DARB: A Density-Adaptive Regular-Block Pruning for Deep Neural Networks

Ao Ren,[†‡*] Tao Zhang,[†] Yuhao Wang,[†] Sheng Lin,[‡] Peiyan Dong,[‡]
Yen-kuang Chen,[†] Yuan Xie,[†] Yanzhi Wang[‡]
[†]Alibaba DAMO Academy
[‡]Northeastern University
{t.zhang, yuhao.w, yk.chen, y.xie}@alibaba-inc.com, {lin.sheng, dong.pe, ren.ao}@husky.neu.edu,
yanz.wang@northeastern.edu

Abstract

The rapidly growing parameter volume of deep neural networks (DNNs) hinders the artificial intelligence applications on resource constrained devices, such as mobile and wearable devices. Neural network pruning, as one of the mainstream model compression techniques, is under extensive study to reduce the model size and thus the amount of computation. And thereby, the state-of-the-art DNNs are able to be deployed on those devices with high runtime energy efficiency. In contrast to irregular pruning that incurs high index storage and decoding overhead, structured pruning techniques have been proposed as the promising solutions. However, prior studies on structured pruning tackle the problem mainly from the perspective of facilitating hardware implementation, without diving into the deep to analyze the characteristics of sparse neural networks. The neglect on the study of sparse neural networks causes inefficient trade-off between regularity and pruning ratio. Consequently, the potential of structurally pruning neural networks is not sufficiently mined.

In this work, we examine the structural characteristics of the irregularly pruned weight matrices, such as the diverse redundancy of different rows, the sensitivity of different rows to pruning, and the position characteristics of retained weights. By leveraging the gained insights as a guidance, we first propose the novel *block-max weight masking (BMWM)* method, which can effectively retain the salient weights while imposing high regularity to the weight matrix. As a further optimization, we propose a *density-adaptive regular-block (DARB)* pruning that can effectively take advantage of the intrinsic characteristics of neural networks, and thereby outperform prior structured pruning work with high pruning ratio and decoding efficiency. Our experimental results show that DARB can achieve 13× to 25× pruning ratio, which are 2.8× to 4.3× improvements than the state-of-the-art counterparts on multiple neural network models and tasks. Moreover, DARB can achieve 14.3× decoding efficiency than block pruning with higher pruning ratio.

1 Introduction

With the rapid development of deep neural networks (DNNs), artificial intelligence (AI) has penetrated into various application domains. The success of the state-of-the-art deep learning models highly relies on the large number of parameters and the resulting extensive computations. And the trend of new models is to grow even larger and deeper. However, the enormous number of parameters and computations prevents the models from being widely deployed in the scenarios that have limited hardware resources, such as running these models on mobile or edge devices. To address this issue, a lot of research efforts have been paid to reduce the model size and accelerate the inference process of DNNs (Han, Mao, and Dally 2016; Park, Ahn, and Yoo 2017; Zhou et al. 2017; Leng et al. 2017; Wang et al. 2018b; Ren et al. 2019; Ye et al. 2019).

Neural network pruning is one of the major compression techniques to reduce model size by removing the least important weights, i.e., the weights with small absolute value. It has been empirically proved in prior works (Han, Mao, and Dally 2016; Ren et al. 2019) that such irregular pruning can achieve the highest compression ratio than all other compression techniques for neural networks, such as structured pruning, feature/activation pruning, and weight quantization. Nonetheless, irregular pruning is mostly opted out in reality due to its notorious positional irregularity of retained weights, which incurs inefficient index decoding and high index storage overhead. Structured pruning techniques are developed to address the drawbacks of irregular pruning. They reduce indexing overhead and achieve acceleration mainly through the trade-off among regularity, pruning ratio, and model accuracy. That is, given the same accuracy, the looser constraints on the regularity of the weights, the higher pruning ratio can be achieved. However, little attention has been given to the intrinsic characteristics of neural networks for efficient trade-off.

In this work, we endeavor to find a more effective structured pruning method by studying the structural characteristics of the irregularly pruned weight matrix. In particular, we investigate the distribution of *row density* in the matrix and its correlation with the pruning ratio[1]. Our study focuses on recurrent neural networks (RNNs), such as long short term memory (LSTM) and gated recurrent unit (GRU),

[*]The work was done during Ao's internship at Alibaba DAMO Academy.

[1]Row density is defined as the percentage of retained weights in a row after pruning, which is the compliment of sparsity.

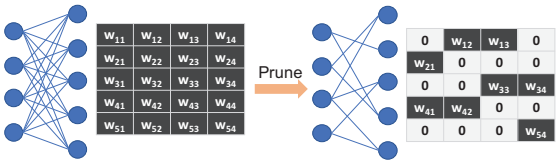

Figure 1: Neural network pruning.

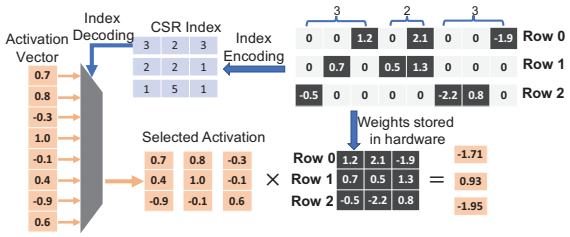

Figure 2: Index encoding and decoding process.

and the fully-connected layers of convolutional neural networks (CNNs). Our study reveals that: (i) maintaining the variety of row density helps sustain model accuracy; (ii) the dense rows are more sensitive than the sparse rows to further pruning; (iii) when dividing all rows into equally sized blocks and selecting in each block one weight with the largest magnitude, these locally salient weights have similar salience to the weights globally selected over the whole weight matrix. Based on the gained insights, we innovatively propose a *block-max weight masking (BMWM)* method and a *density-adaptive regular-block (DARB)* pruning method that can achieve high pruning ratio, low index storing cost, efficient index decoding, and sustained accuracy, *simultaneously*. Our experimental results show that the proposed DARB significantly outperforms the state-of-the-art counterparts by up to 4.3× higher pruning ratio on various neural network models and tasks. Moreover, DARB can beat block pruning by 14.3× higher index decoding efficiency and meanwhile achieve higher pruning ratio.

In summary, the main contributions of this work are:

- We analyze the irregularly pruned weight matrices and figure out that each row intrinsically has different density and denser rows are more sensitive to further pruning, which requires attention to be paid when developing pruning algorithms.
- We study the positional characteristic of weights, and propose the block-max weight masking (BMWM) method that is more effective in retaining the salient weights than prior structured pruning methods.
- We propose the density-adaptive regular-block (DARB) pruning to achieve high regularity, high pruning ratio, and sustained accuracy, simultaneously.

2 Background and Related Work

Neural networks are becoming larger and deeper to achieve the state-of-the-art performance in many domains, such as (Krizhevsky, Sutskever, and Hinton 2012a; Simonyan and Zisserman 2014; He et al. 2016; Szegedy et al. 2017) in computer vision (CV) and (Vaswani et al. 2017; Devlin et al. 2018; Yang et al. 2019; Simonyan and Zisserman 2014) in natural language processing (NLP). Nonetheless, the large number of parameters and the resulting computations impede the deployment of the models on devices that have limited on-chip resources. Consequently, neural network compression techniques have gained increasing attraction in both industry and academia. There are two major compression techniques, *pruning* and *quantization*, where the former is mainly intended to reduce the redundancy existing in the number of weights (Han, Mao, and Dally 2016; Frankle and Carbin 2019; Ren et al. 2019) while the latter is to reduce the redundancy in the representation precision of each single weight (Leng et al. 2017; Park, Ahn, and Yoo 2017; Zhou et al. 2017). These two techniques are orthogonal to each other and often combined to achieve the best compression ratio. This work only concentrates on the pruning technique, which solely can achieve higher compression ratio.

Irregular Pruning and Index Decoding As illustrated in Figure 1, pruning refers to removing the connections between the neurons of two adjacent layers, which results in a sparse weight matrix. Since the removed weights do not need to be stored and involved in computation, pruning is able to reduce storage and computation overhead, which is critical for resource constrained devices.

Irregular pruning is the most straightforward pruning technique. The idea behind is that the important weights have larger magnitude (i.e., absolute value), so keeping only the top-K weights in magnitude should have little impact on accuracy. Although it can achieve impressive high pruning ratio (Han, Mao, and Dally 2016; Frankle and Carbin 2019; Ren et al. 2019), the positions of the pruned weights are rather random, and consequently, a large number of indices are required to record the positions of the retained weights. To save the index storage, the relative compressed sparse row (CSR) format (Han, Mao, and Dally 2016) is usually adopted, which encodes each index by the relative distance (i.e., the number of zeros) between two adjacent non-zero weights. Besides, a decoding process is needed to select the corresponding activations for the retained weights. Figure 2 illustrates the encoding and decoding process. The main drawback of irregular pruning is that decoding one index requires a search over the whole activation vector, and thus it brings little acceleration and even speed degradation.

Structured Pruning and Related Work Structured pruning techniques are proposed to address the drawbacks of irregular pruning, and they can be categorized into two types.

The first type of structured pruning techniques requires no index or decoding, such as matrix factorization-based methods (Thakker et al. 2019; Sainath et al. 2013; Kim, Khan, and Kyung 2019), block-circulant algorithm (Wang et al. 2018a), and PermDNN (Deng et al. 2018). The low-rank matrix factorization proposed in (Sainath et al. 2013) factorizes the original high-rank weight matrix into low-rank sub-matrices and thereby achieves 30%-50% parameter reduction. (Thakker et al. 2019) proposes to use Kronecker Product to decompose the high-rank matrix. However, this

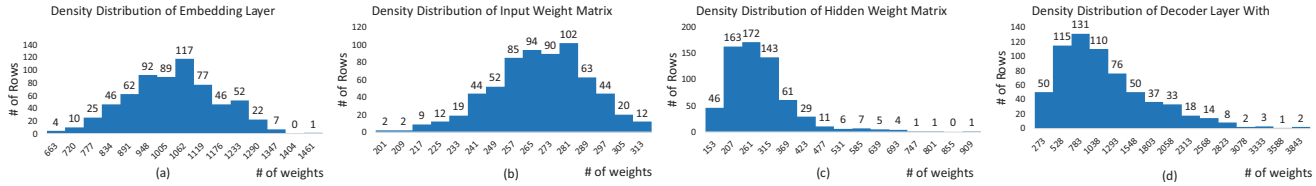

Figure 3: Row density distribution of the four weight components of a medium LSTM with 90% sparsity: (a) embedding layer, (b) input weight matrices, (c) hidden weight matrices, (d) decoder layer.

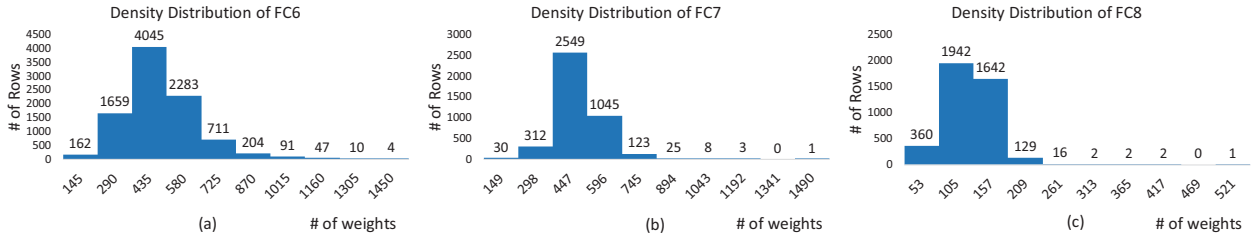

Figure 4: Row density distribution of FC6-FC8 of AlexNet with 90% sparsity: (a) FC6, (b) FC7, (c) FC8.

method achieves only little computation reduction, because the Kronecker Product computation to reconstruct a high-rank matrix from the decomposed sub-matrices is not trivial.

Block-circulant pruning proposed in (Wang et al. 2018a) divides the original weight matrix into sub-blocks and enforces each block to be a circulant matrix. Since this method requires storing only the first row of each circulant block, it can reduce the space complexity from $O(n^2)$ to $O(n)$ and the computation complexity from $O(n^2)$ to $O(n \log n)$. However, the computation is mainly performed in the frequency domain, which significantly diminishes the benefit of computation reduction because of the Fast Fourier Transformation and complex domain computation. PermDNN (Deng et al. 2018) decomposes the weight matrix into blocks and each block is a permuted diagonal matrix. Nevertheless, this method considers only the simplification of hardware implementation, resulting in limited pruning ratio.

The other type of structured pruning works in the way of sharing index among multiple neighboring weights, so that the index storage is reduced and the *decoding efficiency*[2] can be improved. Column pruning removes one column of the weight matrix (Wen et al. 2017). Although this method is effective in convolutional layers, it fails to work in the fully-connected layers, where removing one column can cause significant information loss as it is equivalent to removing one input activation. Prior work (Wen et al. 2017; Wang et al. 2019) adopts this strategy in RNNs but only achieves about 2× parameter reduction. Block pruning performs pruning at the scale of blocks (Van Keirsbilck, Keller, and Yang 2019), but grouping neighboring weights into a specific structure is a strong constraint which is not an effective way to keep the salient weights. As a result, only limited pruning ratio can be achieved.

[2]Decoding efficiency is defined as the number of activations selected per clock cycle for the corresponding retained weights.

3 Methodology

Based on the survey of the related work, we can conclude that (i) irregular pruning can achieve the highest compression ratio among all compression techniques mentioned above; (ii) structured pruning works in the way of trading off the regularity with pruning ratio at a given accuracy. In order to make the pruned weight matrix structural, two types of constraints are usually applied. One is to enforce all rows in the weight matrix to have the same number of weights retained, and the other is to group neighboring weights into certain structures, such as the block pruning (Van Keirsbilck, Keller, and Yang 2019) or block-circulant-based compression (Wang et al. 2018a).

Even though they can more or less achieve trade-off improvement, those previous studies have never discussed the fundamental question: *whether or not the the hardware-friendly algorithm can lead to high pruning ratio, which is another vital factor for achieving high performance and energy efficiency.* To answer this question, it is favorable to explore the intrinsic characteristics of the pruned neural networks to help us understand the essence of pruning. Since irregular pruning can achieve the highest compression ratio, in this work we study the structure characteristics of irregularly pruned weight matrices. In addition, we ultimately propose a structured pruning method that can achieve both high pruning ratio and decoding efficiency without accuracy loss.

Structure Characteristics of a Weight Matrix

In this work, we first obtain the pruning mask by applying the irregular pruning into the matrix. The pruning mask is a bitmap where '0' ('1') indicates if the weight needs to be pruned (retained). We then examine the row density distribution and the position distribution of the weights

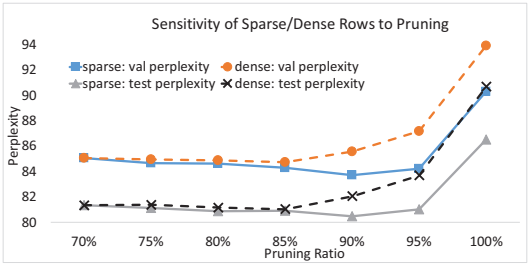

Figure 5: Sensitivity study of sparse and dense rows. The dense rows are more sensitive to further pruning as the solid curves are always above the dashed curves.

within each row. For the study, two models are selected and each is the representative of a specific domain. The first model is a medium size LSTM for language modeling (Zaremba, Sutskever, and Vinyals 2014), which has the architecture of (embedding: 10k×650)–(LSTM: 650)–(LSTM: 650)–(decoder: 650×10k). The second model is AlexNet for image classifications (Krizhevsky, Sutskever, and Hinton 2012b), and all of its fully-connected layers (i.e., FC6-FC8) are studied.

Figure 3 and 4 plots the row density distribution of the LSTM model and AlexNet (FC6-FC8 only), respectively. Both neural networks have been pruned irregularly to reach 90% sparsity. From the two figures, we can find that the row density distributions vary significantly not only across different neural networks, but also across different layers within the same network, and even across different rows within the same layer. For example, the densest rows can have up to 14× more weights than the sparsest rows.

Nowadays, enforcing all rows in a layer to have the same density is widely adopted to assure the regularity. Nevertheless, due to our observation on the wide variation in the row density distribution, we question this as a reasonable strategy. Therefore, we further investigate the sensitivity of a row for further pruning. We have done the experiments by firstly performing irregular pruning with an overall 70% pruning ratio to obtain the pruning mask. The density of each row is then calculated (i.e., counting 1s in the mask) and sorted. We evenly divide the sorted rows into two segments, the first half is denoted as *sparse rows* and the second half as *dense rows*. Then, we anchor the pruning ratio of the dense rows and increase the pruning ratio of the sparse rows. The model is then retrained to recover the perplexity[3]. As a comparison, we also anchor the pruning ratio of the sparse rows while increasing the ratio of the dense rows. Please refer to Section 4 for the detail of experiment setup.

Figure 5 illustrates the sensitivity of the sparse and dense rows to pruning ratio, respectively. The dashed curves show the results when sweeping pruning ratio in the dense rows. The solid curves show the results when sweeping pruning ratio in the sparse rows. The validation and test perplexities are plot separately. As shown, the dense rows are more sensitive to further pruning than the sparse rows, as the solid curves are always above the dashed curves. Generally, the sparse rows are resilient to further pruning. Thanks to the regularization effect, the model performance can be even improved with moderate pruning ratio (e.g., 70%-90%). However, as the pruning ratio goes high enough (e.g., ≥95%), many rows are pruned entirely, which leads to a sharp performance degradation.

[3]Perplexity is the metric to evaluate a language model and it is the lower the better.

Table 1: The portion of blocks that contain different number of weights. Block size is ten.

Layer	# of weights =0	# of weights =1	# of weights > 1
Embedding	34.86%	38.79%	26.35%
LSTM1	34.85%	38.75%	26.40%
LSTM2	34.94%	38.63%	26.43%
Decoder	34.95%	38.66%	26.39%

In addition to the distribution of row density, we also conduct the characterization study on weight positions. In the study, we first extract the pruning mask by performing irregular pruning with 90% pruning ratio on the medium size LSTM. Note that only the pruning mask is generated at this step, while the matrix is not really pruned. According to the pruning mask, we then measure the relative distance between two adjacent non-zero weights in the same row. We figure out that the average relative distance in all layers is ten, which corresponds to the sparsity (i.e., one out of ten weights is retained for 90% pruning ratio). As summarized in Table 1, when we divide each row into blocks whose sizes are ten, about 26% blocks across all layers have more than one non-zero weights, around 39% blocks have exactly one non-zero weight, and the remaining 35% blocks are empty.

To determine if those empty blocks contain only in-salient weights, we define a new metric *relative difference* to quantify the significance of weights in the empty blocks. Firstly, we pick the largest weight in each empty block and calculate the arithmetic mean of their absolute values as $\bar{W}_{empty}$. We then turn to the blocks that have more than one weights and pick all retained weights but the largest one (all-but-largest, or abl for short) to calculate their mean value as $\bar{W}_{abl}$. Finally, we compare the two mean values and denote the value $\frac{|\bar{W}_{abl}-\bar{W}_{empty}|}{\bar{W}_{abl}}$ as their relative difference. In this way, a large relative difference indicates that empty blocks only have in-salient weights while a small difference implies that empty blocks also have salient weights, even though they may be slightly less important than the retained weights selected by irregular pruning.

In our experiment, the relative difference is about 27.6% for both embedding and decoder layers, and 12.3% for the LSTM layers. As the relative difference is small, it reveals that these empty blocks also have salient weights. This finding inspires us to reform the weight selection strategy by devising the new *block-max weight masking (BMWM)* method. In contrast to the irregular pruning, BMWM simply picks the largest weight in magnitude from each block. Specifically, BMWM consists of three steps: (1) It first divides a row into equally sized blocks. For instance, given a row that has 1000 elements and the block size is 10, the row should

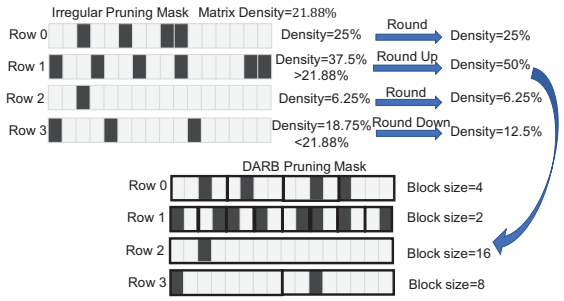

Figure 6: The generation of a DARB pruning mask. Irregular pruning mask is used as the starting point.

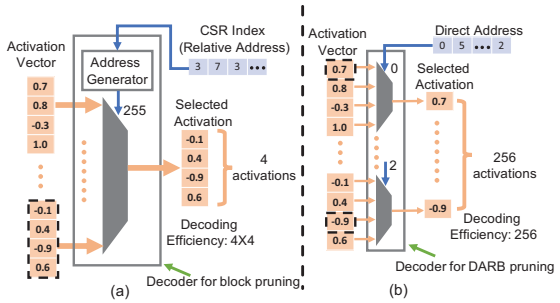

Figure 7: Decoding efficiency for (a) block pruning, (b) DARB pruning.

Table 2: The Comparison of Relative Difference of Retained Weights between BMWM and Block Pruning

Layer	BMWM to Irregular Pruning	Block Pruning 4×4 to Irregular Pruning
Embedding	8.8%	45.3%
LSTM1	4.3%	31.1%
LSTM2	4.3%	31.1%
Decoder	8.9%	45.2%

have 1000/10=100 blocks. (2) Within each block, it retains the largest weight in magnitude. (3) It repeats (1) and (2) for each row of the weight matrix.

To demonstrate the effectiveness of BMWM, we compare the salience of the retained weights after BMWM with that after irregular pruning. Specifically, the mean value of the magnitude of the retained weights after BMWM is calculated as $\bar{W}_{bmwm}$, and that after irregular pruning is $\bar{W}_{irr}$, their relative difference is calculated as $\frac{|\bar{W}_{irr}-\bar{W}_{bmwm}|}{\bar{W}_{irr}}$. As shown in Table 2, the relative difference for the embedding layer and decoder layer is reduced to 8.8%, and that for the LSTM layers drops to 4.3%. In other words, BMWM can effectively sustain the salience of the weight matrix, similar to irregular pruning. On the other hand, we also evaluate the salience of the retained weights selected by the state-of-the-art block pruning method with 4×4 block size. The relative difference between block pruning and irregular pruning is 45% in the embedding and decoders layer, and that in the LSTM layers is 31%, which is much larger than BMWM. Therefore, we are confident that BMWM is simple yet effective in selecting the salient weights.

Density-Adaptive Regular-Block Pruning

As discussed above, maintaining the variety of row density helps sustain accuracy. Therefore, it is desirable to consider this characteristic when designing a structured pruning method. However, maintaining this variety incurs difficulty in storing the retained weights in a compact and regular manner. In addition, even though BMWM is effective to select salient weights, the block size can be hardware unfriendly. To fully exploit the discovered characteristics of neural networks, we propose a density-adaptive regular-block (DARB) pruning method.

As illustrated in Figure 6, DARB first needs to obtain the pruning mask generated in irregular pruning. Note that this step is to calculate the row density, rather than perform real pruning. Like BMWM, DARB then divides each row into blocks and keeps only the weight that has the largest magnitude in each block. Distinct from BMWM, DARB allows different rows to adopt different block sizes so that the irregularity of row density can be maintained to some extent. To make it hardware friendly, DARB intentionally constrains the block size to be in power of two, such as 2, 4, ..., etc.

To obey the constraint on block size, the row density sometimes needs to be adjusted. Given that dense rows are more sensitive to further pruning, maintaining the density of dense rows helps sustain the model accuracy better. Also, we need to avoid the density from being rounded in the same direction that can change the pruning ratio significantly. Therefore, when the density of a row is greater (less) than the matrix density, it is rounded up (down). For instance, suppose the matrix density after irregular pruning is 21.88%, if the density of a row is 37.5%, it is rounded up to 50% so that the block size is two (=1/0.5). On the other hand, if the density of a row is 18.75%, it needs to be rounded down to 12.5% and the resulting block size is eight (=1/0.125).

In this way, DARB can achieve excellent trade-off between regularity and pruning ratio with the following benefits. Firstly, the block size is hardware friendly to facilitate index decoding. As illustrated in Figure 7(a), the conventional CSR encoded indices need a big multiplexer to accomplish the indexing, which makes the entire decoding process quite slow since it can only be done in sequence. Even though block pruning can share the index among multiple weights, it is still hard to significantly improve the decoding performance since the block size is substantially limited by the consideration of accuracy. In contrast, thanks to the small equally-sized blocks, DARB supports highly efficient decoding by leveraging multiple small multiplexers in parallel, as shown in Figure 7(b). Suppose the row size is 1,024, and the block size is four, DARB can simply select 256 activations in parallel with similar area cost.

Secondly, the constraint on the block size enforces the row density to be clustered into different groups, and the density ratio between two groups is still in the power of two. Figure 6 explains the motivation. In the figure, the density of row 0 and 1 is 25% and 50%, respectively. Therefore, the

Table 3: Pruning Results and Comparison on Large Size LSTM for PTB

Method	Perplexity	Para. No.	Pruning Ratio
baseline	(82.20, 78.40)	66.00M	1.00×
Irregular	(82.20, 78.40)	3.00M	20.00×
ISS	(82.59, 78.65)	21.80M	3.03×
BBS	(—, 79.20)	—	4.00×
DARB-1 (ours)	(82.15, 78.51)	5.02M	13.14×
DARB-2 (ours)	(82.65, 79.17)	4.26M	15.48×

density of row 1 is 2^1 times of row 0. Such regularity among row densities eventually facilitates the weights storing.

4 Evaluation

To evaluate the efficacy of DARB pruning, we test it on five neural networks. The pruning ratio and the accuracy of the pruned models are compared with the state-of-the-art work, including both irregular pruning and structured pruning methods. In addition, the efficiency of index decoding is evaluated by counting the number of activations that can be selected per clock cycle in an exemplary hardware implementation. In this work, we adopt the ADMM-NN pruning framework (Ren et al. 2019), which has achieved the state-of-the-art irregular pruning ratio. The ADMM-based pruning framework decomposes the pruning problem into two subproblems, one is to find a good pruning mask, and the other is to train the model via this mask. By iteratively solving these two subproblems, the target pruning ratio can be achieved.

Algorithm Evaluation

Language Modeling. For this task, a large size two-layer LSTM (Zaremba, Sutskever, and Vinyals 2014) is built to perform the word-level prediction for Penn Tree Bank (PTB) dataset (Marcus, Marcinkiewicz, and Santorini 1993), whose vocabulary size is 10k words. The training data, validation data, and test data of PTB dataset has 929k, 73k, and 82k words, respectively. The architecture of the model is (embedding: 10k×1500)–(LSTM: 1500)–(LSTM: 1500)–(decoder: 1500×10k). The model is trained with 20 batches and 35 unrolling steps, which has the same configurations as the prior arts (Zaremba, Sutskever, and Vinyals 2014; Wen et al. 2017). The dropout configuration for DARB is (0.35, 0.75) in the ADMM regularization step and (0.35, 0.7) in the retrain step, where the former in the parentheses is the dropout for LSTM layers, and the latter is for other layers. For the sake of fair comparison, two pruning ratios are applied separately to make sure that each achieves the similar perplexity to its counterpart.

Table 3 shows the comparison between DARB and the state-of-the-art pruning methods on the LSTM model. The first number in the perplexity tuple is the validation perplexity, and the second is the test perplexity. Compared with ISS (Wen et al. 2017) , which has similar perplexity with DARB-1, DARB-1 achieves 13.14× pruning ratio and outperforms ISS by 4.34×. In addition, DARB-2 can achieve 15.48× pruning ratio, which is 3.78× higher than BBS (Cao et al. 2019). Note that ISS can only prune the LSTM and decoder layers but leave the embedding layer untouched due to the high sensitivity to accuracy loss[4]. Instead, DARB can prune all four layers of the model by 13.14× and meanwhile sustain the model accuracy. Despite the 20× pruning ratio, the extremely low decoding efficiency makes irregular pruning inapplicable in real applications. We will discuss the decoding efficiency at the end of this section.

Table 4: Pruning Results ad Comparison on LSTMP for TIMIT

Method	PER Degradation	Para. No.	Pruning Ratio
baseline	20.70% → 20.70%	3.25M	1.0×
Irregular	20.70% → 20.90%	0.16M	20.0×
C-LSTM	24.15% → 24.57%	0.41M	8.0×
C-LSTM	24.15% → 25.48%	0.20M	16.0×
BBS	23.50% → 23.75%	0.41M	8.0×
DARB-a (ours)	20.70% → 20.80%	0.41M	8.0×
DARB-b (ours)	20.70% → 20.90%	0.20M	16.0×
DARB-c (ours)	20.70% → 21.00%	0.16M	20.0×

Table 5: Pruning Results and Comparison on GRU for TIMIT

Method	PER	Para. No.	Pruning Ratio
baseline	18.8%	12.7M	1.0×
Irregular	19.0%	0.40M	32.0×
Block-circulant-1	19.4%	1.59M	8.0×
Block-circulant-2	20.1%	0.79M	16.0×
DARB-A (ours)	18.8%	1.59M	8.0×
DARB-B (ours)	19.1%	0.40M	32.0×

Speech Recognition. This task is evaluated with TIMIT (Garofolo et al. 1990), which is an acoustic-phonetic speech corpus. It contains broadband recordings from 630 speakers, and each speaker reads ten phonetically rich sentences in eight major dialects of American English. We build two models for the task. The first model is a projected LSTM (LSTMP) (Zia and Zahid 2019), which has two LSTM layers with 1,024 hidden units and a projection layer with 512 hidden units. The second model is a two-layer gated recurrent unit (GRU) with 1,024 hidden units in each layer. For this task, the evaluation metric is phone error rate (PER), which is the smaller the better.

Table 4 shows the comparison results. Note that DARB has three variants with different pruning ratios. For this model, we prune only one layer of LSTMP, same as the previous work. In addition, since those prior studies have different baselines, we present PER degradation after pruning. When pruning ratio is 8×, the best prior work is BBS, which has 0.25% PER degradation. DARB-a instead has only 0.1% degradation. Comparing to C-LSTM (Wang et al. 2018a) that achieves 16× pruning ratio with 1.32% PER degradation, DARB-b can achieve the same pruning

[4]It is unclear if BBS can prune the entire network.

Table 6: Pruning Results and Comparison on FC6,FC7,FC8 of AlexNet for ImageNet

Method	Top-5 Acc.	Para. No.	Pruning Ratio
AlexNet	80.2%	58.6M	1.0×
PermDNN on AlexNet	80.0%	6.5M	9.0×
AlexNet-BN	83.4%	58.60M	1.0×
Irregular	83.4%	2.44M	24.0×
DARB-I on AlexNet-BN	83.4%	2.75M	21.3×
DARB-II on AlexNet-BN	83.2%	2.34M	25.0×

ratio with only 0.2% degradation. When the pruning ratio increases to 20×, DARB has only 0.3% PER degradation, which is 0.1% worse than irregular pruning. As a result, the performance of DARB is close to irregular pruning.

Table 5 shows the pruning results on the GRU model, and DARB is compared with the block-circulant algorithm. The Block-circulant algorithm sees 0.6% and 1.3% PER degradation with 8× and 16× pruning ratio, respectively. In contrast, DARB-A (8× pruning ratio) has no degradation and DARB-B (32× pruning ratio) has only 0.3% degradation. Furthermore, DARB-B even beats the block-circulant algorithm with 8× pruning ratio. Again, DARB achieves similar PER performance as irregular pruning.

Image Classification. To evaluate the feasibility of DARB on non-NLP models, we train an AlexNet (Krizhevsky, Sutskever, and Hinton 2012b) with ImageNet (Deng et al. 2009). Table 6 gives the comparison between DARB and PermDNN (Deng et al. 2018) when pruning FC6-FC8 of AlexNet. Note that the baseline of PermDNN is the vanilla AlexNet, while ours is AlexNet-BN that is trained with batch-normalization. Although the baselines are different, the original AlexNet-BN has higher accuracy, which is intuitively more difficult to prune. To assure fair comparison, we focus on the relative accuracy loss. As shown, DARB-I achieves 21.3× pruning ratio with no accuracy loss, which is remarkable because it is close to irregular pruning (24×), while PermDNN has 0.2% accuracy loss with 9× pruning ratio. If the same accuracy loss is allowed, DARB-II is able to achieve 25× pruning.

Also, we evaluate DARB in the convolutional layers of VGG-16 (Simonyan and Zisserman 2014) on CIFAR-10 (Krizhevsky, Hinton, and others 2009). It achieves 18.7× pruning ratio with only 0.6% accuracy degradation. The result suggests that DARB is promising for CNNs. We will optimize DARB for state-of-the-art CNNs as our future work.

Demonstration of Efficient Index Decoding

To demonstrate the index decoding efficiency of DARB, we define the decoding efficiency as the number of activations that can be selected from an activation vector per clock cycle. Block pruning is used as our baseline. The size of the activation vector in the aforementioned LSTM model is 1500. We implement six decoders in RTL to concurrently support the block size of 2, 4, ..., and 64. They are synthesized in CMOS $40nm$ process to measure the hardware area. We also design and synthesize the decoders for the baseline to support two types of block pruning. As shown in Table 7, given the same hardware area, we can equivalently deploy 6.41 and 6.45 decoders for the block-pruning design whose block size is 4×4 and 8×8, respectively. From the table, we can find that the decoding efficiency of DARB outperforms the other two designs by up to 14.3×. Moreover, once no perplexity degradation is allowed, DARB can achieve 1.76× and 2.45× higher pruning ratio than the other two designs. It is worth noting that the decoding efficiency of irregular pruning is only 7 with the same footprint. Such low decoding efficiency explains why it is rarely adopted in reality.

Table 7: Decoding Efficiency and Pruning Ratio Comparison

Metric	DARB	Block-4×4	Block-8×8
Area (μm^2)	58928	58928	58928
Decoder No.	6	6.41	6.45
Decoding Efficiency (activations/cycle)	1478	103	413
Pruning Ratio	13.14×	7.48×	5.37×

Regarding the index storage overhead, block pruning only has minor advantage over DARB. In general, block pruning has significantly lower pruning ratio than DARB, which indicates that its overall storage is still dominated by the weight itself. For instance, the original model has 66 million weights. After block pruning, 12.29 million weights (v.s. DARB's 5.02 million) are retained so that 0.2(=12.29/64) million indices are required. Assuming weights are quantized into INT8, it needs at least 12.29MB to store weights. Therefore, even if it only needs <0.1MB for relative CSR index, the overall storage saving is minor. On the other hand, DARB needs 5.02(=66/13.14) million indices. It seems huge at the first glance. However, the weight index in DARB just indicates its position within the block. As the block size is a power of two, a block whose size is m only requires $\log_2 m$ bits for indexing. Therefore, the average bit width of each index is less than four, and the total index storage overhead of DARB is less than 0.63MB, which is slightly larger than block pruning but still minimal.

5 Conclusion

In this work, we study the intrinsic characteristics of irregularly pruned weight matrices, including the row density distribution, the sensitivity of different rows to further pruning, and the positional characteristic of weights within each row. Motivated by our observations, we propose a block-max weight masking method that is superior to previous work in selecting salient weights. Then we devise a density-adaptive regular-block pruning method that can simultaneously achieve high pruning ratio and decoding efficiency, with almost no accuracy loss. Experimental results show that DARB outperforms the state-of-the-art prior in terms of pruning ratio by 2.8× to 4.3×. And it achieves up to 14.3× decoding efficiency over the block pruning.

On the other hand, inspired by prior work (Liu et al. 2018; Frankle and Carbin 2019), we believe that mining more

characteristics of sparse neural networks has great potential in not only pushing the state-of-the-art pruning ratio, but also facilitating neural architecture search. We will extend our research in this direction.

References

Cao, S.; Zhang, C.; Yao, Z.; Xiao, W.; Nie, L.; Zhan, D.; Liu, Y.; Wu, M.; and Zhang, L. 2019. Efficient and Effective Sparse LSTM on FPGA with Bank-Balanced Sparsity. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays (FPGA)*, 63–72. ACM.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 248–255. IEEE.

Deng, C.; Liao, S.; Xie, Y.; Parhi, K. K.; Qian, X.; and Yuan, B. 2018. PermDNN: Efficient Compressed DNN Architecture with Permuted Diagonal Matrices. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 189–202. IEEE.

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

Frankle, J., and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Garofolo, J. S.; Lamel, L. F.; Fisher, W. S.; Fiscus, J. G.; Pallett, D. S.; and Dahlgren, N. L. 1990. Timit acoustic-phonetic continuous speech corpus. *National Institute of Standards and Technology*.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *International Conference on Learning Representations (ICLR)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

Kim, H.; Khan, M. U. K.; and Kyung, C.-M. 2019. Efficient neural network compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 12569–12577.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012a. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012b. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of Advances in neural information processing systems (NeurIPS)*, 1097–1105.

Leng, C.; Li, H.; Zhu, S.; and Jin, R. 2017. Extremely low bit neural network: Squeeze the last bit out with admm. *arXiv preprint arXiv:1707.09870*.

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics* 19(2):313–330.

Park, E.; Ahn, J.; and Yoo, S. 2017. Weighted-entropy-based quantization for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7197–7205.

Ren, A.; Zhang, T.; Ye, S.; Li, J.; Xu, W.; Qian, X.; Lin, X.; and Wang, Y. 2019. Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 925–938. ACM.

Sainath, T. N.; Kingsbury, B.; Sindhwani, V.; Arisoy, E.; and Ramabhadran, B. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6655–6659. IEEE.

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

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Thakker, U.; Beu, J.; Gope, D.; Zhou, C.; Fedorov, I.; Dasika, G.; and Mattina, M. 2019. Compressing rnns for iot devices by 15-38x using kronecker products. *arXiv preprint arXiv:1906.02876*.

Van Keirsbilck, M.; Keller, A.; and Yang, X. 2019. Rethinking full connectivity in recurrent neural networks. *arXiv preprint arXiv:1905.12340*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, S.; Li, Z.; Ding, C.; Yuan, B.; Qiu, Q.; Wang, Y.; and Liang, Y. 2018a. C-LSTM: Enabling Efficient Lstm using Structured Compression Techniques on FPGAs. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 11–20. ACM.

Wang, Y.; Ding, C.; Li, Z.; Yuan, G.; Liao, S.; Ma, X.; Yuan, B.; Qian, X.; Tang, J.; Qiu, Q.; et al. 2018b. Towards ultra-high performance and energy efficiency of deep learning systems: an algorithm-hardware co-optimization framework. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wang, S.; Lin, P.; Hu, R.; Wang, H.; He, J.; Huang, Q.; and Chang, S. 2019. Acceleration of Lstm With Structured Pruning Method on FPGA. *IEEE Access* 7:62930–62937.

Wen, W.; He, Y.; Rajbhandari, S.; Zhang, M.; Wang, W.; Liu, F.; Hu, B.; Chen, Y.; and Li, H. 2017. Learning intrinsic sparse structures within long short-term memory. *arXiv preprint arXiv:1709.05027*.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Ye, S.; Xu, K.; Liu, S.; Cheng, H.; Lambrechts, J.-H.; Zhang, H.; Zhou, A.; Ma, K.; Wang, Y.; and Lin, X. 2019. Adversarial robustness vs. model compression, or both? ICCV.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent Neural Network Regularization. *arXiv preprint arXiv:1409.2329*.

Zhou, A.; Yao, A.; Guo, Y.; Xu, L.; and Chen, Y. 2017. Incremental network quantization: Towards lossless cnns with low-precision weights. In *International Conference on Learning Representations (ICLR)*.

Zia, T., and Zahid, U. 2019. Long short-term memory recurrent neural network architectures for Urdu acoustic modeling. *International Journal of Speech Technology* 22(1):21–30.