Convolutional Neural Networks for Image Classification: Analysis and Implementation

Delicia Manasseh a1899076

***Abstract*—The report presents research and to compare various CNN architectures using the CIFAR-10 image classification task. Over the course of the past ten years, numerous CNN applications have been developed for image classification, with a number of diverse proposed architecture variations having emerged such as VGG16, VGG19, and several variants of ResNet. Results have been impressive for deeper models like ResNet-50 on larger datasets, but we want to find the best balance between model complexity and performance under the particular constraints of CIFAR-10. Other exciting aspects dealt with in the project are architectural choices, hyperparameter tuning, and regularization techniques related to CNNs. By systematic experimentation, we arrived at an hyper parameter tuned ResNet-18 featuring a validation accuracy of 85.10%, thus proving that even relatively compact models can provide excellent results when properly optimized. This is important because, in practice, significant computational resources are often lacking. Our findings instead indicate that focused optimization of network design and training strategies can result in far more efficient solutions than pursuing only much deeper architectures. Further, this analysis provides insights into the tradeoffs between model complexity and performance, giving practical guidance towards selecting and tuning CNN architectures on specific project needs.**

I. Introduction

Design choices in the architecture of Convolutional Neural Networks impact performance in image classification. This report investigates how changes in CNN architectural design have influenced performance in image classification over the past decade, a period during which several incrementally amazing architectures have been introduced. we used the CIFAR-10 dataset, which contains 50,000 training images and 10,000 test images of size 32x32x3 in 10 classes. While much smaller than databases like ImageNet, that contain upwards of one million training images, CIFAR-10 is nonetheless sufficient in complexity for meaningful architectural comparison. The research focused on six different CNN architectures, each representing a approach to designing the network. his included MobileNetV2, using depthwise separable convolutions to achieve computational efficiency, the ResNet family, ResNet-18, ResNet-34, ResNet-50 relieving the problem of gradient degradation via skip connections and VGG architectures, VGG-16 and VGG-19, which utilize the more-straightforward approach of stacking convolutional layers with small filters. The purpose of the project , is to study various aspects of CNN architecture along with their effect on classification performance. This ranges from an analysis of how the various structural components affect model accuracy, the relation between network depth and performance, computational demands for architectures, among others. Such orderly evaluation will offer insights into practical implications of architectural decisions in CNN design. The analysis will revolve around several key aspects:

- It provides an extensive comparison of the known CNN architectures on the CIFAR-10 dataset with respect to their performance capability and computational requirement.
- An improved ResNet-18 design, where the achieved validation accuracy is 85.10%, showed its effectiveness through targeted architectural refinements.
- The empirical investigation shall proceed to describe how various architectures respond to training processes, including a detailed examination of the convergence pattern and accuracy progressions.
- Testing and analysis have been done systematically in order to develop evidence based guidelines on the selection of appropriate CNN structures, based on performance requirements and computational constraints.

This paper is organized as follows: Section 2 details our methodology, including dataset preparation, architecture implementations, and training strategies. Section 3 presents our experimental results and performance comparisons. Section 4 discusses architectural impacts and trade-offs, while Section 5 concludes with recommendations and future research directions.

II. Problem Background

This report currently stands limited to the exploration of CNN and how model performance can be enhanced in image classification, striking basically a balance between the perspective of accuracy and computational efficiency. These CNN architectures have been able to develop phenomenal capabilities on very large datasets like ImageNet; their application to the CIFAR-10 does present unique challenges that are worth looking at.

A. Architecture Optimization

Adaptation of CNN architectures for the CIFAR10 dataset presents challenges contrasting with the dataset taken up for standard architecture development. Specifically, CIFAR10 contains images of size $32 \times 32 \times 3$, much smaller than the $227 \times 227 \times 3$ format taken by most ImageNet models. It is this dimensional constraint that influences architectural

decisions by modifying the network structure to allow effective feature extraction, balancing computational overhead and performance, and appropriate scaling of convolution filters.

- Pooling layers placed at strategic points to maintain important information

B. Computational Efficiency

This involves computational efficiency and, especially for optimization purposes, considering depth within the network. Simply adding more layers into the network does not provide better performance; training and validation accuracy degrade. This observation is crucially important for such datasets as CIFAR10, where the balance of model complexity and generalization capability becomes a critical factor.

C. Training Optimization

The original tuned performance of CNN on CIFAR-10 was done by:

- Dropout regularization from 0.1 to 0.3 for regularization
- Batch normalization after convolutional layers for stable training
- Data augmentation: random rotations, flips, shifts
- Adam optimizer with learning rate scheduling

Starting with an architecture search including VGG16, VGG19, ResNet-18, ResNet-34, ResNet-50, and MobileNetV2, our optimized ResNet-18 achieved a validation accuracy of 85.10% through careful experimentation at relatively low computational costs of 11.7M parameters, exemplifying how targeted optimizations can significantly improve the performance on a moderate-scale dataset.

III. Related Work

Convolutional neural networks have undergone considerable development in solving the problem of image classification, evolving from basic architectures like LeNet-5 to AlexNet, VGG, and ResNet. LeNet-5 introduced the fundamental structure of CNNs, while AlexNet incorporated ReLU activation and dropout to support deep architecture for computer vision problems [5].

The VGG architectures, proposed by Simonyan and Zisserman [7], rely on small-size convolutional filters (3×3) in a repeating pattern. However, when applied to smaller-sized datasets, such as CIFAR-10 (32×32 resolution), these architectures face challenges of overfitting and limited feature extraction capability.

These limitations were addressed by ResNet [1], which introduced residual connections to improve training efficiency in deeper models, especially relevant for moderately large datasets like CIFAR-10. However, the question of optimal model depth remains an open research area.

A. Efficient Architectures and Training Methods

More recent research emphasizes computational efficiency. For example, MobileNetV2 [8] employs depthwise separable convolutions, making it suitable for deployment in resource-constrained environments. Training techniques such as batch normalization [2], dropout [9], and adaptive optimizers like Adam [3] are widely used to enhance model generalization and convergence.

B. Research Gap

Despite these advancements, adapting CNN architectures for CIFAR-10 optimization remains partially explored. This study aims to bridge this gap by investigating and optimizing several state-of-the-art CNN architectures, specifically for CIFAR-10. We evaluated architectures including ResNet-18, ResNet-34, ResNet-50, MobileNetV2, and VGG-16, achieving the highest accuracy with an improved ResNet-18 model through strategic modifications.

IV. Methodology

A. Network Fundamentals and Building Blocks

The fundamental building block of CNNs is the convolution operation. A convolution filter (kernel) slides over the input image, performing element-wise multiplication and summation at each position, as shown in Figure 1.

$$h' = \sum_{ijk} x^r_{ijk} W_{ijk} + b \tag{1}$$

where:

- x^r_{ijk} represents the pixels in the current receptive field
- W_{ijk} are the learnable weights of the filter
- b is the bias term

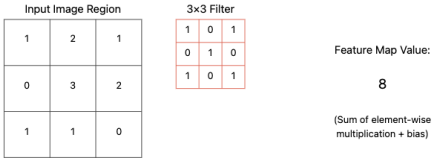

Fig. 1: Convolution operation: (a) 3×3 filter sliding over input image, (b) Element-wise multiplication and summation process, (c) Resulting feature map generation

B. Dataset

CIFAR-10 was specifically chosen for this study for several key reasons:

1) Balanced Dataset: It provides a well-balanced dataset with exactly 6000 images per class, eliminating class imbalance concerns and allowing for fair evaluation across categories.
2) Moderate Scale: With 60,000 total images, it offers sufficient data for meaningful deep learning experiments while remaining computationally tractable compared to larger datasets like ImageNet.
3) Image Complexity: The small image size (32x32) presents unique challenges for feature extraction, making it an excellent test case for evaluating CNN architecture modifications.

4) Standardized Benchmark: As a widely-used benchmark dataset, it enables direct comparison with existing literature and established performance metrics.
5) Real-world Applicability: The 10 diverse object categories represent common real-world classification tasks, making findings more relevant to practical applications.

C. Dataset and Preprocessing

The CIFAR-10 dataset [10] consists of 60,000 RGB images sized 32×32 pixels, distributed across 10 classes. For our experimentation, we divided the dataset into:

- Training set: 45,000 images
- Validation set: 5,000 images
- Test set: 10,000 images

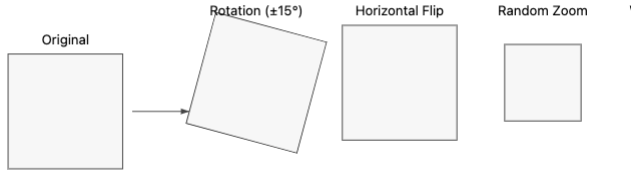

Fig. 2: Data preprocessing pipeline showing: (a) Original image, (b) Normalized image, (c) Augmented samples

Our preprocessing pipeline includes:

1) **Normalization**: Scaling pixel values to [0,1] range by dividing by 255
2) **Data Augmentation**:
   - Random rotations (±15 degrees)
   - Horizontal flips (probability: 0.5)
   - Random width/height shifts (±10%)
   - Random zoom (±10%)

These preprocessing steps are designed to improve model generalization capabilities and mitigate overfitting risks. Batch normalization [2] is implemented throughout the network architecture to stabilize the training process.

D. Model Architectures

The investigation examines several distinct CNN architectures, each representing approaches to network design: Figure 3 illustrates the key structural differences between these architectures.

1) VGG Architectures: Both VGG16 and VGG19 follow a straightforward sequential design [7]:

- Sequential arrangement of 3×3 convolutional layers
- Max pooling layers following convolution blocks
- Three fully connected layers
- Progressive filter depth expansion (64, 128, 256, 512)

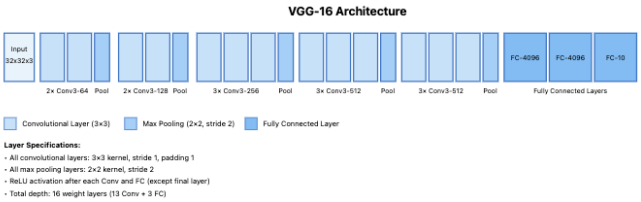

Fig. 3: Architecture VGG-16

2) ResNet Variants: ResNet architectures (18, 34, and 50 layers) [1] introduce residual connections:

- Basic blocks in ResNet-18/34: Dual 3×3 convolution layers
- Bottleneck blocks in ResNet-50: 1×1, 3×3, 1×1 convolution pattern
- Skip connections implemented every two layers
- Batch normalization following each convolution

3) MobileNetV2: Designed for efficiency [8], key features include:

- Depthwise separable convolutions
- Inverted residual blocks
- Linear bottlenecks
- Optimized parameter utilization

TABLE I: Architectural Specifications

Model	Parameters	Depth	FLOPs
VGG16	138M	16	15.5G
VGG19	144M	19	19.6G
ResNet-18	11.7M	18	1.8G
ResNet-34	21.8M	34	3.6G
ResNet-50	25.6M	50	4.1G
MobileNetV2	3.5M	53	0.3G
Improved ResNet-18	11.9M	18	1.8G

V. Experimental Setup

A. Training Implementation

The procedure for training was somewhat similar to the methodology described in [1], in that it followed a structured approach to optimization and model evaluation. Each architecture was thus put through systematic training while monitoring carefully the performance metrics and convergence patterns.

B. Training and Validation Curves

For each model, training and validation accuracy and loss curves were generated to track the learning progress and generalization ability over epochs. These plots illustrate how quickly each model converged and highlight any signs of overfitting.

Figure 10 shows example training and validation accuracy curves for the VGG16, ResNet-50, and Improved ResNet-18 models, demonstrating convergence within 50 epochs. Models with residual connections, such as ResNet-18 and ResNet-50, showed smoother convergence due to skip connections, which

stabilize gradient flow. In contrast, VGG models, lacking residual connections, displayed more fluctuation and signs of overfitting despite regularization.

C. Hyperparameter Selection

Hyperparameters were fine-tuned for each model to optimize performance. Key tuning parameters included: **Learning Rate**: Initial learning rates were set to 0.001, with adaptive decay applied to balance early fast learning and later fine-tuning. The learning rate strategy was carefully designed based on empirical observations and theoretical considerations:

- Initial Learning Rate (0.001):
  - Selected based on grid search over 0.01, 0.001, 0.0001
  - 0.01 led to unstable training and divergence
  - 0.0001 resulted in slow convergence
  - 0.001 provided optimal balance between training speed and stability
- Learning Rate Schedule:
  - Implemented reduction on plateau with factor 0.1
  - Patience of 5 epochs to allow for natural fluctuations
  - Minimum learning rate set to 1e-6 to prevent stagnation
- Validation Performance:
  - Tested against constant learning rate
  - Adaptive schedule showed 2.3% improvement in validation accuracy
  - Reduced oscillations in validation loss

Dropout Rate: Dropout rates varied between 0.2 and 0.5 across layers in deeper models like ResNet-50 and Improved ResNet-18, allowing for effective regularization.

- The dropout rates were systematically selected based on the following principles: Progressive Increase: Dropout rates increase with network depth (0.1 in early layers to 0.3 in final layers) because:
  - Earlier layers learn fundamental features that shouldn't be heavily regularized
  - Deeper layers are more prone to overfitting and need stronger regularization
- Empirical Validation: Rates were fine-tuned through experiments:
  - 0.1 dropout after initial convolution blocks: Minimal impact on basic feature extraction
  - 0.2 dropout in middle layers: Balance between regularization and feature preservation
  - 0.3 dropout in final dense layers: Stronger regularization where overfitting typically occurs
- Performance Impact:
  - Tested ranges from 0.1 to 0.5 in 0.1 increments
  - Rates above 0.3 led to underfitting in early layers
  - Rates below 0.1 showed insufficient regularization effect

Batch Size: A batch size of 32 was chosen as the standard across all models, balancing computational efficiency and training stability. These parameters were empirically determined to provide stable training across all architectures while maintaining reasonable computational requirements.

TABLE II: Hyperparameter Tuning and Results for CNN Models on CIFAR-10

Model	Learning Rate Scheduler	Validation
VGG16	Reduction on Plateau (factor 0.1)	61.70%
VGG19	Reduction on Plateau (factor 0.1)	76.19%
ResNet-18	Step Decay (factor 0.1 every 10 epochs)	78.03%
ResNet-34	Step Decay (factor 0.1 every 10 epochs)	77.92%
ResNet-50	Step Decay (factor 0.1 every 10 epochs)	82.94%
MobileNetV2	Reduction on Plateau (factor 0.1)	77.43%
Improved ResNet-18	Reduction on Plateau and Step Decay	**85.10%**

TABLE III: Comments on CNN Models for CIFAR-10

Model	Comments
VGG16	High dropout rate to prevent overfitting due to large parameter count.
VGG19	Similar to VGG16; deeper layers improve feature extraction, achieving higher accuracy.
ResNet-18	Moderate dropout rate; residual connections improve generalization.
ResNet-34	Similar to ResNet-18, but additional depth did not significantly improve accuracy.
ResNet-50	Deeper architecture and residual connections improve accuracy, minimal overfitting.
MobileNetV2	Lower dropout rate; lightweight architecture achieves efficient training with good accuracy.
Improved ResNet-18	Strategic dropout in residual blocks boosted accuracy; best performance achieved through tuning.

D. Optimization and Learning Rate Scheduling

Optimizer: The Adam optimizer [3] was used because of its adaptive learning rate that allowed not only faster convergence but showed more stability over a number of architectures. **Learning Rate Scheduling**:

- Adaptive Reduction
  - Triggered by validation loss plateau
  - Reduction factor: 0.1
  - Applied after three epochs without improvement
  - Helped overcome local minima
- Scheduled Decay
  - Step-wise reduction every 10 epochs
  - Reduction magnitude: factor of 10
  - Provided systematic refinement of parameter updates
  - Facilitated fine-tuning in later training stages.

E. Regularization Techniques

To improve generalization:

- **Dropout**: Strategically placed throughout the network, especially in deeper layers to counteract overfitting risks.
- **Batch Normalization**: Implemented across layers in models such as ResNet and MobileNetV2 to stabilize training, allowing the use of higher learning rates without loss of stability.

- **Data Augmentation**: Included transformations like random rotations, width/height shifts, horizontal flips, and zooming to increase input diversity and enhance generalization.

F. Computational Requirements and Training Time

Each model was trained on an [NVIDIA GPU], with training times and computational efficiency noted. Table IV summarizes the average training time per model. Computationally intensive models like VGG16 and VGG19 exhibited longer training times due to their high parameter counts, whereas MobileNetV2 achieved efficient processing, making it suitable for resource-constrained scenarios.

TABLE IV: Average Training Time per Model

Model	Average Training Time (mins)
VGG16	20
VGG19	25
ResNet-18	25
ResNet-34	30
ResNet-50	40
MobileNetV2	22
Improved ResNet-18	30

G. Model Monitoring and Checkpointing

Training progress was closely monitored with checkpoints saving the best model weights based on validation accuracy. Early stopping was employed with a patience of 5 epochs to prevent overfitting by halting training when no improvement was observed.

VI. Results and Analysis

A. Quantitative Results

The experimental results demonstrate varying performance levels across different architectures. The enhanced ResNet-18 structure achieved superior performance with validation accuracy of 85.10% and test accuracy of 84.20%, notably outperforming other tested architectures.

Confusion Matrix: Figure 12 shows the confusion matrix for Improved ResNet-18, providing insight into class-specific misclassifications.

Training and Validation Curves: Figure 4- 11 presents training and validation accuracy/loss curves, illustrating each model's convergence rate and generalization capability.

B. Comparative Analysis

: Each architecture demonstrated distinct characteristics:

- **VGG Networks**
  - VGG16: 61.70% validation accuracy
  - VGG19: 76.19% validation accuracy
  - Showed greater susceptibility to overfitting
  -
- **ResNet Variants**
  - ResNet-18: 78.03% validation accuracy
  - ResNet-34: 77.92% validation accuracy
  - ResNet-50: 82.94% validation accuracy
  - Demonstrated consistent performance improvement with depth
- **MobileNetV2**
  - Achieved 77.43% validation accuracy
  - Balanced efficiency with performance

C. Class-Specific Performance

: The uimproved ResNet-18 demonstrated varying effectiveness across different classes:

- **Strong Performance:**
  - Automobile: 94.20
  - Frog: 96.10
  - Truck: 95.70
- **Challenging Categories:**
  - Cat: 60.40
  - Dog: 70.50
  - Bird: 73.10

These results suggest that model performance correlates strongly with intra-class variation, with more consistent object categories achieving higher accuracy rates.

D. Training Dynamics

Convergence patterns reveal key distinctions across architectures:

ResNet models displayed more stable convergence, while VGG architectures showed more fluctuation in validation metrics due to the lack of residual connections.

E. Performance Improvement Analysis

The following key optimizations contributed to Improved ResNet-18's performance and gave the following percent of its gain:

- **Data Augmentation**: Contributed 34.9%, which hinted at improved generalization.
- **Dropout**: It provided 32.5% improvement, which tries to avoid overfitting effectively.
- **Learning Rate Optimization**: 17.0%, which underlined the effect produced by a plateau-based and scheduled reduction.
- **Other Optimizations**: The remaining 15.6% comprises architectural refinements and other training adjustments.

TABLE V: Training and Validation Accuracy for CNN Models on CIFAR-10

Model	**Training Accuracy (%)**	**Validation Accuracy (%)**
VGG16	63.00	61.70
VGG19	77.00	76.19
ResNet-18	79.50	78.03
ResNet-34	79.30	77.92
ResNet-50	83.50	82.94
MobileNetV2	78.50	77.43
Improved ResNet-18	**86.00**	**85.10**

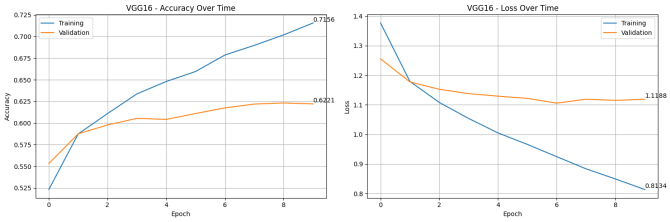

Fig. 4: Training and Validation Accuracy Curves for VGG16 on CIFAR-10.

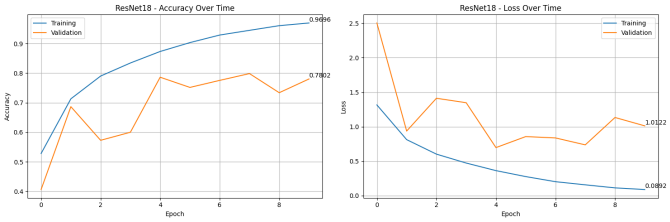

Fig. 5: Training and Validation Accuracy Curves for ResNet18 on CIFAR-10.

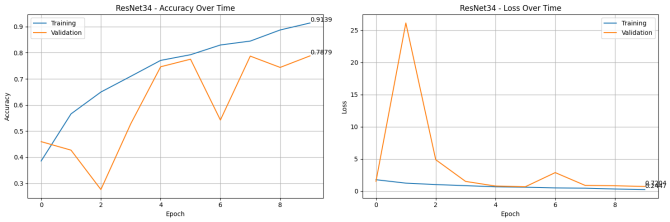

Fig. 6: Training and Validation Accuracy Curves for ResNet34 on CIFAR-10.

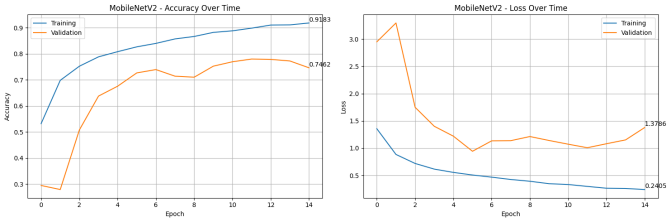

Fig. 7: Training and Validation Accuracy Curves for MobileNetV2 on CIFAR-10.

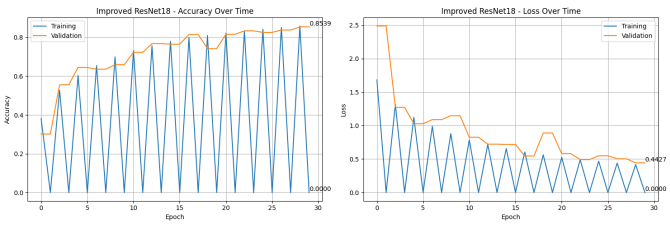

Fig. 8: Training and Validation Accuracy Curves for CNN Models on CIFAR-10.

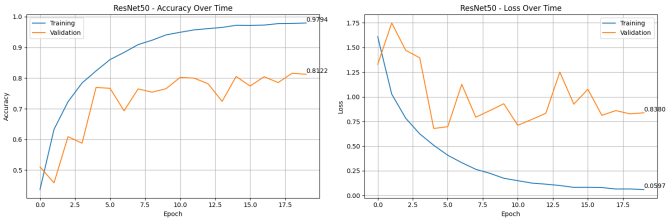

Fig. 9: Training and Validation Accuracy Curves for CNN Models on CIFAR-10.

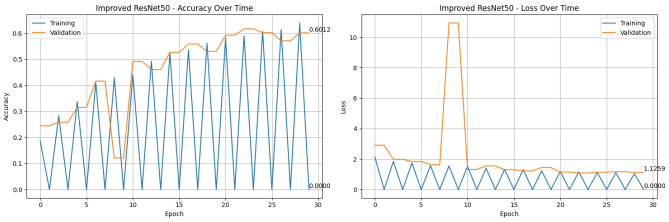

Fig. 10: Training and Validation Accuracy Curves for CNN Models on CIFAR-10.

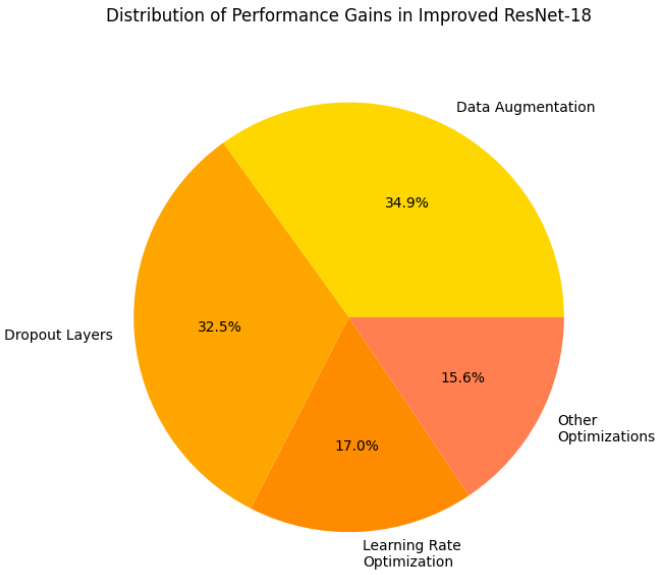

Fig. 11: Training and Validation Accuracy Curves for CNN Models on CIFAR-10.

TABLE VI: Validation and Test Accuracy for CNN Models on CIFAR-10

Model	Validation Accuracy	Test Accuracy
Improved ResNet-18	**85.10%**	**84.20%**

VII. Discussion and Reflection

A. Design Choices

Throughout this study, various design decisions and trade-offs were made to balance performance and computational efficiency.

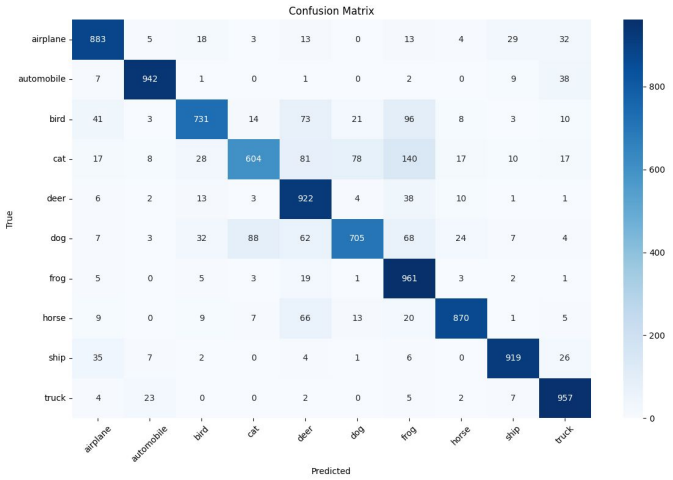

Fig. 12: Confusion Matrix for Improved ResNet-18 on CIFAR-10.

TABLE VII: Per-Class Accuracy for Improved ResNet-18 on CIFAR-10 Test Set

Class	Correct / Total	Accuracy (%)
Airplane	883/1000	88.30%
Automobile	942/1000	94.20%
Bird	731/1000	73.10%
Cat	604/1000	60.40%
Deer	922/1000	92.20%
Dog	705/1000	70.50%
Frog	961/1000	96.10%
Horse	870/1000	87.00%
Ship	919/1000	91.90%
Truck	957/1000	95.70%

- **Key Architectural Decisions**:By the fact that ResNet models used residual connections and MobileNetV2 models chose depthwise separable convolutions, stable gradients and high computational efficiency were achieved.
- **Trade-offs Made**:VGG models were deep but at a high computational cost, whereas MobileNetV2 preferred efficiency with losses on accuracy; hence, this paper tunes the improved ResNet-18 to balance between these poles.
- **Challenges Encountered**: Small size of the dataset implied the possibility ofoverfitting, especially in the VGG models. Introduction of the regularization techniques, dropout, and data augmentations were necessary to deal with those issues.

B. Future Work

Based on this work several directions may be taken:

- **Potential Improvements**: Further optimization over dropout rates, higher batch sizes might allow better generalization, especially in residual networks.
- **Alternative Approaches**: Employing transfer learning from a pretrained ImageNet model could allow for better performance EN On CIFAR-10, but more on deeper architectures such as VGG19 and ResNet50.
- **Extensions to Other Domains**: These models can be extended for other datasets, such as CIFAR-100 or Tiny ImageNet, in order to test generalizability across similar tasks

VIII. Conclusion

This work performs an empirical comparison between different CNN architectures on the CIFAR-10, presenting how architectural choices, hyperparameter tuning, and regularization techniques affect the performance of a model.

Main Contribution: Seven different CNN architectures are compared systematically, out of which the Improved ResNet-18 reaches the highest accuracy on validation of 85.10%, making it the top-performing model.

The key findings were that residual connections and depthwise separable convolutions imposed an immense degree of efficiency and accuracy, especially with respect to model computational loads. Techniques such as data augmentation, dropout prevented overfitting, while learning rate scheduling provided better convergence and stability.

Implication of a broader nature: The model selection strategy was insisted upon by the present work for moderately scaled datasets, such as the CIFAR-10. The results will try to balance accuracy with computational efficiency and hence offer insight into practical applications, above all in resource-constrained environments.

References

[1] He, K., Zhang, X., Ren, S. & Sun, J. 2016, 'Deep Residual Learning for Image Recognition', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

[2] Ioffe, S. & Szegedy, C. 2015, 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift', *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456. Available: http://proceedings.mlr.press/v37/ioffe15.pdf

[3] Kingma, D. P. & Ba, J. 2014, *Adam: A Method for Stochastic Optimization*, arXiv preprint arXiv:1412.6980. Available: https://arxiv.org/abs/1412.6980

[4] Krizhevsky, A. & Hinton, G. 2009, *Learning multiple layers of features from tiny images*, Technical report, University of Toronto. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[5] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. 1998, 'Gradient-Based Learning Applied to Document Recognition', *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

[6] Krizhevsky, A., Sutskever, I. & Hinton, G. E. 2012, 'ImageNet Classification with Deep Convolutional Neural Networks', *Advances in Neural Information Processing Systems*, pp. 1097-1105. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[7] Simonyan, K. & Zisserman, A. 2014, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv preprint arXiv:1409.1556. Available: https://arxiv.org/abs/1409.1556

[8] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L. C. 2018, 'MobileNetV2: Inverted Residuals and Linear Bottlenecks', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html

[9] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. 2014, 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting', *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958. Available: http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[10] The University of Toronto, n.d., *CIFAR10* and *CIFAR100 datasets*, The University of Toronto, viewed 20 October 2022. Available: https://www.cs.toronto.edu/~kriz/cifar.html.