

0 0 0 0

LỚP CS231.Q11

NHẬP MÔN THỊ GIÁC MÁY TÍNH

BÁO CÁO ĐỒ ÁN: PHÂN LOẠI KHUÔN MẶT ĐEO KHẨU TRANG



Thành viên:
Nguyễn Công Phát - 23521143
Nguyễn Lê Phong - 23521168
Vũ Việt Cương - 23520213

0 0 0 0

NỘI DUNG

I/ LÝ DO THỰC HIỆN BÀI TOÁN

II/ PHÁT BIỂU BÀI TOÁN

III/ PHƯƠNG PHÁP THỰC HIỆN

IV/ THỰC NGHIỆM

V/ DEMO



0000

LÍ DO CHỌN ĐỀ TÀI

- Bối cảnh:**

- Trong các môi trường nhiều nguy cơ như khu công nghiệp, nhà máy, xưởng sản xuất hoặc bối cảnh dịch bệnh truyền nhiễm, đường hô hấp của công nhân luôn chịu nhiều rủi ro.
- Giải pháp cơ bản nhất và rẻ nhất là đeo khẩu trang.

- Mục tiêu:**

- Xây dựng mô hình phân loại khuôn mặt, tích hợp vào camera giám sát để tự động phát hiện công nhân có đeo khẩu trang hay không.
- Đánh giá độ hiệu quả của các kỹ thuật trích xuất đặc trưng (HOG, LBP) và mô hình học máy khác nhau..

III/ PHÁT BIỂU BÀI TOÁN

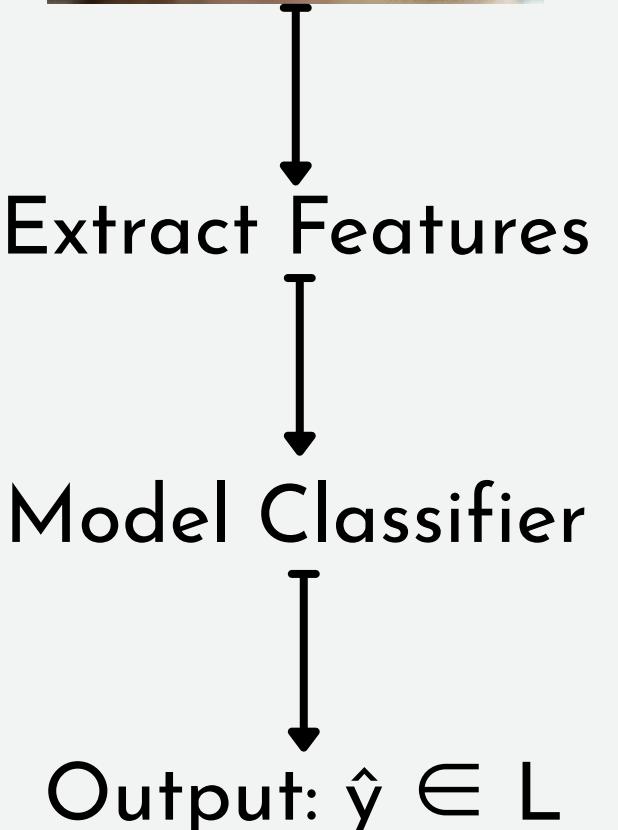
Input:

- Tập dữ liệu huấn luyện N phần tử: $D = \{(x_i, y_i)\}$ với i chạy từ 1 đến N .
- Trong đó $x_i \in \mathbb{R}^{(128 \times 128 \times 1)}$: Ảnh đầu vào đã chuẩn hóa kích thước.
- Tập nhãn $L = \{0: \text{WithoutMask}, 1: \text{WithMask}\}$.
- Ảnh x có chứa khuôn mặt

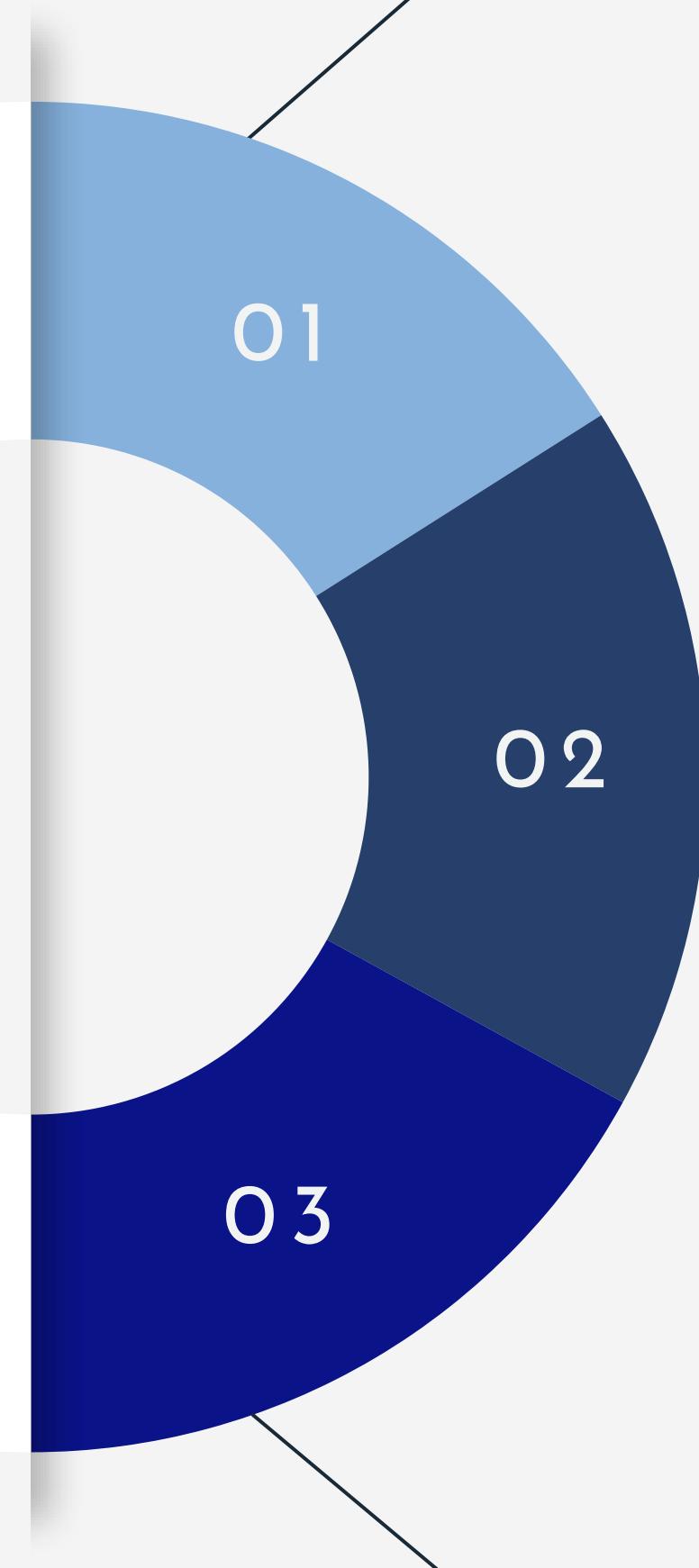
Output:

- $\hat{y} \in L$: Giá trị nhãn dự đoán của ảnh x

INPUT: Ảnh có chứa khuôn mặt



III/ PHƯƠNG PHÁP THỰC HIỆN



TRÍCH XUẤT ĐẶC TRƯNG

CÁC MÔ HÌNH
MÁY HỌC

PHƯƠNG PHÁP TỐI
ƯU SIÊU THAM SỐ

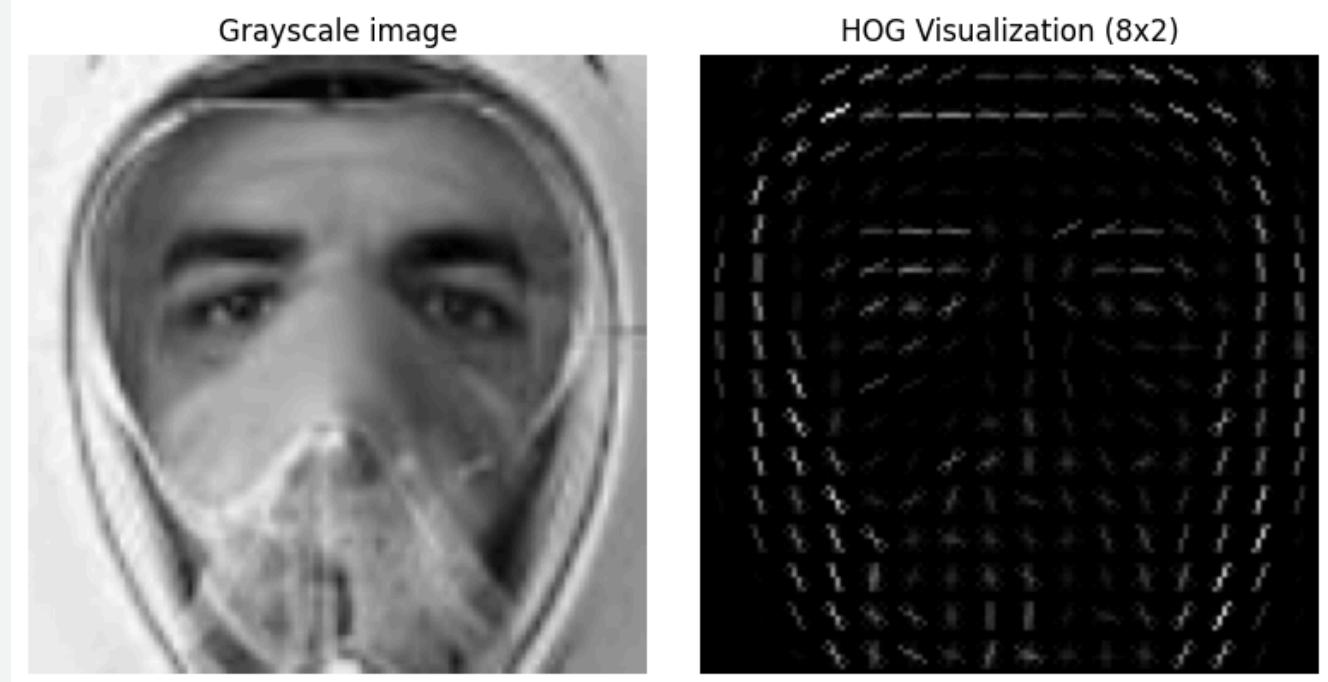
ĐẶC TRƯNG HOG (HISTOGRAM OF ORIENTED GRADIENTS)

1. Định nghĩa

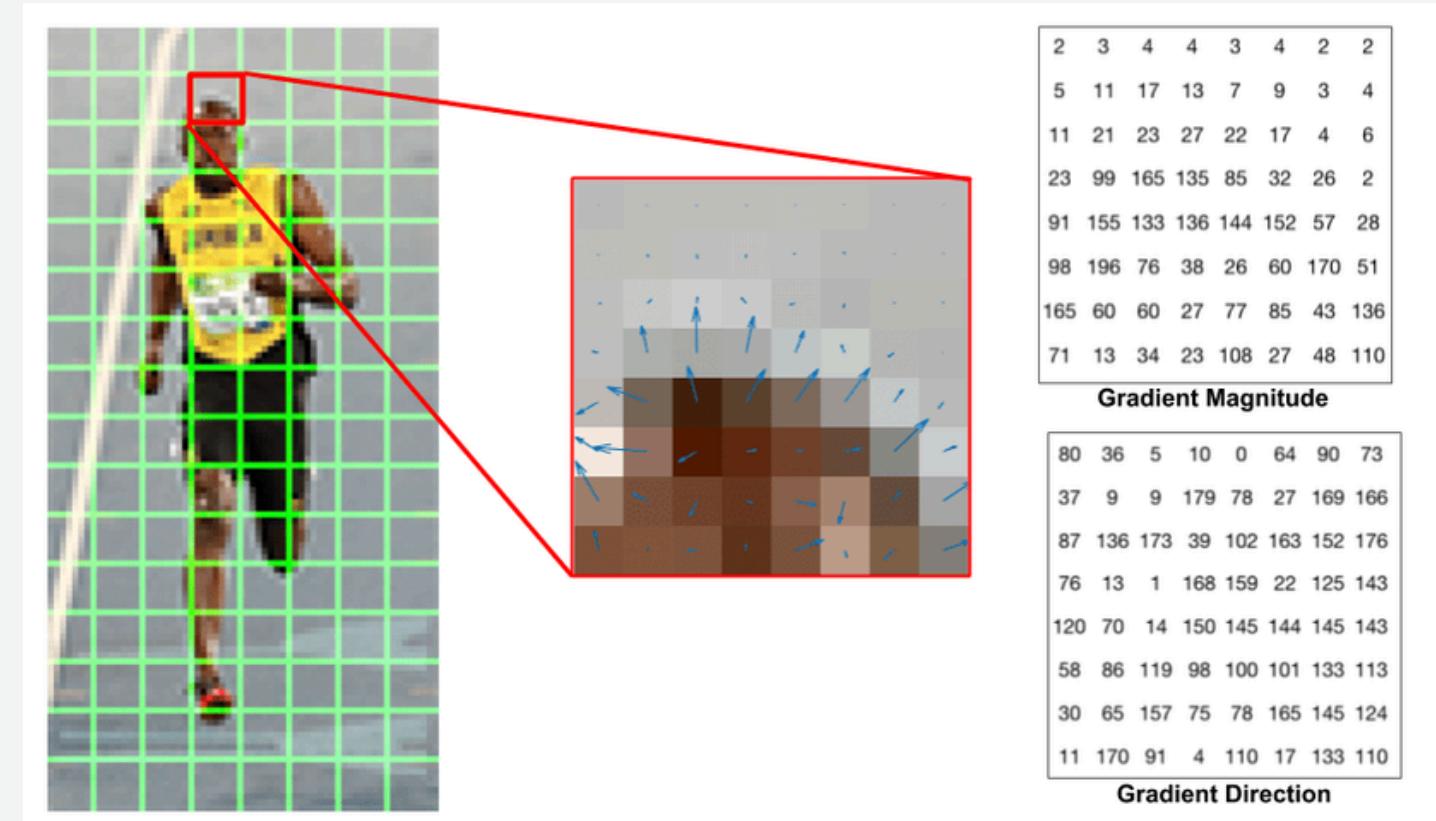
HOG là đặc trưng mô tả hình dạng của đối tượng bằng cách thống kê hướng cạnh (gradient orientations) trong từng vùng nhỏ của ảnh.

2. Nguyên lý hoạt động

- **Tính Gradient:** Tính độ lớn & hướng của gradient tại mỗi pixel.
- **Chia ảnh thành ô (Cell):** Chia ảnh thành các vùng nhỏ (ví dụ: 8x8 pixels).
- **Tạo Histogram hướng:** Mỗi cell tạo histogram hướng (9 bins) để mô tả hướng cạnh cục bộ.
- **Chuẩn hóa theo khối (Block Normalization):** Gom nhiều cell thành 1 block và chuẩn hóa để giảm nhiễu ánh sáng.



Hình 1. Trục quan hóa HOG Feature trên ảnh Grayscale



Hình 2. Trục quan hóa nguyên lý hoạt động

CÀI ĐẶT HOG TRONG BÀI TOÁN

1. Cấu hình chung

- **Input:** Ảnh được Resize về 128×128 , chuyển sang Grayscale.
- **Orientations:** 9 hướng
- **Chuẩn hóa:** L2-Hys



Hình 3. Quy trình thực hiện cài đặt HOG trong bài toán

2. Hai cấu hình HOG nhóm sử dụng

```
IMAGE_SIZE = (128, 128)

# HOG parameters (6x3)
HOG_PARAMS = {
    "orientations": 9,
    "pixels_per_cell": (6, 6),
    "cells_per_block": (3, 3),
    "block_norm": "L2-Hys"
}
```

Hình 4. HOG 6x3

```
IMAGE_SIZE = (128, 128)

# HOG parameters (8x2)
HOG_PARAMS = {
    "orientations": 9,
    "pixels_per_cell": (8, 8),
    "cells_per_block": (2, 2),
    "block_norm": "L2-Hys"
}
```

Hình 5. HOG 8x2

- So sánh mức độ chi tiết → đánh giá tác động của cell/block
- Tìm cấu hình HOG tối ưu cho bài toán

ĐẶC TRƯNG LBP (LOCAL BINARY PATTERN)

1. Định nghĩa

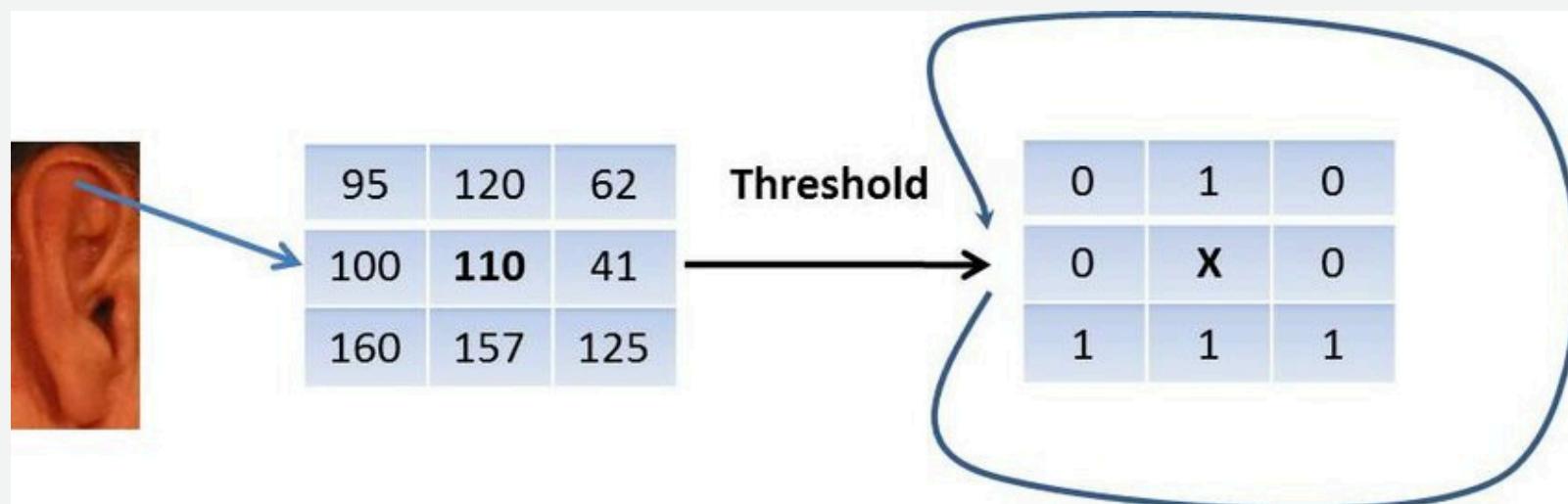
LBP là đặc trưng mô tả kết cấu (texture) của ảnh bằng cách so sánh giá trị pixel trung tâm với các pixel lân cận để tạo mã nhị phân.

2. Nguyên lý hoạt động:

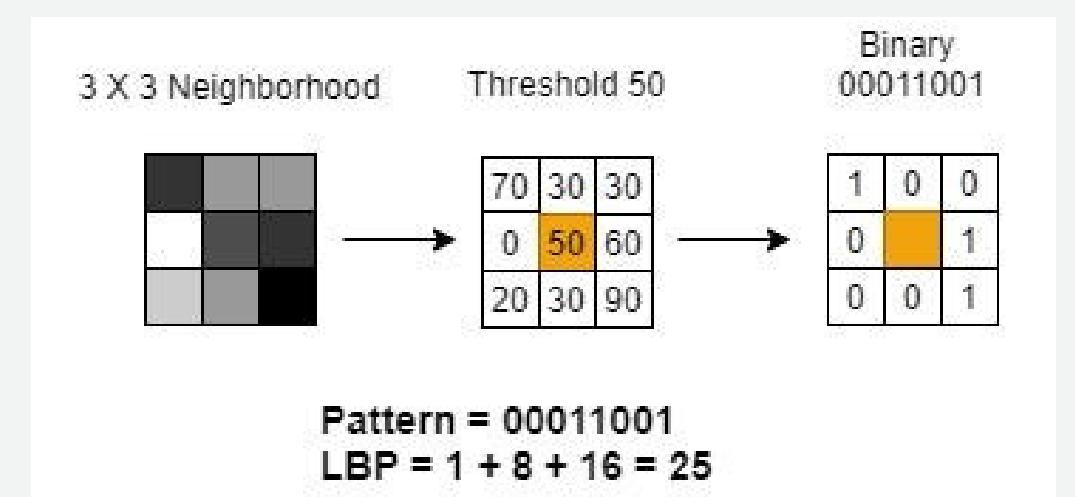
- So sánh pixel:** So sánh pixel trung tâm với P điểm lân cận trong bán kính R.
- Tạo mã nhị phân:** Lân cận \geq Trung tâm $\rightarrow 1$, Ngược lại $\rightarrow 0$. Chuỗi bit này là giá trị LBP.
- Tạo Histogram LBP:** Thống kê tần suất các mã LBP xuất hiện trong vùng/ảnh để biểu diễn kết cấu (texture).
- LBP Uniform ($P=8, R=1$):** Chỉ giữ các pattern có ≤ 2 lần chuyển trạng thái $0 \leftrightarrow 1$, giúp giảm chiều và ổn định.



Hình 6. Trực quan hóa LBP Feature trên ảnh Grayscale



Hình 7. Basic LBP operator - [Link](#)



Hình 8. Illustrates the example of the LBP algorithm - [Link](#)

CÀI ĐẶT LBP TRONG BÀI TOÁN

1. Cấu hình LBP trong bài toán

```
IMAGE_SIZE = (128, 128)  
  
LBP_RADIUS = 1  
LBP_POINTS = 8 * LBP_RADIUS  
LBP_METHOD = "uniform"
```

Hình 10. Cấu hình LBP

Giải thích:

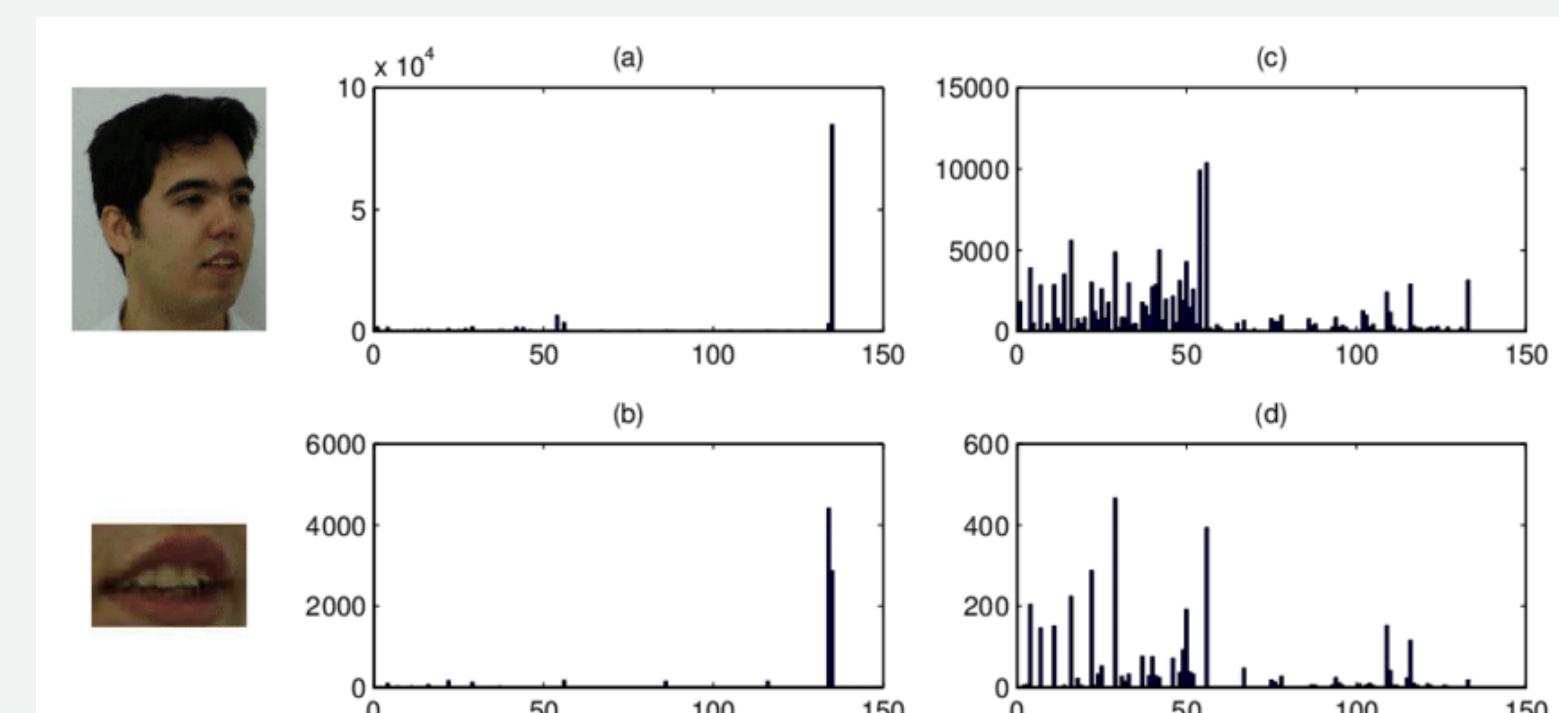
- R=1, P=8: Mô tả kết cấu cục bộ tốt, phù hợp với ảnh khuôn mặt.
- 'uniform': Giảm chiều vector, ổn định và ít nhạy cảm với nhiễu ánh sáng đơn điệu.

2. Ưu điểm LBP

- Giỏi phân biệt Texture (kết cấu) giữa da người và vải khẩu trang.
- Tính toán nhanh, phù hợp với dataset lớn.
- Ổn định với thay đổi sáng đơn điệu (Illumination Invariant).

3. Nhược điểm LBP

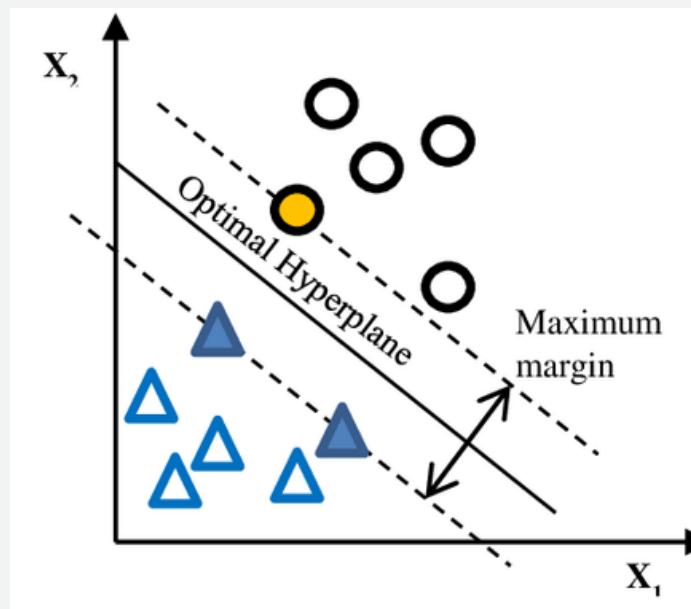
- Không mô tả hình dạng tổng thể (shape) như HOG.
- Nhạy với nhiễu và mờ (blur) nếu ảnh không rõ.
- Kém hiệu quả nếu khẩu trang có kết cấu quá trơn



Hình 11. Minh họa - [Link](#)

CÁC MÔ HÌNH HỌC MÁY

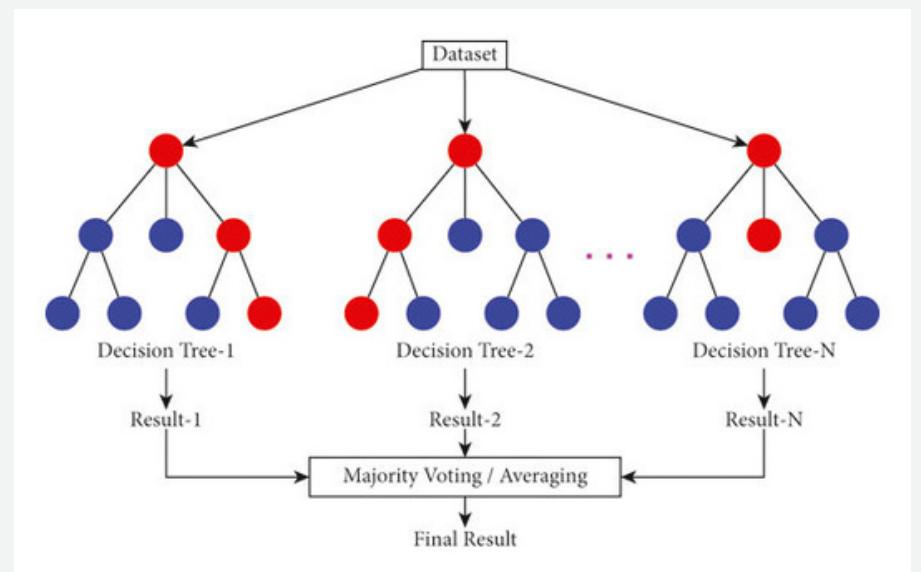
1. SVM (Support Vector Machine): Tìm siêu phẳng tối ưu để phân tách 2 lớp dữ liệu.



- Tìm siêu phẳng tối ưu tách hai lớp với margin lớn nhất.
- Kernel (RBF, Linear) giúp xử lý dữ liệu phi tuyến.

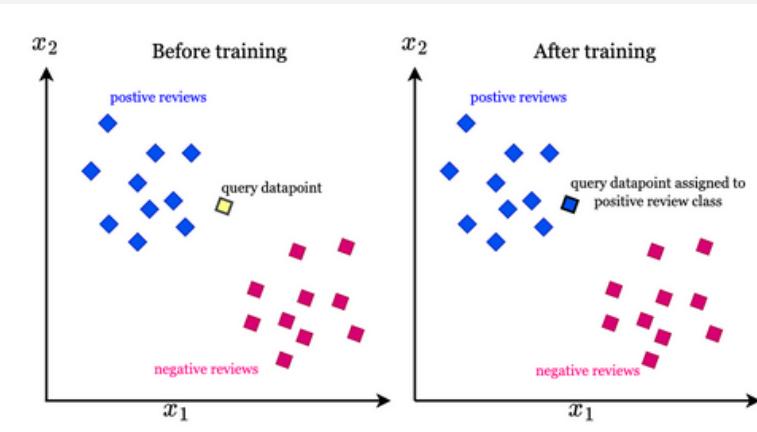
Hình 12. the maximum margin hyperplane of SVM- [Link](#)

2. Random Forest (Rừng ngẫu nhiên): Sử dụng kỹ thuật Bagging với nhiều cây quyết định.



Hình 13. Illustration of random forest trees. - [Link](#)

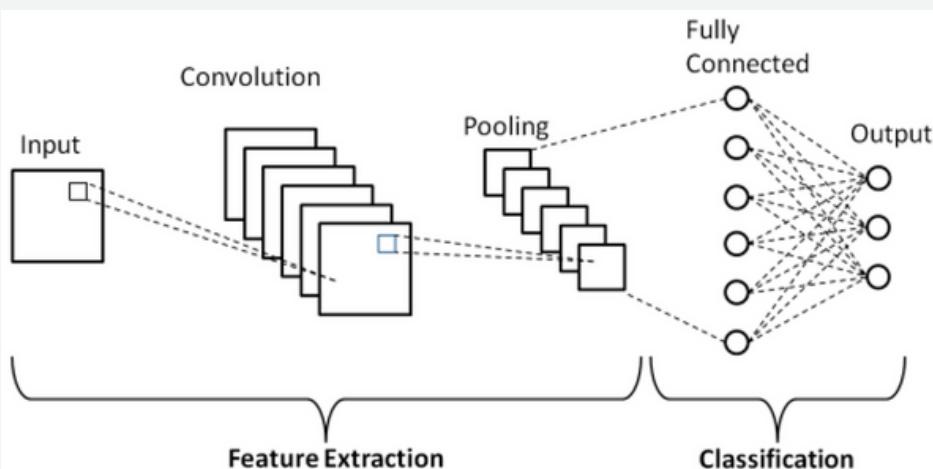
3. KNN (K-Nearest Neighbors): Phân loại dựa trên khoảng cách đến các điểm dữ liệu gần nhất



- Tìm **k** điểm gần nhất trong không gian đặc trưng.
- Dự đoán theo voting của các hàng xóm.

Hình 14. KNN - [Link](#)

4. CNN: dùng phép tích chập (convolution) để tự động trích xuất đặc trưng từ dữ liệu đầu vào



Hình 15. Basic CNN architecture - [Link](#)

- Mỗi cây vote → kết quả cuối là majority vote.
- Giảm overfitting so với Decision Tree đơn lẻ.

PHƯƠNG PHÁP TỐI ƯU SIÊU THAM SỐ - OPTUNA & KERAS TUNER

1. Optuna

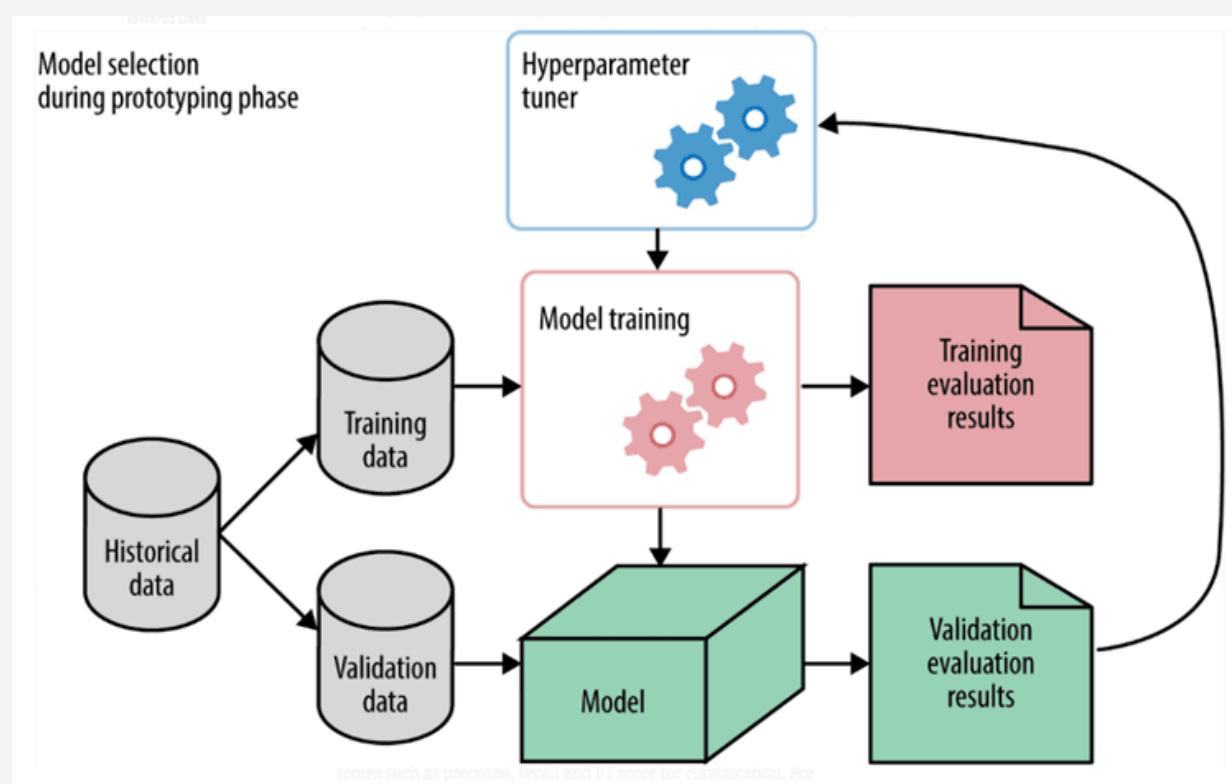
Nguyên lý hoạt động:

- Dùng chiến lược Bayesian Optimization (sampler TPE) để tìm tham số tối ưu.
- Cải thiện tốc độ nhờ pruning (bỏ sớm mô hình không tiềm năng).
- Tự động khám phá không gian tham số → hiệu quả hơn Grid Search / Random Search.

Ưu điểm: nhanh, hiệu quả, tìm tham số tốt.

Nhược điểm: cài đặt phức tạp hơn.

Ứng dụng: tuning SVM, RF, KNN.



Hình 16. Optuna - [Link](#)

2. Keras Tuner

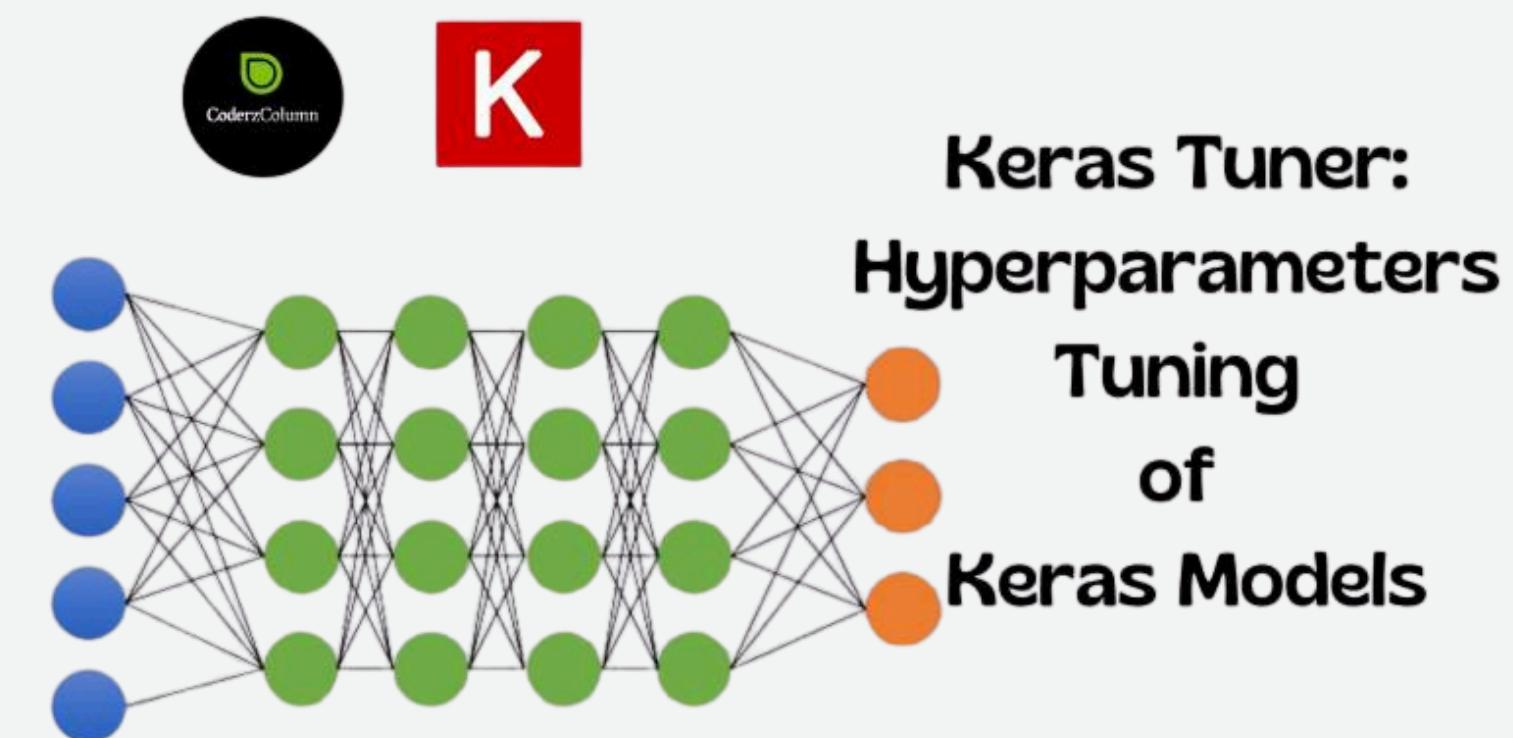
Nguyên lý hoạt động:

- Tìm cấu hình CNN tốt nhất bằng thuật toán Hyperband, RandomSearch, BayesianOptimization.
- Tự sinh nhiều kiến trúc CNN khác nhau: số filters, dense units, dropout, learning rate.
- Dừng sớm trial kém để tiết kiệm thời gian.

Ưu điểm: mạnh cho Deep Learning.

Nhược điểm: tốn GPU.

Ứng dụng: tuning filters, dense units, dropout, LR cho CNN.



Hình 17. Keras Tuner - [Link](#)

OPTUNA + SVM

```
def objective(trial):
    params = {
        "C": trial.suggest_float("C", 1e-2, 1e2, log=True),
        "kernel": trial.suggest_categorical("kernel", ["linear", "rbf"]),
        "gamma": trial.suggest_categorical("gamma", ["scale", "auto"]),
        "probability": False, # that probability đe speed-up
        "random_state": 42,
    }

    clf = SVC(**params)
    clf.fit(X_train, y_train_enc)
    pred = clf.predict(X_val)
    return accuracy_score(y_val_enc, pred)

study = optuna.create_study(
    direction="maximize",
    pruner=MedianPruner(n_startup_trials=5),
    sampler=TPESampler(multivariate=True),
)

study.optimize(objective, n_trials=30, n_jobs=1)

print("Best validation accuracy:", study.best_value)
print("Best params:", study.best_params)
```

```
def objective(trial):
    kernel = trial.suggest_categorical("kernel", ["linear", "poly", "rbf"])
    C = trial.suggest_categorical("C", [1, 5, 10])

    if kernel == "poly":
        gamma = trial.suggest_float("gamma_poly", 0.01, 1.0)
        degree = trial.suggest_int("degree", 3, 15)
    else:
        gamma = trial.suggest_categorical("gamma", ["scale", "auto"])

    model = SVC(kernel=kernel, C=C, gamma=gamma)
    model.fit(X_train, y_train)
    pred = model.predict(X_val)
    return accuracy_score(y_val, pred)

study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=50)
```

1. HOG + SVM (Optuna tuning)

→ Ý nghĩa: Tối ưu nhanh, nhẹ, tuning SVM theo 3 tham số chính (C, kernel, gamma).

2. LBP + SVM (Conditional Optuna tuning)

→ Ý nghĩa: Nếu kernel thay đổi thì không gian tham số cũng thay đổi

OPTUNA + KNN

```
def objective(trial):
    # Không gian tham số cho KNN
    params = {
        # số láng giềng, thử số lẻ 1-31 để tránh tie
        "n_neighbors": trial.suggest_int("n_neighbors", 1, 31, step=2),
        # uniform: mọi hàng xóm nhau, distance: hàng xóm gần hơn trọng số lớn hơn
        "weights": trial.suggest_categorical("weights", ["uniform", "distance"]),
        # p = 1: Manhattan, p = 2: Euclidean
        "p": trial.suggest_int("p", 1, 2),
        # metric để mặc định minkowski (dùng chung với p)
        "metric": "minkowski",
    }

    clf = KNeighborsClassifier(**params)
    clf.fit(X_train, y_train_enc)

    y_val_pred = clf.predict(X_val)
    acc = accuracy_score(y_val_enc, y_val_pred)
    return acc

study_knn = optuna.create_study(
    direction="maximize",
    pruner=MedianPruner(n_startup_trials=5, n_warmup_steps=1),
    sampler=TPESampler(multivariate=True),
)

study_knn.optimize(objective, n_trials=30, n_jobs=1)

print("[KNN] Best validation accuracy:", study_knn.best_value)
print("[KNN] Best params:", study_knn.best_params)
```

Hình 22. HOG + KNN + Optuna

I. KNN + HOG + Optuna

→ **Ý nghĩa:** Optuna tối ưu KNN trên đặc trưng HOG: điều chỉnh số láng giềng, trọng số và chuẩn khoảng cách; sử dụng MedianPruner để dừng sớm trial kém.

```
def objective(trial):
    # Không gian tham số cho KNN
    params = {
        # số láng giềng lẻ để tránh tie
        "n_neighbors": trial.suggest_int("n_neighbors", 1, 31, step=2),
        # uniform: mọi hàng xóm nhau, distance: gần hơn -> trọng số lớn hơn
        "weights": trial.suggest_categorical("weights", ["uniform", "distance"]),
        # p = 1: Manhattan, p = 2: Euclidean
        "p": trial.suggest_int("p", 1, 2),
        # metric để mặc định = "minkowski" (dùng chung với p)
        "metric": "minkowski",
    }

    clf = KNeighborsClassifier(**params)
    clf.fit(X_train, y_train_enc)

    y_val_pred = clf.predict(X_val)
    acc = accuracy_score(y_val_enc, y_val_pred)
    return acc

study_knn = optuna.create_study(
    direction="maximize",
    pruner=MedianPruner(n_startup_trials=5, n_warmup_steps=1),
    sampler=TPESampler(multivariate=True, seed=42),
)
study_knn.optimize(objective, n_trials=30, n_jobs=1)
```

Hình 23. LBP + KNN + Optuna

2. KNN + LBP + Optuna

→ **Ý nghĩa:** Optuna tune tham số KNN trên đặc trưng LBP uniform: n_neighbors, weights và p; dùng TPE Sampler và MedianPruner để tăng tốc tìm kiếm tham số.

OPTUNA + RANDOM FOREST

```
def objective(trial):
    params = {
        'max_depth': trial.suggest_int('max_depth', 5, 50),
        'min_samples_split': trial.suggest_int('min_samples_split', 2, 20),
        'min_samples_leaf': trial.suggest_int('min_samples_leaf', 1, 10),
        'max_features': trial.suggest_categorical('max_features', ['sqrt','log2']),
        'warm_start': True, 'n_jobs': 1, 'random_state': 42
    }
    clf = RandomForestClassifier(**params)
    for n in [50, 100, 150, 200]:
        clf.set_params(n_estimators=n)
        clf.fit(x_train, y_train)
        acc = accuracy_score(y_val, clf.predict(x_val))
        trial.report(acc, n)
        if trial.should_prune(): raise optuna.TrialPruned()
    return acc
```

Hình 20. HOG + RF + Optuna

```
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 50, 300),
        'max_depth': trial.suggest_int('max_depth', 5, 50),
        'min_samples_split': trial.suggest_int('min_samples_split', 2, 20),
        'min_samples_leaf': trial.suggest_int('min_samples_leaf', 1, 10),
        'max_features': trial.suggest_categorical('max_features',['sqrt','log2',None]),
        'n_jobs': -1, 'random_state': 42
    }
    clf = RandomForestClassifier(**params)
    clf.fit(x_train, y_train)
    return accuracy_score(y_val, clf.predict(x_val))
```

Hình 21. LBP + RF + Optuna

1. HOG + RF với Warm Start + Pruning

→ Ý nghĩa: Dùng incremental training để prune trial kém sớm → giảm thời gian train.

2. LBP + RF (Standard Search Space)

→ Ý nghĩa: Vector nhỏ → chỉ cần tối ưu tham số cơ bản, không cần warm_start.

CNN + KERAS TUNER

```
def build_model(hp):
    model = tf.keras.Sequential()
    hp_filters = hp.Int('filters_1', 32, 96, step=32)
    model.add(Conv2D(hp_filters, (3,3), activation='relu'))
    model.add(MaxPool2D(2,2))

    hp_units = hp.Int('dense_units', 64, 256, step=64)
    hp_dropout = hp.Float('dropout', 0.2, 0.5, step=0.1)
    hp_lr = hp.Choice('learning_rate', [1e-2, 1e-3, 1e-4])

    model.add(Flatten())
    model.add(Dense(hp_units, activation='relu'))
    model.add(Dropout(hp_dropout))
    model.add(Dense(2, activation='softmax'))

    model.compile(optimizer=Adam(hp_lr),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model

tuner = kt.Hyperband(build_model, objective='val_accuracy')
tuner.search(X_train, Y_train, validation_data=(X_val, Y_val))
```

- Keras Tuner tự động tìm kiến trúc CNN tối ưu bằng Hyperband và EarlyStopping.
- Tối ưu các tham số quan trọng: filters, dense units, dropout, learning rate.
- Chọn mô hình có validation accuracy cao nhất và huấn luyện lại để đạt hiệu suất tối đa.

Hình 24a. Tối ưu kiến trúc CNN bằng Keras Tuner (Hyperband).

CNN + KERAS TUNER

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_4 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_5 (Conv2D)	(None, 28, 28, 64)	36,928
max_pooling2d_5 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_2 (Dense)	(None, 192)	2,408,640
dropout_1 (Dropout)	(None, 192)	0
dense_3 (Dense)	(None, 2)	386

Total params: 7,396,040 (28.21 MB)

Trainable params: 2,465,346 (9.40 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 4,930,694 (18.81 MB)

Hình 24b. Cấu trúc của Neuron Network

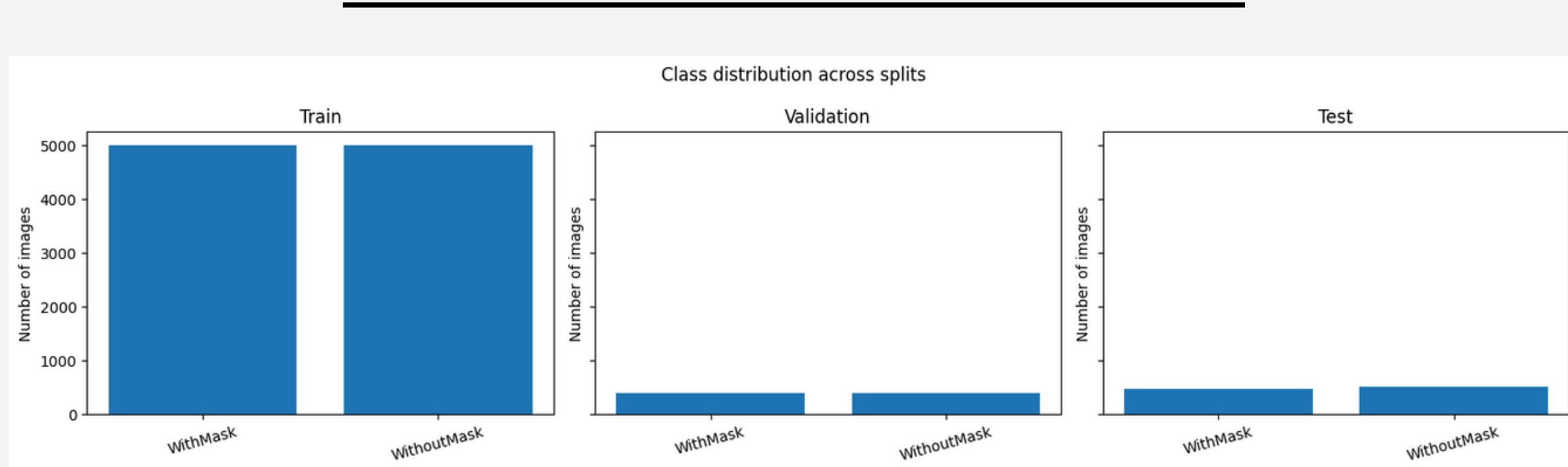
○ ○ ○

IV/ THỰC NGHIỆM

○ ○ ○

DATASET

- **Dataset:** [Face Mask 12k Images Dataset \(Kaggle\)](#).
- **Số lượng:** Tổng cộng ~12,000 ảnh.
- **Phân chia dữ liệu (Data Split):**
 - **Train:** 10,000 ảnh (Dùng để huấn luyện).
 - **Validation:** 800 ảnh.
 - **Test:** 992 ảnh



Hình 25. Plot class distribution



Hình 26. Sample Training Images

TIỀN XỬ LÝ - ĐỘ ĐO

Tiền xử lý

- Resize ảnh về 128x128.
- Chuẩn hóa pixel (Normalize) về khoảng [0, 1].
- Chuyển sang ảnh xám (Grayscale) cho HOG/LBP và giữ RGB cho CNN.



- Accuracy
- Precision
- recall
- f1-score
- weighted avg
- macro avg

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{số lượng mẫu}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

TP = True positive

TN = True negative

FP = False positive

FN = False negative

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Độ đo

KẾT QUẢ THỰC NGHIỆM

KNN:

Đặc trưng: HOG8x2

Test Accuracy (HOG 8x2 + KNN Optuna): 0.9839

Classification report (KNN):

	precision	recall	f1-score	support
WithMask	1.00	0.97	0.98	483
WithoutMask	0.97	1.00	0.98	509
accuracy			0.98	992
macro avg	0.98	0.98	0.98	992
weighted avg	0.98	0.98	0.98	992

Đặc trưng: HOG6x3

Test Accuracy (HOG 6x3 + KNN Optuna): 0.9748

Classification report (KNN):

	precision	recall	f1-score	support
WithMask	0.99	0.95	0.97	483
WithoutMask	0.96	0.99	0.98	509
accuracy			0.97	992
macro avg	0.98	0.97	0.97	992
weighted avg	0.98	0.97	0.97	992

Đặc trưng: LBP

Test Accuracy (LBP + KNN Optuna): 0.9556

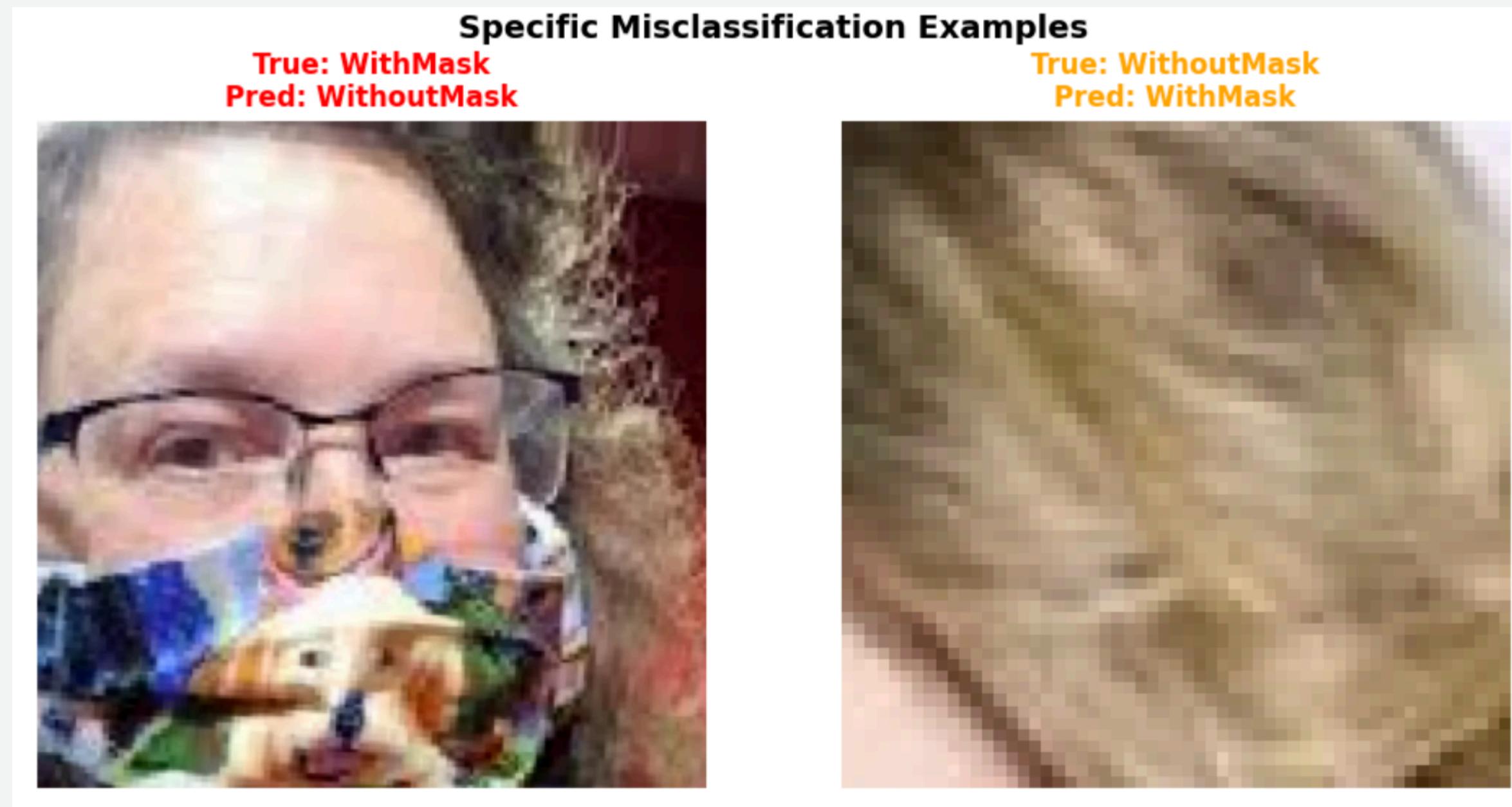
Classification report (KNN + LBP):

	precision	recall	f1-score	support
WithMask	0.99	0.92	0.95	483
WithoutMask	0.93	0.99	0.96	509
accuracy			0.96	992
macro avg	0.96	0.95	0.96	992
weighted avg	0.96	0.96	0.96	992

KẾT QUẢ THỰC NGHIỆM

KNN:

Đặc trưng HOG8x2:



KẾT QUẢ THỰC NGHIỆM

KNN:

Đặc trưng HOG6x3:



KẾT QUẢ THỰC NGHIỆM

KNN:

Đặc trưng LBP:



KẾT QUẢ THỰC NGHIỆM

SVM:

Đặc trưng: HOG8x2

Test Accuracy: 0.9899				
Classification Report:				
	precision	recall	f1-score	support
WithMask	0.98	1.00	0.99	483
WithoutMask	1.00	0.98	0.99	509
accuracy			0.99	992
macro avg	0.99	0.99	0.99	992
weighted avg	0.99	0.99	0.99	992

Đặc trưng: HOG6x3

Test Accuracy: 0.9879				
Classification Report:				
	precision	recall	f1-score	support
WithMask	0.98	1.00	0.99	483
WithoutMask	1.00	0.98	0.99	509
accuracy			0.99	992
macro avg	0.99	0.99	0.99	992
weighted avg	0.99	0.99	0.99	992

Đặc trưng: LBP

--- 1. CÁC CHỈ SỐ CHI TIẾT ---				
	precision	recall	f1-score	support
WithMask	0.98	0.96	0.97	483
WithoutMask	0.97	0.98	0.97	509
accuracy			0.97	992
macro avg	0.97	0.97	0.97	992
weighted avg	0.97	0.97	0.97	992

KẾT QUẢ THỰC NGHIỆM

SVM:

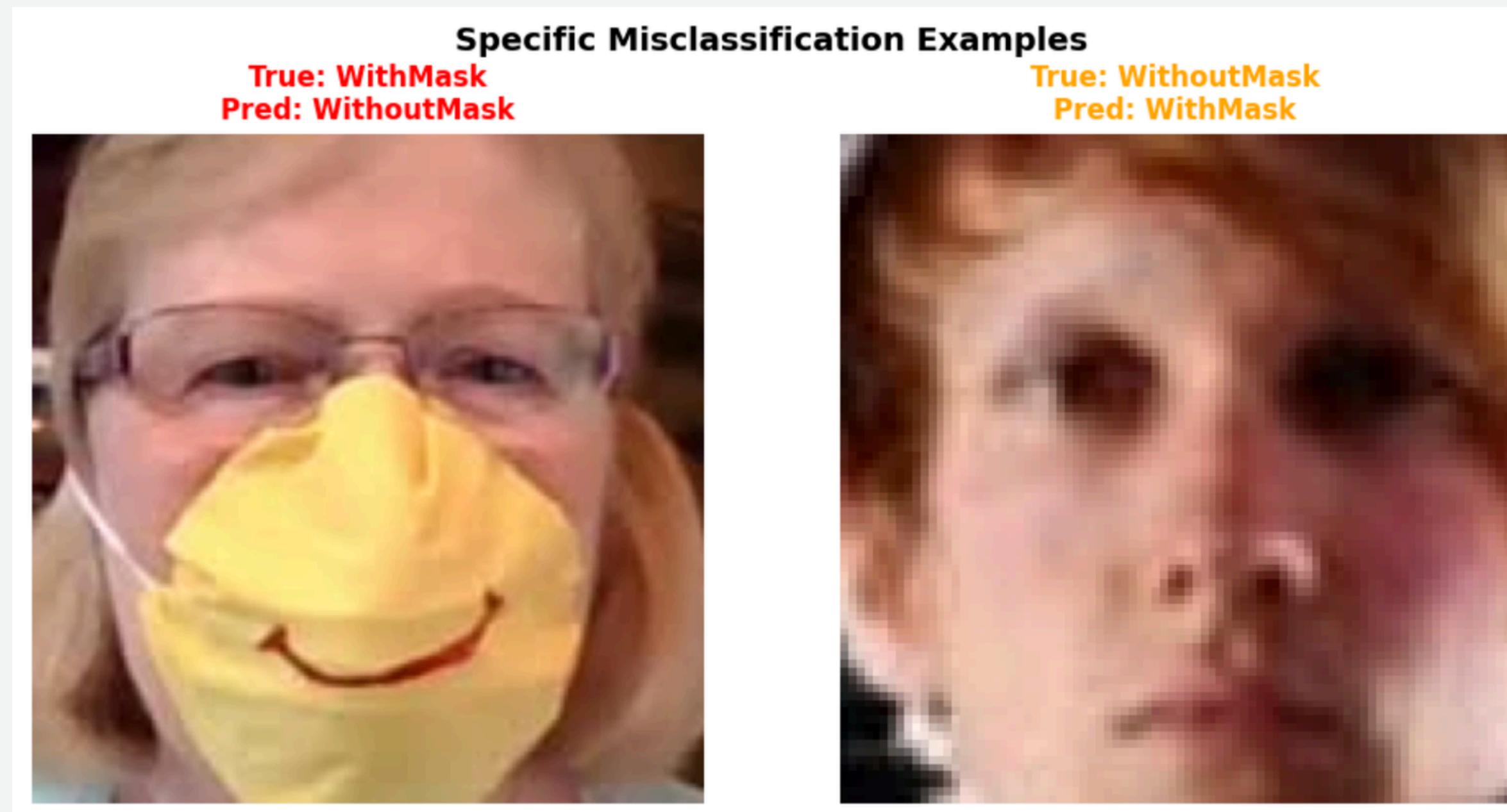
Đặc trưng HOG8x2:



KẾT QUẢ THỰC NGHIỆM

SVM:

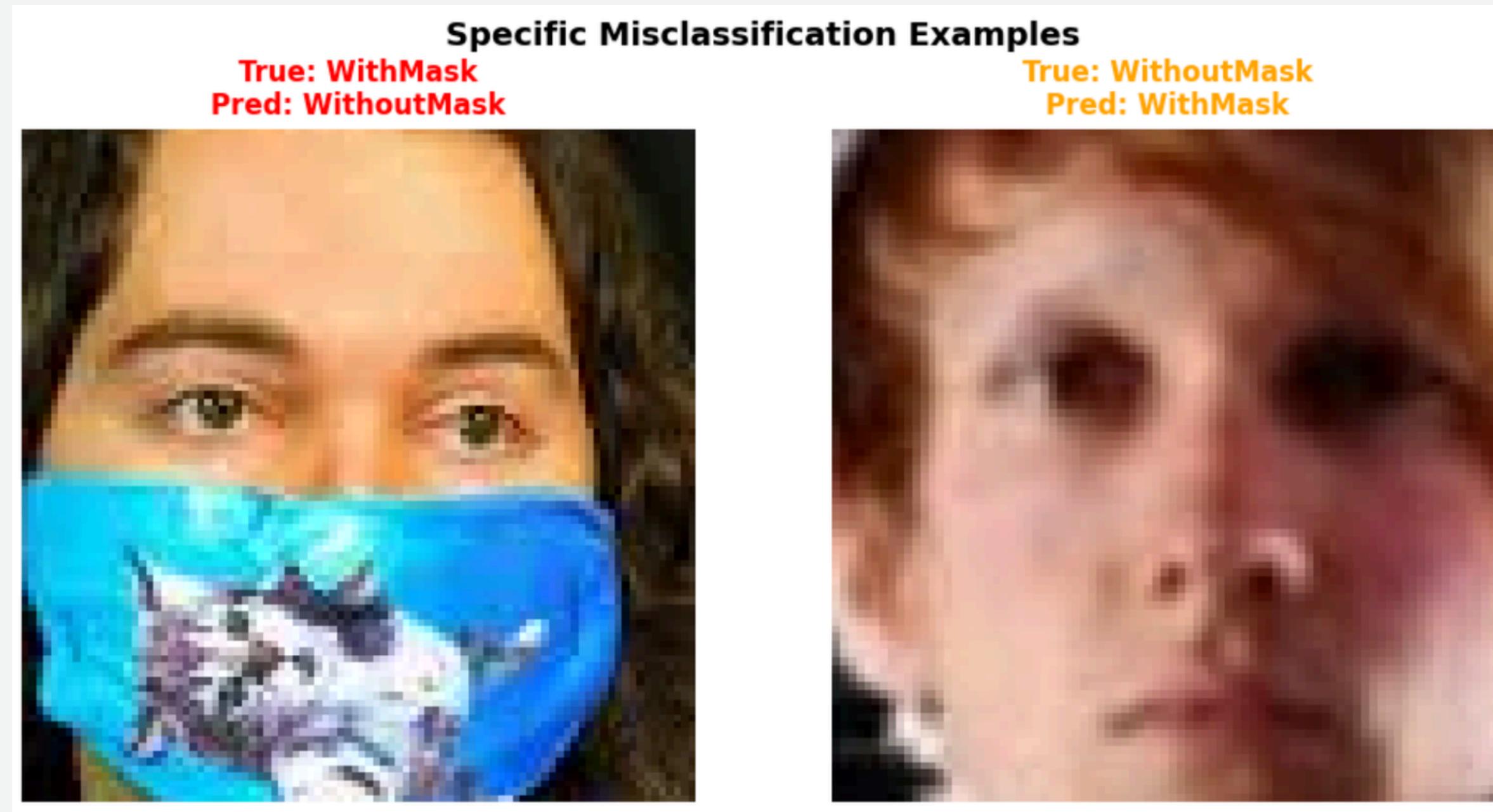
Đặc trưng HOG6x3:



KẾT QUẢ THỰC NGHIỆM

SVM:

Đặc trưng LBP:



KẾT QUẢ THỰC NGHIỆM

Random Forest:

Đặc trưng: HOG8x2

```
--- Evaluating ---  
Test Accuracy: 0.9818548387096774  
  
Classification Report:  
precision    recall   f1-score   support  
  
WithMask      0.97     1.00      0.98      483  
WithoutMask    1.00     0.97      0.98      509  
  
accuracy      0.98      0.98      0.98      992  
macro avg     0.98     0.98      0.98      992  
weighted avg  0.98     0.98      0.98      992
```

Đặc trưng: HOG6x3

```
--- Evaluating ---  
Test Accuracy: 0.9818548387096774  
  
Classification Report:  
precision    recall   f1-score   support  
  
WithMask      0.97     0.99      0.98      483  
WithoutMask    0.99     0.97      0.98      509  
  
accuracy      0.98      0.98      0.98      992  
macro avg     0.98     0.98      0.98      992  
weighted avg  0.98     0.98      0.98      992
```

Đặc trưng: LBP

```
Test Accuracy: 0.9092741935483871  
  
Classification Report:  
precision    recall   f1-score   support  
  
WithMask      0.92     0.89      0.90      483  
WithoutMask    0.90     0.93      0.91      509  
  
accuracy      0.91      0.91      0.91      992  
macro avg     0.91     0.91      0.91      992  
weighted avg  0.91     0.91      0.91      992
```

KẾT QUẢ THỰC NGHIỆM

Random Forest:

Đặc trưng HOG8x2:



KẾT QUẢ THỰC NGHIỆM

Random Forest:

Đặc trưng HOG6x3:



KẾT QUẢ THỰC NGHIỆM

Random Forest:

Đặc trưng LBP:



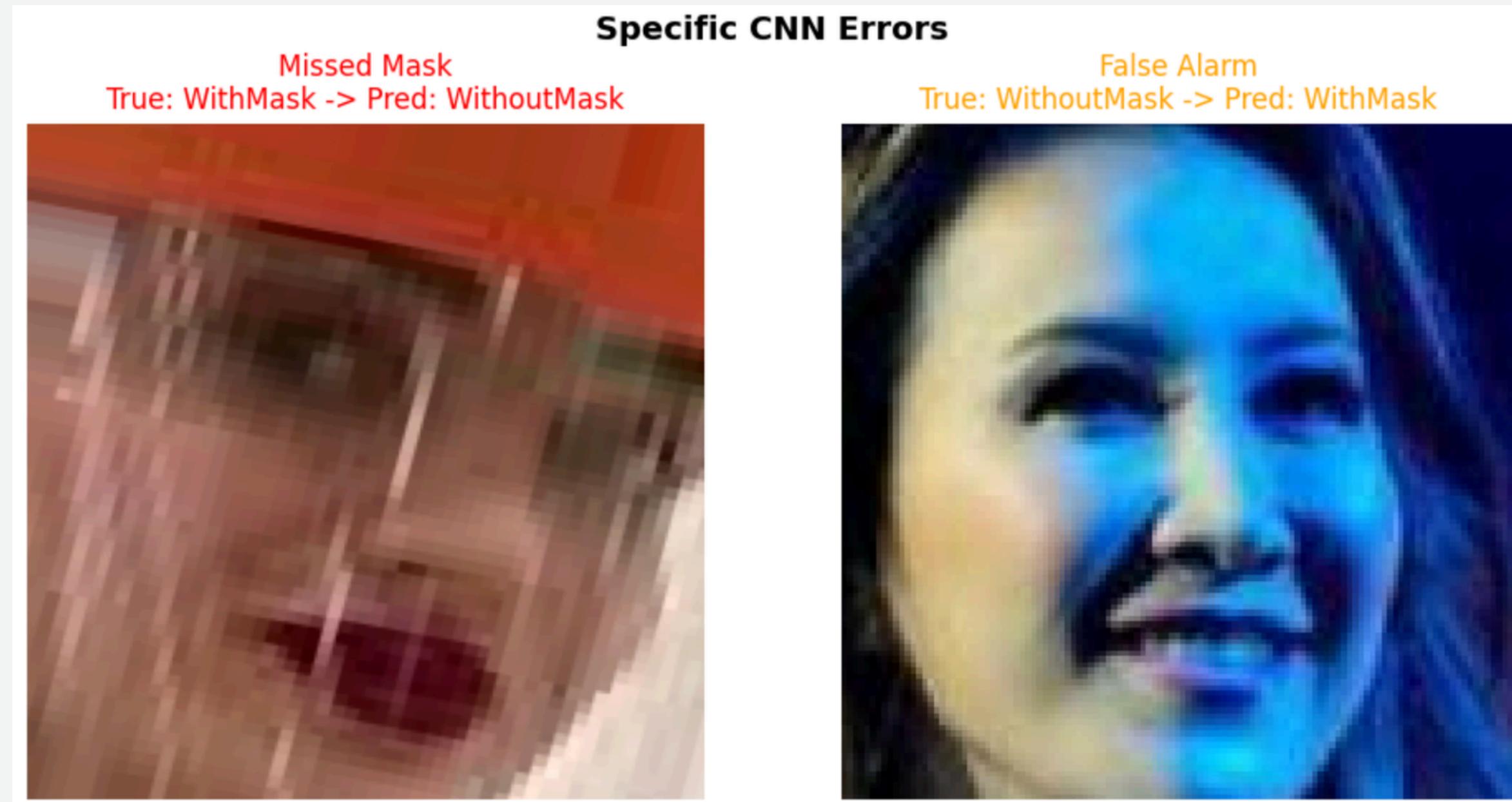
KẾT QUẢ THỰC NGHIỆM

CNN:

--- CLASSIFICATION REPORT ---				
	precision	recall	f1-score	support
Mask	0.99	0.99	0.99	483
No Mask	0.99	0.99	0.99	509
accuracy			0.99	992
macro avg	0.99	0.99	0.99	992
weighted avg	0.99	0.99	0.99	992

KẾT QUẢ THỰC NGHIỆM

CNN:



KẾT QUẢ THỰC NGHIỆM

Mô hình	Feature	Accuracy
CNN	None	0.99
	HOG8x2	0.9818
Random Forest	HOG6x3	0.9818
	LBP	0.9092
SVM	HOG8x2	0.9899
	HOG6x3	0.9879
	LBP	0.97
KNN	HOG8x2	0.9839
	HOG6x3	0.9748
	LBP	0.9556

KẾT QUẢ THỰC NGHIỆM

CNN (Convolutional Neural Network)

- **Hiệu suất:** Accuracy: 0.99
- **Nhận xét:**
 - CNN tự động học các đặc trưng từ ảnh thô mà không cần trích xuất thủ công (như HOG hay LBP). Các chỉ số Precision, Recall và F1-score đều đạt 0.99 cho cả hai nhãn (Mask và No Mask).
 - Tốn tài nguyên tính toán hơn các mô hình ML truyền thống (dù với bài toán nhỏ này thì không đáng kể).

SVM (Support Vector Machine)

- **Hiệu suất:** Accuracy: ~0.9899 với HOG 8x2
- **Nhận xét:**
 - SVM phân chia siêu phẳng (hyperplane) rất tốt trong không gian nhiều chiều tạo ra bởi HOG. Đây là minh chứng cho thấy sự kết hợp HOG + SVM cực kỳ mạnh mẽ cho bài toán phát hiện đối tượng/phân loại ảnh đơn giản.
 - Hoạt động tốt ổn định trên cả các đặc trưng khác nhau.

KẾT QUẢ THỰC NGHIỆM

KNN (K-Nearest Neighbors)

- **Hiệu suất:** Khá tốt (Accuracy: ~0.98 với HOG).
- **Nhận xét:**
 - KNN vẫn thấp hơn SVM và CNN một chút. KNN có thể bị ảnh hưởng bởi khi vector đặc trưng HOG có số chiều lớn, làm khoảng cách giữa các điểm dữ liệu trở nên khó phân biệt hơn so với việc tìm siêu phẳng của SVM.

Random Forest

- **Hiệu suất:** Thấp nhất trong số các mô hình thử nghiệm, đặc biệt kém khi kết hợp với LBP (Accuracy: ~0.90).
- **Nhận xét:**
 - Random Forest hoạt động dựa trên việc chia nhỏ không gian đặc trưng. Các đặc trưng trích xuất (đặc biệt là LBP) không tạo ra các ngưỡng (thresholds) đủ rõ ràng để các cây quyết định phân loại chính xác so với cách tiếp cận không gian vector của SVM.

○ ○ ○

V/ DEMO

○ ○ ○



THANKS FOR
LISTENING