📄 Major Project 2 Report

Facial Recognition System for Secure Employee Identification

🧠 1. Project Overview

This project focuses on developing a facial recognition system that securely verifies employee identities using facial embeddings and machine learning. The system aims to automate identity verification, reduce manual intervention, and improve overall security in access control systems.

📌 2. Problem Definition

In various sectors such as banking, online education, and corporate access control, verifying a person's identity accurately and quickly is essential. Manual verification is time-consuming and prone to human error.

Goal: To verify whether a new facial input (image or video) belongs to a registered employee using AI and face embeddings.

📦 3. Dataset and Loading

📁 Dataset: lfw_arnie_nonarnie.csv

- Derived from the Labeled Faces in the Wild (LFW) dataset.
- Contains 128-dimensional facial embeddings, not raw images.
- Each row represents a single facial embedding vector.
- Label column: person (e.g., arnold_schwarzenegger or others)

python

df = pd.read_csv("lfw_arnie_nonarnie.csv")

df['target'] = df['person'].apply(lambda x: 1 if x == 'arnold_schwarzenegger' else 0)

📊 4. Data Exploration and Preprocessing

✅ Class Distribution

Visualized to check dataset balance between target classes:

python

sns.countplot(x='target', data=df)

plt.title("Class Distribution (1 = Arnie, 0 = Others)")

🖌️ Preprocessing Steps

- Feature Scaling using StandardScaler
- Train-Test Split (typically 80/20)
- Label Encoding to convert names into binary values

python

```
X = df.drop(['person', 'target'], axis=1)
y = df['target']
X_scaled = StandardScaler().fit_transform(X)
```

🧠 5. Model Development

🔷 Deep Learning Model (Keras)

A simple feedforward neural network was created:

python

```
model = Sequential([
    Dense(128, activation='relu', input_dim=128),
    Dropout(0.4),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
```

Trained using:

- Epochs: 50
- Batch Size: 32
- Callback: EarlyStopping to avoid overfitting

📈 6. Visualizations & Evaluation

🔹 a. Accuracy Plot

Shows how model accuracy changed over training epochs:

python

```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training vs. Validation Accuracy')
```

plt.legend()

🔹 b. Confusion Matrix

python

from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Non-Arnie", "Arnie"]).plot()

Highlights true positives and false classifications.

💉 7. Model Metrics Summary

Metric	Value
Test Accuracy	78.95%
Loss Function	Binary Crossentropy
Optimizer	Adam
Confusion Matrix	Displayed via ConfusionMatrixDisplay
Overfitting	Low — accuracy and loss were stable

💻 8. Interface Design (Prototype)

A simple Streamlit UI was prototyped to simulate deployment:

python

```
st.title("Employee Face Recognition")
uploaded_file = st.file_uploader("Upload a Face Image")

if uploaded_file:
    image = Image.open(uploaded_file).resize((224,224))
    img_array = np.array(image)/255.0
    prediction = model.predict(img_array.reshape(1, 224, 224, 3))
    label = np.argmax(prediction)
    confidence = np.max(prediction)
    st.success(f"Prediction: Class {label} (Confidence: {confidence:.2f})")
```

🚀 9. Deployment Strategy

- Model Hosting: AWS EC2, Render, or Heroku
- App Backend: Flask / FastAPI / Streamlit
- Security: HTTPS, encrypted storage, user authentication
- Database: Encrypted SQLite or PostgreSQL for identity storage

---

🔐 10. Ethical & Legal Considerations

- Comply with GDPR and India's DPDP Act
- Take explicit user consent for image usage
- Provide a data deletion request option
- Avoid use in non-consensual surveillance

📌 11. Conclusion

This project successfully demonstrates:

- Use of facial embeddings for identity verification
- Building a deep learning model on numerical features
- Achieving 78.95% accuracy, with potential for real-time use
- Developing a simple user interface for demonstration
- Laying groundwork for future scalable identity verification systems

---

🔮 12. Future Improvements

- Integrate OpenCV for real-time webcam verification
- Add liveness detection (blink, head movement)
- Expand to multi-class identity classification
- Use FaceNet/MobileNet for improved embeddings
- Optimize for edge deployment (e.g., mobile apps)

📁 Appendix

📂 Files Included

- lfw_arnie_nonarnie.csv – Face embedding dataset
- Face_identification_model.ipynb – Notebook with training code and results