evaluate the performance of our method against all these baselines by the FLOPs-accuracy trade-off of BERT$_{BASE}$ on the GLUE and SQuAD benchmarks. In the experimental results, to simplify the notation, we will indicate gradient-free with (!∇).

4.2. Experimental Results

Table 2 compares the accuracy drop of KCM against prior structured pruning methods in Table 1. Since the baseline accuracy differs slightly from paper to paper, we compare the amount of the accuracy drop from the baseline instead of the absolute accuracy. Similar to Kwon et al. (2022), we use the results without knowledge distillation and data augmentation reported in each paper since these add extra overhead. As one can see, the highest accuracy drop of KCM across all task is -3.62 which reduces 40% of the original FLOPs. Worth mentioning that, while all the baselines require labeled data and leverage the backward pass, our proposed method is gradient-free with unlabeled data.

Next, we perform a more thorough evaluation against Mask-Tuning (Kwon et al., 2022), Weight-Magnitude (Li et al., 2016) and Weight-Magnitude-Scale, since their problem setup is closer to ours (Table 1). Figure 2 shows the results on BERT$_{BASE}$ as we vary the FLOPs constraint from 90% to 60%, i.e, reducing 10% to 40% of the original FLOPs. Clearly, KCM outperforms Weight-Magnitude and Weight-Magnitude-Scale, highlighting the effectiveness of our approach in the absence of labeled data. For the 70% and 60% FLOPs constraints, Mask-Tuning performs slightly better than ours. Table 3 shows the gap more clearly. This gap can

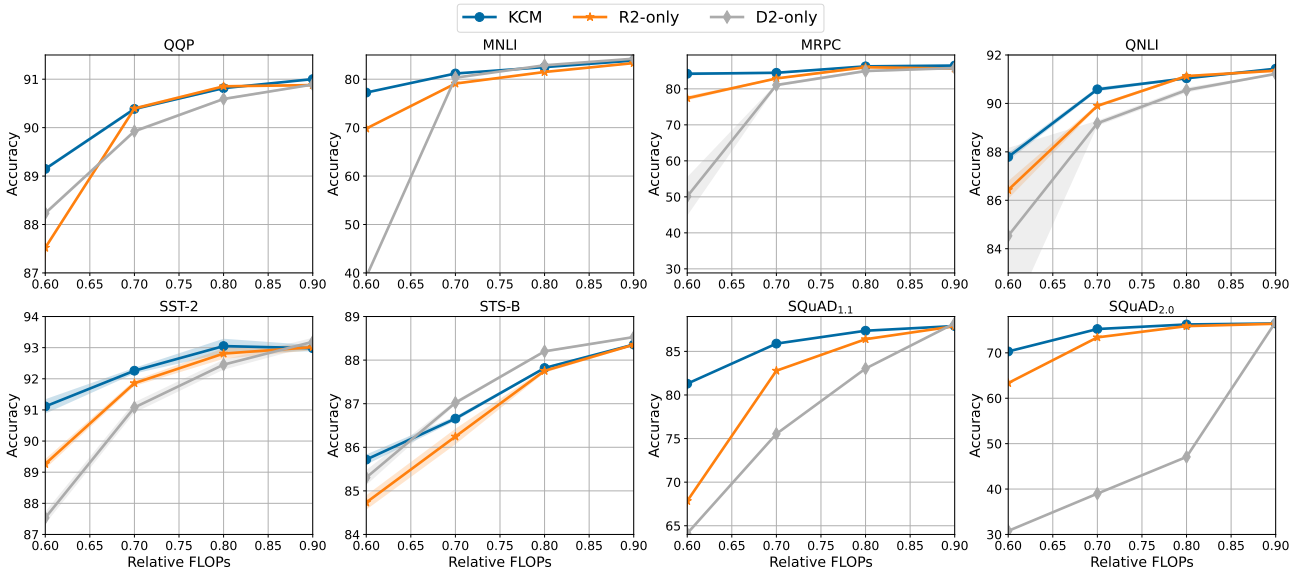

Figure 3. Ablation Study to investigate the importance of our ranking techniques. D2-only, and R2-only are our KCM that either uses *Data-Driven* or *Representative Ranking*. R2-only performs better than D2-only in all tasks except STS-B. But KCM that combines them by *R2D2* is able to leverage both of these rankings and shows improvement across all tasks considered.

be explained by the fact that, unlike Mask-Tuning, KCM is gradient-free with unlabeled data.

We further evaluate the performance of KCM against Mask-Tuning (Kwon et al., 2022) on DistilBERT for the 70% and 60% FLOPs constraints. As shown in Table 4, interestingly, even though Mask-Tuning leverages the backward pass and labeled data, the proposed KCM performs better than Mask-Tuning on the $\text{SQUAD}_{2.0}$ benchmark. Moreover the results of both approaches on QQP, and STS-B are quite comparable, showing that even without labeled data and no backward pass the accuracy loss of the pruned model by our KCM method can be minimal.

4.3. Ablation Studies

Importance of our ranking techniques: *R2D2* is the core component of our proposed approach KCM that combines the ranking of Representative Ranking (*R2*) and Data-Driven (*D2*) (Section 3). We run an ablation study to investigate the importance of these ranking techniques. Recall that *R2* ranks N filters based on the weights of the FFNs, while *D2* ranks them by the output of the activation function. Figure 3 illustrates how the performance changes if we only use one of these in our framework. While *D2*-only is our KCM without the *R2*, *R2*-only only uses the *R2* ranking. As one can see, except in STS-B where *D2*-only slightly outperforms KCM, using *R2*-only performs better than *D2*-only. More importantly, when *R2D2* combines them it allows our KCM to leverage both rankings and demonstrates improvement across all tasks. Note that the results of *D2*-only also confirm that using only the output of the activation functions is not always sufficient for pruning and highlights the impact of using the trained model weights (More results in Figure 5).

Dynamic neuron selection: Another important feature of our KCM is the fact that it dynamically decides how many neurons from each layer to prune. This feature is an outcome of merging the result of *R2D2* across all L layers. Figure 4 illustrates how KCM affects different layers of the BERT_{BASE}. Clearly more pruning occurs over the last three layers, and more than half of the filters in the first two layers are pruned. From KCM point of view, the middle layers seem to be more important across all tasks.

4.4. Discussion

Our KCM is a gradient-free structured pruning framework that neither requires retraining nor labeled data. Here we would like to discuss what if we have a limited labeled data and how our approach can be extended to leverage that.

Recall that *R2D2* uses the statistics from the unlabeled data to rank filters based on layer-wise output. Thus a simple add-on would be to freeze the trained model, use the limited labeled data and only do one forward-backward pass and gather the gradient over the mask variables. Note that unlike Mask-Tuning (Kwon et al., 2022), we do not calculate the Fisher information since we just want to use the gradient as the new signal for the pruning. To do so, for example

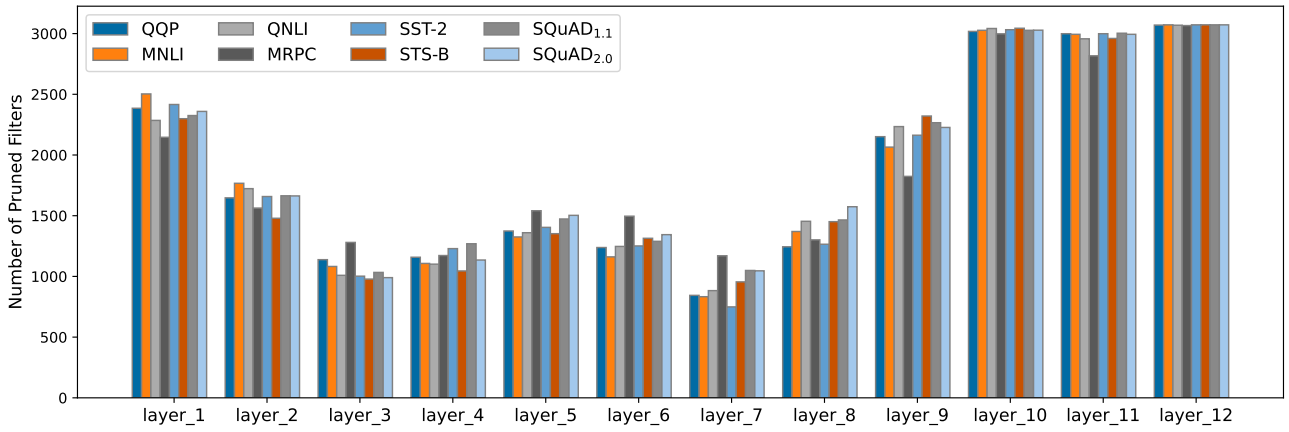

Figure 4. Ablation Study to investigate how many filters from each layer are pruned by KCM on BERT$_{BASE}$. While middle layers seems to less get affected by KCM, many filters from the last three layers and first layer are pruned.

1) the gradient information can be used as a new ranking criteria that can be combined into our *R2D2* or 2) one can use it to refine the top-k results of our KCM. Specifically, let us assume f_i be the least important filter in top-k result of the KCM, and f_j be the most important one from the gradient scores. If f_j is not already in the top-k results, we can switch f_i with f_j if the total gradient of the top-k results increases.

We implemented this simple greedy solution as an add-on to our KCM and show that indeed having a limited labeled data contributes to improve the accuracy drop. Table 5 shows the result on DistilBERT where only 512 sampled label data is available. Since this is out of the scope of this work, We leave a more thorough investigation as a future work.

5. Related Work

There has been a lot of work on efficient transformers that improve inference speed and reduce memory usage, including efficient architecture design (Kitaev et al., 2020; Iandola et al., 2020; Sun et al., 2020; Wang et al., 2020b; Wu et al., 2020; Fakoor et al., 2020a; Lan et al., 2019; Lee et al., 2022), neural architecture search (So et al., 2021; Chen et al., 2020a; So et al., 2019; Wang et al., 2020a; Xu et al., 2021; Yin et al., 2021), knowledge distillation (Sun et al., 2020; Jiao et al., 2019; Sanh et al., 2019; Sun et al., 2019; Fakoor et al., 2020b), quantization (Kim et al., 2021; Shen et al., 2020; Zadeh et al., 2020; Zafrir et al., 2019), and hardware-software co-design (Ham et al., 2021; Tambe et al., 2021; Wang et al., 2021; Gu et al., 2022; Shi et al., 2018).

Pruning is an important area of research for model sparsity that removes insignificant weights in neural networks. While Kurtic et al. (2022); Sanh et al. (2020); Gale et al. (2019); Zhang et al. (2022) proposed second-order, first-order, and magnitude-based pruning methods for Transformers, Chen et al. (2020b;c); Prasanna et al. (2020) explored the lottery ticket hypothesis. These methods can significantly reduce the model size; however, they might not offer significant inference speedup since the hardware and cannot efficiently utilize the unstructured sparse patterns.

Structured pruning methods, on the other hand, target removing groups of parameters. For example, low-rank factorization (Gu et al., 2022; Wang et al., 2019), corsets based techniques (Mussay et al., 2021; 2019; Liebenwein et al., 2019; Baykal et al., 2018), block-wise sparsity (Li et al., 2020), and tile-wise sparsity (Guo et al., 2020a) prune structured sets of parameters in weight matrices. Additionally, attention head pruning (Michel et al., 2019; Voita et al., 2019) and layer dropping (Fan et al., 2019; Sajjad et al., 2023; Peer et al., 2022) have been commonly used as more coarse-grained methods. Recent research has also explored combining different pruning granularity and principles to maximize model efficiency in all dimensions (Chen et al., 2021; Khetan & Karnin, 2020; Lagunas et al., 2021; Lin et al., 2020; Liu et al., 2021a; Xia et al., 2022; Yao et al., 2021). Another approach is to dynamically prune Transformers during inference time (Fan et al., 2019; Hou et al., 2020; Liu et al., 2021b; Xin et al., 2020; Zhou et al., 2020).

Even though structured pruning methods can be effective for compression and speedup, they can be difficult to implement in practice due to the high computational cost and complexity of the process. Additional training during or after pruning can be up to 10 times more expensive than original model training (Lagunas et al., 2021; Xia et al., 2022), and the pruning pipeline often requires rewriting the training code and involves many additional hyperparameters to adjust (Hou et al., 2020; Lan et al., 2019; Liu et al., 2021a; Yao et al., 2021).

Table 5. How few labeled data improves accuracy of our pruning framework KCM on DistilBERT.

!∇	Method	QQP		MNLI		MRPC		QNLI	
		60%	70%	60%	70%	60%	70%	60%	70%
	baseline	89.99		82.11		84.80		88.56	
✗	Mask-Tuning	88.71 ± 0.22	89.66 ± 0.06	80.51 ± 0.19	81.65 ± 0.09	84.73 ± 0.71	84.83 ± 0.35	87.72 ± 0.38	88.43 ± 0.07
✓	**KCM**	88.16 ± 0.03	89.28 ± 0.03	78.05 ± 0.08	80.60 ± 0.05	79.66 ± 0.27	83.01 ± 0.16	85.93 ± 0.09	86.93 ± 0.13
✗	**Extension(512 labeled data)**	88.76 ± 0.25	89.45 ± 0.07	80.02 ± 0.25	81.37 ± 0.11	83.70 ± 1.40	84.49 ± 0.49	87.21 ± 0.54	88.21 ± 0.15
✗	**Extension(1k labeled data)**	88.92 ± 0.20	89.53 ± 0.08	80.41 ± 0.11	81.50 ± 0.12	84.17 ± 0.45	84.68 ± 0.49	87.60 ± 0.31	88.29 ± 0.16
!∇	Method	SST-2		STS-B		$\text{SQuAD}_{1.1}$		$\text{SQuad}_{2.1}$	
		60%	70%	60%	70%	60%	70%	60%	70%
	baseline	91.40		86.12		85.73		68.84	
✗	Mask-Tuning	90.44 ± 0.41	90.93 ± 0.24	85.73 ± 0.07	85.96 ± 0.10	83.20 ± 0.16	84.64 ± 0.09	62.36 ± 1.40	65.32 ± 0.48
✓	**KCM**	88.38 ± 0.25	90.61 ± 0.25	85.26 ± 0.02	85.55 ± 0.03	76.92 ± 0.11	82.65 ± 0.06	64.56 ± 0.11	68.19 ± 0.06
✗	**Extension(512 labeled data)**	89.32 ± 0.54	90.38 ± 0.35	85.83 ± 0.11	86.02 ± 0.07	81.41 ± 0.26	83.30 ± 0.09	66.51 ± 0.24	67.72 ± 0.18
✗	**Extension(1k labeled data)**	89.86 ± 0.56	90.62 ± 0.40	85.90 ± 0.11	86.04 ± 0.06	81.16 ± 0.22	83.34 ± 0.11	66.35 ± 0.32	67.74 ± 0.18

To tackle this, post-training model compression has been recently studied in Kwon et al. (2022); Hubara et al. (2021); Frantar & Alistarh (2022). While Hubara et al. (2021; 2020); Banner et al. (2019) improves post training neural quantization, Kwon et al. (2022) proposed a fast post-training structured pruning framework for Transformers. Even though this approach avoids expensive retraining, it requires labeled data in the pruning pipeline.

Pruning in an unsupervised setting has been studied in Guo et al. (2020b); Browne et al. (2020; 2021) for spiking neural networks and fully-connected layers; however, the pruning either happens during training or still requires retraining of the pruned model. In contrast, our structured pruning method neither requires retraining nor labeled data.

nsupervised: (Guo et al., 2020b) proposed an unsupervised online adaptive weight pruning method that dynamically removes non-critical weights from a spiking neural network (SNN). (Browne et al., 2020; 2021) uses unsupervised k-means clustering to detect clusters of similar filters, and nodes in fully-connected layers, and prunes those that are redundant. (Aghasi et al., 2020): convex post-processing module, which prunes (sparsifies) a trained network layer by layer, while preserving the internal responses.

representative selection problem, low rank decomposition: There has been a lot of work on finding such a representative set (Kazemi et al., 2022; Killamsetty et al., 2021; You et al., 2020) that require training. However since $W_\ell^{(2)} H_\ell^{(1)} + b_\ell^{(2)}$ is a linear function this problem can be reduced to finding convex hull over $W_\ell^{(2)}$.

6. Conclusion

In this work, we studied the problem of structured pruning with unlabeled data and no backward pass. We proposed a gradient-free structured pruning framework that prunes the filters with the help of our proposed *R2D2* that combines two ranking techniques called *Representative Ranking* (*R2*) and *Data-Driven* (*D2*). We empirically evaluated our framework on GLUE and SQuAD benchmarks using BERT_{BASE} and DistilBERT. Compared to when the labeled data is available, our approach achieved up to 40% FLOPs reduction with less than 4% accuracy loss over all tasks considered.

References

Aghasi, A., Abdi, A., and Romberg, J. Fast convex pruning of deep neural networks. *SIAM Journal on Mathematics of Data Science*, 2(1):158–188, 2020.

Banner, R., Nahshan, Y., and Soudry, D. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019.

Baykal, C., Liebenwein, L., Gilitschenski, I., Feldman, D., and Rus, D. Data-dependent coresets for compressing neural networks with applications to generalization bounds. *arXiv preprint arXiv:1804.05345*, 2018.

Browne, D., Giering, M., and Prestwich, S. Pulsenetone: fast unsupervised pruning of convolutional neural networks for remote sensing. *Remote Sensing*, 12(7):1092, 2020.

Browne, D., Giering, M., and Prestwich, S. Unsupervised pulsenet: Automated pruning of convolutional neural networks by k-means clustering. In *International Conference on Machine Learning, Optimization, and Data Science*, pp. 172–184. Springer, 2021.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. Semantic textual similarity-multilingual and cross-lingual focused evaluation. In *Proceedings of the 2017 SEMVAL International Workshop on Semantic Evaluation (2017). https://doi. org/10.18653/v1/s17-2001*, 2017.

Chen, D., Li, Y., Qiu, M., Wang, Z., Li, B., Ding, B., Deng, H., Huang, J., Lin, W., and Zhou, J. Adabert: Task-adaptive bert compression with differentiable neural architecture search. *arXiv preprint arXiv:2001.04246*, 2020a.

Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., and Carbin, M. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020b.

Chen, T., Cheng, Y., Gan, Z., Yuan, L., Zhang, L., and Wang, Z. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021.

Chen, X., Cheng, Y., Wang, S., Gan, Z., Wang, Z., and Liu, J. Earlybert: Efficient bert training via early-bird lottery tickets. *arXiv preprint arXiv:2101.00063*, 2020c.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment: First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, pp. 177–190. Springer, 2006.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ding, C. H., Li, T., and Jordan, M. I. Convex and semi-nonnegative matrix factorizations. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):45–55, 2008.

Dolan, B. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Fakoor, R., Chaudhari, P., Mueller, J., and Smola, A. J. Trade: Transformers for density estimation. *arXiv preprint arXiv:2004.02441*, 2020a.

Fakoor, R., Mueller, J. W., Erickson, N., Chaudhari, P., and Smola, A. J. Fast, accurate, and simple models for tabular data via augmented distillation. *Advances in Neural Information Processing Systems*, 33:8671–8681, 2020b.

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

Frantar, E. and Alistarh, D. Optimal brain compression: A framework for accurate post-training quantization and pruning. *arXiv preprint arXiv:2208.11580*, 2022.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Ganesh, P., Chen, Y., Lou, X., Khan, M. A., Yang, Y., Sajjad, H., Nakov, P., Chen, D., and Winslett, M. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, 9:1061–1080, 2021.

Gu, J., Keller, B., Kossaifi, J., Anandkumar, A., Khailany, B., and Pan, D. Z. Heat: Hardware-efficient automatic tensor decomposition for transformer compression. *arXiv preprint arXiv:2211.16749*, 2022.

Guo, C., Hsueh, B. Y., Leng, J., Qiu, Y., Guan, Y., Wang, Z., Jia, X., Li, X., Guo, M., and Zhu, Y. Accelerating sparse dnn models without hardware-support via tile-wise sparsity. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15. IEEE, 2020a.

Guo, W., Fouda, M. E., Yantir, H. E., Eltawil, A. M., and Salama, K. N. Unsupervised adaptive weight pruning for energy-efficient neuromorphic systems. *Frontiers in Neuroscience*, 14:598876, 2020b.

Ham, T. J., Lee, Y., Seo, S. H., Kim, S., Choi, H., Jung, S. J., and Lee, J. W. Elsa: Hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 692–705. IEEE, 2021.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Hou, L., Huang, Z., Shang, L., Jiang, X., Chen, X., and Liu, Q. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020.

Huang, C., Wu, Y., Min, G., and Ying, Y. Kernelized convex hull approximation and its applications in data description tasks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2018.

Hubara, I., Nahshan, Y., Hanani, Y., Banner, R., and Soudry, D. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518*, 2020.

Hubara, I., Chmiel, B., Island, M., Banner, R., Naor, J., and Soudry, D. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in Neural Information Processing Systems*, 34: 21099–21111, 2021.

Iandola, F. N., Shaw, A. E., Krishna, R., and Keutzer, K. W. Squeezebert: What can computer vision teach nlp about efficient neural networks? *arXiv preprint arXiv:2006.11316*, 2020.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Kazemi, S. M., Tsitsulin, A., Esfandiari, H., Bateni, M., Ramachandran, D., Perozzi, B., and Mirrokni, V. Tackling provably hard representative selection via graph neural networks. *arXiv preprint arXiv:2205.10403*, 2022.

Khetan, A. and Karnin, Z. schubert: Optimizing elements of bert. *arXiv preprint arXiv:2005.06628*, 2020.

Killamsetty, K., Zhao, X., Chen, F., and Iyer, R. Retrieve: Coreset selection for efficient and robust semi-supervised learning. *Advances in Neural Information Processing Systems*, 34:14488–14501, 2021.

Kim, S., Gholami, A., Yao, Z., Mahoney, M. W., and Keutzer, K. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pp. 5506–5518. PMLR, 2021.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kurtic, E., Campos, D., Nguyen, T., Frantar, E., Kurtz, M., Fineran, B., Goin, M., and Alistarh, D. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. *arXiv preprint arXiv:2203.07259*, 2022.

Kwon, W., Kim, S., Mahoney, M. W., Hassoun, J., Keutzer, K., and Gholami, A. A fast post-training pruning framework for transformers. *arXiv preprint arXiv:2204.09656*, 2022.

Lagunas, F., Charlaix, E., Sanh, V., and Rush, A. M. Block pruning for faster transformers. *arXiv preprint arXiv:2109.04838*, 2021.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Lee, M., Han, K., and Shin, M. C. Littlebird: Efficient faster & longer transformer for question answering. *arXiv preprint arXiv:2210.11870*, 2022.

Levesque, H., Davis, E., and Morgenstern, L. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

Li, B., Kong, Z., Zhang, T., Li, J., Li, Z., Liu, H., and Ding, C. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. *arXiv preprint arXiv:2009.08065*, 2020.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Liebenwein, L., Baykal, C., Lang, H., Feldman, D., and Rus, D. Provable filter pruning for efficient neural networks. *arXiv preprint arXiv:1911.07412*, 2019.

Lin, Z., Liu, J. Z., Yang, Z., Hua, N., and Roth, D. Pruning redundant mappings in transformer models via spectral-normalized identity prior. *arXiv preprint arXiv:2010.01791*, 2020.

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

Liu, Z., Li, F., Li, G., and Cheng, J. Ebert: Efficient bert inference with dynamic structured pruning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 4814–4823, 2021b.

Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

Miller, J., Krauth, K., Recht, B., and Schmidt, L. The effect of natural distribution shift on question answering models. In *International Conference on Machine Learning*, pp. 6905–6916. PMLR, 2020.

Mussay, B., Osadchy, M., Braverman, V., Zhou, S., and Feldman, D. Data-independent neural pruning via coresets. *arXiv preprint arXiv:1907.04018*, 2019.

Mussay, B., Feldman, D., Zhou, S., Braverman, V., and Osadchy, M. Data-independent structured pruning of neural networks via coresets. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7829–7841, 2021.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Peer, D., Stabinger, S., Engl, S., and Rodríguez-Sánchez, A. Greedy-layer pruning: Speeding up transformer models for natural language processing. *Pattern Recognition Letters*, 157:76–82, 2022.

Prasanna, S., Rogers, A., and Rumshisky, A. When bert plays the lottery, all tickets are winning. *arXiv preprint arXiv:2005.00561*, 2020.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

Sajjad, H., Dalvi, F., Durrani, N., and Nakov, P. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429, 2023.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Sanh, V., Wolf, T., and Rush, A. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389, 2020.

Shankar, I., Nikhil, D., and Kornel, C. First quora dataset release: question pairs (2017). *URL https://www. quora. com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs*, 2017.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. 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.

Shi, Y., Nguyen, L., Oh, S., Liu, X., Koushan, F., Jameson, J. R., and Kuzum, D. Neuroinspired unsupervised learning and pruning with subquantum cbram arrays. *Nature communications*, 9(1):1–11, 2018.

So, D., Le, Q., and Liang, C. The evolved transformer. In *International Conference on Machine Learning*, pp. 5877–5886. PMLR, 2019.

So, D., Mańke, W., Liu, H., Dai, Z., Shazeer, N., and Le, Q. V. Searching for efficient transformers for language modeling. *Advances in Neural Information Processing Systems*, 34:6010–6022, 2021.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.

Tambe, T., Hooper, C., Pentecost, L., Jia, T., Yang, E.-Y., Donato, M., Sanh, V., Whatmough, P., Rush, A. M., Brooks, D., et al. Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 830–844, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, H., Wu, Z., Liu, Z., Cai, H., Zhu, L., Gan, C., and Han, S. Hat: Hardware-aware transformers for efficient natural language processing. *arXiv preprint arXiv:2005.14187*, 2020a.

Wang, H., Zhang, Z., and Han, S. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 97–110. IEEE, 2021.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020b.

Wang, Z., Wohlwend, J., and Lei, T. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

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

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45. Association for Computational Linguistics, 2020.

Wu, Z., Liu, Z., Lin, J., Lin, Y., and Han, S. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020.

Xia, M., Zhong, Z., and Chen, D. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

Xin, J., Tang, R., Lee, J., Yu, Y., and Lin, J. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020.

Xu, J., Tan, X., Luo, R., Song, K., Li, J., Qin, T., and Liu, T.-Y. Nas-bert: task-agnostic and adaptive-size bert compression with neural architecture search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1933–1943, 2021.

Yao, Z., Ma, L., Shen, S., Keutzer, K., and Mahoney, M. W. Mlpruning: A multilevel structured pruning framework for transformer-based models. *arXiv preprint arXiv:2105.14636*, 2021.

Yin, Y., Chen, C., Shang, L., Jiang, X., Chen, X., and Liu, Q. Autotinybert: Automatic hyper-parameter optimization for efficient pre-trained language models. *arXiv preprint arXiv:2107.13686*, 2021.

You, C., Li, C., Robinson, D., and Vidal, R. Self-representation based unsupervised exemplar selection in a union of subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Zadeh, A. H., Edo, I., Awad, O. M., and Moshovos, A. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 811–824. IEEE, 2020.

Zafrir, O., Boudoukh, G., Izsak, P., and Wasserblat, M. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pp. 36–39. IEEE, 2019.

Zhang, Q., Zuo, S., Liang, C., Bukharin, A., He, P., Chen, W., and Zhao, T. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International Conference on Machine Learning*, pp. 26809–26823. PMLR, 2022.

Zhou, W., Xu, C., Ge, T., McAuley, J., Xu, K., and Wei, F. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341, 2020.

A. Experimental Details

We implemented our framework with PyTorch (Paszke et al., 2019) using the HuggingFace Transformers (Wolf et al., 2020) library. We fine-tuned the pre-trained checkpoints of the BERT$_{BASE}$ (Devlin et al., 2018) and DistilBERT (Sanh et al., 2019) downloaded from the HuggingFace repository on GLUE (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2018; 2016) benchmarks.

GLUE (Wang et al., 2018) includes following tasks. 1)Sentence similarity (QQP (Shankar et al., 2017), MRPC (Dolan & Brockett, 2005), STS-B (Cer et al., 2017)) with 364K, 4k and 6k training examples. 2) Sentiment classification (SST-2 (Socher et al., 2013)) with 67K training example, 3)Textual entailment (RTE (Dagan et al., 2006)) with 3K training examples. 4) Natural language inference (MNLI (Williams et al., 2017), QNLI (Rajpurkar et al., 2016)) with 392K, 105K training examples. We exclude CoLA (Warstadt et al., 2019) and WLNI (Levesque et al., 2012) due to their unstable behaviors. SQuAD 1.1 (Rajpurkar et al., 2016) and SQuAD 2.0 (Rajpurkar et al., 2018) are question and answering tasks, each of which contains 88K and 130K training examples. SQuAD$_{2.0}$ is an extension of SQuAD$_{1.1}$ by including unanswerable questions whose answers are not stated in the given contexts.

All results are the averaged over the runs with 10 different seeds. For SQuAD tasks we report F1 score and for all GLUE tasks except STS-B we report accuracy. For STS-B we report Spearman Correlation.

For Pruning we only use 2K raw data from the training sets. Note that we only use the input features and not their labels. For our Representative Ranking calculation, in Algorithm 2, the width of Gaussian kernel σ, and convergence rate α are the hyperparameters. In our experiments, we set $\sigma = 1.0$ and $\alpha = 0.01$. On the average it only takes less than 20 iterations to converge.

B. More Experimental Results

B.1. Comparison with Gradient-free, Retrain-free, Supervision-free Baselines

Figure 5 shows the performance of KCM against Weight-magnitude, and layer-wise Output-magnitude using BERT$_{BASE}$ on GLUE and SQuAD tasks. All these methods are gradient-free (no backward pass), retrain-free (no retrain), supervision-free (no labeled data), and run in a matter of minutes. Clearly, KCM outperforms across all tasks considered.

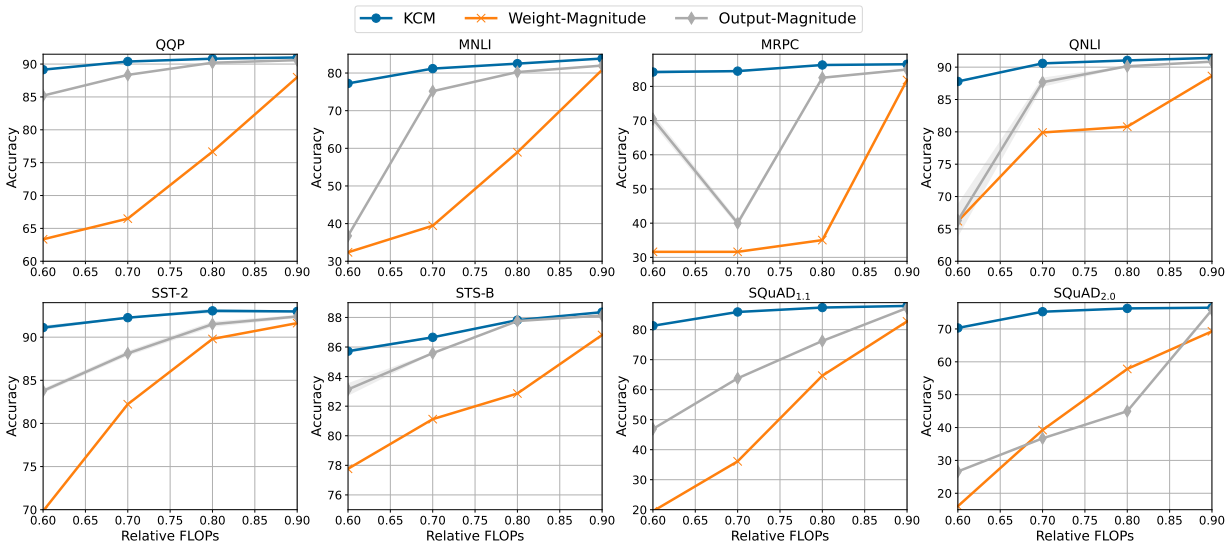

Figure 5. KCM outperforms Weight-magnitude, and outputs-magnitude across all GLUE and SQuAD tasks.

B.2. Impact of Unlabeled Data Sample Size

Figure 6 shows how the performance of KCM on DistilBERT changes while we varied unlabeled data sample size.

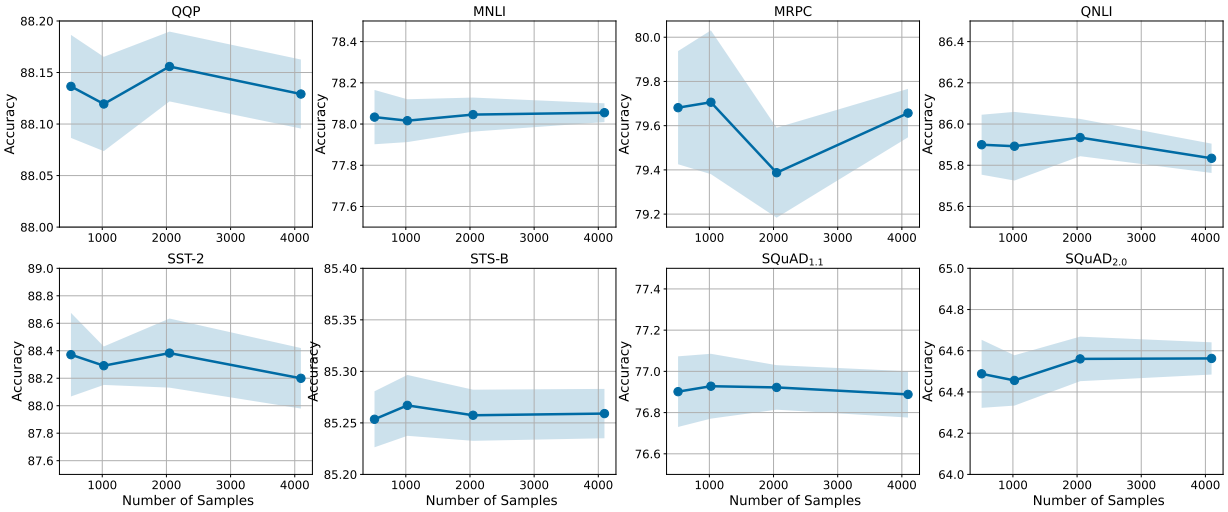

Figure 6. Performance of KCM on DistilBERT by varying unlabeled data sample size.

B.3. Speedup

We evaluated the latency on real hardware and obtained the speedup of KCM on BERT$_{BASE}$ on a single NVIDIA V100 GPU for 60% Flops constraint:

Table 6. Speedup of KCM on BERT$_{BASE}$ on a single NVIDIA V100 GPU for 60% Flops constraint.

Method	QQP	MNLI	MRPC	QNLI	SST-2	STS-B	SQuAD$_{1.1}$	SQuAD$_{2.0}$
speedup	1.58x	1.46x	1.53x	1.57x	1.58x	1.59x	1.47x	1.44x

C. SQuAD$_{1.1}$ Task on BERT$_{LARGE}$

We conducted additional experiments on a larger-scale model, BERT$_{LARGE}$ over SQuAD$_{1.1}$. The results in Table 7 indicate that our method outperforms unsupervised baselines, providing further evidence of its efficacy.

Table 7. KCM outperforms Weight-magnitude, and Weight-magnitude-Scale on BERT$_{LARGE}$ over SQuAD$_{1.1}$.

Method	60%	70%	80%	90%
Weight-Magnitude (Li et al., 2016)	2.3029	2.8365	4.5763	52.5013
Weight-Magnitude-Scale	2.3178	24.6811	83.6422	91.4438
KCM (ours)	$\mathbf{85.86 \pm 0.14}$	$\mathbf{88.57 \pm 0.06}$	$\mathbf{91.40 \pm 0.04}$	$\mathbf{92.72 \pm 0.03}$

Table 8. How few labeled data improves accuracy of our pruning framework KCM on BERT$_{LARGE}$ for SQuAD$_{1.1}$.

Method	60%	70%	80%	90%
KCM	85.86 ± 0.14	88.57 ± 0.06	91.40 ± 0.04	92.72 ± 0.03
Extension(512 labeled data)	89.44 ± 0.1	91.85 ± 0.06	92.40 ± 0.05	93.00 ± 0.01
Extension(1k labeled data)	89.48 ± 0.16	91.92 ± 0.07	92.69 ± 0.02	93.17 ± 0.04

D. Train-Test Data Discrepancy

We ran new experiments with a new dataset called `new-Wiki` to further evaluate the effectiveness of the proposed method under training-test data discrepancy. As outlined in Miller et al. (2020), `new-Wiki` is different from the original SQuAD$_{1.1}$ dataset and was generated using the Wikipedia dataset.

We explored various scenarios involving the sampling of unlabeled data from datasets that differ from the evaluation dataset. In particular, we sample unlabeled data from 1) SQuAD$_{1.1}$-train 2) SQuAD$_{1.1}$-val or 3) new-Wiki and evaluate on SQuAD$_{1.1}$-val. As evident from the results in Table 9, sampling unlabeled data from new-Wiki (and evaluating on SQuAD$_{1.1}$-val) yielded improved performance compared to sampling from SQuAD$_{1.1}$-train and evaluating on SQuAD$_{1.1}$-val. This finding further supports our assertion regarding the applicability and effectiveness of our approach.

Table 9. Train-Test data discrepancy

Unlabeled Sample	Evaluation	60%	70%
SQuAD$_{1.1}$-train	SQuAD$_{1.1}$-val	76.92 ± 0.11	82.65 ± 0.06
SQuAD$_{1.1}$-val	SQuAD$_{1.1}$-val	77.17 ± 0.073	82.75 ± 0.074
new-Wiki	SQuAD$_{1.1}$-val	77.45 ± 0.076	82.80 ± 0.031

We further evaluate the effectiveness of our approach on using the finetuned model on SQuAD$_{1.1}$ but evaluate on new-Wiki and results are as follows:

Table 10. Performance of Finetuned model on SQuAD$_{1.1}$ on new-Wiki dataset.

Unlabeled Sample	Evaluation	60%	70%
SQuAD$_{1.1}$-train	new-Wiki	75.26 ± 0.08	81.14 ± 0.096
new-Wiki	new-Wiki	75.72 ± 0.09	81.08 ± 0.13