Final Project : Face Recognition

Name of the Project : Face Recognition

Goal: Face Recognition using Classification, Main goal of the project is, to classify photos as "George Bush" and "Not George Bush" and to classify photos as "Serena Williams" and "Not Serena Williams".

Dataset: There are 13,233 photos in total which are 64x64 and gray-scale. We used following three datasets for the project. **X.csv:** The file consists of all photos where each row corresponds to a photo and columns correspond to pixels. **y_bush_vs_others.csv:** The file consists of labels for task **y_williams_vs_others.csv:** The file consists of labels for task 2.

There are four phases of the project. All phases are described below.

PHASE 1 :

In phase 1,The experiment was on the following classifiers on both Bush and Williams datasets; they are given below.

1. KNeighborsClassifier, with n_neighbors = 1, 3, and 5.
2. SVC with various C values, kernels, and various parameters for those kernels (such as gamma for rbf, degree for poly, etc).

The tasks to be performed in this phase are as follows and these are done twice, once for Bush and once for Williams:

=>. Loading the datasets.

```
X = pd.read_csv("data/X.csv", sep=' ', header=None, dtype=float)
X = X.values
```

```
y = pd.read_csv("data/y_bush_vs_others.csv", header=None)
y_bush = y.values.ravel()
```

```
y = pd.read_csv("data/y_williams_vs_others.csv", header=None)
y_williams = y.values.ravel()
```

=>. Pick one classifier, KNeighbourClassifier and SVC; and perform various combination as follows.

```
knn = KNeighborsClassifier(n_neighbors=1)
```

And same for n_neighbours = 2 and 5.

The SVC can be set with various parameters as follows; examples for that is given in below figures.

```
svc1 = SVC( C = 0.1, kernel = 'linear')
```

```
svc1 = SVC( C = 1000, kernel = 'rbf', gamma = 0.0001)
```

```
svc1 = SVC( C = 10, kernel = 'poly', degree = 6, coef0=1)
```

=>. We then performed three-fold cross validation using the following code:

stratified_cv_results = cross_validate (clf, X, y, cv=StratifiedKFold (n_splits = 3, shuffle=True, random_state = 7392), scoring=('precision', 'recall', 'f1'), return_train_score=False).

clf is the classifier and can be set to knn or svc.

=>. After performing the cross validation, we check for the mean F1 results for n_neighbors=1, n_neighbors=3, n_neighbors=5 and SVC and choose the best SVC result.

Bush

=>. For KNN for

KNN Result

Classifier	KNeighborsClassifier			
Parameters	**n_neighbors=1**			
	result1	**result2**	**result3**	**mean result**
fit_time	5.40908599	5.03097796	4.99072194	5.1435952981
score_time	841.52749586	808.44048500	833.14367104	827.7038839658
test_f1	0.18289086	0.10795455	0.13031161	0.1403856719
test_precision	0.19135802	0.10857143	0.1299435	0.1432909854
test_recall	0.17514124	0.10734463	0.13068182	0.1377225646
Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=3**			
	result1	**result2**	**result3**	**mean result**
fit_time	6.51542020	5.52433205	5.82914495	5.9562990665
score_time	808.20556283	870.69530129	840.76913714	839.8900004228
test_f1	0.06896552	0.06730769	0.03636364	0.05754562
test_precision	0.26923077	0.22580645	0.09090909	0.19531544
test_recall	0.03954802	0.03954802	0.02272727	0.03394111
Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=5**			
	result1	**result2**	**result3**	**mean result**
fit_time	5.80563498	4.81308889	4.67249894	5.0970742702
score_time	765.92551994	802.09543490	755.14202189	774.3876589139
test_f1	0.01104972	0.02150538	0.02116402	0.0179063738
test_precision	0.25000000	0.22222222	0.15384615	0.2086894587
test_recall	0.00564972	0.01129944	0.01136364	0.0094375963

=>. The best SVC results in this phase are:

Best (in terms of mean F1) SVC result I got				
Parameters	C = 100, kernel = 'poly', degree = 3, coef0 = 1			
	result1	**result2**	**result3**	**mean result**
fit_time	59.23813605	56.15270615	55.172333	56.8543917333
score_time	72.70101810	66.58980393	64.87317395	68.0546653267
test_f1	0.67128028	0.61073826	0.62116041	**0.6343929833**
test_precision	0.86607143	0.75206612	0.77777778	0.7986384433
test_recall	0.54802260	0.51412429	0.51704545	0.5263974467

=>. Overall SVC is the best result for bush.

Best (in terms of mean F1) SVC result I got				
Parameters	C = 100, kernel = 'poly', degree = 3, coef0 = 1			
	result1	**result2**	**result3**	**mean result**
fit_time	59.23813605	56.15270615	55.172333	56.8543917333
score_time	72.70101810	66.58980393	64.87317395	68.0546653267
test_f1	0.67128028	0.61073826	0.62116041	**0.6343929833**
test_precision	0.86607143	0.75206612	0.77777778	0.7986384433
test_recall	0.54802260	0.51412429	0.51704545	0.5263974467

Williams

KNN Result

		Classifier	KNeighborsClassifier	
Parameters	**n_neighbors=1**			
	result1	**result2**	**result3**	**mean result**
fit_time	5.34192014	4.83722806	5.31970692	5.1662850380
score_time	771.88136172	770.05597210	789.46003294	777.1324555874
test_f1	0.16666667	0.10526316	0.33333333	0.2017543860
test_precision	0.33333333	0.50000000	0.57142857	0.4682539683
test_recall	0.11111111	0.05882353	0.23529412	0.1350762527
		Classifier	**KNeighborsClassifier**	
Parameters	**n_neighbors=3**			
	result1	**result2**	**result3**	**mean result**
fit_time	5.50879574	4.99511290	4.71376014	5.0725562572
score_time	767.57621312	763.16864514	765.78671575	765.5105246703
test_f1	0.00000000	0.00000000	0.00000000	0.00000000
test_precision	0.00000000	0.00000000	0.00000000	0.00000000
test_recall	0.00000000	0.00000000	0.00000000	0.00000000
		Classifier	**KNeighborsClassifier**	
Parameters	**n_neighbors=5**			
	result1	**result2**	**result3**	**mean result**
fit_time	5.15044880	4.83519816	5.27125812	5.0856350263
score_time	774.67728996	803.81123400	784.83699679	787.7751735846
test_f1	0.00000000	0.00000000	0.00000000	0.00000000
test_precision	0.00000000	0.00000000	0.00000000	0.00000000
test_recall	0.00000000	0.00000000	0.00000000	0.00000000

SVC Result.

Best (in terms of mean F1) SVC result I got				
Parameters	C = 100, kernel = 'poly', degree = 3, coef0 = 2			
	result1	**result2**	**result3**	**mean result**
fit_time	13.10019803	13.05727196	12.14418697	12.7672189871
score_time	13.47802186	12.57174110	12.45664001	12.8354676565
test_f1	0.44444444	0.61538462	0.53846154	**0.5327635328**
test_precision	0.66666667	0.88888889	0.77777778	0.7777777778
test_recall	0.33333333	0.47058824	0.41176471	0.4052287582

=>. Overall SVC is the best result for williams.

Best (in terms of mean F1) SVC result I got				
Parameters	C = 100, kernel = 'poly', degree = 3, coef0 = 2			
	result1	**result2**	**result3**	**mean result**
fit_time	13.10019803	13.05727196	12.14418697	12.7672189871
score_time	13.47802186	12.57174110	12.45664001	12.8354676565
test_f1	0.44444444	0.61538462	0.53846154	**0.5327635328**
test_precision	0.66666667	0.88888889	0.77777778	0.7777777778
test_recall	0.33333333	0.47058824	0.41176471	0.4052287582

PHASE 2 :

The goal of the phase is to implement principal component analysis (PCA) and to experiment with several PCA parameters and classifier parameters to achieve the best mean F1 results.
The tasks to be performed in this phase are as follows and these are done twice, once for Bush and once for Williams:
=>. We loaded the datasets as shown in the phase : 1.
=>. After loading, fit and transform the full dataset using PCA as follows.

```
pca = PCA(n_components=60)
pca.fit_transform(X)
```

=>. Again same as phase 1 pick a classifier i.e., experiment with KNeighborsClassifier and SVC.
=>. We perform three-fold cross validation using the following code:

stratified_cv_results = cross_validate (clf, X, y, cv=StratifiedKFold (n_splits = 3, shuffle=True, random_state = 7392), scoring= ('precision', 'recall', 'f1'), return_train_score=False).
In this case, X is the PCA transformed data, with good parameter settings and clf can be set to knn or svc.

=>. After completion of above step - record and check for best mean F1 results for KNeighborsClassifier and SVC.

=>. For this phase the best results are shown in below figures.

1. Bush Dataset

bush dataset

Phase 1 results		
Classifier	**Parameters**	**Mean F1**
KNeighborsClassifier	n_neighbors=1	0.14038767
KNeighborsClassifier	n_neighbors=3	0.05754562
KNeighborsClassifier	n_neighbors=5	0.01790637
SVC (Best result)	C = 100, kernel = 'poly', degree = 3, coef0 = 1	0.6343929833
Phase 2 best results		
Best result for KNeighborsClassifier		
PCA parameters	n_components = 200	
KNeighborsClassifier parameters	n_neighbors = 1	
Mean F1	**0.1424657128**	
Best result for SVC		
PCA parameters	n_components = 3000	
SVC parameters	C = 100, kernel = 'poly', degree = 3, coef0 = 1	
Mean F1	**0.6339804292**	

2. Williams Dataset

williams dataset

Phase 1 results		
Classifier	**Parameters**	**Mean F1**
KNeighborsClassifier	n_neighbors=1	0.2017543860
KNeighborsClassifier	n_neighbors=3	0.0000000000
KNeighborsClassifier	n_neighbors=5	0.0000000000
SVC (Best result)	C = 100, kernel = 'poly', degree = 3, coef0 = 2	0.5327635328
Phase 2 best results		
Best result for KNeighborsClassifier		
PCA parameters	n_components = 200	
KNeighborsClassifier parameters	n_neighbors = 1	
Mean F1	**0.2231545046**	
Best result for SVC		
PCA parameters	n_components = 1000	
SVC parameters	C = 5, kernel = 'poly', degree = 9, coef0 = 100	
Mean F1	**0.5415491835**	

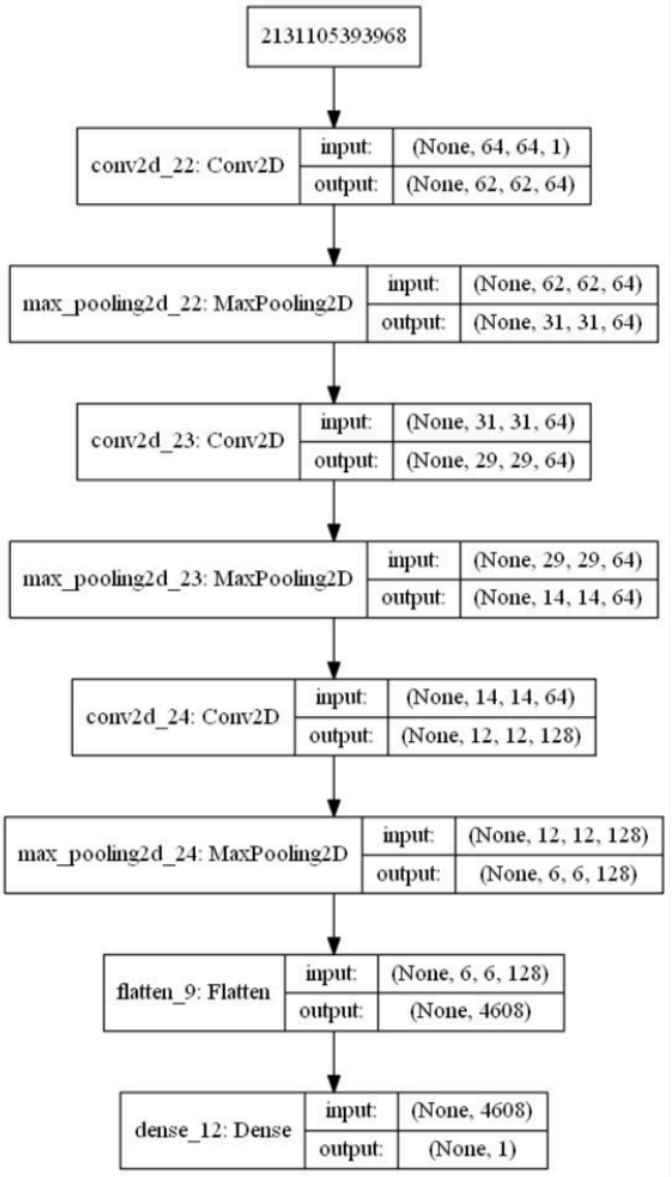
PHASE 3 :

Goal : The goal of this phase is to develop and learn a deep learning model, design deep learning model for image classification by using Convolutional Neural Networks (CNN's) and max-pooling. We first perform CNNs followed by max-pooling, The output layer is a single node with sigmoid activation and one dense layer before that.

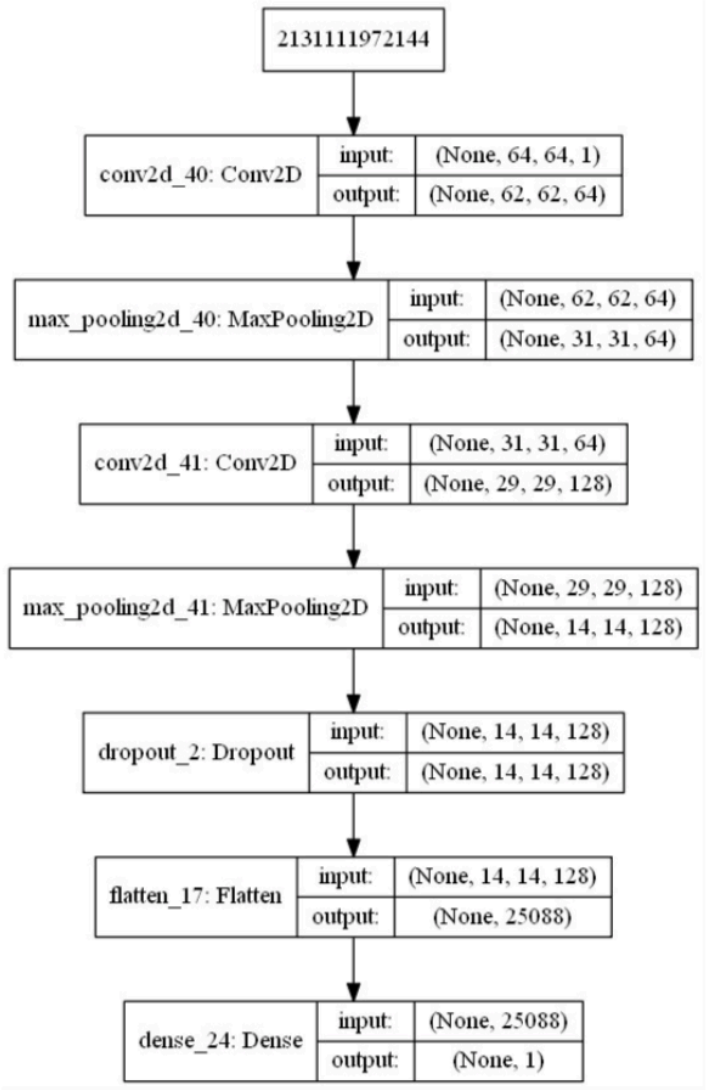
Architecture of Bush:
The images are of the size 64 x 64 and is reshaped as 64 x 64 x 1 to provide as input to the network. Here for the project, use three convolutional layers as described below: **1.** The first layer will have 64 – 5 x 5 filters; **2.** The second layer will have 128 – 5 x 5 filters and; **3.** The third layer will have 256 – 5 x 5 filters.
The above three layers have RELU function, the dense layer have linear activation and the output layer has the sigmoid function; Following each convolutional layer, we have three max-pooling layers each of size 2 x 2; The architecture used for Bush is in below figures.

Architecture of Williams:
The images are of the size 64 x 64 and is reshaped as 64 x 64 x 1 to provide as input to the network. We use three convolutional layers as described below: **1.** The first layer will have 64 – 5 x 5 filters; **2.** The second layer will have 128 – 5 x 5 filters and; **3.** The third layer will have 256 – 5 x 5 filters.
The above three layers and the dense functions have linear activation and the output layer has the sigmoid function; Following each convolutional layer, we have three max-pooling layers each of size 2 x 2; The architecture used for Williams is in below figures.

Model for Bush



Model for Williams



The tasks to be performed in this phase are as follows and these are done twice, once for Bush and once for Williams :
=>. How to load the datasets (Bush and Williams) is shown in above two phases.
=>. After loading we perform a train-test split evaluation as follows:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X, y_bush, stratify=y_bush, test_size=1./3, random_state=6196, shuffle=True)
```

The above is for Bush. We also perform for Williams in a similar way.

=>. We then reshape the images as follows:

```
X_train = X_train.reshape(-1, 64,64, 1)
X_test = X_test.reshape(-1, 64,64, 1)
```

=>. The model implemented is as follows:

For Bush dataset

```
from keras import layers
from keras import models
from keras.layers.core import Activation
model = models.Sequential()
model.add(layers.Conv2D(64,(3,3),activation='relu',input_shape=(64,64,1)))
model.add(layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
model.add(layers.Conv2D(64, (3,3),activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
model.add(layers.Conv2D(128, (3,3),activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

For Williams dataset

```
from keras import layers
from keras import models
from keras.layers.core import Activation
model = models.Sequential()
model.add(layers.Conv2D(64,(3,3),activation='relu',input_shape=(64,64,1)))
model.add(layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
model.add(layers.Conv2D(128, (3,3),activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
#model.add(layers.Conv2D(64, (3,3),activation='relu'))
#model.add(layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
model.add(layers.Dropout(0.5))
model.add(layers.Flatten())

#model.add(layers.Dense(64, activation='tanh'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

=>. The below is the summary of the model:

Bush:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_22 (Conv2D)           (None, 62, 62, 64)        640
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 29, 29, 64)        36928
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 12, 12, 128)       73856
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 6, 6, 128)         0
_________________________________________________________________
flatten_9 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 4609
=================================================================
Total params: 116,033
Trainable params: 116,033
Non-trainable params: 0
_________________________________________________________________
```

Williams:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_40 (Conv2D)           (None, 62, 62, 64)        640
_________________________________________________________________
max_pooling2d_40 (MaxPooling (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 29, 29, 128)       73856
_________________________________________________________________
max_pooling2d_41 (MaxPooling (None, 14, 14, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 128)       0
_________________________________________________________________
flatten_17 (Flatten)         (None, 25088)             0
_________________________________________________________________
dense_24 (Dense)             (None, 1)                 25089
=================================================================
Total params: 99,585
Trainable params: 99,585
Non-trainable params: 0
_________________________________________________________________
```

=>. Then, we fit, evaluate and test the model as follows.

```
model_train = model.fit(X_train, y_train, batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(X_test, y_test))
```

```
test_eval = model.evaluate(X_test, y_test, verbose=1)
```

```
predicted_classes = model.predict(X_test)
```

=>. We finally get the F1 results as follows:

1. For Bush:

```
p = [f1_bush1, f1_bush]
print (p)
```

```
[0.9843971631205675, 0.8275862068965518]
```

2. For Williams:

```
p = [f1_williams1, f1_williams]
print (p)
```

```
[0.923076923076923, 0.5217391304347826]
```

PHASE 4 :

This is the final phase of the project and its goal is to perform a simple form of transfer learning. In this we need to find an image classification dataset and pre-train the model. We first start with a randomly-initialized model and train it on a different image classification dataset.

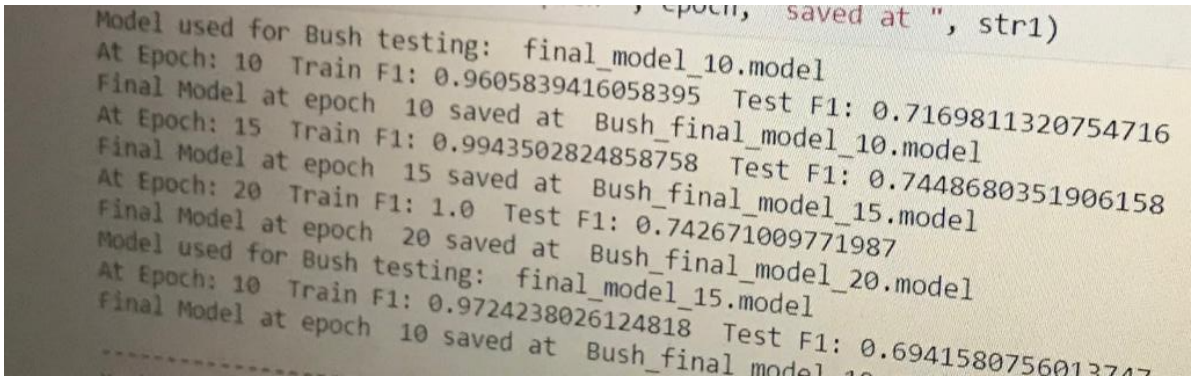
=>.We first load the dataset. The dataset I used is from the following url:

URL: https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data

Total images: The train folder contains 25,000 images of dogs and cats; The train folder contains 25,000 images of dogs and cats. Each image in this folder has the label as part of the filename. The test folder contains 12,500 images, named according to a numeric id. For each image in the test set, you should predict a probability that the image is a dog (1 = cat, 0 = dog).

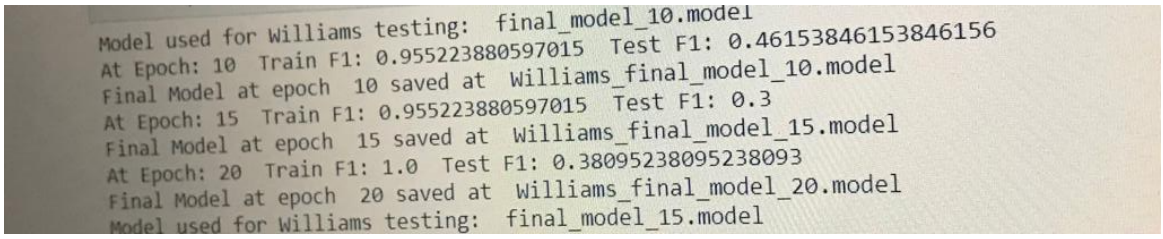
=>. Then, we pre-train our model on the processed image dataset.
=>. Perform the tasks specified in phase 3 for both Bush and Williams dataset.
=>. After that, we load our model and train it on Bush (or Williams) train split.
=>. The result is as follows:

For Bush:



For epoch 15 got **Train F1 :** 0.994350282485 and **Test F1 :** 0.7448680351906
After running above combinations.

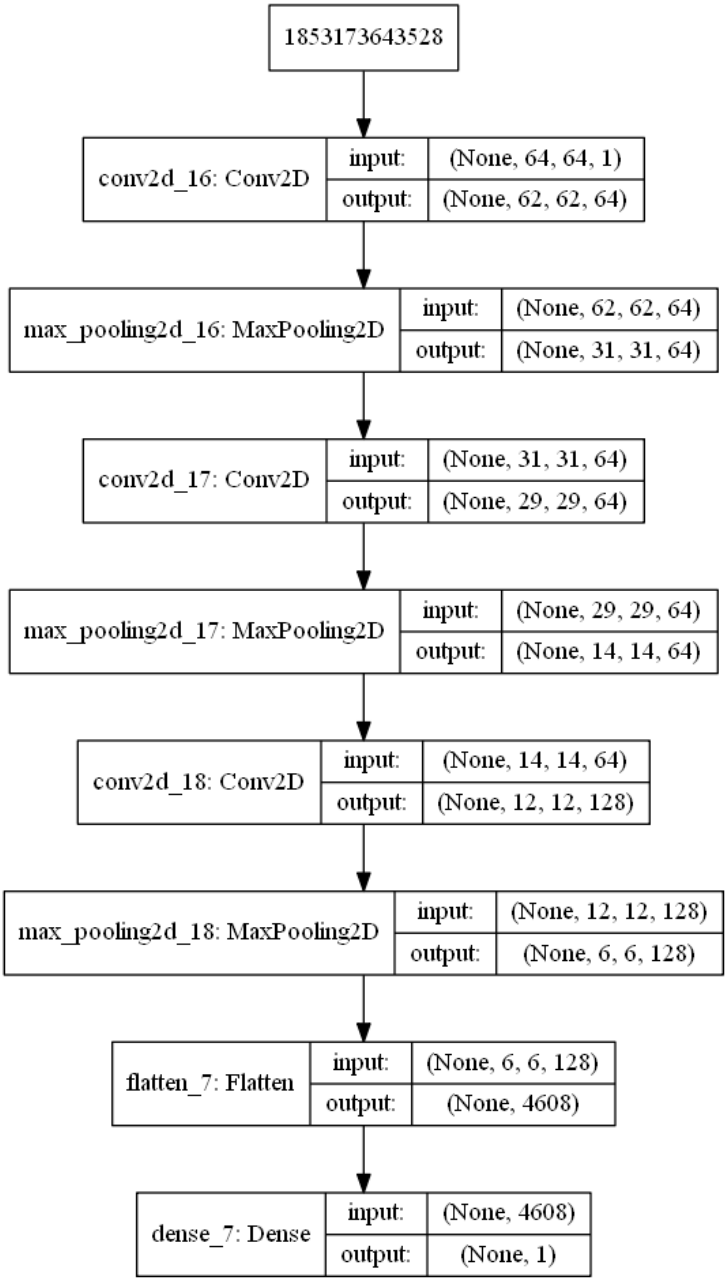
For Williams:



For epoch 15 got **Train F1** : 0.939393939393 and **Test F1 :** 0.3
After running above combinations.

Graphical representation of the model:

For Bush :


1853173643528
conv2d_16: Conv2D
input: (None, 64, 64, 1)
output: (None, 62, 62, 64)
max_pooling2d_16: MaxPooling2D
input: (None, 62, 62, 64)
output: (None, 31, 31, 64)
conv2d_17: Conv2D
input: (None, 31, 31, 64)
output: (None, 29, 29, 64)
max_pooling2d_17: MaxPooling2D
input: (None, 29, 29, 64)
output: (None, 14, 14, 64)
conv2d_18: Conv2D
input: (None, 14, 14, 64)
output: (None, 12, 12, 128)
max_pooling2d_18: MaxPooling2D
input: (None, 12, 12, 128)
output: (None, 6, 6, 128)
flatten_7: Flatten
input: (None, 6, 6, 128)
output: (None, 4608)
dense_7: Dense
input: (None, 4608)
output: (None, 1)

For Williams :