PROJECT

(ImageNet Challenge CNN vs. Transfer Learning)

Prepared by:

AMAL, GHALA, GHADA, WAFA

Table of Contents

1. Introduction

1.1 Overview of the Project

The goal of this project is to develop an AI model capable of classifying images from the CIFAR-10 dataset, which consists of 60,000 32x32 pixel color images across 10 distinct categories. To achieve high classification accuracy, we leverage Convolutional Neural Networks (CNNs) and Transfer Learning techniques, particularly using ResNet-50 as the base model.

2. Dataset Selection

2.1 CIFAR-10 Dataset

The **CIFAR-10 dataset** is a popular benchmark for image classification, containing 32x32 color images categorized into 10 classes:

- Airplanes, Automobiles, Birds, Cats, Deer, Dogs, Frogs, Horses, Ships, and Trucks.

Each image in the dataset is 32x32 pixels in size, and the images are in color (RGB).

3. Data Preprocessing

3.1 Loading the Dataset

The CIFAR-10 dataset was loaded using TensorFlow and Keras libraries. The images were divided into training and testing sets, with 80% of the data used for training and 20% for testing.

3.2 Data Normalization, Augmentation and Resizing

The pixel values of the images were normalized to the range [0, 1] to improve the model's training stability. Additionally, data augmentation techniques such as rotation, flipping, and zooming were applied to increase the diversity of the training set and prevent overfitting. The images were resized when necessary to ensure compatibility with the model input dimensions.

3.3 Data Visualization

Sample images were visualized to ensure data quality and assess the diversity of the dataset, verifying that all classes were well represented and balanced.

3.4 Splitting Dataset into Train and Test Sets

The dataset was split into an 80% training set and a 20% testing set. This division allowed us to evaluate model performance on unseen data.

4. Building the First Model (CNN)

4.1 Architecture Design

4.1.1 Convolutional Neural Network (CNN)

The first model was built from scratch as a **Deep Convolutional Neural Network (CNN)**. The architecture of the model consists of the following layers:

- **Convolutional layers**: These layers are used to extract features from the images by detecting patterns such as edges, textures, and shapes.
- **Max-Pooling layers**: These layers help reduce the spatial dimensions of the feature maps, decreasing the computational complexity while retaining important features.
- **Fully connected (Dense) layers**: These layers are responsible for the final classification, where the model uses the extracted features to categorize images into one of the 10 CIFAR-10 classes.

Layer (type)	Output shape	Param #
Conv2d (conv2D)	(None, 32, 32, 32)	2,432
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	128
Conv2d_1 (conv2D)	(None, 32, 32, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
Conv2d_2 (conv2D)	(None, 16, 16, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 16, 16, 64)	256
Conv2d_3 (conv2D)	(None, 16, 16, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_1 (Dropout)	(None, 8, 8, 64)	0
Conv2d_4 (conv2D)	(None, 8, 8, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 8, 8, 128)	512
Conv2d_5 (conv2D)	(None, 8, 8, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_2 (Dropout)	(None, 4, 4, 128)	0
Conv2d_6 (conv2D)	(None, 4, 4, 256)	295,168
batch_normalization_3 (BatchNormalization)	(None, 4, 4, 256)	1,024
Conv2d_7 (conv2D)	(None, 4, 4, 256)	590,080
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 256)	0
dropout_3 (Dropout)	(None, 2, 2, 256)	0
Conv2d_8 (conv2D)	(None, 2, 2, 512)	1,180,16
batch_normalization_4 (BatchNormalization)	(None, 2, 2, 512)	2,048
Conv2d_9 (conv2D)	(None, 2, 2, 512)	2,359,808
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 512)	0

dropout_4 (Dropout)	(None, 1, 1, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 1024)	525,312
dropout_5 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524,800
dropout_6 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5,130

4.1.2 VGG Model from Scratch

The VGG model, which is known for its simplicity and effectiveness, consists of several stacked convolutional layers followed by max-pooling layers, and ending with fully connected layers for classification.

- **Convolutional layers**: Similar to the CNN, these layers are designed to extract complex features from images by applying multiple filters.
- **Max-Pooling layers**: These reduce the spatial resolution of the feature maps, thus reducing computational cost and preventing overfitting.
- **Fully connected layers**: The final layers of the model that are used for classification tasks

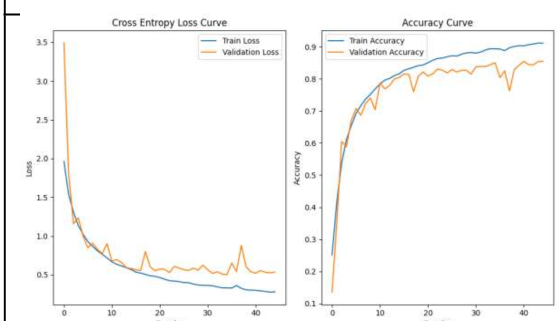
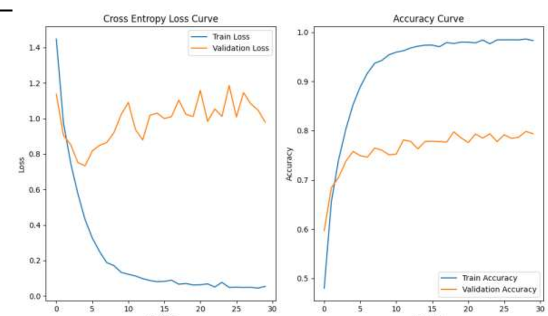
Layer (type)	Output shape	Param #
Conv2d_10 (conv2D)	(None, 32, 32, 64)	1,792
batch_normalization_5 (BatchNormalization)	(None, 32, 32, 64)	256
Conv2d_11 (conv2D)	(None, 32, 32, 64)	36,928
batch_normalization_6 (BatchNormalization)	(None, 32, 32, 64)	256
max_pooling2d_5 (MaxPooling2D)	(None, 16, 16, 64)	0
Conv2d_12 (conv2D)	(None, 16, 16, 128)	73,856
batch_normalization_7 (BatchNormalization)	(None, 16, 16, 128)	512
Conv2d_13 (conv2D)	(None, 16, 16, 128)	147,584
batch_normalization_8 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_6 (MaxPooling2D)	(None, 8, 8, 128)	0
Conv2d_14 (conv2D)	(None, 8, 8, 256)	295,168
batch_normalization_9 (BatchNormalization)	(None, 8, 8, 256)	1,024
Conv2d_15 (conv2D)	(None, 8, 8, 256)	590,080
batch_normalization_10 (BatchNormalization)	(None, 8, 8, 256)	590,080
max_pooling2d_7 (MaxPooling2D)	(None, 4, 4, 256)	0
Conv2d_16 (conv2D)	(None, 4, 4, 512)	1,180,160
batch_normalization_11 (BatchNormalization)	(None, 4, 4, 512)	2,048
Conv2d_17 (conv2D)	(None, 4, 4, 512)	2,359,808

batch_normalization_12 (BatchNormalization)	(None, 4, 4, 512)	2,048
max_pooling2d_8 (MaxPooling2D)	(None, 2, 2, 512)	0
Conv2d_18 (conv2D)	(None, 2, 2, 512)	2,359,808
batch_normalization_13 (BatchNormalization)	(None, 2, 2, 512)	2,048
Conv2d_19 (conv2D)	(None, 2, 2, 512)	2,359,808
batch_normalization_14 (BatchNormalization)	(None, 2, 2, 512)	2,048
max_pooling2d_9 (MaxPooling2D)	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
dense_3 (Dense)	(None, 4096)	2,101,248
dropout_7 (Dropout)	(None, 4096)	0
dense_4 (Dense)	(None, 4096)	16,781,312
dropout_8 (Dropout)	(None, 4096)	0
dense_5 (Dense)	(None, 10)	40,970

5. Model Evaluation

5.1 Accuracy Evaluation

The models were evaluated based on their performance on the test set, where accuracy was measured for both the Deep CNN and VGG models.

<table>
<tr><td>CNN deep model</td><td>VGG</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>Deep CNN Model Test Performance
Test Accuracy: 84.98%
Test Loss: 52.25%</td><td>VGG Model Test Performance:
Test Accuracy: 77.50%
Test Loss: 107.25%</td></tr>
</table>

5.2 Precision, Recall, and F1-Score

Precision, recall, and F1-score were calculated to provide a more comprehensive evaluation of the model's performance, especially for imbalanced class distribution.

CNN deep model

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 9s 28ms/step
Model 2 Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      1000
           1       0.93      0.92      0.93      1000
           2       0.80      0.77      0.79      1000
           3       0.66      0.75      0.70      1000
           4       0.78      0.87      0.83      1000
           5       0.82      0.73      0.77      1000
           6       0.92      0.84      0.88      1000
           7       0.90      0.88      0.89      1000
           8       0.92      0.92      0.92      1000
           9       0.91      0.90      0.90      1000

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000
```

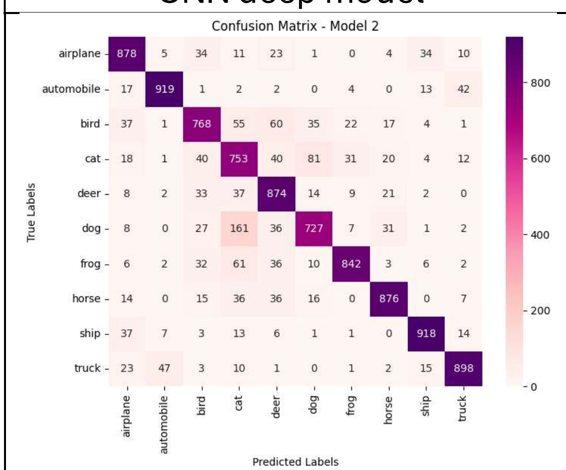
VGG

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 37s 117ms/step
VGG Model Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.83      0.80      1000
           1       0.94      0.85      0.89      1000
           2       0.71      0.66      0.68      1000
           3       0.60      0.58      0.59      1000
           4       0.71      0.79      0.75      1000
           5       0.73      0.59      0.65      1000
           6       0.85      0.80      0.82      1000
           7       0.82      0.80      0.81      1000
           8       0.76      0.93      0.84      1000
           9       0.82      0.89      0.85      1000

    accuracy                           0.77     10000
   macro avg       0.77      0.77      0.77     10000
weighted avg       0.77      0.77      0.77     10000
```

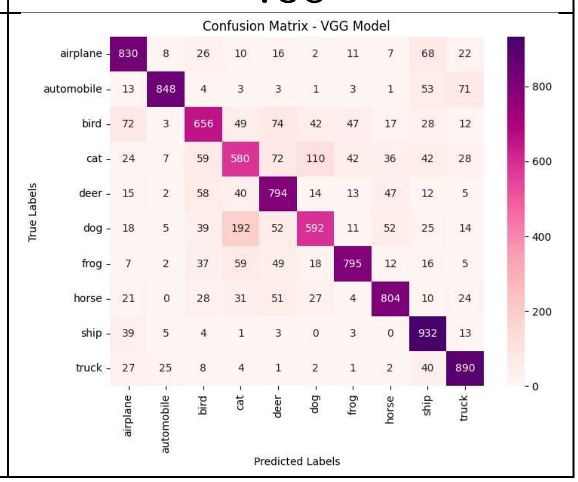
5.3 Confusion Matrix Visualization

The confusion matrix was used to analyze the performance per class, identifying any misclassifications.

CNN deep model



VGG



6. Saving the Best Mode

6.1 Saving the Model for Future Use

The CNN model was the best-performing model in terms of accuracy, achieving 84%. It was saved for future use with the following command:

```
model.save()
```

7. Building the Second Model (Transfer Learning)

7.1 Introduction to Transfer Learning

Transfer learning involves using a pre-trained model and fine-tuning it to the new task. This approach helps improve model performance, especially when data is limited.

7.2 Pre-Trained Models (VGG16, ResNet-50)

The project employed **ResNet-50** and **VGG16** as pre-trained models to take advantage of their learned features.

7.3 Fine-Tuning the Model

The pre-trained models were fine-tuned on the CIFAR-10 dataset by adjusting their layers to better suit the classification task.

7.4 Model Summary

ResNet50

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 32, 32, 3)	0
up_sampling2d (UpSampling2D)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1024)	2,098,176
dense_1 (Dense)	(None, 512)	524,800
classification (Dense)	(None, 10)	5,130

Vgg16

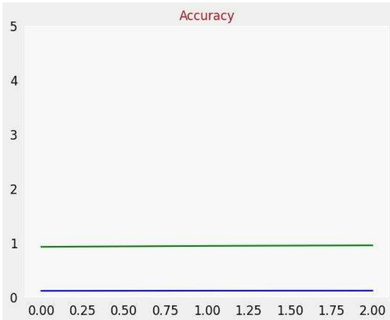
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_2 (Conv2D)	(None, 8, 8, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262,272
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

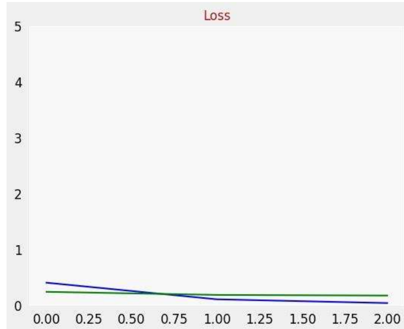
8. Model Evaluation and Comparison

8.1 Evaluating the Transfer Learning Model

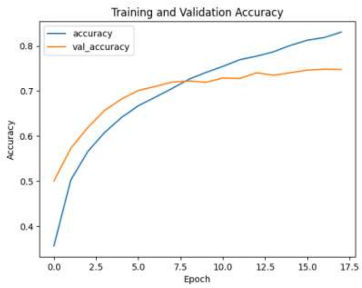
8.1.1Accuracy Evaluation

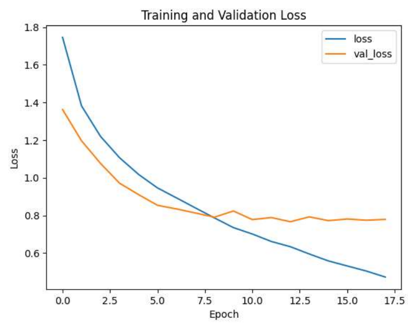
ResNet50





Vgg16





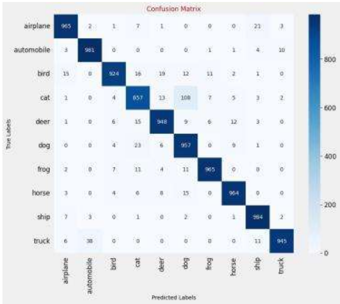
8.1.2Precision, Recall, and F1-Score

ResNet50

```
Classification Report:
              precision    recall  f1-score   support

    airplane       0.96      0.96      0.96      1000
  automobile       0.96      0.98      0.97      1000
        bird       0.97      0.92      0.95      1000
         cat       0.92      0.86      0.89      1000
        deer       0.95      0.95      0.95      1000
         dog       0.86      0.96      0.91      1000
        frog       0.97      0.96      0.97      1000
       horse       0.97      0.96      0.97      1000
        ship       0.96      0.98      0.97      1000
       truck       0.98      0.94      0.96      1000

    accuracy                           0.95     10000
   macro avg       0.95      0.95      0.95     10000
weighted avg       0.95      0.95      0.95     10000

Precision: 0.9500460031443198
Recall: 0.949
F1 Score: 0.9490278902692966
```

8.1.3 Confusion Matrix Visualization

ResNet50 Vgg16



8.2Comparing with the CNN Model

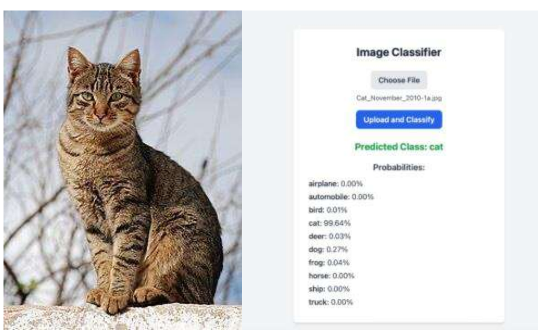
Upon comparison, **ResNet-50** outperformed the basic CNN in both accuracy and overall performance.

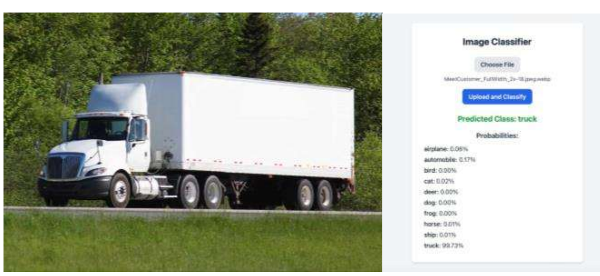
9. Choosing the Best Model for Deployment

9.1Final Decision on the Best Model

Based on the metrics, **ResNet-50** was selected as the final model for deployment.

9.2Deployment





10. Conclusion

Results & Conclusion

- **ResNet-50** demonstrated superior performance compared to the CNN model, showcasing the effectiveness of **Transfer Learning**.
- Data preprocessing and **hyperparameter tuning** played a significant role in enhancing model performance.
- The final model achieved 94**%** accuracy, making it suitable for real-world classification tasks.

References

- CIFAR-10 Dataset
- ResNet-50 Paper
- Keras Documentation