CNN for CIFAR-10 Image Classification

Simon B Siagian
The University of Adelaide
simonbaharja.siagian@student.adelaide.edu.au

Abstract

This study explores the optimization and evaluation of Convolutional Neural Network (CNN) architectures on the CIFAR-10 dataset. This study focuses on standard CNN models such as ResNet, MobileNet, and EfficientNet, tuning hyperparameters like learning rate, batch size, and optimization methods to assess their impact on model performance. Among the tested architectures, a modified ResNet-18 emerges as the best-performing model, achieving a final validation accuracy of 87.1% and a test accuracy of 84.8% after iterative hyperparameter tuning. Additionally, we compare these models with a custom-built CNN to highlight the role of architecture design and parameter selection in enhancing classification performance.

1 Introduction

Machine learning algorithms are widely used for image recognition and detection, where image recognition involves identifying and inferring information from digital images. Convolutional Neural Networks (CNNs) have become the leading approach for these tasks due to their ability to learn relevant features from data automatically. In 2012, Alex Krizhevsky and colleagues introduced a breakthrough CNN model that achieved high classification accuracy on ImageNet [1], propelling CNNs into the forefront of deep learning for applications across domains from computer vision to speech processing [2].

In computer vision research, the focus has shifted from feature engineering to network design engineering [3], resulting in numerous advanced architectures over time. Starting with foundational models such as AlexNet [4] and VGGNet [5], successive models have consistently pushed performance boundaries on benchmarks like ImageNet and CIFAR-10.

This study aims to optimize and evaluate several renowned CNN architectures, including ResNet, MobileNet, and EfficientNet, assessing the impact of hyperparameters on model performance. Additionally, we compare these architectures against a custom-built CNN model developed from scratch.

2 Method

2.1 Dataset

The CIFAR-10 dataset is a well-known benchmark in computer vision research. It was created by the Canadian Institute for Advanced Research (CIFAR) [6] to evaluate the performance of machine learning algorithms in image classification tasks. It consists of 60,000 32×32 color images across 10 distinct classes, each representing a unique object category: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class contains 6,000 images, ensuring a balanced dataset ideal for multiclass classification.

2.2 Convolutional Neural Network (CNN)

Compared to regular Artificial Neural Networks (ANN), CNN can capture spatial information within the image using filters or kernels to extract the information from the image or feature map [7]. The architecture of CNN is inspired by human visual perception [8], where the filters represent receptors that can recognize various features. CNN has several advantages over regular ANN:

1. Neuron connections — each neuron from the previous layer is not connected to all neurons in the next layer, resulting in fewer parameters and faster convergence.

2. Weight share — a connection can share the weight with another connection.

3. Dimension reduction — CNN use a pooling layer to sample the image while retaining the information within the image

There are four main components in CNN model:

1. Convolutional Layers: where each layer applies filters or kernels to the input image to generate feature maps highlighting specific patterns such as edges, textures, or colors.

2. Activation function: after each convolution operation, activation function such as ReLu is applied to introduce nonlinearity into the network. This non-linearity enables the CNN to model complex patterns by stacking multiple layers.

3. Pooling Layers: these layers are used to reduce the spatial dimensions of the feature maps, which decreases the number of parameters and computations in the network. By retaining only the most significant features, pooling layers help make the network invariant to small shifts and distortions in the input image.

4. Fully Connected (dense) Layers: these layers are applied at the end of the CNN layer to combine the extracted features and perform the final classification typically using sigmoid function. These layers take the high-level patterns identified in the convolutional and pooling layers. Figure 1 shows the architecture of a full CNN.

2.3 Common CNN Architectures

2.3.1 ResNet

This architecture was introduced by He et al. [9] in 2015. The revolutionary idea of this architecture is to utilize bypass pathway [[2]][10]. This model combines a conventional feed-forward network with a residual connection. The mathematical concept for a residual block can be summarized as:

$$Output = F(x, \{W_i\}) + x \tag{1}$$

where $F(x, W_i)$ represents the transformation applied to input x using weights W_i, and x is the identity mapping. This formulation makes learning an identity function easier and allows the network to preserve information through each layer. In this study, the architecture of ResNet-18 will be used. Resnet consists of several blocks with multiple convolutional and skip connections [7].

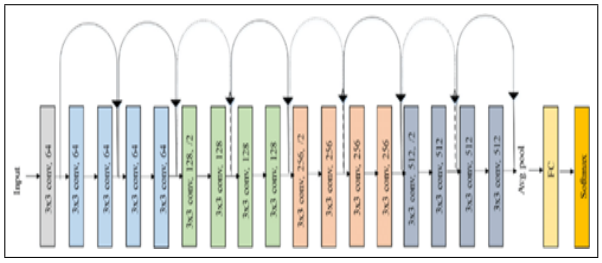


Figure 2: ResNet-18 Architecture

Figure 2 shows the architecture of ResNet-18, which consists of 18 layers, including 17 convolutional layers and 1 dense layer at the end. The original Resnet18 utilizes a filter size of 7×7 for ImageNet classification; however, in this study, the first layer is modified to use a 3×3 filter since the input image size is only 32×32. This smaller size ensures that more spatial information can be preserved throughout the model [11].

2.3.2 MobileNet

MobileNet is one of lightweight convolutional neural networks designed specifically for mobile and embedded vision applications where computational resources, such as processing power and memory, are limited [12]. The core innovation in MobileNet is the use of *depthwise separable convolutions*, which significantly reduces the computational cost and model size [3]. Depthwise separable convolution breaks this down into two operations:

- Depthwise Convolution: This operation applies a single convolutional filter to each input channel separately, effectively reducing the number of operations.

- Pointwise Convolution (1×1 convolution): After depthwise convolution, a 1x1 convolution is applied to combine the channels, introducing interactions between channels.

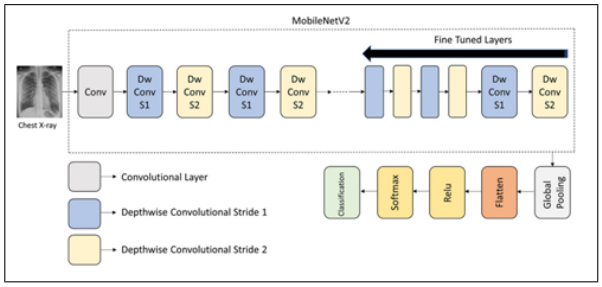


Figure 3: MobileNetV2 Architecture

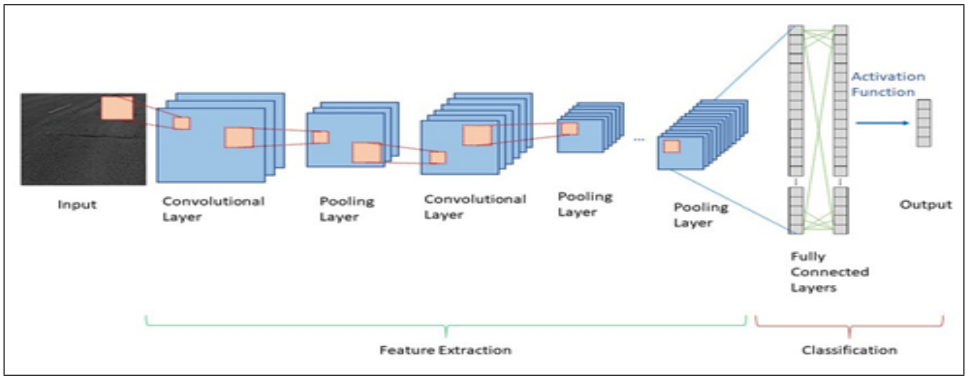


Figure 1: Architecture of CNN

2.3.3 EfficientNet

EfficientNet is a family of convolutional neural networks that balances model accuracy and efficiency [13]. The EfficientNet models use a unique compound scaling method that uniformly scales the model's depth, width, and resolution, achieving state-of-the-art accuracy with significantly fewer parameters and computational cost than previous architectures. Whereas architectures like ResNet and MobileNet typically scale up model dimensions (depth) when increasing model capacity, EfficientNet introduces a compound scaling method that scales depth, width, and input resolution in a balanced way. This allows the network to grow in capacity while preserving its efficiency.

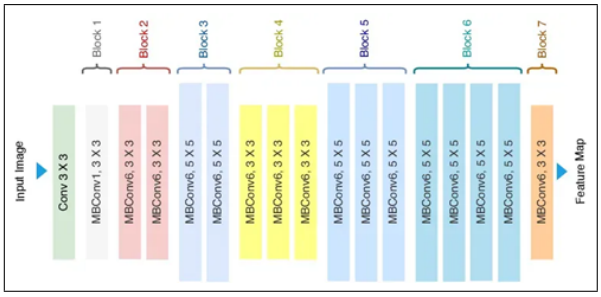


Figure 4: EfficientNet Architecture

MBConv layers, also used in MobileNetV2, use depthwise separable convolutions and inverted residuals to maintain high efficiency while preserving the model's representational power [13]. These layers effectively capture spatial and channel-wise information while being computationally light.

2.4 Modelling and Evaluation

In this study, all models will be trained on the CIFAR-10 dataset. Each model will be trained using the same preprocessing steps and evaluation metrics to ensure experiment comparability.

2.4.1 Experiment Setup

Data augmentation is a critical step in the experiment, especially for smaller datasets like CIFAR-10, as it artificially increases data diversity, helping models generalize better and avoid overfitting [14]. We follow the data augmentation process suggested in [14] by randomly cropping to 32×32 size with 4-pixel padding applied to the edges. Each image is then randomly flipped horizontally with a probability of 70%. Random rotations up to 15 degrees are also applied to the training image, which helps the model learn the rotational invariance. Finally, the pixels of the image are normalized to have a mean of 0 and a standard deviation of 1 across each channel. This normalization standardizes the input range, helping the model converge faster during training.

Training Process. The CIFAR-10 dataset is divided into a training set (40,000 images), a validation set (10,000 images), and a test set (10,000 images) [14]. The validation set is used for hyperparameter tuning and to monitor for overfitting, while the test set provides an unbiased evaluation of the final model performance.

2.4.2 Model Evaluation and Hyperparameter Tuning

In the training process, hyperparameter selection is done iteratively by evaluating the model accuracy on the validation set. Several hyperparameters will be tuned to get the model's best performance:

- **Batch size**. Batch size is the number of training samples processed before the model's internal parameters are updated. In this study, common batch sizes of 32, 64, and 128 are tested.

Table 1: Model Summary

Model	# ConvLayer	# DenseLayer	Train Acc (%)	Val Acc (%)	# Parameters
Base Model 1	2	1	53.7	49.2	60.3k
Base Model 2	3	2	79.8	74.6	89k
Base Model 3 (dropout)	4	2	87.8	76.5	274k
Base Model 4 (batch norm + dropout)	5	2	91.0	76.2	1.01M
ResNet-18 + SGD + DO	17	1	89.0	81.0	11.17M
ResNet-18 (Mod first kernel) + Adam + Batch norm + DO	17	1	95.8	87.1	11.17M
MobileNet	27	1	80.2	70.4	2.23M
EfficientNet	16 MBConv	1	85.0	72.3	4.02M

- **Learning rate**. The learning rate is a critical hyperparameter that determines the step size during model parameter updates. Learning rates ranging from 0.1 to 0.001 [12] and learning rates with step and one-cycle scheduling are tested.
- **Optimizer**. This hyperparameter affects the speed and quality of convergence. This study will evaluate two common options: Adam Optimizer and SGD Optimizer [15].
- **Dropout**. This hyperparameter acts as a regularization technique that randomly drops units from the network during training to prevent co-adaptation of neurons and improve generalization.
- **Batch Normalization**. This layer standardizes the inputs to each layer, stabilizing and speeding up training. It can also act as a regularization technique.
- **Epoch and early stopping**. Setting too many epochs can result in overfitting. Thus, in this study, epoch with an early stopping technique will be used to terminate the training process when there is no improvement in model loss.

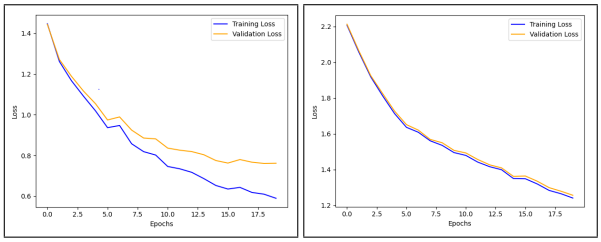
After each trial with one set of hyperparameters, we evaluate the model's overall accuracy on the validation set while still considering the performance on each class. Moreover, to provide information on underfitting, overfitting, and convergence, we use learning curve plotting loss and accuracy.

3 Result and Analysis

Base Model. The base model is built from scratch, and more layers are added iteratively. The initial model only has 2 convolutional layers with one dense layer. This model only reached a validation accuracy of 49.2%. The baseline model lacked sufficient complexity to capture and produce feature maps of the image. Subsequently, the model was improved by adding more layers to capture more important information from the image. Base Model 2 with 3 convolutional layers improved significantly, with a final accuracy 74% at epoch 20. This base model experiment tested different learning rates (0.01 and 0.001). Results show that the model with a learning rate 0.01 produces slightly better final validation accuracy.

To evaluate more about the model architecture, we add more layers to increase the model complexity. In base models 3 and 4, more convolutional and dense layers were added. However, the chance of overfitting also increases due to the number of parameters. In base models 3 and 4, regularization techniques such as dropout and batch normalization were used to minimize overfitting. While dropout and batch normalization in the baseline model can reduce the overfitting, it did not fully eliminate the overfitting; there is still a sign of overfitting, especially at epoch ≥ 20. Surprisingly, results show that different optimization methods result in different learning processes. In this base model, using SGD almost eliminates the overfitting and produces a smoother learning curve, shown by the identical training and validation loss curve in Figure 5. However, using SGD optimizer still showed signs of overfitting for epoch ≥ 24. Finally, for the final baseline model, the best accuracy is 76.2% after 30 iterations.

ResNet-18 Architecture. In this study, ResNet-18 performance as one of the lightweight architectures in CNN is evaluated on the CIFAR-10 dataset. Originally. ResNet-18 is designed to process ImageNet with input 224×224. However, due to the limited computational resources, we



(a) Base model with Adam optimizer (b) Base model with SGD optimizer

Figure 5: Learning curve for model with Adam and SGD optimizer

proceed with an input size of 32×32. Initially, the original architecture of ResNet-18 is tested, and the CIFAR-10 image is upscaled to match the model input. Table 2 shows the comparison of the final performance and training time of ResNet-18 with different input sizes.

Table 2: Comparison of different input size

Input Size	**Val Acc (%)**	**Training time**
32×32	87.10	28.16 m
224×224	96.11	74 m

In this study, the final pooling layer is changed to AdaptivePooling to make the tensor size match the dense layer input. Furthermore, we set a small filter size at the initial layer to get more representative feature maps over the convolutional layer [11]. Instead of the original 7×7, we use 3×3 filter size. The results show that using a smaller filter size at the first layer improves the model performance significantly. Using the original kernel size, the final validation accuracy after 30 epochs is 80.98%; after changing the kernel size, the accuracy is improved to 87.1%.

In the experiment with ResNet-18 architecture, we also evaluated the effect of different learning rates on the model. Two types of learning rate schedulers are used to train the model: stepwise scheduling and OneCycle scheduling. Both models with learning rate schedules do not perform better than those with a fixed learning rate in terms of final validation accuracy. However, the learning process using a learning rate scheduler is more stable. This is because learning rate scheduling stabilizes the learning process by starting with a higher rate for faster convergence and then gradually reducing it to avoid overshooting and improve precision. This approach reduces oscillations and enables the model to generalize better. Figure 6 shows the comparison of different settings of learning rates in ResNet-18 model training.

The effect of batch size is also evaluated in this ResNet-18 experiment. Common batch sizes used in deep learning are tested: 32, 64, and 128. Result (Fig 7) show that using a batch size of 64 results in better performance than the other.

Finally, the best model for ResNet-18 architecture uses a batch size of 64, a fixed learning rate of 0.01, and Adam optimization, with several dropout and batch normalization layers added.

MobileNet. In this study, we also evaluated MobileNet architecture with CIFAR-10 dataset. In this experiment, the model was not built from scratch. Instead, it was imported using a torch library with a pre-trained set to False to ensure the model was trained from scratch. Results show that although more convolutional layers are being used in this model, it performed worse than the ResNet-18 model with the best accuracy of around 70.4% after training with 20 epochs, while ResNet-18 achieved 80% accuracy on that epoch. This is might due to the ability of ResNet-18 to maintain higher representational power compared to the depthwise and pointwise convolutional layer of MobileNet. Although the MobileNet model reduces the number of parameters and is more efficient, it may also reduce the model's representational capacity, affecting performance on big datasets.

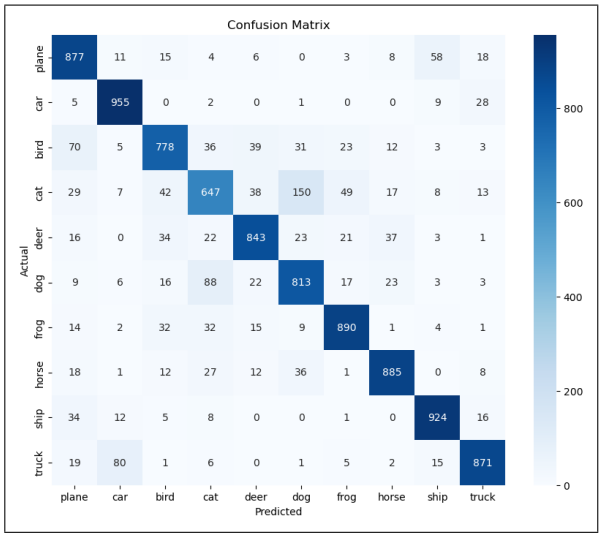
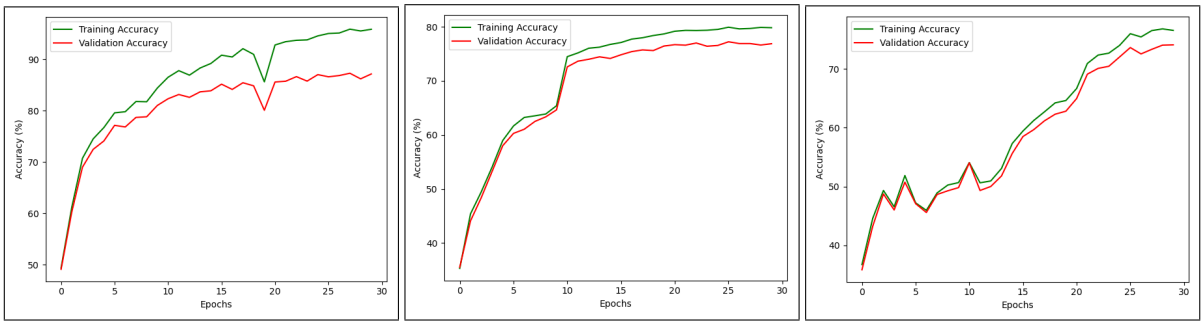


Figure 8: Confusion matrix of the best model on test set

EfficientNet. Like MobileNet, EfficientNet did not perform better than the ResNet model, with a final validation accuracy of around 73 %. This might be because both MobileNet and EfficientNet compound scaling methods are tailored for higher-resolution data, such as ImageNet, whereas



(a) ResNet with fixed learning rate (b) Stepwise scheduling (c) OneCycle scheduling

Figure 6: Comparison of learning rate strategies in ResNet-18 training

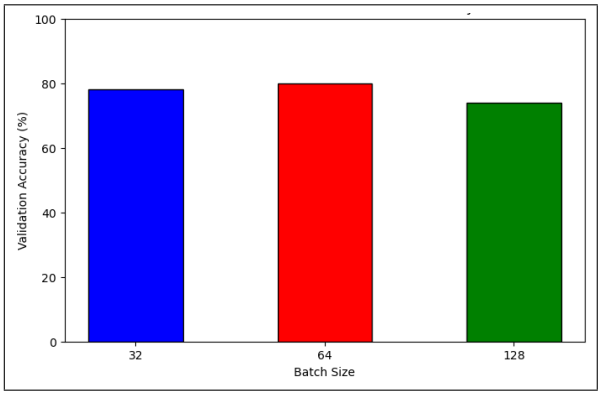


Figure 7: Model ResNet-18 comparison with different batch size

ResNet-18 is more robust across various resolutions.

Table 1 shows the performance summary of all models in this experiment. The best model in this experiment is ResNet-18, with some modifications applied to it, such as a smaller initial filter size, dropout layer, and a modified pooling layer. After obtaining the best model with hyperparameter tuning using a validation set, the best model was tested against a test set to measure the final generalization performance. The accuracy of the best model on the test set is 84.83%.

Figure 8 shows the confusion matrix of the final best model. Results show that the model performed relatively well for all classes, indicating that the model can generalize on the test set. In examining the confusion matrix and individual class accuracies, we can observe that even the best-performing model struggles with certain classes. Specifically, there is notable confusion between similar classes, such as "cat" and "dog." Despite achieving high overall accuracy, the model has difficulty distinguishing between these two classes, which may be due to their similar visual features, like fur textures, shapes, and common backgrounds. This similarity can lead the model to misclassify images of cats as dogs and vice versa. The lower individual accuracy for the "cat" class (64.7%) and moderate accuracy for the "dog" class (81.3%) suggest that these two classes frequently overlap in the model's predictions.

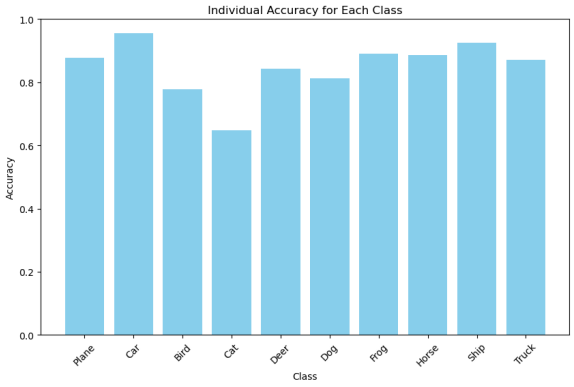


Figure 9: Individual class accuracy of the best model

Figure 9 shows the individual class accuracy. This analysis reveals that the "car" class is the most distinguished, achieving the highest individual accuracy. This strong performance likely stems from cars' unique and distinct visual features, such as their defined shapes, colours, and patterns that are less commonly shared with other objects in the dataset. These characteristics allow the model to recognize cars with minimal confusion, as there are fewer similar visual cues among the other classes, making misclassification less likely.

Table 3: Model Hyperparameter Summary

Model	Learning Rate	Optimizer	Dropout	Epoch
Baseline best model	0.01	SGD	0.5	20
ResNet-18	0.001	Adam	0.3	30
MobileNetV1	0.001	Adam	–	20
EfficientNet	0.001	Adam	–	20

Table 3 shows the set of hyperparameters for each model in this study. Results show that although the SGD optimizer is better for convergence, as shown in the base model, the best final performance is still shown by the model with the Adam optimizer. All models also work better with the addition of batch normalization, meaning that batch normalization can help the network learn more quickly and efficiently. As a result, batch normalization improves convergence and enables the model to reach higher accuracy in fewer epochs (20-30), optimizing training time and model performance.

4 Conclusion and Future Work

This study evaluated the performance of several CNN architectures on the CIFAR-10 dataset, with a focus on ResNet-18, MobileNet, and EfficientNet. After extensive experimentation and hyperparameter tuning, the modified ResNet-18 model emerged as the most effective, achieving a validation accuracy of 87.1% and a test accuracy of 84.8%. This model's success highlights the impact of architectural adjustments—such as using a smaller initial filter size and applying dropout layers—on improving performance.

The findings also demonstrate that while lightweight architectures like MobileNet and EfficientNet are valuable for resource-limited applications, they may not match the representational power of ResNet-18 on lower-resolution datasets like CIFAR-10.

Future work could focus on exploring more advanced architectures, such as Vision Transformers or ensemble methods, to further enhance classification accuracy. Applying these models to more complex datasets, like ImageNet, alongside different data augmentation techniques, may provide additional insights into the capabilities of modern CNN architectures.

5 Code

The code for this study, including model architectures, training processes, and evaluation metrics, is available on GitHub repository.

References

[1] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.

[2] A. J. H. Laith Alzubaidi, Jinglan Zhang, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, 2021.

[3] D. M. A. S. L. Chaoning Zhang, Philipp Benz, "Mobilenetv2 model for image classification," *ResNet or DenseNet? Introducing Dense Shortcuts to ResNet*, pp. 3550–3559, 2021.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

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

[6] Kaggle, "Cifar-10 - object recognition in images," 2024.

[7] Y. Bu, W. Xiong, and L. Zhu, "Performance of convolutional neural networks on cifar and dog breed classification dataset," *Proceedings of the 2023 International Conference on Machine Learning and Automation*, pp. 199–207, 2023.

[8] W. Y. S. P. Zewen Li, Fan Liu, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Neurol and Learning Systems*, vol. 33, no. 12, 2022.

[9] 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)*, pp. 770–778, 2016.

[10] J. Liang, "Image classification based on resnet," *Journal of Physics: Conference Series*, 2020.

[11] M. A. L. Owais Mutjaba Khanday, Samad Dadvandipour, "Effect of filter sizes on image classification in cnn: a case study on cfir10 and fashion-mnist datasets," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 4, pp. 872–878, 2021.

[12] C. Z. Y. L. Ke Dong, Yihan Ruan, "Mobilenetv2 model for image classification," *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, vol. 14, no. 3, pp. 476–480, 2020.

[13] L. B. X. L. Jiangchuan Liu, Mantao Wang, "Efficientnet based recognition of maize diseases by leaf image classification," *Journal of Physics: Conference Series*, vol. 1693, 2020.

[14] M. H. P. M. H. Reza Fuad Rachmadi, I Ketut Eddy Purnama, "A systematic evaluation of shallow convolutional neural network on cifar dataset," *IAENG International Journal of Computer Science*, vol. 46, no. 2, 2019.

[15] C. X. Xiaoling Xia, "Inception-v3 for flower classification," *2017 2nd International Conference on Image, Vision and Computing*, 2017.

[16] M. C. Siekavizza, *Optimizing CNN Architectures for Image Classification: A Case Study on CIFAR-10*. PhD thesis, Harrisburg University of Science and Technology.

[17] J. V. Felipe Giuste, "Cifar-10 image classification using feature ensembles," 2020.

[18] N. Jinliang, "Cifar-10 image classification based on resnet," 2020.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.