Image Classification using CNN

Yuchen Liu
The University of Adelaide
yuchen.liu01@adelaide.edu.au

Abstract

Image classification has become a cornerstone task in computer vision, benefiting from advancements in deep learning, especially Convolutional Neural Networks (CNNs). This study evaluates the performance of three classic CNN architectures—LeNet, VGG, and ResNet—on the CIFAR-10 dataset. Additional enhancements, such as Data Augmentation and Dropout Layers, were applied to improve generalization and prevent overfitting. Experimental results indicate that each architecture exhibits unique strengths when handling small sample sizes, with ResNet demonstrating notable robustness in accuracy. These findings provide insights into the adaptability of CNN structures in resource-limited environments, contributing to the ongoing exploration of CNN-based image classification. The code for this project is available on the GitHub page (`https://github.com/MorainingErin/COMP_SCI_7318_DLF/tree/main/A2`*).*

1. Introduction

Image data is important across various domains because it aligns closely with human perception. As a form of visual information, images naturally reflect how humans sense and interact with the world, making them essential for understanding complex patterns, objects, and scenes. The ability to process image data has significant meanings in fields such as healthcare, autonomous driving, and security, where visual information plays a critical role in decision-making processes.

One of the fundamental tasks in image analysis is the classification problem. This task involves determining the class or label of images based on their content. This topic has been at the forefront of computer vision research for decades, with numerous algorithms developed to enhance the accuracy and efficiency of image classification.

In recent years, deep learning networks, particularly Convolutional Neural Networks (CNNs), have demonstrated remarkable success in addressing image classification challenges. CNNs are specifically designed to exploit the spatial structure of image data, allowing them to extract relevant features and patterns. By leveraging convolutional layers, pooling operations, and hierarchical feature extraction, CNNs have become the state-of-the-art approach for handling image data, offering significant improvements over traditional machine learning methods.

In this assignment for the course 'Deep Learning Fundamentals'[7], the performance of CNNs in the context of image classification tasks is explored and studied. Through a series of experiments, this report investigates the capability of several classic CNN architectures in handling image classification problems, particularly when dealing with small datasets. The designed experimental workflow focuses on how well these networks perform under such constraints, providing insights into their applicability and effectiveness in practical scenarios.

The organization of this report is as follows: Sec.1 introduces the background and motivation for the assignment. Sec.2 summaries the related research on CNN networks in this field. The problem is defined in Sec.3, and the pipeline is described in detail. Section 4 presents the performance results of both the baseline and improved models, along with additional analytical experiments that validate the effectiveness of the improvements. The code for this assignment is available on the GitHub page (Sec.5). Finally, Section 6 summarizes the work and proposes potential directions for future improvements.

2. Related Works

Image classification has become a widely researched topic for decades. Early approaches to image classification relied on handcrafted features and traditional machine learning algorithms, such as Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN). Although these methods achieved reasonable results, they were often limited in their ability to generalize across diverse datasets due to their reliance on manually engineered features. In recent years, this classification problem has been largely driven by advancements in deep learning, especially Convolutional Neural Networks.

Nowadays, the use of Convolutional Neural Networks

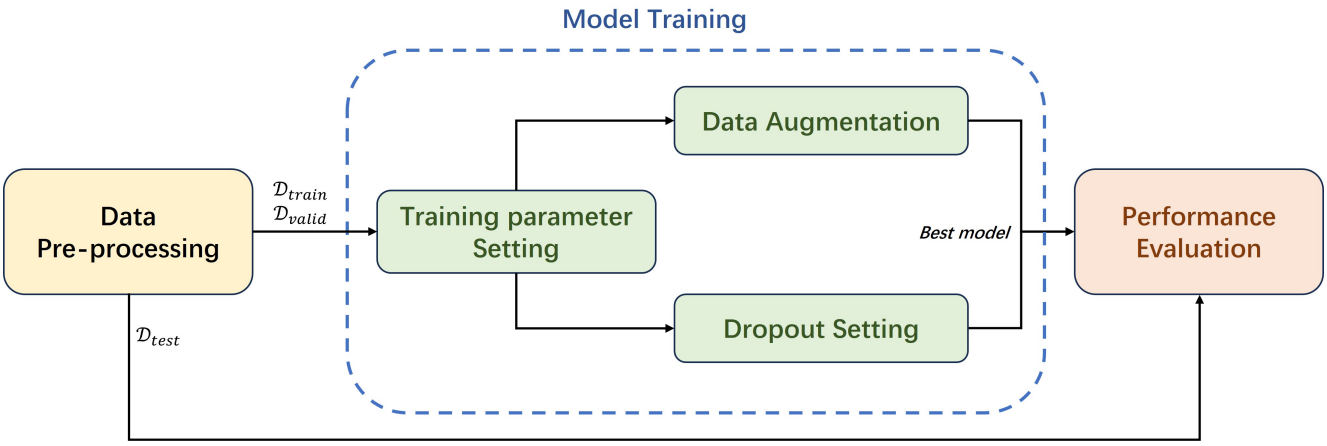

Figure 1: **Overview of the model training pipeline.** The process begins with data pre-processing. Within the model training stage, key settings are established for training parameters, data augmentation, and dropout configuration. The best model is selected based on performance on the validation set, followed by performance evaluation on the test set.

(CNNs) in image processing has been a foundational topic in computer vision research. Over the past few decades, numerous CNN architectures have emerged, each aiming to improve accuracy, efficiency, or both. Researchers have thoroughly explored these structures, with many comprehensive surveys detailing their development[2, 9, 13]. One of the earliest models, LeNet [5], introduced by LeCun et al. in 1998, set the stage for CNN-based image recognition by demonstrating the effectiveness of convolutional operations in extracting meaningful image features. Following this, various classic architectures have continued to shape the field. For instance, VGGNet [10], GoogLeNet [12] and ResNet [1] have each contributed distinct structural innovations that advanced the capabilities of CNNs for image classification tasks.

In this assignment, constrained by computational resources, the CIFAR-10 dataset [4] was used to perform experiments and validate results. Due to the small size of training dataset, several lightweight network architectures were selected for training. LeNet [5], as one of the earliest and most seminal CNN models, was used as a baseline due to its simplicity and foundational significance. The VGG architecture [10], with its layered sampling approach and multi-scale feature extractions, is now widely adopted in many CNNs. Finally, ResNet [1] introduced the residual estimation framework, significantly enhancing training precision and establishing itself as a common foundation in subsequent models. Given the importance of these architectures in CNN-based image processing, this study focuses on testing LeNet, VGG, and ResNet models.

3. Method

The model training pipeline is illustrated in the Fig. 1. It starts with data pre-processing, where the CIFAR-10 dataset is prepared and divided into training, validation, and test subsets. The core training process includes setting training parameters, applying data augmentation, and configuring dropout, each contributing to improved model generalization and robustness. The best model, determined from validation results, is then evaluated on the test set for final performance assessment.

3.1. Problem Definition

For the image classification problem, the objective is to identify a classifier f, which is defined as follows:

$$f : \mathbf{X} \rightarrow y, \tag{1}$$

where $\mathbf{X}$ represents an RGB image with the resolution of $[H, W, 3]$ (H for height and W for width of the RGB image), and $y \in \{i\}_{i=1,2,\ldots,K}$ corresponds to the class label indicating the object class its belongs to. There are a total of K classes for all the images.

In this work, a deep neural network is employed to fit the classifier f, with the network denoted as f_N, where N stands for 'Network'. To facilitate gradient-based optimization, the network's output differs slightly from the original definition in Eq.1:

$$f_N : \mathbf{X} \rightarrow y_N, \tag{2}$$

where the output y_N is a continues vector with length of K, representing the score for each class. This design facilitates more effective training of the neural network. The final class label is then determined using the $argmax$ operation.

3.2. Data Preprocessing

In this assignment, the dataset CIFAR-10[4], denoted as $\mathcal{D}$, was utilized. This dataset is then split into three subsets:

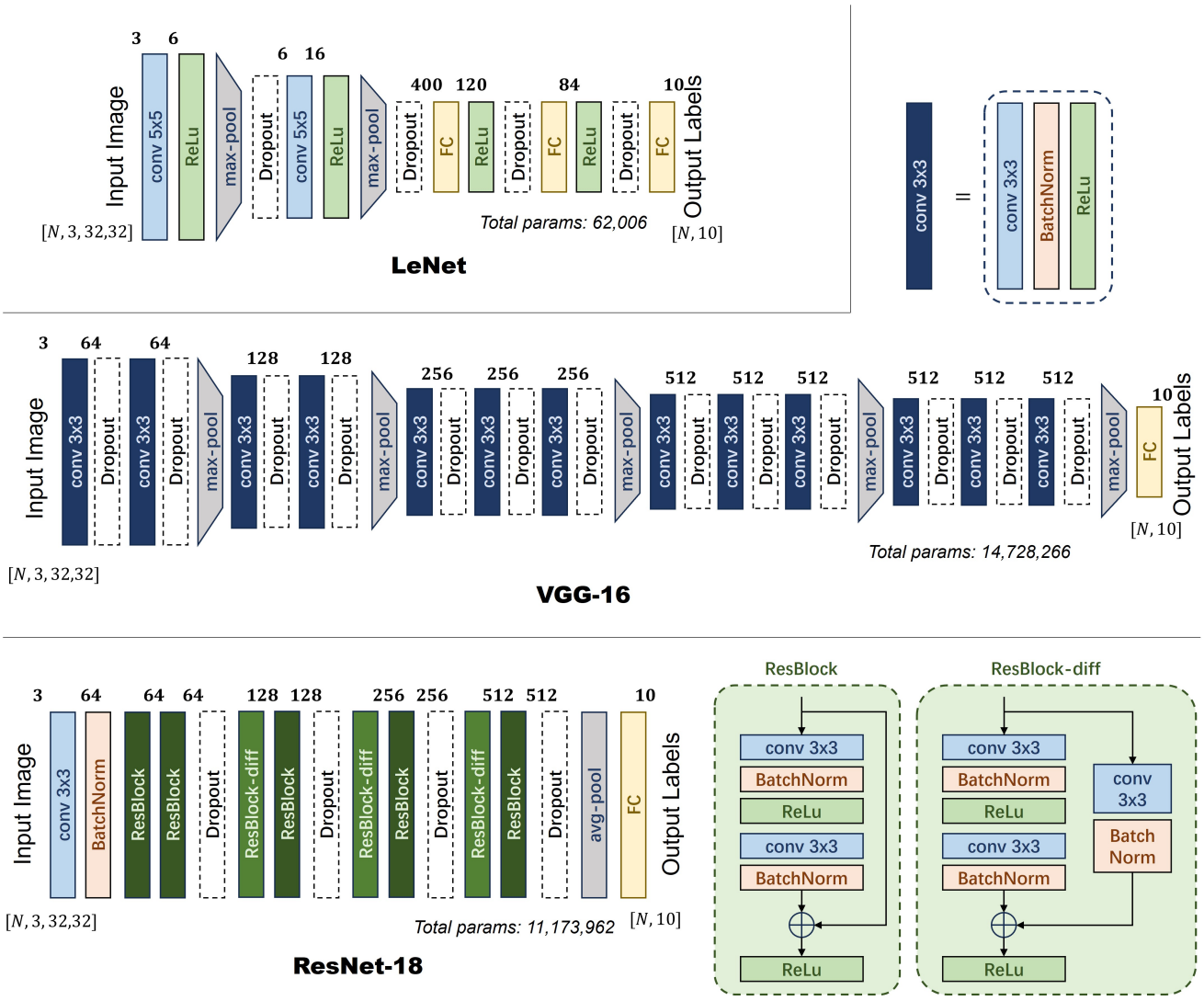

Figure 2: **Overview of the three network architectures used in this study.** The architectures include LeNet [5], VGG-16 [10] and ResNet-18 [1]. The input and output channels of convolutional and linear layers are marked on their top.

The training dataset $\mathcal{D}_{train}$, the validation dataset $\mathcal{D}_{valid}$ and the test dataset $\mathcal{D}_{test}$. The test dataset $\mathcal{D}_{test}$ is reserved exclusively for the final evaluation. The training dataset $\mathcal{D}_{train}$ is employed for training the network, while the validation dataset $\mathcal{D}_{valid}$ is utilized for hyperparameter tuning and model selection by assessing performance during the training process. As the CIFAR-10 dataset is balanced across classes and contains no missing values, the pipeline bypasses any missing value processing steps.

3.3. Model training

3.3.1 Model architecture

In this assignment, three CNN-based network architectures are employed for evaluation: LeNet, VGG, and ResNet. Each architecture has distinct structural characteristics and has proven effective for image classification tasks. Fig.2 shows the general framework of three proposed architecture used in this report.

LeNet [5] is one of the pioneering CNN architectures, originally designed for handwritten digit recognition. The model consists of two convolutional layers, each followed by a subsampling (pooling) layer, and three fully connected layers. Due to its straightforward architecture and historical importance, LeNet serves as the baseline in this study, providing a foundational comparison point for more complex models.

VGG-16 [10] network is known for its deep, homogeneous design. VGG utilizes multiple stacked 3x3 convolutions, which increases the depth of the network while maintaining manageable computational complexity. Each block

Figure 3: **Examples of data augmentation strategies applied in this study. [6]** The top row ("Geo") demonstrates geometric transformations using the RandomCrop + RandomHorizontalFlip function, where images undergo random geometric transform to introduce perspective variations. The bottom row ("Color") illustrates color adjustments using the ColorJitter function, with random alterations to brightness, contrast, and hue, enhancing the network's ability to generalize to diverse lighting and color conditions.

of convolutional layers is followed by a max-pooling layer, reducing the spatial resolution and allowing the network to focus on essential features. In this assignment, VGG is selected for its layered sampling technique. Although the original VGG model is designed for images with larger resolution, it still can be applied on the 32×32 dataset.

ResNet-18 [1] addresses the issue of vanishing gradients in deep networks through the use of residual connections, allowing information to bypass certain layers. This residual learning approach enables the training of very deep networks by using "identity shortcuts" that skip one or more layers, making it easier to optimize deeper architectures. This structure has made ResNet a foundational element in many state-of-the-art models. In this assignment, ResNet's residual framework is particularly valuable, as it improves model accuracy and stability, providing insights into how deep networks can perform under limited data conditions.

3.3.2 Training parameter selection

Since each network has a unique architecture, different combinations of training parameters, such as batch size, optimizer, and learning rate, are required for optimal performance. Considering the time constraints for network training, a comprehensive parameter search was not conducted; instead, parameter choices were informed by established deep learning practices and prior training experience with similar models.

Optimizer: For all networks, the Adam optimizer was chosen as the training optimizer due to its adaptive learning rate capabilities and its widespread success in CNN-based image classification tasks. This optimizer allows each model to adjust learning rates automatically, enhancing convergence stability and performance across diverse network structures.

Batch size: A batch size of 128 was applied uniformly across all networks. This batch size was feasible because the image data used in this study is relatively small, allowing the GPU memory to accommodate the increased batch size without significant overhead. A larger batch size also benefits training stability by providing a more comprehensive gradient estimation with each update, reducing the variance in parameter adjustments.

Learning rate: For the learning rate, which has a critical impact on training performance, a cyclic learning rate policy was adopted to identify the initial learning rate. This initial experiment progressively adjusts the learning rate to identify an effective range and schedule. The details of this learning rate setting are outlined in the Sec.4.2.

Loss function. As the dataset includes labeled ground truth for each class, cross-entropy loss was used as the training loss function. Given the network output y_N and the groundtruth y_{gt}, the cross entropy loss $\mathcal{L}$ is calculated as follows:

$$\mathcal{L} = -\log \frac{\exp \rho(y_N, y_{gt})}{\sum_{i=1}^{K} \exp \rho(y_N, i)}, \tag{3}$$

where $\rho(y_N, i)$ denotes the predicted score class i from the network output y_N.

3.3.3 Improvements

To further enhance the performance of the networks, some modifications were implemented in both the training process and network structure. These adjustments included Data Augmentation and the incorporation of Dropout Layers. Data Augmentation was employed to artificially expand the training dataset by applying random transformations, which help to improve generalization by exposing the network to a broader variety of input variations. Fig. 3 shows the examples of image augmentation by geometry transformation and color variation [6]. Dropout Layers were introduced within the network architectures to prevent overfitting by randomly deactivating a subset of neurons during each training iteration, thereby promoting more robust feature learning. This study explored the impact of these optimizations on network performance. Different levels of augmentation and dropout probabilities were tested to evaluate their effects on the final model accuracy and generalization capabilities. Details on the specific implementation and results of these optimizations are discussed in Sec.4.4.

Model	Acc. (%)	Prec. (%)	F1. (%)
LeNet	46.06	45.69	45.18
LeNet-imp	10.00	10.00	10.00
VGG	69.09	69.40	69.18
VGG-imp	72.73	73.46	72.84
ResNet	67.66	68.17	67.62
ResNet-imp	**80.38**	**80.71**	**80.44**

Table 1: **Performance comparison between baseline and improved models.** The metrics include Accuracy (Acc.), Precision (Prec.), and F1 Score (F1). Improvements with data augmentation and dropout are evident for VGG and ResNet, with ResNet-imp achieving the highest overall performance. However, the modifications had an adverse effect on LeNet, as shown by the significantly reduced scores across all metrics.

3.4. Performance evaluation

After completing the model training, performance is evaluated on the test dataset $\mathcal{D}_{test}$. To prevent data leakage, for all training parameters setting and improvements above, only the validation dataset $\mathcal{D}_{valid}$ and $\mathcal{D}_{train}$ are used.

4. Experiments

In this section, the experimental setup is detailed, and the effectiveness of the improvement methods is demonstrated.

4.1. Experimental Details

The CIFAR dataset was obtained from the official source [4]. The dataset contains 60000 images of 10 classes with the resolution of 32×32 in total. There are 5 batches of training data and one batch of test data in the CIFAR-10. In our experiments, we use batch $1-4$ as the training dataset $\mathcal{D}_{train}$, the batch 5 as the validation dataset $\mathcal{D}_{valid}$, and the test batch as the test dataset $\mathcal{D}_{test}$.

All experiments were conducted on a GPU platform (RTX2080Ti), and the neural network implementation was carried out using PyTorch[8].

4.2. Parameter setting

Base on empirical observations, a batch size of 128 was used for all models, paired with the Adam optimizer [3] and its default parameters ($\beta_1 = 0.9, \beta_2 = 0.99$). These settings were chosen to ensure a stable and effective training process while accommodating the computational constraints of the available hardware.

To determine an appropriate learning rate for each network, an experimental approach was taken by utilizing cyclical learning rate (CLR) [11]. This method involves varying the learning rate cyclically within a specified range, allowing the model to explore different learning rates dynamically during training. This is particularly effective for quickly identifying suitable learning rates without extensive grid search. By tuning the learning rate from a small value (1×10^{-10} and gradually increasing that, the suitable learning rate can be identified. The learning rate schedules and corresponding training curves for each model are shown in Fig. 5.

Based on the insights gained from the CLR approach, starting learning rates of 10^{-2} were selected for all three networks. This initial rates were found to provide stable convergence while accommodating the varying depths and complexities of each network. For subsequent training, this learning rate will be decreased by half if there is no improvement in validation accuracy for 50 epochs. This reduction allows for fine-tuning as the models approached optimal convergence. By following this dynamic adjustment strategy, each model was effectively trained with an appropriate learning rate throughout the training process.

4.3. Main Experiments

Model performance was evaluated using three metrics on the test dataset: accuracy, precision, and f1 score. The recall metric is not included, as all classes are balanced, in which case it will always be equal to accuracy in a multi-class setting. For the final comparison, two models were assessed: The original baseline model, which has the fixed training setting but no dropout nor data augmentation is added, and the model with improvements (referred to as *-imp). The quantitative results are presented in Table 1. Additionally, confusion matrices for both models are provided in Fig.4 to offer a more detailed visualization of their performance.

4.4. Ablation Study

During the training process, overfitting was observed in the models, especially in VGG and ResNet, where near-perfect accuracy was reached on the training data relatively early. To mitigate this overfitting, two strategies were applied in this assignment: Dropout and Data Augmentation. This section presents ablation experiments conducted to address the specific impact of these methods on model performance. Notice that all the metrics were measured on both the validation dataset $\mathcal{D}_{valid}$ and test dataset $\mathcal{D}_{test}$. Validation results are particularly relevant for model selection, while test results offer insight into the practical performance of each strategy.

Firstly, experiments were conducted using Dropout. Dropout layers were added at specific locations in the network (as illustrated in Fig.2). The dropout rates vary from 0.0 (no dropout) to 0.5. The baseline model without dropout is indicated by $p = 0.0$. Tab. 2 shows the results of the dropout experiments. For VGG and ResNet, a dropout rate of 0.3 led to the highest validation accuracy and test accu-

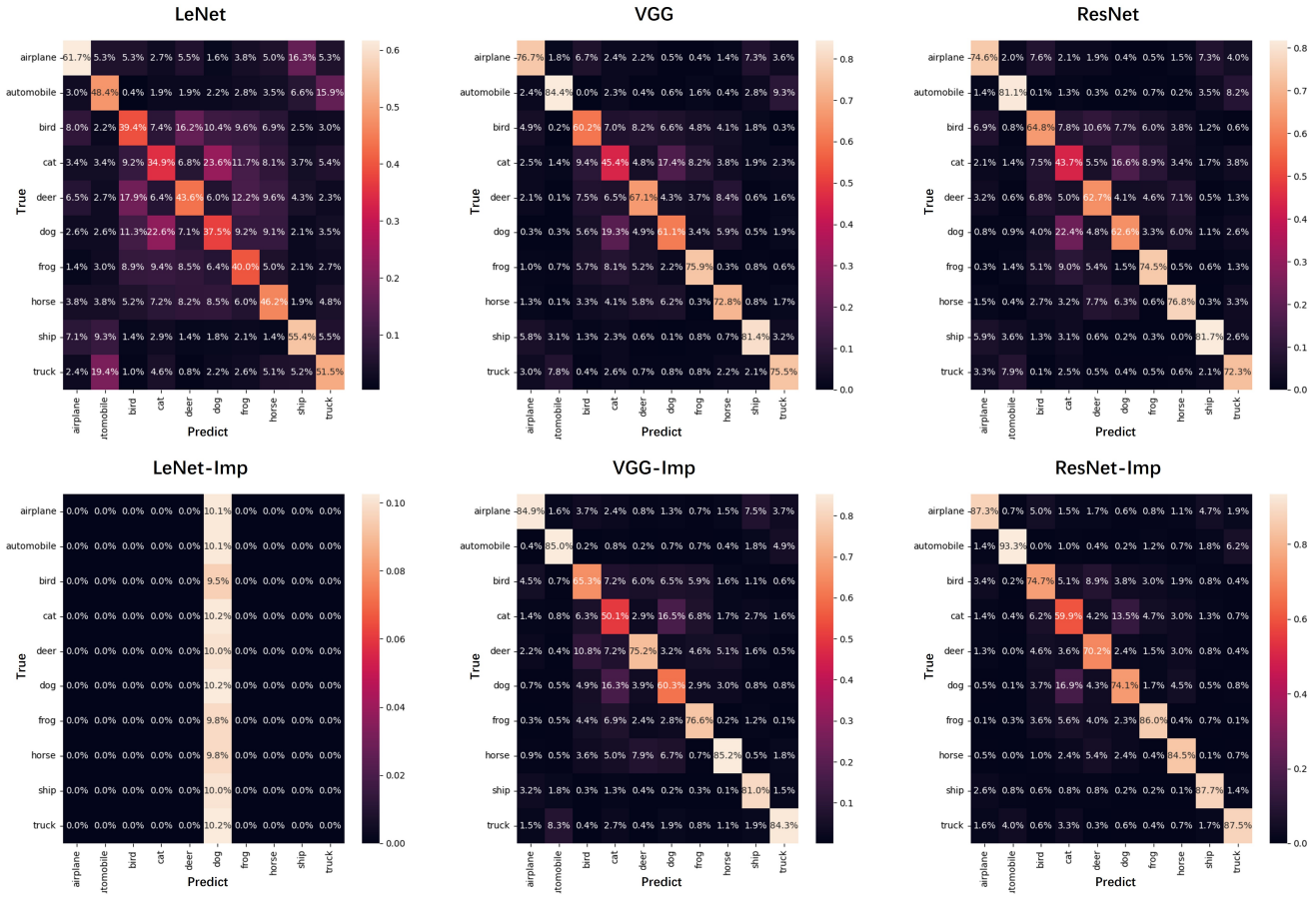

Figure 4: **Confusion matrices for models on CIFAR-10 test data.** The top row shows baseline results, while the bottom row (denoted by "-Imp") shows improved models with dropout and data augmentation applied. Each matrix displays the distribution of predicted labels against true labels, highlighting areas where models commonly misclassify certain classes. Brighter colors indicate higher classification accuracy within each class, and darker colors reveal misclassification results.

Dropout	Validation ($\mathcal{D}_{valid}$)			Test($\mathcal{D}_{test}$)		
	Acc. (%)	Prec. (%)	F1. (%)	Acc. (%)	Prec. (%)	F1. (%)
LeNet (0.0)	46.06	45.84	45.26	46.06	45.69	45.18
LeNet (0.1)	10.00	1.03	1.86	10.00	1.00	1.82
LeNet (0.3)	43.28	43.21	42.59	43.60	43.61	42.99
LeNet (0.5)	10.00	1.03	1.86	10.00	1.00	1.82
VGG (0.0)	69.76	70.05	69.86	69.09	69.40	69.18
VGG (0.1)	74.15	73.94	73.99	73.18	72.99	73.04
VGG (0.3)	**75.14**	**75.16**	**75.03**	**74.02**	**74.00**	**73.84**
VGG (0.5)	67.13	67.27	66.73	65.96	66.15	65.51
ResNet (0.0)	68.94	69.48	68.96	67.66	68.17	67.62
ResNet (0.1)	70.32	70.29	70.27	69.13	69.12	69.06
ResNet (0.3)	72.10	72.44	72.14	70.87	71.17	70.86
ResNet (0.5)	66.88	67.40	67.00	66.31	66.81	66.43

Table 2: **The ablation study for dropout on validation and test datasets.** Different dropout rates (0.0, 0.1, 0.3, and 0.5) were applied to observe their effects on accuracy, precision, and F1 scores. Moderate dropout rates, particularly 0.3, improved performance for deeper models like VGG and ResNet, while LeNet showed less benefit.

Data Aug	Validation ($\mathcal{D}_{valid}$)			Test($\mathcal{D}_{test}$)		
	Acc. (%)	Prec. (%)	F1. (%)	Acc. (%)	Prec. (%)	F1. (%)
LeNet (none)	46.06	45.84	45.26	46.06	45.69	45.18
LeNet (geo)	31.42	31.30	30.42	32.06	31.74	30.93
LeNet (color)	10.00	1.03	1.86	10.00	1.00	1.82
LeNet (all)	28.87	30.01	27.29	29.28	31.10	27.98
VGG (none)	69.76	70.05	69.86	69.09	69.40	69.18
VGG (geo)	67.85	70.46	68.30	68.59	71.23	69.05
VGG (color)	68.39	69.05	68.60	67.39	68.00	67.60
VGG (all)	70.73	71.15	70.88	70.53	70.79	70.60
ResNet (none)	68.94	69.48	68.96	67.66	68.17	67.62
ResNet (geo)	78.95	79.24	78.97	78.87	79.16	78.90
ResNet (color)	70.70	70.79	70.69	69.99	70.13	69.99
ResNet (all)	**81.04**	**80.96**	**80.95**	**80.86**	**80.73**	**80.76**

Table 3: **The ablation study for data augmentation on validation and test datasets.** Different augmentation strategies were tested: none, geometric transformations (geo), color adjustments (color), and a combination of both (all). VGG and ResNet achieved the best performance with combined augmentations, indicating improved generalization on test data.

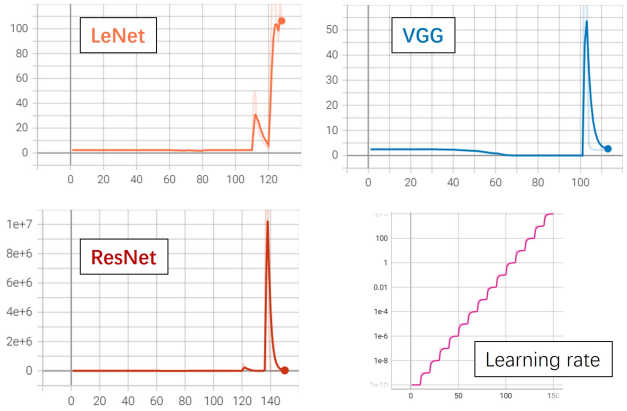

Figure 5: **Loss curves and learning rate schedule for determining optimal starting learning rates.** The top-left, top-right, and bottom-left plots show loss curves for LeNet, VGG, and ResNet, respectively, highlighting points where the learning rate becomes too high. The bottom-right plot illustrates the cyclical learning rate schedule used to explore effective starting rates during training.

racy, suggesting that moderate dropout rates are effective in reducing overfitting. In contrast, for LeNet, the addition of dropout resulted in reduced performance, indicating that simpler models may not benefit from dropout as much as deeper architectures.

Secondly, an ablation study was conducted on Data Augmentation. Two types of augmentation were applied: (1) ***-color**: Random adjustments to brightness, contrast, and hue of the images, and (2) ***-geo**: Random geometric transformations to increase the model's robustness to varied perspectives. These augmentations were implemented using functions from Torchvision, with examples shown in Fig. 3. The results are displayed in Table 3. For VGG and ResNet, the combination of geometric and color augmentations ("all") achieved the best results, demonstrating that complex models benefit significantly from diverse data augmentation strategies.

5. Code

The code for this project is available on my GitHub page (`https://github.com/MorainingErin/COMP_SCI_7318_DLF/tree/main/A2`). A readme file is also attached with the project for better understanding.

6. Conclusion

This study evaluated the performance of three CNN architectures—LeNet, VGG, and ResNet—on the CIFAR-10 image classification task. Using a set of training parameters, a baseline comparison was established to assess each network's generalization capability on limited sample sizes. Data Augmentation and Dropout Layers were introduced to enhance model robustness, showing promising improvements in generalization and reducing overfitting. Among the tested models, ResNet demonstrated notable resilience in maintaining accuracy under constrained data conditions, highlighting the advantages of its residual learning framework. These results underscore the significance of architecture selection in resource-limited environments and suggest directions for further optimization.

Limitations: This study has several limitations. First, due to computational constraints, only a limited set of

network architectures and parameter configurations were explored, potentially restricting the performance for each model. Additionally, while basic data augmentation techniques were applied, more advanced augmentation strategies could further improve generalization. Lastly, the reliance on the CIFAR-10 dataset, which consists of relatively low-resolution images, may limit the applicability of these findings to more complex datasets with higher-resolution images. Future research could address these limitations by investigating a wider range of architectures, experimenting with advanced data augmentation techniques, and testing on larger, more diverse datasets.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition, (CVPR)*, pages 770–778, 2016.

[2] Mahbub Hussain, Jordan J Bird, and Diego R Faria. A study on cnn transfer learning for image classification. In *Advances in Computational Intelligence Systems: Contributions Presented at the 18th UK Workshop on Computational Intelligence*, pages 191–202, 2019.

[3] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[4] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

[5] Yann. Lecun, Leon. Bottou, Yoshua. Bengio, and Patrick. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the ACM International Conference on Multimedia*, page 1485–1488, 2010.

[7] The University of Adelaide. Course outlines: COMP SCI 7318 - Deep learning fundamentals. `https://www.adelaide.edu.au/course-outlines/110026/1/sem-2/2020/`, 2024. [Accessed 01-10-2024].

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019.

[9] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[11] Leslie N Smith. Cyclical learning rates for training neural networks. In *IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[13] Wei Wang, Yujing Yang, Xin Wang, Weizheng Wang, and Ji Li. Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering*, 58(4):040901–040901, 2019.