Data Driven Designing of Convolutional Neural Networks Architectures for Image Classification

Sajad Ahmad Kawa, *Member, IAENG,* M. Arif Wani

***Abstract*—The process of designing Convolutional Neural Networks (CNN) architecture has been automated using Neural Architecture Search (NAS) algorithms, however, these algorithms pose a significant computational demand due to the large search space of possible architectures. This paper proposes a two-stage algorithm to address the computational resource demand of NAS by generating architectures directly from the dataset. In the first stage, the complexity tree is generated from the dataset based on the complexity of individual images. The image complexity is determined by color variation and changes in color intensity. This complexity tree is then utilized in the second stage for designing the CNN architecture. As the complexity tree solely necessitates one-time generation for the dataset, our proposed model does not have significant computational demands. In the experiment that was performed utilizing standard benchmark datasets, the generated (CNN) models attained an accuracy of 97.35% on CIFAR-10 and 72.57% on CIFAR-100 datasets. Our proposed model was compared with state-of-the-art models and yielded a notable improvement in accuracy. Furthermore, it outperformed all other models in terms of computational resource requirement.**

***Index Terms*—CNN Architecture, Image Complexity, Macro Architecture, NAS**

I. Introduction

ARTIFICIAL intelligence models utilizing deep neural networks have had significant impacts on image classification, object detection, cancer classification, accident prediction, and other related areas of interest, as referenced in several scholarly sources [1], [2], [3], [4], [5] and [6]. These models have demonstrated considerable performance improvements over traditional image classification techniques [5], largely due to their capacity for automatic feature learning [7]. Unlike traditional approaches, deep learning models can learn the features required for classification or regression tasks during the training phase, without the need for feature engineering by a human expert. The training process of deep learning models is entirely automatic and end-to-end, with only the model architecture and training parameters requiring manual input from a human expert.

The architecture of a CNN model is defined by a set of parameters including the size and strides of filters, depth of layers, types of layers and activation functions, and various training parameters. These parameters collectively determine the performance level of the network [8], [9]. While human experts have traditionally designed CNN architectures, this process has now been automated using Neural Architecture Search (NAS) [10]. NAS is a search process for discovering architectures that perform better among a defined set of architectures within a specific search space. Various NAS methods and methodologies have been utilized to design CNN architectures, including traditional random and grid search methods, as well as more recent evolutionary, reinforcement learning, and Bayesian optimization approaches.

The conventional approach to NAS entails random search [11], [12] or grid-based search methods without incurring any additional overhead computation. While random search samples values for different parameters of CNN architecture from a pre-defined range of values, grid search conducts an exhaustive search that covers the entire search space. However, these traditional methods may fail to generate the best-performing architecture, as random search is not exhaustive, and grid search is inefficient for large search spaces and not scalable. In recent years, advanced methods for NAS have emerged. For instance, evolutionary algorithms inspired by biological evolution generate architectures by evaluating a large number of candidate architectures [13], [14]. Bayesian optimization, utilized by [15], builds a probabilistic model over the search space and uses it to search for architectures. Reinforcement learning-based NAS, demonstrated by [16], [17], [18], has yielded significant performance improvements but comes at a high computational cost. Despite the advanced architectural achievements of NAS models, NAS methods still exhibit significant limitations. Notably, traditional NAS methods tend to produce suboptimal results, while more sophisticated NAS methods place heavy demands on computational resources. Moreover, all of these NAS methods necessitate the training and evaluation of a considerable number of models to generate a high-performing model.

In our proposed model, we address the search space and architecture search issues in a unique manner. We leverage the dataset to devise the CNN architecture, which obviates the need for searching an architecture that delivers optimal performance on a given dataset. The CNN architectures in our proposed model are derived from the complexity tree of the dataset, which is created prior to the design of the architectures and only requires parsing of the dataset once. This approach eliminates the need to train and evaluate a large number of architectures. We further mitigate the computational resource issue in NAS by utilizing parameter sharing among the architectures that are formed based on the complexity tree. Overall, our approach is novel and efficient in addressing the challenges associated with NAS.

The paper is divided into several distinct sections. Section II features a comprehensive review of the pertinent literature on various approaches and techniques employed in NAS. The proposed model is expounded in detail in Section III. In Section IV, the experimental setup and corresponding results are described, including a comparative analysis with other contemporary models. Section V is dedicated to presenting the conclusions.

Manuscript received July 27, 2022; revised April 19, 2023.

S. Kawa is a Research Scholar at the Department of Computer Sciences, University of Kashmir, Srinagar, 190001, India (Corresponding Author Mobile: +91-7780-949848; e-mail: sajadkawa9@gmail.com).

M. A. Wani is a Professor at the Department of Computer Sciences, University of Kashmir, Srinagar, 190001, India (e-mail: awani@uok.edu.in).

II. Related Works

A. Neural Architecture Search

NAS is defined by a search space and the architecture search method [10], [19]. The exploration space of NAS is often vast, requiring considerable computational resources as noted by previous research [14]. The resource constraint is addressed by different authors in multiple ways. According to [20], [12], Monte Carlo analysis provides a viable option for estimating parameters. However, the random approach used in architecture search algorithms lacks adaptivity, which renders it incapable of utilizing previous iterations to inform subsequent architecture selection. Moreover, the search process may select parameter values from suboptimal regions of the search space, where performance is minimal and importance is negligible.

The authors in [21] employ grid search in their work on NAS to ensure reproducible outcomes. Nonetheless, the implementation of this technique necessitates considerable computational resources, underscoring the limitations of grid search in managing large search spaces and underscoring the urgency for more efficient search methods. Bayesian optimization is a popular technique for optimizing the search space of parameters using a probabilistic model. This method is typically built upon a surrogate model, as proposed by [22], which is based on a probabilistic framework and is trained on the actual objective function, which in the case of architecture search is the architecture itself. To determine the next set of candidate parameters to evaluate, an acquisition function is utilized. Since the relationship between the architecture parameters and performance is not explicitly known, the surrogate model can be constructed using a variety of methods, such as a Gaussian process or a Parzen estimator with Bayes rules [23]. The choice of acquisition function is also flexible and can vary depending on the problem at hand. Some examples include a random forest-based decision tree [24] or an iterative optimization method combined with gradient descent [25], [26].

In their works,[27], [28] introduce the use of evolutionary algorithms to define the architecture of NAS. The NAS architecture is represented as a graph layout consisting of nodes and edges, where nodes represent layers, and edges represent connections among these layers. Additionally, the architecture can be specified in a binary chromosome format, as suggested by [29] and [30] The single-level architecture specification has been identified with various issues, which have been addressed by a multi-level architecture specification as proposed by [31] and [32], where individual components are encoded at one level, and the complete architecture is encoded at a higher level [13]. Despite the success of evolutionary algorithms in generating good architectures, it is noted that this approach requires significant generations to arrive at a good architecture, thus imposing a substantial computational demand.

According to [18], the architecture parameters are represented as network embeddings produced by a Long Short-Term Memory (LSTM) layer, which are optimized through particle swarm optimization [33]. Additionally, the authors employed a Q-learning-based reinforcement learning approach [34] for iterative development of the architecture parameters with greedy exploration. The reinforcement learning technique can also be employed to develop the architecture through network transformation, whereby an agent takes the action of network transformation, or based on policy gradient search for a subgraph [35].

The resource constraint issue in NAS has also been addressed by using a constrained exploration space [36], [26], reducing the dimensions of the search space by limiting parameter values to specific ranges, or exploring only the architecture of a cell or block [18] rather than the entire macro architecture. In a study by the authors in [17], the resource constraint is addressed by incrementally constructing the components of the architecture layer by layer with the use of a resource-efficient Recurrent Neural Network (RNN) controller [36]. This method involves a layer-by-layer search approach, which is subsequently assembled to construct the macro architecture. However, there is still a requirement to explore the layers and module structure. By employing a layer stacking technique, in conjunction with the Q-learning [18] strategy and a relaxed continuous exploration space [26], the resulting architectures exhibit high efficiency and superior performance. The authors in [37] employ stochastic implicit gradients to optimize the architecture search process. The challenge of resource limitations has also been tackled through the development of a performance estimation approach [38] that reduces the necessity of training and evaluating multiple architectures by assigning scores to untrained architectures [16], but still the requirement of estimating the score requires computation resource in itself.

B. Image Complexity

In the domain of computer vision, the assessment of image complexity has attracted significant interest. The methods for evaluating image complexity have traditionally relied on subjective human judgment, which may lead to inconsistencies and bias. To address this issue, [39] proposed a method for objectively measuring image complexity using fuzzy clustering techniques for edge detection. Another measure of image complexity was proposed by [40], which is based on the spatial distribution of color intensity values and partitioning regions calculated using a divergence value. Image complexity has also been defined using multiple factors such as color variation, texture, and spatial frequency. A weighted average of the fuzzy sets is then utilized to generate an overall measure of image complexity [41]. It has also been noted by authors in [42] and [43] that the complexity of an image can be determined using the color similarity and spatial pixel distribution, while [44] takes into consideration the resolution of the image along with the total number of colors in the image for defining the image complexity.

The majority of approaches for assessing image complexity have been designed for quantifying general image quality or complexity. Nevertheless, some recent studies, such as the one cited in [45], have utilized image complexity as a means of compressing neural networks and making predictions regarding the accuracy of a neural network model on an image dataset. In contrast, the authors of [46] have extended these methods by leveraging image complexity to inform decisions regarding network depth selection and assess the impact of these choices on image segmentation.

III. Proposed Model

The proposed model uses a novel approach that utilizes a two-stage algorithm for developing a CNN architecture tailored to a specific dataset. The first stage of the algorithm involves the application of a unique image-splitting method to each image in the dataset. This method is designed to generate the complexity tree of an image, which takes into account its color variation and intensity levels. This allows for the creation of a structured representation of the image's complexity. These complexity trees are then used to construct the complexity tree for the entire dataset. In the second stage of the algorithm, the complexity tree of the dataset is utilized to establish the CNN architecture for this particular dataset. The architecture is designed to leverage the information contained in the complexity tree to optimize its performance in handling the specific characteristics of the dataset. By tailoring the CNN architecture to the dataset, it is expected to achieve better performance in terms of accuracy and efficiency.

A. Stage One: Complexity Tree

In this stage, an image-splitting method is applied to each image in the dataset, which recursively divides the images into two images at multiple depth levels based on the color and intensity variations present in the image. The dataset's complexity tree is constructed by averaging the complexity trees of all the training images, which depicts the data complexity in the dataset in the form of a tree structure. This splitting method is inspired by the functioning of CNN, where the filter applied to the input image or feature maps generates an output feature map that highlights features in the image. As the filter weights are learned during the training process, it convolutes or highlights several inherent aspects of the image. The feature map generated by a filter has the activated region of images that the filter has learned while training. In a similar manner, the splitting method assesses the complexity of an image by recursively splitting it at various depth levels.

The split process initiates from the input image, which splits into two images of the same size as that of the original input image, with one image having color intensity values below a specific threshold value and the other having intensity values above the same threshold value. The threshold is computed from the largest variation within the image's intensity values. The two resultant images are passed to a lower depth level, where the same splitting process continues further. The output is a binary tree structure containing the same image with different informational content at different depth levels. The split process terminates for an image in a branch when the image's intensity variation reduces to such a low level that further splitting is not possible. These variations within the image indicate different parts or objects in an image, similar to how human perception perceives objects and parts of an image based on the changes in color intensity. The splitting process assumes that significant variations in the image represent different parts or objects in the image.

1) Complexity of an Image: Algorithm 1 and Algorithm 2 presented below delineate the split method employed for generating the complexity tree for an image. This split method is uniformly applied to all images in the training set of the dataset, yielding a complexity tree for each image. The split algorithm for one of the images is illustrated in Fig. 1

Algorithm 1 Algorithm for generating the complexity tree of an image

Require: $image$ ▷ Input Image
Ensure: $tree$ ▷ Complexity tree of Image
1: Make empty binary $tree$
2: Insert $image$ in $tree$
3: SplitFunction($tree$,$tree.root$)

Algorithm 2 Split Function

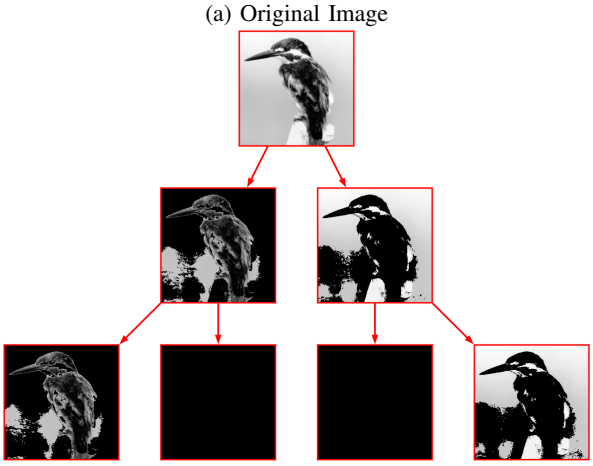
1: **procedure** SplitFunction($tree$,$node$)
2: **if** $node.hasNonZeros \geq 8$ **then**
3: r, g, b = $node.getRGB$
4: Sort(r, g, b)
5: $diff$ = subtract adjacent value in r, g, b
6: $thres$ = Max($diff$)
7: $nodeLeft$ = $r, g, b < thres$
8: $nodeRight$ = $r, g, b > thres$
9: $tree.leftChild \leftarrow nodeLeft$
10: $tree.rightChild \leftarrow nodeRight$
11: SplitFunction($tree$,$tree.leftChild$)
12: SplitFunction($tree$,$tree.rightChild$)
13: **end if**
14: **end procedure**

2) Complexity of dataset: The complexity tree of a specific dataset is formed through the aggregation of the complexity trees of individual images present in the training dataset, as outlined in Algorithm 3. This is accomplished by computing a weighted average of all feasible branches resulting from the splitting method. The weight assigned to each branch is determined by the number of branches having a particular depth. To ensure comprehensive coverage, all possible branches within the complexity trees are considered when computing the average. Notably, the calculation accounts for the possibility of sub-branches of varying lengths within each branch, thereby taking the average of each produced branch and its sub-branches within the dataset. The resulting complexity trees of the datasets used in this paper are presented in Fig. 2

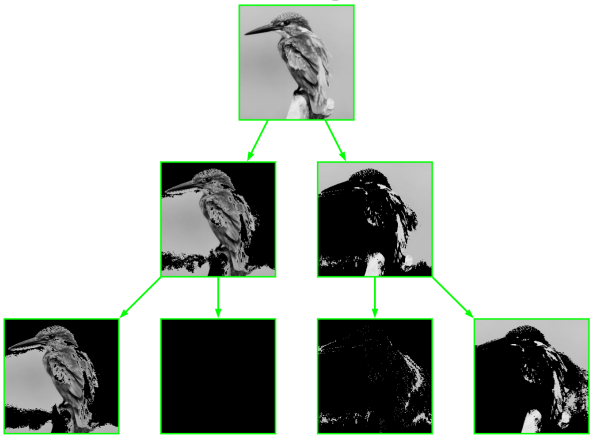
B. Stage Two: Designing the CNN architecture

The architecture design of the CNN model is based on the complexity tree of the dataset, which is a binary tree structure, that is developed in stage one. The binary tree is utilized in defining the architecture model for the CNN. The macro architecture of CNN reflects the branches and nodes of the complexity tree, where the nodes correspond to the split images at different levels of depth in the tree as shown in Fig. 3 for the CIFAR-10 dataset. Specifically, the complexity tree nodes represent the image segments of various depths, while the architecture nodes denote the computation nodes utilized for learning image features during training. The computation nodes in the architecture represent the features of the images

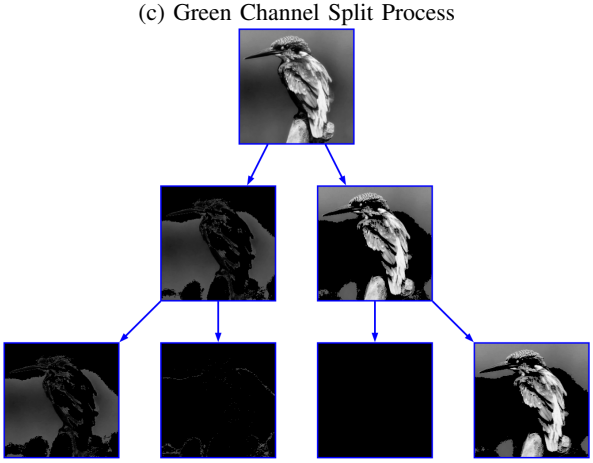
(a) Original Image



(b) Red Channel Split Process



(c) Green Channel Split Process



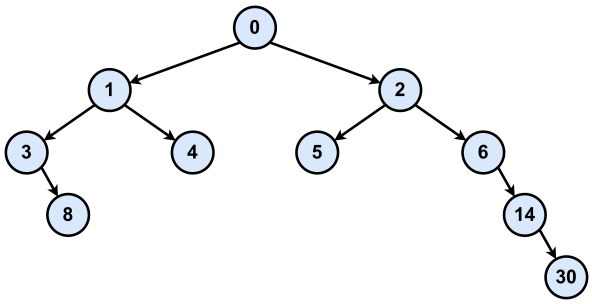
(d) Blue Channel Split Process

Fig. 1: Split Process on an Image

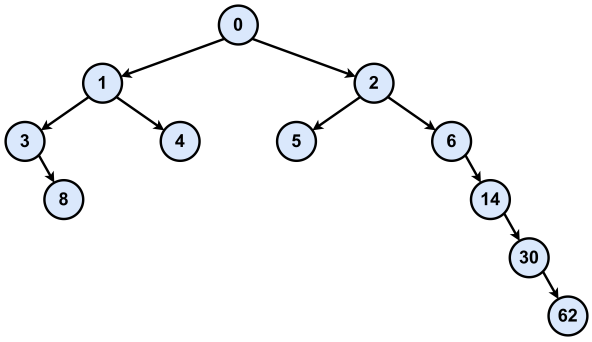
Algorithm 3 Algorithm for generating complexity tree of dataset

Require: $treeList$ ▷ Complexity trees of all images
Ensure: $tree$ ▷ Complexity tree of Dataset

for all $branch$ in $treeList$ **do**
  $branchW = branch.Depth * branch.Count$
  $branchC = branchC + branch.Count$
end for
$avgDepth = branchW \ / \ branchC$
for all $branch$ in $treeList$ **do**
  if $branch.Depth == avgDepth$ **then**
    $branchF \leftarrow branch$
  end if
end for
$tree \leftarrow$ MakeTree($branchF$)



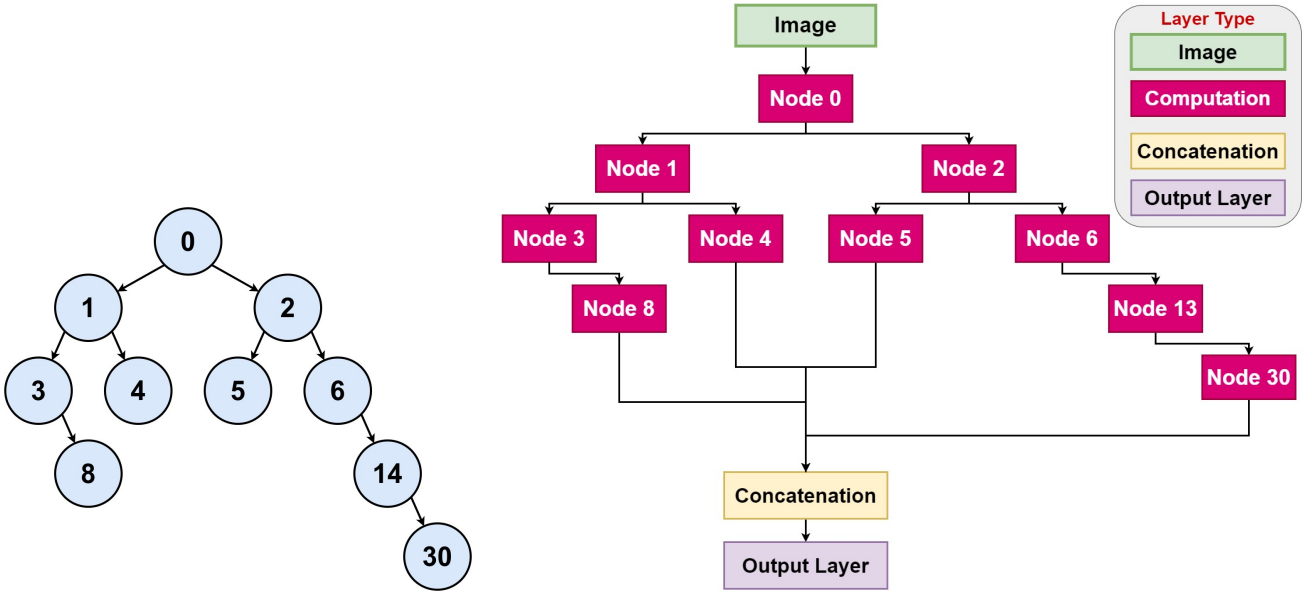
(a) CIFAR-10 Complexity Tree



(b) CIFAR-100 Complexity Tree

Fig. 2: The Complexity Trees of the Datasets

during the training phase. Consequently, we have obtained a single CNN architecture with multiple branches, in which the nodes share a parent node, leading to a smaller number of parameters and faster training. The complexity tree nodes are transformed into computation nodes, and the branches are used for the connection between them, resulting in the formation of different branches in the CNN. The output of all the leaf nodes is combined to create a single node that serves as input to the Output layer. For actual computation nodes, we have primarily used the convolution layer, the Inception layer as introduced by [1], and Resnet block as introduced by [2]. During training and evaluation, three distinct CNN models were utilized for each dataset, denoted as TreeNAS-Conv, TreeNAS-Inception, and TreeNAS-Resnet, each of which incorporated the convolution layer, Inception layer, and Resnet block as computation nodes respectively.



(a) CIFAR-10 Complexity Tree

(b) CNN Architecture for CIFAR-10

Fig. 3: CIFAR-10 Architecture designed from the Complexity Tree

IV. Experiment and Results

A. Datasets

1) CIFAR-10: The CIFAR-10 dataset is a prevailing benchmark dataset to assess the performance of various NAS techniques. The dataset includes 60,000 images that are divided into ten distinct classes, namely airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Furthermore, the dataset has been split into a standard train and test set, with 50,000 and 10,000 images, respectively. This standardized partitioning enables consistent performance evaluation among different architectural formation approaches. Each image in the dataset is composed of 32x32 pixels with three Red Green Blue (RGB) color values.

2) CIFAR-100: The CIFAR-100 dataset is a benchmark dataset for image classification tasks. This dataset contains 100 classes of images, with each class consisting of 600 images. The total number of images in the dataset is 60,000. Each image in the CIFAR-100 dataset is of size 32x32 pixels with three color channels, i.e., Red, Green, and Blue (RGB). The dataset has been divided into training and test sets, with 50,000 and 10,000 images, respectively. The CIFAR-100 dataset is commonly used in research for evaluating the performance of image classification models due to its diverse and challenging set of classes.

B. Training of Models

The study employs three different computation nodes, which are the convolutional layer, the Inception layer [1], and Resnet Block [2]. As such, three CNN models are trained for each dataset, namely TreeNAS-Conv, TreeNAS-Inception, and TreeNAS-Resnet. Although the macro architecture of these models is identical, they differ at the computation node level. Specifically, the TreeNAS-Conv model uses a convolution layer with a 3x3 kernel size, followed by batch normalization and relu activation as computation nodes. The TreeNAS-Inception model utilizes the Inception layer, while the TreeNAS-Resnet model adopts the Resnet Block. The investigation reveals that the TreeNAS-Resnet model produces the best outcomes. Therefore, the Resnet Block used in this model is depicted in Fig. 4. All CNN architectures trained have a Maxpooling layer after the second depth, with a pooling size of 2x2. The output of the leaf nodes is concatenated, followed by a Global Average pooling layer, a Dense layer with 512 neurons, and, finally, an output layer based on the number of classes in the dataset.

Each of these models underwent training and testing on the Google Colab platform, a cloud-based Jupyter notebook environment equipped with GPU support. The models underwent 300 epochs of training, utilizing the Adam optimizer and implementing an early stop strategy with patience of 30 epochs.

C. Results

In this study, we conducted experiments to evaluate the performance of three different models, namely TreeNAS-conv, TreeNAS-inception, and TreeNAS-Rsenet, on two standard image datasets, CIFAR-10 and more complex CIFAR-100. We summarize the results of our experiments in Table I, which includes information on the number of parameters learned, the duration of the training, and the accuracy of each model. Our findings indicate that TreeNAS-Resnet outperformed the other two architectures in all three parameters. Consequently, we selected TreeNAS-Resnet for further analysis in this study. Moreover, we conducted a comparative analysis of our model (TreeNAS-Resnet) against six other state-of-the-art models for Neural Architecture Search (NAS). We compared the models based on the metrics of classification accuracy and training time, which are commonly used in evaluating competing methods. We report the results of our analysis in Table II for CIFAR-10 and Table III for CIFAR-100. Our findings indicate that our model consistently outperformed the other models on both datasets.

Results from the experiment indicate that the TreeNAS-Resnet model outperforms existing models on both bench-

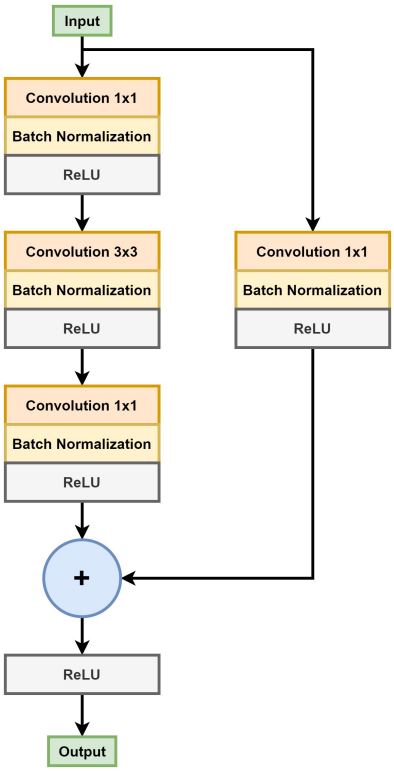
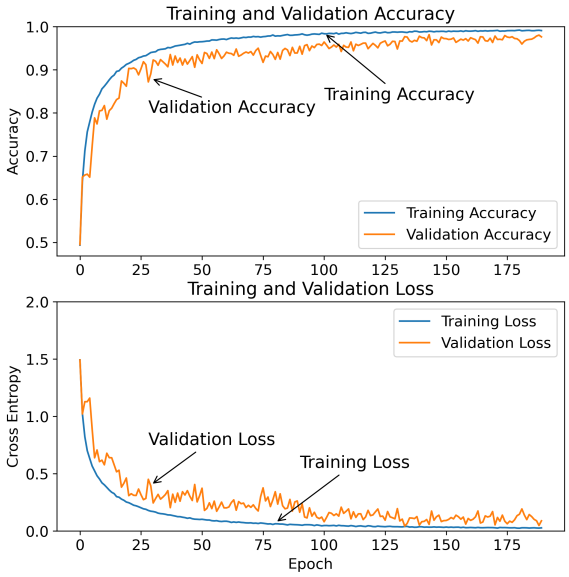
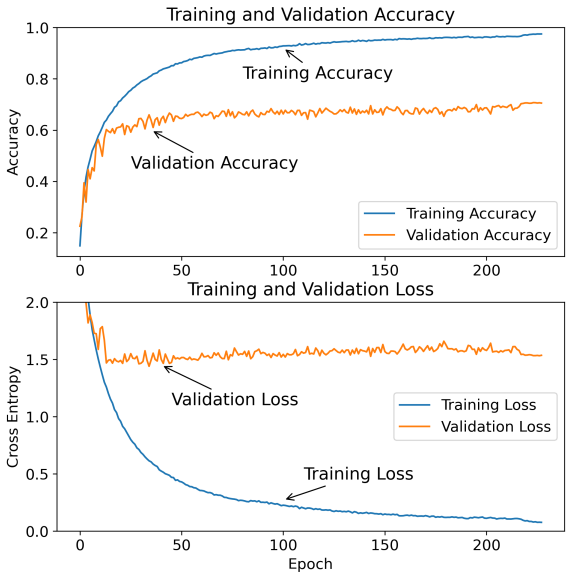


Fig. 4: Resnet Block used in this paper



(a) CIFAR-10 TreeNAS-Resnet Performance Plot



(b) CIFAR-100 TreeNAS-Resnet Performance Plot

Fig. 5: TreeNAS-Resnet Model Accuracy and Loss Plot

mark datasets. Moreover, the two-stage algorithm yields significant computational savings, as it does not require extensive training of numerous architectures. The TreeNAS-Resnet model generates a tree structure with branches of specific depth, which leads to better results. Notably, the Resnet block utilized in this study capitalizes on skip connections, and the different branches of the architecture serve as macro skip connections that enhance performance. The experiment further demonstrates that considering the complexity measure of the dataset images to define the depth and structure of the CNN model is a valid approach. In addition, the accuracy metrics obtained during the training and validation stages of the TreeNAS-Resnet model, applied to both the CIFAR-10 and CIFAR-100 datasets, are illustrated in Fig. 5. This figure provides a visual representation of the accuracy metrics and helps to better understand the performance of the TreeNAS-Resnet Model on both datasets.

V. Conclusion

In this study, we introduced a two-stage algorithmic approach to address the challenge of designing a Convolutional Neural Network (CNN) model architecture. Our novel technique avoids the need for training a plethora of CNN models to identify an optimal architecture. Instead, we have proposed a unique method for evaluating the complexity of an image, which is not only highly efficient but also low on computational resources. Specifically, the complexity of an image is determined by the complexity tree generated by the application of a split method on the image. The dataset complexity tree is then derived by averaging the complexity trees of all the training images in the dataset. This tree is subsequently employed to design the macro CNN architecture, thereby limiting the search for optimal architectures to a small number of potential designs derived from the dataset's complexity tree. To test the efficacy of our approach, experiments were conducted on two standard im-

TABLE I
Overview of Trained Models

Dataset	Model	Params (M)	Accuracy (%)	Training Time GPU Days
CIFAR-10	TreeNAS-Conv	1.55	93.87	0.16
	TreeNAS-Inception	6.7	88.90	0.25
	TreeNAS-Resnet	5.9	97.35	0.25
CIFAR-100	TreeNAS-Conv	1.6	65.45	0.16
	TreeNAS-Inception	6.9	62.37	0.25
	TreeNAS-Resnet	6.1	72.57	0.25

TABLE II
COMPARISON WITH STATE OF ART NAS MODELS ON CIFAR-10

Model	Accuracy (%)	Training Time GPU Days
Chu [38]	90.07	5
Lopes [16]	92.63	4
Zhang [37]	93.76	2
Sun [13]	95.22	35
Barret Zoph [17]	95.53	10
Zhau Zhong [18]	96.46	90
TreeNAS-Resnet	97.35	0.25

TABLE III
COMPARISON WITH STATE OF ART NAS MODELS ON CIFAR-100

Model	Accuracy (%)	Training Time GPU Days
Lopes [16]	70.10	4
Chu [38]	71.00	5
Zhang [37]	71.11	2
TreeNAS-Resnet	72.57	0.25

age classification datasets, CIFAR-10 and CIFAR-100. The results of our investigations demonstrate that the complexity tree of the dataset can indeed be leveraged to generate the CNN architecture. The performance of our proposed model in terms of accuracy achieved and computation resource requirements surpassed that of the state-of-the-art models. In conclusion, our two-stage algorithm offers a promising avenue for addressing the challenge of CNN model architecture design, particularly in terms of computational efficiency and accuracy.

REFERENCES

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] D. U. Wutsqa, A. M. Abadi *et al.*, "Breast cancer classification using a hybrid model of fuzzy and neural network." *IAENG International Journal of Computer Science*, vol. 49, no. 2, pp. 550–557, 2022.

[4] L. Ronghui and W. Xinhong, "Application of improved convolutional neural network in text classification." *IAENG International Journal of Computer Science*, vol. 49, no. 3, pp. 762–767, 2022.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, may 2017. [Online]. Available: https://doi.org/10.1145/3065386

[6] J. N. Mogan, C. P. Lee, K. S. M. Anbananthen, and K. M. Lim, "Gait-densenet: A hybrid convolutional neural network for gait recognition." *IAENG International Journal of Computer Science*, vol. 49, no. 2, pp. 393–400, 2022.

[7] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[9] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," *arXiv preprint arXiv:1905.01392*, 2019.

[10] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[11] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, no. 1. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 115–123.

[12] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012. [Online]. Available: http://jmlr.org/papers/v13/bergstra12a.html

[13] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing cnn architectures using the genetic algorithm for image classification," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.

[14] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 06–11 Aug 2017, pp. 2902–2911. [Online]. Available: https://proceedings.mlr.press/v70/real17a.html

[15] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 19–34.

[16] V. Lopes, S. Alirezazadeh, and L. A. Alexandre, "Epe-nas: Efficient performance estimation without training for neural architecture search," in *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part V*. Springer, 2021, pp. 552–563.

[17] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=r1Ue8Hcxg

[18] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE Computer Society, 2018, pp. 2423–2432.

[19] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–34, 2021.

[20] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

[21] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97. PMLR, 09–15 Jun 2019, pp. 7105–7114.

[22] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[23] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, p. 3460–3468.

[24] J. Dong, A. Cheng, D. Juan, W. Wei, and M. Sun, "Ppp-net: Platform-aware progressive search for pareto-optimal neural architectures," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=B1NT3TAIM

[25] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[26] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1eYHoC5FX

[27] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 497–504. [Online]. Available: https://doi.org/10.1145/3071178.3071229

[28] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, and O. F. and, "Chapter 15 - evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Academic Press, 2019, pp. 293–312.

[29] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1388–1397.

[30] A. Brock, T. Lim, J. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rydeCEhs-

[31] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=BJQRKzbA-

[32] G. Michel, M. A. Alaoui, A. Lebois, A. Feriani, and M. Felhi, "Dvolver: Efficient pareto-optimal neural network architecture search," *arXiv preprint arXiv:1902.01654*, 2019.

[33] D. Chhachhiya, A. Sharma, and M. Gupta, "Designing optimal architecture of recurrent neural network (lstm) with particle swarm optimization technique specifically for educational dataset," *International Journal of Information Technology*, vol. 11, pp. 159–163, 2019.

[34] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=S1c2cvqee

[35] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[36] Y. Zhou, S. Ebrahimi, S. Ö. Arık, H. Yu, H. Liu, and G. Diamos, "Resource-efficient neural architect," *arXiv preprint arXiv:1806.07912*, 2018.

[37] M. Zhang, S. W. Su, S. Pan, X. Chang, E. M. Abbasnejad, and R. Haffari, "idarts: Differentiable architecture search with stochastic implicit gradients," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 12 557–12 566.

[38] X. Chu, B. Zhang, and R. Xu, "Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search," in *Proceedings of the IEEE/CVF International Conference on computer vision*, 2021, pp. 12 239–12 248.

[39] I. Mario, M. Chacon, D. Alma, and S. Corral, "Image complexity measure: a human criterion free approach," in *NAFIPS 2005-2005 Annual Meeting of the North American Fuzzy Information Processing Society*. IEEE, 2005, pp. 241–246.

[40] J. Rigau, M. Feixas, and M. Sbert, "An information-theoretic framework for image complexity," in *Proceedings of the First Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*, ser. Computational Aesthetics'05. Goslar, DEU: Eurographics Association, 2005, p. 177–184.

[41] M. Cardaci, V. Di Gesù, M. Petrou, and M. E. Tabacchi, "On the evaluation of images complexity: A fuzzy approach," in *Fuzzy Logic and Applications: 6th International Workshop, WILF 2005, Crema, Italy, September 15-17, 2005, Revised Selected Papers 6*. Springer, 2006, pp. 305–311.

[42] S. Yang, P. Gao, F. Meng, X. Jiang, and H. Liu, "Objective image quality assessment based on image complexity and color similarity," in *2013 Fourth World Congress on Software Engineering*, 2013, pp. 5–9.

[43] B. Zhou, S. Xu, and X. xin Yang, "Computing the color complexity of images," in *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2015, pp. 1898–1902.

[44] M. Ivanovici and N. Richard, "A naive complexity measure for color texture images," in *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*, 2017, pp. 1–4.

[45] S. Mishra, D. Z. Chen, and X. S. Hu, "Image complexity guided network compression for biomedical image segmentation," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 18, no. 2, pp. 1–23, 2021.

[46] T. M. Khan, S. S. Naqvi, and E. Meijering, "Leveraging image complexity in macro-level neural network design for medical image segmentation," *Scientific Reports*, vol. 12, no. 1, p. 22286, 2022.

Sajad Ahmad Kawa Sajad Ahmad Kawa received his BCA and MCA Degrees from University of Kashmir. He is currently a PhD Research scholar in the Department of Computer Sciences at University of Kashmir. His research is focused on Deep Learning and CNN. His current research interests include architecture design of CNN, development of efficient CNN models and Deep learning models

M. Arif Wani M. Arif Wani received his Bachelor of Engineering degree from University of Kashmir, Masters in Computer Technology from Indian Institute of Technology, New Delhi, and Ph. D. from Cardiff University, UK. He worked as a postdoctoral researcher at Cardiff University and served as a faculty member in the UK for a period of four years. He served as the director of Machine Learning and Visualization laboratory at California State University. Currently he is serving as Professor in the Postgraduate Department of Computer Sciences at the University of Kashmir, India