MSDS 458 Deep Learning
Juan Campos
Assignment Final Project
8/31/2019

1. Executive Summary

This analysis builds on the research conducted in Assignment 2/3. The goal of Assignment 2/3 was to gain practical experience and an understanding of designing, training, and evaluating a CNN. Using a dataset of cat and dog images, an image classification CNN that distinguishes between cats and dogs was constructed. The constructed CNN was unable to discriminate between the two classes. It was concluded that the complexity of both the dataset and the CNN architecture used in the study adversely impacted the CNN's performance.

This phase of the research sets out to examine the capabilities of a stripped-down Visual Geometry Group -16 (VGG-16) Convolutional Neural Network (CNN) architecture and to determine if the model's classification accuracy can be improved by using a reduced architecture and images with less clutter/noise. The results of this study confirm that a simpler approach can improve the accuracy of the CNN.

2. Data Preparation, Exploration, Visualization

The images were obtained from the Kaggle Dogs vs. Cats Competition. The original dataset contains 25,000 images. The images are in color and 280 x 300 in size. Each image is labeled with the word cat or dog and a sequential number appendix (i.e. cat.01, dog.4094). The animals are displayed in varying positions and angles on the images. Some images show the full body of the animals while others show only a headshot (viewpoint variation). Different breeds and a myriad of colors of the cats and dogs are represented in the images.

Most of the images in the dataset also contain noise in the form of clutter in the environment. For instance, blankets with designs, cages, leashes, human arms and hands, and even human faces appear in the images (Figure 1). In many of the images, the animals are also occluded. Therefore only a few pixels in the image are associated with the object of interest. Prompted by the conclusions of Assignment 2/3, an examination of the dataset also confirmed that some of the images were incorrectly labeled and assigned to the wrong class.

Figure 1: Example of noise in the images

To overcome and mitigate some of the challenges with the images, 300 images were manually selected for this analysis with the focus of reducing the complexity of the dataset. Images with minimal background clutter; and images containing headshots or showing the full body of the animals were selected. However, there is an imbalance in the distribution of the images in the dataset. The images containing headshots exceed the number of images that show the full bodies. Figure 2 displays examples of the images in the dataset. Finally, to improve the algorithm's ability to generalize the images, the CNN algorithm was programmed to randomly flip, zoom, or angle the training set images (Figure 3).

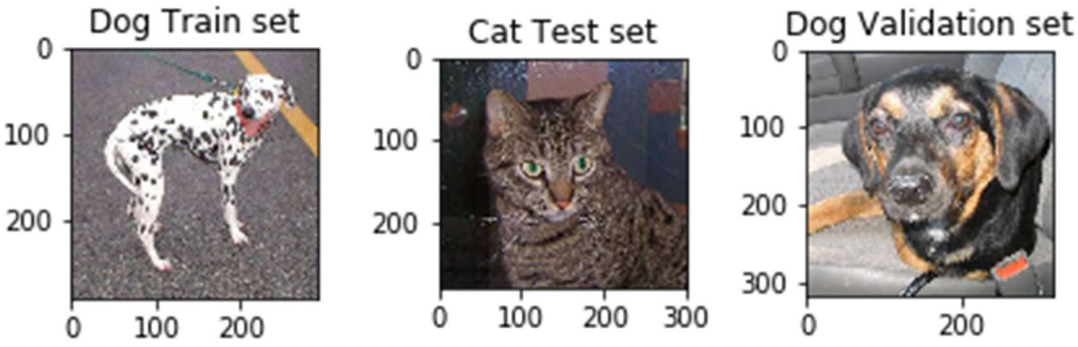


Figure 2: Example of dataset images

Figure 3: Example random flipping, zoom-in, and angling of images

The dataset was split into three sub-datasets for training of the CNN and testing and validating its classification ability. The split consists of 200 Train images, 50 Test images, and 50 Validate images. The two classification categories are represented 50/50 in each of the sub-datasets. The labels of the images were set as factor type with the values: 'cat' or 'dog'.

3. Implementation and Modeling

The Python code and the Jupyter Notebook environment were used to conduct the study. Several Python packages were used to conduct the study with the packages Keras and the Tensorflow backend being employed to perform the Deep Learning tasks such as training the CNN.

A modified version of the Visual Geometry Group -16 (VGG-16) CNN architecture was selected for Assignment 2/3. This network architecture uses an input size of 224 x 224. The input size of the images used in the study was

downsized to 128 x 128, converted to gray-scale, and processed in batches to satisfy computer demands. Other parameters that are distinctive of the VGG-16 architecture are that it has a 3 x 3 receptive field; a convolution stride of 1; padding of 1 for the receptive field of 3 x 3; five max pooling layers of 2 x 2 with a 2 pixel stride, five convolution layers followed by two fully connected layers (first connected layer has 4096 channels and the second has 1,000 channels, one for each class), the activation function of ReLU, and the final layer is a Softmax classification layer. All hidden layers are equipped with ReLU.

In this phase of the study, three models were constructed using three levels of simplification of the original VGG-16 model. The first model (Model 1) contains one CNN block with 3 x 3 x 64 CNN layer. The second model (Model 2) uses one CNN block with 3 x 3 x 64 CNN layers. The final model (Model 3) contains two CNN blocks starting with a width of 3 x 3 x 64 and ending with a width of 128 channels (3 x 3 x 128) and a total of four CNN layers. Two formulas were applied to the steps_per_epoch and the validation_steps parameters. The steps are the number of batch iterations performed before a training epoch is completed. The formula total train samples (200)/total batches (32) were utilized for Model 1. This resulted in 6 steps per epoch being performed. The formula len(total train samples) was used for Models 2 and 3 and resulted in 7 steps per epoch being performed. The models were run using 1,000 epochs for Models 1 and 2 and 25 epochs for Model 3.

Because regularization worsened the accuracy of the models in Assignment 2/3, regularization was not implemented in this analysis. To force all of the nodes to learn a portion of the input, a dropout rate of .50 was applied to each convolutional layer block. In addition, early stopping was applied to the model with a patience value of 10. This helps prevent the model from over-fitting. If the loss function does not decrease after 10 epochs, the model will stop learning and will move on. Ideally, this will help save training time as well. Finally, the learning rate was reduced each time that the accuracy of the model did not improve by 2. The RMSProp optimizer is used to minimize binary-cross-entropy for all models. Performance metrics were calculated and graphed. Finally, SHAP was employed to explain the output of the CNN.

4. Results

The early stopping feature prompted Model 1 to stop learning at epoch 22 and Model 2 to stop learning at epoch 35 because the loss function did not decrease after 10 epochs. Model 3 completed the 25 defined epochs.

The Assignment 2/3 model classification accuracy score settled between the range .50 and .52 for each run. The area under the ROC curve was also in the range of .50 and .52 after each run, indicating that the algorithms were not fitting the data well. Model's 1 and 2 performed similarly with each model achieving an Accuracy score of 52%, and an AUC score of .458 and .629, respectively. Model 3 showed improvement with an overall Accuracy score of 60% and an AUC score of .579. Figure 4 shows the performance metric graphs for Model 3.

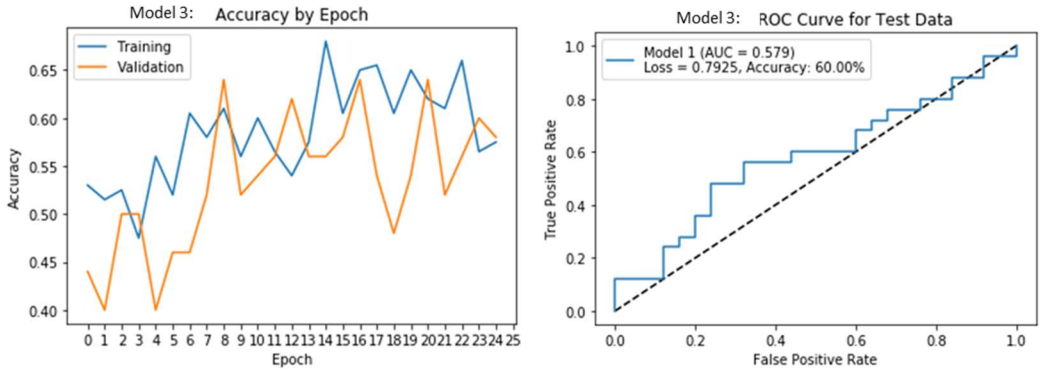


Figure 4: Model 3 ROC Curve Graph, Accuracy Score, and Accuracy Score History per Epoch

Figure 4 shows that the training accuracy fluctuated up and down with the lowest accuracy being about .4800 in epoch number 3 and the highest accuracy being approximately .6800 in epoch 14. The final Accuracy score achieved by epoch 25 was approximately .5700. The classification accuracy of the validation set was more erratic with very wide ranges between the increases and decreases in Accuracy score values. In epochs 1 and 4 the model's accuracy decreased from .5000 to .4000. By epoch 8, the model's accuracy increased to nearly .6500. The accuracy continued to rise and fall by large increments during the final two epochs ending with .5700 accuracy. Appendix A and Appendix B display

the performance metric graphs for Models 1 and 2. Although Model 2 achieved the highest AUC score of .6290, its overall Accuracy score was lower than Model 3's Accuracy score by 8 percentage points. Also, as can be seen in the Accuracy per Epoch graph in Appendix B, Model 2 began to immediately over-fit the data. Beginning in epoch 1, the validation accuracy was greater than the training accuracy and remained higher through the entire process.

CNN's are black-box models. It is difficult to obtain an understanding of what they learned due to the many deep layers. This is especially true of the deepest layers. Although these layers learn the higher-level features of the images, they are very far removed from the images. To help understand the CNN output, the Shapely values or SHAP was employed to explain the results. Figure 5 shows the SHAP values and graphs that help explain how Model 3 classified the images.

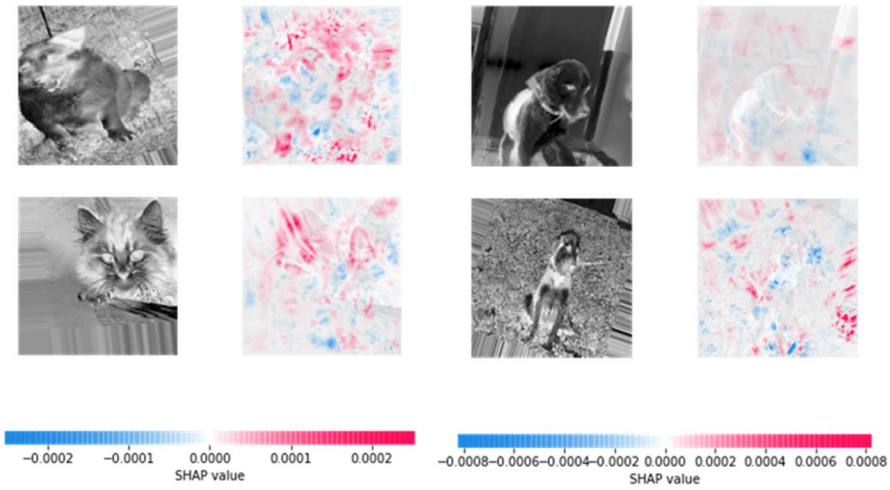


Figure 5: SHAP Value Graphs: Model 3

The images in the first and third columns are input images themselves. The images in the second and fourth columns display the SHAP analytics. The SHAP output suggests that the model was unable to detect/extract the object of interest from the images to help it distinguish between the two animals. Referring to the cat images, although the model was able to extract many features (highlighted in red), the extracted features pertain mainly

to the background objects and not the animal. For instance, the couch on the top cat image was almost completely highlighted in red while the majority of the cat was highlighted in gray and blue. This suggests that the CNN filters were unable able to 'see' the cat features in the image nor to isolate the animal from the background. The same is true for the second cat image on the bottom. As regards the dog images, the CNN was unable to extract many features on these images. Extracted features appear sporadic and are mainly background objects. Meanwhile, no features were extracted for the dog objects which are highlighted in mostly gray and blue colors indicating that the model was unable to extract features or assign credit to the images to classify them.

In examining the first convolutional layer (Figure 6), it does not appear that the CNN filters are detecting the edges nor separating the elements in the images. However, when examining the second CNN block (Figure 7), the model is beginning to detect the edges and to separate the elements in the images.

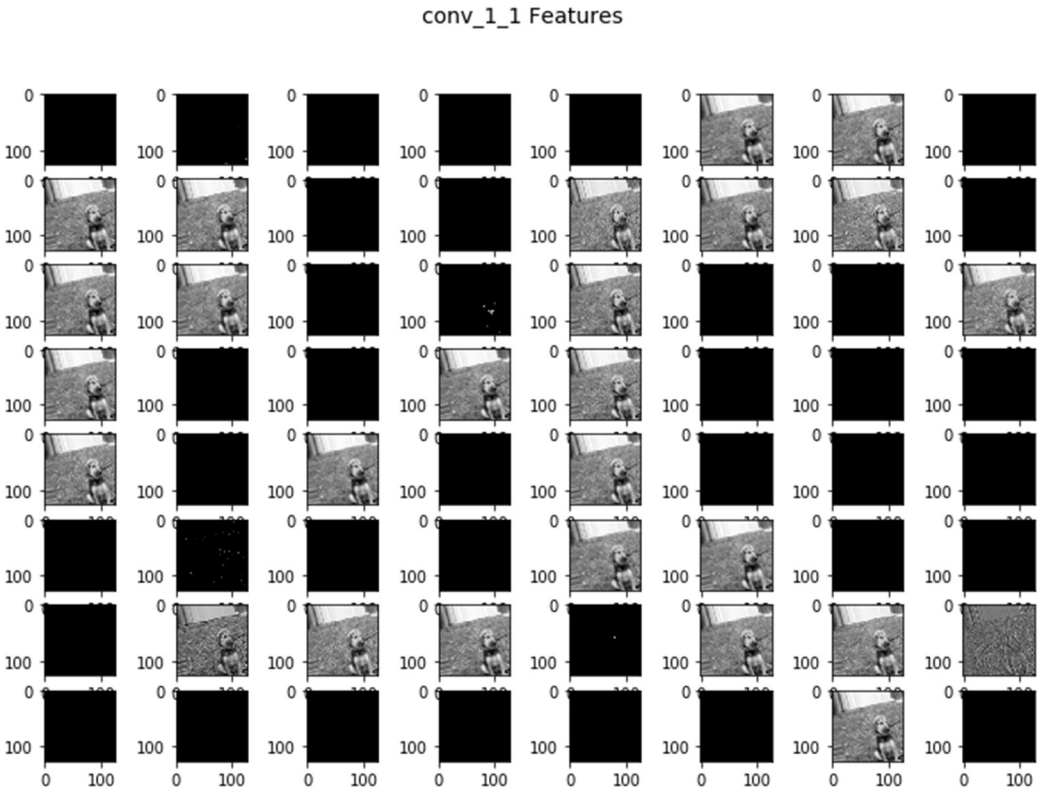


Figure 6: Convolution Layer 1 Features

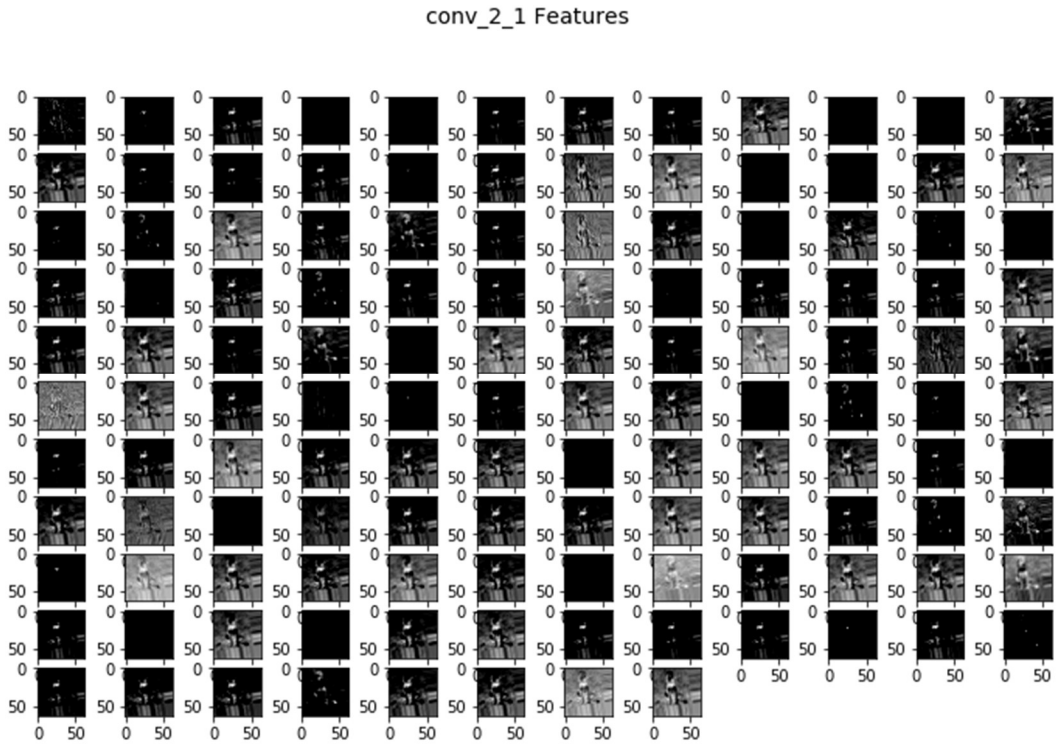


Figure 7: Convolution Layer 3 Features

5. Conclusion and Recommendations

In this study, the CNN continued to have difficulty distinguishing between the two classes. Two of the models were not able to distinguish between a cat or a dog more effectively than random guessing. The third model showed improved accuracy but performed poorly in discriminating between the cat and dog classifications. Nevertheless, this study satisfied the goal of determining whether or not the classification accuracy could be improved by using a simpler approach than that used in Assignment 2/3. Tuning the number of epochs and the steps_per_epoch parameter also helped improve the performance of the CNN.

As regards the image complexity, the models appear to find it more difficult to extract features from the dog images than the cat images. It is suspected that the high variance in dog morphology is a contributing factor. The noise/clutter; the scale and viewpoint variations; the occlusion of the animals; and the mislabeling errors in the dataset add to the complexity of the dataset and make it more difficult for the CNN algorithm to discern the object of interest. However, by manually selecting images that were believed to be easier to be read by the algorithm helped to slightly improve the accuracy of the CNN. But a wider dataset, with many dog breeds being represented, is needed to more adequately train the CNN. Finally, the algorithms were not able to extract the objects of interest from their background. The models treated the background objects as part of animals. To continue to improve the accuracy of the CNN, it is recommended that algorithms that remove the backgrounds (leaving only the object of interest visible) or object detection algorithms (image segmentation) be employed in future tests. Image segmentation should be performed with algorithms such as Mask Region-Based CNN (Mask R-CNN).

Bibliography

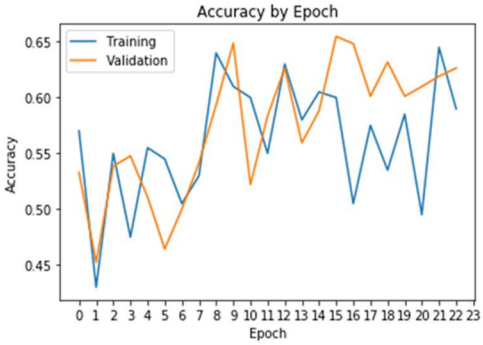
How to set batch_size, steps_per_epoch and validation steps. Retrieved from:

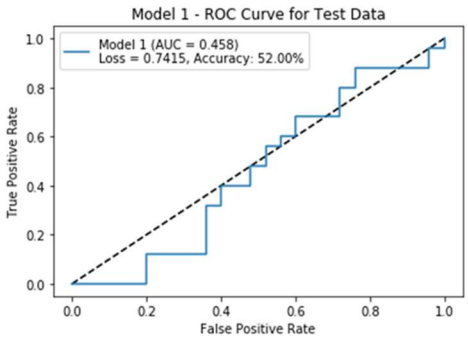
https://datascience.stackexchange.com/questions/29719/how-to-set-batch-size-steps-per-epoch-and-validation-steps

Extract from sourse:

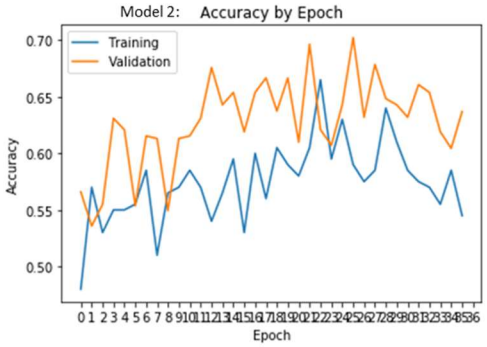
- **batch_size** determines the number of samples in each mini batch. Its maximum is the number of all samples, which makes gradient descent accurate, the loss will decrease towards the minimum if the learning rate is small enough, but iterations are slower. Its minimum is 1, resulting in stochastic gradient descent: Fast but the direction of the gradient step is based only on one example, the loss may jump around. batch_size allows to adjust between the two extremes: accurate gradient direction and fast iteration. Also, the maximum value for batch_size may be limited if your model + data set does not fit into the available (GPU) memory.
- **steps_per_epoch** the number of batch iterations before a training epoch is considered finished. If you have a training set of fixed size you can ignore it but it may be useful if you have a huge data set or if you are generating random data augmentations on the fly, i.e. if your training set has a (generated) infinite size. If you have the time to go through your whole training data set I recommend to skip this parameter.
- **validation_steps** similar to steps_per_epoch but on the validation data set instead on the training data. If you have the time to go through your whole validation data set I recommend to skip this parameter.

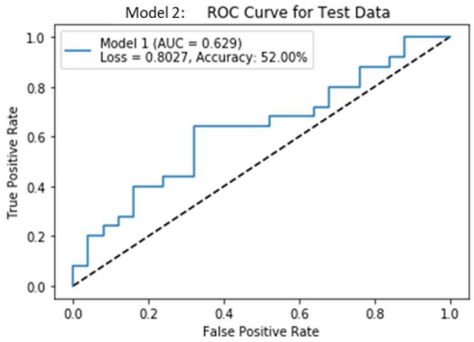
Appendix A


Accuracy by Epoch
Training
Validation
Accuracy
Epoch
0.65
0.60
0.55
0.50
0.45
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23


Model 1 - ROC Curve for Test Data
Model 1 (AUC = 0.458)
Loss = 0.7415, Accuracy: 52.00%
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.2 0.4 0.6 0.8 1.0

Appendix B


Model 2: Accuracy by Epoch
Training
Validation
Accuracy
Epoch


Model 2: ROC Curve for Test Data
Model 1 (AUC = 0.629)
Loss = 0.8027, Accuracy: 52.00%
True Positive Rate
False Positive Rate