Neuron-based Network Pruning Based on Majority Voting

Ali Alqahtani *†, Xianghua Xie*, Ehab Essa*‡, and Mark W Jones*
* Department of Computer Science, Swansea University, Swansea, UK
† Department of Computer Science, King Khalid University, Abha, Saudi Arabia
‡ Department of Computer Science, Mansoura University, Mansoura, Egypt

***Abstract*—The achievement of neural networks in a variety of applications is accompanied by a dramatic increase in computational costs and memory requirements. In this paper, we propose an efficient method to simultaneously identify the critical neurons and prune the model during training without involving any pre-training or fine-tuning procedures. Unlike existing methods, which accomplish this task in a greedy fashion, we propose a majority voting technique to compare the activation values among neurons and assign a voting score to quantitatively evaluate their importance. This mechanism helps to effectively reduce model complexity by eliminating the less influential neurons and aims to determine a subset of the whole model that can represent the reference model with much fewer parameters within the training process. Experimental results show that majority voting efficiently compresses the network with no drop in model accuracy, pruning more than 79% of the original model parameters on CIFAR10 and more than 91% of the original parameters on MNIST. Moreover, we show that with our proposed method, sparse models can be further pruned into even smaller models by removing more than 60% of the parameters, whilst preserving the reference model accuracy.**

I. Introduction

In recent years, deep learning algorithms have shown their robust ability in representation learning and achieved considerable success in many tasks. Deep learning utilizes a hierarchical level of neural networks with respect to different kinds of neural networks, such as multi-layer perceptron (MLP), convolutional neural networks (CNNs), and recurrent neural networks (RNNs), to perform the learning process. This hierarchical representation allows models to learn features at multiple levels of abstraction, meaning that complicated concepts can be learned from simpler ones. Neurons in earlier layers of a network learn low-level features, while neurons of later layers learn more complex concepts.

Thanks to large amount of data and advanced computing power, deep learning models have advanced into wider and deeper architectures, driving the state-of-the-art performances in various tasks. However, its significant redundancy in the parameterization has become a widely-recognized property of deep learning [1]. The over-parametrized and redundant nature of deep neural networks present significant challenges for many applications. For instance, deploying sizeable deep learning models to a resource-limited device leads to various constrains, as on-device memory is limited [2]. Moreover, training with more parameters than necessary incurs expensive computational costs and high storage requirements.

In an attempt to confront these challenges, several approaches have been developed to visually understand the importance of intermediate neurons in neural networks [3], [4], [5], and measure the influence of hidden units [6], [7], [8]. Although these approaches provide different ways to measure the importance of individual hidden units, most of the focus on gaining a better understanding of the network's behavior, with limited attention being paid to pruning studies. The ways in which neuron-based pruning assist in decreasing the complexity of large scale networks are not the focus of the current research. Influential neurons usually identify essential features or high-level concepts on a trained network. Recognizing the importance of such neurons can help to reduce the model complexity by discarding less important units. Reducing the complexity of models while maintaining their powerful performance is always desirable.

Pruning approaches can be applied to any part of deep neural networks, including weights [9], [10], [11], neurons [12], [13], filters [14], and channels [15]. Most of the existing methods tend to focus on compressing networks rather than on discovering informative neurons for effective pruning. The fact that not all nodes deliver essential information for the final prediction of the model motivates us to fundamentally rely on applying the importance method when pruning non-informative neurons. Most of the existing methods also tend to compress the networks through the following three-step procedure: training, pruning, and fine-tuning; in contrast, we train our models from scratch without the use of any pre-training or fine-tuning. We integrate the pruning procedure into the learning process, aiming to find a smaller, well-suited architecture to the target task at the training phase. The main goal of pruning algorithms is to obtain a subnetwork with much fewer parameters without harming accuracy. The pruned version, a subset of the whole model, can represent the reference model at a smaller size or much fewer parameters. Hence, overparameterized networks can be efficiently compressed while maintaining the property of better generalization [16].

In this paper, the focus is on a neuron-pruning approach that is carried out according to level of importance. We propose a majority voting technique which votes for crucial neurons and removes redundant nodes accordingly. Our activation-based method aims to compute a measure of relevance that identifies the most critical neurons by assigning a voting score

to evaluate their importance. In order to gather conclusive evidence to evaluate the effectiveness of our method, an experiment based on ablation analysis in trained models was carried out. By comparing our importance method with several baselines, we show that our method substantially outperforms others in terms of neurons' effective measurement. We also introduce a network-wide holistic approach to prune neurons based on our majority voting method during training, without involving any pre-training or fine-tuning procedures. We have evaluated our pruning model on MNIST and CIFAR10 datasets. The experimental results show that the proposed method efficiently reduces the number of parameters without harming the accuracy.

II. Related Work

A. Neuron Importance Methods

Some neuron-based strategies corresponding to different measures of importance were explored. Their core aim is to evaluate neuron importance, which provides meaningful insight into the characteristics of the internal representations of neural networks. The developed methods have attempted to visually understand and characterize the deep representations, mainly focusing on single-neuron properties [4] and pixel-level annotations [3], [5] in relation to object invariance. Analyzing individual neurons and looking for an explanation of specific activation does not help to intuitively quantify the sufficient associations and decision linkages between neurons with a massive number of parameters.

The quantitative assessment of neurons' property has been successfully adopted to understand neuron property and evaluate neurons' importance. Such techniques introduce objectives to measure the activation values of each neuron and assign to them a score. Dhamdhere et al. [6] utilize integrated gradients by summing the gradients of the output prediction with respect to the input, in order to evaluate the importance of hidden neurons. Amjad et al. [7] also proposed a method to compute internal neuron importance, utilizing information-theoretic quantities (i.e. entropy and mutual information) to understand the outputs of individual neurons of trained neural networks. Moreover, Morcos et al. [17] investigated the relationship between the classification performance of neural networks and the output of individual neurons to estimate class selectivity and mutual information for the activation of each neuron. Furthermore, Na et al. [8] have recently used the highest mean activation to measure the importance of individual units on language tasks, showing that different units are selectively responsive to specific morphemes, words and phrases. Although these methods provide an intuitive process to determine criteria for neuron selection for effective pruning, most of the previously mentioned methods focus on gaining a better understanding of the network's behaviour, with limited attention being paid towards pruning methods.

B. Pruning Methods

Pruning approaches have received considerable attention as a way to tackle over-parameterization and redundancy. Existing works for the purpose of network compression can be classified into two categories: weight-based methods [9], [10], [11], [14], [18] and neuron-based methods [12], [19], [13]. Weight-based pruning eliminates unnecessary, low-weight connections between layers of a neural network, while neuron-based methods remove all weight connections to a specific neuron, where both income or outgoing weights are removed.

Several weight-based methods have been proposed to prune non-informative connections. Recently, Han et al. [11] introduced a pruning method to remove connections whose absolute values are smaller than a predefined threshold value. The threshold calculated using the standard deviation of a layer's weights. The network is, thereafter, retrained to recover the dropped accuracy. Although Han's framework received significant attention and has become a typical method used for network pruning, it focuses on weights' magnitudes, relies on iterative pruning and fine-tuning, and requires a particular software/hardware accelerator which is not supported by off-the-shelf libraries. Moreover, the reliance on a predefined threshold is less practical and proves too inflexible for some applications.

Li et al. [14] also proposed a pruning method based on the absolute weighted sum to judge the importance of the intermediate weights, where pruning is carried out according to the lowest scores. The method requires multiple iterations to regain lost accuracy, which can be time-consuming.

Liu et al. [20] showed the possibility of overriding the retraining phase by random reinitialization before the retraining step. This still delivers equal accuracy with comparable training time. Furthermore, Mocanu et al. [18] replaced the fully-connected layers with sparsely-connected layers by applying initial topology based on the Erdős–Rényi random graph. During training, fractions of the smallest weights are iteratively removed and replaced with the new random weights. Applying initial topology allows for the finding of a sparse architecture before training; however, this requires expansive training steps and obviously benefits from iteratively random initialization.

It can be argued that the use of weight-based methods alone suffers from certain limitations. The need to remove low-weight connections means that important neurons whose activation does not contribute enough, due to low-magnitude income or outgoing connections, could be ignored. However, these methods can be applied in combination with our proposed one.This combination adds extra compression value. Moreover, the overall impact of weight-based pruning on network compression can be less compared to neuron-based ones. Pruning a neuron eliminates entire rows or columns of the weight matrices from both the former and later layers connected to that neuron, while weight-based methods only prune the low-weight connections between layers.

Neuron-based methods represent another pruning approach that is proposed to eliminate less important neurons. He et al. [12] propose a simple neuron-based pruning strategy. They evaluate the importance of a neuron by summing the output weights of each neuron; based on this, the unimportant

nodes are eliminated. They also apply neuron-based pruning utilizing neuron activation entropy. Their entropy function evaluates the activation distribution of each neuron based on a predefined threshold, which is only suitable with a sigmoid activation function. Since this method damages the network's accuracy, additional fine-tuning is required to obtain satisfactory performance. Srinivas et al. [19] also introduced a neuron-based pruning method by evaluating the weights similarity of neurons in a layer. A neuron is removed when its weights are similar to that of another in its layer. Moreover, Mariet et al. [13] introduced Divnet, which selects a subset of diverse neurons and subsequently merges similar neurons into one. The subset is selected based on activation patterns by defining a probability measure over subsets of neurons. These non-structured pruning methods require a particular software/hardware accelerator that is not supported by off-the-shelf libraries. They also require a multi-step procedure to prune neurons; in contrast, our method prunes the model during training, leading to better solutions.

Most of the existing methods tend to compress the networks through multi-step procedures. Undertaking enough retraining is the only technique which can be used to regain the initial accuracy, as there is no guarantee that accuracy will be preserved throughout the compression phase. One definite explanation is that the standard compression approach mainly benefits from the substantial retraining step, especially when the selection criteria is simple and does not adequately measure the importance, due to the adoption of less efficient measurement standards. In order to overcome these issues, our proposed method involves a mechanism which introduces competent neuron measurement into the pruning process. This mechanism helps to reconcile the significant importance measurement and effective pruning. We train our models from scratch without involving any pre-training or iterative fine-tuning procedures. This saves time that is needed for the initial training as well as the retraining phases; these could require twice the amount of time that is usually necessary to train a model from scratch.

III. Proposed Method

Pruning redundant neurons always requires a more careful approach. There is no standard guidance for choosing the best network architecture; a model may have a certain level of redundancy to guarantee excellent quality performance. Most of the existing methods tend to compress networks by adopting straightforward selection criteria, such as relying on a predefined threshold [11], calculating absolute summed values [14], or simply summing the output weights[12]. These methods seem to focus on compressing networks without discovering a well-suited architecture or adopting efficient measurement standards to evaluate importance. In contrast, our neuron importance measurement method, majority voting, selects a subset of neurons based on voting scores. It measures the importance of neurons in each layer and determines a subset of neurons whose activation patterns are the most influential. In this paper, we introduce a comprehensive approach to prune network's neurons based on our majority voting method during training, without involving any pre-training or fine-tuning procedures. The proposed method introduces a mechanism for measuring the importance of neurons and pruning them accordingly into the body of the learning phase, aiming to obtain a subset of the whole model, which represents the reference model with much fewer parameters. In this section, we introduce our overall proposed framework, which consists of two parts. First, the measurement of neuron importance is discussed; this includes utilizing the majority voting approach in order to determine the importance of neurons in each layer. Then, we introduce a network-wide holistic approach which can be used to prune network neurons during training. The details are provided below.

A. Importance of Individual Neuron via Majority voting (MV)

We aim to detect influential neurons in neural networks by evaluating their activation. Feeding the training data through the network, each example is represented differently and has individual activation throughout all neurons in the network. We apply forward passing through an optimized model to find the output of each neuron, called activation. This can be viewed as random variables, and different input images can sample more instances. To evaluate their importance, a voting technique is applied; it only votes for a neuron when all the instances agree, which is what majority voting refers to. Our method was named majority voting (MV) as it utilizes a majority voting strategy to measure the importance of neurons. This measure of importance is discussed in depth below. Each layer has weights that are multiplied with an input example, x, to produce an output corresponding n activation. The activation at j-th neuron is computed as the weighted sum of activations from all neurons in the $i-1$-th layer. The output of the j-th unit in the i-th layer of the neural network is defined as:

$$t_j^{(i)}(x_n) = \sigma\left(b_j^{(i)} + \sum_p w_{p,j}^{(i-1)} t_p^{(i-1)}(x_n)\right), \qquad (1)$$

where x_n denotes the n-th data example at the input, σ is the activation function, b_j^i denotes the corresponding bias for the j-th unit in the i-th layer, $w_{p,j}^{(i-1)}$ is the weight that connects p-th neuron from the previous layer $(i-1)$ with the j-th unit in the i-th layer (existing layer).

After this, the activation matrix is obtained by Eq.(1). For each row, we set the top l largest activation neurons to 1 and others to 0 by using the following form:

$$v_j^{(i)}(x_n) = \begin{cases} 1 & \text{If argsort } (t_j^{(i)}(x_n))[1:l] \\ 0 & \text{Otherwise} \end{cases} \qquad (2)$$

As a result, a binary matrix is obtained with $J * N$ dimension in the i-th layer, where J is the number of neurons and N the number of input examples. The obtained matrix determines how important a neuron is for a given example, where 1 indicates the most influential neuron and 0 otherwise. Then, we sum over columns (examples) to score of the number of times that the j-th neuron is one of the top neurons for given

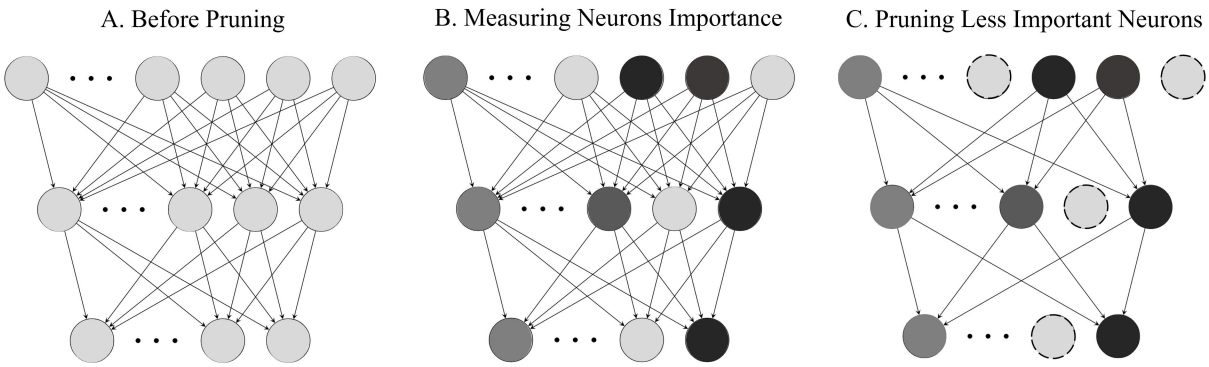

Fig. 1: Neuron-based pruning method. (A) The initial state of the fully-connected layers. After training the network for t epochs, (B) Measuring neuron importance, where the dark circles in the diagram indicate important neurons. (C) Pruning the less important neurons, based on which the income or outgoing connections are removed.

examples, voting for the crucial neurons. This is given by the following form:

$$y_j^{(i)} = \sum_{n=1}^{N} v_j^{(i)}(x_n) \tag{3}$$

$$\psi_j^{(i)} = y_j^{(i)} = \begin{cases} 1 & \text{If argsort } (y_j^{(i)})[1 : k * J] \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

We set a k percentage of the J neurons, which have the largest voting scores, to 1 and the remaining to 0. Here, k denotes the percentage of the largest index of y. For every layer, we will come up with a binary vector that indicates whether such neurons are important or not, where 1 denotes that the neuron is important and 0 otherwise.

Algorithm 1 Pruning algorithm using Majority Voting (MV)

Input: Training set (x, y), Validation set ($\hat{x}$, $\hat{y}$), t, and k
Output: A pruned model
initialization
best accuracy $\leftarrow 0$
for $e \leftarrow 1$ *to* E **do**
  Preform standard training procedure
  Preform weights update
  accuracy $\leftarrow$ model accuracy
  if $e \bmod t = 0$ *and* *accuracy* $>$ *best accuracy* **then**
    best accuracy $\leftarrow$ accuracy
    for *each layer* **do**
      Compute the activation for each neuron Eq.(1)
      Vote for largest activations Eq.(2)
      Compute the amount of times a neuron has been voted Eq.(3)
      Vote for $k\%$ of largest voting-score neurons Eq.(4)
      Prune the non-important neurons
    end
  end
end

B. Pruning algorithm

We introduce a method that measures the importance of a network's neurons and prunes them accordingly during training. Our criterion for measuring the value of individual neuron and finding less important ones is critical, as it allows us to effectively identify and prune redundant neurons. As shown in Fig.1, our pruning algorithm starts with standard network architecture and preform standard training. After a certain number of iterations t, we measure the neurons' importance, applying our majority voting method, as described in the previous section III-A. For every layer, we come up with a binary vector that indicates $k\%$ of important neurons, which is a result of the MV method Eq.4. To eliminate the non-informative neurons, we remove a certain number of neurons that have the lowest voting scores based on the predefined percentage. The complete algorithm is given in Algorithm 1.

Starting with standard training, we begin to apply our pruning algorithm based on certain conditions, including a certain number t of epochs and the observation of the model's accuracy. Pruning neurons at the beginning might lead to the permanent removal of essential neurons; we therefore start the pruning after a certain number t of epochs. We continue to employ our pruning algorithm while surveying the conditions. If we prune neurons in each epoch, the final number of neurons would be too small to maintain reasonable accuracy. This setting allows our model to learn and retain important parameters which provide our pruning algorithm with valuable guidance to identify non-important neurons and remove them accordingly.

IV. Experiments and Discussion

We empirically study the performance of our proposed method using two different datasets: MNIST and Cifar10. For fully-connected models, the network architecture consists of three fully-connected layers, which is adopted by the base architecture proposed in [18]. Specifications of the datasets and their architecture are presented in Table. I. We first compare our evaluative importance method with several baselines,

using an ablation study. The experimental results on both datasets show that our method substantially outperforms the baselines. After this, we apply the proposed method to remove redundant nodes and compress the neuron network during training. Then, we integrate our neuron-pruning method with existing sparsely-connected network models. Experimental results show that our method adds substantial compression and further reduces the number of parameters, without harming the accuracy. Lastly, we empirically study our method with convolutional neural networks architecture. The details are described below.

Dataset	*Examples*	*Image Size*	*FC Architecture*
MNIST	70000	28x28x1	784-1000-1000-1000-10
CIFAR10	60000	32x32x3	3072-4000-1000-4000-10

TABLE I: Details of Datasets and their FC Architectures used in our experiments.

A. Measuring Neuron Importance via Ablation

Classification performance was used in order to evaluate the impact of our majority voting method. An ablation study, as a commonly used technique, allows for the evaluation of the effectiveness of measuring neuron importance quantitatively. This procedure typically refers to the removal of some parts of the model and the study of its performance, as crucial neurons capture meaningful information and contribute substantially to the model's final performance [21]. We ablate unimportant neurons by forcing the activation to be zero and compute the classification accuracy on the test-set. Quantifying the effect of the ablation on the classification performance allows for an impartial evaluation in order to measure a neuron's importance and distinguish the most important neurons in a neural network, allowing for layer-wise and whole-network comparisons. This method not only enables the evaluation of neurons' importance, but can also detect the unimportant, redundant neurons, which can be removed while compressing the network during training.

To evaluate the effectiveness of our proposed importance method, it was compared with several baseline methods. These methods are briefly summarized as follows:

- **Random.** Neurons are randomly ablated.
- **Weights sum [12], [14].** Neurons (p) with lowest absolute weights sum values are ablated: $\psi_p = \sum_p \mid w_{p,j} \mid$.
- **Activation Mean[14]**. $\psi_p = \frac{1}{N} \sum mean(t_p)$, where t_p is the activation values for neuron p, and N denotes the size of data.
- **Activation standard deviation (SD) [14]**. $\psi_p = \frac{1}{N} \sum std(t_p)$.
- **Activation $l1$-norms[14]**. $\psi_p = \frac{1}{N} \sum \|t_p\|_1$.
- **Activation $l2$-norms[14]**. $\psi_p = \frac{1}{N} \sum \|t_p\|_2$.

All these baseline methods consider neurons with higher values as more important, which is motivated by the intuition that unimportant activation has influential outputs to the final prediction of a model. Following [14], we calculate neurons' importance measure on the activation of the neurons before batch normalization or non-linear activation.

Table. II and Table.III summarize the classification results on CIFAR10 and MNIST test-sets respectively. These tables provide layer-wise comparisons, where neuron importance was evaluated using different selection criteria in a fully trained model utilizing the ablation approach. A compression ratio of 0.7 was set, where 70% of the important neurons in each layer are preserved. The tables show layer-wise results for each layer, where we ablate layer by layer and calculate the accuracy for each layer separately. We also examine the whole network cumulative ablation, where the same ratio from all layers in the whole network are ablated. For random selection criteria, the mean value of three runs are reported.

Our ablation study has shown that MV achieves higher classification performance compared with other baselines, especially in the case of the whole network cumulative ablation. In CIFAR, Table. II, MV achieves 68.28% when having compression ratio of 0.7 for each layer of the reference model. This shows that MV has less impact on the dropping of model accuracy compared with the second place method, which used activation standard deviation, where it achieves 66.33%. This demonstrates the robustness of our proposed method in identifying the most important neurons.

One interesting finding is that ablating neurons with random selection shows the first layer has stronger negative effects and has more synergistic neurons compared with higher hidden layers. It can also be seen that the higher hidden layers are significantly redundant and more class-specific. This observation is consistent with a previous theoretical proposal [22]. One reasonable explanation is that the neuron networks hierarchically learn representations. Hence, the first layer is not relevant to a specific object. Still, it builds feature representations of all input images that are joined to form more relevant object features in the later layers. By ablating these fundamental features, deeper layers fail to produce class-specific features and have more negative impacts on the overall accuracy.

Although a random selection is not robust and not applicable in practice [15], it presents insight and demonstrates that the detection of principal neurons is a critical approach when pruning redundant neurons. The experiment empirically confirms that our importance method is sufficient, given that ablating neurons with low values in the layers only has the most negligible impact on the overall accuracy compared with all baselines. As shown in Table.II and Table. III, the experimental results on both datasets show that the method substantially outperforms the baselines. Our proposed method to measure neuron importance helps not only to remove redundant nodes and compress the neuron network, but also to understand their inter-relationships and how said neurons impact the model. The experiment confirms that selecting the right criteria to evaluate neurons' importance throughout all layers can guarantee a successful pruning approach.

	1st Layer	*2nd Layer*	*3rd Layer*	*Cumulative Ablation*
Random	45.46%	61.84%	65.07%	21.39%
Weights Sum	63.75%	67.47%	67.04%	48.62%
Activation Mean	68.97%	68.47%	68.48%	64.75%
Activation SD	69.49%	68.90%	69.22%	66.33%
Activation $l1$-norms	69.39%	68.71%	69.32%	65.94%
Activation $l2$-norms	69.45%	68.73%	69.31%	65.81%
MV	**69.77**%	**69.39**%	**69.66**%	**68.28**%

TABLE II: Examining neuron importance via ablation study with different selection criteria on CIFAR10.Note. the compression ratio=0.7 i.e., 70% of the importance neurons in each layer are preserved.

	1st Layer	*2nd Layer*	*3rd Layer*	*Cumulative Ablation*
Random	95.4%	97.96%	98.41%	85.32%
Weights Sum	95.00%	98.39%	98.57%	94.63%
Activation Mean	97.88%	98.52%	98.58%	97.98%
Activation SD	98.58%	98.73%	98.68%	98.44%
Activation $l1$-norms	98.56%	98.72%	98.65%	98.40%
Activation $l2$-norms	98.51%	98.73%	98.67%	98.37%
MV	**98.68**%	**98.75**%	**98.76**%	**98.68**%

TABLE III: Examining neuron importance via ablation study with different selection criteria on MNIST. Note. the compression ratio=0.7 i.e., 70% of the importance neurons in each layer are preserved.

B. Pruning redundant Neurons during Training

The proposed method was implemented using Keras and Tensorflow in Python and evaluated on two computer vision benchmark datasets: MNIST and CIFAR10. The models were trained end-to-end from scratch without involving any pre-training and fine-tuning procedures. All weights were initialized randomly. *Stochastic gradient descent* optimizer was used, where each batch contained 100 random shuffled images. An initial learning rate of 0.006 with a momentum of 0.9 and weight decay of 0.0002 were used. For our experiments, the value of t is set to be 20, and the value of k is set to be 0.05 as a large value of k leads to the removal of many neurons and the remaining neurons would be too small to maintain reasonable accuracy.

During training, we pruned the network's neurons, after having applied our method to measure the importance of each neuron independently. Consequently, we extended the TensorFlow framework to prune the neurons of a network during training. TensorFlow allows us to apply a constraint function to the weights matrix, which in turn means that constraints can be set on network parameters during optimization. For every pruned layer, we utilized the output binary vector, which is obtained via the importance measure method (MV) Eq.4, for the constraint function. The binary vector contributes to generating a binary mask variable which has the same size as the layer's weight matrix. The binary mask determines the participation of the weights in the forward procedure.In the back-propagation, gradients pass through the binary masks, and the masked weights in the forward-propagation are not updated in the back-propagation phase.

Determining and eliminating the non-informative neurons results in a significant additional increment into the body of the learning process. This approach is aimed at forming a kind of structure that enhances the identification of non-informative neurons and removes any redundant parts of the model during training. The comparative results are shown in Table. IV, and supports our hypothesis that significantly fewer parameters can add enough discriminative ability without harming the original accuracy of the baseline dense models. These are considered a solution to overcoming the over-parametrized and redundant nature of deep learning models. It can be observed that the models' accuracy were maintained or even improved after the removal of unimportant neurons.

Table. IV demonstrates how the pruned version of models outperform the original, fully-connected models with only significantly fewer parameters. With regards to the CIFAR10 dataset, it has been shown that a significant gain can be obtained with only 20% of the weights of the original fully-connected model. Based on CIFAR10-related literature, [23] is considered as one of the state-of-the-art fully-connected models, which achieves a classification accuracy of 74.1% with $31,600K$ parameters, while our model reached a comparable accuracy of 74.21% with only around $4,245K$ parameters.

	FC		*MV Pruning*	
Dataset	***Accuracy***	n^W	***Accuracy***	n^W
MNIST	98.78%	2,794K	98.88%	232K
CIFAR10	71.90%	20,328K	74.21%	4,245K

TABLE IV: Summarization of our experimental results with fully-connected networks.

C. Integrating our Pruning Method to Existing Sparse Neural Network

To thoroughly investigate the abilities of our proposed method, we also evaluated its performance with sparsely-connected networks (SC). SC by Mocanu et al. [18] is an interesting approach that replaces fully-connected layers with sparsely-connected ones. They have introduced a way to connect nodes in neural networks before training by applying an initial sparse topology based on the Erdős–Rényi random

graph and by starting training using standard optimization techniques. They iteratively remove the weakest connections and replace them with the new random initialization, which leads to a substantial reduction in connections and, therefore, to increased memory and computational efficiency.

A massive number of neurons is still a significant challenge, as it can lead to significant redundancy. Although sparsely-connected layers remove unnecessary connections without significant performance degradation, neuron-pruning methods are much more beneficial. This is because unimportant neurons do not contribute much to the final model performance, as shown in the previous section, therefore, all of their income or outcome connections (weights) are trivial and non-informative. Eliminating unimportant neurons can guarantee the removal of extra parameters, as pruning a neuron removes entire rows or columns of the weight matrices from the former and later layers.

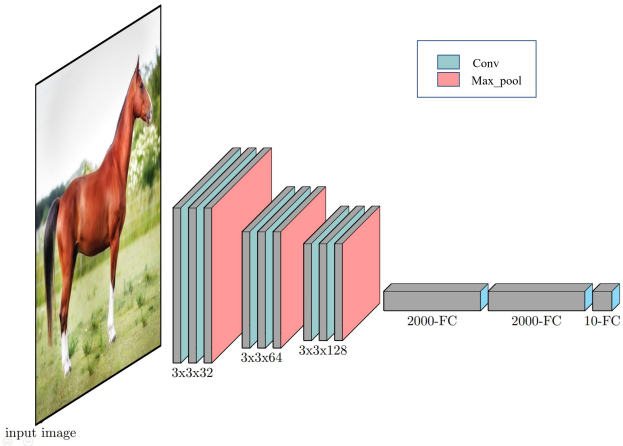

Fig. 2: The architecture of the CNN model.

Our experiment began with sparse topology random graph, after which [18] the weakest connections were iteratively replaced with new initialized ones in the training phase. In the meantime, we computed a measure of relevance that identified the less critical neurons and pruned them accordingly. Table. V demonstrates that on the MNIST dataset, with our neuron-pruning method, we can prune up to 60% of parameters from the original sparsely-connected models without harming the performance. This supports our hypothesis that pruning unimportant neurons is just as essential as pruning unimportant weights, and that combining both can lead to competitive results, as shown in our findings.

	SC [18]		***MV Pruning***	
Dataset	***Accuracy***	n^W	***Accuracy***	n^W
MNIST	98.74%	89K	98.84%	34K
CIFAR10	74.84%	278K	75.05%	214K

TABLE V: Summarization of our experimental results with sparsely-connected networks.

D. Extension to Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [24] are one of the most famous kinds of neural networks. They replace fully-connected layers with convolution and pooling layers, which significantly decreases the number of parameters. CNN architecture usually comprises convolutional layers for spatially-related feature extraction and fully-connected layers used for classification. However, while CNNs still maintain fully-connected layers, they can additionally benefit from our pruning method.

We studied our pruning method with CNN architectures, where we compressed their fully-connected layers, as they form the majority of the CNN parameters. For instance, VGG16 [25] comprises 90% of its parameters in fully-connected layers. Our model architecture consists of three convolutional blocks, where each block has two convolutional layers with a filter size of 3 x 3 with 32 kernels in the first block, 64 kernels in the second block, and 128 kernels in the third block. Each block ends with a max-pooling layer. This is followed by three fully-connected layers consisting of 2000, 2000, and 10 neurons respectively. A standard Relu activation function was utilized. The detailed architecture of the CNN model is presented in Fig. 2. Because the fully-connected layers form 96.6% of the overall parameters of our architecture, it justifies our focus on the fully-connected layers in the convolutional neural networks.

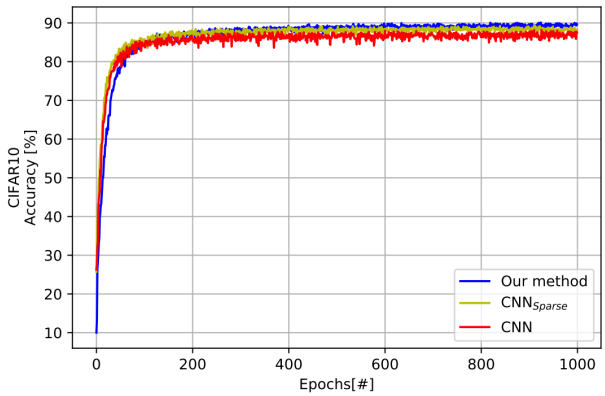

Fig. 3: Changing of accuracy during training for three different models.

Our experiments were performed on the CIFAR10 dataset. To quantify this, it should be noted that our experiment reached a maximum of 90.12% accuracy, SC achieves a maximum of 89.30% accuracy, while standard CNN achieves a maximum of 87.65% accuracy. Each model has 8,407,018, 456,077, and 393,549 weights for CNN, SC and our method, respectively. Our pruned model has shown better accuracy than standard CNN, having removed more than 95% of its parameters. In other words, with only less than 5% of the CNN weights, our method can achieve better accuracy. Fig. 3 also shows the changes in validation accuracy in classification tasks with the number of training cycles, which clearly indicates

that classification stably converges using an iterative pruning scheme.

V. Conclusion

In this paper, we propose a pruning framework which simultaneously identifies the most critical neurons and removes redundant nodes accordingly. The experimental results have demonstrated the effectiveness of our pruning method in maintaining or even improving accuracy after removing unimportant neurons. The results also demonstrate that our proposed method is applicable to weight-based pruning methods and adds extra compression. Our potential future work is to extend this framework to filters in convolutional neural networks and experiment with more difficult datasets.

Acknowledgment

This work was supported by The Engineering and Physical Sciences Research Council (EP/N028139/1) and a Sêr Cyrum II Fellowship (663830-SU158).

References

[1] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," in *Advances in neural information processing systems*, 2013, pp. 2148–2156.

[2] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4820–4828.

[3] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6541–6549.

[4] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[5] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "Gan dissection: Visualizing and understanding generative adversarial networks," *International Conference on Learning Representations*, 2019.

[6] K. Dhamdhere, M. Sundararajan, and Q. Yan, "How important is a neuron?" *International Conference on Learning Representations*, 2019.

[7] R. A. Amjad, K. Liu, and B. C. Geiger, "Understanding individual neuron importance using information theory," *arXiv preprint arXiv:1804.06679*, 2018.

[8] S. Na, Y. J. Choe, D.-H. Lee, and G. Kim, "Discovery of natural language concepts in individual units of cnns," *International Conference on Learning Representations*, 2019.

[9] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[10] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in neural information processing systems*, 1993, pp. 164–171.

[11] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[12] T. He, Y. Fan, Y. Qian, T. Tan, and K. Yu, "Reshaping deep neural network for fast decoding by node-pruning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 245–249.

[13] Z. Mariet and S. Sra, "Diversity networks: Neural network compression using determinantal point processes," *International Conference on Learning Representations*, 2016.

[14] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *International Conference on Learning Representations*, 2017.

[15] J. Luo, H. Zhang, H. Zhou, C. Xie, J. Wu, and W. Lin, "Thinet: Pruning cnn filters for a thinner net," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2525–2538, Oct 2019.

[16] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang, "Stronger generalization bounds for deep nets via a compression approach," *arXiv preprint arXiv:1802.05296*, 2018.

[17] A. S. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick, "On the importance of single directions for generalization," *International Conference on Learning Representations*, 2018.

[18] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature communications*, vol. 9, no. 1, p. 2383, 2018.

[19] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," in *Proceedings of the British Machine Vision Conference (BMVC)*, M. W. J. Xianghua Xie and G. K. L. Tam, Eds. BMVA Press, September 2015, pp. 31.1–31.12. [Online]. Available: https://dx.doi.org/10.5244/C.29.31

[20] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *International Conference on Learning Representations*, 2019.

[21] A. Alqahtani, X. Xie, J. Deng, and M. W. Jones, "Learning discriminatory deep clustering models," in *International Conference on Computer Analysis of Images and Patterns*. Springer, 2019, pp. 224–233.

[22] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein, "On the expressive power of deep neural networks," in *International Conference on Machine Learning*. JMLR. org, 2017, pp. 2847–2854.

[23] G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson, "Do deep convolutional nets really need to be deep and convolutional?" *International Conference on Learning Representations*, 2017.

[24] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, 2015.