CIFAR-10 Image Classification Using CNNs

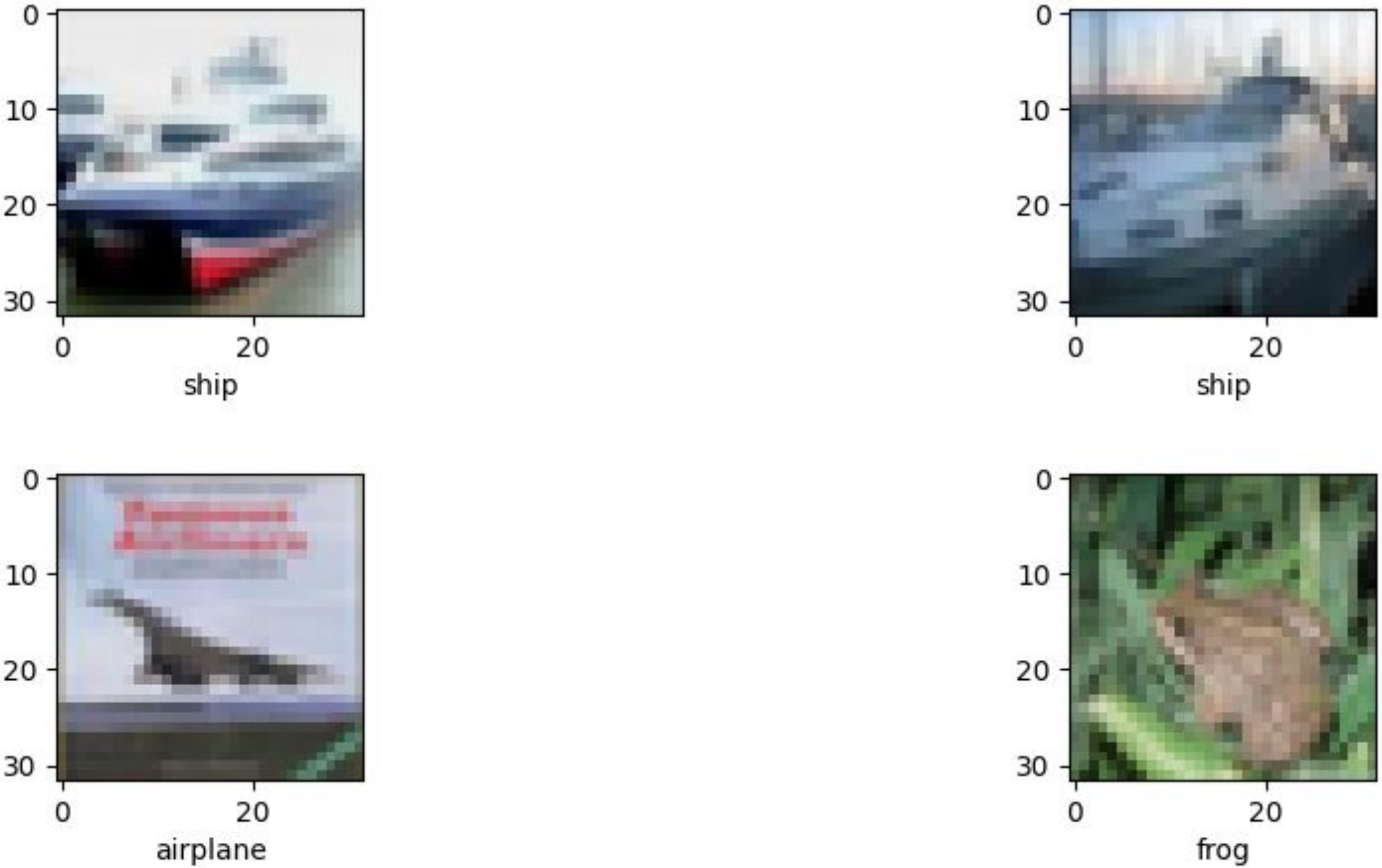
Noah Wiley

Project Overview

- Goal: Build and evaluate a deep learning model for CIFAR-10
- Apply CNNs, training strategies, and hyperparameter tuning
- Compare baseline vs. improved model
- Final accuracy achieved: **78%**

Dataset Overview

- CIFAR-10: 60,000 images, 10 balanced classes
- 32×32 RGB images
- Classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
- Preprocessing:
  - ToTensor()
  - Normalization to mean=0, std=1


ship
ship
airplane
frog

Why CIFAR-10?

Popular benchmark for computer vision

Small enough for fast experimentation

Ideal for testing CNN architectures learned in class

Balanced dataset → clean evaluation metrics

Baseline Model

Started with PyTorch tutorial CNN

Architecture:

- 2 convolutional layers
- Max pooling
- 3 fully connected layers

Performance:

- **~60% accuracy**

Limitations: Too shallow, insufficient feature extraction

Final Model Architecture

5 Convolutional Layers:

- Filters: 32 → 64 → 128 → 256 → 512
- 3×3 kernels
- BatchNorm after each convolution
- ReLU activation
- Max Pooling after first 4 layers
- Fully connected head:
    - 2048 → 64 → 10 classes

Why This Architecture?

-

Progressive filter increase → deeper feature extraction

BatchNorm → stabilizes training

Reduced over-pooling → retains spatial detail

Deeper model learned more complex visual patterns

Hyperparameter Exploration

- Optimizers: SGD → Adam
- Learning rate tuning significantly impacted performance
- Batch size: 4 → 128
- Epochs: 2 → 30 (final choice: **20**)
- Tried altering filter sizes & kernel sizes
  - Many configurations reduced accuracy
- BatchNorm gave +2–3% performance boost

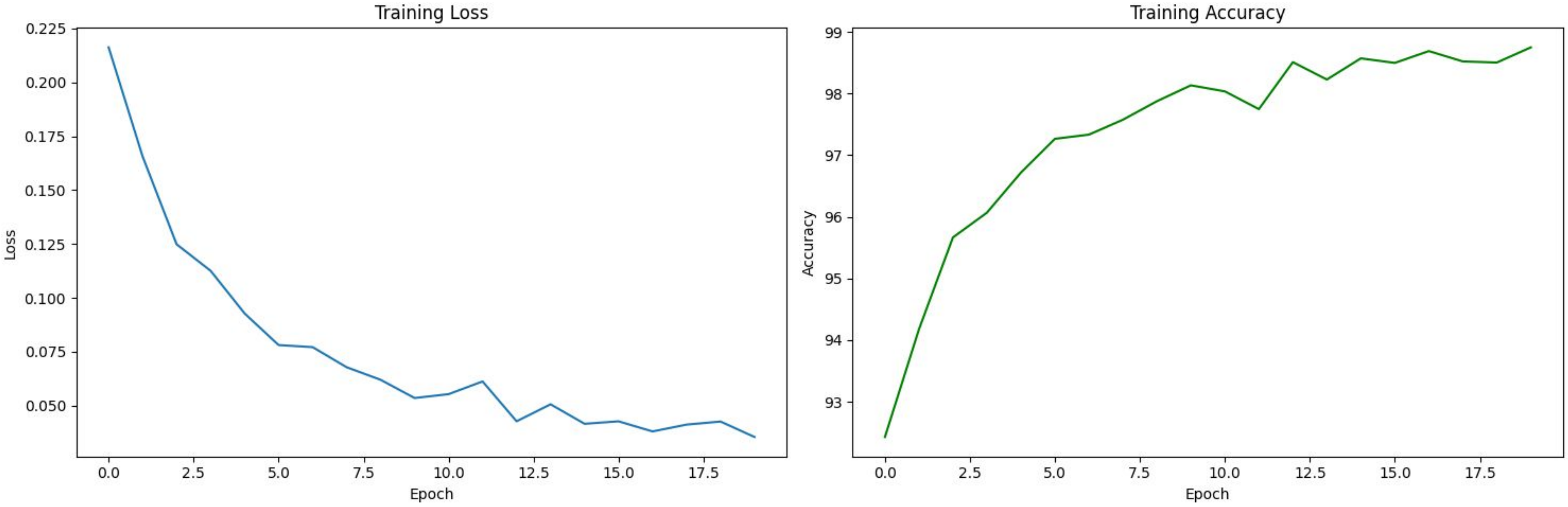
Training Results

Training accuracy reached **~98%**

Loss minimized over epochs

Training curves showed plateau around epoch 20

Indicates model was effectively learning the data

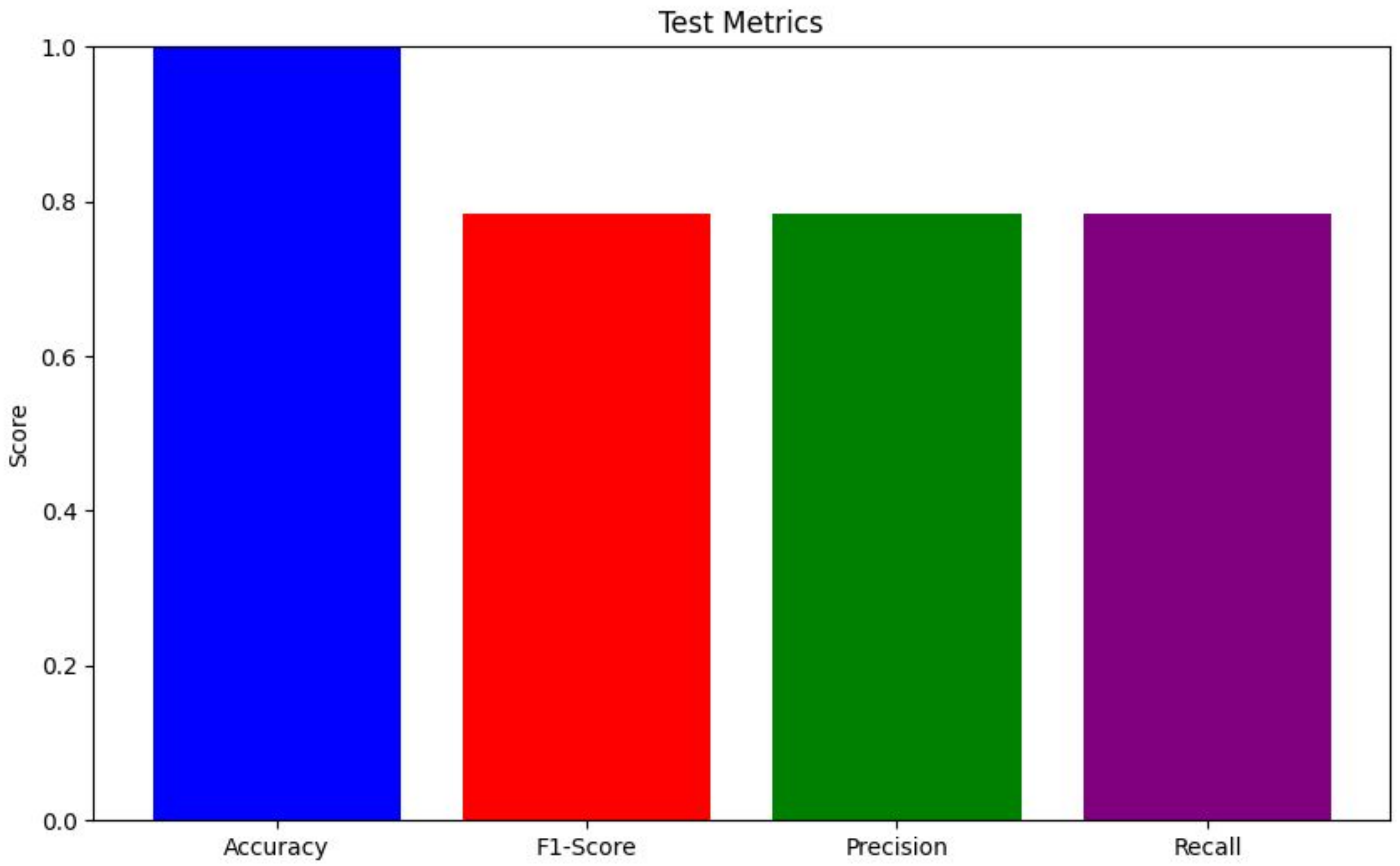

Training Loss
Loss
Epoch
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Training Accuracy
Accuracy
Epoch
99
98
97
96
95
94
93

Test Results

- Final test accuracy: **78%**
- Precision: ~0.80
- Recall: ~0.79
- F1-score: ~0.79
- Model generalizes reasonably well
- Performance consistent across most classes

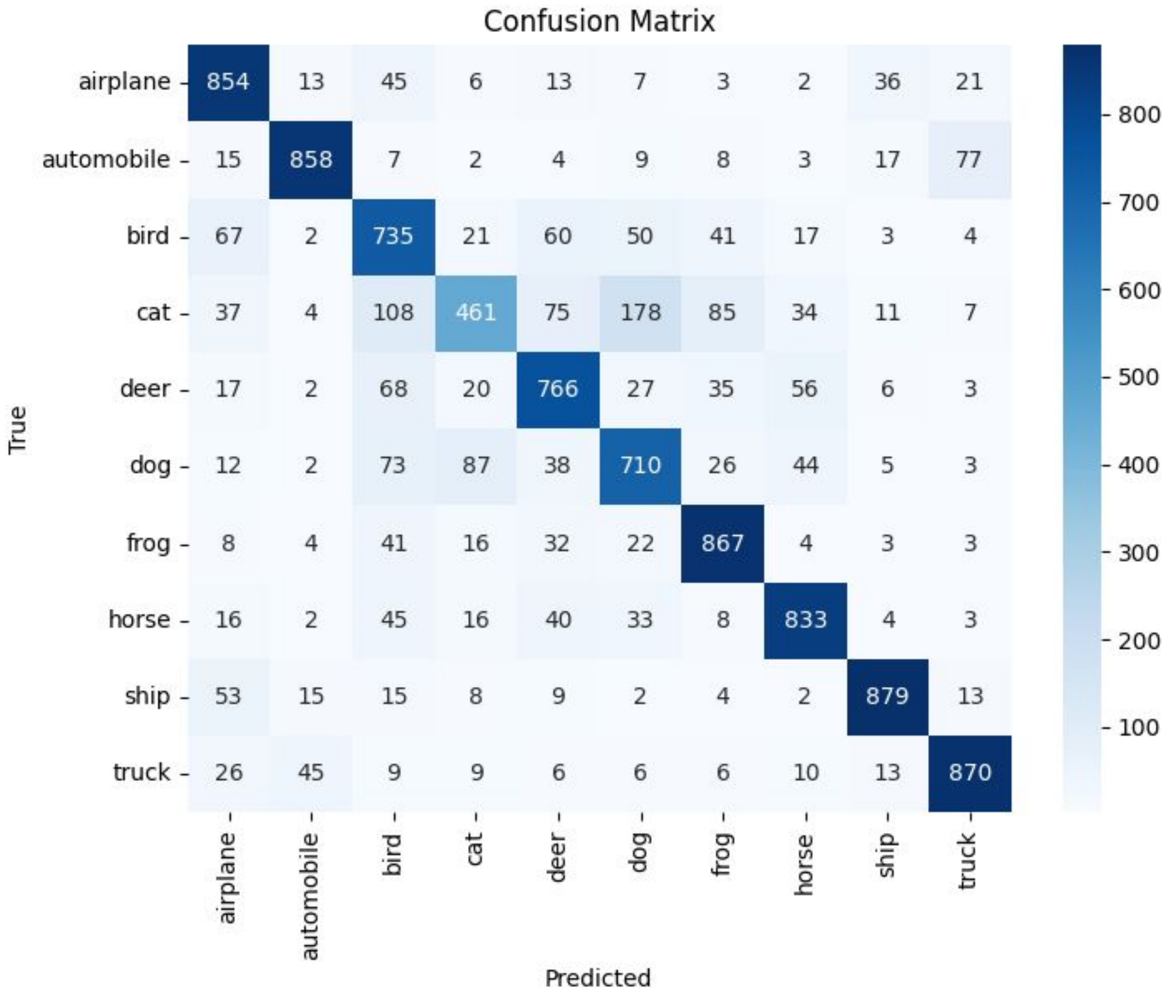
Classification Report:

	precision	recall	f1-score	support
airplane	0.77	0.85	0.81	1000
automobile	0.91	0.86	0.88	1000
bird	0.64	0.73	0.68	1000
cat	0.71	0.46	0.56	1000
deer	0.73	0.77	0.75	1000
dog	0.68	0.71	0.69	1000
frog	0.80	0.87	0.83	1000
horse	0.83	0.83	0.83	1000
ship	0.90	0.88	0.89	1000
truck	0.87	0.87	0.87	1000
accuracy			0.78	10000
macro avg	0.78	0.78	0.78	10000
weighted avg	0.78	0.78	0.78	10000


Test Metrics
Score
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
F1-Score
Precision
Recall

Confusion Matrix Insights

- Common confusions:
  - Cat ↔ Dog
  - Automobile ↔ Truck
  - Bird ↔ Airplane
- Diagonal dominance shows overall effective classification
- Useful for understanding misclassified patterns



Confusion Matrix
True
Predicted
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
854 13 45 6 13 7 3 2 36 21
15 858 7 2 4 9 8 3 17 77
67 2 735 21 60 50 41 17 3 4
37 4 108 461 75 178 85 34 11 7
17 2 68 20 766 27 35 56 6 3
12 2 73 87 38 710 26 44 5 3
8 4 41 16 32 22 867 4 3 3
16 2 45 16 40 33 8 833 4 3
53 15 15 8 9 2 4 2 879 13
26 45 9 9 6 6 6 10 13 870
800
700
600
500
400
300
200
100

Discussion

Architectural decisions mattered more than optimizer tweaks

Excessive pooling was initially a major issue

BatchNorm essential for convergence

Larger models are not always better — must balance depth & spatial resolution

Limitations

Limited exploration of filter sizes & kernel sizes

Training time increased significantly as model deepened

Only one dataset explored

No data augmentation used (could improve accuracy)

Lessons Learned

CNN depth must be balanced with spatial retention

Training curves help determine optimal epoch count

Hyperparameter tuning is iterative and time-consuming

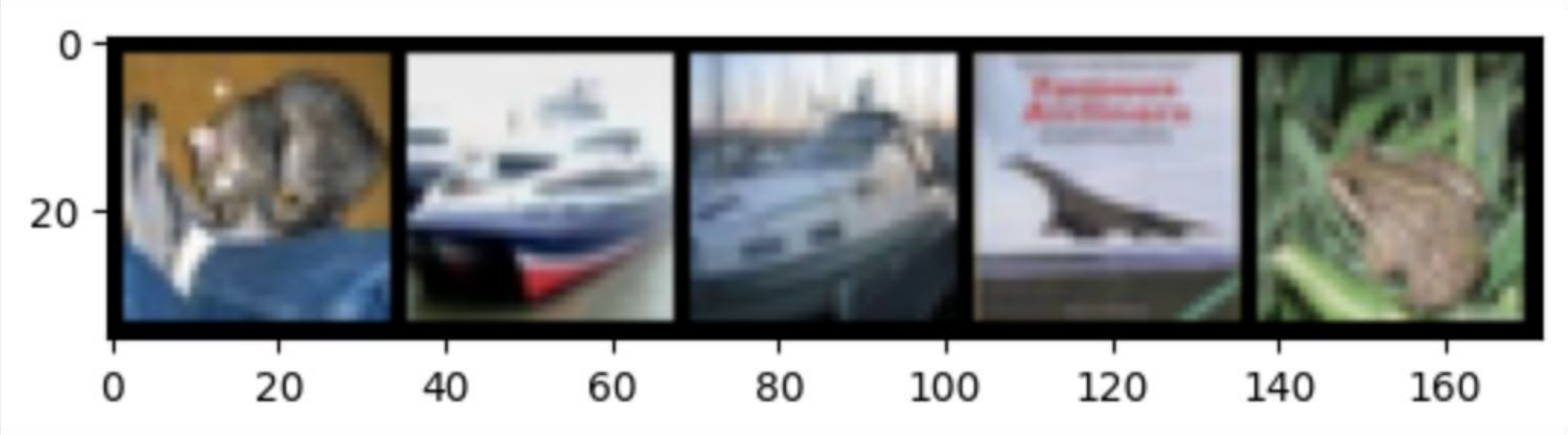
Understanding architecture principles is more important than copying models

Future Work

Try:

- Data augmentation
- Dropout or weight decay
- Different kernel sizes
- Automated hyperparameter optimization
- Advanced CNNs like ResNet or DenseNet

Evaluate on CIFAR-100




```
GroundTruth:  cat   ship  ship  airplane frog
Predicted:  cat   ship  cat   airplane frog
```

Conclusion

Built and trained a deeper CNN from scratch

Improved from **60% → 78%** test accuracy

Demonstrated full DL workflow: dataset prep → modeling → tuning → evaluation

Learned practical insights into CNN design and hyperparameter effects

Thank You