An Efficient Image Classification Method Using Depthwise Separable Convolution by a Supervised Learning Approach

Md Safayet Islam
Undergraduate Thesis
Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

October 2022

Abstract

Image classification models have been widely used in various fields, from medicine to autonomous driving. While existing models have achieved high accuracy, there is a growing need for a computational and data-efficient model for the real-time classification of objects. This research proposes a new CNN architecture to address this challenge by using depthwise separable convolution. To improve the performance of the depthwise convolution layer, we used two additional 1*1 convolution layers for the reduction of an image's input channels. Additionally, we mixed the spatial dimensions and the reduced channel dimensions using a patch-based representation of the input image with convolution. The residual connections help us to build a deep architecture. We have proposed three different models that are distinguished based on the trainable parameters and training time of the model, keeping in mind the trade-off between accuracy and training time of a neural network. To assess our model's ability to classify images, we used the CIFAR10 dataset. We have achieved some competitive results after experimenting with our models. On the CIFAR10 dataset, our smallest model, with only 130,442 trainable parameters and 0.0325 GFLOPS, has a top-1 accuracy of 93%, while our largest model, with 998,666 parameters and 1.0092 GFLOPS, had an accuracy of 97%.

1 Introduction

Major applications of computer vision include image classification, segmentation, object detection, and localization. Image classification is the most fundamental task and serves as the foundation for many other computer vision challenges. Image classification models are used in medical imaging, object recognition in satellite images, traffic control systems, brake light detection, and other applications. As image classification systems become more widely used, there is a demand for ones that are accurate in identifying objects and lightweight and computationally efficient.

1.1 Motivation

Convolutional neural networks have become the most used machine learning algorithm for recognizing visual objects. Though they were introduced more than 20 years ago, it wasn't until recently that advancements in computer hardware and network architecture made it possible to train truly deep CNNs. VGG [1] had 19 layers, the original LeNet [2] had 5, and Residual Networks (ResNets) [3] broke the 100-layer barrier. The parameters' size and the training data volume have increased in tandem with the adoption of Transformers [4] as the benchmark for language processing and their improvements in computer vision. Recent research has demonstrated that models based on transformers, most prominently the Vision Transformer (ViT) [5], would perform better in particular circumstances. More crucially, pre-trained on the huge JFT-300M dataset with weak labels, ViT produces results in comparison to those of state-of-the-art (SOTA) ConvNets, suggesting that Transformer models may be more scale-efficient than ConvNets. Nevertheless, ViTs cannot be extended to bigger dimensions through the use of patch embeddings, which combine small portions of the picture into a single input element. This is because the self-attention stages in Transformers have quadratic complexity. Additionally, convolutional networks (ConvNets) may exhibit some beneficial inductive biases that plain Transformer layers may not, necessitating the use of a large quantity of data and computer power to make up for this. Much recent research has attempted to integrate the inductive biases of convolutional networks into transformer architecture by imposing local receptive fields for attention layers or enhancing the attention and feed-forward network layers with definitive or implicit convolutional operational processes. These approaches, however, are either ad hoc or concentrated on infusing a specific characteristic and, therefore, lack a comprehensive knowledge of the relative functions of convolution and attention while coupled. It is difficult for everyone to possess a large dataset or have large computing resources due to the fact that computer vision is being employed in many different sectors and on many different platforms. Therefore, the requirement for an image classifier with lower complexity and acceptable accuracy exists. In this thesis, we looked into patch-based image representation using depth-separable convolution for image classification. To create a novel image classification architecture with fewer parameters and better accuracy, we experimented with the usage of Depthwise Separable Convolution [6] in this study.

1.2 Literature Review

1.2.1 Case 1

Krizhevsky et al. [7] have suggested using five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers as ConvNet architecture. To incorporate non-linearity, the suggested model employed the Rectified Linear Activation (ReLU) Activation Function rather than Tanh activation. The dropout regularization technique was also applied to deal with overfitting training data. The network size was decreased by using the Max Pooling Layer. This model has relatively little depth, which makes it difficult for it to learn features from image sets. There are just eight learnable layers in it. Compared to contemporary ConvNet models, it takes longer to obtain results with higher accuracy. The network design has 62.3 million learnable parameters, which is a very high quantity. This model's accuracy on the

CIFAR10 dataset is about 80.80%.

1.2.2 Case 2

Simonyan et al. [1] suggested using a small (3*3) filter size for convolution layers to increase the depth of the CNN. They suggested multiple CNN models of various sizes built on the same architecture. VGG-16 has 16 layers, and VGG-19 has a depth of 19 layers. Non-linearity increased as the number of layers with smaller kernels increased. The Vanishing Gradient issue exists in that proposed CNN model. VGG-19 takes a very long time to train compared to the more recent ResNet Architecture. The network design features 143.7 million learnable parameters, which is a very high quantity. The model's accuracy on the CIFAR10 dataset is 91.2%.

1.2.3 Case 3

He et al. [3] have recommended employing skip connections to deepen a network while avoiding vanishing gradient issues. By including residual Block, they were able to address the Deep Neural Network degradation issue. By doing so, they were able to create a model that was considerably deeper and did not have the exploding/vanishing gradient problem. To reduce overfitting, they utilized batch normalization. The complexity of the architecture was boosted by the deep model. Later, it was discovered that accuracy reached saturation with an increase in data. The ResNet-50 model on an NVIDIA M40 GPU requires 14 days to train to complete 90 epochs on the ImageNet dataset. There are 25.6 million learnable parameters in the network design. On the CIFAR10 dataset, the model's accuracy is 95.84%.

1.2.4 Case 4

Dosovitskiy et al. [5] proposed using Self-Attention instead of Convolution layers for image classification. The various components of an image are represented via patch embedding. No convolution was utilized; hence, there was no bias in the architecture that was image-specific. With more data available, they observed an improvement in the model's accuracy. However, when the model was trained on insufficient amounts of data, it did not generalize effectively. For the model to achieve state-of-the-art accuracy, 300M images were used in training. This architecture has a Large CO2 footprint because it requires about 30 days to train a ViT on an 8-core, typical cloud TPUv3. On the CIFAR10 dataset, the model's accuracy is around 90%.

1.3 Thesis Objective

1. Select a suitable dataset to evaluate the effectiveness and accuracy of the suggested model.
2. Create a new architecture for deep learning-based image classification.
3. Reduce the model's parameters while maintaining accuracy to improve it.
4. To analyze the impact of different hyperparameters on the output of the model.
5. To analyze the impact of different optimizers, loss functions, and activation functions.
6. Use various state-of-the-art training paradigms to get the highest possible accuracy.

7. Compare the proposed model with other deep learning image classification models.
8. Obtain classification reports for diverse datasets of images from different domains.
9. Reduce the training time of the model using various techniques.

2 Methodology

To improve the accuracy of our model, we adhered to a few well-suited methodologies that were most appropriate for our research. This includes picking the right dataset, applying the right preprocessing and augmentation methods, and using the right layers, kernels, and convolution layer filter sizes. We also adhered to standard practices when training a deep learning model, such as comparing various learning rate schedulers and selecting an appropriate optimizer after careful consideration. Due to all of these choices, we can obtain a competitive accuracy on the specified dataset with fewer training parameters and a simpler computational requirement.

2.1 Network Architecture

The building block of our classification model is the convolution layer. We used depthwise separable convolution in our model. In addition, we also used Residual Skip Connections to prevent vanishing gradient problems and to help the network converge faster.

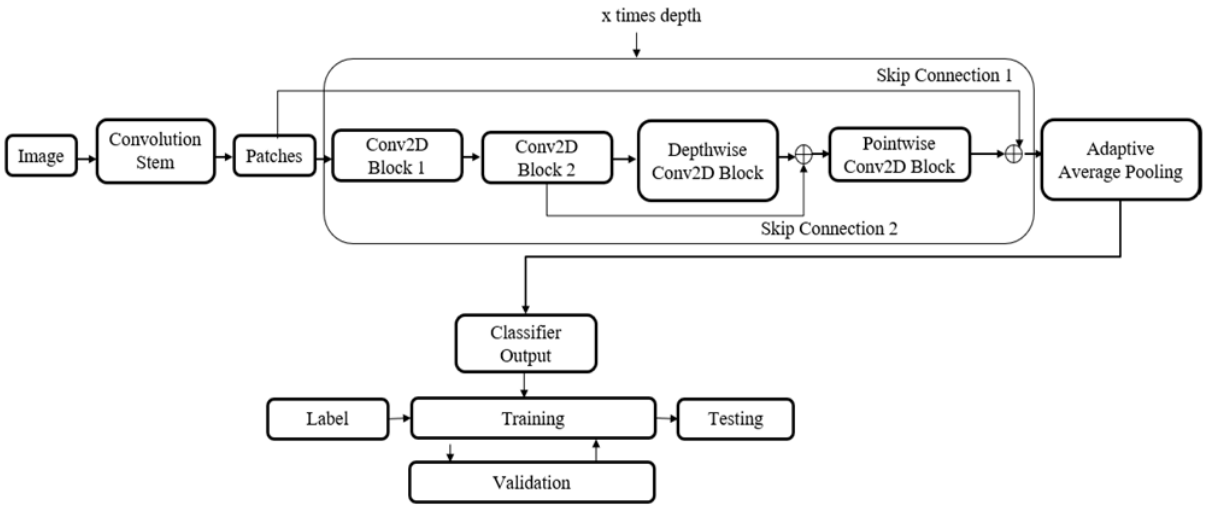

Figure 1: Proposed CNN Model Architecture

2.1.1 Convolution Stem Block

Over the initial image, the stem layer functions as a compression technique. As a result, the spatial extent of the activations is quickly reduced, which lowers memory and computing costs. Patches of the image are then separated. A convolution layer with c input channels and h output channels, kernel size, and stride equal to the patch size make up this layer. The projection dimension of the patches is indicated by the number of filters employed in this layer. An activation function and a post-activation batch normalization [8] come afterward. The model's initial element is this layer. The output of this layer can be represented as:

$$z_0 = BN(\sigma Conv_{p \to h}(X, \text{stride} = p, \text{kernelsize} = p)) \tag{1}$$

2.1.2 Image Patches

The images were divided into several patches that were of the same length and width, **P**.

$$x \in \mathbb{R}^{H \times W \times C} \Rightarrow x \in \mathbb{R}^{N \times P^2 C} \tag{2}$$

Where,

- $N = \frac{HW}{P^2}$ is the number of patches
- **P** is the height and width of the patch with a patch dimension (**P, P, C**)
- **C** is the number of channels

The stride of the convolution stem layer works as a hyperparameter as it determines the size of the image patches.

2.1.3 Con2D Block 1

The stack of convolution layers utilized in each block starts with this layer. This layer serves as a channel downsizing block. The filter size functions as a hyperparameter while the kernel size of the convolution layer is held constant at 1. The multiplication required for the subsequent layers is minimized by narrowing the channel. The kernel size half of the projection dimension produces the best results for the CIFAR10 dataset. The convolution layer is followed by the GELU [9] activation layer and then the batch normalization layer.

2.1.4 Con2D Block 2

The convolution block after that functions as a channel dimensionality reduction method as well. Here, the filter size is set to one-quarter of the projection dim, and the kernel size is left at 1. Similarly, batch normalization and GELU activation are applied after this convolution layer.

2.1.5 Depthwise Convolution Block

Depthwise convolution is performed over the reduced channel. The depthwise convolution layer's kernel size is crucial to the model's accuracy. A bigger kernel size promotes information exchange between adjacent pixels but also raises the cost of calculation. In the suggested architecture, it has been discovered that the kernel sizes of 5 and 7 offer a fair balance between precision and complexity. A residual connection is established between the output of Conv2D block 2 and this output for efficient information transfer after the activation function and batch normalization.

2.1.6 Pointwise Convolution Block

A 1x1 kernel is used in the Pointwise Convolution method, which iterates over every point. The depth of this kernel is the same as the number of channels in the input image. To construct

depthwise-separable convolutions, it is combined with depthwise convolutions. The filter size, in this case, is maintained at the same level as the patch projection dimension. Another residual link is formed between the input of Conv2D block 1 and this block's output after activation and normalization have been completed.

3 Results & Discussion

The experimental findings of our study, which were obtained after classifying the dataset using the methods we suggested, will be discussed in this part. Additionally, a thorough explanation of the experimental design and steps used to arrive at the results will be provided. To assess the outcomes, we made use of the TensorFlow [10] and scikit-learn [11] libraries. The output is visualized using the Matplotlib [12] Library. The key concepts for analyzing our findings are covered first.

3.1 Experimental Setup

Without any pretraining or supplementary data, we assess our suggested model largely using the CIFAR10 [13] dataset. Typical data augmentation methods like RandAugment [14], AutoAugment [15], MixUp [16], and CutMix [17] were employed. We employed an exponential learning rate scheduler [18] and the AdamW [19] optimizer. The learning rate for the AdamW optimizer started out at 1e-5 and rose to 0.01 after five warm-up epochs. After that, the learning rate dropped gradually to 5e-5. We used a predetermined number of epochs for our experiments. Our models were trained over 300 epochs. We just did a minimal amount of hyperparameter tuning due to restricted computing resources. As a result, our models might be either over or under-fitted, and the accuracy we claim probably understates our model's potential. We executed all the experimentations on the Kaggle notebook with a TPU v3 hardware accelerator.

3.2 Model Specifications

We propose three different convolutional models based on the same architecture. The patch size, the kernel size of the depthwise separable layer, the projection dimension of the patches, and the depth of the model are differentiators among these models. We call these there versions Small, Medium, and Large based on the training parameters of these models. Although each type of model has a different patch projection dimension, the ratio of channel reduction before each Depthwise separable convolution was kept the same. Each of the models has two 1*1 convolution layers for shrinking the channel size of the image before depthwise convolution, which resulted in fewer multiplicative and memory-intensive parameters. This also reduces the FLOPS of the model.

Table 1: Model Specifications.

Model Size	Patch Size	Projection Dim	Depth Conv. Kernel Size	Model Depth	Trainable Parameters	Flops
Small	2	128	5	8	130,442	0.0325G
Medium	2	256	5	8	490,250	0.1238G
Large	1	256	7	16	998,666	1.0092G

3.3 Experimental Results on the CIFAR10 Dataset

Table 2: Accuracy on the CIFAR10 dataset.

Model Size	Trainable Parameters	Training Accuracy	Testing Accuracy	Epochs	FLOPS
Small	130,442	71%	93%	300	0.0325G
Medium	490,250	76%	95%	300	0.1238G
Large	998,666	82%	97%	300	1.0092G

3.4 Discussion

This chapter has covered our research on CIFAR10 image classification based on our proposed image classifier. Our three suggested models, which are based on the same architecture, have been tested to see how well they perform. We have observed that the model's accuracy increases with the increase in the depthwise separable convolution kernel size. However, excessive kernel sizing has negative effects on model parameters, FLOPS, and training time. In our experiment, shrinking the channels of the image before depthwise convolution resulted in fewer multiplicative and memory-intensive parameters. We have also observed the significance of patch size on the model's overall classification accuracy. We have found that the accuracy of these models is highest when a patch size is as small as 1. The model's performance was also significantly influenced by the depth of the model. Although adding depth increases complexity, the improvement in accuracy shows that the added depth is not likely to cause the model to overfit. The model was not overfitted because of the extensive use of augmentation techniques that served as regulation techniques; instead, testing accuracy was higher than training accuracy. This demonstrates that our model is more broadly generalizing the training data set. So, the proposed model might be ideal in the case where the training data amount is small and the computational budget is limited.

4 Conclusion & Future Work

4.1 Conclusion

In this research, we applied supervised learning to the problem of image classification. The primary goal of this research was to develop a new image classification architecture with fewer parameters and complexity. In comparison to other larger CNN or Vision Transformer [5] models,

our models may not achieve state-of-the-art accuracy, but the experimental data demonstrates a good speed-accuracy tradeoff. A two-stage 1*1 convolutional layer, the main building block of our network, reduces the input channels for depthwise convolution. We can achieve fewer parameters and a shorter training time thanks to this reduction in dimensionality. We have employed pointwise convolution in an unconventional manner. The pointwise convolution typically employs the same number of filters as the depthwise convolution of the preceding layer. However, we suggested using pointwise convolution to expand the input channel's dimension. By doing so, we can maintain a residual connection to the input before the dimension is reduced. As a result, the flow of information remains constant. We have observed a 3-7% increase in accuracy when using Batch Normalization [8], despite Layer Normalization [20] becoming more and more popular in Transformer, MLP, and even CNN models. The use of the AdamW [19] optimizer instead of the Adam [21] optimizer resulted in a 2-3% increase in accuracy. The regularization techniques used aided in preventing overfitting. The findings indicate that none of the models were overfitted. One of the most significant key findings of our experiment is that the patch-based image representation method introduced in Vision Transformers also aids the performance of the CNNs. Although positional encoding is necessary for Vision Transformers [5], our model did not require it because our architecture did not call for self-attention. Accuracy is improved by convolution's inductive bias over the image patches. Another finding of our experiment was the use of batch size. The accuracy was lower when we used a batch size of 128 than when we used a batch size of 512, even though the smaller batch size should increase accuracy. As a result, we kept the batch size at 512 for all of our tests. It might be the result of using Cloud TPUs for our training rather than GPUs. As the use of AI in our daily lives grows, so does the demand for compact image classifiers. The internet has made data shortages less frequent, but some edge cases still need an effective image classification model with a good complexity-accuracy tradeoff. We are witnessing a rapid rise in computer vision in many industries, including the medical sector, satellite imaging, agriculture, and medicine, but the amount of data available in these industries is insufficient. We anticipate that our suggested architecture will serve as the foundation for many computer vision tasks in these fields, including segmentation, object detection, and classification.

4.2 Future Work

The CIFAR10 dataset, which has images with a size of 32*32, served as the basis for the entire research process. High-resolution image data was not available for us to test our model on. This resulted from the fact that we had limited computational resources. Additionally, we limited the testing of our suggested model to image classification tasks. Therefore, testing and evaluating our model on large, high-resolution image datasets like ImageNet would make up the majority of our study's future work. We did not evaluate the effectiveness of our model in other computer vision fields. Future research can also be extended to evaluate how well our model performs on other computer vision tasks like object detection and image segmentation. The efficiency of the model as a whole could have been increased by further hyperparameter tuning. To put our theoretical research into practice, we want to implement our classification model on edge devices like smartphones for real-time image classification.

References

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

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[4] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[6] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1800–1807.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[9] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[10] T. Developers, *Tensorflow*, 2022.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2012.

[12] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[13] A. Krizhevsky, *Cifar-10 and cifar-100 datasets*, `https://www.cs.toronto.edu/~kriz/cifar.html`, 2009.

[14] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," *arXiv preprint arXiv:1909.13719*, 2019.

[15] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[16] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[17] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," *arXiv preprint arXiv:1905.04899*, 2019.

[18] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," *arXiv preprint arXiv:1910.07454*, 2019.

[19] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[20] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.