Report

April 7th, 2019

Regarding:

Cat and Dog Classification

Springboard Course Work
Capstone Project #2

Cliff Robbins
cliff@gearforgesoftware.com
M 612.701.2998

Introduction

This project is leveraging deep learning to create a program that will detect whether an image is a cat or dog.

The project is broken up into multiple phases:

1. Cat or Dog Detection
2. Specific Cat or Dog Detection

For capstone project two, will focus on phase one and tackling the remaining with time permitting.

Client

The client identified is a dog trainer that will leverage the software to sell with his current training regime. This will include further work outside the scope of capstone project 2.

History

Roughly 89.7 million dogs live with families in the United States[1]. Approximately, 3.3 million dogs are booked into animal shelters each year, with 670,000 euthanized[3]. Various reasons are stated for why animals are given to shelters. In one study, it was estimated that 96% of surrendered dogs had not received obedience training[2].

DataSet

It is recommended to have at least 100,000 images to train a deep learning model. I will combine various data sets of dogs, cats and other animals/objects to get the necessary images for training the model. Here is where those images are pulled from along with the information about each dataset.

Kaggle Cat vs Dog

- 10,000 images
- 5,000 dogs and 5,000 cats
- JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, baseline, precision 8, 300x335, frames 3
- https://www.kaggle.com/c/dogs-vs-cats

Kaggle Dog Breed Identification

- 10,223 images
- All dogs
- JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, baseline, precision 8, 500x375, frames 3
- https://www.kaggle.com/c/dog-breed-identification/data

Stanford Dogs Dataset

- 20,580 images
- All dogs
- JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, baseline, precision 8, 250x188, frames 3
- http://vision.stanford.edu/aditya86/ImageNetDogs/main.html

Image-net

This data set only provide the urls. I wrote a small python program to pull the images from the listed urls.

- 1827 images of cats
- 1594 images of dogs
- http://image-net.org/explore
- JPEG image data, JFIF standard 1.01, resolution (DPI), density 300x300, segment length 16, baseline, precision 8, 500x375, frames 3

PASCAL VOC

The PASCAL Visual Object Classes challenge has five years of competition data that has 20 classes with various images for each year. This data needs to be collated into a single folder.

- 2006 - 2,618 images in 10 classes
- 2007 9,963 images in 20 classes
- 2008 4,340 images in 20 classes
- 2009 7,054 images in 20 classes
- 2010 10,103 images in 20 classes
- 2011 11,530 images in 20 classes
- 2012 11,530 images in 20 classes
- JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, baseline, precision 8, 486x500, frames 3
- http://host.robots.ox.ac.uk/pascal/VOC/

The specific animal images will come from various images from the clients dogs that he currently boards and trains. Those images will be broken up into various images using OpenCV to create respective datasets.

One challenge with using various image datasets are the different sizes of each image.

Analysis

To conduct classification for the two phases, deep learning algorithms are used to create a baseline and then modify the models and hyperparameters for additional tuning.

Approach

Data Acquisition and Wrangling

Data Investigation

The first step is to import the data from the various locations and integrate the data.

The data is located in 11 different directories with each having different structures and organization. Each dataset has its own challenges that need addressed.

20 different categories were established based on the data, which allows the various images to get stored in sub-directories in those category folders. The directories/categories are:

1. cat
2. car
3. dog
4. pottedplant
5. person
6. sheep
7. boat
8. chair
9. bus
10. motorbike
11. sofa
12. bird

13. aeroplane
14. tvmonitor
15. train
16. bicycle
17. horse
18. bottle
19. cow
20. diningtable

Kaggle Cat vs Dog & Kaggle Dog Breed Identification

Both of these datasets are broken into train/test folders where all the files reside. They both have sub-directories for cat and dog. Those were manually copied to the corresponding cat/dog folder.

Stanford Dogs Dataset

The dataset is broken into 127 different folders that each represent a different dog breed. To bring the data together, a python program was written to parse the directories and copy the images into a single image directory under the dog category.

Image-net

This data set only provide the urls. A small python program was written to retrieve the files and then stored locally under the correct dog or cat directory.

PASCAL VOC

The PASCAL VOC consists of 7 different projects. Each of those have 20 classes. The project has an annotation file for each class that lists the file name along with a 1 or -1 to indicate if it belongs to that class. A python program was written to iterate through all 7 projects and the 20 annotation files and read the file then sort the images into the correct location based on class.

After the images were all moved into a single source directory, a bash script was used to get the total image count based on class.

cat	11621
car	5519
dog	53579
pottedplant	765
person	10699

sheep	1345
boat	1648
chair	2841
bus	761
motorbike	875
sofa	2274
bird	3746
aeroplane	3195
tvmonitor	1956
train	2174
bicycle	2080
horse	2250
bottle	3089
cow	1369
diningtable	402

There are a total of 112,188 images.

Data Cleaning

Since these data are all images, a simple way to test if the image is a proper image is to open the image. A python program was written to loop through all of the images and try to open each image and capture the size.

While running the program, it was found that 1,169 files were bad. Those files were either corrupted or had zero bytes of data. Those files were removed.

After removing the bad files, all of the images were copied, via a python program, to a new directory to ensure the original files were still accessible.

Dealing with Missing Data Values

Because the images are all valid, the remaining data are all acceptable for processing.

Data Quality

A key part of working with images are understanding the size and aspect ratio of the image. A python program was used to list out all the different size variations of the 110,604 images. The images have 8,497 different sizes with 6,992 different aspect ratios.

The following data provides additional information regarding basic statistics for the aspect, width and height.

	aspect	width	height
count	110604	110604	110604
mean	1.22	453.93	384.14
std	0.30	105.98	95.32
min	0.19	43	39
25%	1.00	375	333
50%	1.33	500	375
75%	1.38	500	414
max	7.04	3264	2562

To ensure data quality, each of the min and max images were opened to ensure they were not outside of norms. This is when a wrong image was discovered for the min height of 33. It ended up being an image of a YAHOO! banner instead of a cat.

Scaling the Data

Before the images can get used by the models, the data needs to be in the same shape. For an image, that will mean taking that image and turning it into a vector that represents the image in a series of numbers. The issue is that if the aspect is not the same for all of the images, the vectors will also not be the same after they are converted. This will not work for the models.

Based on the information above regarding aspect, width and height, it is evident that the images are not all the same. All of the images need to be updated so they are all the same aspect ratio[2].

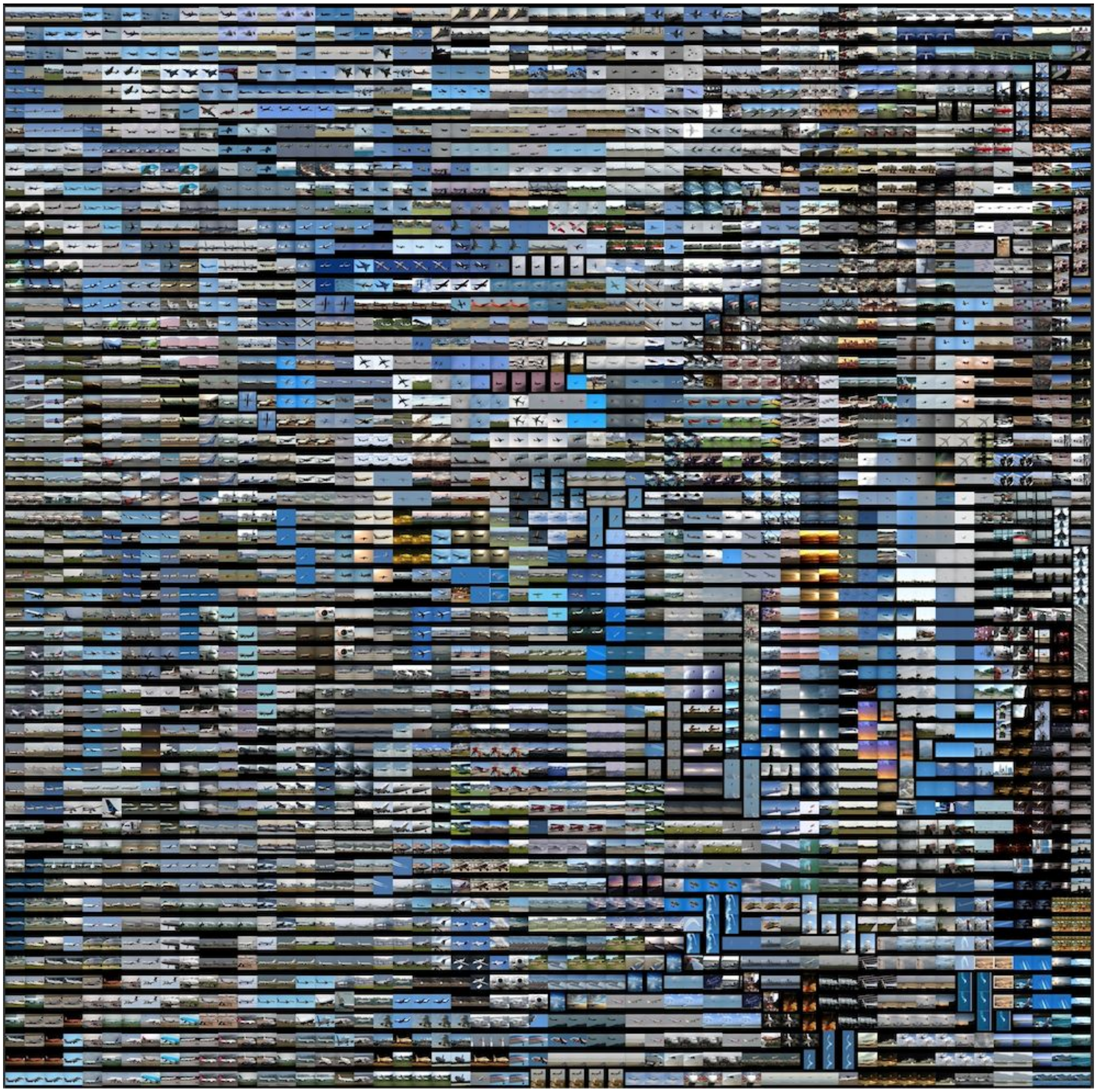
Therefore, each image will get sized to a 300 by 300 image[1]. To achieve this, each image needs opened and a new size needs calculated based on the old size and then padding applied to the image to ensure we have a square 300 by 300 image. 300 was decided due to the statistics that stated that 50% of the height were at 375.

A python program was written to first test the image resizing and then another was written to resize all of the images. Once all images were resized and verified, the unresized images were removed (note, the original image data are still kept in a raw data folder).

Now that the images are broken out into each classification folder, the images are ready for processing by the model.

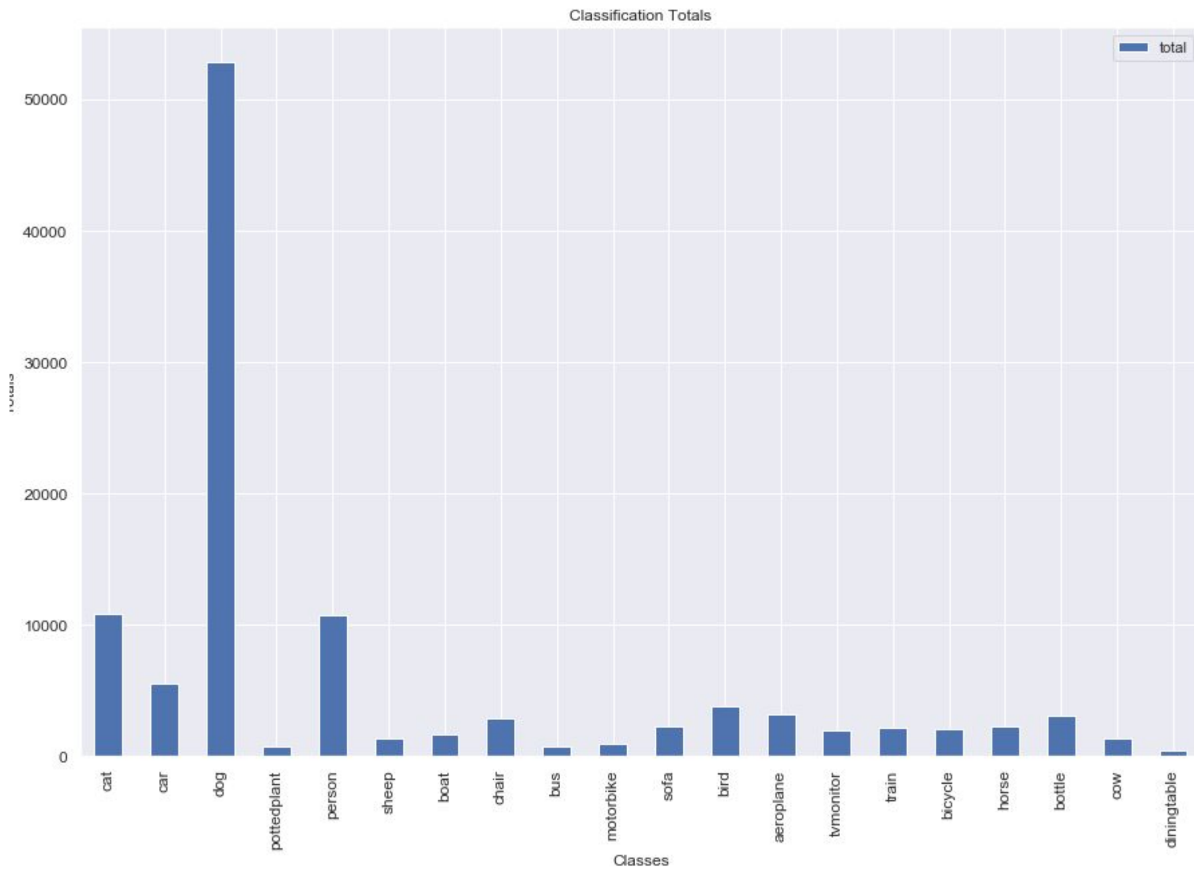
Storytelling and Inferential Statistics

For a deeper analysis, a TSNE grid was created for the aeroplane class to demonstrate visual similarities. Upon inspection, duplicates were detected and subsequently cleaned. Due to the size of the data, it was not possible to create an entire combined TSNE grid.

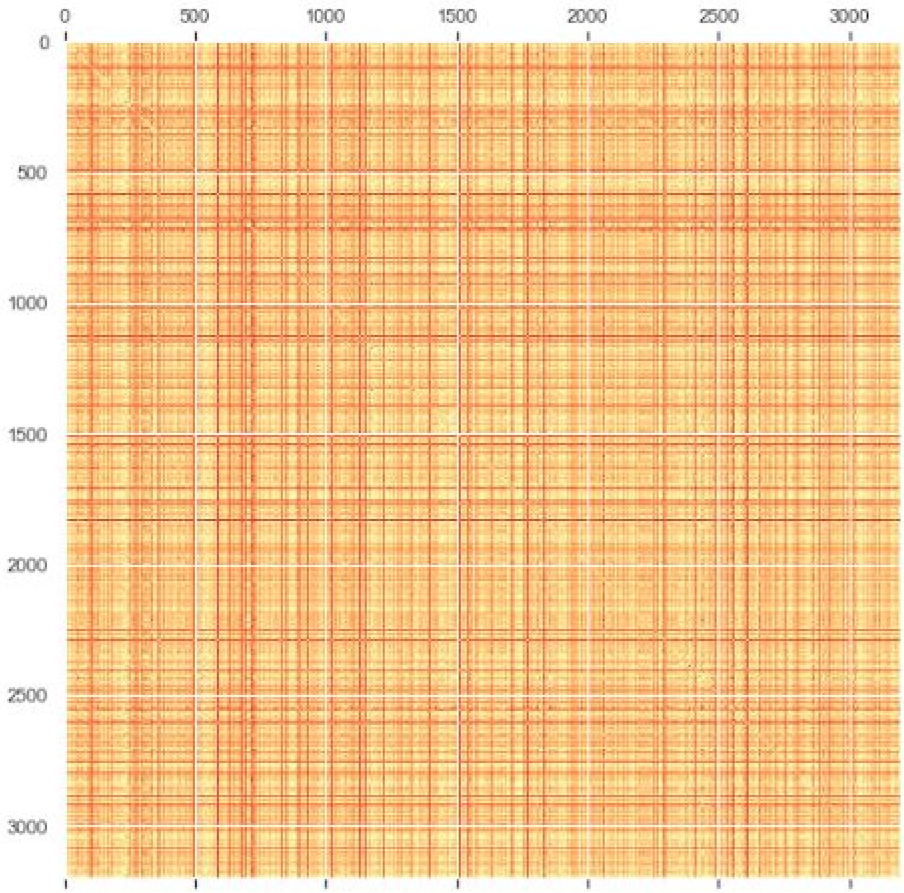


Inferential Statistics Investigation

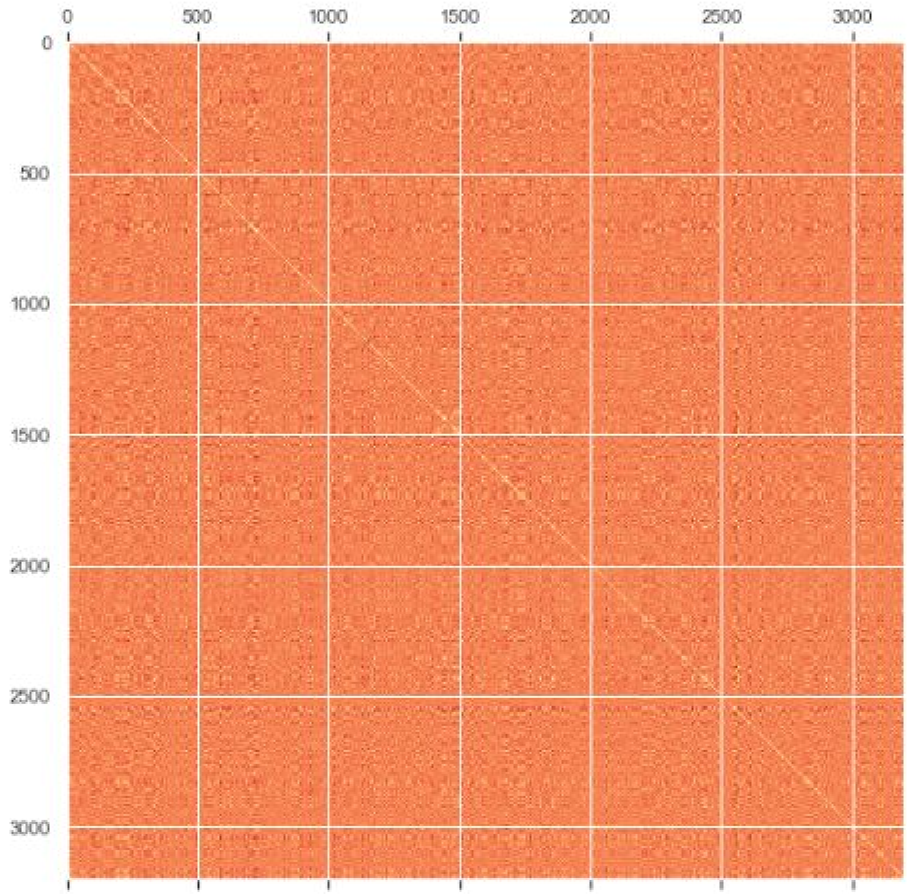
The various classifications available along with the data for each classification varies. The dog class has over 50,000 instances. As seen in the graph below, the number of instances vary based on class.



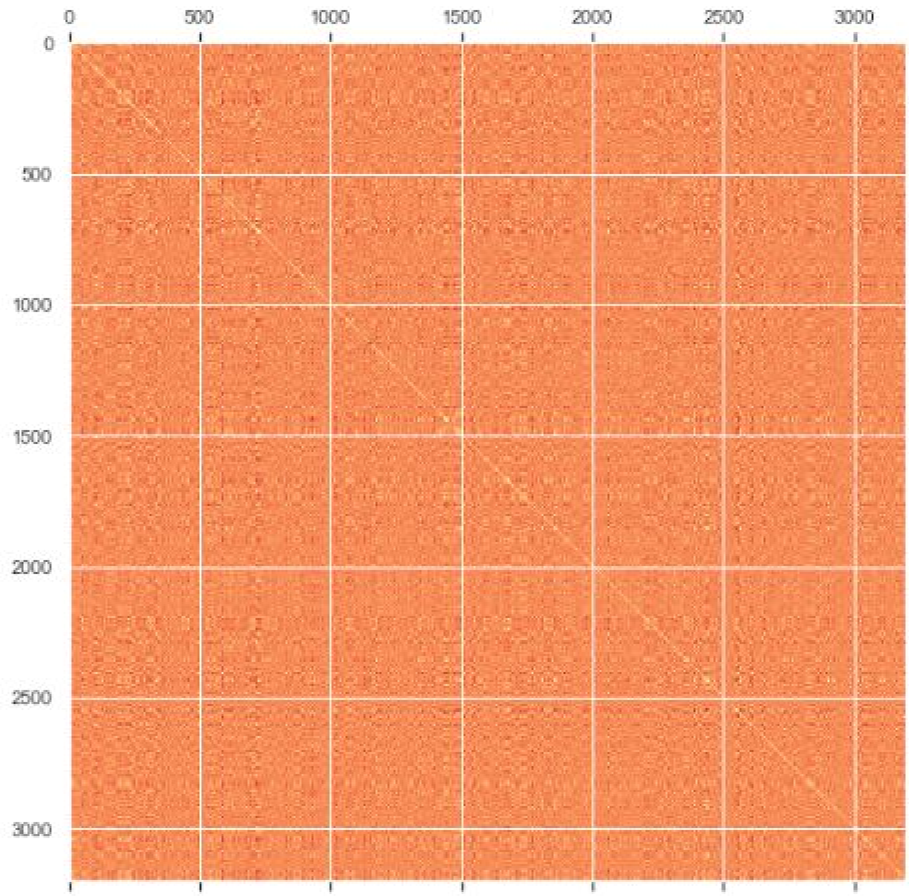
A covariance matrix of the aeroplane class was created with all of the images in the class. The shape of the image was 300 by 300 by 3 (RGB). The images were flattened before doing the comparison, which changed the shape to 270,000 columns.
Here is the first covariance with the flattened data.



Here is another covariance after the data is normalized around 0.



The images were then resized to 32 by 32 by 3 and normalized which produced the following matrix.



This matrix looks very similar to the covariance with data 300 by 300 by 3. Due to the size of the data and not having done any preprocessing the covariance data is inconclusive.

Baseline Analysis and Preliminary Results

The goal of the analysis is to establish a starting point and then consider whether results can be improved from there. The evaluation of the model will use accuracy along with performance metrics.

The original goal is to predict the correct class based on various images that have little to none pre-processing.

Convolutional Neural Network

A Convolutional Neural Network (CNN) is used as the model for the 110,604 images belonging to 20 classes.

The CNN used for this model is constructed of fourteen layers based on previous image classification projects. The model is sequential and leverages Keras.

1. 2D convolutional layer (32,3,3)
2. Activation layer (relu)
3. 2D max pooling layer
4. 2D convolutional layer (32,3,3)
5. Activation layer (relu)
6. 2D max pooling layer
7. 2D convolutional layer (64,3,3)
8. Activation layer (relu)
9. 2D max pooling layer
10. Flatten layer
11. Dense layer
12. Activation layer
13. Dropout layer
14. Dense layer (top layer) using softmax

The CNN ran with 50 epochs which took ~5,600 seconds for each epoch. This took ~78 hours to process the entire 110,604 images of 20 classes.

The data was split into a test and train segment with 20% of the data used for validation.

Here are a few of the results from the epics:

Epic	**Time**	**Loss**	**Accuracy**
1	5377s	0.1319	0.9589
25	5374s	0.1403	0.9554
40	5400	0.1414	0.9544
50	5407	0.1381	0.9562

Once the CNN completed running, the weights were saved to a file for further use, if necessary.

The accuracy rate for the model was roughly 95%. That is really good for accuracy without any optimization. The next step is a classification report to understand how each class performed.

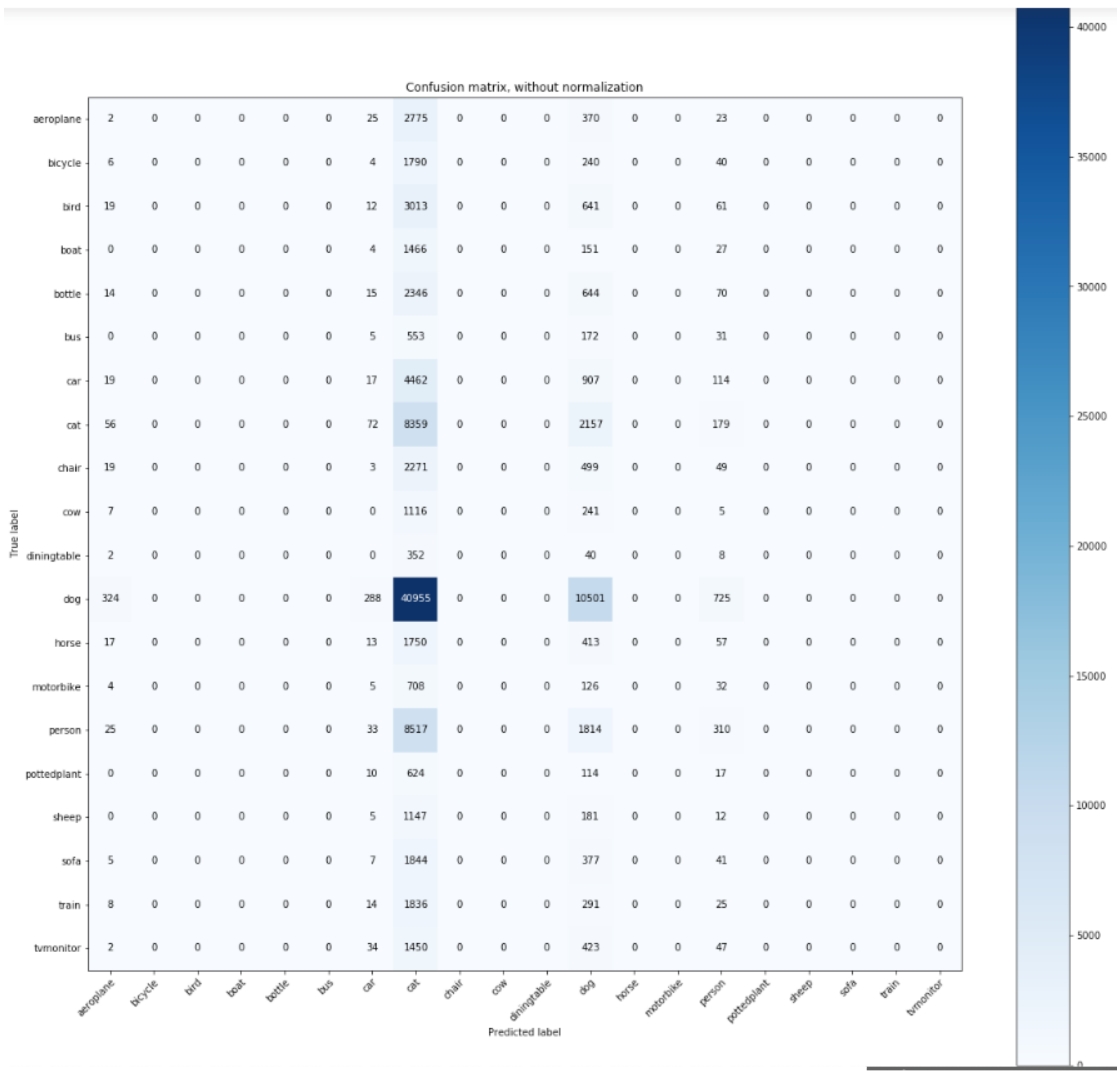
A classification report was generated to further understand how the model performed.

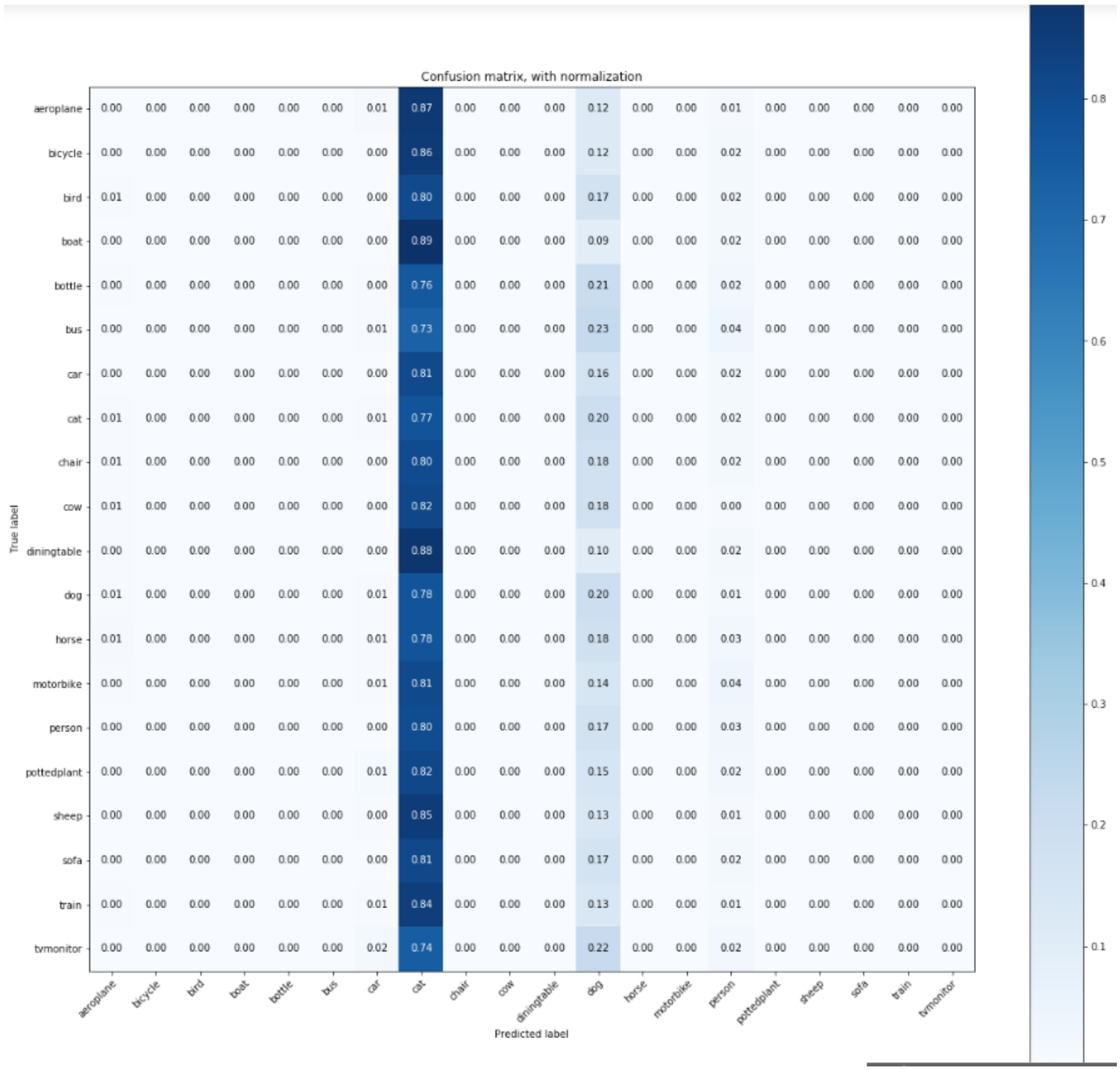
Labels	Precision	Recall	F1-Score	Support
aeroplane	0.00	0.00	0.00	3195
bicycle	0.00	0.00	0.00	2080
bird	0.00	0.00	0.00	3746
boat	0.00	0.00	0.00	1648
bottle	0.00	0.00	0.00	3089
bus	0.00	0.00	0.00	761
car	0.03	0.00	0.01	5519
cat	0.10	0.77	0.17	10823
chair	0.00	0.00	0.00	2841
cow	0.00	0.00	0.00	1369
diningtable	0.00	0.00	0.00	402
dog	0.52	0.20	0.29	52793
horse	0.00	0.00	0.00	2250
motorbike	0.00	0.00	0.00	875
person	0.17	0.03	0.05	10699
pottedplant	0.00	0.00	0.00	765
sheep	0.00	0.00	0.00	1345
sofa	0.00	0.00	0.00	2274
train	0.00	0.00	0.00	2174
tvmonitor	0.00	0.00	0.00	1956
Micro Avg	0.17	0.17	0.17	110604

Macro Avg	0.04	0.05	0.03	110604
Weighted Avg	0.27	0.17	0.16	110604

The results of the classification report are concerning due to zeros for almost all classes. The cat class had the best recall of 0.77 and dog had the best prediction of 0.52.

A confusion matrix was created in a non-normalized and normalized to further understand the data.





The confusion matrix is showing the issue that was indicated in the classification report. A proper distribution would have a diagonal approach from the upper left to lower right corner. The normalized confusion matrix indicates that the cat and dog class are skewing the results with the cat class being the drastic case.

Extended Analysis and Final Results

Due to the poor performance of the base model, the CNN was adjusted along with the loss function and metrics used for the model. The model was rerun with the following layers:

1. 2D convolutional layer (32,3,3)

2. Activation layer (relu)
3. 2D convolutional layer (64,3,3)
4. 2D max polling layer
5. Dropout layer (.25)
6. Flatten layer
7. Dense layer (128)
8. Dropout layer (0.5)
9. Top dense layer

The model now leverages SGD for the optimizer versus Adam.
The CNN ran with 1 epoch which took 24,672 seconds for the epoch. This took ~8 hours to process the entire 110,604 images of 20 classes.

The data was split into a test and train segment with 20% of the data used for validation.

Here are a few of the results from the epic:

Epic	**Time**	**Loss**	**Accuracy**
1	24,672	8.4247	0.4772

Once the CNN completed running, the weights were saved to a file for further use, if necessary.

The accuracy rate for the model was roughly 47%. That is okay for accuracy without any optimization. The next step is a classification report to understand how each class performed.

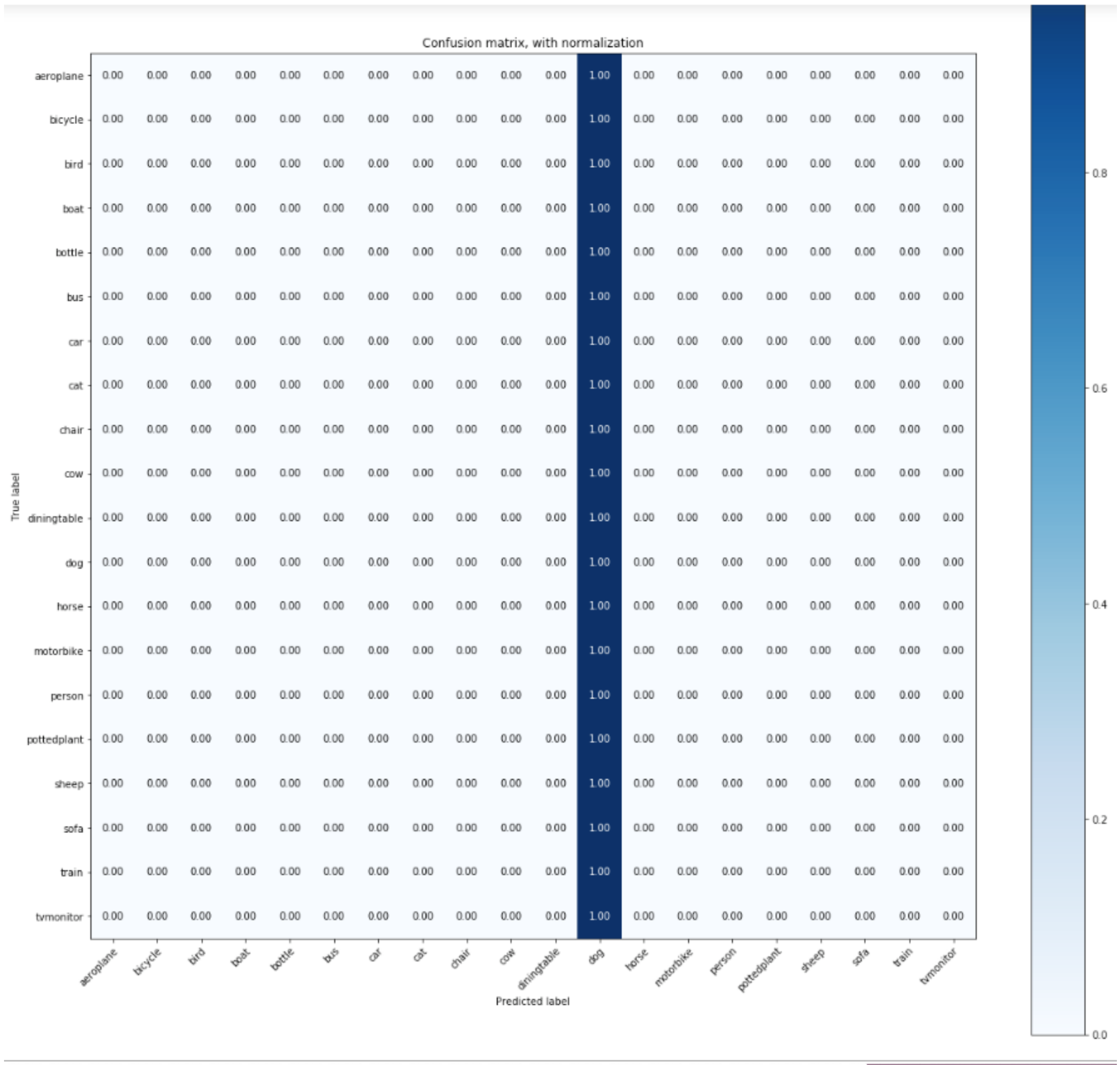
A classification report was generated to further understand how the model performed.

Labels	Precision	Recall	F1-Score	Support
aeroplane	0.00	0.00	0.00	3195
bicycle	0.00	0.00	0.00	2080
bird	0.00	0.00	0.00	3746
boat	0.00	0.00	0.00	1648
bottle	0.00	0.00	0.00	3089
bus	0.00	0.00	0.00	761

car	0.00	0.00	0.00	5519
cat	0.00	0.00	0.00	10823
chair	0.00	0.00	0.00	2841
cow	0.00	0.00	0.00	1369
diningtable	0.00	0.00	0.00	402
dog	0.48	1.00	0.65	52793
horse	0.00	0.00	0.00	2250
motorbike	0.00	0.00	0.00	875
person	0.00	0.00	0.00	10699
pottedplant	0.00	0.00	0.00	765
sheep	0.00	0.00	0.00	1345
sofa	0.00	0.00	0.00	2274
train	0.00	0.00	0.00	2174
tvmonitor	0.00	0.00	0.00	1956

The results of the classification report are concerning due to zeros for almost all classes. The dog class had the best recall of 1.00.

A confusion matrix was created normalized to further understand the data.



Additional Testing V1

Due to the poor performance of the extended model, the CNN was adjusted along with only running 2 models: Dog and Cat. The model was rerun with the following layers:

1. 2D convolutional layer (32,3,3)
2. Activation layer (relu)
3. 2D convolutional layer (32,3,3)
4. 2D max pooling layer
5. Dropout layer (.2)
6. 2D convolutional layer (64,3,3)

7. Activation layer (relu)
8. 2D convolutional layer (64,3,3)
9. 2D max pooling layer
10. Dropout layer (.2)
11. 2D convolutional layer (128,3,3)
12. Activation layer (relu)
13. 2D convolutional layer (128,3,3)
14. 2D max pooling layer
15. Dropout layer (.2)
16. Flatten layer
17. Dense layer (128)
18. Dropout layer (0.5)
19. Top dense layer

The model leverages SGD for the optimizer.
The CNN ran with 3 epochs which took ~24,672 seconds per epoch. This took ~8 hours to process the entire 110,604 images of 2 classes.

The data was split into a test and train segment with 20% of the data used for validation.

Here are the results from the epics:

Epic	**Time**	**Loss**	**Accuracy**
1	9864	0.4391	
2	8701	0.3967	
3	9248	0.3680	

Once the CNN completed running, the weights were saved to a file for further use, if necessary.

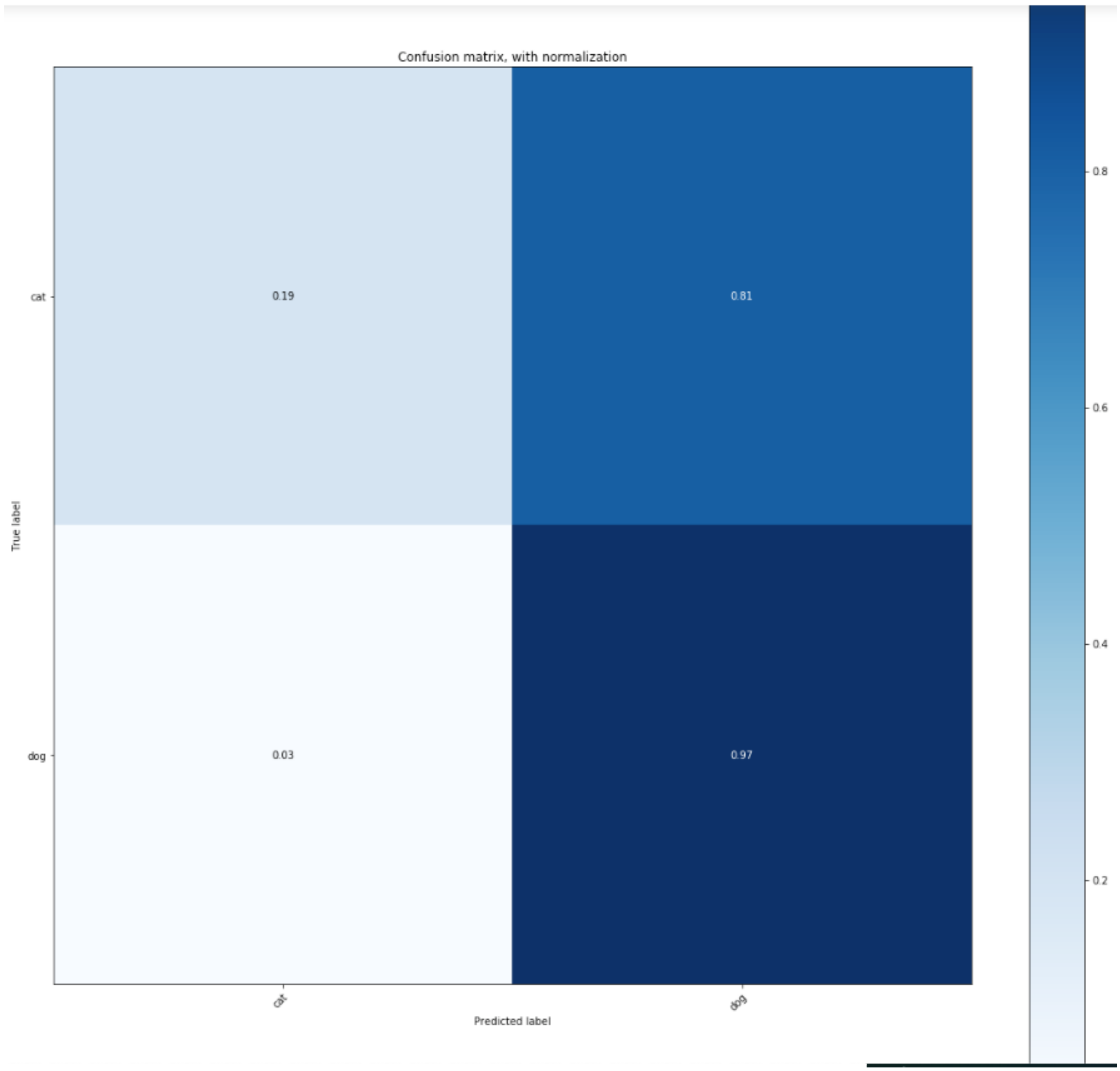
Due to the model change, accuracy was not reported. All results are based on the classification report below.

A classification report was generated to further understand how the model performed.

Labels	Precision	Recall	F1-Score	Support
cat	0.59	0.19	0.29	10823
dog	0.85	0.97	0.91	52793

The results of the classification report are looking better than before. However, the cat class is still low.

A confusion matrix was created normalized to further understand the data.



Additional Testing V2

Due to the poor performance of the previous model, the CNN is the same as V1. The images now are the same amount ensuring there is not an imbalance in the classes. The model was rerun with the following layers:

1. 2D convolutional layer (32,3,3)

2. Activation layer (relu)
3. 2D convolutional layer (32,3,3)
4. 2D max pooling layer
5. Dropout layer (.2)
6. 2D convolutional layer (64,3,3)
7. Activation layer (relu)
8. 2D convolutional layer (64,3,3)
9. 2D max pooling layer
10. Dropout layer (.2)
11. 2D convolutional layer (128,3,3)
12. Activation layer (relu)
13. 2D convolutional layer (128,3,3)
14. 2D max pooling layer
15. Dropout layer (.2)
16. Flatten layer
17. Dense layer (128)
18. Dropout layer (0.5)
19. Top dense layer

The model leverages SGD for the optimizer.
The CNN ran with 3 epochs which took ~2,847 seconds per epoch. This took ~5 hours to process the entire 21,646 images of 2 classes.

The data was split into a test and train segment with 20% of the data used for validation.

Here are the results from the epics:

Epic	**Time**	**Loss**	**Accuracy**
1	2919	0.6938	0.4998
2	2878	0.6942	0.5002
3	2867	0.6935	0.4993
4	2759	0.6931	0.5023
5	2847	0.6932	0.4953

Once the CNN completed running, the weights were saved to a file for further use, if necessary.

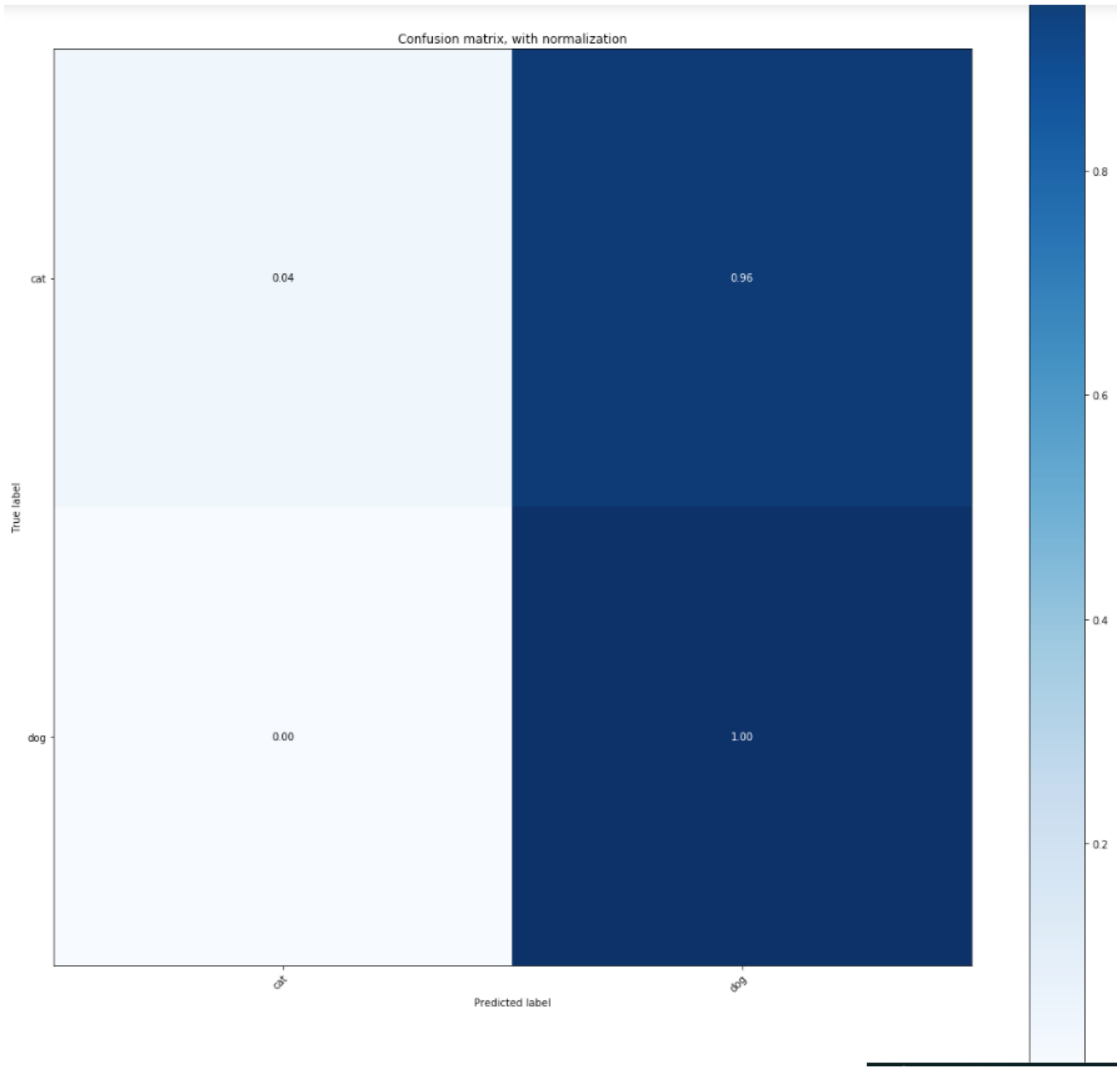
The accuracy ranged from 49% to 50% accurate.

A classification report was generated to further understand how the model performed.

Labels	Precision	Recall	F1-Score	Support
cat	0.94	0.04	0.07	10823
dog	0.51	1.00	0.67	10823

The results of the classification report are looking better than before. However, the recall rate for cat is still poor.

A confusion matrix was created normalized to further understand the data.



Conclusions and Future Work

After running various models the dog and cat classes tend to skew the results which could be related to an imbalance in data on the dog and cat classes. The CNN does well in accuracy once the correct model is selected. Further work needs done on data augmentation and photo preprocessing. Leveraging the CNN for this work is a viable option.

Future Work

Additional work can be done to improve the results further. The following areas of study are suggested:

1. Additional photo pre-processing
2. Hyper parameter tuning of layers
3. Try different network architectures. Try deeper and shallower networks.
4. Try adding BatchNormalization layers to the network.
5. Experiment with different weight initializations
6. Try different learning rates and schedules
7. Make an ensemble of models
8. Try normalization of input images
9. Data augmentation

Recommendations for Client

The client should continue to pursue perfection of the CNN as 95% accuracy is possible. The images of the animals and preprocessing those is an important factor that needs resolved along with productionizing the entire process.

Resources / Resources Used

The following is a list of the various resources used in this study.

1. https://github.com/DeadBigRedDog/animal-classification
2. https://jdhao.github.io/2017/11/06/resize-image-to-square-with-padding/
3. https://machinelearningmastery.com/how-to-load-and-manipulate-images-for-deep-learning-in-python-with-pil-pillow/
4. https://datascience.stackexchange.com/questions/29223/exploratory-data-analysis-with-image-datset
5. https://thebittheories.com/t-sne-visualization-of-instagram-posts-d5915ae99e63
6. https://github.com/prabodhhere/tsne-grid/blob/master/tsne_grid.py
7. https://github.com/shivanichander/tSNE/tree/master/Code
8. https://cloudkul.com/blog/mounting-s3-bucket-linux-ec2-instance/
9. https://github.com/IBM/image-classification-using-cnn-and-keras/blob/master/notebooks/Image%20Classification%20of%20Documents.ipynb
10. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
11. https://gist.github.com/fchollet/0830affa1f7f19fd47b06d4cf89ed44d
12. https://gist.github.com/fchollet/f35fbc80e066a49d65f1688a7e99f069
13. https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3
14. https://www.dlology.com/blog/how-to-use-keras-sparse_categorical_crossentropy/
15. https://chsasank.github.io/keras-tutorial.html