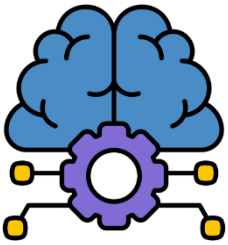


Deep Learning-Image Classification with CNN

Hassa Mohammed
Amjad Aloufi
Arwa AbuFeeh

Index:

1. Introduction

This project explores image classification using CNN architectures on the CIFAR-10 dataset. The experiment is divided into two phases: designing CNN models from scratch and implementing transfer learning using pretrained models (VGG16 and ResNet50). The goal is to evaluate and compare the effectiveness of these models in terms of accuracy and generalization.

2. Dataset Description

The CIFAR-10 dataset contains 60,000 color images of size 32×32 pixels across 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is balanced, with each class containing an equal number of images.

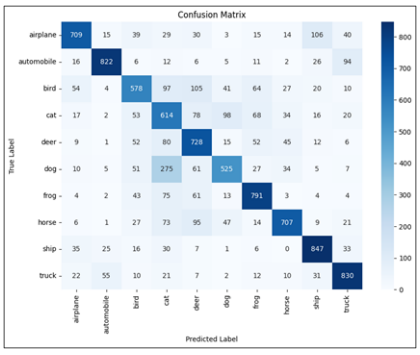
3. Model Architectures

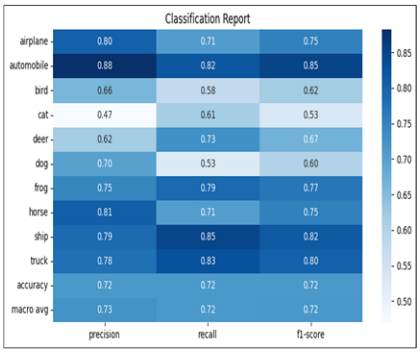
Custom CNN Models**:**

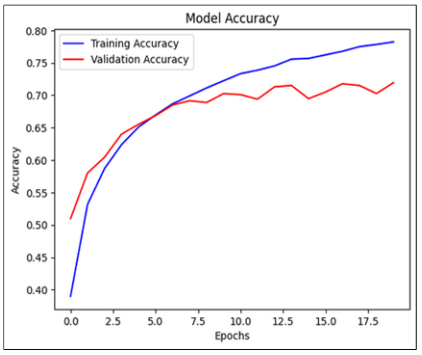
Model 1:

- **Preprocessing**: Normalization (0-1 range) and one-hot encoding.
- **Architecture**:
  - Convolutional layers: 32-64 filters with ReLU activation.
  - MaxPooling layers to reduce spatial dimensions.
  - Dense layers with 64 neurons, dropout (30%), and softmax activation.
- **Training** :
  - Optimizer: Adam
  - Loss Function: Sparse categorical cross-entropy
  - Epochs: 20
  - Accuracy: 73.1%

Below are the confusion matrix, classification report, and training & validation accuracy plot for Model 1.:



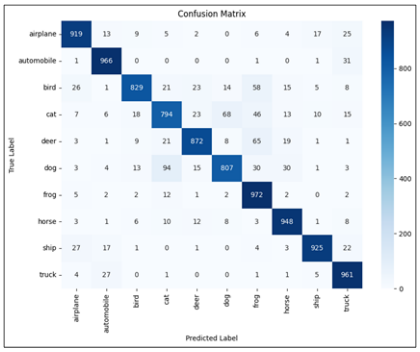


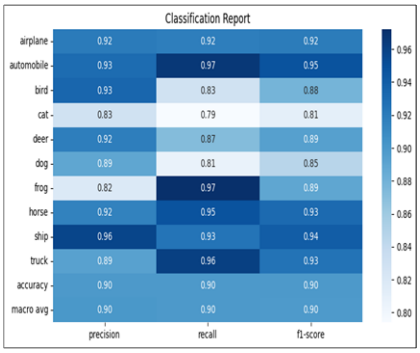


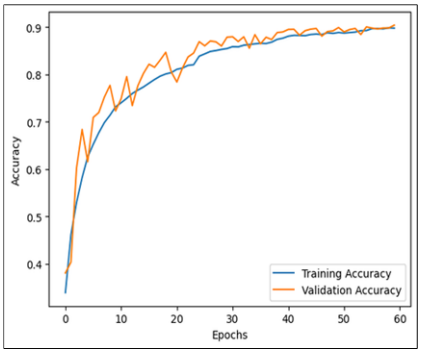
Model 2:

- **Preprocessing**: Similar to Model 1 but with added data augmentation (rotation, shifting, shear, zoom, and horizontal flips) to prevent overfitting.
- **Architecture**:
  - Increased filter size (64-128-256) with batch normalization.
  - MaxPooling layers and dropout (30% convolution layers, 50% fully connected layers).
  - Softmax activation in the output layer.
- **Training**:
  - Optimizer: Adam
  - Callbacks: ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
  - Accuracy: 81.2%

Below are the confusion matrix, classification report, and training & validation accuracy plot for Model 2:



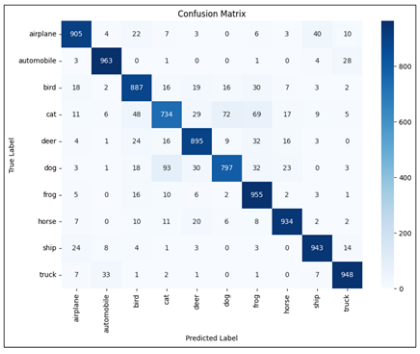


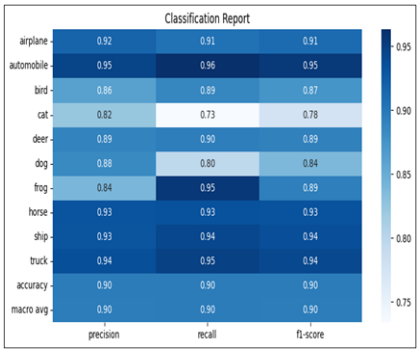


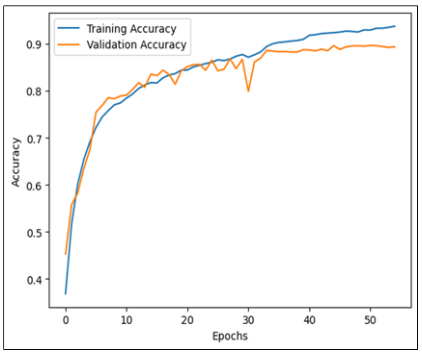
Model 3:

- **Preprocessing**: Normalization, one-hot encoding, and enhanced data augmentation (rotation, horizontal/vertical shifts, nearest neighbor interpolation for missing pixels)
- **Architecture**:
  - Four convolutional blocks (32-256 filters), ReLU activation, batch normalization.
  - MaxPooling and dropout (50%) for regularization.
  - Dense layers with 512 neurons.
- **Training**:
  - Optimizer: Adam, Learning rate: 0.001
  - Epochs: 100, Batch size: 64
  - Accuracy: 90.17%

Below are the confusion matrix, classification report, and training & validation accuracy plot for Model 3:





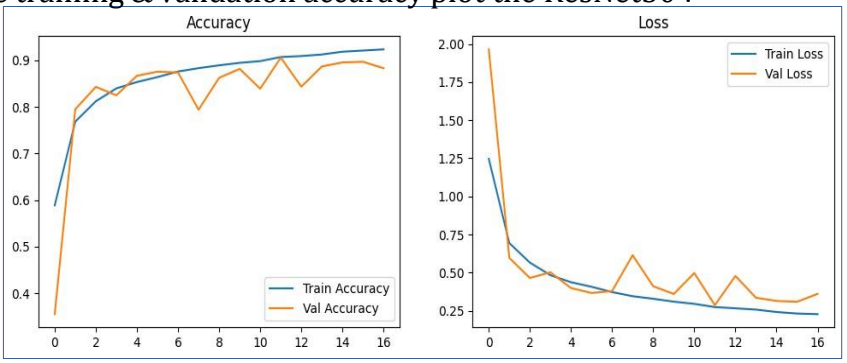


Transfer Learning Models:

ResNet50:

- **Architecture**: 50-layer deep network with residual connections to handle vanishing gradients.
- **Training**:
  - Optimizer: SGD
  - Accuracy: 91% (training), 88% (validation)
  - Observed overfitting

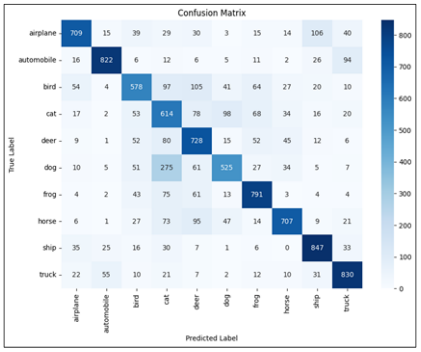
Below are the training & validation accuracy plot the ResNet50 :

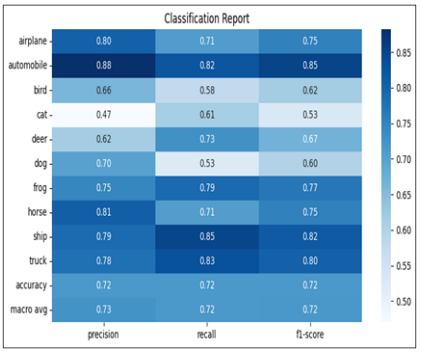


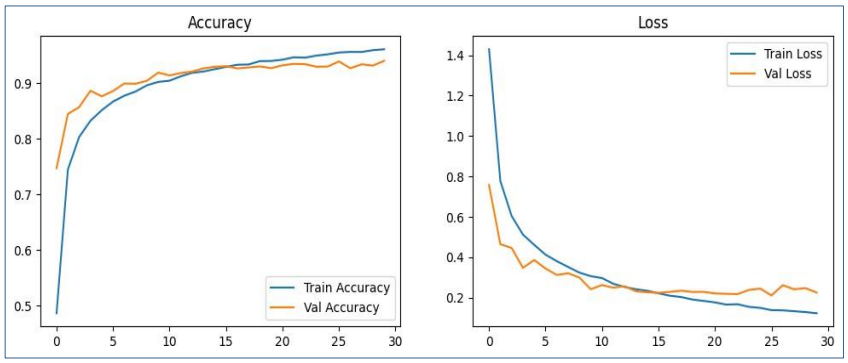
ResNet50, despite its deep architecture with residual connections designed to tackle vanishing gradients, showed lower performance compared to VGG16. It achieved 91% accuracy on the training set and 88% on the validation set. Although it has potential, we observed overfitting, as the training accuracy was higher than validation accuracy, indicating that the model struggled to generalize well to unseen data. Given these issues, VGG16 outperformed ResNet50 in terms of generalization and overall performance.

VGG16:

- **Architecture**: Simpler design with stacked convolutional layers.
- **Training**:
  - Optimizer: SGD
  - Accuracy: 96% (training), 93% (validation)
  - Better generalization compared to ResNet50.
  -
- Below are the confusion matrix, classification report, and training & validation accuracy plot for the VGG16:
  -







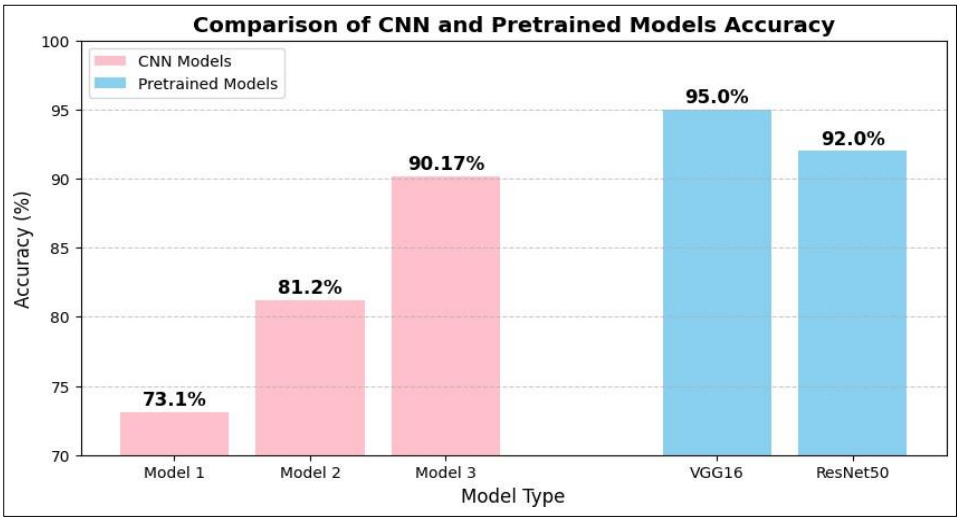
We chose VGG16 as the best model due to its simpler architecture, which uses stacked convolutional layers to effectively capture features while being easy to implement. It achieved 96% accuracy on the training set and 93% on the validation set, demonstrating strong performance and good generalization. Compared to ResNet50, VGG16 showed better generalization, avoiding overfitting. These factors, along with its reliable performance metrics, made VGG16 the best choice for this task.

4. Results and Best Model

Our experiments revealed a progressive improvement in model performance as we refined our architecture and utilized transfer learning techniques. The results of our models are summarized below:

Model	Accuracy
Model 1	73.1%
Model 2	81.2%
Model 3	90.17%
ResNet50	92%
VGG16 (Best Model)	95%

The figure below compares the accuracy of custom CNN models and pretrained models (VGG16, ResNet50) :



Among the models we tested, VGG16 emerged as the best performer with a validation accuracy of 93%. The model demonstrated better generalization capabilities compared to the other architectures, likely due to its well-optimized pre-trained features. In contrast, ResNet50, despite achieving high training accuracy, exhibited some degree of overfitting, with a validation accuracy slightly lower than VGG16.

Our custom CNN models showed a clear trend of improvement with increasing depth and additional regularization techniques. Model 3, which incorporated data augmentation and

dropout, significantly outperformed Model 1 and Model 2, reaching 90.17% accuracy. However, even our best custom model was surpassed by the transfer learning approaches, highlighting the advantages of leveraging pre-trained networks for image classification tasks.

- **Training and Validation Accuracy Plot**: Shows how each model's accuracy evolved over epochs.
- **Confusion Matrices**: Displays the classification performance for each model.
- **Bar Chart Comparison**: Highlights the accuracy differences between custom CNNs and transfer learning models.

5. Key Insights

- Deeper and more optimized architectures result in improved accuracy.
- Data augmentation significantly enhances model generalization.
- Transfer learning (VGG16 & ResNet50) provides superior performance due to pre- trained feature extraction.
- Regularization techniques (dropout, batch normalization, early stopping) help mitigate overfitting.

6. Final thoughts

While custom CNN models improved with depth and tuning, VGG16 proved to be the best-performing model**.** The benefits of transfer learning were evident in achieving higher accuracy with less training effort. For practical applications**,** fine-tuning pretrained models is a more efficient approach compared to building CNNs from scratch.

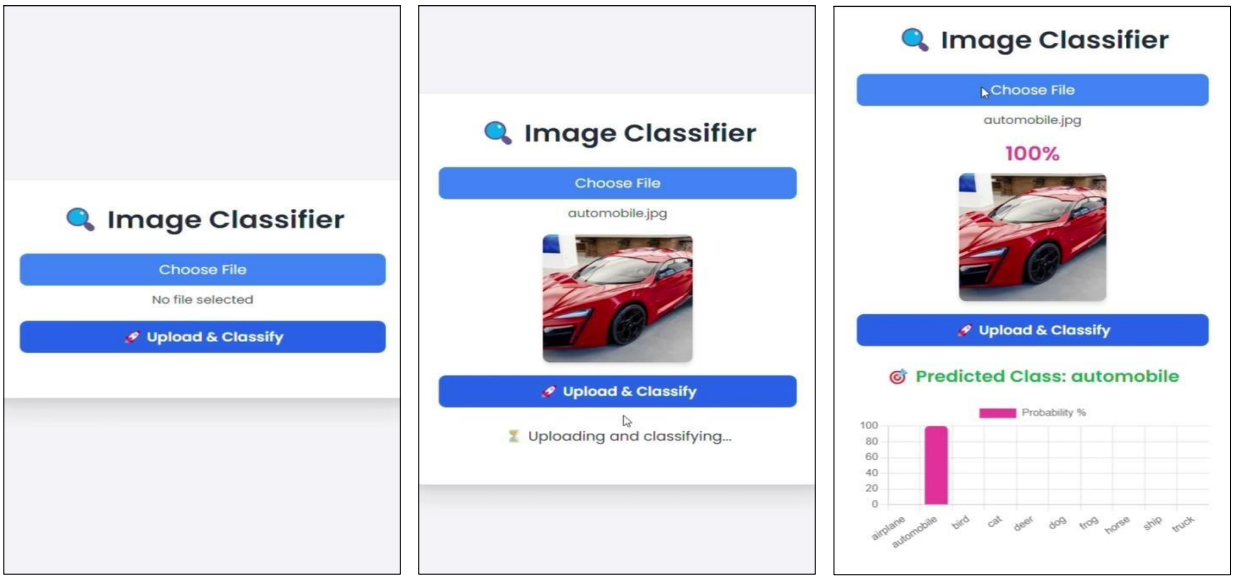
7. Deployment

During our deployment process, we also visualized:

- Real-time Predictions**:** Graphical representation of model predictions.
- Loss and Accuracy Over Time**:** Interactive plots to track performance trends.

As a team, we worked together to deploy our best-performing model as a web application using Flask for the backend and HTML/CSS for the frontend. This phase was both challenging and rewarding, as we had to integrate our trained model into an interactive user interface.

Here's some pictures :



Our deployment process included:

- Saving the best model as cifar10_best_model.h5.
- Building an API with Flask to handle image uploads and return predictions.
- Developing a user-friendly frontend where users can upload images for classification.
- Testing and debugging to ensure smooth functionality.

This experience enabled us to bridge the gap between machine learning concepts and real- world applications. Deploying our model provided valuable insights into integrating AI solutions into practical use cases, reinforcing our understanding of how AI can be effectively implemented to solve real-world problems.