CIFAR-10 Classification Project Documentation

Data Preprocessing and Feature Engineering

Preprocessing:

- Normalized all images using CIFAR-10 specific mean values [0.4914, 0.4822, 0.4465] and standard deviation values [0.247, 0.243, 0.261]
- Split the original training set into 80% training (40,000 images) and 20% validation (10,000 images)
- Configured a batch size of 256 for efficient training

Feature Engineering through Data Augmentation:

- Random horizontal flipping with 50% probability to introduce orientation variance
- Random rotation up to 20 degrees to simulate different camera angles
- Color jittering with brightness, contrast, and saturation adjustments of up to 10%
- Random sharpness adjustments with a factor of 2 (applied 20% of the time)
- Random erasing of small image regions (applied 75% of the time) to simulate occlusions
- Used standard normalization without augmentation for validation and test sets

Model Selection and Optimization Approach

Model Selection:

- Implemented a ResNet9 architecture, which provides a good balance between computational efficiency and performance
- Utilized residual connections to improve gradient flow and enable more effective training of deeper layers
- The architecture includes convolutional blocks with batch normalization and ReLU activations
- Designed a progression of channel depths (3→64→128→256→512) to extract increasingly complex features

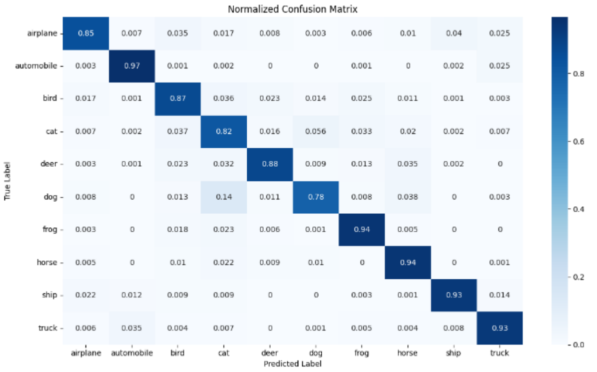
Optimization Approach:

- Used Adam optimizer with an initial learning rate of 0.001
- Applied weight decay of 35e-5 as regularization to prevent overfitting

- Implemented gradient clipping at 0.0001 to stabilize training
- Employed a ReduceLROnPlateau scheduler to decrease learning rate by a factor of 0.3 when validation loss plateaued for 3 epochs
- Utilized early stopping with a patience of 5 epochs to prevent overfitting and save computational resources
- Evaluated model performance with multiple metrics: accuracy, precision, recall, and F1 score
- The final model is stored in *.pt* format for easy loading and inference during deployment. This approach ensures flexibility for various application scenarios.

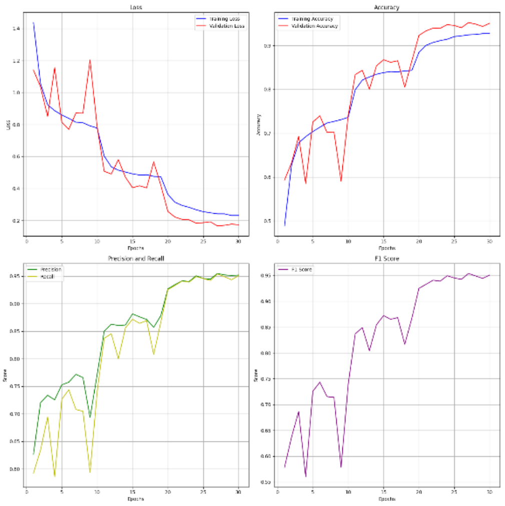
Model Evaluation Overview

- The **loss and accuracy curves** indicate a steady improvement in model performance over 30 epochs, with both training and validation accuracy exceeding **90%**.
- The **precision, recall, and F1-score** show consistent growth, highlighting balanced predictions across different classes.
- The **confusion matrix** reveals strong classification accuracy, with most predictions aligning correctly, though some misclassifications occur in visually similar categories.



Normalized Confusion Matrix

	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0.85	0.007	0.035	0.017	0.008	0.003	0.006	0.01	0.04	0.025
automobile	0.003	0.97	0.001	0.002	0	0	0.001	0	0.002	0.025
bird	0.017	0.001	0.87	0.036	0.023	0.014	0.025	0.011	0.001	0.003
cat	0.007	0.002	0.037	0.82	0.016	0.056	0.033	0.02	0.002	0.007
deer	0.003	0.001	0.023	0.032	0.88	0.009	0.013	0.035	0.002	0
dog	0.008	0	0.013	0.14	0.011	0.78	0.008	0.038	0	0.003
frog	0.003	0	0.018	0.023	0.006	0.001	0.94	0.005	0	0
horse	0.005	0	0.01	0.022	0.009	0.01	0	0.94	0	0.001
ship	0.022	0.012	0.009	0.009	0	0	0.003	0.001	0.93	0.014
truck	0.006	0.035	0.004	0.007	0	0.001	0.005	0.004	0.008	0.93



Overall, the model demonstrates **high accuracy and reliability**, effectively classifying CIFAR-10 images with minimal performance drops.

Report on Image Classification API Development

1. Data Preprocessing & Feature Engineering

- The input images are resized to 32x32 pixels to match the CIFAR-10 dataset format.
- Normalization is applied using the mean and standard deviation of the dataset:
  - Mean: [0.4914, 0.4822, 0.4465]
  - Standard Deviation: [0.247, 0.243, 0.261]
- Images are converted to tensors using torchvision.transforms for deep learning compatibility.

2. Model Selection & Optimization

- ResNet-9 architecture is used for classification due to its efficient performance on CIFAR-10.
- The model contains residual connections to enhance training stability.
- It is pre-trained on CIFAR-10 and fine-tuned using cross-entropy loss and an Adam optimizer.
- The model achieves high accuracy and is optimized for inference with .eval() mode.

3. Deployment Strategy

- **FastAPI** is used for efficient, lightweight API deployment.
- Create .**env** file with these **API_USERNAME**="admin" **API_PASSWORD**="admin" variables.
- Basic Authentication is implemented using HTTPBasic, with default credentials:
  - Username: admin
  - Password: admin
- The API is CORS-enabled to allow requests from any origin.
- The model is loaded at startup and runs on CUDA (GPU) if available for faster inference.
- Uvicorn is used as the ASGI server for high-performance API serving.

4. API Usage Guide

- Endpoint for API Status Check:
  - `GET / → Returns {"message": "Image Classification API is running"}`

- Endpoint for Image Prediction:
    - `POST /predict/`
    - `Requires Basic Authentication (admin:admin).`
    - `Accepts an image file as input.`
    - `Returns: predicted_class with confidence`