DoomCrossing

Using machine learning to identify the two side of an unlikely friendship

Background

- In March of 2020 two very different video games were released on the same day

Background

- In March of 2020 two very different video games were released on the same day
- This event lead to unlikely comradery between these disparate communities

Background

- In March of 2020 two very different video games were release on the same day
- This event led to unlikely comradery between these disparate communities
- ...and to a wealth of fan art and dank memes

The Data

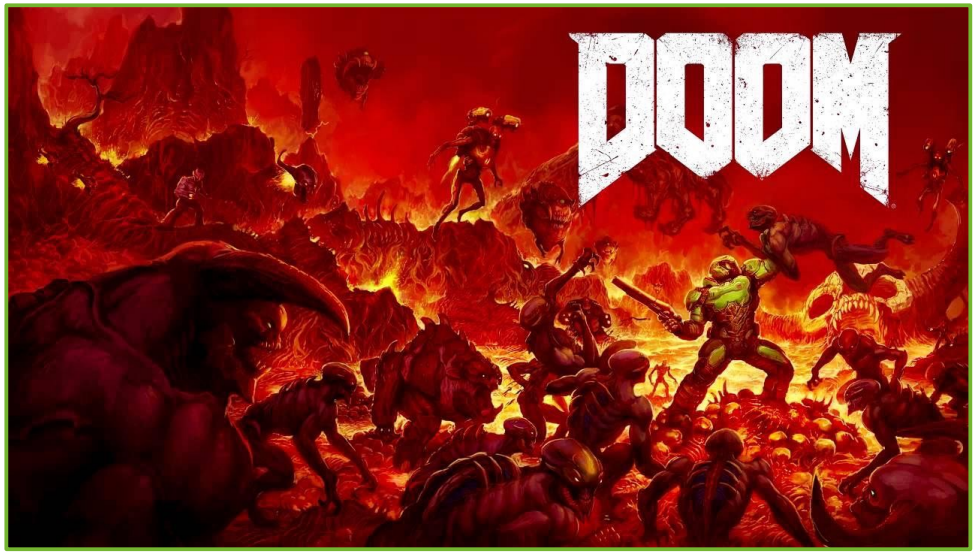
- Collection of images scraped from the Doom and Animal Crossing subreddits
  - Also includes information about the post in which the picture appeared
- Total of 1597 images; 757 from animal crossing and 840 from Doom
- Split into a training/test set
  - 10% into test set
- Images are highly varied
  - Some show the games
  - Others are text or memes

Data Examples: Animal Crossing

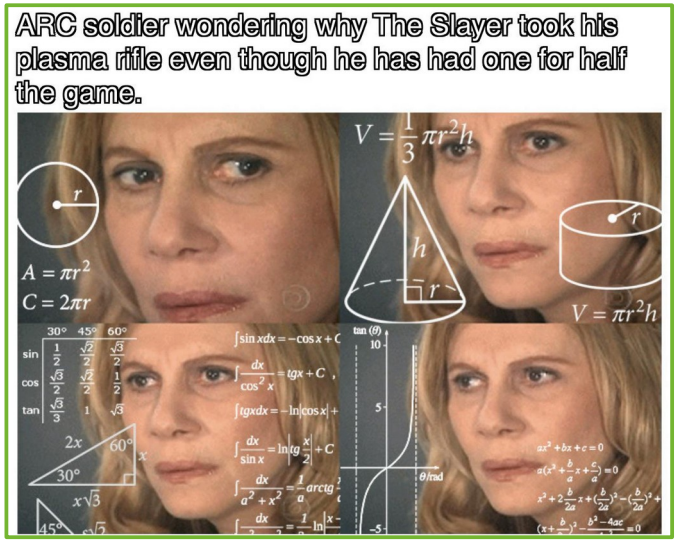
Data Examples: Animal Crossing

Data Examples: Animal Crossing

Data Examples: Doom



Data Examples: Doom



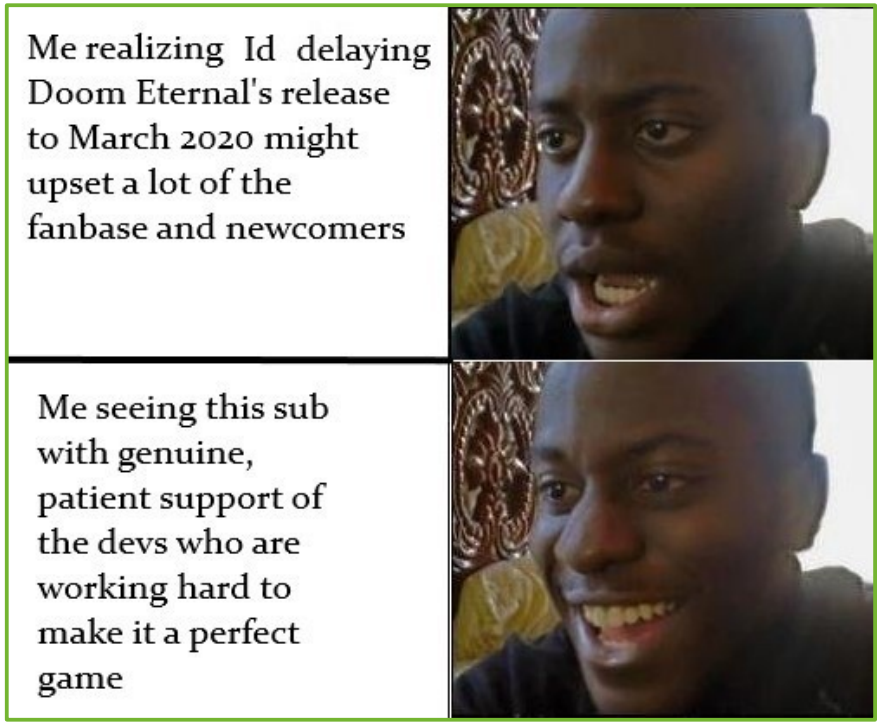
Data Examples: Shared

Doom

Animal Crossing

Data Examples: Shared

Doom



Animal Crossing

Data Examples: Complicated

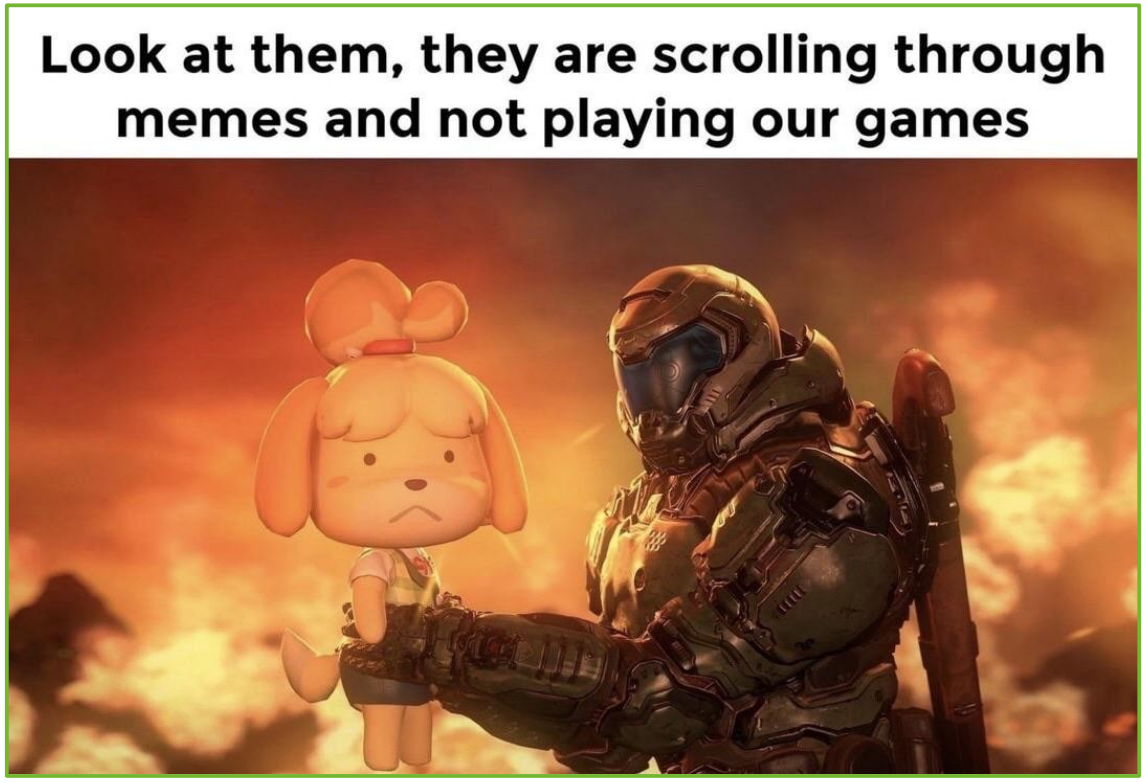
Doom

Doom

Animal Crossing

Data Examples: Complicated

Doom



Classification Models

- Hierarchical Clustering
- Tree Based Model
- Simple Neural Net
- PCA Optimized Neural Net
- Custom Convolutional Neural Nets
- Edge Detecting Neural Nets
- Inception Model
- ResNet18 Model

Preprocessing

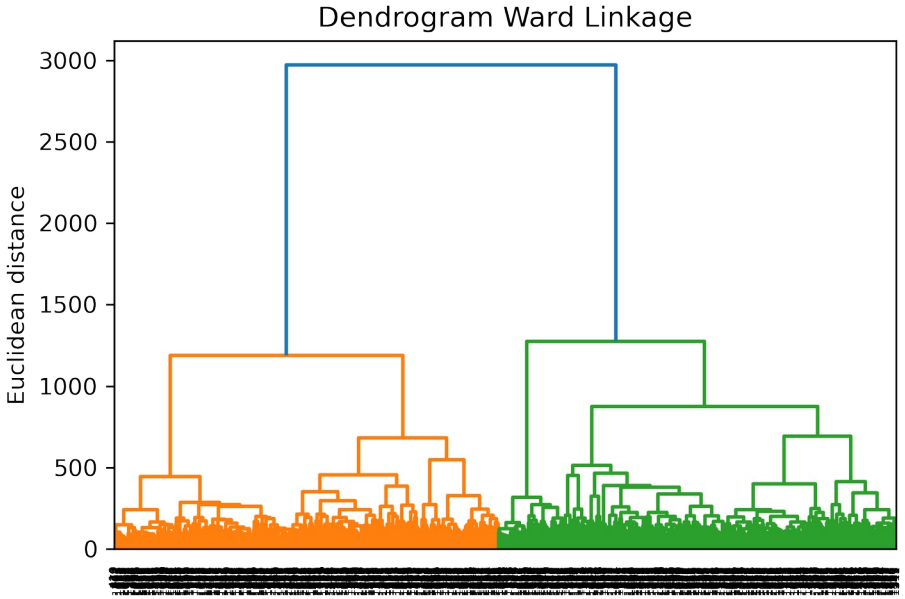
- Resize and crop
  - 200x200 for clustering/tree based model
  - 256x256 for neural nets
- Random horizontal flip on training set
- Normalization

Hierarchical Clustering

- Hierarchical clustering using four different linkage methods
  - Ward Method, Complete, Average, and Single
  - Ward method seeks to minimize inter-cluster variance
- Although the classification based on clustering does not necessarily align with the true labels, the confusion matrix can give some sense of how close the cluster proportions match the true proportions.
- Mostly just an excuse to make dendrograms

Hierarchical Clustering

- Ward Linkage



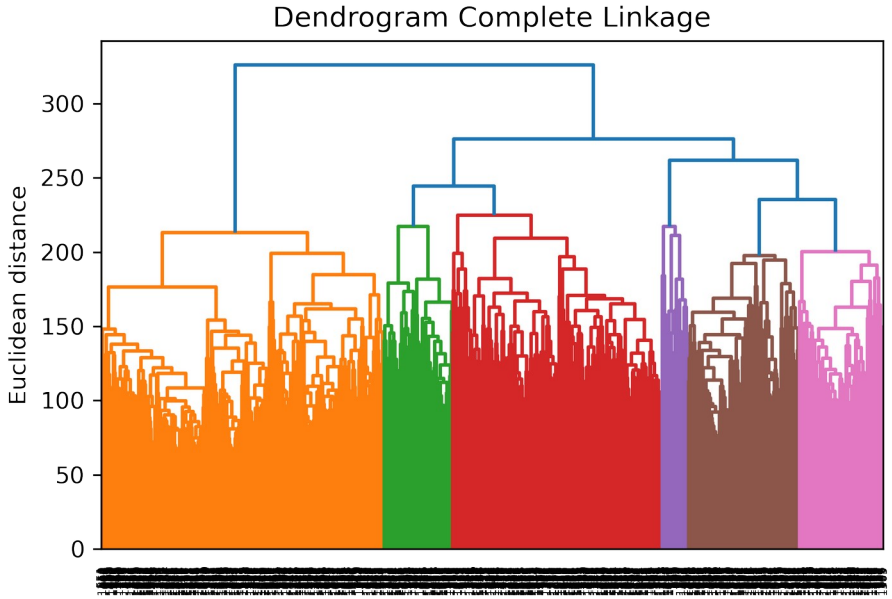
	precision	recall	f1-score	support
0.0	0.58	0.63	0.60	676
1.0	0.64	0.60	0.62	760
accuracy			0.61	1436
macro avg	0.61	0.61	0.61	1436
weighted avg	0.61	0.61	0.61	1436

Confusion Matrix

```
[[423 253]
 [307 453]]
```

Tree Based Models

- Complete Linkage

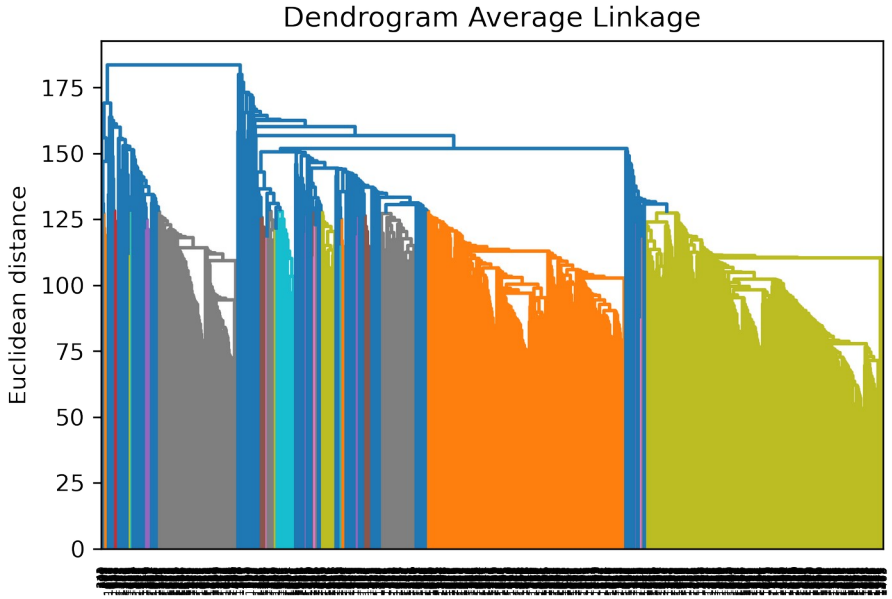


	precision	recall	f1-score	support
0.0	0.56	0.76	0.64	676
1.0	0.68	0.47	0.55	760
accuracy			0.60	1436
macro avg	0.62	0.61	0.60	1436
weighted avg	0.62	0.60	0.59	1436

Confusion Matrix
[[511 165]
 [406 354]]

Hierarchical Clustering

- Average Linkage

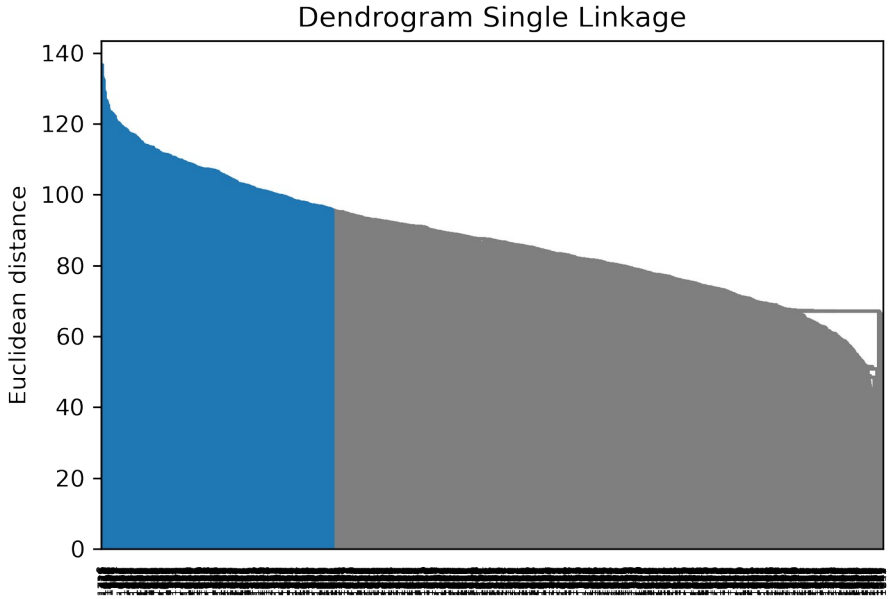


	precision	recall	f1-score	support
0.0	0.47	0.82	0.60	676
1.0	0.53	0.17	0.26	760
accuracy			0.48	1436
macro avg	0.50	0.50	0.43	1436
weighted avg	0.50	0.48	0.42	1436

Confusion Matrix
[[557 119]
 [628 132]]

Hierarchical Clustering

- Single Linkage

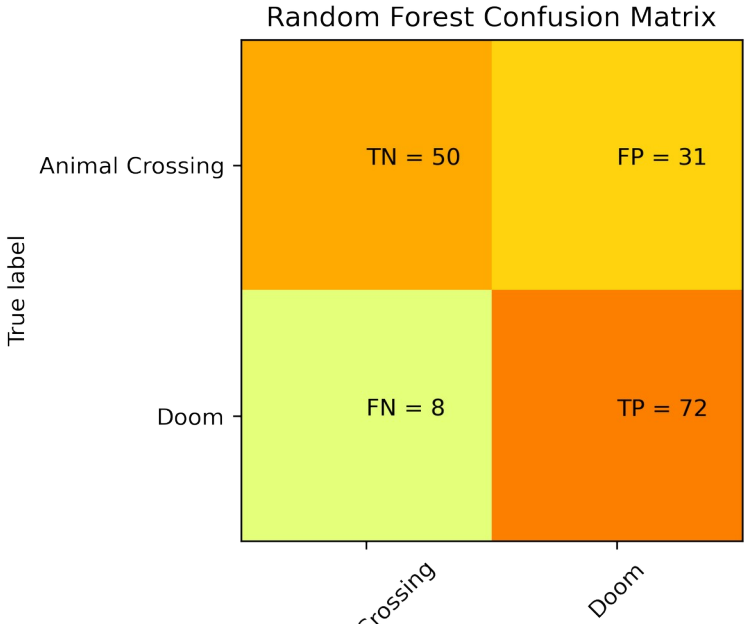


	precision	recall	f1-score	support
0.0	0.47	1.00	0.64	676
1.0	1.00	0.00	0.00	760
accuracy			0.47	1436
macro avg	0.74	0.50	0.32	1436
weighted avg	0.75	0.47	0.30	1436

Confusion Matrix
[[676 0]
[759 1]]

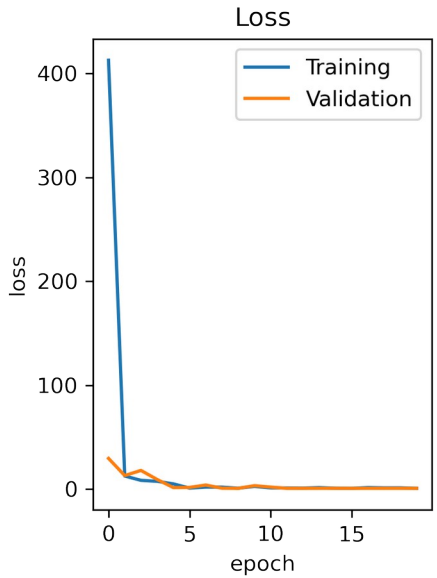
Tree Based Model

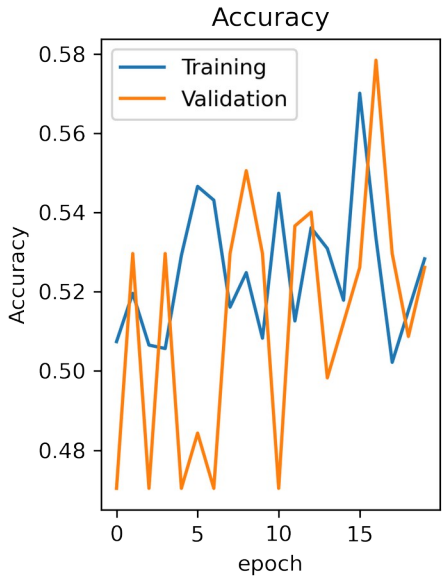
- Random Forest Model



	precision	recall	f1-score	support
0.0	0.86	0.62	0.72	81
1.0	0.70	0.90	0.79	80
accuracy			0.76	161
macro avg	0.78	0.76	0.75	161
weighted avg	0.78	0.76	0.75	161

Simple Neural Net





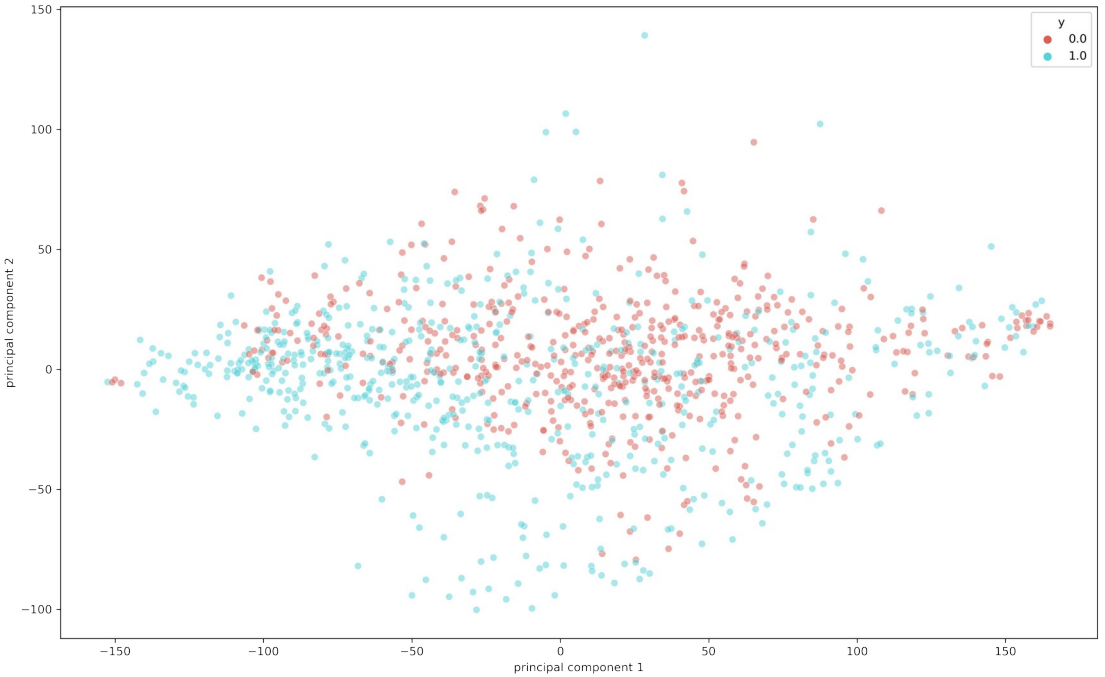

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1024)              302080
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_3 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 514
=================================================================
Total params: 2,008,322
Trainable params: 2,008,322
Non-trainable params: 0
_________________________________________________________________
```

Maximum validation accuracy: 57.8 %

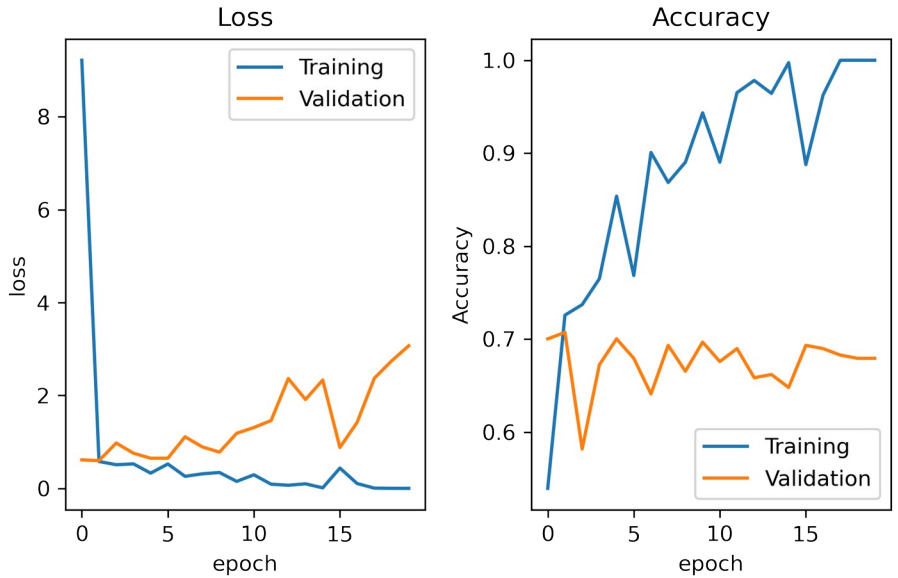
Test set accuracy: 53.3 %

PCA Optimized Neural Net

- PCA is ran on the flattened images
- The model is ran on the principal components that capture 90% of the variance in the data
- Training runs extremely quickly
- Variance captured in the first two PCs
  - [0.379 0.070]
  - Many PCs required but much less than actual pixels



PCA Optimized Neural Net

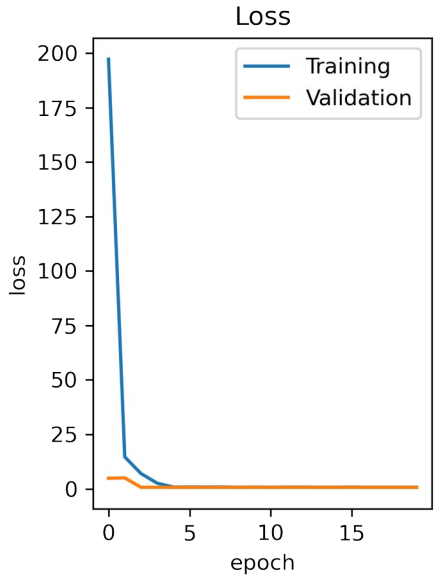


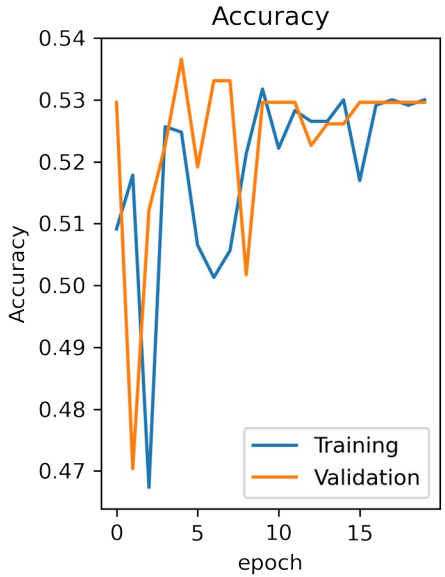

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1024)              302080
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_3 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 514
=================================================================
Total params: 2,008,322
Trainable params: 2,008,322
Non-trainable params: 0
_________________________________________________________________
```

Maximum validation accuracy: 70.7 %

Test set accuracy: 70.7 %

Simple Neural Net



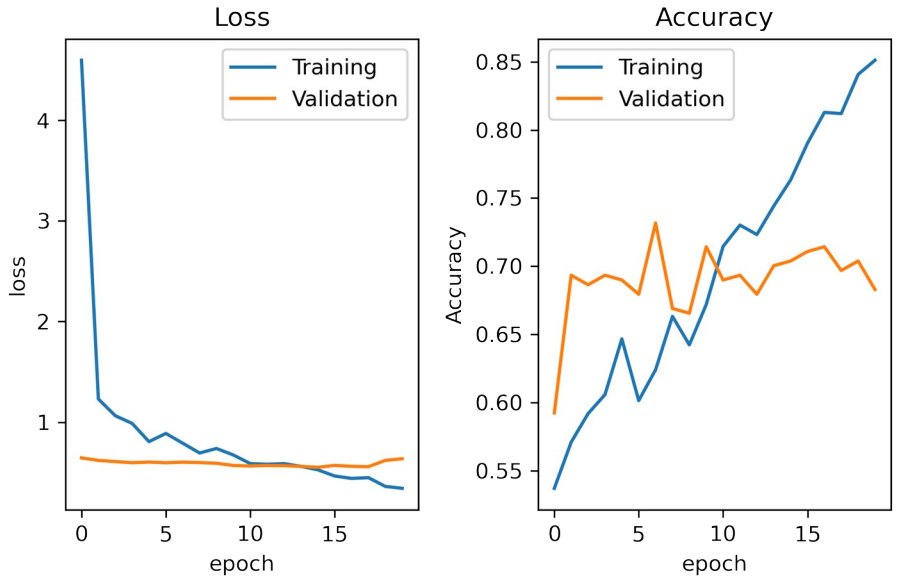



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_15 (Dense)             (None, 1024)              122881024
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_16 (Dense)             (None, 1024)              1049600
_________________________________________________________________
dropout_5 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_17 (Dense)             (None, 512)               524800
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_18 (Dense)             (None, 256)               131328
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_19 (Dense)             (None, 2)                 514
=================================================================
Total params: 124,587,266
Trainable params: 124,587,266
Non-trainable params: 0
_________________________________________________________________
```

Maximum validation accuracy: 53.6 %

Test set accuracy: 53.0 %

PCA Optimized Neural Net



Model: "sequential_2"

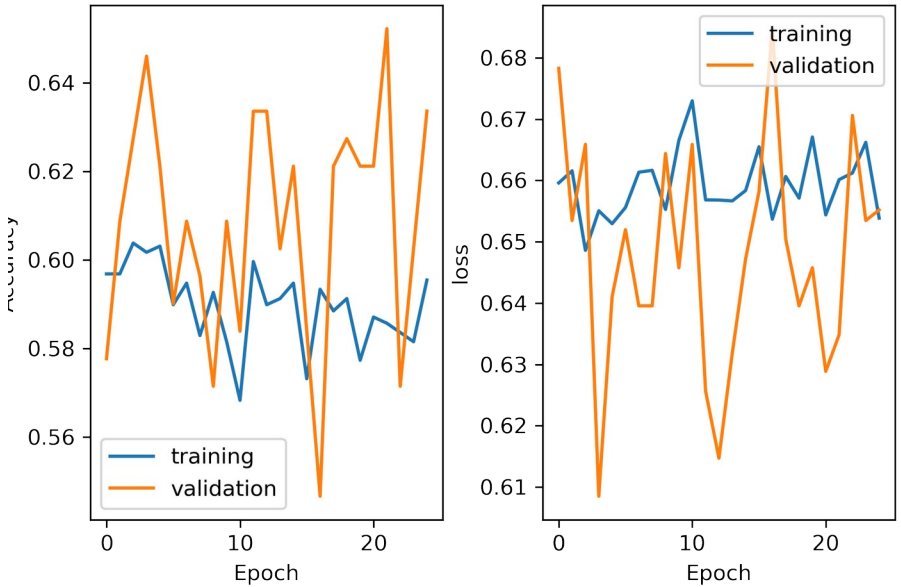
Layer (type)	Output Shape	Param #
dense_10 (Dense)	(None, 1024)	302080
dropout (Dropout)	(None, 1024)	0
dense_11 (Dense)	(None, 1024)	1049600
dropout_1 (Dropout)	(None, 1024)	0
dense_12 (Dense)	(None, 512)	524800
dropout_2 (Dropout)	(None, 512)	0
dense_13 (Dense)	(None, 256)	131328
dropout_3 (Dropout)	(None, 256)	0
dense_14 (Dense)	(None, 2)	514

Total params: 2,008,322
Trainable params: 2,008,322
Non-trainable params: 0

Maximum validation accuracy: 73.1 %

Test set accuracy: 71.4 %

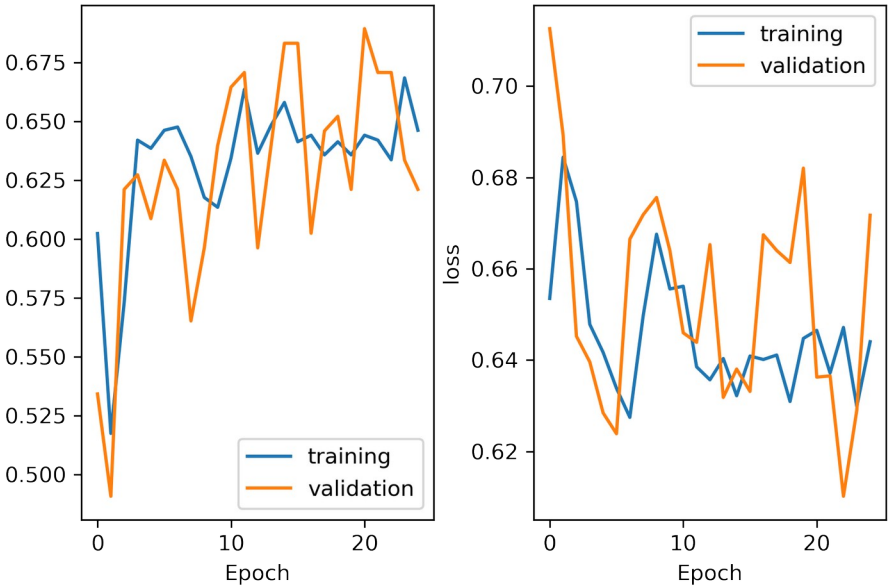
Custom Convolutional Neural Nets



- Built with PyTorch
- One convolutional layer
  - out_channels: 128
  - Relu activation
  - 3x3 kernels
  - MaxPool (2x2 kernel)
- One fully connected layer
  - Softmax activation
- Adam Optimizer
  - With learning rate scheduler

Best Validation Accuracy: 65.2%

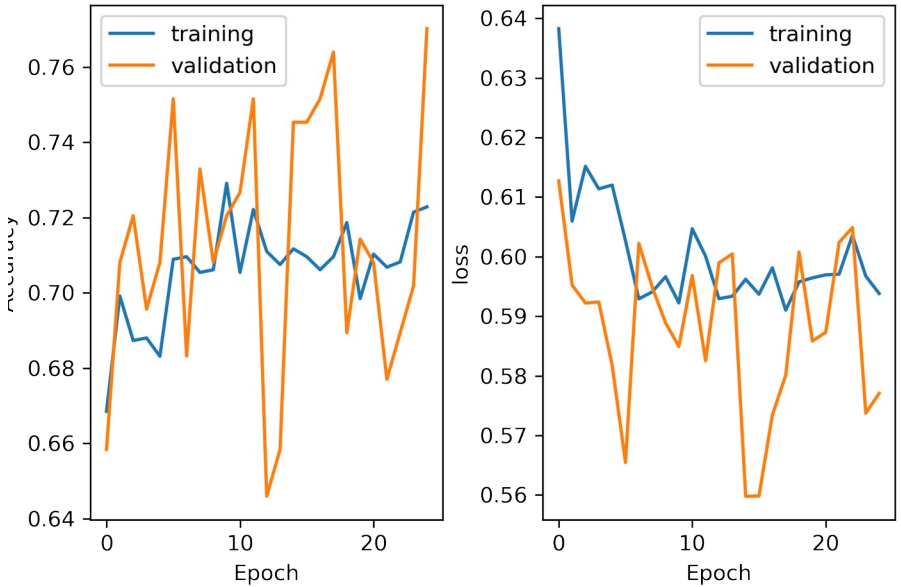
Custom Convolutional Neural Nets



- Built with PyTorch
- Two convolutional layers
  - out_channels: 16/32
  - Relu activation
  - 3x3 kernels
  - MaxPool (2x2 kernel)
- One fully connected layer
  - Softmax activation
- Adam Optimizer
  - With learning rate scheduler

Best Validation Accuracy: 68.9%

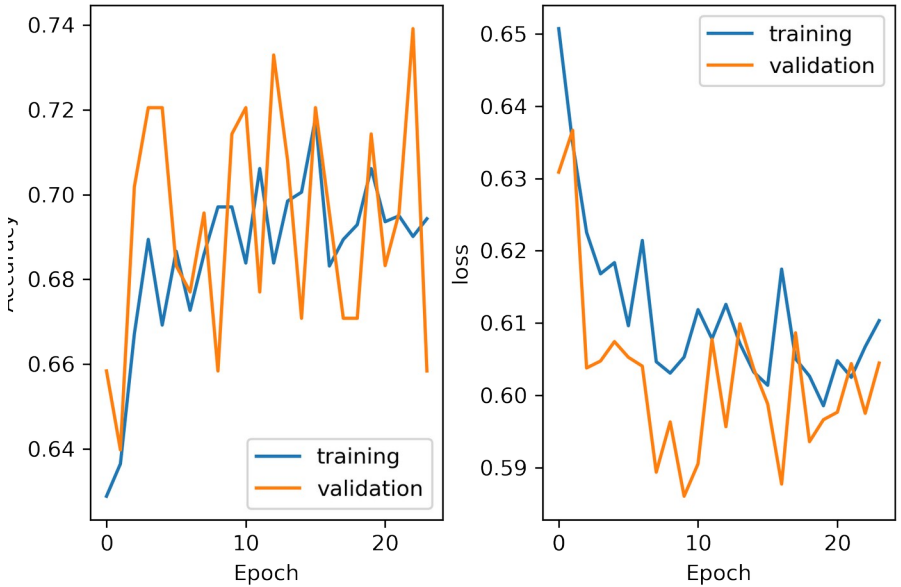
Custom Convolutional Neural Nets



- Built with PyTorch
- Two convolutional layers
  - out_channels: 16/32
  - Relu activation
  - 3x3 kernels
  - MaxPool (2x2 kernel)
- Two fully connected layer
  - Softmax activation
- Adam Optimizer
  - With learning rate scheduler

Best Validation Accuracy: 77.0%

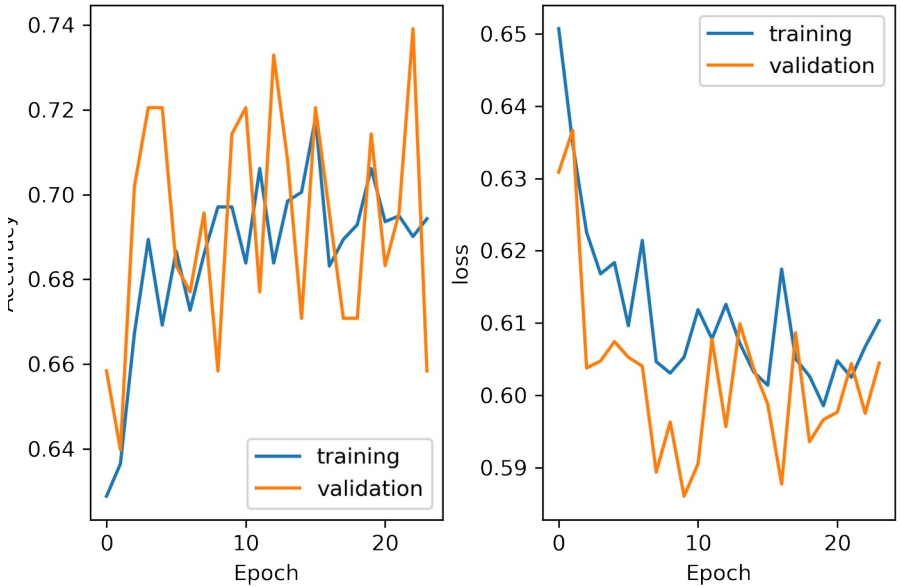
Custom Convolutional Neural Nets



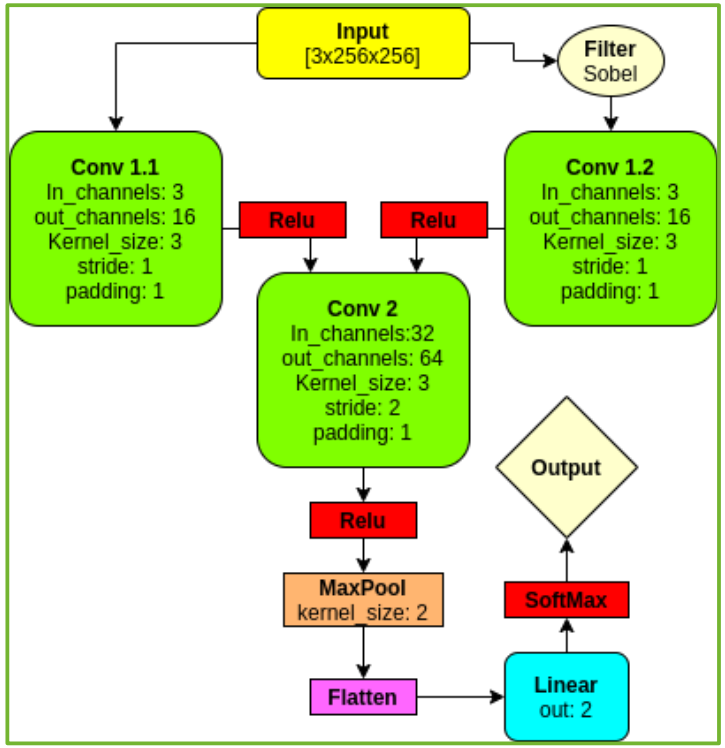
Best Validation Accuracy: 73.9%

- Parallel convolutions
  - One on raw input and another with filtered input (sobel)
- Three convolutional layers
  - out_channels: 16/16/64
  - Relu activation
  - 3x3 kernels
  - MaxPool (2x2 kernel)
- Two fully connected layer
  - Softmax activation
- Adam Optimizer
  - With learning rate scheduler

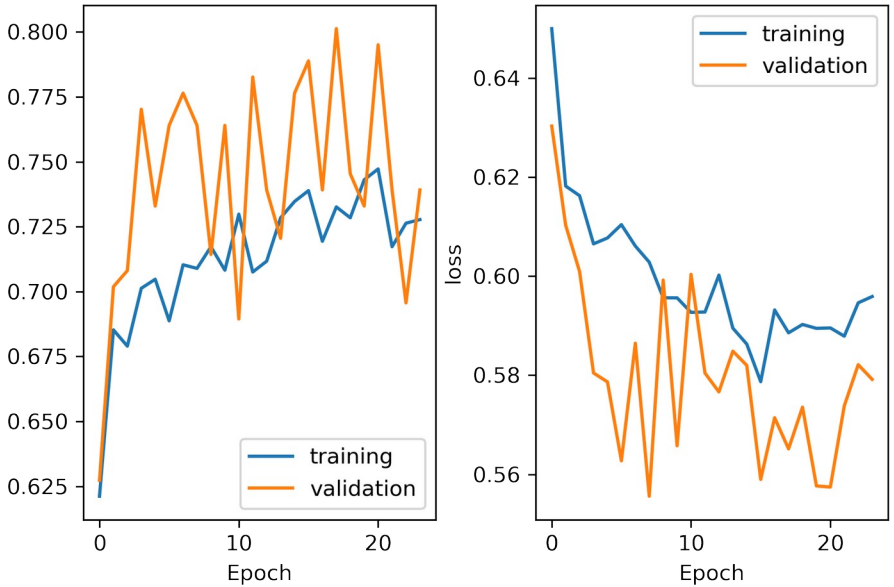
Custom Convolutional Neural Nets



Best Validation Accuracy: 73.9%



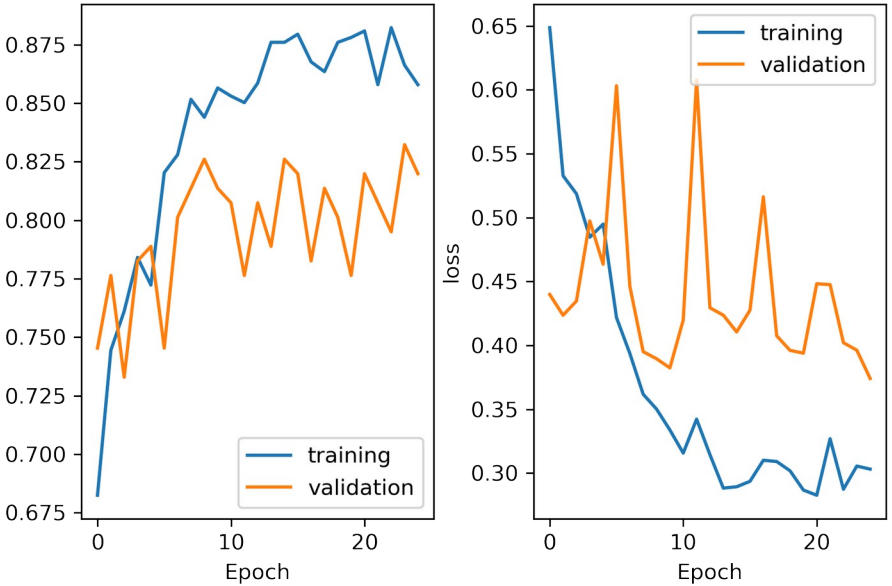
Custom Convolutional Neural Nets



Best Validation Accuracy: 80.1%

- Parallel convolutions
  - One on raw input and another with filtered input (sobel)
- Five convolutional layer
  - out_channels: 8/8/16/16/64
  - Relu activation
  - 3x3 kernels
  - MaxPool (2x2 kernel)
- Two fully connected layer
  - Softmax activation
- Batch normalization of final convolution

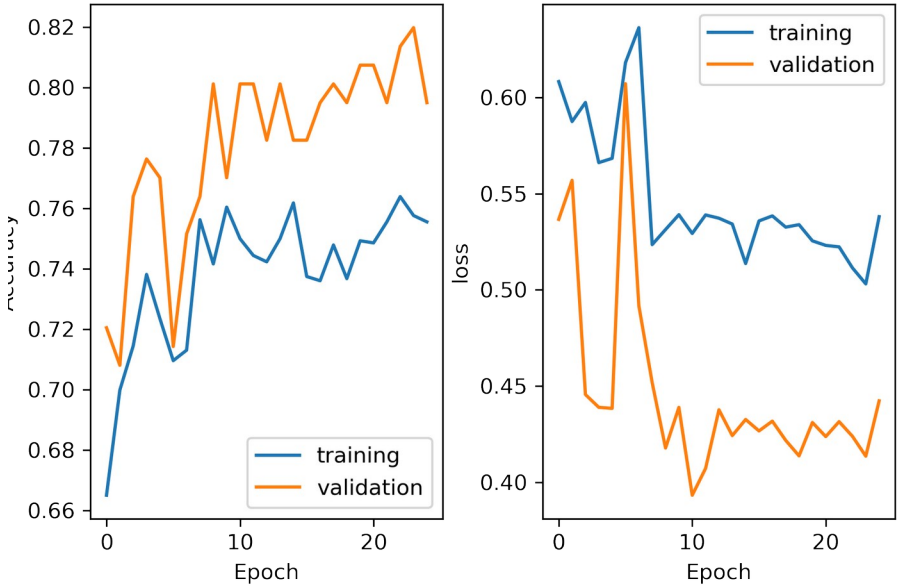
Inception v3



- All weights trained
- Cross entropy loss
- Stochastic gradient descent

Best Validation Accuracy: 83.2%

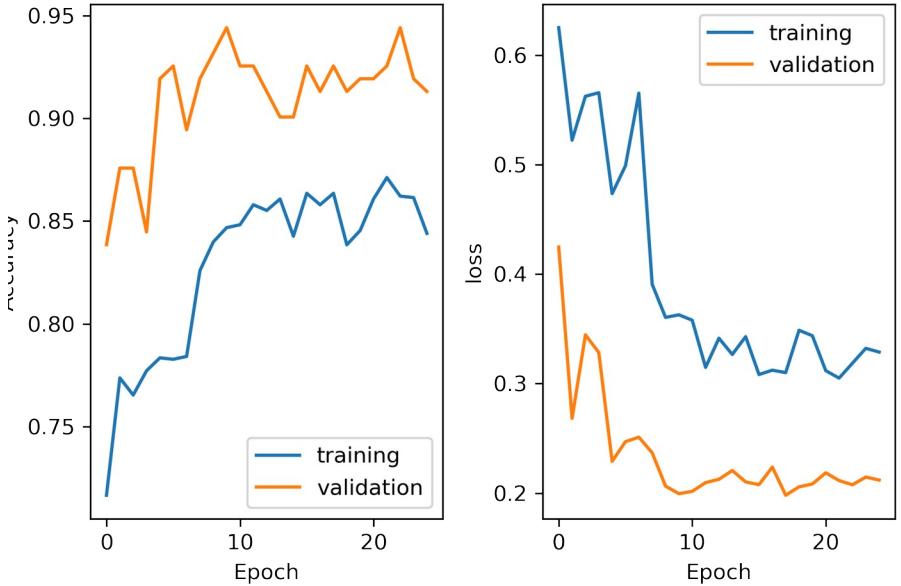
Inception v3



- As a feature extractor
  - Only last layer trained
- Cross entropy loss
- Stochastic gradient descent

Best Validation Accuracy: 82.0%

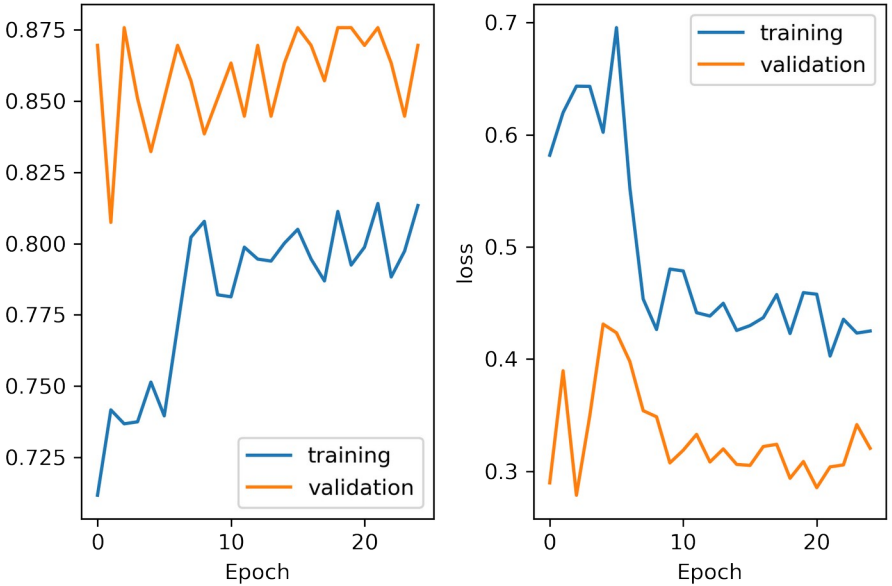
ResNet 18



- All weights trained
- Cross entropy loss
- Stochastic gradient descent

Best Validation Accuracy: 94.4%

ResNet 18



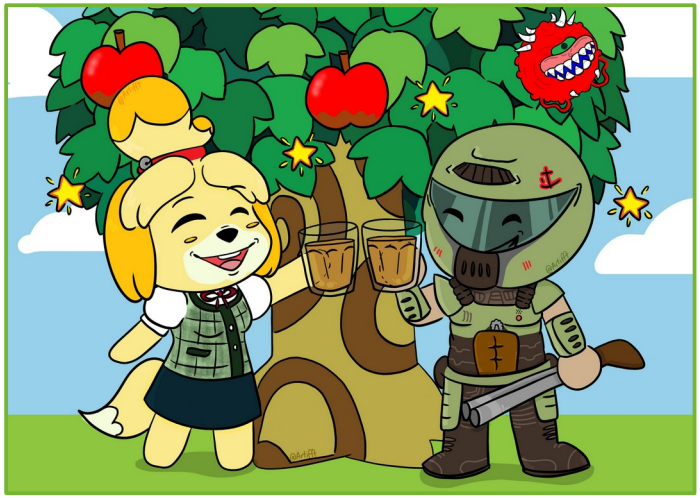
- As a feature extractor
  - Only last layer trained
- Cross entropy loss
- Stochastic gradient descent

Best Validation Accuracy: 87.6%

Misclassification Examples

Predicted: Animal Crossing
Actual: Doom

Predicted: Animal Crossing
Actual: Doom



Predicted: Doom
Actual: Animal Crossing

Misclassification Examples

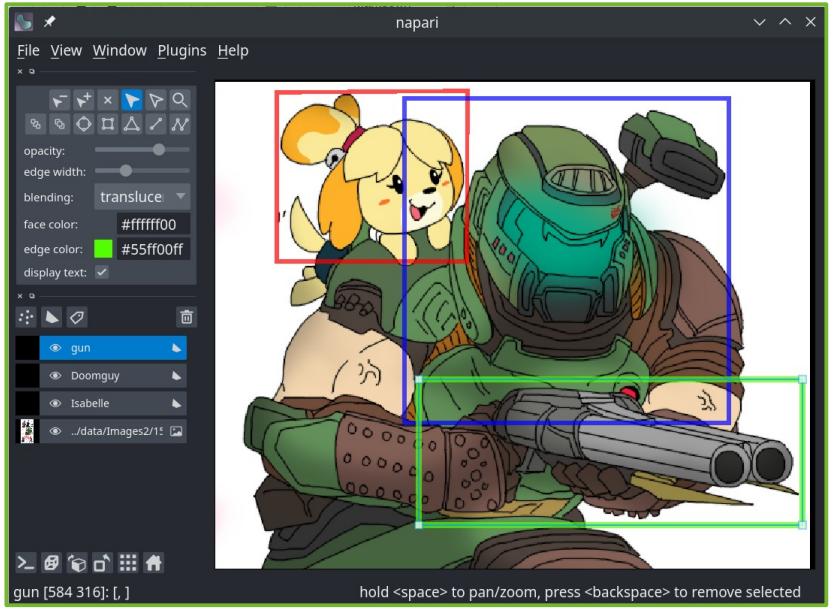
Predicted: Animal Crossing
Actual: Doom

Predicted: Doom
Actual: Animal Crossing

Predicted: Doom
Actual: Animal Crossing

Next Steps

- Creating a model that generates a caption for an image
  - The data contains all of the reddit post information including the title which could be used to train a generative model
  - A region based CNN would be used to identify features in the image and a recurrent neural network would generate text based on this output using the true captions as examples
    - Requires annotation of classes in training images (in progress)
    - Success would be generation of a legible caption that had some level of relation to the image



Thank You

Questions?