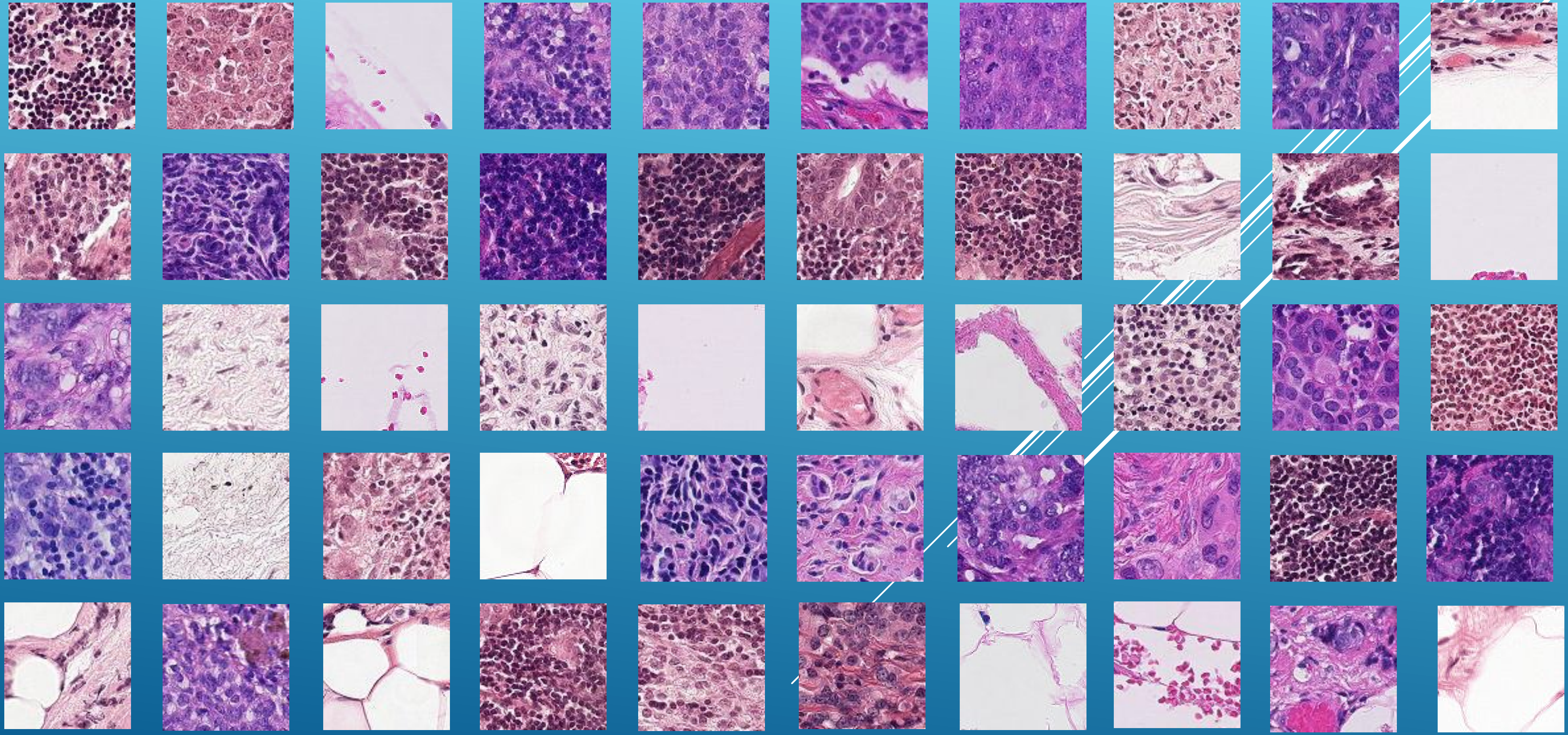
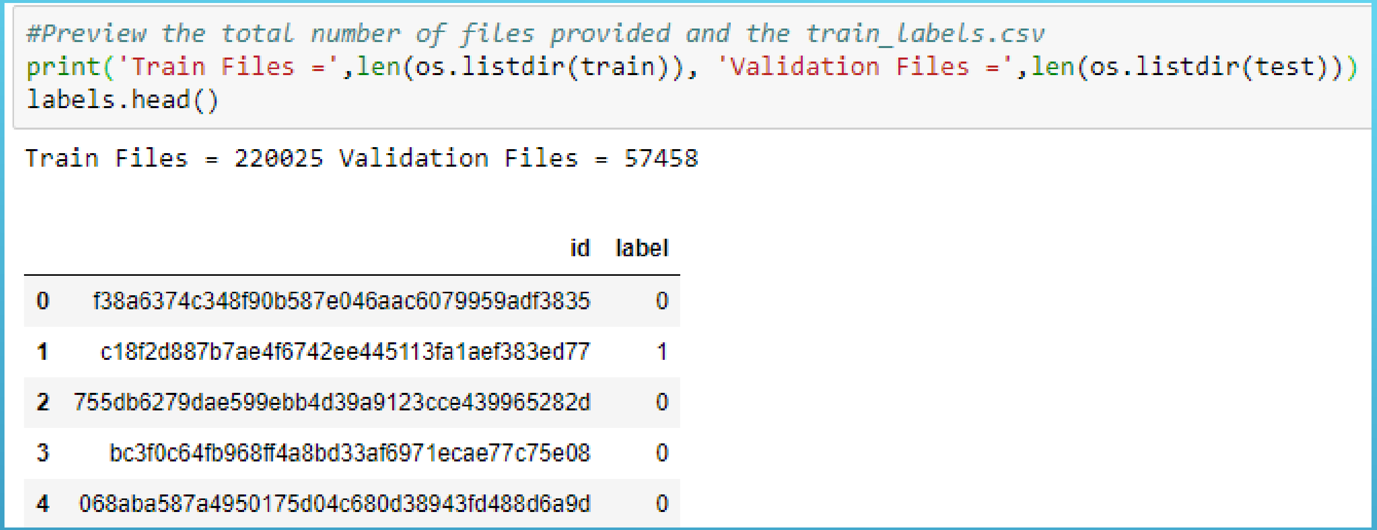
HISTOPATHOLOGIC CANCER DETECTION

https://www.kaggle.com/c/histopathologic-cancer-detection



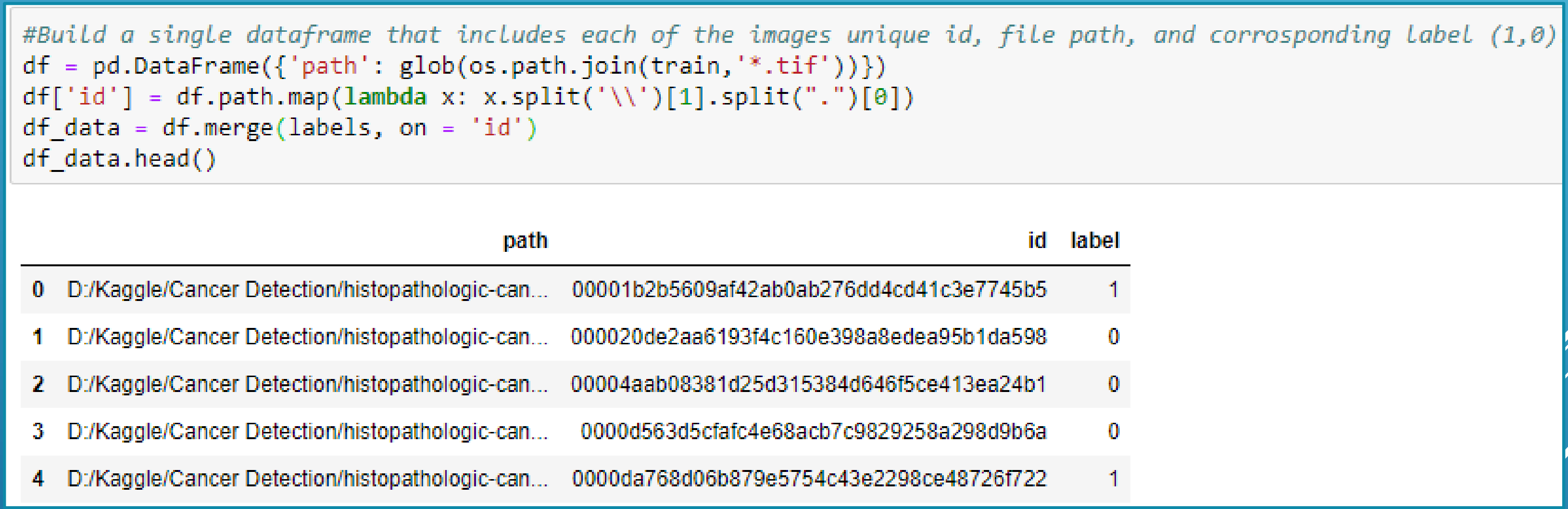
DATA STRUCTURE AND PIPELINE



```
#Preview the total number of files provided and the train_labels.csv
print('Train Files =',len(os.listdir(train)), 'Validation Files =',len(os.listdir(test)))
labels.head()
```

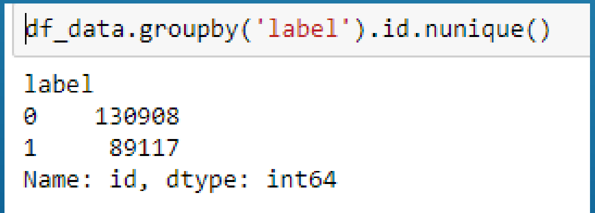
```
Train Files = 220025 Validation Files = 57458
```

	id	label
0	f38a6374c348f90b587e046aac6079959adf3835	0
1	c18f2d887b7ae4f6742ee445113fa1aef383ed77	1
2	755db6279dae599ebb4d39a9123cce439965282d	0
3	bc3f0c64fb968ff4a8bd33af6971ecae77c75e08	0
4	068aba587a4950175d04c680d38943fd488d6a9d	0



```
#Build a single dataframe that includes each of the images unique id, file path, and corresponding label (1,0)
df = pd.DataFrame({'path': glob(os.path.join(train,'*.tif'))})
df['id'] = df.path.map(lambda x: x.split('\\')[1].split(".")[0])
df_data = df.merge(labels, on = 'id')
df_data.head()
```

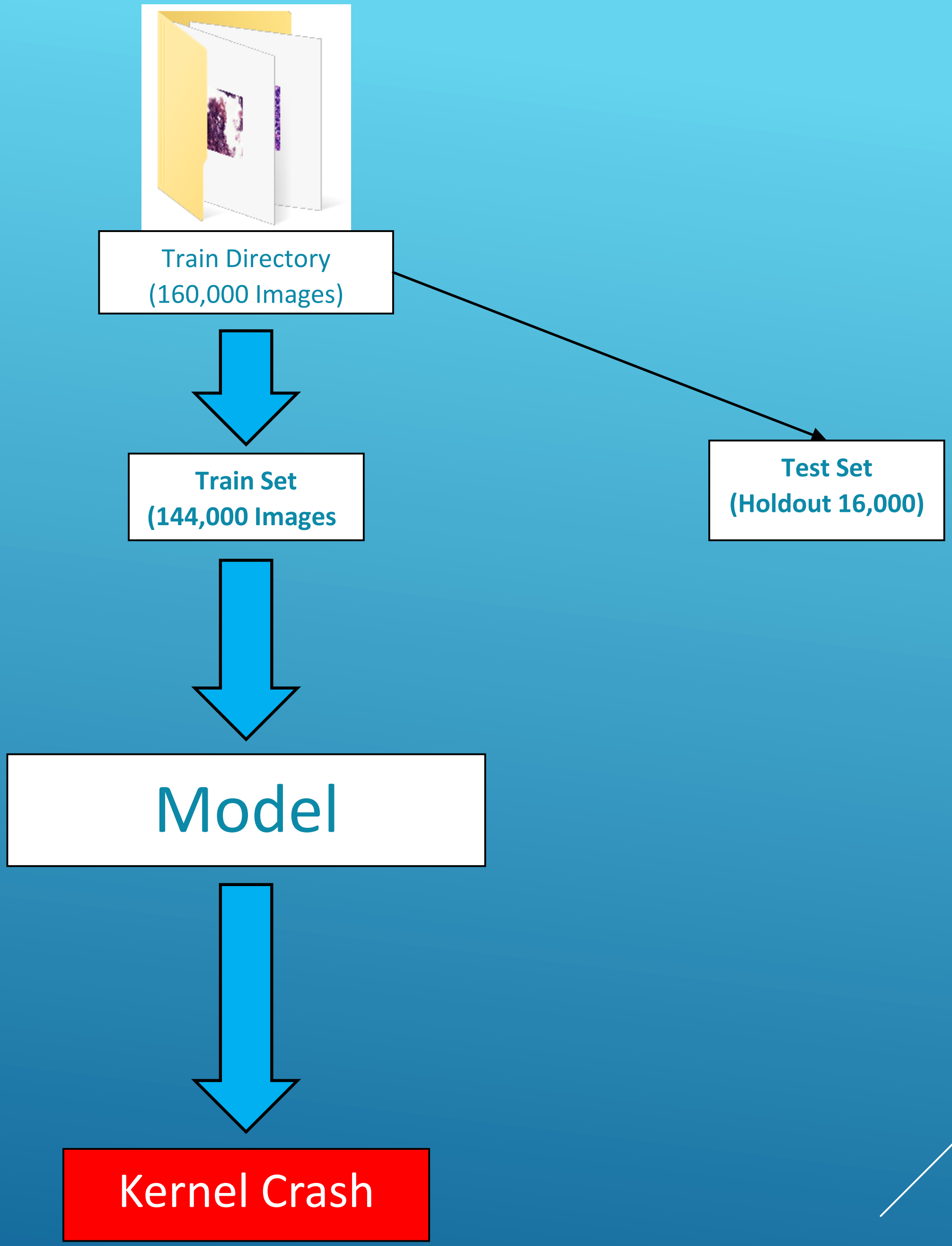
	path	id	label
0	D:/Kaggle/Cancer Detection/histopathologic-can...	00001b2b5609af42ab0ab276dd4cd41c3e7745b5	1
1	D:/Kaggle/Cancer Detection/histopathologic-can...	000020de2aa6193f4c160e398a8edea95b1da598	0
2	D:/Kaggle/Cancer Detection/histopathologic-can...	00004aab08381d25d315384d646f5ce413ea24b1	0
3	D:/Kaggle/Cancer Detection/histopathologic-can...	0000d563d5cfafc4e68acb7c9829258a298d9b6a	0
4	D:/Kaggle/Cancer Detection/histopathologic-can...	0000da768d06b879e5754c43e2298ce48726f722	1



```
df_data.groupby('label').id.nunique()
```

```
label
0    130908
1     89117
Name: id, dtype: int64
```

Naive Assumption = 40%



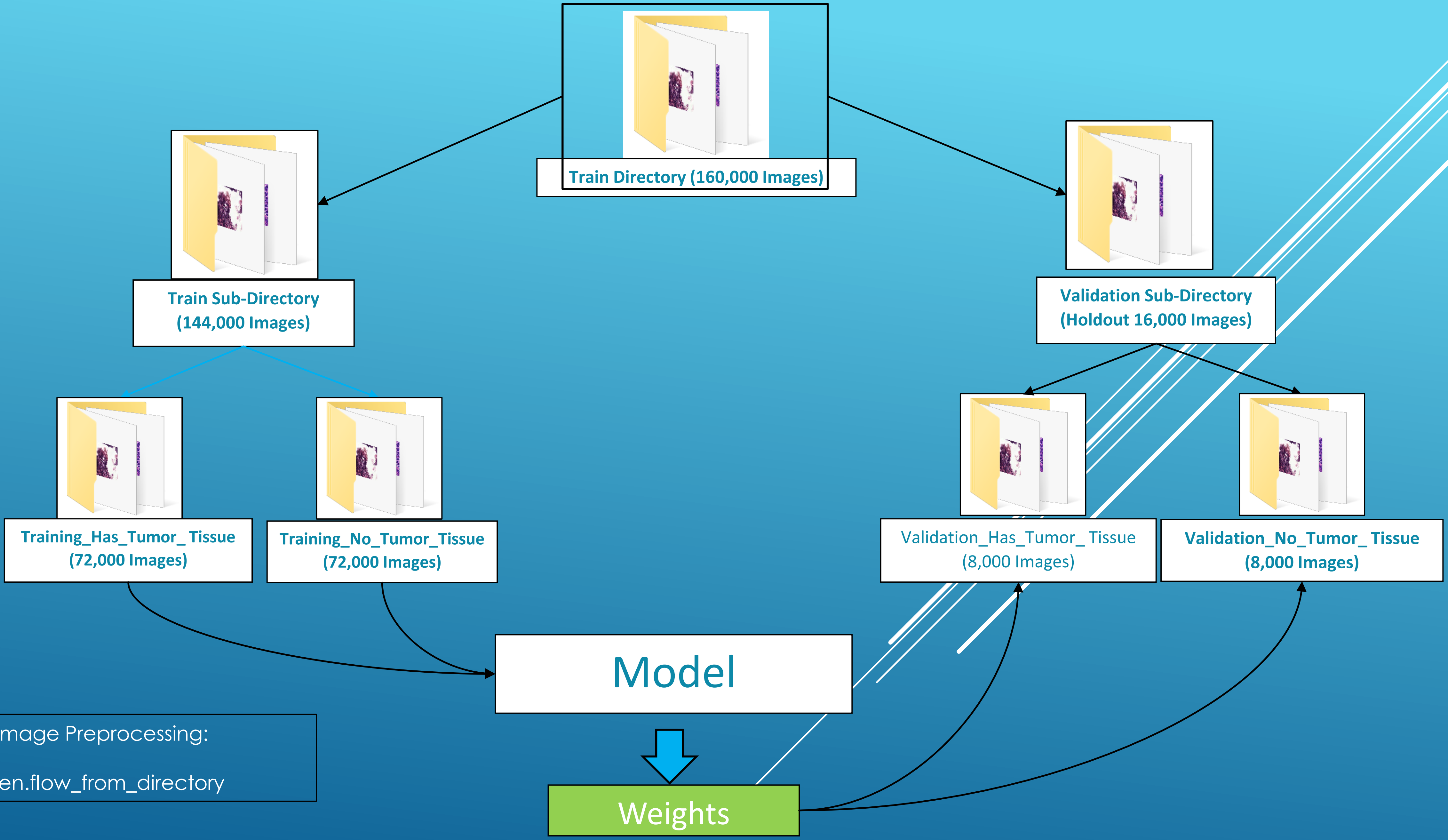
Train Directory
(160,000 Images)

Train Set
(144,000 Images

Test Set
(Holdout 16,000)

Model

Kernel Crash



Train Directory (160,000 Images)

Train Sub-Directory (144,000 Images)

Validation Sub-Directory (Holdout 16,000 Images)

Training_Has_Tumor_ Tissue (72,000 Images)

Training_No_Tumor_Tissue (72,000 Images)

Validation_Has_Tumor_ Tissue (8,000 Images)

Validation_No_Tumor_ Tissue (8,000 Images)

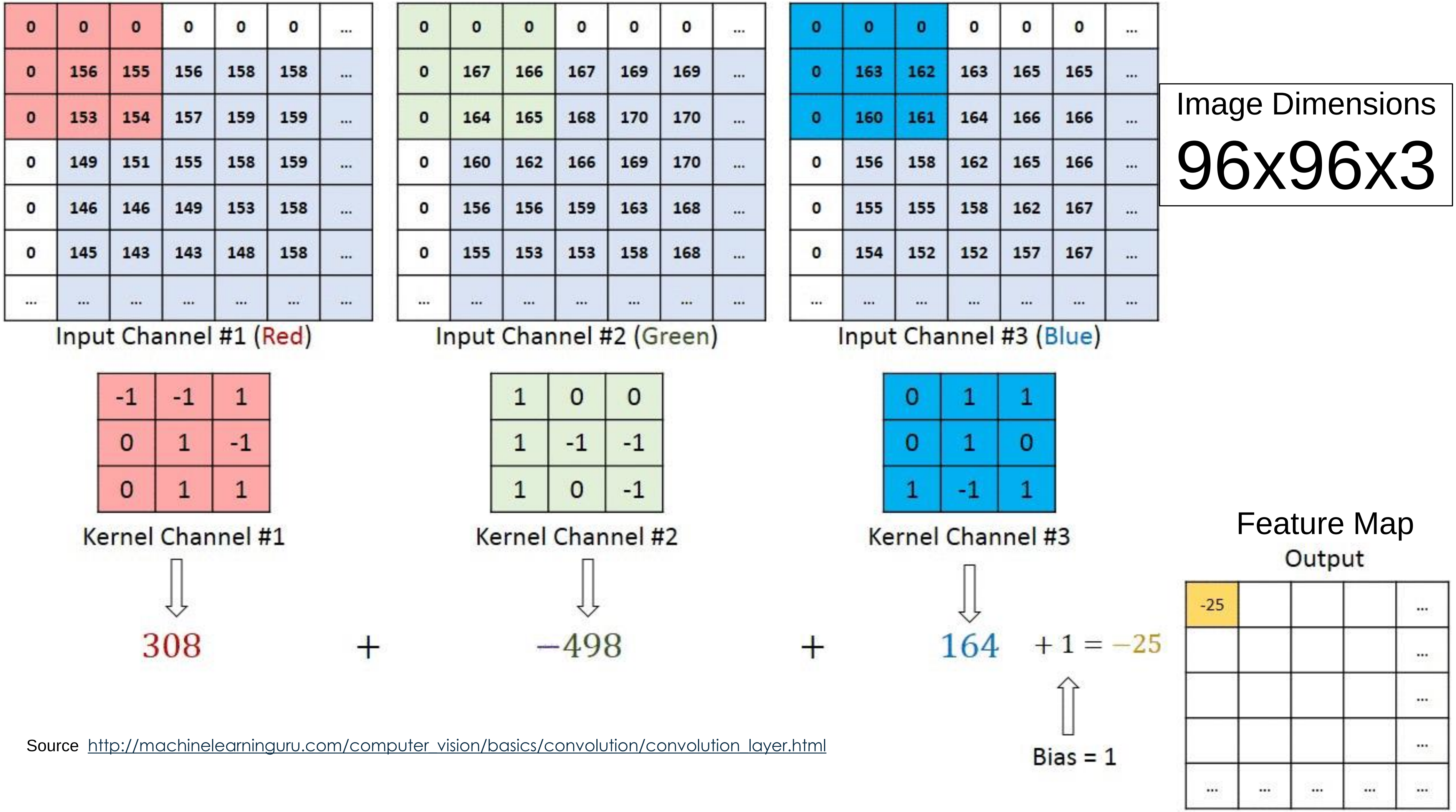
Model

Weights

Keras Image Preprocessing:

Datagen.flow_from_directory

Conceptualize and Build our Model



0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

Image Dimensions

96x96x3

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$$308 \qquad + \qquad -498 \qquad + \qquad 164 \quad + 1 = -25$$

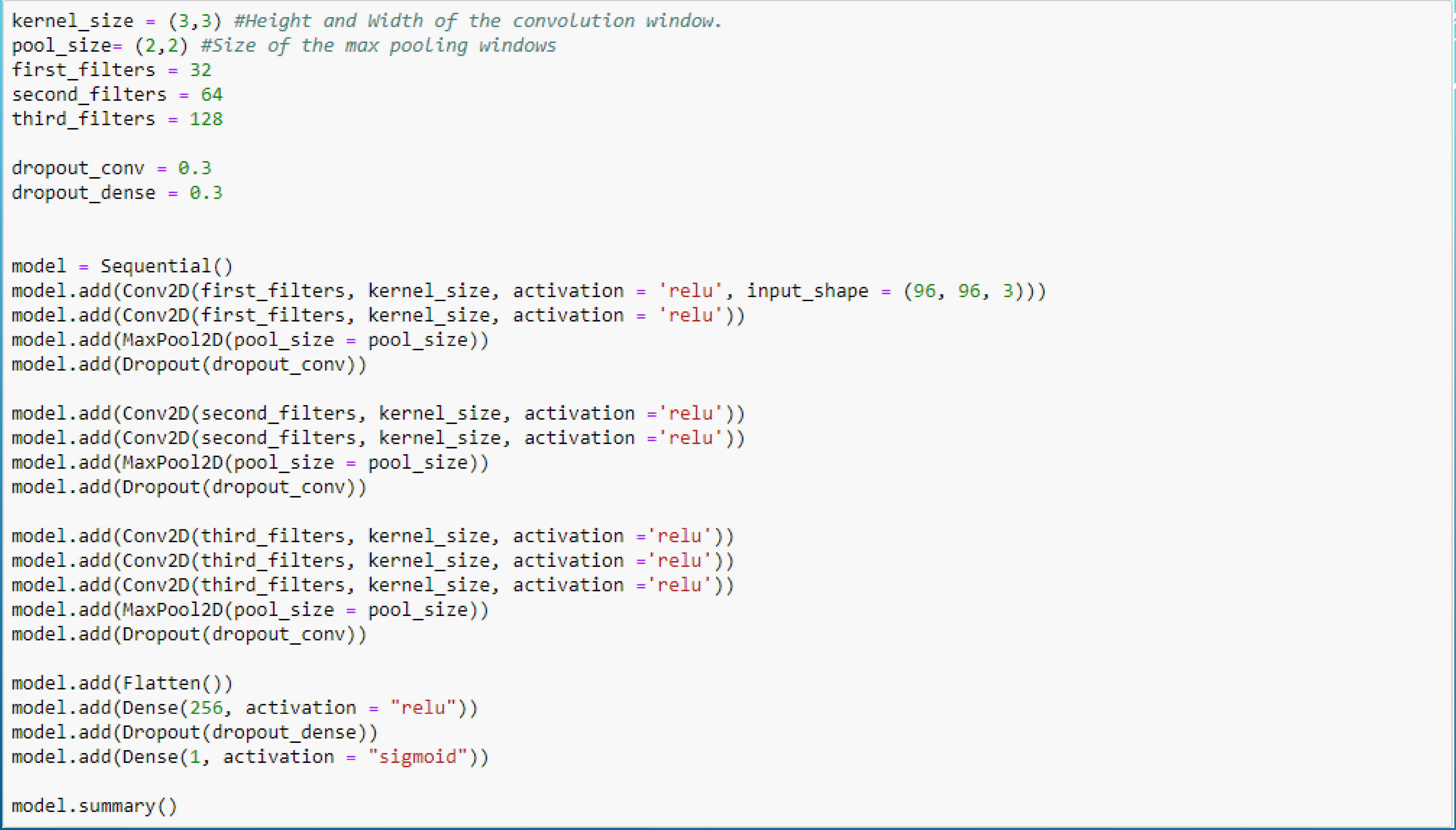
Bias = 1

Feature Map

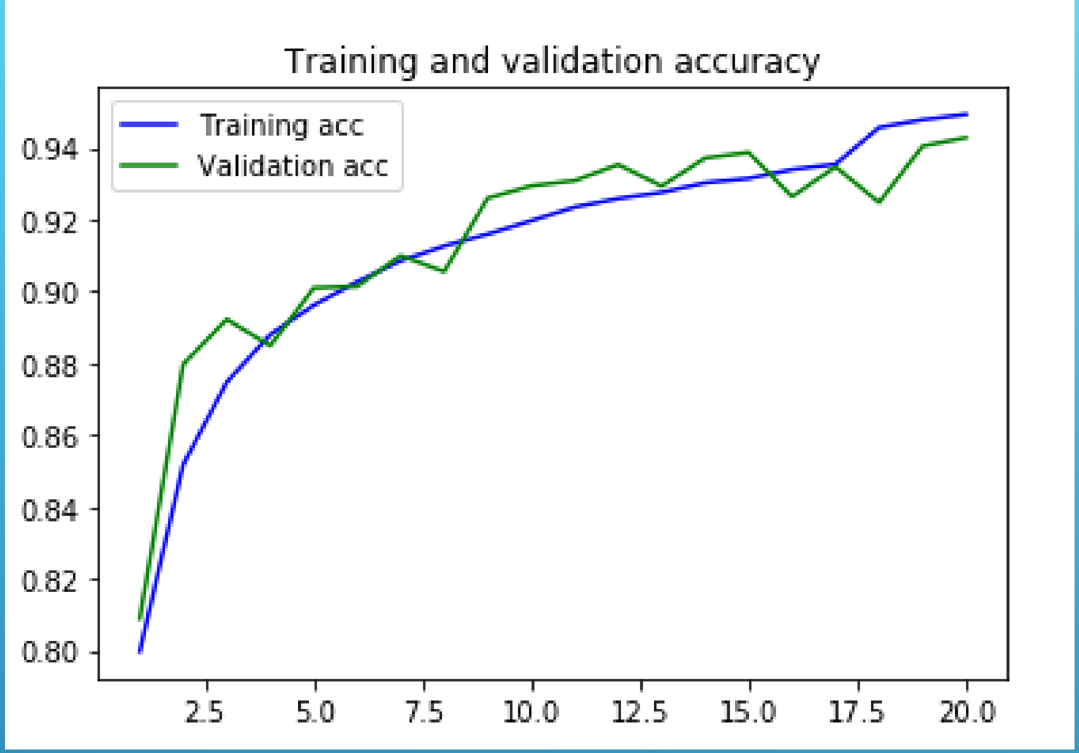
Output

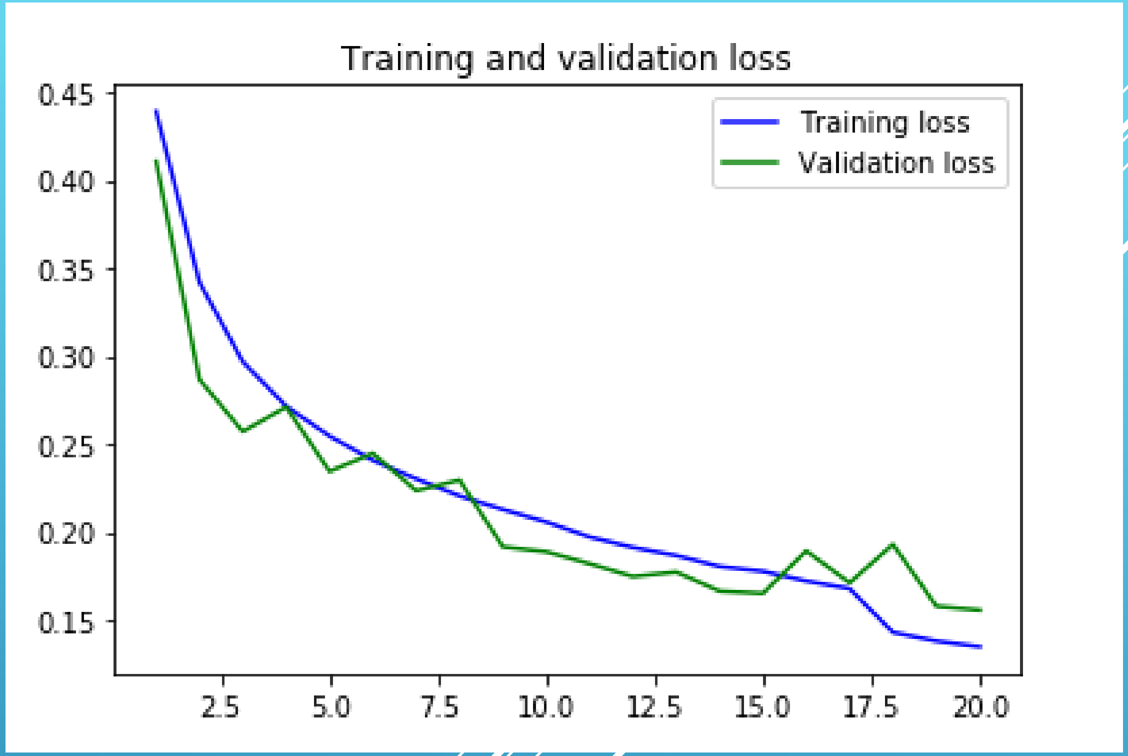
-25				...
				...
				...
				...
...	...	...	...	...

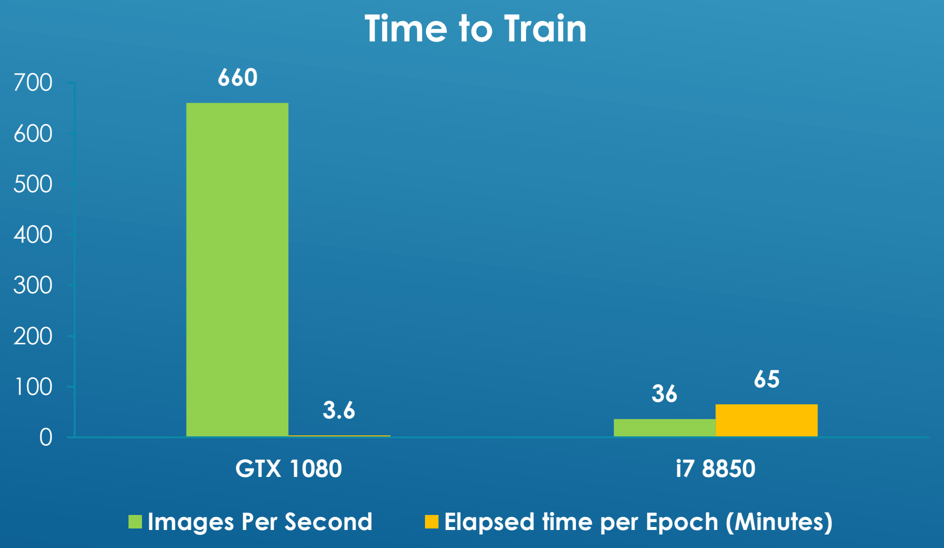
Source http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html



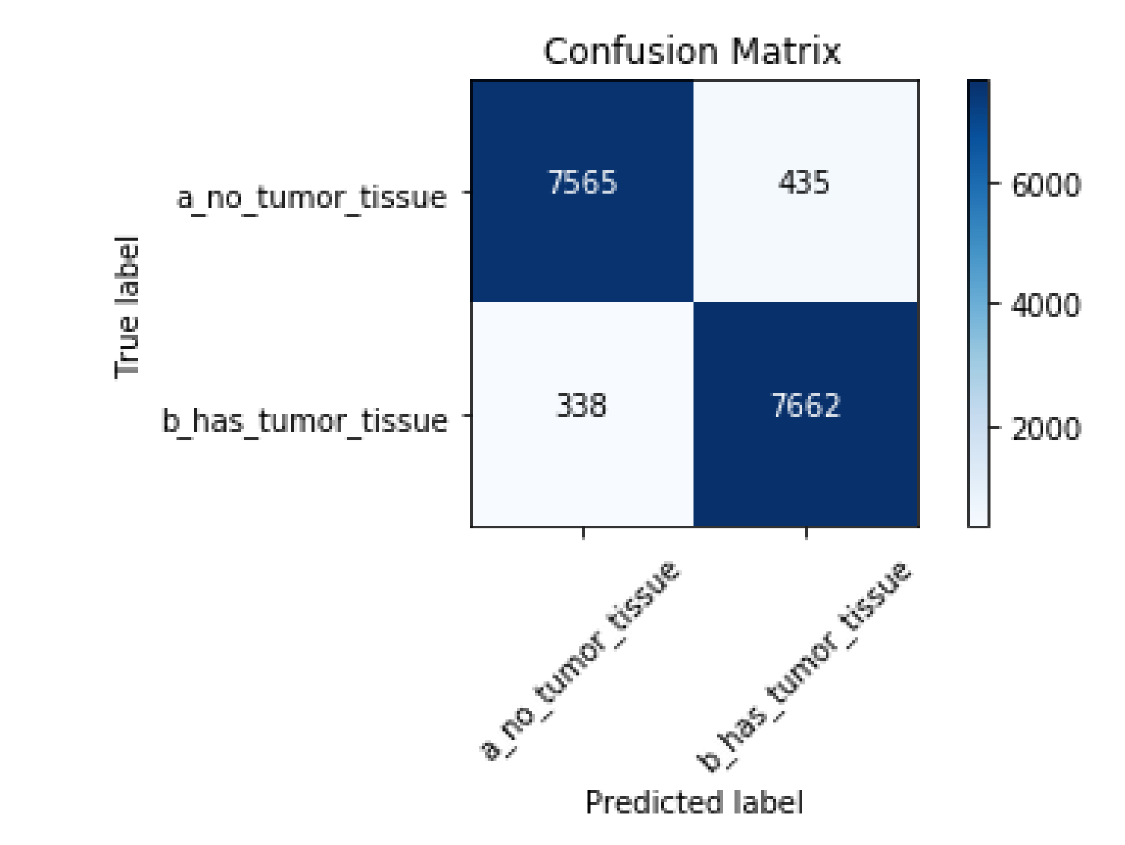
```
kernel_size = (3,3) #Height and Width of the convolution window.
pool_size= (2,2) #Size of the max pooling windows
first_filters = 32
second_filters = 64
third_filters = 128

dropout_conv = 0.3
dropout_dense = 0.3


model = Sequential()
model.add(Conv2D(first_filters, kernel_size, activation = 'relu', input_shape = (96, 96, 3)))
model.add(Conv2D(first_filters, kernel_size, activation = 'relu'))
model.add(MaxPool2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(MaxPool2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(MaxPool2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(dropout_dense))
model.add(Dense(1, activation = "sigmoid"))

model.summary()
```

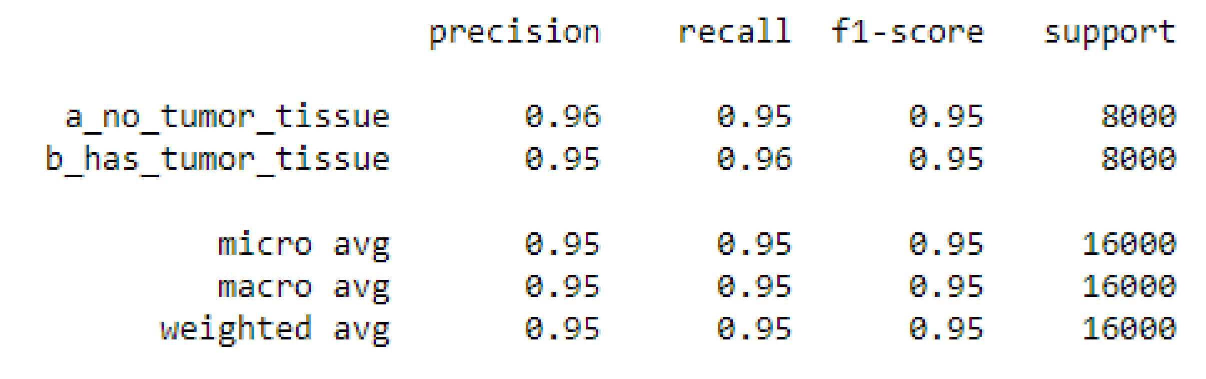

Training and validation accuracy
Training acc
Validation acc
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation loss
Training loss
Validation loss
0.45
0.40
0.35
0.30
0.25
0.20
0.15
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Time to Train
700
600
500
400
300
200
100
0
660
3.6
36
65
GTX 1080
i7 8850
Images Per Second
Elapsed time per Epoch (Minutes)

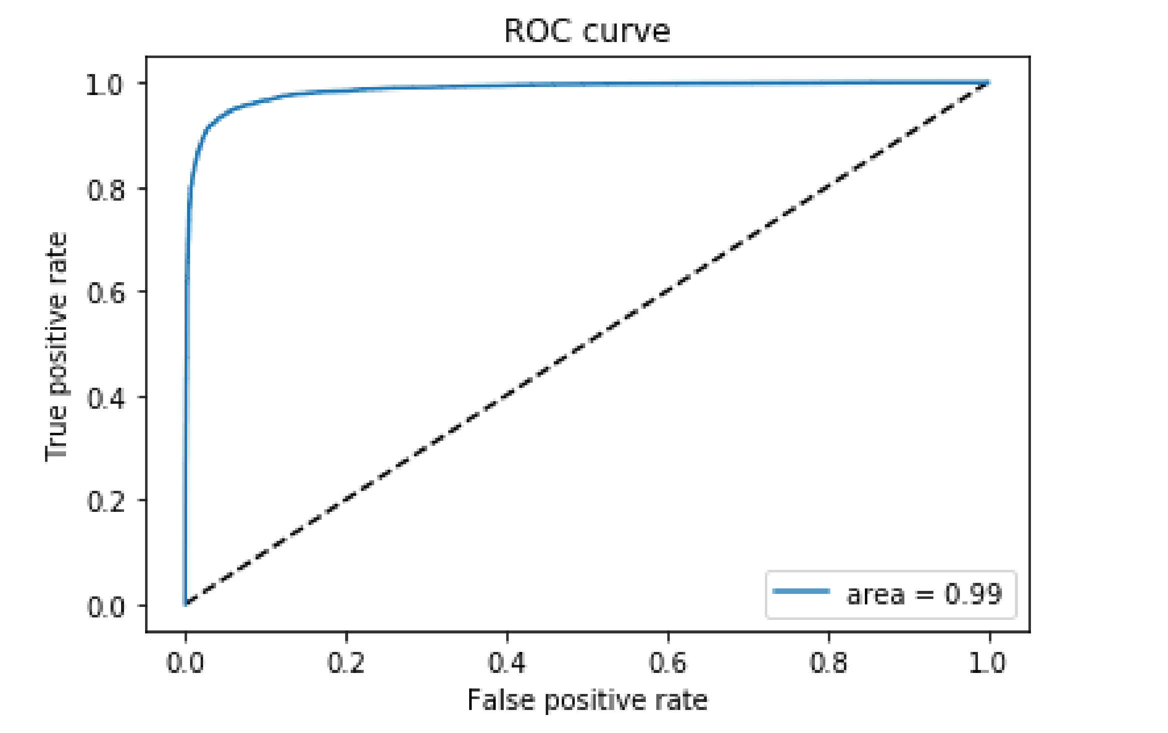
Model Validation





```
                    precision    recall  f1-score   support

 a_no_tumor_tissue       0.96      0.95      0.95      8000
b_has_tumor_tissue       0.95      0.96      0.95      8000

         micro avg       0.95      0.95      0.95     16000
         macro avg       0.95      0.95      0.95     16000
      weighted avg       0.95      0.95      0.95     16000
```

AUC = 0.989



REFERENCES

- **Stanford CNN Overview -** http://cs231n.github.io/convolutional-networks/

- **Public Kernel Referenced -** https://www.kaggle.com/vbookshelf/cnn-how-to-use-160-000-images-without-crashing

- **Keras Documentation -** https://keras.io/