19Z610 - MACHINE LEARNING LABORATORY

SMART ATTENDANCE TRACKING

Aadil Arsh SR	**(21Z201)**
Aaditya Rengarajan	**(21Z202)**
Gaurav Vishnu N	**(21Z217)**
Hareesh S	**(21Z218)**
S Karun Vikhash	**(21Z247)**
Sanjay Kumaar Eswaran	**(21Z248)**

PROBLEM STATEMENT

Develop a Smart Attendance System leveraging facial recognition technology to streamline classroom attendance management. The system enables teachers to capture a group photo, automatically counting the number of present students in class and thus preventing proxy. Students authenticate themselves by taking a selfie, cross-referenced against their ID card photos for accurate attendance recording. Attendance data is seamlessly transmitted to the teacher's portal, enhancing efficiency and accountability in classroom administration.

DATASET DESCRIPTION

The students of a class upload their ID card images through a separate login, the face and the OCR details (student name and roll number) are extracted from the ID card images.
The extracted face is converted to binary and stored in our database along with the student name and roll number.

https://drive.google.com/drive/folders/1It4_Negbf52Wj6gLRDTGk4xB2tv4-sFx

METHODOLOGY / MODELS USED

- **CNN**
  Convolutional Neural Networks (CNNs) extract features and carry out challenging visual tasks using mathematical operations including convolutions, pooling, and matrix multiplication.
  Convolution is a mathematical technique that creates a third function by combining two functions. By swiping filters over the image, features are extracted from input photos in CNN contexts.
  The following is the formula for 2D convolution between a filter/kernel (K) and an input image (I):
  $(I * K)(x, y) = \Sigma i\ \Sigma j\ I(x + i, y + j) \cdot K(i, j)$

Where:
$(I * K)(x,y)$ is the output value at position (x, y) in the feature map.
$I(x+i,y+j)$ represents the pixel value at the position (x+i, y+j) in the input image.
$K(i,j)$ represents the filter/kernel weight at position (i, j).
Pooling helps in minimizing the size of feature maps while retaining most of the important information. Max pooling is a common pooling function.
The formula for 2D max pooling with a pooling window of size p×p is:
$\text{Max Pool}(x, y) = \text{maxij } (I(x \cdot p + i, y \cdot p + j))$
Where:
Max Pool(x,y) yields the output value at position (x, y) in the pooled feature map.
I(x p+i,y p+j) represents the input value at position (x \c.p + i, y
\c. p + j) in the feature map.
Matrix Multiplication: Matrix multiplication is a fundamental operation used in neural networks, including fully connected layers. It calculates the dot product between matrices.
The following is the formula matrix product of A and B:
$C = A \cdot B$
Given that m x n are the dimensions for matrix A and n x p are the dimensions of matrix B Resulting in a matrix C with m x p dimensions.
Convolutional neural networks (CNNs) slide filters across images to gather features and reduce spatial dimensions while preserving important information. They excel in extracting features and completing complex visual tasks, making them crucial for computer vision and deep learning. CNNs leverage various filter widths to capture both fine-grained and global characteristics, improving image identification. Additionally, CNNs have diverse applications beyond computer vision, including supporting decision-making in socially responsible investing using multi-attribute benchmarks.

- **InceptionV3**
  InceptionV3 is a convolutional neural network architecture, that is renowned for its effectiveness and precision in image recognition applications. It has effective grid size reduction techniques, auxiliary classifiers to enhance training, and factorized convolutions to save computing costs. With these improvements, it can effectively manage computing loads and get good results on picture classification benchmarks.

- **Keras**
  Neural network construction and training are commonly conducted using Keras, an open-source deep learning framework. It is renowned for having an intuitive interface that enables programmers to rapidly build and test various network topologies. A high-level API is offered by Keras, which supports combinations of recurrent and

convolutional networks. Because of its modular, adaptable, and extensible design, it integrates with other well-known deep learning libraries like TensorFlow and Theano with ease. A large portion of deep learning's complexity is abstracted away by Keras, making it understandable to both novices and professionals.

- **Tesseract**
  Tesseract is an open-source optical character recognition (OCR) engine capable of recognizing text from images.
  Here we use OCR for

TOOLS USED

- **Python:**
  Python is an interpreted, high-level programming language that is renowned for its ease of use and readability. It is extensively utilized in several fields, including artificial intelligence, data science, web development, and more.

- **TensorFlow:**
  A popular tool for creating and refining deep learning models for a range of applications, TensorFlow is an open-source machine learning framework created by Google that provides community support, scalability, and flexibility.
  Here we use it to create a model for detecting the presence of faces in the classroom for the HeadCount feature.

- **Fast API:**
  FAST API is a high-performance Python web framework for building APIs quickly and efficiently, leveraging modern Python features and automatic interactive documentation to streamline development processes.

- **PostgreSQL:**
  An open-source relational database management system known for its extensibility, robustness, and adherence to SQL standards.

- **Docker**
  A platform for developing, shipping, and running applications in isolated, portable containers. Here we use docker containerization for portability.

- **Dlib:**
  Dlib is a flexible C++ library that's frequently used in Python machine-learning applications, especially those involving computer vision. It provides implementations of

several machine learning algorithms, such as clustering, object identification, and facial recognition. Because of its reliability and effectiveness, Dlib is a well-liked option for developers working on complex machine-learning projects.

- **face-recognition:**
  Python module created utilizing dlib's cutting-edge deep learning-based face recognition technology. For the Labeled Faces in the Wild benchmark, the accuracy of the model is 99.38%.

- **Opencv:**
  Opencv is widely used for computer vision tasks like image processing, object detection, and feature extraction, offering a comprehensive suite of tools for analyzing images and videos.
  Here we use it to create a model for detecting the presence of faces in the classroom for the HeadCount feature and for the face extraction from the ID card images for dataset creation.

- **Google-vision API:**
  Google Vision API is a cloud-based service that offers powerful image analysis capabilities such as object detection, OCR, and face detection, simplifying integration into applications for advanced image processing tasks.

- **psycopg2:**
  psycopg2 is a Python library that serves as a PostgreSQL adapter, enabling Python applications to interact with PostgreSQL databases.

- **pandas:**
  Pandas is a Python library for data manipulation and analysis, providing powerful data structures like DataFrame. Here we use pandas for processing image files.

- **numpy:**
  NumPy is a Python library for numerical computing, offering support for large arrays and matrices along with a collection of mathematical functions. Here we use numpy arrays to store processed face-embeddings.

CHALLENGES FACED

- Software cross-compatibility during the development phase.

- Difficulty in adjusting parameter values for the HeadCount Model due to different lighting conditions and resolutions in dataset images.

CONTRIBUTION OF TEAM MEMBERS

Roll No.	Name	Contribution
21z201	Aadil Arsh SR	Back-end for Teacher's Login, Student's Login, Dataset creation, Database Connection.
21z202	Aaditya Rengarajan	Front-end and Back-end Integration, Docker Containerization, API development
21z217	Gaurav Vishnu N	Front-end for Student registration, Teacher registration, Teacher Login
21z218	Hareesh S	Front-end for CRUD operations, FaceRecognition Back-end
21z247	S Karun Vikhash	Front-end and Back-end Integration, HeadCounting Backend Model, API development.
21z248	Sanjay Kumaar Eswaran	Ideation and Conceptualization, FaceRecognition Backend, Database connectivity, Report.

ANNEXURE I: Code

OCR:

```
import pytesseract
def extract_text(img_path):
    img = cv2.imread(img_path)
    ocr_text = pytesseract.image_to_string(img)
    # Define regular expressions to match the patterns
    name_pattern = r"(?<=\n\n)[A-Z\s]+(?=\nBE)"
    name_match = re.search(name_pattern, ocr_text)
    # Extract the matched strings
    if name_match:
        name = name_match.group().strip()
    else:
```

```
        name = None
    print("Name:", name)
    substring = "BE"
    output = extract_text_near_substring(ocr_text, substring)
    output=output[7:]
    output=output[:2]+'z'+output[3:]
    print("ID:", output)
    n1=name
    o1=output
    return name,output
```

Head_Count Model:

```
local_weights_file =
'./content/Face_Counting/model/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

pre_trained_model = InceptionV3(input_shape = (150, 150, 3),
                                include_top = False,
                                weights = None)

pre_trained_model.load_weights(local_weights_file)

# Make all the layers in the pre-trained model non-trainable
for layer in pre_trained_model.layers:
  layer.trainable = False

# Print the model summary
pre_trained_model.summary()

from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import Adadelta
from tensorflow.keras import layers
# Flatten the output layer to 1 dimension
x = layers.Flatten()(last_output)
# Add a fully connected layer with 1,024 hidden units and ReLU activation
x = layers.Dense(256, activation='relu')(x)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dense(1024, activation='relu')(x)
# Add a dropout rate of 0.40
x = layers.Dropout(0.40)(x)
# Add a final sigmoid layer for classification
```

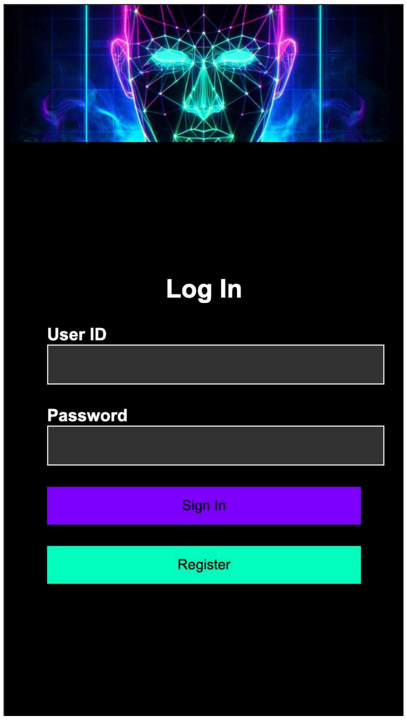
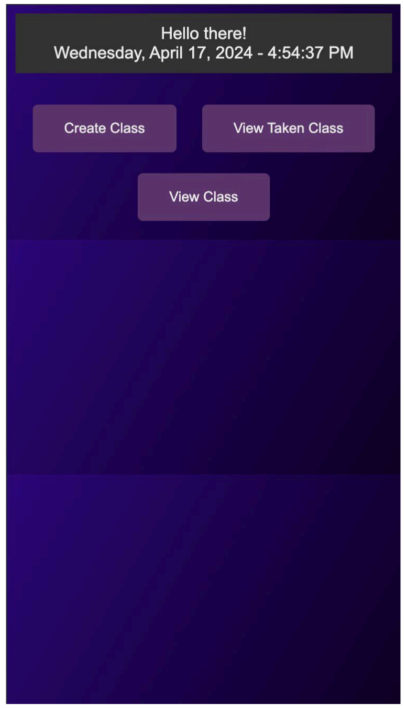
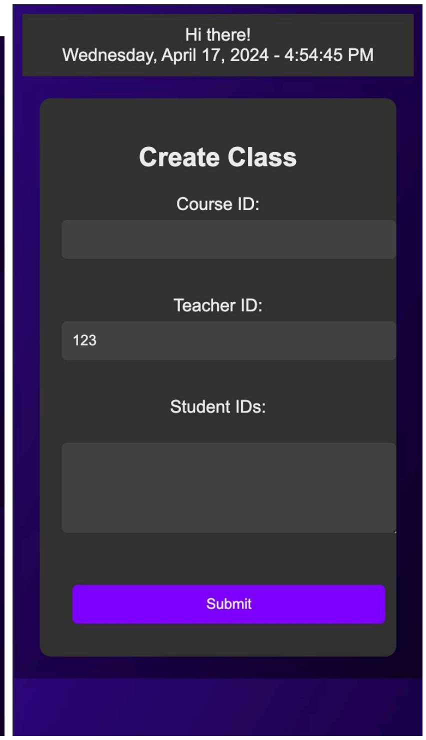
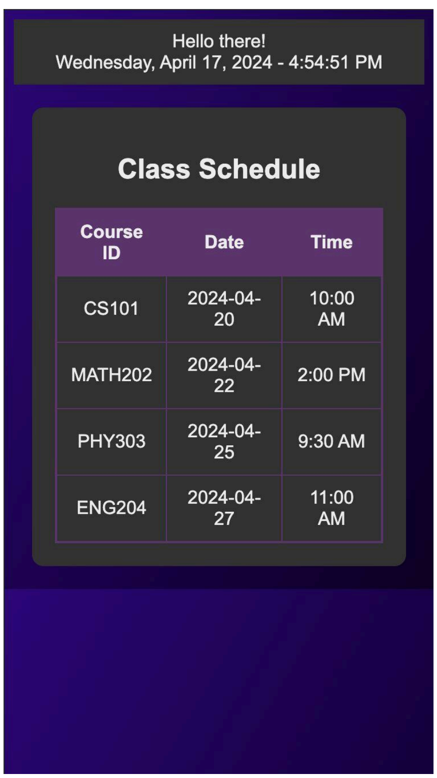
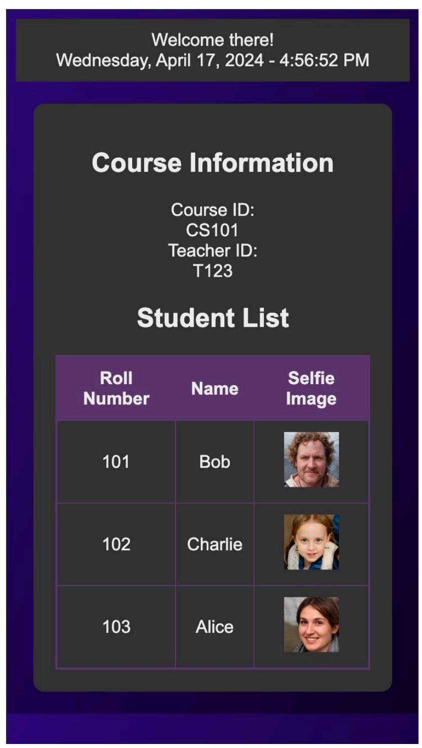
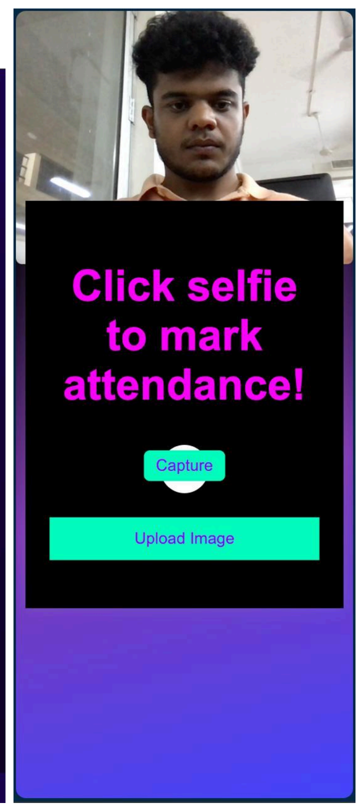
```
x = layers.Dense(1, activation='linear')(x)

model = Model( pre_trained_model.input, x)

model.compile(optimizer = Adadelta(lr=0.001),
            loss = 'mean_squared_error')

model.summary()

history = model.fit_generator(
        train_generator_df,
        validation_data = validation_generator_df,
#           steps_per_epoch = 100,
        epochs = 10,
#           validation_steps = 50,
        verbose = 2,
        callbacks=[callbacks])

plt.figure(figsize=(12,5))
val_loss = history.history['val_loss']
loss = history.history['loss']


epochs = range(len(val_loss))

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()
```

FaceRecognition:

```
def recognize_faces(input_image, dataset_folder  = "./Face_Recog/dataset", threshold=50):
   """Recognize faces in the input image against a dataset."""
   # Get face encodings from input image
   input_face_encodings = get_face_encodings(input_image)

   # Initialize results dictionary
   match_results = {}
```

```
    results = []

    # Iterate over images in the dataset folder
    for filename in os.listdir(dataset_folder):
        if filename.endswith(('.jpg', '.jpeg', '.png', '.JPG')):
            print(filename)
            dataset_image_path = os.path.join(dataset_folder, filename)
            dataset_image = load_image(dataset_image_path)
            dataset_face_encodings = detect_faces(dataset_image)

            if len(input_face_encodings) == 0 or len(dataset_face_encodings) == 0:
                print("No face detected")
                match_results[filename] = 0, "No face detected"
                continue

            # Compare face encodings
            match_percentage = compare_faces(input_face_encodings, dataset_face_encodings)

            if match_percentage >= threshold:
                print(f"Image: {filename}, Match Percentage: {match_percentage:.2f}%, Result: The
faces are the same")
                results.append(f"Image: {filename}, Match Percentage: {match_percentage:.2f}%,
Result: The faces are the same")
                match_results[filename] = match_percentage, "The faces are the same"
                # Save the matching image
                save_path = os.path.join('matches', filename)
                cv2.imwrite(save_path, input_image)
            else:
                print(f"Image: {filename}, Match Percentage: {match_percentage:.2f}%, Result: The
faces are different")
                results.append(f"Image: {filename}, Match Percentage: {match_percentage:.2f}%,
Result: The faces are different")
                match_results[filename] = match_percentage, "The faces are different"

    return results
```

ANNEXURE II: Snapshots of the Output


Log In
User ID
Password
Sign In
Register

Hello there!
Wednesday, April 17, 2024 - 4:54:37 PM
Create Class
View Taken Class
View Class

Hi there!
Wednesday, April 17, 2024 - 4:54:45 PM
Create Class
Course ID:
Teacher ID:
123
Student IDs:
Submit

Hello there!
Wednesday, April 17, 2024 - 4:54:51 PM
Class Schedule
Course ID
Date
Time
CS101
2024-04-20
10:00 AM
MATH202
2024-04-22
2:00 PM
PHY303
2024-04-25
9:30 AM
ENG204
2024-04-27
11:00 AM

Welcome there!
Wednesday, April 17, 2024 - 4:56:52 PM
Course Information
Course ID:
CS101
Teacher ID:
T123
Student List
Roll Number
Name
Selfie Image
101
Bob
102
Charlie
103
Alice

Click selfie to mark attendance!
Capture
Upload Image

REFERENCES

1) Author, F.: Gavkare, A., Mandlik, A., Mutkule, A., Chinchane, A., Sheikh, A. Article title: Smart Attendance System Using HOG and SVM Journal: International Journal of Innovative Research in Technology (IJIRT) 2(5), 1–4 (2021)

2) Ballings, M., Verbraken, T., Van den Poel, D.: Using convolutional neural networks for price direction prediction. Expert Systems with Applications 72, 1-10 (2017)

3) Chiang, T-W., Yang, C-Y., Chiou, G-J., Lin, F. Y-S., Lin, Y-N., Shen, V. R. L., Juang, T. T-Y., Lin, C-Y.: Development and Evaluation of an Attendance Tracking System Using Smartphones with GPS and NFC. Applied Artificial Intelligence 36(1), 2083796 (2022)

4) Chollet, F.: Deep learning with Python. 1st edn. Manning Publications (2017)

5) Dodiya, H., et al.: Attention, Emotion, and Attendance Tracker with Question Generation System Using Deep Learning. In: 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1–6. IEEE (2021).

6) Khawla Alhanaeea, Mitha Alhammadia, Nahla Almenhalia, Maad Shatnawia: Face Recognition Smart Attendance System using Deep Transfer Learning. 25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Procedia Computer Science 192 (2021) 4093-4102.

7) Lin, C.-L., Huang, Y.-H.: The Application of Adaptive Tolerance and Serialized Facial Feature Extraction to Automatic Attendance Systems, pp.1-15 (2022).

8) Mridha, K., Yousef, N. T.: Study and Analysis of Implementing A Smart Attendance Management System Based on Face Recognition Technique Using OpenCV and Machine Learning. IEEE Conference Publication, 2021, pp. 1-5 (June 2021)

9) O'Shea K and R. Nash: An Introduction to Convolutional Neural Networks. In: arXiv preprint, (2015).

10) Prompinit, T., Cheawcharnthong, S., Mongkolnam, P., Chan, J.H.: Facial Recognition Attendance Checker. In: The Joint Symposium Computational Intelligence (JSCI), pp. 1-2. IEEE Computational Intelligence Society Thailand Chapter, Bangkok (2018).