Video-based Face recognition

Nurdaulet Zhuzbay, Zhamilya Saparova, Alisher Toregali

I. Abstract

This paper presents a system for automatic face recognition in video streams. The system is composed of four main parts: face detection, liveness detection, face tracking, and face recognition. In the liveness detection part, we employ two different methods to determine whether a face in a video frame is real or fake, which helps prevent spoofing attacks on the face recognition system. In the face tracking part, we track the position and movement of faces across multiple video frames. This helps the system maintain consistent face recognition even when faces move or change in appearance. In the face recognition part, we match detected faces against a database of known faces using deep learning models that learn features. The system outputs the identity of the detected face if it is recognized as a match in the database.

Keywords: face recognition, video-based, deep learning, face detection, liveness detection, face tracking, spoofing, security.

II. Introduction

NOWADAYS, face recognition is one of the most actively studied problems in computer vision and biometrics. Especially, video-based face recognition offers a plethora of potential applications in the real-world including visual surveillance, access control, and video content analysis. Recognizing faces from the video is generally more challenging than still image-based face recognition. This is because of the following reasons: a much larger amount of data to be processed and significant intra/inter-class variations caused by motion blur, low video quality, occlusion, frequent scene changes, and unconstrained acquisition conditions.

This project develops an automatic system that allows users to input the videos taken from surveillance cameras to efficiently detect and recognize human faces that appeared in the scene. The system should also enable the “liveness detection”, i.e., realize if the face is from the real person (not taken from the picture).

III. Literature Review

Video-based face detection has been a topic of interest in computer vision for several years now. With the increase in popularity of video content on social media, the need for accurate and efficient face detection algorithms has become more important than ever.

In recent years, deep learning-based approaches have gained popularity in the field of face detection. One of the most successful methods is the Region-based Convolutional Neural Network (R-CNN) [7]. R-CNN uses a combination of object proposals and a deep neural network to detect faces. It has been shown to outperform traditional methods like Viola-Jones in terms of accuracy, especially under challenging conditions like low resolution and occlusion [8].

Another popular approach researched by Garg et al. is YOLO (You Only Look Once) [9]. It is a highly efficient and accurate method for detecting faces in real-time video streams. YOLO is a deep learning algorithm that uses a single neural network to detect objects, including faces, in images and videos. The YOLO algorithm operates by dividing an image into a grid and predicting bounding boxes and class probabilities for each grid cell. This approach allows YOLO to quickly and accurately detect objects, including faces, in a video stream.

In video-based face detection, face tracking is an important task that involves detecting and tracking faces over time. One method for face tracking is the Kanade-Lucas-Tomasi (KLT) algorithm, which uses optical flow to track faces [10]. Another method is the Mean-Shift algorithm, which uses color histograms to track faces. Face tracking is a challenging task due to variations in lighting, pose, and occlusion, and therefore requires robust algorithms.

As deep learning architectures require millions of training samples and access to powerful computational resources like Graphical Processing Units (GPUs) in order to optimize millions of parameters and learn the multi-stage weights from scratch, Ghazi and Ekenel [27] in their paper propose transfer learning method for face recognition. According to their paper, transfer learning can be applied in two ways with respect to the volume of dataset. The first method involves adjusting the weights of the pre-trained network. This approach is only advised for sufficiently large datasets [29]. The second strategy involves directly applying learnt weights to the targeted issue in order to extract and subsequently categorize features. This method works exceptionally well when there are few classes or a tiny fresh dataset [30]. Authors concluded that comparing the VGGFace model to the Lightened CNN model, the former has demonstrated a greater transferability. This may be explained by its more complex construction, which produces a more abstract representation. Moreover, Qawaqneh et al. highlited the efficiency of VGGFace over other CNN based models. They observed that models VGGFace outperforms the models for age estimation task, which demonstrates that a CNN model can be used for age estimation to increase performance. Also, they state that the usage of pre-trained CNN models can resolve over-fitting issue on a large database for face recognition task [28].

Face liveness detection is the process of determining whether a facial biometric sample is live or spoofed. The importance of liveness detection has increased significantly with the widespread use of facial recognition systems in various applications. One of the traditional techniques for liveness detection is based on liveness cues, eye-blinking [11],

[12], [13], face and head movement [14], [15] (e.g., nodding and smiling), gaze tracking. However, these liveness cues should be obtained from the continuous video streams, which is computationally inefficient and can be easily attacked. Most of the recent papers treat the problem of anti-spoofing as a binary classification problem (e.g., '0' for live while '1' for spoofing faces, or vice versa) thus supervised by a simple binary cross-entropy loss [16], [17], [18]. Other works use deep pixel-wise information for these types of challenges [19]. Other work [20] proposes a method for face anti-spoofing that uses semi-supervised learning to improve the generalizability of the model. The proposed method uses a combination of a supervised and semi-supervised learning approach. The supervised component is trained on a labeled dataset, while the semi-supervised component is trained on both labeled and unlabeled data. The semi-supervised component uses a consistency loss function to encourage the model to produce similar outputs for the same input, even when the input is perturbed.

In recent years, face recognition using FaceNet model became on of the most popular techniques used. William et. al shows that that FaceNet has better results than CASIA-Webface and VGGFace2 models on public datasets such as YALE, JAFFE, AT T, Georgia Tech, and Essex [21]. In another research, which was conducted by Cahyono et. al, FaceNet was found to be better than the architectural model of Openface [22]. Moreover, FaceNet model in combination with SVM has reached an accuracy of 100%.

IV. Datasets

For the face liveness detection part "Large Crowdcollected Face Anti-Spoofing Dataset" was used. It contains images in "png" format for training, validation and testing. The images are pre-processed, and faces are extracted. For the training data it has 8299 images, 1223 are real and 7066 are fake images. For the validation data it has 2948 images, 405 are real and 2543 are fake images. For the test data it has 7580 images, 314 are real and 7266 are fake images. All the images are collected from Youtube, Amazon Mechanical Turk and Yandex Toloka web services [23].

YouTube Faces dataset is a large-scale benchmark dataset that contains over 3,000 videos of 1,595 individuals from YouTube, with each individual having at least one video. The videos are of various lengths, ranging from 48 frames to 6,070 frames, and contain different variations in pose, lighting, and facial expression. The dataset was created to facilitate research on video-based face recognition and facial expression analysis.

The problem statement for a video-based face recognition project using the YouTube Faces dataset is to develop an algorithm that can accurately identify individuals across different videos and video frames based on their facial features. This involves several challenges, including dealing with variations in lighting, pose, and expression, as well as identifying the same person across different videos that may have significant visual differences.

The goal of our project would be to improve the accuracy and robustness of video-based face recognition algorithms, which have numerous applications in security, surveillance, and personalization. By using the YouTube Faces dataset, researchers can develop and evaluate their algorithms on a large and diverse dataset that represents real-world challenges in video-based face recognition.

The YouTube Faces dataset is annotated with several types of information that can be useful for video-based face recognition and facial expression analysis research. The annotations include:

- Identity labels: Each video in the dataset is labeled with the identity of the person appearing in the video. The identities are represented by unique integer identifiers.
- Frame-level bounding boxes: Each frame in a video is annotated with a bounding box that tightly encloses the face of the person appearing in the frame. The bounding boxes are provided in the form of (x,y) coordinates of the top-left corner of the bounding box and its width and height.
- Pose labels: The dataset also includes pose labels for a subset of the videos. The pose labels indicate the orientation of the head.

The annotations in the YouTube Faces dataset enable researchers to develop and evaluate algorithms for various tasks, such as face detection, face tracking, face recognition, and facial expression analysis. The identity labels and frame-level bounding boxes, in particular, are useful for developing face recognition algorithms that can identify individuals across videos with varying lighting, pose, and expression. The pose and expression labels, on the other hand, can be used for developing algorithms that can recognize different facial expressions or head poses. The quality scores can help researchers filter out videos with poor annotations, which can improve the accuracy of their algorithms.

V. Methodology

Four different tasks were completed in this project: face detection, liveness detection method, face tracking and face recognition.

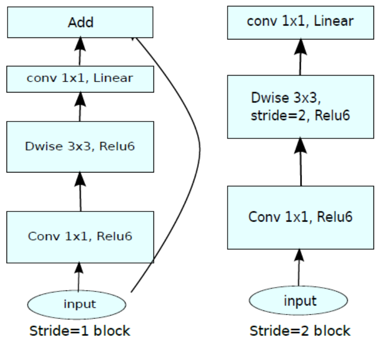
A. Face Liveness Detection

Face liveness detection is the process of identifying whether the given image is a real image or a printed, 3-D mask, i.e. fake. For this task "Large Crowdcollected Facial Anti-Spoofing Dataset" was used, because of its lightweight and public availability. Since the data was pre-processed, the face detection was not implemented for this part. This task was implemented using plain CNN and MobileNetV2 architectures.

1) Baseline: For the baseline, we implemented simple CNN based classification method in Tensorflow. The model consists of 3 2D convolution layers and 2 Max pooling layer and 2 Dense layers. All convolution layers have ReLU activation function for feature extraction. The input to the network is of size (224, 224, 3). The last 2 Dense layers have sigmoid activation function for binary classification. As an optimizer Adam was used and binary cross-entropy loss was set. The model was trained on NVIDIA GeForce 1650 GTX GPU with batch size 24 and 20 epochs.

2) Main approach: For the main approach more complex network was implemented. For feature extraction MobileNetV2 architecture was used with the pre-trained weights "ImageNet". MobileNetV2 is a great choice because it was designed to be computationally efficient and lightweight. Despite its lightweight architecture, MobileNetV2 can achieve high accuracy on image classification tasks. Its depthwise separable convolutions help to preserve spatial information while reducing computational complexity [24].

Fig. 1: MobileNetV2 architecture



After the output of the pre-trained MobileNetV2 model additional layers were added for classification. Dropout layer was added to prevent overfitting. The last fully-connected layer's activation function is sigmoid for classification. Optimizer was set as Adam and loss is binary cross entropy. Batch size is 24 with 20 epochs. Model was trained on NVIDIA GeForce 1650 GTX GPU.

B. Face Recognition

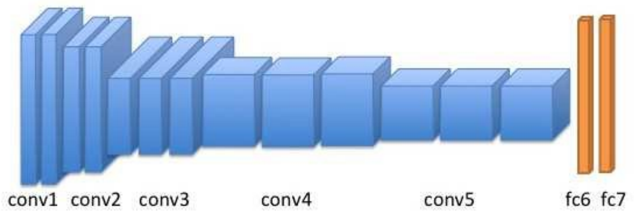
Face recognition is the process of identifying and verifying the identity of a person based on their facial features. It is a key application of computer vision. The process of face recognition in computer vision typically involves several steps. In our case with the given Youtube Faces dataset, the first step would be to detect the face bounding box and crop them from the original image frames. The MTCNN algorithm was used for the face detection. It is a popular face detection algorithm that uses deep learning techniques to detect and localize faces in images with high accuracy. There are two versions of MTCNN, one for keras library and the other is for pytorch library. As the first implementation took more time for us to process all the images, we chose the second, which is for pytorch library. We saved the output in .npz file. The parameters of MTCNN:

```
mtcnn = MTCNN(image_size=160, margin=0,
min_face_size=20,factor=0.709,
select_largest=False,
post_process=False, device=device)
```

We split the dataset into train, validation and test set with 80:10:10 ratio, respectively. Two popular methods were applied for the face recognition: VGGFace and FaceNet.

1) Baseline: We pass the train and validation set to VGFace. The baseline model for face recognition is VGGFace algorithm, which is a deep learning model for face recognition trained on over 2 million images [26, 27, 28]. The model is based on the popular VGGNet architecture, which consists of several convolutional layers and fully connected layers. The original VGGFace architecture has 26 layers [26]. We replace the seven layers, which represented the three fully connected output layers to custom layers. The newly additional layers will be taught to recognize the images from the given dataset. As we already have trained 19 layers by the VGGFace16 model, we will only train the new added layers. The input to the network is a face image, which is resized to 160x160 pixels. The first few layers of the network consist of convolutional layers with small 3 x 3 filters, followed by max-pooling layers with a 2 x 2 filter and stride of 2 . The last few layers of the network are fully connected layers, which are used to compute the identity of the face. The output of the network is a vector of size 512, which represents the identity of the face [27]. Below can be seen visual demonstration of VGGFace architecture.

Fig. 2: VGGFace architecture [28]



All the proccess has gone through GPUs (Graphical Processing Units). We set loss to sparse categorical crossentropy and evaluation metrics to accuracy. For the optimizer, we tested three different popular optimizers: Adam (Adaptive Moment Estimation), Adamax and SGD(Stochastic Gradient Descent). SGD is a simple and widely used optimization algorithm that updates the weights of a neural network based on the average gradient of the loss function with respect to the weights [31]. Coming to the Adam, it is a more advanced optimization algorithm that combines the advantages of several optimization techniques [31]. Adamax is a variant of the popular Adam optimizer, which is widely used in deep learning for updating the weights of neural networks during training. Adamax is designed to address some of the limitations of Adam, especially in situations where the gradients are very sparse or the learning rates are very high [31].

2) Main Approach: As a main approach for face recognition, we used feature extraction from an image using FaceNet model and pass train and validation data to neural networks for classification. For the sake of comparison, we used following neural networks: Support Vector Classifier (SVC) and Multi-Layer Perceptron (MLP). Dataset was divided in a same manner as it was described in a baseline approach (80:10:10).

FaceNet is a CNN model that was developed by Google researches for face recognition. It takes an image as an input to map its values into Euclidean spaces. As an output, FaceNet returns an array of 512 features, that will be used to find similar faces. An architecture of the model can be seen at figure 3 and 4. Basically, triplet loss function makes inputs that are similar closer, while different images become more distant. Face detection using MTCNN was used in order to crop the face and use as an input [21].

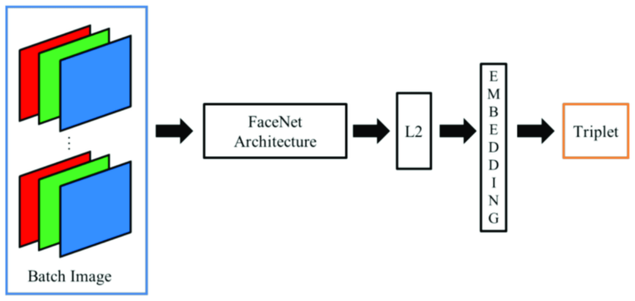


Fig. 3: FaceNet architecture

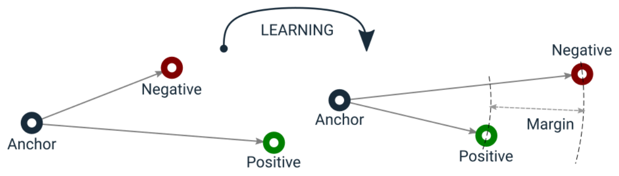


Fig. 4: Triplet Loss

For face recognition classification we use widely used neural networks: SVC and MLP. We could not try optimizing the model by training with several parameters due to the computational limitation. For SVC, we trained with two kernel types for comparison, which are 'linear' and 'rbf'. In a case of MLP, we used for parameter options for number of neurons: 10, 12, 13, 14 and 15.

C. Face Tracking

For the final task, all of the research conducted was summarized and face tracking algorithm was implemented. The methodology of face tracking consists of several steps: input preprocessing, face detection, face recognition and processing the output. Preprocessing of the video was conducted by dividing it into frames using Open Source Computer Vision Library (OpenCV). For each frame, face detection method using MTCNN was applied. By cropping the face from the image, it was used as input for the face recognition technique. Eventually, video file with a face box and name tag is provided as an output file.

VI. Evaluation Metric

To evaluate the results of this paper, accuracy score will be used by dividing the number of correct predictions by the total number of predictions made, and then multiplying by 100 to convert the result to a percentage. This is a common evaluation metric used in machine learning to measure how well a model is able to predict the correct outcome for a given task. Also, another popular metrics, confusion matrix, will be used to evaluate classification tasks. It compares the actual values of the target variable with the predicted values produced by the model, and shows how often the model makes correct or incorrect predictions.

VII. Results & Analysis

A. Face liveness Detection

1) Baseline: The CNN model was trained with batch size of 24 and 20 epochs. Although the training accuracy score was high on the test set it failed to show good results. From Fig. 5 it can be seen that the accuracy on the training set is approximately 85 percent.

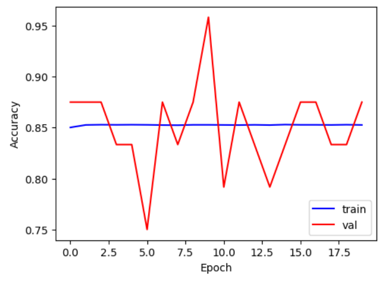


Fig. 5: Adam optimization accuracy

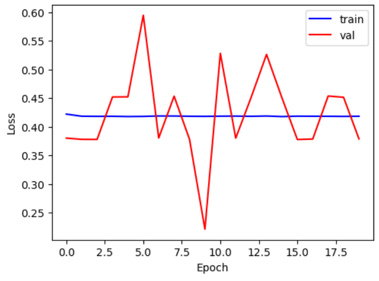


Fig. 6: Adam optimization loss

On the test set it showed precision score of 95 percent. However, it predicted all images as fake ones. It can be seen from the confusion matrix from below. It might be due to overfitting, since no regularization or dropout layers were added to this baseline model.

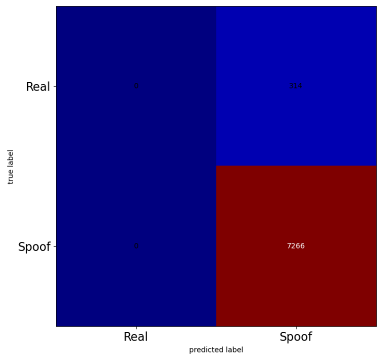


Fig. 7: Confusion matrix - baseline

2) Main approach: The main approach with MobileNetV2 architecture and Dropout layers on the other hand performed well on training and test sets. From Fig. 8 it can be seen that the model reached almost 99.5 percent accuracy and 0.5 percent loss.

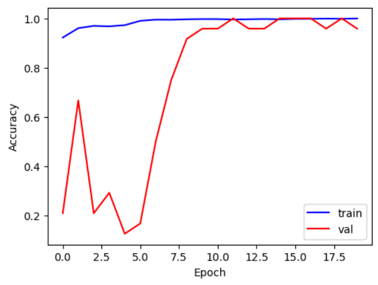


Fig. 8: Adam optimization accuracy

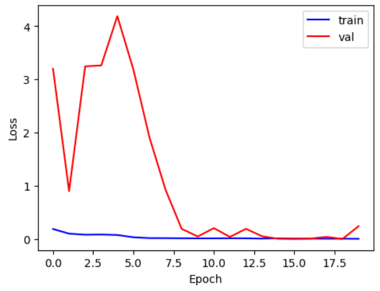


Fig. 9: Adam optimization loss

Also, from the confusion matrix from Fig. 10 it can be seen that the main approach model performed well on test set. The precision score is 99.35 percent.

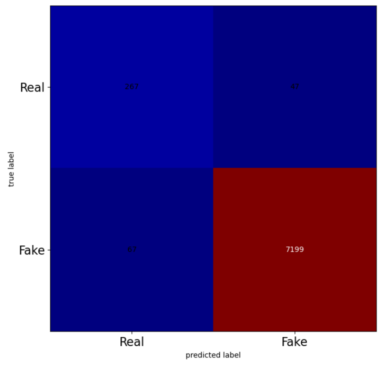


Fig. 10: Confusion matrix - MobileNetV2

Fig. 11: Adam optimization accuracy and loss

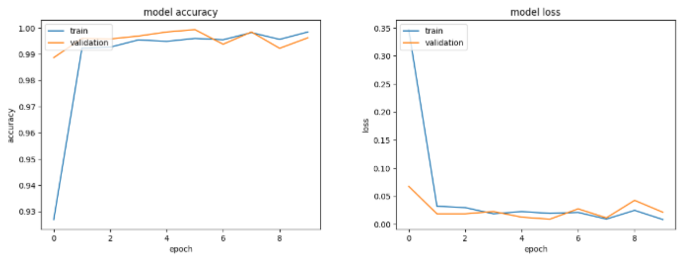


Fig. 12: SGD optimization accuracy and loss

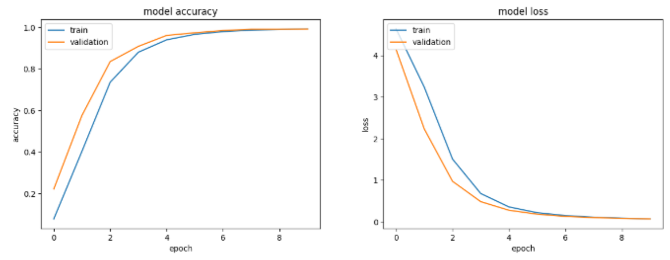
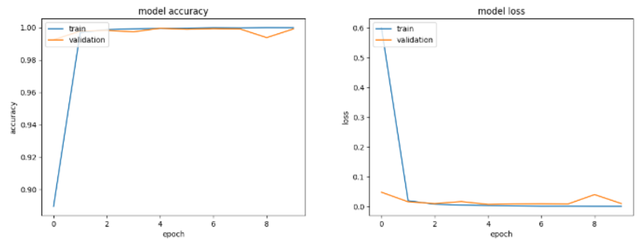


Fig. 13: Adamax optimization accuracy and loss

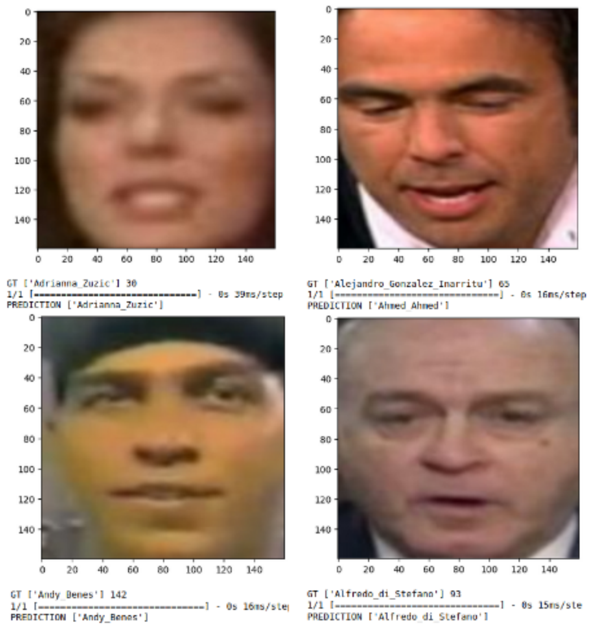


B. Face Recognition

1) Baseline: VGGFace was experimented with three different optimization techniques. On the Fig. 11, 12, 13 can be seen the demonstration of model accuracy and loss on train set and validation set using three optimizers. It can be noticed from the figures that Adam optimizer fluctuates more compared to Adamax and SGD. Also, SGD shows smoothest transition over each epoch, which means that the model took more time to learn.

The model was verified with test set. The test accuracy score for the Adam optimizer is 97.667%, for the Adamax is 98% and for the SGD is 88.289%. Overall, from the accuracy score values, it can be noticed that the Adamax optimizer shows the best performance.

Also, below can be seen examples of predictions and ground truth from the test set.



```
GT ['Adrianna_Zuzic'] 30
1/1 [==============================] - 0s 39ms/step
PREDICTION ['Adrianna_Zuzic']

GT ['Alejandro_Gonzalez_Inarritu'] 65
1/1 [==============================] - 0s 16ms/step
PREDICTION ['Ahmed_Ahmed']

GT ['Andy_Benes'] 142
1/1 [==============================] - 0s 16ms/step
PREDICTION ['Andy_Benes']

GT ['Alfredo_di_Stefano'] 93
1/1 [==============================] - 0s 15ms/ste
PREDICTION ['Alfredo_di_Stefano']
```

Fig. 14: Prediction example

2) *Main Approach:* According to the Figure 15, SVC has performed well on our dataset. This model has shown a minor overfitting, because mean train accuracy is slightly higher than the mean test result. However, it is negligible due to the insignificant difference. The mean test accuracy obtained using kernel 'rbf' during cross-validation was 96.4%, while mean train score reached 99.9%.

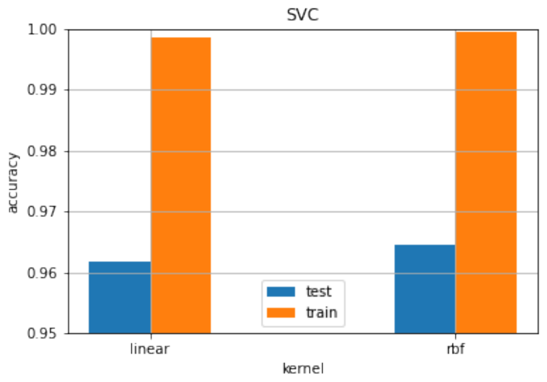


Fig. 15: Mean train and mean test accuracy values for SVC with two kernel variations

MLP has shown worse results than SVC. For all of the parameters, the best result obtained was using 15 neurons in hidden layer and adam solver, which can be seen at Figures 9 and 10 . The best mean testing score reached was about 88.1%, while the training score was found to be 99.5%. Besides lower testing score, there is a larger difference between traing and testing accuracy.

SVC has shown best results, that is why we compute its test accuracy on out test set. The results are following: train accuracy is 99.95%, while test accuracy is 99.91%.

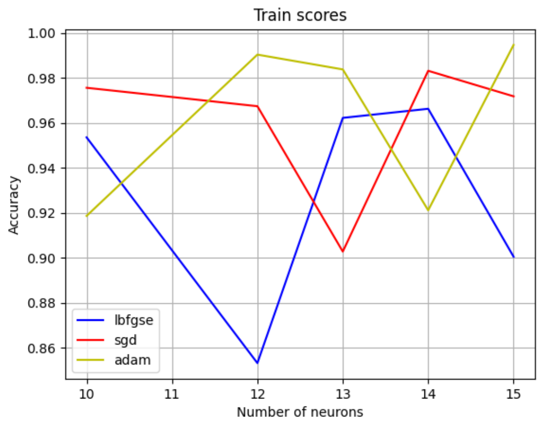


Fig. 16: Mean train accuracy values for MLP

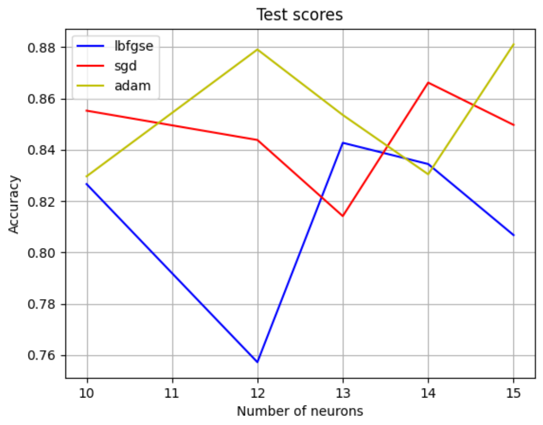


Fig. 17: Mean test accuracy values for MLP

3) *Face Tracking:* An algorithm for taking as an input the name of the folder with frames and returning an output of the video with a name label and box around the face was implemented. FaceNet model with SVC proved to be better than VGGFace for out dataset, that is why FaceNet was used for face recognition in this task. The output frame of the algorithm can be seen at the Figure 18.

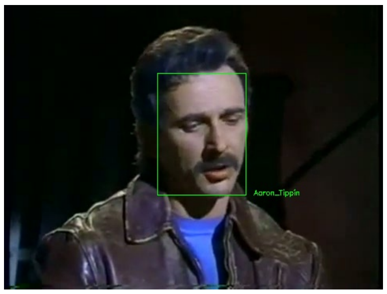


Fig. 18: Output of face tracking implementation

VIII. Error Analysis

According to the results of this paper, there are no major errors that could lower results significantly during our project.

One of the considerable limitations of our project is the dataset we used. Despite the fact it consists of a large number of people and frames from videos per each of them, it had several drawbacks. Firstly, there are some people, for whom there was only one video, which limits the diversity in training data and increases overfitting. Another problem is the fact that for some of the people, there were 2 people in the video. If someone will have only 1 video, on which there are 2 people, the model will not understand which face has to be processed. That is why we have discarded this kind of data. Also, it should be mentioned that the dataset weighted a lot and was computationally cost. Due to this problem, only approximately 10% of the whole dataset was used. Models could not train on large scale dataset and many of parameters for neural networks were neglected to include in the optimization.

IX. Future Work

For the future work, it is planned to train the algorithms on more powerful hardware to cover whole dataset. The dataset could also be manually preprocessed to get rid of untrainable data. The problem of videos with multiple people could be adapted. Also, to join all four task, face detection, liveness detection method, face tracking and face recognition, to result in one automatic face recognition system. Moreover, as all members of team did not have web camera, we could not test the algorithm on real-time based video.

X. Conclusion

This paper demonstrates video-based face recognition system, which consist of four subunits: face detection, liveness detection, face tracking, and face recognition. For the face detection, current state-of-the-art MTCNN was employed as it detects and localizes faces in images with high accuracy. For the liveness detection task, two methods were compared: CNN based model and MobileNetV2. In the comparison between two architectures, both models performed well. However, due to an unbalanced dataset, the CNN-based model incorrectly classified all images as fake, whereas MobileNetV2 produced more accurate results. For the face recognition task, also two widely known algorithms were used: VGGFace and FaceNet. We observed that training a new model from scratch is computationally expensive, thus involved transfer learning. According to obtained results, FaceNet model in combination with SVC has reached better results than with MLP classifier. Moreover, this model has also outperformed VGGFace model, by getting test accuracy 1% higher. Thus, FaceNet model in combination with SVC classification algorithm was used for face recognition task. However, this difference is not major and this issue has to be researched further. In the face tracking, the position and movement of a face within each frame of a video was detected. Video files can be captured into frames and processed by the algorithm implemented in the project. Moreover, it could modified to serve as a real-time face detector and recognizer.

XI. References

[1] Wolf, L., Hassner, T., Maoz, I. (2011, June). Face recognition in unconstrained videos with matched background similarity. In CVPR 2011 (pp. 529-534). IEEE.

[2] Kim, K., Yang, Z., Masi, I., Nevatia, R., Medioni, G. (2018, March). Face and body association for video-based face recognition. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 39-48). IEEE.

[3] Taigman, Y., Yang, M., Ranzato, M. A., Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1701-1708).

[4] Schroff, F., Kalenichenko, D., Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

[5] Parkhi, O. M., Vedaldi, A., Zisserman, A. (2015). Deep face recognition.

[6] Bao, W., Li, H., Li, N., Jiang, W. (2009, April). A liveness detection method for face recognition based on optical flow field. In 2009 International Conference on Image Analysis and Signal Processing (pp. 233-236). IEEE.

[8] Jiang, H. and Learned-Miller, E. (2017) "Face detection with the faster R-CNN," 2017 12th IEEE International Conference on Automatic Face amp; Gesture Recognition (FG 2017) [Preprint]. Available at: https://doi.org/10.1109/fg.2017.82.

[8] Vikram, K. and Padmavathi, S. (2017) "Facial parts detection using Viola Jones algorithm," 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS) [Preprint]. Available at: https://doi.org/10.1109/icaccs.2017.8014636.

[9] Garg, D. et al. (2018) "A deep learning approach for face detection using Yolo," 2018 IEEE Punecon [Preprint]. Available at: https://doi.org/10.1109/punecon.2018.8745376.

[10] Barnouti, N.H., Al-Mayyahi, M.H. and Al-Dabbagh, S.S. (2018) "Real-time face tracking and recognition system using Kanade-Lucas-Tomasi and two-dimensional principal component analysis," 2018 International Conference on Advanced Science and Engineering (ICOASE) [Preprint]. Available at: https://doi.org/10.1109/icoase.2018.8548818.

[11] G. Pan, L. Sun, Z. Wu, and S. Lao, "Eyeblink-based anti-spoofing in face recognition from a generic webcamera," in ICCV, 2007.

[12] H.-K. Jee, S.-U. Jung, and J.-H. Yoo, "Liveness detection for embedded face recognition system," International Journal of Biological and Medical Sciences, 2006.

[13] J.-W. Li, "Eye blink detection based on multiple gabor response waves," in ICMLC, vol. 5. IEEE, 2008, pp. 2852–2856.

[14] L.Wang, X. Ding, and C. Fang, "Face live detection method based on physiological motion analysis," Tsinghua Science Technology, vol. 14, no. 6, pp. 685–690, 2009.

[15] W. Bao, H. Li, N. Li, and W. Jiang, "A liveness detection method for face recognition based on optical flow field," in ICASSP, 2009.

[16] K. Patel, H. Han, and A. K. Jain, "Cross-database face antispoofing with robust feature representation," in CCBR, 2016.

[17] A. Jourabloo, Y. Liu, and X. Liu, “Face de-spoofing: Anti-spoofing via noise modeling,” in ECCV, 2018.

[18] S. Jia, X. Li, C. Hu, G. Guo, and Z. Xu, “3d face anti-spoofing with factorized bilinear coding,” arXiv preprint arXiv:2005.06514, 2020.

[19] A. George and S. Marcel, “Deep pixel-wise binary supervision for face presentation attack detection,” in ICB, no. CONF, 2019.

[20] Sergievskiy, Nikolay, Roman Vlasov, and Roman Trusov. ”Generalizable Method for Face Anti-Spoofing with Semi-Supervised Learning.” arXiv preprint arXiv:2206.06510 (2022).

[21] William, I. et al. (2019) “Face recognition using FaceNet (survey, Performance Test, and comparison),” 2019 Fourth International Conference on Informatics and Computing (ICIC) [Preprint]. Available at: https://doi.org/10.1109/icic47613.2019.8985786.

[22] Cahyono, F., Wirawan, W. and Fuad Rachmadi, R. (2020) “Face recognition system using Facenet algorithm for employee presence,” 2020 4th International Conference on Vocational Education and Training (ICOVET) [Preprint]. Available at: https://doi.org/10.1109/icovet50258.2020.9229888.

[23] Timoshenko, D., Simonchik, K., Shutov, V., Zhelezneva, P. and Grishkin, V., 2019. Large crowdcollected facial anti-spoofing dataset. 2019 Computer Science and Information Technologies (CSIT), pp.123-126.

[24] Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. ”Mobilenetv2: Inverted residuals and linear bottlenecks.” In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510-4520. 2018.

[25] Melinte, D. O., Vladareanu, L. (2019). Facial Expressions Recognition for Human–Robot Interaction Using Deep Convolutional Neural Networks with Rectified Adam Optimizer. Sensors, 19(19), 4106.

[26] Nakada, M., Wang, H., Terzopoulos, D. (2016). AcFR: Active Face Recognition Using Convolutional Neural Networks. In Proceedings of the 2016 ACM on Multimedia Conference (pp. 68-72).

[27] Ghazi, M. M., Ekenel H.K. (2018). A Comprehensive Analysis of Deep Learning Based Representation for Face Recognition. CVPR workshop.

[28] Qawaqneh, Z., Abu Mallouh, A., Barkana, B. D. (2017). Deep Convolutional Neural Network for Age Estimation based on VGG-Face Model. In 2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT) (pp. 1-6). IEEE.

[29] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How trans- ferable are features in deep neural networks? In Advances in Neural Information Processing Systems, pages 3320–3328, 2014.

[30] Y.LeCun, Y.Bengio,and G.Hinton. Deep Learning. Nature, 521(7553):436–444, 2015.

[31] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.