

# Facial Recognition System Project

# Team project

Basel Tarek Abdelmonsif  
Mohammed Essam Shehata  
Mostafa Sayed Salah  
Naser Ali Naser  
Zain Tamer Zain El-Abdin Awad

# Executive Summary

- 5 key milestones of the project
- Data collection & preprocessing (LFW dataset)
- Model development (FaceNet with transfer learning)
- Deployment (Streamlit on Hugging Face)
- MLOps implementation (MLflow, Kubeflow)
- Performance analysis

# Project Overview

- Problem: Challenges in facial recognition  
(varying lighting, poses, demographic bias)
- Solution: End-to-end production-ready system
- Application domains: Identity verification,  
access control, security

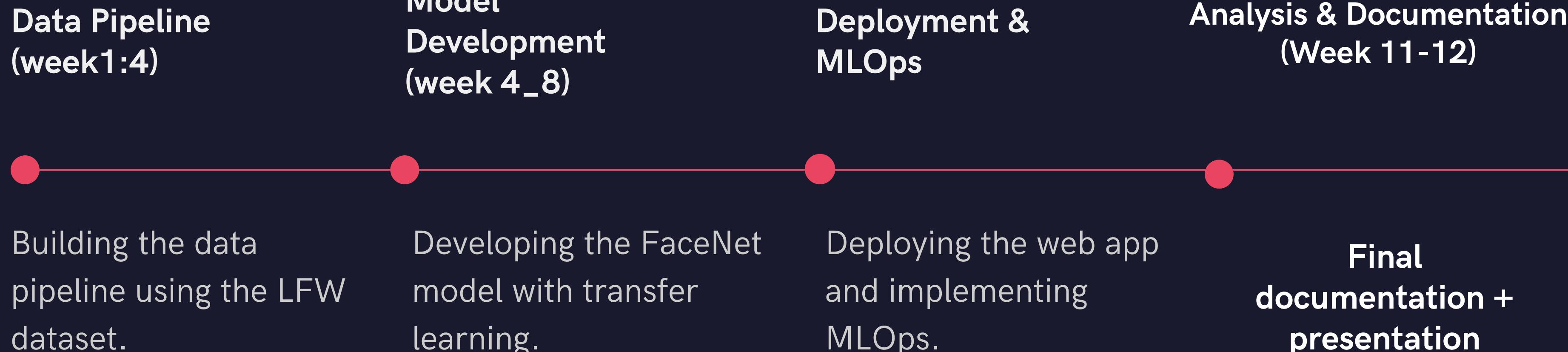
# Project Objectives

## Achieving High Accuracy and Efficiency

- Achieve high accuracy (>80%) on LFW dataset
- Implement comprehensive MLOps practices
- Build production-ready web application
- Optimize performance for real-time inference
- Ensure fairness across demographic groups

# Project Timeline

**Data Pipeline**  
**(week 1-4)**



Building the data pipeline using the LFW dataset.

**Model Development**  
**(week 4-8)**

Developing the FaceNet model with transfer learning.

**Deployment & MLOps**

Deploying the web app and implementing MLOps.

**Analysis & Documentation**  
**(Week 11-12)**

Final documentation + presentation

# Team Roles

## Allocation of Resources and Responsibilities

Our project relies on a **dedicated team** working on enhancing the dataset by different augmentation with utilizing GPU resources for training, along with MLflow for tracking and integrating this work into a fully functional web app.



# LFW dataset exploration

This dataset consists of **13233** images distributed among different  
**5749** classes



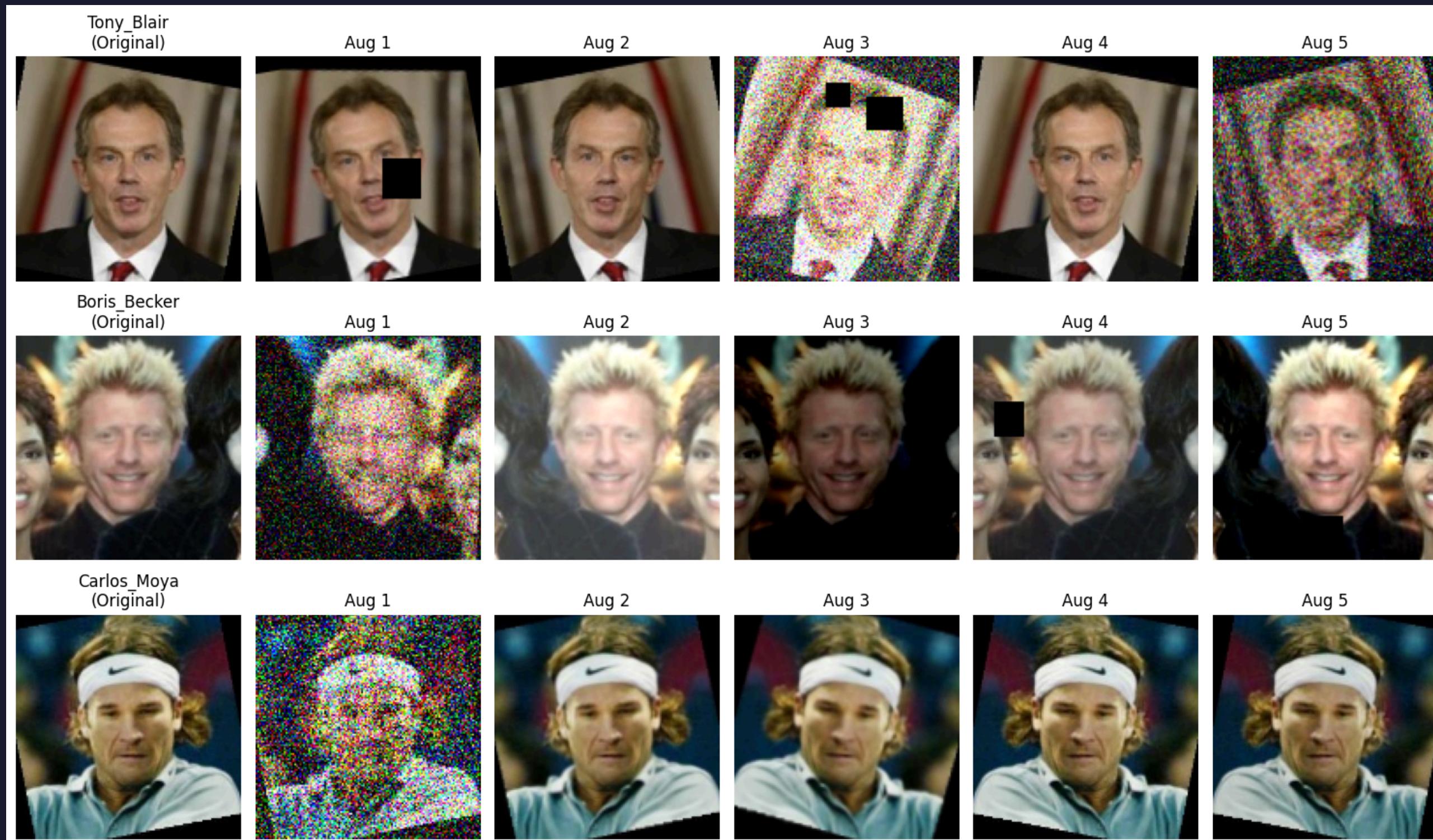
# Data Augmentation

```
[6] > augment = A.Compose([
    A.HorizontalFlip(p=0.5),
    A.Rotate(limit=15, p=0.5),
    A.RandomBrightnessContrast(brightness_limit=0.3, contrast_limit=0.3, p=0.6),
    A.GaussianBlur(blur_limit=(1, 3), p=0.3),
    A.GaussNoise(var_limit=(5, 15), p=0.3),
    A.RandomScale(scale_limit=0.1, p=0.3),
    A.CoarseDropout(max_holes=8, max_height=8, max_width=8, p=0.2),
])

def apply_aug(img):
    return augment(image=img)["image"]
```

% Generate + Code + Markdown

# Data Augmentation



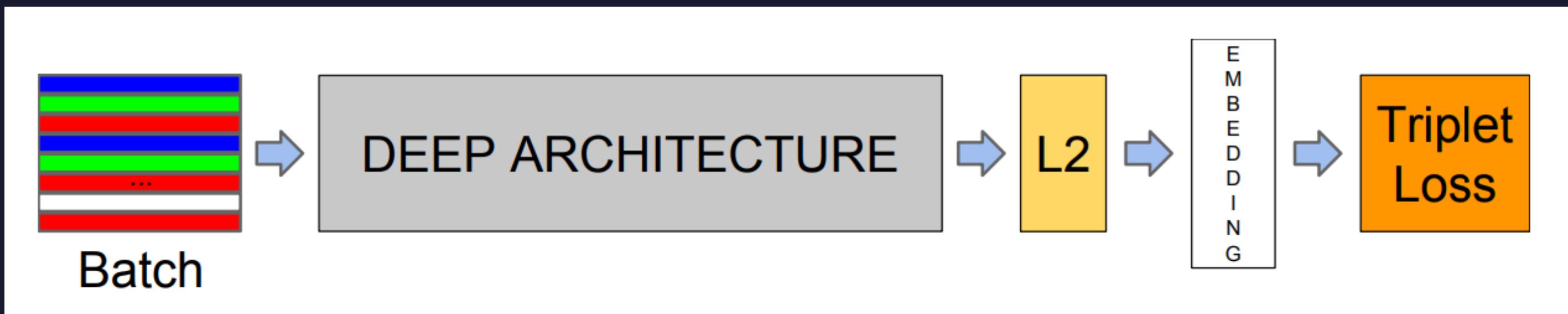
# FaceNet Model

FaceNet is a deep learning model for face recognition. It does not classify faces directly — instead, it learns to map each face image to a 128-dimensional embedding vector.

Architecture	Parameters	Embedding Size
FaceNet	23M	128
VGGFace2 (ResNet-50)	25M	2048
ArcFace (ResNet-100)	65M	512
MobileFaceNet	1M	128

# FaceNet Model

Architecture:



# Apply Transfer Learning Head for FaceNet Model

Model: "facenet\_transfer\_head"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 256)	33,024
batch_normalization (BatchNormalization)	(None, 256)	1,024
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 50)	12,850

Total params: 46,898 (183.20 KB)

Trainable params: 46,386 (181.20 KB)

Non-trainable params: 512 (2.00 KB)

# System Design Overview

01

## Presentation Layer

User interfaces facilitate interaction and display results seamlessly.

02

## Application Layer

Core logic processes requests and manages data flow efficiently.

03

## ML Layer

Machine learning models perform tasks such as face recognition.

# Face Verification Flow

The face verification process ensures accurate matching through a systematic approach, enhancing security and user experience.



# Testing and Metrics

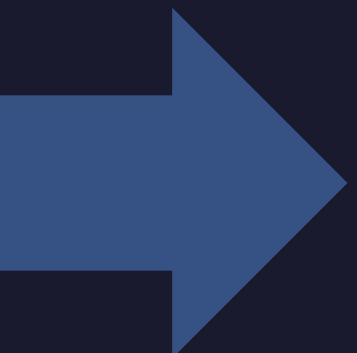
## Ensuring Quality and Reliable Performance

Comprehensive testing improved accuracy from 70% to 80% after transfer learning, achieving over 99% uptime and maintaining inference times under 500 milliseconds for efficiency.



# Accuracy

70



80

Pre-trained FaceNet  
accuracy

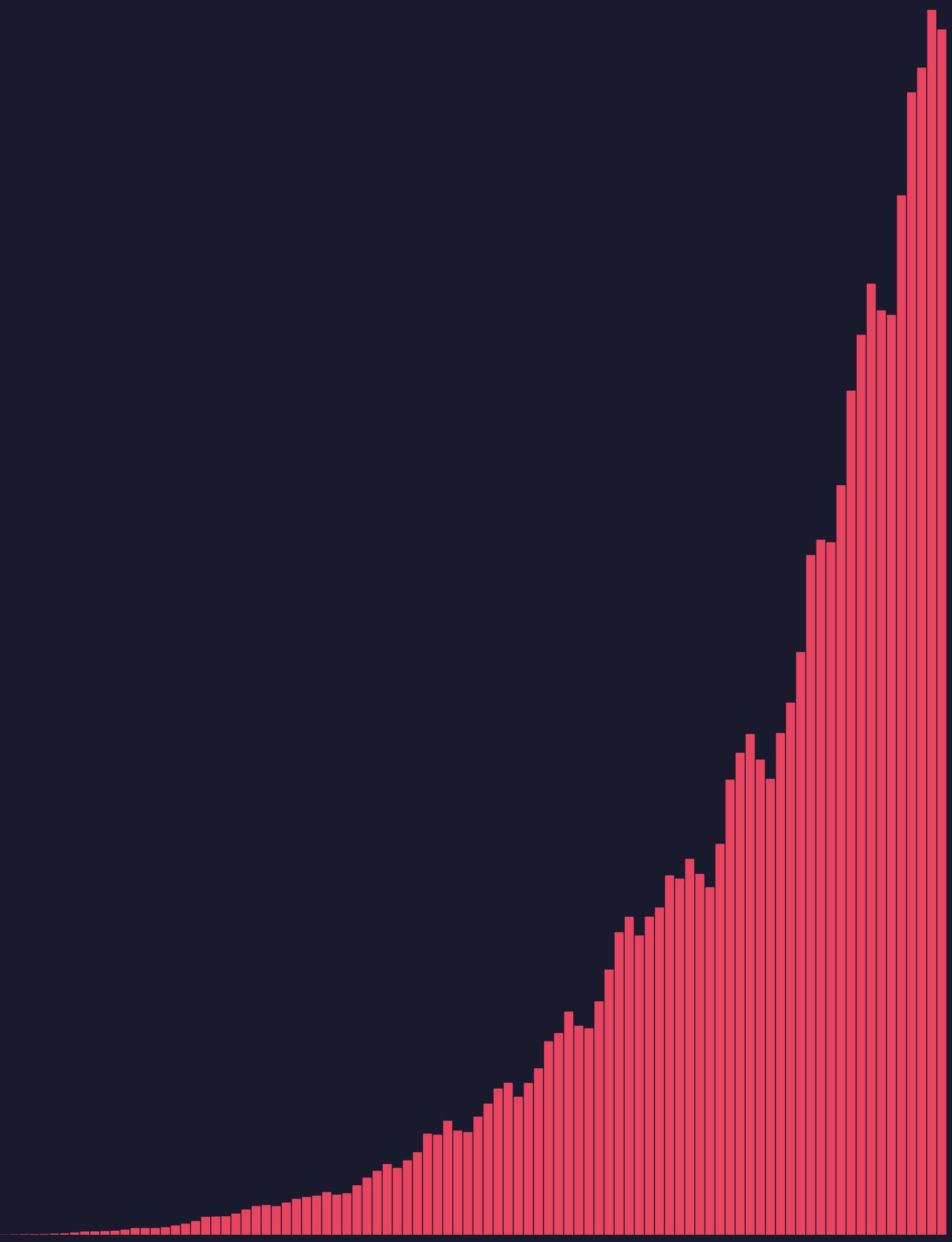
Fine-tuned FaceNet  
accuracy

# Key Performance Statistics

80% Recognition  
Accuracy

**<500 ms**

Inference time for real-time processing



# Web App

We built web app using Streamlit

Streamlit is a lightweight Python framework for building interactive web applications.

In our project, we used Streamlit to build the web app for our facial recognition system. It handled image uploads, displayed results, and provided an easy-to-use interface for running FaceNet-based face verification.



# Deployment

We hosted our web app using Hugging Face Spaces

Hugging Face Spaces is an online platform that allows developers to host and deploy machine learning applications easily. It supports frameworks like Streamlit and others, making model deployment simple and accessible. In our project, we used Hugging Face as the hosting platform for our facial recognition web app, allowing users to access and test the system directly through the browser.

# Ethical Considerations in Facial Recognition Systems



## Bias Mitigation

Ensuring **balanced datasets** for fair outcomes is crucial.

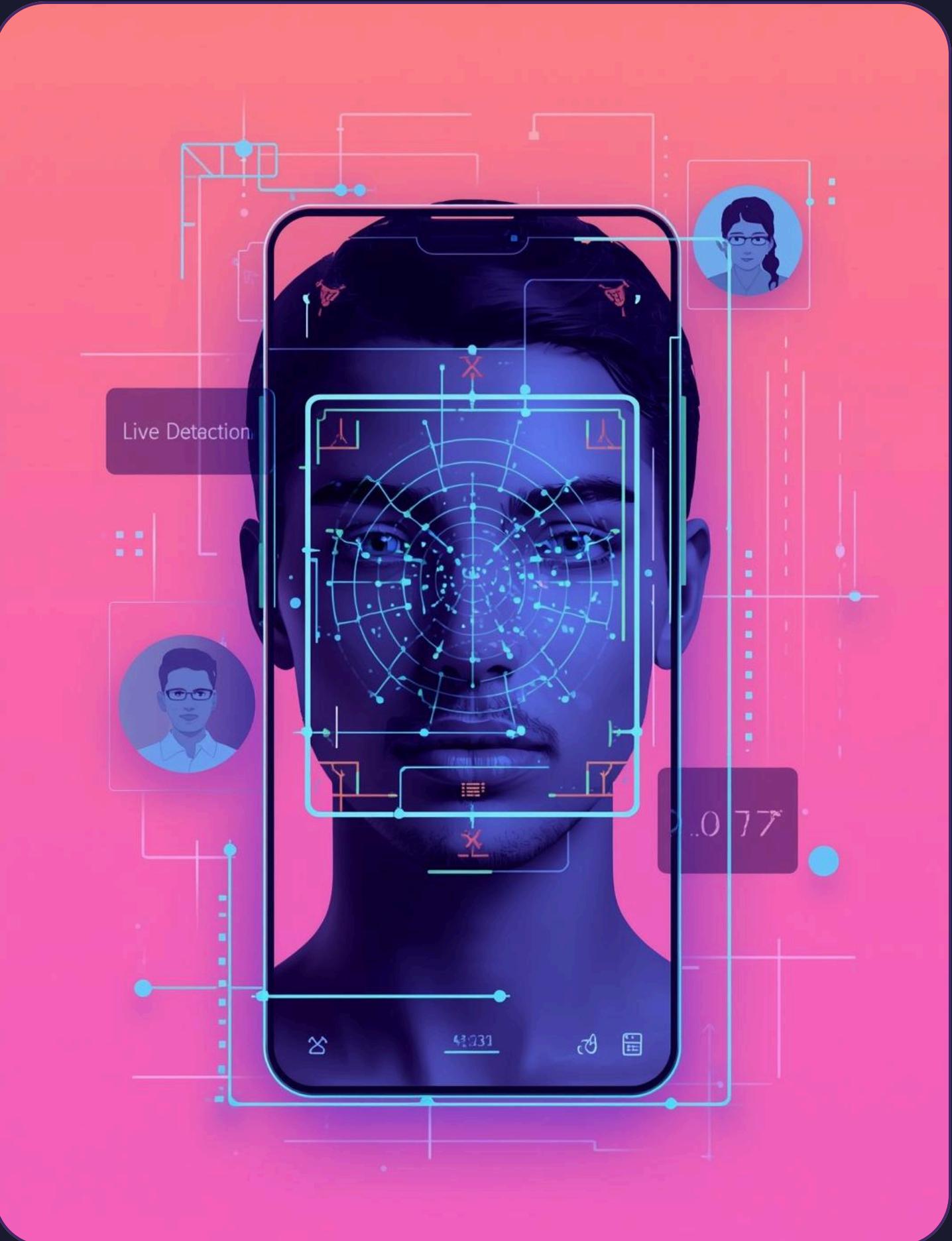


## Privacy Protection

Implementing **data anonymization** measures to safeguard identities.

# Key Achievements

- Successfully achieved 80% accuracy on LFW dataset
- Production-ready system with MLOps best practices
- Robust deployment on Hugging Face Spaces
- Scalable microservices architecture
- Commitment to fairness and privacy



# Challenges & Resolutions

- Challenge 1: Making classes in the dataset close to us

Resolution: Used augmentation techniques changing the brightness and tone

- Challenge 2: Live video lag

Resolution: Implemented frame skipping (1 recognition/second)

# Demo



# Thank You

