Final SOC Report

A Journey into Machine Learning and Face Recognition

Prithvi Bhardwaj (24b0406)

July 2025

Contents

1 Introduction

The Self-Organized Course (SOC) 2025 was an opportunity for self-paced exploration into the world of machine learning. This report documents my end-to-end journey, covering everything from Python basics to a full-fledged face recognition system using deep learning. Through the process, I faced challenges, experimented, failed, learned, and finally succeeded in building models that actually work. This final report covers all my post-midterm progress in detail.

2 Python Foundations Recap

Before diving into ML, I revised the Python basics:

- **Data Structures:** Lists, Dictionaries, Sets, Tuples
- **Functions:** Arguments, Return Values, *args, **kwargs
- **File Handling:** Read/Write CSV, JSON
- **Modules:** math, os, random, sys
- **OOP:** Classes, Objects, Inheritance

2.1 Code Example: Email Validation using Regex

```
import re

def validate_email(email):
    pattern = r"^[\w\.-]+@[\w\.-]+\.\w{2,4}$"
    return re.match(pattern, email) is not None

print(validate_email('test@example.com'))
```

3 Exploration of Data Science Libraries

3.1 NumPy

Worked with arrays, shapes, reshaping, and broadcasting.

```
import numpy as np
A = np.array([[1, 2], [3, 4]])
B = np.array([[2, 0], [1, 3]])
print(np.dot(A, B))
```

3.2 Pandas

Explored DataFrames, missing data handling, filtering, and group operations.

```
import pandas as pd
df = pd.read_csv("data.csv")
df.fillna(method='ffill', inplace=True)
print(df.groupby("category").mean())
```

3.3 Matplotlib

Used for data visualization: scatter plots, bar graphs, histograms.

```
import matplotlib.pyplot as plt
plt.hist(df['score'], bins=10)
plt.title("Score␣Distribution")
plt.show()
```

4 Deep Learning Foundations

4.1 Getting Started with TensorFlow & Keras

- Built my first neural network
- Learned about activation functions (ReLU, softmax)
- Understood loss functions and optimizers

4.2 Model Code: Image Classifier

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout

model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

4.3 OpenCV Basics

```
import cv2
image = cv2.imread('photo.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray␣Image", gray)
cv2.waitKey(0)
```

5 Siamese Neural Network: One-shot Learning

5.1 Concept

Two identical convolutional networks generate embeddings. We compare embeddings using Euclidean distance.

5.2 Loss Function

Contrastive loss:

$$L = y \cdot D^2 + (1 - y) \cdot \max(0, m - D)^2$$

5.3 My Implementation

- Preprocessed image pairs to 105x105
- Built CNN with shared weights
- Used sigmoid on final similarity layer

5.4 Model Code Snippet

```
def build_siamese():
    input = Input(shape=(105, 105, 1))
    x = Conv2D(64, (10,10), activation='relu')(input)
    x = MaxPooling2D()(x)
    x = Flatten()(x)
    x = Dense(128, activation='sigmoid')(x)
    return Model(inputs=input, outputs=x)
```

5.5 Training

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit([pairs_a, pairs_b], labels, epochs=20, validation_split=0.2)
```

6 Linear Regression: Core ML Model

6.1 What I Explored

- Supervised vs Unsupervised learning
- Cost Function: MSE (Mean Squared Error)
- Gradient Descent
- Polynomial regression and feature scaling

6.2 Simple Linear Regression Code

```
def predict(x, theta):
    return theta[0] + theta[1] * x

def compute_cost(X, y, theta):
    m = len(y)
    predictions = predict(X, theta)
    return (1/(2*m)) * np.sum((predictions - y)**2)
```

6.3 Gradient Descent Code

```
def gradient_descent(X, y, theta, alpha, iterations):
    m = len(y)
    for _ in range(iterations):
        error = predict(X, theta) - y
        theta[0] -= alpha * (1/m) * np.sum(error)
        theta[1] -= alpha * (1/m) * np.dot(error, X)
    return theta
```

7 Face Recognition Project

7.1 Pipeline

1. Face detection using Haar Cascades
2. Data collection and labeling
3. CNN training for classification
4. Real-time recognition with webcam

7.2 Code for Real-time Detection

```
cap = cv2.VideoCapture(0)
while True:
    _, frame = cap.read()
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        face_img = frame[y:y+h, x:x+w]
        label = model.predict(face_img)
        cv2.putText(frame, str(label), (x, y-10), font, 1, (255,0,0), 2)
    cv2.imshow('Face␣Recognition', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

8 Challenges and How I Solved Them

- **TensorFlow-Version Issues:** Downgraded OpenCV to maintain compatibility.
- **Normalization Errors:** Fixed preprocessing pipeline.
- **Incorrect Labeling:** Added delay and manual confirmation.

9 Results and Accuracy

- Siamese Network Validation Accuracy: ˜82%
- Linear Regression Cost: Reduced steadily with learning rate tuning
- Face Recognition: >90% accuracy in closed dataset

10 Key Learnings and Reflections

- Importance of Data Preprocessing
- Learned how training/testing split affects performance
- Confidence in using Jupyter + Git for experiments

11 Next Steps

- Implement triplet loss
- Try ResNet backbone for feature extraction
- Deploy app via Flask or Streamlit

12 Conclusion

The SOC program turned out to be a launchpad for my machine learning journey. From basic scripting to deploying face recognition in real-time, the journey was intense, mistake-filled, but rewarding.

13 References

- https://numpy.org
- https://pandas.pydata.org
- https://www.tensorflow.org/tutorials
- https://docs.opencv.org
- YouTube Playlist: https://youtube.com/playlist?list=PLgNJO2hghbmhHuhURAGbe6KWpiYZt0AMH