Notebook

November 22, 2024

1 Import libraries

```
import os
import sys
import cv2
import math
import json
import joblib
import nbformat
import numpy as np
import pandas as pd
import seaborn as sns
from tqdm import tqdm
from sklearn.svm import SVC
from datetime import datetime
import matplotlib.pyplot as plt
from nbconvert.exporters import PDFExporter
from skimage.feature import hog as skimage_hog
from sklearn.preprocessing import LabelEncoder
from IPython.display import display, Javascript
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix
from scipy.spatial.distance import cityblock, cosine, correlation, sqeuclidean
```

2 Load data

```
project_dir = os.getcwd()
project_dir = os.path.dirname(project_dir)
```

```
width = 64
height = 64
```

```
data_dir = project_dir + "\\data"

train_path = os.path.join(data_dir, "train")
```

```
test_path = os.path.join(data_dir, "test")

train_images = []
test_images = []

train_labels = []
test_labels = []

for path in (train_path, test_path):
    if (path.split('\\')[-1] == "train"):
        for dir in os.listdir(path):
            label_path = os.path.join(path, dir)
            label = dir.split('\\')[-1]
            for image in os.listdir(label_path):
                image_path = os.path.join(label_path, image)
                image = cv2.imread(image_path)
                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                image = cv2.resize(image, (width, height))
                train_images.append(image)
                train_labels.append(label)
    else:
        for dir in os.listdir(path):
            label_path = os.path.join(path, dir)
            label = dir.split('\\')[-1]
            for image in os.listdir(label_path):
                image_path = os.path.join(label_path, image)
                image = cv2.imread(image_path)
                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                image = cv2.resize(image, (width, height))
                test_images.append(image)
                test_labels.append(label)
```

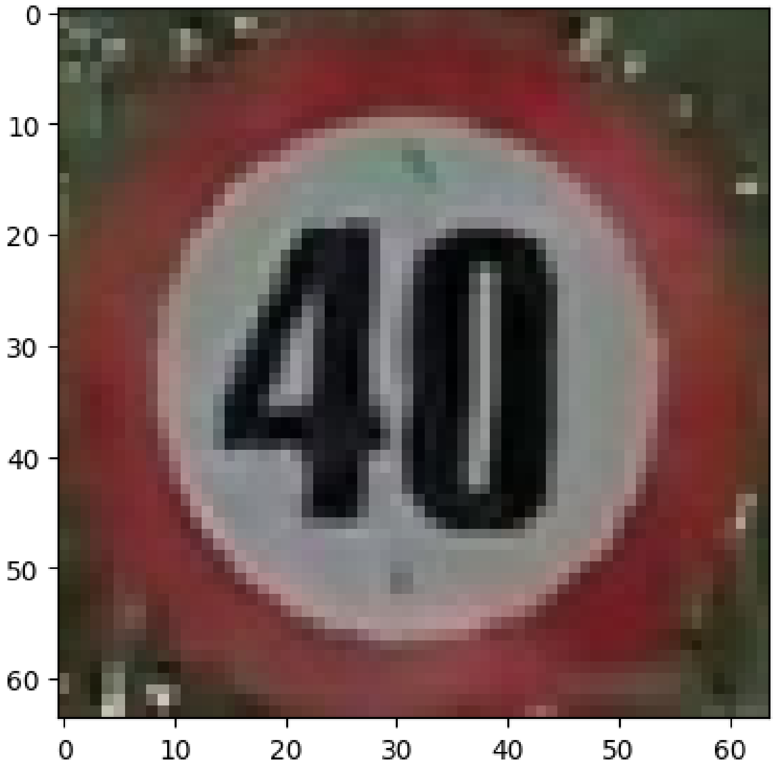
```
label_encoder = LabelEncoder()
train_labels_encoded = label_encoder.fit_transform(train_labels)
test_labels_encoded = label_encoder.transform(test_labels)
```

```
joblib.dump(train_images, project_dir + '\joblib\\train_images.joblib')
joblib.dump(test_images, project_dir + '\joblib\\test_images.joblib')
joblib.dump(train_labels_encoded, project_dir + '\joblib\\train_labels_encoded.
 ↪joblib')
joblib.dump(test_labels_encoded, project_dir + '\joblib\\test_labels_encoded.
 ↪joblib')
joblib.dump(label_encoder, project_dir + '\joblib\\label_encoder.joblib')
```

```
['d:\\ASUS\\Deploy-Traffic-Sign-Classification-through-
Images\\joblib\\label_encoder.joblib']
```

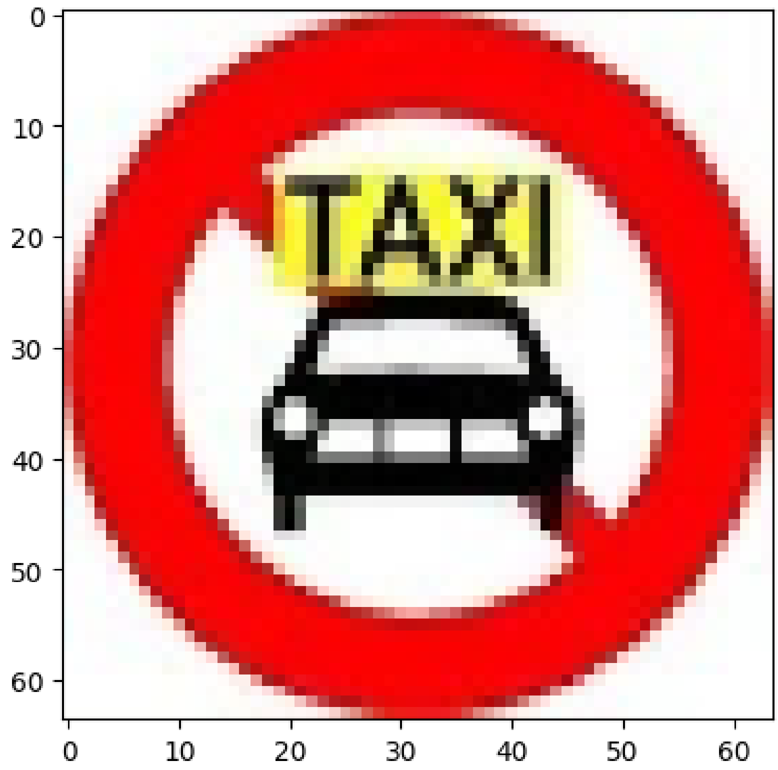
```
plt.imshow(test_images[0])
```

```
<matplotlib.image.AxesImage at 0x1f5b1264250>
```



```
plt.imshow(train_images[1])
```

```
<matplotlib.image.AxesImage at 0x1f5b12559d0>
```



3 Extract features

```python
def blur_image(image):
    blurred_image = cv2.medianBlur(image, 5)
    return blurred_image
```

```python
def color_histogram(image):
    # image = cv2.cvtColor(image, cv2.COLOR_RGB2LUV)
    row, column, channel = image.shape[:3]
    size = row * column
    feature = []
    for k in range(channel):
        histogram = np.squeeze(cv2.calcHist([image], [k], None, [32], [0, 256]))
        histogram = histogram / size
        feature.extend(histogram)
    return feature
```

```python
def hog(image):
    # image = cv2.cvtColor(image, cv2.COLOR_RGB2LUV)
    hog_features = skimage_hog(image, orientations=9, pixels_per_cell=(8, 8),␣
  ↪cells_per_block=(2, 2), visualize=False, block_norm='L2-Hys',␣
  ↪transform_sqrt=True, channel_axis=2)
    return hog_features
```

```
def extract_features(images):
    blurred_images = [blur_image(image) for image in tqdm(images, desc="Blur␣
  ↪Images")]
    color_features = [color_histogram(image) for image in tqdm(blurred_images,␣
  ↪desc="Extracting Color Features")]
    hog_features = [hog(image) for image in tqdm(blurred_images,␣
  ↪desc="Extracting HOG Features")]
    combined_features = [np.concatenate((color_feature, hog_feature))
                         for color_feature, hog_feature in␣
  ↪tqdm(zip(color_features, hog_features), desc="Combining Features")]

    return combined_features
```

```
train_features = extract_features(train_images)
joblib.dump(train_features, project_dir + '\joblib\\train_features.joblib')
```

```
Blur Images:   0%|          | 0/1415 [00:00<?, ?it/s]

Blur Images: 100%|      | 1415/1415 [00:01<00:00, 1056.08it/s]
Extracting Color Features: 100%|      | 1415/1415 [00:00<00:00, 9148.61it/s]
Extracting HOG Features: 100%|      | 1415/1415 [00:08<00:00, 173.08it/s]
Combining Features: 1415it [00:00, 24137.55it/s]

['d:\\ASUS\\Deploy-Traffic-Sign-Classification-through-
Images\\joblib\\train_features.joblib']
```

```
test_features = extract_features(test_images)
joblib.dump(test_features, project_dir + '\joblib\\test_features.joblib')
```

```
Blur Images: 100%|      | 150/150 [00:00<00:00, 1101.31it/s]
Extracting Color Features: 100%|      | 150/150 [00:00<00:00, 7931.84it/s]
Extracting HOG Features: 100%|      | 150/150 [00:00<00:00, 155.71it/s]
Combining Features: 150it [00:00, 42889.47it/s]

['d:\\ASUS\\Deploy-Traffic-Sign-Classification-through-
Images\\joblib\\test_features.joblib']
```

4 Distance metrics KNN

```
def chi_square_distance(x, y):
    return cv2.compareHist(np.array(x, dtype=np.float32), np.array(y, dtype=np.
  ↪float32), cv2.HISTCMP_CHISQR)
```

```python
def bhattacharyya_distance(x, y):
    return cv2.compareHist(np.array(x, dtype=np.float32), np.array(y, dtype=np.float32), cv2.HISTCMP_BHATTACHARYYA)

def intersection_distance(x, y):
    return 1 - cv2.compareHist(np.array(x, dtype=np.float32), np.array(y, dtype=np.float32), cv2.HISTCMP_INTERSECT)
```

5 Load Best Model

```python
knn_model = joblib.load(project_dir + '\\joblib\\best_knn_model.joblib')
svm_model = joblib.load(project_dir + '\\joblib\\best_svm_model.joblib')

y_pred_knn = knn_model.predict(test_features)
y_pred_svm = svm_model.predict(test_features)
```

6 Gridsearch KNN

```python
# knn_model = KNeighborsClassifier()
# knn_model.fit(train_features, train_labels_encoded)
# y_pred_knn = knn_model.predict(test_features)
```

```python
# param_grid = {
#     'n_neighbors': [3, 4, 5, 6, 7, 10],
#     'weights': ['uniform', 'distance'],
#     'leaf_size': [5, 10, 20, 30, 40, 50],
#     'metric': [
#         cityblock,
#         cosine,
#         # correlation,
#         sqeuclidean,
#         chi_square_distance,
#         bhattacharyya_distance,
#         intersection_distance
#     ]
# }

# knn_model = KNeighborsClassifier()
# grid_search_knn = GridSearchCV(
#     knn_model,
#     param_grid,
#     cv=3,
#     scoring='f1_macro',
#     verbose=3
# )

# grid_search_knn.fit(train_features, train_labels_encoded)
```

```
# best_knn = grid_search_knn.best_estimator_
# print(f"Best Params: {grid_search_knn.best_params_}")


# y_pred_knn = best_knn.predict(test_features)

# joblib.dump(best_knn, project_dir + '\joblib\\best_knn_model.joblib')
```

7 Gridsearch SVM

```
# svm_model = SVC()
# svm_model.fit(train_features, train_labels_encoded)
# y_pred_svm = svm_model.predict(test_features)
```

```
# param_grid = {
#     'C': [0.1, 0.2, 0.3, 0.4],
#     'kernel': ['rbf', 'linear', 'poly', 'sigmoid'],
#     'gamma': ['scale', 'auto', 0.1, 0.01, 0.001],
#     'degree': [2, 3, 4],
# }

# svm_model = SVC()

# grid_search_svm = GridSearchCV(
#     estimator=svm_model,
#     param_grid=param_grid,
#     cv=3,
#     scoring='f1_macro',
#     verbose=3,
# )

# grid_search_svm.fit(train_features, train_labels_encoded)
```

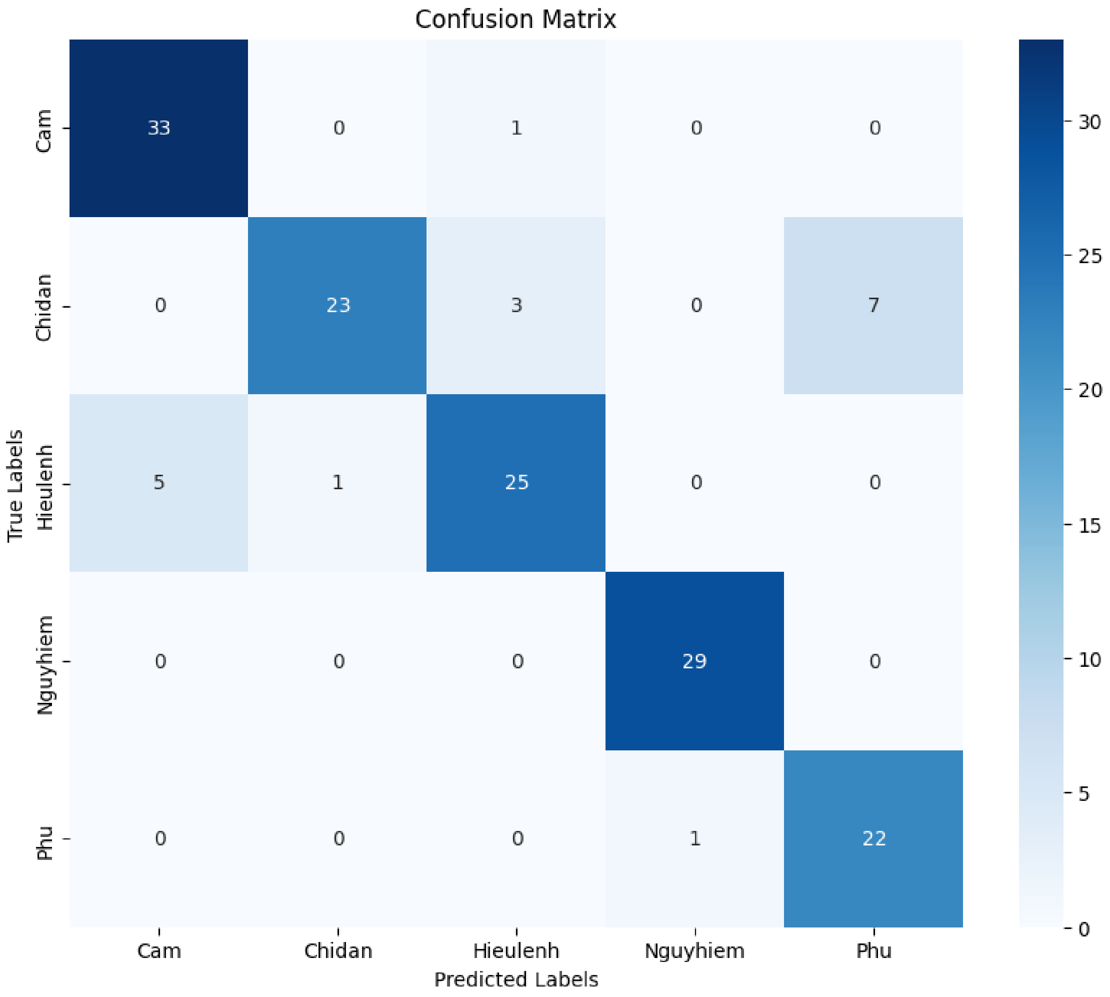
```
# best_svm = grid_search_svm.best_estimator_
# # Get the best parameters and score
# print("Best parameters:", grid_search_svm.best_params_)

# y_pred_svm = best_svm.predict(test_features)

# joblib.dump(best_svm, project_dir + '\joblib\\best_svm_model.joblib')
```

8 Predict on test images for KNN

```
report_knn = classification_report(test_labels_encoded, y_pred_knn,␣
 ↪target_names=label_encoder.classes_)
print(report_knn)
```

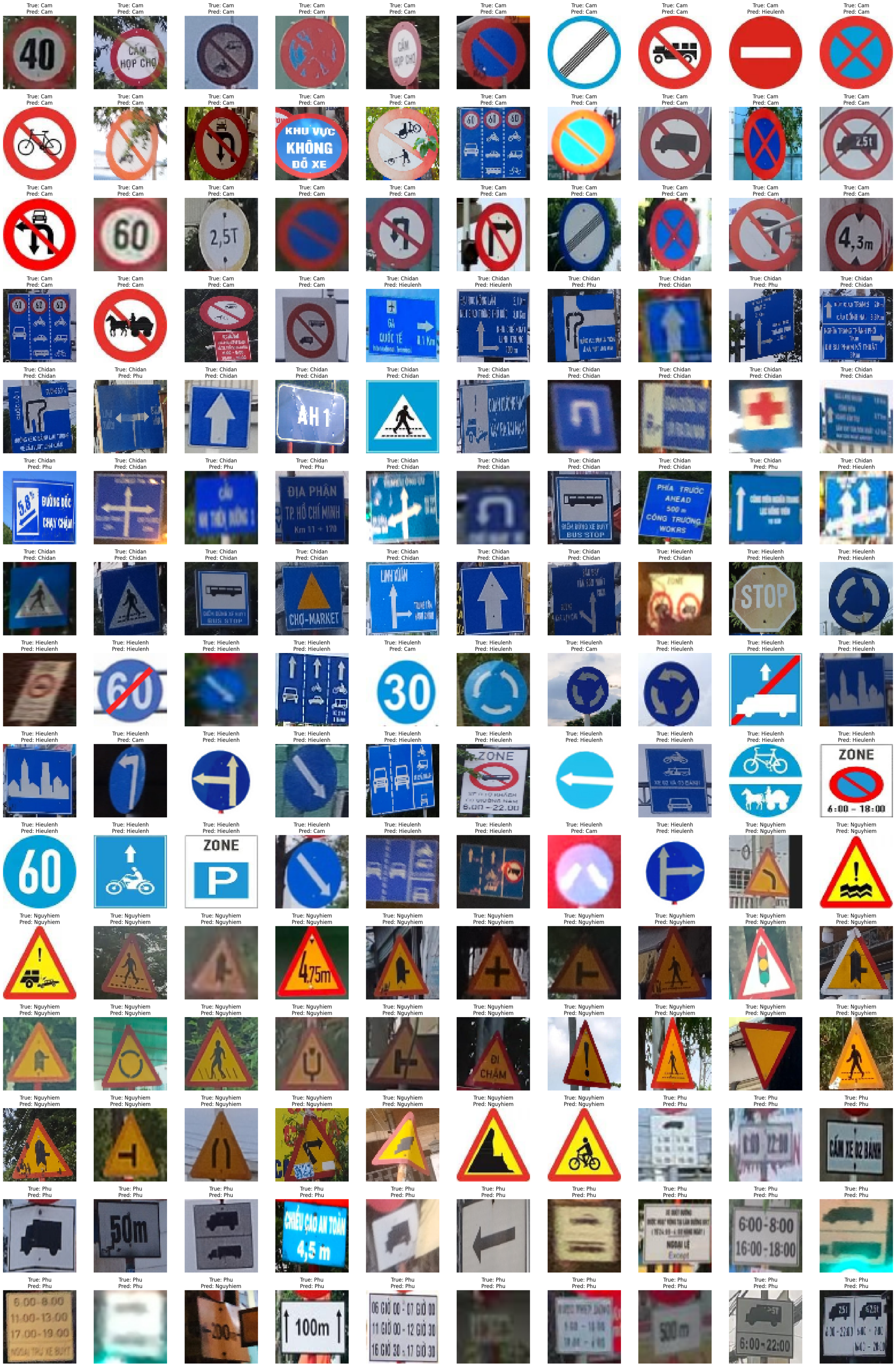
```
              precision    recall  f1-score   support

         Cam       0.87      0.97      0.92        34
      Chidan       0.96      0.70      0.81        33
    Hieulenh       0.86      0.81      0.83        31
    Nguyhiem       0.97      1.00      0.98        29
         Phu       0.76      0.96      0.85        23

    accuracy                           0.88       150
   macro avg       0.88      0.89      0.88       150
weighted avg       0.89      0.88      0.88       150
```

```
heatmap_label_knn = confusion_matrix(test_labels_encoded, y_pred_knn)

plt.figure(figsize=(10, 8))
sns.heatmap(heatmap_label_knn, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```



```
n_columns = 10
n_rows = math.ceil(len(test_images) / n_columns)

fig, axes = plt.subplots(n_rows, n_columns, figsize=(30, n_rows * 3))

for idx, (image, true_label, pred_label) in enumerate(zip(test_images, test_labels_encoded, y_pred_knn)):
    row = idx // n_columns
    col = idx % n_columns

    axes[row, col].imshow(image)
    axes[row, col].set_title(f'True: {label_encoder.classes_[true_label]}\nPred: {label_encoder.classes_[pred_label]}')
    axes[row, col].axis('off')
```

```python
for ax in axes.flat:
    if not ax.has_data():
        ax.axis('off')

plt.tight_layout()
plt.show()
```

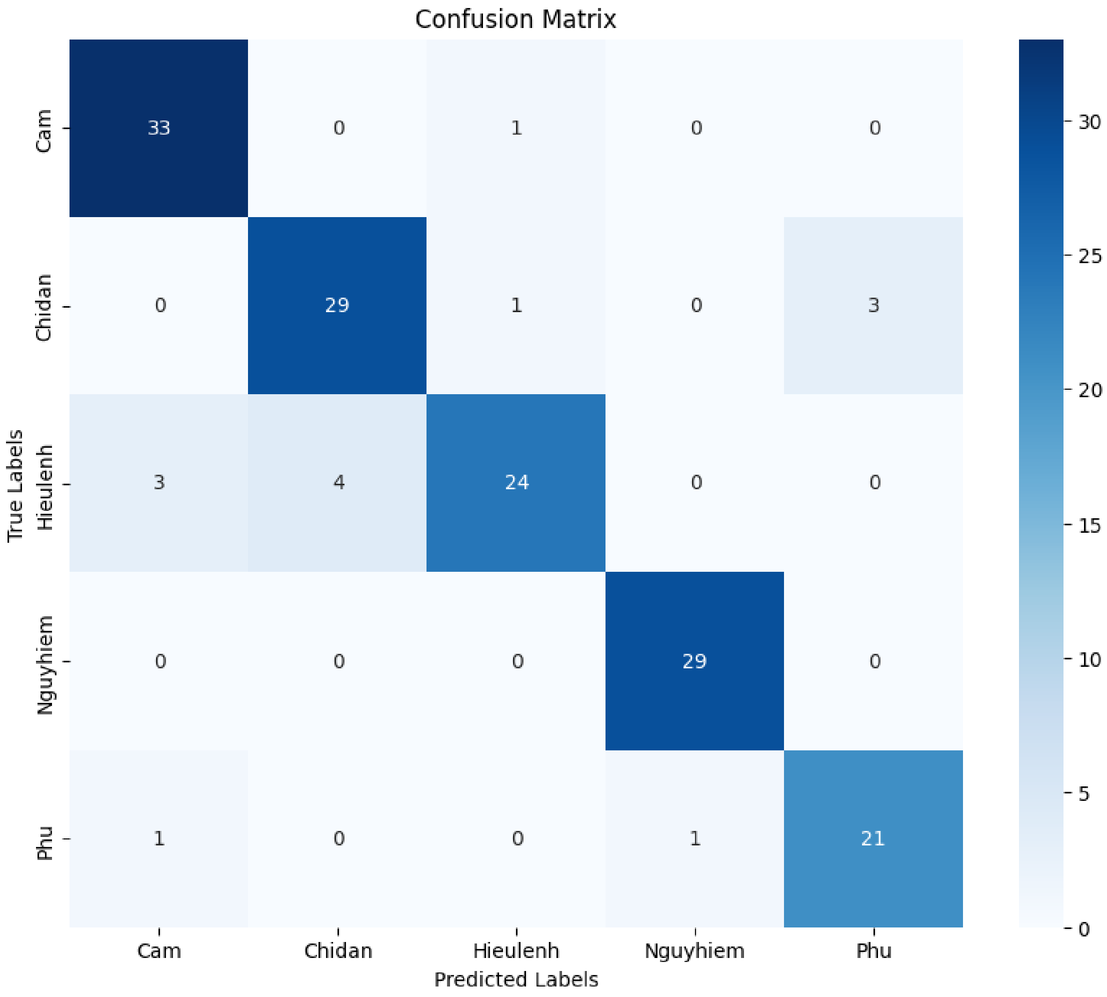

True: Cam
Pred: Cam
40
True: Cam
Pred: Cam
CẤM HỌP CHỢ
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
CẤM HỌP CHỢ
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Hieulenh
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
KHU VỰC KHÔNG ĐỖ XE
True: Cam
Pred: Cam
True: Cam
Pred: Cam
60 60 60
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
2,5t
True: Cam
Pred: Cam
True: Cam
Pred: Cam
60
True: Cam
Pred: Cam
2,5T
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
4,3m
True: Cam
Pred: Cam
True: Cam
Pred: Cam
True: Cam
Pred: Cam
CẤM
True: Cam
Pred: Cam
True: Chidan
Pred: Hieulenh
True: Chidan
Pred: Hieulenh
True: Chidan
Pred: Phu
True: Chidan
Pred: Chidan
True: Chidan
Pred: Phu
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
True: Chidan
Pred: Phu
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
AH 1
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
True: Chidan
Pred: Phu
5,8% ĐƯỜNG DỐC CHẠY CHẬM
True: Chidan
Pred: Chidan
True: Chidan
Pred: Phu
True: Chidan
Pred: Phu
ĐỊA PHẬN TP. HỒ CHÍ MINH Km 11 + 170
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
BUS STOP
True: Chidan
Pred: Chidan
PHÍA TRƯỚC AHEAD 500 m CÔNG TRƯỜNG WOKRS
True: Chidan
Pred: Phu
True: Chidan
Pred: Hieulenh
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
BUS STOP
True: Chidan
Pred: Chidan
CHỢ-MARKET
True: Chidan
Pred: Chidan
LINH XUÂN
True: Chidan
Pred: Chidan
True: Chidan
Pred: Chidan
True: Hieulenh
Pred: Chidan
True: Hieulenh
Pred: Hieulenh
STOP
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
60
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Cam
30
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Cam
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Cam
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
ZONE
6:00 – 22:00
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
ZONE
6:00 – 18:00
True: Hieulenh
Pred: Hieulenh
60
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
ZONE
P
True: Hieulenh
Pred: Cam
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Hieulenh
True: Hieulenh
Pred: Cam
True: Hieulenh
Pred: Hieulenh
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
4,75m
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
ĐI CHẬM
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Nguyhiem
Pred: Nguyhiem
True: Phu
Pred: Phu
True: Phu
Pred: Phu
6:00 22:00
True: Phu
Pred: Phu
CẤM XE 02 BÁNH
True: Phu
Pred: Phu
True: Phu
Pred: Phu
50m
True: Phu
Pred: Phu
True: Phu
Pred: Phu
CHIỀU CAO AN TOÀN 4,5 m
True: Phu
Pred: Phu
True: Phu
Pred: Phu
True: Phu
Pred: Phu
True: Phu
Pred: Phu
NGOẠI LỆ
Except
True: Phu
Pred: Phu
6:00-8:00
16:00-18:00
True: Phu
Pred: Phu
True: Phu
Pred: Phu
6.00-8.00
11.00-13.00
17.00-19.00
NGOẠI TRỪ XE BUÝT
True: Phu
Pred: Phu
True: Phu
Pred: Nguyhiem
True: Phu
Pred: Phu
100m
True: Phu
Pred: Phu
06 GIỜ 00 - 07 GIỜ 00
11 GIỜ 00 - 12 GIỜ 30
16 GIỜ 30 - 17 GIỜ 30
True: Phu
Pred: Phu
True: Phu
Pred: Phu
True: Phu
Pred: Phu
500 m
True: Phu
Pred: Phu
6:00-22:00
True: Phu
Pred: Phu

9 Predict on test images for SVM

```
report_svm = classification_report(test_labels_encoded, y_pred_svm,␣
 ↪target_names=label_encoder.classes_)
print(report_svm)
```

```
              precision    recall  f1-score   support

         Cam       0.89      0.97      0.93        34
      Chidan       0.88      0.88      0.88        33
    Hieulenh       0.92      0.77      0.84        31
    Nguyhiem       0.97      1.00      0.98        29
         Phu       0.88      0.91      0.89        23

    accuracy                           0.91       150
   macro avg       0.91      0.91      0.91       150
weighted avg       0.91      0.91      0.91       150
```

```
heatmap_label_svm = confusion_matrix(test_labels_encoded, y_pred_svm)

plt.figure(figsize=(10, 8))
sns.heatmap(heatmap_label_svm, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```



```
n_columns = 10
n_rows = math.ceil(len(test_images) / n_columns)

fig, axes = plt.subplots(n_rows, n_columns, figsize=(30, n_rows * 3))

for idx, (image, true_label, pred_label) in enumerate(zip(test_images,␣
 ↪test_labels_encoded, y_pred_svm)):
    row = idx // n_columns
    col = idx % n_columns

    axes[row, col].imshow(image)
    axes[row, col].set_title(f'True: {label_encoder.classes_[true_label]}\nPred:
 ↪ {label_encoder.classes_[pred_label]}')
    axes[row, col].axis('off')
```

```
for ax in axes.flat:
    if not ax.has_data():
        ax.axis('off')

plt.tight_layout()
plt.show()
```

10 Save grid search results

```
def export_notebook_to_pdf(notebook_path, project_dir):
    results_dir = os.path.join(project_dir)
    os.makedirs(results_dir, exist_ok=True)

    # Đọc notebook
    with open(notebook_path, 'r', encoding='utf-8') as f:
        nb = nbformat.read(f, as_version=4)

    # Cấu hình PDF exporter
    pdf_exporter = PDFExporter()
    pdf_exporter.exclude_input_prompt = True
    pdf_exporter.exclude_output_prompt = True

    # Thêm template và style cơ bản
    pdf_exporter.template_name = 'classic'

    # Chuyển đổi sang PDF
    pdf_data, resources = pdf_exporter.from_notebook_node(nb)

    # Tạo tên file với timestamp
    current_time = datetime.now().strftime('%Y-%m-%d_%H_%M_%S')
    pdf_file = os.path.join(results_dir, f"notebook_export_{current_time}.pdf")

    # Lưu file PDF
    with open(pdf_file, 'wb') as f:
        f.write(pdf_data)

    print(f"Đã xuất file PDF thành công: {pdf_file}")
    return pdf_file
```

```
# project_dir = os.path.dirname(project_dir)
notebook_path = project_dir + "\\model\\main.ipynb"
proj_dir = project_dir + "\\grid_search_results"

export_notebook_to_pdf(notebook_path, proj_dir)
```