Deep Compressed Pneumonia Detection for Low-Power Embedded Devices

Hongjia Li$^1$, Sheng Lin$^1$, Ning Liu$^1$, Caiwen Ding$^2$, and Yanzhi Wang$^1$

$^1$ Northeastern University, Boston MA 02115, USA
{li.hongjia, lin.sheng, liu.ning}@husky.neu.edu, yanz.wang@northeastern.edu
$^2$ University of Connecticut, Storrs CT 06269, USA
caiwen.ding@uconn.edu

Abstract. Deep neural networks (DNNs) have been expanded into medical fields and triggered the revolution of some medical applications by extracting complex features and achieving high accuracy and performance, etc. On the contrast, the large-scale network brings high requirements of both memory storage and computation resource, especially for portable medical devices and other embedded systems. In this work, we first train a DNN for pneumonia detection using the dataset provided by RSNA Pneumonia Detection Challenge [4]. To overcome hardware limitation for implementing large-scale networks, we develop a systematic structured weight pruning method with filter sparsity, column sparsity and combined sparsity. Experiments show that we can achieve up to 36x compression ratio compared to the original model with 106 layers, while maintaining no accuracy degradation. We evaluate the proposed methods on an embedded low-power device, Jetson TX2, and achieve low power usage and high energy efficiency.

Keywords: Pneumonia detection · YOLO · structured weight pruning.

1 Introduction

There are approximately 450 million people globally (about 7% of the population in the world) suffering from pneumonia, and results in about 4 million deaths per year [14,9]. In the United States, pneumonia accounts for over 500,000 visits to emergency departments [3] and over 50,000 deaths in 2015 [1], keeping the ailment on the list of top 10 causes of death in the country. To accurately diagnose and localize pneumonia, a general diagnostic process requires review of a chest radiograph (CXR) by highly trained specialists and confirmation through clinical history, blood exams and vital symptoms.

To improve the efficiency and reach of diagnostic services, many researchers have extensively studied from medical fields and also computer aided design. In the past years, DNNs have been experiencing a rapid and tremendous progress thanks to the new era of big data. Especially for computer vision problems, deep learning and large-scale annotated image datasets drastically improved the performances of object recognition, detection and segmentation. Through the training processing based on large-scale datasets, DNNs can rapidly learn the complex
features and provide helpful functions of diagnose and localization. Many recent works have discussed medical image detection using large-scale neural networks. Based on Chest X-ray dataset [16], recurrent neural cascade model proposed by [15], CheXNet developed by [10], and Text-Image Embedding network (TieNet) introduced by [17]. Despite the promising results obtained by these works, one of the biggest challenges is that all these networks adopted a deep architecture with multiple layers, leading to a large memory storage and computation resource requirement. These make it difficult to implement large DNN models in portable medical devices and embedded systems [8,7].

In order to deploy DNNs on these embedded devices, DNN model compression techniques such as weight pruning, have been proposed for storage reduction and computation acceleration. Recently, works such as [5,20] have made breakthrough on the weight pruning methods for DNNs while maintaining the network accuracy. However, the network structure and weight storage after pruning become highly irregular and therefore the storage of indexing is non-negligible, which undermines the compression ratio and the performance. Therefore, the structured pruning is proposed to incorporate structured sparsity into the weight pruning algorithm [6,18]. The structured sparsity of DNN introduced by pruning methods is hardware-friendly, and it efficiently improves the evaluation of DNNs on embedded devices.

In this work, we develop a pneumonia detector based on you only look once (YOLO) [11]. We select a dataset provided by RSNA Pneumonia Detection Challenge [4]. In the pre-processing stage, the labeled images are resized to 320 × 320, along with the corresponding coordinates of bounding boxes, as shown in Figure 1. YOLOv3 [13] is adopted as the base feature detector with our costly anchor box priors, due to the speed boost and high average precision. It can achieve detection accuracy of 71.23 mAP. Moreover, in order to enhance the network performance, we utilize training optimizations including learning rate warm up, cosine learning rate decay and mixup training. To further maintain the precision obtained by the 106-layer network, we apply the ADMM-based unified model pruning algorithm on the original model, incorporated with structured sparsity (filter-wise sparsity and column-wise sparsity). Experimental result shows that without accuracy loss, our YOLOv3-based network can be pruned up to 36x. The number of parameters is reduced from 61.5M to 1.7 M, which undoubtedly reduces the memory storage and computation resource requirement for embedded systems. To validate our proposed method, we implement our model on Jetson.
and it achieves low power usage and high energy efficiency. Therefore, it verifies that our proposed method is very suitable for pneumonia detection with the characteristics of real-time and low-power on portable medical devices.

2 Model Design

2.1 YOLOv3

YOLO is an unified, real-time object detection framework. Compared with other object detection classifiers, YOLO frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities [11]. Recently, two improved versions of YOLO have been developed, namely YOLO9000 [12] and YOLOv3 [13]. In this work, we adopt YOLOv3 based detector due to its speed boost and high average precision.

YOLOv3 is a fully convolutional network, containing 75 convolutional layers, with skip connections and upsampling layers. The YOLOv3 adopts a convolutional layer with stride 2 as downsampling layer instead of pooling layer. A custom deep architecture Darknet-53 is utilized as the feature extractor since it can achieve a promising performance while with fewer floating point operations and more speedup [13]. In our work, we initialize the weights using a pretrained DarkNet-53 weights based on ImageNet.

YOLOv3 predicts boxes at 3 different scales. For each scale, detection layers that comprised of convolutional layers are constructed, respectively. The last layer predicts a 3D tensor containing bounding box coordinates, object prediction, and class predictions. In our work, the class number is 1 and the number of predicted boxes at each scale is 3, thus the tensor is $N \times N \times [3 \times (4 + 1 + 1)]$ for the 4 bounding box offsets, 1 object prediction, and 1 class predictions. YOLOv3 predicts bounding boxes using dimension clusters as anchor boxes. The network predicts 4 coordinates for each bounding box. K-means clustering is adopted to determine our anchor boxes. Same as YOLOv3, we choose 9 clusters and 3 scales. On our data, we modify the 9 clusters as following: $(40 \times 39), (63 \times 49), (48 \times 69), (75 \times 74), (58 \times 102), (83 \times 108), (67 \times 148), (89 \times 154), (94 \times 202)$.

2.2 Training optimization

Inspired by [19], we absorb several training optimization methods to enhance the network performance. **Learning rate warmup:** Instead of using a too large learning rate directly at the beginning, we use a small learning rate and then smooth back to the initial learning rate. To be specific, we use a gradual warmup strategy, which increases the learning rate from 0 to the original initial learning rate linearly. **Cosine learning rate decay:** For the learning rate decay, a cosine annealing strategy is applied, in which the learning rate gets decreased from the initial value to 0 by the following function: $l_{r_t} = 0.5 \times (1 + \cos(t \pi / T))l_{r_0}$, where $t$ denotes the current batch and $T$ denotes the total number of batches, and $l_{r_0}$ is the initial learning rate. **Mixup:** For data augmentation, we adopt
mixup method, in which each time we randomly sample two examples \((x_i, y_i)\) and \((x_j, y_j)\). Then a new example is obtained by a weighted linear interpolation of these two examples: 

\[
x' = \lambda x_i + (1 - \lambda) x_j, \quad y' = \lambda y_i + (1 - \lambda) y_j, \quad \text{where } \lambda \in [0, 1]
\]

is a random number drawn from the Beta(\(\alpha, \alpha\)) distribution. The new example \((x', y')\) will be used as our training data.

### 3 Model Compression

#### 3.1 Unified weight pruning algorithm

We develop an unified systematic framework containing three phases: pre-pruning, masked mapping and retraining. The objective of the weight pruning is to minimize the loss function while satisfying the weight constraints, the whole problem is defined as:

\[
\begin{align*}
\text{minimize } & f_{\text{loss}}(\{W_i\}_{i=1}^N, \{b_i\}_{i=1}^N), \quad \text{subject to} \quad W_i \in \mathcal{S}_i, \quad i = 1, \ldots, N. \\
\end{align*}
\]

where \(W_i\) and \(b_i\) denotes the sets of weights and biases of the \(i\)-th (CONV or FC) layer in an \(N\)-layer DNN, respectively. The set \(\mathcal{S}_i = \{W_i|\text{card}(W_i) \leq \alpha_i\}\) denotes the constraint for weight pruning, and ‘card’ refers to cardinality. It meets the goal that the number of non-zero elements in \(W_i\) is limited by \(\alpha_i\) in layer \(i\).

In the pre-pruning phase, we add the ADMM-based regularization on an original DNN model. The regularization is operated by introducing auxiliary variables \(Z_i\)'s, and dual variables \(U_i\)'s. In each iteration, while keeping on minimizing network regularized loss, we also reduce the error of Euclidean projection from \(W_i^{k+1} + U_i^{k}\) onto the set \(\mathcal{S}_i\). Because under the constraint that \(\alpha_i\) is the desired number of weights after pruning in the \(i\)-th layer, the Euclidean projection can keep \(\alpha_i\) elements in \(W_i^{k+1} + U_i^{k}\) with the largest magnitudes and set the remaining weights to zeros. Then the dual variables \(U_i\) is updated as following: 

\[
U_i^{k+1} = U_i^{k} + W_i^{k+1} - Z_i^{k+1}.
\]

In the second phase, with the obtained intermediate \(W_i\) solutions, we first perform the Euclidean projection (mapping) to satisfy that at most \(\alpha_i\) weights in each layer are non-zero. And then in the retraining phase, the zero weights are gradient masked and non-zero weights are retrained using training sets to restore partial accuracy.

#### 3.2 Structured pruning

As mentioned before, irregular pruning methods introduce extra storage for index and undermines the compression ratio and the performance. In order to develop an algorithm more friendly on hardware implementation, we incorporate structured pruning with the unified weight pruning algorithm.

In a typical convolutional layer, there are two structured sparsities: filter-wise sparsity, channel-wise sparsity, and shape-wise sparsity. For fully-connected layers, there are two types: row-wise sparsity and column-wise sparsity. We mainly focus on compressing convolutional layer in our design, since it is the most
During the convolutional computation, the feature map tensor and weights tensor are converted to 2D matrices and performed the general matrix multiplication (GEMM), as shown in Figure 2. Filter-wise sparsity corresponds to row-wise sparsity, while channel-wise sparsity and shape-wise sparsity correspond to column-wise sparsity. Therefore, filter pruning leads to reducing the number of rows of matrix, and correspondingly, channel and shape pruning result in the reduction of column number. The process of our structured pruning method can be explained as follows.

**Filter pruning** As we mentioned in Equation 1, the constraint set \( S_i \) here indicates the number of nonzero filters in \( W_i \) that is less than a predefined value \( \alpha_i \). To determine the limited number of nonzero filters, we perform \( l_2 \) norm on each filter and select the \( \alpha_i \) filters with most magnitude and set the remaining as zero.

**Column pruning** In the pre-pruning phase, we prune the convolutional weight by first converting 4D weight tensor into a 2D matrix. Therefore, the constraint set \( S_i \) for column pruning indicates the number of nonzero column in converted \( W_i \) that is less than a threshold value. The largest \( \alpha_i \) columns evaluated by \( l_2 \) norm are kept and the remaining column values are set to zero.

**Combined pruning** To take advantage of utilization in structured pruning on hardware implementation, we propose a approach by combination of these two structured pruning, which decreases the dimension in GEMM while still maintaining a full matrix. We first perform either one type pruning, filter for example. With the filter-pruned model, we first mask the zero filters and then perform the column pruning. In this way, we can keep the desired number of nonzero filter and obtain a higher sparsity on the column-wise.
4 Experimental Results

In this section, we evaluate the proposed model compression technique, starting from original model training, systematic structured weight pruning, and the hardware implementation on embedded device.

4.1 Data preprocessing

In our project, we use the dataset provided by RSNA Pneumonia Detection Challenge [4]. The dataset is derived from National Institutes of Health Clinical Center for publicly providing the Chest X-Ray dataset [19]. In our experiments, only the labeled images are selected, loaded from Digital Imaging and Communications in Medicine (DICOM) image format and resized into $320 \times 320$ from the original $1024 \times 1024$. The corresponding coordinates are also re-calculated from the original size. The whole dataset contains 6,002 images, of which 5,400 are considered as our training dataset and the remaining 602 are test dataset.

4.2 Model training

We apply the weight pruning method and train the pneumonia detector on Nvidia GeForce GTX2080 using Pytorch. During the training, we warmup our learning rate from $10^{-5}$ to our initial learning rate $10^{-3}$ during the first epoch. In the rest epochs, the learning rate decreased from $10^{-3}$ to $4^{-8}$ using the cosine function. The $\alpha$ for the Beta distribution in data mixup is 0.2.

4.3 Model evaluation

To evaluate the performance of the model, we use mean average precision (mAP) at different intersection over union (IoU) thresholds. The metric sweeps over a range of IoU thresholds, at each point calculating an average precision value. The threshold values range from 0.4 to 0.75 with a step size of 0.05: (0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75). To be specific, if we use 0.5 as the threshold, only when IoU is greater than 0.5 the object can be considered as detected. The result of original model is listed on the first row in Table 1 under different IoU thresholds. When IoU threshold is 0.5, we can achieve detection accuracy of 71.23 mAP.

Next, the unified structured weight pruning method is applied on filter pruning, column pruning and combined pruning, respectively. The detailed evaluation results of models with various prune ratio are shown in Table 1. Without accuracy loss, the prune ratio can be increased up to 36x. For this model, we prune 3.56x filters and 9.68x columns. The size of model parameters is reduced from 61.5 M to 1.7 M, which results the model storage saved from 246.4 MB to 6.84 MB. The original floating point operations (FLOPs) is 38.63 Bn. In total, the FLOPs can be significantly reduced to 1.32 Bn. In this way, not only the requirement of memory storage and computation resource decreased, but also facilitate acceleration on embedded devices.
Table 1: Localization accuracy (mAP) using IoU where T(IoU)=0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75.

| T(IoU) | 0.4 | 0.45 | 0.5 | 0.55 | 0.6 | 0.65 | 0.7 | 0.75 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| Original model | 81.2 | 76.3 | **71.2** | 63.4 | 54.7 | 42.3 | 30.7 | 19.1 |
| Filter pruned | 11.55x | 81.4 | 75.9 | **71.5** | 63.8 | 53.0 | 40.9 | 29.7 | 17.9 |
| | 16.26x | 80.6 | 76.2 | **71.2** | 62.5 | 53.6 | 42.3 | 30.0 | 18.7 |
| | 19.33x | 80.7 | 76.1 | **71.1** | 62.3 | 52.9 | 41.2 | 30.0 | 18.6 |
| Column pruned | 11.66x | 81.2 | 76.9 | **71.9** | 64.5 | 53.3 | 41.8 | 29.8 | 18.4 |
| | 16.36x | 80.7 | 76.0 | **71.3** | 64.1 | 55.9 | 41.5 | 28.4 | 19.1 |
| | 19.55x | 80.6 | 76.1 | **71.0** | 63.7 | 53.5 | 42.3 | 29.7 | 18.7 |
| Combined pruned | 36.02x | **81.2** | 76.3 | 71.0 | **63.5** | 53.4 | **42.8** | **31.2** | 19.3 |
| | 51.97x | 81.0 | 76.0 | **70.6** | 62.8 | 53.0 | 41.7 | 29.0 | 18.3 |

### 4.4 Hardware implementation

To validate our method on the embedded low-power devices, we implement our pruned model on Jetson TX2, which is considered as the fastest, most power-efficient embedded AI computing device [2]. It’s built by a 256-core NVIDIA Pascal-family GPU and the memory is 8 GB with 59.7 GB/s bandwidth. The power consumption of our model is 7.3 W and the energy efficiency is 0.69 IPS/W. The low power usage and high energy efficiency show a high feasibility and compatibility of our weight pruning method on DNN for low-power real-world devices.

### 5 Conclusion

In this work, we developed a YOLOv3-based detector for pneumonia detection with 71.23 mAP. In order to reduce the storage memory and computational resource requirement by the 106-layer fully convolution network, we applied a systematic structured weight pruning method on filter sparsity, column sparsity and combined sparsity. Without accuracy loss, the prune ratio can achieve up to 36x, which reduce the model size from 61.5 M to 1.7 M. To validate our method on the real-world low-power device, we implemented and evaluated our model on Jetson TX2, which resulted a low power usage and high energy efficiency.

### References

1. Deaths: Final data for 2015. supplemental tables. [https://www.cdc.gov/nchs/nvsr/nvsr66/nvsr66_06_tables.pdf](https://www.cdc.gov/nchs/nvsr/nvsr66/nvsr66_06_tables.pdf) [Online; accessed 24-May-2019]
2. Jetson tx2 module. [https://developer.nvidia.com/embedded/buy/jetson-tx2](https://developer.nvidia.com/embedded/buy/jetson-tx2)
3. National ambulatory medical care survey: 2015 emergency department summary tables. [https://www.cdc.gov/nchs/nhamcs/web_tables/2015_ed_web_tables.pdf](https://www.cdc.gov/nchs/nhamcs/web_tables/2015_ed_web_tables.pdf) [Online; accessed 24-May-2019]
4. Rsna pneumonia detection challenge (2018). [https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/overview](https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/overview)
5. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
6. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: International Conference on Computer Vision (ICCV). vol. 2 (2017)
7. Li, H., Liu, N., Ma, X., Lin, S., Ye, S., Zhang, T., Lin, X., Xu, W., Wang, Y.: Admm-based weight pruning for real-time deep learning acceleration on mobile devices. In: Proceedings of the 2019 on Great Lakes Symposium on VLSI. pp. 501–506. ACM (2019)
8. Lin, S., Liu, N., Nazemi, M., Li, H., Ding, C., Wang, Y., Pedram, M.: Fft-based deep learning deployment in embedded systems. In: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 1045–1050. IEEE (2018)
9. Lodha, R., Kabra, S.K., Pandey, R.M.: Antibiotics for community-acquired pneumonia in children. Cochrane Database of Systematic Reviews (6) (2013)
10. Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langelotz, C., Shpanskiaya, K., et al.; CheXnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225 (2017)
11. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
12. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7263–7271 (2017)
13. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
14. Ruuskanen, O., Laiho, E., Jennings, L.C., Murdoch, D.R.: Viral pneumonia. The Lancet 377(9773), 1264–1275 (2011)
15. Shin, H.C., Roberts, K., Lu, L., Demner-Fushman, D., Yao, J., Summers, R.M.: Learning to read chest x-rays: Recurrent neural cascade model for automated image annotation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2497–2506 (2016)
16. Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., Summers, R.M.: Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2097–2106 (2017)
17. Wang, X., Peng, Y., Lu, L., Lu, Z., Summers, R.M.: Tienet: Text-image embedding network for common thorax disease classification and reporting in chest x-rays. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 9049–9058 (2018)
18. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: Advances in Neural Information Processing Systems. pp. 2074–2082 (2016)
19. Xie, J., He, T., Zhang, Z., Zhang, H., Zhang, Z., Li, M.: Bag of tricks for image classification with convolutional neural networks. arXiv preprint arXiv:1812.01187 (2018)
20. Zhang, T., Ye, S., Zhang, K., Tang, J., Wen, W., Fardad, M., Wang, Y.: A systematic dnn weight pruning framework using alternating direction method of multipliers. arXiv preprint arXiv:1804.03294 (2018)