CS 584 Machine Learning Project report

Chethan Kothwal

A20423243

Phase 1:

Given dataset of 64x64 grayscale images having Bush and Williams SVM and KNN classifiers were used to develop a model and predict on new unseen data. The given dataset was split into test and train sets. The test and train folds are made by preserving the percentage of samples of each class. The value of nearest neighbours for KNN classifier was changed to 1, 3 and 5. The results were noted. More number of parameters had to be explored for SVM classifier. Support Vectors Classifier tries to find the best hyperplane to separate the different classes by maximizing the distance between sample points and the hyperplane. The following observations were made by parameter tuning to the SVM classifier:

- The kernel parameters select the type of hyperplane used to separate the data.
- Higher the gamma values of the SVM classifier, the more it tries to fit to the training set. It may lead to overfitting and results in lesser F1 on the test data.
- C is the penalty trade-off between smooth decision boundary and classifying the training points correctly. Increasing C will lead to overfitting.
- Degree is a parameter used for 'poly' kernel. Increasing degree leads to higher training times.

The best results obtained were:

Dataset	Best Test f1
Bush(KNN) n_neighbors =1	0.121255877
Williams (KNN) n_neighbors =1	0.24242424
Bush(SVM)	0.659746877
Williams(SVM)	0.538110573

Phase 2:

Principal component analysis is used to reduce the dimensionality of the data which speeds up the fitting of the machine learning algorithm which in this case was KNN and SVM classifier. The learning algorithm is too slow in phase 1 because the input dimension is too high, then using PCA to speed it up is a reasonable choice. The given dataset had 4096 features (64*64) which is reduced to around 1500 features which does not help much in improving F1 results but helps in better fitting time.

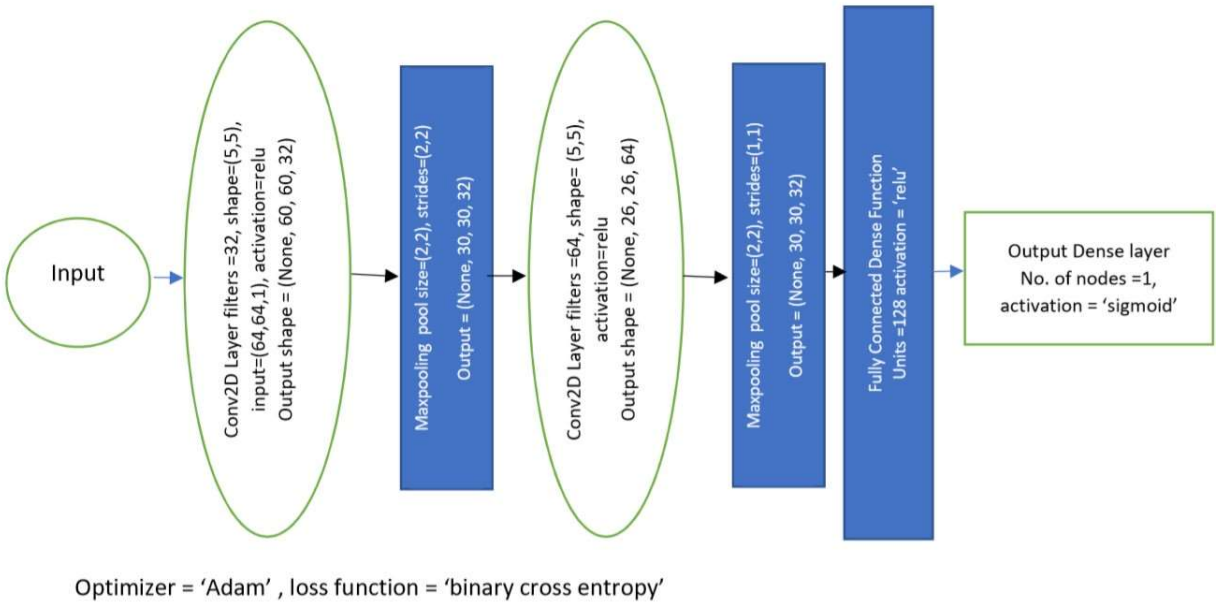
F1 scores:

Dataset	Best test f1
Bush KNN	0.131494287
Williams KNN	0.292238627
Bush SVM	0.661564337
Williams SVM	0.552529377

Phase 3:

A convolutional neural network was setup as it is an Image dataset. Use of CNN helps to reduce the number of parameter required for images over the regular NN. Also, it helps to do the parameter sharing so that it can possess translation invariance.

Bush CNN model description:



The sequential () model is used from keras. The input is given to a 2D convolution layer followed by a maxpooling operation on the resultant features. The feature vector is then an input to a 2nd 2D convolution layer and again pooled using maxpooling. The pooled images are then converted into a continuous vector through flattening. The 2D array is converted into a one-dimensional single vector. A dense function is added which creates a fully connected layer and to this layer we are going to connect the set of nodes obtained after the flattening step. These nodes will act as an input layer to these fully connected Layers. The final layer is the output layer which contains only one node as it is binary classification. Sigmoid activation function is used for the final layer. The model was trained for 10 epochs.

First Convolutional layer: filters =32, shape of the filter = (5,5), input= (64,64,1), activation=relu

Output shape = (None, 60, 60, 32)

Max Pooling: pool size= (2,2), strides= (1,1) Output = (None, 30, 30, 32)

Second convolutional Layer: Conv2D Layer filters =32, shape= (5,5), input= (64,64,1), activation=relu

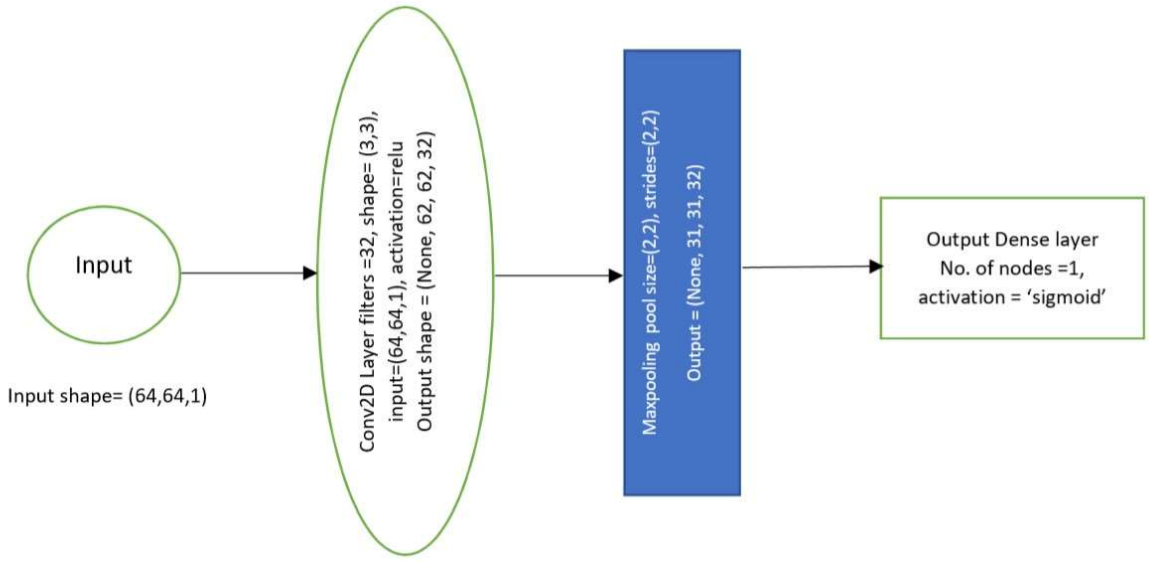
Output shape = (None, 60, 60, 32)

Max pooling: pool size= (2,2), strides= (1,1) Output = (None, 30, 30, 32)

Dense Layer: Units =128 activation = 'relu'

Output Dense Layer: Number of output nodes = 1, activation = 'sigmoid'

Williams CNN model:



The input is given to a 2D convolution layer with number of filters = 32 and shape pf each filter = (3,3). The activation used is 'relu' rectifier function. It is then followed by a maxpooling operation with pool size = (2,2) and strides = (2,2) on the resultant features. The output layer is a dense layer with one output node and sigmoid activation function as it is binary classification. The model was trained for 5 epochs. Training for more than 5 epochs resulted in overfitting and having very less to almost zero F1 on the test set.

First Convolutional layer: filters =32, shape of the filter = (3,3), input= (64,64,1), activation=relu Output shape = (None, 61, 61, 32)

Max Pooling: pool size= (2,2), strides= (1,1) Output = (None, 31, 31, 32)

Output Dense Layer: Number of output nodes = 1, activation = 'sigmoid'

To compile: Optimizer = 'Adam', loss function = 'binary cross entropy'

F1 scores were much improved when using neural networks than using SVM or KNN classifier to develop a model.

Dataset	Train F1	Test f1
Bush	0.98778833	0.759878
Williams	0.88888888	0.625

Phase 4:

A simple form of transfer learning was implemented in this phase. The initialization of the model and its weights is done on a separate dataset. A CNN is implemented for a similar classification task. The same model is then used to fit the given dataset of Bush and Williams. using pre-trained network weights as initialisations or a fixed feature extractor helps in solving most of the problems in hand. Even with many data augmentation strategies it is difficult to achieve decent accuracy. Training these networks with millions of parameters generally tend to overfit the model. This is where transfer learning is useful.

The initial “other” model was developed on a Cats vs Dogs dataset. It was converted to a binary classification case where Dog represented the true class 1 and cat images represented the false class 0. The dataset contained 12500 samples of Dog and cat images respectively.

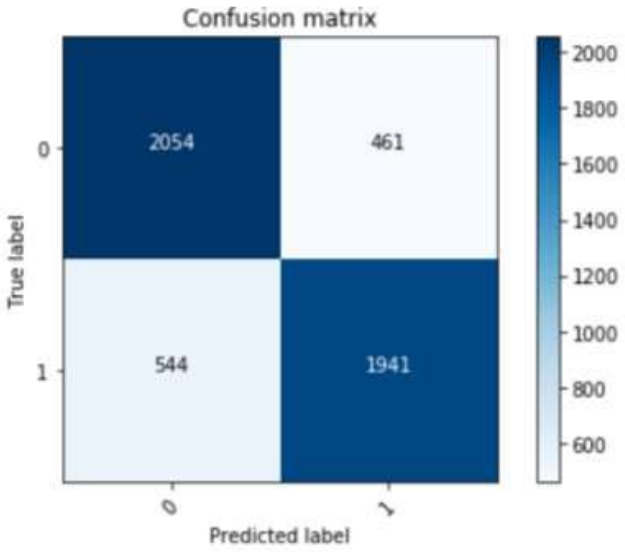
Dataset: https://www.kaggle.com/c/dogs-vs-cats/data

Pre-processing:
The given dataset had images of different sizes and in RGB color channels which was converted to 64x64 size grayscale images. These operations on the image was performed using OpenCV library.
Cv2.imread() function was used to read the images in grayscale.
Cv2.resize() function is then used to the image to (64,64) size image.
Each dog image was labelled as 1 and each cat image was labelled as 0.
The data is then split into train and test split using train_test_split() function from sklearn.
The training set contains of 20000 images and the test set consists of 5000 images.

CNN model:
Summary of the CNN model implemented:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 8)	80
max_pooling2d_1 (MaxPooling2	(None, 32, 32, 8)	0
conv2d_2 (Conv2D)	(None, 32, 32, 16)	1168
max_pooling2d_2 (MaxPooling2	(None, 16, 16, 16)	0
conv2d_3 (Conv2D)	(None, 16, 16, 32)	4640
max_pooling2d_3 (MaxPooling2	(None, 8, 8, 32)	0
conv2d_4 (Conv2D)	(None, 8, 8, 64)	18496
max_pooling2d_4 (MaxPooling2	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 100)	102500
dense_2 (Dense)	(None, 1)	101

The Cnn consists of 4 convolutional layers followed by maxpooling after each layer. There is a dense layer before the final output layer. The output layer is a dense layer with binary sigmoid activation. The CNN model was trained for 20 epochs.



Confusion matrix when the model was used to predict the values of the test set.

Precision = 0.80807
Recall = 0.7810
F1 = 0.7963

The trained model was then saved and loaded to apply to Bush and Williams dataset. The last 2 layers of the CNN was set to false in case of the Williams dataset. The following results were obtained:

Dataset	**Train f1**	**Test f1**
Bush	0.99453551	0.82191780
Williams	0.89655172	0.57142857

There was a 7% increase in the F1 score in the Bush dataset and a decrease of 4% in f1 score of Williams dataset.