Exploring the Correlation Between Random Convolutional Architectures and the Trained Equivalent

1st Nicholas Evans
Australian National University
Canberra, Australia
u3095460@anu.edu.au

2nd Jo Plested
Australian National University
Canberra, Australia
jo.plested@anu.edu.au

3rd Tom Gedeon
Australian National University
Canberra, Australia
tom.gedeon@anu.edu.au

***Abstract*—In this paper we explore the correlation between Convolutional Neural Network (CNN) architectures with random weights in the convolutional layers to the same architectures with trained weights. We show that this correlation extends to deep CNN architectures of up to 10 or even 12 layers to the extent that untrained model accuracy could be a useful proxy for trained model accuracy. We also find that for models with fewer layers much of this relationship comes from the strong correlation between the number of features output from the final CNN layer and final accuracy. With 10 and 12 layers there is a moderate correlation even when the size of the fully connected layer is held constant. We anticipate our findings in extending these correlations to deeper networks will be useful in designing faster Neural Architecture Search (NAS) models. Analytically solving for the weights of the final prediction layer is orders of magnitude faster than training the weights via backpropagation**

***Index Terms*—Neural Networks, Convolutional Neural Networks, Neural Architecture Search, NAS, random features**

I. Introduction

A large body of previous work has shown that leaving the weights of a neural network (NN) untrained and just solving the final prediction layer can result in strong performance that correlates well with an equivalent fully trained NN. The benefit is significantly lower training time. [1] left the weights from the hidden layer feed forward NN untrained and solved for the weights of the final classification layer using the Fisher solution. They showed performance comparable to the same models trained with backpropagation on synthetic datasets. [2] did the same again but solved for the weights of the final layer using a minimum norm least squares approach. They demonstrated performance comparable with support vector machine (SVM) models on house price prediction and medical imaging datasets.

The tendency of the weights from the directly solved solution to over-fit to the training set was identified in [3]. A regularised version of the minimum norm least squares approach was proposed which produced better performance than SVMs on a range of classification tasks. An alternative to inverting the large feature matrix for large numbers of training examples was also proposed. [4] reported that single layer convolutional architectures with random, untrained weights perform very well, compared to single layer CNNs on object recognition tasks. The authors also showed that a strong relationship exists between the accuracy of the trained architecture and the accuracy of the untrained equivalent. They suggested that this relationship could be used as a form of fast architecture search.

Neural Architecture Search (NAS) models [5] have produced results comparable to the top human performance for predicting the design features of the best deep NN architecture to solve a particular task. The original model from [5] was run on 800 gpus for 28 days making it out of reach for all but the largest research organisations. So focus has turned to finding good proxies to quickly predict the accuracy of large fully trained models [6] [7]

[8] successfully applied a heuristic with a moderate correlation to the trained accuracy in a NAS implementation. They showed that even this moderate correlation is likely to indicate an effective proxy.

Our paper extends [4] to include architectures with multiple convolutional layers. We define a Random Feature Model (RFM) to be a model with N untrained convolutional layers and a single classification layer that is solved analytically similar to [3]. The use of the term RFM is purely for simplicity and does not imply a newly created or novel technique.

Our paper shows how the correlation between the accuracy of RFMs and the equivalent trained model changes as the number of layers in a deep CNN are increased and other factors are taken into account. We make two contributions which show that:

1) The correlation remains as strong up to a 9 layer CNN (Pearson correlation of 0.746594) and even though it drops off at 11 layers (Pearson correlation of 0.527108) it is still strong enough to be useful as a proxy for trained accuracy.
2) The correlation is highly dependent on the number of features output from the final CNN layer, and this relationship should be considered when relying on the correlation between an RFM and equivalent trained model.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

Since RFM architectures are orders of magnitude quicker to evaluate, correlation between the RFM accuracy and the trained equivalent could be useful in the context of Neural Architecture Search (NAS).

II. BACKGROUND

A. Random convolutional architectures

[4] sample 11 different single layer convolutional architectures, each parameterised by 4 variables; filter size 4x4, 8x8,12x12, 16x16, pooling size 3x3, 5x5, 9x9 and filter stride 1, 2. The authors plotted the classification performance of each random-weight architectures against the trained weight equivalent. The random weights architectures used a linear support vector machine (SVM) to perform classification and the trained architecture used a pre-trained and fine-tuned approach. The authors observed that the architectures that performed well with random weights also tended to perform well with trained weights. They go on to suggest that the relationship between the accuracy of the random weight architecture and its trained equivalent model can be used as a form of fast architecture search.

B. Analytical Solutions to Classification Layer Weights

[1] [9] [2] used analytical methods for finding the weights of a final classification layer in an otherwise random weight neural network. [2] found the weights using an ordinary least squares approach. If H is the matrix formed by feeding each training example through the randomised part of the network and T is the matrix of targets (assuming a multi-class classification problem) . Then the weights of the final layer can be found by solving the following system for β:

$$H\beta = T \tag{1}$$

Solving for β we have:

$$\beta = H^{+}T \tag{2}$$

Where H^{+} is the Moore Penrose Psuedo Inverse of the matrix H. Huang extended the above to a regularised version:

$$\beta = \left(\lambda I + H^{T}H\right)^{-1} H^{T}T \tag{3}$$

The regularised version has been shown to improve generalisation to unseen data [3].

III. METHOD

A. General method

1) Define a search space
2) Randomly sample candidate models from that search space
3) Train candidates models for 10 epochs
4) Construct and solve an RFM from the candidate architecture
5) Record the validation accuracy that is achieved by both the RFM and trained model

B. Data set and preprocessing

CIFAR-10 [10] image dataset is used for the experiment. The provided split of 50000 training examples and 10000 test examples is used. Unlike [4] the images are not converted to black and white since we are interested in finding architectures that perform well on the colour images. We use a pre-processing scheme, similar to [5] since these steps have been shown to give top results on the CIFAR-10 dataset. For each image:

1) the image is padded by 4 pixels and a random 32*32 crop is sampled
2) random horizontal flips are performed
3) the image is normalised using the mean (0.4914, 0.4822, 0.4465) and standard deviation of each channel (0.2023, 0.1994, 0.2010)

C. Architecture Search Space

We perform five distinct correlation experiments for networks of varying numbers of layers. Experiments are conducted for networks 3, 5, 7, 9 and 11 convolutional layers. Each convolutional layer that is sampled is described by two variables, convolutional filter size, chosen from 3, 5 and multiplier, chosen from 1, 1.25, 1.5. The multiplier describes how many features are output from the convolution operation by applying the following rule:

*output_feature_maps = input_features_maps * multiplier* (4)

The number of feature maps output from the initial convolution operation is set to 24. Modern architectures such as [11] will often use 64; we scale this down to match the relative complexity of the architectures being sampled. We note here that the resulting models do not have an equal number of parameters and this is also true of the initial experiment [4]. Given that the total number of unique convolutional layer configurations is 6, there are 6^{n} unique architectures, where n is the number of layers sampled. The simplified search space also did not include depthwise separable convolutions [15], or non-square filter dimensions [5] which have been shown to produce top results. This allowed us to test purely whether correlations between CNN models with trained and untrained weights extend to deeper models without other complexities affecting the results.

TABLE I
PARAMETER SETTINGS

Param	Value
Momentum	0.9
Weight decay	0.00004
Learning Rate Cosine annealing	Max 0.1, Min 0.001
Batch Size	32

D. Constructing the Network

A neural network is constructed for each sampled model, with the filter sizes and number of feature maps being determined by the sampled architecture string. Additionally, 2

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE II
CORRELATION RESULTS – TRAINED VS RANDOM FEATURES

Depth	Pearson coefficient	Spearman coefficient
3	0.664021	0.675301
5	0.70894	0.715277
7	0.709527	0.699675
9	0.746594	0.73908
11	0.527108	0.585333

[a]All values are statistically significant.

max pooling layers are placed into each network at even spacing. Each "convolutional layer" as we describe it in this paper is actually a block of convolution $\rightarrow$ ReLU $\rightarrow$ batch normalisation. Along with the limitations in the search space no modern architectural design complexities such as skip connections, dropout, auxiliary head etc generally used in deep CNNs were included. This and the search space limitations allowed a straightforward answer to the question of "does the relationship between CNN models with untrained and trained weights hold as the number of layers increases?". We do not think this limits the comparison within the scope of this work, but it does mean that the final accuracy cannot be fully compared with other modern results.

E. Classification with the RFM

The weights of the final classification layers are computed using equation 3:

$$\beta = \left(\lambda I + H^T H\right)^{-1} H^T T \tag{5}$$

H is the matrix formed by feeding each training example through the randomised part of the network, T is the matrix of targets, β is the weight matrix that is being solved and λ is a user defined regularisation constant. λ is optimised against a validation set of 5000, taken from the test set. An optimal value of 0.7 was found and this value was used for all other experiments.

For each of the sampled architectures, three random initialisations are used and the results on the test set are averaged. For each RFM that is sampled, the corresponding CNN is constructed and trained for 10 epochs using back-propagation. The hyper parameters that are used are consistent with those used in [5] and are listed in Table 1.

IV. RESULTS AND DISCUSSION

A. Correlation results

Figures 2 & 3 along with Table II show there is still a strong correlation between the accuracy from the RFM models and trained models when the numbers of CNN layers are increased to 7 and 9. The Pearson and Spearman correlations are 0.71 and 0.74 respectively. In figure 4, when the number of CNN layers is increased to 11 the correlation becomes more moderate at 0.52. As per [8] a moderate correlation means it is still likely to be a useful proxy for fast architecture search.

We trained to convergence the models with the top 5 predicted accuracies from both the 10 epoch and RFM models for 9 and 11 convolutional blocks. Table III reports the average and standard deviation of these results. While the average accuracies for the 10 epoch and RFM models are very similar the standard deviation for the RFM models are over twice as high. This indicates that using RFM as a heuristic for fully trained models results in a noisier estimate.

The accuracy of the 11 convolutional block model being lower than the 9 convolutional block model shows the limitations of not including modern architectural design features. A particular limitation is that as per [12] [11] [13] the modern architectures that produce the best results on CIFAR 10 follow the general rule of doubling the number of feature maps when the spatial size of the feature maps is halved to remove representational bottlenecks whereas our models do not follow this rule. In our models we include 2 evenly spaced downsampling layers in each model no matter how many convolutional blocks we have. While this limitation does reduce the best final accuracies that can be produced by the models we have found no reason to think it affects the comparison between the trained vs the RFM models.

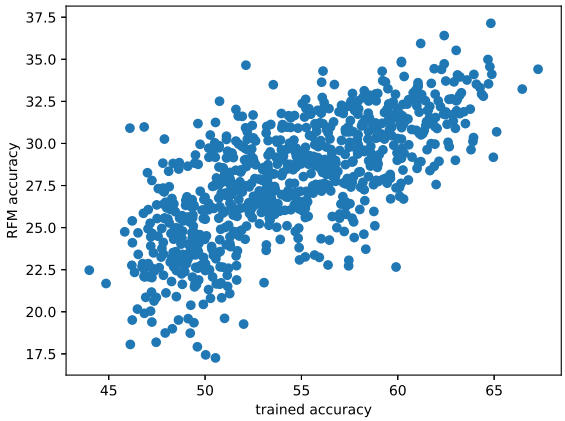

Fig. 1. RFM accuracy as a percentage (y axis) compared to trained accuracy (x axis) for a model with 5 convolutional layers (6 layers total)

TABLE III
TOP FINAL FULLY TRAINED RESULTS

Depth	Top 5 by 10 epoch accuracy	by RFM accuracy
9	85.23±0.4%	84.75±0.87%
11	84.55±0.46%	83.62±1.23%

TABLE IV
CORRELATION RESULTS - FINAL LAYER SIZE VS ACCURACY

Depth	Spearman coefficient trained	Spearman coefficient RFM
3	0.912154	0.726757
5	0.916082	0.744238
7	0.900444	0.697415
9	0.872316	0.715059
11	0.26569	0.593903

[a]All values are statistically significant.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

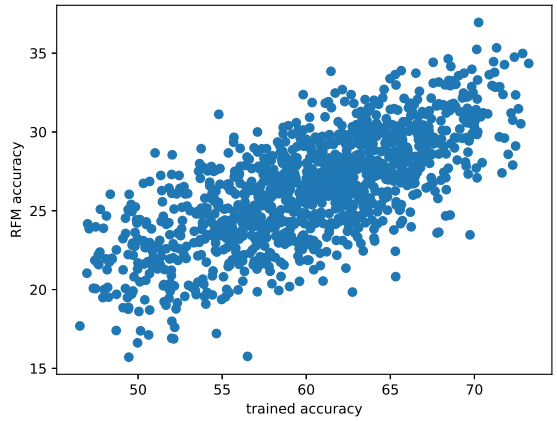

Fig. 2. RFM accuracy as a percentage (y axis) compared to trained accuracy (x axis) for a model with 7 convolutional layers (8 layers total)

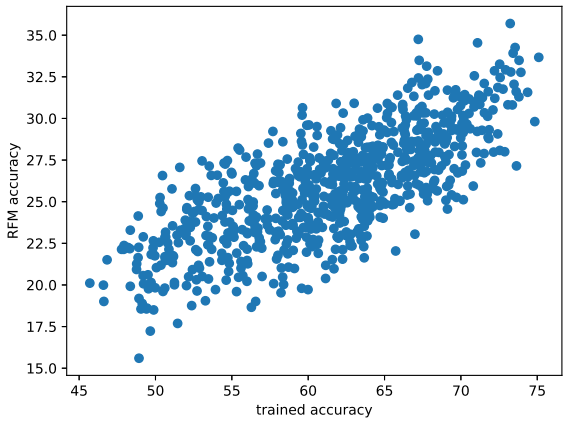

Fig. 3. RFM accuracy as a percentage (y axis) compared to trained accuracy (x axis) for a model with 9 convolutional layers (10 layers total)

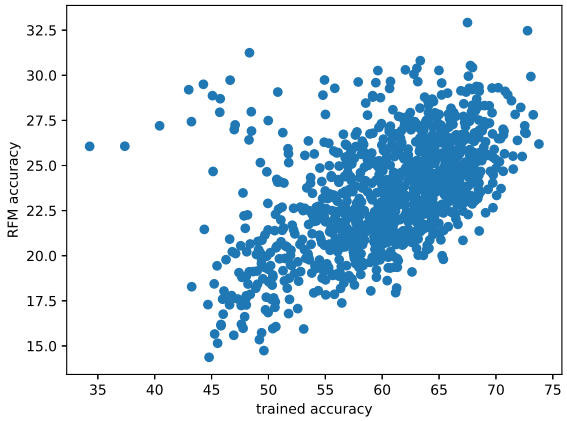

Fig. 4. RFM accuracy as a percentage (y axis) compared to trained accuracy (x axis) for a model with 11 convolutional layers (12 layers total)

B. Correlation with constant feature size

We additionally calculate the Spearman coefficient between the validation accuracy and the size of the final classification layer, the results are recorded in Table IV.

We observe that the accuracy of both the RFM and 10 epoch trained models correlates very strongly with the final layer size.

The first thing to note is that the correlations between final layer size and accuracy are far higher for models with a smaller number of layers than a greater number. This seems to indicate that the models with less layers are underfitting the data and thus perform better with more features (a larger final layer size). Whereas for the models with a higher number of layers the correlation is roughly the same as or even lower than the correlation between the trained and RFM model.

We now aim to determine if a correlation exists when the size of the final layer is held constant. We know that both the RFMs and our partially trained models perform well when the final layer is large. But are RFMs a useful proxy for finding good architectures when the final layer size is held constant? In order to answer this question we calculate the correlation amongst models with a similar final layer size. For each network depth (5, 7, 9, 11), we group the models into 20 evenly spaced bins and calculate the correlation within these groups. The results are shown in Table V

TABLE V
CORRELATIONs TRAINED VS RFM WITHIN SAME FINAL LAYER SIZE

Layers-(Final Layer Size Range]	Pearson	Spearman
11 - (3033.6, 3507.2]	0.358299	0.297166
11 - (1612.8, 2086.4]	0.321455	0.339726
11 - (182.528, 665.6]	0.526462	0.517113
9 - (1660.8, 1824.0]	0.393931	0.467433
9 - (681.6, 844.7]	0.408181	0.410049
9 - (188.7, 355.2]	0.288102	0.288102
7 - (1142.4, 1248.0]	0.354627	0.27737
7 - (614.4, 720.0]	0.317162	0.331108
7 - (403.2, 508.8]	0.207533	0.220077
5 - (806.4, 857.6]	0.20966	0.275019
5 - (499.2, 550.4]	0.279318	0.201833
5 - (190.976, 243.2]	0.0694111	0.0636323

It can be seen from table V that for 5-9 convolutional blocks the correlation between the RFM and trained accuracy is much higher with a larger number of features. However with 11 layers the correlation reduces slightly for a larger number of features. This indicates that the model is underfitting with a smaller number of layers and that 11 convolutional blocks is a reasonable size for this task. Both table V and figures 5 to 8 show that while all the Final Classification layer (FC) sizes shown for 9 and 11 convolutional blocks have a significant correlation, the smaller FC layer sizes for 5 and 7 convolutional blocks do not. Again leading us to conclude that 9 and 11 convolutional blocks is a better size for this problem. The fact that the correlation is still strong for these sized models makes the RFM a useful proxy for trained model accuracy for the CIFAR 10 classification problem.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

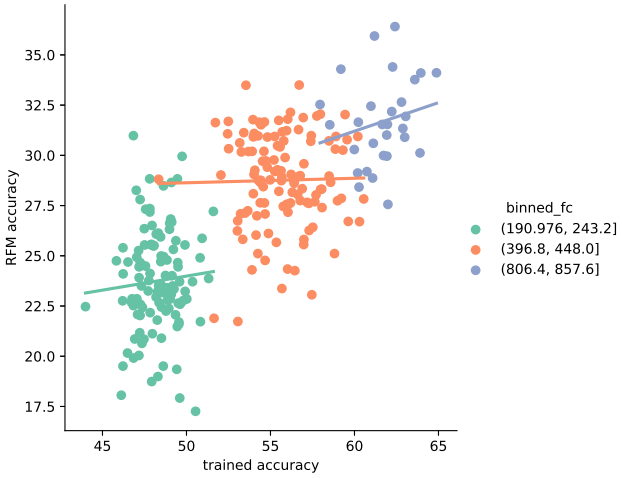

Fig. 5. RFM accuracy as a percentage (y axis) compared to trained accuracy (x axis) for a model with 5 convolutional layers (6 layers total) with low, medium and high fully conenected layer size

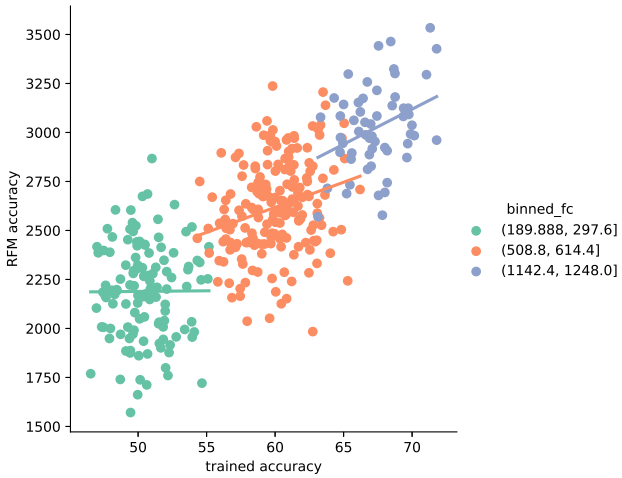

Fig. 6. RFM accuracy as a percentage (y axis) compared to trained accuracy (x axis) for a model with 7 convolutional layers (8 layers total) with low, medium and high fully connected layer size

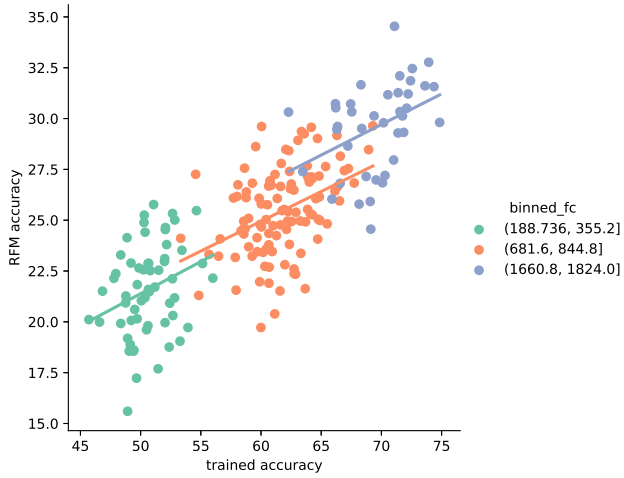

Fig. 7. RFM accuracy as a percentage (y axis) compared to trained accuracy (x axis) for a model with 9 convolutional layers (10 layers total) with low, medium and high fully connected layer size

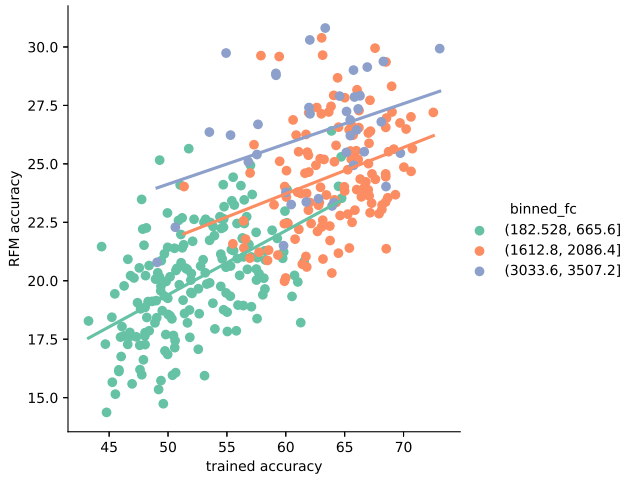

Fig. 8. RFM accuracy as a percentage (y axis) compared to trained accuracy (x axis) for a model with 11 convolutional layers (12 layers total) with low, medium and high fully connected layer size

C. Discussion

We conclude from our results that our original questions of ”does the relationship between CNN models with untrained and trained weights hold as the number of layers increases?” and ”can an RFM model be a useful proxy for trained models?” can be answered affirmatively. Care is needed when relying on these correlations to make sure the models being compared are an adequate size, and not underfitting, so that the correlation does not come mostly from the FC layer size.

We conclude that a correlation of over 0.5 for a model size that is reasonable for this task shows the potential of RFMs to be used as a heuristic to test architectures that are relevant to modern tasks. Additional limitations that may be uncovered if the comparison is expanded to more complex modern architectures in future work.

V. Conclusion and Future Work

This paper has quantified the correlation between deep CNNs with random weights and the trained equivalent models and found a strong correlation for up to 9 CNN layers. The correlation was still moderate when the number of layers was increased to 11. When the correlation is performed among models with the same size fully connected layer the correlation is considerably weaker, particularly with more shallow models. We conclude that the observed strong correlation appears to be in part because both RFMs and the trained models perform better when the number of inputs into the final classification layer is high. This is particularly the case with shallow models that are more likely to be underfitting. Caution should be used in consider an RFM as a proxy for a trained model in these cases. A moderate correlation within the same size

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

fully connected groups was still present across all groups analysed for the deeper models. [8] successfully applied a heuristic with a moderate correlation to the trained accuracy in a NAS implementation so there is evidence that random feature models are worth exploring as a proxy for trained models to reduce the cost of evaluating candidate architectures.

A limitation of this work is that the search space and CNN architecture are relatively simple so the results so far do not compare to the state of the art on CIFAR 10. Additional research is needed to determine if the observed correlation still holds when the search space is expanded to include modern state of the art architectural features.

REFERENCES

[1] Schmidt, W.F., Kraaijveld, M.A. and Duin, R.P., 1992, August. Feed forward neural networks with random weights. In International Conference on Pattern Recognition (pp. 1-1). IEEE COMPUTER SOCIETY PRESS.

[2] Huang, G.B. and Siew, C.K., 2004, December. Extreme learning machine: RBF network case. In ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004. (Vol. 2, pp. 1029-1036). IEEE.

[3] Huang, G.B., Zhou, H., Ding, X. and Zhang, R., 2011. Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 42(2), pp.513-529.

[4] Saxe, A.M., Koh, P.W., Chen, Z., Bhand, M., Suresh, B. and Ng, A.Y., 2011, June. On random weights and unsupervised feature learning. In ICML (Vol. 2, No. 3, p. 6).

[5] Zoph, B. and Le, Q.V., 2016. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.

[6] Pham, H., Guan, M.Y., Zoph, B., Le, Q.V. and Dean, J., 2018. Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268.

[7] Zela, A., Klein, A., Falkner, S. and Hutter, F., 2018. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. arXiv preprint arXiv:1807.06906.

[8] Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J. and Murphy, K., 2018. Progressive neural architecture search. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 19-34).

[9] Pao, Y.H. and Takefuji, Y., 1992. Functional-link net computing: theory, system architecture, and functionalities. Computer, 25(5), pp.76-79.

[10] Krizhevsky, A., Nair, V. and Hinton, G., 2014. The cifar-10 dataset. online: http://www. cs. toronto. edu/kriz/cifar. html, 55.

[11] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

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

[13] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[14] Zoph, B., Vasudevan, V., Shlens, J. and Le, Q.V., 2018. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8697-8710).

[15] Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

978-1-7281-6926-2/20/$31.00 ©2020 IEEE