Selecting Update Blocks of Convolutional Neural Networks using Genetic Algorithm in Transfer Learning

Md. Mehedi Hasan[1], Muhammad Ibrahim[2*], Md. Sawkat Ali[1]

[1]*Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh*

[2]*Department of Computer Science and Engineering, University of Dhaka, Dhaka, Bangladesh*

**Corresponding author: Muhammad Ibrahim ibrahim313@du.ac.bd*

Abstract The performance of convolutional neural networks (CNN) for computer vision problems depends heavily on their architectures. Transfer learning performance of a CNN strongly relies on selection of its trainable layers. Selecting the most effective update layers for a certain target dataset often requires expert knowledge on CNN architecture which many practitioners do not possess. General users prefer to use an available architecture (e.g. GoogleNet, ResNet, EfficientNet etc.) that is developed by domain experts. With the ever-growing number of layers, it is increasingly becoming difficult and cumbersome to handpick the update layers. Therefore, in this paper we explore the application of a genetic algorithm to mitigate this problem. The convolutional layers of popular pre-trained networks are often grouped into modules that constitute their building blocks. We devise a genetic algorithm to select blocks of layers for updating the parameters. By experimenting with EfficientNetB0 pre-trained on ImageNet and using three popular image datasets – namely Food-101, CIFAR-100 and MangoLeafBD – as target datasets, we show that our algorithm yields similar or better results than the baseline in terms of accuracy, and requires lower training and evaluation time due to learning a smaller number of parameters. We also devise a measure called block importance to measure each block's efficacy as an update block and analyze the importance of the blocks selected by our algorithm.

Keywords: computer vision; transfer learning; convolutional neural network; EfficientNet; genetic algorithm.

1. Introduction

High-performance image recognition models are often developed using Convolutional Neural Networks (CNN). As the prevalent approach of deep learning in image classification, CNNs have shown exceptional supremacy over many approaches in various real-world machine-learning applications, especially in the broad area of computer vision [18]. The performance of a CNN depend heavily on its architecture [28], [27], and hence, all of the state-of-the-art CNNs, such as GoogleNet [30], ResNet [13], DenseNet [14] etc., are handcrafted by experts who have rich domain knowledge. As it is not always feasible for general practitioners of CNN to acquire such expertise, these users often opt to use a pre-designed architecture that suits their need. CNNs are usually designed with a fixed computational resource budget, and then scaled up at a later time for better accuracy as more computational resources become available.

The training process of CNNs requires large sized datasets because these models need to learn a huge number of parameters. Since the parameter space is colossal, sufficient quantity of training data are warranted to learn complex patterns. This requirement of having large datasets, however, can be relaxed using the transfer learning setting [25] where practitioners reuse existing pre-trained networks, thereby reducing the training time significantly. In transfer learning, during training phase, the parameters of some layers of the pre-trained network are kept fixed while updating the rest with the (target) dataset at hand. Generally, early layers, i.e., the layers near the input, of a CNN detect low-dimensional information like the color and edges of an image, and the later layers, i.e., the layers near the output, extract high-dimensional features that help to identify the ground truth labels [37]. Therefore, for transfer learning, usually the early layers of the pre-trained model are kept frozen while the parameters of the later layers are updated.

Some recent empirical studies, however, report good results by applying the opposite practice, i.e., keeping the parameters of the later layers fixed instead of that of earlier ones. To mention a few such works: Zunair et al [38] use a VGG16 network [28] pre-trained with ImageNet for the prediction task of Bangla characters and report that the best accuracy is found when the first input layer and early fully-connected layers are selected as update layers. Gafoorian et al [10] apply transfer learning on MRI images and report that the best performance is achieved when parameters of only six input layers are updated. Therefore, it is intriguing to investigate into the appropriate layers to be updated for transfer learning, thereby triggering this research.

Nowadays, common CNN architectures contain a lot of layers. For example, VGG16, InceptionV3, and GoogleNet have 16, 94, and 22 layers, respectively. EfficientNetB0 and EfifcientNetB7 [32] have 237 and 813 layers, respectively. When the general users want to use a pre-trained model, such a large number of layers makes it hard for them to manually select the appropriate layers to be updated. Manually selecting the layers to be updated involves a trial-and-error approach which is time consuming and tiresome.

To mitigate this difficulty, there are studies that apply the so-called metaheuristic optimization algorithms, such as genetic algorithm, to automatically select the effective layers to be updated (we discuss these works in detail in Section 2). However, we have not found any work that investigates the problem of selecting the appropriate blocks – not individual layers – to be updated in a transfer learning setting using genetic algorithms. Our investigation is dedicated to this endeavor.

In this paper, we develop a genetic algorithm-based [11,22] method to automatically select blocks of layers – instead of individual layers – to update the parameters. This technique is expected to significantly minimize the training time of CNNs while maintaining similar accuracy. We also adapt a recently proposed metric named OTDD [2] to calculate the importance of the blocks of layers. Using this metric we calculate the contribution of each block in identifying the features of the data. In all these investigations we use three target datasets, namely, Food-101, CIFAR-100, and MangoLeafBD. As for the CNN, we use EfficientNet [32] models pre-trained on ImageNet [8].

The rest of the paper is organized as follows. Section 2 discusses the relevant existing works. Section 3 presents the background knowledge required to understand the paper. Section 4 presents the framework and methodology of the investigation. Section 5 demonstrates the experimental results. Finally, Section 6 concludes the paper.

2. Related Work

Deep learning and transfer learning are well-known for their effectiveness in image classification task [23]. It is well-known that the structure and performance of a CNN rely heavily on its hyper-parameters. Hyper-parameters are often manually chosen by experts to obtain a model with expected performance. However, different datasets may require different model structure, and hence, choosing them by trial and error can be tedious. AszemiDomic [4] discuss the usefulness of different search and optimization methods to find a suitable set of hyper-parameters. The authors also develop a hybrid optimization method combining a genetic algorithm with local search that can be used to optimize CNN structure.

To improve CNN performance, some studies have explored the possibility of training multiple CNNs at the same time using different means to promote cooperation and specialization among them. Such sets of CNNs are called committees. Bochinski et al. [5] propose a way of defining the CNN structure in terms of its hyper-parameters and a framework to automatically find out the best set of hyper-parameters using an evolutionary optimization algorithm. Additionally, they extend their framework to optimize a CNN committee. The goal of this study is to establish a framework to optimize CNN committees for better performance.

Yanan and Bing et al. [29] are inspired by the success of ResNet and DenseNet and propose a genetic algorithm-based evolutionary approach of automatically designing CNNs using blocks from ResNet and DenseNet.The proposed algorithm is self-sufficiently automatic and does not expect any specific domain expertise from the user.

XieYuille et al. [35] venture to find a way to automatically build effective CNN structures for a given dataset. As the number of possible network structures increase exponentially with the number of layers in the network, the authors employ a genetic algorithm to navigate through the expanding search space. They propose a fixed length encoding strategy which represents a network architecture in the population and a genetic algorithm that operates on this population to produce better generations. Experiments with the CIFAR10 and ILSVRC2012 datasets show that their method produces CNNs with competitive or better recognition accuracy.

Lee et al. [19] propose a genetic algorithm that considers CNN structure and its hyper-parameters both in the optimization space and produces a CNN with optimal architecture and hyper-parameter values for the given dataset. The performance of the algorithm is evaluated with 18F-Florbetaben Amyloid PET/CT images for classification of Alzheimer's disease.

Loussaie et al. [20] observe that hyper-parameters of a network such as network depth, number of filters, and their sizes dramatically affect the performance of a CNN. They propose a genetic algorithm that finds the optimal values of those parameters for a given dataset, and thus produces an optimal CNN architecture.

Tian and Chen [33] develop a new genetic algorithm to find out the best suited pre-trained model for different datasets. They come up with a new genetic encoding model that represents different pre-trained CNN models in the population and an evolutionary approach to promote the best performing models in each generation. Experimental results have evidently shown that their approach outperforms some of the existing classification methods.

Cai and Luo [7] propose neural architecture search which is a resource-heavy search mechanism that automatically searches for CNN architectures. The authors devise an evolutionary framework that employs a multi-task, multi-objective search approach to find optimally balanced CNN architecture for a given task.

Finally, as discussed earlier, Nagae et al. [24] propose a method to automatically select effective layers using a genetic algorithm for InceptionV3 network. The authors also coin a term called Optimal Transport Dataset Distance (OTDD) to quantitatively evaluate a particular layer's efficacy as an update layer. They use OTDD to estimate a layer's importance as an update layer and compare a layer's contribution to accuracy with its OTDD score to reveal that layer OTDD score is indicative of a layer's capability of detecting features from the target dataset.

From the above discussion we see that although there are several works that utilize genetic algorithm to select the best setting of hyper-parameters and network architectures, to the best of our knowledge, there is no existing work that employs a genetic algorithm to select the best blocks to be updated in a transfer learning setting. Our investigation presented in this paper has filled this gap in the literature.

3. Background Study

As mentioned earlier, in this investigation we use EfficientNet [32] models pre-trained on ImageNet [8], though our developed framework is equally applicable to other types of CNNs. In this section we briefly discuss the architecture of EfficientNet and PathNet. We also discuss a metric called Optimal Transport Dataset Distance (OTDD).

3.1. EfficientNet

CNNs are often scaled up to achieve better performance. For example, ResNet [13] can be scaled up from ResNet-18 to ResNet-200 by incorporating more layers. GPipe's

ImageNet [15] top-1 accuracy is improved to 84.3% by scaling up the baseline model by 4 times. Scaling CNNs up by their depth [13] or width [36] are the most commonly performed by the practitioners, but scaling up the models by image resolution [15] is also a popular method. Only one of the three dimensions – depth, width, and image size – is usually scaled. Although two or three dimensions can be arbitrarily scaled, it involves tedious manual tuning, and yet often fails to provide better accuracy and efficiency. To resolve this, EfficientNet [32] proposes a simple yet effective compound scaling method that uses a constant scaling ratio to scale all three dimensions of network in a controlled and balanced way. The intuition is: as the input image size increases, it makes sense that the model will need more layers and more channels to extract even finer details from larger images. In fact, previous studies [21, 26, 36] have shown that there is a certain correlation between network width and depth. The compound scaling method uniformly scales network width, depth, and resolution with a set of constant scaling coefficients. In particular, if 2^N times more computational resources become available, then this method simply scales up the network depth by α^N, width by β^N, and image size by γ^N, where α, β, γ are constant coefficients chosen through a small grid search on the baseline network.

3.2. PathNet and StepwisePathNet

When performing transfer learning to the target datasets, each layer of a pre-trained CNN detects features that are common among the source and target datasets [37]. Therefore, it is imperative to efficiently select layers that are effective feature detectors for the target datasets and then update their parameters while keeping the other layers frozen. Additionally, as the network architectures have become more complex due to the increased availability of computational resources, an efficient way of selecting effective layers without manual labor is required. The StepwisePathNet method [16] tries to address this specific need. This algorithm expands DeepMind's PathNet [9] to select the update layers in a straight-chain network. StepwisePathNet algorithm labels each layer as either fixed or updated, and the selection is optimized by a genetic algorithm that employs a tournament selection mechanism.

3.3. Improvement on StepwisePathNet

Citing the limitations of StepwisePathNet, Nagae et al [24] improves the algorithm by applying another genetic algorithm. The authors work with InceptionV3 architecture [31] and apply a genetic algorithm to automatically select the effective layers of the network to be updated during learning for the target dataset.

3.4. Optimal Transport Dataset Distance (OTDD)

Evaluation of the distance between two labeled datasets has been explored in studies utilizing the optimal transport distance [2], which provides a way to quantify the difference between two datasets and correlate dataset distance with transfer learning efficacy. Optimal transport deals with the issue of transferring material from one place to another at minimum cost, and can also be used to mathematically compare two different probability distributions [34].

Optimal Transport Dataset Distance (OTDD) [2] is a measure to compute the distance between two different labeled datasets. Using this metric, in [24], the importance of a layer is calculated by estimating its effectiveness as an update layer for transfer learning and its potential for detecting common traits in both source and target datasets. A subset of feature maps generated at layer l for both source and target datasets are taken and denoted respectively as A_l^{source} and A_l^{target}. Then, the layer importance LI is expressed as:

$$\mathrm{LI}(l) = \frac{\mathrm{OTDD}(A_l^{\text{source}}, A_l^{\text{target}})}{\mathrm{OTDD}(A_l^{\text{source}}, A_l^{\text{source}'}) + \epsilon}\,, \tag{1}$$

were OTDD is calculated using Eq. (16) of [24] and ϵ is a small nonzero positive number. The denominator in Eq. (1) denotes the optimal transport distance between two different subsets, source and source′, of the source dataset, where the difference is created due to the difference in sampling. This ratio basically captures the difference between the features maps generated for the source and target datasets at a particular layer. Datasets with similar feature sets should result in similar feature maps at an effective layer, yielding a lower LI. A lower value of LI indicates higher adaptability of the model for the target dataset.

4. Proposed Framework and Methodology

It is well-known to the research community that the performance of CNNs is highly dependent on their architecture, and hence all high performing CNNs like GoogleNet, ResNet, DenseNet, EfficientNet, InceptionV3 etc. have been manually designed by experts who possess profound knowledge on CNNs. Unfortunately, such deeper understanding of CNNs and expertise in machine learning cannot be expected from all general practitioners of these models. Hence, general users often opt to use a pre-designed architecture that suits their need, thereby giving rise to the notion of transfer learning. In this setting, the user of a CNN does not need to train the model on a large dataset, and instead takes advantage of a pre-trained model which has already been trained on a large source dataset. The intuition behind this strategy is as follows. Some layers of CNNs extract low level information such as edge, color, shape etc. from input images, while other layers detect high level features like ground truth labels of the instances.

Since in image classification task, the edges and low-level shapes of images are needed to be extracted irrespective of the domain at hand, there is no benefit to re-extract these features, and so the parameters that contribute to extracting these low level features are no longer needed to be re-learnt across different domains/datasets. Therefore, in a transfer learning setting, it is imperative to decide which layers should be kept fixed (i.e., no new learning of parameters are needed) and which layers should be learnt/updated anew for the target dataset. The challenge, however, is, as the depth and complexity of CNNs are rapidly growing with the increasing availability of computational resources, it is increasingly becoming infeasible for general practitioners to handpick effective layers for update.

Many popular CNN architectures such as ResNet, MnasNet, GoogleNet, EfficientNet etc. are constituted of groups of convolutional layers which are called blocks. Each group of layer or block helps the model identify a low or high level characterizing feature. From this intuition, we pose the research question: *Can we automatically select appropriate blocks for update using a genetic algorithm that would yield at least similar accuracy to the baseline model?* The effect will be lesser blocks in the network (i.e., reduced number of parameters) to be learnt for transfer learning, thereby reducing the execution time.

4.1. Automatic Block Selection for Update by Genetic Algorithm

In our proposed scheme, a genetic algorithm is devised to select the blocks to be updated so as to reduce the training time, and, at the same time, to yield good accuracy in prediction for the target datasets. A genetic algorithm is a metaheuristic search algorithm that selects a good-enough solution from the vast search space of potential solutions. It trades off between exploration and exploitation, which means, optimizing a potential solution while escaping the local minima. This algorithm, broadly, works as follows [11,22]. It begins its journey in the solution space with some potential solutions whose set is called *population*. It then selects two *parent* solutions from the solution pool based on some fitness function, and then applies two operations, namely crossover and mutation, to generate *children* solutions. This process is repeated until a good-enough solution is found. Use of the fitness function ensures exploitation of the search space, and use of randomization allows exploration of the search space.

In our algorithm, we maintain a binary array to denote the blocks of the network where each block is denoted by 0 or 1. A 1 means the block is selected for an update. Thus, for each solution or genotype g, if the ith gene g_i is equal to 1, then the corresponding ith block of the network, i.e., the feature extracted model, is selected for an update, which means, all of the layers of this block are selected as update layers. If the block is not selected, then its layers are frozen, i.e., the parameters of its layers are not updated. Initially, all the genes of the genotypes are set either 0 or 1 uniformly at random. Then, each genotype is evaluated against an evaluation function, which, in our case, is the classification accuracy. Then, the best two genotypes are chosen, and the crossover and mutation operations are performed in order to get offspring genotypes. This crossover and mutation operations enforce exploration in the search space. We iterate this process for 100 rounds (known as epochs) to obtain the final (best) model.

4.2. Block Accuracy and Block Importance

In order to understand the impact of each block on accuracy of the target dataset, we evaluate the obtained test accuracy of the fine-tuned model on target datasets when only a particular block is selected to be updated. For each block, only that particular block in the feature extracted model is selected to be updated and the rest are frozen. The model is then fine-tuned on the target dataset for up to 20 epochs, and the evaluation accuracy of the final model is recorded as the block accuracy denoted by BA.

Activation feature map datasets for all the source and target datasets are generated for each block of the pre-trained model, which are then used to calculate the Optimal Transport Dataset Distance (OTDD) between the pairs of source and target datasets.

The OTDD is calculated according to the method of [2] using their implementation [3].

Following the definition of layer importance (defined in Eq. (1)), we define block importance, BI for bth block as follows:

$$\mathrm{BI}(b) = \frac{\mathrm{OTDD}(A_b^{\text{source}}, A_b^{\text{target}})}{\mathrm{OTDD}(A_b^{\text{source}}, A_b^{\text{source}'}) + \epsilon}\,, \tag{2}$$

where source and source′ denote two different subsets of the source dataset (as explained below Eq. (1)). This ratio adeptly captures the difference between the feature maps generated for the source and target datasets at a particular block. Datasets with similar feature sets should result in similar feature maps at an effective block, yielding a lower BI. Lower value of BI indicates higher adaptability of the model for target dataset.

4.3. Model

We apply our proposed scheme on a popular pre-trained network for transfer learning called EfficientNetB0 which has 237 layers grouped in 8 blocks. For larger models of the EfficientNet family, similar results are expected since they are just scaled up versions of the base model and the basic building blocks remain the same. Moreover, since our method is generic, we expect similar result on other pre-trained networks besides EfficientNet.

As mentioned earlier, EfficientNetB0 is pre-trained on ImageNet dataset. As target datasets, we employ three datasets, namely CIFAR-100, Food-101, and MangoLeafBD (details of these datasets are discussed in the next section). In contrast to the existing practice of selecting individual layers (as done in [24]), our genetic algorithm select blocks of layers for update blocks from the feature-extracted model. The fully connected layers of the model are always selected as update layers.

Now we discuss the technical details of our algorithm. Regarding the values of hyper-parameters, we set up some empirical values as follows. We set the population size to 7. Elite plus roulette method is used as the selection method. The mutation probability is set to 1%.

The training phase is performed for 100 epochs with the Adam optimizer (with 0.0001 as the learning rate). We note here that while it is natural to try different combinations of values of these hyper-parameters, in this research our primary goal is to investigate the efficacy of a generic genetic algorithm. The best model obtained from the genetic algorithm is then trained on the target dataset. The training accuracy, validation accuracy, training time and the number of parameters are recorded as evaluation metrics.

5. Experimental Results

In this section we analyze the experimental results and discuss the findings. We measure the performance against five parameters: update layer/block selection time, training time, evaluation time, accuracy, and the number of parameters to be learnt.

5.1. Datasets and Experimental Settings

We use three target datasets: (1) CIFAR-100 [17], (2) Food-101 [6], (3) MangoLeafBD [1]. CIFAR-100 is an object recognition dataset with 100 classes each having 500 images for training and 100 images for testing. Food-101 is a food recognition dataset with 101 classes each of which has 750 training and 250 testing images. MangoLeafBD is a recently released dataset containing 4000 images of mango leaves with 8 classes of diseases. Figures 1, 2, and 3 show some sample images from the three datasets. All input images are shuffled, resized to 224×224 pixels, batched and pre-fetched for optimal data loading and training performance. All experiments are performed in a single NVIDIA GeForce RTX 2060 with a batch size of 32.

5.2. Baseline: Automatic Layer Selection by Genetic Algorithm

As the baseline method, we compare our method with the work of Nagae et al. [24]. The authors of this work use a genetic algorithm to select the appropriate update layers for the target datasets. So we think this work is an appropriate baseline for our proposed method.

Fig. 1. Sample images from CIFAR10 dataset.

Tab. 1. Performance comparison between the existing benchmark algorithm and our proposed algorithm on three datasets. LayerSelect – layer selection algorithm by [24]; BlockSelect – our proposed block selection scheme. Parameters: Runtime – time taken by the layer/block selection algorithm; Training time – neural network's training time after selecting layers/blocks; Evaluation time – testing time; Accuracy – percentage of correct predictions on test set; # parameters – number of parameters to be learnt for learning the target datasets.

Dataset →	Food-101		CIFAR-100		MangoLeafBD	
Parameter ↓	LayerSelect	BlockSelect	LayerSelect	BlockSelect	LayerSelect	BlockSelect
Runtime	3 h	43 min	1 h 34 min	39 min	12 min 19 s	2 min 42 s
Training time	31 min	21 min	28 min	19 min	19 min	18 min
Evalation time	42 ms	32 ms	31 ms	31 ms	58 ms	99 ms
Accuracy	0.77	0.79	0.81	0.82	1.0	0.997
# parameters	5, 79, 813	31, 13, 413	2, 56, 500	8, 30, 758	3, 61, 896	4, 24, 784

5.3. Performance Comparison

Table 1 presents the experimental results of Food-101, CIFAR-100, and MangoLeafBD target datasets. Here, LayerSelect indicates the layer selection algorithm of [24], and BlockSelect indicates our proposed algorithm. In the 1st column, runtime means the time taken by the layer/block selection algorithm, training time is the neural network's training time after selecting layers/blocks, evaluation time means the testing time, accuracy is the percentage of correct prediction on test set, and finally # parameters is the number of parameters to be learnt for learning the target datasets.

From the experimental results we see that our proposed block selection algorithm works much faster than the existing baseline, i.e., the layer selection algorithm [24]. Still we achieve slightly better results in two datasets and slightly worse result in the

Fig. 2. Sample images from Food-101 dataset.

other dataset. Overall, the block selection algorithm is found to maintain similar level of accuracy while reducing the training and evaluation time.

From the experimental results it is evident that the block selection algorithm is faster than the layer selection method and yields a model that has similar or better accuracy and inference time.

5.4. Block Importance and Block Accuracy

Table 2 presents the values of Block Importance (BI) and Block Accuracy (BA) metrics for the target datasets, and Fig. 4 illustrates the results graphically. BI is calculated

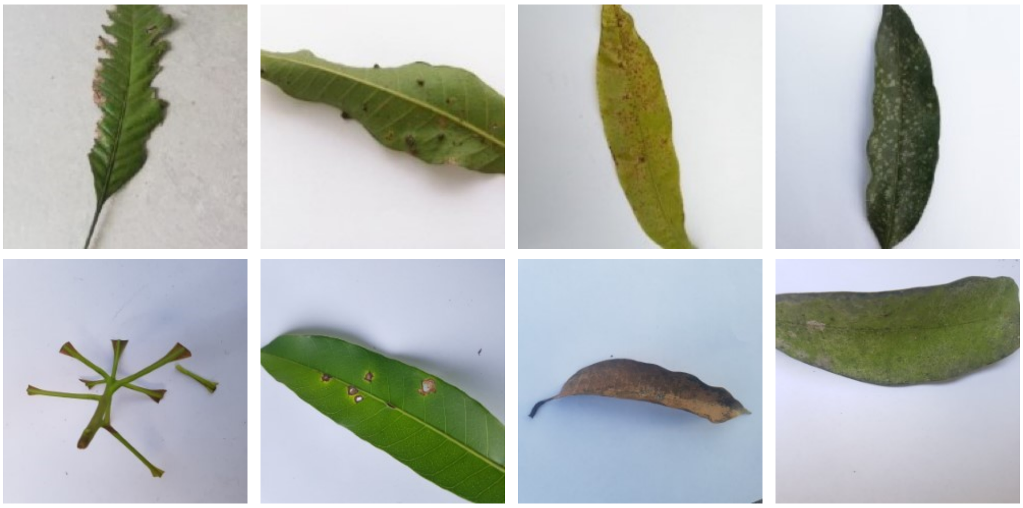


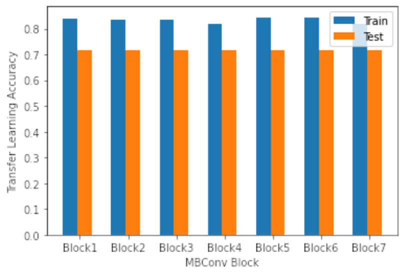
Fig. 3. Sample images from MangoLeafBD dataset.

Tab. 2. Block Importance (BI) and Block Accuracy (BA) of various blocks for all three datasets.

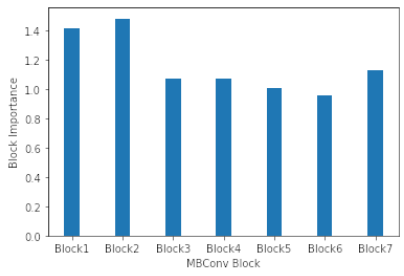
	Food-101			CIFAR-100			MangoLeafBD		
Block	BI	Train BA	Test BA	BI	Train BA	Test BA	BI	Train BA	Test BA
1	1.420	0.84	0.72	1.952	0.88	0.72	1.491	1	0.993
2	1.484	0.83	0.72	1.471	0.89	0.72	1.459	1	0.995
3	1.073	0.83	0.72	1.152	0.89	0.72	1.259	1	0.995
4	1.072	0.82	0.72	0.964	0.90	0.71	1.078	1	0.995
5	1.011	0.84	0.72	1.021	0.89	0.72	0.029	1	0.995
6	0.959	0.84	0.72	1.000	0.90	0.71	0.815	1	0.995
7	1.132	0.82	0.72	0.965	0.89	0.72	0.022	1	0.995

using Eq. (2). BA is the obtained from the training and test accuracy when transfer learning is performed considering only the update layers of that block.

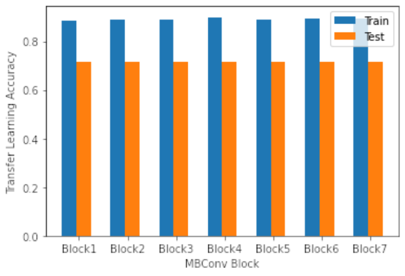
Conventionally, in transfer learning, it is believed that updating the layers close to the output side of the network is more effective. From our experimental data we cannot decisively claim this conjecture to be true, but it is evident that updating output side blocks works pretty well. We also see that updating blocks with lower BI values results in high training and testing accuracy. Though we cannot conclusively claim that updating the blocks with higher BI does not work, the trends shown by the experimental data evidently suggests that updating blocks with lower BI is effective for transfer learning.



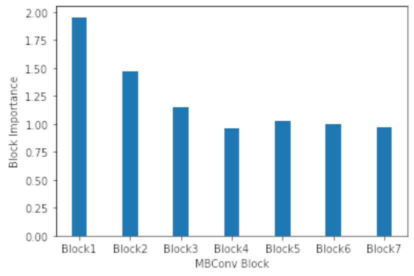
a BA



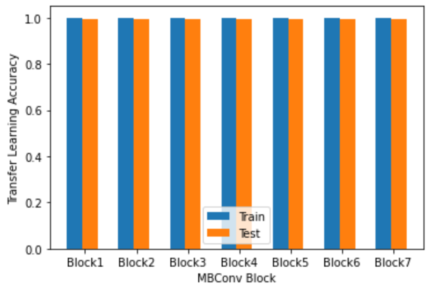
a BI



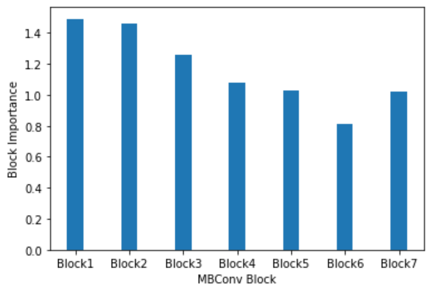
b BA



b BI



c BA



c BI

Fig. 4. Block Accuracy (**BA**) and Block Importance (**BI**) of various blocks of different datasets. (**a**) FOOD-100 dataset; (**b**) CIFAR-100 dataset; (**c**) MangoLeafBD dataset.

6. Conclusion

In order to assist practitioners of transfer learning select appropriate parameters of a convolutional neural network, in this research we have proposed a genetic algorithm-based solution that automatically selects CNN blocks, i.e., groups of layers, for the parameters to be updated only in these layers, for effective transfer learning. The proposed block selection algorithm selects blocks of the network instead of layers, which results in less computational time requirement and yet yields similar or slightly better accuracy over the baseline method. Currently the proposed block selection algorithm is only implemented for a popular CNN called EfficientNet. This algorithm may easily be extended to other types of CNNs. The genetic algorithm devised here can also be honed using sophisticated methods and hyper-parameter tuning. In addition, other metaheuristic algorithms (like simulated annealing [12]) can be investigated to better select the layers and blocks for update.

References

[1] S. I. Ahmed, M. Ibrahim, M. Nadim, M. M. Rahman, M. M. Shejunti, et al. MangoLeafBD: A comprehensive image dataset to classify diseased and healthy mango leaves. *Data in Brief*, 47:108941, 2023. doi:10.1016/j.dib.2023.108941.

[2] D. Alvarez-Melis and N. Fusi. Geometric dataset distances via optimal transport. In: *Proc. 33th Int. Conf. Neural Information Processing Systems (NeurIPS)*, pp. 21428–21439. Curran Associates, 2020. `https://proceedings.neurips.cc/paper_files/paper/2020/hash/f52a7b2610fb4d3f74b4106fb80b233d-Abstract.html`.

[3] D. Alvarez-Melis, Microsoft Open Source, and C. Yang. `microsoft / otdd`. GitHub. `https://github.com/microsoft/otdd`.

[4] N. M. Aszemi and P. D. D. Dominic. Hyperparameter optimization in convolutional neural network using genetic algorithms. *International Journal of Advanced Computer Science and Applications*, 10(6):269–278, 2019. doi:10.14569/IJACSA.2019.0100638.

[5] E. Bochinski, T. Senst, and T. Sikora. Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms. In: *Proc. 2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3924–3928, 2017. doi:10.1109/ICIP.2017.8297018.

[6] L. Bossard, M. Guillaumin, and L. V. Gool. Food-101 – Mining discriminative components with random forests. In: *Proc. European Conference on Computer Vision (ECCV 2014)*, pp. 446–461, 2014. doi:https://doi.org/10.1007/978-3-319-10599-4_29.

[7] R. Cai and J. Luo. Multi-task learning for multi-objective evolutionary neural architecture search. In: *Proc. 2021 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1680–1687, 2021. doi:10.1109/CEC45853.2021.9504721.

[8] D. J. Dong, W. Socher, R. Li, Li-Jia, K. Li, et al. ImageNet: A large-scale hierarchical image database. In: *Proc. 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009. doi:10.1109/CVPR.2009.5206848.

[9] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, et al. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv*, 2017. ArXiv.1701.08734. doi:10.48550/ARXIV.1701.08734.

[10] M. Ghafoorian, A. Mehertash, T. Kapur, N. Karssemeijer, E. Marchiori, et al. Transfer learning for domain adaptation in mri: Application in brain lesion segmentation. In: *Proc. International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, p. 516–524, 2017. doi:https://doi.org/10.1007/978-3-319-66179-7_59.

[11] D. E. Goldberg. *Genetic Algorithms*. Pearson Education India, 2013.

[12] M. S. Haque, M. Fahim, and M. Ibrahim. An exploratory study on simulated annealing for feature selection in learning-to-rank. *International Journal of Intelligent Systems and Applications (IJISA)*, 16:86–103, 2024. doi:10.5815/ijisa.2024.04.06.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 770–778, 2016. doi:10.1109/CVPR.2016.90.

[14] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 2261–2269, 2017. doi:10.1109/CVPR.2017.243.

[15] Y. Huang, Y. Cheng, D. Chen, H. Lee, J. Ngiam, et al. GPipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv*, 2019. ArXiv:1811.06965v5. doi:10.48550/arXiv.1811.06965.

[16] S. Imai, S. Kawai, and H. Nobuhara. Stepwise pathnet: A layer-by-layer knowledge selection-based transfer learning algorithm. *Scientific Reports*, 10:8132, 2020. doi:10.1038/s41598-020-64165-3.

[17] A. Krizhevsky. *Learning multiple layers of features from tiny images.* M.sc. thesis, University of Toronto, 2009. `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In: *Proc. Advances in Neural Information Processing Systems 25 (NIPS)*, p. 1097–1105. Curran Associates, 2012. `https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html`.

[19] S. Lee, J. Kim, H. Kang, D.-Y. Kang, and J. Park. Genetic algorithm based deep learning neural network structure and hyperparameter optimization. *Applied Sciences*, 11(2):744, 2021. doi:10.3390/app11020744.

[20] S. Loussaief and A. Abdelkrim. Convolutional neural network hyper-parameters optimization based on genetic algorithms. *International Journal of Advanced Computer Science and Applications*, 9(10), 2018. doi:10.14569/IJACSA.2018.091031.

[21] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In: *Proc. Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017. `https://proceedings.neurips.cc/paper/2017/hash/32cbf687880eb1674a07bf717761dd3a-Abstract.html`.

[22] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, Berlin, Heidelberg, 2013. doi:10.1007/978-3-662-07418-3.

[23] A. Mimi, S. F. T. Zohura, M. Ibrahim, R. R. Haque, O. Farrok, et al. Identifying selected diseases of leaves using deep learning and transfer learning models. *Machine Graphics and Vision*, 32(1):55–71, 2023. doi:10.22630/MGV.2023.32.1.3.

[24] S. Nagae, D. Kanda, S. Kawa, and H. Nobuhara. Automatic layer selection for transfer learning and quantitative evaluation of layer effectiveness. *Neurocomputing*, 469:151–162, 2022. doi:10.1016/j.neucom.2021.10.051.

[25] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi:10.1109/TKDE.2009.191.

[26] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. SohlDickstein. On the expressive power of deep neural networks. In: *Proc. Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017. `https://proceedings.neurips.cc/paper/2017/hash/32cbf687880eb1674a07bf717761dd3a-Abstract.html`.

[27] H. Rehana, M. Ibrahim, and M. H. Ali. Plant disease detection using region-based convolutional neural network. *arXiv*, 2023. ArXiv.10.48550. doi:10.48550/arXiv.2303.09063.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In: *Proc. 3rd International Conference on Learning Representations (ICLR)*, 2015. Accessible in arXiv. doi:10.48550/arXiv.1409.1556.

[29] Y. Sun, B. Xue, M. Zhang, and G. G. Yen. Automatically evolving CNN architectures based on blocks. *arXiv*, 2019. ArXiv.1810.11875v2. doi:10.48550/arXiv.1810.11875v2.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, et al. Going deeper with convolutions. In: *Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015. doi:10.1109/CVPR.2015.7298594.

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In: *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016. doi:10.1109/CVPR.2016.308.

[32] M. Tan and Q. V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In: *Proc. International Conference on Machine Learning (ICML)*, vol. 97 of *Proceedings of Machine Learning Research*, p. 6105–6114, 2019. https://proceedings.mlr.press/v97/tan19a.html.

[33] H. Tian, S. Pouyanfar, J. Chen, S. Chen, and S. S. Iyengar. Automatic convolutional neural network selection for image classification using genetic algorithms. *Proc. 2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 444–451, 2018. doi:10.1109/IRI.2018.00071.

[34] C. Villani. *Optimal transport, Old and New.* Springer, 2008. doi:10.1007/978-3-540-71050-9.

[35] L. Xie and A. Yuille. Genetic CNN. In: *Proc. 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1388–1397, 2017. doi:10.1109/ICCV.2017.154.

[36] S. Zagoruyko and N. Komodakis. Wide residual networks. In: *Proc. British Machine Vision Conference (BMVC)*, pp. 87.1–87.12, 2016. https://bmva-archive.org.uk/bmvc/2016/papers/paper087/index.html.

[37] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In: *Proc. European Conference on Computer Vision (ECCV 2014)*, vol. 8689 of *Lecture Notes in Computer Science*, p. 818–833, 2014. doi:10.1007/978-3-319-10590-1_53.

[38] H. Zunair, N. Mohammed, and S. Momen. Unconventional wisdom: A new transfer learning approach applied to Bengali numeral classification. *Proc. 2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, 21:1–6, 2018. doi:10.1109/ICBSLP.2018.8554435.