Image Classification: CNN with Transfer Learning on CIFAR-10

Arjun B. Vaghasiya
CSULB, CA
CECS 550
ID:032166509

Daksh N. Goti
CSULB, CA
CECS 550
ID:032172125

Yagnik K. Beladiya
CSULB, CA
CECS 550
ID:030821490

Project Link: https://github.com/AV015/CECS_550_Pattern_Recognition

Abstract

This Project explore how to use Convolutional language Network and apply the transfer learning technique to improve the image classification accuracy on CIFAR-10 datasets. CIFAR-10 has 60,000 32*32 colour images in 10 classes, making it with high accuracy. Transfer Learning involve using pre-trained model to extract the insight from the data for new task, reducing a need for extensive training data. Our approach fine-tunes a CNN on CIFAR-10, optimizes hyperparameters, evaluates performance. Through experimentation, we aim to demonstrate the effectiveness of CNN with transfer learning for image classification.

1. Introduction

In the world of computers, understanding what's in a picture is a big deal. That's what image classification is all about – training computers to learn how to classify images. And the reason why computers are getting good at it comes down to a particular type of software that exploits a mathematical family of functions known as Convolutional Neural Networks (CNNs). The idea behind these networks is, essentially, to just let the computer learn all the features it needs to know directly from the data in the pixels.

But even better than that, there's this other use known as transfer learning. In this trick, you borrow knowledge from a super-genius computer that already knows a lot about images. That way, even your own not-so-smart image-sorting programs can get smarter – as long as you don't have a huge pile of pictures you can train them on.

So, for this project, we're going to use a CIFAR-10 (a difficult dataset made up of many small, colourful images in different categories) datasets to try various types of object-sorting programs and see how they behave on our task; we'll also play with ways to configure them to fit CIFAR-10's particular challenges, and see how much better they get when we use transfer learning.

Our goal is to find the best way to sort images in CIFAR-10 and hopefully learn some new things that can help computers get even better at understanding pictures in the future.

2. Dataset and Related Work

In Section 3.1, we provide an overview of the CIFAR-10 dataset, highlighting its structure, characteristics, and significance as a benchmark dataset for image classification tasks. In Section 3.2 explores existing research efforts and literature related to image classification using the CIFAR-10 dataset, analysing various approaches, methodologies, and insights gleaned from prior studies.

2.1 CIFAR-10 Dataset

The CIFAR-10 dataset [1] is a widely-used benchmark for image classification. It consists of 60,000 32x32 colour images across 10 mutually exclusive classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is split into 50,000

training images and 10,000 test images, with 6,000 images per class. The small image size and diverse classes make CIFAR-10 challenging for classification tasks.

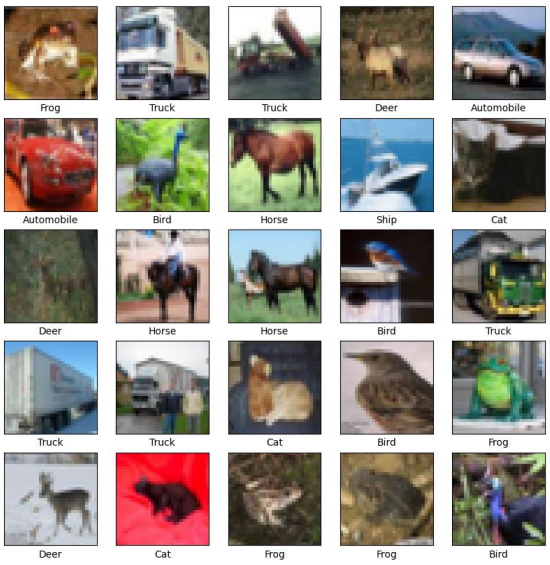

Figure 1: CIFAR-10 Datasets

2.2 Related Work

Numerous studies have explored image classification on CIFAR-10, proposing various CNN architectures and techniques to improve performance. ResNet [2] introduced residual connections, enabling the training of deep networks and achieving 93.57% accuracy with a 110-layer model. DenseNet [3] proposed dense connections, reaching 96.54% accuracy with a 190-layer model and a growth rate of 40. Other notable architectures include Wide ResNet [4] and PyramidNet [5].

Recent work has investigated transfer learning, leveraging pre-trained models on large-scale datasets like ImageNet for faster convergence and improved accuracy. Meta-learning techniques, such as few-shot learning [6], have been explored to learn from limited training data. Data augmentation methods, including AutoAugment [7], have been employed to increase data diversity and improve generalization.

Regularization techniques like dropout [8] and weight decay have been used to prevent overfitting, while ensemble methods, such as model averaging and snapshot ensembling [9], have been utilized to enhance accuracy and robustness.

This project builds upon these advancements, comparing custom CNN architectures and pre-trained models to gain insights into their effectiveness on CIFAR-10. The impact of data augmentation techniques on model performance and generalization is also investigated.

3. Methodology

3.1 Custom CNN Architectures:

We designed three custom CNN architectures for CIFAR-10 classification:

CNN1:

This architecture consists of five convolutional block followed by three dense layers. The conv layers have increasing filter depths (32, 64, 128, 256, 512) and use 3x3 and 5x5 kernels with ReLU activation. Max pooling and dropout are applied after each conv block. The dense layers have 1024, 512, and 10 units respectively.

CNN2:

CNN2 is similar to CNN1 but with different filter depths (64, 128, 256, 512) and an additional dense layer of 256 and 512 units. This model aims to capture more complex features with increased depth.

CNN3:

This is a deeper architecture with 12 conv layers organized into five blocks. Each block contains two or three conv layers with filter depths of 64, 128, 256, and 512. 1x1 conv layers are used in the last two blocks to reduce dimensionality. The model ends with three dense layers of 2048, 1024, and 10 units.

All CNNs use batch normalization for faster convergence and regularization.

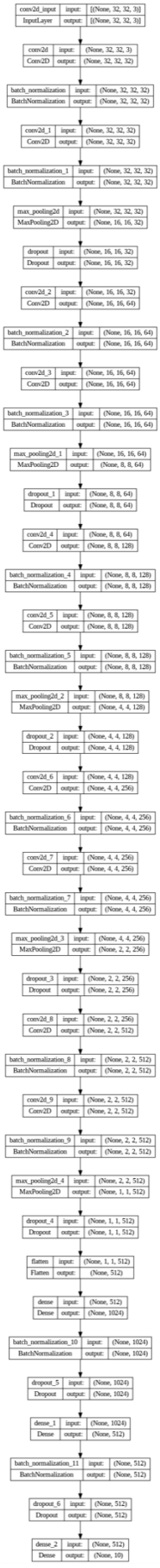

Figure 2: CNN 1

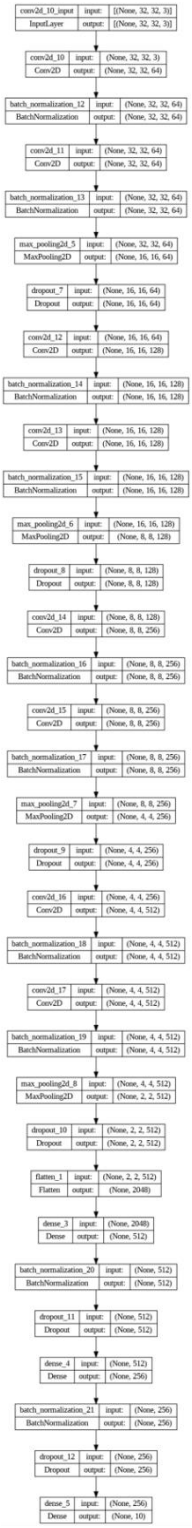

Figure 3: CNN 2

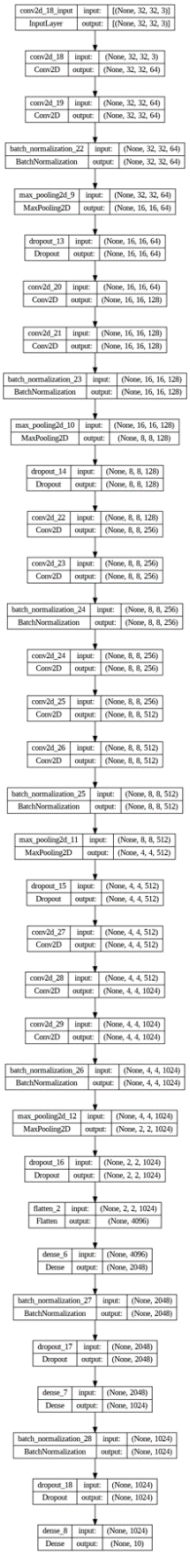

Figure 4: CNN 3

3.2 Transfer Learning

We explored transfer learning using three pre-trained CNN architectures:

VGG16 [6]:

VGG16 is a deep CNN with 16 layers trained on ImageNet. We loaded the pre-trained weights, froze the base layers, , and added dense layers of 256 and 10 units for CIFAR-10 classification.

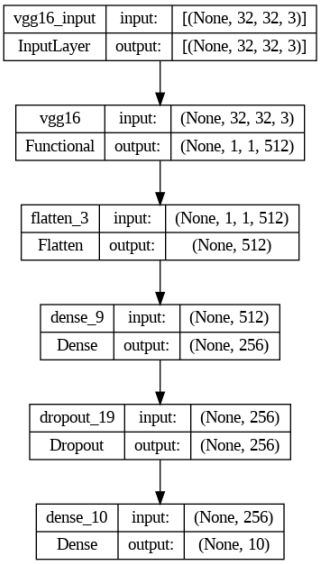

Figure 5: VGG16

MobileNet [7]:

MobileNet is an efficient CNN architecture optimized for mobile devices. Similar to VGG16, we loaded pre-trained MobileNet weights, froze the base model, and added custom dense layers.

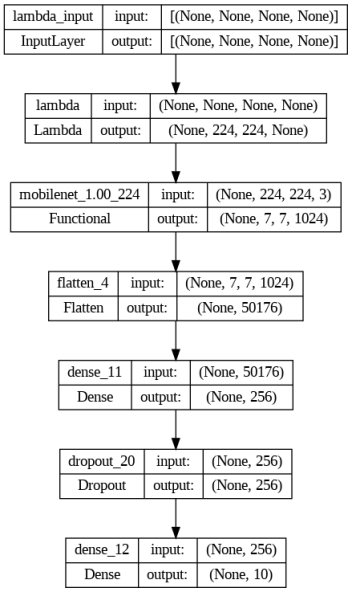

Figure 6: MobileNet

InceptionV3 [8]:

InceptionV3 is a sophisticated CNN with inception modules that perform parallel convolutions at different scales. We loaded the pre-trained model, resized CIFAR-10 images to match InceptionV3's input size, and fine-tuned the last layers.

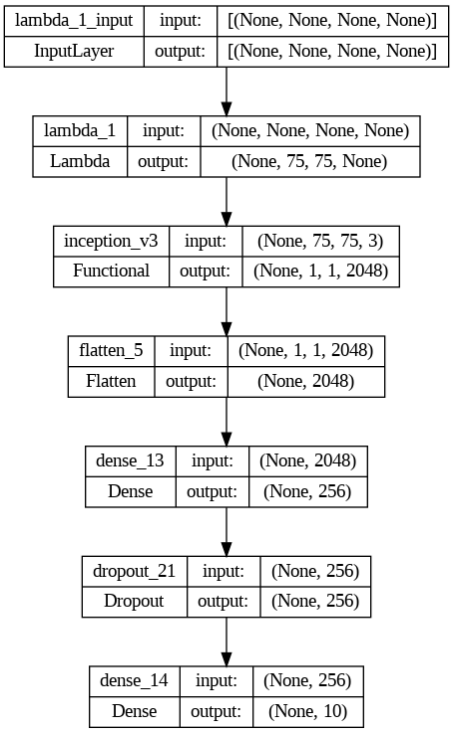

Figure 7: InceptionV3

Transfer learning allows leveraging knowledge learned from large-scale datasets like ImageNet to boost performance on smaller target datasets.

3.3 Data Augmentation

To improve model generalization, we applied data augmentation techniques during training:

- Random rotation between -15 and 15 degrees
- Width and height shift of 10%
- Random horizontal flip

Data augmentation helps the models learn invariances and reduces overfitting.

4. Experimental Setup

4.1 Implementation

The project was implemented using Python 3.10, TensorFlow 2.15, and Keras. Models were trained on an NVIDIA L4 GPU. The code was structured modularly with separate functions for data loading, preprocessing, model building, training, and evaluation.

4.2 Hyperparameters

We used the following hyperparameter settings:

- **Optimizer:** Adam with learning rate 0.001
- **Batch size:** 128
- **Epochs:** 30
- **Early stopping:** Patience of 10 epochs monitoring validation loss

Models were compiled with categorical cross-entropy loss and accuracy metric.

5. Results Measurement

5.1 Training and Validation Metrics

For each model, we logged the training and validation accuracy & loss across epochs. This helps assess model convergence and overfitting. The training curves were visualized using matplotlib.

Custom CNN Architectures Accuracy:

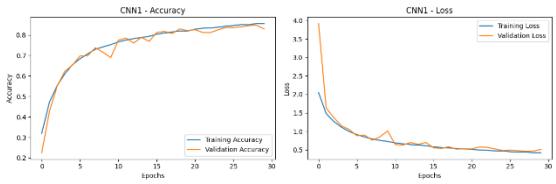

Figure 8: CNN1 Accuracy & Loss

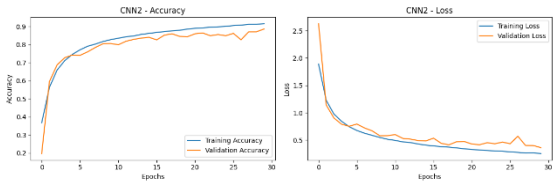

Figure 9: CNN2 Accuracy & Loss

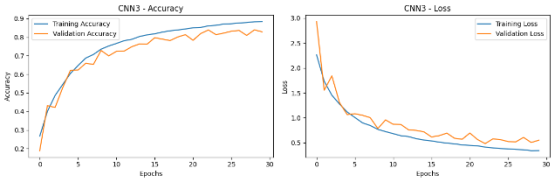

Figure 10: CNN3 Accuracy & Loss

Transfer Learning Accuracy:

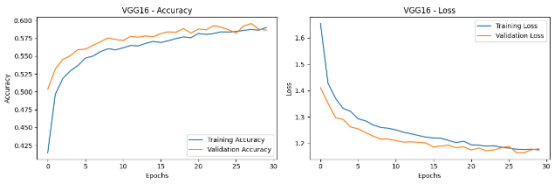

Figure 11: VGG16 Accuracy & Loss

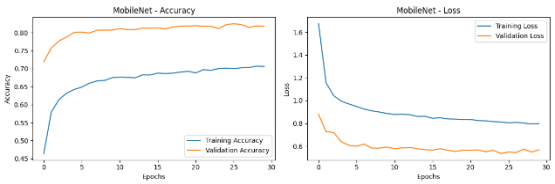

Figure 12: MobileNet Accuracy & Loss

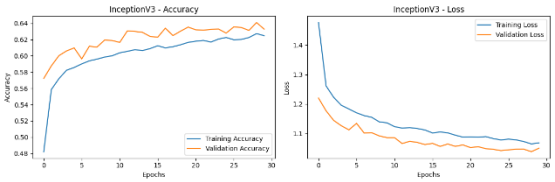

Figure 13: InceptionV3 Accuracy & Loss

5.2 Test Set Evaluation

We quantified the performance of the trained models on the CIFAR-10 test set using the following metrics:

- **Accuracy:** fraction of images that were correctly classified by the model.
- **Confusion Matrix:** a table that tracks what classes the model does and does not do a good job at predicting. This can help identify what classes are commonly misclassified.
- **Precision, Recall, F1 score:** these give a more nuanced view of how well the model is doing in relation to true positives (correct), false positives (errors) and false negatives (passing up a chance to predict correctly).

CNN Architectures Confusion Metrix:

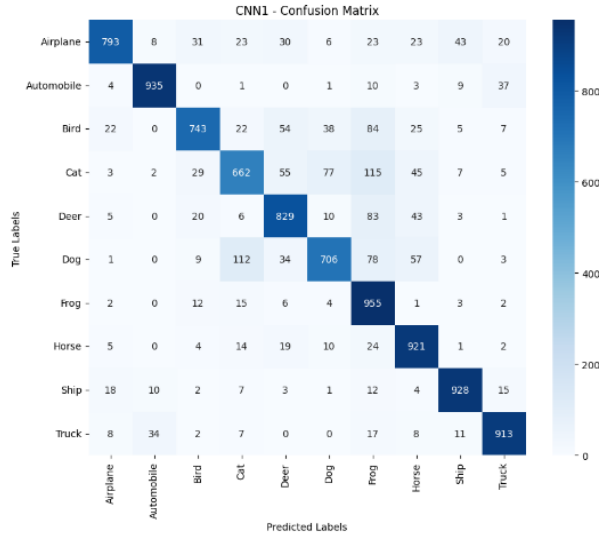

Figure 14: CNN1 Confusion Metrix

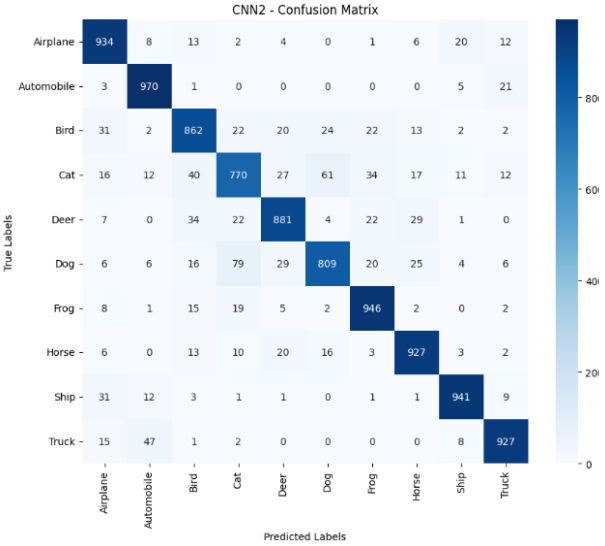

Figure 15: CNN2 Confusion Metrix

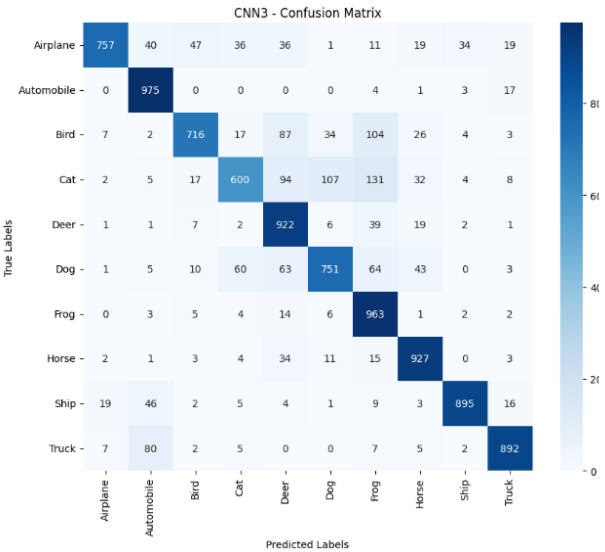

Figure 16: CNN3 Confusion Metrix

Transfer Learning Confusion Metrix:

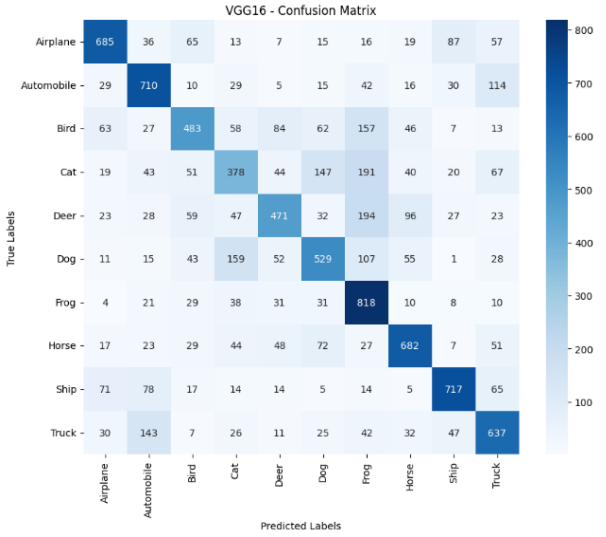

Figure 17: VGG16 Confusion Metrix

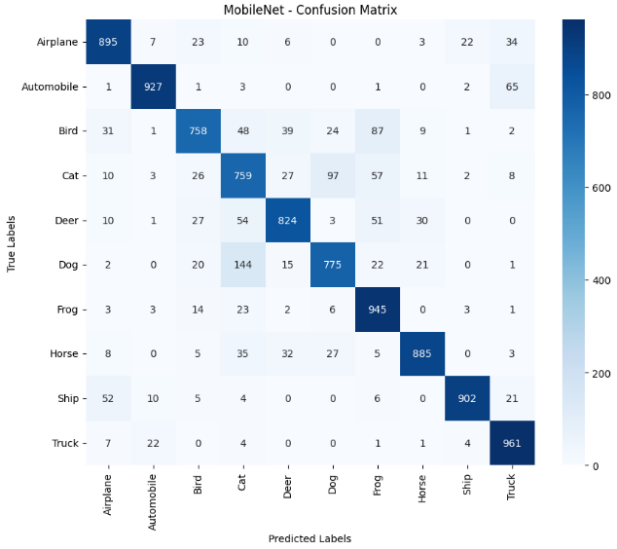

Figure 18: MobileNet Confusion Metrix

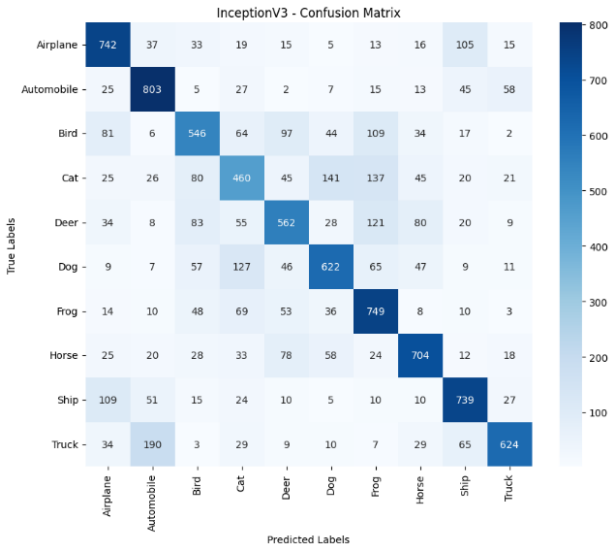

Figure 19: InceptionV3 Confusion Metrix

5.3 Inference Speed

We measured the average inference time per image for each model. This is an important consideration for real-time applications.

6. Results Analysis and Comparison

6.1 Custom CNN Architectures

Among the custom CNNs, CNN2 achieved the best test accuracy of 89.67%, followed by CNN3 with 83.98% and CNN1 with 83.85%. CNN2's higher accuracy can be attributed to its increased depth and additional dense layer compared to CNN1. However, CNN3's deeper architecture did not yield further improvements, possibly due to overfitting.

The confusion matrices reveal that the custom CNNs struggled most with the "cat" and "dog" classes. This is unsurprising given the visual similarities between these animals. Precision and recall scores were generally high across classes, indicating balanced performance.

6.2 Transfer Learning Models

MobileNet outperformed the other transfer learning approaches with 86.31% test accuracy. VGG16 and InceptionV3 achieved lower accuracies of 61.10% and 65.51% respectively. The lower performance of these models might be due to the mismatch between their input sizes and the smaller CIFAR-10 images.

MobileNet's strong results demonstrate the effectiveness of its efficient architecture and the benefits of transfer learning. Fine-tuning the pre-trained models could potentially improve their performance further.

6.3 Comparison and Intuition

The best custom CNN (CNN2) surpassed the best transfer learning model (MobileNet) by a margin of 3.36% accuracy. This suggests that a well-designed CNN architecture tailored to the specific dataset can outperform pre-trained models.

However, transfer learning offers several advantages. Pre-trained models encode rich features learned from diverse datasets, which can be effectively adapted to new tasks. They also converge faster and require less training data compared to training from scratch.

The choice between custom CNNs and transfer learning depends on factors like dataset size, similarity to pre-training data, computational resources, and required accuracy. For CIFAR-10, a custom CNN provided the best results, but transfer learning remains a powerful technique worth exploring.

We're going to look at a bunch of different models in this report. We'll check out how accurate they are and how much loss they have. This way, we can figure out which one is the best.

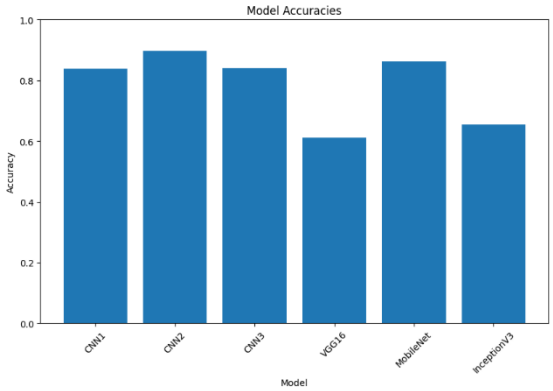

Figure 20: Comparison Accuracy of models

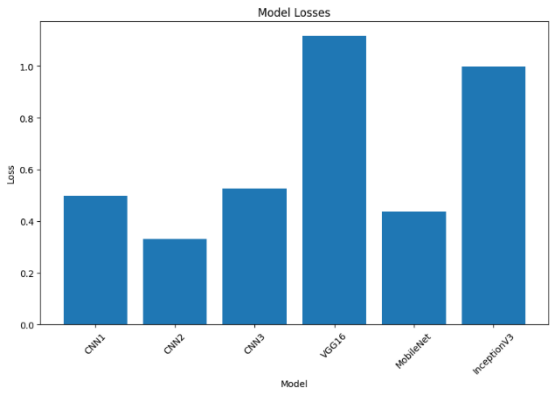

Figure 21: Comparison loss of models

7. Conclusion

The custom architectures tested had three CNNs – which we called CNN1, CNN2 and CNN3, and they were training to solve Image Classification on CIFAR-10. An important finding was that a good 4-layer CNN (CNN2) can beat all other structures on CIFAR-10. Another interesting finding was that transfer learning with MobileNet can perform well, as seen in the result for MobileNet. Leveraging a pre-trained model can help achieve high accuracy on CIFAR-10. Further work can be done by testing more advanced architectures, such as ResNet , DenseNet or EfficientNet. It will also be interesting to fine-tune the pre-trained models in order to enhance performance.

Other approaches, such as SVMs (which we did not test in this project) can potentially yield high accuracy, although our high accuracy results using CNNs on CIFAR-10 indicate that the CNNs excel at this task.

This project has given us insight into the various types of model structures that can be used to develop an image classification system, while smart learning techniques can yield better results.

8. Student Contribution

This section details the individual contributions of each student to the project.

Arjun Vaghasiya (#032166509):

- Implemented the CNN2 model architecture from scratch
- Utilized MobileNet transfer learning to build upon pre-trained weights
- Preprocessed and analyzed the CIFAR-10 dataset for model training and evaluation
- Designed and applied data augmentation techniques to enhance model robustness
- Optimized the models and tuned hyperparameters for improved performance

Daksh Goti (#032172125) :

- Implemented the CNN1 model architecture from scratch
- Applied VGG16 transfer learning to leverage pre-trained weights
- Set up the experimental environment using Python, TensorFlow, and Keras
- Structured the project codebase into modular, reusable components
- Implemented evaluation metrics to assess model performance

Yagnik Beladiya (#030821490):

- Implemented the CNN3 model architecture from scratch
- Employed InceptionV3 transfer learning to harness pre-trained weights
- Conducted extensive experiments to compare the performance of different models
- Visualized and interpreted the experimental results using graphs and metrics
- Contributed significantly to the analysis and comparison of results across models

9. References

[1] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In CVPR (pp. 770-778).

[3] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In CVPR (pp. 4700-4708).

[4] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.

[5] Han, D., Kim, J., & Kim, J. (2017). Deep pyramidal residual networks. In CVPR (pp. 5927-5935).

[6] Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. In NIPS (pp. 4077-4087).

[7] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2019). AutoAugment: Learning augmentation strategies from data. In CVPR (pp. 113-123).

[8] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. JMLR, 15(1), 1929-1958.

[9] Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., & Weinberger, K. Q. (2017). Snapshot ensembles: Train 1, get M for free. arXiv preprint arXiv:1704.00109.

[10] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[11] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[12] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In CVPR (pp. 2818-2826).