Image Classification Using CNN in PyTorch

- Objective: Build a CNN-based image classifier using a custom dataset organized in class-wise folders.
- Task: Train a model to accurately classify images into their respective categories.
- Tools & Frameworks: PyTorch, Google Colab, Pretrained AlexNet

Dataset Description

- Source: Custom dataset uploaded to Google Drive
- Structure: Folder-based format (/Train and /Test)
- Total Classes: N (replace with actual number)
- Class Names: ['cat', 'dog', 'snake', ...]
- Training samples: X, Testing samples: Y

Modified AlexNet Architecture

- Base Model: Pretrained AlexNet
- Feature Extractor: Frozen
- Classifier: Linear(9216 → 1024) → ReLU → Dropout(0.4) → Linear(1024 → num_classes) → LogSoftmax
- Loss Function: NLLLoss
- Optimizer: Adam (LR=0.001)

Training Workflow

- Data Augmentation: Rotation, HorizontalFlip, Normalize
- Dataloaders: Batch size = 10, Shuffle enabled
- Device: GPU/CPU support
- Training: 50 epochs, accuracy & loss tracking, batch limits (800/300)

Accuracy & Loss Plots

- Accuracy vs Epochs: (Insert plot image)
- Loss vs Epochs: (Insert plot image)
- Final Test Accuracy: 93.00%
- Training Duration: 756 seconds

Sample Inference & Evaluation

- Example Inference: Test image with predicted label
- Evaluation Metrics: Confusion Matrix (Insert heatmap)
- Visualizes correct and incorrect predictions

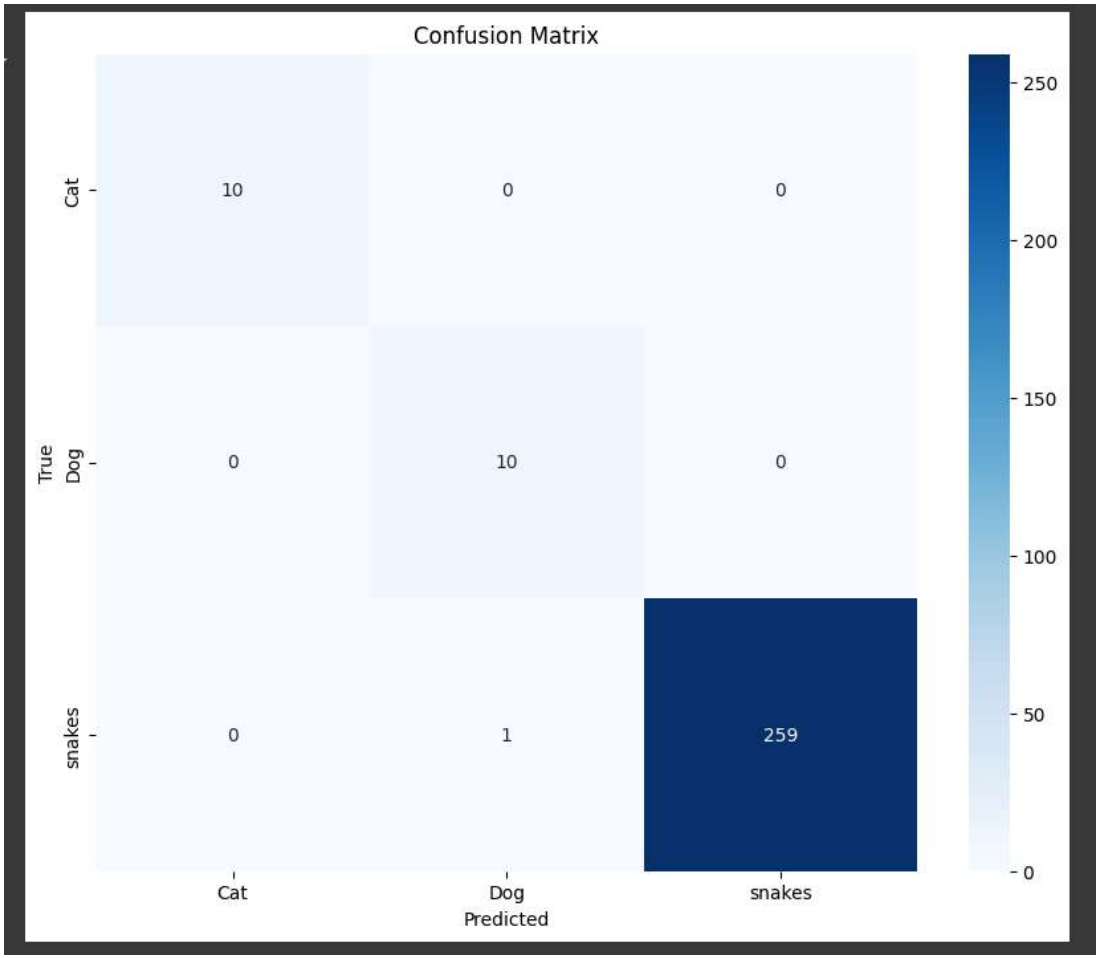
Challenges Faced

- Overfitting: Handled with dropout & augmentation
- Class Imbalance: Ensured balanced dataset and shuffling
- Large Model: Froze feature extractor layers
- Hardware Limits: Used batch caps and GPU acceleration

Conclusion & Learnings

- Successfully implemented a CNN-based classifier in PyTorch
- Used transfer learning (AlexNet) for faster convergence
- Model generalizes well to unseen data
- Hands-on experience in tuning, preprocessing, evaluation

Confusion Matrix Output



Predicted Output