Layer-wise Model Pruning based on Mutual Information

Chun Fan[♠⋆▲], **Jiwei Li**[♦♣], **Xiang Ao**[▼],
Fei Wu[♦], **Yuxian Meng**[♣] and **Xiaofei Sun**[♣]
[♣] Shannon.AI, [♦]Zhejiang University
[▲]National Biomedical Imaging Center, Peking University
[♠]Computer Center of Peking University, [⋆]Peng Cheng Laboratory
[▼]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences
{yuxian_meng, jiwei_li, xiaofei_sun}@shannonai.com
fanchun@pku.edu.cn, aoxiang@ict.ac.cn, wufei@zju.edu.cn

Abstract

Inspired by mutual information (MI) based feature selection in SVMs and logistic regression, in this paper, we propose MI-based layer-wise pruning: for each layer of a multi-layer neural network, neurons with higher values of MI with respect to preserved neurons in the upper layer are preserved. Starting from the top softmax layer, layer-wise pruning proceeds in a top-down fashion until reaching the bottom word embedding layer. The proposed pruning strategy offers merits over weight-based pruning techniques: (1) it avoids irregular memory access since representations and matrices can be squeezed into their smaller but dense counterparts, leading to greater speedup; (2) in a manner of top-down pruning, the proposed method operates from a more global perspective based on training signals in the top layer, and prunes each layer by propagating the effect of global signals through layers, leading to better performances at the same sparsity level. Extensive experiments show that at the same sparsity level, the proposed strategy offers both greater speedup and higher performances than weight-based pruning methods (e.g., magnitude pruning, movement pruning).

1 Introduction

In spite of impressive results of neural networks, the huge model size has hindered their applications in cases where computation and memory resources are limited.[1] As a result, training and using existing huge models not only requires rich hardware resources, but also consumes high environmental costs (Strubell et al., 2019).

Model pruning, reduces model sizes by dropping a fraction of the model parameters, to reduce computation intensity and memory footprint of large models at the lowest cost of accuracy on end tasks (Joulin et al., 2016; Ganesh et al., 2020; Gordon et al., 2020). Among pruning techniques, weight based pruning is a widely-used group of methods. It focuses on removing weights according to their importance under different specific criteria, e.g., the magnitude (Han et al., 2015b,a), first-order derivative (Lee et al., 2018; Sanh et al., 2020) and second-order derivative information (LeCun et al., 1990; Hassibi and Stork, 1993), and it has been successfully applied to a large variety of model architectures (Guo et al., 2016; Gale et al., 2019; Molchanov et al., 2019) and downstream tasks (McCarley, 2019; Gordon et al., 2020).

While weight-based methods have been successfully applied to a wide range of neural models for model pruning, they come with the following shortcomings: (1) weights in matrices are pruned irregularly, which lead to irregular memory access, resulting in runtime inefficiency; (2) weight matrices are pruned independently, and this neglect of global supervision from training signals at the top layer and ignorance of information propagation between consecutive layers may result in sub-optimality of pruned networks.

In this paper, inspired by mutual information (MI) based feature selection (Kuncheva, 2007) in SVMs and logistic regression, we propose MI based layer-wise pruning, to address the aforementioned drawbacks of weight-based pruning methods in NLP. For each layer of a multi-layer neural network, neurons with higher values of MI with respect to the preserved neurons in the upper layer are preserved. Starting from the top softmax layer, layer-wise pruning proceeds until reaching the bottom input word embedding layer in a top-down fashion. Once the preserved neurons in each layer are selected, the redundant dimensions along with the corresponding rows and columns of the weight matrices can be pruned or squeezed, inducing model sparsity at different levels.

[1]For example, the GPT-3 model (Brown et al., 2020) has 175B parameters in total, with 96 layers and 96 attention heads (Vaswani et al., 2017) per layer.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3079–3090
November 7–11, 2021.

The proposed pruning strategy naturally addresses the aforementioned two shortcomings of weight-based methods: (1) it avoids irregular memory access since it squeezes the pruned representations and matrices into their smaller but dense counterparts. This enables significantly faster computations than weight-based pruning methods at the same sparsity level; (2) rather than viewing each weight matrix separately based on their own weight values, the proposed method operates from a more global perspective based on training signals at the top layer, and prunes each layer by propagating the effect of global training signals through consecutive layers in a top-down fashion. This leads to better performances at the same sparsity level.

We conduct extensive experiments on both generative tasks (MT) and discriminative tasks (question answering) in NLP to examine the effectiveness of the proposed strategy. We show that compared to weight-based pruning methods including magnitude pruning (Han et al., 2015b), movement pruning (Sanh et al., 2020) and L_0 pruning (Louizos et al., 2017), the proposed method yields greater speedup along with better performances for the same sparsity levels on generative NLP tasks of WMT'14 En→Fr and WMT'14 En→De, and discriminative NLP tasks of SQuAD v1.1 (Rajpurkar et al., 2016), MNLI (Williams et al., 2017) and SST-5 (Socher et al., 2013). In addition, we also show that the proposed method serves the feature selection purposes, where we observe significant performance boosts when fixing preserved neurons and relearning the pruned ones, leading to a state-of-the-art performance of 43.9 BLEU score for En→Fr translation in setups without back-translation or external data.

2 Related Work

2.1 Model Pruning

Generic Model Pruning Model pruning refers to reducing the model size by dropping a fraction of the model parameters, which dates back to early works of Optimal Brain Damage (PBD) (LeCun et al., 1990) and Optimal Brain Surgeon (OBS) (Hassibi and Stork, 1993). One major branch of neural model pruning methods is magnitude pruning (Han et al., 2015b; See et al., 2016; Narang et al., 2017; Molchanov et al., 2019; Gale et al., 2019; Frankle et al., 2020), which prunes model parameters measured by their importance scores. Han et al. (2015b) removed all parameters with weight values below a threshold, and then retrained the remaining sparse network. Guo et al. (2016) proposed *dynamic network surgery*, allowing for model connection recovery from incorrect pruning decisions made in previous iterations. Michael H. Zhu (2018) adopted a gradual pruning schedule, in which the sparsity level increases from an initial sparsity value to a specified final sparsity value during training. Other methods for neural model pruning include L_0 regularization pruning (Louizos et al., 2017), variational dropout pruning (Kingma et al., 2015; Molchanov et al., 2017; Gomez et al., 2019) and movement pruning (Sanh et al., 2020), etc. Recent works have proposed a line of techniques to prune and produce sparsity in a structured way (Anwar et al., 2017; Zhou et al., 2016; Hu et al., 2016; Liu et al., 2019b), which aims at pruning full convolutional filters or whole layers. Methods for structured pruning mainly include group Lasso (Alvarez and Salzmann, 2016; Wen et al., 2016; He et al., 2017), sparsity regularization (Li et al., 2016; Liu et al., 2017; Huang and Wang, 2018; Gordon et al., 2018) and automatic network searching (He et al., 2018; Yu and Huang, 2019; Dong and Yang, 2019; Ding et al., 2019).

Pruning Transformers Pruning Transformer based models has been of growing interest (Guo et al., 2019; Chen et al., 2020; Li et al., 2020). Fan et al. (2019) proposed *LayerDrop* to reduce Transformer depth. Michel et al. (2019) proposed to use *head importance score* to prune BERT attention heads. Attention heads can also be pruned by using L_0 regularization (Voita et al., 2019) and cascade pruning (Wang et al., 2021). Wang et al. (2020) combined L_0 regularization with matrix factorization to prune BERT. Gordon et al. (2020) proposed that BERT can be pruned once during pre-training rather than separately for each task without sacrificing performance.

2.2 Mutual Information Feature Selection

Feature selection is the process of selecting a proper subset of features for better model performances (Kira and Rendell, 1992; Guyon and Elisseeff, 2003; Chandrashekar and Sahin, 2014; Bolón-Canedo et al., 2016; Cai et al., 2018). A widely used method for feature selection is *Mutual Information Based Feature Selection* (Vergara and Estévez, 2014; Liu et al., 2009; Beraha et al., 2019), which selects features that minimize the redundancy and maximize the relevance w.r.t. the target variable. Various approaches including minimum-Redundancy-Maximum-Relevance (mRMR) (Estévez et al., 2009; Brown et al., 2012; Bennasar et al., 2015) are proposed to accurately select features.

3 Model

3.1 Overview for Model Pruning

Given a set of inputs $\mathcal{M} = \{(X, Y)\}$, where each input is a word sequence $X = \{x_1, ..., x_t, ..., x_{N_x}\}$ and N_x denotes the length of the input, our goal is to predict the label(s) for X, denoted by Y.

In a standard multi-layer neural network setup, the input layer first maps each input word x_t to a vector representation $h_t^0 \in \mathbb{R}^{D\times 1}$, where D denotes the dimensionality. On top of the input layer, the model stacks L intermediate neural layers. Let $h_t^l \in \mathbb{R}^{D\times 1}$ denote the representation for token x_t at the l^{th} layer. $H^l \in \mathbb{R}^{D\times N}$ is the concatenation of representations at the l^{th} layer for all tokens in the input X. Each layer of the network involves multiple operations such as fully connected operations, ReLU, self-attentions or residual connections. The group of all operations within layer l is denoted by F_l, which maps H^l to H^{l+1}:

$$H^{l+1} = F_l(H^l) \tag{1}$$

The output from the last layer h_t^L is fed to the final softmax layer for predictions. To prune a neural network model, let $m^l \in \{0, 1\}^{D\times 1}$ denote the mask for representation dimensions at layer l. The number of 1s in m^l is a pre-defined hyper-parameter, denoted by K, controlling the sparsity of the network. $M^l \in \{0, 1\}^{D\times N}$ makes N copies of m^l, making the dimensionality of the mask the same as that of layer representations for X. Let u^l denote the set of indexes for preserved dimensions, where $m^l[j \text{ for } j \text{ in } u^l] = 1$. Eq.(1) can be rewritten as:

$$H^{l+1} = F_l(H^l \otimes M^l) \tag{2}$$

where $\otimes$ is the Hadamard product. We need special attentions for the uppermost softmax layer. No dimension should be pruned for this layer since each dimension corresponds to an output label. $m^{\text{softmax}} = [1]^{|\mathcal{Y}|}$, where $|\mathcal{Y}|$ denotes the size of the output label set.

3.2 Layer-wise Pruning

The key point of layer-wise pruning is to construct correlations between dimensions in two consecutive layers $l-1$ and l. Then based on the correlations, we can prune the network in a top-down fashion: with respect to output labels in the final softmax layer, we select the top K correlated dimensions in the L^{th} layer based on the correlation measure, zeroing out the rest. Let $I(A, B)$ denote correlation between two set of dimensions:

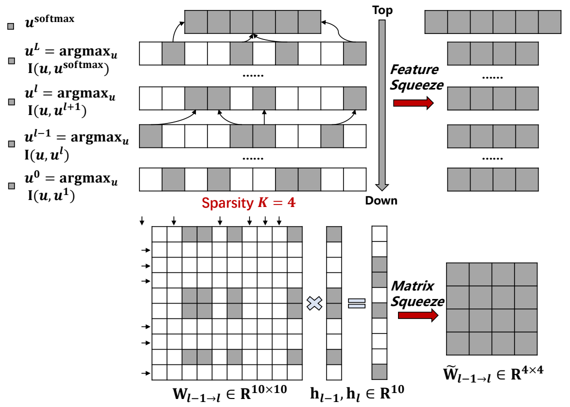

Figure 1: An overview of the proposed layer-wise pruning method. The top part shows pruning at the feature level, and the bottom part shows the weight matrix level pruning. Layer-wise pruning first selects feature dimensions in each layer regarding some correlation criterion $I(\cdot,\cdot)$, and then prunes matrix rows and cols according to the selected dimensions at consecutive layers, after which both features and matrices can be squeezed.

$$u^L = \arg\max_u I(u, u^{\text{softmax}}) \ \text{ s.t. } \ |u^L| = K \tag{3}$$

Next, we go to the $(L-1)^{\text{th}}$ layer, preserving dimensions in the $(L-1)^{\text{th}}$ layer that are most correlated with preserved dimensions in the L^{th} layer

$$u^{L-1} = \arg\max_u I(u, u^L) \ \text{ s.t. } \ |u^{L-1}| = K \tag{4}$$

This process proceeds until the bottom input embedding layer. An illustration of the proposed layer-wise pruning method is show in Figure 1. Algorithm 1 describes the pruning process.

3.3 Mutual Information between Dimensions

Here, we describe quantitative ways to compute correlation scores $I(A, B)$ between dimensions in layer $l-1$ and layer l using MI.

3.3.1 MI for Dimension Selection

Mutual information (MI) is a measure between two random variables to quantify the amount of information obtained about one variable through the other variable. In our case, we wish to compute the

Algorithm 1: Layer-wise Pruning

Input : A trained model F before pruning; the correlation function between two sets of dimensions $I(\cdot,\cdot)$; a specified sparsity K; u^{softmax}

Output : Sets of indexes for preserved dimensions $u^1, \cdots, u^L$ in each layer

$u^L = \arg\max_u I(u, u^{\text{softmax}})$ s.t. $|u^L| = K$;

```
// Top-down layer-wise pruning
```

for $i \leftarrow L-1$ **to** 0 **do**

  $u^i = \arg\max_u I(u, u^{i+1})$ s.t. $|u^i| = K$;

end

MI between dimensions u^l at layer l and dimensions u^{l-1} at layer $l-1$. Let $v_{d_k^l}$ denote the variable for the neuron value of the d_k^l-th dimension at the l^{th} layer. MI between u^l and u^{l-1} is given by:

$$I(u^l,u^{l-1}) = H(u^l) - H(u^l|u^{l-1}) \tag{5}$$

To tangibly compute Eq.(5), we make assumptions that both $v_{d_1^l}, ..., v_{d_K^l}$ and $v_{d_1^{l-1}}, ..., v_{d_K^{l-1}}$ are samples from Gaussian distributions:

$$\begin{aligned} v_{d_1^l}, ..., v_{d_K^l}, v_{d_1^{l-1}}, ..., v_{d_K^{l-1}} &\sim \mathcal{N}(\eta_{u^l}^{l-1,l}, \Sigma_{u^l}^{l-1,l}) \\ v_{d_1^l}, ..., v_{d_K^l} &\sim \mathcal{N}(\eta_{u^l}^{l}, \Sigma_{u^l}^{l}) \\ v_{d_1^{l-1}}, ..., v_{d_K^{l-1}} &\sim \mathcal{N}(\eta_{u^l}^{l-1}, \Sigma_{u^l}^{l-1}) \end{aligned} \tag{6}$$

where $\eta_{u^l}^{l-1,l} \in \mathbb{R}^{2K\times 1}$; $\eta_{u^l}^{l-1}, \eta_{u^l}^{l} \in \mathbb{R}^{K\times 1}$; $\Sigma_{u^l}^{l-1,l} \in \mathbb{R}^{2K\times 2K}$; $\Sigma_{u^l}^{l-1}, \Sigma_{u^l}^{l} \in \mathbb{R}^{K\times K}$. η and Σ can be estimated using maximum likelihood. Specifically, for all $(X, Y) \in \mathcal{M}$, we first compute the neuron values for all instances for all layers. $\eta_{u^l}^l$ and $\Sigma_{u^l}^l$ are given as follows:

$$\begin{aligned} \eta_{u^l}^l &= \frac{1}{\sum_{X\in\mathcal{M}} |N_x|} \sum_{X\in\mathcal{M}} \sum_{t\in[1,N_x]} v_{t,d_{u^l}^l} \\ \Sigma_{u^l}^l &= \frac{1}{\sum_{X\in\mathcal{M}} |N_x|} \sum_{X\in\mathcal{M}} \sum_{t\in[1,N_x]} \\ &\qquad (v_{t,d_{u^l}^l} - \eta^l)^\top (v_{t,d_{u^l}^l} - \eta^l) \end{aligned} \tag{7}$$

where $v_{t,d_{u^l}^l}$ is a vector of length K, corresponding to a sub-vector within h_t^l with dimension u^l. $\eta_{u^l}^{l-1}$, $\eta_{u^l}^{l,l-1}$, $\Sigma_{u^l}^{l-1}$, $\Sigma_{u^l}^{l,l-1}$ can be computed similarly.

It is worth noting that the proposed model relies on the Gaussian assumption for MI computations, and several recent efforts have been proposed to release this strong assumption, such as training independent neural nets to estimate MI (Belghazi et al., 2018), using variational distributions to approximate the distribution (Cheng et al., 2020; Poole et al., 2019). These workarounds to avoid the Gaussian assumption requires learning another model (an independent neural model in Belghazi et al. (2018) and variational distributions in Cheng et al. (2020)) through gradient updates, and thus cannot be adapted to the scale in our situation, where we have to estimate MI for all dimensions across all layers. The adopted Gaussian model is efficient in estimating MI values in bulk, and achieve satisfying performances. We leave how to relax this assumption to future work.

3.3.2 Greedy Selection

Selecting u^l based on Eq.(5) is an NP-hard optimization problem, because the set of possible combinations of dimensions grows exponentially since there are C_D^K combinations of dimensions (D is the dimension of vector and K is the number of dimensions to pick). We thus turn to a greedy forward step-wise selection strategy, a widely used strategy in mutual-information based feature selection. Specifically, let $u_{(k)}^l$ be the set of selected dimensions at time step $k \leq K$. At each time step, we incrementally add one dimension d_k^l to $u_{(k-1)}^l$ by selecting the dimension that leads to the biggest increase. We repeat this process K times:

$$d_k^l = \underset{d\notin u_{(k-1)}^{l-1}}{\arg\max}\, I(u^l, u_{(k-1)}^{l-1} \cup d) \tag{8}$$

Inspired by Brown et al. (2012), further assumptions are made that the selected dimensions are independent and class-conditionally independent given unselected features, transforming Eq.(8) to the following form:

$$\begin{aligned} d_k^l = \underset{d\notin u_{(k-1)}^{l-1}}{\arg\max} \{ I(u^l, d) - \\ [\alpha I(d, u_{(k-1)}^l) - \beta I(d, u_{(k-1)}^l | u^l)] \} \end{aligned} \tag{9}$$

It is straightforward to see that the first part of Eq.(9), i.e., $I(u^l, d)$ models the relevance of selected dimensions, against the redundancy compared to the dimensions already selected , manifested in the second and the third part. The model degenerates to the model of Maximum Relevancy Minimum Redundancy (mRMR) (Peng et al., 2005) when $\beta = 0$.

3.3.3 Squeezing Weights and Features

For weight matrixes W and feature H^l involved in the matrix manipulation WH^l, we do not need to actually compute the Hadamard product in Eq2. Instead, for H, we squeeze all preserved dimensions to the left side and truncate the rest. For W, rows and columns that correspond to pruned dimensions will be erased and the remaining dimensions will be squeezed. For example, with $m^l = [1, 1, 0, 1]$ and $m^{l+1} = [0, 1, 1, 1]$, the third row and first column of the original matrix $W = [w_{ij}]$ can be pruned, the result of which is squeezed into a smaller matrix:

$$W = \begin{bmatrix} \cancel{w_{11}} & w_{12} & w_{13} & w_{14} \\ \cancel{w_{21}} & w_{22} & w_{23} & w_{24} \\ \cancel{w_{31}} & \cancel{w_{32}} & \cancel{w_{33}} & \cancel{w_{34}} \\ \cancel{w_{41}} & w_{42} & w_{43} & w_{44} \end{bmatrix} \Rightarrow \begin{bmatrix} w_{12} & w_{13} & w_{14} \\ w_{22} & w_{23} & w_{24} \\ w_{42} & w_{43} & w_{44} \end{bmatrix} \tag{10}$$

This avoids irregular memory accesses and thus can significantly speed up matrix-vector product. Figure 1 gives a tangible illustration.

3.4 Iterative Pruning

Instead of aggressively reducing dimensions from D to K in only one iteration, iterative pruning (Han et al., 2015b) gradually reduces model dimensions in multiple steps: in each iteration, pruning is followed by model retraining using preserved dimensions. As we will show in experiments, this strategy achieves better performances than the single-step pruning with the same sparsity levels.

3.5 Retraining Pruned Dimensions

The proposed MI based pruning strategy can not only be used for reducing model size, but also for improving model performances. We can view the MI pruning model from a feature selection perspective: given fixed size of features (where we view each neural dimension as a feature), we wish that all features in each neural layer be informative and relevant. To this end, we can first remove redundant or irrelevant features, add new features, retrain the model, and repeat this process. This strategy is akin to feature selection methods in SVMs or logistic regression (Kuncheva, 2007).

In the neural setup, we can achieve this goal by (1) pruning irrelevant dimensions; (2) reinitializing pruned dimensions (adding new features); and (3) retraining the model. Preserved dimensions and weight matrices are fixed during model retraining, and we only update pruned dimensions. We report the performances of pruning and retraining 60% dimensions. It is worth noting that the strategy of retraining pruned dimensions does not serve as the goal of speedup and model compressing, as pruned dimensions are relearned, making the model of the same size as the model before pruning. We as view retraining pruned dimensions as a byproduct of the pruning, with the goal of improving performances.

3.6 Discussions

For the Wh matrix multiplication in neural models, we refer to W as weights, and h as features. Weight-based methods (Han et al., 2015a,b) prune networks based on values of W, removing features with smaller weights, which are comparable to $L1$ or $L2$ regularizers for feature selection (Ng, 2004; Ravikumar et al., 2010). MI-based pruning method is comparable to MI based feature selection, which attaches attentions to the features by measuring feature-label correlations (Kuncheva, 2007; Yu et al., 2008).

4 Experiments

We conduct experiments on both generative and discriminative NLP tasks. For generative tasks, we conduct experiments on WMT14 En-Fr and WMT14 En-DE. The WMT14 En-Fr dataset consist of 36M and is split into 32000 word-piece vocabulary. The WMT 2014 En-DE dataset consisting of about 4.5 million sentence pairs. We use BPE (Sennrich et al., 2016b) to maintain a source-target vocabulary of 37,000. We use Transformers (Vaswani et al., 2017) as the model backbone. We use En-Fr to perform comprehensive analysis where we use four model setups: extra-large, large, base and tiny. The model statistics are shown in Table 1. It is worth noting that the large and base models are identical to models in Vaswani et al. (2017). We train different models with 16 V100 GPUs with 32G memories. We follow protocols in Vaswani et al. (2017). Adam (Kingma and Ba, 2014) is used for all models with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-6}$. A dropout rate of 0.1 is applied to all layers across all models, and the strategy of label smoothing (Szegedy et al., 2016) is used with smoothing value set to 0.1.[2] We use beam search with a beam size of 20, with no penalty on length. We report BLEU scores based on `multi-bleu.perl` of single models (no ensemble), average floating-point operations (FLOPs), and average practical speedup.

[2] Since our goal is to test the performances of different pruning techniques in the vanilla supervised setup, no advanced MT techniques such as backtranslation (Sennrich et al., 2016a; Edunov et al., 2018), self-learning (He et al., 2020; Sun et al., 2020), data noising (Xie et al., 2017; Bengio et al., 2015), nearest neighbor search (Khandelwal et al., 2020; Meng et al., 2021; Zheng et al., 2021) are used.

Model	d_{model}	d_{ff}	L	H	# Params
Extra-Large	2,048	8,192	8	16	1.1B
Large	1,024	4,096	6	16	275M
Base	512	2,048	6	8	93M
Tiny	256	1,024	6	8	35M

Table 1: Model statistics. d_{model}, d_{ff}, L and H respectively denote input/output dimensionality, inner-layer dimensionality, # layers and # heads.

For discriminative tasks, we followed the current trend of LM pretraining (Devlin et al., 2018; Liu et al., 2019a; Jiao et al., 2019; Radford et al., 2019; Lan et al., 2019; Brown et al., 2020; Clark et al., 2020; Sun et al., 2021). We test different pruning models on the tasks of question answering (Rajpurkar et al., 2016, 2018), natural language inference (Bowman et al., 2015; Williams et al., 2017) and text classification (Socher et al., 2013; Tang et al., 2014; Howard and Ruder, 2018; Chai et al., 2020; Lin et al., 2021). We use BERT (Devlin et al., 2018) as the backbone, and fine-tune BERT on different datasets. Adam (Kingma and Ba, 2014) is used for all models, with batch size, learning rate and the number of epochs treated as hyper-parameters to be tuned on the dev set. We compare the proposed strategy with the following weight based pruning models:

- *Magnitude Pruning* (Han et al., 2015b): removing weights based on their absolute weight values.
- *Movement Pruning* (Sanh et al., 2020): removing weights based on the first-order derivative.
- *L0 Pruning* (Louizos et al., 2017): using the L_0 loss to regularize the number of non-zero weights.

4.1 MT Results

MT results are shown in Tables 2 and 3. Observations can be summarized as follows: (1) When comparing with movement and magnitude pruning, at the same levels of sparsity, the proposed MI method yields greater speedup. This is due to the fact that using MI, the weight matrix W can be squeezed avoiding irregular memory accesses. For magnitude and movement pruning: though W is sparse, pruned dimensions in W are scattered and irregular memory accesses are inevitable.

(2) The MI model yields not only speedup but also performance boosts: we find that the proposed MI pruning consistently works better, both in the low-sparsity and high-sparsity situations. This is because the mutual information strategy provides a more global feature (dimension) selection strategy based on the output label, rather than focusing on the local matrix weights in matrix manipulations. Regarding magnitude pruning and movement pruning, we find that movement pruning underperforms magnitude pruning at lower sparsity levels but works better at higher sparsity levels.

Model	BLEU	FLOPs	Speedup	# Params
	Original Models			
Extra-Large	43.3	100%	1	100%
Large	41.8	24 %	× 2.7	25%
Base	37.9	4.2%	× 8.6	8.5%
Tiny	32.4	2.3%	× 13.7	3.2%
	Without Retraining: Pruning Extra-Large			
MI (to large)	42.4	22%	× 2.6	25%
MI (to base)	39.6	4.4%	× 8.8	8.5%
MI (to tiny)	34.9	2.1%	× 13.6	3.2%
Magnitude (to large)	41.7	23%	× 2.1	25%
Magnitude (to base)	37.3	4.1%	× 4.5	8.5%
Magnitude (to tiny)	32.3	2.3%	× 7.5	3.2%
Movement (to large)	42.0	24%	× 1.9	25%
Movement (to base)	38.2	4.6%	× 4.7	8.5%
Movement (to tiny)	33.6	2.6%	× 6.1	3.2%
L0 (to large)	42.0	25%	× 2.1	25%
L0 (to base)	38.0	3.9%	× 3.9	8.5%
L0 (to tiny)	33.8	2.3%	× 5.8	3.2%
	Without Retraining: Pruning Large			
MI (to base)	38.6	4.1%	× 8.5	8.5%
MI (to tiny)	33.6	2.4%	× 14.1	3.2%
Magnitude (to base)	38.3	4.5%	× 4.0	8.5%
Magnitude (to tiny)	32.7	2.6%	× 6.5	3.2%
Movement (to base)	38.1	4.8%	× 4.7	8.5%
Movement (to tiny)	33.3	2.4%	× 8.3	3.2%
L0 (to base)	38.2	4.4%	× 4.6	8.5%
L0 (to tiny)	32.8	2.9%	× 6.9	3.2%
	Without Retraining: Pruning Base			
MI (to tiny)	33.1	2.3%	× 13.5	3.2%
Magnitude (to tiny)	32.5	2.5%	× 8.4	3.2%
Movement (to tiny)	32.8	2.7%	× 8.7	3.2%
L0 (to tiny)	32.7	2.4%	× 6.9	3.2%
	Retraining Pruned Dimensions			
MI+Extra-Large	43.9 (+0.6)	100%	1	100%
MI+Large	42.3 (+0.5)	24 %	× 2.7	25%
MI+Base	38.4 (+0.5)	4.2%	× 8.6	8.5%

Table 2: Test results for WMT14 En-Fr. "MI" stands for the propose MI based pruning method, "Magnitude" stands for magnitude pruning, "Movement" stands for movement pruning and "L0" stands for L0 pruning. *to* X means pruning the original model to X, and X is thus smaller than the original model. 60% dimensions are pruned and then retrained for the retraining setup.

(3) Based on MI, training a big model and then pruning it to a smaller one outperforms directly training a smaller model of the same size, e.g., pruning extra-large to large yields a BLEU score of 42.4 for En-Fr, which is +0.6 higher than vanilla large (41.8). This is also the case with pruning extra-large to base and tiny, and pruning large to base and tiny. The explanations are as follows: a directly trained model contains redundant and irrelevant dimensions; for the large-training-then-

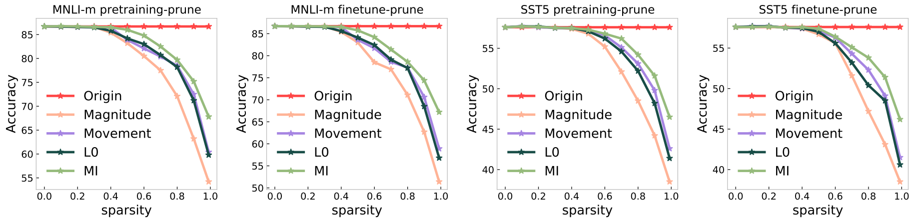

Figure 2: Performances of *pretrain-prune* and *finetune-prune* on MNLI-m and SST-5.

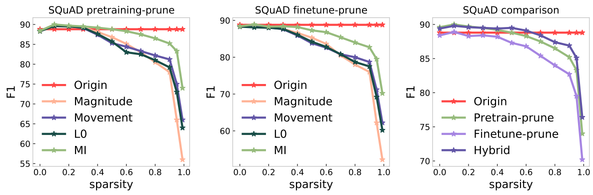

Figure 3: Performances of *pretrain-prune*, *finetune-prune* and *hybrid* on SQuAD.

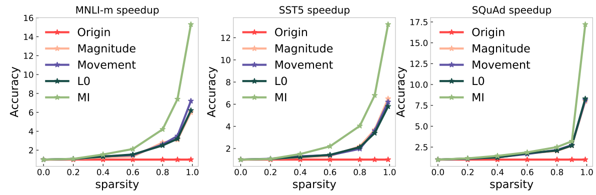

Figure 4: Speedups of different models for *pretrain-prune* on MNLI-m, SST-5 and SQuAD.

Model	**BLEU**	**FLOPs**	**Speedup**	**# Params**
	Original Models			
Large	28.4	100 %	× 1	100%
Base	27.3	17.5%	× 3.1	34%
Tiny	23.6	9.6%	× 5.1	13%
	Without Retraining: Pruning Large			
MI (to base)	27.9	19.2%	× 2.6	34%
MI (to tiny)	25.8	12.4%	× 4.9	13%
Magnitude (to base)	27.3	24.2%	× 1.8	34%
Magnitude (to tiny)	24.8	14.1%	× 2.8	13%
Movement (to base)	27.6	22.4%	× 1.7	34%
Movement (to tiny)	25.5	13.0%	× 3.5	13%
L0 (to base)	27.6	21.9%	× 1.5	34%
L0 (to tiny)	25.8	13.6%	× 2.7	13%
	Retraining Pruned Dimensions			
MI+Large	28.8 (+0.4)	17.2 %	× 3.2	34%
MI+Base	27.9 (+0.6)	9.8%	× 5.0	13%

Table 3: Test results for WMT14 En-De.

pruning strategy, the model first learns a larger set of feature dimensions, and then prunes irrelevant ones. This makes the model consist of fewer irrelevant feature dimensions than the one directly trained, leading to better performances.

(4) Pruning and then retraining yields consistent performance boosts over direct training: +0.6 for extra-large (43.3 vs 43.9), +0.5 for large (41.8 vs 42.3) and +0.5 for base (37.9 vs 38.4) for En-Fr. This is because direct training introduces redundant and less relevant features; retraining pruned dimensions can help the model replace less relevant dimensions with relevant ones, obtaining a state-of-the-art performance of 43.9 BLEU score for En→Fr translation in setups without back-translation or external data. Similar phenomenon are observed for En-De with +0.4 for the large model, and +0.6 for base models.

4.2 BERT Pruning

We carry out experiments on the pretrained model of BERT-large[3]. We select different degrees of sparsities from 0% to 90% at an interval of 10%. Model pruning can happen either in the pretraining stage (*pretrain-prune*), the fine-tune stage (*finetune-prune*), and both (*hybrid*): For *hybrid*, pruning happens at both stages, with the ultimate sparsity level γ being the product of the sparsity level of two stages, γ_{pretrain} and γ_{finetune}. We compare the performance of the three strategies on the SQuAD v1.1, MNLI and and SST-5 in Figure 2 and Figure 3. Generally, *pretrain-prune* works consistently better than *finetune-prune* with the same level of sparsity. This is because the training objective at the pretraining stage is a more general one than that at the finetuning stage, with more training data points and categories. Pruning at the finetuning stage is more prone to overfitting, leading to inferior performances. The *hybrid* method outperforms the *pretrain-prune* strategy if the sparsity levels at two stages are carefully calibrated. This is because the *hybrid* model can progressively prune less relevant dimensions in pretraining and then less relevant dimensions in task-specific finetuning, leading to better final performances.

For both *pretrain-prune* and *finetune-prune*, we find that the proposed MI method offers greater speedup and better performances at the same sparsity levels. Similar phenomenon are found for

[3]which contains 24 layers, 1,024 hidden units per layer, 16 heads per layer and 340M parameters in total

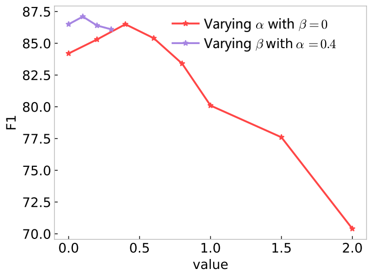

Figure 5: The effect of α and β.

# Iterations	1	2	3	4
F1	85.1	86.5	86.7	**86.8**

Table 4: The effect of iterative pruning.

MNLI and SST-5. Figure 4 shows the speedup gains for different models for the *pretrain-prune* setup. With the same sparsity, random pruning and the proposed MI based pruning lead to the largest speedup, followed by magnitude pruning, movement pruning and L_0 pruning. This observation validates that condensed weights serve as an effective remedy for irregular memory access.

5 Ablation Studies

In this section, we conduct ablation studies to get a better understanding of model behaviors. We use SQuAD for analysis, where BERT-large is used.

5.1 The Effect of α and β

The value of α and β in Eq.(9) controls the tradeoff between selecting relevant dimensions and removing redundant dimensions. Based on the *pretrain-prune* strategy with sparsity level of 20%, we can see from Figure 5 that the model works best when the value of α is set to 0.4, and then deteriorates as α increases when fixing $\beta = 0$. With fixed value of $\alpha = 0.4$, we find that the influence from β is less significant. This shows that given the conditional independency assumption, the improvement from the class-conditionally independent assumption is marginal. We thus suggest omitting this part if computing resources are limited.

5.2 The Effect of Iterative Pruning

Table 4 presents results with different number of pruning iterations, where we use linear interpolation to obtain sparsity levels for different iterations. As can be seen, though more pruning iterations lead to better performances, the boost becomes marginal when iteration number exceeds 2.

γ_{finetune}	1.0	0.8	0.6	0.4	0.2
F1	86.5	**87.4**	86.3	85.1	84.0

Table 5: The effect of γ_{pretrain} and γ_{finetune}.

Method	Inverted Pyramid	Vanilla	Pyramid
F1	**87.9**	87.4	87.2

Table 6: Layers with different sparsity values

5.3 The Effect of γ_{pretrain} and γ_{finetune}

Fixing the overall sparsity of 0.2, we explore the effect of γ_{pretrain} and γ_{finetune}. When $\gamma_{\text{finetune}} = 1$, it means we only perform pruning at the pretraining stage; When $\gamma_{\text{finetune}} = 0.2$, it means we only perform pruning at the finetuning stage. As can be seen from Table 5, performance peaks when γ_{finetune} is slightly lower than 1 ($\gamma_{\text{finetune}} = 0.8$, $\gamma_{\text{pretrain}} = 0.25$), and then declines as we increase γ_{finetune}. This further validates that the final performance benefits more when most pruning happens at the pretraining stage.

5.4 Layers with Different Sparsity Values

We explore the situation where given fixed overall sparsity value, different layers can have different levels of sparsity. We additionally consider two setups, *pyramid*, where lower layers are denser and thus less sparse than upper layers, and *inverted pyramid* where upper layers are less sparse than lower layers. For *pyramid*, with the overall sparsity of 0.2, the lowest word embedding starts with a sparsity level of 0.1, with the sparsity of all layers forms an arithmetic sequence. *inverted pyramid* has the same overall sparsity value of 0.2, with the lowest word embedding starts with a sparsity level of 0.3. Results are shown in Table 6. We can observe that *inverted pyramid* outperforms *vanilla*, which outperforms *pyramid*. These results illustrate that to obtain better performances in model pruning with fixed overall sparsity, upper layers should be less sparse than lower layers. This is because upper layers contain more high-level and dense information about the input. Therefore, pruning upper layers does more harm to the model. Lower layers contain more noise, and thus hurt the model less when get pruned.

6 Conclusion and Future Work

In this paper, we propose MI based methods for model pruning in NLP. The proposed model avoids the issue of irregular memory access, leading to higher speedup with the same level of sparsity. Also, the proposed strategy prunes the model in a top-down fashion based on global training signals, and thus achieves higher accuracies. In future work, we should release the strong assumption that neuron values come from a Gaussian distribution.

Acknowledgement

This work was supported by Key-Area Research and Development Program of Guangdong Province (No. 2019B121204008). We thank the High-Performance Computing Platform at Peking University and the PCNL Cloud Brain for providing platforms of data analysis and model training.

References

Jose M Alvarez and Mathieu Salzmann. 2016. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, volume 29, pages 2270–2278. Curran Associates, Inc.

Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. 2017. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual information neural estimation. In *International Conference on Machine Learning*, pages 531–540. PMLR.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Mohamed Bennasar, Yulia Hicks, and Rossitza Setchi. 2015. Feature selection using joint mutual information maximisation. *Expert Systems with Applications*, 42(22):8520–8532.

Mario Beraha, Alberto Maria Metelli, Matteo Papini, Andrea Tirinzoni, and Marcello Restelli. 2019. Feature selection via mutual information: New theoretical insights. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE.

Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. 2016. Feature selection for high-dimensional data. *Progress in Artificial Intelligence*, 5(2):65–75.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. 2012. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *The journal of machine learning research*, 13(1):27–66.

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

Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. 2018. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79.

Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. 2020. Description based text classification with reinforcement learning. In *International Conference on Machine Learning*, pages 1371–1382. PMLR.

Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. *arXiv preprint arXiv:2007.12223*.

Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. 2020. Club: A contrastive log-ratio upper bound of mutual information. In *International Conference on Machine Learning*, pages 1779–1788. PMLR.

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

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*.

Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan. 2019. Approximated oracle filter pruning for destructive CNN width optimization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1607–1616, Long Beach, California, USA. PMLR.

Xuanyi Dong and Yi Yang. 2019. Network pruning via transformable architecture search. In *Advances in Neural Information Processing Systems*, volume 32, pages 760–771. Curran Associates, Inc.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. 2009. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2):189–201.

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

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2020. Compressing large-scale transformer-based models: A case study on bert. *arXiv preprint arXiv:2002.11985*.

Aidan N. Gomez, Ivan Zhang, Siddhartha Rao Kamalakara, Divyam Madaan, Kevin Swersky, Yarin Gal, and Geoffrey E. Hinton. 2019. Learning sparse networks using targeted dropout.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. 2018. Morphnet: Fast & simple resource-constrained structure learning of deep networks.

Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155, Online. Association for Computational Linguistics.

Fu-Ming Guo, Sijia Liu, Finlay S Mungall, Xue Lin, and Yanzhi Wang. 2019. Reweighted proximal pruning for large-scale language representation. *arXiv preprint arXiv:1909.12486*.

Yiwen Guo, Anbang Yao, and Yurong Chen. 2016. Dynamic network surgery for efficient dnns. *arXiv preprint arXiv:1608.04493*.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

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*, 28:1135–1143.

Babak Hassibi and David Stork. 1993. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, pages 164–171. Morgan-Kaufmann.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations*.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800.

Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

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*.

Zehao Huang and Naiyan Wang. 2018. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320.

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*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest neighbor machine translation. *arXiv preprint arXiv:2010.00710*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Durk P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28:2575–2583.

Kenji Kira and Larry A Rendell. 1992. A practical approach to feature selection. In *Machine Learning Proceedings 1992*, pages 249–256. Elsevier.

Ludmila I Kuncheva. 2007. A stability index for feature selection. In *Artificial intelligence and applications*, pages 421–427.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yann LeCun, John Denker, and Sara Solla. 1990. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pages 598–605. Morgan-Kaufmann.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

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

Xiaoya Li, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. Sac: Accelerating and structuring self-attention via sparse adaptive connection. *arXiv preprint arXiv:2003.09833*.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. Bertgcn: Transductive text classification by combining gcn and bert. *arXiv preprint arXiv:2105.05727*.

Huawen Liu, Jigui Sun, Lei Liu, and Huijie Zhang. 2009. Feature selection with dynamic mutual information. *Pattern Recognition*, 42(7):1330–1339.

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

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019b. Rethinking the value of network pruning. In *International Conference on Learning Representations*.

Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

J Scott McCarley. 2019. Pruning a bert-based question answering model. *arXiv preprint arXiv:1910.06360*.

Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2021. Fast nearest neighbor machine translation. *arXiv preprint arXiv:2105.14528*.

Suyog Gupta Michael H. Zhu. 2018. To prune, or not to prune: Exploring the efficacy of pruning for model compression.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024. Curran Associates, Inc.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. 2017. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. 2017. Exploring sparsity in recurrent neural networks. *arXiv preprint arXiv:1704.05119*.

Andrew Y. Ng. 2004. Feature selection, <i>l</i>₁ vs. <i>l</i>₂ regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA. Association for Computing Machinery.

Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. 2019. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

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

Pradeep Ravikumar, Martin J. Wainwright, and John D. Lafferty. 2010. High-dimensional ising model selection using l 1 -regularized logistic regression. *Annals of Statistics*, 38(3):1287–1319.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning.

Abigail See, Minh-Thang Luong, and Christopher D Manning. 2016. Compression of neural machine translation models via pruning. *arXiv preprint arXiv:1606.09274*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and

Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Zijun Sun, Chun Fan, Xiaofei Sun, Yuxian Meng, Fei Wu, and Jiwei Li. 2020. Neural semi-supervised learning for text classification under large-scale pretraining.

Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. 2021. Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. *arXiv preprint arXiv:2106.16038*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565.

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

Jorge R Vergara and Pablo A Estévez. 2014. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24(1):175–186.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162, Online. Association for Computational Linguistics.

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*, 29:2074–2082.

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

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. *arXiv preprint arXiv:1703.02573*.

Jiahui Yu and Thomas Huang. 2019. Autoslim: Towards one-shot architecture search for channel numbers.

Lei Yu, Chris Ding, and Steven Loscalzo. 2008. Stable feature selection via dense feature groups. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 803–811.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021. Adaptive nearest neighbor machine translation. *arXiv preprint arXiv:2105.13022*.

Hao Zhou, Jose M Alvarez, and Fatih Porikli. 2016. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer.