CSL2050
Pattern Recognition & Machine Learning

Course Project

“ Face Identification”



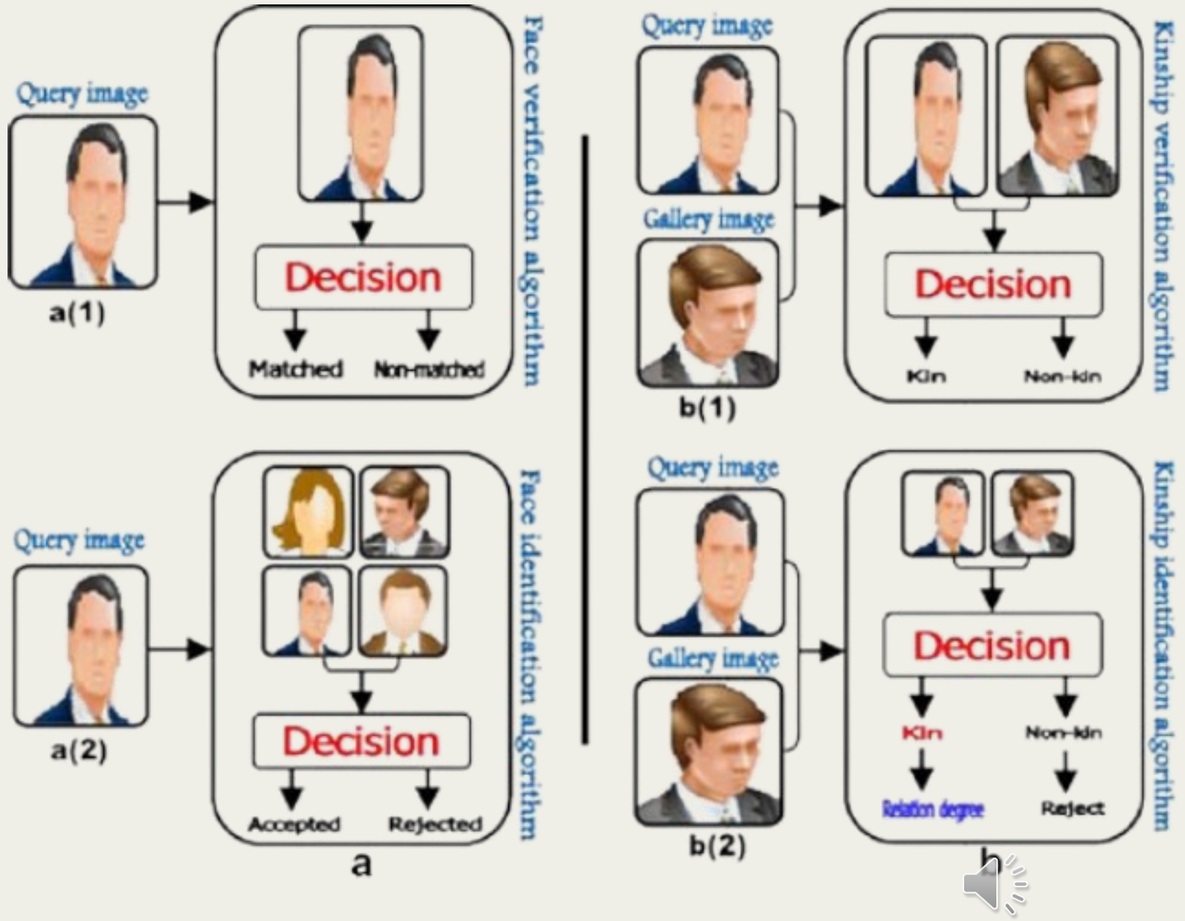


Sandeep Soni, Atanu Kayal, Japneet Singh,
Varchasva Saxena, Vishal

Introduction & Motivation

Problem Statement:

Build a robust face-identification system that classifies any given face image into one of K known identities



Introduction & Motivation

Problem Statement:

Build a robust face-identification system that classifies any given face image into one of K known identities

Key Application:

Automated Attendance – mark people in/out in real time using face recognition

Datasets

LFW Dataset

(Labeled Faces in the Wild)

- ~13,000 images of 5,749 individuals
- Captured in uncontrolled settings

Problems:

- Few images per identity → overfitting
- Imbalanced classes → biased models
- Poor generalization for deep models

Only good for face verification, not identification
Initially used for prototyping

VGGFace2

(Final Dataset)

- 3.3M images across 9,131 identities
- High intra-class variation
- Balanced distribution

Our subset:

- 50 identities
- 300–350 images each (total ≈16,000 images)
- Cleaned and preprocessed for consistency

Datasets

LFW Dataset

(Labeled Faces in the Wild)

- ~13,000 images of 5,749 individuals
- Captured in uncontrolled settings

Problems:

- Few images per identity →overfitting
- Imbalanced classes → biased models
- Poor generalization for deep models

Only good for face verification, not identification
Initially used for prototyping

VGGFace2

(Final Dataset)

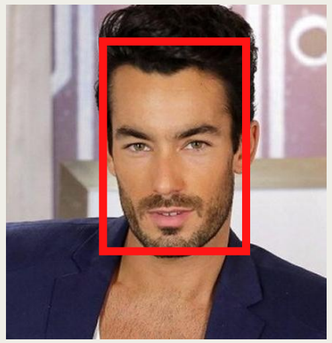
- **3.3M images across 9,131 identities**
- **High intra-class variation**
- **Balanced distribution**

Our subset:

- **50 identities**
- **300–350 images each (total ≈16,000 images)**
- **Cleaned and preprocessed for consistency**

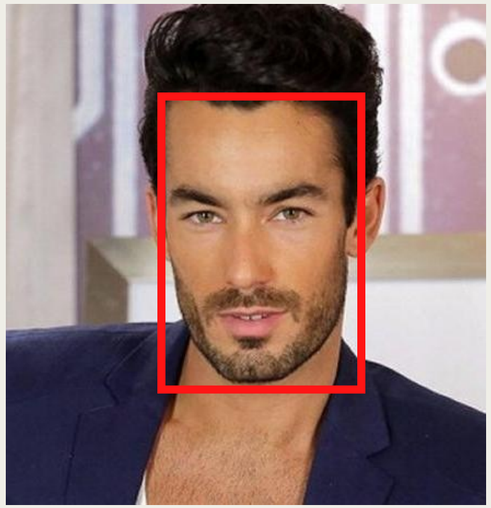
Preprocessing Pipeline

Original Image

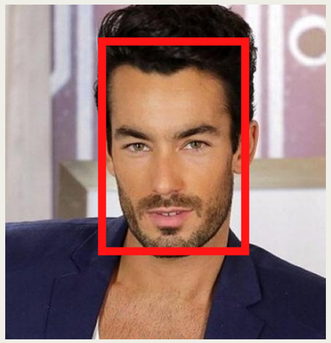


Preprocessing Pipeline

Detected bounding box overlay



Preprocessing Pipeline



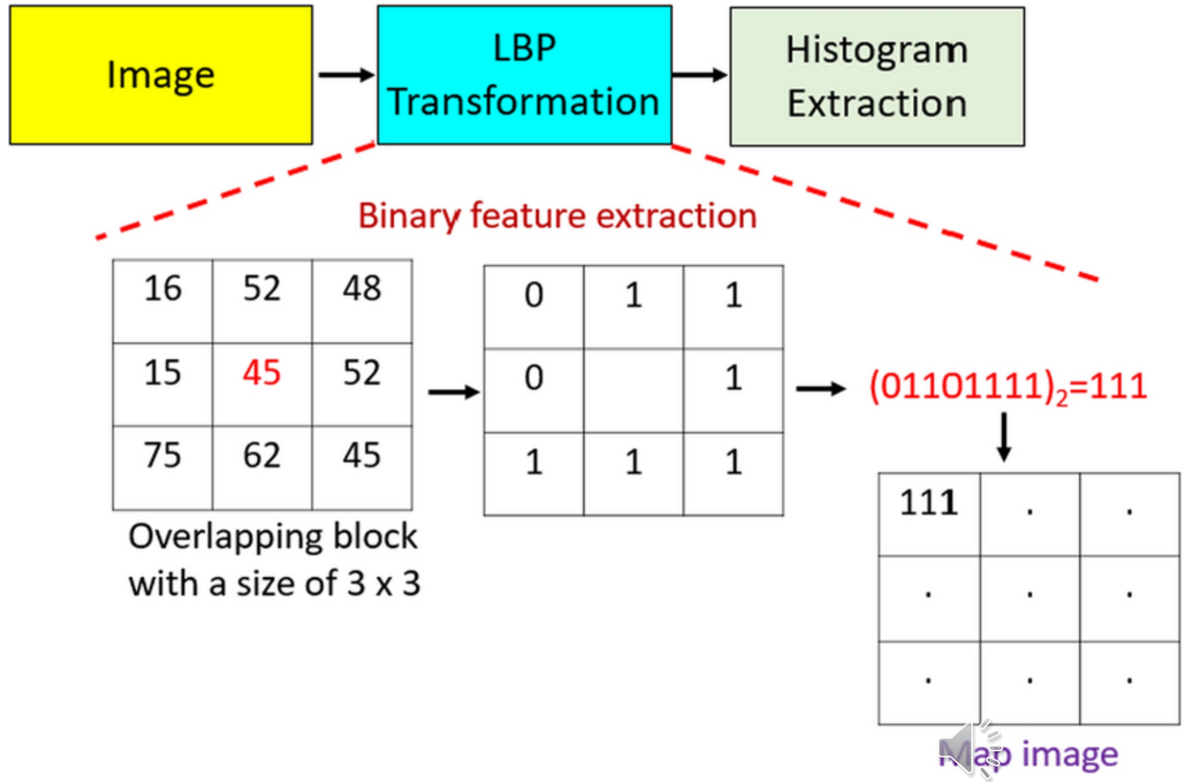
Cropped & resized face Image

Feature Extraction Techniques

Convert face images into numerical feature vectors that models can learn from. Each method captures different aspects of facial information.

1 LBP (Local Binary Patterns)

- Captures local texture patterns
- Robust to lighting variation
- Outputs: Histogram of binary patterns
- Works on grayscale images
- Fast but limited context

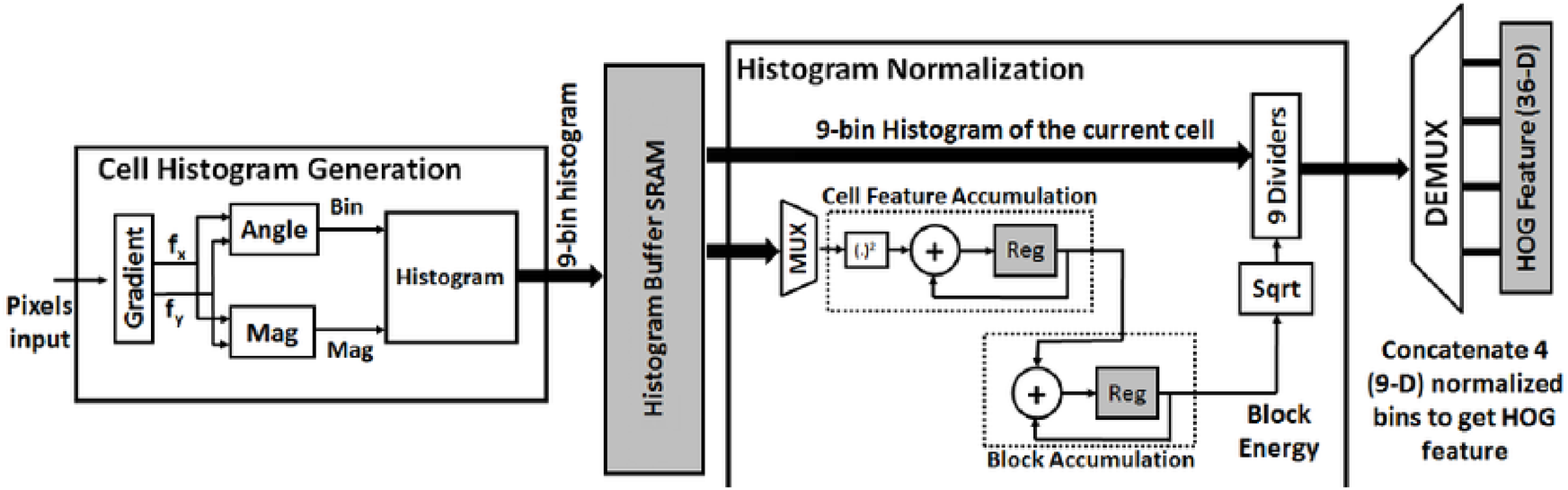


Feature Extraction Techniques

Convert face images into numerical feature vectors that models can learn from. Each method captures different aspects of facial information.

2 HoG (Histogram of Oriented Gradients)

- Captures edge orientation & structure
- Divides image into cells, computes gradient histograms
- Good for shape information
- Lacks semantic detail

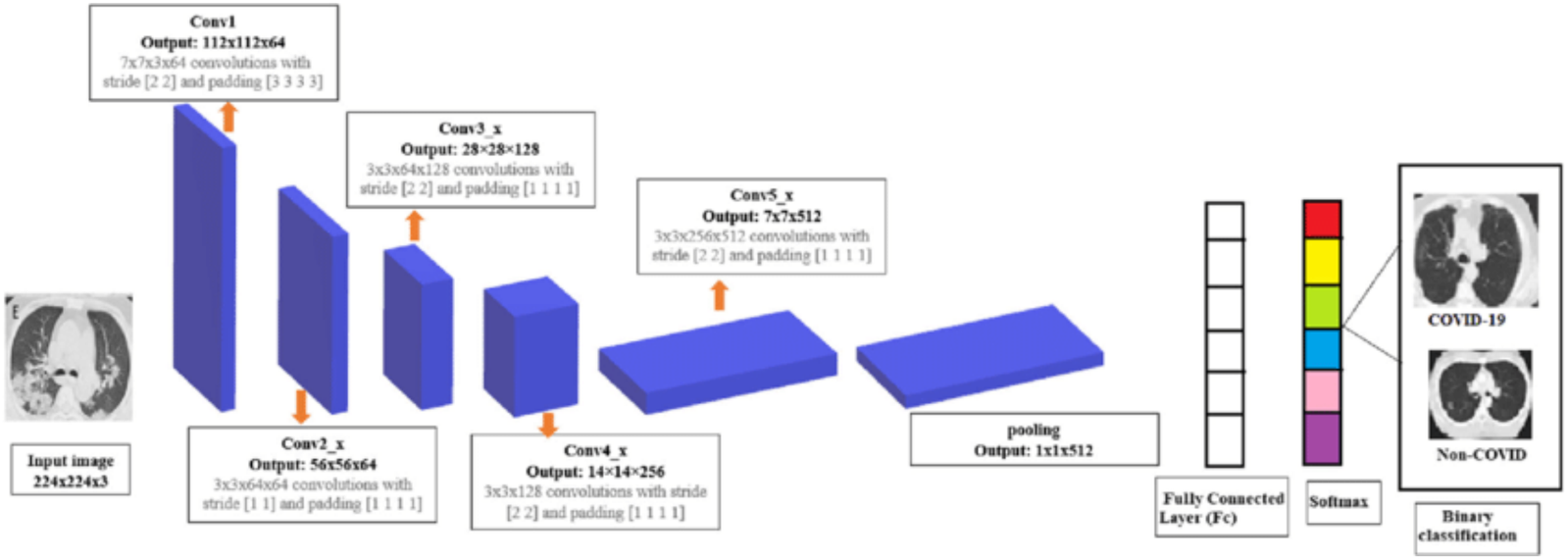


Feature Extraction Techniques

Convert face images into numerical feature vectors that models can learn from. Each method captures different aspects of facial information.

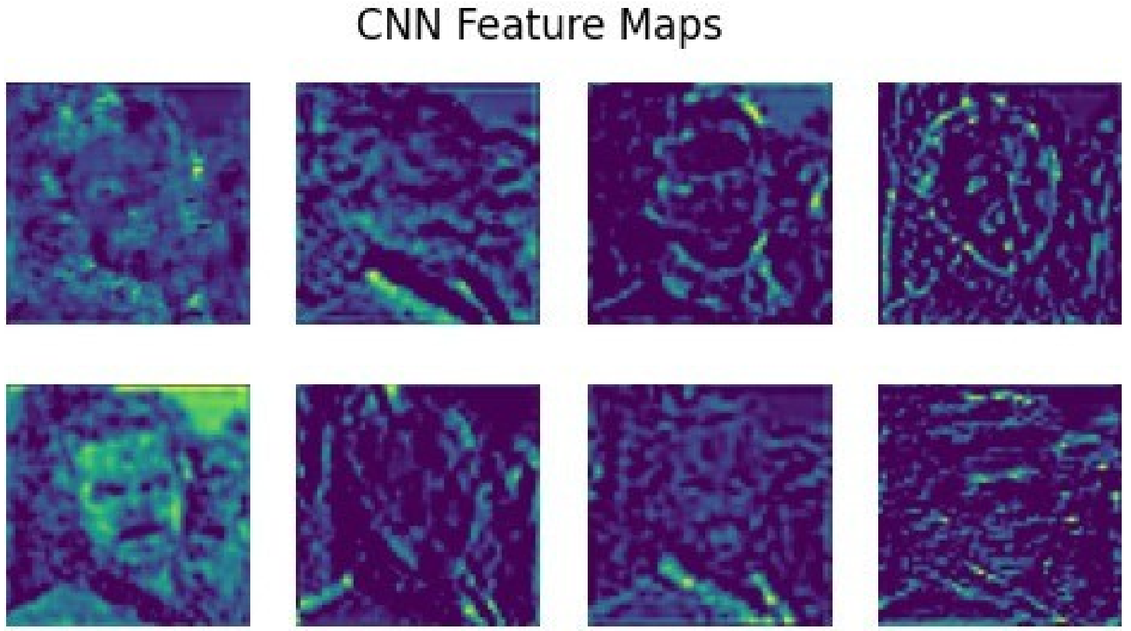
3 CNN Features (ResNet18)

- Deep semantic feature embeddings
- Pretrained on large datasets
- Outputs: 512-dimensional vector
- Most accurate & robust , uses RGB images



Feature Extraction Techniques

Technique	Sample Image	Output
LBP	Cropped face	LBP pattern histogram (grayscale overlay)
HoG	Same face	Gradient cell overlay visualization
CNN	Same face	Activation heatmap / embedding visualization (e.g., PCA projected)

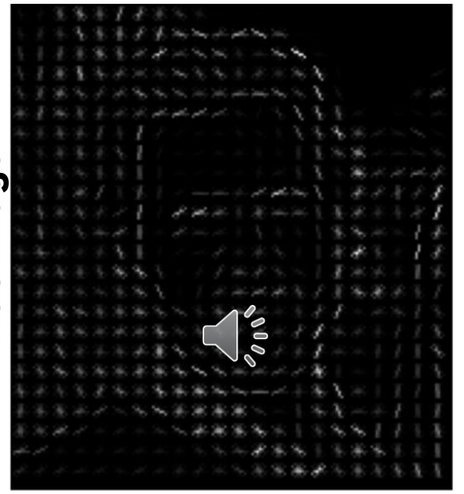


Original Image

LBP Image

Original Image

HoG Image



Feature Fusion

Concatenation: LBP + HoG + CNN vector

Why ?

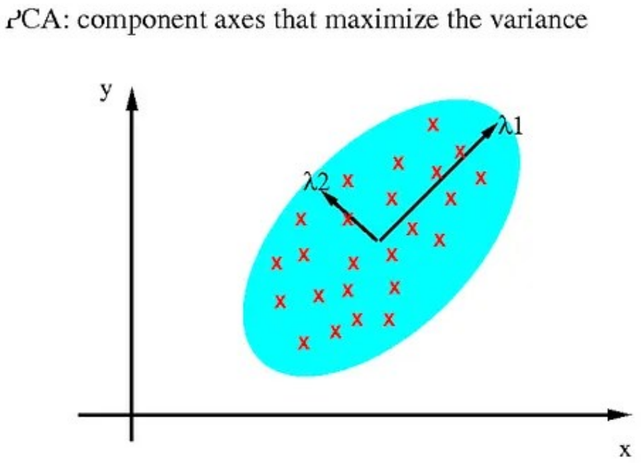
- LBP captures textures
- HoG adds shape details
- CNN provides deep semantics

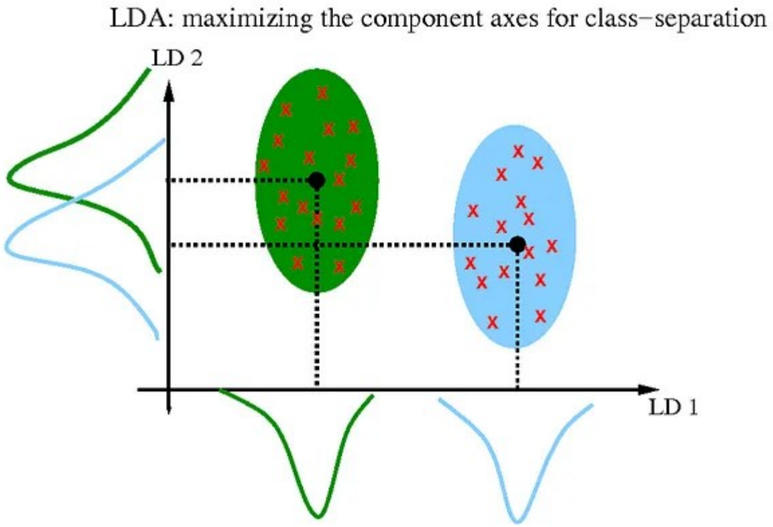
→ Fusion creates a rich representation of each face

Dimensionality Reduction Techniques:

PCA (Principal Component Analysis)

- Projects data to lower dimensions
- Keeps variance
- Improves training time





LDA (Linear Discriminant Analysis)

- Supervised reduction
- Maximizes class separation
- Better for classification

Classification Techniques

How We Classified the Faces ?

Once we had our feature vectors (from LBP, HoG, CNN, or combined), we experimented with various machine learning and deep learning classifiers to predict the correct identity.

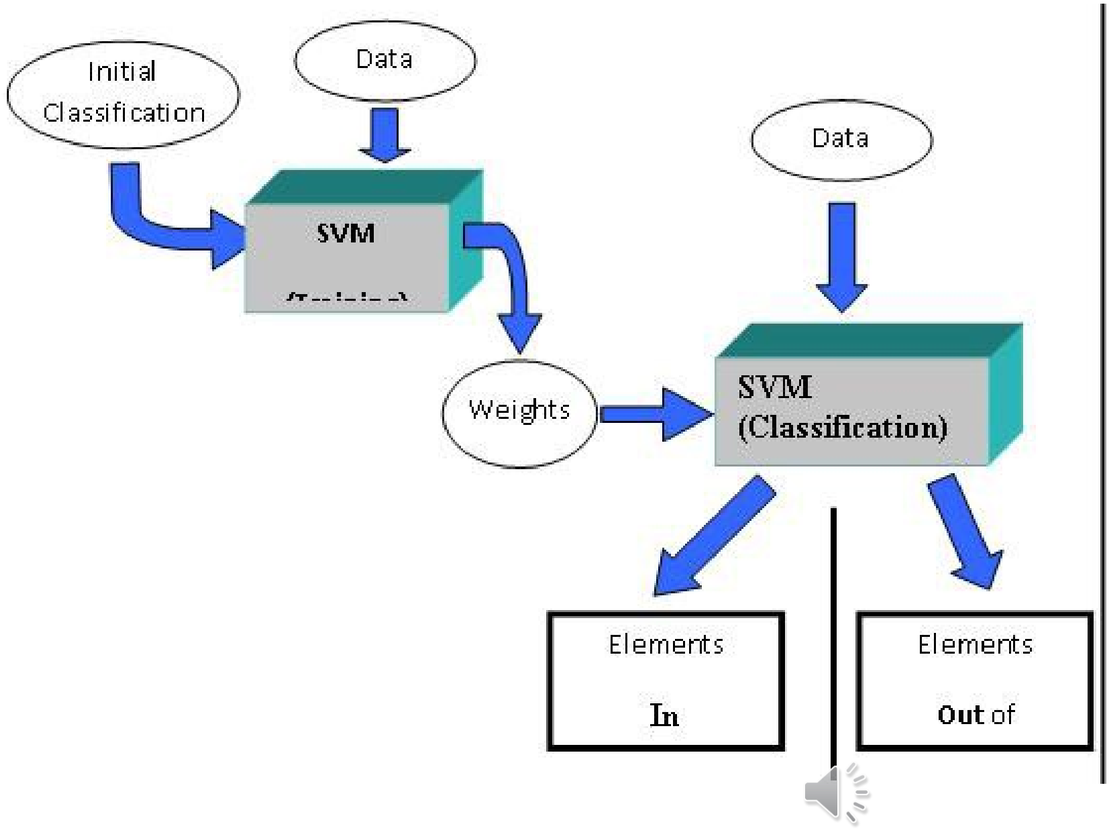
Traditional ML Models (using features):

- SVM (Support Vector Machine)
- KNN (K-Nearest Neighbors)
- Logistic Regression
- Random Forest & XGBoost
- ANN (Shallow Neural Network)

Classification Techniques

1. SVM (Support Vector Machine)

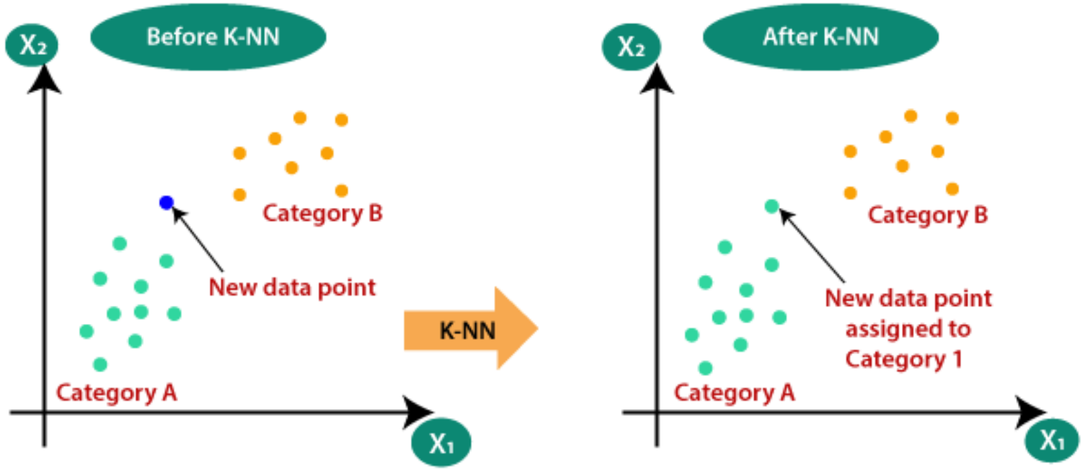
- Best performer on extracted features
- Handles high-dimensional data well
- Kernel used: RBF
- Accuracy: 76%



Classification Techniques

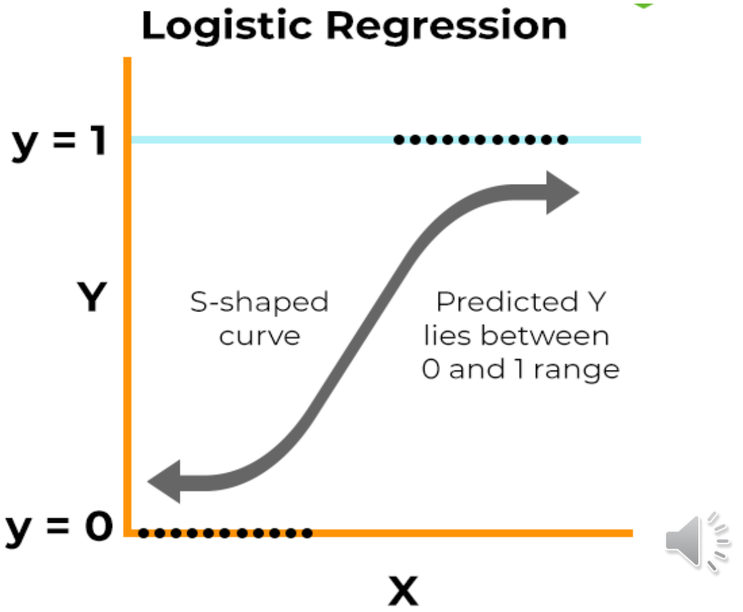
2. KNN (K-Nearest Neighbors)

- Distance-based, simple
- Accuracy: 58%



3. Logistic Regression

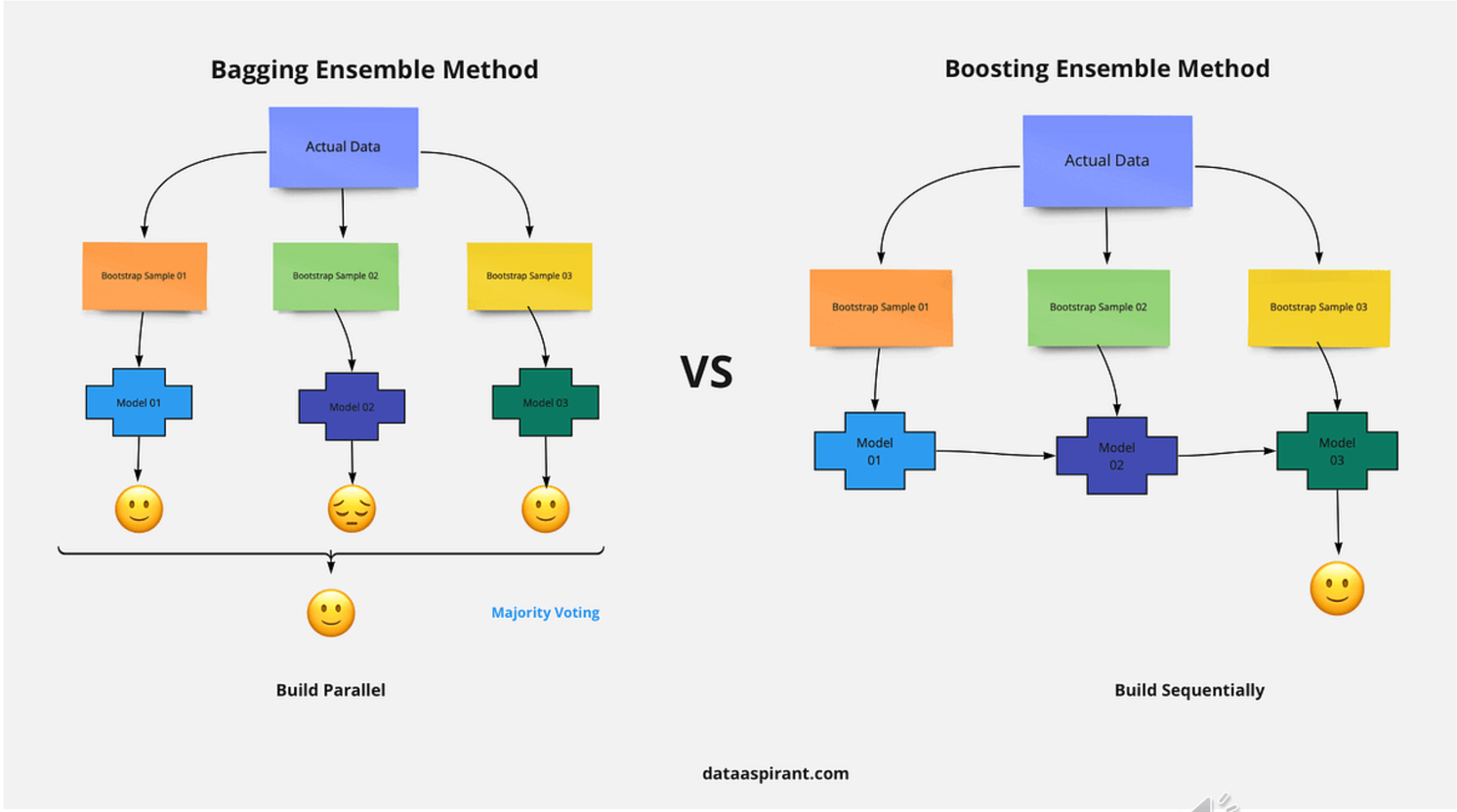
- Probabilistic model
- Accuracy: 66%



Classification Techniques

4. Random Forest & XGBoost :

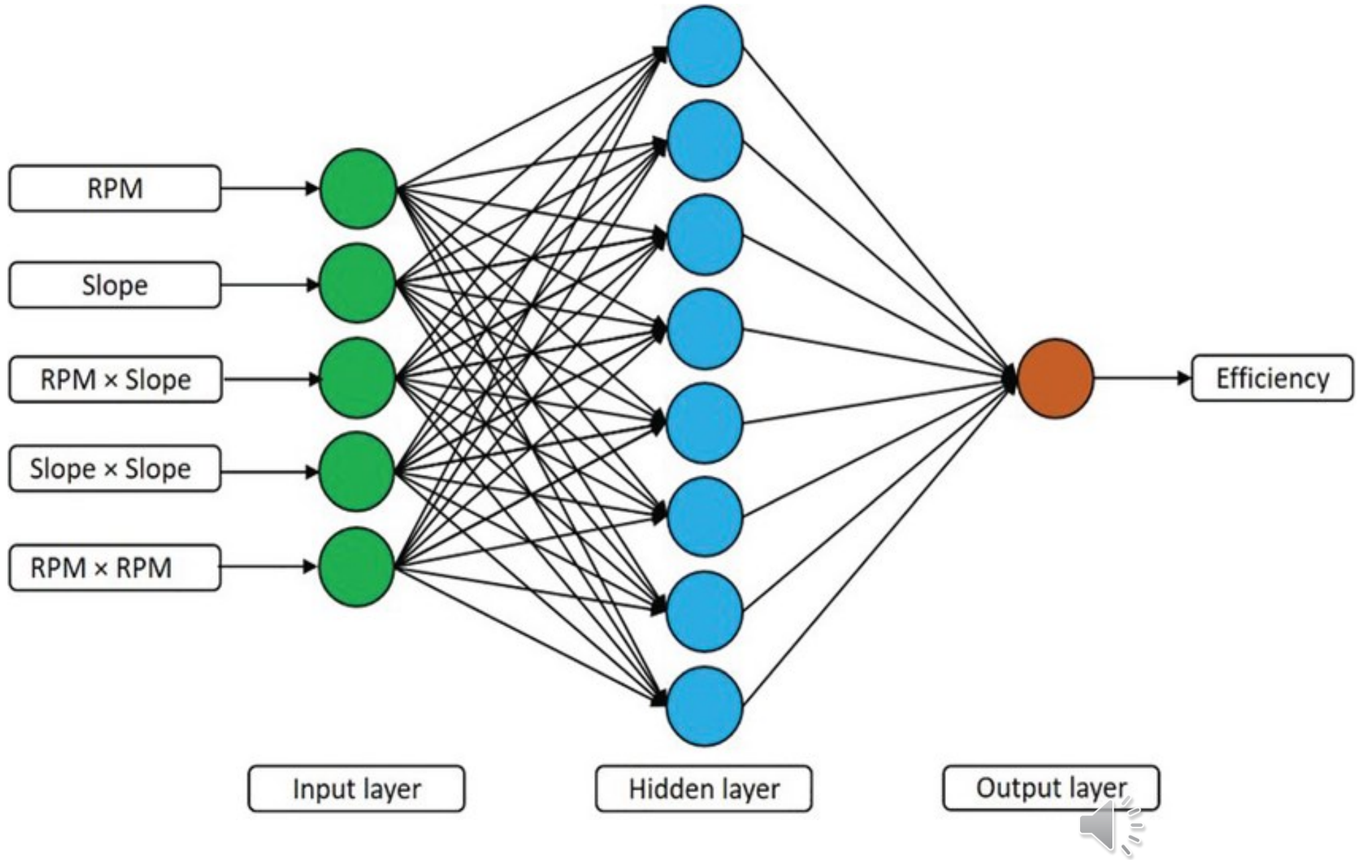
- Ensemble-based
- Struggled with high-dimensional fused features
- RF Accuracy: 57%, XGBoost: 63%



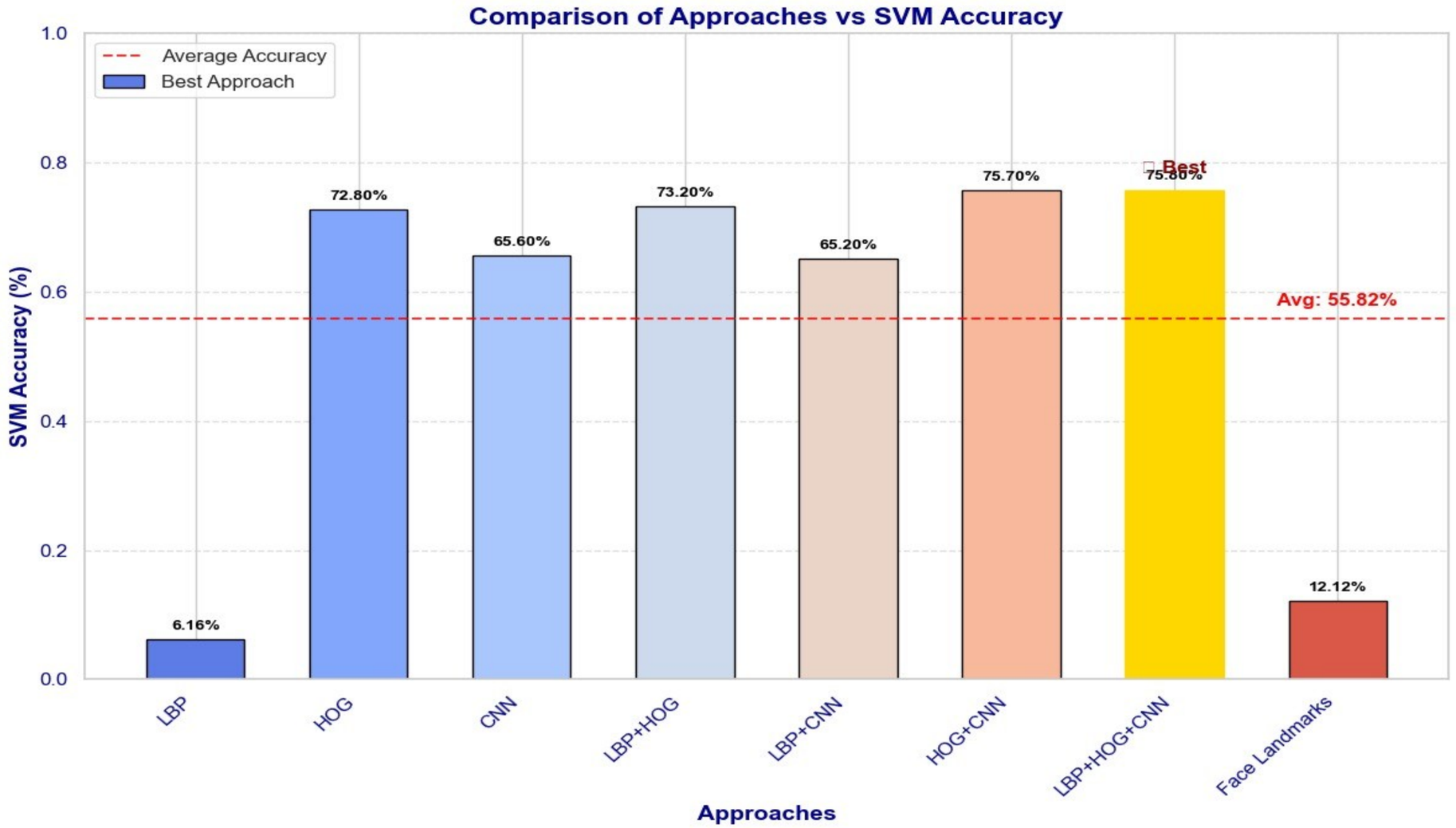
Classification Techniques

5. ANN (Artificial Neural Network)

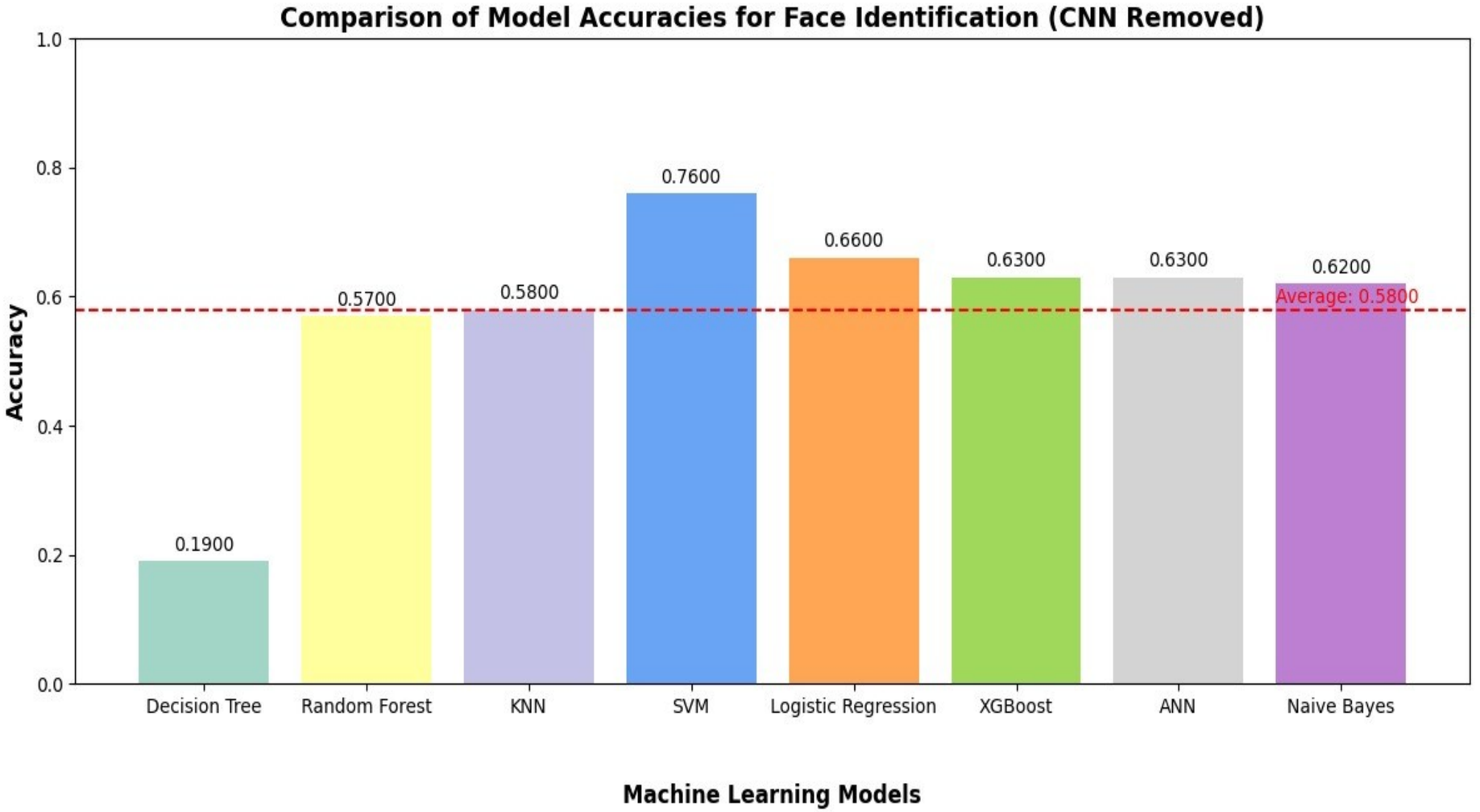
- 2 hidden layers with ReLU
- Accuracy: 63%



Accuracy with Different Features (using SVM):



Accuracy with Different ML Models:



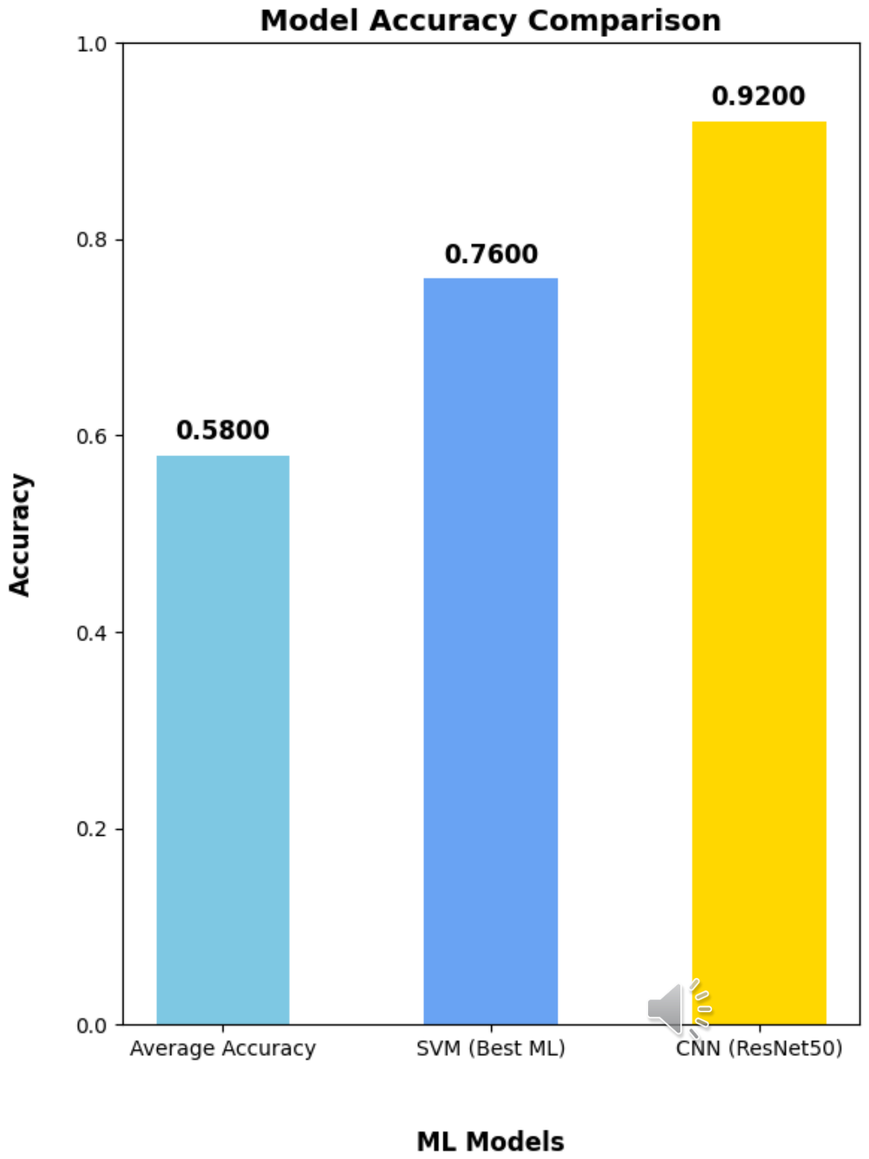
Best Approach & Trade-Offs

Actual Best Model Implemented:

Support Vector Machine (SVM) – Achieved **76% accuracy**, outperforming other classical ML models.

Top performer Model: ResNet50 (92% Accuracy)

ResNet50 showed the highest simulated accuracy at 92%, but was not implemented in our project.



Tradeoffs :

SVM (Used):

✓ Simple, fast, interpretable
✗ Lower accuracy on complex data

ResNet50 (not used):

✓ Highest accuracy (92%), learns features automatically
✗ Complex, needs more data & compute

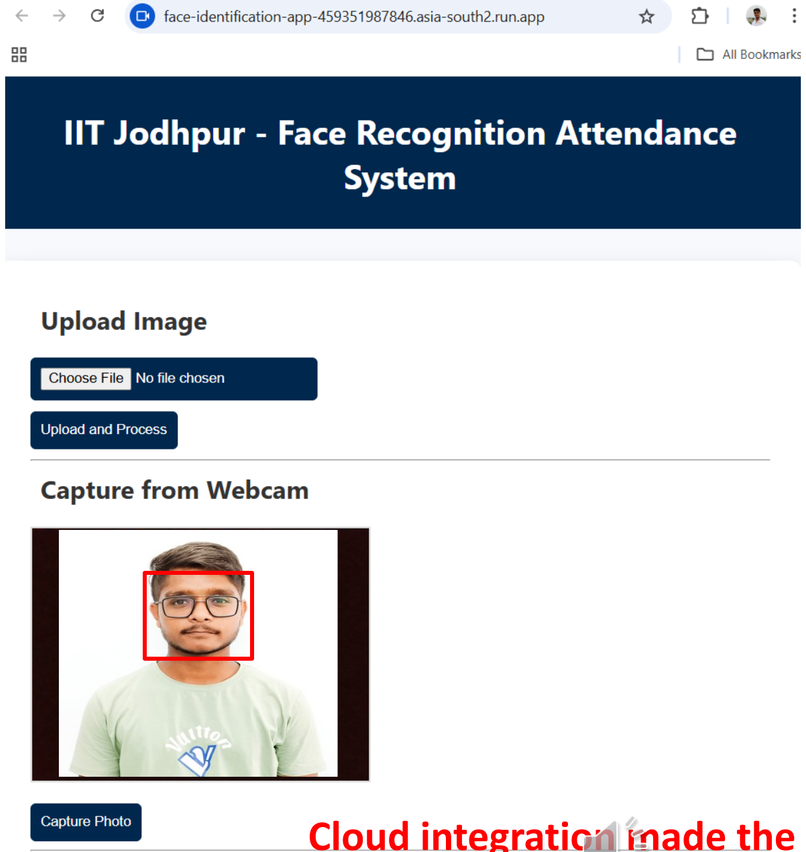
Final Summary & Deployment

System Overview:

- Real-time face recognition attendance using SVM
- Features extracted via LBP, HoG, and simulated CNN
- GUI for easy user interaction and attendance marking

Google Cloud Deployment:

- **Firebase**: Real-time attendance updates & secure database
- **Cloud Storage**: Stores models & face data for fast access
- **Web Hosting**: Deployed project interface on GCP
- **Scalable & remotely accessible**



Cloud integration made the system scalable, fast, and accessible from anywhere.

☺ Thankyou ! ☺