INDIAN INSTITUTE OF TECHNOLOGY (INDIAN SCHOOL OF MINES), DHANBAD

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT REPORT
SESSION (2017-18)
IV SEMESTER

TOPIC: Image Processing Using Convolutional Neural Networks (Implementing Cats and Dogs Image Classifier)

SUBMITTED TO

Dr. Annavarapu Chandra Sekhara Rao

(CSE DEPARTMENT)

SUBMITTED BY

Sambhavesh Haralalka
16JE002482
Mahabir Saha
16JE002439

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING IIT (ISM) DHANBAD

CERTIFICATE

This is to certify that the project entitled "**Image processing using Convolutional Neural Network**"- submitted by **Sambhavesh Haralalka (Adm No. 16JE002482)** and **Mahabir Saha (Adm No.- 16JE002439)** under the guidance of **Dr. A.C.S Rao,** Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines) Dhanbad have successfully completed the project work in the academic session 2017-2018. The result embodied in this project have not been reproduced for any purpose in any institute or university.

Dr A.C.S RAO

Assistant Professor
Department of CSE
IIT (ISM) Dhanbad

DECLARATION

We hereby declare that this project report is an authentic record of the research work carried out by us. To the best of our knowledge it contains no material previously published or written by another person or material which has been accepted for the award of any degree or diploma of the university or any other institutes of higher learning where due acknowledgement has been made in the text.

Date:

SAMBHAVESH HARALALKA
Admission No. 16JE002482

MAHABIR SAHA
Admission No. 16JE002439

ACKNOWLWFGEMENT

We would like to express heartfelt gratitude and regards to my project guide **Dr. A.C.S Rao,** Department of Computer Science and Engineering, IIT (ISM) Dhanbad. I convey a humble thanks to him for his valuable cooperation, support and suggestion throughout the project work which made this project successful. I shall remain indebted throughout my life fr his noble help and guidance.

We are thankful to all the faculties of the Department of Computer Science and Engineering, IIT (ISM) Dhanbad for their encouraging words and valuable suggestions towards the project work. Last but not the least I want to acknowledge the contribution of my parents, family members, and friends for their constant and never-ending motivation.

We would like to take this opportunity too express my gratitude to the people who have been instrumental in the successful completion of the project.

Date:
Place:

SAMBHAVESH HARALALKA
Admission No. 16JE002482

MAHABIR SAHA
Admission No. 16JE002439

ABSTRACT

The project titled **Image processing using Convolutional Neural Network** focuses on classifying images of Cats and Dogs in their respective classes. The project is developed in Python using various packages such as tflearn, numpy, opencv etc.

A grayscale image serves as input to the convolutional network which is processed based on various filters and the class (whether the image is of a cat or a dog) is given as output.

24000 labelled images were used to train the network.

1000 labelled images were used to test the network.

12500 unlabelled images output was recorded.

TABLE OF CONTENTS

Chapter 1

1. Introduction

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually **Image Processing** system includes treating images as two-dimensional signals while applying already set signal processing methods to them.

Image processing basically includes the following three steps:

- Importing the image with optical scanner or by digital photography.
- Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.
- Output is the last stage in which result can be altered image or report that is based on image analysis.

There are two types of **methods used for Image Processing.** They are **Analog and Digital** Image Processing.

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing

Analog image processing is any image processing task conducted on two-dimensional analog signals by analog means. Analog image processing is done on analog signals. It includes processing on two dimensional analog signals. In this type of processing, the images are manipulated by electrical means by varying the electrical signal. The common example include is the television image.

Digital image processing has dominated over analog image processing with the passage of time due its wider range of applications.

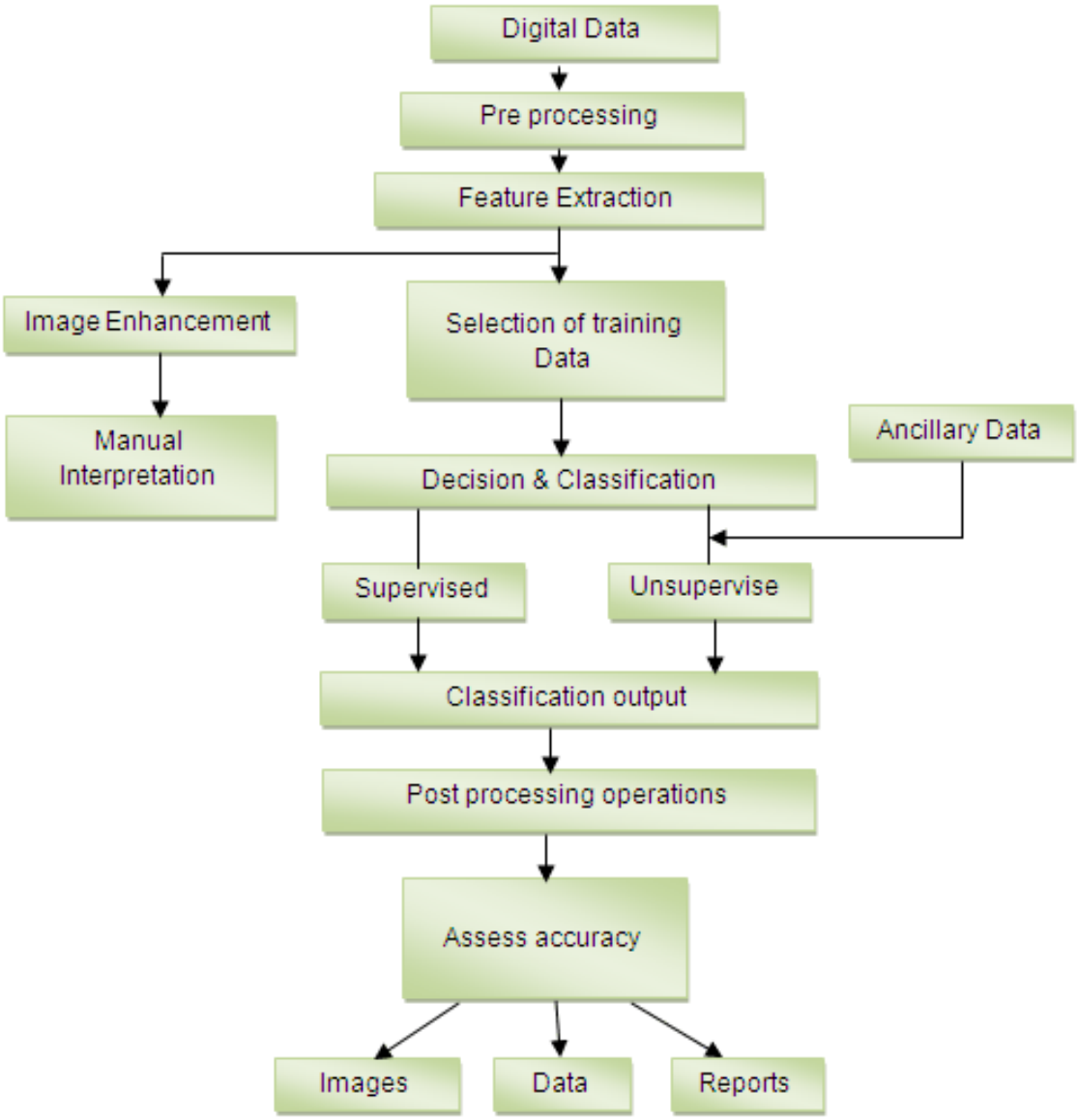


Fig 1.1 Representational Flowchart for image Processing

Purpose of Image processing

The purpose of image processing is divided into 5 groups. They are:

1. Visualization - Observe the objects that are not visible.
2. Image sharpening and restoration - To create a better image.
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern – Measures various objects in an image.
5. Image Recognition – Distinguish the objects in an image.

Image Processing is one of the most rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines.

Here we perform image processing to work on image recognition of Cats and Dogs.

Chapter 2

2.1. Convoluted Neural Networks

Convolutional Neural Networks (**ConvNets** or **CNNs**) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars.

A **Convolutional neural network** (**CNN**, or **ConvNet**) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as **shift invariant** or **space invariant artificial neural networks** (**SIANN**), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.
They have applications in image and video recognition, recommender systems and natural language processing.

There are several design parameters associated with CNN:

1. Convolution
2. Normalizaion (Rectified Linear Units ReLU)
3. Pooling
4. Fully Connected layer
5. Loss layer
6. Choosing hyperparameters

Convolution:

The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

ConvNets derive their name from the “convolution” operator

Normalization:

A ReLU layer performs a threshold operation to each element of the input, where any value less than zero is set to zero.

This operation is equivalent to

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0. \end{cases}$$

Pooling:

Sometimes when the images are too large, we would need to reduce the number of trainable parameters. It is then desired to periodically introduce pooling layers between subsequent convolution layers. Pooling is done for the sole purpose of reducing the spatial size of the image. Pooling is done independently on each depth dimension therefore the depth of the image remains unchanged. The most common form of pooling layer generally applied is the max pooling.

Pooling each filtered image number of times reduces the size of the array of pixels.

Each such set of operations are repeated a number of times so that the resultant image is more filtered (due to convolution), lesser in number of pixels (due to pooling) and with Rectified Linear Units(ReLU) by a process called **deep-stacking**.

Fully Connected layer:

Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

Choosing hyperparameters

CNNs use more hyperparameters than a standard MLP. While the usual rules for learning rates and regularization constants still apply, the following should be kept in mind when optimising.

Number of filters

Since feature map size decreases with depth, layers near the input layer will tend to have fewer filters while higher layers can have more. To equalize computation at each layer, the feature x pixel position product is kept roughly constant across layers. Preserving more information about the input would require keeping the total number of activations (number of feature maps times number of pixel positions) non-decreasing from one layer to the next.

The number of feature maps directly controls capacity and depends on the number of available examples and task complexity.

Filter shape

Common field shapes found in the literature vary greatly and are usually chosen based on the dataset. The challenge is thus to find the right level of granularity so as to create abstractions at the proper scale, given a particular dataset.

Max pooling shape

Typical values are 2x2. Very large input volumes may warrant 4x4 pooling in the lower-layers. However, choosing larger shapes will dramatically reduce the dimension of the signal, and may result in excess information loss. Often, non-overlapping pooling windows perform best.

2.2 Tensor Flow

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google, often replacing its closed-source predecessor, DistBelief.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open source license on November 9, 2015.

The various commands used in the project are given as follows :

1. Converting to grayscale

- tf.image.rgb_to_grayscale
- tf.image.grayscale_to_rgb

2. Encoding and Decoding :

- tf.image.decode_jpeg
- tf.image.encode_jpeg

3. Resizing

- tf.image.resize_images
- tf.image.resize_area

4. Cropping

- tf.image.crop_and_resize

5. Flipping Rotating and Transposing

- tf.image.flip_up_down
- tf.image.random_flip_up_down
- tf.image.flip_left_right
- tf.image.random_flip_left_right
- tf.image.transpose_image

Chapter 3

3.1 Objective

To apply Image processing techniques to train our system with a classified set of images of cats and dogs (12,500 each) and to test our module for image recognition by feeding it with a set of undetermined or unclassified images of cats and dogs.

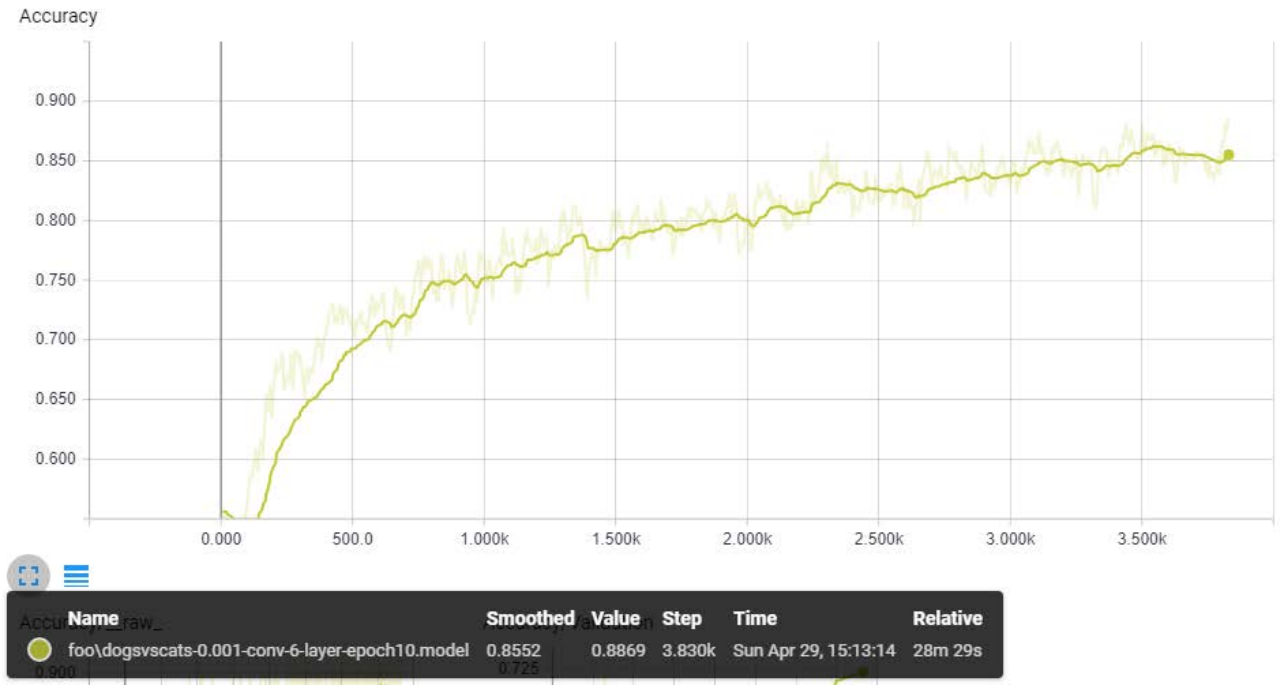
To see observable changes in training and test accuracy levels with change in the numbers of epoch values and convolutional layer numbers.

We first make a network with one convolutional layer with epoch three and then train and test our network. Increase the number of layers one by one and finally network with seven convolutional layers with epoch ten is tested and results are recorded.

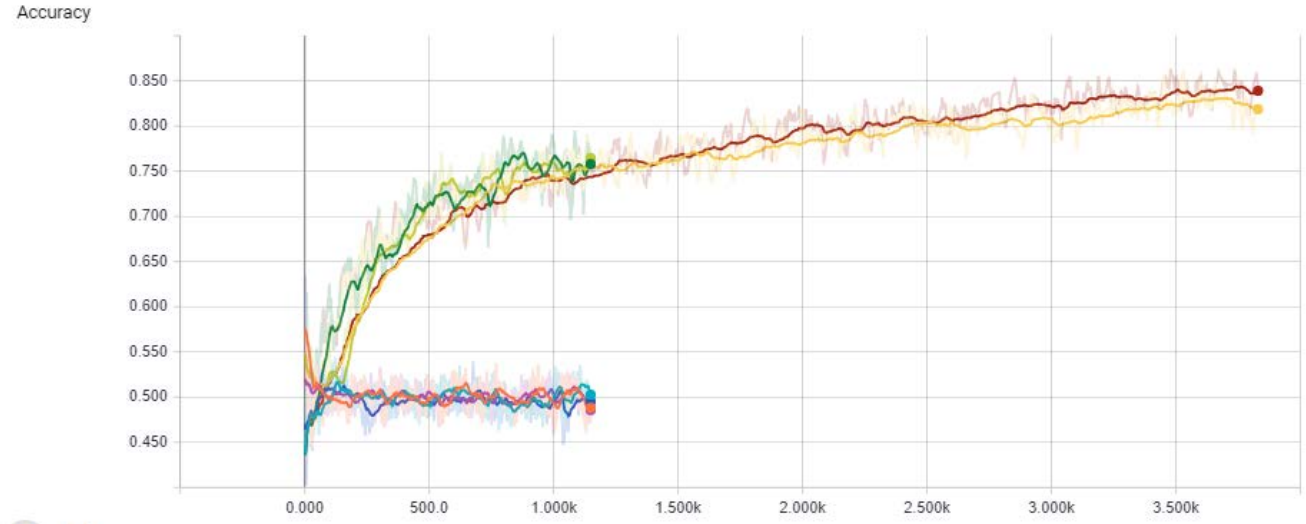
3.2 Result

Layer	Epoch	Training Accuracy	Training Loss	Testing Accuracy	Testing Loss
1	3	0.4743	12.10578	0.494	11.65108
2	3	0.4805	11.96085	0.494	11.65108
3	3	0.4787	12.00434	0.494	11.65108
4	3	0.4909	0.69326	0.494	0.69318
5	3	0.747	0.49279	0.714	0.56349
6	3	0.7729	0.49811	0.738	0.55641
5	10	0.8207	0.37571	0.764	0.52397
6	10	0.8869	0.26646	0.778	0.61134
7	10	0.8441	0.35310	0.756	0.61737

Best Result:



Comparing All:



Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-1-layer-epoch3.model	0.4881	0.4743	1.149k	Sun Apr 29, 12:16:09	5m 43s
foo\dogsvscats-0.001-conv-2-layer-epoch3.model	0.5027	0.4805	1.149k	Sun Apr 29, 12:57:49	6m 36s
foo\dogsvscats-0.001-conv-3-layer-epoch3.model	0.4857	0.4787	1.149k	Sun Apr 29, 13:31:58	6m 57s
foo\dogsvscats-0.001-conv-4-layer-epoch3.model	0.4951	0.4909	1.149k	Sun Apr 29, 13:57:42	6m 49s
foo\dogsvscats-0.001-conv-5-layer-epoch10.model	0.8185	0.8207	3.830k	Sun Apr 29, 15:55:14	21m 56s
foo\dogsvscats-0.001-conv-5-layer-epoch3.model	0.7578	0.7470	1.149k	Sun Apr 29, 14:14:24	6m 45s
foo\dogsvscats-0.001-conv-6-layer-epoch3.model	0.7645	0.7729	1.149k	Sun Apr 29, 14:34:29	8m 46s
foo\dogsvscats-0.001-conv-7-layer-epoch10.model	0.8389	0.8441	3.830k	Sun Apr 29, 17:10:06	47m 19s

References

1) https://pythonprogramming.net/tflearn-machine-learning-tutorial/
2) https://numpy.org
3) https://tflearn.org
4) https://keras.io/
5) https://opencv.org
6) https://matplotlib.org
7) https://towardsdatascience.com
8) https://tensorflow.org
9) https://deeplearning.net
10) https://developer.google.com

APPENDIX - 1 (CODE)

Preprocessing

First, we'll get our imports and constants for preprocessing:

In [1]:
```
import cv2                  # working with, mainly resizing, images
import numpy as np          # dealing with arrays
import os                   # dealing with directories
from random import shuffle # mixing up or currently ordered data that might lead our network astray in training.
from tqdm import tqdm       # a nice pretty percentage bar for tasks.

TRAIN_DIR = 'C:/Users/Sambhavesh/Documents/Project/train'
TEST_DIR = 'C:/Users/Sambhavesh/Documents/Project/test'
IMG_SIZE = 50
LR = 1e-3

MODEL_NAME = 'dogsvscats-{}-{}.model'.format(LR, '5conv-basic-epoch10-LR1e03')
```

Our images are labeled like "cat.1" or "dog.3" and so on, so we can just split out the dog/cat, and then convert to an array:

In [2]:
```
def label_img(img):
    word_label = img.split('.')[-3]
    # conversion to  array [cat,dog]
    #                           [cat, no dog]
    if word_label == 'cat': return [1,0]
    #                            [no cat, dog]
    elif word_label == 'dog': return [0,1]
```

Function to fully process the training images and their labels into arrays:

In [3]:
```
def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data
```

In [4]:
```
def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR,img)
        img_num = img.split('.')[0]
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data
```

Now, we can run the training:

In [5]:
```
train_data = create_train_data()
# If you have already created the dataset:
#train_data = np.load('train_data.npy')
```

Convolutional Neural Network

1 Layer:

In [6]:
```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

2 Layer:

In [6]:
```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

3 Layer:

In [6]:
```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5,
activation='relu') convnet = max_pool_2d(convnet, 5)


convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

4 Layer:

In [6]:
```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5,
activation='relu') convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

5 Layer:

In [6]:
```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5,activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

6 Layer:

In [6]:
```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5,activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5,activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

7 Layer:

In [6]:
```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5,activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 256, 5,activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5,activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

In [7]:
```
if os.path.exists('C:/Users/Sambhavesh/Documents/Project/{}.meta'.
    format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model loaded!')
```

Now, let's split out training and testing data:

In [8]:
```
train = train_data[:-500]
test = train_data[-500:]
```

In [9]:
```
X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
test_y = [i[1] for i in test]
```

Now we fit for 3 epochs:

In [10]:
```
model.fit({'input': X}, {'targets': Y}, n_epoch=3, validation_set=({'input':
    test_x}, {'targets': test_y}), snapshot_step=500, show_metric=True,
    run_id=MODEL_NAME)
```

Now we fit for 10 epochs:

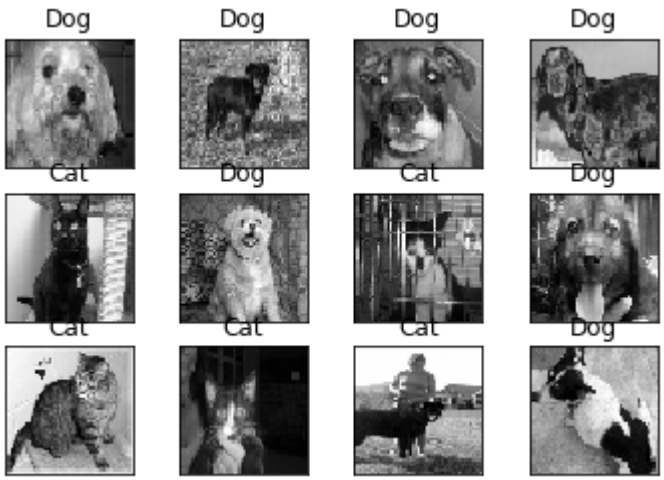
In [10]:
```
model.fit({'input': X}, {'targets': Y}, n_epoch=10, validation_set=({'input':
    test_x}, {'targets': test_y}), snapshot_step=500, show_metric=True,
    run_id=MODEL_NAME)
```

In [11]:
```
model.save(MODEL_NAME)
#tensorboard --logdir=foo:C:\Users\Sambhavesh\Documents\Project\log
```

Visually inspecting our network against unlabeled data

In [12]:
```
import matplotlib.pyplot as plt

# if you need to create the data:
#test_data = process_test_data()
# if you already have some saved:

test_data=np.load('test_data.npy)

fig=plt.figure()

for num,data in enumerate(test_data[:12]):
    # cat: [1,0]
    # dog: [0,1]

    img_num = data[1]
    img_data = data[0]

    y = fig.add_subplot(3,4,num+1)
    orig = img_data
    data = img_data.reshape(IMG_SIZE,IMG_SIZE,1)
    #model_out = model.predict([data])[0]
    model_out = model.predict([data])[0]

    if np.argmax(model_out) == 1: str_label='Dog'
    else: str_label='Cat'

    y.imshow(orig,cmap='gray')
    plt.title(str_label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)
plt.show()
```

```
100%|██████████| 12500/12500 [01:05<00:00, 190.50it/s]
```



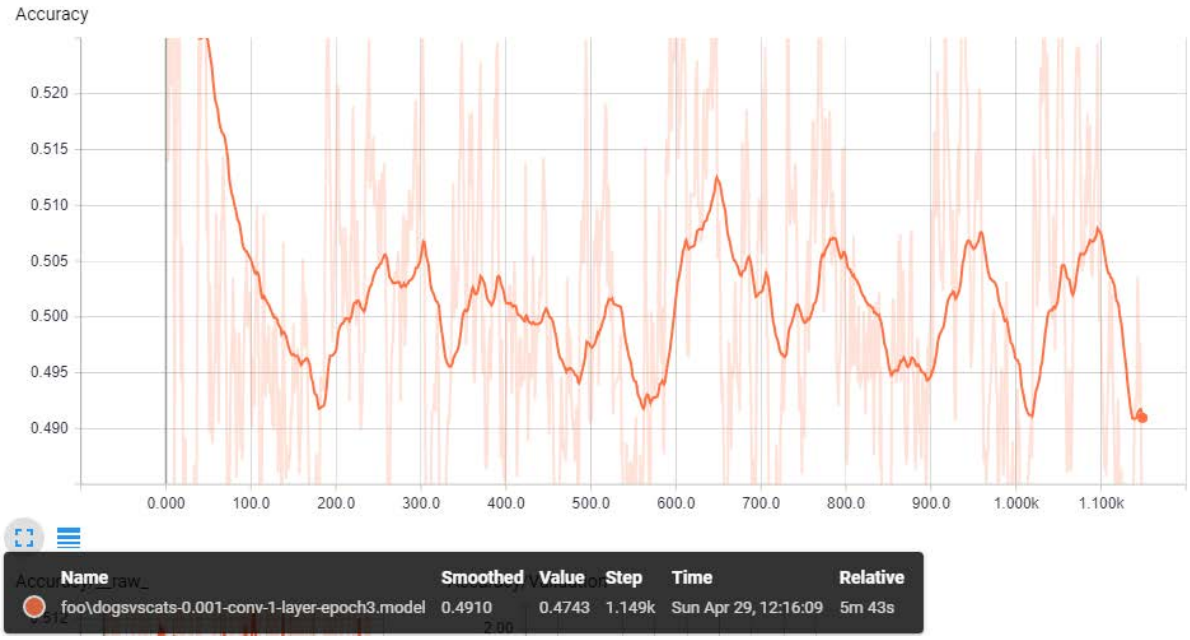
In [13]:
```
with open('submission_file.csv','w') as f:
    f.write('id,label\n')

with open('submission_file.csv','a') as f:
    for data in tqdm(test_data):
        img_num = data[1]
        img_data = data[0]
        orig = img_data
        data = img_data.reshape(IMG_SIZE,IMG_SIZE,1)
        model_out = model.predict([data])[0]
        f.write('{},{}\n'.format(img_num,model_out[1]))
```

```
100%|██████████| 12500/12500 [00:47<00:00, 263.45it/s]
```

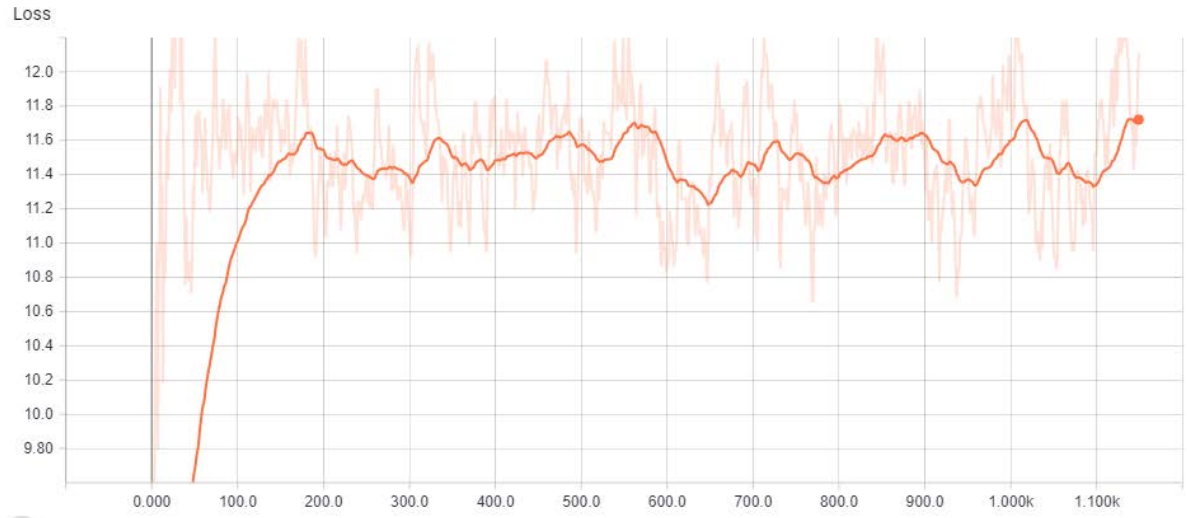
In [ ]:

APPENDIX - 2 (GRAPH)

1) Layer(1) , Epoch(3) :

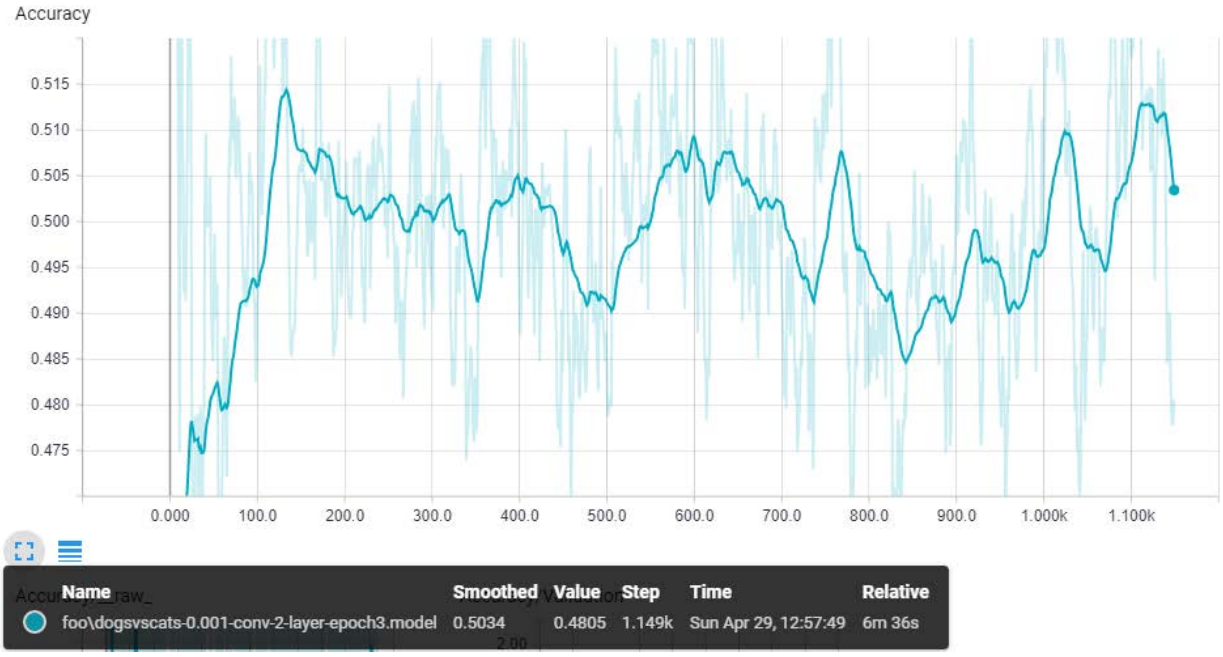


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-1-layer-epoch3.model	0.4910	0.4743	1.149k	Sun Apr 29, 12:16:09	5m 43s

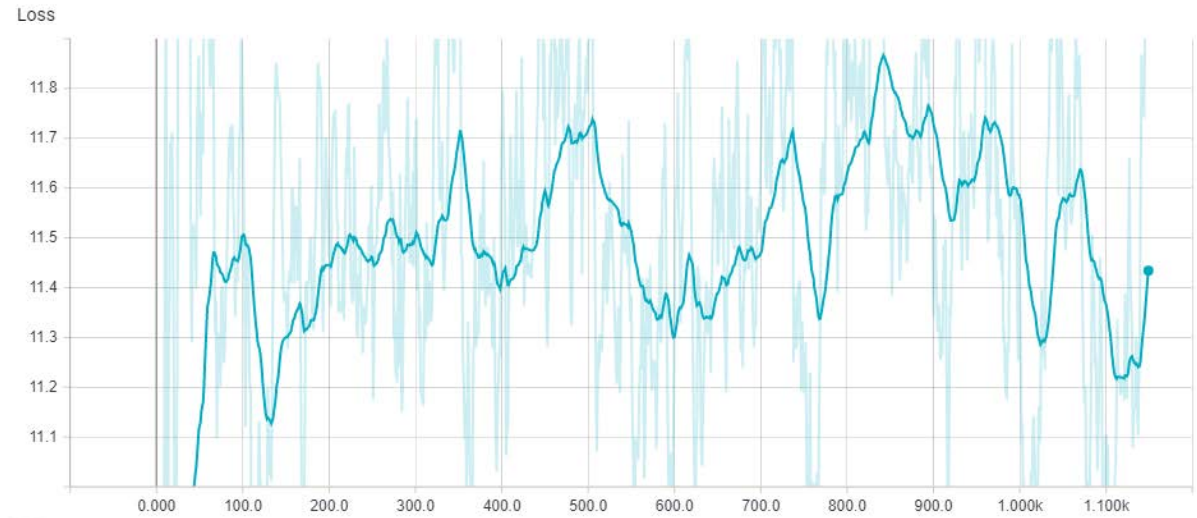


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-1-layer-epoch3.model	11.72	12.11	1.149k	Sun Apr 29, 12:16:09	5m 43s

2) Layer(2), Epoch(3):

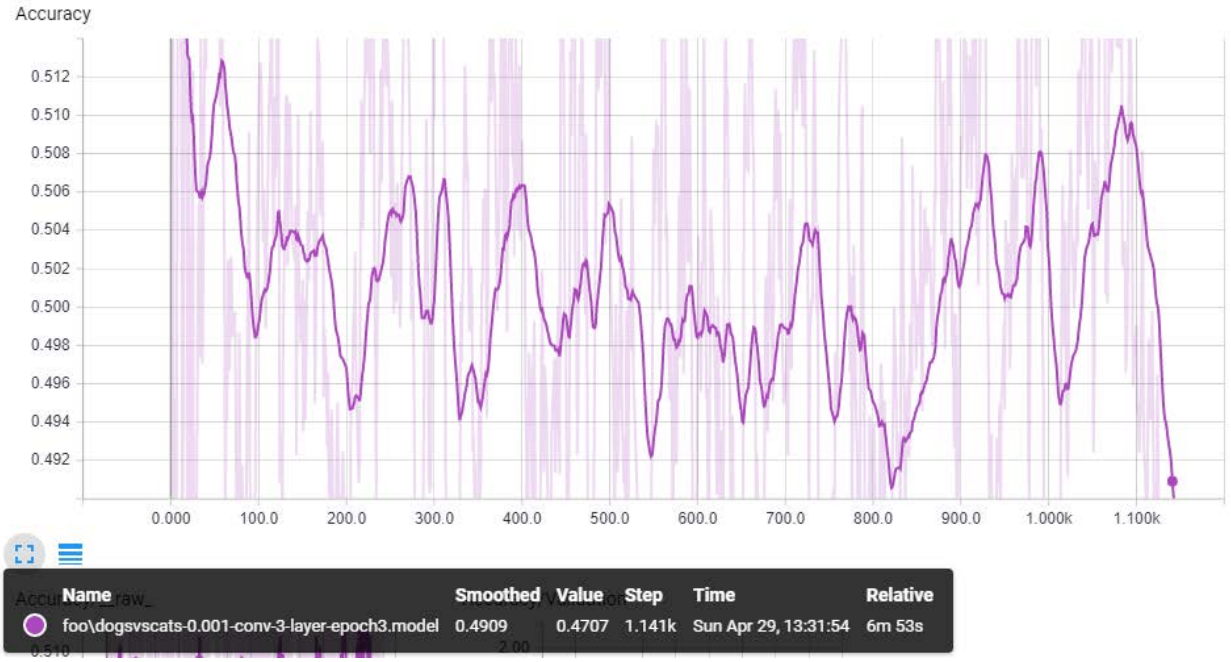


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-2-layer-epoch3.model	0.5034	0.4805	1.149k	Sun Apr 29, 12:57:49	6m 36s

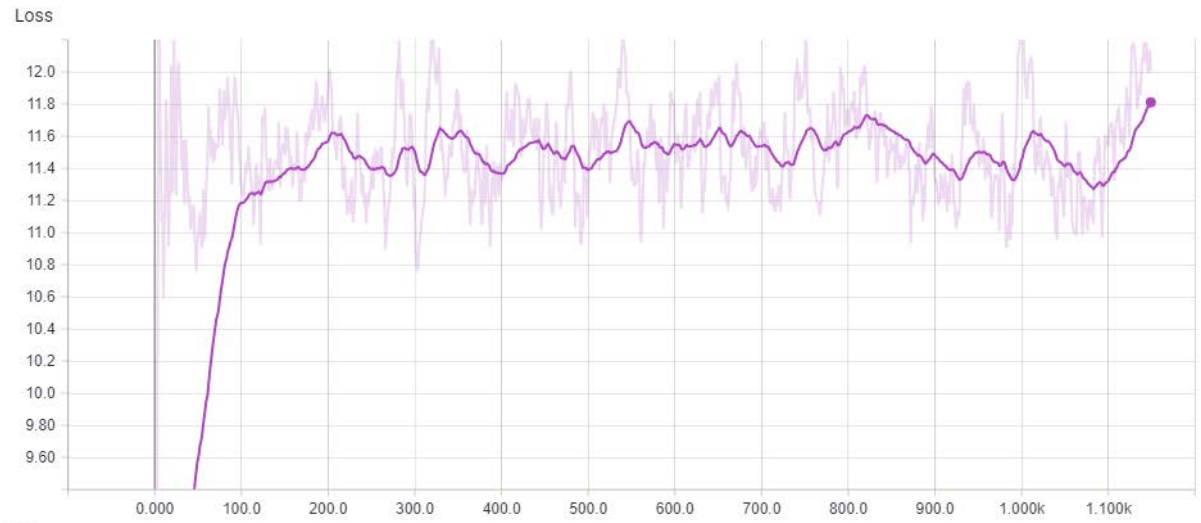


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-2-layer-epoch3.model	11.43	11.96	1.149k	Sun Apr 29, 12:57:49	6m 36s

3) Layer(3), Epoch(3):

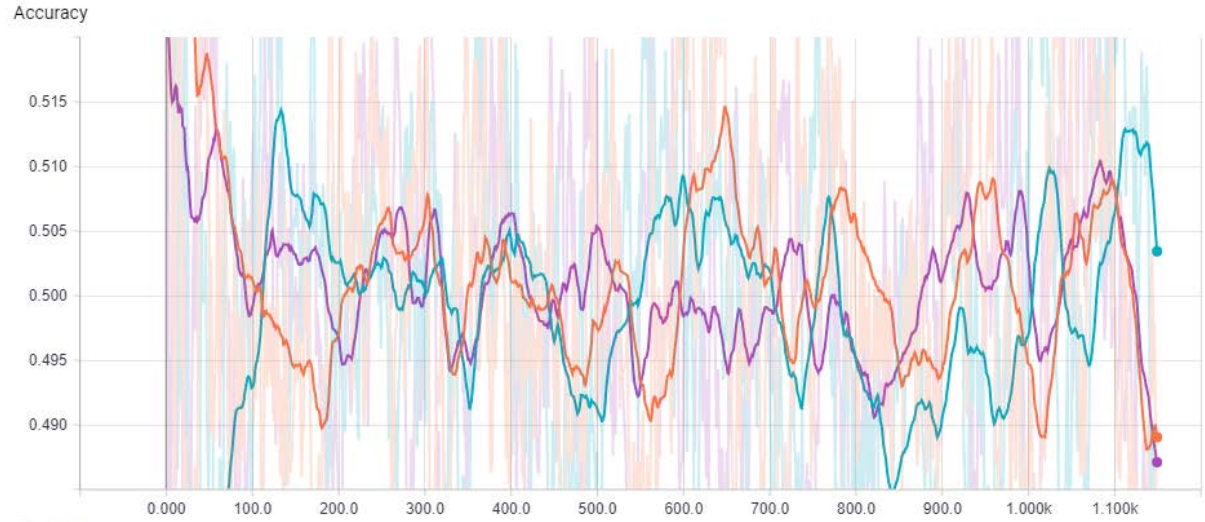


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-3-layer-epoch3.model	0.4909	0.4707	1.141k	Sun Apr 29, 13:31:54	6m 53s

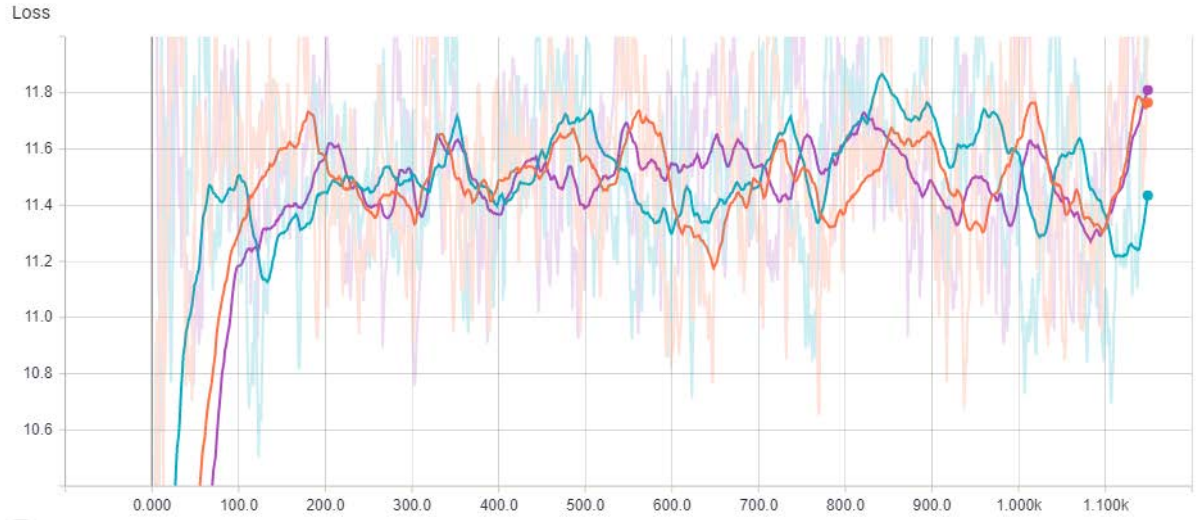


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-3-layer-epoch3.model	11.81	12.00	1.149k	Sun Apr 29, 13:31:58	6m 57s

4) Comparision among Layer(1,2,3), Epoch(3):

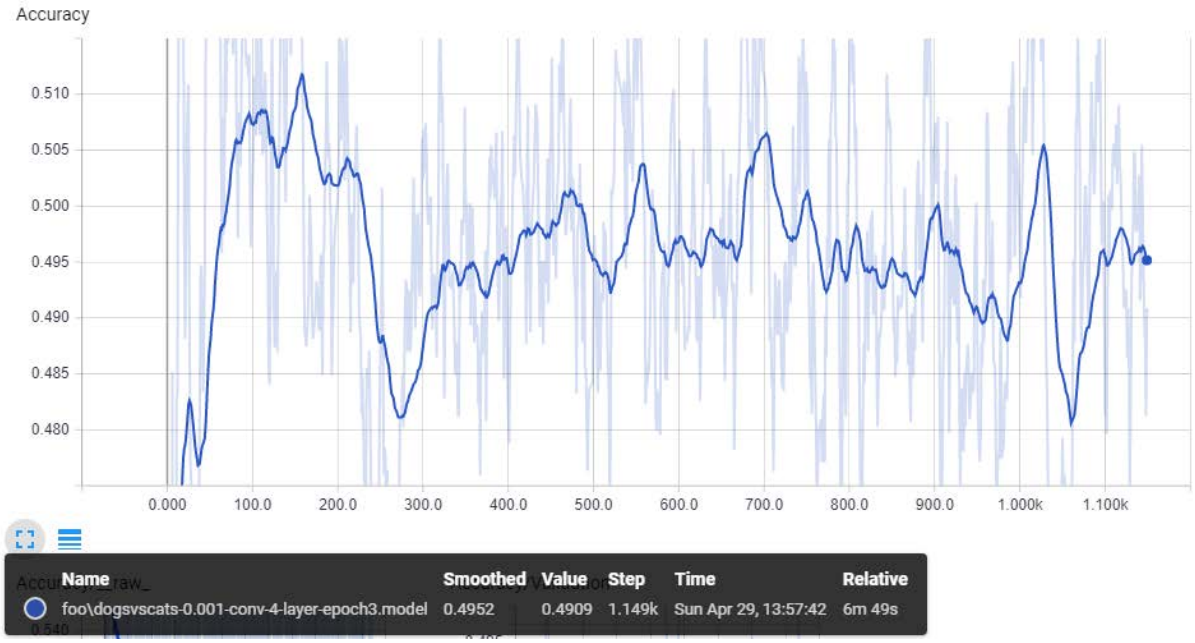


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-1-layer-epoch3.model	0.4890	0.4743	1.149k	Sun Apr 29, 12:16:09	5m 43s
foo\dogsvscats-0.001-conv-2-layer-epoch3.model	0.5034	0.4805	1.149k	Sun Apr 29, 12:57:49	6m 36s
foo\dogsvscats-0.001-conv-3-layer-epoch3.model	0.4871	0.4787	1.149k	Sun Apr 29, 13:31:58	6m 57s

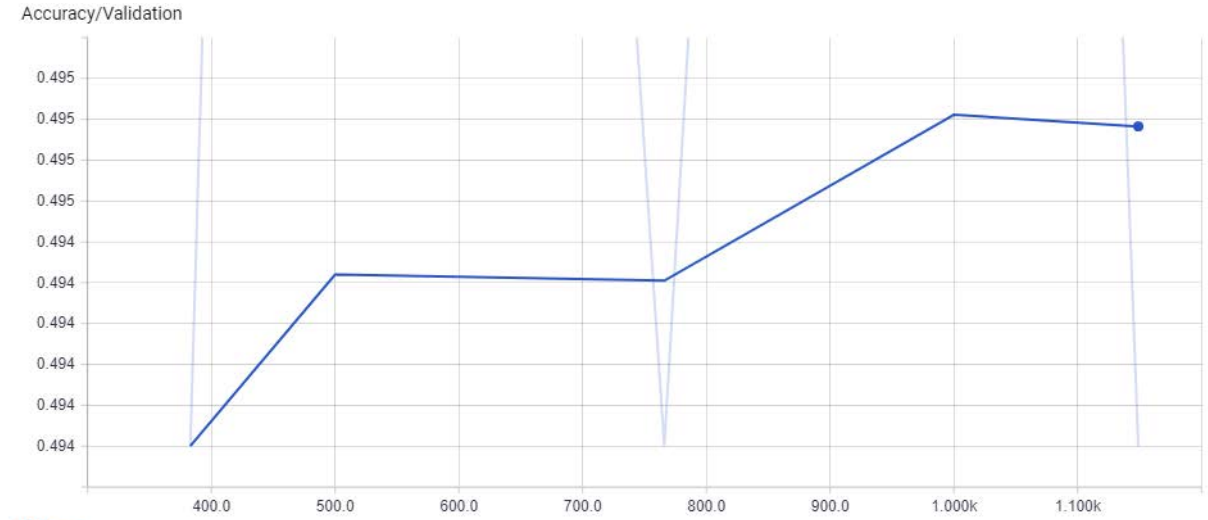


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-1-layer-epoch3.model	11.77	12.11	1.149k	Sun Apr 29, 12:16:09	5m 43s
foo\dogsvscats-0.001-conv-2-layer-epoch3.model	11.43	11.96	1.149k	Sun Apr 29, 12:57:49	6m 36s
foo\dogsvscats-0.001-conv-3-layer-epoch3.model	11.81	12.00	1.149k	Sun Apr 29, 13:31:58	6m 57s

5) Layer(4), Epoch(3):

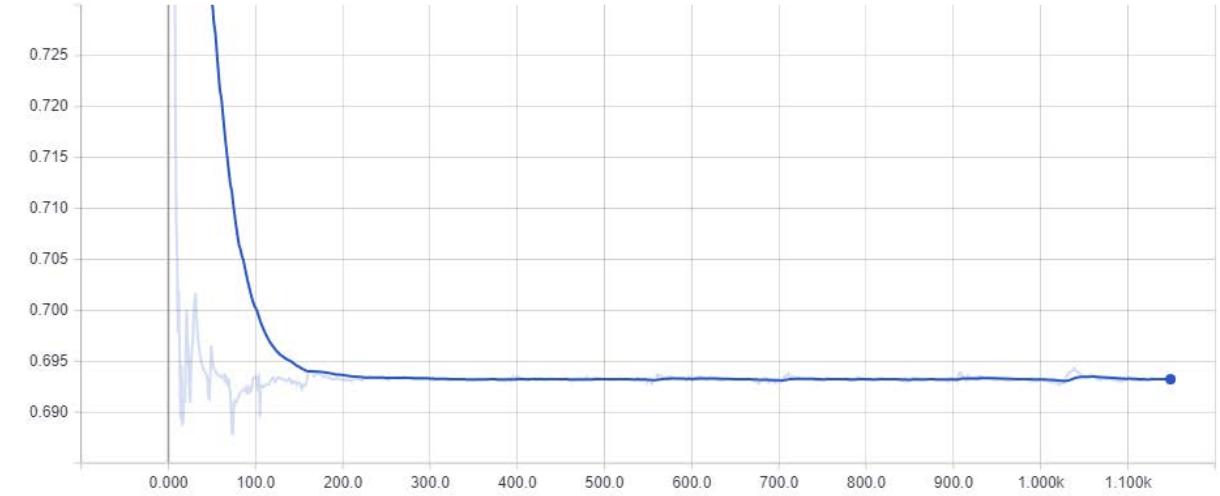


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-4-layer-epoch3.model	0.4952	0.4909	1.149k	Sun Apr 29, 13:57:42	6m 49s



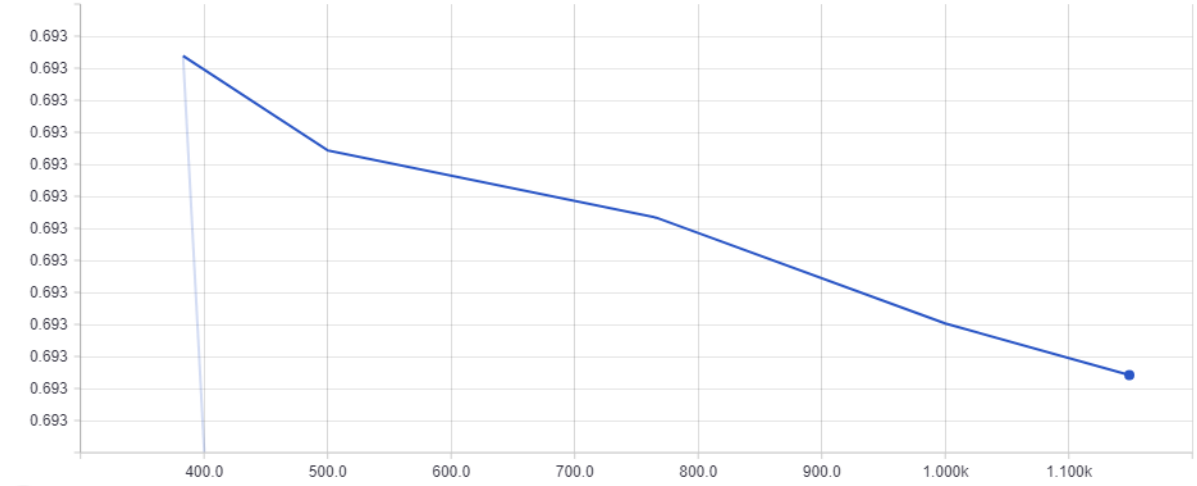
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-4-layer-epoch3.model	0.4948	0.4940	1.149k	Sun Apr 29, 13:57:42	4m 26s

Loss



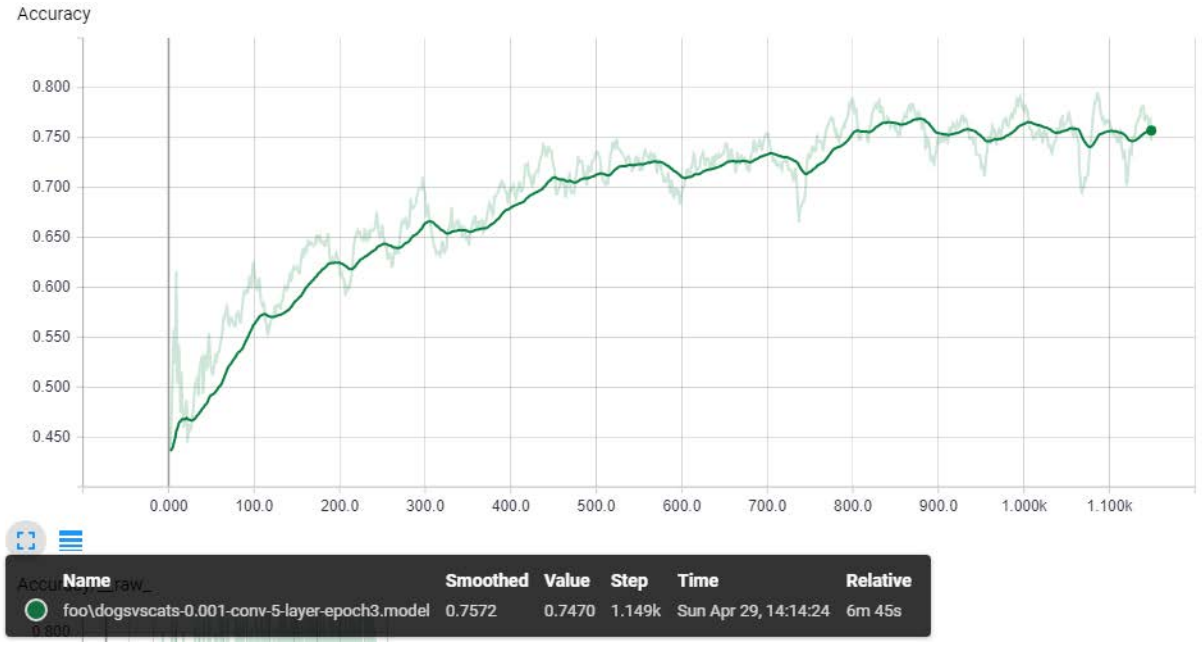
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-4-layer-epoch3.model	0.6932	0.6933	1.149k	Sun Apr 29, 13:57:42	6m 49s

Loss/Validation

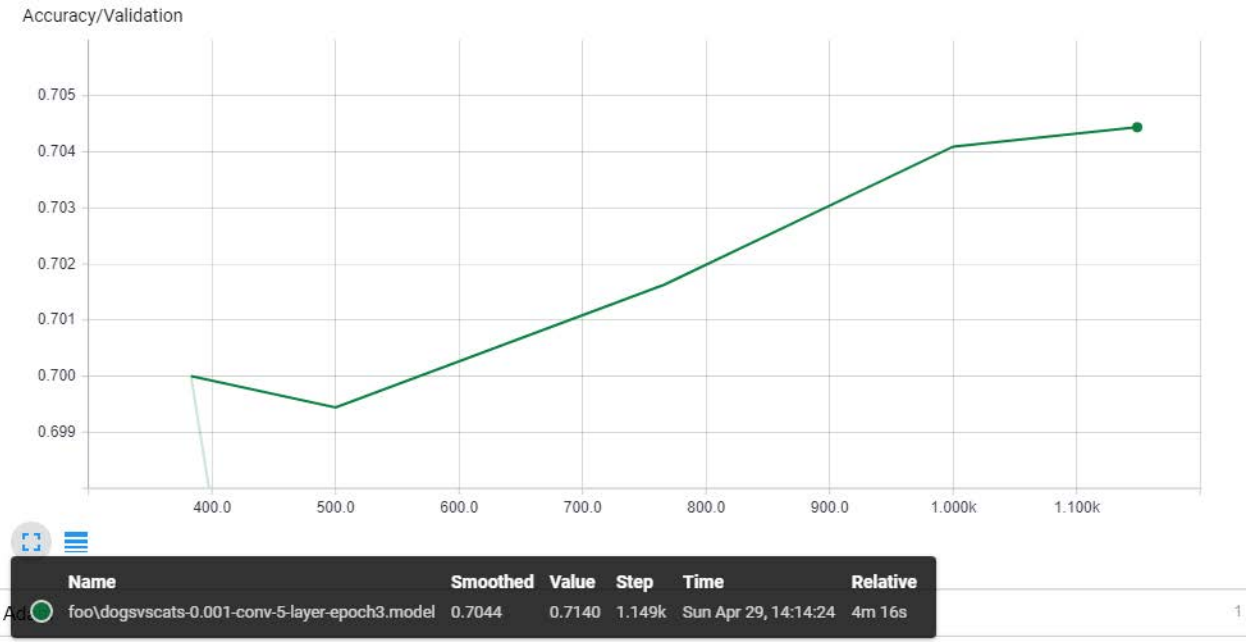


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-4-layer-epoch3.model	0.6933	0.6932	1.149k	Sun Apr 29, 13:57:42	4m 26s

6) Layer(5), Epoch(3):

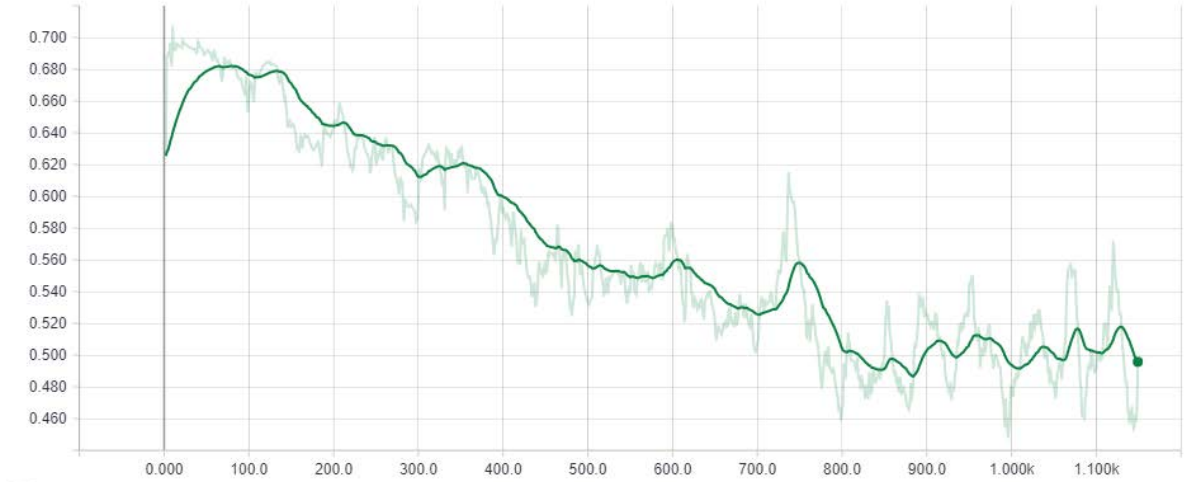


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch3.model	0.7572	0.7470	1.149k	Sun Apr 29, 14:14:24	6m 45s



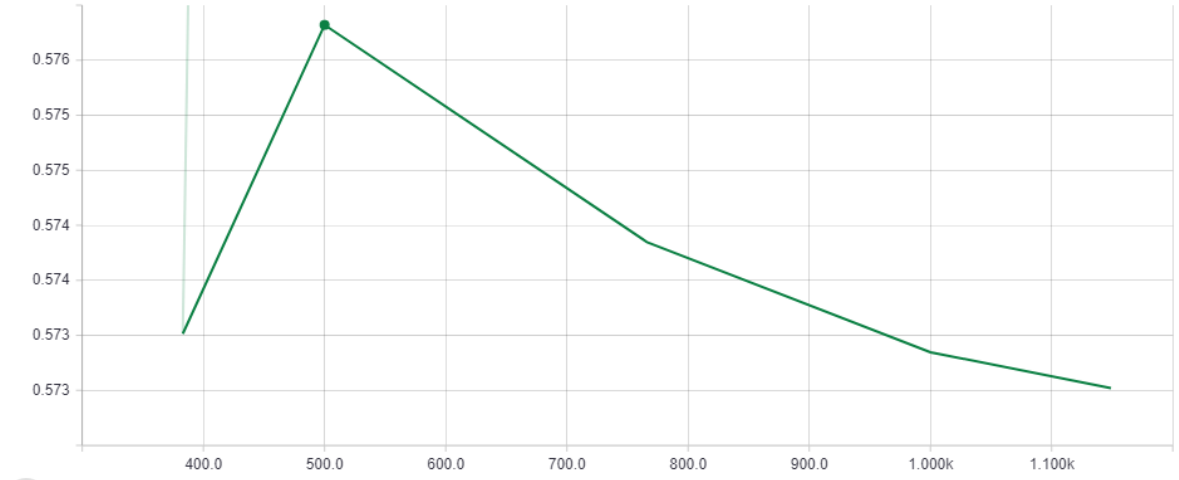
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch3.model	0.7044	0.7140	1.149k	Sun Apr 29, 14:14:24	4m 16s

Loss



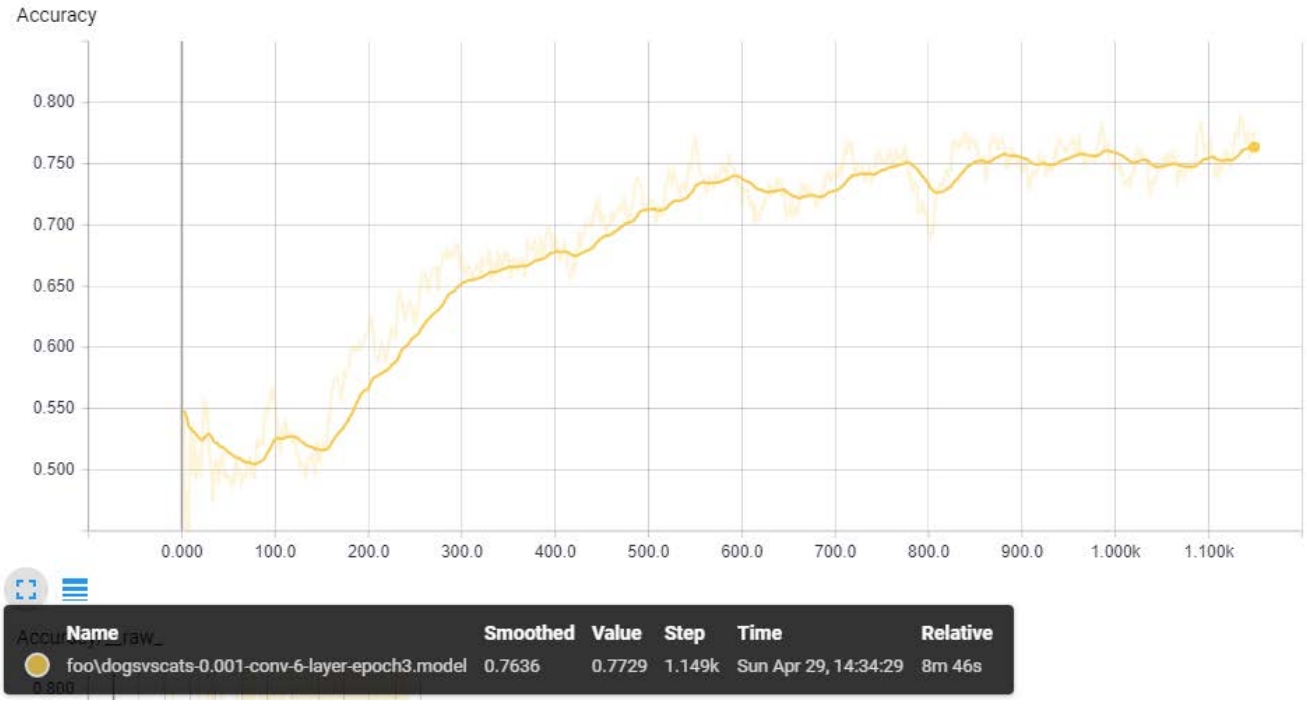
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch3.model	0.4957	0.4928	1.149k	Sun Apr 29, 14:14:24	6m 45s

Loss/Validation

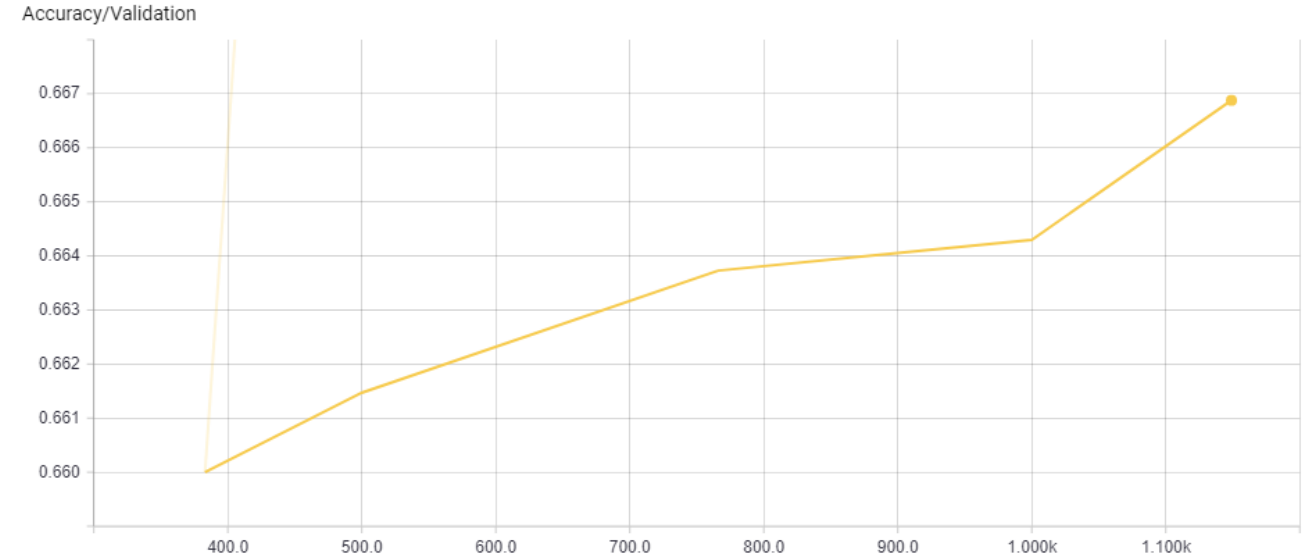


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch3.model	0.5758	0.6532	500.0	Sun Apr 29, 14:10:52	44s

7) Layer(6), Epoch(3):

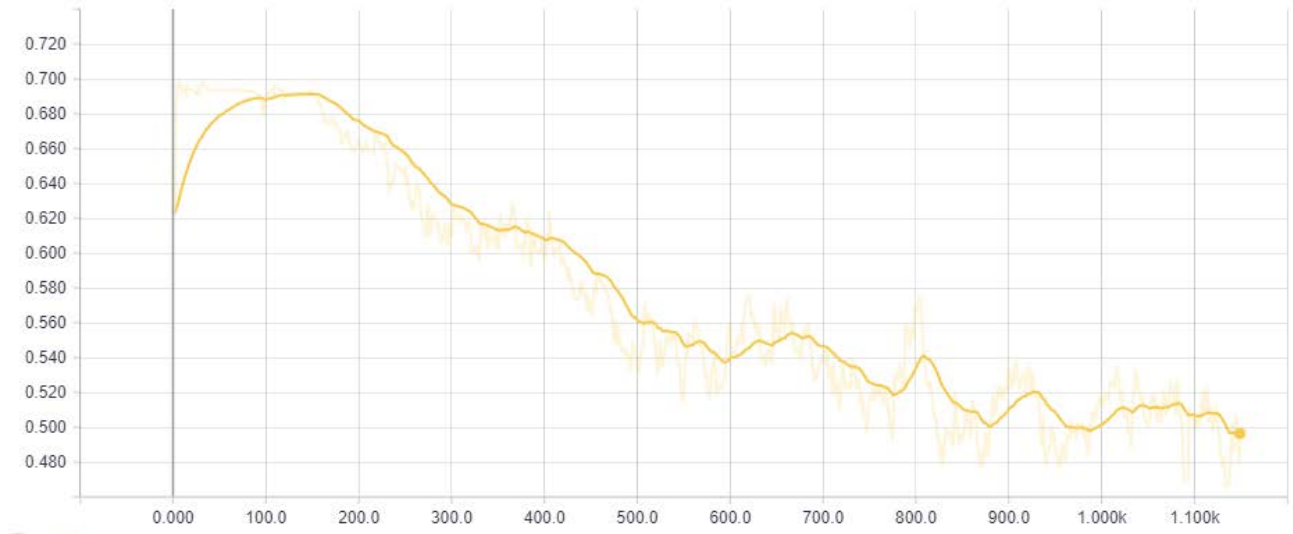


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-6-layer-epoch3.model	0.7636	0.7729	1.149k	Sun Apr 29, 14:34:29	8m 46s



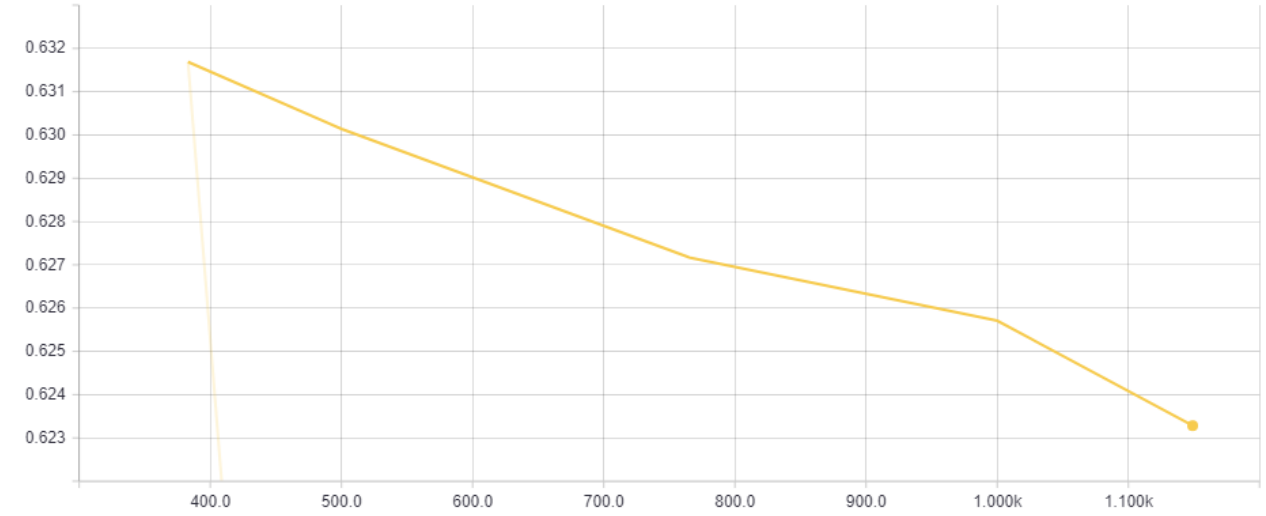
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-6-layer-epoch3.model	0.6669	0.7380	1.149k	Sun Apr 29, 14:34:29	5m 55s

Loss



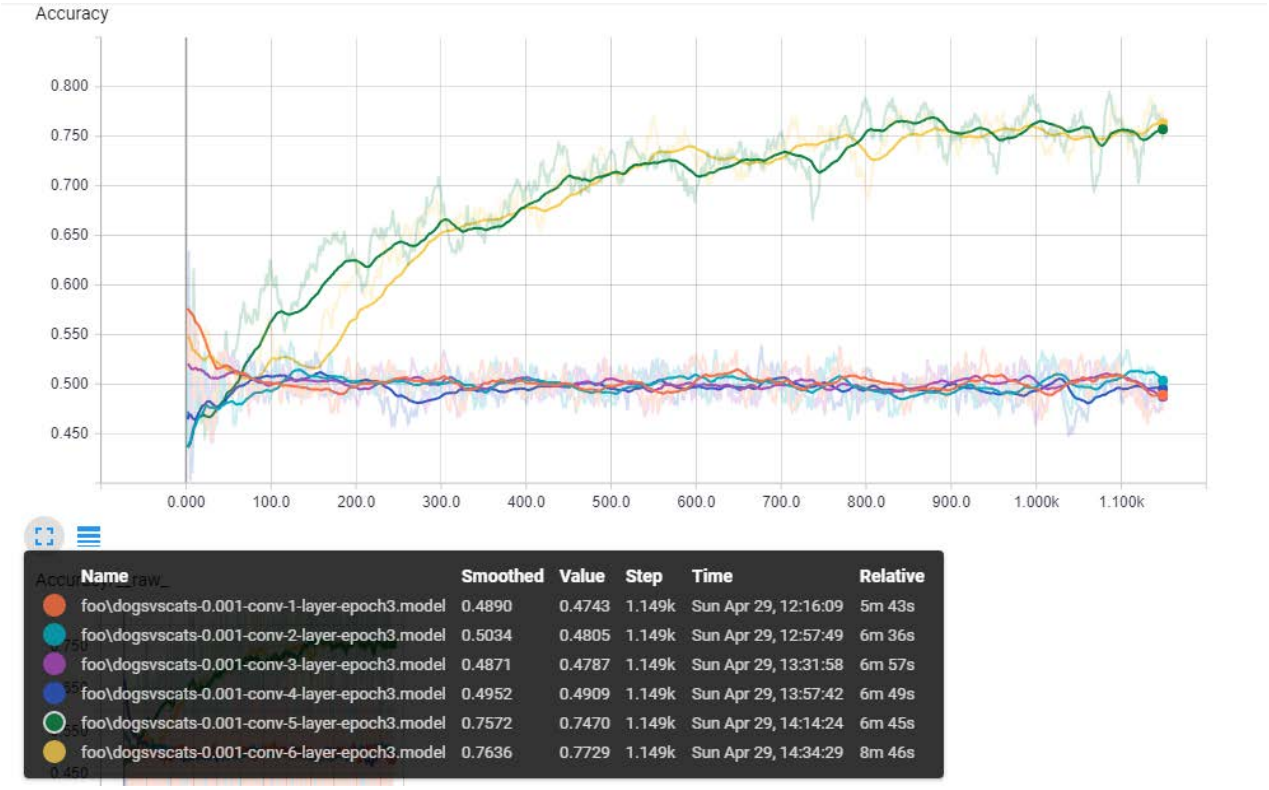
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-6-layer-epoch3.model	0.4964	0.4981	1.149k	Sun Apr 29, 14:34:29	8m 46s

Loss/Validation

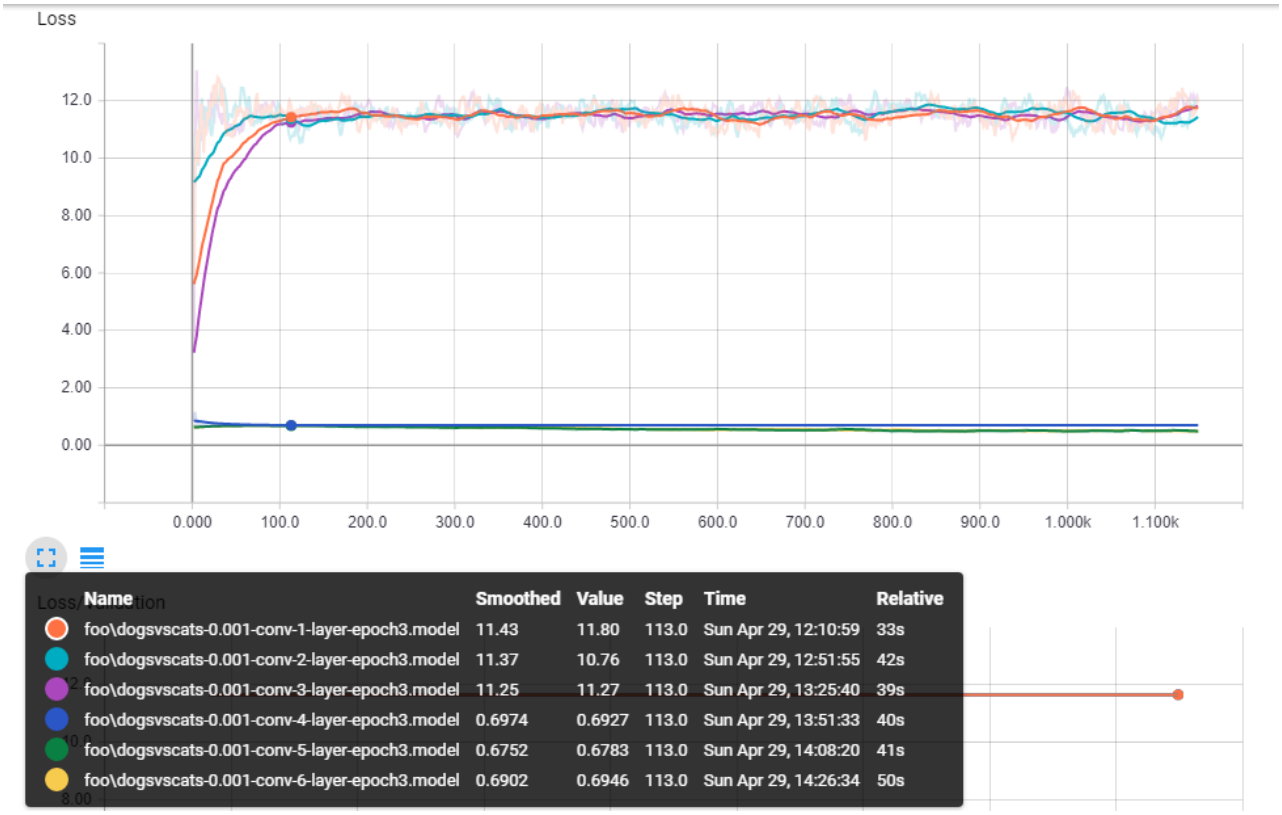


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-6-layer-epoch3.model	0.6233	0.5564	1.149k	Sun Apr 29, 14:34:29	5m 55s

8) Comparision among Layer(1,2,3,4,5,6), Epoch(3):

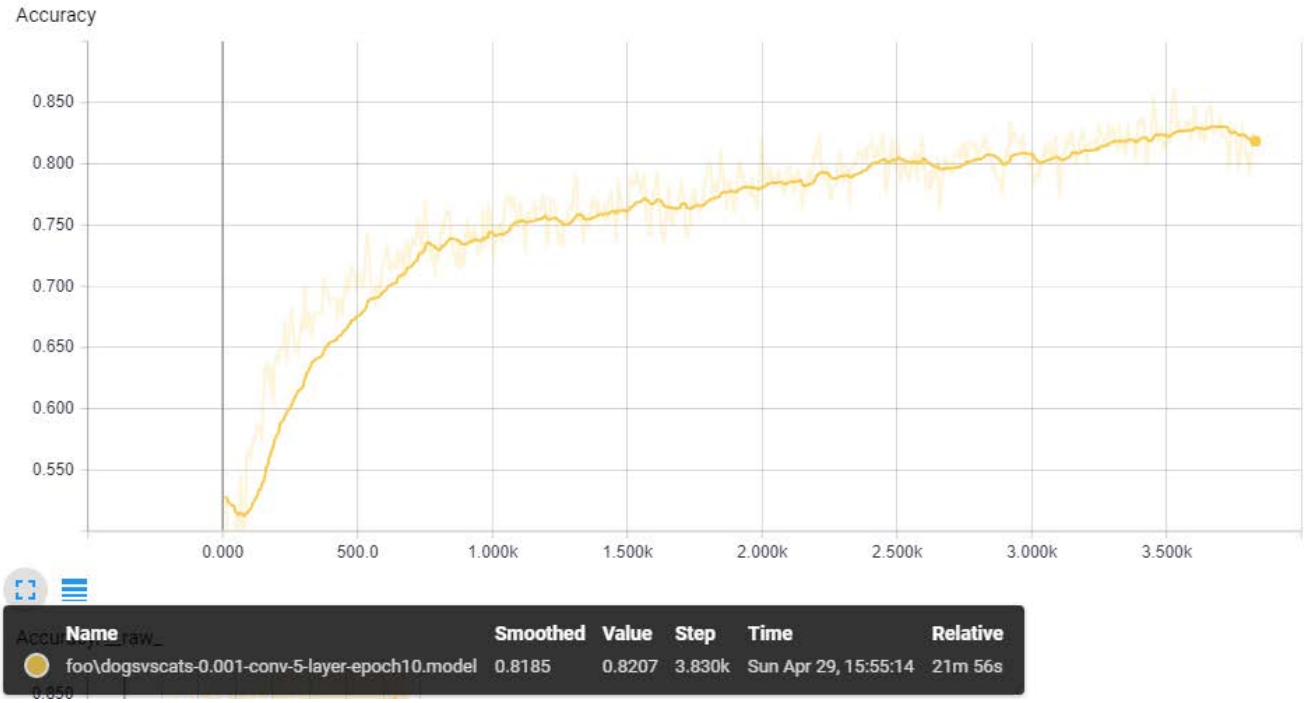


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-1-layer-epoch3.model	0.4890	0.4743	1.149k	Sun Apr 29, 12:16:09	5m 43s
foo\dogsvscats-0.001-conv-2-layer-epoch3.model	0.5034	0.4805	1.149k	Sun Apr 29, 12:57:49	6m 36s
foo\dogsvscats-0.001-conv-3-layer-epoch3.model	0.4871	0.4787	1.149k	Sun Apr 29, 13:31:58	6m 57s
foo\dogsvscats-0.001-conv-4-layer-epoch3.model	0.4952	0.4909	1.149k	Sun Apr 29, 13:57:42	6m 49s
foo\dogsvscats-0.001-conv-5-layer-epoch3.model	0.7572	0.7470	1.149k	Sun Apr 29, 14:14:24	6m 45s
foo\dogsvscats-0.001-conv-6-layer-epoch3.model	0.7636	0.7729	1.149k	Sun Apr 29, 14:34:29	8m 46s

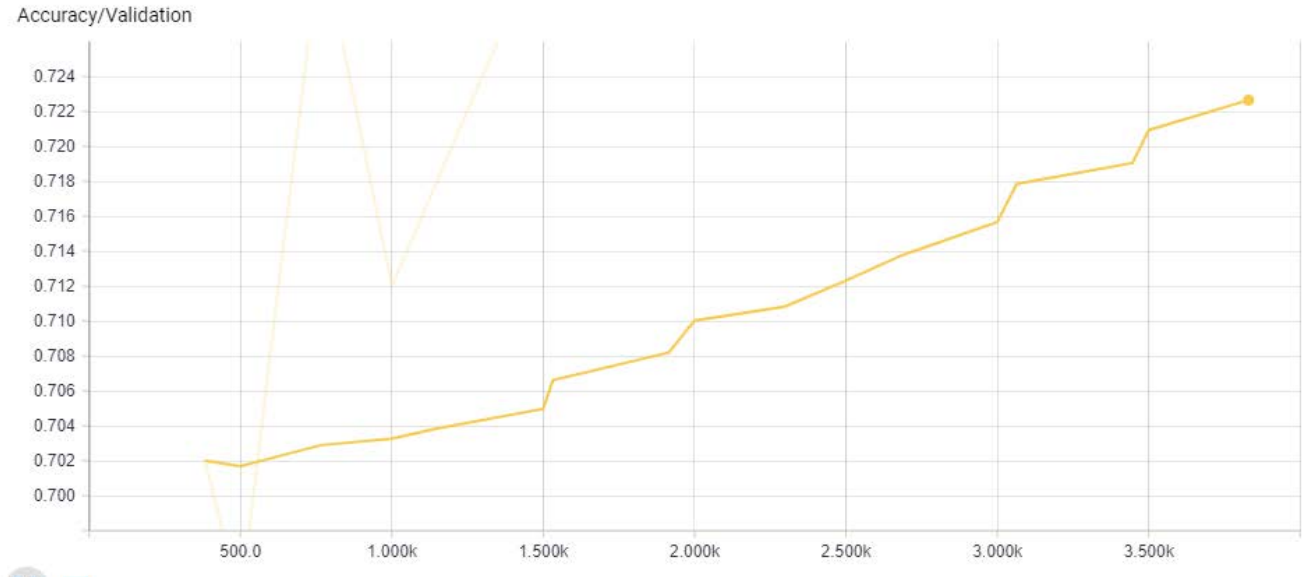


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-1-layer-epoch3.model	11.43	11.80	113.0	Sun Apr 29, 12:10:59	33s
foo\dogsvscats-0.001-conv-2-layer-epoch3.model	11.37	10.76	113.0	Sun Apr 29, 12:51:55	42s
foo\dogsvscats-0.001-conv-3-layer-epoch3.model	11.25	11.27	113.0	Sun Apr 29, 13:25:40	39s
foo\dogsvscats-0.001-conv-4-layer-epoch3.model	0.6974	0.6927	113.0	Sun Apr 29, 13:51:33	40s
foo\dogsvscats-0.001-conv-5-layer-epoch3.model	0.6752	0.6783	113.0	Sun Apr 29, 14:08:20	41s
foo\dogsvscats-0.001-conv-6-layer-epoch3.model	0.6902	0.6946	113.0	Sun Apr 29, 14:26:34	50s

9) Layer(5), Epoch(10):

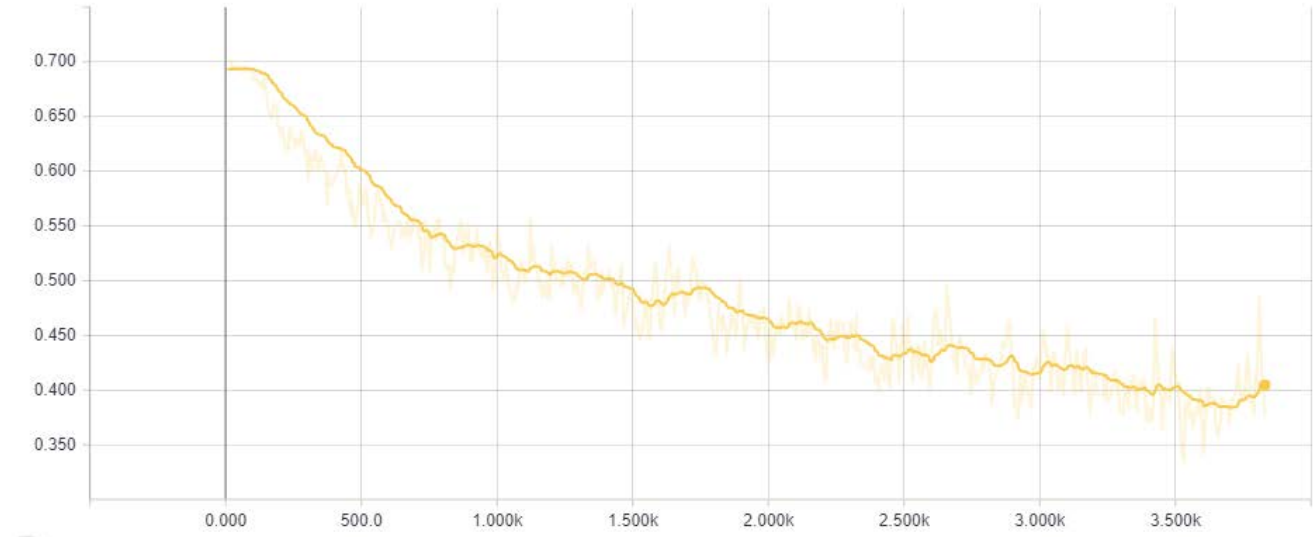


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch10.model	0.8185	0.8207	3.830k	Sun Apr 29, 15:55:14	21m 56s



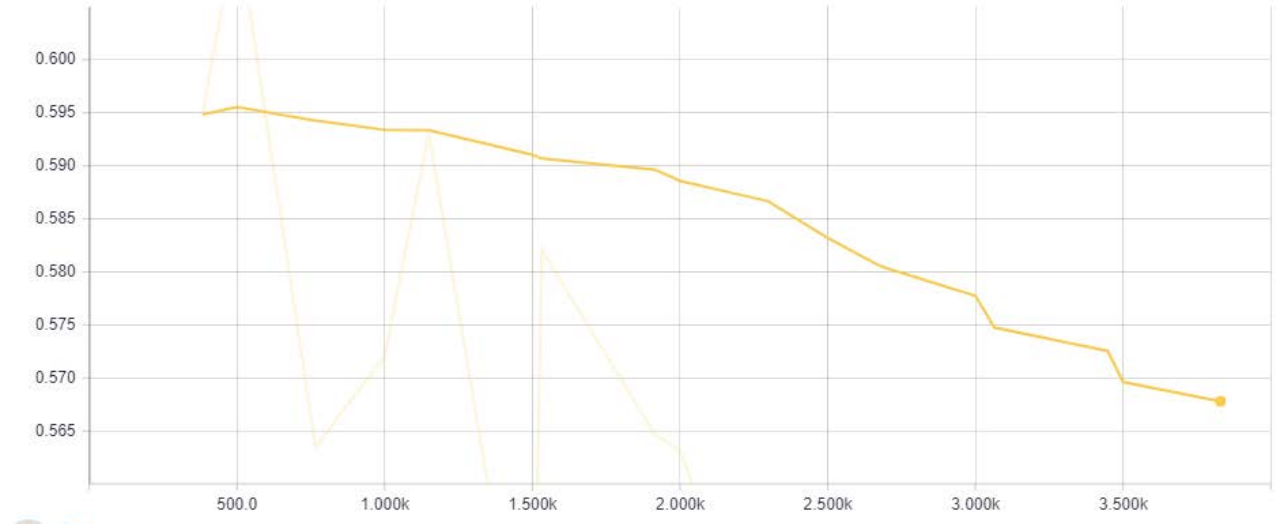
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch10.model	0.7226	0.7640	3.830k	Sun Apr 29, 15:55:14	19m 47s

Loss



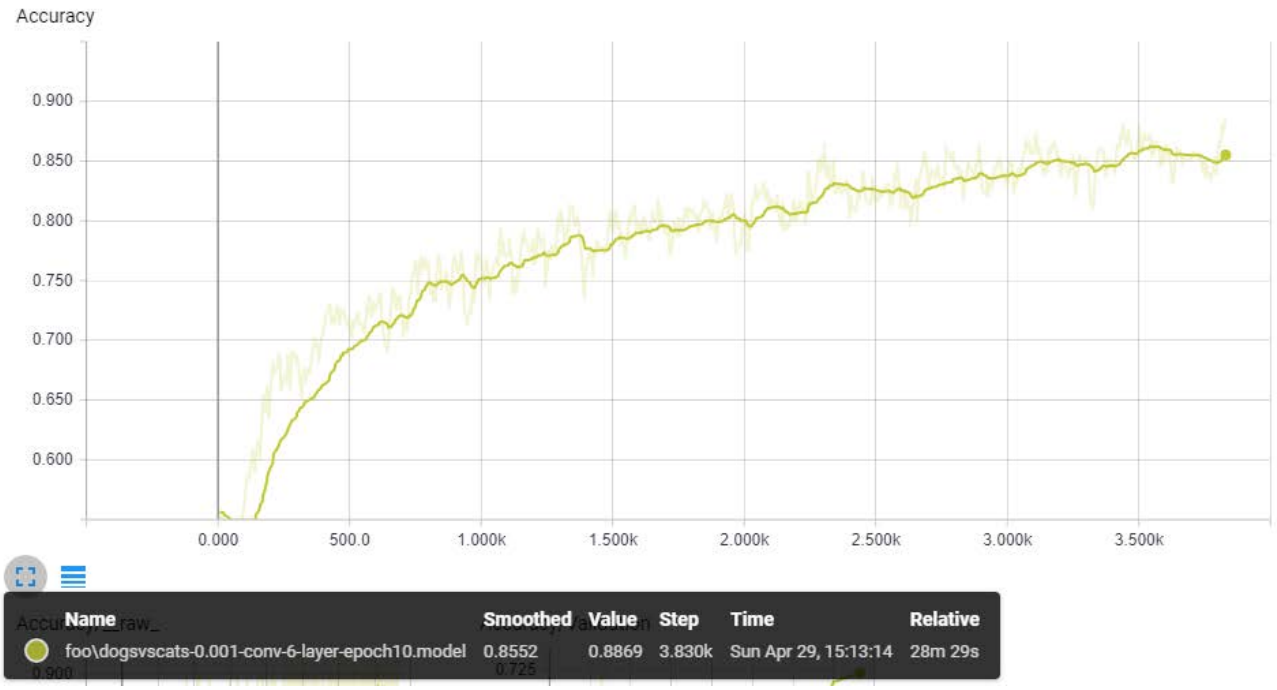
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch10.model	0.4046	0.3757	3.830k	Sun Apr 29, 15:55:14	21m 56s

Loss/Validation

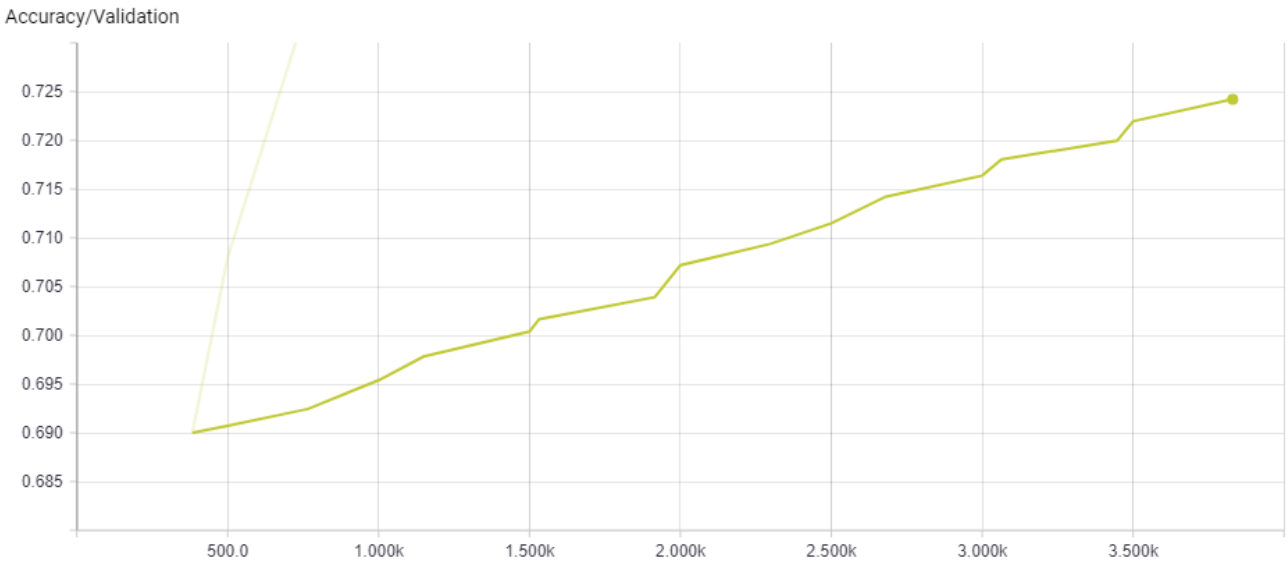


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch10.model	0.5678	0.5240	3.830k	Sun Apr 29, 15:55:14	19m 47s

10) Layer(6), Epoch(10):

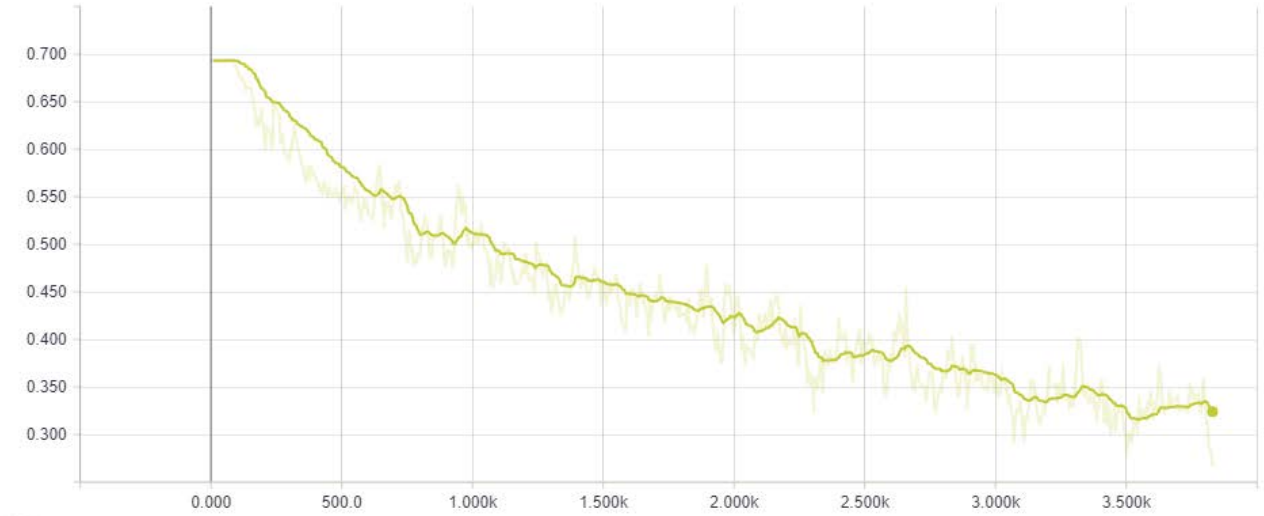


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-6-layer-epoch10.model	0.8552	0.8869	3.830k	Sun Apr 29, 15:13:14	28m 29s



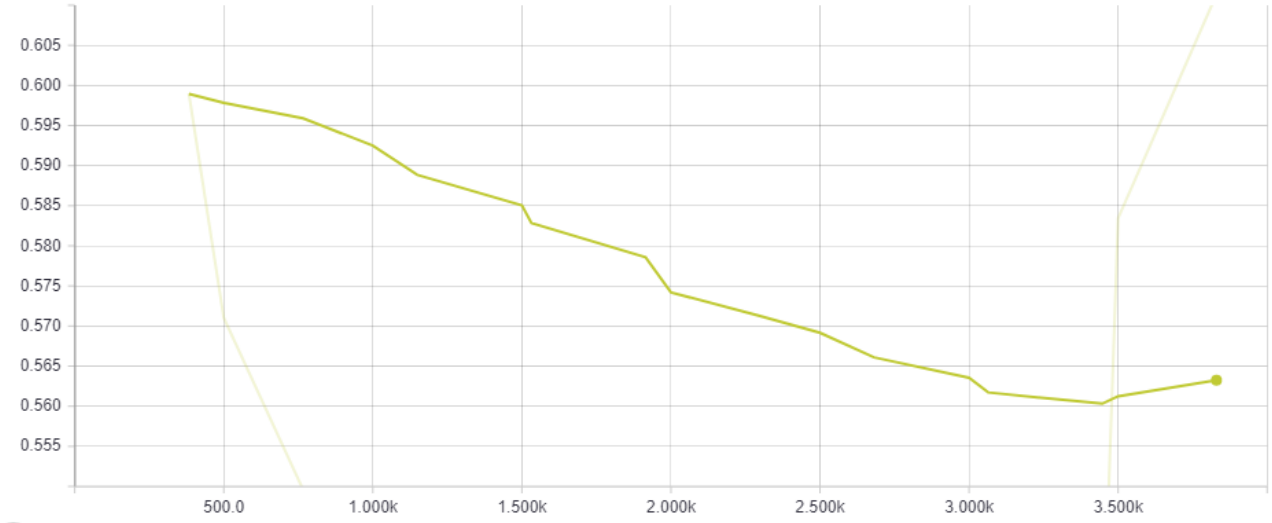
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-6-layer-epoch10.model	0.7242	0.7780	3.830k	Sun Apr 29, 15:13:14	25m 46s

Loss



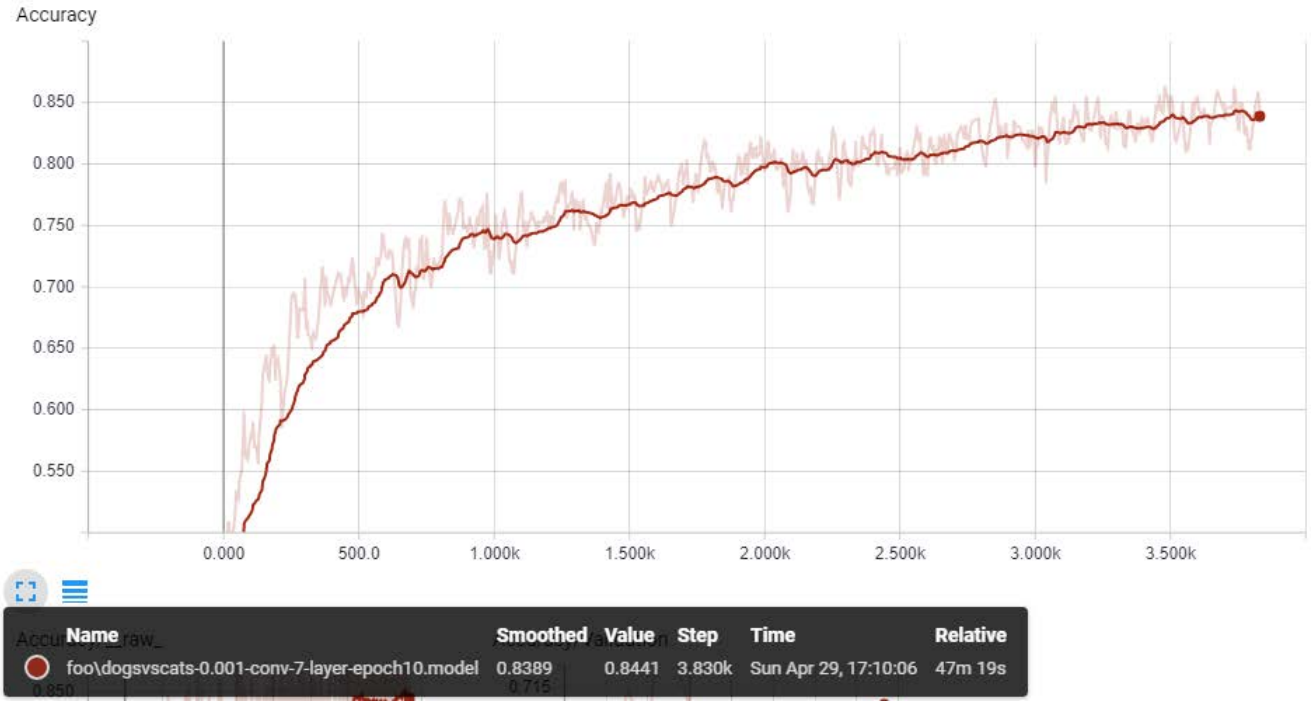
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-6-layer-epoch10.model	0.3240	0.2665	3.830k	Sun Apr 29, 15:13:14	28m 29s

Loss/Validation

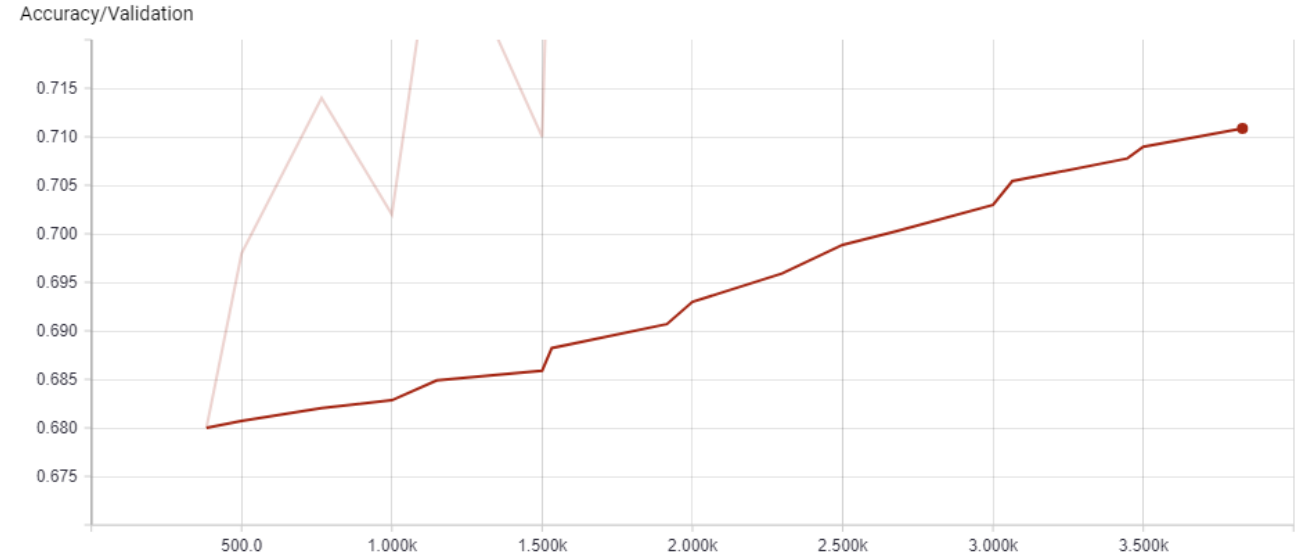


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-6-layer-epoch10.model	0.5632	0.6113	3.830k	Sun Apr 29, 15:13:14	25m 46s

11) Layer(7), Epoch(10):

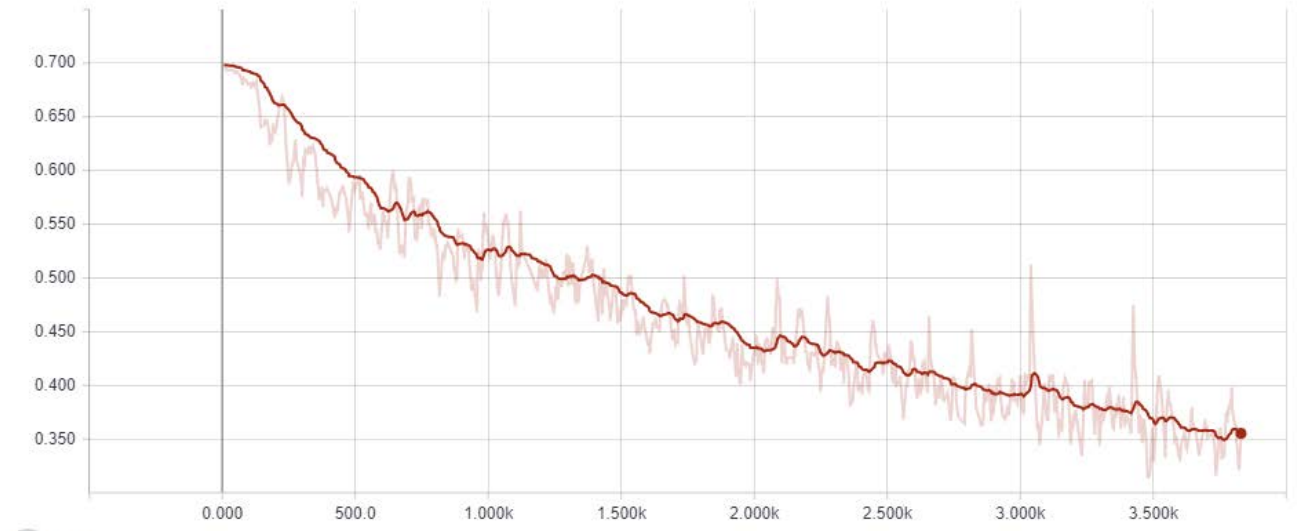


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-7-layer-epoch10.model	0.8389	0.8441	3.830k	Sun Apr 29, 17:10:06	47m 19s



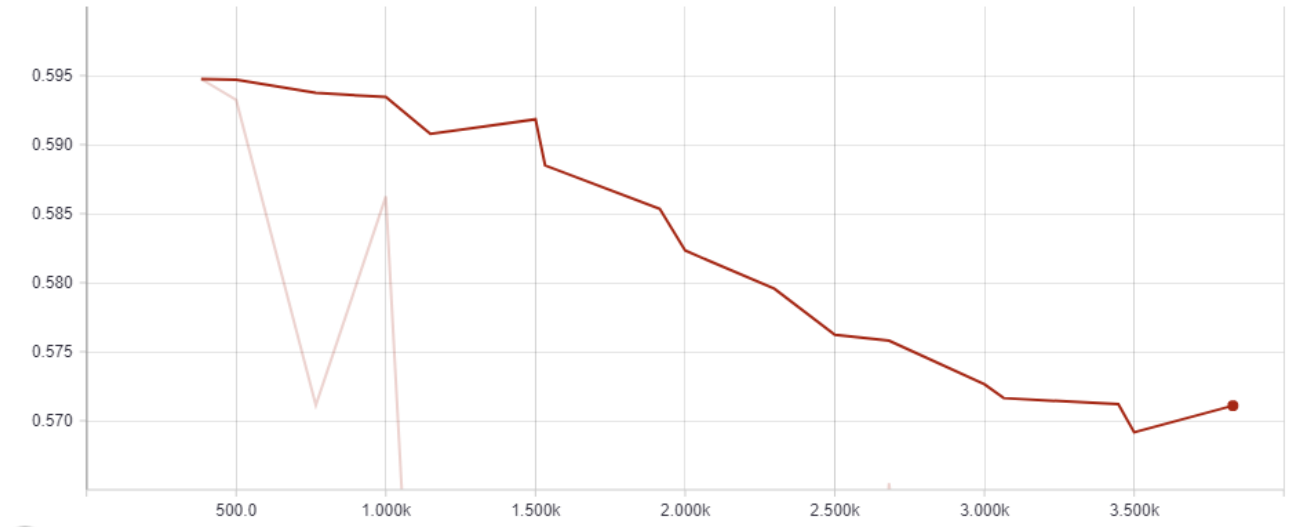
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-7-layer-epoch10.model	0.7109	0.7560	3.830k	Sun Apr 29, 17:10:06	42m 14s

Loss



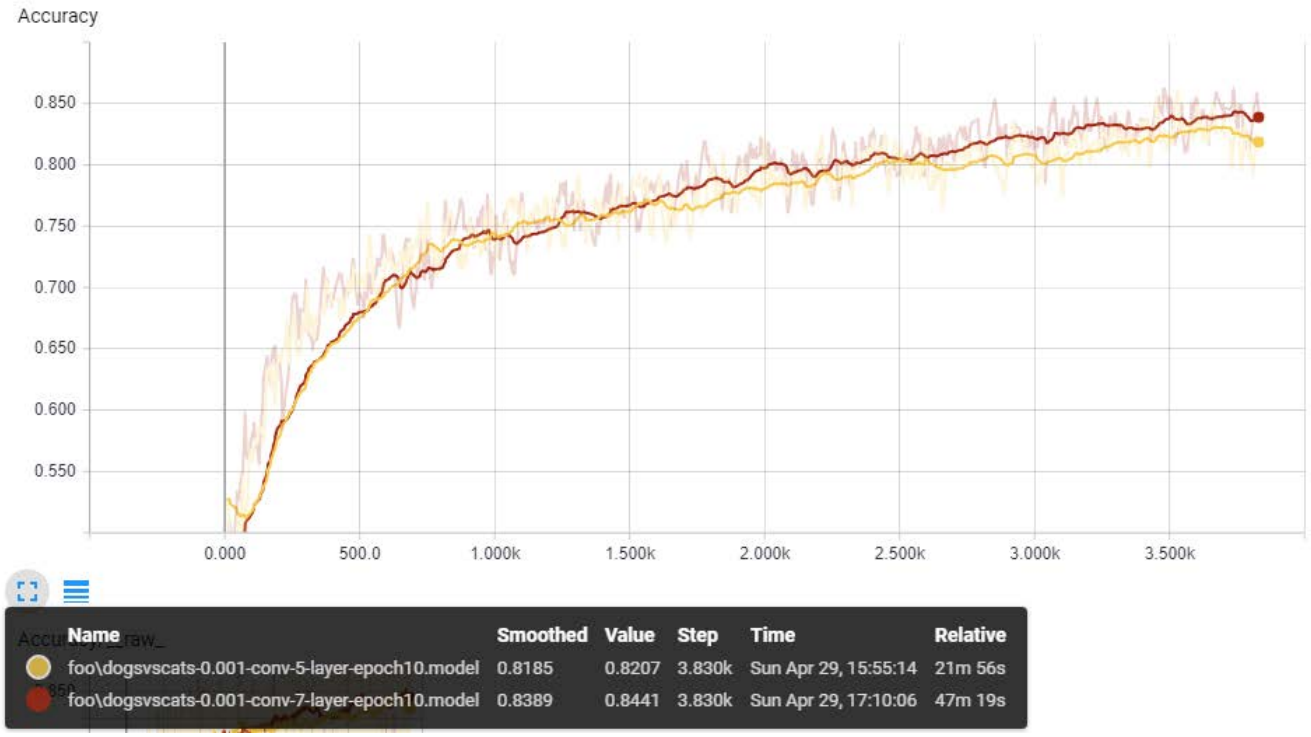
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-7-layer-epoch10.model	0.3553	0.3531	3.830k	Sun Apr 29, 17:10:06	47m 19s

Loss/Validation

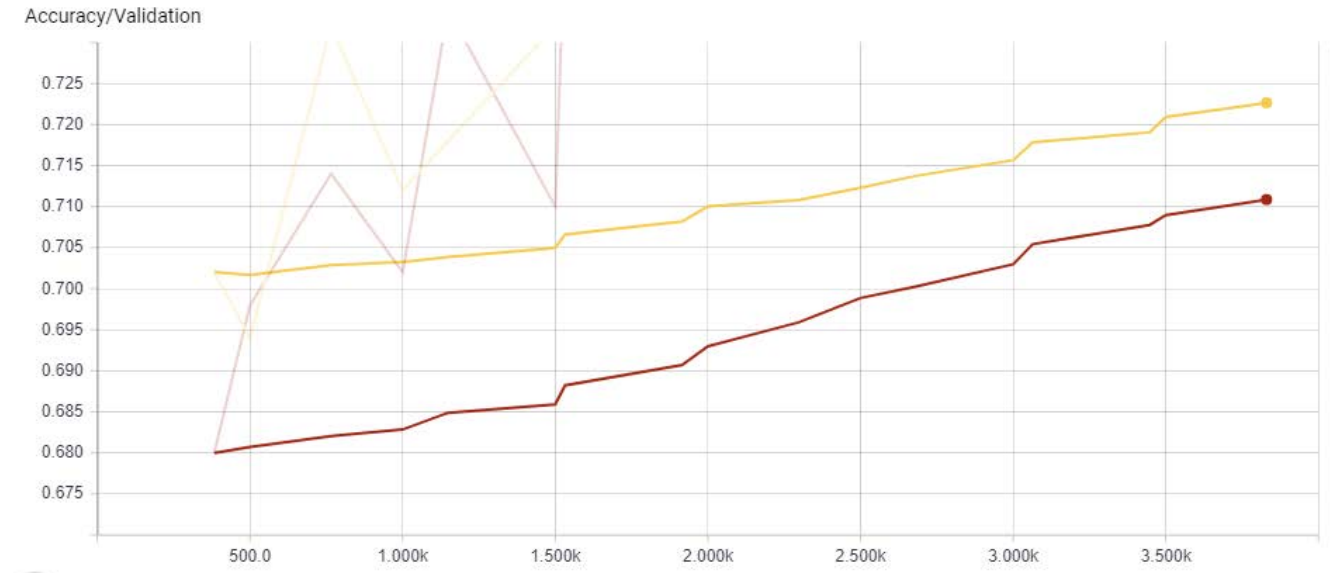


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-7-layer-epoch10.model	0.5711	0.6174	3.830k	Sun Apr 29, 17:10:06	42m 14s

12) Comparision among Layer(1,2,3,4,5,6), Epoch(3):

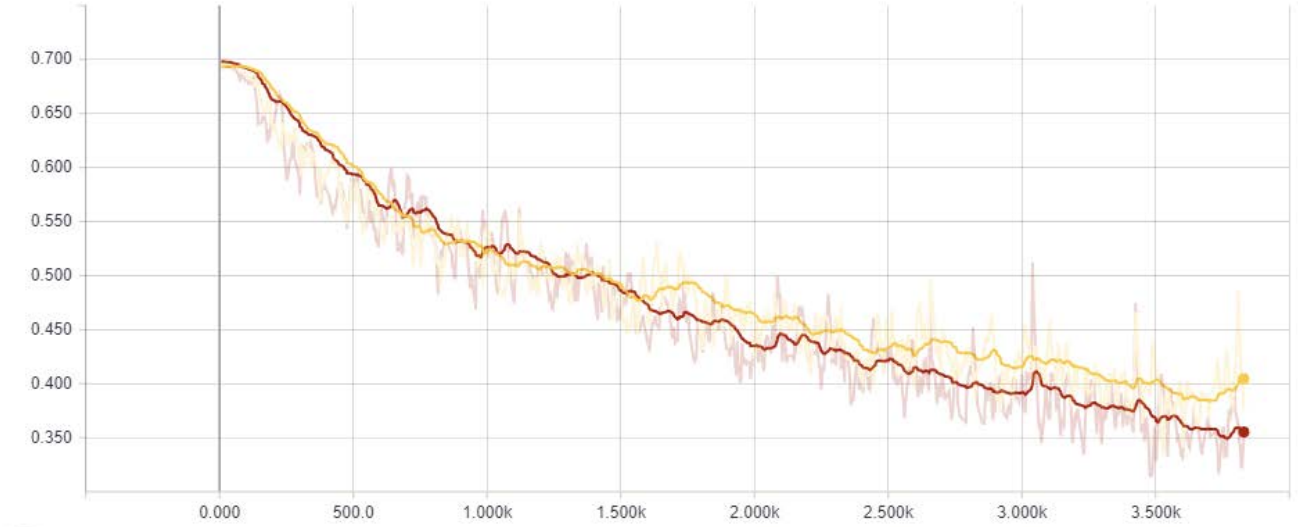


Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch10.model	0.8185	0.8207	3.830k	Sun Apr 29, 15:55:14	21m 56s
foo\dogsvscats-0.001-conv-7-layer-epoch10.model	0.8389	0.8441	3.830k	Sun Apr 29, 17:10:06	47m 19s



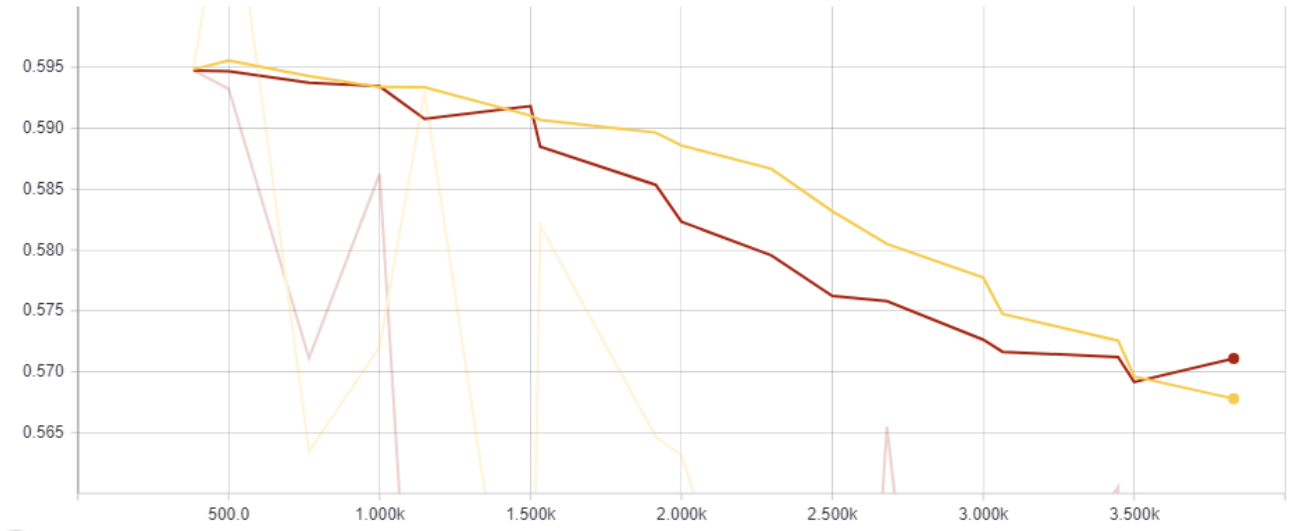
Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch10.model	0.7226	0.7640	3.830k	Sun Apr 29, 15:55:14	19m 47s
foo\dogsvscats-0.001-conv-7-layer-epoch10.model	0.7109	0.7560	3.830k	Sun Apr 29, 17:10:06	42m 14s

Loss



Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch10.model	0.4046	0.3757	3.830k	Sun Apr 29, 15:55:14	21m 56s
foo\dogsvscats-0.001-conv-7-layer-epoch10.model	0.3553	0.3531	3.830k	Sun Apr 29, 17:10:06	47m 19s

Loss/Validation



Name	Smoothed	Value	Step	Time	Relative
foo\dogsvscats-0.001-conv-5-layer-epoch10.model	0.5678	0.5240	3.830k	Sun Apr 29, 15:55:14	19m 47s
foo\dogsvscats-0.001-conv-7-layer-epoch10.model	0.5711	0.6174	3.830k	Sun Apr 29, 17:10:06	42m 14s