Project Report

Introduction

This project is on Face Recognition where we took datasets containing various images and had to identify whether it’s a Bush/Williams face vs all other faces present in the dataset using various machine learning strategies. We used various parameters for measuring accuracy, precision, recall and/or f1_score.

- Dataset Information:
  - We had 3 csv files which had following details:
    - X.csv: All the photos, where each row was one photo and each column represent pixels of that photo. The photos were in the form of 64x64x1 (here 1 means Gray-scaled image). There are total of 13,233 photos in this file.
    - Y_bush.csv: Labels for the bush images vs all other images.
    - Y_williams.csv: Labels for the Williams images vs all other images.
- The project is divided into four phases where each phase had different things to be calculated and analyzed. The description of all the phases is below.

Phase 1

- This phase requires us to use various classifiers to calculate f1_score and report the results.
- The classifiers used where:
  - KNeighborsClassifier with neighbors = [1,3,5]
  - SVC with various values for for the parameter C, kernel, gamma and degree.
- For KNeighborsClassifier :
  - First, we read the X.csv, Y_bush.csv and y_williams.csv files and then for neighbors = 1,3 and 5, we initialize KNeighborsClassifier's object.
  - Once this is done, we use StratifiedKFold for performing cross validation on the input data.
  - Using this result, we retrieve f1_score for all the neighbor parameters and report it.
  - I also used n_jobs = -1, which enables all cores to be used for your calculation.
  - We do this for both Bush and Williams separately and note down the results accordingly by using appropriate y_bush and y_williams files.
- For SVC,
  - We initialize SVC classifier's object by changing various parameters for C (10, 1000, 10,000, …), kernel (poly, linear, rbf, sigmoid), degree (1,2,3,5, …), gamma(0.1, 0.001, 0.0001, …), coef0(100, 200, …).
  - Using SVC's object, we perform cross validation by using StratifiedKFold.
  - I also used n_jobs = -1, which enables all cores to be used for your calculation.
  - These steps are repeated for Bush as well as Williams by using appropriate label files.
  - The best f1_score is retrieved from the cross validation's result for various parameters and is reported.
- The f1_scores were pickled and stored.
- Results:

I. Bush:

Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=1**			
	result1	result2	result3	mean result
fit_time	9.67905235	9.40821075	9.79015207	9.62580506
score_time	1024.382646	1021.918452	1028.556778	1024.95263
test_f1	0.13095238	0.16286645	0.14646465	0.14676116
test_precision	0.13836478	0.19230769	0.13181818	0.15416355
test_recall	0.12429379	0.14124294	0.16477273	0.14343649

Table 1

Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=3**			
	result1	result2	result3	mean result
fit_time	8.81445074	9.15254259	8.7925868	8.91986004
score_time	1023.137986	1018.936136	1021.999099	1021.35774
test_f1	0.08530806	0.07920792	0.07476636	0.07976078
test_precision	0.26470588	0.32	0.21052632	0.2650774
test_recall	0.05084746	0.04519774	0.04545455	0.04716658

Table 2

Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=5**			
	result1	result2	result3	mean result
fit_time	9.20155191	8.81941533	8.62457275	8.88184666
score_time	1024.201898	1021.816223	1021.74425	1022.58746
test_f1	0.06349206	0.02173913	0.03191489	0.03904869
test_precision	0.5	0.28571429	0.25	0.3452381
test_recall	0.03389831	0.01129944	0.01704545	0.02074773

Table 3

SVC classifier parameters that returned a mean zero f1				
C	**kernel**	**degree**	**gamma**	**coef0**
1000	poly	2000	N/A	
2	poly	1	N/A	3
1	poly	2	N/A	
1	rbf	N/A	0.000884832	
1	rbf	N/A	0.000244141	
1	sigmoid	N/A	0.000244141	

Table 4

Best (in terms of mean F1) SVC result I got				
Parameters	C = 1000000	kernel = linear		
	result1	result2	result3	mean result
fit_time	70.42077136	74.69381189	75.87962961	73.6647376
score_time	73.16128469	76.15634441	76.2215147	75.1797146
test_f1	0.6116208	0.64150943	0.62573099	0.62628707
test_precision	0.66666667	0.72340426	0.64457831	0.67821641
test_recall	0.56497175	0.57627119	0.60795455	0.58306583

Table 5

II. **Williams**:

Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=1**			
	result1	result2	result3	mean result
fit_time	9.13664937	9.44431186	9.31736064	9.299440623
score_time	1116.657835	1115.540049	1117.202809	1116.466897
test_f1	0.19047619	0.15384615	0.34782609	0.230716143
test_precision	0.66666667	0.22222222	0.66666667	0.51851852
test_recall	0.11111111	0.11764706	0.23529412	0.154684097

Table 6

Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=3**			
	result1	result2	result3	mean result
fit_time	10.42673373	10.39670777	10.21153235	10.34499128
score_time	1151.078742	1149.67174	1151.463399	1150.73796
test_f1	0	0	0	0
test_precision	0	0	0	0
test_recall	0	0	0	0

Table 7

Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=5**			
	result1	result2	result3	mean result
fit_time	11.27881193	11.57988501	11.04236054	11.30035249
score_time	1103.933918	1102.088767	1104.687204	1103.569963
test_f1	0	0	0	0
test_precision	0	0	0	0
test_recall	0	0	0	0

Table 8

SVC classifier parameters that returned a mean zero f1				
C	**kernel**	**degree**	**gamma**	**coef0**
0.001	poly	1	N/A	1
1	rbf	N/A	0.0001	
1000	rbf	N/A	0.000001	
1	rbf	N/A	0.001	
1	rbf	N/A	0.000244141	
1	sigmoid	N/A	0.000244141	

Table 9

Best (in terms of mean F1) SVC result I got				
Parameters	C = 100	kernel = linear		
	result1	result2	result3	mean result
fit_time	15.45404792	14.06917262	14.87692523	14.80004859
score_time	16.14169788	15.11384058	16.27359056	15.84304301
test_f1	0.6	0.34782609	0.5	0.482608697
test_precision	0.75	0.66666667	0.63636364	0.684343437
test_recall	0.5	0.23529412	0.41176471	0.382352943

Table 10

Analysis:

- The f1_score was higher for Bush as compared to Williams because there are more images for Bush in the dataset as compared to Williams.
- For Bush, as we increased the number of neighbors, the f1_Score seem to be decreasing.
- Similarly, for Williams, as we increase number of neighbors, the f1_score went down. For neighbors 3 and 5, mean f1_score was 0.0
- SVC classifier was tried with various parameters and we got 0.0 for many of those parameters listed in tables above for Bush and Williams respectively.
- The f1_score for best SVC parameter is listed above for both Bush and Williams respectively.

Phase 2

- This phase uses PCA component analysis for calculating the f1_score.
- We first read the data from csv files, then we initialize PCA's object with various n_components parameters.
- The input data is then fitted and transformed using above PCA's object.
- Now, we perform KNeighbors classification and SVC classification by changing various parameters of PCA, KNN and SVC for Bush as well as Williams as done in the phase 1 of this project (except for PCA in this case).
- We use cross validation technique using three-fold cross-validation and report best f1_score for Kneighbors and SVC.
- The f1_score for Bush and Williams using KNeighbors and SVC is reported.
- Results:

i. **Bush**:

Best result for KNeighborsClassifier	
PCA parameters	n_components=64
KNeighborsClassifier parameters	n_neighbors=1
Mean F1	0.162520803

Table 11

Best result for SVC	
PCA parameters	n_components=3000, random_state = 6925
SVC parameters	C = 1000, kernel = 'poly', degree = 15,gamma = 0.001, coef0 = 200
Mean F1	0.634924692

Table 12

ii. **Williams**:

Phase 2 best results	
Best result for KNeighborsClassifier	
PCA parameters	n_components=170
KNeighborsClassifier parameters	n_neighbors=1
Mean F1	0.32764359

Table 13

Best result for SVC	
PCA parameters	n_components=2250, random_state = 6925
SVC parameters	C = 1, kernel = 'sigmoid', gamma = 0.000244141
Mean F1	0.51594203

Table 14

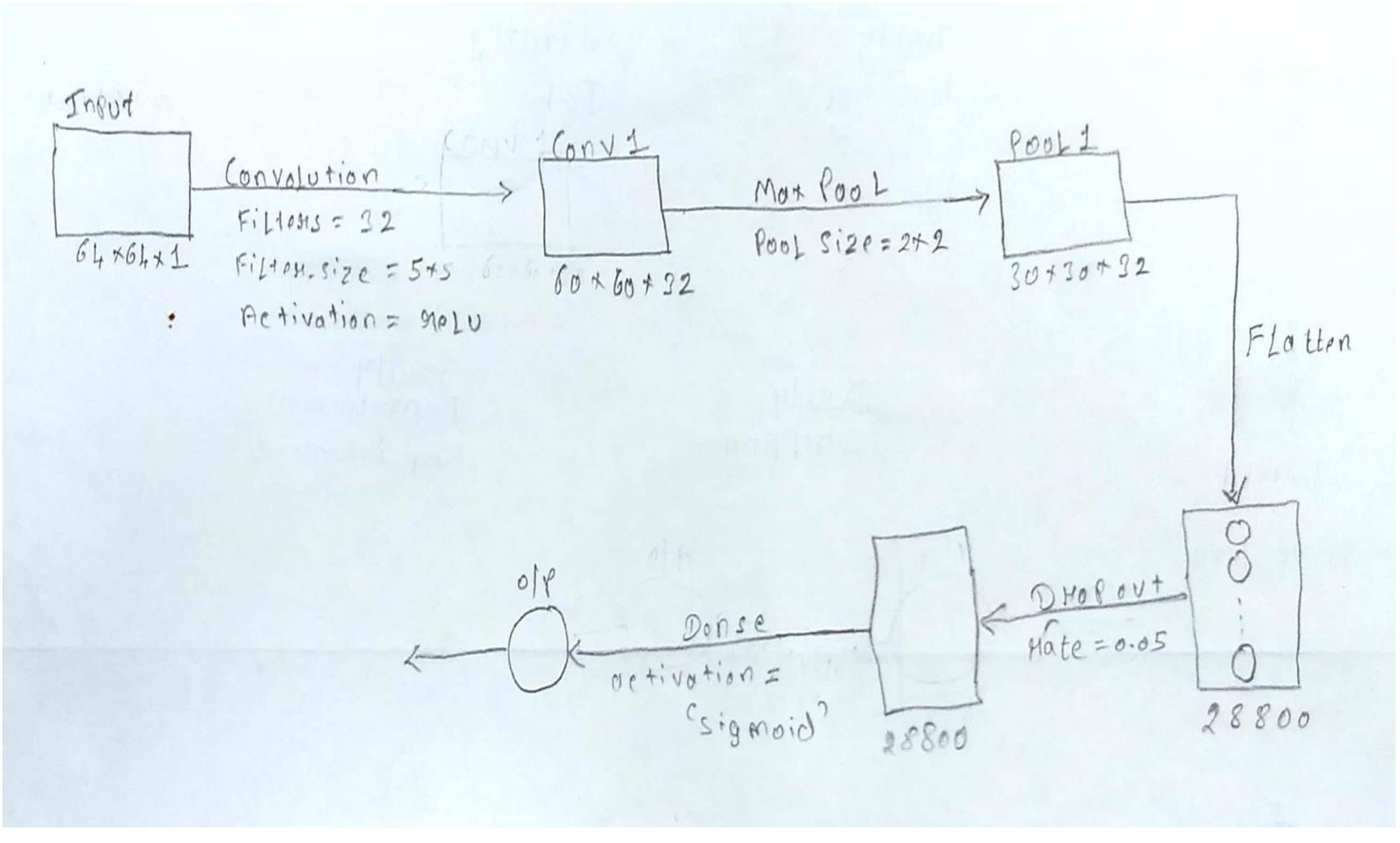
Analysis:

- I used n_components=64 for Bush and n_components=170 for Williams for performing KNeighbors classification.
- For SVC, I used n_components=3000, random_state = 6925 for Bush and n_components=2250, random_state = 6925 for Williams.
- The f1_scores are reported in the tables above.
- Also, SVC gives better f1_score as compared to Knn for this dataset.

Phase 3

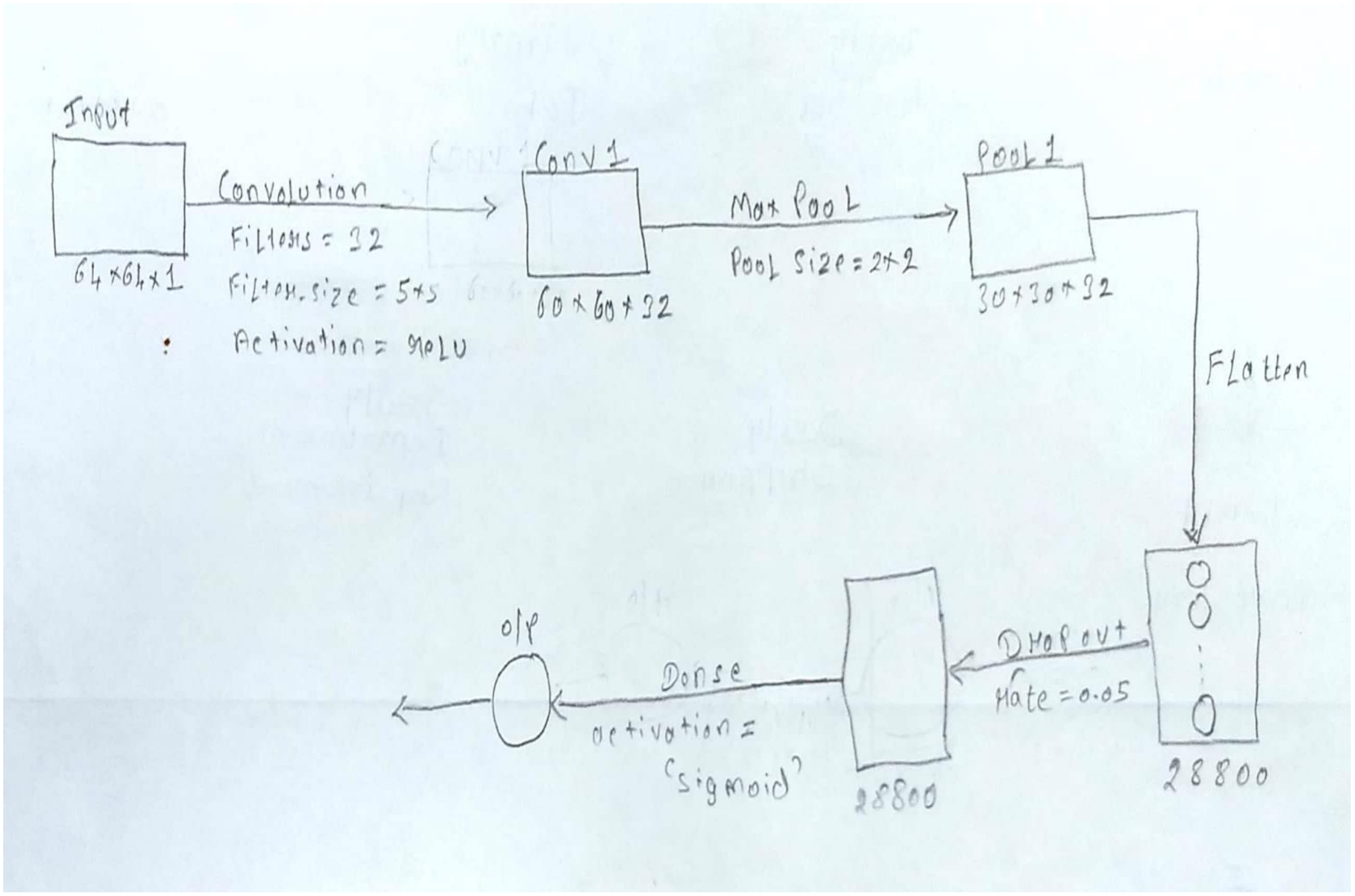
- We have to develop a deep learning model for Bush and Williams.
- Since it is a image classification model, using convolution and maxpooling layers would give good results for F1 scores.
- We perform Train-Test split evaluation for calculating F1_Scores.
- Test data = 1/3rd of original data.
- Shuffle= True for train_test_split and stratify was set to appropriate y_bush or y_williams for Bush and Williams respectively.
- Sequential model is ideal for Image classification and the layers were:
    - Convolution layer with relu as activation function.
    - Maxpool layer
    - Flatten
    - Dropout
    - Dense with sigmoid as activation.
- The model is fitted on the data and ran for epochs = 10 for Bush and epochs = 8 for Williams
- We save the model and do predict_classes on this model and then calculate f1_score using sklearn's merics.
- The f1_score is calculated for train and test data for Bush and Williams and is reported below.
- Reports:

I. **Bush**:



- The above architecture which shows the Bush model with different layers showing their dimensions along with input and output size.
- It also shows various activation function used at different type of layers.
- My model has following details:
  - Input:
    - 64x64x1
  - Convolution layer:
    - Filters = 32
    - Kernel size = 5x5
    - Activation = relu
    - Output = 60x60x32
  - Max Pooling:
    - Pool size = 2x2
    - Output = 30x30x32
  - Flatten:
    - Output = 1D – 28800
  - Dropout:
    - Rate = 0.05
    - Output: 1D – 28800
  - Output Layer:
    - Dense
    - Activation = sigmoid
- The f1 score that I could achieve was:
  - Train data: 0.9800000000000001
  - Test data: 0.7761194029850746

ii. **Williams**:



- The above architecture which shows the Williams model with different layers showing their dimensions along with input and output size.
- It also shows various activation function used at different type of layers.
- My model has following details:
    - Input:
        - 64x64x1
    - Convolution layer:
        - Filters = 32
        - Kernel size = 5x5
        - Activation = relu
        - Output = 60x60x32
    - Max Pooling:
        - Pool size = 2x2
        - Output = 30x30x32

- Flatten:
  - Output = 1D – 28800
- Dropout:
  - Rate = 0.05
  - Output: 1D – 28800
- Output Layer:
  - Dense
  - Activation = sigmoid

-The f1 score that I could achieve was:

- Train data: 0.8923076923076922
- Test data: 0.7096774193548386

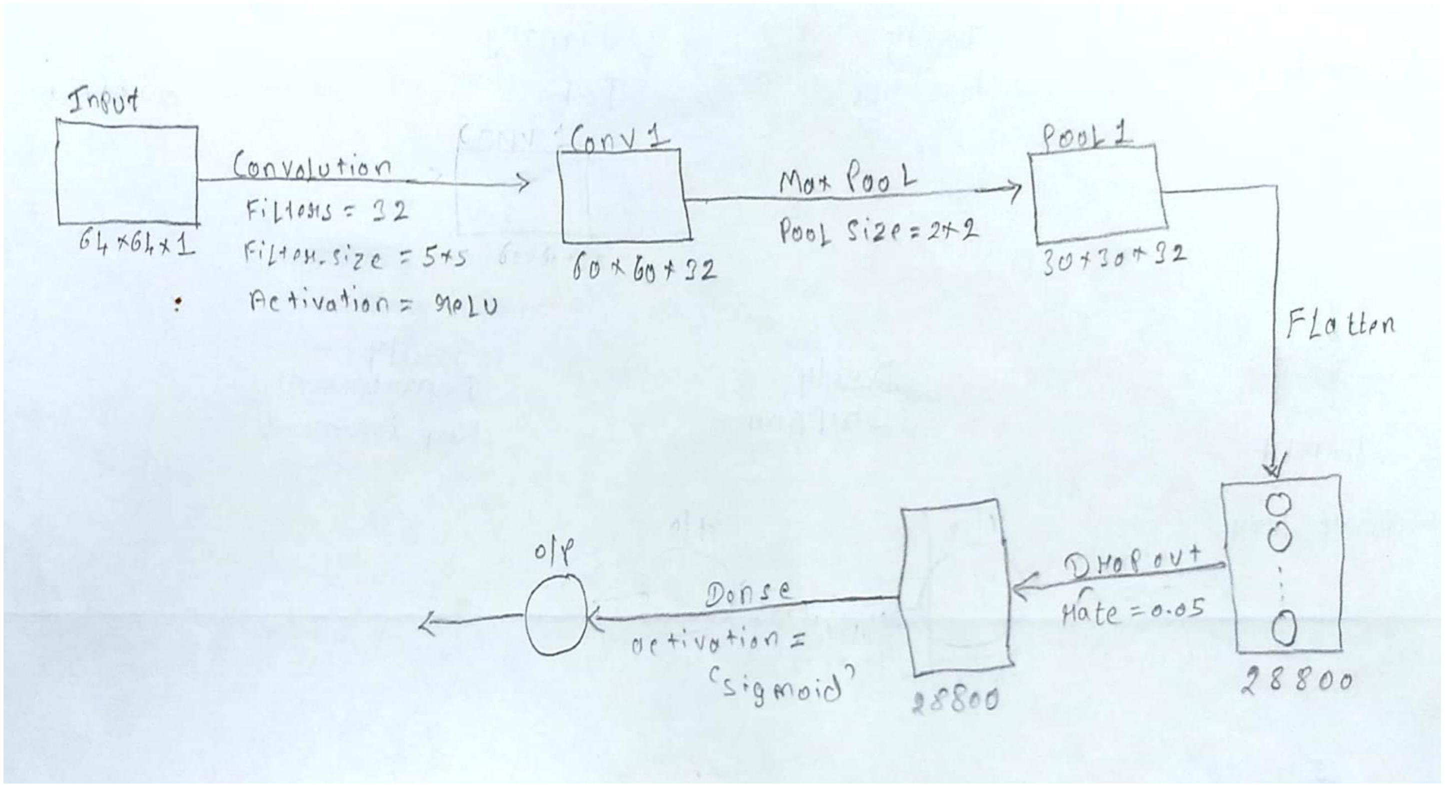
Analysis:

- The number of epochs I varied was from 3 to 15 for both of them. I got good result around 10 or 11 but the result kept changing as I re-run my program.
- I have used simple model as the dataset size is not that sufficient and if I use complex model, it may overfit the data.
- The f1 score for Bush is more comparatively to Williams as there are more examples for Bush in training set as compared to Williams and hence better training and prediction on the data which ultimately leads to more f1-score.

Phase 4

- This phase is about Transfer Learning – where we train our model on a different dataset and then use this trained model to our original dataset split and check f1_score.
- About the dataset:
  - The dataset I have used contains colored images of cats and dogs.
  - There are 10,500 images in total.
  - Link: https://www.kaggle.com/c/dogs-vs-cats/data
- The data was pre-processed in order to fit to our existing system as described below.
- Firstly, I read the image file and converted to gray-scale image using image.convert('L').
- After that I have resized the image to 64x64 and then converted image to an array of 2D pixels.
- The values of the arrays are normalized by dividing each entry with 255 to receive all values between 0 and 1.
- The label array is binary where, if the image is 'cat' it is 1 otherwise it is set to 0.
- We then create the model and train on this dataset and save this model as 'Initialize-model.keras'
- We run this task for various number of epochs and try by changing various layers of the model.
- After this, load the 'Intialize-model.keras' model and train it on Bush split and calculate f1_score from sklearn and report it.
- We do the same process for Williams model too.
- Save both Bush.model and Williams.model for the best f1_score achieved.
- The above steps are pretty similar to phase 3 except for the fact that instead of using randomly initialized model, we use a model which is trained on some other image dataset.
- Since it is a image classification model, using convolution and maxpooling layers would give good results for F1 scores.
- Sequential model is ideal for Image classification and the layers are:
  - Convolution layer with relu as activation function.
  - Maxpool layer
  - Flatten
  - Dropout
  - Dense with sigmoid as activation.
- We save the model and do predict_classes on this model and then calculate f1_score using sklearn..
- The f1_score is calculated for train and test data for Bush and Williams and is reported below.
- Reports:

i. **Bush**:

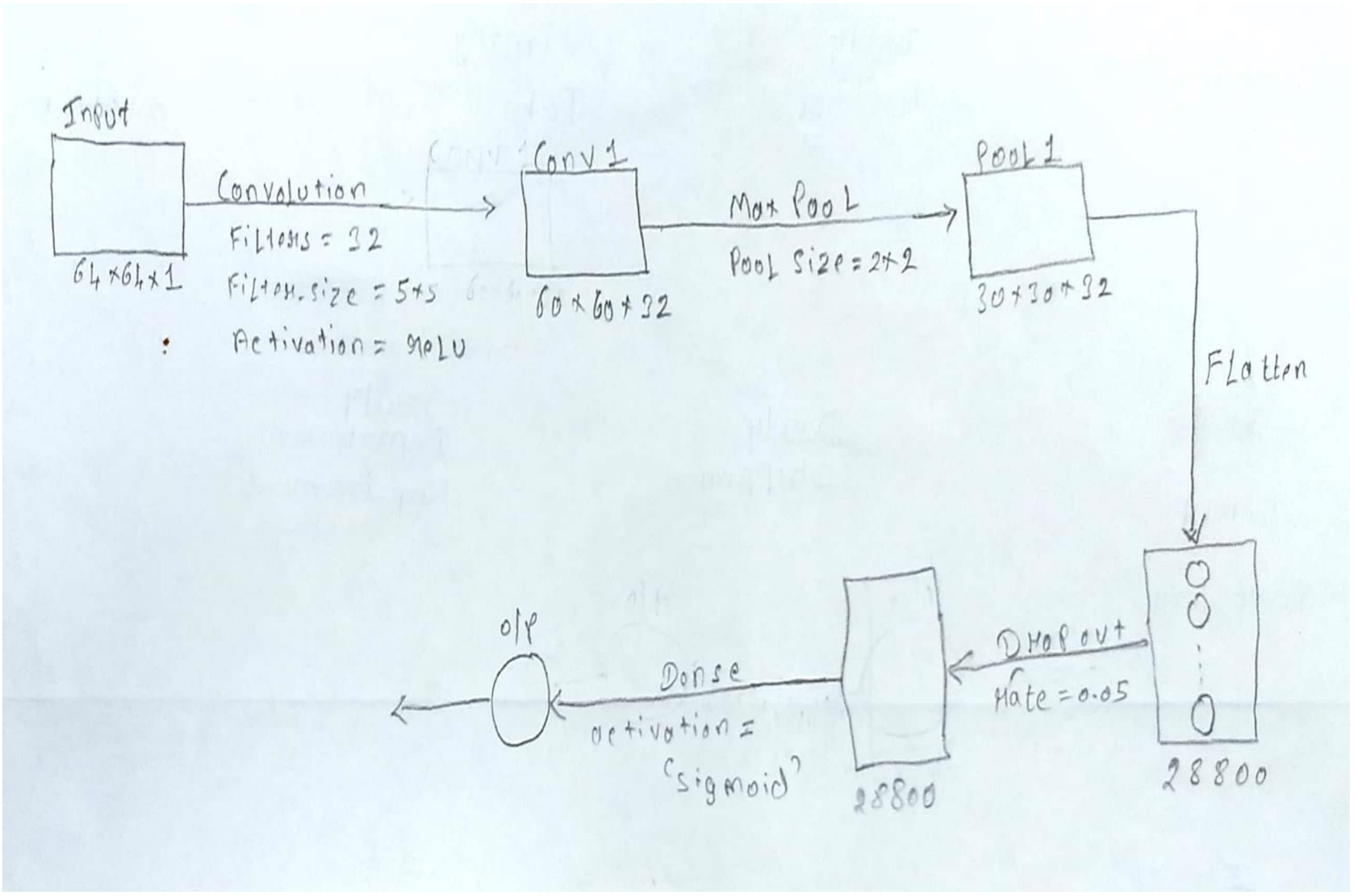


- The above architecture which shows the Bush model with different layers showing their dimensions along with input and output size.
- It also shows various activation function used at different type of layers.
- My model has following details:
    - Input:
        - 64x64x1
    - Convolution layer:
        - Filters = 32
        - Kernel size = 5x5
        - Activation = relu
        - Output = 60x60x32
    - Max Pooling:
        - Pool size = 2x2
        - Output = 30x30x32
    - Flatten:
        - Output = 1D – 28800
    - Dropout:
        - Rate = 0.05
        - Output: 1D – 28800
    - Output Layer:

    - Dense
    - Activation = sigmoid

- The f1 score that I could achieve was:
  - Train data: 1.0
  - Test data: 0.7854984894259819

II. **<u>Williams</u>**:



- The above architecture which shows the Williams model with different layers showing their dimensions along with input and output size.
- It also shows various activation function used at different type of layers.
- My model has following details:
  - Input:
    - 64x64x1
  - Convolution layer:
    - Filters = 32

        - Kernel size = 5x5
        - Activation = relu
        - Output = 60x60x32
    - Max Pooling:
        - Pool size = 2x2
        - Output = 30x30x32
    - Flatten:
        - Output = 1D – 28800
    - Dropout:
        - Rate = 0.05
        - Output: 1D – 28800
    - Output Layer:
        - Dense
        - Activation = sigmoid

-The f1 score that I could achieve was:
- Train data: 0.9428571428571428
- Test data: 0.7058823529411765

<u>Analysis</u>:

- The number of epochs that I used for 'Initialized-keras/model' are 30 for Bush and 10 for Williams.
- The number of epochs used for Bush.model are 40 and, for Williams it is 18.
- The f1_score increases till certain number of epochs and then it starts coming down for both Bush and Williams.
- The f1 score for Bush is more comparatively to Williams as there are more examples for Bush in training set as compared to Williams and hence better training and prediction on the data which ultimately leads to more f1-score.

<u>Conclusion</u>:

- The f1_score is best for transfer learning case.
- The f1_score increased as we moved from Knn, SVC to knn,SVC using PCA and then it further increased for deep learning model which was overtaken by Transfer Learning model.
- The mean f1 or the f1_score is more for Bush as compared to Williams as the training set has more Bush data compared to Williams.