Face Spoofing Detection Based on Local Ternary Label Supervision in Fully Convolutional Networks

PROJECT REPORT

Submitted By:

ANUBHAB NANDY (MIT2020043)

MTech IT (SE) 2nd Semester

UNDER THE SUPERVISION OF

Dr. Satish Singh

(A UNIVERSITY ESTABLISHED UNDER SEC.3OF UGC ACT, 1956 VIDE NOTIFICATION NO. F.9-4/99-U.3 DATED 04.08.2000 OF THE GOVT. OF INDIA)

A CENTRE OF EXCELLENCE IN INFORMATION TECHNOLOGY ESTABLISHED BY GOVT.OF INDIA

CONTENT

I. Abstract

Despite of not being much secured, face recognition has become the most popular security service medium. The inevitable result was a threat of fraud caused by face spoofing which is achieved by using images or videos of the users. To stop this, this paper suggests a method based on multi task cascaded convolutional neural network (MTCNN), Facenet and Support Vector Machine (SVM) classifier. This method detects the faces in the images using MTCNN, extracts the feature vectors using FaceNet and ultimately classify the images to be real or spoofed based on a SVM Classifier. This method is applied on the NUAA public anti face spoofing dataset and has achieved 79% accuracy on an average on test dataset. However using bagging or boosting technique with tree classifier in case of classification and training the model over bigger datasets like Celeb-A can improve the accuracy of the models' prediction.

Keywords: FaceNet, Multi Task Cascaded Convolutional Neural Network, Classifier, Support Vector Machine, Face Spoofing, Bagging, Boosting, Accuracy, Tree Classifier

II. Introduction

Face recognition is not a much secured way of ensuring a gateway security. But with the introduction of smartphone and face camera it has become a very popular and easy way of providing security service. So unlike other much secured security media like iris recognition or fingerprint recognition face recognition is easier target of frauds.

Some of the different ways the fraud can breach into a face recognition based security are:

a. Using the image or video of the person and presenting or playing the video in front of the camera to spoof someone's facial identity.
b. Using the 3d masking technology to imitate someone's facial features.
c. Using tempered photograph or video to get into the system.

There has been previously many approaches applied to solve this problem. W. Sun, Y. Song, C. Chen, J. Huang and A. C. Kot [1] worked on the CASIA and CELEB-A dataset for spoofing detection. They have used ternary label classifier. This classifier network focussed on local pixels and it was created in the style of VGG-16 and was trained from scratch by them. They used a fully convolutional network for this. In this approach the local level pixels were taken into consideration but on the other hand the self-made FCN is very time consuming to train.

A.Pinto, S. Goldenstein, A. Ferreira, T. Carvalho, H. Pedrini and A. Rocha [2] used Shape for Shading Algorithm before using a shallow neural network to classify the data. They worked on CASIA, REPLAY ATTACK datasets mainly. The albino, depth and reflectance information of the images are taken into consideration. The Model used in this approach is shallow and time efficient compared to previous one but this model gives little attention to pixel level to analysis.

D. C. Garcia and R. L. de Queiroz [3], mainly focussed on moire pattern detection. They worked on PRINT ATTACK and REPLAY ATTACK datasets. Moire patterns are very commonly found in spoofed images. This paper tries to solve spoofing detection using moire pattern detection. This approach is a very simple approach using Fourier Transform and band pass filter to determine whether moire pattern is present or not. But in this approach if the Moire pattern is not found then the image will be classified into real image which may not be correct always.

Fully Connected layers have been used by many researchers in their works for spoofing attack detection. A. George and S. Marcel [4] have used pixel level binary supervision along with FCN model to detect presentation attack over REPLAY ATTACK dataset.

Y. Liu, A. Jourabloo, and X. Liu [5] have used CNN-RNN model to classify the images into spoof or real.

J. Yang, Z. Lei, D. Yi, and S. Z. Li [6] however used a different approach. They have used domain based training which trains the model subject wise to have a better analysis on the individual subject images.

In all these methods the spoofed images or videos always have some of the evidences in the presented media which can be detected by a trained model. Our goal in this project is to detect those facial data which changes due to spoofing. For example, any spoofed image will always contain some Moire pattern which can be detected using Local Binary pattern (LBP) recogniser.

So the local pixel data always play an important role in this method. So in this project our main goal is to analyse local level pixels and determine the unimportant pixels from important pixels and ultimately considering the important pixels for further analysis. To do that normally in this project background to foreground separation using face detection model is used. Then those pixels are further analysed using already established image recogniser model to extract some important feature vectors. And finally a classifier is used over these feature vectors to classify the images into either real or spoofed.

The problem in this project is to detect spoofed images from real images using Fully Connected Layers and analyse the pixel level data to eliminate unimportant pixels. Local level filters will be used to separate the foreground from the background and then an already trained fully connected layer will be used to extract feature vectors from the face images. Now these vectors are finally classified to be spoofed or real using a Classifier.

III. Objective

There are mainly three objectives of this project which can be depicted as:

1. <u>To analyse the images on pixel level and discard the pixels which are unnecessary</u>:-
   This is the first objective of the project. In this step for face spoofing it is important that only relevant pixels are chosen for making the further analysis easier. The way this can be achieved is by separating foreground from background. In this example detecting the face and concentrating over the face pixels can be taken as an important consideration. So for spoofed images the face detection will not be done properly in most of the cases which will ensure certain distinction between the real image and spoofed images because in real images face can be detected very easily.
2. <u>To use a pre-trained image based model to identify the features present in the images</u>:-
   To extract the image features a pre-trained model can be used rather than using a user defined model. In many of the cases of the previous papers where face spoofing was addressed, user defined models were used. The main drawback of that is training over a moderate face image dataset takes a huge amount of time. So using a pre-trained model takes comparatively way less time than that. If a correct model can be chosen for face image feature recognition, even in low time high accuracy can be achieved.
3. <u>Using a classfier to classify into real or spoofed images and calculating accuracy</u>:-
   Any standard classifier needs to be used in this step to predict the output over the test data and calculate the accuracy. With hit and trial it needs to be determined which classifier works best.

IV. Methodology

The proposed model contains mainly 4 different steps altogether. Which can be shown with this following diagram:

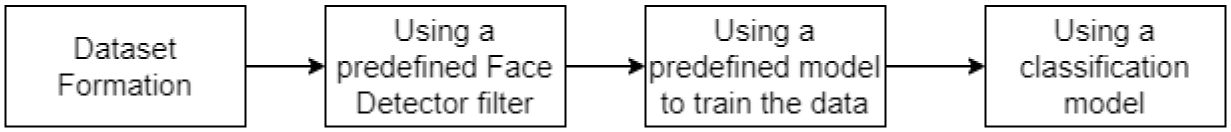

Dataset Formation
Using a predefined Face Detector filter
Using a predefined model to train the data
Using a classification model

Figure 1:- Simple Block Diagram of the steps applied in the project

a. **Dataset formation:-**

The way the dataset is actually available is different from the dataset on which the project is done. The dataset is present in the form of only two folders original and spoof and it contains the different individuals' images in separate folders. Following the instructions of text files which contains the details of the images which needs to be present in the training or test dataset, a simple python script was written and run to actually develop the dataset.

b. **Using a predefined Face Detector Filter:-**

A face recogniser model is used to mainly identify the face from a photo. Normally it is used to identify the face from an image using a box. Two different Face Detectors are used in this project. Which are,

1. HAAR Cascade:-

Unlike other very strong deep learning based face recogniser, HAAR Cascade mainly uses a primitive idea of edge detection using black and white horizontal vertical filters and edge detection.

The way it works is first of all the image needs to be converted into grey and then the edge detection matrices work on the images to detect edges on that black and white image. Now before applying all feature matrices available it is determined which of the features are needed for the image to be used by boosting the model and once the edges are detected these edges are used to isolate the face from the background.

A simple working process of HAAR Cascade would look like:

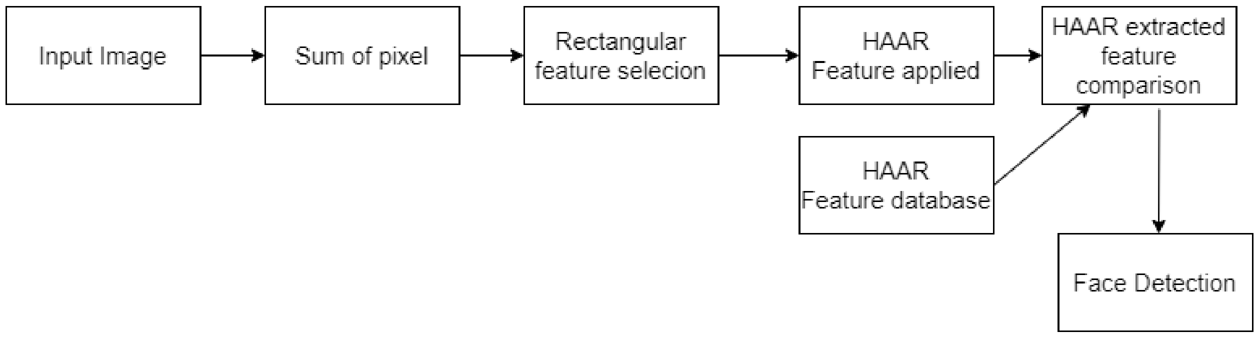


Figure 2:- This is a simple architecture following which HAAR Cascade works.

2. Multi-task Cascaded Convolutional Networks (MTCNN):-

MTCNN is a CNN based model having on total 3 CNN models in it which are namely P-Net, R-Net and O-Net. The work flow of the whole model can be subdivided in some parts.

In first stage (P-Net) the image pyramid is created using different scaling of images and then over the image pyramid filters of different shapes are used to find traces of faces from different scaled images. Now all these possibly detected faces are boxed up and along with that certain confidence of every detection is as added.

In stage two (R-Net) these detected possible faces are padded if necessary. Bounded boxes are re-evaluated and many of them are discarded.

Finally in the last stage (O-Net) all the confidence and the detected boxes are reanalysed and final detected face is published.

c. Using a predefined model to train the data:-

Two methodologies are mainly used in this step.

1. Using predefined keras imagenet models with transfer learning to train over the dataset:-

   Different predefined and pre-trained models are used with the transfer learning approach in this method.

   i. VGG-16 (Contains 16 layers, trained over ImageNet dataset)

   ii. ResNet50 (Contains over 50 layers, trained over ImageNEt dataset)

   Mainly these two models are used to train over the images. But as these models are trained over a big dataset called imagenet and already very good at image recognition, the weights of the lower levels of both these are kept frozen. Only 5-6 top layers are used in training. As these were already trained models on ImageNet where there were 1000 classes in that dataset the top most layer is not included while importing these models.

   In this way these two pre-trained models are used to train over the dataset so that it will identify the necessary features and extract them for classification.

2. Using Keras FaceNet model to extract feature vectors from the train and test images:-

FaceNet is a model used for embedding of images rather than training and classifying over like normal CNN models. FaceNet mainly tries to extract the most meaningful feature vector depending on embedding of images. These feature vectors are later used in classification. FaceNet is a very well-known and popular model in different face recognition application. FaceNet takes input of size (160, 160, 3) in RGB and the output is a set of 128 feature vectors which is used for further classification.

FaceNet works on L2 distance between similar and different images. It decreases the distance between similar images and increases the distance between dis-similar images. That's how they form these feature vectors. A very rough architecture of FaceNet will look like:-

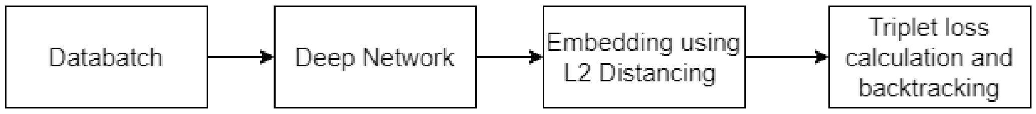

Databatch
Deep Network
Embedding using L2 Distancing
Triplet loss calculation and backtracking

Figure 3:- FaceNet architecture in blocks

d. <u>Using a classification model</u>:-

1. <u>For pre-trained imagenet models</u>:-

   Pre-trained models are trained using the train dataset and the trained model is able to identify the features of the test dataset. I used a classifier layer with softmax activation over the already existing VGG-16 or ResNet50 and trained the whole model over train dataset keeping only the top 5-6 layers trainable.

2. <u>For FaceNet extracted features</u>:-

   FaceNet extracted features contain 128 feature set, this can be classified using a standard classification model. In this project, Support Vector Machine and tree based classifier have been used to classify the vectors into two of the classes which are spoof and original.

V. Experimental Setup

1. <u>Dataset</u>:-

In this project a dataset called NUAA is used. It is a publicly available dataset containing 15 sets of images of 15 different people which are real and 16 sets of images of 16 different people which are spoofed. On total almost 11000 images are present in the dataset. These dataset is subdivided into the train and test dataset following the train and test image text files which are provided with the dataset. Almost 4000 data are present in the train and 7000 data are present in the test set. The size of test dataset is set to be higher than train dataset to test for a better performance check of the proposed model whether it will work better for newer images or not.

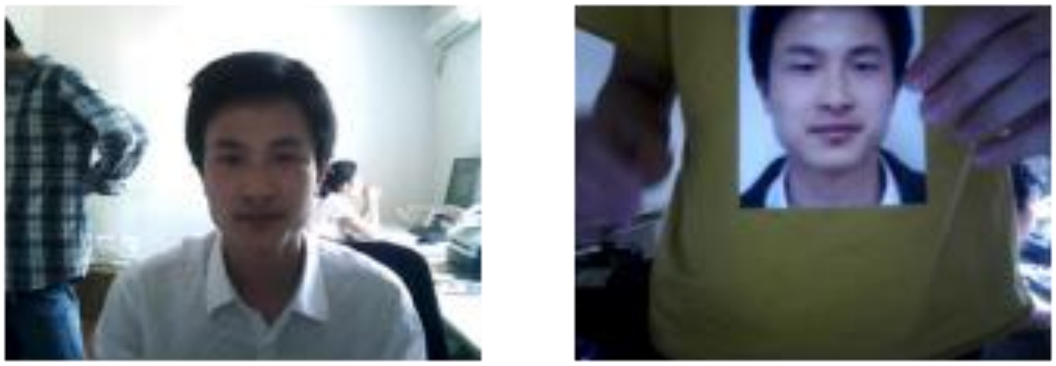


Figure 4:- the left image is a sample original image and the right one is a sample spoofed image.

http://parnec.nuaa.edu.cn/_upload/tpl/02/db/731/template731/pages/xtan/NUAAImposterDB_download.html

2. Resources Used:-
    a. Google Collab (GPU enabled) is used as platform to train and test the model.
    b. Python with Tensorflow and keras are used as the main bulding blocks of the project.
    c. Numpy and Pandas libraries are used for data manipulation.
    d. The FaceNet and MTCNN has to be installed separately in the colab environment.
    e. Python PIL library is used for image manipulation.
    f. For classifier model related works sklearn models are used.
    g. The dataset is imported directly from google drive mounting it on collab environment.
    h. The filtered and detected faces as well as embedded features vectors are saved separately once trained so that multiple training would not be necessary as these operations are time consuming.

    i. Also the trained models along with checkpoints are saved for easier testing.

3. <u>Measurement Standards</u>:-

The main measuring parameter of this project is accuracy with which the test image dataset can be identified as real or spoof by the model.

The accuracy of any classifier model has the following formula:

$$Accuracy=\frac{Number\ of\ True\ postives+Number\ of\ True\ Negatives}{Total\ sample\ data}$$

Equation 1:- Formula of Accuracy of a binary classifier

4. <u>Alternative solutions</u>:-
    a. With better resources like google collab pro or cloud services the filtering and training can be done in lesser time.
    b. Also GPU enabled Machines can be used to enhance the speed of the training.
    c. The system can be further enhanced using some more vast datasets like CELAB-A(containing almost 3 lakhs of celebrity images) or video datasets like CASIA.

VI. Results and Discussion

a. <u>Selecting the face detector</u>:-
   HAAR Cascade is a very simple and fast acting model but in case of our dataset, it was unable to detect the faces from some of the original images and for some of them the detection was wrong.
   For normal original image HAAR classifier acted like this:

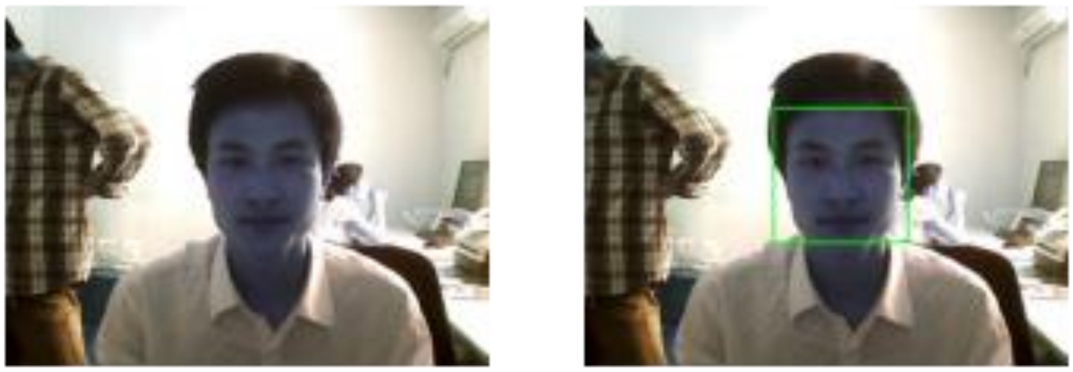


Figure 5: using HAAR face detection face recognition of an original image. Original (left), face recognised (right)

But for the following image the detector was unable to detect the face

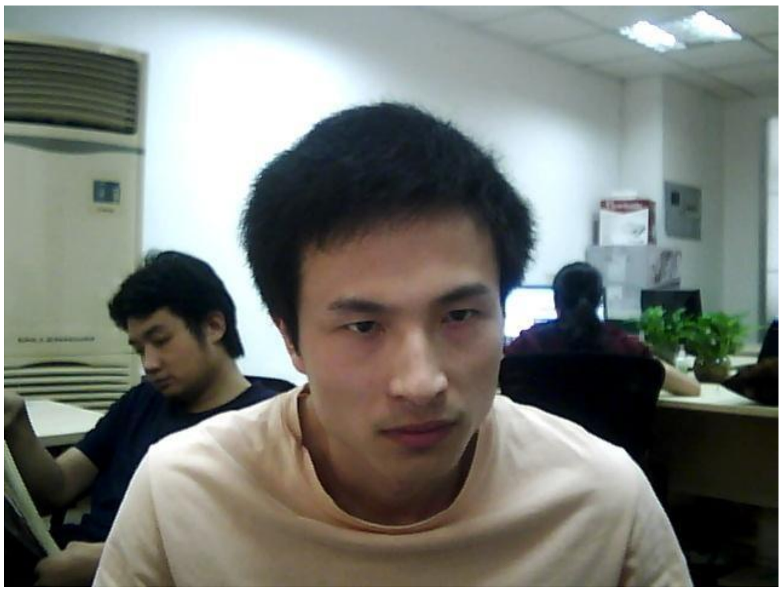


Figure 6:- Using HAAR Cascade face could not be detected from this image.

Because of these problems MTCNN, a more powerful face detector is used. MTCNN detector acts on the dataset like following:

The problem is approached in different methods which are chronologically mentioned along with the results:

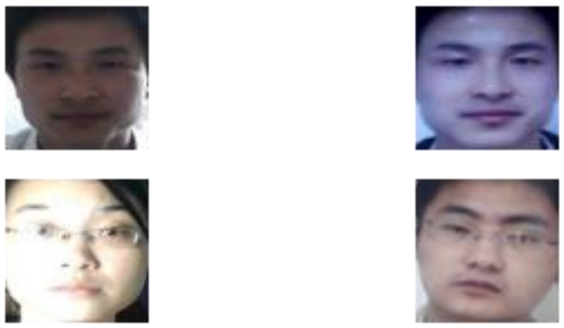


Figure 7:- Face recognition using MTCNN face detection. These 4 images are representative of 4 parts of the dataset. The upper row is from train dataset and the top left one is original and the right one is spoofed. Similarly for the bottom row is taken from test dataset. The bottom left one is original and right one is spoofed.

The above example shows that MTCNN is capable of recognising faces very efficiently as representative images from each class is presented.

b. Selecting the training model:-

Different models are used in the projects and they are accepted or declined based on the accuracy score of the test dataset. These are described in the following table.

Methods used	**Avg Train accuracy**	**Avg Test Accuracy**
VGG-16	83%	58%
ResNet50	85%	62%
ResNet50+Transfer Learning	93%	66%
ResNet50+MTCNN+Transfer Learning	95%	73%
MTCNN+FACENET+SVM classifier	**97%**	**79%**
MTCNN+FACENET+TreeClassifier	96%	75%

Table 1:- A comparative study of different applied models based on Accuracy

i. VGG16 and ResNet50:- VGG16 and ResNet50 are pre-trained models with imageNet, because of this the models do not work good with this new dataset of ours. Only the classifier on the top was trained in this process.

ii. ResNet50 + Transfer Learning:- ResNet50 is considerably better image classifier than VGG16 so further work has been done only on ResNet50. Transfer Learning is a technique where the upper layers of an already trained model is trained with new dataset to make the model recognize the new features of the dataset. Using this method the result got better but because of using a very deep pre-trained model there are many layers which contain the previous weights which was kept frozen during the train. This causes an imbalance in the training the new dataset because of which the results were not quite good.

iii. ResNet50 + MTCNN + Transfer Learning:- MTCNN is a face detector which in the first step detects the faces from the images and then the ResNet and transfer learning is applied on the faces only. In this was the unimportant pixels get discarded by the filter and only over the face pixels the model is applied which improves the accuracy. But as the ResNet is trained over ImageNet dataset which contains animal images face detection was not that effective for ResNet50, only 73% test accuracy was achieved after all these pre-processing.

iv. **FaceNet+MTCNN+Classifier:**- Using FaceNet to extract the embedded feature vectors was a huge improvement in terms of accuracy. For classifier network mainly two different classifiers were used to classify the feature vectors into the two classes of original and spoof respectively. Between TreeClassifier and SVM **SVM** performed better.

So the final Model is proposed to be the one with highest test accuracy with MTCNN, FaceNet and SVM classifier which gives results like following on the dataset:

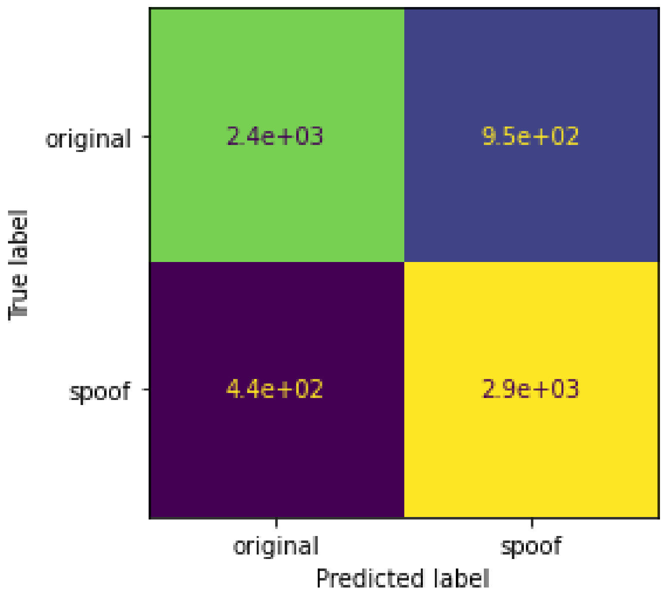
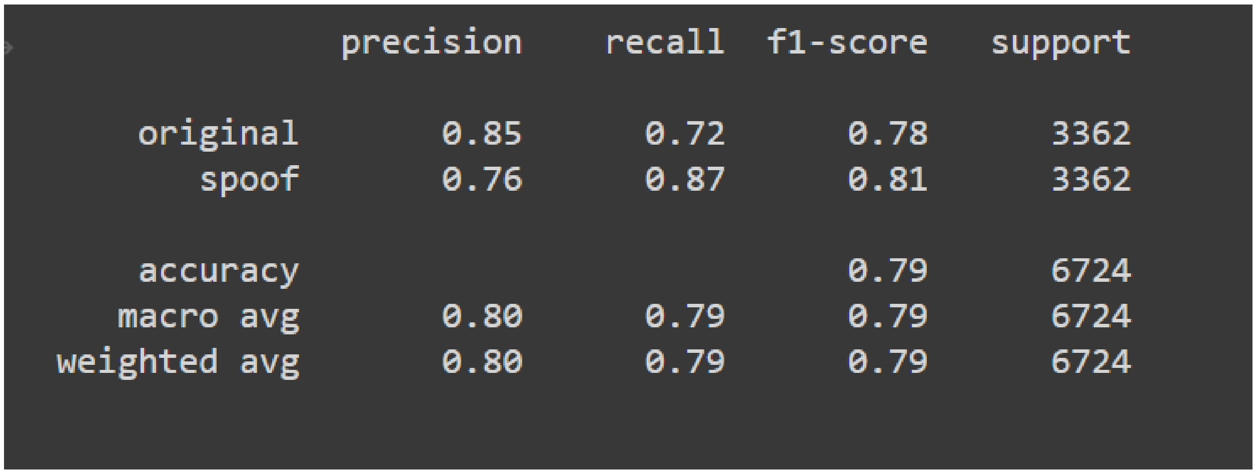


Figure 8:- Confusion matrix of proposed model



```
              precision    recall  f1-score   support

    original       0.85      0.72      0.78      3362
       spoof       0.76      0.87      0.81      3362

    accuracy                           0.79      6724
   macro avg       0.80      0.79      0.79      6724
weighted avg       0.80      0.79      0.79      6724
```

Figure 9:- Classification Report of the proposed model

VII. Conclusion

With proper resources used the model should take lesser time to train compared to the previously worked on models. This model is trained only on the NUAA dataset which is a very standard dataset, but training the model with more data and comparing inter dataset accuracy will give more insight of the model's performance. Using bigger datasets like Celeb-A or famous video spoofing dataset like CASIA can improve the models predicting power. Also better classification models at the last level can be used, bagging and boosting of tree classifiers can improve the accuracy drastically.

VII. References

[1] W. Sun, Y. Song, C. Chen, J. Huang and A. C. Kot, "Face Spoofing Detection Based on Local Ternary Label Supervision in Fully Convolutional Networks," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3181-3196, 2020

[2] A. Pinto, S. Goldenstein, A. Ferreira, T. Carvalho, H. Pedrini and A. Rocha, "Leveraging Shape, Reflectance and Albedo From Shading for Face Presentation Attack Detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3347-3358, 2020.

[3] D. C. Garcia and R. L. de Queiroz, "Face-spoofing 2D-detection based on moiré-pattern analysis," *IEEE Trans. Inf. Forensics Security*, vol. 10,no. 4, pp. 778–786, Mar. 2015.

[4] A. George and S. Marcel, "Deep pixel-wise binary supervision for face presentation attack detection," in *Proc. Int. Conf. Biometrics (ICB)*, Jun. 2019.

[5] Y. Liu, A. Jourabloo, and X. Liu, "Learning deep models for face antispoofing: Binary or auxiliary supervision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 389–398.

[6] J. Yang, Z. Lei, D. Yi, and S. Z. Li, "Person-specific face antispoofing with subject domain adaptation," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 4, pp. 797–809, Apr. 2015.

VIII. Appendix

a. Using MTCNN face recognition with the following function :

```
def face(fil):
  final_size=(160, 160)
  img = Image.open(fil)
  img = img.convert('RGB')
  arr = asarray(img)
  ret = MTCNN().detect_faces(arr)
  x1, y1, width, height = ret[0]['box']

  if int(height)==0 and int(width)==0:
    return arr

  x1, y1 = abs(x1), abs(y1)
  x2, y2 = x1 + width, y1 + height
  extracted_face = arr[y1:y2, x1:x2]
  img = Image.fromarray(extracted_face)
  img = img.resize(final_size)
  ans = asarray(img)
  return ans
```

b. Using following function to create face extracted dataset X :

```
def make_X(dir):
  X = list()
  path=glob.glob(dir)
  for src in path:
    X.append(face(src))
  return X
```

c. To form the X as well as labels of each train and test the following function is used :

```
def load(dir):
  X=list()
  y=list()
  for subdr in listdir(dir):
    path = dir + subdr +"/*jpg"
    f = make_X(path)
    label = [subdr for _ in range(len(f))]
    X.extend(f)
    y.extend(label)
    return asarray(X), asarray(y)
```

Now for both the train and test the face detected dataset is created by the following code:

```
Xtrain, ytrain = load('/content/drive/MyDrive/data/train/')
Xtest, ytest = load('/content/drive/MyDrive/data/test/')
```

d. As all the operations are time expensive after the face detection is done I saved the data in form of zip

```
np.savez_compressed('/content/drive/MyDrive/face_data_train.npz',  Xtrain,
 ytrain)
np.savez_compressed('/content/drive/MyDrive/face_data_test.npz',  Xtest, y
test)
```

e. Once the Face detection is over the following function is used for the embedded feature extraction

```
def embed(model, f):
  f = f.astype('float32')
  m, s = f.mean(), f.std()
  f = (f - m) / s
  samples = np.expand_dims(f, axis=0)
  y = model.predict(samples)
  return y[0]
```

f. Now the from the face data the embedded feature vectors are extracted both by train and test dataset

```
model = load_model('/content/drive/MyDrive/facenet_keras.h5')

data = np.load('/content/drive/MyDrive/face_data_train.npz')
Xtrain, ytrain= data['arr_0'], data['arr_1']

nXTrain = list()
for face in trainX:
  embeddings = embed(model, face)
  nXTrain.append(embeddings)
nXTrain = np.asarray(nXTrain)

data = np.load('/content/drive/MyDrive/face_data_test.npz')
testX, testy = data['arr_0'], data['arr_1']

nXTest = list()
for face in trainX:
  embeddings = embed(model, face)
  nXTest.append(embeddings)
nXTest = np.asarray(nXTest)
```

g. Finally saving these again as a separate zip file

```
np.savez_compressed('/content/drive/MyDrive/face_test_embed.npz',nXTest, y
test)
np.savez_compressed('/content/drive/MyDrive/face_train_embed.npz',nXTrain,
 ytrain)
```

h. The following code is for the SVM classifier :
    a. Data pre-processing

```
data = np.load('/content/drive/MyDrive/face_train_embed.npz')
Xtrain, ytrain = data['arr_0'], data['arr_1']
data = np.load('/content/drive/MyDrive/face_test_embed.npz')
Xtest, ytest = data['arr_0'], data['arr_1']

n = Normalizer(norm='l2')
trainX = n.transform(Xtrain)
testX = n.transform(Xtest)

out = LabelEncoder()
out.fit(ytrain)
trainy = out.transform(ytrain)
testy = out.transform(ytest)
```

    b. Model training

```
model = SVC(kernel='linear')
model.fit(trainX, trainy)
```

c. Testing and results

```
yhat_test = model.predict(testX)
yhat_train = model.predict(trainX)
score_train = accuracy_score(trainy, yhat_train)
score_test = accuracy_score(testy, yhat_test)
print('Accuracy: train=%.3f, test=%.3f' % (score_train*100, score_test*100
))

metrics.plot_confusion_matrix(model,testX,testy)



print(metrics.classification_report(testy,yhat_test , target_names=["origi
nal","spoof"]))
```