International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307 (Online), Volume-14 Issue-5, November 2024

# Efficient Layer Optimizations for Deep Neural Networks

**Shafayat Mowla Anik, Kevyn Kelso, Byeong Kil Lee**

***Abstract*: Deep neural networks (DNNs) have technical issues such as long training time as the network size increases. Parameters require significant memory, which may cause migration issues for embedded devices. DNNs applied various pruning techniques to reduce the network size in deep neural networks, but many problems still exist when applying the pruning techniques. Among neural networks, several applications applied autoencoders for reconstruction and dimension reduction. However, network size is a disadvantage of autoencoders since the architecture of the autoencoders has a double workload due to the encoding and decoding processes. In this research, we chose autoencoders and two deep neural networks – AlexNet and VGG16 to apply out-of-order layer pruning. We perform the sensitivity analysis to explore the performance variations for the network architecture and network complexity through an out-of-order layer pruning mechanism. As a result of applying the proposed layer pruning scheme to the autoencoder, we developed the accordion autoencoder ($A^2E$) and applied credit card fraud detection and MNIST classification. Our results show 4.9% and 13.6% performance drops, respectively, but we observe a significant reduction in network complexity, 85.1% and 94.5% for each application. We extend the out-of-order layer pruning to deeper learning networks. In our approach, we propose a simple yet efficient scheme, accuracy-aware structured filter pruning based on the characterization of each convolutional layer combined with the quantization of fully connected layers. We investigate the accuracy and compression rate of each layer using a fixed pruning ratio, and then the pruning priority is rearranged depending on the accuracy of each layer. Our analysis of layer characterization shows that the pruning order of the layers does affect the final accuracy of the deep neural network. Based on our experiments using the proposed pruning scheme, the parameter size in the AlexNet can be up to 47.28x smaller than the original model. We also obtained comparable results for VGG16, achieving a maximum compression rate of 35.21x.***

***Keywords*: Deep Neural Network; Machine Learning; Filter pruning; Network Compression; Layer Pruning.**

## I. INTRODUCTION

Deep neural networks (DNNs) require huge computational workloads with large memory capacity to train weight values in deeper layers and conduct inference operations. Also, large-scale DNNs are hard to apply to embedded devices with limited computational resources. To resolve those issues, model compression techniques (e.g., pruning and quantization) for neural networks have been proposed to reduce the model redundancy without significantly degrading accuracy and performance.

Several applications applied autoencoders for dimensionality reduction, data reconstruction, anomaly detection, and classification. For dimensionality reduction, the lower dimension of data can generalize sets of higher dimensions that provide intuitive features. In modern applications, generative models use lower dimensional latent spaces to change the features of the output. Also, autoencoders have been used in quantum data compression problems [1]. However, autoencoders are not used in practical data compression because other modern algorithms perform better without the required training. It would also be impractical to gather a dataset for each compression application because the autoencoder networks are not performing well on dataset transfer learning. This lack of dataset flexibility is one of the main reasons that autoencoders are one of the potential candidates for anomaly detection. Once the autoencoder network has trained with a particular dataset, any data fed through the network slightly outside the training dataset results in a high reconstruction loss. However, network size is a disadvantage of autoencoders since the architecture of the autoencoders has a double workload due to the encoding and decoding processes.

As the first application for layer characterization and layer optimization, in this paper, we select autoencoders to perform the sensitivity analysis to explore the performance variations for the network architecture and the network complexity through an out-of-order layer pruning mechanism. The motivation behind exploring different autoencoder architectures is their practical uses for applications such as anomaly detection, classification, and other usages in generative models generating novel output using the latent space in the autoencoder. Additionally, understanding the mechanism to improve dimensionality reduction may help in understanding the behaviors of the human brain.

We extend the out-of-order layer optimization to deeper learning networks.

**Manuscript received on 25 September 2024 | Revised Manuscript received on 14 October 2024 | Manuscript Accepted on 15 November 2024 | Manuscript published on 30 November 2024.**

*Correspondence Author(s)
**Shafayat Mowla Anik**, Department of Electrical and Computer Engineering, University of Colorado Colorado Springs, 1420 Austin Bluffs Parkway, Colorado Springs, USA. Email: sanik@uccs.edu,
**Kevyn Kelso**, Department of Electrical and Computer Engineering, University of Colorado Colorado Springs, 1420 Austin Bluffs Parkway, Colorado Springs, USA. Email: kkelso@uccs.edu,
**Byeong Kil Lee***, Department of Electrical and Computer Engineering, University of Colorado Colorado Springs, 1420 Austin Bluffs Parkway, Colorado Springs, USA. Email: blee@uccs.edu. ORCID ID: 0000-0002-0260-2238



*Retrieval Number: 100.1/ijsce.E365014051124*
*DOI: 10.35940/ijsce.E3650.14051124*
*Journal Website: www.ijsce.org*

*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)*

The computational complexity of the network depends on the effectiveness of the pruning algorithm used in identifying and preserving the essential information from the original network. Various research proposes to reduce the complexity of workloads. In our approach, we develop a simple and accuracy-aware structured filter pruning based on the characterization of each convolutional layer combined with the quantization of fully connected layers. Among various pruning schemes, filter pruning is a naturally structured prune method without introducing sparsity. Therefore, it does not require sparse libraries or specialized hardware [2]. We investigate the accuracy and compression rate of each layer using a fixed pruning ratio, and the pruning priority is rearranged based on the accuracy of each layer. Our analysis of layer characterization shows that the pruning order of the layers does affect the final accuracy of the deep neural network. Based on our experiments using the proposed pruning scheme, the parameter size of AlexNet can be reduced up to 47.28x compared to the original model with an accuracy loss of less than 1%. Similar results are also obtained with VGG16, achieving a maximum compression rate of 35.21x.

The rest of the paper has organized as follows. In Section II, we describe related work. Section III describes the layer characterization of autoencoders and DNNs. Section IV describes the proposed out-of-order layer pruning method and experimental results. In Section V, we describe the obtained results and analyze them. Finally, we conclude in Section VI.

## II. RELATED WORK

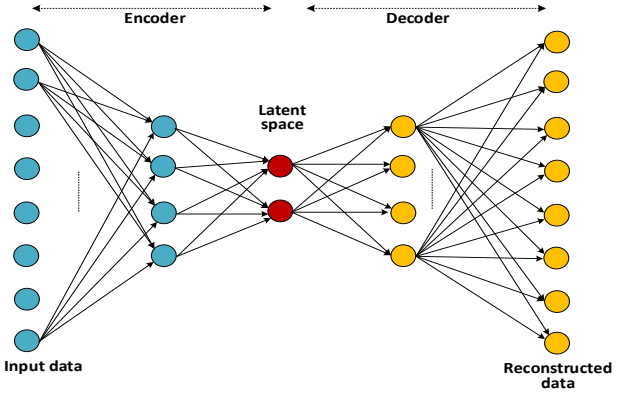
Tirumala [3] introduced a simple pruning strategy in that all connections with weights below a threshold are excluded from training to reduce computational workloads. This process had to be followed by fine-tuning to recover the accuracy lost by eliminating weights. To avoid the limitations of non-structured pruning, other papers [4] and [5] argued that filter-level pruning could be a better option. In [4], a compressed model proposes to prune the unimportant filters of CNN models in both training and testing stages, providing performance acceleration and network compression. Also, in [5], Luo studied the effect of pruning sensitivity to decide the pruning rate for the different convolutional layers. For further compression, some research works have implemented post-training quantization after pruning. In [6], the model size has reduced with three stages of the compression pipeline: weight pruning, quantization, and Huffman coding. However, one noticeable problem when using pruning is the large number of epochs required for retraining the model after pruning. In our approach, we will introduce one parameter: the retraining order of the layers. We will show how this parameter can affect the fine-tuning process and the accuracy with a smaller number of epochs.

## III. LAYER CHARACTERIZATION

### A. Implementation of Pruning

An autoencoder is a fully connected neural network that compresses the input into a smaller and more meaningful representation and then decodes that representation into an output resembling the original input [7]. Usually, decreasing the dimensionality of the input is the goal of the autoencoder. Generally, the autoencoder learns distinguishable attributes of the data represented in the latent space. The latent space is a lower-dimensional vector and the smallest middle layer. These attributes tend to be generalizations of the data representing one or many features in the input. The latent attributes can sometimes resemble the results of principal component analysis (PCA) but often perform better due to the nonlinearity of the encoder and decoder layers [8][9]. Figure 1 shows the architecture of autoencoders with two encoding processes, two decoding processes, and one latent space.

We use Tensorflow 2 with Keras to implement all network models. For the performance metrics, we use the F1 score, recall, precision, accuracy, loss, and complexity for the autoencoder. Two applications are used for the experiments, including anomaly detection in credit card transaction data and image classification using the MNIST dataset [10][11]. Generally, precision is a relevance-based metric based on a percentage of correctly predicted or classified instances, and recall is the percentage of retrieved instances. One could classify all instances as one category and have 100% recall but with 0% precision. A problem-specific balance between the two is optimal for best performance. The F1 score is the harmonic mean between precision and recall. Also, the F1 score is considered the primary performance metric for both datasets. The F1 score has adjustable weights to show the importance of precision versus recall. To keep things simple and maintain the F1 score meaningful between datasets, precision and recall are equally weighted in the F1 score for all results.



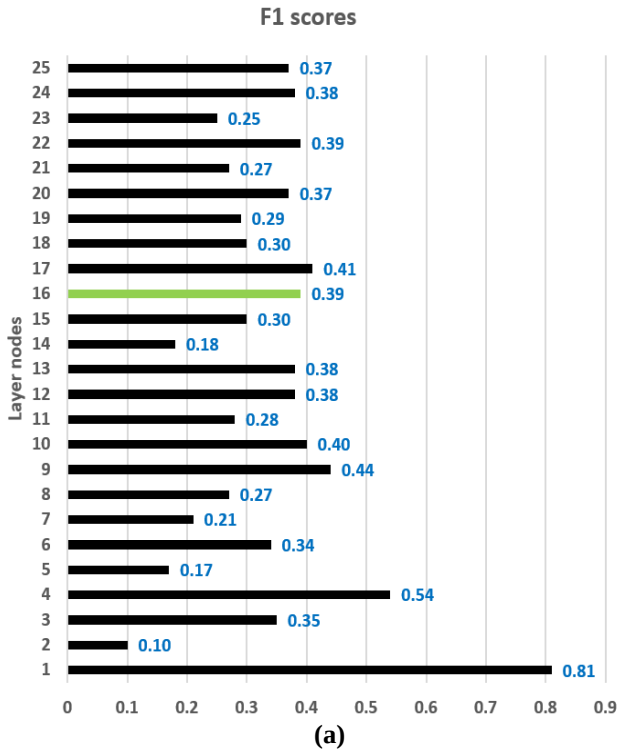

**[Fig. 1: The Architecture of Autoencoder [8-4-2-4-8]]**

There is no established standard for autoencoder architecture for classification or anomaly detection. For characterizing the autoencoder network in this research, we use the architecture [16-8-4-2-4-8-16] as a baseline architecture [10]. The first layer of the baseline autoencoder's encoding module has 16 nodes, contributing to the model's complexity. One of the goals of this research is to reduce network complexity. We perform the performance sensitivity analysis by changing the number of nodes in the first layer (at the encoding side) and the seventh layer (at the decoding side) to preserve the symmetry of the architecture. Figure 2 (a) shows the performance (F1 score) sensitivities for Layer 1 and 7.

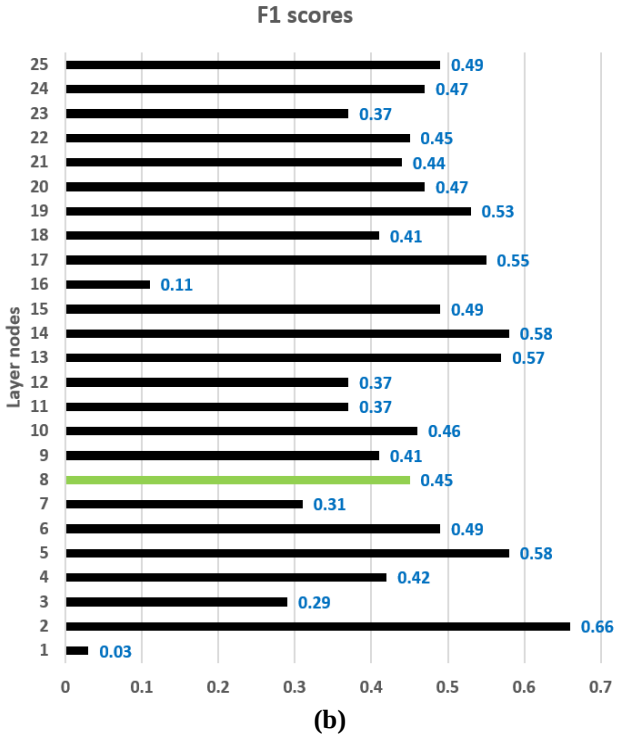
*Retrieval Number: 100.1/ijsce.E365014051124*
*DOI: 10.35940/ijsce.E3650.14051124*
*Journal Website: www.ijsce.org*

*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)*

Based on our observation of sensitivity analysis, we observe notable improvement with the smaller number of nodes rather than the larger ones. If we reduce the number of nodes of Layer 1 into Layer 4, we can get better performance (37.5% improvement) and lower complexity (41.5% reduction). The performance can vary in other applications or datasets, but our results from the sensitivity analysis led us to further characterization and performance evaluation.




(a)

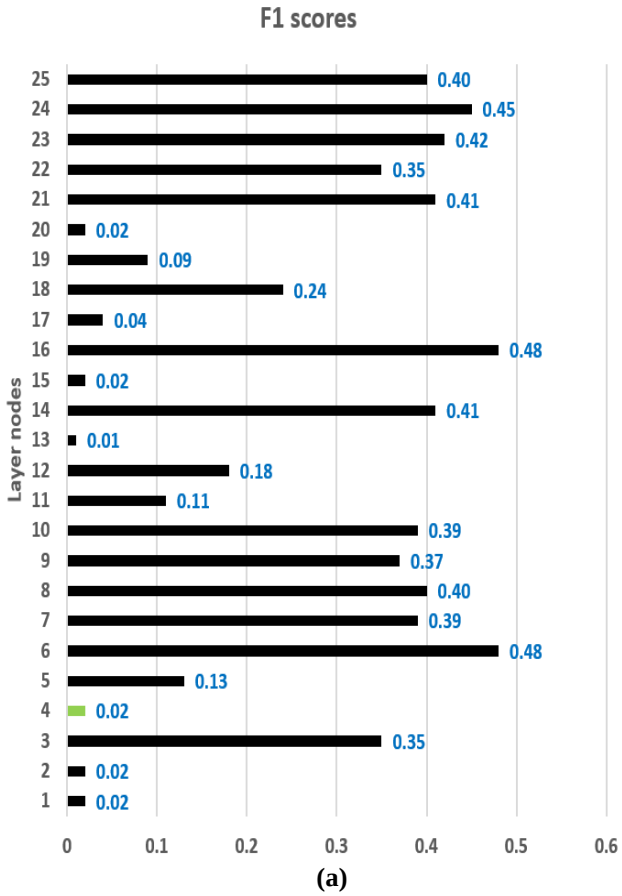



(b)

**[Fig.2: Performance Sensitivity Analysis: F1 Score vs. the Number of Nodes in Layer 1-7 Pair and 2-6 Pair: (a) Layer 1 and Layer 7 variations (Layer 16 is the Original Node with Green Bar); (b) Layer 2 and Layer 6 Variations (Layer 8 is the Original Node with Green Bar)]**

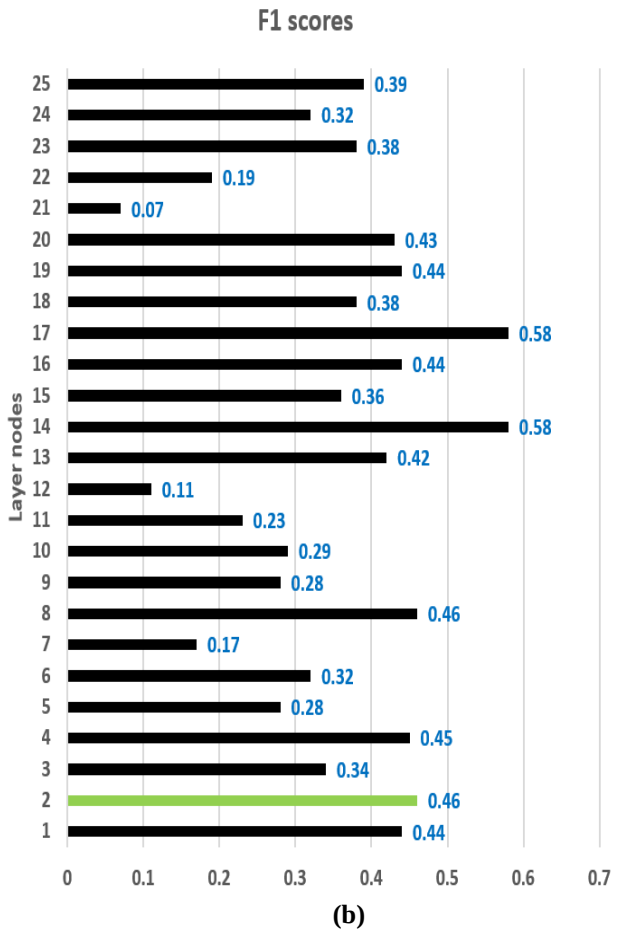
In the second step, after changing the number of nodes in Layer 2 and Layer 6 while maintaining four nodes in Layer 1 and Layer 7, we observe the performance variations. Based on the results from the experiments, 2 is the best-performing number of nodes for layers 2 and 6 in terms of the F1 score shown in Figure 2 (b). As the next step for layers 3 and 5, we observe that the best number of nodes is six from performance (F1 score) based on the sensitivity analysis shown in Figure 3 (a). As a final step, we change the number of nodes in the latent space to see the performance variations. However, distinct performance improvement has not observed from the sensitivity analysis. As shown in Figure 3 (b), performance variations are limited, and the original number of nodes is among the best performance groups.

The sensitivity analysis performs on a specific application with the dataset. It is hard to say the scheme we used for extracting the best-performing number of nodes is a general solution. However, it provides meaningful results where some performance numbers (F1 score, precision, complexity, and recall) are improved while the accuracy number is marginally dropped (13.6%). Based on our analysis, accuracy has some inverse relationships to the F1 classification metrics, where, up to a threshold, lower accuracy scores provide improved F1, precision, and recall scores.




(a)

*Retrieval Number: 100.1/ijsce.E365014051124*
*DOI: 10.35940/ijsce.E3650.14051124*
*Journal Website: www.ijsce.org*



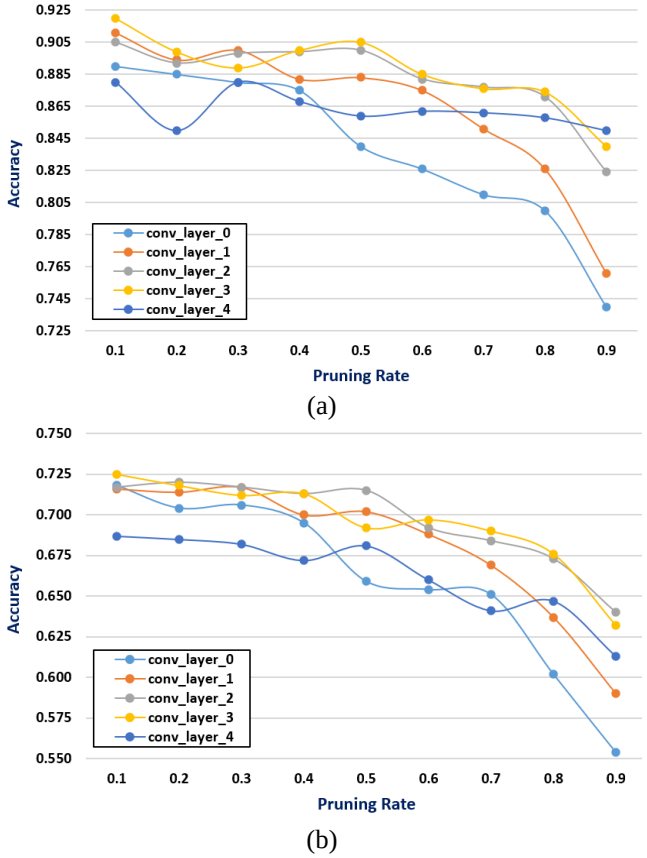

(b)

**[Fig.3: Performance Sensitivity Analysis: F1 Score vs. the Number of Nodes in Layer 3-5 Pair and the Latent Space: (a) Layer 3 and Layer 5 variations (Layer 4 is the Original Node with Green Bar); (b) The Latent Space Variations (Layer 2 is the Original Node with Green Bar)]**

### B. Layer Characterization of DNNs

We perform the layer characterization in two different models: AlexNet and VGG16 [12][13]. The two datasets chosen are CIFAR-10 and CIFAR-100 [14][15]. CIFAR-10 consists of 60,000 images that belong to 10 different classes. 50,000 images are used for training and 1,000 for testing. CIFAR-100 is another version of CIFAR-10 with the same number of images, but it contains 100 classes instead of 10. Pytorch [16] has used to conduct the proposed experiments. Transfer learning is applied as both CNN architectures are previously pre-trained with the ImageNet [17] dataset. We used a batch size of 64, a learning rate 0.001, and SGD as the optimizer. As for the hardware, all experiments has done with NVIDIA Quadro P6000.

AlexNet with CIFAR-10 and CIFAR-100: Our first DNN characterizations are pruning AlexNet with CIFAR-10 and CIFAR-100. AlexNet has five convolutional layers: conv0 to conv4. As a first step, we applied the pruning sensitivity sweep for both datasets separately. Figure 4 shows pruning results with CIFAR-10. It is noticeable that the accuracy for convolutional layer 4 starts at a lower point than the rest of the layers and does not go over 0.9 in all the sweeps.




**[Fig.4: AlexNet Pruning Sweep Using CIFAR-10 and CIFAR-100: (a) Accuracy for Pruning Sweep with CIFAR-10; (b) Accuracy for Pruning Sweep with CIFAR-100]**

VGG16 with CIFAR-10 and CIFAR-100: We also apply our method to deeper networks to analyze its performance in more complex architectures. Figure 5 shows the pruning sweep of VGG16 using CIFAR-100. In this case, the threshold factor is 0.99 for CIFAR-10 and 0.997 for CIFAR-100. Likewise, with AlexNet, the pruning is not applied to the last convolutional layer because it would damage the performance of the final model. The model has retrained with five epochs after pruning each layer.

## IV. OUT-OF-ORDER LAYER PRUNING

### A. Accordion Autoencoders ($A^2E$)

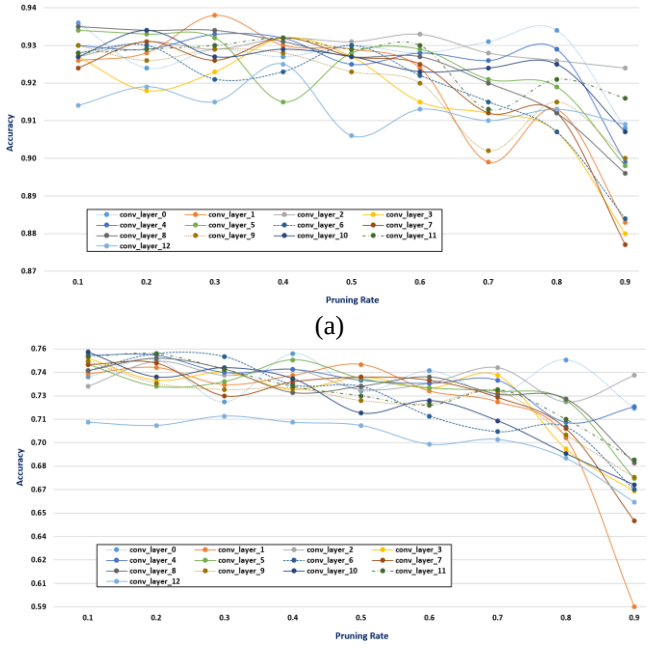
Generally, the architecture of the autoencoders has two parts – the down-sampling part (encoding) and up-sampling (decoding). The complexity is monotonically decreased and increased throughout the process. Based on our sensitivity analysis of fraud detection cases, the extracted best-performing architecture [4-2-6-2-6-2-4] does not have the feature of monotonicity. The number of nodes in each layer is up and down. We name it accordion autoencoders ($A^2E$) because the shape of the architecture is similar to the accordion, as shown in Figure 6. In computer vision, the pyramid representation, a multi-scale image data representation, is used to extract the features through subsampling.

*Retrieval Number: 100.1/ijsce.E365014051124*
*DOI: 10.35940/ijsce.E3650.14051124*
*Journal Website: www.ijsce.org*



*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)*

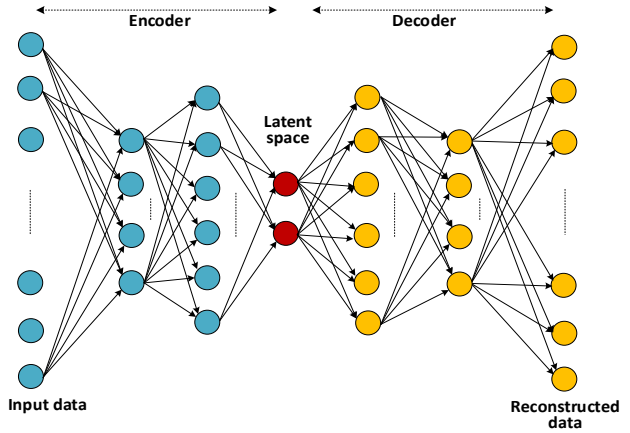
In autoencoders, we can extract features from the first layer. Those features are typically lower dimensions than the input size. Then, as in accordion autoencoders, if we increase the dimension from the subsampled data in the second layer, many meaningful features will be lost or deformed. However, we can keep the features through the iterative cost reduction in deep learning, and the output images have been used for another dimension reduction and feature extraction. The accordion architecture will anneal the features through the dimension up/downs and the iterative optimizations.




(a)

(b)

**[Fig.5: VGG16 Pruning Sweep using CIFAR-10 and CIFAR-100 (Convolution layers 0-12): (a) Accuracy for Pruning Sweep with CIFAR-10; (b) Accuracy for Pruning Sweep with CIFAR-100]**

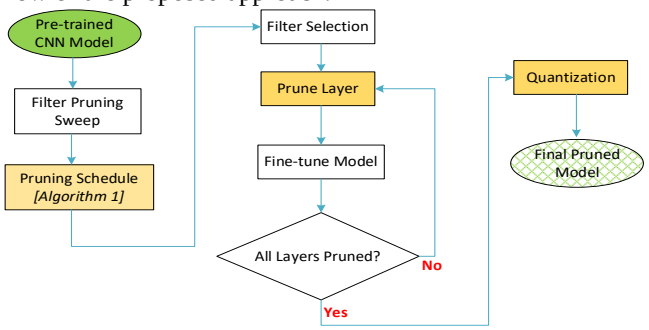



**[Fig.6: The Accordion Autoencoder (A²E) [N-4-6-2-6-4-N]]**

The general concept of the accordion autoencoder is that data gets compressed and decompressed multiple times to get more meaningful features recorded at each compression stage. Essentially, it is an extension of the ability of the classic autoencoder to record meaningful features due to the memory capacity in the expansion layers immediately following each latent space layer. Presumably, the most meaningful features will end up in the last latent space after being 'autoencoded' several times to ensure only features that affect the reconstruction loss.

### B. Out-of-order Layer Pruning for DNNs

Firstly, we describe the filter pruning technique used for initial compression through characterizing individual layers. Then, we explain how the pruning rate and the order of the layers are selected. Finally, we use quantization in fully connected layers to achieve further compression while maintaining the model's accuracy. Figure 7 shows the overall flow of the proposed approach.




**[Fig.7: Overall Flow of DNN Out-of-Order Layer Pruning]**

We consider a CNN network with $L$ convolutional layers. The weight matrix in each layer, $W_i$, can be expressed as $W_i \in \mathbb{R}^{N_{i+1} \times N_i \times k \times k}$ where $N_i$ denotes the number of input channels of the convolutional layer $i$ and $N_{i+1}$ next layer's input channels. The number of input channels is the same as the number of output channels for the layer $i$. The constant $k$ represents the kernel dimension, considering that height and width are the same. We denote the kernel as $\mathcal{K} \in \mathbb{R}^{k \times k}$. Every filter $j$ in layer $i$ can be expressed as $\mathcal{F}_{i,j} \in \mathbb{R}^{N_i \times k \times k}$. The $P_i$ is the pruning rate for each layer.

The first step of the proposed method is applying filter pruning. The process consists of identifying those filters considered unimportant or redundant. The importance of the filters needs to be evaluated according to a criterion to identify the redundant filters. After evaluation of different norms through our experiments, we select $\ell 2$ norm, also known as the Euclidean norm, as a criterion:

$$\|\mathcal{F}_{i,j}\|_2 = \sqrt[2]{\left|\sum_{n=1}^{N_i}\sum_{k=1}^{K}\sum_{k=1}^{K}\left|\mathcal{F}_{i,j}(n,k,k)\right|\right|^2} \quad \ldots (1)$$

Evaluating each layer of the CNN model, we can rank the filters by assuming the ones with a smaller norm will lead to lower activation values and less impact in the final classification. Filter selection for every layer using pruning with $\ell 2$ norm can be formulated mathematically as:

$$\forall i \in \{1, \ldots, L\}: \min_{j \in S} \ell_2(\mathcal{F}_{i,j}) \quad \text{s.t } S \subset \{1, \ldots N_{i+1}^{\mathcal{S}\mathcal{S}}\}, N(S) = N_{i+1} \times \mathcal{P}_i \quad (2)$$

$S$ represents a subset of all the possible output channels. N(S) is the total number of elements in the S. The S is the number of filters that need pruning. After pruning the selected filters in the layers $i$, we will fine-tune the model by retraining it for a determined number of epochs.



Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

A greedy layer-wise approach applies to layer *i+1*. This process will continue until $i = L$.

Before the final pruning stage, the sensitivity analysis should be performed on each layer. The accuracy values from the sweeping are used to determine which pruning order would be well-fit and which pruning rate would be adequate for each layer.

**Pruning Scheduling:** In the proposed pruning schedule algorithm, the final pruning rate $P_i$ s for each layer has determined according to $\ell_2$ norm criteria. We will perform a sweep to characterize every layer going from $P_{min} = 0.1$ to $P_{max} = 0.9$ with a step size of 0.1. As a result, only $N_{i+1}$ x $P_i$ among the $N_{i+1}$ channels will be removed, meaning that the final number of output channels in the layer *i* will be $N_{i+1}$ x (1 - $P_i$).

To characterize individual layers, we keep changing the pruning rate from $P_{min}$ to $P_{max}$ until all corresponding accuracy values are available from pruning each layer. Once the sweep has done, enough information will be ready to apply Algorithm 1. The algorithm aims to achieve the maximum compression rate of the model while minimizing the accuracy loss. It is necessary to set a different pruning rate for each layer.

In Algorithm 1, we compare the accuracy result from pruning a layer of the model with a specific $P_i$ to baseline accuracy of the original model. The threshold value is multiplied by a factor from 0.985 to 0.997 depending on the model and dataset to avoid excessively restrictive algorithms. These values are determined experimentally to balance the trade-off between accuracy and compression results. The original accuracy multiplied by this factor is what we consider the threshold. If the accuracy of any layer is lower than the threshold, pruning will not apply to the layer. If the threshold is set too low, the accuracy will be significantly lower compared to the original, while the compression rate of the model will be high. However, if the threshold is set too high, the testing accuracy of the new model will also be higher, but the compression rate will drop. Several experiments have performed with two different models. We observe that the threshold should be closer to the original accuracy to obtain high accuracy in the final model. It occurs when the model has more layers to be pruned, and retrained loss in the final model is likely to accumulate and be higher. That is why the AlexNet compression rate is higher than in VGG16.

Applying Algorithm 1, we obtain the maximum pruning rate to make every layer have the same accuracy or higher than the threshold. Based on individual layers' accuracy, layers has arranged by descending order.

With the results obtained from Algorithm 1, we can proceed to the final pruning of the model. The process is the same as in sweeping, but now we have a fixed pruning rate for every layer based on the analysis of their pruning sensitivity. After pruning one layer, we retrain the model for several epochs. The order for pruning the layers will not be sequential but specified by the pruning schedule. This way, layers that individually achieve the highest accuracies were pruned first. Since previous retraining provides performance improvement, pruning should apply to the layers with relatively poor performance. We observe that changing the order of the layers in the pruning schedule can cause a faster convergence, requiring fewer epochs during retraining to achieve better accuracy results in the final model. When the pruning applies to all the selected layers in the specified order, we can consider that the new and efficient network model is produced and ready for testing or inference.

**Algorithm 1**: Order and $P_i$ for Convolutional Layers

**Input**: A list of test accuracies and corresponding pruning rate for each layer; A list of convolutional layers in the model.

**Output**: Ordered list of convolutional layers and the $P_i$.

1. **for** each *l* in [1, L] **do:**
2. **for** each $P$ in $[P_{min}, P_{max}]$ **do**:
3. **if** testing accuracy ≥ threshold:
4. layer, pruning rate ← *l*, $P$
5. **else**:
6. layer, pruning rate ←not found
7. **end**
8. **end**
9. Layers' order and pruning rate according to the highest accuracy

**Quantization:** In CNNs, fully connected (FC) layers have many parameters. Therefore, parameter reduction of FC layers is becoming essential if maximized compression is required. If the pruning scheduling algorithm, used in convolutional layers, is applied with pruning neurons in FC layers, it is possible to lose the accuracy. An alternative is using quantization to reduce the number of bits for representing parameters.

In CNNs, weights can be represented with 32-bit floating point numbers, while the most common form of quantization is into 8-bit integers. Also, hardware support for INT8 computations is 2 to 4 times faster than FP32 compute [18]. We are applying a dynamic quantization with the Pytorch [19] library that converts only weights to INT8. It is a post-training quantization method since the model uses FP32 for training before quantizing it. By default, this method can apply to layers like the fully connected layers of a CNN model. There is a slight loss of accuracy in testing, but it is not comparable when it is s necessary to retrain the network with pruning.

## V. EXPERIMENTS AND RESULTS

### A. Experiments and Analysis for A$^2$E

**Applications and Dataset:** The applications for autoencoders explored in this paper involve anomaly detection that translates to detecting fraudulent credit card transactions and recognizing handwritten digits. The datasets are 'Credit Card Fraud Detection' from Kaggle and 'MNIST' version 3.0.1 from Tensorflow/Keras, respectively. For the Credit Card Dataset: Models were trained to reconstruct the 29 features of only the clean dataset. The dataset was reduced to 29 unlabeled fields using PCA [18]. For testing to classify either fraud or clean, the mean absolute deviation (MAD) score was used [20]. Since the model has trained on clean data, it should have trouble reconstructing the fraud data that provides a noticeably higher loss. It is an unsupervised approach. Table I shows the information for the fraud dataset.

Table II shows the information for the MNIST dataset, where MNIST image classification used a standard approach.

We fixed the output layer of the model to ten nodes - one for each classification category.

*Retrieval Number: 100.1/ijsce.E365014051124*
*DOI: 10.35940/ijsce.E3650.14051124*
*Journal Website: www.ijsce.org*

The softmax was used with argmax to determine the classification categories.

**Table 1: Characteristics of Credit Card Fraud Database**

| Characteristics | Count |
|---|---|
| Total transactions | 84,807 |
| Total fraud transactions | 492 |
| Number of features | 29 |

**Table 2: Characteristics of MNIST Database**

| Characteristics | Count |
|---|---|
| Train images | 60,000 |
| Test images | 10,000 |
| Number of features | 784 |

**A$^2$E architectures:** As mentioned in Section IV, the A$^2$E architecture for credit card fraud detection is [4-2-6-2-6-2-4]. It has the same number of layers as the baseline architecture. We updated the number of nodes at Layer 1-7 pair, Layer 2-6 pair, and Layer 3-5 pair while the latent space remained the same. The number of nodes is small enough, and based on the sensitivity analysis, we could not see any big performance difference. For the MNIST classification, we use a 5-layer autoencoder [128-64-32-64-128] as the baseline autoencoder architecture [8]. The network needs a relatively large number of nodes in each layer to perform well due to the input size of 784 (28x28). Based on the sensitivity analysis of performance vs. the number of nodes in each layer, all performance numbers are similar, and it is hard to find an optimal architecture. Therefore, we perform the sensitivity analysis by directly changing the number of nodes and making an A$^2$E architecture by switching the number of nodes in Layers 1-2 and in Layers 4-5. Table III shows the performance comparison of AE and A$^2$E. We observe that the complexity is reduced by 43%~49% while the accuracy is dropped by 0.3%~5.0%, as shown in Table III. Considering a 4.9% accuracy drop compared to the performance of the baseline AE, the effectively performing architecture for MNIST classification is [8-16-4-16-8] structure providing a 94.5% reduction in complexity. We exclude the last case with [4-8-2-8-4] because it shows an accuracy drop [21].

**Accuracy and network size comparison:** We apply the accordion autoencoder, which has multiple compressions and decompressions for annealing the features to reduce the reconstruction error, to two applications mentioned in the previous subsection and measures the performance (accuracy). Figure 8 compares the accuracy and complexity of AE and A$^2$E on credit card fraud detection and MNIST classification. Our results show 13.6% and 4.9% performance drops, respectively, but we observe a reduction in network complexity, 85.1% and 94.5% for each application.

**Table 3: Node Sensitivity Analysis: AE vs. A2E**

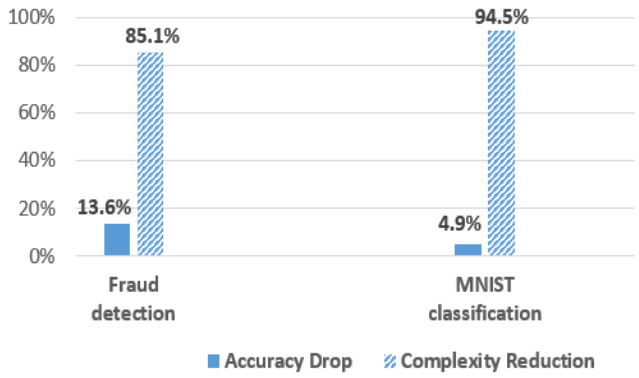
| Performance Metrics / Autoencoder vs. Accordion AE | Loss | Accuracy | Precision | Recall | F1-score | Complexity |
|---|---|---|---|---|---|---|
| 128-64-32-64-128 | 0.09 | 0.97 | 0.97 | 0.97 | 0.97 | 122,538 |
| 64-128-32-128-64 | 0.09 | 0.97 | 0.97 | 0.97 | 0.97 | 75,818 |
| 64-32-16-32-64 | 0.10 | 0.97 | 0.96 | 0.96 | 0.96 | 56,154 |
| 32-64-16-64-32 | 1.12 | 0.96 | 0.96 | 0.96 | 0.96 | 31,770 |
| 16-8-4-8-16 | 0.22 | 0.94 | 0.94 | 0.94 | 0.94 | 13,086 |
| 8-16-4-16-8 | 0.26 | 0.92 | 0.92 | 0.92 | 0.92 | 6,798 |
| 8-4-2-4-8 | 0.46 | 0.88 | 0.89 | 0.88 | 0.88 | 6,468 |
| 4-8-2-8-4 | 0.59 | 0.84 | 0.84 | 0.84 | 0.84 | 3,308 |

**Implications:** The latent space in the autoencoders has condensed information from the input through multiple dimension reduction layers, and the latent space information is becoming more meaningful through the iterative decoding and encoding processes in deep autoencoders. The latent space can be a feature space. The proposed A$^2$E architecture has multiple latent spaces through repeated decoding and encoding processes. Information propagates from feature spaces into the A$^2$E network, which will help reduce the reconstruction loss even with lower complexity and trainable parameters.

## B. Experiments and Analysis for DNNs

Two different models, the AlexNet and the VGG16, are used for layer characterizations with CIFAR-10 and CIFAR-100.

**AlexNet with Cifar-10 and Cifar-100:** The threshold factor used in the pruning schedule algorithm for both datasets is 0.985. As expected, conv4 remains unpruned because no accuracy value is above the threshold. The model will be retrained for 20 epochs while pruning every layer. Table IV shows the algorithm results obtained for AlexNet using CIFAR-10 and CIFAR-100. Table V shows the accuracy and compression results of the proposed method. We see that the final accuracy of the pruned model using our method is higher than using a sequential order. Results for testing the accuracy of the final model are the mean of three different runs, so it could be certified that there was a higher accuracy tendency when changing the order. Due to the filter pruning, the model performance can be reduced in floating point operations per second (FLOPs). Also, by applying quantization in fully connected layers, the reduction in parameter size goes up to 47.28x compared to the original model [22][23][24][25][26][27].




**[Fig.8: Performance Comparison: AE vs. A$^2$E]**

**Table 4: Proposed Layer Order and Pruning Rate**

| Model | Dataset | Order of Conv.Layers | Pruning rate |
|---|---|---|---|
| AlexNet | CIFAR-10 | [3,1,0,2] | [0.4,0.1,0.1,0.4] |
| | CIFAR-100 | [3,2,1,0] | [0.2,0.3,0.3,0.1] |


Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

**Table 5: Accuracy and Compression Results of the Accuracy-Aware Method**

| Baseline | | | | |
|---|---|---|---|---|
| **Model** | **Dataset** | **Accuracy (%)** | **PS (MB)** | **FLOPs (E6)** |
| AlexNet | CIFAR-10 | 92.19 | 61.10 | 710.63 |
| | CIFAR-100 | 72.86 | | |

| Accuracy-Aware Pruning Method | | | | | |
|---|---|---|---|---|---|
| **Original Model** | **Dataset** | **Order Conv** | **Acc. (%) (Accuracy drop)** | **PS (MB)** | **FLOPs (E6)** |
| AlexNet | CIFAR-10 | O-O-O | 91.85 (▼ 0.34%) | 4.93 (PC: 47.28x) | 414.94 (FC: 1.72x) |
| | | Seq. | 91.04 (▼ 1.15%) | | |
| | CIFAR-100 | O-O-O | 72.50 (▼ 0.36%) | 5.71 (PC: 40.82x) | 419.39 (FC: 1.69x) |
| | | Seq. | 72.08 (▼ 0.78%) | | |

** PS: Parameter Size; PC: Parameter size Compression; FC: FLOPs compression; OOO: Out-of-Order*

**VGG16 with CIFAR-10 and CIFAR-100:** We also apply our method to deeper networks to analyze its performance in more complex architectures. In this case, the threshold factor is 0.99 for CIFAR-10 and 0.997 for CIFAR-100. As observed in the case of AlexNet, pruning will be applied to the last layer because it would damage the performance of the final model. The model has tuned for five epochs after pruning each layer. Table VI presents the results of the pruning schedule algorithm, and Table VII shows the accuracy and compression of the pruned model.

**Table 6: Proposed Layer Order and Pruning Rate**

| Model | Dataset | Order of Conv. Layers | Pruning Rate |
|---|---|---|---|
| VGG16 | CIFAR-10 | [0, 1, 9, 11, 6, 4, 5, 8, 2, 3, 7, 10] | [0.8, 0.5, 0.4, 0.6, 0.5, 0.8, 0.6, 0.5, 0.8, 0.5, 0.5, 0.5] |
| | CIFAR-100 | [6, 1, 2, 0, 4, 8, 9, 11, 7, 3, 5, 10] | [0.3, 0.5, 0.7, 0.8, 0.4, 0.3, 0.1, 0.3, 0.5, 0.7, 0.5, 0.4] |

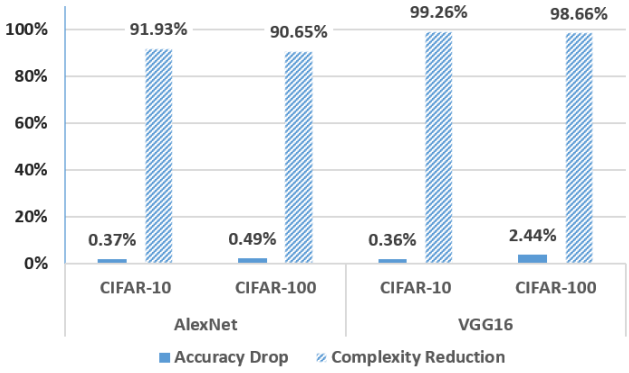
**Table 7: Accuracy and Compression Results of the Accuracy-Aware Method**

| Baseline | | | | |
|---|---|---|---|---|
| **Model** | **Dataset** | **Accuracy (%)** | **PS (MB)** | **FLOPs (E6)** |
| VGG16 | CIFAR-10 | 92.81 | 527.79 | 15.48 |
| | CIFAR-100 | 74.09 | | |

| Accuracy-Aware Pruning Method | | | | | |
|---|---|---|---|---|---|
| **Original Model** | **Dataset** | **Order Conv** | **Acc. (%) (Accuracy drop)** | **PS (MB)** | **FLOPs (E6)** |
| VGG16 | CIFAR-10 | O-O-O | 92.48 (▼ 0.33%) | 3.93 (PC: 35.21 x) | 2.74 (FC: 5.66x) |
| | | Seq. | 91.15 (▼ 1.66%) | | |
| | CIFAR-100 | O-O-O | 72.80 (▼ 1.29%) | 7.09 (PC: 19.50 x) | 4.76 (FC: 3.25x) |
| | | Seq. | 72.28 (▼ 1.81%) | | |

** PS: Parameter Size; PC: Parameter size Compression; FC: FLOPs compression; OOO: Out-of-Order*

**VGG16 with Cifar-10 and Cifar-100:** The graph in Figure 9 represents the training accuracy of the model after pruning one layer at a time and training the model for five epochs. When applying our method, the last layer provides a slightly higher accuracy than the final layers in the existing sequential order. These performance changes can explain the differences in the final accuracy of the pruned model.




**[Fig.9: Pruning Performance Comparison: Original Order vs. Out-of-Order]**

## VI. CONCLUSION

Deep neural networks (DNNs) have several technical issues as the network size gets more complex. Those issues include computational complexity, redundancy, and parameter size. Many parameters require high memory capacity. That might cause migration problems to embedded devices. Many pruning techniques are applied to reduce the network size in deep neural networks, but various issues still exist when DNNs apply the pruning techniques.

Several applications use autoencoders among neural networks, including dimensionality reduction, data reconstruction, anomaly detection, and classification problems. On the other hand, autoencoders are not used in practical data compression problems because modern algorithms perform better without the required training. It would also be impractical to gather a dataset for each compression application because the autoencoder provides performance less on transfer learning. Also, network size is a disadvantage of autoencoders. The architecture of the autoencoders has a double workload due to the encoding and decoding processes. In this research, we choose autoencoders and two deep neural networks – AlexNet and VGG16. We perform the sensitivity analysis to explore the performance variations for the network architecture and network complexity through an out-of-order layer pruning mechanism.

Accordion AutoEncoder ($A^2E$) provides good performance because it has multiple compressions and decompressions for annealing the features to reduce the reconstruction error and for credit card fraud detection and MNIST classification and measure the performance (accuracy). Our results show 4.9% and 13.6% performance drops, respectively, but we observe a reduction in network complexity, 85.1% and 94.5% for each application.

*Retrieval Number: 100.1/ijsce.E365014051124*
*DOI: 10.35940/ijsce.E3650.14051124*
*Journal Website: www.ijsce.org*





*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)*

We extend the out-of-order layer pruning to deeper learning networks. Various research proposes methodologies to reduce the complexity. In our approach, we propose a simple yet efficient scheme, accuracy-aware structured filter pruning based on the characterization of each convolutional layer combined with the quantization of fully connected layers. We investigate the accuracy and compression rate of each layer using a fixed pruning ratio and reorder the pruning priority depending on the accuracy of each layer. Our analysis of layer characterization shows that the pruning order of the layers does affect the final accuracy of the deep neural network. Based on our experiments using the proposed pruning scheme, the parameter size can be reduced up to 47.28 times in AlexNet compared to the original model. Similar results are also obtained with VGG16, achieving a maximum compression rate of 35.21x.

For the future work, we will optimize pruning and compression rates while maintaining overall performance. Also, it will be meaningful work to apply our method to much deeper architectures.

## DECLARATION STATEMENT

After aggregating input from all authors, I must verify the accuracy of the following information as the article's author.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been sponsored or funded by any organization or agency. The independence of this research is a crucial factor in affirming its impartiality, as it has been conducted without any external sway.
- **Ethical Approval and Consent to Participate:** The data provided in this article is exempt from the requirement for ethical approval or participant consent.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Authors Contributions:** The authorship of this article is contributed equally to all participating individuals.

## REFERENCES

1. D. Bank, N. Koenigstein, R. Giryes, "Autoencoders," Book Chapter in Machine Learning for Data Science Handbook, 2023. https://doi.org/10.1007/978-3-031-24628-9_16
2. S. Lin, R. Ji, Y. Li, C. Deng, X. Li "Towards Compact ConvNets via Structure-Sparsity Regularized Filter Pruning," IEEE Transactions on Neural Networks and Learning Systems, 2020. https://doi.org/10.1109/tnnls.2019.2906563
3. S. S. Tirumala, "A Novel Weights of Weights Approach for Efficient Transfer Learning in Artificial Neural Networks," Procedia Computer Science, 2022. https://doi.org/10.1016/j.procs.2022.11.013
4. J. Luo, J. Wu, W. Lin, "ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression," IEEE International Conference on Computer Vision, 2017. https://doi.org/10.1109/iccv.2017.541
5. J. Guo, M. Potkonjak, "Pruning ConvNets Online for Efficient Specialist Models," IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017. https://doi.org/10.1109/cvprw.2017.58
6. P. Hu, X. Peng, H. Zhu, M. Aly, J. Lin, "OPQ: Compressing Deep Neural Networks with One-shot Pruning-Quantization," Proceedings of the AAAI Conference on Artificial Intelligence, 2021. https://doi.org/10.1609/aaai.v35i9.16950
7. G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," SCIENCE Vol. 313, Issue 5786, pp. 504-507, 2006. https://doi.org/10.1126/science.1127647
8. M. Kon and L. Plaskota, "Information Complexity of Neural Networks," Neural Network, 2000. https://doi.org/10.1016/s0893-6080(00)00015-0
9. Neloy, M. Turgeon, "A Comprehensive Study of Auto-Encoders for Anomaly Detection: Efficiency and Trade-Offs," SSRN, 2024. http://dx.doi.org/10.2139/ssrn.4757204.
10. S. Bhatia, R. Bajaj, S. Hazari, "Analysis of Credit Card Fraud Detection Techniques," International Journal of Science and Research (IJSR), 2006. https://doi.org/10.21275/v5i3.nov162099.
11. J. Romero, J. P. Olson, and A. Aspuru-Guzik, "Quantum autoencoders for efficient compressions of quantum data," arXiv:1612.02806, 2017. https://doi.org/10.1088/2058-9565/aa8072
12. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Proceedings of the 25th International Conference on Neural Information Processing Systems, 2017. https://doi.org/10.1145/3065386
13. J. Kim, J. Lee, K. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. https://doi.org/10.1109/cvpr.2016.182
14. Jiang, G, Goldsztein, "Convolutional Neural Network Approach to Classifying the CIFAR-10 Dataset," Journal of Student Research, 2023. https://doi.org/10.47611/jsrhs.v12i2.4388.
15. Y. Zheng, H. Huang, J. Chen, "Comparative analysis of various models for image classification on Cifar-100 dataset," Journal of Physics, 2024. https://doi.org/10.1088/1742-6596/2711/1/012015.
16. N. Ketkar, J. Moolayil, "Introduction to PyTorch," Book Chapter in Deep Learning with Python, 2021. https://doi.org/10.1007/978-1-4842-5364-9_2
17. F. Nielsen, "Linking ImageNet WordNet Synsets with Wikidata," WWW '18: Companion Proceedings of the The Web Conference, 2018. https://doi.org/10.1145/3184558.3191645.
18. J. Naylor, K. P. Li, "Analysis of a neural network algorithm for vector quantization of speech parameters," ITT Defense Communications Division, 1988. https://doi.org/10.1016/0893-6080(88)90341-3.
19. Li, D. Chen, L. Shi, B. Jin, J. Goh, and S. Ng, "MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks," arXiv:1901.04997, 2019. https://doi.org/10.1007/978-3-030-30490-4_56
20. E. Bisong, "Autoencoders," Book Chapter in Building Machine Learning and Deep Learning Models on Google Cloud Platform, 2019. https://doi.org/10.1007/978-1-4842-4470-8_37.
21. K. Kelso and B. Lee, "Accordion Autoencoders (A2E) for Generative Classification with Low Complexity Network," International Symposium on Computational Intelligence, December 2021. https://doi.org/10.1109/csci54926.2021.00152
22. M. Carballo and B. Lee, "Accuracy-aware Structured Filter Pruning for Deep Neural Networks," International Symposium on Artificial Intelligence, December 2020. https://doi.org/10.1109/csci51800.2020.00122
23. K. Sai krishna, G. Sreenivasa Raju, P. Praveen Kumar, A Deep Neural Network for face Recognition. (2019). In International Journal of Innovative Technology and Exploring Engineering (Vol. 8, Issue 12S, pp. 420–423). https://doi.org/10.35940/ijitee.l1105.10812s19
24. Young, L., York, J. R., & Kil Lee, B. (2023). Implications of Deep Compression with Complex Neural Networks. In International Journal of Soft Computing and Engineering (Vol. 13, Issue 3, pp. 1–6). https://doi.org/10.35940/ijsce.c3613.0713323
25. Das, S., S, S., M, A., & Jayaram, S. (2021). Deep Learning Convolutional Neural Network for Defect Identification and Classification in Woven Fabric. In Indian Journal of Artificial Intelligence and Neural Networking (Vol. 1, Issue 2, pp. 9–13). https://doi.org/10.54105/ijainn.b1011.041221
26. Chellatamilan, T., Valarmathi, B., & Santhi, K. (2020). Research Trends on Deep Transformation Neural Models for Text Analysis in NLP Applications. In International Journal of Recent Technology and Engineering (IJRTE) (Vol. 9, Issue 2, pp. 750–758). https://doi.org/10.35940/ijrte.b3838.079220
27. Magapu, H., Krishna Sai, M. R., & Goteti, B. (2024). Human Deep Neural Networks with Artificial Intelligence and Mathematical Formulas. In International Journal of Emerging Science and Engineering (Vol. 12, Issue 4, pp. 1–2). https://doi.org/10.35940/ijese.c9803.12040324

## AUTHORS PROFILE

**Shafayat Mowla Anik** is a Ph.D. student in Electrical and Computer Engineering at the University of Colorado Colorado Springs. He received his bachelor's and master's degrees from the University of Dhaka, Bangladesh in 2015 and 2017 respectively. He worked at LiCA (Laboratory for Intelligent Computer Architecture) as a Graduate Research Assistant (GRA). Currently, he is a Graduate Teaching Assistant (GTA) for Electrical and Computer Engineering. His teaching classes include Circuit and System, Introduction to Signals and Systems Lab, Advanced Digital Design Methodology, and VLSI-1. His research interests include cache replacement, deep learning, computer architecture, malware detection, cybersecurity, low power, logic optimization, embedded processor design, and performance modeling.

**Kevyn Kelso** is a master's student in Electrical and Computer Engineering at the University of Colorado Colorado Springs. He received his bachelor's degree in Electrical and Computer Engineering (Computer Engineering major) from the University of Colorado Colorado Springs in 2021. He worked for EM Microelectronic as an intern during his senior year. After graduation, he joined EM Microelectronic as a full-time software engineer. He worked at LiCA (Laboratory for Intelligent Computer Architecture) as a URA (Undergraduate Research Assistant) during his undergraduate career. His research interests include computer architecture, logic optimization, embedded processor design, malware detection, GPU computations, neural network optimization and pruning, deep learning, and generative images.

**Byeong Kil Lee** received a Ph.D. degree in computer engineering from The University of Texas, Austin, TX, in 2005. He is an Associate Professor with the Department of Electrical and Computer Engineering at the University of Colorado at Colorado Springs (UCCS). Dr. Lee worked at Samsung as a Vice President for five years before joining UCCS. Also, he was an Assistant Professor at The University of Texas at San Antonio (UTSA) and a Senior Design Engineer at Texas Instruments (TI) for five years and four years, respectively. He was also a senior research staff at the Agency for Defense Development (ADD), Korea, for ten years. His research interests include computer architecture, workload characterization of emerging applications, deep learning, application-specific processors, low-power mobile processors, and cybersecurity. He serves on the program and organizing committee for conferences and workshops such as ISCA, ISPASS, IISWC, ICCD, HPCA, ASAP, PACT, ICPP, and UCAS.