## Facial Recognition

#### Deepface

Deepface is a lightweight [face recognition](https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/) and facial attribute analysis ([age](https://sefiks.com/2019/02/13/apparent-age-and-gender-prediction-in-keras/), [gender](https://sefiks.com/2019/02/13/apparent-age-and-gender-prediction-in-keras/), [emotion](https://sefiks.com/2018/01/01/facial-expression-recognition-with-keras/) and [race](https://sefiks.com/2019/11/11/race-and-ethnicity-prediction-in-keras/)) framework for python. It is a hybrid face recognition framework wrapping state-of-the-art models: [VGG-Face](https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/), [Google FaceNet](https://sefiks.com/2018/09/03/face-recognition-with-facenet-in-keras/), [OpenFace](https://sefiks.com/2019/07/21/face-recognition-with-openface-in-keras/), [Facebook DeepFace](https://sefiks.com/2020/02/17/face-recognition-with-facebook-deepface-in-keras/), [DeepID](https://sefiks.com/2020/06/16/face-recognition-with-deepid-in-keras/), [ArcFace](https://sefiks.com/2020/12/14/deep-face-recognition-with-arcface-in-keras-and-python/), [Dlib](https://sefiks.com/2020/07/11/face-recognition-with-dlib-in-python/) and SFace.

Pros:

1. High Accuracy: DeepFace has demonstrated impressive accuracy in face recognition tasks, surpassing human-level performance on benchmarks like the Labeled Faces in the Wild (LFW) dataset.
2. Robustness to Variations: DeepFace is designed to handle variations in pose, lighting conditions, and image quality, making it effective in recognizing faces under different scenarios.
3. Easy Integration: DeepFace provides a pre-trained model and a user-friendly API, making it relatively easy to integrate into applications and systems.
4. Developed by Facebook: DeepFace is developed by Facebook AI Research, benefiting from the expertise and resources of one of the leading tech companies.

Cons:

1. Limited Flexibility: DeepFace is a pre-trained model with limited customization options. It may not be suitable for specialized use cases that require extensive model modifications or fine-tuning.

#### Face\_recognition

The world's simplest facial recognition api for Python and the command line.

Built using [dlib](http://dlib.net/)'s state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the [Labeled Faces in the Wild](http://vis-www.cs.umass.edu/lfw/) benchmark.

Pros:

1. User-Friendly API: face\_recognition library provides a simple and intuitive API for face recognition tasks. It offers convenient functions for face detection, feature extraction, and face matching.
2. Easy to Use: The library is easy to install and use, even for users with limited programming experience.
3. Support for Multiple Face Recognition Algorithms: face\_recognition supports different face recognition algorithms, including HOG (Histogram of Oriented Gradients) and deep learning-based models, providing flexibility in choosing the best algorithm for specific use cases.

Cons:

1. Performance and Accuracy: While face\_recognition is easy to use, its performance and accuracy might be lower compared to more advanced deep learning-based models like FaceNet or ArcFace.

#### Facenet-pytorch

Pretrained Pytorch face detection (MTCNN) and facial recognition (InceptionResnet) models

Pros:

1. Deep Learning-based: facenet-pytorch is based on deep learning techniques, specifically FaceNet, which has shown strong performance in face recognition tasks.
2. Flexibility and Customization: facenet-pytorch allows for customization and fine-tuning of the FaceNet model architecture, making it suitable for specialized use cases or research purposes.
3. PyTorch Integration: The library is built on PyTorch, a popular deep learning framework that provides flexibility, ease of use, and access to a wide range of pre-trained models and resources.

Cons:

1. Requires Deep Learning Expertise: Working with facenet-pytorch may require some understanding of deep learning techniques and PyTorch framework, making it less approachable for users without prior experience in these areas.
2. Training Data and Resources: Training FaceNet models from scratch or fine-tuning them require large-scale face datasets and significant computational resources.

## Tracking

#### DeepSort

Pros:

1. Robust Multi-Object Tracking: DeepSort is specifically designed for multi-object tracking, making it suitable for scenarios where tracking multiple objects simultaneously is required.
2. Integration with Object Detection: DeepSort can be easily integrated with object detection algorithms to perform joint object detection and tracking, providing accurate tracking results.
3. Association and Track Management: DeepSort utilizes techniques like the Kalman Filter and Hungarian algorithm for robust object association and track management, ensuring accurate and consistent track predictions.

Cons:

1. Limited in Handling Occlusions: DeepSort may struggle in situations with heavy occlusions, where objects are partially or fully obscured by other objects in the scene.
2. Dependency on Object Detection: DeepSort heavily relies on the accuracy of the object detection stage for initializing and updating tracks. Errors or inaccuracies in the object detection stage can affect the overall tracking performance.

#### SiamRPN++

Pros:

1. High Tracking Accuracy: SiamRPN++ achieves state-of-the-art tracking accuracy by utilizing a Siamese network architecture and a refined training strategy.
2. Real-Time Performance: SiamRPN++ is designed to operate in real-time, making it suitable for tracking applications that require low-latency responses.
3. Robustness to Scale and Occlusion: SiamRPN++ is known for its robustness in handling scale variations and partial occlusions, making it effective in challenging tracking scenarios.

Cons:

1. Limited in Handling Fast Motion: SiamRPN++ may struggle in tracking objects with rapid or abrupt motion, as it relies on the assumption of smooth object motion between consecutive frames.
2. Lack of Multi-Object Tracking Capability: SiamRPN++ is primarily a single-object tracker, and it may require additional methods or algorithms for multi-object tracking scenarios.

#### FairMOT

Pros:

1. High Tracking Accuracy: FairMOT achieves state-of-the-art tracking accuracy by combining a deep learning-based object detector with a motion-aware tracker.
2. Real-Time Performance: FairMOT is designed to run in real-time, making it suitable for applications that require tracking in live video streams or in resource-constrained environments.
3. Open-Source Implementation: FairMOT provides an open-source implementation, making it accessible to developers and researchers for customization and further development.

Cons:

1. Limited in Handling Crowded Scenes: FairMOT may face challenges in tracking objects in crowded scenes where multiple objects are densely packed together, as it relies on accurate object detection and association.
2. Complexity and Resource Requirements: FairMOT utilizes a combination of deep learning models for object detection and tracking, which may require significant computational resources for training and inference.

I found 3 candidates for facial recognition and tracking.

I have to determine final candidate by testing during the progress of this project.

For video, we can consider the combination of facial recognition and tracking.

Let me explain the algorithm(provisional) in case of video.

We may not need tracking if facial recognition works well for video.

Once we recognize face of special person, then we set tracker with the face.

From then, we do facial recognition and tracking together.

1. Initialize Tracker

Once we succeeded in facial recognition for target person, we initialize tracker with the face.

1. Track Face using facial recognition + tracking

For every frame in video:

If we succeeded in facial recognition, we can cover the face of target person with the detected area.

Else we determine the face of target person using tracker + facial detection.

(Even though we failed in facial recognition, facial recognition engine can still detect faces. If we failed in facial detection, then it means there is not any frontal or side faces in the frame. So we can skip the frame.)