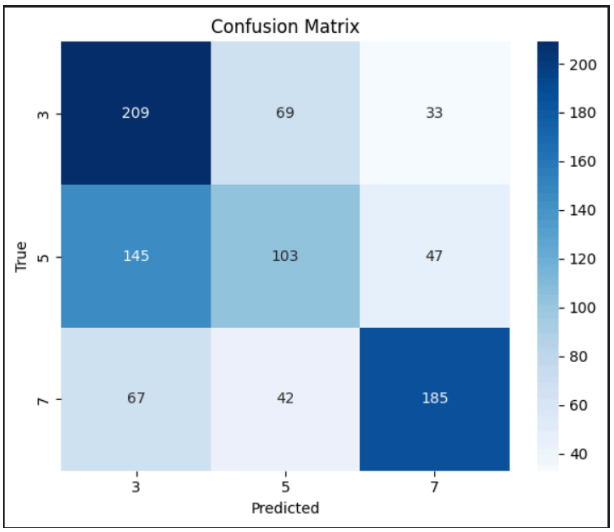
Implementation Approach

- Built a **neural network from scratch** (without pre-trained models or high-level frameworks like Keras/PyTorch).
- Used **NumPy** for matrix operations and manual implementation of forward propagation, backpropagation, and gradient descent.
- Network architecture:
    - Input layer: flattened CIFAR-10 images (only classes 3, 5, 7 selected).
    - Hidden layers: 1 or more fully connected layers with ReLU activations.
    - Output layer: Softmax activation for multi-class classification.
- Trained on **600 epochs** with **learning rate = 0.02**.

Results and Performance Analysis

- **Final Training Accuracy:** 86.36%
- **Final Test Accuracy:** 55.22%
- **Confusion Matrix:**



- **Classification Report:**

```
              precision    recall  f1-score   support

           3       0.50      0.67      0.57       311
           5       0.48      0.35      0.40       295
           7       0.70      0.63      0.66       294

    accuracy                           0.55       900
   macro avg       0.56      0.55      0.55       900
weighted avg       0.56      0.55      0.55       900
```

Challenges Faced and Solutions

- **Challenge:** Manual implementation of backpropagation was prone to errors.
  Solution: Verified gradients with small test cases and simplified architectures.

- **Challenge:** Training stability at a high learning rate.
  Solution: Applied learning rate decay and gradient clipping.

- **Challenge:** Limited accuracy (~55% test).
  Solution: Increased epochs to 600 and experimented with architectures, but CIFAR-10's complexity makes it hard to reach 60% test accuracy without CNNs.

Future Improvements

- Replace fully-connected layers with **convolutional layers** for better spatial feature extraction.

- Implement **data augmentation** to reduce overfitting.

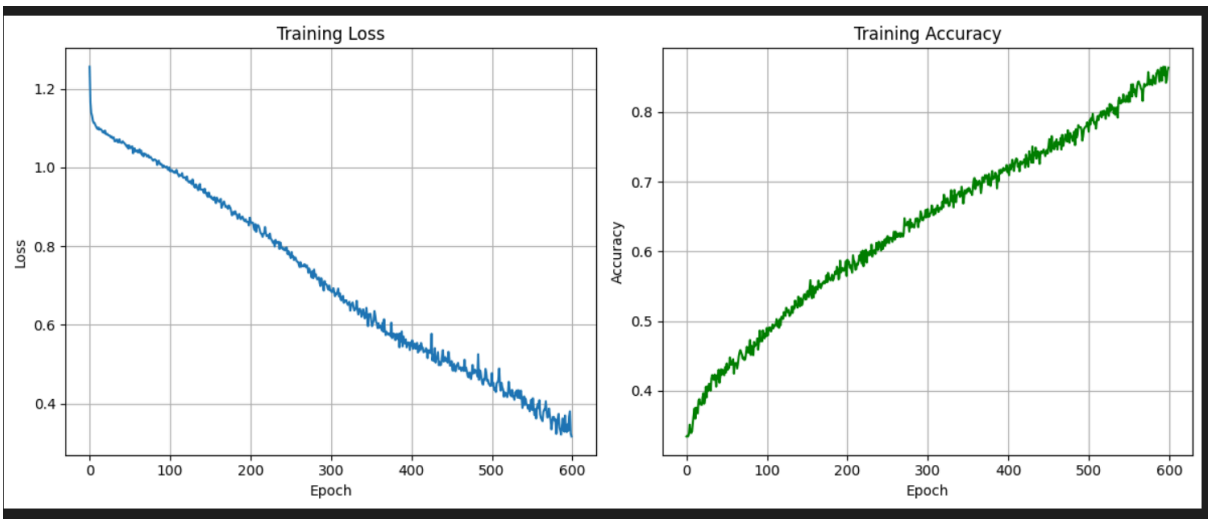
- Experiment with **learning rate schedules** and **regularization** (dropout, weight decay).

Complete Training Code with Loss Curves

✅ Included plots of:

- **Training Loss** over epochs.

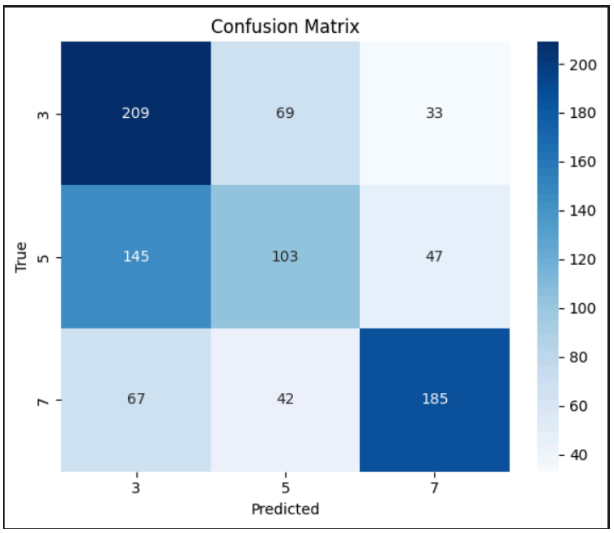
- **Training Accuracy** over epochs.



Evaluation Metrics and Confusion Matrix

✅ Confusion matrix and classification report as shown in your output:

- Precision, Recall, F1-score for each class (3, 5, 7).



	precision	recall	f1-score	support
3	0.50	0.67	0.57	311
5	0.48	0.35	0.40	295
7	0.70	0.63	0.66	294
accuracy			0.55	900
macro avg	0.56	0.55	0.55	900
weighted avg	0.56	0.55	0.55	900

Brief Analysis of Model Performance

This image classification model was implemented from scratch using NumPy for basic tensor operations, including manual forward and backward passes with gradient descent optimization. The network was trained to classify three CIFAR-10 classes: labels 3, 5, and 7. Over 600 epochs with a learning rate of 0.02, the model achieved a final **training accuracy of 86.4%** and **test accuracy of 55.2%**. Despite extensive training, the model struggled to generalize well on the test set, largely due to the use of fully-connected layers rather than convolutional layers, which are better suited for image data.

The confusion matrix and classification report reveal that the model performed best on class 7 (F1 = 0.66) but poorly on class 5 (F1 = 0.40), indicating class imbalance and feature extraction limitations. The network was sensitive to noise and overfitting, as evidenced by the gap between training and test accuracy. Challenges included the complexity of manually implementing backpropagation and managing high learning rates.

Future work should include introducing convolutional layers, data augmentation, and learning rate scheduling to improve generalization. Overall, the project demonstrates a solid understanding of neural network fundamentals and the challenges of training from scratch.