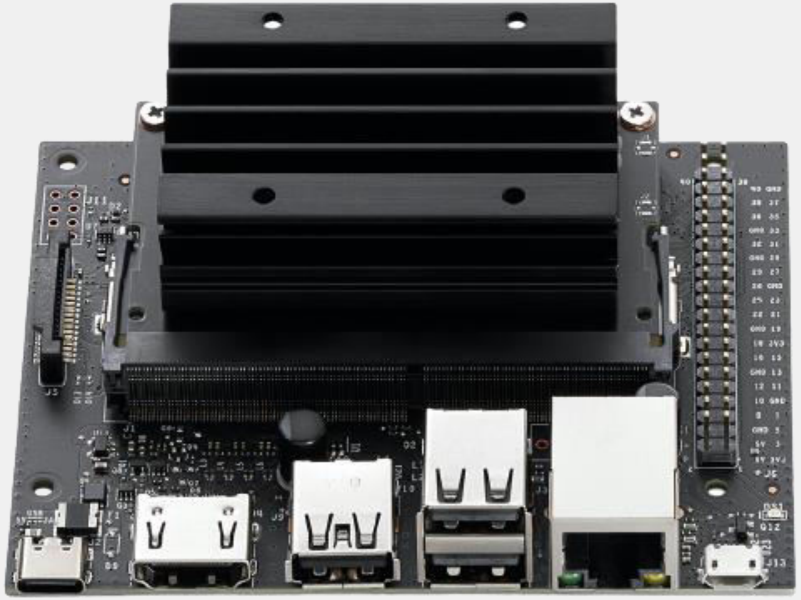
AUGMENTED STARTUPS
Computer Vision | AI | Robotics

Face Recognition and Attendance

INTRODUCTION

Jetson Nano

AUGMENTED STARTUPS
Computer Vision | AI | Robotics



AUGMENTED STARTUPS
Computer Vision | AI | Robotics

Equipment Required

- Nvidia Jetson Nano
- A Camera
- Good Internet

Required Dependencies?

- Python 3.6
- Dlib
- Face_Recognition

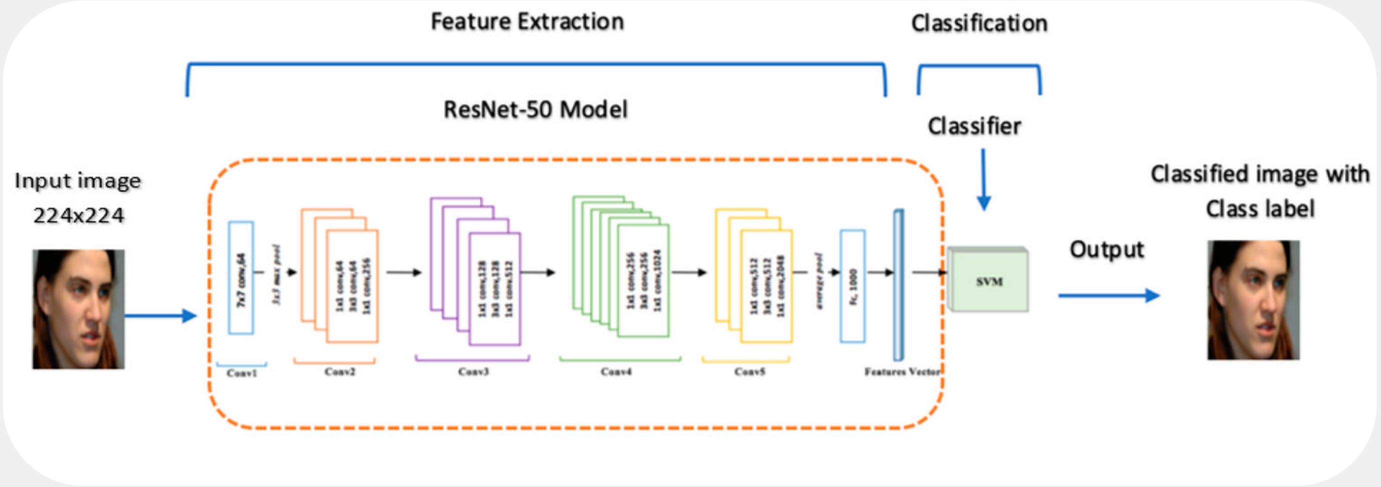
AUGMENTED STARTUPS
Computer Vision | AI | Robotics

HOG and CNN

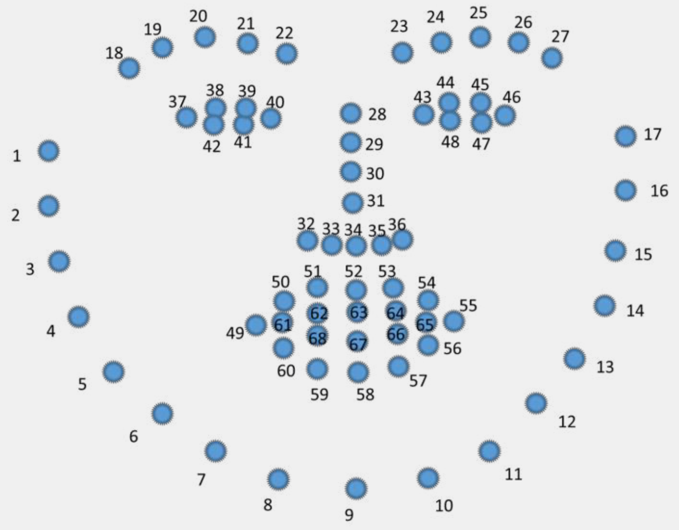
- Histogram Oriented
- Linear Classifier
- Faster

- Convolutional Neural Network
- Slow
- Better Accuracy

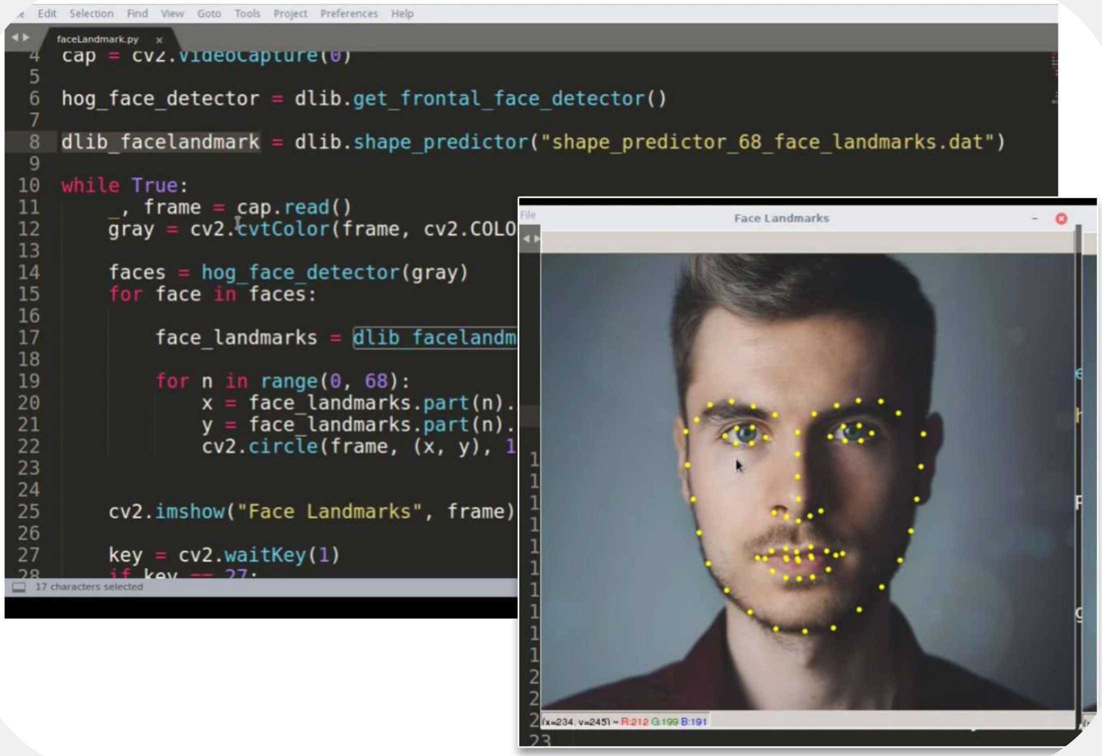
CNN Approach



Dlib Face Encodings

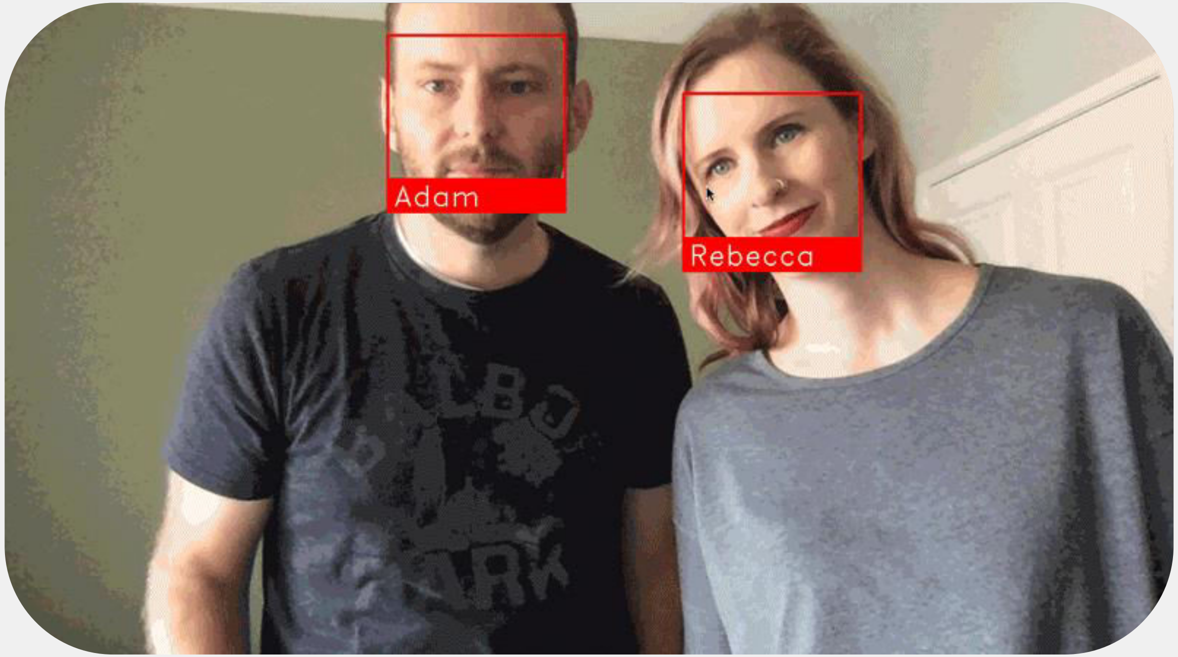


Dlib Face Recognition


Edit Selection Find View Goto Tools Project Preferences Help
faceLandmark.py
17 characters selected
Face Landmarks
(x=234, y=245) ~ R:212 G:199 B:191

```
cap = cv2.VideoCapture(0)

hog_face_detector = dlib.get_frontal_face_detector()

dlib_facelandmark = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLO

    faces = hog_face_detector(gray)
    for face in faces:

        face_landmarks = dlib_facelandm

        for n in range(0, 68):
            x = face_landmarks.part(n).
            y = face_landmarks.part(n).
            cv2.circle(frame, (x, y), 1


    cv2.imshow("Face Landmarks", frame)

    key = cv2.waitKey(1)
    if key == 27:
```

Facial Recognition Result



Training Steps

- Register face
- Look up face
- Keep count

Register Face

```
def register_new_face(face_encoding, face_image):
    known_face_encodings.append(face_encoding)
known_face_metadata.append({
        "first_seen": datetime.now(),
        "first_seen_this_interaction": datetime.now(),
        "last_seen": datetime.now(),
        "seen_count": 1,
        "seen_frames": 1,
        "face_image": face_image,
    })
```

Look up face

```
def lookup_known_face(face_encoding):
    metadata = None
    if len(known_face_encodings) == 0:
        return metadata
    face_distances = face_recognition.face_distance(
        known_face_encodings,
        face_encoding
    )
    best_match_index = np.argmin(face_distances)
```

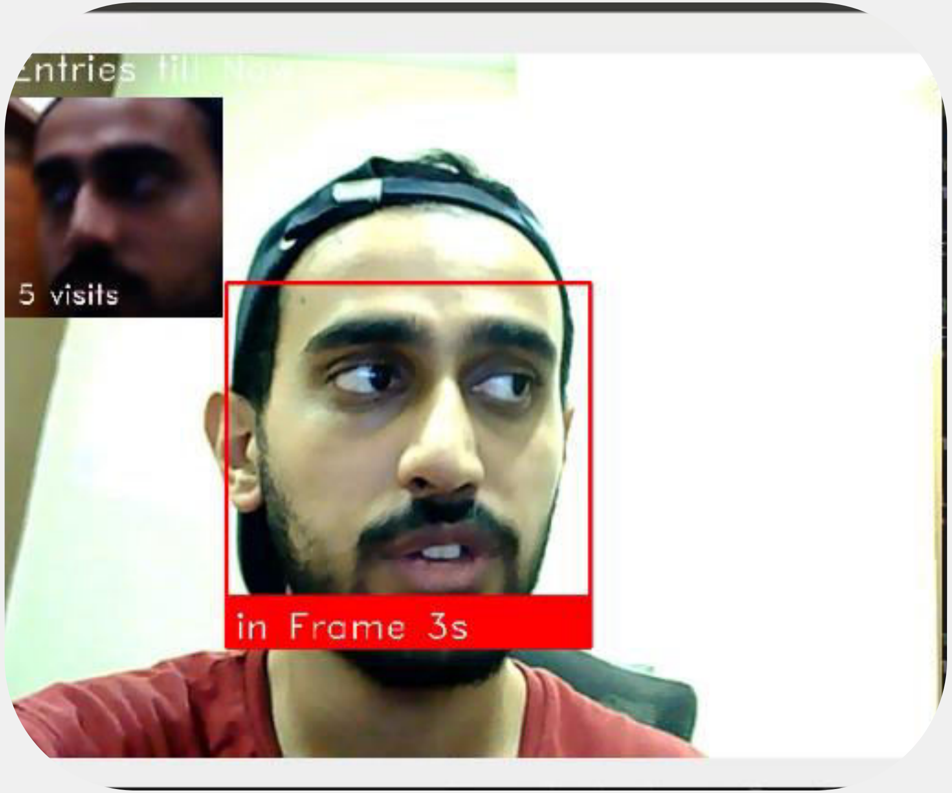
Look up face

```
if face_distances[best_match_index] < 0.65:
        metadata = known_face_metadata[best_match_index]
        metadata["last_seen"] = datetime.now()
        metadata["seen_frames"] += 1
        if datetime.now() - 
metadata["first_seen_this_interaction"] 
                > timedelta(minutes=5):
            metadata["first_seen_this_interaction"] = 
datetime.now()
            metadata["seen_count"] += 1
    return metadata
```

Keep Count

```
    if metadata is not None:
        time_at_door = datetime.now() -
            metadata['first_seen_this_interaction']
        face_label = f"At door
{int(time_at_door.total_seconds())}s"
    else:
        face_label = "New visitor!"
        # Grab the image of the face
        top, right, bottom, left = face_location
        face_image = small_frame[top:bottom, left:right]
        face_image = cv2.resize(face_image, (150, 150))
        # Add the new face to our known face data
        register_new_face(face_encoding, face_image)
```

Results


Entries till Now
5 visits
in Frame 3s