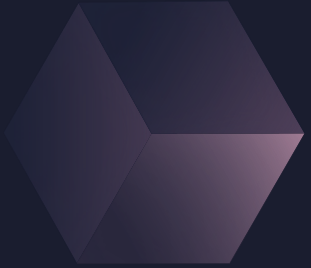


ML6: Convolutional Neural Networks

Manuel, Ma. Janelle G.

AP 186





objectives

The main objective of this activity is to classify images (dogs and cats) using CNN.

Improving the algorithm

Dogs were categorized as 1 while cats 0.

Training data: 18,000

Validation data: 4,500

For my algorithm, whose parameters were mostly based on an algorithm by [Uysim Ty](#), **3 convolutional layers were used**. The first layer was made to have 32 filters, while the second and third had 64 and 128, respectively. Ultimately, this made the model learn more complex features from the images as compared to the original algorithm thus **providing a better model performance**. However, training took SO long! Epoch was set to 15 and each epoch took ~18 minutes to run! The wait was worth it because at **epoch 1**, the accuracy was already equal to the accuracy the author achieved (75%).

Finally, after waiting for ~7 hours, the final accuracy is at 88% with validation loss of 23%.

```
Epoch 1/15
1333/1333 [=====] - 1345s 1s/step - loss: 0.5150 - acc: 0.7530 - val_loss: 0.5756 - v
Epoch 2/15
1333/1333 [=====] - 1314s 986ms/step - loss: 0.4758 - acc: 0.7773 - val_loss: 0.4842
08
Epoch 3/15
1333/1333 [=====] - 1283s 962ms/step - loss: 0.4487 - acc: 0.7945 - val_loss: 1.0691
69
Epoch 4/15
1333/1333 [=====] - 1273s 955ms/step - loss: 0.4247 - acc: 0.8092 - val_loss: 0.3612
34
Epoch 5/15
1333/1333 [=====] - 1118s 839ms/step - loss: 0.4073 - acc: 0.8194 - val_loss: 0.3608
08
Epoch 6/15
1332/1333 [=====>.] - ETA: 0s - loss: 0.3898 - acc: 0.8300
Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
1333/1333 [=====] - 1160s 870ms/step - loss: 0.3898 - acc: 0.8301 - val_loss: 0.4816
Epoch 7/15
1333/1333 [=====] - 1107s 831ms/step - loss: 0.3606 - acc: 0.8430 - val_loss: 0.2710
21
Epoch 8/15
1333/1333 [=====] - 1135s 852ms/step - loss: 0.3467 - acc: 0.8509 - val_loss: 0.2817
01
Epoch 9/15
1332/1333 [=====>.] - ETA: 0s - loss: 0.3289 - acc: 0.8590
Epoch 00009: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
1333/1333 [=====] - 1096s 822ms/step - loss: 0.3288 - acc: 0.8591 - val_loss: 0.2579
05
Epoch 10/15
1333/1333 [=====] - 1557s 1s/step - loss: 0.3222 - acc: 0.8632 - val_loss: 0.2670 - v
Epoch 11/15
1332/1333 [=====>.] - ETA: 1s - loss: 0.3115 - acc: 0.8683
Epoch 00011: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
1333/1333 [=====] - 1665s 1s/step - loss: 0.3116 - acc: 0.8683 - val_loss: 0.3059 - v
Epoch 12/15
1333/1333 [=====] - 1360s 1s/step - loss: 0.3012 - acc: 0.8705 - val_loss: 0.2239 - v
Epoch 13/15
1333/1333 [=====] - 1360s 1s/step - loss: 0.2961 - acc: 0.8764 - val_loss: 0.2316 -
Epoch 14/15
1333/1333 [=====] - 1245s 934ms/step - loss: 0.2936 - acc: 0.8748 - val_loss: 0.2191
23
Epoch 15/15
1333/1333 [=====] - 1415s 1s/step - loss: 0.2889 - acc: 0.8808 - val_loss: 0.2336 -
```

Figure 1. Python output for training accuracy

Results: Testing the model

The model with three convolution layers reached **an accuracy of 88%**. Sample predictions are shown below:

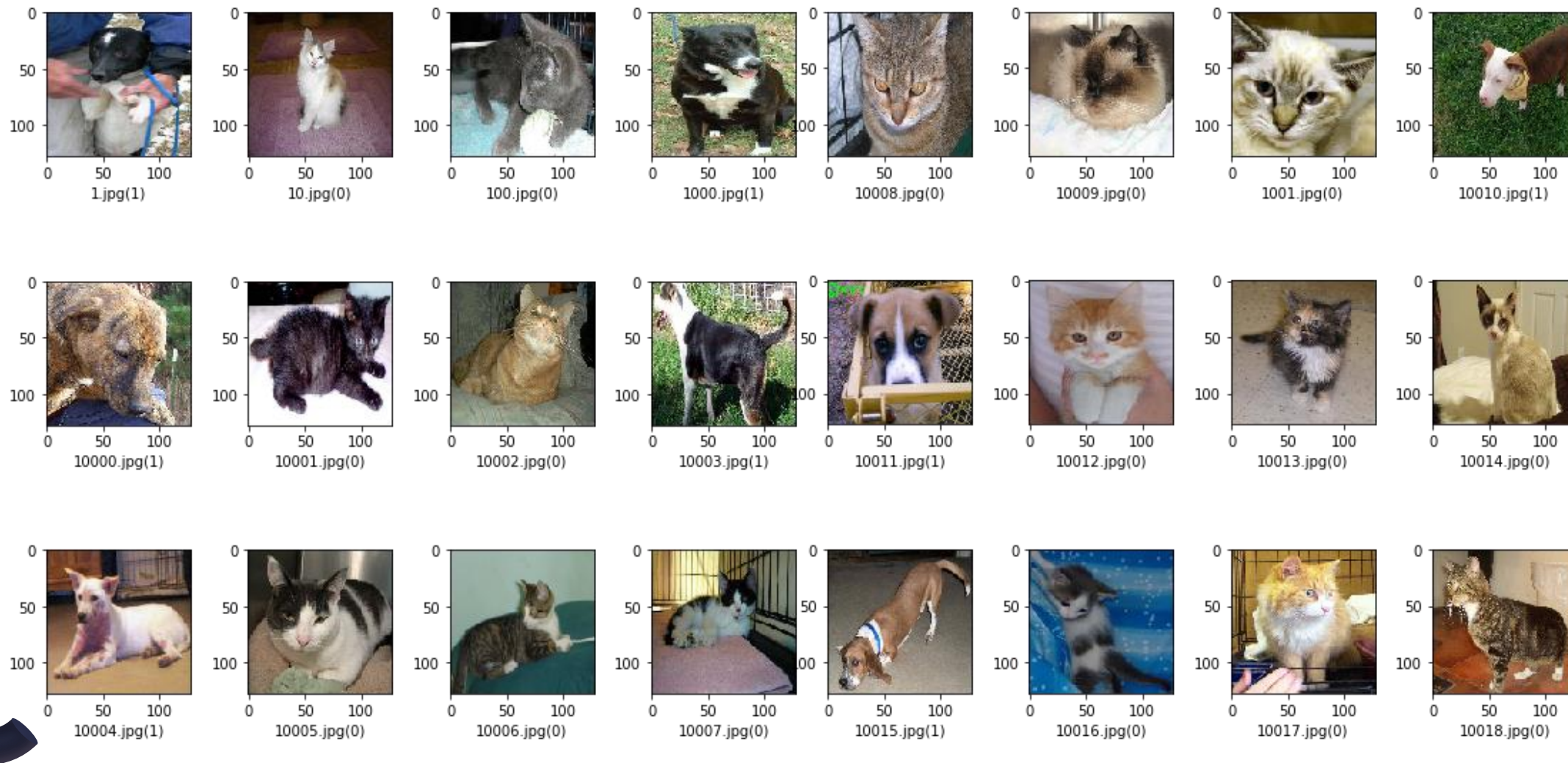


Figure 2. Randomly chosen images and their predicted category enclosed in (). 1: dog, 0: cat.

Analysis

Images were successfully classified without deriving extracted features using Convolutional Neural Networks.

The model's accuracy is highly dependent on how patient you are. A model with more convolutional layers with fully connected layers having more filters produces better results. The tradeoff here is that the better performance, the longer the training time.

If there are time constraints, it is better to increase the fully connected layers' size than to increase the quantity of convolutional filters.



Reflection

Rating: 12/10

This activity was overwhelming! Thankfully, the step by step tutorial provided was clear and easy to follow. It was easy to check if the algorithm was correct since the tutorial provided expected results. I also love that I am 12,500 dog pictures richer now.



References:

Soriano, M. (2020). ML6 – Convolutional Neural Networks.

Algorithm: [Convolutional Neural Networks: A Python Tutorial Using TensorFlow and Keras \(kdnuggets.com\)](https://www.kdnuggets.com/2016/05/convolutional-neural-networks-a-python-tutorial-using-tensorflow-and-keras.html)

Uysim Ty's algorithm: <https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification/comments#Prepare-Traning-Data>

