الأكاديمية السعودية الرقمية
SAUDI DIGITAL ACADEMY

Deep Learning: Image Classification With CNN

Convolutional Neural Network (CNN) Report
Prepared by: Group 4

Introduction

Convolutional Neural Networks (CNNs) have significantly advanced image classification by automating feature extraction and capturing hierarchical patterns in visual data. This report presents the implementation of a CNN model for classifying images from the CIFAR-10 dataset, which consists of 60,000 images categorized into ten classes: airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

The objective of this project was to develop a robust CNN architecture that effectively classifies images while minimizing overfitting. The model was trained using various preprocessing techniques, optimization strategies, and hyperparameter tuning to improve accuracy. This report details:

- The chosen CNN architecture.
- Data preprocessing techniques applied to enhance model performance.
- The training process, including hyperparameters and regularization methods.
- An evaluation of the model's accuracy and generalization capabilities.
- Insights gained from experimentation and recommendations for future improvements.

Through systematic testing of various architectures and training strategies, this study aims to determine the most efficient and accurate CNN model for CIFAR-10 classification.

Step 1: Understanding and Preprocessing the Data

We started by exploring the **CIFAR-10 dataset**, which consists of **60,000 images (32×32 pixels) across 10 categories**. Given the small image size, feature extraction had to be handled carefully to maximize classification performance. To improve our model's generalization, we implemented the following preprocessing techniques:

- **Normalization:** We scaled pixel values to the range **[0,1]** to stabilize training.
- **One-Hot Encoding:** Converted class labels into categorical vectors for multi-class classification.
- **Data Augmentation:** To improve robustness and prevent overfitting, we applied:
    - **Random Rotations (up to 20°)**
    - **Width and Height Shifts (20%)**
    - **Zoom Transformations (20%)**
    - **Shear Transformations (20%)**
    - **Horizontal Flipping.**

Step 2: Designing the CNN Architecture

Since we were building the model from scratch, we experimented with different configurations before finalizing our deep CNN. The final architecture consisted of**:**

- **Four Convolutional Blocks:** Each containing:
    - **Two Conv2D layers** with **ReLU activation** and increasing filters **(64 → 128 → 256 → 512).**
    - **Batch Normalization** after every convolutional layer for faster convergence.
    - **MaxPooling layers** to downsample spatial dimensions.
    - **Dropout layers** to prevent overfitting.
- **Flattening Layer:** To convert feature maps into a 1D vector.
- **Fully Connected Layers:**
    - **512 Neurons (ReLU activation) + Batch Normalization + Dropout (50%)**
    - **Softmax Layer (10 neurons) for multi-class classification.**

Step 3: Training the Model – Overcoming Challenges

During training, we faced several optimization challenges:

- **Choosing the Right Learning Rate**:
    - Initially, we set the learning rate too high, leading to unstable training.
    - After experimentation, we settled on 0.001 with the Adam optimizer for adaptive learning rate adjustments.
- **Overfitting Issues:**
    - Our initial model overfitted the training data.
    - We mitigated this by applying L2 regularization (1e-4) on Conv2D layers and adding Dropout layers (30-50%).
- **Choosing the Right Batch Size & Epochs:**
    - Training for 200 epochs was too long, so we used Early Stopping to halt training when validation loss stopped improving.
    - A batch size of 64 provided the best balance between training speed and stability.

4. Results & Model Comparison – Evaluating Performance:

After successfully training our CNN model from scratch, we compared its performance against two benchmarks:

1. **Baseline Model** – A simple CNN architecture trained from scratch with fewer layers.
2. **Best Model** – Our optimized deep CNN, incorporating **batch normalization, dropout, and L2 regularization**.
3. **VGG16 (Pre-trained)** – A well-established deep CNN trained on ImageNet and fine-tuned for CIFAR-10.

Model	Total params	Trainable params	Non-trainable params	Number of layers	LR	Last/epochs
Bale line model	2,609,034	2,607,114	1,920	4	0.0005	44/60
Our Model	5,749,322	5,744,458	4,864	10	0.0001	96/200
VGG16	14,714,688	0	14,714,688	16	0.0001	15/50

To assess performance, we used **accuracy, precision, recall, and F1-score** as evaluation metrics. The results for the **Baseline Model, Custom CNN Model, and VGG16 (Pre-Trained Model)** are presented in the table below:

Metric	Base line	Our Model	VGG16 (Pre-trained)
Accuracy	0.7799	0.9215	0.9350
Precision	0.8115	0.9183	0.9246
Recall	0.8118	0.9187	0.9239
F1 Score	0.8088	0.9181	0.9239

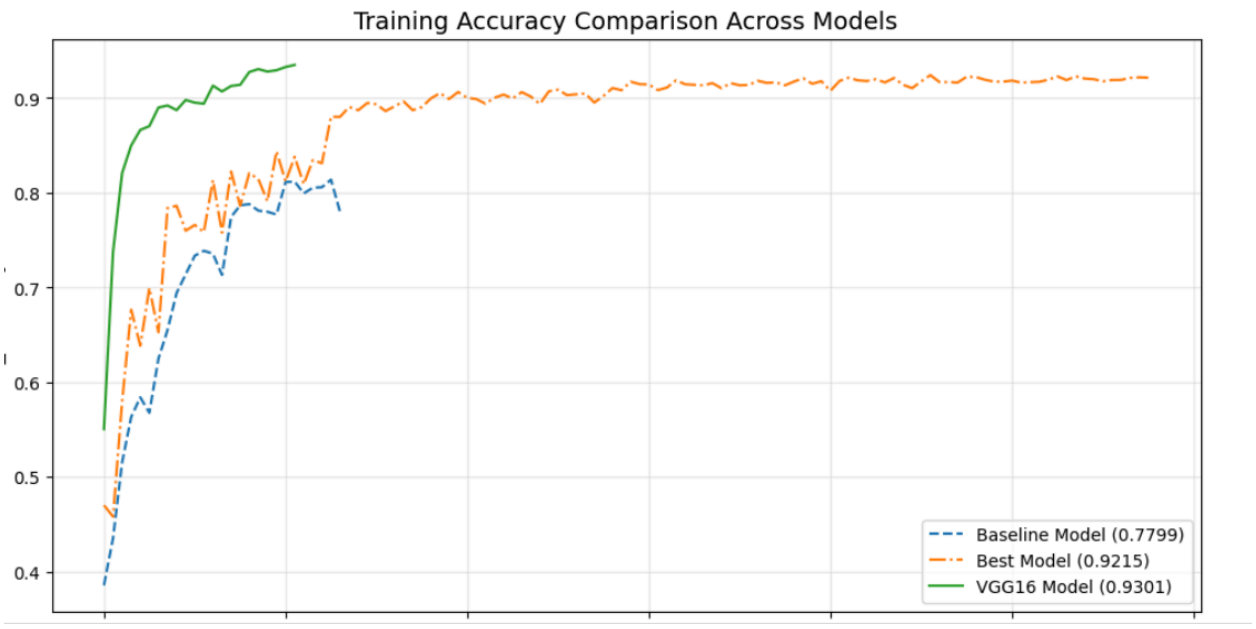
Best Model Selection & Justification:

Our **best-performing model** was **VGG16 (Pre-Trained Model)**, which achieved the highest accuracy of **93.50%**, surpassing both the **Baseline Model (77.99%)** and our **Custom CNN Model (92.15%)**. The primary factors contributing to VGG16's superior performance include its **pre-trained feature extractors, deeper architecture, and higher input resolution**.

However, our **Custom CNN Model remains a strong alternative**, achieving **high accuracy while being computationally more efficient**.

Key Factors Contributing to VGG16's Superior Performance

1. **Pre-Trained Feature Extractors**
    - VGG16 was trained on the large-scale **ImageNet dataset**, allowing it to recognize complex patterns immediately.
    - Unlike our **custom CNN model, which learned from scratch**, VGG16 utilized pre-trained weights, leading to faster learning and improved accuracy.
2. **Deeper Architecture & Higher Parameter Count**
    - VGG16 consists of **16 layers and 14.7 million parameters**, compared to our **10-layer CNN with 5.7 million parameters**.
    - The additional depth allowed VGG16 to capture **more hierarchical features**, improving classification performance.
3. **Higher Input Resolution (224×224 vs. 32×32)**
    - Our model trained on **32×32 images**, while **VGG16 required resizing images to 224×224**.
    - The larger input size retained more spatial details, enhancing feature extraction.



The figure above illustrates the **training accuracy progression** of three different models—**Baseline Model, Best Model (our CNN), and VGG16 (pre-trained)**—over the training epochs.

Baseline Model (Blue, Dashed Line):

- Initially struggles to learn, with slower accuracy improvement.
- Experiences **more fluctuations** throughout training, indicating instability.
- Converges at **~77.99% accuracy**, significantly lower than the other models.

Our Model (Orange, Dash-Dotted Line):

- **Faster learning rate** compared to the baseline, with improved stability.
- Achieved **92.15% accuracy**, proving the effectiveness of **batch normalization, dropout, and L2 regularization**.

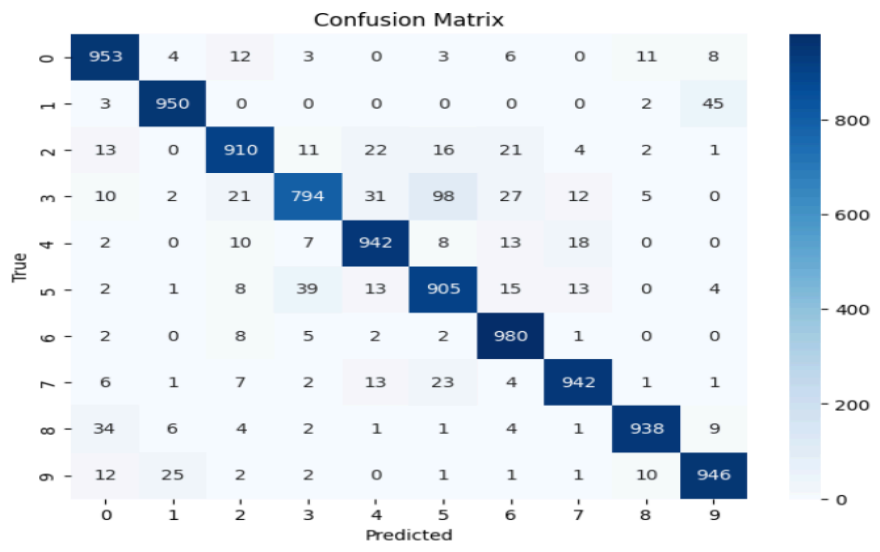
VGG16 Pre-Trained Model (Green, Solid Line):

- Learn **very quickly**, reaching high accuracy within the first few epochs.
- Converges at **93.50% accuracy**, slightly outperforming our best model.
- The rapid improvement suggests that **pre-trained feature extractors** on ImageNet provided a strong starting point.

Confusion Matrix Analysis

To further understand model performance, we analyzed the **confusion matrices** of both models to assess **misclassifications and class-wise performance**.

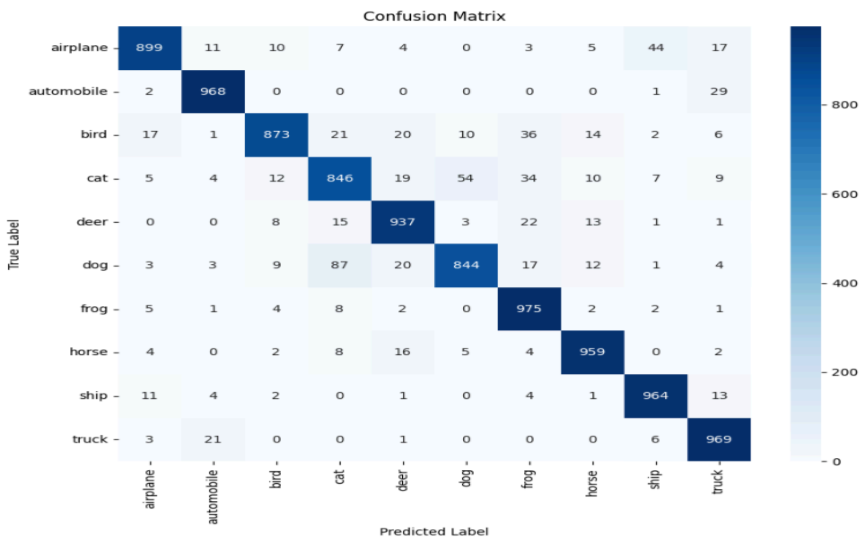
VGG16 Confusion Matrix



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	953	4	12	3	0	3	6	0	11	8
1	3	950	0	0	0	0	0	0	2	45
2	13	0	910	11	22	16	21	4	2	1
3	10	2	21	794	31	98	27	12	5	0
4	2	0	10	7	942	8	13	18	0	0
5	2	1	8	39	13	905	15	13	0	4
6	2	0	8	5	2	2	980	1	0	0
7	6	1	7	2	13	23	4	942	1	1
8	34	6	4	2	1	1	4	1	938	9
9	12	25	2	2	0	1	1	1	10	946

- The VGG16 model correctly classified most images, benefiting from **pre-trained feature extraction**.
- However, **misclassifications occurred in visually similar categories**, suggesting that fine-tuning could further improve performance.

Custom CNN Model Confusion Matrix



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	899	11	10	7	4	0	3	5	44	17
automobile	2	968	0	0	0	0	0	0	1	29
bird	17	1	873	21	20	10	36	14	2	6
cat	5	4	12	846	19	54	34	10	7	9
deer	0	0	8	15	937	3	22	13	1	1
dog	3	3	9	87	20	844	17	12	1	4
frog	5	1	4	8	2	0	975	2	2	1
horse	4	0	2	8	16	5	4	959	0	2
ship	11	4	2	0	1	0	4	1	964	13
truck	3	21	0	0	1	0	0	0	6	969

- While slightly less accurate, the **custom CNN model performed well in most categories**.
- The use of **batch normalization, dropout, and L2 regularization helped reduce overfitting and improve generalization**.

Insights Gained from Experimentation

1. **Transfer Learning Accelerates Training:**

   The VGG16 model benefited from pre-trained weights, achieving high accuracy with fewer training epochs.

2. **Deeper Networks Extract More Features, But at a Cost:**

   While VGG16 performed slightly better, it also required significantly more computational resources due to its large number of parameters.

3. **Optimized Custom Models Can Compete with Pre-Trained Models:**

   Despite being trained from scratch, our custom CNN model reached 92.15% accuracy, proving that a well-designed architecture can rival deep pre-trained networks while being more efficient.

Conclusion

Based on the results, **VGG16** is identified as the **best-performing model** in this study due to its pre-trained feature extraction capabilities, deeper architecture, and ability to process higher-resolution images. However, the custom CNN model demonstrated strong performance while being computationally more efficient, highlighting the potential of well-designed CNN architectures in achieving high classification accuracy without the need for extensive pre-training.