CIFAR-10 Classification with CNNs

JOSHUA LABRANCHE
JUNGHWAN KIM
JONATHAN LAPHAM

Outline

- **Introduction**
- Dataset
- Architecture
- Results
- Conclusions

Introduction

- Constructed original CNN using inspiration form AlexNet, VGG, and ResNet architectures
- Architecture design focused on improving a baseline model with regularization and normalization methods.
- Computational cost was a factor in architecture design, as portability was a desired characteristic
- Lowest error rate achieved: 13.2%

What Did We Do?

Outline

- **Introduction**
- Dataset
- Architecture
- Results
- Conclusions

Dataset

- Architecture was trained and tested using the CIFAR-10 dataset

- Dataset consists of 60,000 32x32x3 images categorized into 10 categories:
  - airplanes
  - cars
  - birds
  - cats
  - deer
  - dogs
  - frogs
  - horses
  - ships
  - trucks.

Part of the larger “80 Million Tine Images” data set found at:

http://groups.csail.mit.edu/vision/TinyImages/

A. Krizhevsky, “Learning multiple layers of features from tiny images,” tech. rep., 2009.

Outline

- **Introduction**
- Dataset
- Architecture
- Results
- Conclusions

Architectures

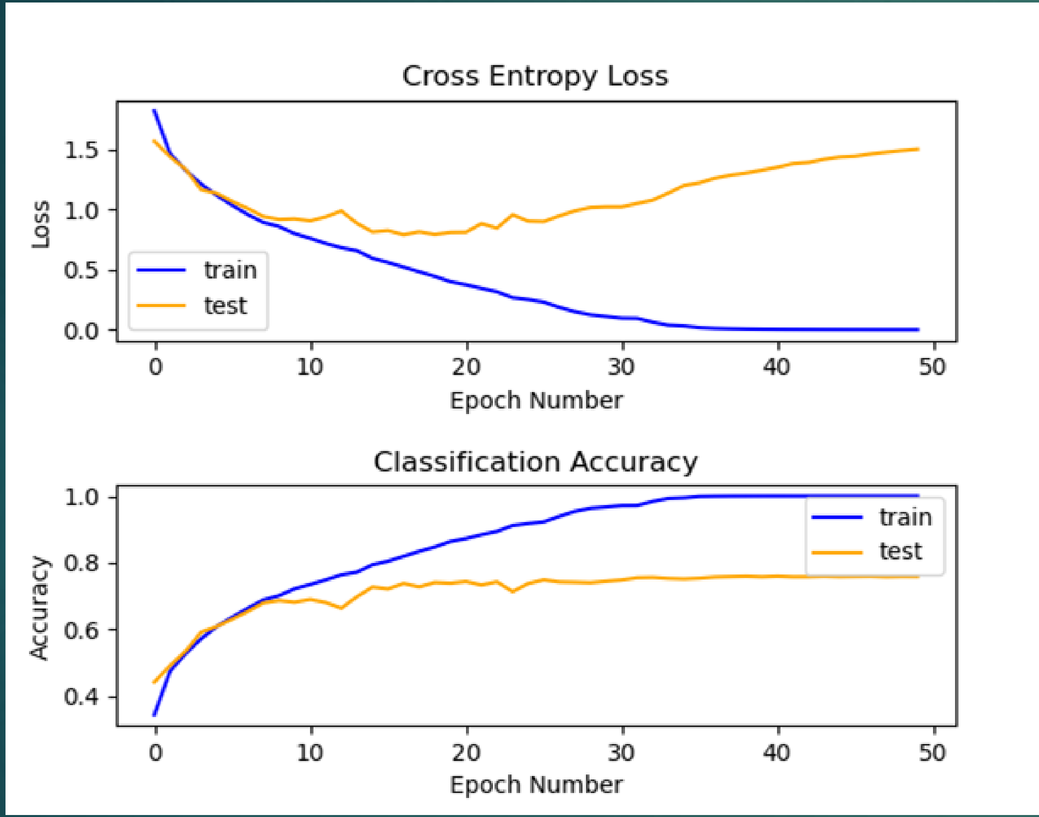
Untuned Model A	Untuned Model B	Untuned Model C	Untuned Model D
2D Conv (256, (3x3)) 2D Conv (256, (3x3)) 2D Conv (256, (3x3)) MaxPool((2x2), stride=2)	2D Conv (64, (3x3)) 2D Conv (64, (3x3)) 2D Conv (64, (3x3)) MaxPool((2x2), stride=2)	2D Conv (256, (3x3)) 2D Conv (256, (3x3)) 2D Conv (256, (3x3)) MaxPool((2x2), stride=2)	2D Conv (256, (3x3)) 2D Conv (256, (3x3)) MaxPool((2x2), stride=2)
2D Conv (128, (3x3)) 2D Conv (128, (3x3)) MaxPool((2x2), stride=2)	2D Conv (64, (3x3)) 2D Conv (64, (3x3)) MaxPool((2x2), stride=1)	2D Conv (256, (3x3)) 2D Conv (256, (3x3)) MaxPool((2x2), stride=2)	2D Conv (256, (3x3)) MaxPool((2x2), stride=1)
2D Conv (64, (3x3)) 2D Conv (64, (3x3)) MaxPool((2x2), stride=2)	2D Conv (32, (3x3)) MaxPool((2x2), stride=2)	2D Conv (256, (1x1)) MaxPool((2x2), stride=1)	2D Conv (256, (1x1)) MaxPool((2x2), stride=1)
NO LAYER	2D Conv (64, (3x3)) 2D Conv (64, (3x3)) MaxPool((2x2), stride=1)	2D Conv (64, (3x3)) MaxPool((2x2), stride=1)	2D Conv (64, (3x3)) MaxPool((2x2), stride=1)
NO LAYER	NO LAYER	2D Conv (256, (1x1)) MaxPool((2x2), stride=2)	2D Conv (256, (1x1)) MaxPool((2x2), stride=2)
Flatten()	Flatten()	Flatten()	Flatten()
Dense(512)	Dense(512)	Dense(512)	Dense(512)
Dense(512)	Dense(512)	Dense(512)	Dense(512)
Dense(10)	Dense(10)	Dense(10)	Dense(10)

Outline

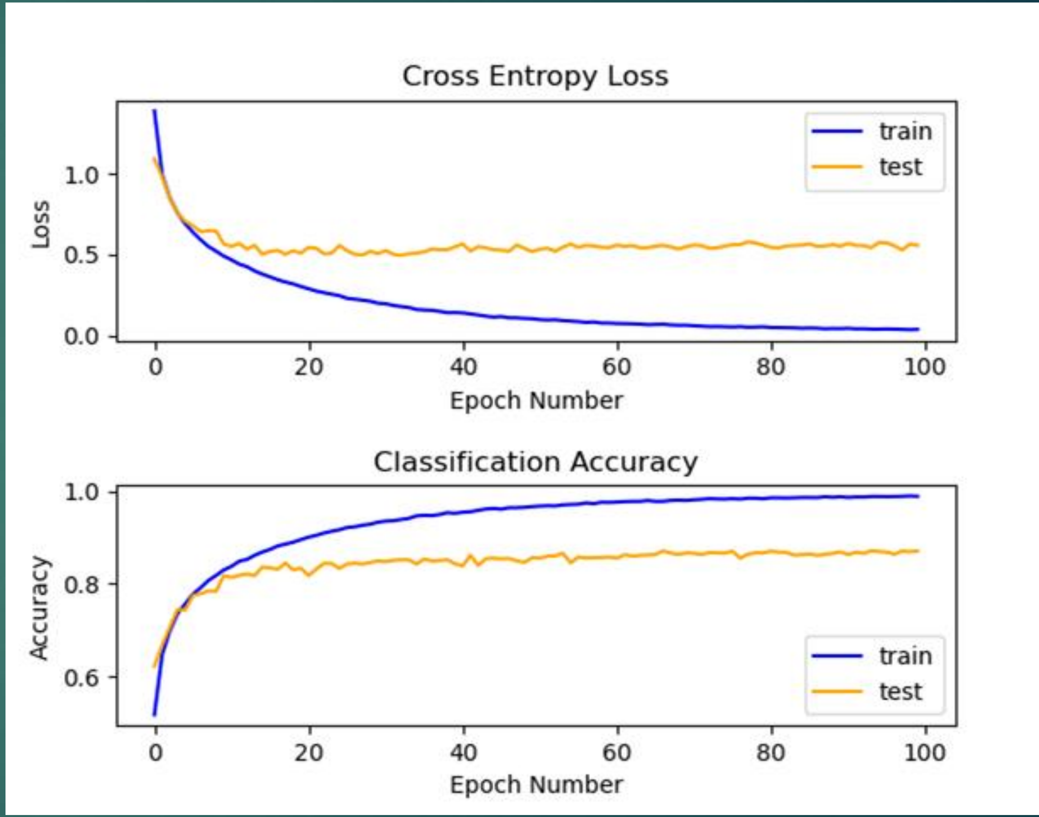
- **Introduction**
- Dataset
- Architecture
- Results
- Conclusions

Results

Baseline Model D



Model D with Data Augmentation, Batch Normalization, and a Reduced Batch Size (64)



Error Rate Minimum: 13.2%

Outline

- **Introduction**
- Dataset
- Architecture
- Results
- Conclusions

Conclusion

- Model D was the best performing of our baseline models.

- We tried to improve performance by adding techniques such as weight decay, drop-out, scaling drop-out, data augmentation, bottlenecking, and batch normalization.

- The best measured performance was the baseline model with the addition of data augmentation and batch normalization, along with a reduced batch size

- Performing only slightly worse was Model D with the addition of data augmentation and constant drop-out with an error rate of 14.8%

Data Augmentation and Batch Normalization!