ResNet50 Unleashed: Mastering CIFAR-10 Image Recognition

EE-627
Final Project

Hebbani Venkatasheshaiah Arun Kashyap
CWID: 20022803
Dept. of Mathematical Science
Stevens Institute of Technology
Hoboken, US
hkashyap1@stevens.edu

Outline

Introduction

- The project is designed to harness the power of ResNet50, a renowned deep learning model, for the purpose of object recognition within the CIFAR-10 dataset, aiming to significantly advance the accuracy and efficiency of image classification tasks.

- CIFAR-10 is chosen as the benchmark dataset due to its diverse array of images spread across ten distinct classes, making it a challenging yet invaluable dataset for testing the robustness and versatility of image recognition models.

- The selection of ResNet50 is strategic, given its proven track record in deep learning achievements for image classification, courtesy of its deep residual networks that alleviate the vanishing gradient problem, thereby facilitating the training of much deeper networks.

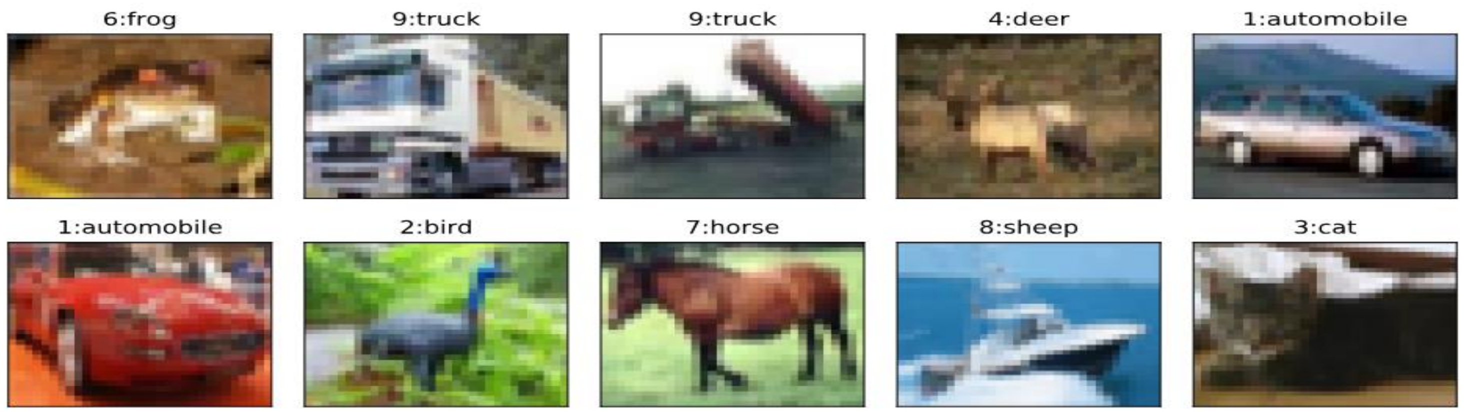


Figure 1: Sample images with their classes from CIFAR-10 Dataset

Dataset Description

- **Data Set Link: https://www.kaggle.com/c/cifar-10/data**
- **Composition:** 60,000 images across 10 classes for object recognition tasks.
- **Image Details:** Standard 32x32 pixel resolution in color.
- Classes: Include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.
- **Splits:** 50,000 training images and 10,000 test images, plus 290,000 non-class 'junk' images in the test set to prevent manual labeling bias.
- **Format:** Available on Kaggle in train.7z and test.7z (PNG files), with a trainLabels.csv for training image labels.

Implementation

Importing the Dataset
The CIFAR-10 dataset will be imported directly into Google Colab from Kaggle using the Kaggle API. This approach simplifies data handling and allows for the seamless integration of the dataset into our project environment.

Data Preprocessing

- **Dataset Acquisition**: The CIFAR-10 dataset is downloaded directly from Kaggle, which hosts the dataset in a compressed format.

- **Extraction and Preparation**: The compressed files, specifically train.7z, are extracted to obtain the training images in a format that can be readily processed.

- **Format Conversion**: The images, initially in a list or another format, are converted into NumPy arrays. This conversion facilitates easier manipulation and processing within data pipelines used in machine learning projects.

- **Normalization:** Each pixel value in the images, ranging from 0 to 255, is scaled down to a range between 0 and 1 by dividing by 255. This normalization step is crucial as it helps in speeding up the training process and improving the convergence of the neural network by providing a common scale for all input features.

Implementation

Numerical Encoding of Labels

- As part of the data preprocessing, the labels for the CIFAR-10 dataset, which denote the ten distinct classes of objects within the images, undergo a process of numerical encoding

- **Identification of Classes:** Each class within the CIFAR-10 dataset (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) is initially associated with a textual label.

- **Conversion to Numerical Encoding:** These textual labels are converted into numerical form, assigning a unique integer (ranging from 0 to 9) to each class. This numerical encoding facilitates the handling of labels by the neural network, allowing for efficient processing and prediction.

Model-1 (Neural Network) Specification

- The model is built using Keras, which is now integrated into TensorFlow, is used as the high-level neural networks API to construct and train the model.
- **Number of Classes:** The model is designed to classify data into 10 different categories, which corresponds to the CIFAR-10 dataset's 10 distinct classes of images.

Architecture:

- **Input Layer:** A Flatten layer that transforms the 2D image data of shape (32, 32, 3) into a 1D array without altering the data.

- **Hidden Layer:** A Dense layer with 64 neurons and ReLU (Rectified Linear Unit) activation function, which introduces non-linearity to the model, allowing it to learn more complex patterns in the data.

- **Output Layer:** Another Dense layer with 10 neurons (one for each class) using the softmax activation function to output a probability distribution over the 10 classes.

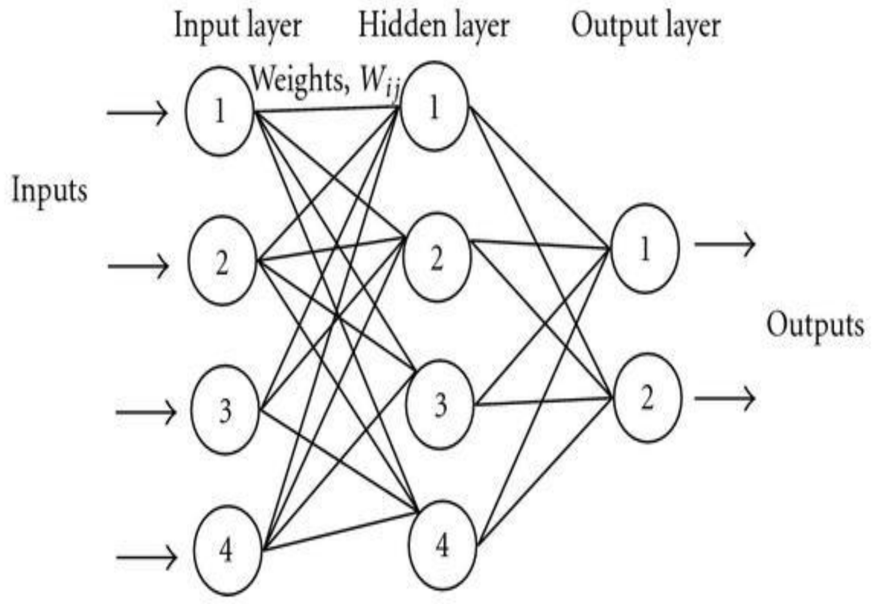


Figure 2: Basic 3-layer Neural Network

Compilation:

- Uses Adam optimizer and 'sparse_categorical_crossentropy' loss for classification.
- Performance measured by accuracy.

Training:

- Preprocessed data (X_train_normalized, Y_train) trained for 25 epochs.
- Uses 10% of data for validation.

Model-1 Validation

Model Performance:

- `Training Accuracy = 45.30%`
- `Training Loss = 1.533`
- `Test Accuracy = 42.17%`
- `Test Loss = 1.6485`

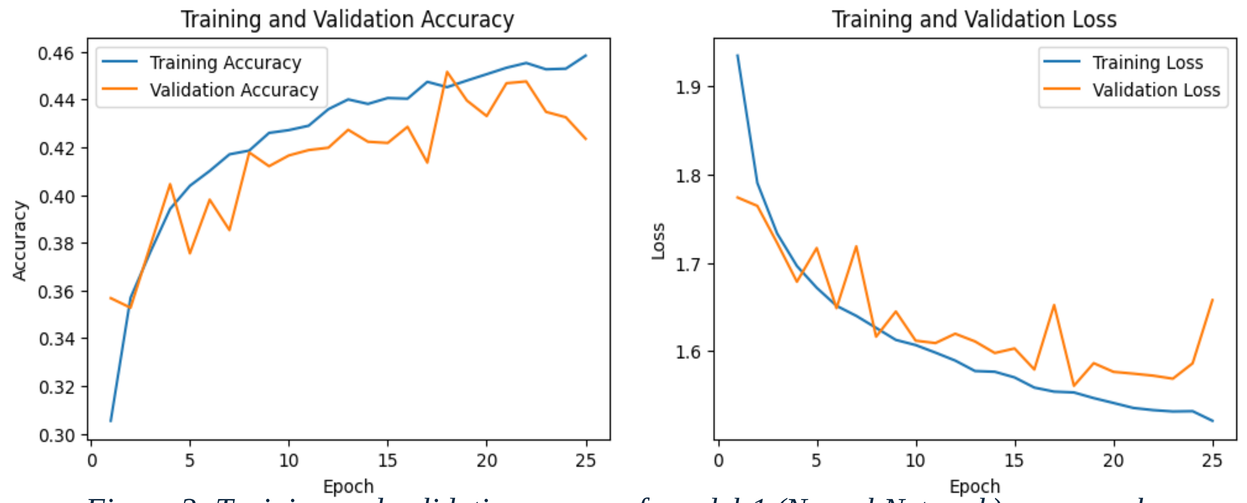
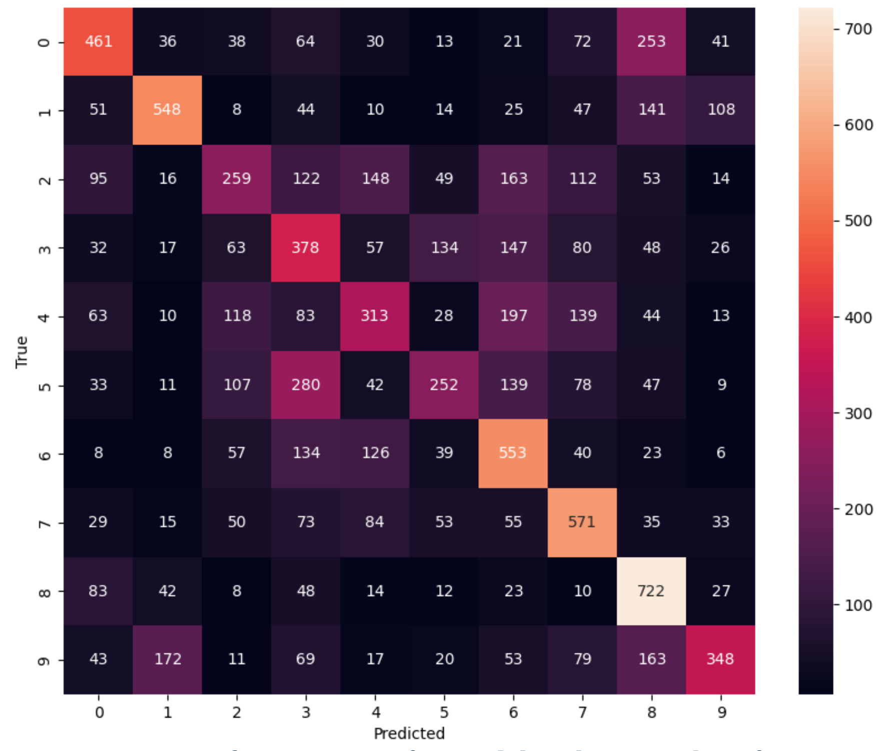


Figure 3: Training and validation curves of model-1 (Neural Network) over epochs



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	461	36	38	64	30	13	21	72	253	41
1	51	548	8	44	10	14	25	47	141	108
2	95	16	259	122	148	49	163	112	53	14
3	32	17	63	378	57	134	147	80	48	26
4	63	10	118	83	313	28	197	139	44	13
5	33	11	107	280	42	252	139	78	47	9
6	8	8	57	134	126	39	553	40	23	6
7	29	15	50	73	84	53	55	571	35	33
8	83	42	8	48	14	12	23	10	722	27
9	43	172	11	69	17	20	53	79	163	348

Figure 4: Confusion matrix for model-1 showing classification performance

- The model shows underfitting, with both training and validation accuracies below 46%.
- Training loss decreases steadily, but inconsistent validation loss suggests the model doesn't generalize well.
- Increasing model complexity could improve performance, indicating a move towards CNN or ResNet50 may be beneficial.

- The model accurately identifies certain classes but confuses others, suggesting good recognition ability for distinct features but difficulty distinguishing similar ones.
- Enhanced feature extraction and a more complex model structure may improve the misclassifications evident across several classes.

Model-2 (Convolution Neural Network) Specification

- **Convolutional Layers:** Starts with two 32-filter layers (3x3), both using ReLU for initial and refined feature detection.
- **Pooling and Dropout:** Includes max pooling for dimension reduction and dropout to combat overfitting.
- **Deeper Feature Extraction:** Adds 64-filter convolutional layers with ReLU, plus max pooling and dropout for complex pattern learning.
- **Fully Connected and Output Layers:** Connects a 512-neuron dense layer with ReLU, followed by dropout, and ends with a 10-neuron softmax output layer for class probability.

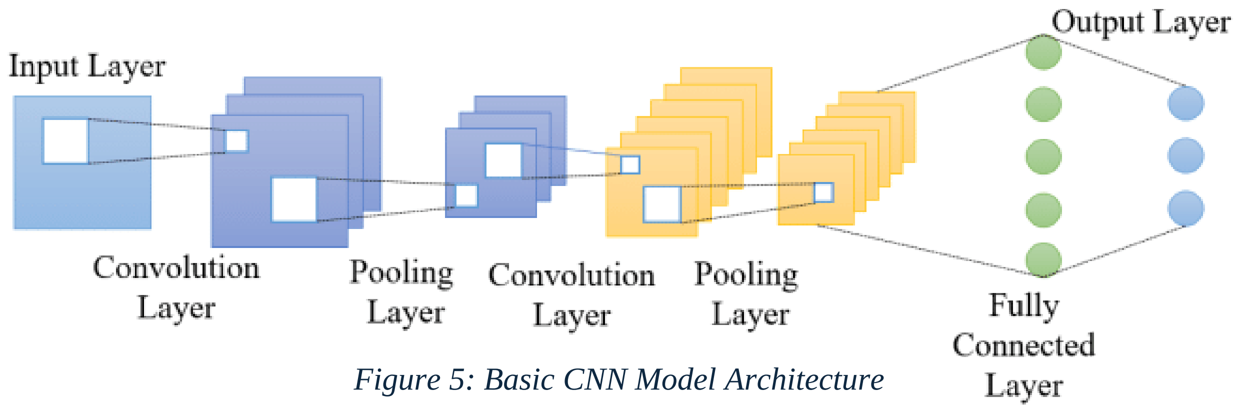


Figure 5: Basic CNN Model Architecture

Compilation:

- Uses Adam optimizer and 'sparse_categorical_crossentropy' loss for classification.
- Performance measured by accuracy.

Training:

- Preprocessed data (X_train_normalized, Y_train) trained for 25 epochs.
- Uses 10% of data for validation.

Model-2 Validation

Model Performance:

- `Training Accuracy = 77.06%`
- `Training Loss = 0.6462`
- `Test Accuracy = 74.11%`
- `Test Loss = 0.7397`

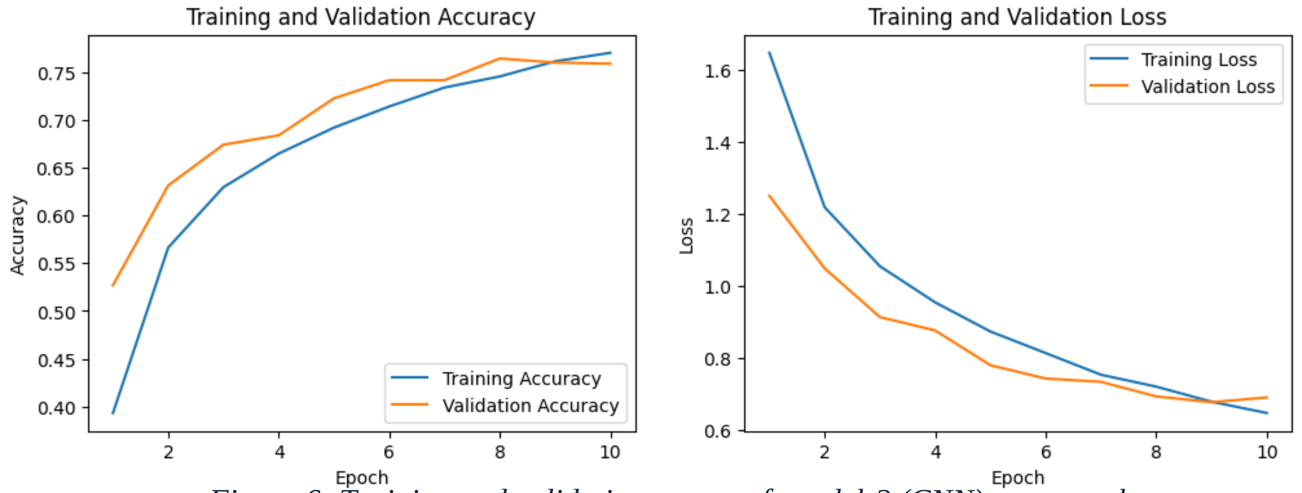
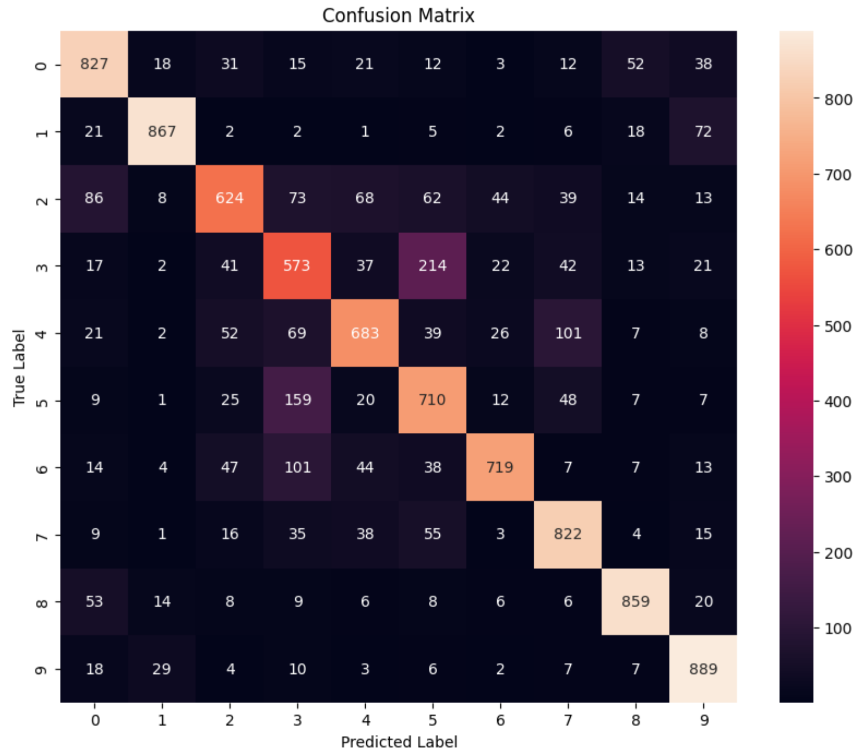


Figure 6: Training and validation curves of model-2 (CNN) over epochs



	0	1	2	3	4	5	6	7	8	9
0	827	18	31	15	21	12	3	12	52	38
1	21	867	2	2	1	5	2	6	18	72
2	86	8	624	73	68	62	44	39	14	13
3	17	2	41	573	37	214	22	42	13	21
4	21	2	52	69	683	39	26	101	7	8
5	9	1	25	159	20	710	12	48	7	7
6	14	4	47	101	44	38	719	7	7	13
7	9	1	16	35	38	55	3	822	4	15
8	53	14	8	9	6	8	6	6	859	20
9	18	29	4	10	3	6	2	7	7	889

Figure 7: Confusion matrix for model-2 showing classification performance

- The model's accuracy converges for training and validation, indicating a balanced fit.
- Both training and validation losses show a consistent decline, reflecting stable learning.
- The closeness of training and validation lines suggests that the model could be scaled up with more complex features to potentially enhance performance.

- The model shows high accuracy on several classes, with particularly strong performance on classes labeled 0 and 9.
- There are some confusions between classes, notably class 2 with classes 3 and 5, which may benefit from improved feature extraction or class-specific tuning.

Why ResNet-50

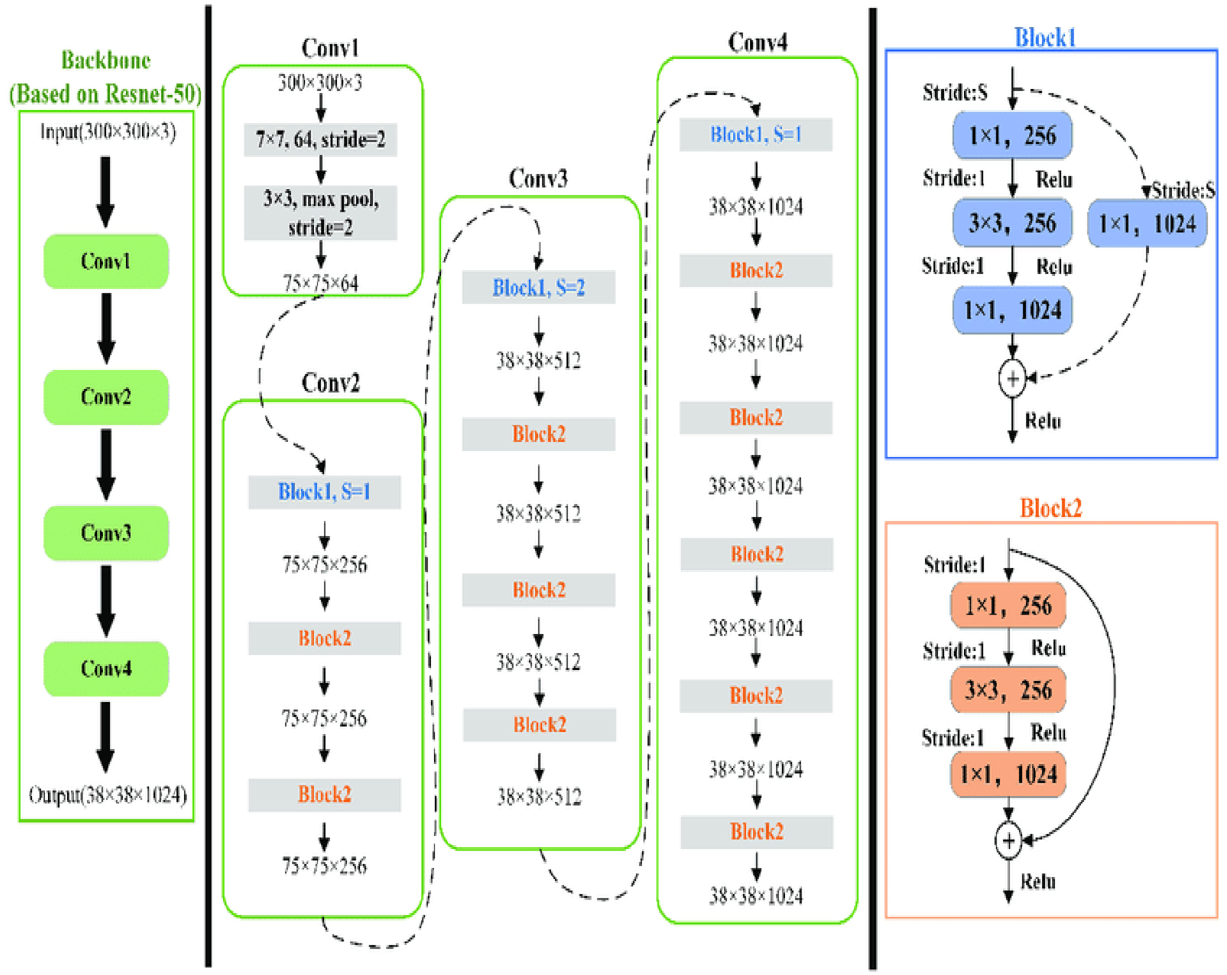


Figure 8: CNN model with ResNet-50 Architecture

Solving Deep Learning Challenges:

•**Residual Learning:** Introduces skip connections to alleviate the vanishing/exploding gradient problem, ensuring effective training of deep networks.
•**Enhanced Training Stability:** ResNet50's architecture allows for deeper networks without increasing training complexity or degrading performance.

Advantages of ResNet50 Architecture:

•**Efficient Training:** Ensures that added layers contribute to performance gains or at the very least, do not harm performance.
•**Layer Nesting:** Maintains optimal function approximation, akin to expanding the search without losing sight of the target.

Model-3 (Convolution Neural Network with ResNet-50) Specification

Upsampling Layers: Three sequential upsampling layers to scale input images to 256x256 pixels.

ResNet50 Backbone: Integrated as a functional layer with 23,587,712 parameters, adapting its robust feature extraction for high-resolution inputs.

Flattening and Normalization: Flattened output from ResNet50 into a 131,072-dimensional vector.

- Applied batch normalization thrice, with the first having 524,288 parameters to standardize the activations.

Dense Layers and Regularization: Dense layer with 128 neurons (16,777,344 parameters), followed by dropout for regularization.

- A second dense layer with 64 neurons (8,256 parameters), again followed by dropout.
- Final output dense layer with 10 neurons (650 parameters) for class prediction.

Compilation:

- Uses Adam optimizer and 'sparse_categorical_crossentropy' loss for classification.
- Performance measured by accuracy.

Training:

- Preprocessed data (X_train_normalized, Y_train) trained for 25 epochs.
- Uses 10% of data for validation.

Hyperparameter Tuning for ResNet50 CNN Model

Methodology:

•Utilized Keras Tuner's RandomSearch to explore hyperparameter space.
•Defined search space for key hyperparameters:

- **Learning Rate:** Logarithmically sampled between 1e-5 and 1e-3.
- **Dense Layer Units:** Varied from 128 to 1024 for the first and 64 to 512 for the second dense layer.
- **Dropout Rate:** Tested between 0.3 to 0.7 to regularize the network and prevent overfitting.

Results:

- Conducted 5 trials with a 1 execution per trial configuration to maintain computational efficiency.
- Each trial involved training for 5 epochs and a validation split of 10% to monitor performance.
- Best hyperparameters obtained were automatically selected based on the highest validation accuracy.

```
Best val_acc So Far: 0.9365000128746033
```

Model-3 Validation

Model Performance:

- `Training Accuracy = 99.08%`
- `Training Loss = 0.0292`
- `Test Accuracy = 93.08%`
- `Test Loss = 0.3245`

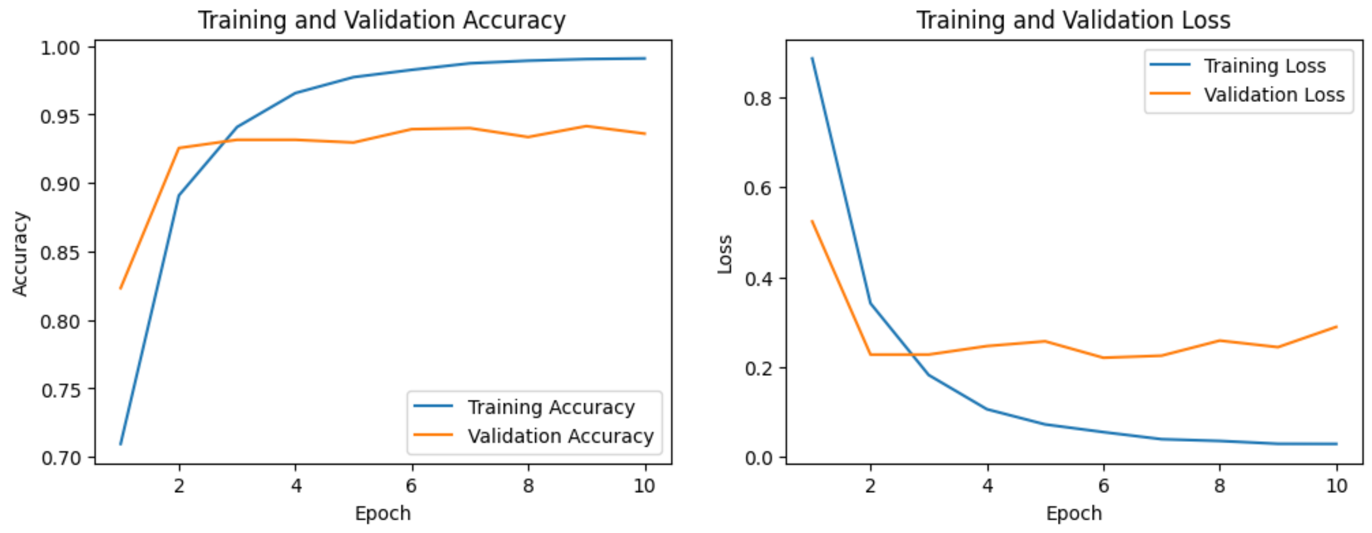
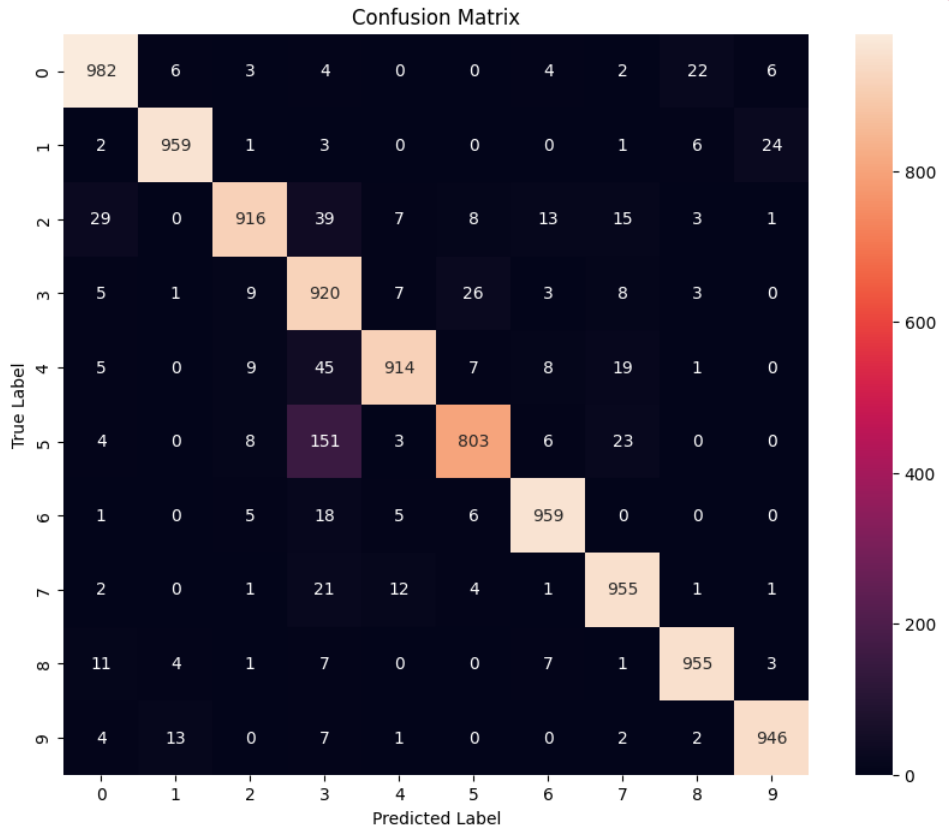


Figure 9: Training and validation curves of model-3 (CNN with ResNet-50) over epochs



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	982	6	3	4	0	0	4	2	22	6
1	2	959	1	3	0	0	0	1	6	24
2	29	0	916	39	7	8	13	15	3	1
3	5	1	9	920	7	26	3	8	3	0
4	5	0	9	45	914	7	8	19	1	0
5	4	0	8	151	3	803	6	23	0	0
6	1	0	5	18	5	6	959	0	0	0
7	2	0	1	21	12	4	1	955	1	1
8	11	4	1	7	0	0	7	1	955	3
9	4	13	0	7	1	0	0	2	2	946

Figure 10: Confusion matrix for model-3 showing classification performance

- The model's learning performance improves as training progresses, with accuracy leveling off, indicating potential for further optimization.
- Losses converge with minor fluctuations in validation, suggesting good generalization with room to reduce overfitting.

- The model accurately predicts most classes, as shown by the dominant diagonal in the confusion matrix.
- Some off-diagonal elements, especially between classes 3 and 5, suggest confusion between certain classes,

Model Comparison

	Neural Network	CNN	CNN with ResNet-50
Test Accuracy	42.17%	74.11%	93.08%
Test Loss	1.64	0.73	0.32

Table 1: Performance metrics of Neural Network, CNN, and CNN with ResNet-50

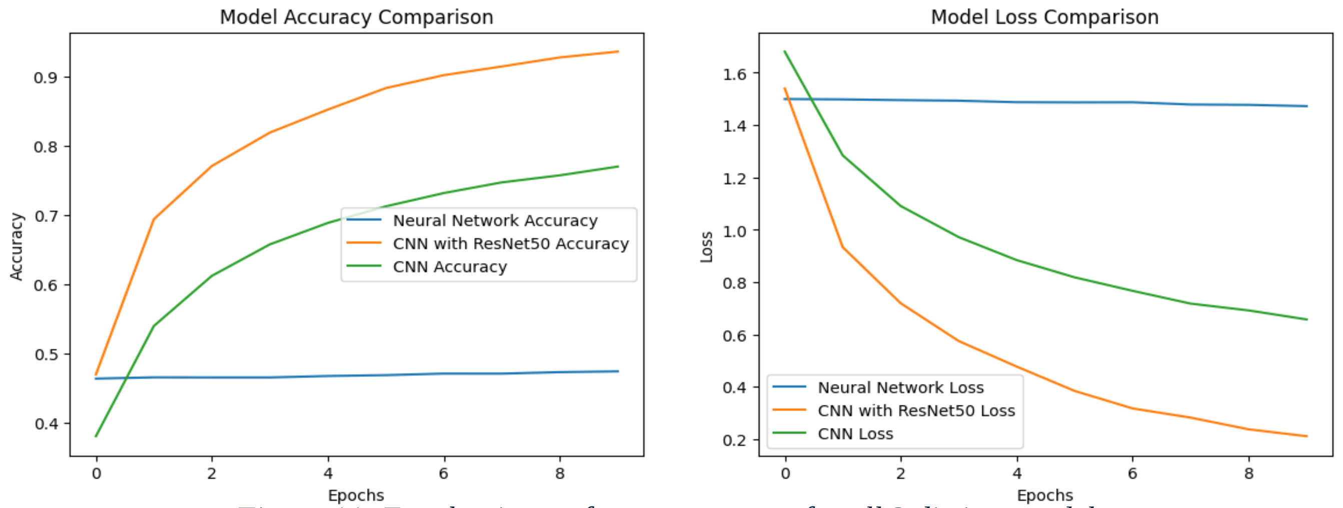


Figure 11: Epoch-wise performance curves for all 3 distinct models

- The CNN with ResNet50 outperforms the basic CNN and neural network in terms of accuracy, achieving the highest values consistently across epochs.
- The neural network shows the least improvement in accuracy over epochs, suggesting limited feature extraction capabilities compared to CNNs.
- Loss decreases most sharply for the CNN with ResNet50, indicating more effective learning and error minimization. The basic neural network model displays a higher loss, suggesting less capacity for learning from the dataset.

Challenges and Considerations

- **GPU Constraints**
    1. Access to Google Colab's limited free-tier GPU resources posed challenges for extensive model training.
    2. Time-bound sessions necessitated careful planning to ensure uninterrupted training for the CNN with ResNet50 architecture.

- **Training and Epoch Limitations**
    1. Restricted by GPU time, the model training was capped at fewer epochs than ideal, potentially impacting the depth of learning.
    2. A delicate balance was sought between extending epochs for higher accuracy and the practical time limitations of the platform.

- **Time Management and Resource Allocation**
    1. Sessions were meticulously scheduled to utilize GPU access fully, requiring precise time management.
    2. The project was adapted to work within the confines of session timeouts and resource availability.

- **Optimization Strategies**
    1. Explored less resource-intensive optimization techniques to compensate for the lack of extended GPU access.
    2. Evaluated the cost-benefit ratio of potentially investing in more resources versus the expected improvement in model performance.

Conclusion

- **Accomplishments:**
  1. Successfully trained three variants of deep learning models, showcasing a clear progression in accuracy from a basic neural network (42% accuracy), to a basic CNN (75%), and finally to a CNN with ResNet50 (93.08%).

- **Key Insights:**
  1. Reinforced the importance of deep architectures like ResNet50 in handling complex image classification tasks effectively.
  2. Demonstrated the balance between model complexity and computational constraints, underscoring the need for efficient resource management.

- **Future Directions:**
  1. Explore extended training sessions with increased epochs to further improve the model's performance.
  2. Consider investing in dedicated GPU resources or seeking partnerships for academic access to overcome the limitations of Google Colab's free tier.

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 (CVPR)*, 770-778.

2. Brownlee, J. (2019). Deep Learning for Computer Vision. Machine Learning Mastery. https://machinelearningmastery.com/deep-learning-for-computer-vision/

3. CIFAR-10 Dataset from Kaggle. https://www.kaggle.com/c/cifar-10

4. The Annotated ResNet-50. Retrieved from Towards Data Science.

5. Google Colaboratory. https://colab.research.google.com/

6. Goodfellow I., Bengio Y., & Courville A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org/ - For foundational concepts in neural networks and deep learning.

7. Krizhevsky A., Sutskever I., & Hinton G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. NIPS.

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

stevens.edu