Object Recognition with the CIFAR-10 Dataset

Alexander Lara, Kyrian Adimora, Ziyu Zhu, Junyi Zhao

Abstract

This project aims to develop and compare the performance of two deep learning models, a Convolutional Neural Network (CNN) and a Residual Network (ResNet), for the task of image classification on the CIFAR-10 dataset. The CIFAR-10 dataset consists of 60,000 32x32 color images divided into 10 classes, with 50,000 images for training and 10,000 for testing. The CNN model follows the LeNet-5 architecture, while the ResNet model is based on the ResNet-18 architecture. Various data preprocessing techniques, such as random horizontal flipping, random vertical flopping, normalization, and batching, are employed to enhance the performance of the ResNet model. The project evaluates the accuracy of both models on the CIFAR-10 dataset and compares their performance. The results provide insights into the effectiveness of these deep learning architectures for image classification tasks and serve as a baseline for further improvements and applications.

1. Introduction

Image classification is a fundamental task in computer vision and has numerous real-world applications, ranging from object detection and recognition to autonomous driving and medical imaging. With the advent of deep learning techniques, convolutional neural networks (CNNs) and their variants have become state-of-the-art models for image classification tasks [1]. However, as the complexity of these models increases, training them effectively becomes a challenge, necessitating the development of advanced architectures and techniques. This project focuses on the implementation and evaluation of two deep learning models, a CNN and a ResNet, for image classification on the CIFAR-10 dataset. The CIFAR-10 dataset [2] is a widely used benchmark dataset in computer vision, consisting of 60,000 32x32 color images divided into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

The CNN model is based on the LeNet-5 architecture [3], a pioneering CNN model introduced in the 1990s. Despite its simplicity, LeNet-5 serves as a baseline for comparison and provides insights into the performance of a traditional CNN architecture on the CIFAR-10 dataset.

On the other hand, the ResNet model is an advanced CNN architecture that incorporates residual connections [4], which have been shown to improve the training of deep neural networks by mitigating the vanishing gradient problem. Specifically, the project implements the ResNet-18 architecture, which consists of several residual blocks and has proven effective in various image classification tasks.

To further enhance the performance of the ResNet model, the project employs various data preprocessing techniques, such as random horizontal flipping, random vertical flipping, normalization, and batching. These techniques aim to introduce additional data variations and regularization, potentially improving the model's generalization capabilities.

2. Related Works

Several works have explored the application of CNNs to image classification tasks. PyTorch provides useful tutorials and tools for working with datasets like CIFAR-10 [5]. Additionally, custom dataset creation and preprocessing techniques have been discussed extensively [6]. ResNet architectures have been studied for

their effectiveness in image recognition tasks, demonstrating improved performance over traditional CNNs [7, 8]. The LeNet-5 architecture, known for its simplicity, serves as a foundational model for comparison in this study [9]. Furthermore, Radford et al. introduced deep convolutional generative adversarial networks (DCGANs) as a strong candidate for unsupervised learning, bridging the gap between supervised and unsupervised learning with CNNs [10]. Convolutional Neural Networks (CNNs) have been extensively explored for image classification tasks, with various architectures and techniques proposed to improve their performance. The CIFAR-10 dataset has served as a popular benchmark for evaluating and comparing these methods.

One of the pioneering works in this domain is AlexNet [11], a deep CNN that achieved breakthrough results on the ImageNet dataset. Subsequently, architectures like VGGNet [12], GoogLeNet [13], and ResNet [4] have been developed, introducing deeper and more efficient network structures. Among these, ResNet has been widely adopted and has demonstrated state-of-the-art performance on the CIFAR-10 dataset. For instance, the ResNet-110 model achieved an error rate of 6.43% [14], outperforming previous architectures like Network in Network (7.25% error rate) [15] and All-Convolutional Net (9.08% error rate) [16]. In addition to architectural innovations, various data augmentation and preprocessing techniques have been explored to improve the performance of CNNs on the CIFAR-10 dataset. These include random cropping, horizontal flipping, color jittering, and cutout regularization [17]. Furthermore, ensemble methods that combine multiple models have been shown to further boost accuracy. For example, the Shake-Shake regularization technique [18], which consists of an ensemble of ResNet models, achieved an impressive 2.86% error rate on CIFAR-10.
While CNNs have dominated image classification tasks, other approaches like Capsule Networks [19] and Vision Transformers [20] have also been investigated. Capsule Networks, which aim to better model spatial relationships and viewpoint invariance, achieved a competitive 3.82% error rate on CIFAR-10 [19]. Vision Transformers, which apply the transformer architecture to image data, have also shown promising results, with the ViT-B/16 model achieving a 4.6% error rate on CIFAR-10 [20].
This project aims to contribute to the ongoing research in image classification by exploring and comparing the performance of two widely used architectures, CNNs and ResNets, on the CIFAR-10 dataset. By employing various data preprocessing techniques and leveraging the strengths of residual connections, the project seeks to improve upon the baseline performance of traditional CNNs and provide insights into the effectiveness of these architectures for this task.

3. Dataset and Formation

Data Collection
Torchvision allows a direct download of the dataset as part of its library. The dataset is pre-split into training and testing. The ratio of training to testing is 5 to 1. There are ten classes: plane, car, bird, cat, deer, dog, frog, horse, ship, and truck. In the training set, each class has 5000 different pictures, and the testing set has 1000 different pictures.
The project further split the training set into training and validation sets for the ResNet-18 model. We applied a validation set split of 5000 images per class to total 45,000 training pictures and 5,000 validation pictures. Below is an example of 4 random images from the training set. Each picture is 3 color channels (RGB), and the resolution is 32 x 32.

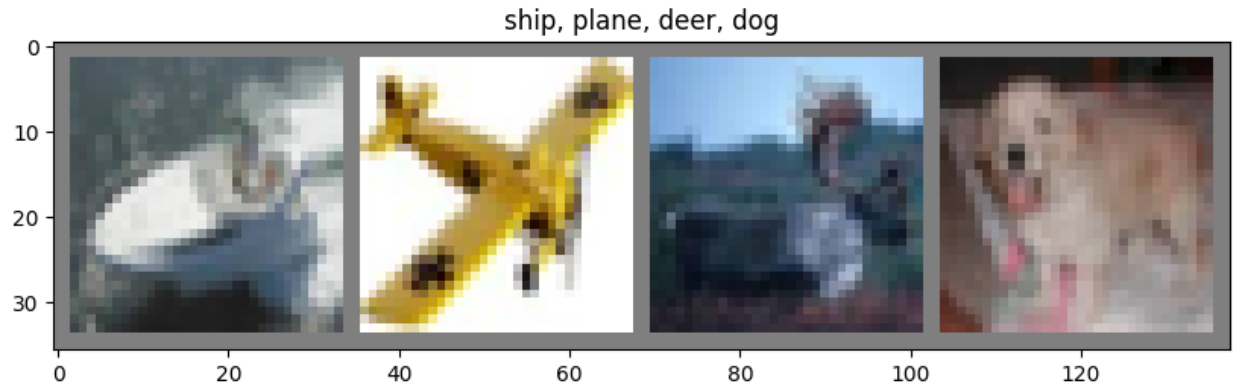

Figure 1. True label of 4 random images

Exploratory Data Analysis

Prior to training classification models, we created histograms of the average pixel intensities per color channel by class. The train and test set histograms produced similar histograms by class. The shape of the histograms tended to look uniformly distributed on the dark half of the pixel intensity range. On the lighter half of the pixel intensities, the distributions have a variety of shapes, but most spiked near the 255-pixel value. Figure 2 is an example of the histogram comparison for the cat class.

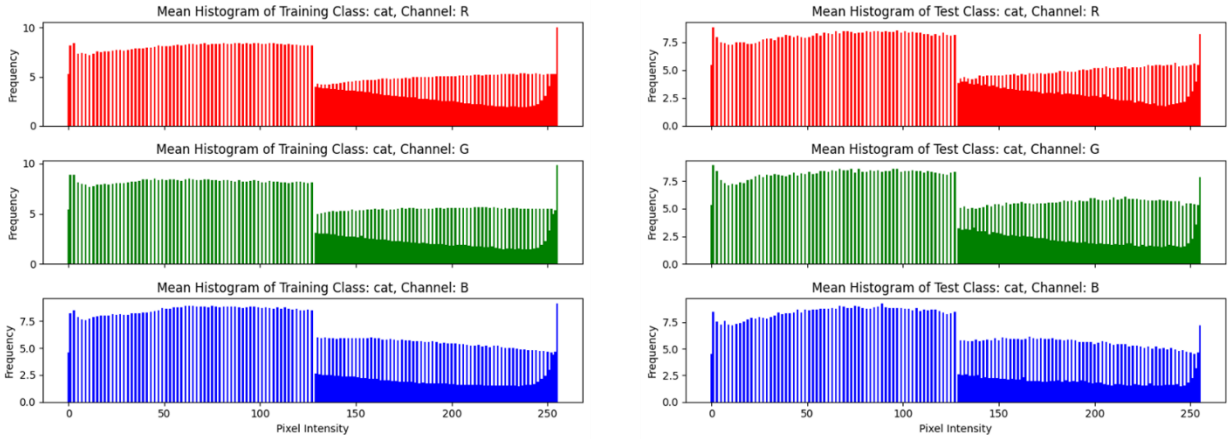

Figure 2. Histogram comparison for the cat class.

Data Preprocessing

To enhance the performance of the models and improve their generalization capabilities, several data preprocessing techniques were employed:

a. **Data Augmentation:** To increase the diversity of the training data and prevent overfitting, various data augmentation techniques were applied, including random horizontal flipping, and vertical flipping.
b. **Normalization:** The pixel values of the input images were normalized to have a mean of .5 and a standard deviation of .5. This step is crucial for ensuring numerical stability during training and improving convergence.
c. **Resize:** The data was resized from 32x32 resolution to 224 x 224. The resize is a necessary step for the RESNET – 18 architecture.
d. **Conversion to Tensor:** The preprocessed images were converted to PyTorch tensors, the standard data format for deep learning models in PyTorch.

e. **Batching and Shuffling:** To improve training efficiency and introduce additional randomness, the preprocessed data was batched into smaller subsets and shuffled before each training epoch.

These preprocessing steps applied to both the training and validation datasets to ensure consistency and enable fair evaluation of the models' performance. Data Augmentation did not apply to the test set, The other preprocessing steps were applied to the test set.

4. Methods

A. Convolutional Neural Network (CNN)

The CNN model used in this project follows the LeNet-5 architecture [3], a pioneering CNN model introduced in the 1990s. The specific implementation details of the LeNet-5 architecture used in this project are as follows:

1. **First Convolution Layer:** The input image first passes through a convolutional layer with 6 filters of size 5x5 and a stride of 1.
2. **Max Pooling Layer:** The output from the first convolutional layer is then passed through a max-pooling layer with a 2x2 kernel and a stride of 2.
3. **Second Convolution Layer:** The pooled output is then passed through another convolutional layer with 16 filters of size 5x5 and a stride of 1.
4. **Flattening:** The output from the second convolutional layer is flattened into a single vector.
5. **First Linear Layer:** The flattened vector is passed through a fully connected linear layer with 120 output units.
6. **Second Linear Layer:** The output from the first linear layer is then passed through another fully connected linear layer with 84 output units.
7. **Third Linear Layer:** The final linear layer has 10 output units, corresponding to the 10 classes in the CIFAR-10 dataset.

B. Residual Network (ResNet)

The ResNet model implemented in this project is based on the ResNet-18 architecture, which incorporates residual connections to improve the training of deep neural networks. The specific architecture used is as follows:

- Initial Convolution Layer: 64 filters of size 5x5 with a stride of 1 and padding of 2.
- Four Residual Layers: Each residual layer consists of two blocks, with each block containing two convolutional layers. The number of output filters ranges from 64to 512.
- Supporting Layers: ReLU activation, average pooling, and normalization layers.
- Final Fully Connected Layer: A fully connected layer for classification.

C. Loss Function and Optimization

Both the CNN and ResNet models were trained using the cross-entropy loss function, which is a common choice for multi-class classification tasks. The cross-entropy loss measures the performance of the model by comparing the predicted probabilities with the true class labels. For CNN optimization, the Stochastic Gradient Descent (SGD) algorithm was used with a learning rate of 0.001 and a momentum of 0.9. SGD is a widely used optimization algorithm that updates the model's weights by computing the gradients of the loss function with respect to the weights. For ResNet optimization, the model used Adam optimizer. Adam optimizer combines AdaGrad and root mean square propagation for computational efficiency.

D. Evaluation Metrics

To assess the performance of the CNN and ResNet models on the CIFAR-10 dataset, the following evaluation metrics were used:

1. **Accuracy:** The overall accuracy of the model, calculated as the fraction of correctly classified samples over the total number of samples.
2. **Precision:** The fraction of true positive samples among the samples predicted as positive by the model.
3. **Recall:** The fraction of true positive samples that were correctly identified by the model.
4. **F1-Score:** The harmonic mean of precision and recall, providing a balanced measure of the model's performance.

These metrics were calculated for each class in the dataset, as well as for the overall performance of the models. The accuracy metric was used as the primary evaluation criterion, while precision, recall, and F1-score provided additional insights into the models' performance in individual classes.

5. Experiments

The project used the LeNet-5 CNN model for preliminary results [5, 9]. The CNN network architecture is as follows:

- Conv1: 6 filters, 5x5 kernel, stride 1
- MaxPool: 2x2 kernel, stride 2
- Conv2: 16 filters, 5x5 kernel, stride 1
- FC1 Linear: 400 inputs, 120 outputs
- FC2 Linear: 120 inputs, 84 outputs
- FC3 Linear: 84 inputs, 10 outputs
- Cross-Entropy Loss Function

LeNet provides a baseline to compare the more complex ResNet model. To train the model, we used a batch size of 4 images and 2 epochs. We used cross-entropy loss for the loss criterion and stochastic gradient descent with a learning rate of 0.001 and momentum of 0.9 to achieve the following train and test results shown in Table 1.

Table 1. Training and testing results for LeNet-5 model.

Class	Testing Accuracy (%)
Overall	54
plane	54.9
car	43.3
bird	43.5
cat	20.7
deer	61
dog	48.1
frog	54.1
horse	63.3
ship	75.1
truck	81.2

While the LeNet-5 model is simple, it performs better than random guessing (10% accuracy) and achieves an overall testing accuracy of 54% with only two training epochs. The model is not overfitted at this point, as the

training and testing accuracies are very similar. However, it is possible that the model is underfitted, and its performance could improve with more extensive hyperparameter tuning.

A. ResNet Results

For the ResNet model, we performed hyperparameter tuning to optimize its performance. The hyperparameters that were tuned include:

- **Learning Rate:** We experimented with learning rates ranging from 0.000005 to 0.001 and found that a learning rate of 0.00001 provided the best results.
- **Batch Size:** We tested batch sizes of 10, 25, 50, 64, 100 and 300, with a batch size of 25 yielding the fastest and highest accuracies.
- **Number of Epochs:** We trained the model for up to 100 epochs and observed that the model converged at different points depending on other parameters. The best model used 96 Epochs to train.
- **Data Augmentation:** No flipping or flopping produced the best testing results.

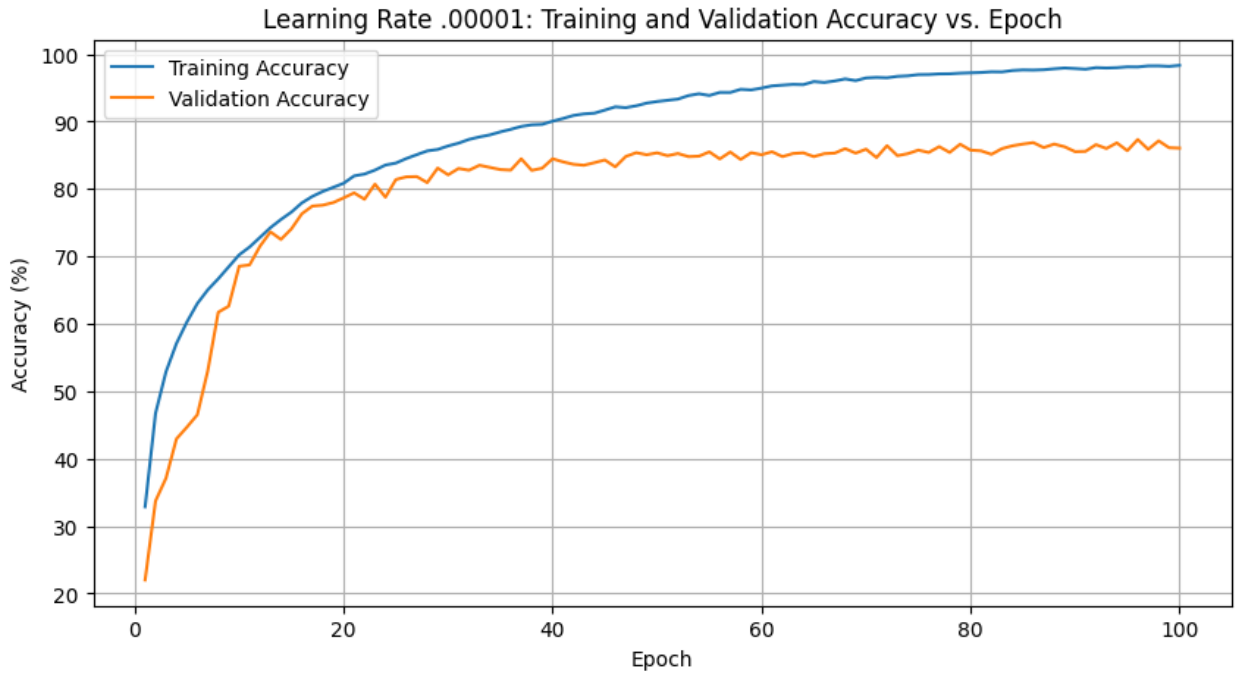

Figure 3. Train and validation accuracy versus Number of Epochs

By adding a validation test set, we visualize model convergence, while comparing underfit and overfit errors. Visual inspection shows the model plateaus at around 40 epochs for validation accuracy. However, our algorithm saves the best validation accuracy. Epoch 96 provides the best validation accuracy.

Table 2. Training and testing results for ResNet model

	ResNet Test Accuracy (%)	LeNet Test Accuracy(%)
Overall	84.67	54
plane	77.1	54.9
car	91	43.3
bird	83.8	43.5
cat	77.9	20.7

deer	78.2	61
dog	64	48.1
frog	89.9	54.1
horse	93.7	63.3
ship	95.7	75.1
truck	95.4	81.2

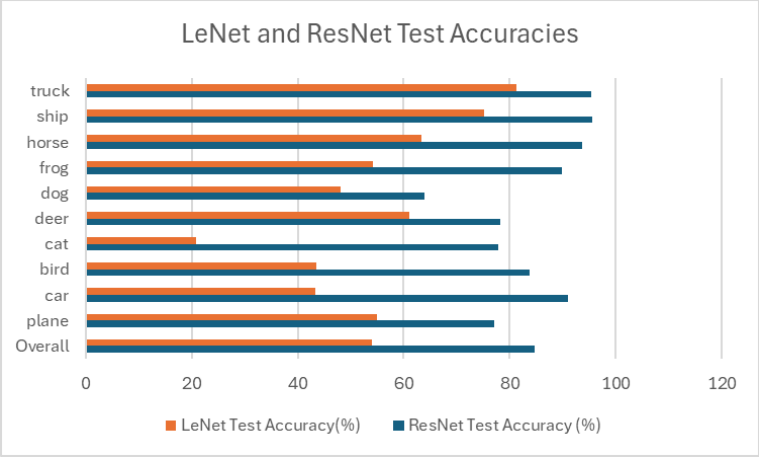

Figure 4: Comparison of the testing accuracies of LeNet-5 and ResNet models

From Table 2, the ResNet model significantly outperforms the LeNet-5 baseline, achieving an overall testing accuracy of 84.7% compared to 54% for LeNet-5. The ResNet model demonstrates improved performance across all classes, with some classes, such as 'cat' and 'bird', exhibiting substantial improvements over the baseline. Figure 3 provides a visual comparison of the testing accuracies for both models, highlighting the superior performance of the ResNet architecture.

B. Qualitative Analysis

In addition to the quantitative results, we performed a qualitative analysis of the models' performance by visualizing misclassified examples and examining the learned features. Figure 4 shows a sample of misclassified images from the ResNet model, along with their true and predicted labels.

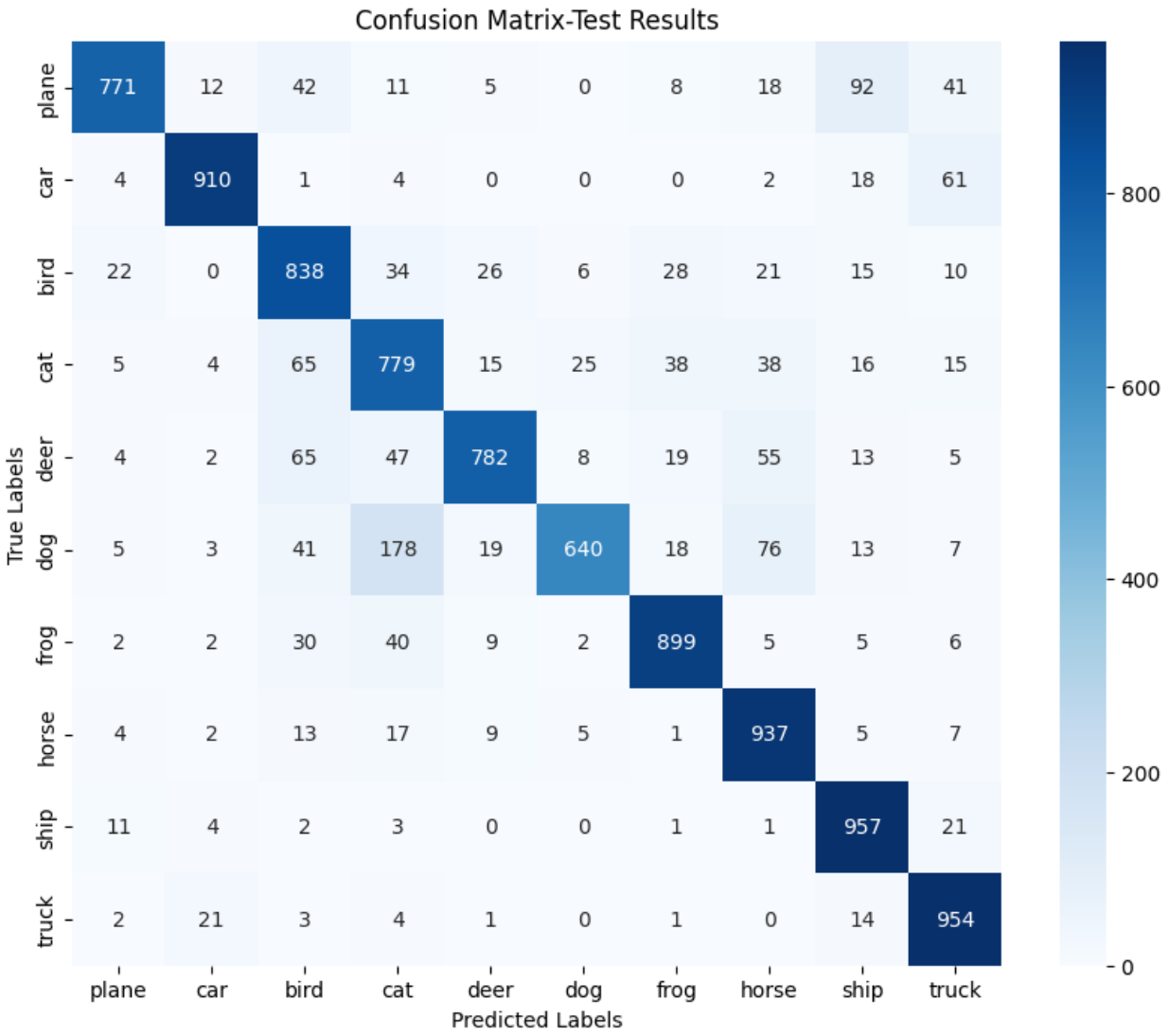

Figure 4: Confusion Matrix of Best ResNet Test Results

From these examples, we can observe that the model struggles with distinguishing between visually similar classes. The animals are more likely to be misclassified as other animals while the machines are more likely to be misclassified as other machines. The one animal and machine that are more likely to predict as each other are ‘bird’ and ‘plane’.

To gain insights into the learned features, we visualized the activations of the convolutional layers for a sample input image. Figure 5 shows the learned filters in the first convolutional layer of the ResNet model, which captures low-level features such as edges, textures, and simple patterns.

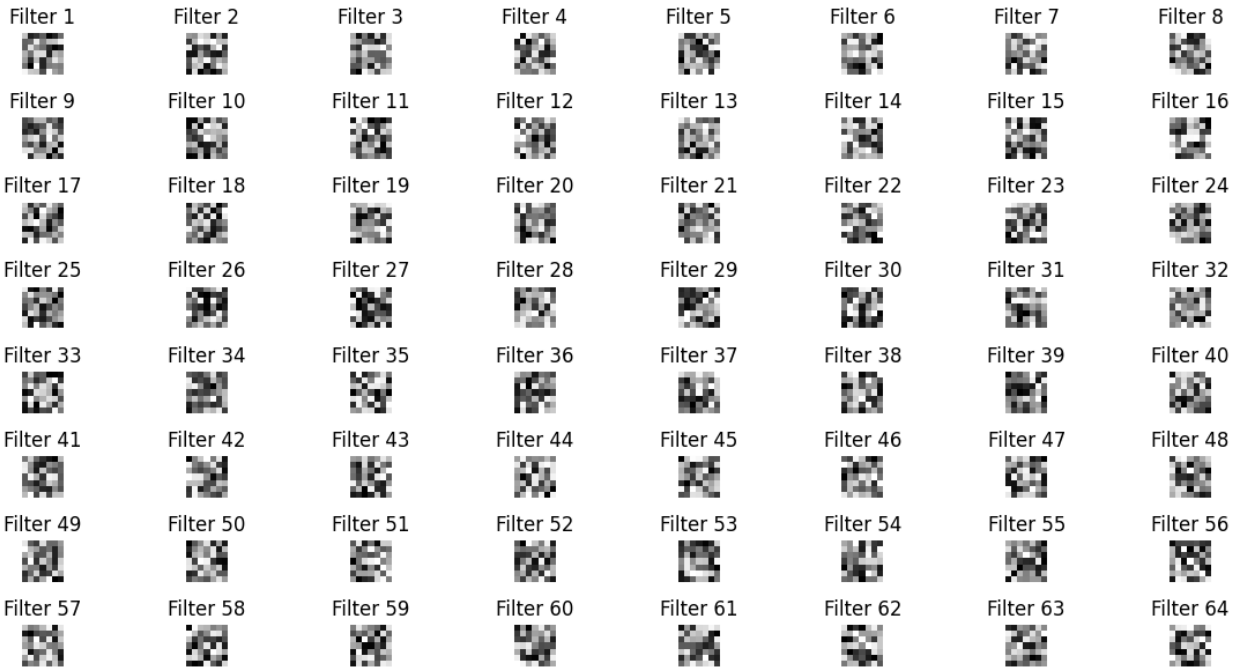

Figure 5: Visualization of learned filters in the first convolutional layer of the ResNet model.

Our visualization creates Figure 5 with 8 rows of 8 images, or 64 images. The dark squares indicate small or inhibitory weights, and the light squares represent large or excitatory weights. Using this intuition, we can see some patters like filter 1 and filter 53 are detecting some edges or corners. These qualitative analyses provide a better understanding of the strengths and limitations of the ResNet model and can inform future improvements, such as incorporating more diverse training data or exploring techniques for handling occlusions and viewpoint variations.

6. Conclusion

This project explored the application of deep learning models, specifically convolutional neural networks (CNNs) and residual networks (ResNets), for image classification on the CIFAR-10 dataset. The CNN model, based on the LeNet-5 architecture, served as a baseline, achieving an overall testing accuracy of 54% after only two training epochs. While this result outperforms random guessing, it highlights the need for more sophisticated architectures to improve performance further. The ResNet model, inspired by the ResNet-18 architecture, was implemented with various data preprocessing techniques, including random horizontal flipping, vertical flopping, normalization, and batching. These preprocessing steps aimed to introduce additional data variations and regularization, potentially enhancing the model's generalization capabilities. Flipping and flopping did not produce improvements for epochs under 100. Perhaps a different test set would require flipping and flopping but not in this study.

The performance of the ResNet model was evaluated using a comprehensive set of hyperparameters, including the number of epochs, loss function, and optimizer. The model's architecture, consisting of multiple residual blocks with skip connections, allowed for more effective training of deeper networks, mitigating the vanishing gradient problem.

Our study did differentiate hardware for training. The ResNet trained with GPU, while LeNet trained with a CPU. The model complexity requires a GPU for ResNet. So, in edge computing or IoT devices, the LeNet might provide an adequate model for image classification.

This project contributes to the ongoing research in image classification by providing a comparative study of two deep learning architectures on a widely used benchmark dataset. The insights gained from this project can inform future research directions and serve as a foundation for developing more sophisticated models tailored to specific applications in computer vision and beyond.
Furthermore, the implementation details and techniques employed in this project can be extended to other image classification tasks or adapted to different domains, such as natural language processing or speech recognition, where deep learning techniques have shown remarkable success.

7. References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[2] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009, Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[5] PyTorch, "CIFAR-10 Tutorial," PyTorch Tutorials, Accessed: Apr. 22, 2024. [Online]. Available: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

[6] J. Alexander, "Building Custom Datasets for PyTorch Deep Learning Image Classification," Medium, Jun. 2020, Accessed: Apr. 22, 2024. [Online]. Available: https://medium.com/@joshuale/building-custom-datasets-for-pytorch-deep-learning-image-classification-29989971652d

[7] PyTorch, "Torchvision Models - ResNet," Accessed: May 8, 2024. [Online]. Available: https://pytorch.org/hub/pytorch_vision_resnet/

[8] Analytics Vidhya, "ResNet: Understand and Implement from Scratch," Medium, Sep. 2020, Accessed: Apr. 30, 2024. [Online]. Available: https://medium.com/analytics-vidhya/resnet-understand-and-implement-from-scratch-d0eb9725e0db

[9] B. Siddhesh, "LeNet-5 Architecture Explained," Medium, Jan. 2022. [Online]. Available:https://medium.com/@siddheshb008/lenet-5-architecture-explained-3b559cb2d52b

[10] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," arXiv:1511.06434 [cs], Jan. 2016, Accessed: May 8, 2024. [Online]. Available: http://arxiv.org/abs/1511.06434

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Adv. Neural Inf. Process. Syst., vol. 25, 2012.

[12] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556 [cs], Sep. 2014.

[13] C. Szegedy et al., "Going Deeper with Convolutions," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," in Computer Vision – ECCV 2016, Cham, 2016, pp. 630–645.

[15] M. Lin, Q. Chen, and S. Yan, "Network In Network," arXiv:1312.4400 [cs], May 2014.

[16] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for Simplicity: The All-Convolutional Net," arXiv:1412.6806 [cs], Apr. 2015.