Convolutional Neural Networks

For

Bird Classification

Taken by Zach Machcek:
https://unsplash.com/@zmachacek

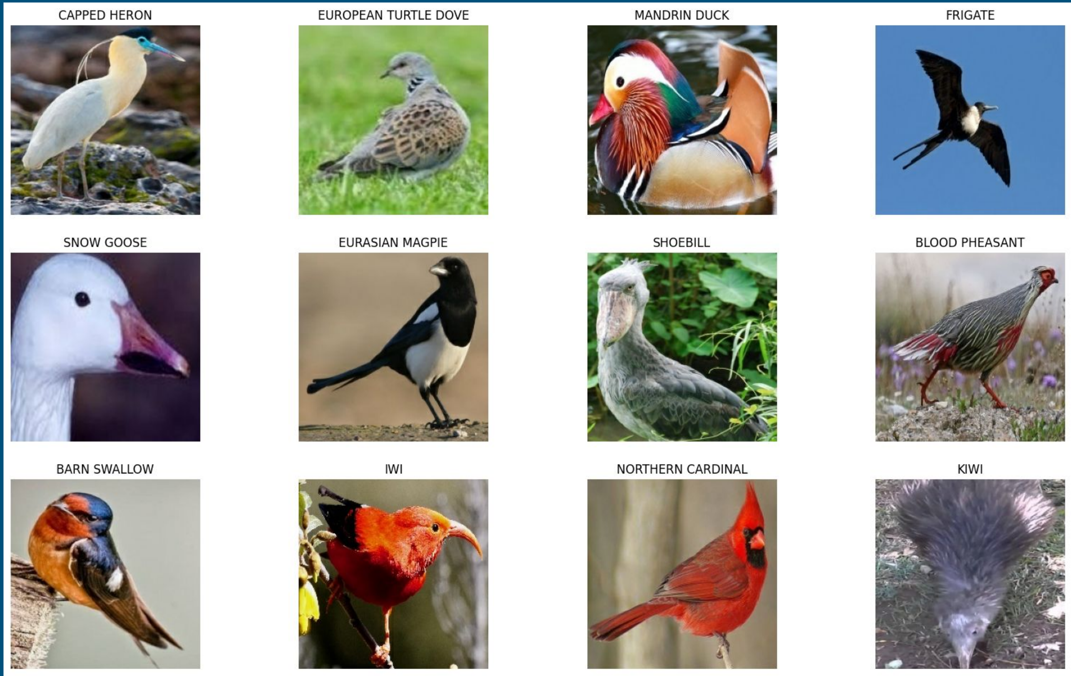
Overall Approach

- Dataset is very large
  - Large number of samples
  - Large size of each sample
  - Large number of classes
- Not practical to learn and explore different models and implementations using the entire dataset

- My approach:
  - Explore and learn on subset of the data
  - Implement “best” model on full dataset
- Goals:
  - Learn and gain confidence in using and understanding CNNs
  - Strengthen understanding of classification tasks
  - Develop an end to end understanding of a somewhat daunting task

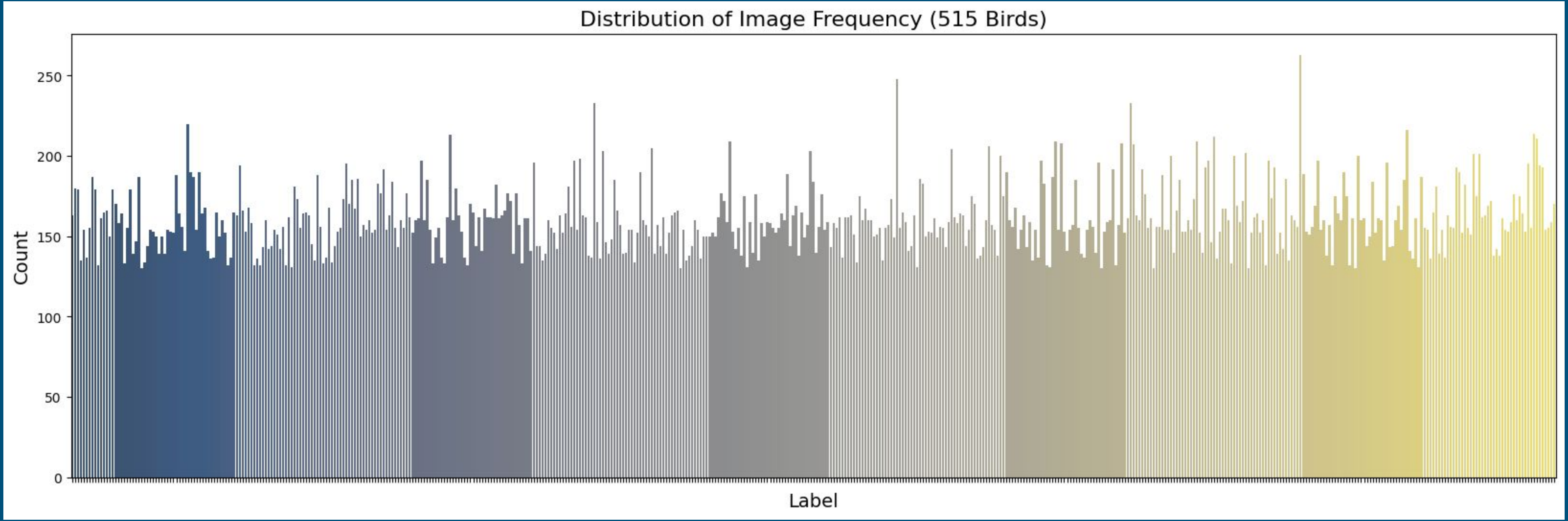
The Dataset: The Ultimate Goal

- 515 bird species/classes
- 82,724 training samples
- 2,586 batches of batch size 32
- Each image of size 224 x 224
- 3rd order tensor (3D array) of 8 bit ints
- 2,575 test images
- 2,575 validation images (5 images per species)



Distribution of Data for Full Dataset by Alphabetical Order

Average Image Count: 160



Most Common: Rufous Treepie
Image Count: 263

Least Common: Red Tailed Thrush
Image Count: 130

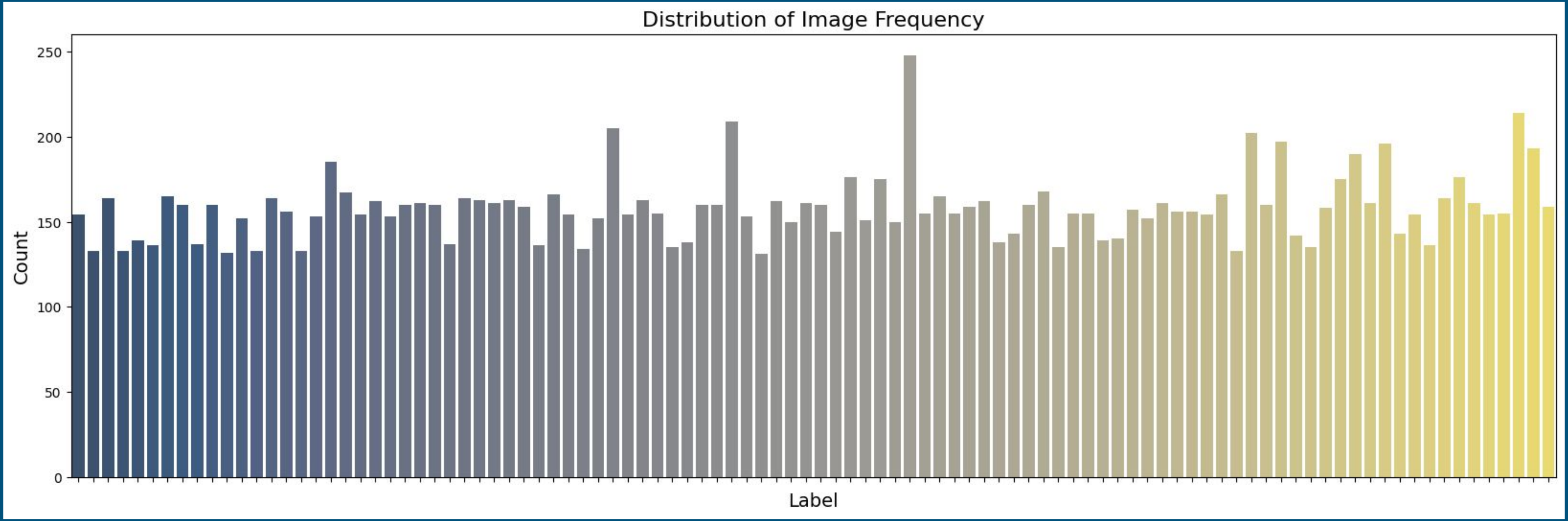
The Dataset: Trimmed

- 100 bird species/classes
- 15,824 training samples
- 495 batches of batch size 32
- Each image of size 224 x 224
- 3rd order tensor (3D array) of 8 bit ints
- 500 test and validation images (5 images per species)

Distribution of data: See Appendix 2

Distribution of Data Frequency by Alphabetical Order - Subset

Average Image Count: 158



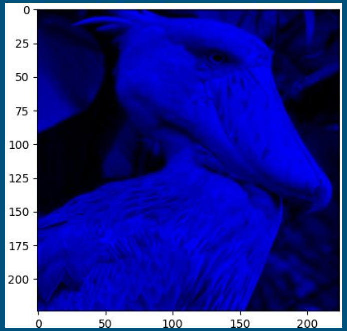
Most Common: House Finch
Image Count: 248

Least Common: Go Away Bird
Image Count: 131

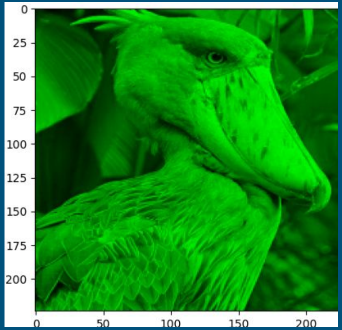
The Images/Sample

RGB

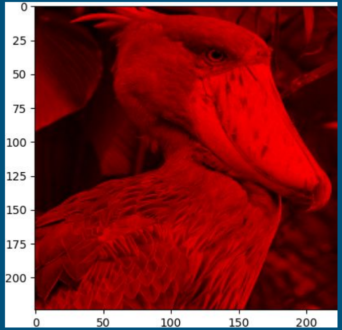
B



G



R



CNN for Image Classification

1. Convolution Layer
2. Pooling Layer
3. Fully Connected Layer

Some other layers:

- Batch Normalization
- Dropout
- Flattening

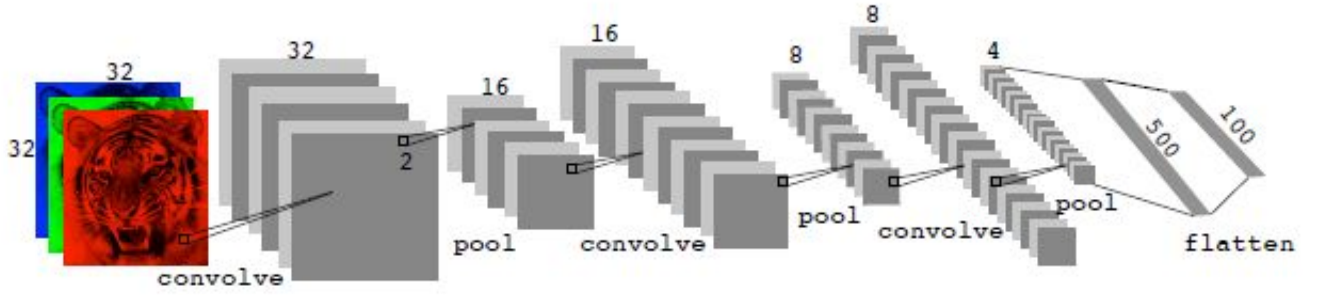
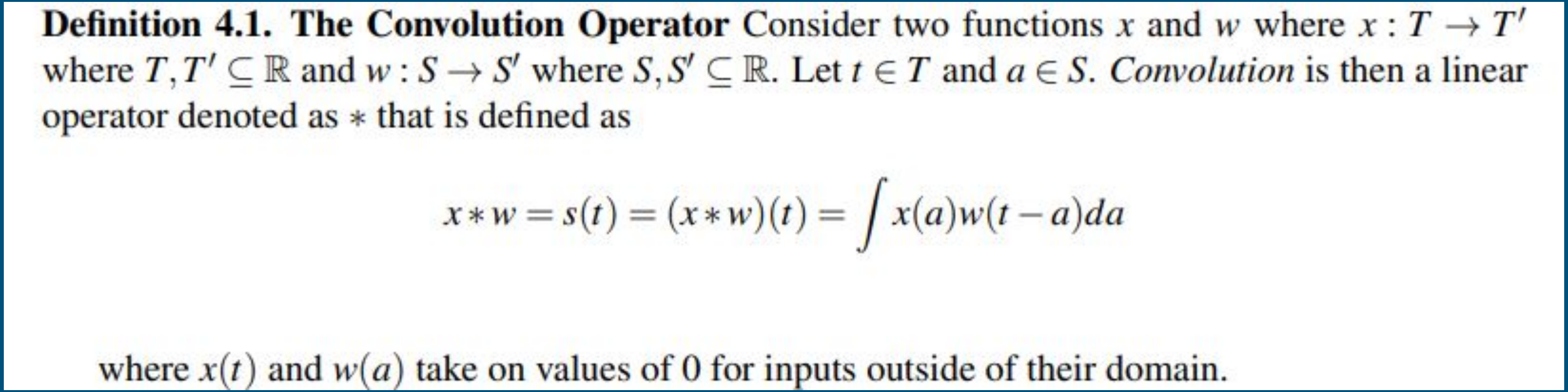


FIGURE 10.8. *Architecture of a deep CNN for the CIFAR100 classification task. Convolution layers are interspersed with 2 × 2 max-pool layers, which reduce the size by a factor of 2 in both dimensions.*

- Image Source: James, G., Witten, D., et al.. (2022). *An introduction to statistical learning: With applications in R*. Springer.

"Neural Networks are like onions"

Convolution



Definition 4.1. The Convolution Operator Consider two functions x and w where $x : T \to T'$ where $T, T' \subseteq \mathbb{R}$ and $w : S \to S'$ where $S, S' \subseteq \mathbb{R}$. Let $t \in T$ and $a \in S$. *Convolution* is then a linear operator denoted as $*$ that is defined as

$$x * w = s(t) = (x * w)(t) = \int x(a)w(t-a)da$$

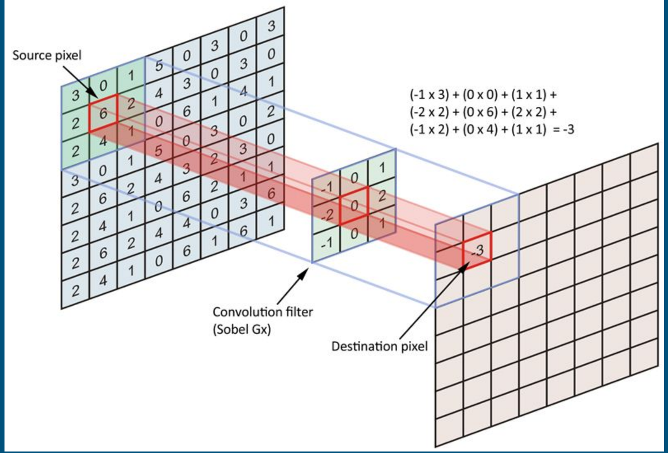
where $x(t)$ and $w(a)$ take on values of 0 for inputs outside of their domain.

The convolution operator essentially flips w around the “y-axis” at point *a* and sums the products evaluated at each point from left to right.

We call x the input and w the kernel.
In our convolutional neural network, the data is the input and the kernel is our filter.

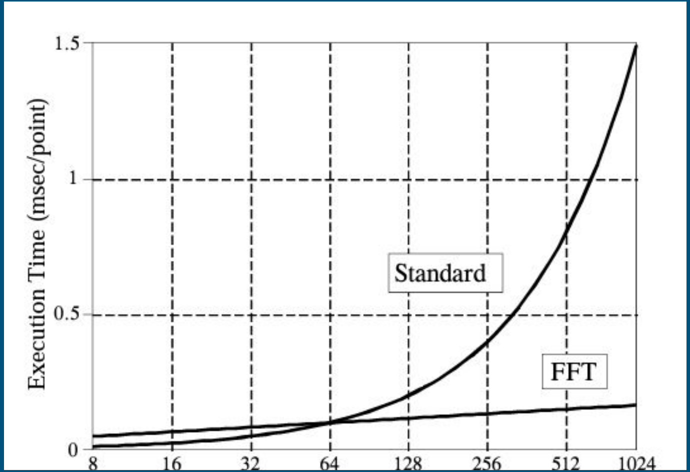
Implementing Convolution

- Our kernel is a sub-matrix of our input matrix consisting of weights
- Kernels form the “filter” which is essentially an entire layer of kernels
- Transforms the input into a convolved output
- Linear transformation. Therefore, we add a non-linear activation function onto the output of the convolution to add non-linearity - we use ReLU
- Sliding method - Center square of the filter ”slides” over the pixel values, transforms, and replaces center pixel
- DFT method - Calculate DFT utilizing FFT algorithm



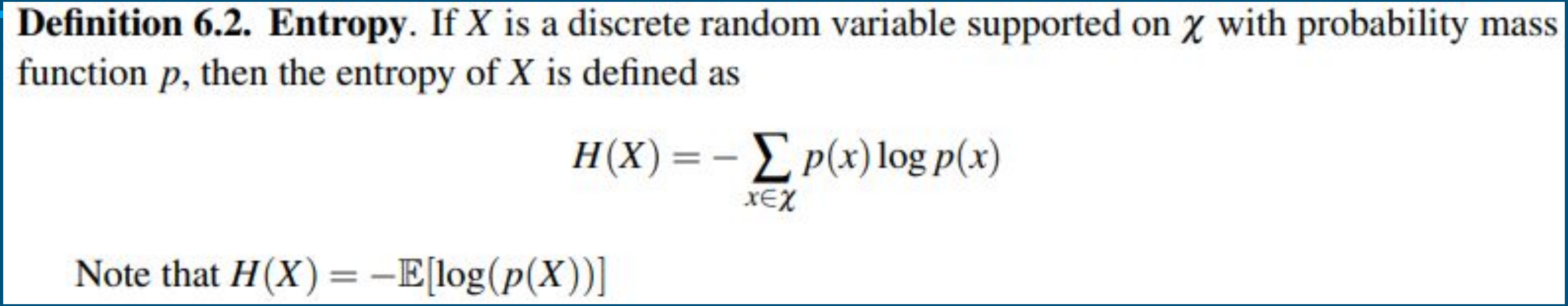
Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

https://datascience.stackexchange.com/questions/23183/why-convolutions-always-use-odd-numbers-as-filter-size



Smith, S. W. (1997). *The scientist and engineer's Guide to Digital Signal Processing*. California Technical Pub.

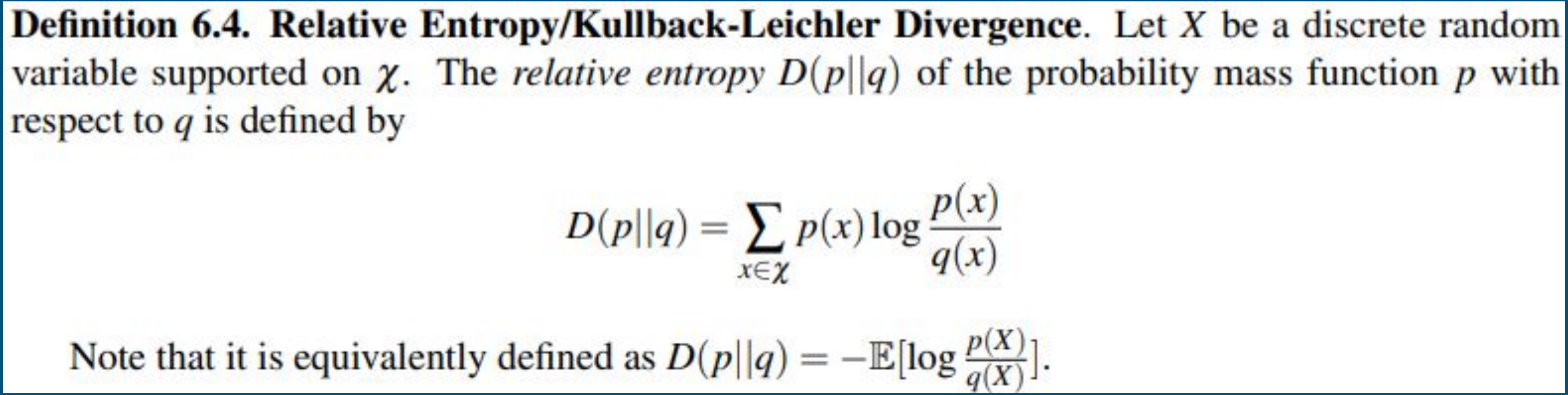
Fitting our CNN - Some Information Theory



Definition 6.2. Entropy. If X is a discrete random variable supported on χ with probability mass function p, then the entropy of X is defined as

$$H(X) = -\sum_{x \in \chi} p(x) \log p(x)$$

Note that $H(X) = -\mathbb{E}[\log(p(X))]$



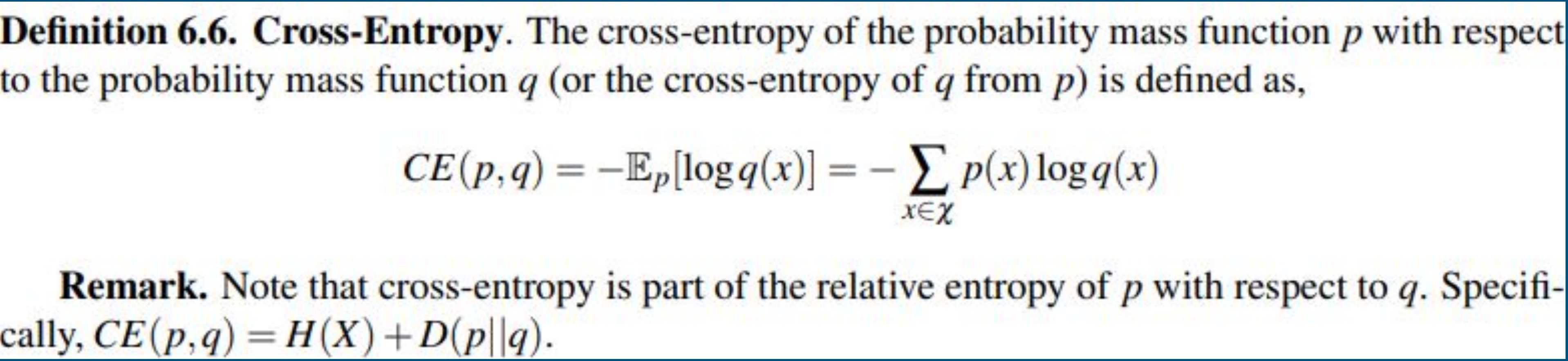
Definition 6.4. Relative Entropy/Kullback-Leichler Divergence. Let X be a discrete random variable supported on χ. The *relative entropy* $D(p||q)$ of the probability mass function p with respect to q is defined by

$$D(p||q) = \sum_{x \in \chi} p(x) \log \frac{p(x)}{q(x)}$$

Note that it is equivalently defined as $D(p||q) = -\mathbb{E}[\log \frac{p(X)}{q(X)}]$.

Fitting our CNN - Loss

- We are dealing with a multi-class classification task
- Need a way to denote a measure of “distance” between our predictions and the actual value
    - ➢ One Possible Answer: Cross Entropy



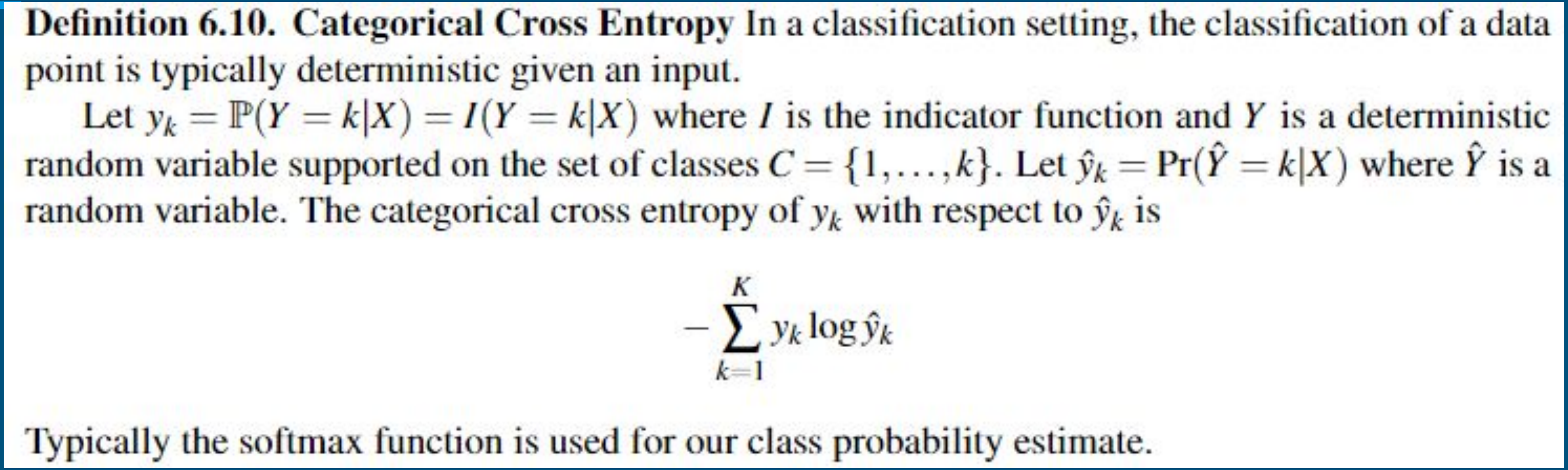
Definition 6.6. Cross-Entropy. The cross-entropy of the probability mass function p with respect to the probability mass function q (or the cross-entropy of q from p) is defined as,

$$CE(p,q) = -\mathbb{E}_p[\log q(x)] = -\sum_{x \in \chi} p(x) \log q(x)$$

Remark. Note that cross-entropy is part of the relative entropy of p with respect to q. Specifically, $CE(p,q) = H(X) + D(p||q)$.

Fitting our CNN - Loss and Cost

- The loss function we used is Cross Entropy
- The cost function is the sample mean of losses which acts as an estimator of loss

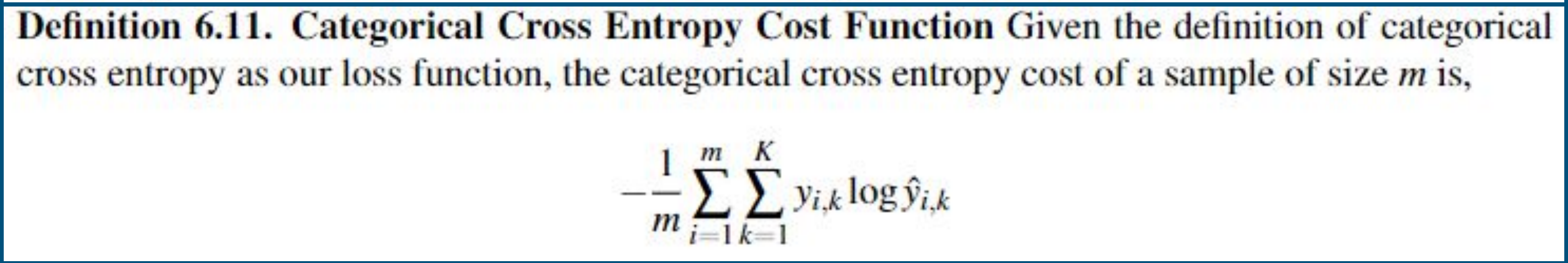


Definition 6.10. Categorical Cross Entropy In a classification setting, the classification of a data point is typically deterministic given an input.

Let $y_k = \mathbb{P}(Y = k|X) = I(Y = k|X)$ where I is the indicator function and Y is a deterministic random variable supported on the set of classes $C = \{1, \ldots, k\}$. Let $\hat{y}_k = \Pr(\hat{Y} = k|X)$ where $\hat{Y}$ is a random variable. The categorical cross entropy of y_k with respect to $\hat{y}_k$ is

$$-\sum_{k=1}^{K} y_k \log \hat{y}_k$$

Typically the softmax function is used for our class probability estimate.



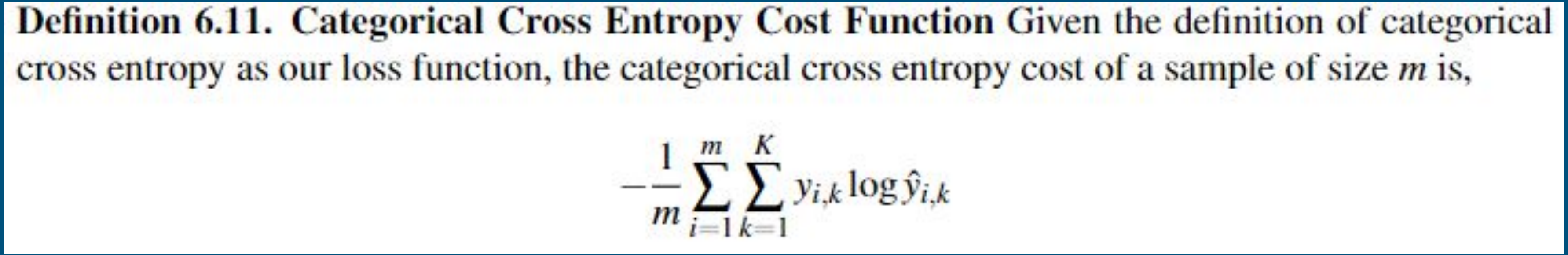
Definition 6.11. Categorical Cross Entropy Cost Function Given the definition of categorical cross entropy as our loss function, the categorical cross entropy cost of a sample of size m is,

$$-\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_{i,k} \log \hat{y}_{i,k}$$

Fitting our CNN

- This is an image multi-class classification task
    - Output - class probability vector
- Chose to use categorical cross entropy as loss function
- Output layer is softmax activation for each class
- Very similar ideas to multinomial logistic regression

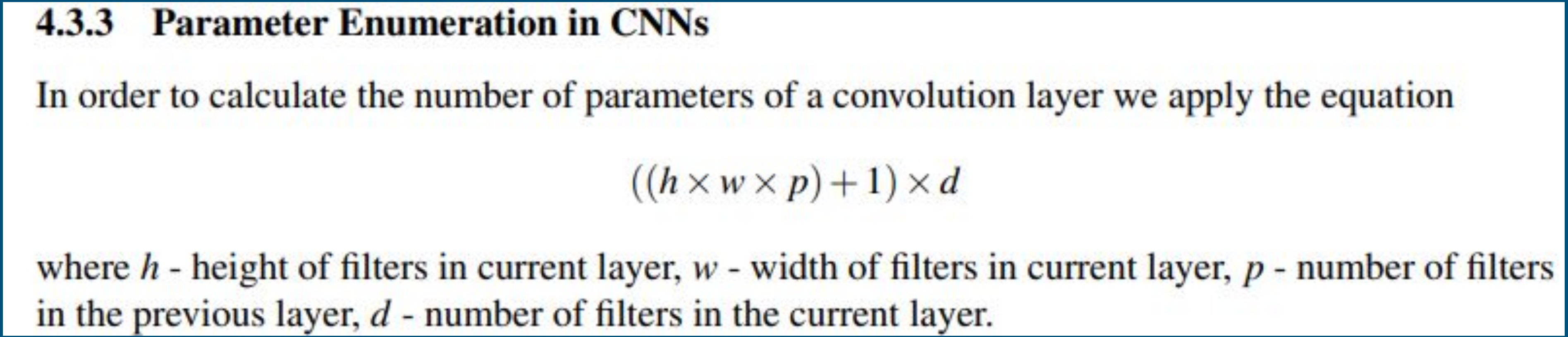
Our Cost Function we use is:



Definition 6.11. Categorical Cross Entropy Cost Function Given the definition of categorical cross entropy as our loss function, the categorical cross entropy cost of a sample of size m is,

$$-\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_{i,k}\log \hat{y}_{i,k}$$

Parameter Enumeration



4.3.3 Parameter Enumeration in CNNs

In order to calculate the number of parameters of a convolution layer we apply the equation

$$((h \times w \times p) + 1) \times d$$

where h - height of filters in current layer, w - width of filters in current layer, p - number of filters in the previous layer, d - number of filters in the current layer.

Optimization Algorithms

- Recall gradient descent as a method of optimization
- Stochastic gradient descent provides a feasible algorithm for performing gradient descent
  - Mini-batch gradient descent
- Improvements on SGD
  - Momentum
  - Nesterov Momentum
  - Adaptive Learning Rates

- Some examples of adaptive learning rate algorithms
  - AdaGrad
  - RMSprop
  - Adam
    - The adaptive method we use

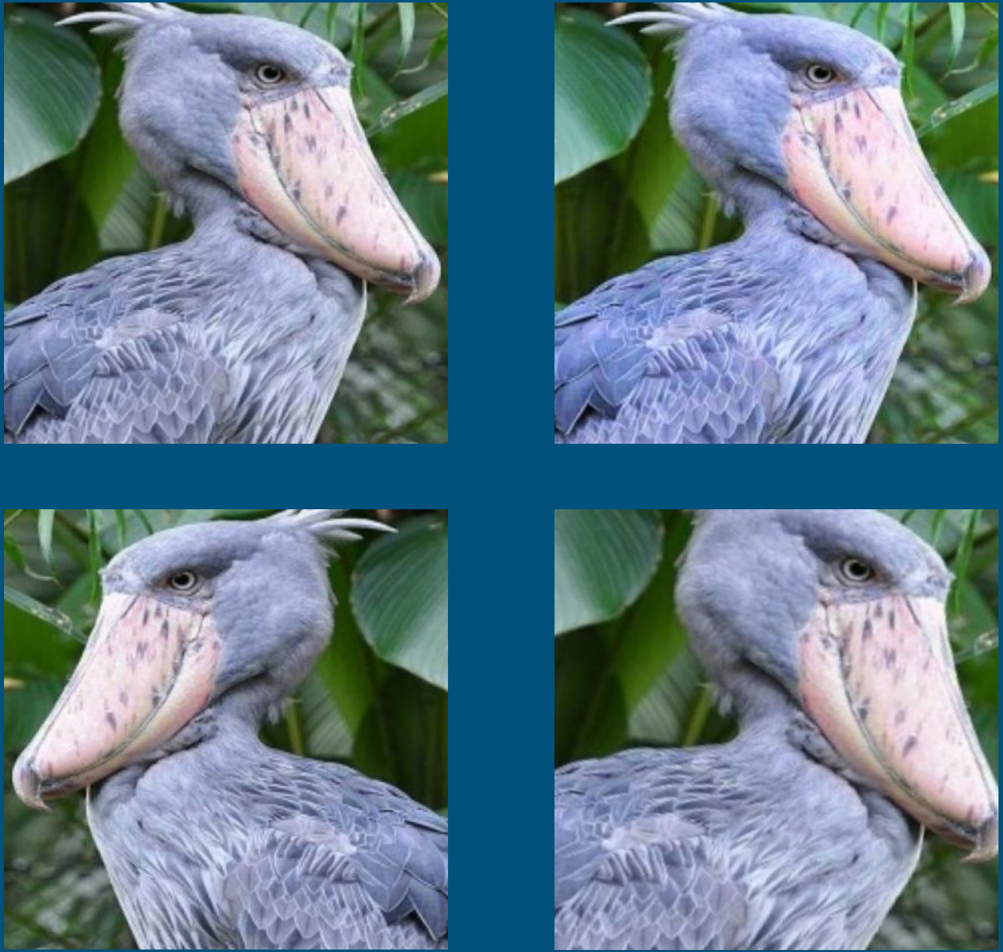
Automatic Differentiation

- Recall the backpropagation algorithm we discussed in class
- We worked through an example with MSE as our cost function. For our classification problem, Mean Cross Entropy loss is our cost function and our output function is the softmax activation function

- Backpropagation is generalized to tensors
- Backpropagation works essentially the same in CNNs as it does FFNs
    - Note: The parameters it tunes in convolution layers are the weights in filters

Regularization

- Dataset Augmentation
    - To the right is an example
        - Original (top left)
        - Increase Contrast (top right)
        - Flip horizontally (bottom left)
        - Rotate cw 10 degrees (bottom right)
- Dropout
- Early Stopping



Transfer Learning

- What is Transfer Learning?
- Benefits and Obstacles
  - Train less parameters
  - But: Your Data != Their Data
- The models we used for Transfer Learning are both trained on ImageNet
  - Pretrained VGG16
  - Pretrained ResNet50
- ImageNet - dataset of over 14 million images in 1000 classes

VGG16

- VGG16 is a 16 layer Convolutional Neural Network
- Output is shape (7, 7, 512)

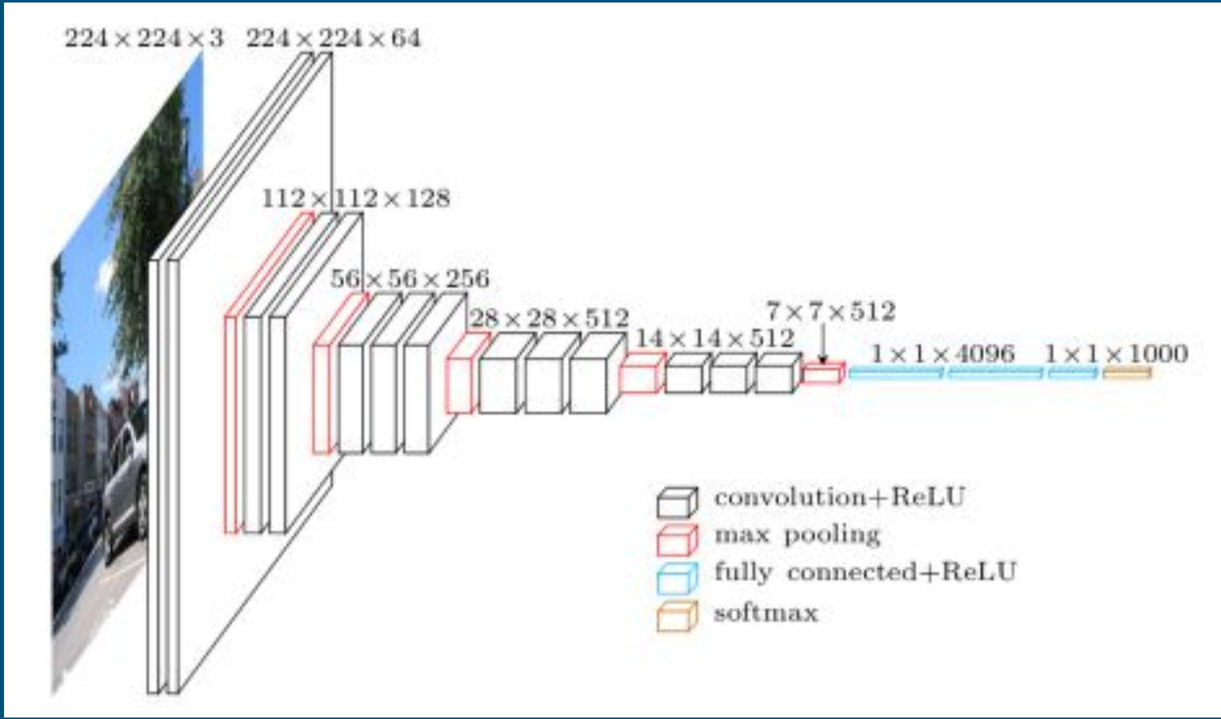


Image Source: https://neurohive.io/en/popular-networks/vgg16/

ResNet50

- 50 Layer Residual CNN
- Similar to ResNet34 but with more layers and instead of stacks of 2 layers, it consists of stacks of 3
- Output of shape (7, 7, 2048)

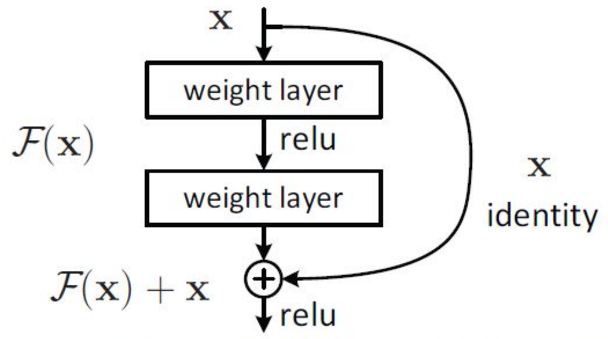

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block.

2 layer stack example of residual learning

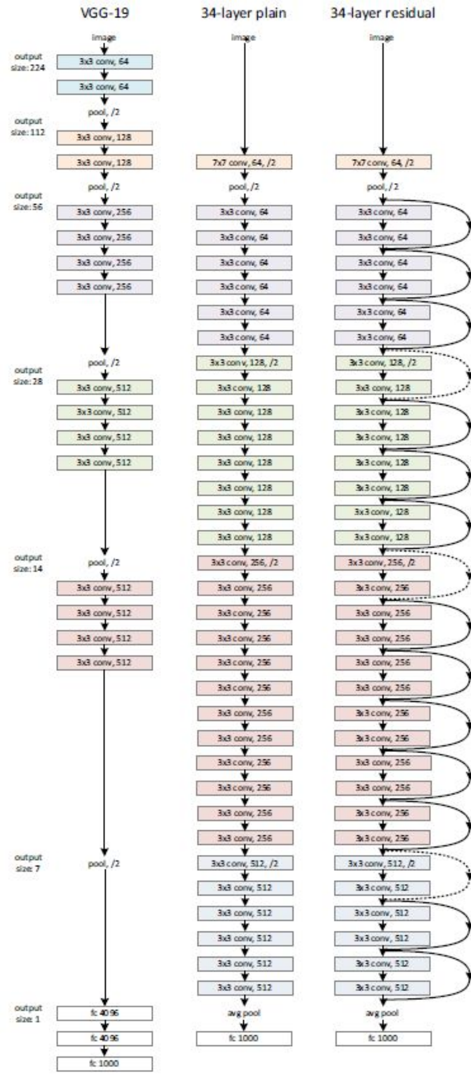


Image Citations:

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2016.90

About the Implementations

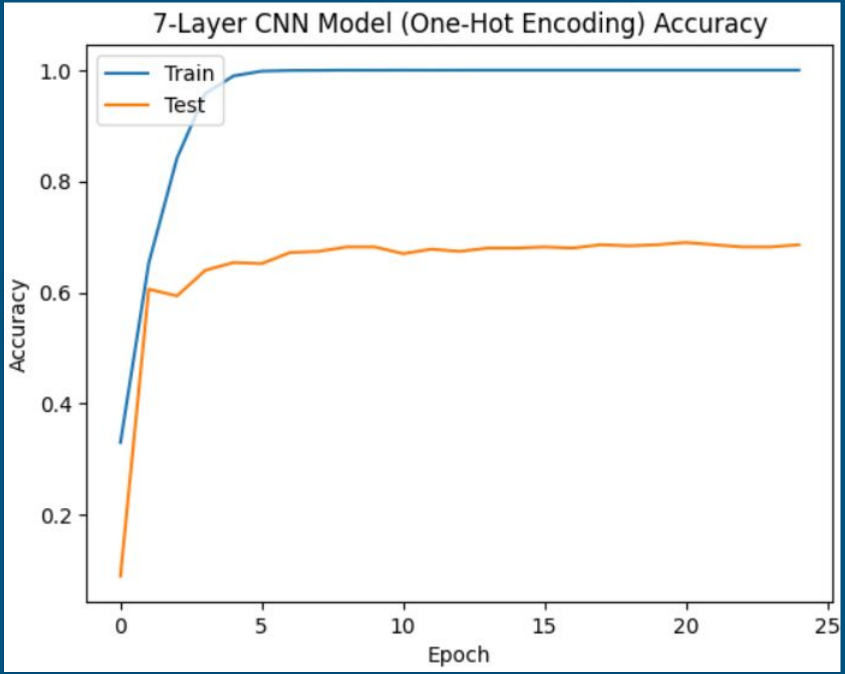
- Explored a 7-Layer architecture and the effects of data augmentation
- Compared SGD to Adam with fixed models and hyperparameters to explore optimizer effectiveness and speed
- Portrayed learning rate effects on the learning rate of a model
- Used an educated guess to build an implement final model and analyze results
- Utilized TensorFlow and the Keras library for processing data and building and training the models

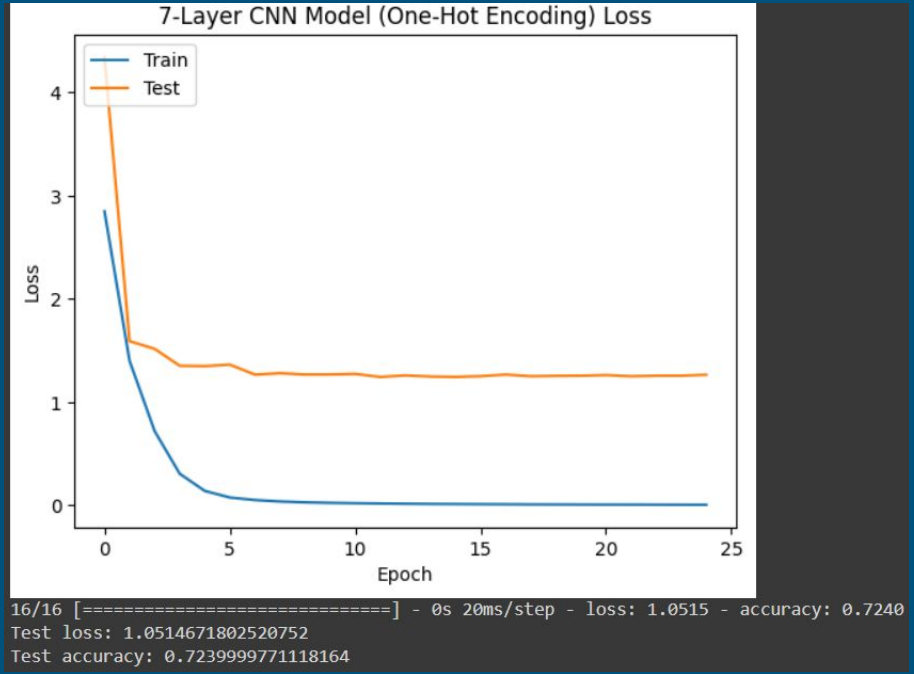
Note: All Transfer Learning use model pre-trained on ImageNet and add one fully connected layer of 512 nodes with ReLU activation function

7-Layer CNN

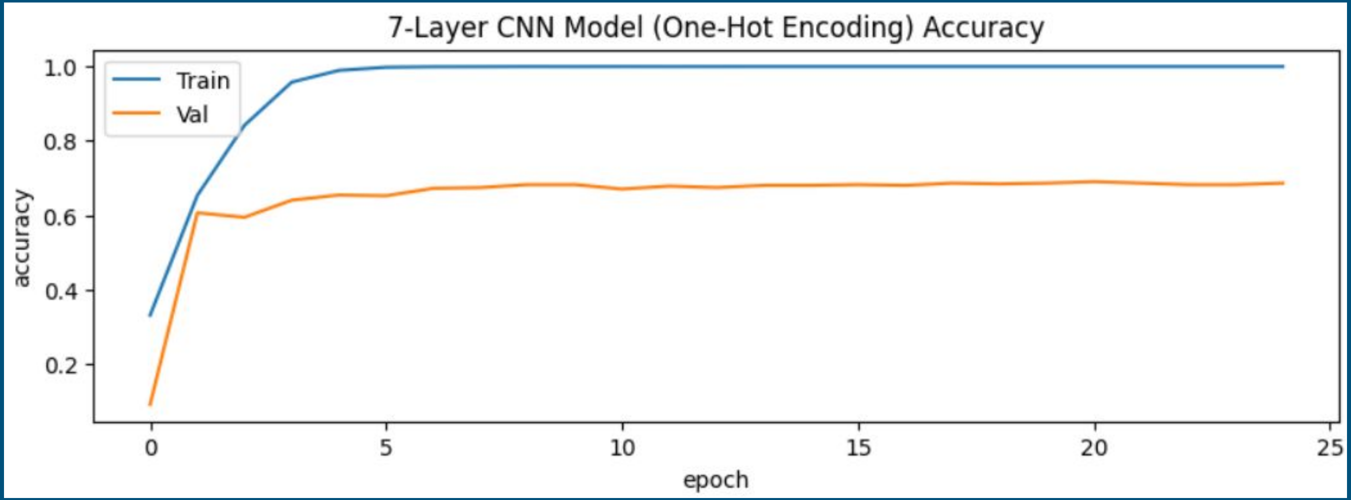
- 7-Layer CNN
- SGD with learning rate = 0.001
- Total Params: 95,754,468
- Loss and Accuracy Curves

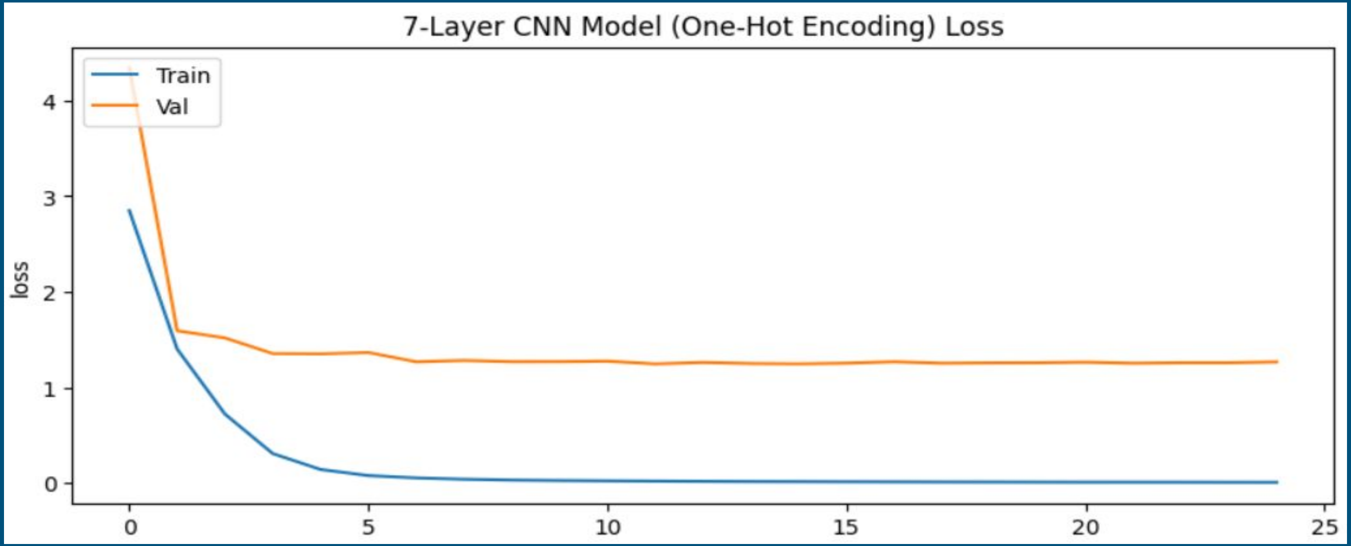
Test Accuracy ~ 72.4%
Test Loss ~ 1.052





```
16/16 [==============================] - 0s 20ms/step - loss: 1.0515 - accuracy: 0.7240
Test loss: 1.0514671802520752
Test accuracy: 0.7239999771118164
```

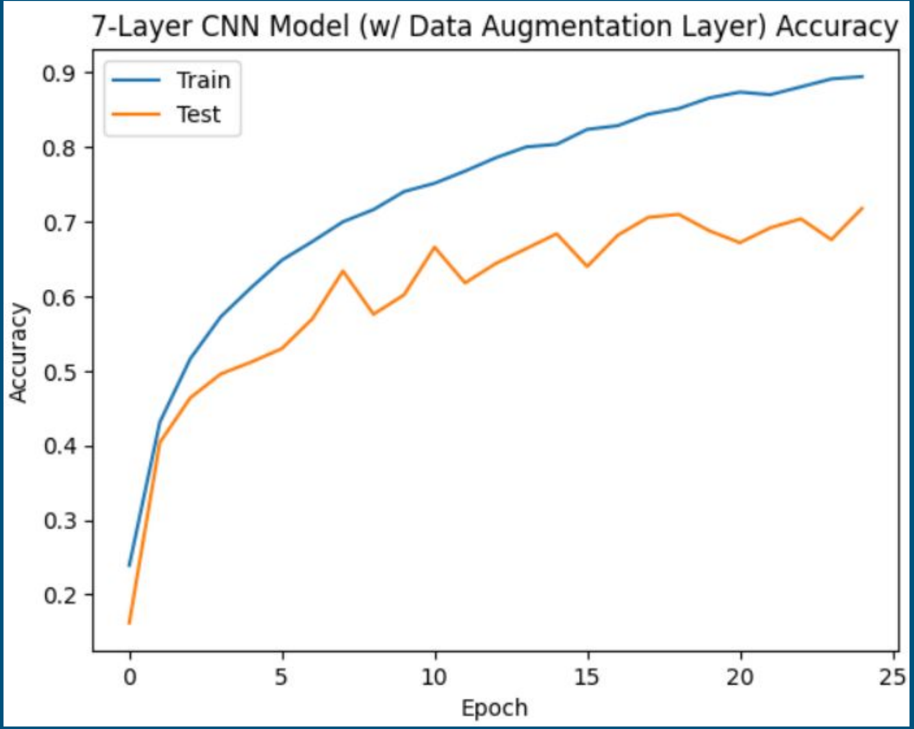

7-Layer CNN Model (One-Hot Encoding) Accuracy
Train
Val
accuracy
epoch
1.0
0.8
0.6
0.4
0.2
0
5
10
15
20
25

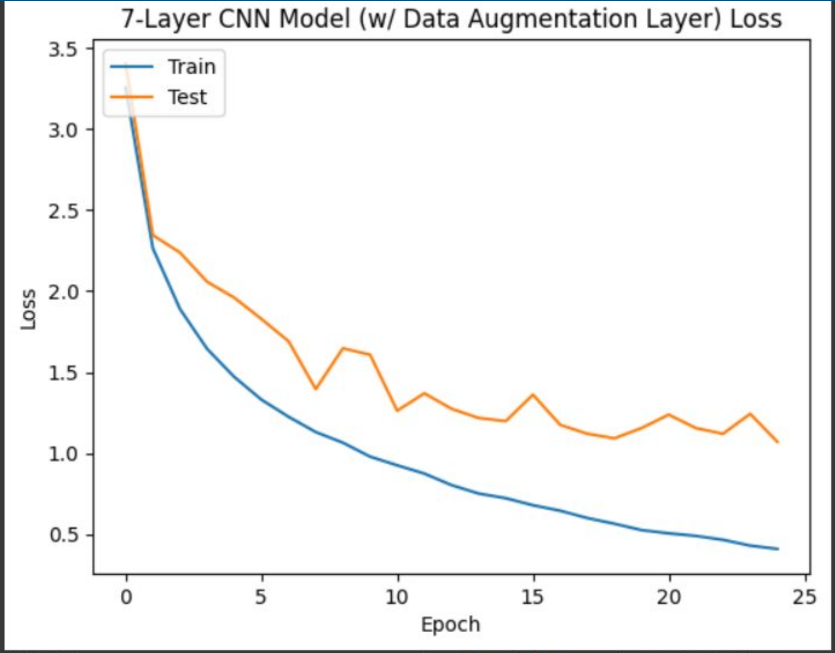

7-Layer CNN Model (One-Hot Encoding) Loss
Train
Val
loss
4
3
2
1
0
0
5
10
15
20
25

7-Layer CNN w/ Data Augmentation Layer

- 7-Layer CNN
- Added Data Augmentation Layer
- SGD with learning rate = 0.001
- Total Params: 95,754,468
- Loss and Accuracy Curves

Test Accuracy ~ 76.4%
Test Loss ~ 0.889



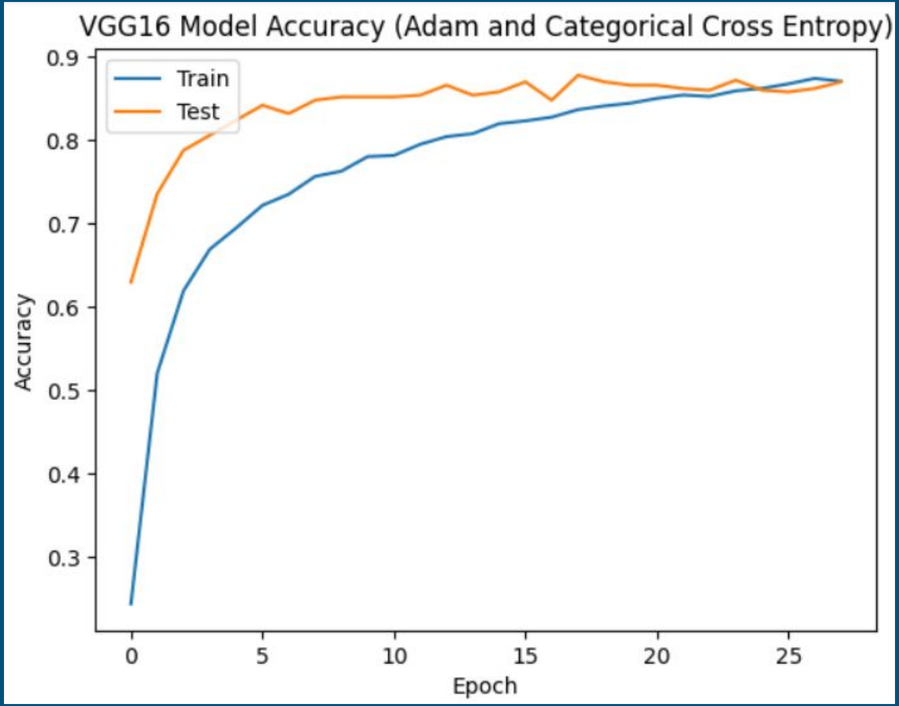


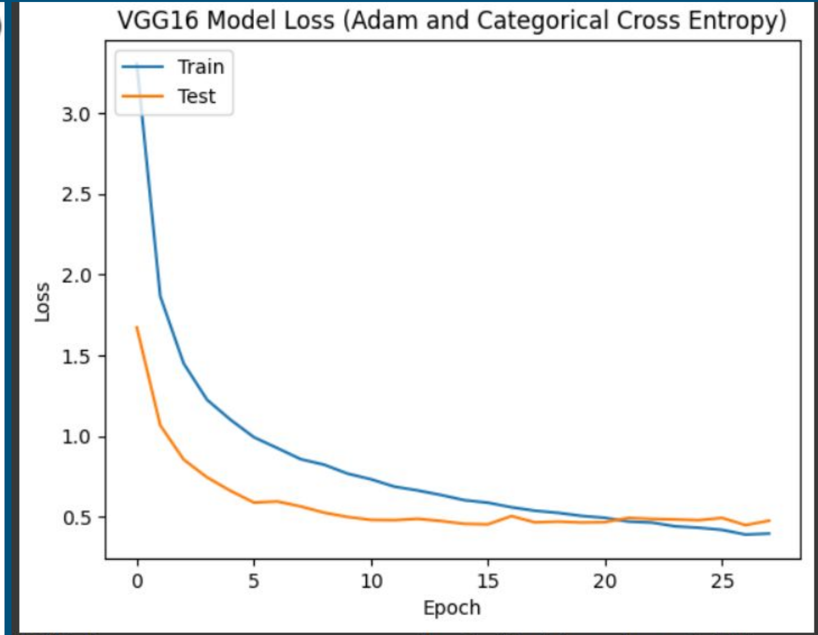
```
16/16 [==============================] - 31s 2s/step - loss: 0.8891 - accuracy: 0.7640
Test loss: 0.8891295194625854
Test accuracy: 0.7639999985694885
```

VGG16 Transfer Learning w/ Adam

- Transfer Learning, VGG16
- Adam with learning rate = 0.001
- Total Params: 15,028,644
- Trainable Params: 313,956
- Loss and Accuracy Curves

Test Accuracy ~ 86.2%
Test Loss ~ 0.425



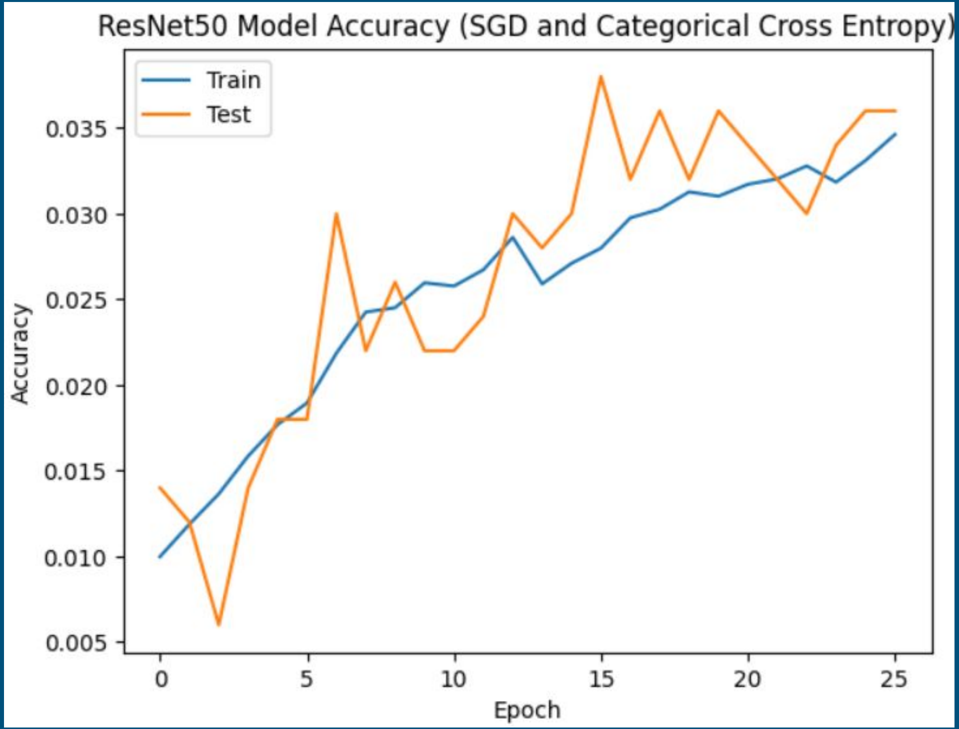


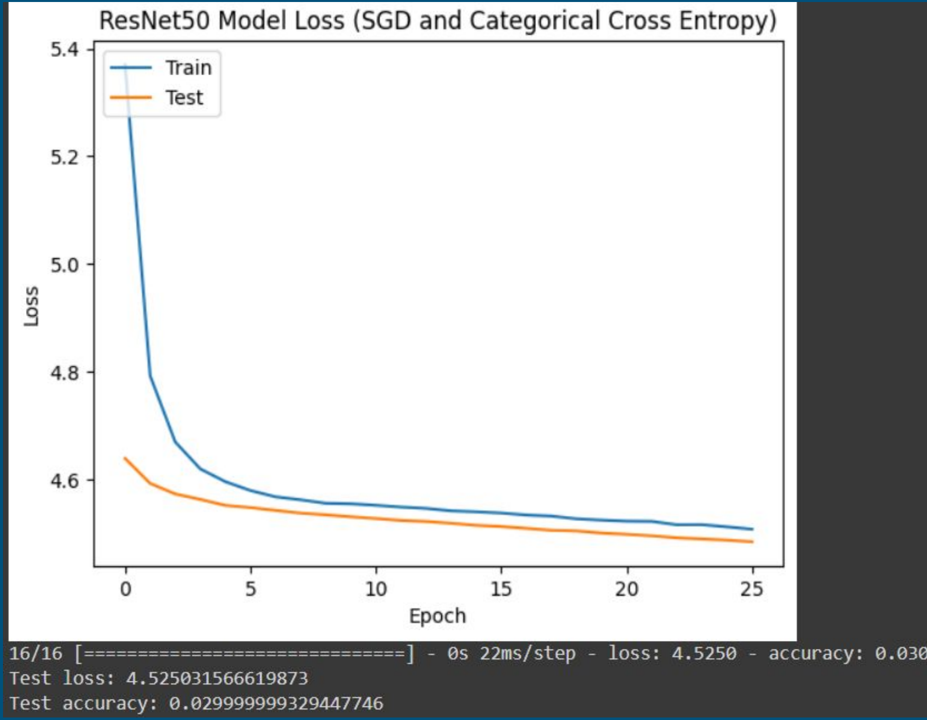
```
16/16 [==============================] - 0s 25ms/step - loss: 0.4249 - accuracy: 0.8620
Test loss: 0.4248928427696228
Test accuracy: 0.8619999885559082
```

ResNet50 Transfer Learning w/ SGD

- Transfer Learning, ResNet50
- SGD with learning rate = 0.0001
- Total Params: 24,688,100
- Trainable Params: 1,100,388
- Loss and Accuracy Curves

Test Accuracy ~ 3.0%
Test Loss ~ 4.525



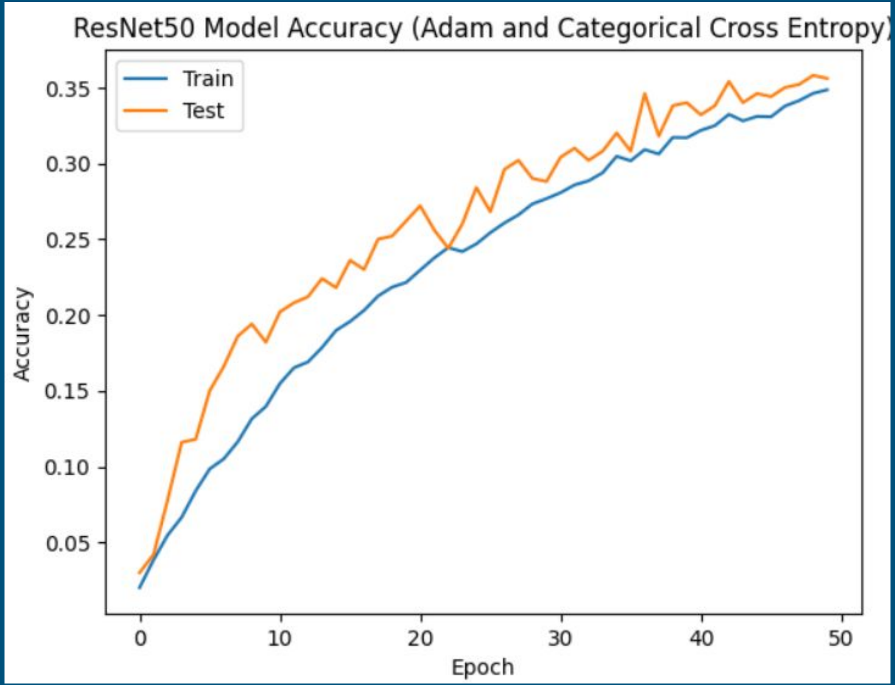


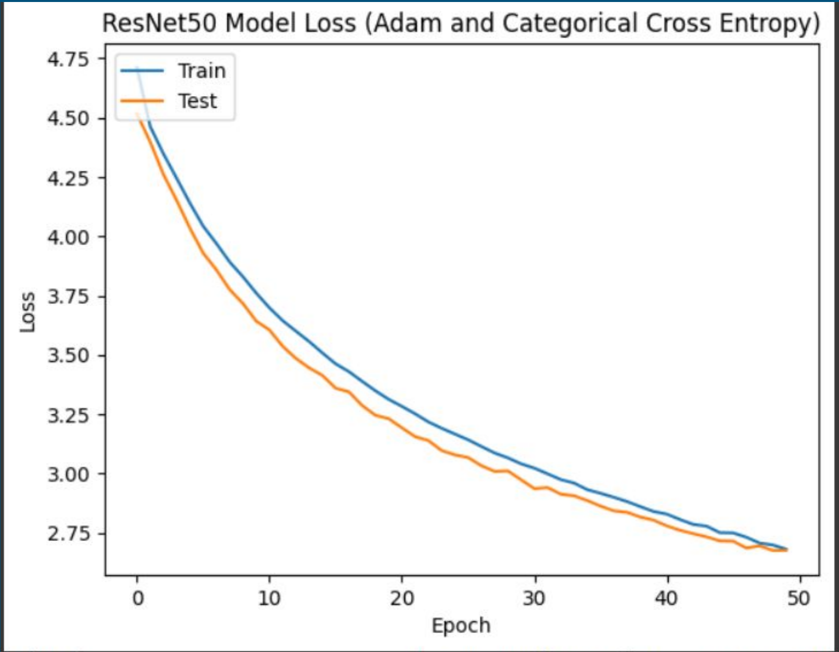
```
16/16 [==============================] - 0s 22ms/step - loss: 4.5250 - accuracy: 0.030
Test loss: 4.525031566619873
Test accuracy: 0.029999999329447746
```

ResNet50 Transfer Learning w/ Adam

- Transfer Learning, ResNet50
- Adam with learning rate = 0.0001
- Loss and Accuracy Curves

Test Accuracy ~ 38.8%
Test Loss ~ 2.521



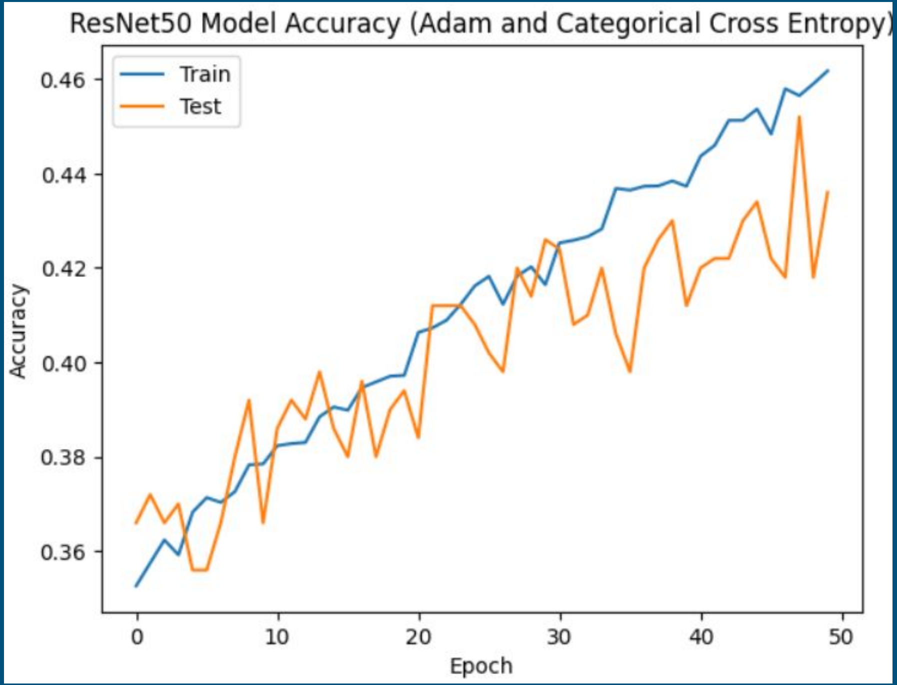


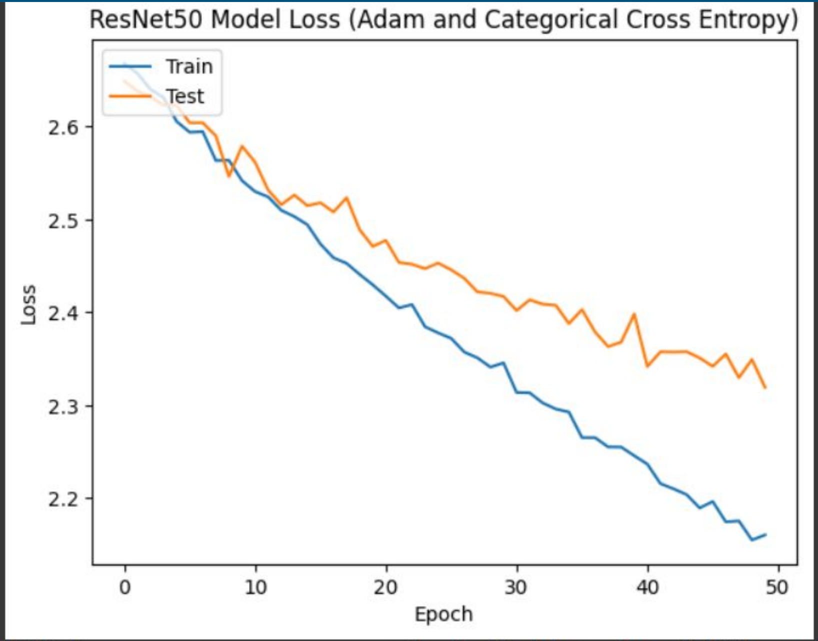
```
16/16 [==============================] - 0s 23ms/step - loss: 2.5213 - accuracy: 0.3880
Test loss: 2.5213265419006348
Test accuracy: 0.38800000114440918
```

ResNet50 Transfer Learning w/ Adam

- 50 more epochs
- Adam with learning rate = 0.0001

Test Accuracy ~ 43.4%
Test Loss ~ 2.158



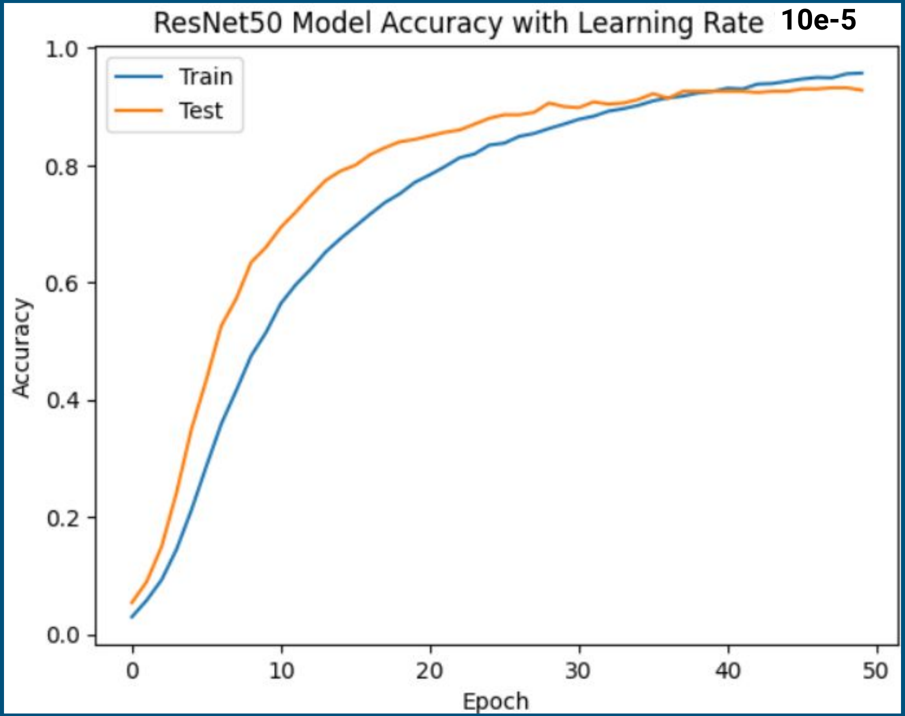


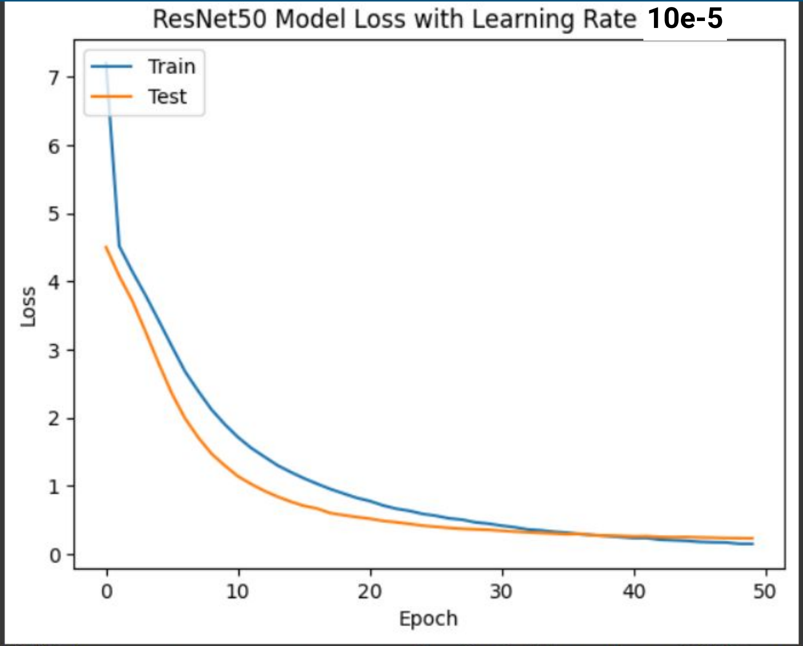
```
16/16 [==============================] - 0s 22ms/step - loss: 2.1581 - accuracy: 0.4400
Test loss: 2.158118724822998
Test accuracy: 0.4399999976158142
```

ResNet50 Transfer Learning w/ Adam

- Transfer Learning, ResNet50
- Adam with learning rate = 10e-5
- Loss and Accuracy Curves

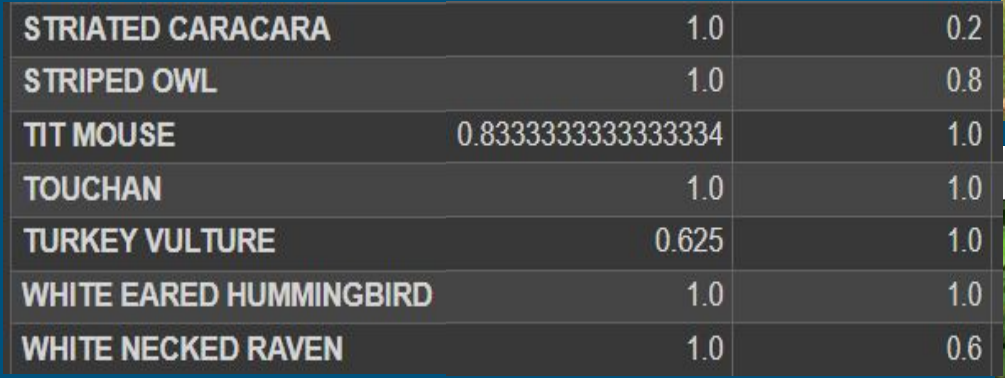
Test Accuracy ~ 95.6%
Test Loss ~ 0.175





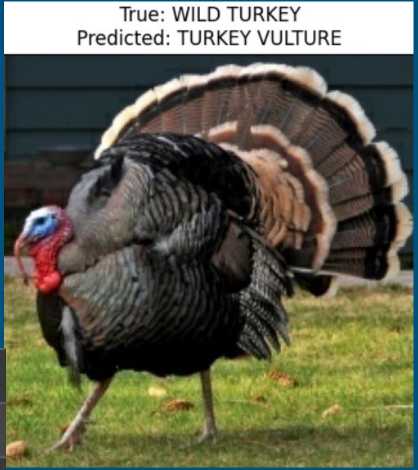
```
16/16 [==============================] - 30s 2s/step - loss: 0.1752 - accuracy: 0.9560
Test loss: 0.1751749962568283
Test accuracy: 0.9559999704360962
```

ResNet50 Transfer Learning
A Sample of the Results

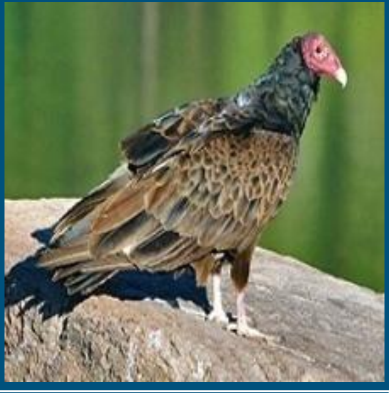


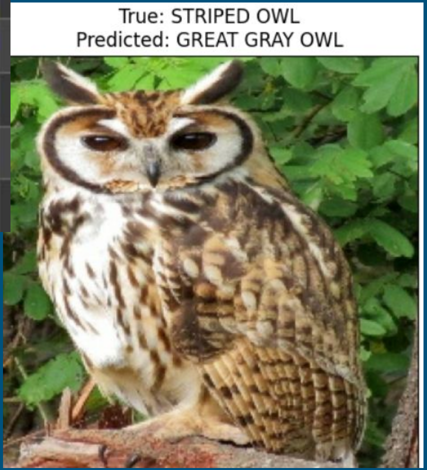
Bird Name	Precision	Recall
STRIATED CARACARA	1.0	0.2
STRIPED OWL	1.0	0.8
TIT MOUSE	0.8333333333333334	1.0
TOUCHAN	1.0	1.0
TURKEY VULTURE	0.625	1.0
WHITE EARED HUMMINGBIRD	1.0	1.0
WHITE NECKED RAVEN	1.0	0.6

Misclassification Examples

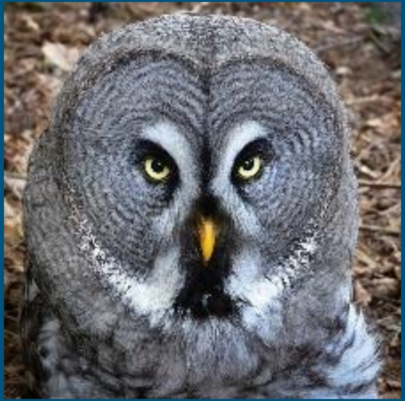


Turkey Vulture





Great Gray Owl



Other methods and Techniques

- Multinomial Logistic Regression
  - ~ 12.5 % Test Accuracy Rate
- Linear Support Vector Classifier
  - ~ 18% Test Accuracy Rate
  - Computationally extremely expensive and very slow to fit
  - Fitting SVC with non-linear kernels is especially slow

- Other methods will potentially benefit from a method of dimensionality reduction such as PCA unlike the CNN which often does not

- Computing power is a huge restriction

Choice of Final Model and Hyperparameters for Full Dataset

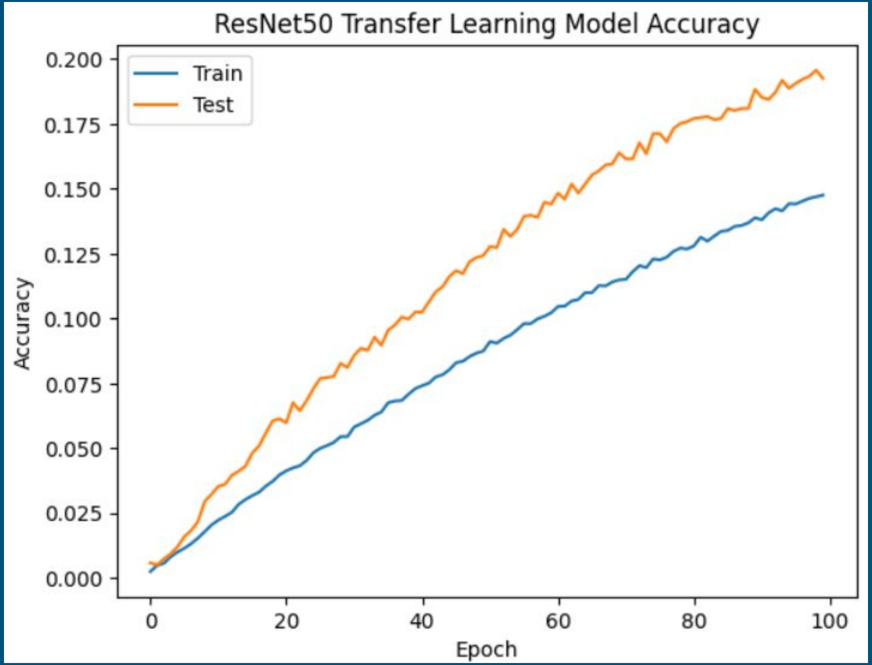
- ResNet50 Transfer Learning
- Adam w/ learning rate of 10e-5
- Added Data Augmentation Layer
  - Random Horizontal Flipping
  - Random Rotation of 10 deg
  - Random Contrast by 10%
  - Random Zoom In by 10%

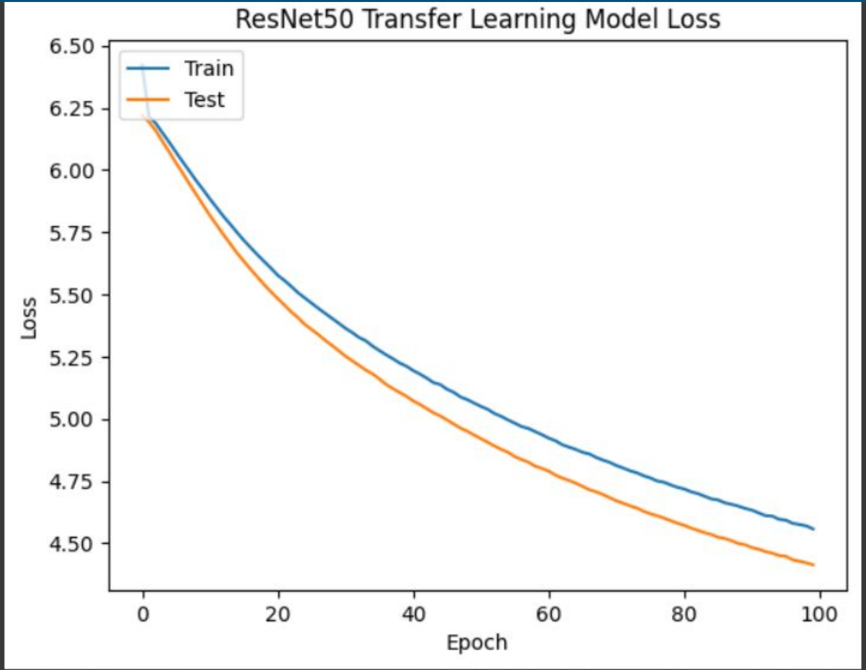
- Why this one?
  - Performed well
    - ~96% Test Accuracy
  - Trained fairly quick,w.r.t. the number of epochs
    - 50 epochs
  - Less parameters to train and computationally (more) feasible

515 Birds Classification Results - Not great

- Transfer Learning, ResNet50
- Adam with learning rate = 10e-5
- Loss and Accuracy Curves

Test Loss ~ 4.341
Test Accuracy ~ 20.08%





```
81/81 [==============================] - 396s 5s/step - loss: 4.3408 - accuracy: 0.2008
Test loss: 4.340805530548096
Test accuracy: 0.20077669620513916
```

Discussion of Final Model

- Transfer Learning is quite restrictive
  - Its benefit is computing power but will generally always perform worse than a fully trained model
  - Could add more layers to top
  - Could train part of the pretrained model
  - May need more similar dataset for pretrained model
- Attempted a decrease and increase in learning rate and did not improve results
- The ResNet50 Transfer Learning model seems to be at its limits, but not for certain

Works Cited

1. Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep learning*. The MIT Press.
2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2022). *An introduction to statistical learning: With applications in R*. Springer.
3. Nielsen, M. A. (1970, January 1). Neural networks and deep learning. Retrieved February 22, 2023, from http://neuralnetworksanddeeplearning.com/index.html
4. *Deep Learning Tutorial*. Unsupervised feature learning and Deep Learning Tutorial. (n.d.). Retrieved February 22, 2023, from http://ufldl.stanford.edu/tutorial/
5. Bagheri, R. (2020, August 28). *An introduction to Deep Feedforward Neural Networks*. Medium. Retrieved February 22, 2023, from https://towardsdatascience.com/an-introduction-to-deep-feedforward-neural-networks-1af281e306cd
6. https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview/

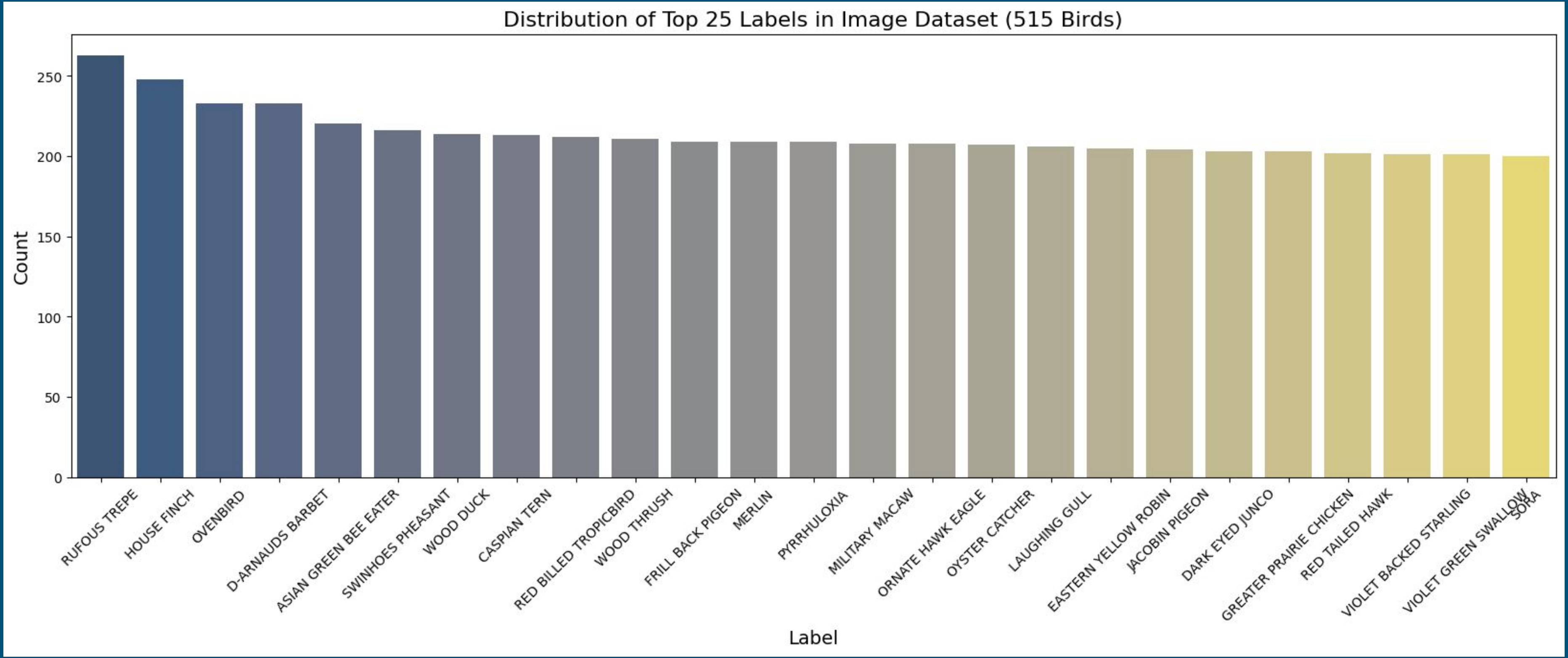
Data and Implementation Sources

The Bird Image dataset source: https://www.kaggle.com/datasets/gpiosenka/100-bird-species

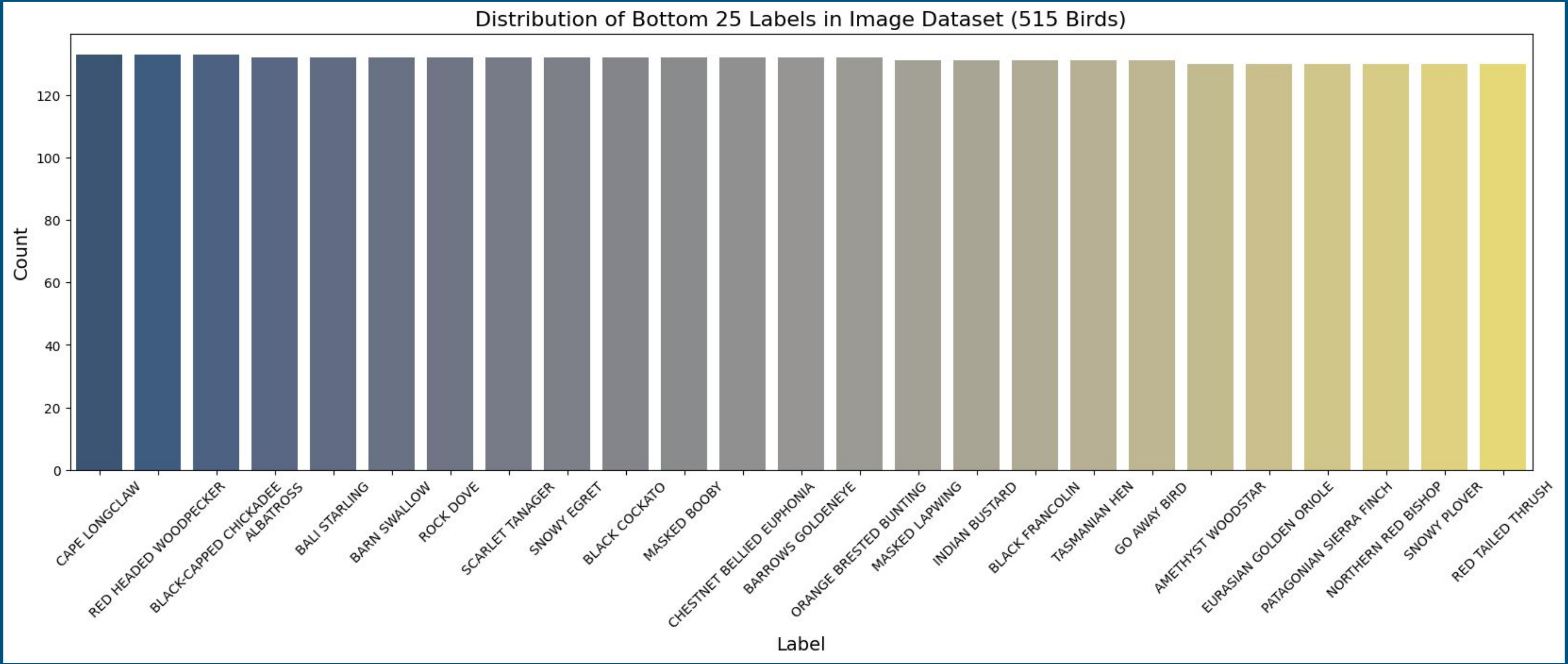
Inspiration behind implementation:
https://www.kaggle.com/code/vencerlanz09/bird-classification-using-cnn-efficientnetb0

Appendix

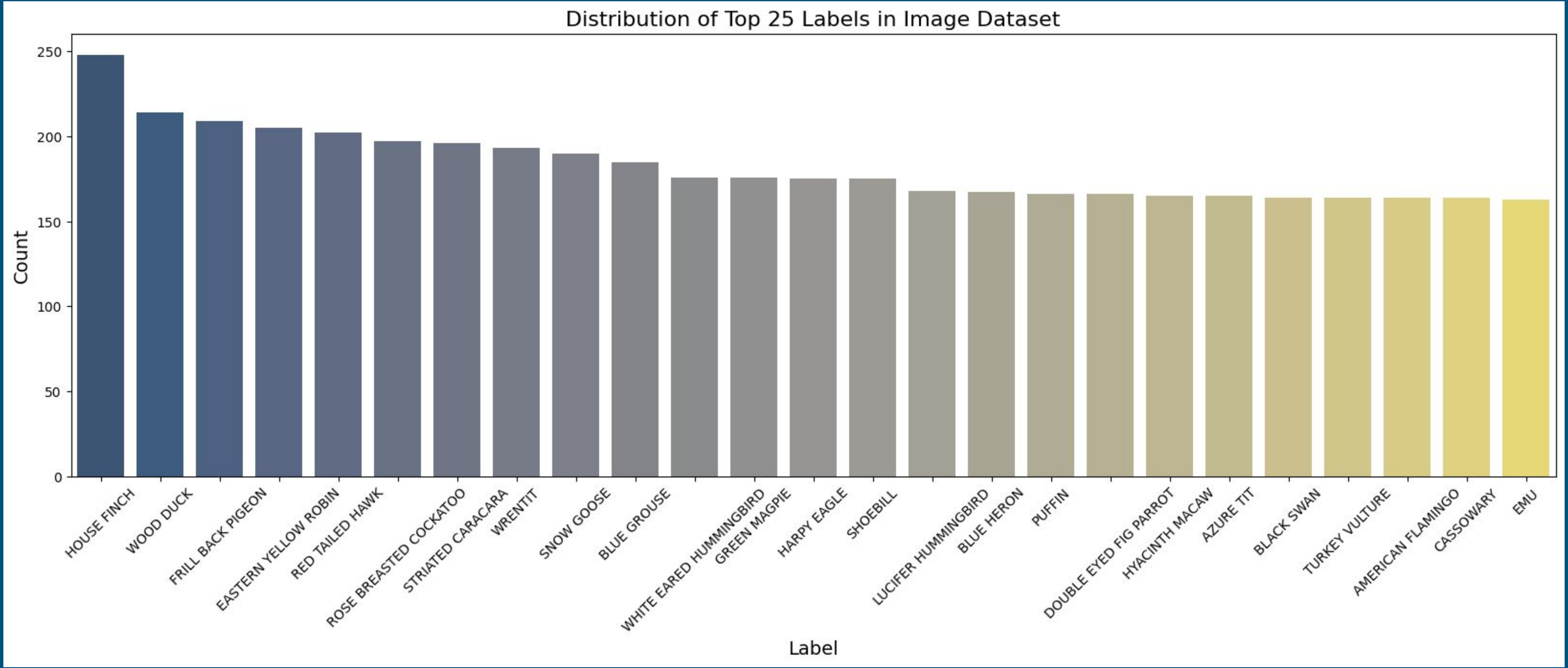
Distribution of Data for Full Dataset



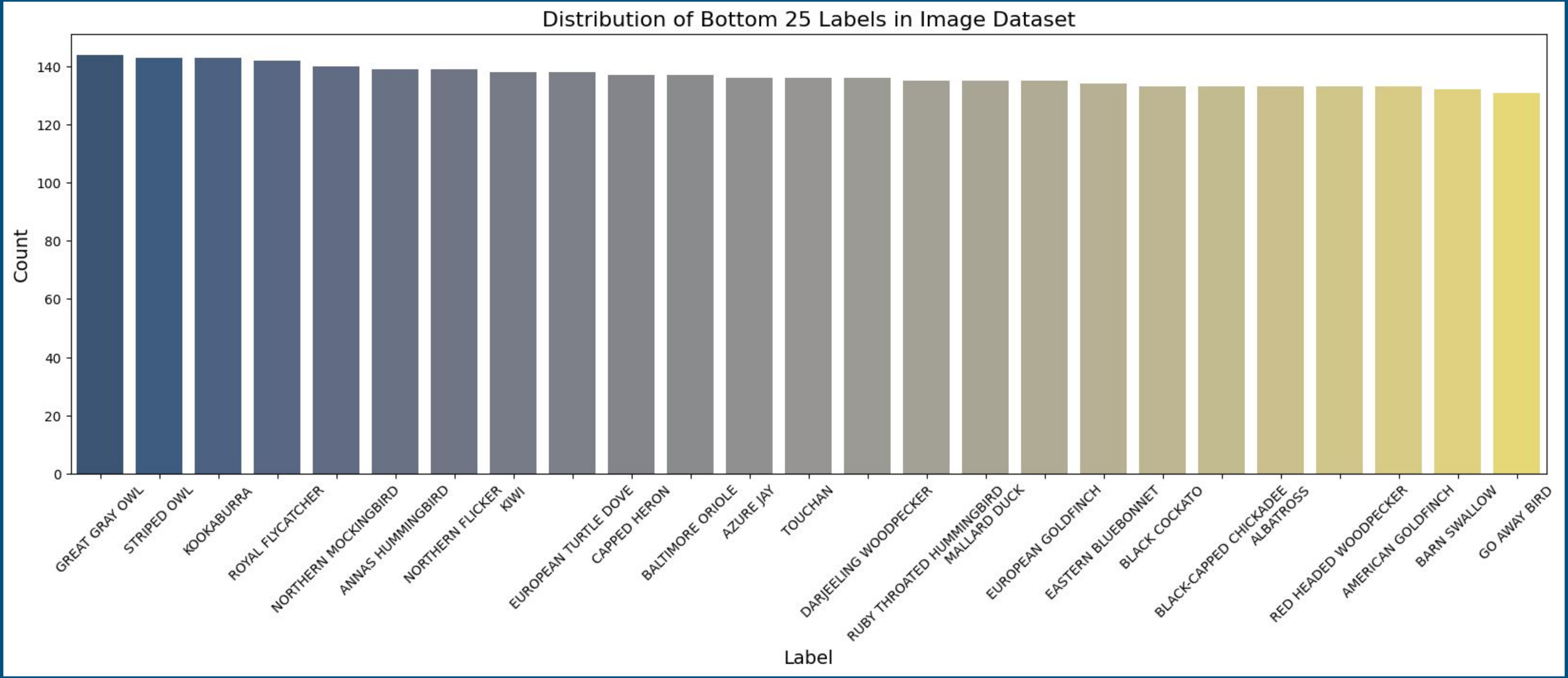
Distribution of Data for Full Dataset



Distribution of Data



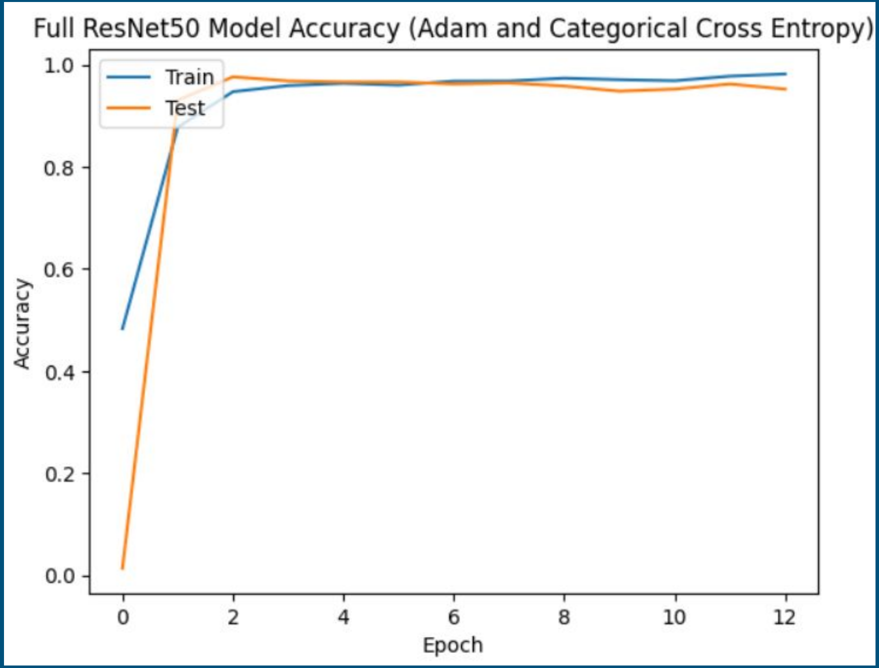
Distribution of Data

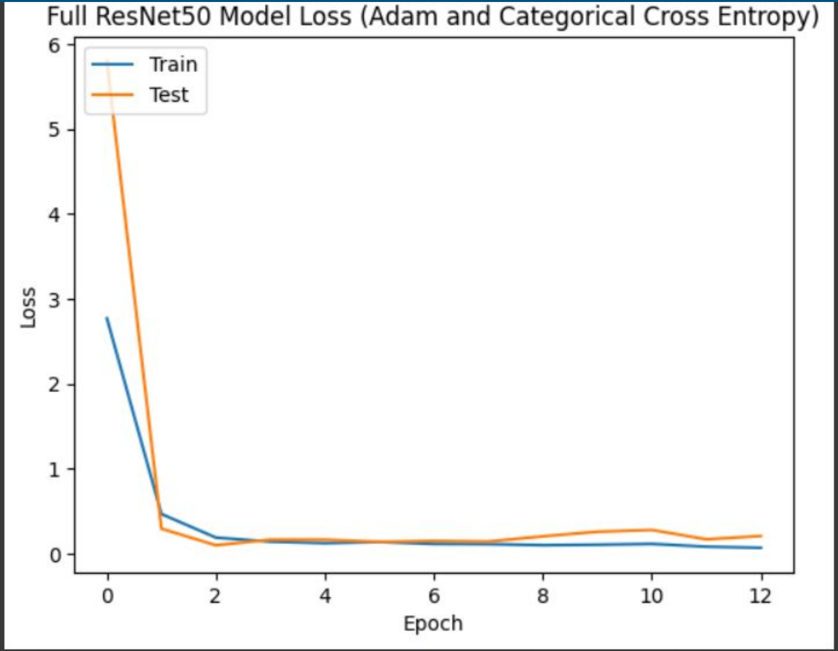


Fully Trained ResNet50

- Fully Trained ResNet50
- Adam with learning rate = 0.0001
- Total Params: 23,587,712
- Loss and Accuracy Curves

Test Accuracy ~ 97.6%
Test Loss ~ 0.059



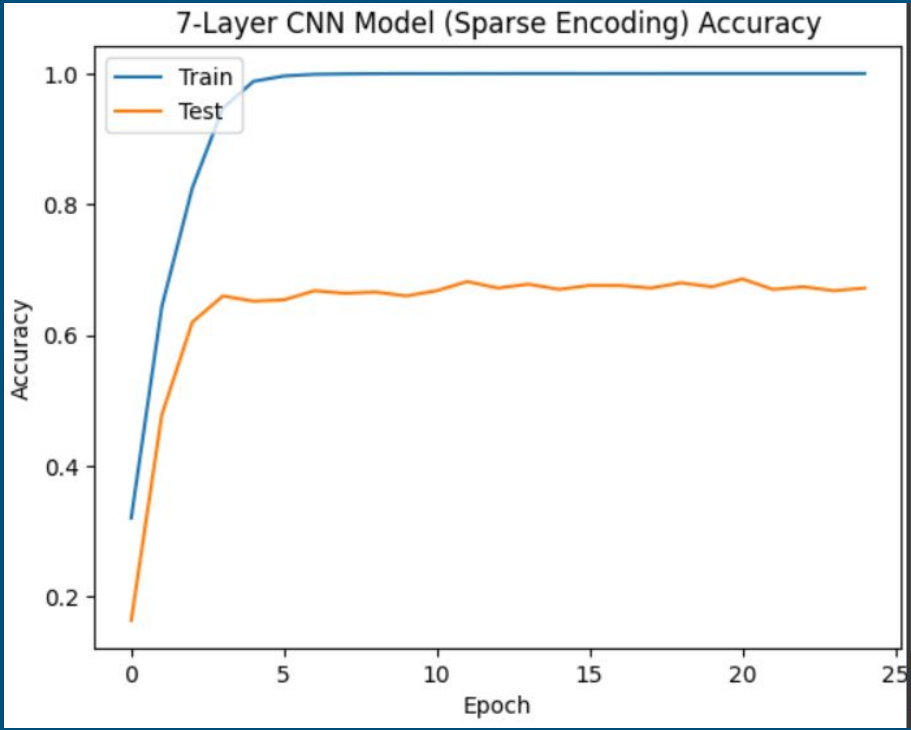


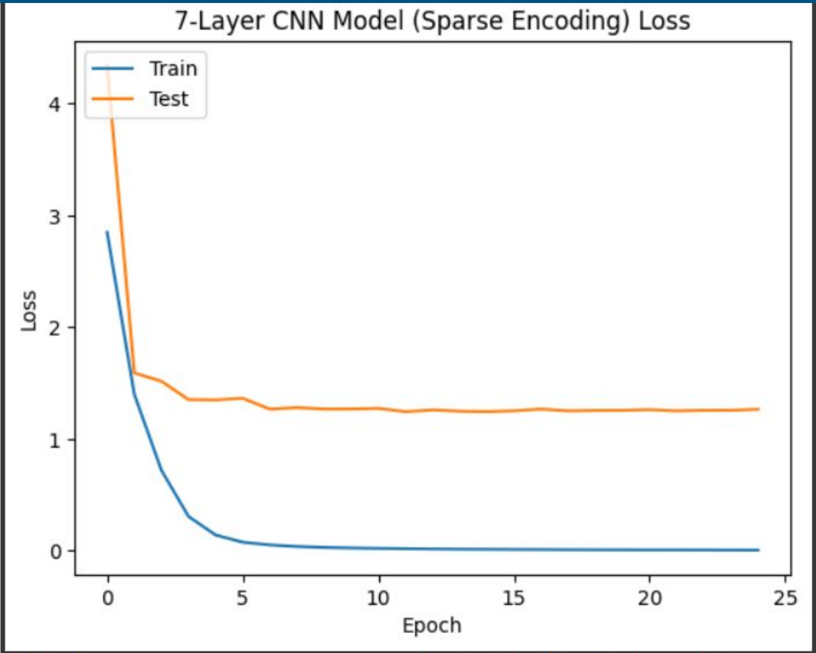
```
16/16 [==============================] - 0s 22ms/step - loss: 0.0593 - accuracy: 0.9760
Test loss: 0.059255003929138184
Test accuracy: 0.9760000109672546
```

7-Layer CNN with Sparse Encoding

- 7-Layer CNN with Sparse Encoding
- SGD with learning rate = 0.001
  - Loss and Accuracy Curves
  - Test Loss and Accuracy

Test Accuracy ~ 72.0%
Test Loss ~ 1.100





```
16/16 [==============================] - 0s 22ms/step - loss: 1.0989 - accuracy: 0.7200
Test loss: 1.0989142656326294
Test accuracy: 0.7200000286102295
```

Fitting our CNN

- How do we actually fit it to our data?
  - We use back propagation!
  - Back propagation is generalized for tensors
- The only essential differences lies in where our trainable parameters are from that we would like to adjust in order to fit our training data
- In the fully connected layer, it is the same as a feedforward NN. However, in the convolution layers, our parameters are the weights in each filter (kernel)
- Notice how sparse the number of parameters compared to fully connected layers