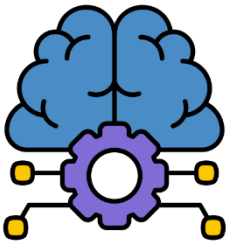


Deep Learning-Image Classification with CNN

Amjad Aloufi Hassa Mohammed
Arwa AbuFeeh

Problem Statement

"Our project is divided into two main parts. The first part involves building a Convolutional Neural Network (CNN) from scratch to classify images into 10 different categories using the CIFAR-10 dataset, aiming for high accuracy and deploying it locally. The second part focuses on utilizing a transfer learning model trained on the CIFAR-10 dataset to predict image classes. Finally, we will compare the results of both approaches to evaluate their performance.

The Dataset

The CIFAR-10 dataset contains 60,000 colour images, each with a resolution of 32×32 pixels. The dataset consists of 10 distinct classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class has an equal number of images, ensuring a balanced distribution. The images are diverse, covering various angles, lighting conditions, and backgrounds

Baseline Model

MODEL 1

1.Data Preprocessing

The CIFAR-10 dataset is normalized to a range of 0 to 1, and the labels are converted to one-hot encoding.

---

2. Convolutional Layers

The first layer is a convolutional layer with 32 filters of size 3x3, using the ReLU activation function to learn complex patterns. It is followed by a MaxPooling2D layer with a 2x2 pool size, reducing the spatial dimensions and helping to prevent overfitting.
The second and third convolutional layers use 64 filters, each followed by MaxPooling2D layers to further downsample the feature maps. The increased filters in later layers allow the model to capture more abstract features.

---

1. Flattening and Dense Layers

After the convolutional and pooling layers, the feature maps are flattened into a 1D vector, passed through a dense layer with 64 neurons using ReLU activation. A Dropout layer with a 30% rate helps prevent overfitting. The final dense layer with 10 units uses the softmax activation function to classify images into one of the 10 CIFAR-10 categories.

---

1. Compilation and Training

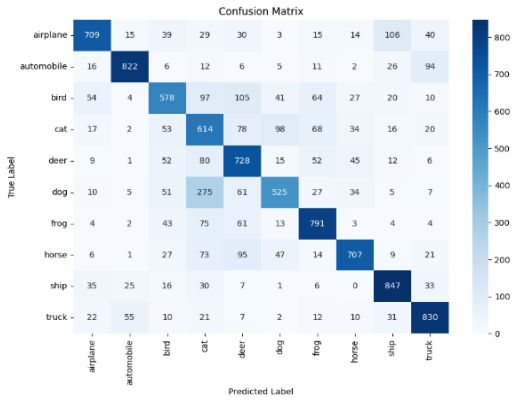
The model is compiled with the Adam optimizer and sparse categorical cross-entropy loss function. It is trained for 20 epochs on the training data to adjust weights and learn features for accurate classification.

---

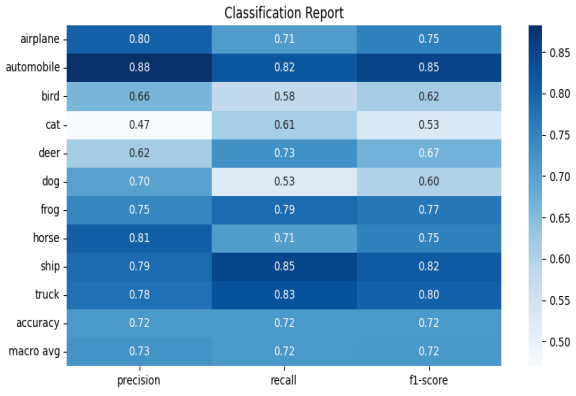
2. Evaluation

After training, the model is evaluated on the test dataset to measure its accuracy in classifying unseen images, ensuring it generalizes well to new data

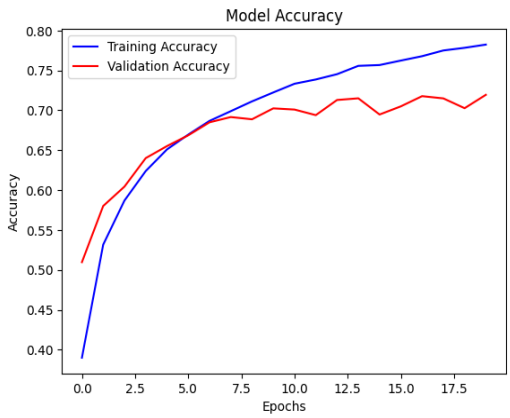
Below is the confusion matrix for Model 1



Below is the classification report



Below is the Training & Validation Accuracy Plot



MODEL 2

1. Data Preprocessing and Augmentation

The CIFAR-10 dataset is normalized to a range of 0 to 1, and the labels are converted to one-hot encoding. Data augmentation is applied using rotation, shifting, shear, zoom, and horizontal flips, which enhances model generalization and prevents overfitting by artificially increasing training data diversity.

---

2. Model Architecture

The CNN model consists of several convolutional layers with increasing filter sizes (64, 128, and 256), followed by batch normalization and max-pooling to reduce spatial dimensions. Dropout layers (30% after convolution blocks and 50% in fully connected layers) help prevent overfitting. The output layer has 10 neurons with a softmax activation function for classification.

---

3. Training Process

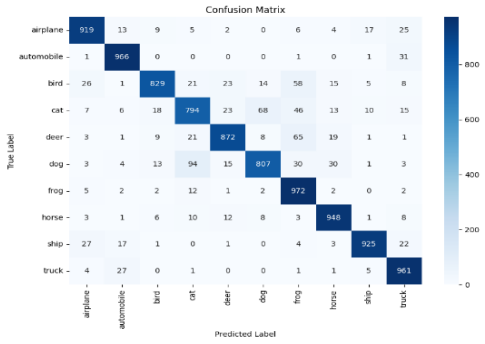
The model is trained using the Adam optimizer with categorical cross-entropy loss. Several callbacks are used: **ModelCheckpoint** saves the best model based on validation accuracy, **ReduceLROnPlateau** reduces the learning rate when validation loss plateaus, and **EarlyStopping** halts training when validation loss doesn't improve, preventing overfitting.

---

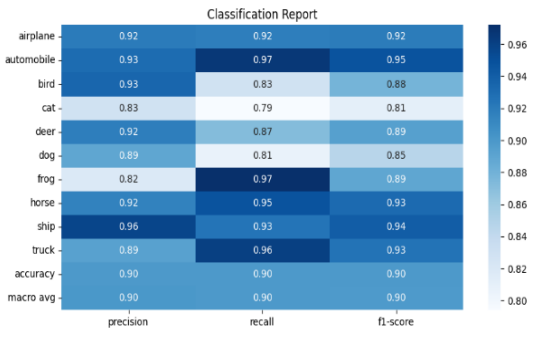
4. Model Evaluation and Performance

After training, the model is evaluated on the test dataset, with the evaluation providing the accuracy score to assess its generalization ability. The combination of techniques ensures high accuracy while minimizing overfitting

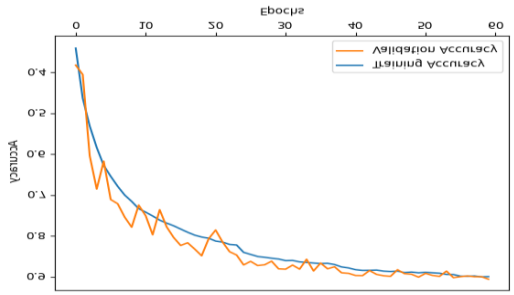
below the confusion matrix for model 2



Below is the classification report



(Training & Validation Accuracy Plot)



MODEL 3

Data Preparation

To enhance training stability, the pixel values are normalized by scaling them between 0 and 1. Additionally, the class labels are converted into a one-hot encoded format, making them compatible with categorical classification during model training.

---

1. Data Augmentation

To improve model generalization, data augmentation is applied using an ImageDataGenerator. This augmentation introduces random transformations such as rotation (up to 15 degrees), horizontal and vertical shifts (up to 10% of the image dimensions), and horizontal flipping. These transformations help the model learn invariant features and prevent overfitting. The missing pixels, if any, are filled using the nearest neighbor strategy.

---

2. Callbacks for Training Optimization

Several callbacks are incorporated to optimize the training process. A ModelCheckpoint ensures that the best-performing model, based on validation accuracy, is saved to prevent overfitting. The ReduceLROnPlateau callback dynamically adjusts the learning rate by reducing it by a factor of 0.5 when validation loss stagnates for five consecutive epochs. Additionally, EarlyStopping halts training when validation loss does not improve for ten consecutive epochs, restoring the best weights recorded during training. These mechanisms help refine the training process and enhance model performance.

---

3. CNN Model Architecture

The CNN model architecture is designed with multiple convolutional layers for feature extraction, followed by fully connected layers for classification. The network comprises four feature extraction blocks, each consisting of three convolutional layers with ReLU activation

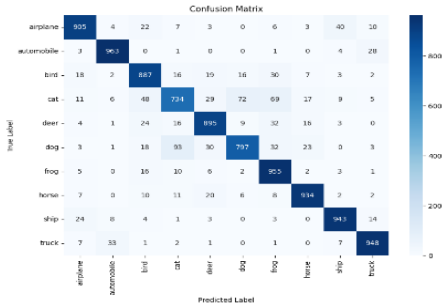
and batch normalization. The number of filters increases progressively from 32 to 256 across these blocks. Each block ends with a max pooling operation to downsample feature maps and a dropout layer to reduce overfitting. After feature extraction, the network transitions to fully connected layers, where the extracted features are flattened and passed through a dense layer with 512 neurons and ReLU activation. A dropout layer with a 50% rate is added to prevent overfitting. Finally, a dense output layer with 10 neurons and softmax activation generates class probabilities.

---

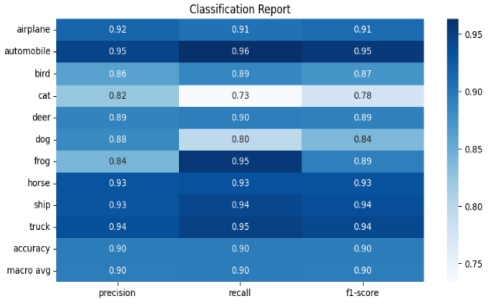
5. Model Compilation and Trainin

The model is compiled using the Adam optimizer with a learning rate of 0.001, and categorical cross-entropy is used as the loss function, which is well-suited for multi-class classification tasks. The training process is conducted over 100 epochs with a batch size of 64, utilizing augmented data for improved generalization. After training, the best model is saved as cifar10_best_model.h5, while the final trained model is stored as cifar10_final_model.h5. The model is then evaluated on the test dataset, and the final accuracy is printed to assess its performance.

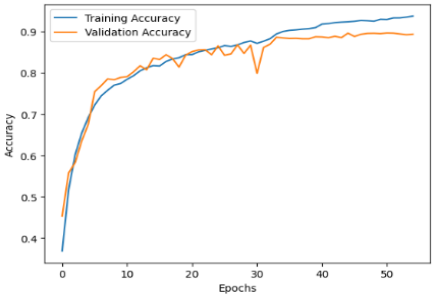
below the confusion matrix for model 3



Here is the classification report



(Training & Validation Accuracy Plot)



PRE-TRAINING MODEL

Training and Evaluation of ResNet50 and VGG16 Models

This report compares the performance of two popular convolutional neural network models, **ResNet50** and **VGG16**, for image classification tasks. Both models were trained on the same dataset to evaluate their accuracy and ability to generalize. While ResNet50 incorporates residual connections to handle deeper architectures, VGG16 uses a simpler design with stacked convolutional layers. Data preprocessing and augmentation techniques were applied to improve model generalization and reduce overfitting. The following sections will provide a detailed analysis of each model's architecture, training process, and performance.

ResNet50

1. Data Preprocessing and Augmentation (ResNet50)

The dataset was normalized to a range of 0 to 1, and labels were converted to one-hot encoding. To improve the model's ability to generalize and prevent overfitting, data augmentation was applied. Techniques such as rotation, shifting, shear, zoom, and horizontal flipping were used to artificially increase the diversity of the training dataset.

---

2. Model Architecture (ResNet50)

The ResNet50 architecture is based on residual learning with 50 layers, using skip connections to mitigate the vanishing gradient problem. These skip connections allow for deeper models to be trained effectively. Despite the advanced architecture, the model experienced overfitting, which hindered its generalization on the validation set.

---

3. Training Process (ResNet50)

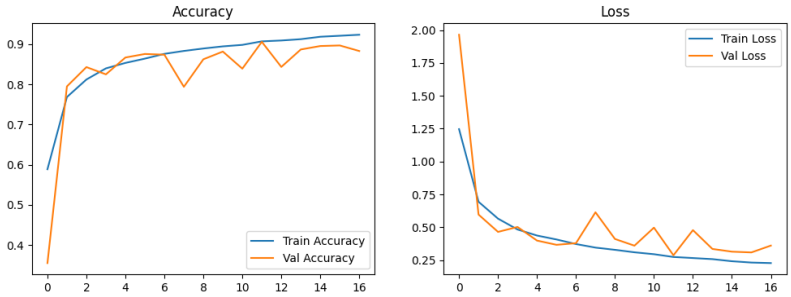
ResNet50 was trained using the SGD optimizer and categorical cross-entropy loss. The model reached **91% training accuracy** but suffered from overfitting, as evidenced by its validation accuracy of **88%**. The training process was halted early to prevent further overfitting, as the validation accuracy declined after a certain point while training accuracy continued to increase.

---

4. Model Evaluation and Performance (ResNet50)

After training, ResNet50 was evaluated on the validation set. Despite high training accuracy, the model showed a significant gap between training and validation accuracy, indicating

overfitting. The training history plot



demonstrates this trend, where training accuracy continued to rise, but validation accuracy plateaued and then decreased.

VGG16

---

1. Data Preprocessing and Augmentation (VGG16)

Similar to the ResNet50 model, the VGG16 model was trained on normalized data (range 0 to 1) with one-hot encoded labels. Data augmentation was applied using techniques such as rotation, shifting, shear, zoom, and horizontal flipping to boost the model's generalization ability and reduce overfitting.

---

2. Model Architecture (VGG16)

VGG16 is a simpler deep neural network architecture consisting of 16 layers, primarily composed of convolutional layers followed by fully connected layers. The straightforward architecture of VGG16 helped it achieve high performance, with a steady increase in both training and validation accuracy throughout the training process.

---

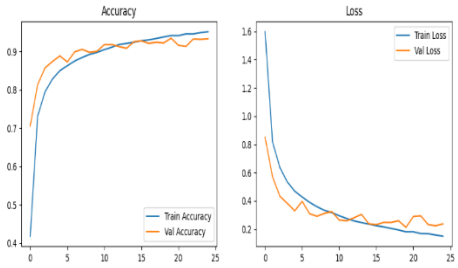
3. Training Process (VGG16)

VGG16 was trained with the SGD optimizer and categorical cross-entropy loss. The model achieved **96% training accuracy** and **93% validation accuracy**. The steady improvement in both training and validation accuracy suggests that VGG16 was able to learn useful features without significant overfitting, making it more robust than ResNet50 in this instance.

---

4. Model Evaluation and Performance (VGG16)

After training, VGG16 was evaluated on the validation set. With **93% validation accuracy**, the model demonstrated better generalization than ResNet50. The training history plot



shows a consistent increase in both training and validation accuracy, confirming the model's ability to generalize well to unseen data and avoid overfitting.reveals a steady increase in both training and validation accuracy, suggesting the model is successfully learning without significant overfitting.

Comparison of CNN Models and Pretrained Models

In this study, we compared the accuracy of three custom **Convolutional Neural Network (CNN) models** with two **pretrained models**, VGG16 and ResNet50. The objective was to analyze the performance differences between **manually designed CNN architectures** and **well-established pretrained models** on the given dataset.

CNN Models Performance

- **Model 1:** Achieved an accuracy of **73.1%**
- **Model 2:** Improved to **81.2%**
- **Model 3:** Reached **90.17%**, showing significant improvement over the first two models

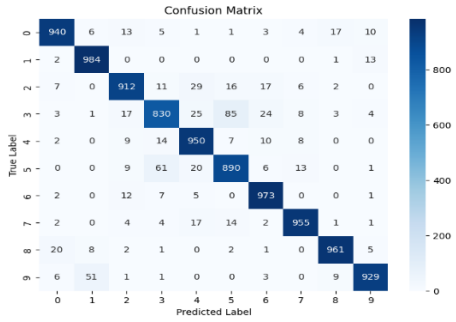
The increasing accuracy from **Model 1 to Model 3** suggests that as the **architecture complexity** increased, the models were able to learn better representations of the data. However, even the best-performing CNN model (**Model 3 at 90.17%**) was still outperformed by the pretrained models.

Pretrained Models Performance

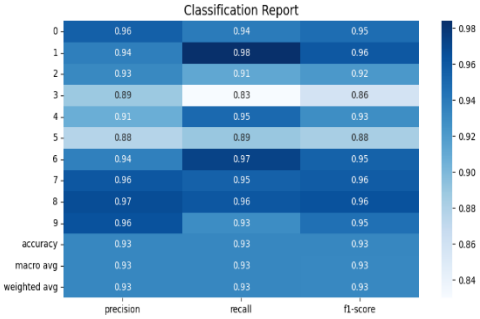
- **VGG16:** Achieved the highest accuracy of **95%**
- **ResNet50:** Reached **92%**, slightly lower than VGG16 but still outperforming all custom CNN models

Pretrained models like **VGG16 and ResNet50** leverage **transfer learning**, which allows them to utilize knowledge gained from large-scale datasets like ImageNet. This gives them an advantage in feature extraction and generalization, leading to higher accuracy compared to

below the confusion matrix

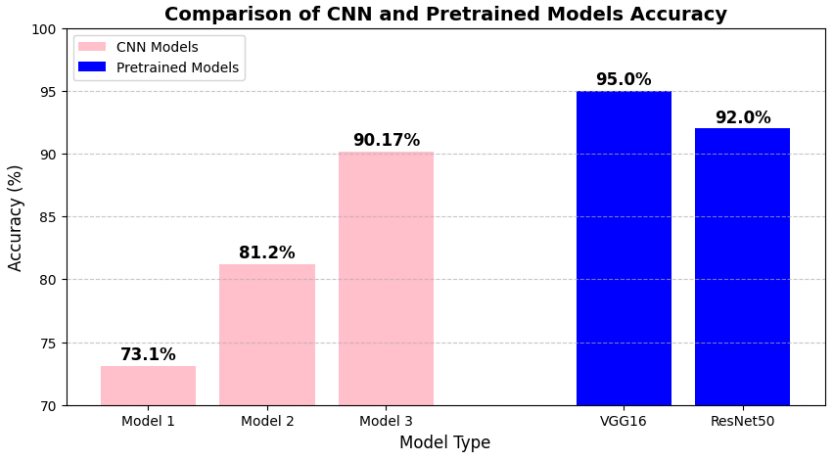


Here is the classification



Graphical Representation

To visualize the performance difference, we plotted a **bar chart** separating the **CNN models** and **pretrained models**. The graph clearly illustrates the accuracy gap, highlighting that while custom CNNs show improvements with increased depth and optimization, **pretrained models still provide superior performance due to their robust feature extraction capabilities**.



Deployment

We developed a Convolutional Neural Network (CNN) model and subsequently experimented with various pre-trained models (transfer learning models) to enhance the performance of the model, as outlined previously. The final stage of the project involved deploying the model locally as a web application using Python, HTML, and the project files. As a result, the project now features an improved interface, making it more accessible and user-friendly for the end user

Conclusion

While manually designed CNN models can achieve competitive performance with **careful tuning and architecture design**, pretrained models like **VGG16 and ResNet50** demonstrate the benefits of **transfer learning** by achieving **higher accuracy with less training time**. This analysis suggests that for practical applications, utilizing pretrained models or fine-tuning them for specific datasets can yield better results than training a CNN from scratch.