Report: ML-Powered Face Recognition for Event Media Management

1. Business Case

Event organizers often struggle with sorting and retrieving images of specific individuals from large collections of event photos. **Manual tagging is inefficient and error-prone**. Machine Learning (ML)-powered **face recognition** can **automatically detect, group, and retrieve people's photos** from event datasets, making media organization more efficient.

2. Business Value of Using ML

Efficiency & Automation

- **Automatically detects and organizes faces** across event photos.
- Eliminates the need for manual sorting and tagging.

Accurate & Fast Retrieval

- Users can **search for a person's event photos** by uploading an image.
- Provides a **confidence score** (e.g., "90% match") for face recognition accuracy.

Scalability & Data Insights

- Can process thousands of images in seconds.
- Tracks how often individuals appear in event photos.

3. ML Framing

Project Archetype

- **Computer Vision: Face Recognition & Matching**

Baseline Model Selection

1. **Face Recognition**:
    - **FaceNet (Schroff et al., 2015)** – Achieves **99.63% accuracy** on the LFW dataset.
    - **Dlib-based Face Recognition** – Lightweight and widely used for facial clustering.
2. **Similarity Matching**:
    - **Cosine similarity** or **Euclidean distance** for face embeddings.

Baseline Feasibility & Justification

- **FaceNet** is state-of-the-art for facial verification and clustering.
- **Dlib** is computationally efficient and widely used for face recognition.
- Pre-trained models available from **Hugging Face** and **OpenAI APIs**.

4. Metrics for Business Goal Evaluation

- **Face Recognition Accuracy** (Precision, Recall, F1-score on LFW dataset).
- **Matching Confidence Score** (Percentage similarity for each face match).
- **Retrieval Speed** (Time taken to find a match in the dataset).

5. Proof of Concept (PoC) Description

The PoC will be built using **Streamlit** for UI, **FastAPI** for backend, and **FaceNet/Dlib** for ML inference.

PoC Features

1. **Upload & Organize Event Photos**
    - Automatically detects and groups people by face.
2. **Search Photos by Face**
    - Users upload a photo to find all images of the same person.
    - Provides a **match confidence score** (e.g., “85% match”).
3. **Event Analytics Dashboard**
    - Tracks how often individuals appear in event photos.

PoC Implementation Steps

1. **Face Detection & Embeddings**
    - Extract facial embeddings using **FaceNet/Dlib**.
2. **Face Matching & Clustering**
    - Group photos by facial similarity.
3. **UI & Deployment**
    - **Streamlit UI** for user interaction.
    - **FastAPI backend** for processing face recognition tasks.

Objective
Develop a web-based **face recognition system** that:

- **Automatically detects and clusters faces** in event images.
- **Allows users to search** for a person’s photos by uploading an image.
- **Provides a similarity score** (confidence percentage) for matches.

Tech Stack

Component	Technology
Frontend	Streamlit (for UI)

Component	Technology
Backend	FastAPI
Face Detection	MTCNN (Multi-task Cascaded Convolutional Networks)
Face Recognition	FaceNet / Dlib
Database	SQLite / PostgreSQL (stores face embeddings)
Storage	Local file system (for images)
Face Similarity Matching	Cosine Similarity / Euclidean Distance

- **Upload event photos** → Faces are detected and grouped by similarity.
- **Search for a person's photos** → User uploads an image, and the system retrieves all matching photos.
- **View match confidence** → Each result shows a **similarity score (%)**.

Technical Overview

- **Frontend**: Streamlit (simple UI for uploads, search, and results).
- **Backend**: FastAPI (handles face detection, embedding extraction, and retrieval).
- **Face Detection**: MTCNN (to locate faces in images).
- **Face Recognition**: FaceNet or Dlib (to generate and compare embeddings).
- **Similarity Matching**: Cosine similarity to rank face matches.
- **Database**: PostgreSQL (stores face embeddings for retrieval).
- **Storage**: Local storage or cloud for event images.

Workflow

1. **Upload Event Photos** → The system extracts facial embeddings and groups similar faces.
2. **Search by Face** → User provides a query image, and the system finds matching faces.
3. **Display Matches** → Results are ranked by confidence score (e.g., "92% match").
4. **Event Insights** → Dashboard shows event stats (most photographed individuals, retrieval speed).

Deployment

- **Model setup & API development** (FaceNet/Dlib + FastAPI).
- **UI design with Streamlit** (upload, search, and result display).
- **Testing & optimization** with event datasets for accuracy and performance.

Model Card: Face Recognition Baseline Model
Model Overview

Model Name: FaceNet + MTCNN
Version: 1.0
Framework: PyTorch
License: Apache 2.0
Model Type: Face Recognition (Feature Extraction & Similarity Search)
Application: Organizing and retrieving faces from event photos based on similarity matching.

Intended Use

This model is designed for **face recognition without classification**. It organizes people based on facial similarity, providing a similarity score to match faces across multiple event images. The model extracts facial embeddings and clusters individuals for efficient photo organization.

Use Cases:
Event photo organization
Face-based retrieval of individuals from large datasets
Improving searchability of media archives
Non-Intended Use:
Real-time security surveillance
Law enforcement or identity verification
Deepfake detection

Model Details

- **Face Detection Model:** MTCNN (Multi-task Cascaded Convolutional Networks)
- **Face Recognition Model:** FaceNet (InceptionResnetV1 trained on VGGFace2)
- **Feature Dimension:** 512-D embedding vector
- **Training Data:** Pretrained on **VGGFace2**, containing 3.3 million images from 9,131 identities.

Baseline Model Selection & Justification

- **MTCNN** is selected for face detection due to its high accuracy in locating multiple faces in images.
- **FaceNet (InceptionResNetV1)** is used for face embedding extraction, as it provides robust 512-dimensional embeddings that preserve facial similarity.
- **Cosine Similarity** is applied to compare embeddings and organize faces with high confidence scores.

Reference Paper:

- *Schroff, Florian, et al. "FaceNet: A Unified Embedding for Face Recognition and Clustering." CVPR 2015.*

State-of-the-Art Comparison:

- FaceNet achieves **99.63% accuracy** on the Labeled Faces in the Wild (LFW) benchmark.
- Works well in **unconstrained environments** (various angles, lighting conditions).

Performance Metrics

Metric	Score (Baseline)
Face Detection Accuracy (MTCNN)	~95%
Face Recognition Accuracy (FaceNet)	~99.6% (on LFW)
Mean Cosine Similarity (Matched Faces)	≥0.8
Mean Cosine Similarity (Unmatched Faces)	≤0.4

- **Threshold Tuning:** Faces are considered a match if similarity **≥ 0.7**.
- **Evaluation Dataset:** LFW & custom event images dataset.

Feasibility & Deployment

- **Baseline Model Availability**:
  Pretrained weights available on Facenet PyTorch Repo.
  Notebook for re-training available: FaceNet Training Notebook.
- **Deployment Recipe:**
  - Convert trained model to **ONNX** for lightweight deployment.
  - Backend via **FastAPI** or **Flask** (Docker-based).
  - Frontend using **Streamlit** (for PoC).

Limitations & Ethical Considerations

- **Bias in Training Data:** VGGFace2 has more representation for certain ethnicities, leading to possible imbalances.
- **Privacy Concerns:** The model does not store personal data but processes facial embeddings, which should be encrypted.
- **Accuracy in Crowded Scenes:** Face occlusion reduces detection performance.

Future Improvements

-**Fine-tuning on event-specific datasets** for improved retrieval accuracy.
-**Integration with clustering techniques** (e.g., DBSCAN) to group images without predefined labels.
-**Edge Deployment Optimization** (TensorRT for low-latency processing).

Model Repository & Code

Baseline Model Repository: FaceNet PyTorch
Dataset: VGGFace2 Dataset
Notebook for Feature Extraction: Colab Notebook