Analysing Efficiency and Accuracy Trade-offs in CNN Architectures for CIFAR-10 Classification

Sanchit Bakshi
The University of Adelaide
Adelaide, Australia
sanchit.bakshi@adelaide.edu.au

Abstract

In this paper, the study explores the Convolutional Neural Networks(CNN's) for image classification on the CIFAR-10 dataset. We have implemented several CNN model architectures and compared their performance (accuracy) with other architectures: Basic CNN, Resnet-18, MobileNetV2 and Alexnet. Different strategies of data preprocessing and training are used to tune the optimal performance, including hyperparameter tuning. The results show how different architectures have different complexity, which affects classification performance, resulting in high accuracy by the models. This study shows the importance of model selection and its performance.

1. Introduction and Problem Statement

Image classification is one of the fundamental tasks in computer vision crucial for a wide range of tasks, including object detection, face recognition etc[2,7,12]. CNN has become one of the leading architectures for image-related tasks due to the hierarchical feature representation having the ability to learn[1,8,13]. This research aims to classify different CNN architectures using the CIFAR-10 dataset, comparing them on their different accuracy as each has its own unique advantages and trade-offs in terms of complexity, efficiency and performance.

Our objective is to evaluate different models of Convolutional Neural Networks(CNN's) architectures - BasicCNN, ResNet-18, MobileNetV2, and AlexNet. We aim to explore these different architectures performance and optimising by doing hyperparameter tuning and applying the transformations to enhance generalisation.

2. Methodology

In this section, we will explain the methodology used in the research study, including about the dataset and data transformation.

2.1. Dataset

The CIFAR-10 dataset was developed by Alex Krizhevsky and Geoffrey Hinton[2]. This dataset consists of 60,000 images where 50,000 labelled images are training samples and 10,000 images are test samples across 10 classes ('airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'). Each image is a 32 X 32 pixel RGB image. In this dataset, each image represents a single object in a frame having limited resolution with various orientations.

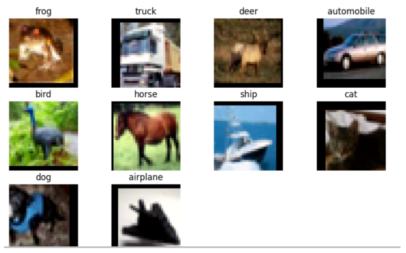

Figure 1. Sample images from the CIFAR-10 dataset with their respective labels.

2.1.1 Data Split

The training data consisting of 50,000 images was further split into training and validation subsets with 80-20 split ratio. The test data remains unchanged for tracking model performance on unseen data.

2.2. Data Transformation

- Random Cropping: Images were cropped by adding a 4-pixels around each image and randomly cropping a 32 X 32 section of the image, which helps in generalising the model better [11].

- Random Horizontal Flip: This was used to add left-right variation by flipping images with a probability of 0.5 .

- Normalisation: It was done by standardising each color channel(RGB) using its mean: [0.4914, 0.4822, 0.4465] and std: [0.2023, 0.1994, 0.2010]. This ensures the pixel values are centered around zero and scaled accordingly [15]. The normalisation formula is

$$x_{\text{normalised}} = \frac{x - \text{mean}}{\text{std}} \tag{1}$$

  where mean and std are the values for each channel as listed above.

3. Model Architectures

In this section, we will explain the different model architectures used in the research study, including different terminologies in CNN along with Basic CNN, Resnet-18, MobileNetV2 and Alexnet.

3.1. Different Terminologies in CNN

- Convolutional Layer: It is a layer that applies a learnable filter (known as kernel) to the input images for the extraction of features. This layer is responsible for detecting patterns.

- ReLU Activation: This activation function is applied to introduce non-linearity, allowing only taking the positive values and making the negative values zero. It is defined by

$$f(x) = \max(0, x) \tag{2}$$

- Pooling Layer: This layer aims to minimise the spatial dimension of feature maps, improving computation. We have used max-pooling during this research. It is defined as

$$P(i, j) = \max\{S(x, y) \mid (x, y) \in \text{receptive field of } (i, j)\} \tag{3}$$

  where P(i, j) is the max value in a region

3.2. Basic CNN

A Convolutional Neural Network is a class of deep learning model phenomenally useful for structured grid data like images and most powerful applications within computer vision. CNNs are designed to utilise the spatial structure present in an image; thus, they detect and learn features from edges and textures to complex and abstract shapes [2,7]. Unlike classic neural networks that connect each neuron, CNNs use a number of filters or kernels on small subregions of the image, maintaining the spatial relations and at the same time reducing the number of parameters greatly. This is the reason why it is computationally efficient and scalable even in cases of large images or large datasets.

The CNN architecture is typically made up of several key layers: convolution, pooling, activation functions such as ReLU, and fully connected layers. The network applies filters in a convolutional layer on the input image, extracting low-level features like edges or gradients of simple textures. Further layers capture higher-level patterns-for instance, combinations or endpoints of edges. Successive layers of pooling, and often more specifically max pooling, perform downsampling in order to reduce the size of the feature maps and retain only the most important features. After some time, this downsampling makes the network invariant to small variations and translations of the inputs. The output from the feature extraction process is usually flattened to a one-dimensional vector and fed into fully connected layers for the final classification. These get stacked, hence CNNs learn varieties of images hierarchically. This enables them to perform exceptionally well on such tasks as image classification, object detection, and segmentation.

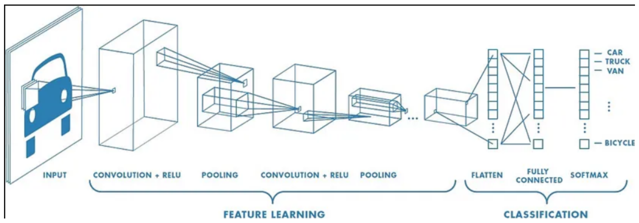

Figure 2. CNN architecture for image classification

The Basic CNN used during this research consists of 2 convolutional layers, the first layer consists of 32 filters and the second has 64 filters, both having 3 X 3 kernel and ReLu activation. After that, it has a max pooling layer. Finally, there are two fully connected layers with 128 hidden layers and 10 outputs.

3.3. ResNet-18

The architecture used during this research uses residual blocks which has two convolutional layers with another shortcut connection that bypasses certain layers to add the input directly with its output [1]. Mathematically, it is defined as

$$y = F(x) + x \tag{4}$$

where,
x is residual block in the network.
$F(x)$ is the residual function learned by the block.
y is the output
It uses multiple convolutional layers where the first layer applies a 7 X 7 convolution followed by ReLu and max pooling, rest layers use a 3 X 3 convolution within the residual block. This model uses global averaging pooling rather than the traditional fully connected (Fc) layers, to reduce each feature map to a single value, which is then fed to the FC in the final later for classification. The final layer output here has 10 classes, in our CIFAR-10.

3.4. MobileNetV2

The architecture followed here is the depthwise separable convolution, it filters out on a single channel input [5], significantly reducing the complexity. A 1 X 1 convolution filter combines the depth-wise convolution with its output.

3.5. AlexNet

This architecture includes several layers starting with the convolutional layers, it has 11 X 11 (needed to adjust the 32 X 32 size in the CIFAR-10 input dataset), followed by ReLu function. Rest of the layers used 5 X 5 and 3 X 3 filters with ReLu after each layer. It also uses max pooling to extract the relevant features from each feature. It has a dropout layer which is used in the fully connected (FC) layers to prevent overfitting. "Dropout" randomly deactivates a fraction of neurons to zero during each forward pass, this helps to improve generalisation.

4. Method Implementation

In this section, we will explain the method implementation used in the research study, including device configuration, training parameters, optimisers and hyperparameter tuning.

4.1. Device Configuration

During this research, all models were trained using GPU acceleration, as a result of the computation was faster which allowed us to take larger batch sizes. PyTorch's CUDA support was leveraged for this efficient computation.

4.2. Training Parameters

Each model was trained by keeping the weights = None, allowing us to randomly initialise weighs, which allows us to evaluate and learn features specially from CIFAR-10 without any pre-trained biases.

The following parameters were consistently used throughout the training models, so as to establish some controlled baseline for experimentation:

- Batch Size: A batch size of 128 was a trade-off between memory consumption and the speed of training. This batch size will enable the GPU memory to maintain stablility.
- Epochs: Both baseline as well as hyperparameter tuning spanned at 10 epochs.
- Loss Function: CrossEntropyLoss was used due to the fact that it is suitable for multi-class classification problems [12]. It calculates difference between predicted prob and actual class labels.

4.3. Optimisers

In this research, they were two optimisers used, depending on the architecture. The optimisers used are:

- Stochastic Gradient Descent (SGD): It is a traditional optimiser that updates the model weights based on the calculation of the gradient of the loss function [19]. SGD was chosen for simpler architectures.

Model used: BasicCNN, OptimisedCNN, Baseline ResNet-18 and MobileNetV2

- Adam optimiser: This algorithm combines the advantage of both SGD with momentum and RMSprop. It adapts the learning rate individually form each parameter. Adam was chosen for complex and adaptive architectures where layer specific would contribute faster and overall performance is enhanced.

Models Used: Optimised ResNet-18 and MobileNetV2, AlexNet

4.4. Hyperparameter Tuning

After initial training, multiple parameters are tuned to enhance performance of the model.

- Learning Rate Scheduler: This dynamically decreases the learning rate when a model plateaus in terms of validation accuracy [15]. It helps in stabilising the training process, which is essential in deeper networks.
- Weight Decay: Among 0.0005, a regularisation parameter was set up because the penalty of larger weights will also avoid overfitting and it will try to use smaller weights.
- Dropout: AlexNet and Optimised CNN used dropout layers in all fully connected layers. This helps in generalizing better because it randomly kills neurons during training. The idea behind this technique is that it compels the model to be less dependent on a particular feature of a neuron, hence making the model less prone to overfitting.

5. Experiments and Analysis

In this section, we will explain in depth analysis of experiments conducted on different CNN architecture: BasicCNN, ResNet-18, MobileNetV2, and AlexNet.

5.1. Basic CNN (Baseline and Optimised CNN Model)

The Basic CNN model has only two convolutional layers saturated rather quickly in terms of its ability to learn complex features. It reached a final training accuracy of 52.39%

Model	Configuration	Training Accuracy	Validation Accuracy	Test Accuracy
BasicCNN	Baseline	52.39%	52.67%	55.45%
	Optimised	73.32%	74.12%	75.94%
ResNet-18	Baseline	59.37%	59.44%	61.84%
	Optimised	69.64%	68.70%	69.71%
MobileNetV2	Baseline	44.47%	45.15%	46.82%
	Optimised	57.34%	57.78%	60.23%
AlexNet	Baseline	44.31%	46.80%	48.54%
	Optimised	77.26%	75.91%	77.66%

Table 1. Training, Validation, and Test accuracies for each model in both Baseline and Optimised configurations.

and a validation accuracy of 52.67%. This corresponded to a test accuracy of 55.45%, which means generalization was rather poor given such a simple architecture. The learning curve for training and validation accuracy increases gradually, but it starts to flatten out at around epoch 8, thereby indicating that the model has a very limited capacity to capture intricate patterns in CIFAR-10 images.

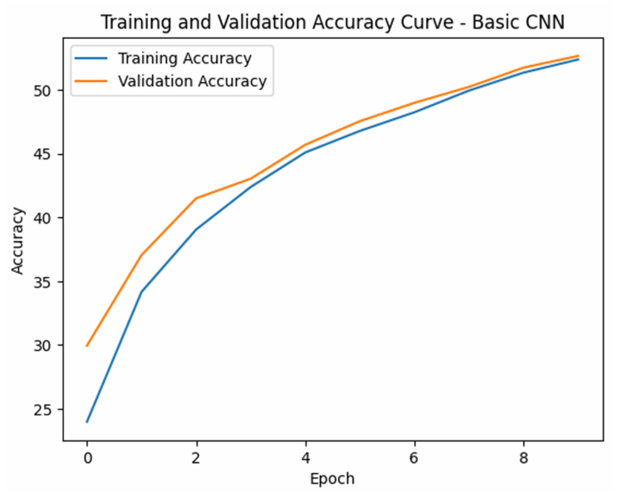

Figure 3. Training and Validation Curve for Basic CNN

To enhance the performance, batch CNN, normalisation and dropout were added for better performance.These further improvements allowed Optimised CNN to achieve increased training accuracy of 73.32% and validation accuracy of 74.12% at the end, while its test accuracy went up to 75.94%, thus demonstrating the benefits of regularisation techniques in action. Optimised CNN has a far more stable and higher increase in both training and validation accuracy, compared to Basic CNN which can be seen in the graph.

5.2. Resnet-18

In the baseline Resnet-18 model, we achieved a training score of 59.73% and a validation accuracy of 59.44%. This model achieved an test accuracy of 61.84% on baseline. The learning curve shows more stable improvement across epochs due to smaller fluctuation between training

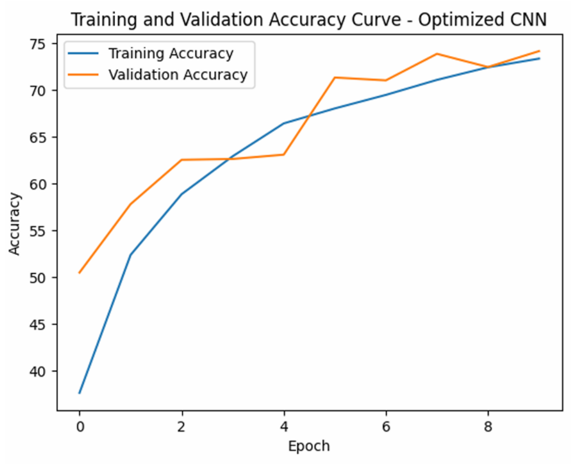

Figure 4. Training and Validation Curve for Optimised CNN

and validation accuracy.

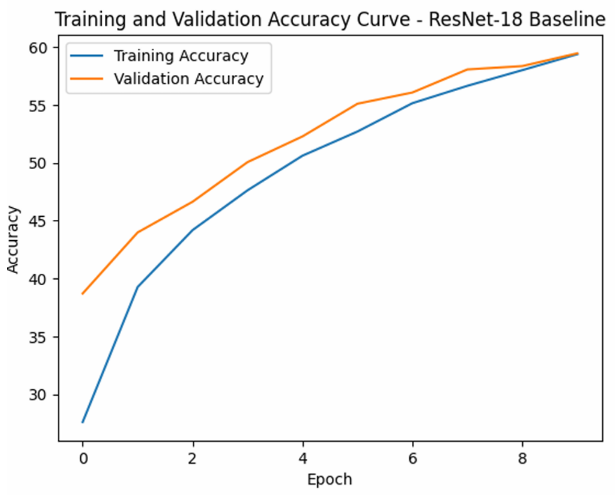

Figure 5. Training and Validation Curve for Resnet-18 Baseline

To enhance the performance of Resnet-18, an Adam optimiser was used in place of SGD with a learning rate of 0.01 and weight decay of 0.0005.Performance after Optimisation had a ResNet-18 training accuracy of 69.64%, a validation accuracy of 68.70%, and a testing accuracy of 69.71%. The optimised learning curve showed an improvement as training and validation accuracy monotonically increased and converged to higher values from the baseline which can be seen in the graphs. Optimisations allowed ResNet-18 to make more use of its deeper structure.

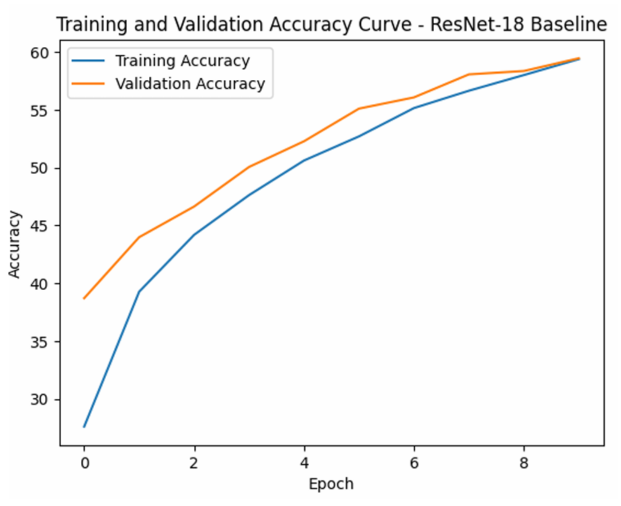

Figure 6. Training and Validation Curve for Optimised Resnet-18

5.3. MobileNetV2

In the baseline setting, MobileNetV2, achieved a training accuracy of 44.47% against a validation accuracy of 45.15%, while it achieved a test accuracy of 46.82%. MobileNetV2's compact design brought in reduced capacity and hence lower performance at the beginning. Training and Validation Curve for the baseline learning curve of MobileNetV2 increased much more gradually, while its accuracy was low on both training and validation sets.

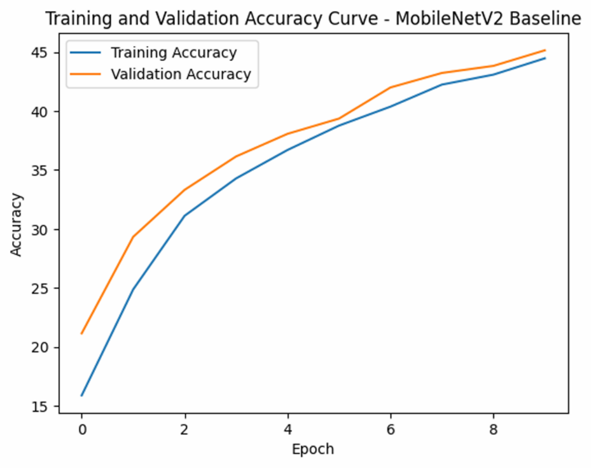

Figure 7. Training and Validation Curve for Baseline MobileNetV2

To improve its performance even further, MobileNetV2 was combined with an Adam optimiser with a learning rate of 0.01 and weight decay of 0.0005. After optimisation, MobileNetV2 obtained a training accuracy of 57.34% and a validation accuracy of 57.78%, with an overall test accuracy of 60.23%. However, the optimised learning curve also had a higher accuracy, as opposed to that of the baseline. Indeed, the Adam optimiser helped the network converge much smoother and more successfully in MobileNetV2. But with regard to performance results, it still lagged behind a deeper network, ResNet-18 and AlexNet, due to its small capacity.

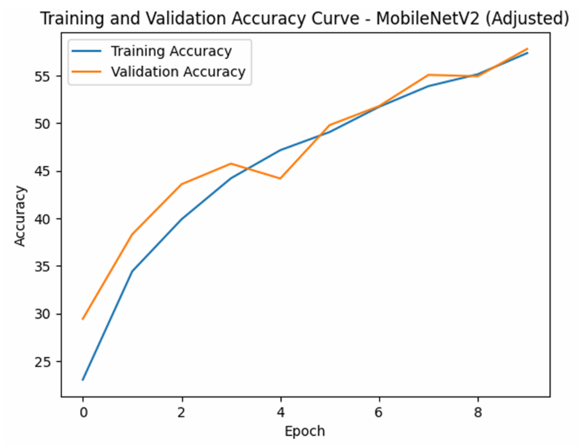

Figure 8. Training and Validation Curve for Optimised MobileNetV2

5.4. AlexNet

A baseline training accuracy of 44.31% and a baseline validation accuracy is 46.80%. However, its test accuracy turned out to be 48.54%. As it has a high capacity, the base model showed a lot of promise but required regularisation to avoid overfitting. The learning curve at baseline for AlexNet showed gradual improvement in accuracy, the model was initially overfitting, as shown by the gap in training and validation accuracy.

To further improve its performance, dropout on fully connected layers was done to avoid overfitting, while we changed the learning rate that was increased to 0.01 with weight decay at 0.0005. Performance after Optimisation: The best performance of AlexNet reached a training accuracy of 77.26%, a validation accuracy of 75.91%, and a test accuracy of 77.66%, which was the highest among all models. The optimised learning curve showed a good rise in the increase in both training and validation accuracy concerning the base, while dropout effectively prevented overfitting. Also, the high trend of accuracy continued right up until the end, reflecting improved generalization of the model.

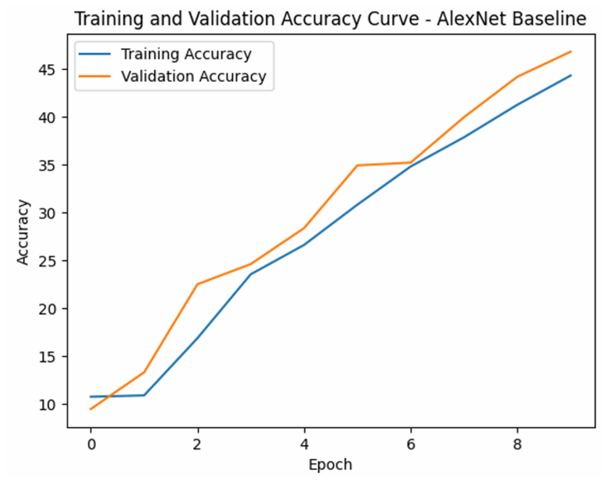

Figure 9. Training and Validation Curve for Baseline AlexNet

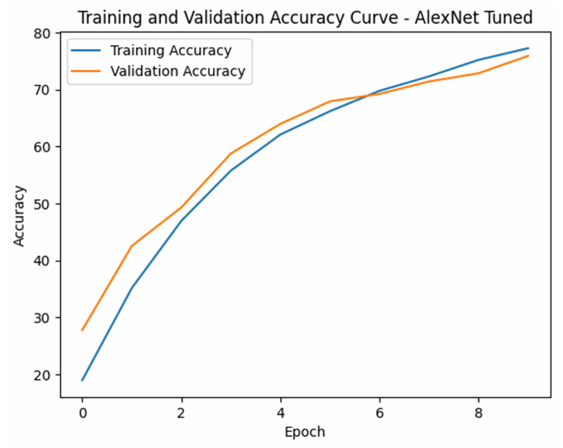

Figure 10. Training and Validation Curve for Optimised AlexNet

5.5. Limitations

There are some limitations found during this research:

- Overfitting in Deeper Models: Deep models such as AlexNet and ResNet-18 tend to overfit if not regularised well enough to provide good generalisation [8].

- Computational Complexity: AlexNet and ResNet-18 resource comsumption makes them less feasible for low-resource or mobile environments.

- The trade-off between Efficiency and Accuracy: Though very efficient, MobileNetV2 comes at a certain cost regarding accuracy [5,20], which is undesirable in applications needing high precision. Shallow models have a limited feature learning capability.

- Dependency on Hyperparameter Tuning: Most of the models require very deep tuning of their hyperparameters, which can be very time-consuming [12].

- Challenges in Generalisation: Models initially trained on CIFAR-10 shall find it difficult should there be a necessity to generalise to other datasets without additional adaptation or transfer learning.

6. Conclusion

This work analysed the performance of BasicCNN, Optimised CNN, ResNet-18, MobileNetV2, and AlexNet architectures. Each one was analysed with baseline performance to find a modification in which certain optimisations due to batch normalisation, dropout, weight decay, and learning rate adjustment resulted. Among the more interesting results from the performed analyses are:

- Generalisation and Model Deepness: The deeper the model, in fact, the more it encompasses regularisation techniques, too. For instance, very good results with strong generalisation were given by ResNet-18 or AlexNet while AlexNet with dropout and higher learning rates showed the best answer with a test accuracy of 77.66

- Residual connections: ResNet-18 availed residual connections for facilitating the flow of gradients efficiently across the layers during training, thus working well for much deeper networks too. In this way, ResNet-18 attains a test accuracy higher than shallower models, which is 69.71

While the lightweight MobileNetV2 was for efficient computation, it achieved good results with fewer parameters. Though this performed worse compared to AlexNet and ResNet-18, the lightweight design of MobileNetV2 will be perfect in applications with less computational resources.

This work demonstrated that much deeper CNN architectures, complemented with efficient regularisation, improve the classification accuracy for complex datasets like CIFAR-10 by a large margin. Optimisations such as dropout, batch normalisation, and adaptive learning rate become really important in letting these models generalise well and preventing overfitting.

Code Section

The link to my GitHub Repository for Deep Learning Assignment 2

References

[1] 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 (CVPR)*, 2016, pp. 770-778.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, 2012, pp. 1097-1105.

[3] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 6105-6114.

[4] C. Szegedy, W. Liu, Y. Jia, et al., "Going Deeper with Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1-9.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510-4520.

[6] O. Russakovsky, J. Deng, H. Su, et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211-252, 2015.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[9] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251-1258.

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700-4708.

[11] T. Zhang, C. Zhang, and Z. Chen, "Understanding Deep Learning Requires Rethinking Generalization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[12] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning," *SIAM Review*, vol. 60, no. 2, pp. 223-311, 2018.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, MA, USA: MIT Press, 2016.

[14] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 249-256.

[15] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015, pp. 448-456.

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431-3440.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788.

[18] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or Deeper: Revisiting the ResNet Model for Visual Recognition," *Pattern Recognition*, vol. 90, pp. 119-133, 2019.

[19] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440-1448.

[20] J. Huang, V. Rathod, C. Sun, et al., "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7310-7311.