Tree-Based Convolutional Neural Networks for Image Classification

Aamir Ahmad Ansari[1], Saba Raees[2], *Nafisur Rahman[3]

aamirahmadansari@outlook.com[1], sweetyraees7@gmail.com[2], nafiis@gmail.com[3]

Master AI & ML Program, Univ.AI[1], MTech. Scholar, Department of Computer Science and Engineering, School of Engineering Sciences and Technology, Jamia Hamdard [2], Assistant Professor, Department of Computer Science and Engineering, School of Engineering Sciences and Technology, Jamia Hamdard[3]

Abstract: In this paper, we explore the use of tree-based Convolutional Neural Networks for image classification problems. The main goal of this image classification technique is to identify the traits, in an image, with precision. In the case of image classification, Convolutional Neural Networks (CNNs) are used because of their high precision. We explore the use of tree data structure to design the architect of our CNN model for image classification. Using n-ary trees we deduce the characteristics and compare them with the state-of-the-art Models. We employ the tiny ImageNet data set prepared by Stanford and the CIFAR-10 dataset [1]. This work shows how increasing the depth and width of a tree-structured CNN compares to the state-of-the-art models in terms of computational benefits, efficiency scores, and resource management.

Keywords: Convolutional Neural Networks, Tree Data Structure, Deep Learning, Classification, Image Classification.

1. Introduction

Convolutional Neural Networks (CNN) have made their own place in the computer vision community since being revived by Yann LeCun, who introduced LeNet [2] in the early ’90s. Since then, more complex and robust CNNs have been introduced and have found their extensive use in the likes of image classification and object detection. Image classification is one of the classic problems in computer vision where, given an image, a computer analyses and identifies it as being a member of one class of several classes. Image classification because of its numerous applications partly is called a classic problem. Self-driving needs fast image classification as a particularly critical primitive. Modern social media and photo sharing and storage applications like Facebook and Google Photos use image classification to boost and personalized user experience on their products.

Leading the rise in the industry as well as in the research field, CNN models with billions of parameters can train networks on extremely large datasets.

ICIDSSD 2022, May 24-25, New Delhi, India

DOI 10.4108/eai.24-5-2022.2318997

Neural networks use graph like data structure widely. Residual Network (ResNet) [3] model is a Convolutional Neural Network (CNN) used in many computer vision tasks used graph data structures but we plan to review how tree-based structures will work for image classification.

2. Literature Review

2.1 Previous Work

In general, the ImageNet challenge is used to benchmark the Convolutional Neural Networks. The CIFAR-10 is another popular dataset collected by the team that proposed AlexNet that destroyed the ImageNet competition in 2012. In the recent past, a transformer-based model ViT-H/14(2020) with 632M parameters achieved the best accuracy score for the CIFAR-10 dataset, followed by CaiT-M-36 U 224(2022) and ResNet (2019). The scores they achieved are given in Table 1.

Table 1. Accuracy score on CIFAR-10 dataset.

Model Name	Accuracy Score
ViT-H/14	99.6
CaiT-M-36 U 224	99.4
BiT-L(ResNet)	99.37

2.2 Contribution

In this paper, we use the power of data structures and use a tree-based Convolutional Neural Network architecture to benchmark on the CIFAR-10 dataset for an image classification problem. We focus on how a tree-based architecture can provide good performance with low computation time and less usage of resources.

The implementation of the Convolutional Neural Network has been done using TensorFlow in Python3. The model with its weights will be provided to the community to work and improve on it.

3. Technical Solution

3.1 Overview

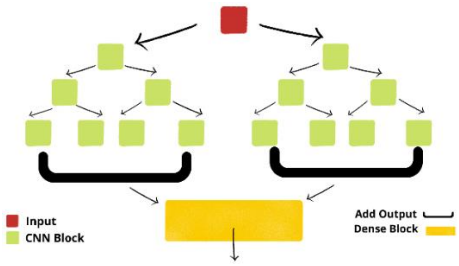
In this section, we will describe our solution and model at a deeper level. We will provide details about the dataset, CIFAR-10. Further, we give a high-level introduction to our model and its construction, describing details on the architecture of the model, the non-linearities, and regularization regularisation techniques used in our model. We define a baseline model by using a binary tree approach and move to a more complex trinary tree-based model.

3.2 Dataset

The dataset that we have used is the popular CIFAR-10 dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton [1]. The CIFAR-10 datasets contain 60,000 images of 10 classes namely dogs, cats, horses, deer, airplane, truck, ship, bird, frog, and automobile. These 60,000 images are then divided into a set of train and test images. The training dataset is a balanced dataset having 5,000 images for each class; it consists of 50,000 images. The other 10,000 images are used as an unseen test set for measuring the generalization of the model. The data set is also available in the TensorFlow dataset API. The size of each image is (32, 32) having a third dimension as for colour channels RGB.

3.3 Model Architecture

We start by building a baseline model, a binary tree approach having 3 levels. The model contained 4,79,402 parameters of which 1,536 were non-trainable due to the use of batch normalization [4]. The model was a complete binary tree where each node had two child nodes, excluding the leaf nodes. The model architecture is as shown in Figure 1.



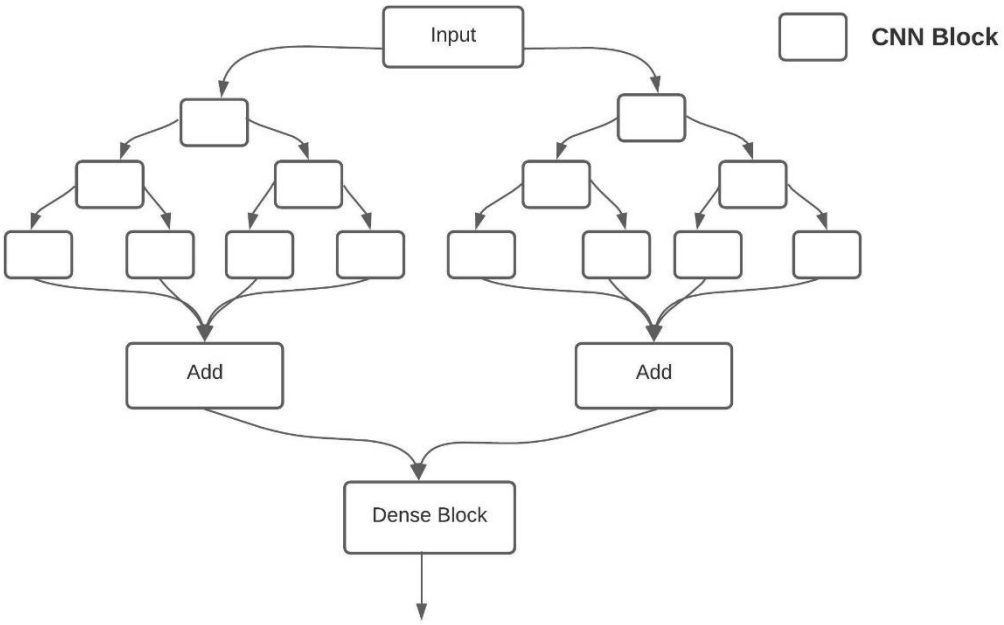


Figure 1. Baseline model architecture

The above network consists of an input of the shape same as the image size with an additional dimension for batch size. The green boxes are convolutional boxes that are structured as shown in Figure 2.

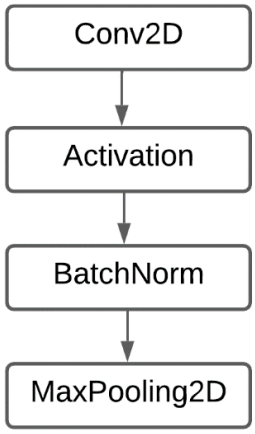


Figure 2. Structure of each CNN block

The initial layer in the block is a convolutional layer, the convolution works in two dimensions. The Conv2D layer consists of 64 kernels of dimension 3*3; 64 outputs (feature maps) are generated using 64, 3*3 kernels where each kernel has 9 weights plus a bias term to be optimized. These feature maps are introduced to non-linearity by the means of the application of the activation layer. ReLU [5] activation has been used for this purpose. ReLU activation provides sparsity in the feature maps which works as a natural regularization technique as the output of a ReLU function for values less than and equal to zero is 0. Another benefit of using ReLU is the well-defined derivatives throughout the domain. It gives an edge over sigmoid activation as we can overcome the vanishing gradients problem while back-propagating. These transformed feature maps are fed to the batch normalization layer, which provides stability in the process as it reduces the internal covariance shift and also helps the model to converge faster. The last layer of this pipeline is the max-pooling layer [30]. We used max-pooling a stride of (2, 2). Max pooling helps to reduce dimensions of the feature maps which also helps in tackling over-fitting.

The output from level 3 of the binary tree is added with respect to the parent they correspond to at level 1. This operation quantifies our information in the feature maps and provides a reduced dimensional output as compared to a concatenation operation hence we save computational time as well as cost. The output of this Addition-layer has the depth equal to the number of feature maps of the last convolutional block and the height and width equal to the reduced dimensions received from the max-pooling layer of the last CNN block. The output from the add operation is then exported to the Dense block. The dense block structure is as shown in Figure 3.

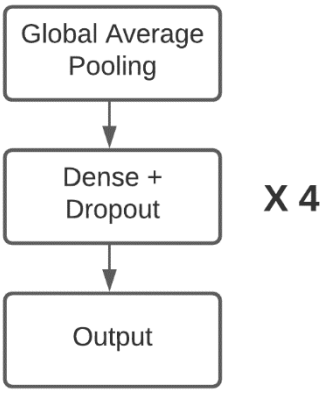


Figure 3. Dense block structure

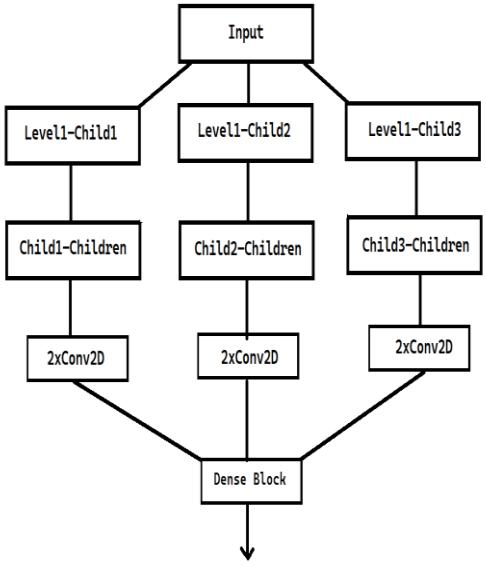
The dense block initially contains the global average pooling [6] layer which provides a scalar, average for each feature map. Then there are four combinations of dense layers and dropout. Dropout helps us to reduce over-fitting as it makes sure that each neuron can learn the information on its own. It randomly drops the specified percentage of neurons in each epoch which provides us with the average output of several unique models created by the random ignorance of neurons in the dense layers. All the dense layers are packed with L1 norm

penalization so the weights do not grow large and sparsity is maintained at the time of generalization as well. The output contains as many neurons as the number of classes with softmax activation as it generates probabilities for each class that sums up to 1.

The loss function used is the categorical cross-entropy. We have used one-hot encoded target values and categorical cross-entropy bodes well with one-hot encoded target and it minimizes the distance between two probability distributions, which is our ultimate goal for an image classification task.

The architecture that we are proposing is a variant of n-ary trees and we use a tri-nary tree instead of a binary tree. There are two levels in this architecture, where the three children of each parent differ from each other as we vary the kernel size, the first child has a kernel size of 3, the second one has 5 and the third one has 7. This provides information to flow through different receptive fields. Level 1 nodes have 128 filters each whereas level 2 nodes are given 64 filters each. The output of the leaf nodes are being added with respect to their immediate parents and passed through two more convolutional blocks, with the same structure as defined above whose results are again added element-wise and passed to the dense block. The convolutional blocks of level 1 and level 2 nodes have exactly the same structure as we used in the baseline model. The architecture shares the structure of dense layers with the baseline model as well.

Adam optimizer [7] was used with learning rate decay; as we reach closer to the minima we do not want to have random walks in the valley due to a large learning rate hence a decay is provided to take smaller steps when near a local minimum. The training conditions for both models were kept the same. We used the early stopping technique to stop before the over-fitting goes beyond control and save the best weights to the model. The model consists of approximately 1.8M parameters. The proposed architecture is shown in Figure 4.



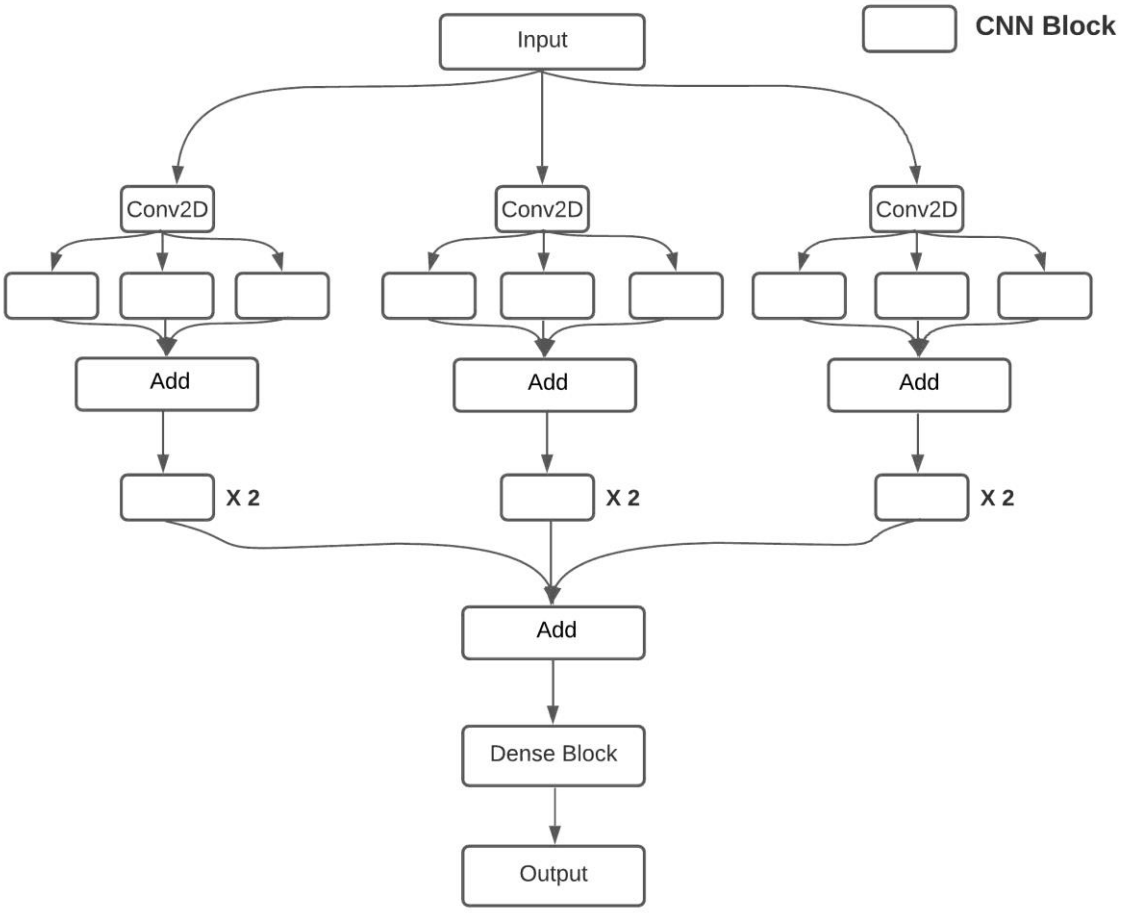


Figure 4. Proposed Model Architecture

The child1, child2, and child3 children blocks contain children of child1, child2, and child3 in which each of these is a 3, 5, and a 7 kernel Conv2D block. The output of child1-children and others are element-wise added feature maps which are followed by two more convolutional blocks for each of them.

4. Results & Discussions

The baseline model registered a training accuracy of 72% and a validation accuracy of 71.04%. The loss plot and the accuracy plots are shown in Figure 5. Notice in Figure 5 the validation loss follows the training loss closely hence the signs of over-fitting were few.

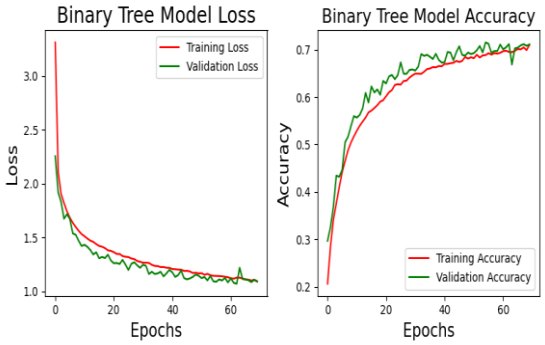


Figure 5. Baseline performance metrics

The early stopping call back prevented any over-fitting in our case, stopping the learning process at the optimal time. The patience value was kept at 7. In the proposed model we achieved greater accuracy and a lower loss metric but over-fitting did become a part of the model. With the use of early stopping the weights were returned to the best weights, as we were monitoring the validation loss, the algorithm intervened when there was no improvement beyond the patience level. The performance metrics can be shown in Figure 6.

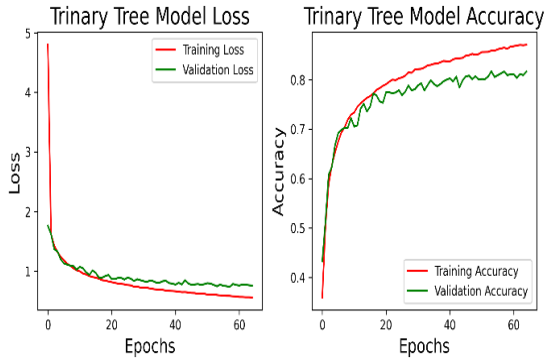


Figure 6. Tree-based CNN performance metrics

The accuracy achieved was 81.14% on the validation set and 87% on the training set.

5. References

[1] Krizhevsky, Alex. (2012). Learning Multiple Layers of Features from Tiny Images. University of Toronto.

[2] LeCun, Yann, Boser, Bernhard, Denker, John S, Henderson, Donnie, Howard, Richard E, Hubbard, Wayne, and Jackel, Lawrence D. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.

[3] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

[4] Ioffe, Sergey & Szegedy, Christian. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

[5] Agarap, Abien Fred. “Deep Learning using Rectified Linear Units (ReLU).” ArXiv abs/1803.08375 (2018): n. pag.

[6] Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv preprint arXiv:1312.4400* (2013).

[7] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[8] Li, Xiang & Chen, Shuo & Hu, Xiaolin & Yang, Jian. (2018). Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift.

[9] Bengio, Y. & Lecun, Yann. (1997). Convolutional Networks for Images, Speech, and Time-Series.

[10] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In ECCV, 2014.

[11] C. M. Bishop. Neural networks for pattern recognition. Oxford University Press, 1995.

[12] R. Girshick. Fast R-CNN. In ICCV, 2015.

[13] G. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In NIPS, 2014.

[14] B. D. Ripley. Pattern recognition and neural networks. Cambridge University press, 1996.

[15] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Le-Cun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR, 2014.

[16] Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3156–3164, 2015

[17] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014

[18] Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014. URL http: //arxiv.org/abs/1409.1556.

[19] Donahue, Jeffrey, Anne Hendricks, Lisa, Guadarrama, Sergio, Rohrbach, Marcus, Venugopalan, Subhashini, Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2625–2634, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M.Bernstein, et al. Imagenet large scale visual recognition challenge. arXiv:1409.0575, 2014.

[23] N. N. Schraudolph. Centering neural network gradient factors. In Neural Networks: Tricks of the Trade, pages 207–226. Springer, 1998.

[24] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010.

[25] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In CVPR, 2007.

[26] Karpathy, Andrej and Fei-Fei, Li. Deep visualsemantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3128–3137, 2015.

[27] Steinkrau, Dave, Simard, Patrice Y, and Buck, Ian. Using gpus for machine learning algorithms. In null, pp. 1115–1119. IEEE, 2005.

[28] Cho, Kyunghyun, Van Merri¨enboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[29] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.

[30] T. Raiko, H. Valpola, and Y. LeCun. Deep learning made easier by linear transformations in perceptrons. In AISTATS, 2012.

[31] Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. In ICCV, 2015.

[32] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. TPAMI, 2012.

[33] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. arXiv:1504.06066, 2015.

[34] Ranzato, M.A., Y.-L. Boureau, and Y.L. Cun. Sparse feature learning for deep belief networks. in Advances in neural information processing systems. 2008.

[35] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).