CS 584 Machine Learning

Final Project Report

Phase1:

Split the datasets for Bush and Williams into train and test and then apply classifiers on the model.
Classifier Used:

BUSH

1)KNN Neighbor

- **When neighbors =1**

	result1	result2	result3	mean result
fit_time	11.35105515	7.51504707	10.77351618	9.8798727
score_time	1383.916674	1190.157048	1169.064649	1247.71279
test_f1	0.12280702	0.15625	0.125	**0.1346856725**
test_precision	0.12727273	0.17482517	0.11979167	0.1406298563
test_recall	0.11864407	0.14124294	0.13068182	0.1301896079

- **When neighbors =3**

	result1	result2	result3	mean result
fit_time	7.07896781	7.11885309	5.90018988	6.699336926
score_time	6715.528166	974.2213357	4860.917852	4183.555784
test_f1	0.06635071	0.04950495	0.06363636	**0.05983067501**
test_precision	0.20588235	0.2	0.15909091	0.1883244207
test_recall	0.03954802	0.02824859	0.03977273	0.03585644581

- **When neighbors =5**

	result1	result2	result3	mean result
fit_time	6.08777809	5.87792873	5.81331944	5.92634209
score_time	996.5334814	967.8878691	1255.434665	1073.285338
test_f1	0.05235602	0	0.02139037	**0.02458213176**
test_precision	0.35714286	0	0.18181818	0.1796536797
test_recall	0.02824859	0	0.01136364	0.01320407464

2)SVC Classifier
Best SVC Results I got

Best (in terms of mean F1) SVC result I got				
Parameters	**C=5096**	**kernel=linear**	**degree=3**	
	result1	result2	result3	mean result
fit_time	145.9461455	144.4341445	142.0430734	144.1411211
score_time	147.3889756	146.7131875	142.8233745	145.6418459
test_f1	0.58709677	0.66463415	0.60534125	**0.6190240556**
test_precision	0.68421053	0.7218543	0.63354037	0.6798684012
test_recall	0.51412429	0.61581921	0.57954545	0.5698296525

WILLIAMS

1)KNN Neighbor

- **When neighbors =1**

	result1	result2	result3	mean result
fit_time	6.40420556	5.93736672	5.89815927	6.079910517
score_time	1070.405849	1811.029706	2575.787887	1819.074481
test_f1	0.16666667	0.1	0.27272727	**0.1797979798**
test_precision	0.33333333	0.33333333	0.6	0.4222222222
test_recall	0.11111111	0.05882353	0.17647059	0.1154684096

- **When neighbors =3**

	result1	result2	result3	mean result
fit_time	5.78136134	5.83775783	5.88008976	5.833069642
score_time	960.3972204	968.7497985	13386.61901	5105.255343
test_f1	0	0	0	**0**
test_precision	0	0	0	0
test_recall	0	0	0	0

- **When neighbors =5**

	result1	result2	result3	mean result
fit_time	6.26647162	7.49847245	6.23395371	6.666299264
score_time	1080.939908	1122.361091	986.4937611	1063.26492
test_f1	0	0	0	**0**
test_precision	0	0	0	0
test_recall	0	0	0	0

2)SVC Classifier
Best SVC Results I got

Parameters	**C=71000**	**kernel=linear**	**degree=3**	
	result1	result2	result3	mean result
fit_time	36.90879774	36.8207283	33.24644923	35.65865843
score_time	35.95784855	36.35571218	33.4667174	35.26009274
test_f1	0.33333333	0.48	0.56	**0.4577777778**
test_precision	0.66666667	0.75	0.875	0.7638888889
test_recall	0.22222222	0.35294118	0.41176	0.3289760349

Conclusion:

BUSH: Best KNN results where obtained when neighbors = 1, **mean f1=0.1346856725**

SVC Result for C=5096,kernel = linear, **mean f1 = 0.6190240556**

WILLIAMNS: Best KNN results where obtained when neighbors = 1, **mean f1=0. 0.1797979798**

SVC Result for C=5096,kernel = linear, **mean f1 = 0. 0.4577777778**

Phase2:

In this phase we loaded and fit the data after transforming using PCA(Principal component analysis) and then apply the classifier, KNN and SVC.

BUSH

Since KNN with **neighbor =1** was the best result in phase 1 we will apply PCA with the same best results.

PCA parameters	**n_components=200,whiten='true',svd_solver='arpack'**
KNeighborsClassifier parameters	n_neighbors=1
Mean F1	**0.1623593424**

SVC Classifier:

Best result for SVC	
PCA parameters	**n_components=3800,random_state=5095**
SVC parameters	C=1000, kernel='linear', degree = 3
Mean F1	**0.6197862422**

WILLIAMS

Since KNN with **neighbor =1** was the best result in phase 1 we will apply PCA with the same best results.

PCA parameters	n_components=50,svd_solver='full'
KNeighborsClassifier parameters	n_neighbors=1
Mean F1	**0.2573260073**

SVC Classifier:

PCA parameters	n_components=2400,random_state=5095
SVC parameters	C=300, kernel='linear', degree = 3
Mean F1	**0.4772649573**

Conclusion:

Phase 2 v/s Phase 1

BUSH: knn n_neighbor=1 **f1=0.1346856725 and increased to f1=0.1623593424**

SVC classfier: f1 = 0.6190240556 and increased to f1 = 0.6190240556

WILLIAMS: knn n_neighbor=1 **f1=0. 0.1797979798 and increased to f1=0.2573260073**

SVC classfier: f1 = 0.4577777778 and increased to f1 = 0.4772649573

Phase3:

In this phase we will use a deep learning model to train the model and use train_test_split for splitting the data in train and test.

The images to be used for deep learning model need to be reshaped into 64*64

Deep learning model we will use has CNN and Maxpooling and then flatten the model and use dense layers before the sigmoid output layer.

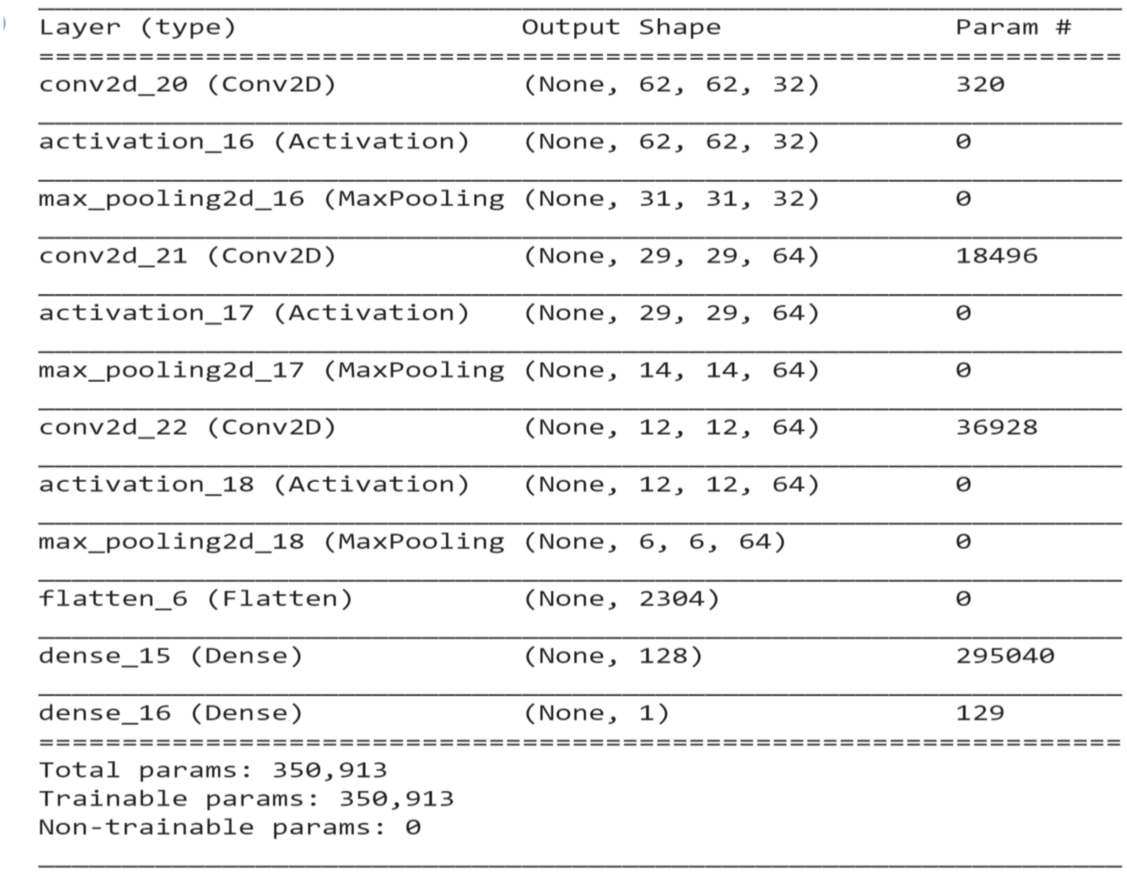
BUSH

Model used for bush is as follows:

3 Convolution layers

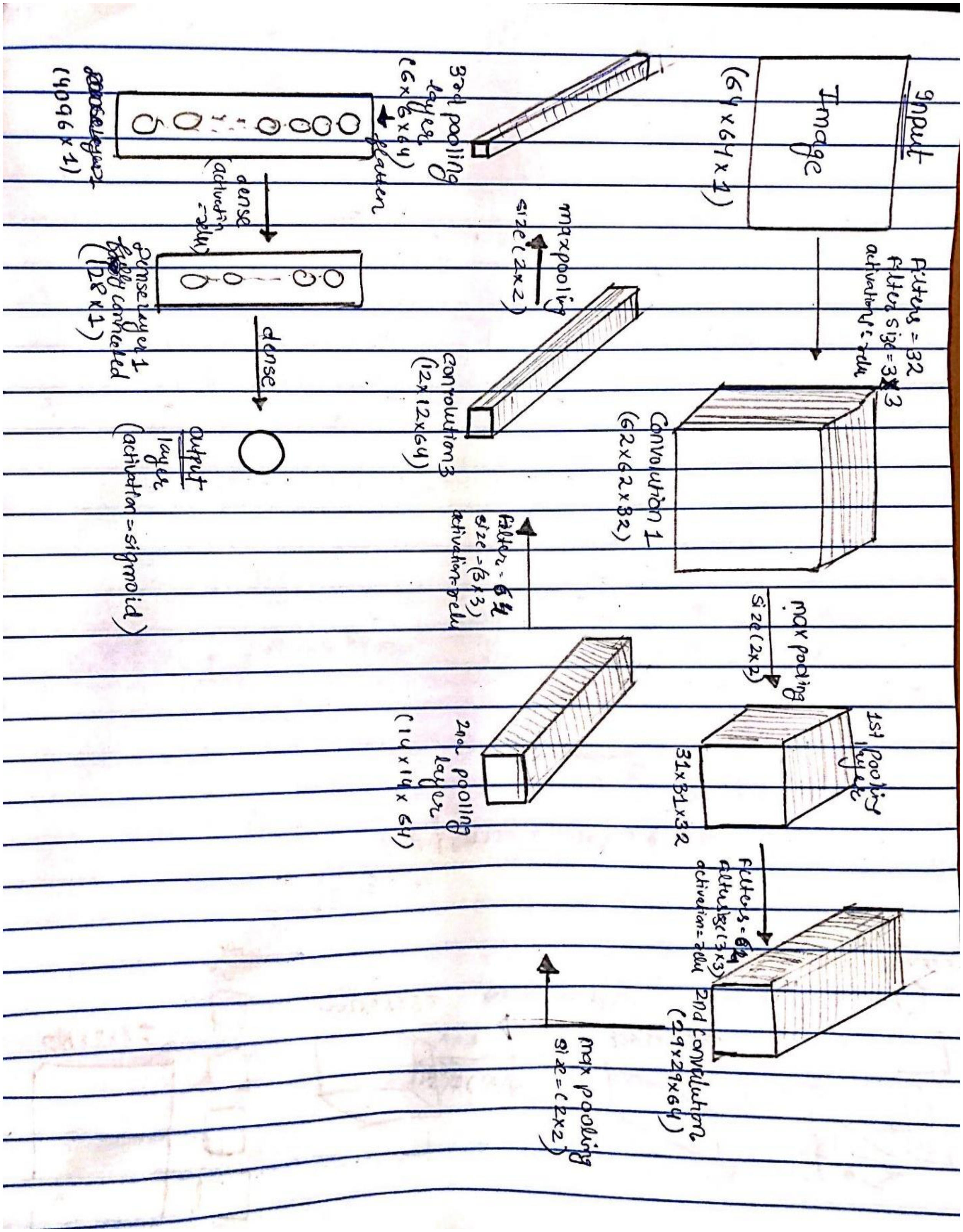
3 Maxpooling

1 Dense Layer(size =128)



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_20 (Conv2D)           (None, 62, 62, 32)        320       
_________________________________________________________________
activation_16 (Activation)   (None, 62, 62, 32)        0         
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 31, 31, 32)        0         
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 29, 29, 64)        18496     
_________________________________________________________________
activation_17 (Activation)   (None, 29, 29, 64)        0         
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 12, 12, 64)        36928     
_________________________________________________________________
activation_18 (Activation)   (None, 12, 12, 64)        0         
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 6, 6, 64)          0         
_________________________________________________________________
flatten_6 (Flatten)          (None, 2304)              0         
_________________________________________________________________
dense_15 (Dense)             (None, 128)               295040    
_________________________________________________________________
dense_16 (Dense)             (None, 1)                 129       
=================================================================
Total params: 350,913
Trainable params: 350,913
Non-trainable params: 0
_________________________________________________________________
```

model.fit(X_reshaped, y_train,**batch_size=32,epochs=20**,verbose=1,validation_data=(Xtest_reshaped, y_test))


Input
Image
(64 x 64 x 1)
Filters = 32
Filter size = 3x3
activations = relu
Convolution 1
(62 x 62 x 32)
max pooling
size (2x2)
1st pooling layer
31x31x32
Filters = 64
Filter size (3x3)
activation = relu
2nd convolution
(29x29x64)
max pooling
size = (2x2)
2nd pooling layer
(14 x 14 x 64)
Filter = 64
size = (3x3)
activation = relu
Convolution 3
(12x12x64)
max pooling
size (2x2)
3rd pooling layer
(6x6x64)
flatten
~~Dense layer~~
(4096 x 1)
dense
(activation = relu)
Dense layer 1
fully connected
(128 x 1)
dense
output layer
(activation = sigmoid)

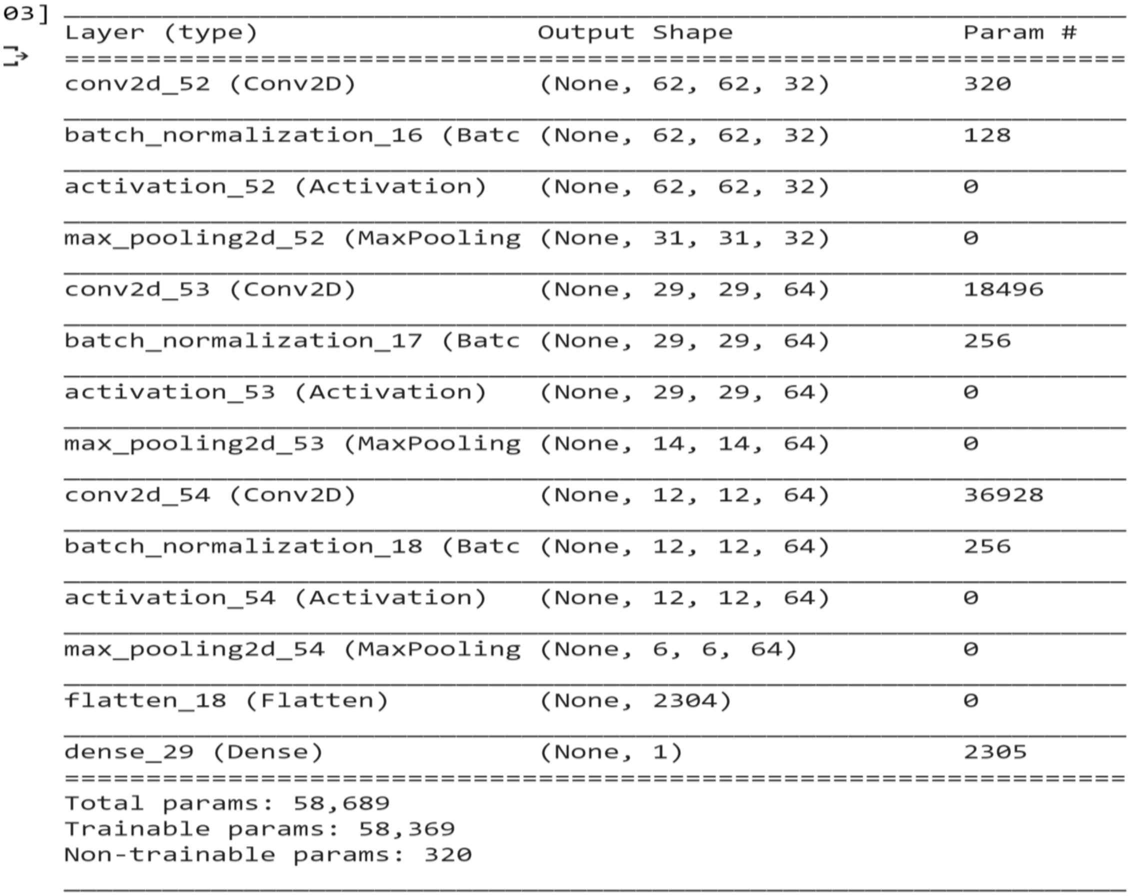
WILLIAMS

Model used for williams with batch processing is as follows:

3 Convolution layers

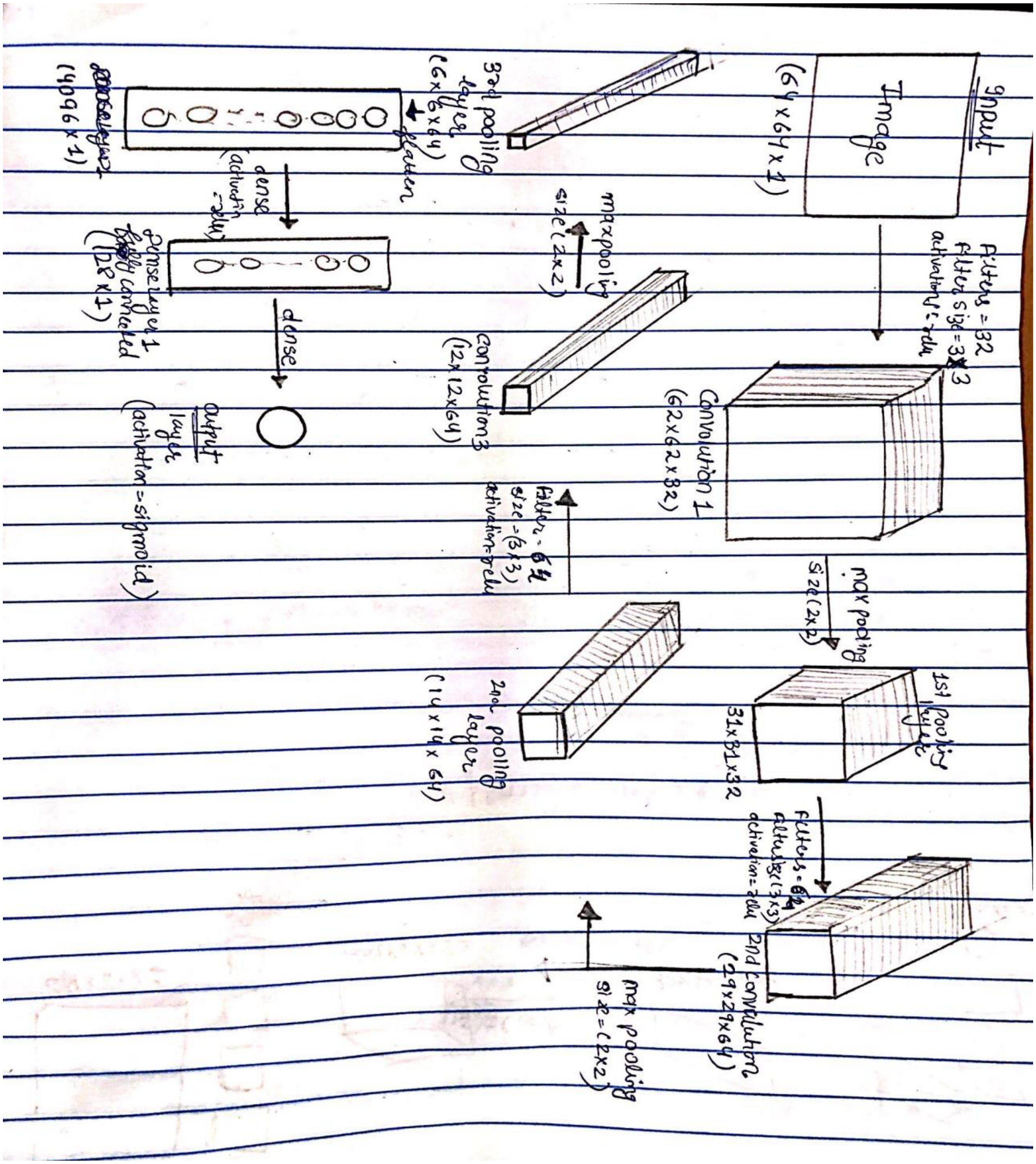
3 Maxpooling

1 output layer (sigmoid)



```
03] _________________________________________________________________
    Layer (type)                 Output Shape              Param #
    =================================================================
    conv2d_52 (Conv2D)           (None, 62, 62, 32)        320
    _________________________________________________________________
    batch_normalization_16 (Batc (None, 62, 62, 32)        128
    _________________________________________________________________
    activation_52 (Activation)   (None, 62, 62, 32)        0
    _________________________________________________________________
    max_pooling2d_52 (MaxPooling (None, 31, 31, 32)        0
    _________________________________________________________________
    conv2d_53 (Conv2D)           (None, 29, 29, 64)        18496
    _________________________________________________________________
    batch_normalization_17 (Batc (None, 29, 29, 64)        256
    _________________________________________________________________
    activation_53 (Activation)   (None, 29, 29, 64)        0
    _________________________________________________________________
    max_pooling2d_53 (MaxPooling (None, 14, 14, 64)        0
    _________________________________________________________________
    conv2d_54 (Conv2D)           (None, 12, 12, 64)        36928
    _________________________________________________________________
    batch_normalization_18 (Batc (None, 12, 12, 64)        256
    _________________________________________________________________
    activation_54 (Activation)   (None, 12, 12, 64)        0
    _________________________________________________________________
    max_pooling2d_54 (MaxPooling (None, 6, 6, 64)          0
    _________________________________________________________________
    flatten_18 (Flatten)         (None, 2304)              0
    _________________________________________________________________
    dense_29 (Dense)             (None, 1)                 2305
    =================================================================
    Total params: 58,689
    Trainable params: 58,369
    Non-trainable params: 320
    _________________________________________________________________
```

model.fit(X_trainreshaped_w, y_train,**batch_size=32,epochs=20**,verbose=1,validation_data=(X_testreshaped_w, y_test))



Conclusion: F_1_train_bush: 1.0, F_1_test_bush: 0.88622

F_1_train_williams: 1.0,F_1_test_williams: 0.749999999

F_1 increases for the same dataset if we use CNN and maxpooling and not classifiers like KNN and SVC.

Phase 4:

In this phase we are performing transfer learning by selecting a different dataset for image classification and then fitting the model as per the data and transforming it using keras libraries into a "initialized-model.keras" and after that we will use the same model to test the accuracy and f1 by trying to fit the given bush and Williams train and test data test using the "initialized-model.keras" saved for metric performance.

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

Datasets used: **Dogs v/s Cats** which has two files train and test.

Train has 25000 images and 12500 images in test.

Steps performed**:**

1) Labeled the images in binary format

Code: if word_label == 'cat': return 0

elif word_label == 'dog': return 1

2)Grayscale the images and reshape it into 64 by 64 size of the original data

Code: img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)

img = cv2.resize(img, (64, 64))

3)**Perform CNN and maxpooling and obtain a initialized-model.keras from the train and test data of dogs and cats images using keras.**

Using this initialized-model.keras we will perform on bush and Williams train and test data and gain the f1 obtained using different image classification dataset and the model fit and transformed from that dataset.

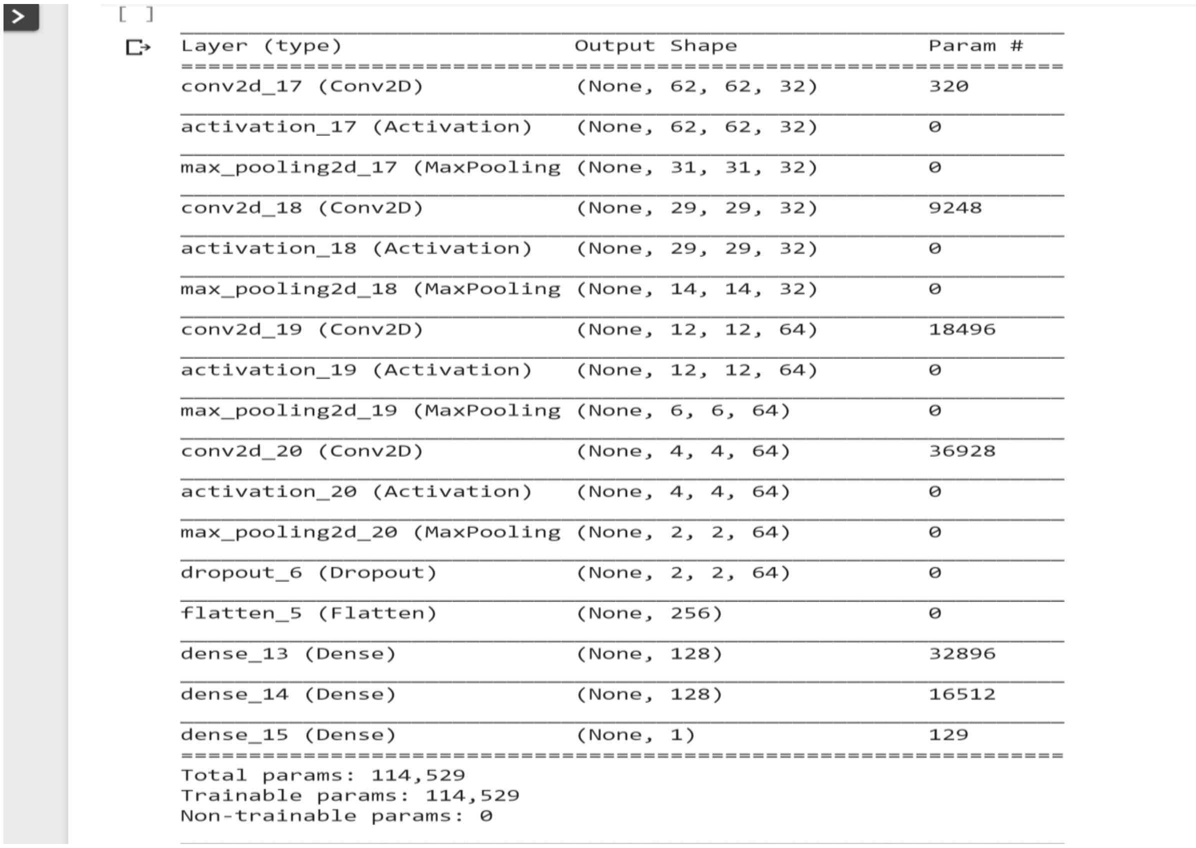
Model used is as follows:

4 Convolution layers(relu)

4 Maxpooling

2 Dense Layer(size1 =128,size2=128,relu)

1 output layer(sigmoid)

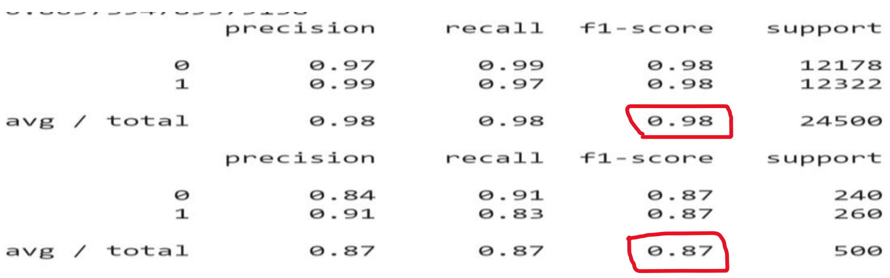


```
[ ]
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 62, 62, 32)        320
_________________________________________________________________
activation_17 (Activation)   (None, 62, 62, 32)        0
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 29, 29, 32)        9248
_________________________________________________________________
activation_18 (Activation)   (None, 29, 29, 32)        0
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 12, 12, 64)        18496
_________________________________________________________________
activation_19 (Activation)   (None, 12, 12, 64)        0
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
activation_20 (Activation)   (None, 4, 4, 64)          0
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 2, 2, 64)          0
_________________________________________________________________
dropout_6 (Dropout)          (None, 2, 2, 64)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_13 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_14 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 129
=================================================================
Total params: 114,529
Trainable params: 114,529
Non-trainable params: 0
_________________________________________________________________
```

Parameters for model

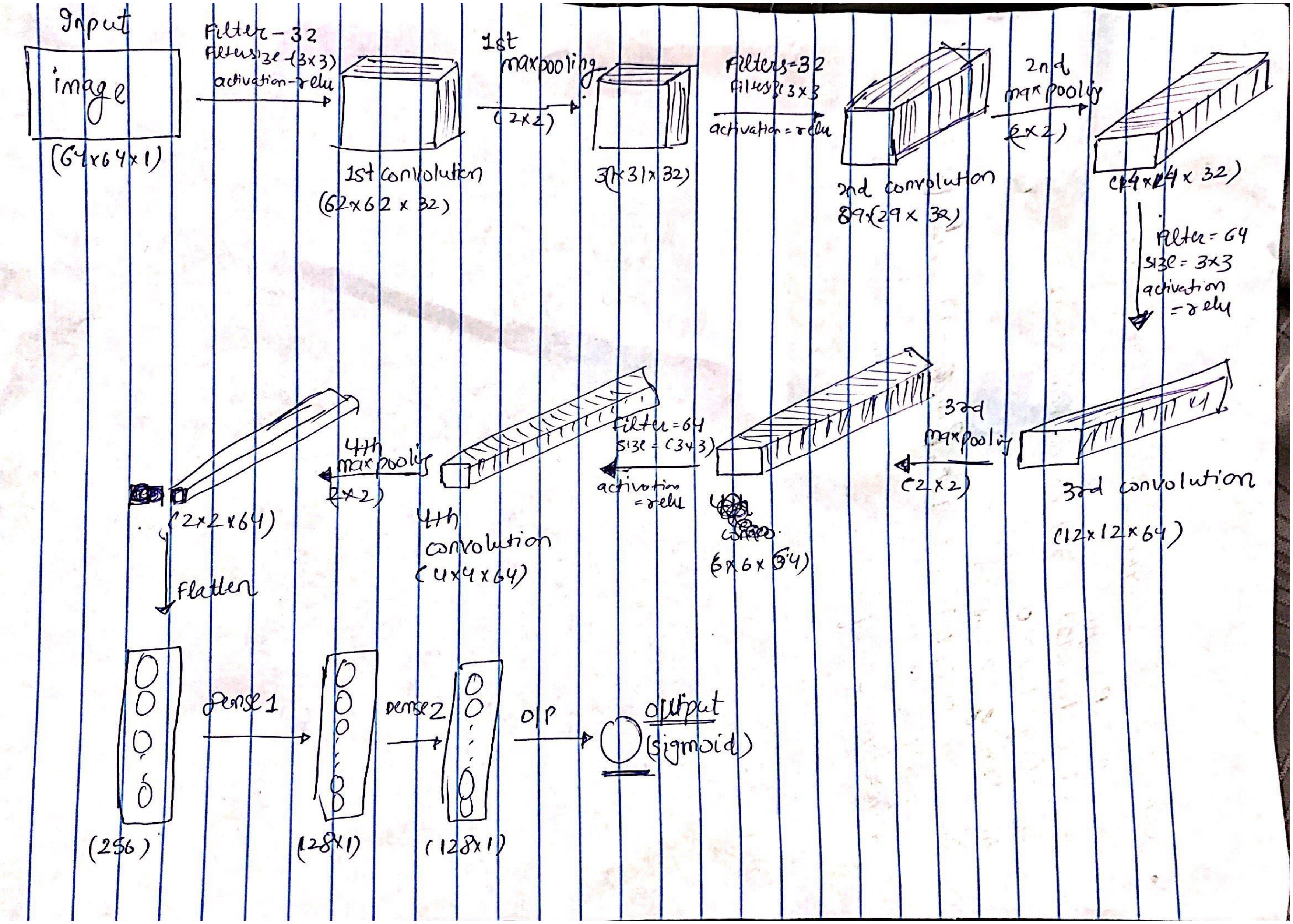
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

m_fit = model.fit(X_train, Y_train, epochs = 20, verbose = 1, validation_data = (X_test, Y_test))



```
             precision    recall  f1-score   support

          0       0.97      0.99      0.98     12178
          1       0.99      0.97      0.98     12322

avg / total       0.98      0.98      0.98     24500
             precision    recall  f1-score   support

          0       0.84      0.91      0.87       240
          1       0.91      0.83      0.87       260

avg / total       0.87      0.87      0.87       500
```

Save this model as “initialized-model.keras”

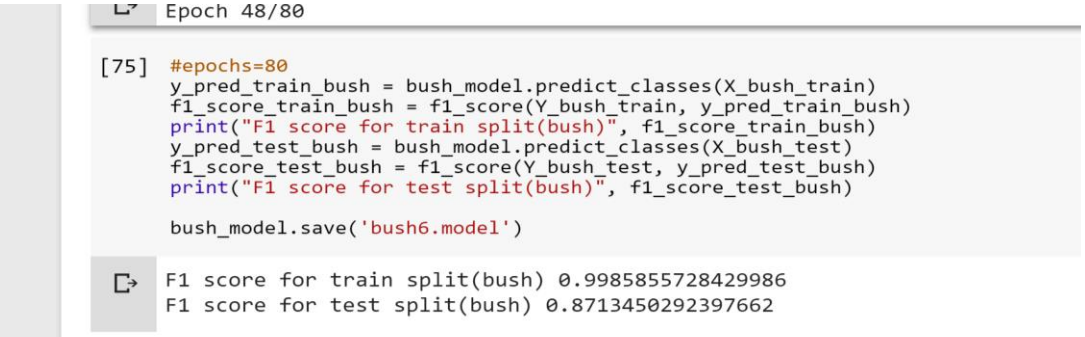

Input
image
(64x64x1)
Filter - 32
Filtersize -(3x3)
activation-relu
1st convolution
(62x62 x 32)
1st
maxpooling
(2x2)
31x31x32)
Filters=32
Filtersize 3x3
activation = relu
2nd convolution
29x29 x 32)
2nd
max pooling
(2x2)
(14x14 x 32)
Filter = 64
size = 3x3
activation
= relu
3rd convolution
(12x12x64)
3rd
maxpooling
(2x2)
(6x6 x 64)
Filter=64
size = (3x3)
activation
= relu
4th
convolution
(4x4x64)
4th
maxpooling
(2x2)
(2x2x64)
Flatten
(256)
Dense 1
(128x1)
Dense2
(128x1)
O/P
output
(sigmoid)

After performing on bush and Williams with this model we get,

BUSH:

bush_model = load_model('initialized-model1.keras')

bushmodel_fit = bush_model.fit(X_bush_train, Y_bush_train, **epochs = 80**, verbose = 1, validation_data = (X_bush_test, Y_bush_test))



```
Epoch 48/80
```

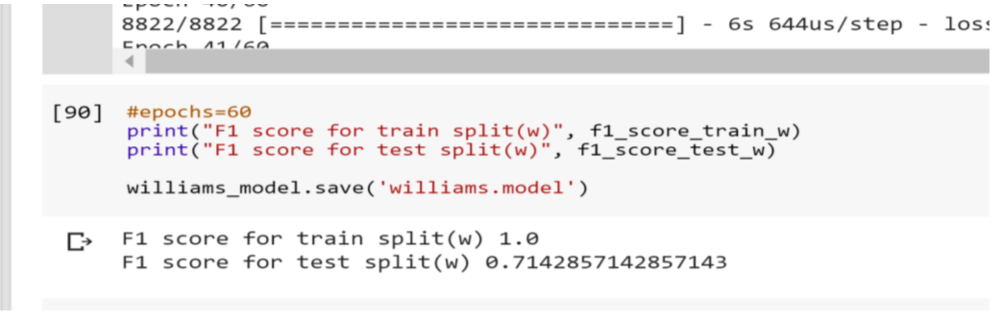
```
[75] #epochs=80
     y_pred_train_bush = bush_model.predict_classes(X_bush_train)
     f1_score_train_bush = f1_score(Y_bush_train, y_pred_train_bush)
     print("F1 score for train split(bush)", f1_score_train_bush)
     y_pred_test_bush = bush_model.predict_classes(X_bush_test)
     f1_score_test_bush = f1_score(Y_bush_test, y_pred_test_bush)
     print("F1 score for test split(bush)", f1_score_test_bush)

     bush_model.save('bush6.model')
```

```
F1 score for train split(bush) 0.9985855728429986
F1 score for test split(bush) 0.8713450292397662
```

WILLIAMS:

williams_model = load_model('initialized-model1.keras')

williamsmodel_fit = williams_model.fit(X_williams_train, Y_williams_train, epochs = 60, verbose = 1, validation_data = (X_williams_test, Y_williams_test))



```
8822/8822 [==============================] - 6s 644us/step - los
```

```
[90] #epochs=60
     print("F1 score for train split(w)", f1_score_train_w)
     print("F1 score for test split(w)", f1_score_test_w)

     williams_model.save('williams.model')
```

```
F1 score for train split(w) 1.0
F1 score for test split(w) 0.7142857142857143
```

Conclusion: **BUSH: F1 score for train:0.9985855728429986**

F1 score for test:0.8713450292397662

Williams: F1 score for train: 1.0

F1 score for test: 0.714285

The results in Phase 3 varies slightly in bush it was 0.88 and now its 0.87 after applying transfer learning and Williams varies a slight by 0.74 to 0. , thus we can say that transfer learning either improves the results or keeps near to constant for the initial model and not decreases the f1 rate by a huge difference.