Experimental Report on CNN Model for dogs-vs-cats Classification Problem using Keras Framework

Purva Taranekar

April 2020

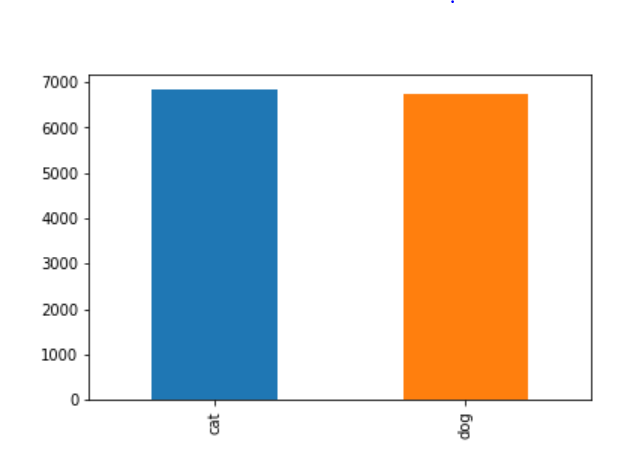
1. Problem Statement

The dogs-vs-cats problem is one of the well-known classification problems used in competitions and for practice as well. The input to the problem consists of train set of images of dogs and cats with true labels, and test set of images of dogs and cats whose label are to be predicted. Each image will contain either a cat or a dog but not both. The expectation from the programmer is to build a model (in this case, a CNN model), to predict whether the given image Is of a cat or a dog. The accuracy of prediction is to be greater than state-of-the-art. The experiment’s train loss, validation loss, validation accuracy and test accuracy are to be reported as solution of the problem.

1. Background

There have been a lot of classifiers build to distinguish cat and dog images. This problem demands the solution to the problem using Keras Framework. The dataset contains images of cats and dogs. These images are named with their category mentioned in filename, For example, dog.123.jpg implies the images contains dog and cat.123.jpg implies the images contains a cat.

The figure shows that equal number of dog and cat images are taken for training.



**Fig 1. Images distribution for train and validation**

Model has been generated to classify unseen test dataset correctly into cats and dogs. The model has been trained using layers of CNN. These are

* 1. Input Layer – The dataset is the input layer of the model. There are 17000 images in train set each with different sizes. Each image gets flattened into 1-D Array for easy computation.
  2. Convolution Layer – Library Conv2D is used to generate Convolution layer on train set.
  3. Pooling Layer – Pooling is done to reduce the size of output. Since each image size is arbitrary, we convert it into fixed size output size of 128 x 128 each. Among various pooling techniques, Max Pooling has been used here which is relatively computationally faster.
  4. ReLU has been used as an activation function
  5. At the last layer, softmax function has been used for predicting values.

The result of Prediction are probabilistic values denoting percentage of probabilities of an image belonging to each class. The larger probability is the winner and its class is assigned as the predicted class of the image.

For accuracy calculation, actual and predicted labels are observed and accuracy found using formula



To fit the model, we are using 3 epochs using training and validation data and results such as training loss, validation loss, validation accuracy are noted down.

1. Setup

For the purpose of experiment, 25000 samples of images were given as training data. Since test dataset’s actual labels were missing, I have taken part of the dataset, initially given for training, as test data. 8000 images were separated from training data for testing.

From remaining training data, 20% code has been reserved as validation and remaining used as training data.

Training images = 13600

Validation Images = 3401

Test Images = 8000

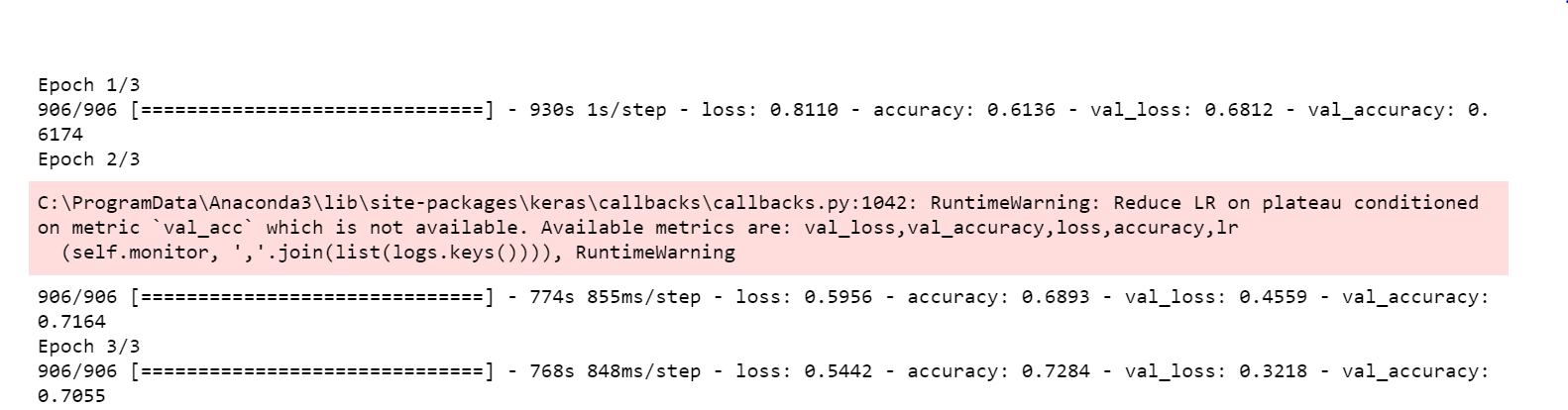
1. Conclusion

As per the problem statement, the results are mentioned in the following table.

|  |  |
| --- | --- |
| **Parameters** | **Experimental Values** |
| Training Loss | 0.72 |
| Validation Loss | 0.3218 |
| Validation Accuracy | 0.7055 |
| Testing Accuracy | 0.708 |

**Fig 2. Experiment Results at epoch = 3**

It is observed that, if number of epochs are increased, there will be improvements in accuracy and reduction in loss.



**Fig 3. Epoch Results showing loss and accuracy of train and validation set**