CS 452 - Deep Learning for Perception

Project Member

Muhammad Nabeel Akhtar Roll # 16F-8253	Khizar Sultan Roll # 16F-8153	Abaidullah 16F-8132

Final Project

Report

CNN Classification

1. Cat & Dog Dataset
2. Sleep Dataset

Submitted to	Dr. Shahnawaz Qureshi

Fast School of Computing

CAT & DOG Project

- ## Data

  Our data includes images. Data includes pictures of cat and dog in which we have to identify that which picture includes cat or dog. It has some trained data and has some test data. The images include pictures of cats and dogs of different color and sizes. We take our data from the website named Kaggle. There are many datasets available on Kaggle. The Kaggle has many data sets of cat and dog, we choose this data set because it includes both training and test data sets.

- ## Pre-Processing

  Yes, we do use pre-processing in our data. We separate the images in different folders like there is training images, Validation images and test images we create different folders for images to classify them. There are 25,000 total images in our dataset in which training data includes 20,000 images and remaining images are in validation dataset. This helps us to classify our data.

- ## Motivation

  We make our model with respect to accuracy. Frequently trained model performs well that's why we train our model as much as we can. There are many layers in our model like convolutional, Relu, Pooling, Flatten, Sigmoid, Dense Layer and many other fully connected layers. We use all these layers according to the data like if there are 2 outputs then sigmoid function is useful and more than 2 outputs then SoftMax function is useful.

- ## Results

  We first train our model which gives 89% accuracy. Then we improve our model as we can and then train it which brings the good accuracy. Now the accuracy is 96% on training dataset.

Model

With 89% accuracy

```
classifier = Sequential()
classifier.add(Conv2D(16,(3,3),input_shape=(IMG_WIDTH,IMG_HEIGHT,3),activation = 'relu'))

classifier.add(BatchNormalization())
classifier.add(MaxPooling2D(pool_size=(2,2),strides=2))

classifier.add(Conv2D(32,(3,3),padding='same',activation='relu'))
classifier.add(BatchNormalization())
classifier.add(MaxPooling2D(pool_size=(2,2),strides=2))

classifier.add(Conv2D(64,(3,3),padding='same',activation='relu'))
classifier.add(BatchNormalization())
classifier.add(MaxPooling2D(pool_size=(2,2),strides=2))


classifier.add(Flatten())

classifier.add(Dense(units=512,activation='relu'))
classifier.add(BatchNormalization())

classifier.add(Dense(units=1,activation='sigmoid'))
```

Epoch 15/180
799/800 [==============================>.] - ETA: 0s - loss: 0.1334 - acc: 0.9488Epoch 1/180
800/800 [==============================] - 87s 108ms/step - loss: 0.1334 - acc: 0.9488 - val_loss: 0.3244 - val_acc: 0.8859

We get 89% accuracy on our model but then we improve and change our model a lit bit like applying SoftMax function instead of sigmoid function.

With 96% accuracy

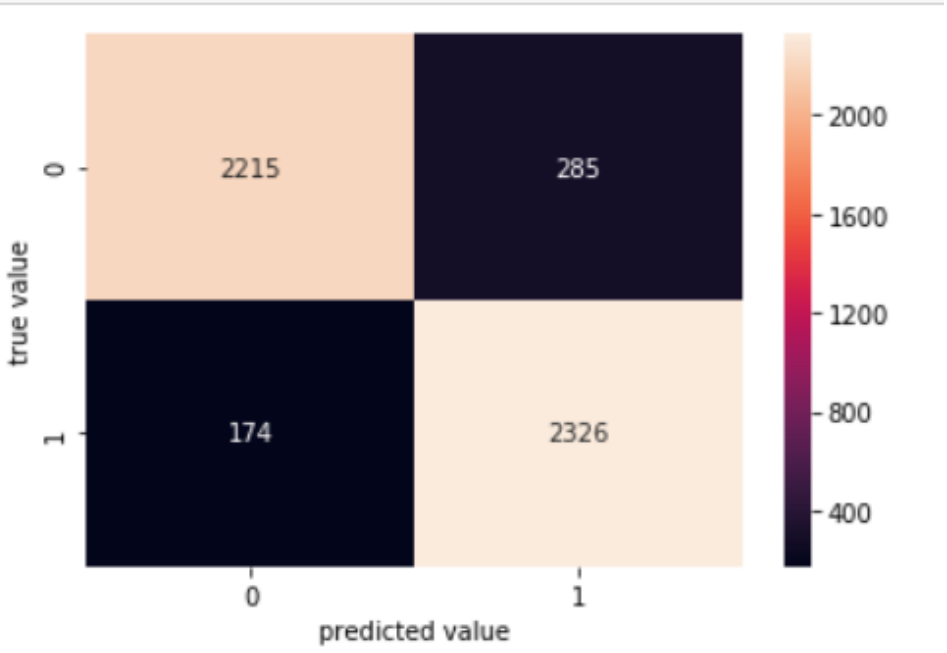
```
classifier = Sequential()

classifier.add(Conv2D(32,(3,3),input_shape=(IMG_WIDTH,IMG_HEIGHT,3),activation = 'relu'))
classifier.add(BatchNormalization()),
classifier.add(MaxPooling2D(pool_size=(2,2),strides=2)) #if stride not given it equal to pool filter size

classifier.add(Conv2D(64,(3,3),activation = 'relu'))
classifier.add(BatchNormalization()),
classifier.add(MaxPooling2D(pool_size=(2,2),strides=2))

classifier.add(Conv2D(128,(3,3),activation = 'relu'))
classifier.add(BatchNormalization()),
classifier.add(MaxPooling2D(pool_size=(2,2),strides=2))

classifier.add(Flatten())
classifier.add(Dropout(0.25))

classifier.add(Dense(units=512,activation='relu'))
classifier.add(BatchNormalization()),
classifier.add(Dropout(0.5))

classifier.add(Dense(units=1,activation='sigmoid'))
```

```
Epoch 13/180
799/800 [============================>.] - ETA: 0s - loss: 0.0957 - acc: 0.9625Epoch 1/180
800/800 [==============================] - 360s 450ms/step - loss: 0.0958 - acc: 0.9625 - val_loss: 0.3602 - val_acc: 0.85
16
Epoch 14/180
799/800 [============================>.] - ETA: 0s - loss: 0.0954 - acc: 0.9616Epoch 1/180
800/800 [==============================] - 357s 446ms/step - loss: 0.0953 - acc: 0.9617 - val_loss: 0.2680 - val_acc: 0.89
22
Epoch 15/180
799/800 [============================>.] - ETA: 0s - loss: 0.0913 - acc: 0.9644Epoch 1/180
 20/800 [..............................] - ETA: 2:08 - loss: 0.2714 - acc: 0.8969
Epoch 00015: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
800/800 [==============================] - 360s 450ms/step - loss: 0.0912 - acc: 0.9645 - val_loss: 0.2714 - val_acc: 0.89
```

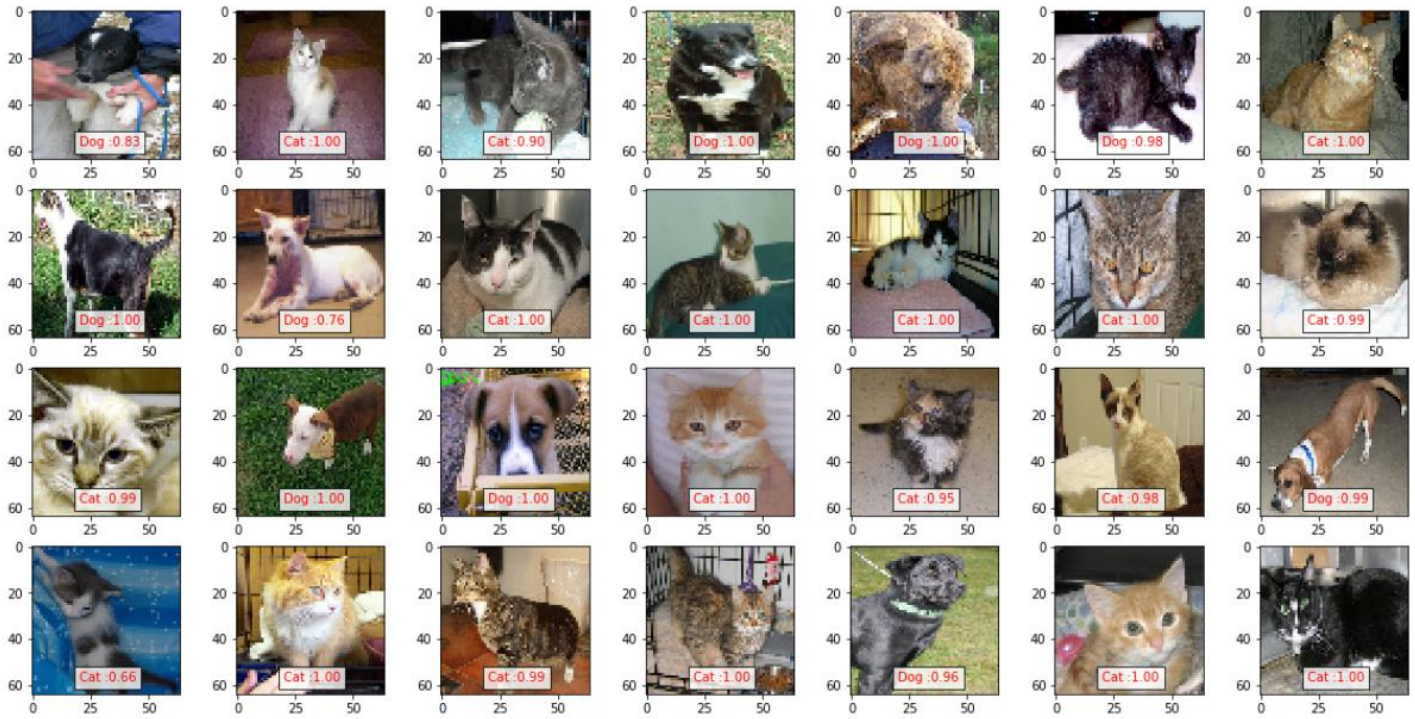
Frequency Table

	filename	Actual	predict
0	cat\cat.0.jpg	0	0
1	cat\cat.1.jpg	0	0
2	cat\cat.10.jpg	0	0
3	cat\cat.100.jpg	0	0
4	cat\cat.1000.jpg	0	0
5	cat\cat.10000.jpg	0	0
6	cat\cat.10001.jpg	0	0
7	cat\cat.10002.jpg	0	0
8	cat\cat.10003.jpg	0	0
9	cat\cat.10004.jpg	0	0

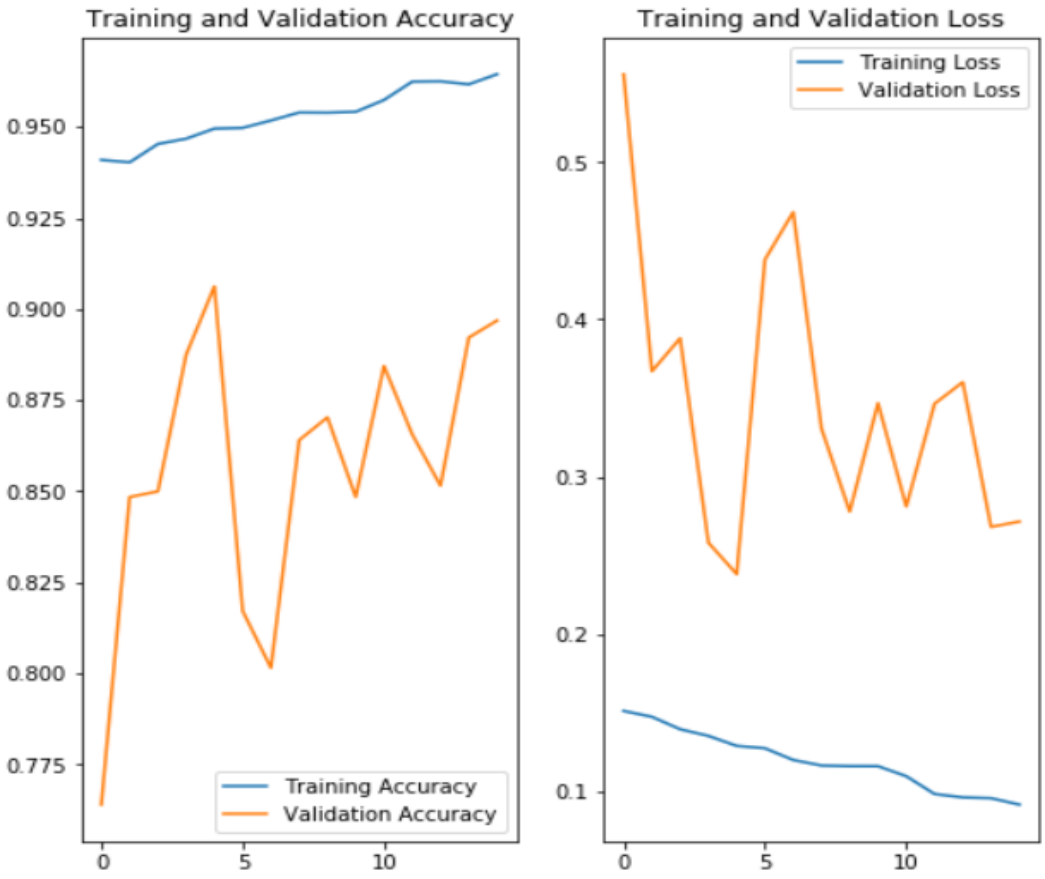
Confusion Matrix



Result (Filters)



Graph



- **Model can be used**
  - Natural Language Processing (NLP)
  - Digital Image Processing (DIP)
  - Business problems
  - Automation
  - Time Series Analysis
  - Computer Vision
  - Speech Recognition
  - Audio Recognition
  - Social Network Filtering
  - Machine Translation

SLEEP Project

- ## Data

  Our data includes images. Data includes pictures of sleeping in which we have to identify that in which picture there was sleeping, awake or rem. It has some trained data and has some test data. The images include pictures of different color and sizes. We take our data from the website named Kaggle. There are many datasets available on Kaggle. The Kaggle has many data sets of sleep, we choose this data set because it includes both training and test data sets.

- ## Pre-Processing

  Pre-Processing is applied on the dataset. We separate the images in different folders like there is training images, validation images and test images. We create different folders for images to classify them**.** There are 25,000 total images in our dataset in which training data includes 20,000 images and remaining images are in validation dataset. This helps us to classify our data.

- ## Motivation

  We make our model to attain maximum accuracy. Frequently trained model performs well that's why we train our model as much as we can. There are many layers in our model like Convolutional Layer, Relu, Pooling, flatten, Dense Layer and many other fully connected layers. We use all layers according to the data like if there are two outputs then sigmoid function is useful and more than two outputs then SoftMax function is useful.

- ## Results

  ### Model

```
model  = None
model = Sequential([
       Conv2D(filters=32,kernel_size=(3,3),padding='same',activation='relu',input_shape = (IMG_WIDTH,IMG_HEIGHT,3)),
       MaxPooling2D((2,2)),

       Conv2D(filters=64,kernel_size=(3,3),padding='same',activation='relu'),
       MaxPooling2D((2,2)),

       Conv2D(filters=128,kernel_size=(3,3),padding='same',activation='relu'),
       MaxPooling2D((2,2)),

       Conv2D(filters=128,kernel_size=(3,3),padding='same',activation='relu'),
       MaxPooling2D((2,2)),
       Flatten(),

       Dense(512,activation='relu'),
       Dense(3,activation='softmax')
])
```

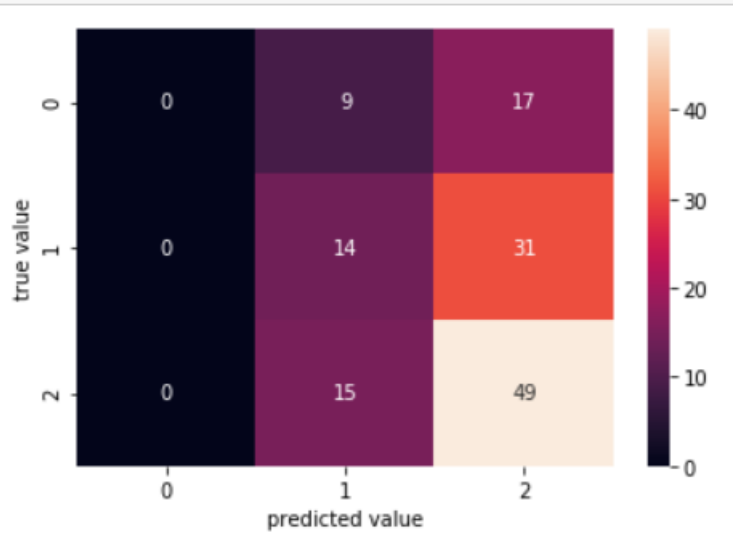
```
15/16 [===========================>..] - ETA: 0s - loss: 0.4633 - acc: 0.8083Epoch 1/200
16/16 [==============================] - 7s 435ms/step - loss: 0.4584 - acc: 0.8086 - val_loss: 0.7158 - val_acc: 0.6875
Epoch 195/200
15/16 [===========================>..] - ETA: 0s - loss: 0.4759 - acc: 0.7584Epoch 1/200
16/16 [==============================] - 7s 420ms/step - loss: 0.4722 - acc: 0.7638 - val_loss: 0.6954 - val_acc: 0.6797
Epoch 196/200
15/16 [===========================>..] - ETA: 0s - loss: 0.4586 - acc: 0.7857Epoch 1/200
16/16 [==============================] - 7s 423ms/step - loss: 0.4517 - acc: 0.7933 - val_loss: 0.6167 - val_acc: 0.7109
Epoch 197/200
15/16 [===========================>..] - ETA: 0s - loss: 0.4583 - acc: 0.7836Epoch 1/200
16/16 [==============================] - 7s 436ms/step - loss: 0.4537 - acc: 0.7835 - val_loss: 0.6155 - val_acc: 0.6641
Epoch 198/200
15/16 [===========================>..] - ETA: 0s - loss: 0.4393 - acc: 0.7983Epoch 1/200
16/16 [==============================] - 7s 430ms/step - loss: 0.4290 - acc: 0.8051 - val_loss: 0.6236 - val_acc: 0.7109
Epoch 199/200
15/16 [===========================>..] - ETA: 0s - loss: 0.4571 - acc: 0.7878Epoch 1/200
16/16 [==============================] - 7s 428ms/step - loss: 0.4651 - acc: 0.7815 - val_loss: 0.7195 - val_acc: 0.7109
Epoch 200/200
15/16 [===========================>..] - ETA: 0s - loss: 0.4419 - acc: 0.8088Epoch 1/200
16/16 [==============================] - 7s 421ms/step - loss: 0.4339 - acc: 0.8150 - val_loss: 0.6587 - val_acc: 0.7500
```

81% accuracy is achieved with our model on sleep dataset.

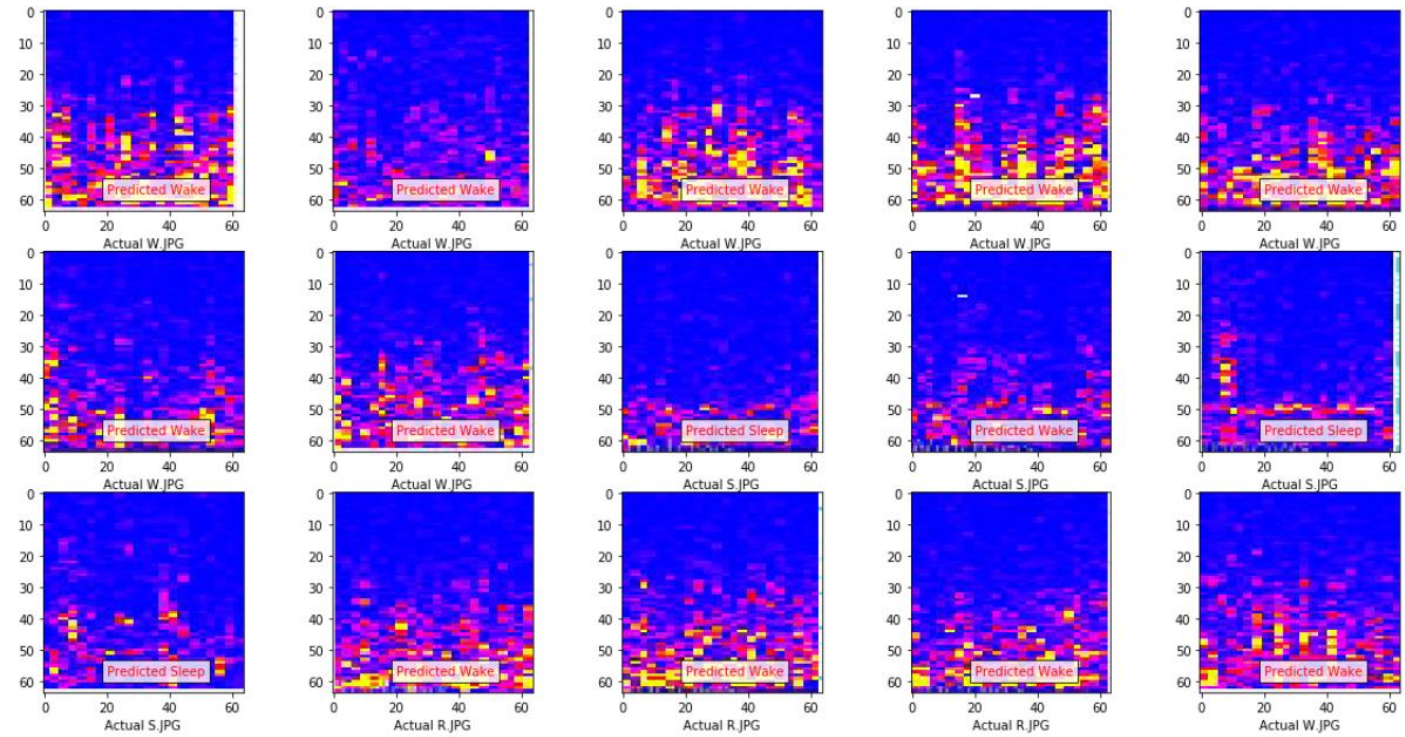
Frequency Table

	Filname	Actual	Prediction
125	wake\img_14_W.JPG	2	2
126	wake\img_150_W.JPG	2	2
127	wake\img_151_W.JPG	2	2
128	wake\img_152_W.JPG	2	2
129	wake\img_153_W.JPG	2	2
130	wake\img_154_W.JPG	2	2
131	wake\img_155_W.JPG	2	2
132	wake\img_156_W.JPG	2	2
133	wake\img_157_W.JPG	2	2
134	wake\img_158_W.JPG	2	2

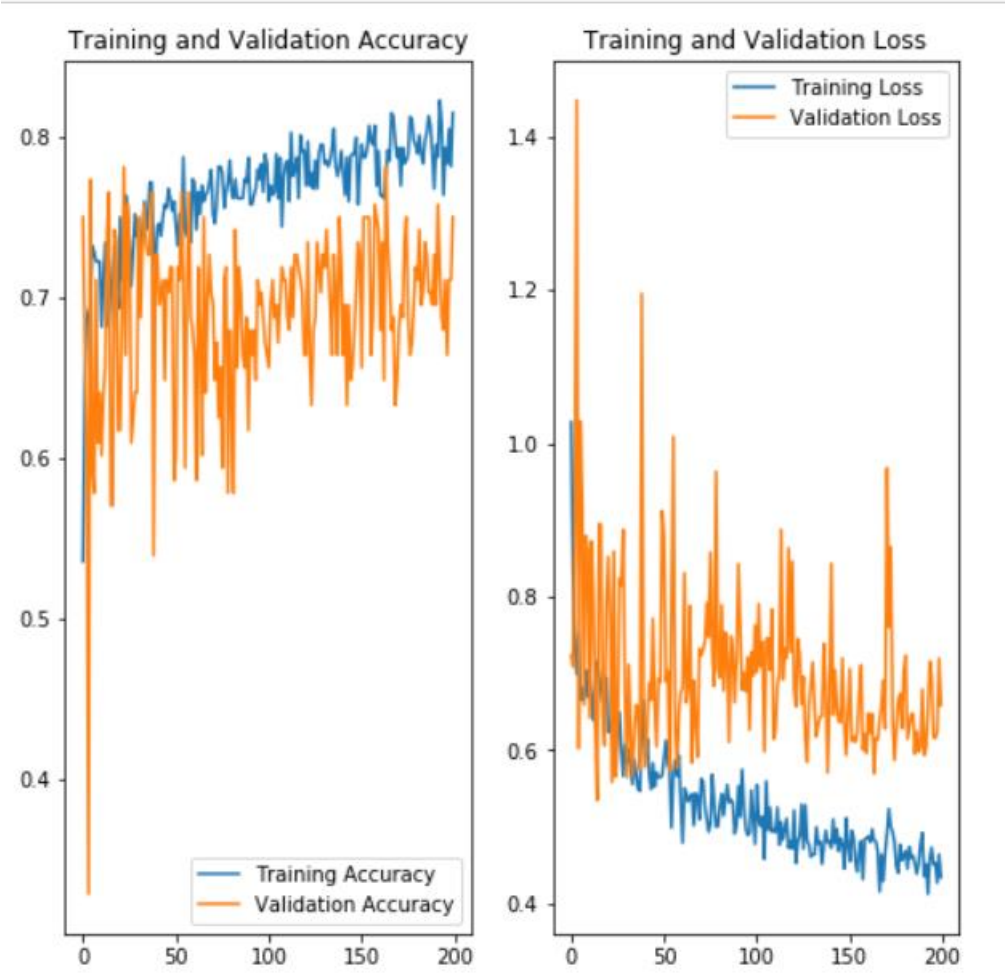
Confusion Matrix



Filters



Graph



- **Model can be used**
  - Natural Language Processing (NLP)
  - Digital Image Processing (DIP)
  - Business problems
  - Automation
  - Time Series Analysis
  - Computer Vision
  - Speech Recognition
  - Audio Recognition
  - Social Network Filtering
  - Machine Translation