Machine Learning Engineer Nanodegree

Capstone Project

Build a Dogs Vs Cats Classifier

Ntepp Jean Marc
09/14/2017

I. Definition

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, education, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification, localization and detection. Recent developments in neural network (aka “deep learning”) approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. Fromdeeplearning4j neural network-overview, we can read a succinct definition of Deep Learning: Deep learning is the name we use for “stacked neural networks”; that is, networks composed of several layers.

The goal of this project is to build a cat/dog image classifier using Deep Learning algorithm called Convolutional Neural network. In other word, we will create an algorithm to distinguish dogs from cats. In this Capstone, we will be resolving a project from kaggle competition and using a dataset from Kaggle.

Problem Statement

Our goal is to build a machine learning algorithm capable of detecting the correct animal (cat or dog) in new unseen images. So we are trying to solve a classification problem.

1. In this project, I will write an algorithm to classify whether images contain either a dog or a cat.
2. Our algorithm has images as input and return whether image contains dogs or cats.

In this project we will apply Convolutional Neural Network(CNN) to predict whether images contains dogs or cats.

Metrics

We have used accuracy metric. Accuracy measures the ratio of correct predictions to the total number of cases evaluated.

$$\mathrm{A}(M) = \frac{TN + TP}{TN + FP + FN + TP}$$

where

- TN is the number of true negative cases

- FP is the number of false positive cases

- FN is the number of false negative cases

- TP is the number of true positive cases

Accuracy is defined as the percentage of correctly predicted classifications, but does not take into account false positives or false negatives. Accuracy works fine if there are a balanced number of false positives and false negatives, but this is rarely the case. Luckily, that is our case.

II. Analysis

Data Exploration

The dataset is divided into training and testing. They can be found in https://www.kaggle.com/c/dogs-vs-cats/data. The training archive contains 25,000 labeled images of dogs(12500 images) and cats(12500 images). We will train our algorithm on these

files and predict the labels (1 = dog, 0 = cat). The sizes of the images are different, the ranges of dimensions is between 1023 X 768 and 60 X 39. Some examples of two differents format of inputs images are presented below.

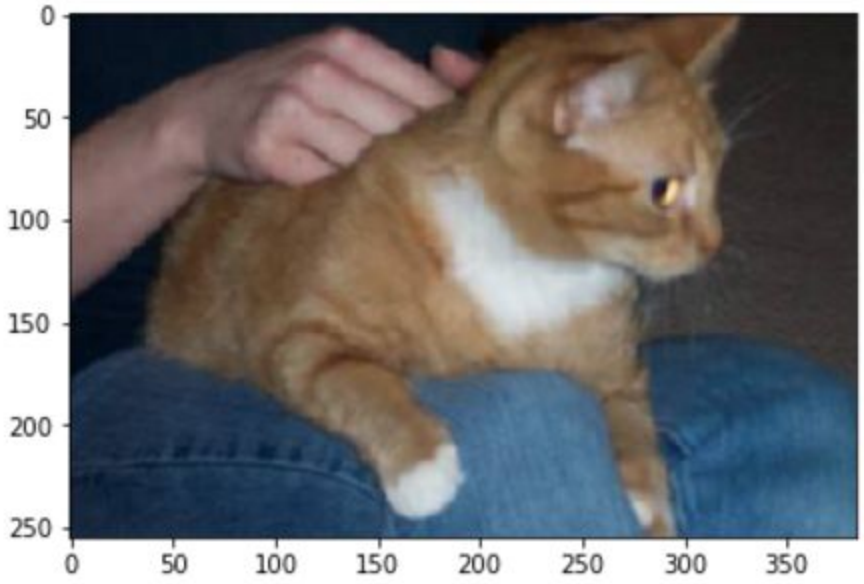
The testing archive contains 12500 images of cats and dogs unlabeled images that we will use for testing our Convolutional Neural Network model. Note here that the labels are contained in the name of the images.

Image1: Dimension: 286 X 270

Image2: 469 X 600

Exploratory Visualization

I plotted the new images below from the training data to present the data. We can observe here that the first image has a size of 250X350 and second has a size of 200X175. This will imply in the preprocessing step to resize all the training images to the same size.




```
print_images(X_train_files[8])
```

Image 3: 2 images of the dataset with different size.

Algorithms and Techniques

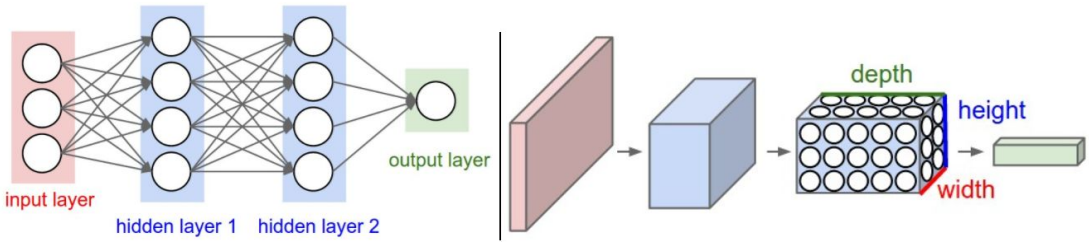
We will start this section by explaining how a Convolutional Neural Network(CNN) works whole. CNN are made up of Neurons and biases similar to ordinary Neural Network.

Recall on neural networks.

An Artificial Neural Network is based on a collection of connected units called artificial neurons. Typically, neurons are organized in layers. Neural Networks receive an input (a single vector), and transform it through a series of *hidden layers*. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the "output layer" and in classification settings it represents the class scores. The network consists of connections, each connection transferring the output of a neuron *i,* i to the input of a neuron *j*. *i* is the predecessor of *j* and *j* is the successor of *i*. Each connection is assigned a weight Wi,j.

CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs.

CNNs operate over Volumes which means that the input is a vector. In our case the input is a multichannel images. For example in a *3D volumes of neurons*, Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. In our implementation the depth is 3 because the input images has 3 channels(Red, Green, Blue channels). Here is a visualization from cs231n which give an a visual explanation of how a CNN is made up of Layers.:



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. From there we can read that, In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

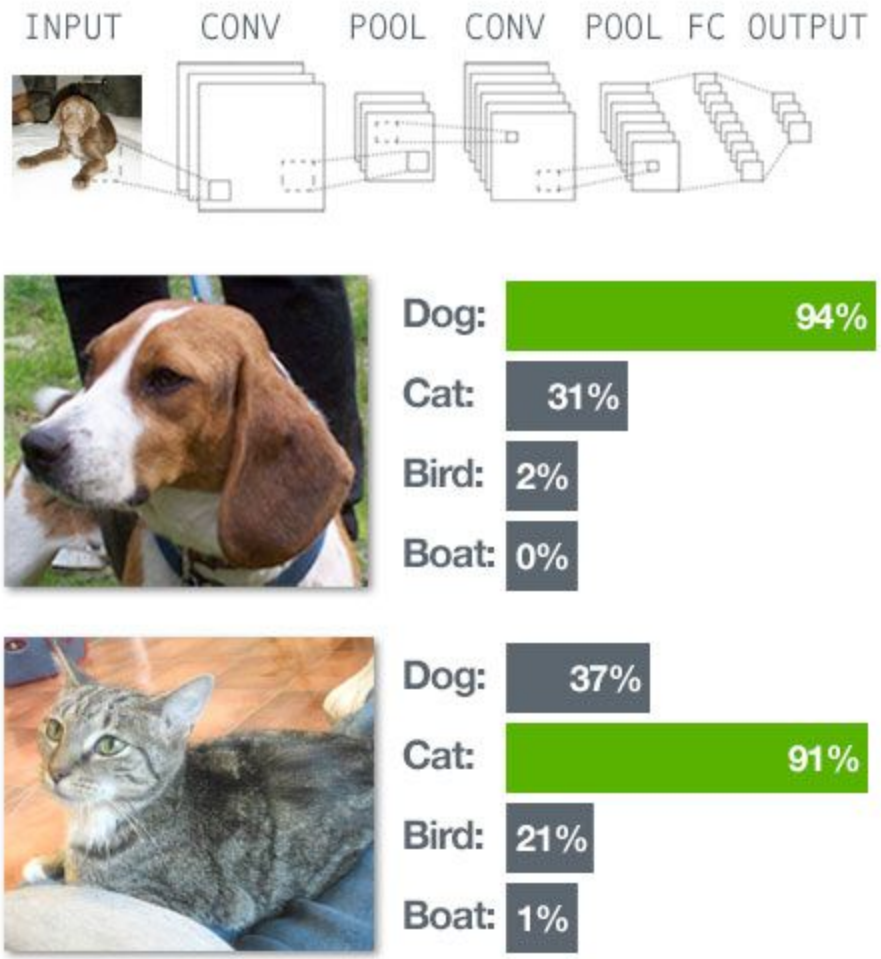
Layers used to build ConvNets

Convolutional Networks architectures are built with three main types of layers: **Convolutional Layer**, **Pooling Layer**, **and Fully-Connected Layer**.

Example Architecture: Overview. We will go into more details below, but a simple ConvNet for our cats dogs classification could have the architecture [INPUT - CONV - RELU - POOL - FC](see our benchmark). In more detail:

- **INPUT** [224x224x3] will hold the raw pixel values of the image, in this case an image of width 224, height 224, and with three color channels R,G,B.
- **CONV** layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [224x224x16] if we decided to use 16 filters.
- **RELU** layer will apply an elementwise activation function, such as the max(0,x). max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([224x224x16]).
- **POOL** layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- **FC** (fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

Please refers to the implementation section to see how we implemented this architecture in keras. Below is a visual explanation for the current problem taken from **pyimagesearch**.



Note: In our case we just have two classes dog or cat (as we want to classify between cats and dogs) the others two in this project are not consider.

Benchmark

I have run a simple CNN model architecture with a convolutional layer and one max pooling layer and get an accuracy of 49% when running the code for 4 epochs. This is choose as my benchmark. we will try to see if this model can beat this benchmark.

We will also compare our solution with the current best solution that use Transfer Learning model named VGG-19 (https://github.com/mrgloom/kaggle-dogs-vs-cats-solution) We will see if our CNN model build from scratch is more powerful that the VGG-19 which has 97.42%. (Test accuracy was measured on train-test split 80%-20%.)

This Benchmark solve the same problem from https://www.kaggle.com/c/dogs-vs-cats kaggle competition.

III. Methodology

Data Preprocessing

We have used 2 pre-processing techniques.

First, as mentioned in the Data Exploration section we have observed that the dataset was not have the same dimension. We have converted all the training and validation images with shape (224, 224, 3).

Second, we have rescaled the images by dividing every pixel in every image by 255. One can ask the reason to rescale images. The reason is that, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. And we are trying to solve here a classification problem.

We finally converted the 3D tensor (224, 224, 3) to 4D tensor with shape (1, 224, 224, 3) and return 4D tensor. This change was made to give a 4D tensor as input to the future Convolutional Network. If we don't make this change, the input sizes will be mismatched.

Implementation

1. **Import Datasets**

We used python glob library to import the dogs and cats dataset images. We populate a few variables through the use of the *load_files* function from the scikit-learn library and use numpy arrays to contain file paths to images. We created a function print_images to print images in the notebook with OpenCv. We printed two images to observe if our datasets imported correctly.

2. **Data Pre-processing**

We will discuss this in the data preprocessing section. However it is important to note that we used *keras.preprocessing image* to preprocess the images.

3. **Build the model**

I have used the *accuracy* metric as explained in the Metric section, for optimizer I have used *Adam* with the default parameters as suggested by keras. It is recommended to leave the

parameters of this optimizer at their default values (except the learning rate, which can be freely tuned). The model runs with 4 epochs. We have created a Convolutional Neural Network that classifies dogs vs cats. We have build our architecture using Keras with TensorFlow backend. Our model architecture is presented**.**The model architecture implemented is defined in **Figure 1**.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 110, 110, 32)      2080
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 54, 54, 64)        8256
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 26, 26, 64)        16448
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 10816)             0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 21634
=================================================================
Total params: 48,626.0
Trainable params: 48,626.0
Non-trainable params: 0.0
_________________________________________________________________
```

Figure 1: Model architecture 1 (`.summary()`)

Conv2D: This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. We added **relu** as non linear activation function in each the Convolutional layer.
MaxPooling2D: The objective of max pooling layer is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.
Flatten layer: The Flatten layer is a utility layer that flattens an input of shape a * b * c * d to a simple vector output of shape a * (a*c*d).

Dense: Dense layer is a Fully connected layers, which perform classification on the features extracted by the convolutional layers and downsampled by the pooling layers. In a dense layer, every node in the layer is connected to every node in the preceding layer.
Dropout: layer was introduced as a regularization technique for reducing overfitting.

I encountered a problem when compiling the model architecture. I have tried keras loss function '*'categorical_crossentropy'*, however this generated an error, I changed it with *'sparse_categorical_crossentropy* after checking on stack-overflow. And this change helps me to progress in my work.

The biggest problem I've encountered is the number of the training dataset preprocessing. In preprocessing step, I encounter the resource limit problem with 25,000 images of cats and dogs. I did several attempts without success. I ended up using 10,000 images for training, 2,500 for validation. The following error was generated in my environment Windows 10, Intel Core i7, 8Gb RAM, NVIDIA GEFORCE 950M.

I use keras API to train and fit the dataset with TensorFlow backend.

Refinement

My initial solution had 59.50% accuracy. I have change the sigmoid activation layer of my network with softmax, I get an accuracy of 89.18%, I also use Adam as optimizer instead of *rmsprop* as I read that rmsprop is more related with Recurrent Neural Network.
At the starting of the project, I planned to train and validate the model in all the 25,000 images. But as the code generate resources exhausted error after more than 3 hours of preprocessing. Note that i have tried this step many day with the same error, I decided to reduce the number of training and validation to 12,500 images from the dataset. After reducing the dataset to 12,500 images I was able to conduct the preprocessing step. I take many time but it finally past.

IV. Results

Model Evaluation and Validation

The model is reasonable when we compare it with our expected solution. We get 89.18% accuracy which is not bad for this project compared with the initial 50%. I reach this result with 4 epochs. To study how the model generalize I have tested the model in 25 new unseen images and the result was quite good 88% accuracy. This image comes from test dataset images of the Kaggle competition. The result is trusted but can be improved.

A resume of the description of our final model architecture is. 4 convolutions layers each with a relu activation layer, and each is followed with a Maxpooling layer. The final layer is a fully connected layer with softmax as activation layer.

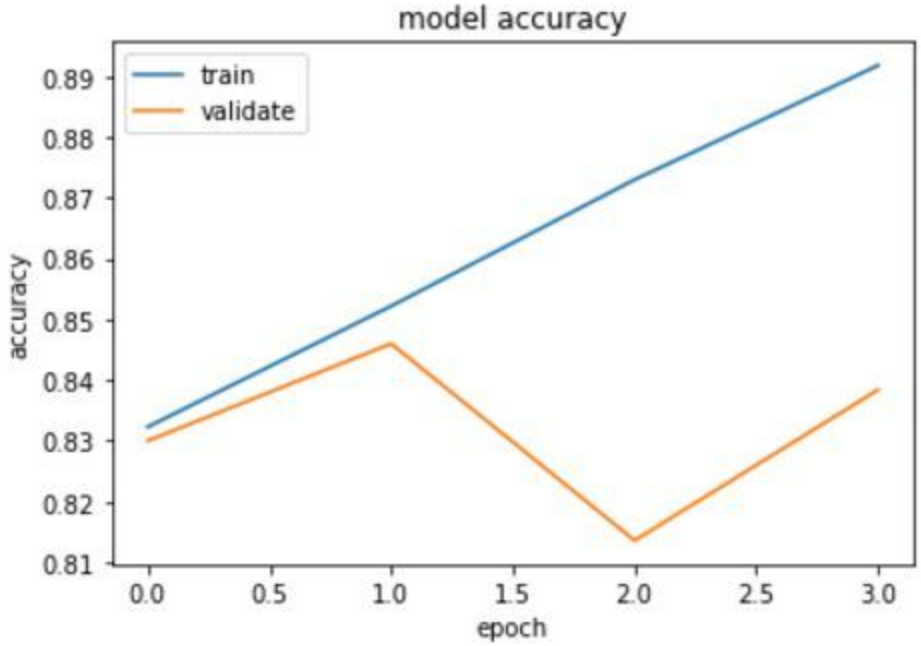
My final model has an accuracy less that the best 97,42% of the others benchmarks defined above. Please refers to the Improvement section to see some advice to do improve my work on this project. I have not try to implement some improvement because of limited resource.

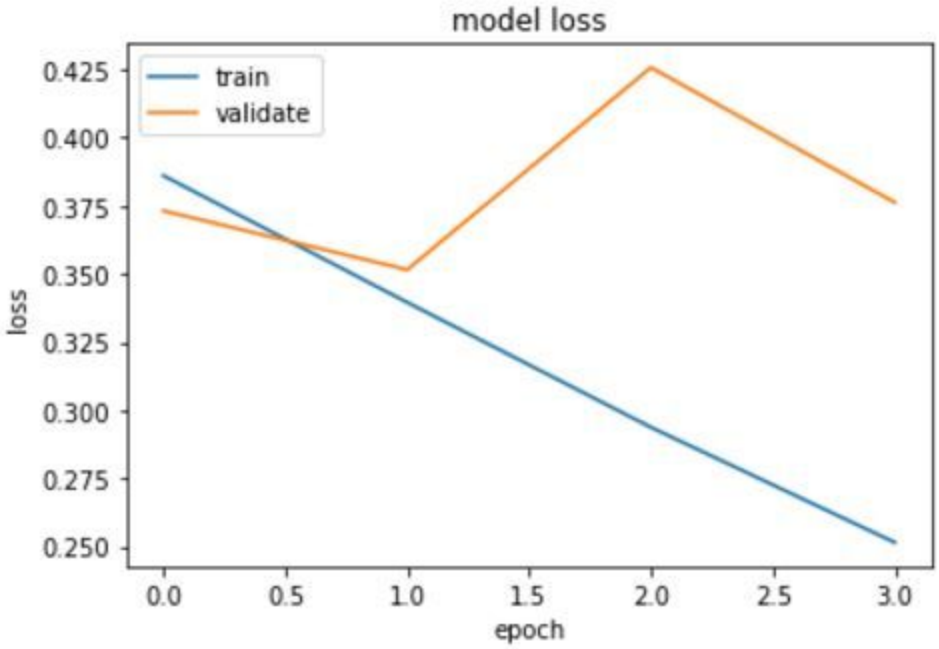
Justification

My final solution 89.18% accuracy is stronger than the benchmark which has >50% accuracy. This solution is a quite good to classify between cats and dogs, as we have tested in 25 unseen images and the model was able to classify 22/25 which give an accuracy of 88%. I have not submitted the final results from the test dataset in Kaggle as this option has been deactivated from Kaggle.

V. Conclusion

Free-Form Visualization

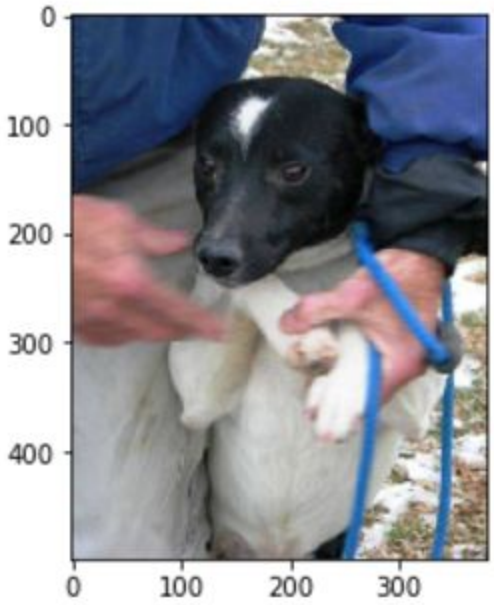




We provided above a visualization of loss and accuracy when epochs increase. We can observe that the training accuracy is increasing when the training loss decrease. This is a good sign to shows that our CNN is learning while the epochs increase. We observe an overfishing start when

the number of epochs exceeds 1. Indeed the accuracy validation is substantially equal to 80% yet the training accuracy is ~ 90%. Below are 2 predictions of our model in the test set. We see that the model build is able to provide the correct prediction of both images.

```
This image contains dog
```



```
This image contains cat
```

VI. Conclusion

Reflection

I have greatly enjoyed working on this project as my personal introduction to Deep Learning as I want to know how computer can classify images.The process I followed to classify dogs is outlined below.

1. I started by downloading the training and testing dataset from Kaggle as this was a kaggle competition.
2. I did some data exploration by loading the dataset, second I calculated the number of data loaded(25000 training images, 12500 testing images), after this step, I printed some images to see if my images were well loaded. we observed that all the images did not contains the same dimension.
3. After data exploration the next step was data preprocessing. I this step we divided the dataset to train-test split 80%-20%. We resized all the images to 224 X 224 to have them with the same dimension. To finish this section, I convert 3D tensor to 4D tensor with shape (1, 224, 224, 3) and return 4D tensor to train the Neural Network.
4. We implemented the Benchmark and the our model architecture as described above. We then Trained the Benchmark and our CNN.
5. Finally we test our model in 25 new unseen data taken from the testing dataset provided by Kaggle. We finally compared our model with the Benchmark defined and we observe that, our model performs more than the benchmark with an accuracy of 89.18 while the Benchmark had 59.5%.

The project was very interesting. I have mainly discovered the limit of my computer and the need of a more powerful machine. As my machine was freezing during the preprocessing of 25,000 images. This was difficult but part but interesting. I finaly end to use 10,000 images for training and 2,500 images for validation. My final solution is acceptable and can be used as a starter application for classify between cats and dogs as the model is accurate about ~90%. As mentioned earlier, the current model can be improve. The following section will focus to somme improvement.

Reflection

The current work done in this project can be improved.

- The First idea is to use all the 25,000 images in a more powerful machine such as AWS dedicated GPU cloud computer for deep Learning. Train the model in more images will allow our model to be more accurate. Here increase of dataset could be helpful as this techniques made its proof(ref:

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)

- Increase the depth of the current Neural Network by adding many others Conv layers and the number of epochs could also be helpful.
- I will also suggest the use of Transfer Learning. The highest score in the other part of the benchmark we defined implement the VGG-19 model and get an accuracy of 97%(ref: https://github.com/mrgloom/kaggle-dogs-vs-cats-solution). Test accuracy was measured on train-test split 80%-20% in all the 25,000 images. I think this solution is more better than the current solution described in this report.
- We can also web application to classify dogs vs cats. The user will upload the file and the application will display if the image contains dog or cat.

Reference:

http://cs231n.github.io/convolutional-networks/
https://www.kaggle.com/c/dogs-vs-cats
https://keras.io/optimizers/
https://stackoverflow.com/questions/42081257/keras-binary-crossentropy-vs-categorical-crossentropy-performance
https://github.com/mrgloom/kaggle-dogs-vs-cats-solution
https://www.kaggle.com/joshuabeard/cats-v-dogs-cnn