EE-628-WS Fall 2020 Final Project Report

Prem Patel

12/17/2020

—

Course: EE628-WS

—

Professor: Rensheng Wang

<u>Algorithm used:</u>

CNN is easily the most popular type of neural network for image classification. These models are ubiquitous for computer vision tasks like image classification, edge or object detection, and image recognition.

We are using three layer and a fully connected layer to obtain the probabilities of images being a dog. Each layer consists of Convolution, BatchNorm, Pooling, and Drop Out variables which are tweaked to obtain the best accuracy of the model.

The first three layers and for feature extraction while the fully connected layer is the classifier.

<u>Parameters tried and changed:</u>

1. <u>Image size:</u>

   To maintain consistence throughout the images, they were resized to a standard 128x128 dimensions. Increasing this number would tremendously increase the computational power.
2. <u>Epoch:</u>

After a lot of trial and error, the final epoch number set is 10. For after processing and for testing, FAST_RUN variable can be set true and it will run the model at 3 epochs. Current model's accuracy at 10 epochs is 84.4% but this can be increased by increasing the number of epochs. However, the mentioned number of epochs was chosen considering the computational power (GPU) that my machine has. EarlyStop function was added to ensure the over fitting of the model if more than 10 epochs are used.

3. Learning Rate:

   Instead of trying multiple learning rates, *ReduceLROnPlateau* was used to reduced the learning rate if the accuracy wasn't increasing by 2 steps.

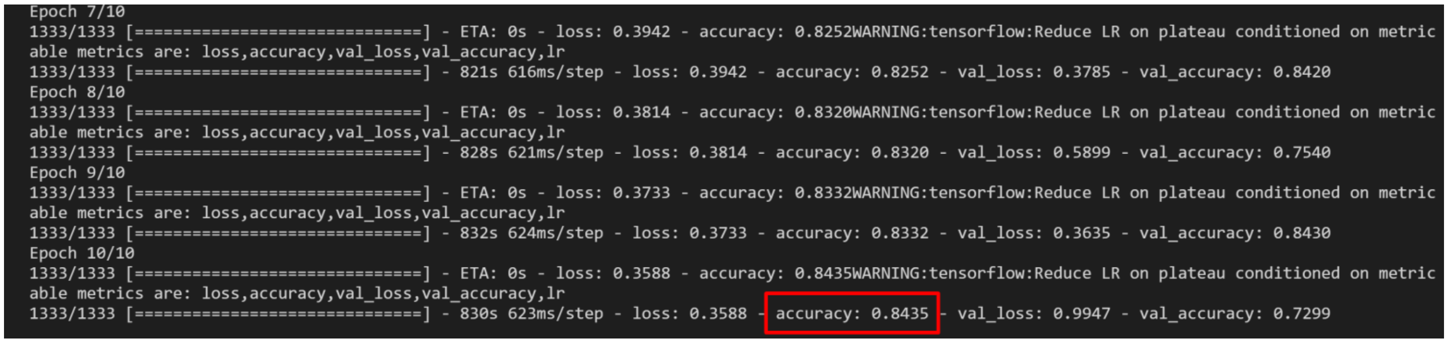
Process:

1. Defining constants, like FAST_RUN – for testing purposes, and other parameters like image size
2. Changing categorized name of the files into category numbers under training data. Dog = 1; Cat = 0
3. Converting an image into a plot by using *load_img* to get pixel values.
4. Building a model and defining parameters of each layer like activation function, Dropout, Compilation loss, compilation optimizer, etc.

5. Defining early stop to prevent overfitting of data if *val_loss* isn't reduced after 10 epochs but this wasn't used as 10 epochs were used to train the model.
6. Function to tweak the learning rate if the accuracy wasn't increasing by 2 steps.
7. Preparing data before feeding it into our model: Categorical names were used instead of categorical numbers as *class_mode='categorical'* is used.
8. Fitting the training model – Epoch 10 was used and the final accuracy score if the model is 84.35%
9. Prediction using the given model. Since the model gives out the probability of the image being a dog, numpy average max was used.
10. Predictions were then converted back into 0 and 1 and the file is saved as a csv.
11. Visualizing a few images as they were classified.

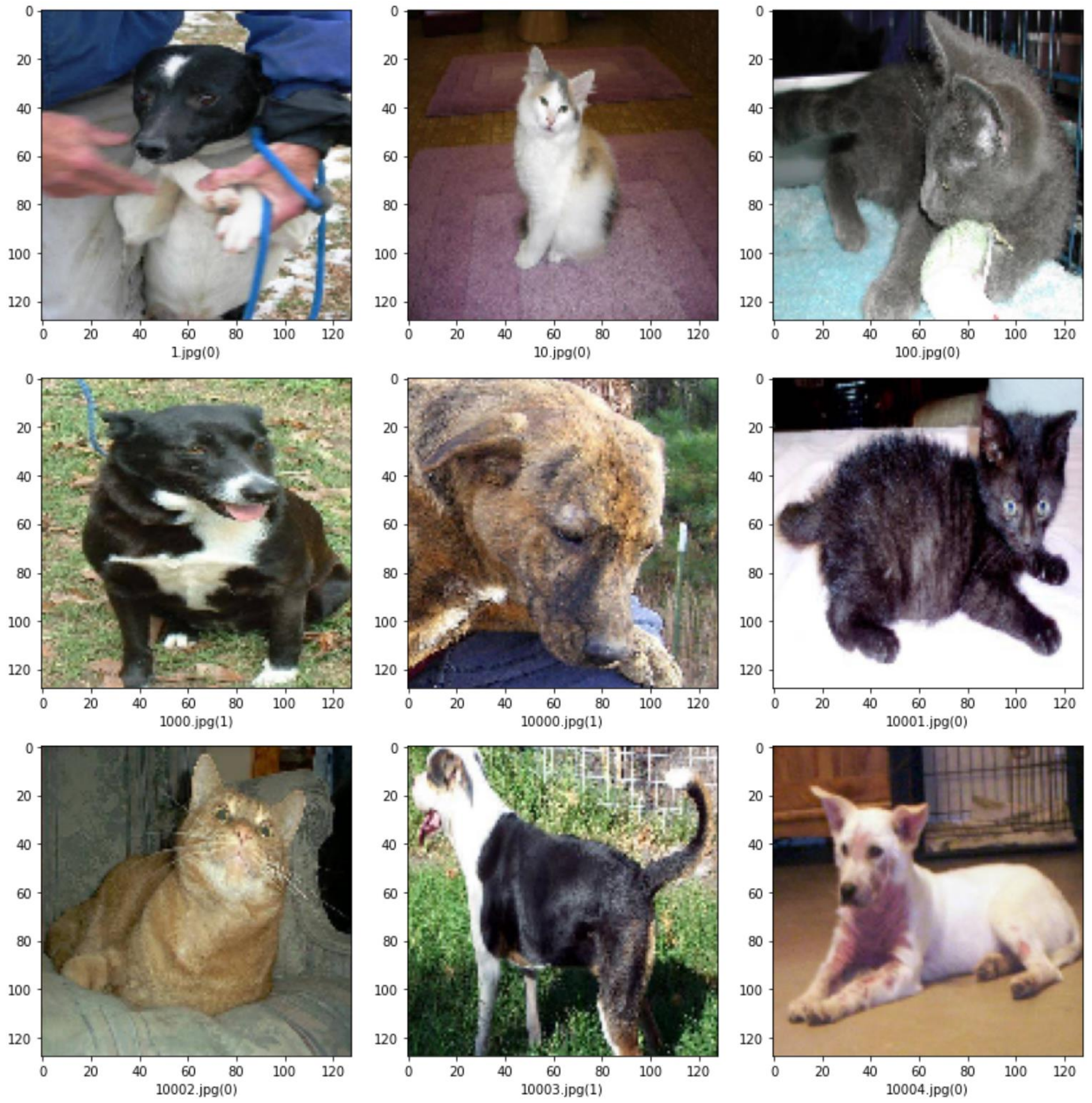
Results Obtained:

Below is the screenshot for the final epochs and highlighted is the final accuracy obtained of the model:



```
Epoch 7/10
1333/1333 [==============================] - ETA: 0s - loss: 0.3942 - accuracy: 0.8252WARNING:tensorflow:Reduce LR on plateau conditioned on metric
able metrics are: loss,accuracy,val_loss,val_accuracy,lr
1333/1333 [==============================] - 821s 616ms/step - loss: 0.3942 - accuracy: 0.8252 - val_loss: 0.3785 - val_accuracy: 0.8420
Epoch 8/10
1333/1333 [==============================] - ETA: 0s - loss: 0.3814 - accuracy: 0.8320WARNING:tensorflow:Reduce LR on plateau conditioned on metric
able metrics are: loss,accuracy,val_loss,val_accuracy,lr
1333/1333 [==============================] - 828s 621ms/step - loss: 0.3814 - accuracy: 0.8320 - val_loss: 0.5899 - val_accuracy: 0.7540
Epoch 9/10
1333/1333 [==============================] - ETA: 0s - loss: 0.3733 - accuracy: 0.8332WARNING:tensorflow:Reduce LR on plateau conditioned on metric
able metrics are: loss,accuracy,val_loss,val_accuracy,lr
1333/1333 [==============================] - 832s 624ms/step - loss: 0.3733 - accuracy: 0.8332 - val_loss: 0.3635 - val_accuracy: 0.8430
Epoch 10/10
1333/1333 [==============================] - ETA: 0s - loss: 0.3588 - accuracy: 0.8435WARNING:tensorflow:Reduce LR on plateau conditioned on metric
able metrics are: loss,accuracy,val_loss,val_accuracy,lr
1333/1333 [==============================] - 830s 623ms/step - loss: 0.3588 - accuracy: 0.8435 - val_loss: 0.9947 - val_accuracy: 0.7299
```

After training the model, images were plotted with their predicted values. A snipped of the plot is found below where ‘0’ stands for a cat and ‘1’ stands for a dog. The below image shows the first nine images and their results.



For the available computational power, 10 epoch is a sweet spot to get a decent accuracy while it can be increased by increasing the epochs but it would need more time or computational power.