ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN VIỄN THÔNG

----------o0o----------

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: MÁY HỌC CƠ BẢN VÀ ỨNG DỤNG

Đề tài: Nhận dạng chữ viết tay bằng mô hình CNN & LSTM vào việc chấm phiếu thi trắc nghiệm

GV: Nguyễn Khánh Lợi

Nhóm 08-L01

Bảng phân chia công việc

Họ và tên	MSSV	Phân công	%
Hoàng Trung Kiên	2012512	Train model	40
Vũ Minh Hiếu	2113366	Tạo chương trình nhận diện	20
Nguyễn Bá Thao	2112274	Tìm, xử lí dữ liệu	20
Trần Trung Kiên	2013550	Viết báo cáo, slide	20

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

I. CƠ SỞ LÝ THUYẾT

1. Tổng quan về học máy và học sâu

1.1. Học máy (Machine Learning)

Học máy là một lĩnh vực con của trí tuệ nhân tạo (AI), trong đó các hệ thống có khả năng học hỏi và cải thiện hiệu suất từ dữ liệu mà không cần được lập trình cụ thể từng bước. Thuật toán học máy sẽ phân tích các mẫu dữ liệu (dataset) để xây dựng mô hình toán học có khả năng dự đoán hoặc phân loại dữ liệu mới. Các kỹ thuật học máy bao gồm học có giám sát (supervised learning), học không giám sát (unsupervised learning), và học tăng cường (reinforcement learning).

1.2. Học sâu (Deep Learning)

Học sâu là một nhánh của học máy sử dụng mạng nơ-ron nhân tạo với nhiều lớp (deep neural networks) để mô hình hóa mối quan hệ phức tạp trong dữ liệu. Học sâu đặc biệt hiệu quả trong các bài toán xử lý ảnh, âm thanh và ngôn ngữ tự nhiên nhờ khả năng tự động trích xuất đặc trưng từ dữ liệu đầu vào mà không cần kỹ thuật thủ công.

2. Nhận dạng ký tự viết tay (Handwritten Character Recognition)

2.1. Định nghĩa và vai trò

Nhận dạng ký tự viết tay là quá trình tự động phân loại và nhận diện các ký tự chữ cái hoặc số được viết tay từ hình ảnh. Đây là một ứng dụng thiết thực trong nhiều lĩnh vực như số hóa tài liệu, xử lý biểu mẫu, nhận dạng địa chỉ trên phong bì thư, bảng số xe, hoặc bài thi trắc nghiệm.

2.2. Thách thức

Việc nhận dạng ký tự viết tay gặp nhiều khó khăn do:

- Sự khác biệt về nét chữ giữa các cá nhân;
- Ký tự có hình dáng tương tự dễ bị nhầm lẫn (ví dụ: "O" và "0", "1" và "l");
- Ảnh đầu vào có thể bị nhiễu, mất nét, mờ hoặc ánh sáng không đồng đều;
- Việc gán nhãn dữ liệu thủ công là tốn công sức và dễ sai sót.

3. Các bước trong hệ thống nhận dạng ký tự

Một hệ thống nhận dạng ký tự thường trải qua các giai đoạn sau:

1) Thu thập dữ liệu: lấy ảnh chứa ký tự viết tay, gán nhãn;
2) Tiền xử lý ảnh: chuyển ảnh về thang xám, chuẩn hóa kích thước, loại bỏ nhiễu;
3) Trích xuất đặc trưng: dùng kỹ thuật thủ công hoặc mạng nơ-ron;
4) Huấn luyện mô hình: xây dựng và huấn luyện mô hình học sâu trên tập train;
5) Đánh giá mô hình: đo lường độ chính xác trên tập kiểm tra;
6) Dự đoán và ứng dụng: sử dụng mô hình đã học để nhận diện ảnh mới.

4. Các mô hình học sâu được sử dụng

4.1. Mạng nơ-ron tích chập (CNN – Convolutional Neural Network)

CNN là kiến trúc chủ đạo trong xử lý ảnh nhờ khả năng phát hiện đặc trưng cục bộ (local features). Cấu trúc CNN bao gồm:

- Lớp tích chập (Convolutional Layer): giúp mô hình học các đặc trưng như đường viền, cạnh, hình khối từ ảnh đầu vào;
- Lớp phi tuyến (Activation – thường dùng ReLU): tăng khả năng biểu diễn phi tuyến của mô hình;
- Lớp gộp (Pooling Layer): giảm kích thước dữ liệu, tăng tốc độ huấn luyện, và giảm overfitting;
- Lớp fully-connected: tổng hợp đặc trưng và đưa ra kết quả phân loại. CNN phù hợp với nhận dạng ký tự do hình ảnh ký tự có các cấu trúc hình học rõ ràng và ổn định.

4.2. Các kỹ thuật tăng cường hiệu suất huấn luyện

- Dropout: giúp tránh overfitting bằng cách ngẫu nhiên "tắt" một số node trong quá trình huấn luyện;
- Batch Normalization: chuẩn hóa đầu ra của từng lớp giúp mô hình huấn luyện nhanh hơn và ổn định hơn;

- EarlyStopping: dừng quá trình huấn luyện nếu mô hình không cải thiện sau một số epoch nhất định;
- ReduceLROnPlateau: tự động giảm learning rate nếu độ lỗi không còn giảm.

4.3. Hàm mất mát và tối ưu hóa

Loss Function: sparse_categorical_crossentropy được sử dụng cho bài toán phân loại nhiều lớp khi nhãn là số nguyên

Optimizer: Adam là bộ tối ưu hóa thích nghi phổ biến, giúp mô hình hội tụ nhanh hơn với learning rate mặc định hoặc tinh chỉnh (ví dụ: 0.0005).

5. Các chỉ số đánh giá mô hình

- Để kiểm tra hiệu quả mô hình sau huấn luyện, một số chỉ số chính được sử dụng: Accuracy: tỷ lệ dự đoán đúng trên tổng số mẫu;
- Confusion Matrix: ma trận thể hiện số lượng mẫu đúng và sai giữa các lớp; Precision, Recall, F1-score: đo lường độ chính xác, độ phủ và độ cân bằng giữa hai yếu tố trên;
- Biểu đồ loss và accuracy: thể hiện quá trình hội tụ và hiệu suất mô hình theo thời gian.

II. THỰC NGHIỆM

1.`Xây dựng mô hình CRNN

1.1. Mô tả dữ liệu

Bộ dữ liệu sử dụng trong bài toán là tập hợp các hình ảnh chứa **ký tự viết tay đơn lẻ**, bao gồm:

- 26 chữ cái in hoa (A-Z)
- 26 chữ cái thường (a-z)
- 10 chữ số (0-9)

Tổng cộng có 62 lớp nhãn. Mỗi hình ảnh chứa một ký tự đơn lẻ, được lưu trữ cùng với nhãn trong một file .csv. Các hình ảnh đã được chuẩn hóa và chia thành ba tập: train, validation và test.

1.2. Tiền xử lý dữ liệu

Quá trình tiền xử lý ảnh đầu vào được thực hiện như sau:

- Đọc ảnh: Từng ảnh được nạp bằng load_img() từ thư viện Keras, với chế độ grayscale để chuyển ảnh thành ảnh đơn kênh.
- Resize ảnh: Mỗi ảnh được thay đổi kích thước về cố định là 128x96 (WxH) pixel để đảm bảo đầu vào thống nhất cho mô hình.
- Chuẩn hóa pixel: Toàn bộ giá trị pixel được chuẩn hóa về khoảng [0, 1] bằng cách chia cho 255.0.
- Mã hóa nhãn: Các nhãn ký tự (A, B, 0, a, …) được mã hóa số nguyên bằng LabelEncoder từ thư viện sklearn, giúp phù hợp với hàm mất mát sparse_categorical_crossentropy.
- Tách tập dữ liệu: Dữ liệu được chia thành train và validation với tỷ lệ 80% - 20% thông qua train_test_split.

Kết quả đầu vào có shape (số mẫu, 96, 128, 1) — ảnh xám có chiều cao 96, chiều rộng 128 và 1 kênh màu.

1.3. Lý do chọn mô hình CRNN

Việc lựa chọn mô hình CRNN (thay vì CNN đơn thuần) được cân nhắc kỹ lưỡng dựa trên đặc thù dữ liệu và yêu cầu bài toán:

Trích xuất đặc trưng không gian bằng CNN: Các lớp Conv2D giúp nhận diện nét, đường cong, cạnh và các đặc trưng cục bộ từ ký tự viết tay.

LSTM cho xử lý chuỗi theo chiều ảnh: Sau khi reshape ảnh thành chuỗi, mô hình sử dụng LSTM hai chiều (Bidirectional) để học các mối quan hệ theo chiều ngang trong ảnh — rất quan trọng trong việc phân biệt các ký tự có nét tương tự.

Giảm độ phức tạp so với mô hình OCR truyền thống: CRNN không cần giải quyết bài toán phân vùng ký tự như các mô hình nhận diện dòng chữ, mà vẫn tận dụng được tính tuần tự của dữ liệu hình ảnh.

Khả năng mở rộng: Mô hình có thể dễ dàng tích hợp với CTC loss để xử lý bài toán nhận dạng chuỗi ký tự viết tay (handwritten text line OCR) trong tương lai.

1.4. Kiến trúc mô hình CRNN

Mô hình sử dụng trong bài là **CRNN (Convolutional Recurrent Neural Network)** — sự kết hợp giữa mạng CNN để trích xuất đặc trưng không gian và mạng RNN (LSTM) để học mối quan hệ tuần tự trong ảnh.

Chi tiết kiến trúc như sau:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 96, 128, 32)	320
batch_normalization (BatchNormalization)	(None, 96, 128, 32)	128
activation (Activation)	(None, 96, 128, 32)	0
max_pooling2d (MaxPooling2D)	(None, 48, 64, 32)	0
dropout (Dropout)	(None, 48, 64, 32)	0
conv2d_1 (Conv2D)	(None, 48, 64, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 48, 64, 64)	256

Layer (type)	Output Shape	Param #
activation_1 (Activation)	(None, 48, 64, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 24, 32, 64)	0
dropout_1 (Dropout)	(None, 24, 32, 64)	0
conv2d_2 (Conv2D)	(None, 24, 32, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 24, 32, 128)	512
activation_2 (Activation)	(None, 24, 32, 128)	0
max_pooling2d_2 (MaxPooling2D)	(None, 12, 16, 128)	0
dropout_2 (Dropout)	(None, 12, 16, 128)	0
reshape (Reshape)	(None, 16, 1536)	0
bidirectional (Bidirectional)	(None, 16, 128)	819,712
batch_normalization_3 (BatchNormalization)	(None, 16, 128)	512
bidirectional_1 (Bidirectional)	(None, 128)	98,816
batch_normalization_4 (BatchNormalization)	(None, 128)	512
dense (Dense)	(None, 64)	8,256
batch_normalization_5 (BatchNormalization)	(None, 64)	256
activation_3 (Activation)	(None, 64)	0

Layer (type)	Output Shape	Param #
dropout_3 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 62)	4,030

Bảng 1 Kiến trúc mô hình

Total params: 1,025,662 (3.91 MB)

Trainable params: 1,024,574 (3.91 MB)

Non-trainable params: 1,088 (4.25 KB)

Tóm tắt lại mô hình:

Thành phần	Thông tin
Input	(96, 128, 1)
Conv2D + BatchNorm + ReLU + MaxPooling + Dropout	3 khối liên tiếp với 32, 64, 128 filters.
Reshape	Biến tensor thành chuỗi (16, 12×128) để đưa vào LSTM
Bidirectional LSTM (x2)	2 lớp LSTM hai chiều với 64 units
Dense + BatchNorm + ReLU + Dropout	1 lớp ẩn với 64 units
Output Dense	Softmax với 62 lớp đầu ra

Bảng 2 Tóm tắt mô hình

Mô hình được xây dựng bằng Keras theo dạng Sequential, được tối ưu bằng thuật toán *Adam* với learning rate 0.0005.

Mô hình học sâu này được thiết kế để xử lý dữ liệu ảnh và phân loại chúng thành 62 lớp khác nhau. Dưới đây là phân tích chi tiết các lớp trong mô hình:

Đầu vào: Dữ liệu đầu vào của mô hình là ảnh có kích thước 96x128 pixel và chỉ có một kênh màu (grayscale), với kích thước đầu vào là (96, 128, 1). Các ảnh này sẽ được xử lý qua các lớp Conv2D để trích xuất các đặc trưng không gian cơ bản và phức tạp từ ảnh.

Các lớp Conv2D + BatchNorm + ReLU + MaxPooling + Dropout: Mô hình sử dụng ba khối Conv2D liên tiếp, mỗi khối bao gồm các lớp Conv2D, BatchNormalization, ReLU Activation, MaxPooling và Dropout. Số lượng filters trong các lớp Conv2D là 32, 64 và 128.

Khối 1: Lớp Conv2D sử dụng 32 filters với kích thước kernel 3x3, giúp phát hiện các đặc trưng cơ bản của ảnh đầu vào. Sau đó, BatchNormalization được sử dụng để chuẩn hóa đầu ra của lớp Conv2D, giúp tăng tốc và ổn định quá trình huấn luyện. ReLU activation tạo tính phi tuyến tính cho mô hình, trong khi MaxPooling giảm kích thước ảnh và số lượng tham số cần tính toán. Dropout giúp giảm thiểu hiện tượng overfitting trong quá trình huấn luyện.

Khối 2 và 3: Tương tự như khối 1 nhưng với số lượng filters tăng dần lên 64 và 128, giúp mô hình phát hiện các đặc trưng phức tạp hơn. Sau mỗi lớp Conv2D, các bước BatchNorm, ReLU, MaxPooling và Dropout được áp dụng để ổn định và giảm overfitting.

Sau khi qua các khối Conv2D, kích thước của ảnh đầu ra giảm dần từ (96, 128, 1) xuống (12, 16, 128), nhờ vào MaxPooling.

Lớp Reshape: Để chuyển dữ liệu qua các lớp LSTM, tensor 4 chiều được chuyển thành 2 chiều. Kích thước của tensor sau khi reshape là (None, 16, 1536), trong đó 16 là số bước thời gian và mỗi bước có 1536 đặc trưng, chuẩn bị cho việc học các đặc trưng theo chuỗi thời gian.

Lớp Bidirectional LSTM (x2): Mô hình sử dụng hai lớp LSTM hai chiều (Bidirectional LSTM), mỗi lớp có 64 units. LSTM hai chiều giúp mô hình học các đặc

trưng từ cả quá khứ và tương lai trong chuỗi dữ liệu. Đây là các lớp quan trọng trong việc học các đặc trưng chuỗi và thời gian từ dữ liệu đầu vào. Sau khi qua hai lớp LSTM, đầu ra của mô hình sẽ là các đặc trưng chuỗi đã được học từ ảnh.

Lớp Dense + BatchNorm + ReLU + Dropout: Đầu ra từ LSTM sẽ được đưa qua một lớp Dense với 64 units. Lớp này giúp học thêm các đặc trưng phi tuyến tính từ đầu ra của LSTM. BatchNormalization được áp dụng để ổn định quá trình huấn luyện, ReLU Activation tạo tính phi tuyến tính và Dropout giúp giảm overfitting trong quá trình huấn luyện.

Lớp Output Dense (Softmax với 62 lớp đầu ra): Cuối cùng, đầu ra của lớp Dense được đưa qua một lớp Dense với 62 units để phân loại ảnh vào một trong 62 lớp. Hàm activation Softmax được sử dụng để chuyển đổi các giá trị đầu ra thành xác suất, từ đó giúp mô hình phân loại ảnh vào lớp có xác suất cao nhất.

1. Huấn luyện mô hình

Các tham số và kỹ thuật huấn luyện bao gồm:

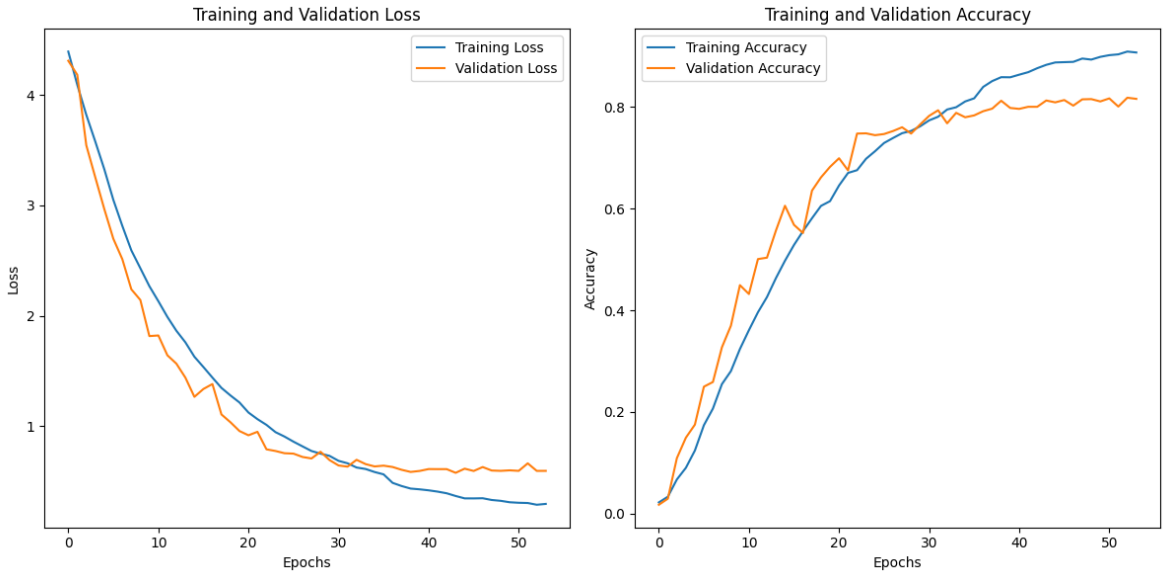
- Loss function: sparse_categorical_crossentropy do nhãn là số nguyên.
- Optimizer: Adam (learning rate = 0.0005)
- Batch size: 32
- Epochs: Tối đa 200, nhưng dừng sớm với EarlyStopping nếu không cải thiện sau 10 epochs.
- ReduceLROnPlateau: Giảm learning rate nếu validation loss không giảm sau 4 epochs.

Kết quả huấn luyện:

- Training Accuracy đạt gần 90%

- Validation Accuracy đạt ~82%
- Không có dấu hiệu overfitting nghiêm trọng nhờ sử dụng Dropout và BatchNormalization.

Biểu đồ loss và accuracy cho thấy quá trình huấn luyện ổn định, loss giảm đều, và accuracy tăng dần theo mỗi epoch.



Hình 1: Kết quả huấn luyện

2. Đánh giá và kiểm thử mô hình

1. Đánh giá tổng thể

Sau khi huấn luyện mô hình CRNN, quá trình kiểm thử được thực hiện trên tập dữ liệu test gồm 2.728 ảnh chứa ký tự viết tay. Mô hình đạt được độ chính xác tổng thể là 81.12% trên tập kiểm thử, cho thấy khả năng tổng quát hóa tương đối tốt, đặc biệt trong điều kiện không sử dụng thêm kỹ thuật data augmentation hay ensemble.

2. Báo cáo phân loại (Classification Report)

	precision	recall	f1-score	support
0	0.459459459	0.386363636	0.419753086	44

1	0.710526316	0.613636364	0.658536585	44
2	0.925	0.840909091	0.880952381	44
3	0.716666667	0.977272727	0.826923077	44
4	0.708333333	0.772727273	0.739130435	44
5	0.863636364	0.863636364	0.863636364	44
6	0.764705882	0.886363636	0.821052632	44
7	0.931818182	0.931818182	0.931818182	44
8	0.772727273	0.772727273	0.772727273	44
9	0.735294118	0.568181818	0.641025641	44
A_caps	0.911111111	0.931818182	0.921348315	44
B_caps	0.836734694	0.931818182	0.88172043	44
C_caps	0.755555556	0.772727273	0.764044944	44
D_caps	0.973684211	0.840909091	0.902439024	44
E_caps	1	0.909090909	0.952380952	44
F_caps	1	0.977272727	0.988505747	44
G_caps	0.918918919	0.772727273	0.839506173	44
H_caps	0.930232558	0.909090909	0.91954023	44
I_caps	0.559322034	0.75	0.640776699	44
J_caps	0.857142857	0.954545455	0.903225806	44
K_caps	0.837209302	0.818181818	0.827586207	44
L_caps	0.936170213	1	0.967032967	44
M_caps	0.955555556	0.977272727	0.966292135	44
N_caps	0.930232558	0.909090909	0.91954023	44
O_caps	0.578947368	0.75	0.653465347	44
P_caps	0.813953488	0.795454545	0.804597701	44
Q_caps	0.953488372	0.931818182	0.942528736	44
R_caps	1	0.909090909	0.952380952	44
S_caps	0.716981132	0.863636364	0.783505155	44

T_caps	0.976744186	0.954545455	0.965517241	44
U_caps	0.893617021	0.954545455	0.923076923	44
V_caps	0.795918367	0.886363636	0.838709677	44
W_caps	0.794117647	0.613636364	0.692307692	44
X_caps	0.885714286	0.704545455	0.784810127	44
Y_caps	0.781818182	0.977272727	0.868686869	44
Z_caps	0.775510204	0.863636364	0.817204301	44
a	0.88372093	0.863636364	0.873563218	44
b	0.777777778	0.954545455	0.857142857	44
c	0.625	0.681818182	0.652173913	44
d	0.872340426	0.931818182	0.901098901	44
e	0.822222222	0.840909091	0.831460674	44
f	0.822222222	0.840909091	0.831460674	44
g	0.693877551	0.772727273	0.731182796	44
h	0.904761905	0.863636364	0.88372093	44
i	0.952380952	0.909090909	0.930232558	44
j	0.928571429	0.886363636	0.906976744	44
k	0.833333333	0.795454545	0.813953488	44
l	0.888888889	0.545454545	0.676056338	44
m	0.866666667	0.886363636	0.876404494	44
n	0.755102041	0.840909091	0.795698925	44
o	0.565217391	0.590909091	0.577777778	44
p	0.707317073	0.659090909	0.682352941	44
q	0.880952381	0.840909091	0.860465116	44
r	0.823529412	0.636363636	0.717948718	44
s	0.642857143	0.409090909	0.5	44
t	0.880952381	0.840909091	0.860465116	44
u	0.921052632	0.795454545	0.853658537	44

v	0.825	0.75	0.785714286	44
w	0.62962963	0.772727273	0.693877551	44
x	0.680851064	0.727272727	0.703296703	44
y	0.947368421	0.818181818	0.87804878	44
z	0.568181818	0.568181818	0.568181818	44
accuracy	0.811217009	0.811217009	0.811217009	0.811217009
macro avg	0.817042308	0.811217009	0.809987098	2728
weighted avg	0.817042308	0.811217009	0.809987098	2728

Bảng 3 Báo cáo phân loại

Báo cáo phân loại chi tiết cho thấy mô hình có độ chính xác cao đối với hầu hết các lớp. Một số lớp ký tự đạt precision, recall và F1-score rất cao (trên 0.95), đặc biệt là:

- F_caps, E_caps, T_caps, M_caps, R_caps, L_caps, i, y, J_caps: Các ký tự này có nét rõ ràng, dễ phân biệt, nên mô hình dễ học và phân loại chính xác.
- Trong khi đó, một số ký tự có kết quả kém hơn (F1-score dưới 0.6), ví dụ: 0, s, z, o, l, W_caps: Đây là các ký tự có hình dạng tương tự nhau, hoặc dễ bị viết nhầm, gây khó khăn trong việc phân biệt.

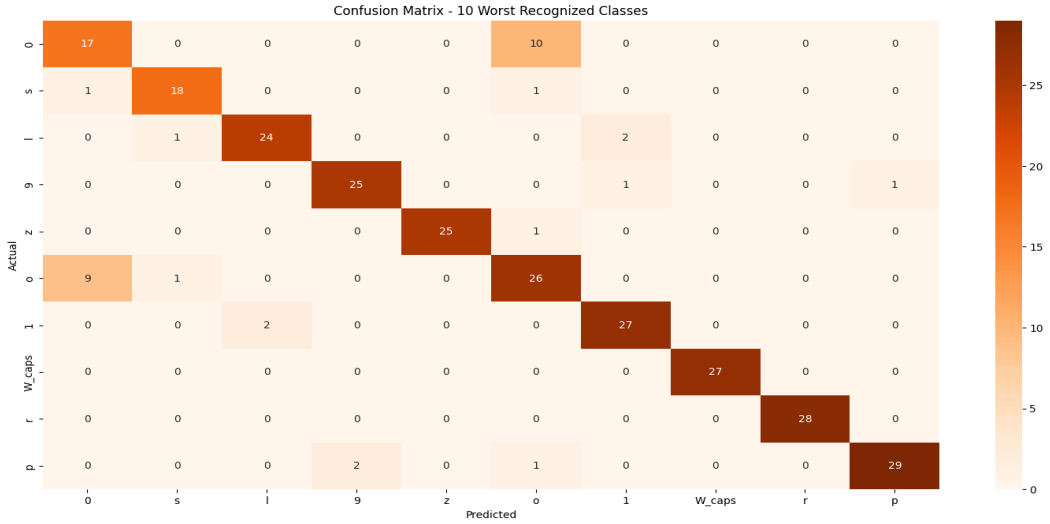
Dưới đây là một số ví dụ về F1-score của các lớp nổi bật:

Ký tự	F1-score	Nhận xét
F_caps	0.99	Gần như hoàn hảo
M_caps	0.97	Nhận diện rất tốt
s	0.50	Rất dễ nhầm với S hoặc 5

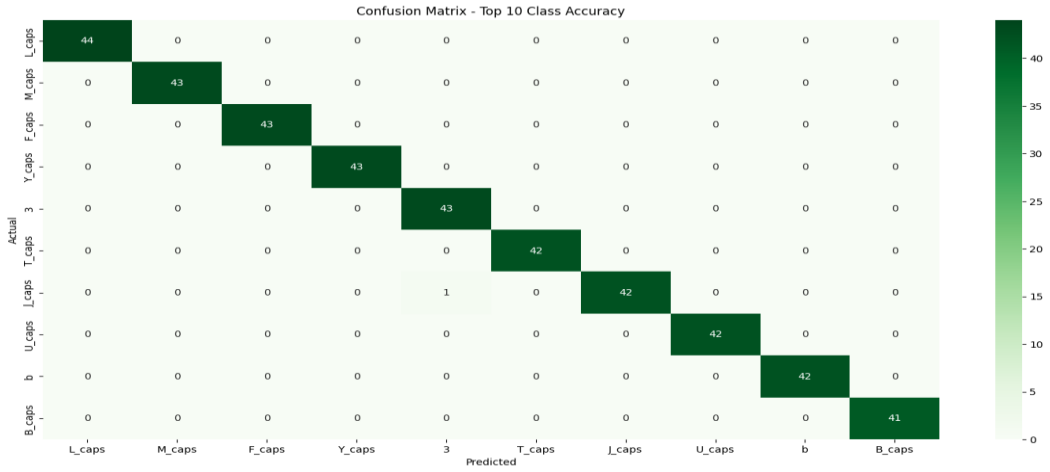
Ký tự	F1-score	Nhận xét
z	0.57	Dễ nhầm với 2, Z

Bảng 4 F1 - score của 1 số lớp

3. Ma trận nhầm lẫn (Confusion Matrix)



Hình 2: Confusion Matrix



Hình 3: Confusion Matrix

Để phân tích sâu hơn, mô hình được đánh giá qua ma trận nhầm lẫn. Đặc biệt, 2 ma trận nhỏ được vẽ cho:

- 10 ký tự nhận diện tốt nhất (F1-score cao nhất)
- 10 ký tự nhận diện kém nhất (F1-score thấp nhất)

Qua ma trận, có thể quan sát rõ:

- Mô hình thường nhầm lẫn giữa các ký tự viết tay có nét tương đồng như 0 và O, s và S, l và 1, z và 2.
- Trong các lớp tốt nhất, hầu như mô hình chỉ dự đoán nhầm 1–2 ảnh (tỉ lệ sai cực thấp).

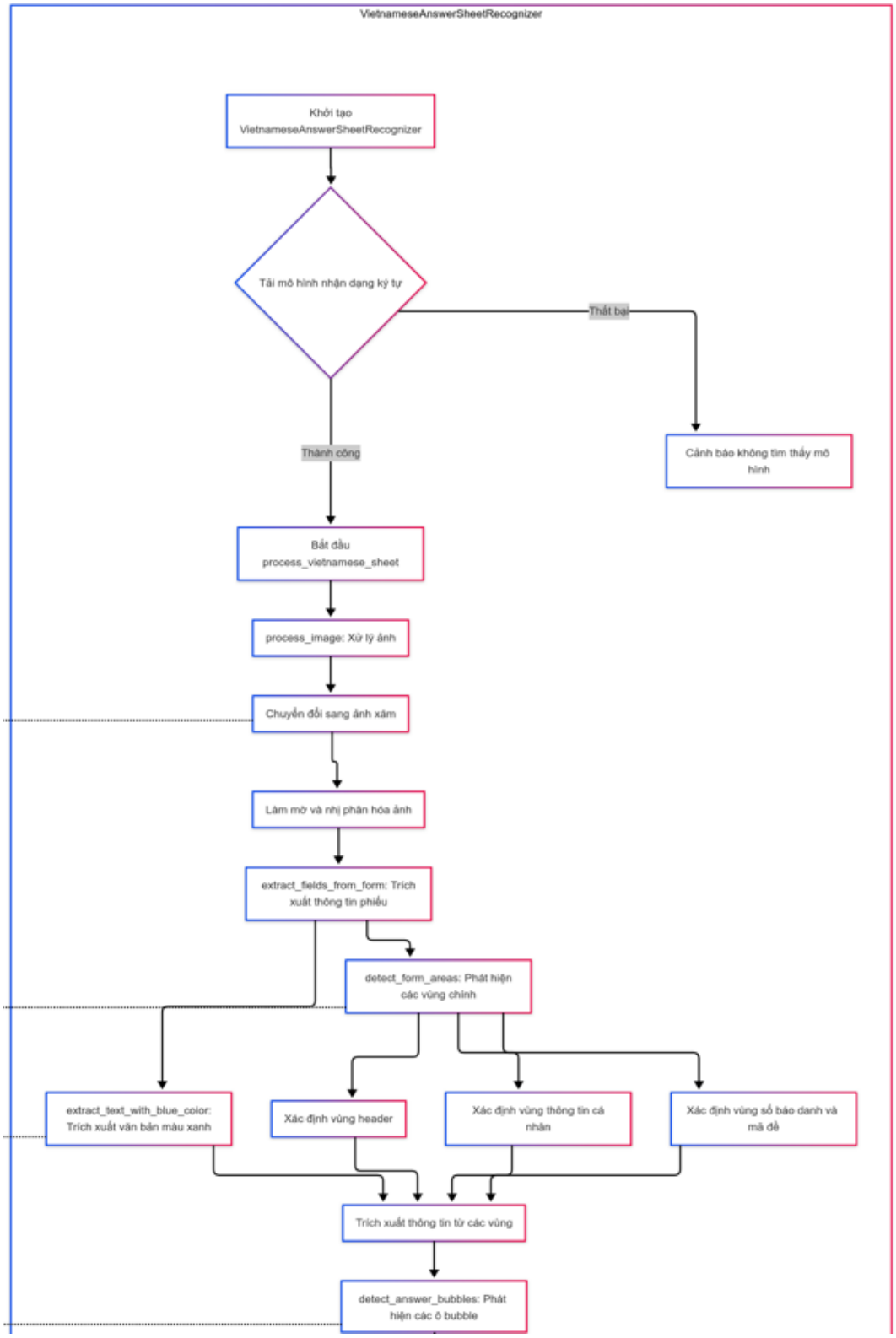
4. Trực quan hóa lỗi

Mô hình đã được kiểm tra bằng cách hiển thị các ảnh dự đoán sai, giúp trực quan hóa các lỗi thường gặp. Những sai sót phổ biến là:

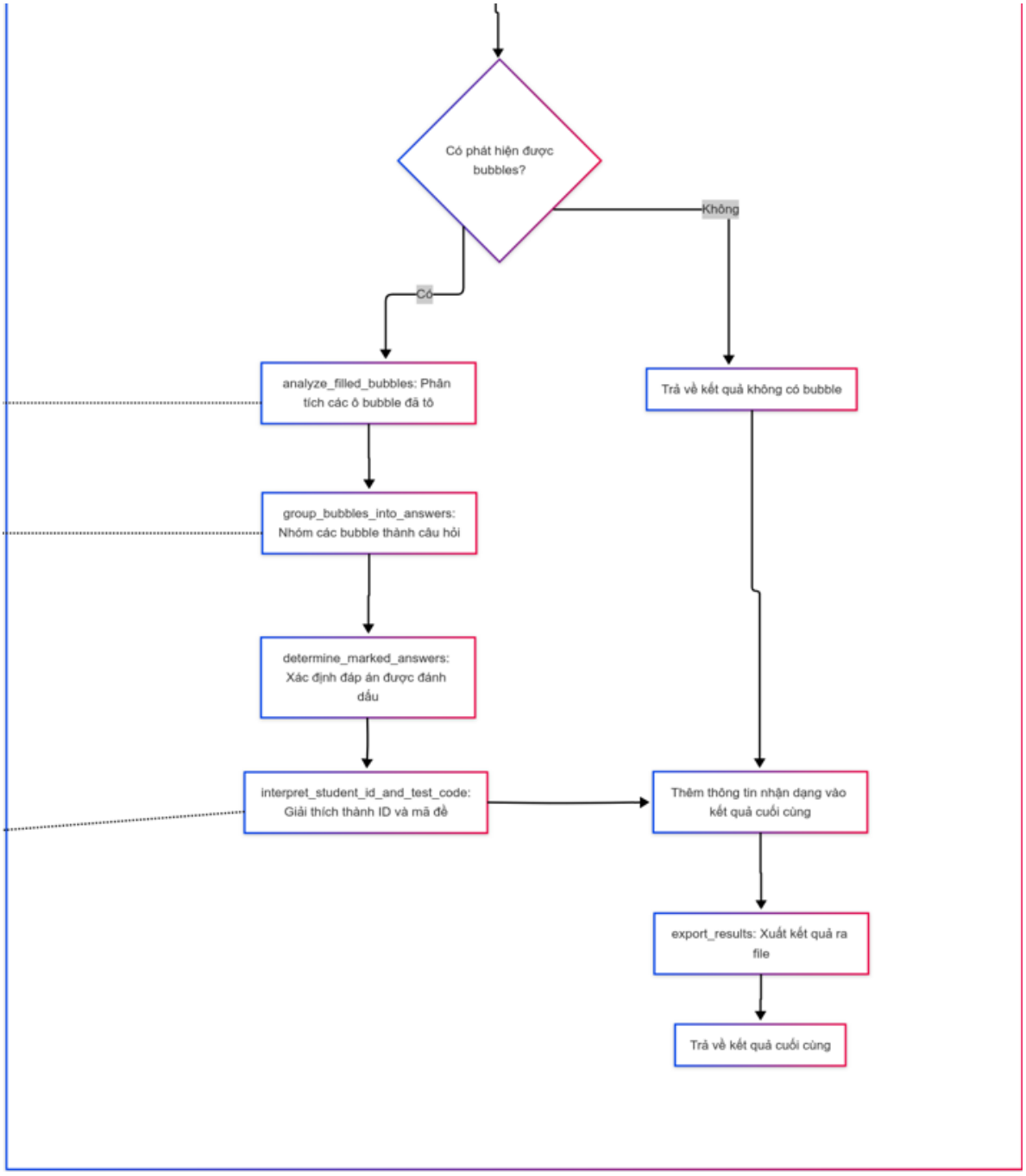
- Ký tự viết tay không rõ nét hoặc viết sai chuẩn.
- Ký tự bị nghiêng, lệch hoặc đè nét (ví dụ: g viết giống q, r bị nhận nhầm thành n).
- Nhiễu ảnh hoặc độ phân giải thấp ảnh hưởng đến nhận diện.

Số lượng ảnh dự đoán sai là 515/2728 (~18.8%), trong đó đa phần tập trung ở các ký tự dễ gây nhầm lẫn đã đề cập.

3. Tóm tắt cách mô hình hoạt động



Hình 4: tóm tắt mô hình hoạt động (1)



Hình 5: tóm tắt mô hình hoạt động (2)

Khởi tạo và cấu hình ban đầu

Lớp VietnameseAnswerSheetRecognizer được thiết kế để hỗ trợ hai phương pháp nhận dạng văn bản: bằng mô hình học sâu (CNN) và bằng OCR (Tesseract). Trong hàm khởi tạo __init__, hệ thống sẽ kiểm tra xem người dùng có cung cấp đường dẫn tới mô

hình CNN (character_model_path) và bộ mã hóa nhãn (encoder_path) hay không. Nếu các tệp này tồn tại và được nạp thành công, hệ thống sẽ sử dụng mô hình CNN cho việc nhận dạng ký tự. Ngược lại, nếu không có hoặc có lỗi khi tải mô hình, hệ thống sẽ chuyển sang sử dụng Tesseract OCR, với điều kiện pytesseract đã được cài đặt trong môi trường. Người dùng cũng có thể ép buộc hệ thống luôn sử dụng CNN thông qua tham số force_cnn.

Tiền xử lý ảnh đầu vào

Phương thức process_image đảm nhận vai trò tiền xử lý ảnh đầu vào trước khi trích xuất thông tin. Ảnh có thể được đưa vào dưới dạng đường dẫn tệp hoặc mảng numpy. Sau khi nạp ảnh, hệ thống chuyển đổi ảnh sang ảnh xám để đơn giản hóa dữ liệu, áp dụng làm mờ Gaussian để giảm nhiễu, và thực hiện nhị phân hóa ảnh bằng kỹ thuật adaptive thresholding. Kết quả là một ảnh nhị phân rõ ràng giữa nền và vùng văn bản, tạo điều kiện thuận lợi cho các bước xử lý tiếp theo như trích xuất văn bản và nhận dạng ký tự.

Phát hiện các vùng có thể chứa văn bản

Sau khi có ảnh nhị phân, phương thức detect_form_areas sẽ tìm các vùng tiềm năng trong ảnh có thể chứa nội dung cần nhận dạng, chẳng hạn như các ô trong phiếu trả lời. Phương pháp này sử dụng hàm cv2.findContours để phát hiện các đường viền (contours) trong ảnh nhị phân, sau đó lọc ra các vùng có diện tích lớn hơn 1% tổng diện tích ảnh, loại bỏ nhiễu và các chi tiết nhỏ không quan trọng. Các vùng này sau đó được sắp xếp theo vị trí từ trên xuống, giúp duy trì trật tự đọc giống như cách con người quét thông tin từ một văn bản.

Trích xuất văn bản màu xanh (mực viết)

Một điểm nổi bật của hệ thống là khả năng trích xuất văn bản viết tay bằng mực xanh – thường là màu mực phổ biến trong phiếu trắc nghiệm. Phương thức extract_text_with_blue_color chuyển ảnh từ không gian màu BGR sang HSV để dễ dàng lọc theo màu sắc. Hệ thống tạo mặt nạ chỉ giữ lại các pixel thuộc dải màu xanh lam, sau đó tách các vùng này ra khỏi ảnh gốc. Sau khi trích xuất, các vùng chứa văn bản xanh được chuyển sang ảnh xám, nhị phân hóa và làm sạch bằng phép toán morphological closing.

Kết quả là một ảnh chỉ chứa văn bản viết tay màu xanh, đã được làm rõ để nhận dạng tốt hơn. Phương pháp này rất hữu ích trong việc loại bỏ các phần in sẵn hoặc nhiễu không liên quan.

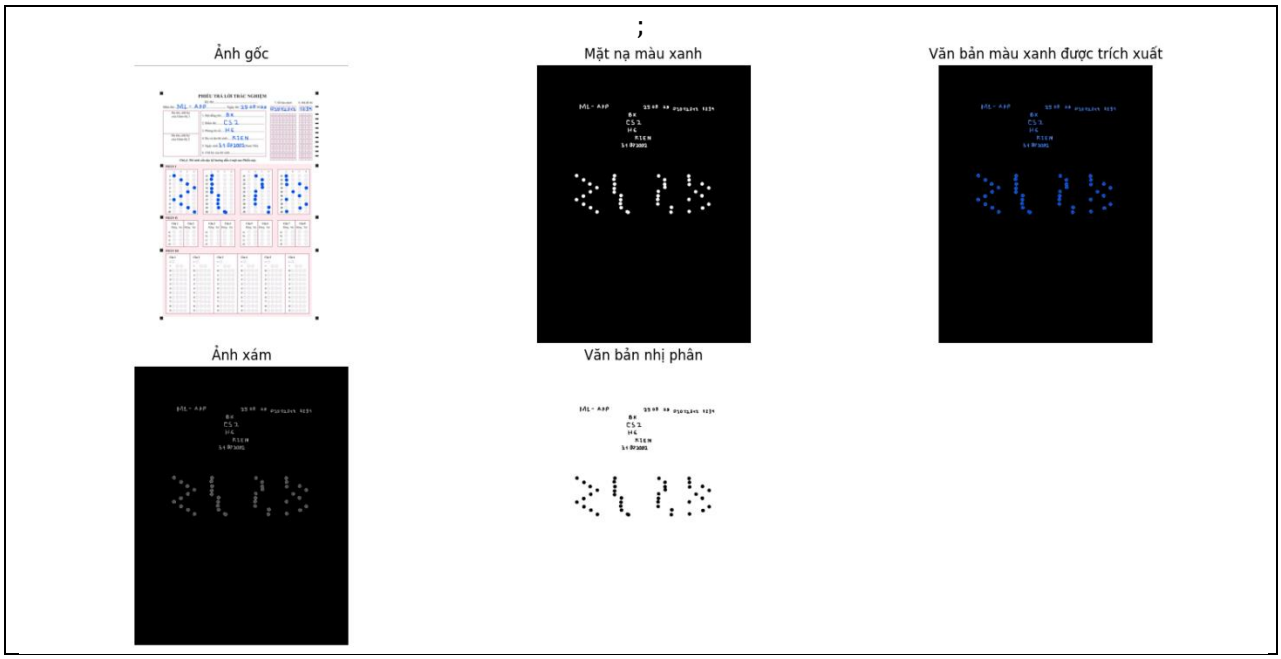
Nhận dạng ký tự bằng mô hình học sâu CNN

Nếu mô hình CNN đã được nạp thành công, phương thức recognize_text_with_cnn sẽ thực hiện việc nhận dạng văn bản từ ảnh đã qua xử lý. Trước tiên, hệ thống tìm các contour đại diện cho từng ký tự hoặc nhóm ký tự. Các contour nhỏ không đáng kể sẽ bị loại bỏ. Sau đó, hệ thống nhóm các contour theo dòng dựa trên vị trí dọc (tọa độ y) và chiều cao trung bình, cho phép phân tách văn bản thành từng dòng. Mỗi ký tự được cắt ra, resize về kích thước chuẩn (ví dụ 128x96), chuẩn hóa giá trị pixel và reshape về dạng tensor để phù hợp với mô hình. Sau đó, mô hình CNN dự đoán lớp cho từng ký tự, kết quả được giải mã thành văn bản bằng LabelEncoder. Văn bản được tái cấu trúc lại theo từng dòng như trong ảnh gốc.

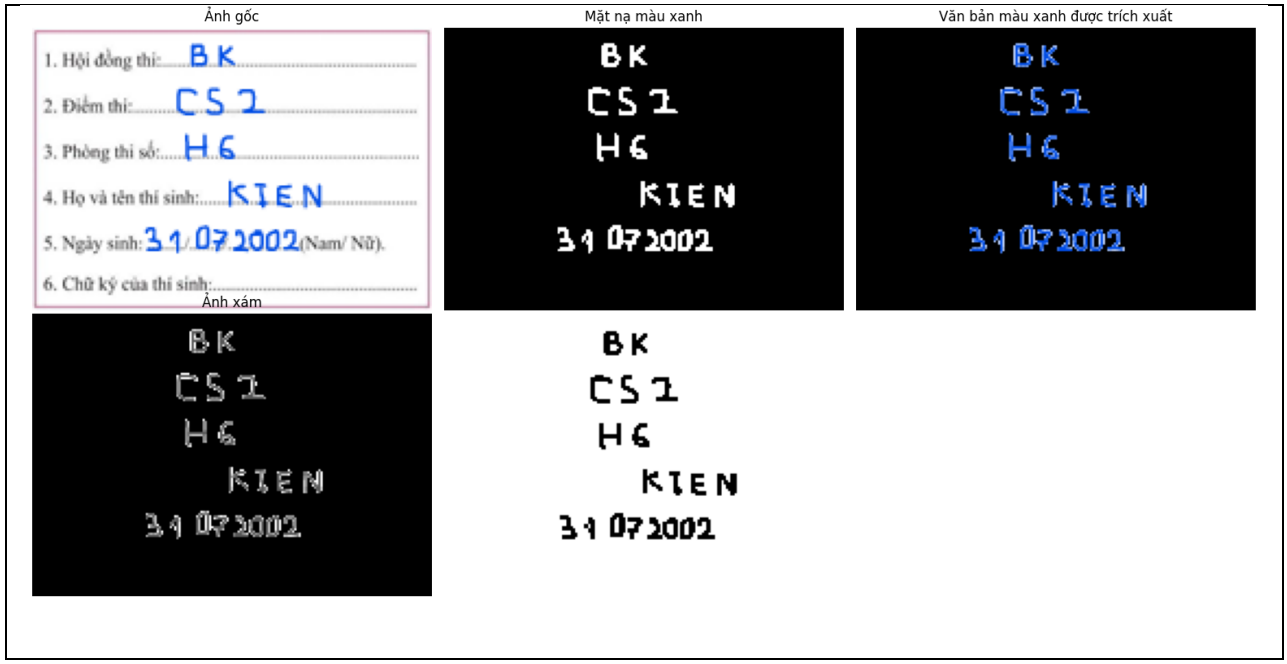
Nhận dạng văn bản bằng Tesseract OCR

Trong trường hợp không có mô hình CNN, hoặc người dùng không ép buộc dùng CNN, hệ thống sẽ sử dụng phương pháp nhận dạng quang học ký tự bằng Tesseract OCR thông qua phương thức perform_ocr. OCR sẽ phân tích toàn bộ ảnh văn bản đầu vào và chuyển đổi trực tiếp thành chuỗi văn bản, hỗ trợ tốt cho ngôn ngữ tiếng Việt nếu Tesseract đã được cấu hình với gói ngôn ngữ phù hợp (lang='vie'). Mặc dù độ chính xác có thể không cao bằng mô hình học sâu tùy chỉnh, đây vẫn là phương pháp dự phòng hiệu quả trong nhiều trường hợp không có dữ liệu huấn luyện đủ mạnh.

4. Kết quả nhận diện



Hình 6: chuyển đổi ảnh sang các chế độ HSV, grayscale và lấy ngưỡng



Hình 7: chuyển đổi ảnh sang các chế độ HSV, grayscale và lấy ngưỡng ở phần vùng thông tin

III. KẾT LUẬN

1. Ưu điểm

Thiết kế linh hoạt, dễ mở rộng và tái sử dụng

Lớp VietnameseAnswerSheetRecognizer được xây dựng với cấu trúc rõ ràng và có tổ chức, cho phép tích hợp mô hình học sâu (CNN) hoặc sử dụng Tesseract như một giải pháp thay thế. Thiết kế này giúp người dùng dễ dàng chuyển đổi giữa các phương pháp nhận diện, đồng thời tạo điều kiện để mở rộng hệ thống bằng các mô hình nâng cao hơn như CRNN hoặc Transformer. Các phương thức được phân chia rõ ràng theo chức năng — từ tiền xử lý ảnh, trích xuất ký tự, đến dự đoán kết quả — làm cho mã nguồn dễ bảo trì, dễ kiểm thử và dễ tích hợp vào những hệ thống lớn hơn như ứng dụng GUI hoặc web.

Tiền xử lý ảnh tối ưu hóa cho văn bản tiếng Việt trên giấy

Một trong những điểm mạnh đáng chú ý là hệ thống tiền xử lý ảnh được thiết kế phù hợp với đặc điểm của văn bản viết tay trên phiếu trả lời ở Việt Nam. Việc chuyển ảnh sang thang xám, áp dụng bộ lọc Gaussian và adaptive threshold giúp làm nổi bật nét chữ, đồng thời giảm nhiễu nền hiệu quả. Đặc biệt, lớp này có tích hợp bộ lọc màu HSV để ưu tiên vùng chữ viết bằng mực xanh — đây là chi tiết thực tế, thường thấy trong bài thi và phiếu trả lời ở Việt Nam. Cách tiếp cận này làm tăng đáng kể độ chính xác của việc nhận diện ký tự, đặc biệt trong môi trường có nhiều nhiễu hoặc ảnh chụp không lý tưởng.

Nhận diện ký tự đơn giản nhưng hiệu quả với phân đoạn contour

Thay vì phụ thuộc vào các kỹ thuật phân đoạn phức tạp, lớp sử dụng phương pháp tìm contour trong ảnh nhị phân để trích xuất các ký tự riêng lẻ. Phương pháp này nhẹ, nhanh và rất phù hợp với bài toán nhận diện ký tự rời rạc như trên phiếu trắc nghiệm. Sau khi tách các ký tự, hệ thống còn sắp xếp lại các vùng theo trục hoành để đảm bảo đúng thứ tự từ trái sang phải. Nhờ cách xử lý hợp lý này, hệ thống đảm bảo đầu ra là chuỗi ký tự được nhận diện một cách chính xác, đồng thời duy trì được tốc độ xử lý cao và độ ổn định trong các tình huống thực tế.

2. Nhược điểm

Phụ thuộc mạnh vào điều kiện ảnh đầu vào

Hệ thống tiền xử lý ảnh như chuyển sang grayscale, làm mờ, và adaptive thresholding chỉ hoạt động tốt trong điều kiện ảnh chụp rõ nét, ánh sáng đều và độ tương phản tốt. Nếu ảnh bị nhòe, nghiêng, hoặc có bóng, chất lượng phân đoạn vùng văn bản và nhận dạng ký tự có thể giảm mạnh. Điều này khiến hệ thống thiếu tính ổn định trong môi trường thực tế với thiết bị quét không đồng đều.

Lọc mực xanh dễ bị sai lệch do ánh sáng hoặc sai màu

Việc lọc văn bản mực xanh sử dụng khoảng giá trị HSV cố định. Trong thực tế, màu mực có thể thay đổi nhẹ theo loại bút, hoặc bị ảnh hưởng bởi điều kiện ánh sáng, bóng đổ, và chất lượng ảnh. Do đó, vùng văn bản mực xanh có thể bị bỏ sót hoặc ngược lại, nhiễu màu nền cũng bị nhận nhầm là văn bản. Hệ thống không có cơ chế tự điều chỉnh hoặc học được ngưỡng màu động.

Không sử dụng mô hình phát hiện ký tự/dòng nâng cao

Việc nhóm contour để chia dòng và tách ký tự chỉ dựa trên vị trí hình học (toạ độ y, x, chiều cao). Điều này làm hệ thống dễ bị sai dòng, gộp hoặc tách nhầm ký tự, nhất là khi chữ viết tay bị lệch dòng hoặc sát nhau. Thiếu các thuật toán phân đoạn ký tự/dòng mạnh hơn như CTPN, EAST, hoặc segmentation bằng học sâu, hệ thống dễ bị lỗi trong trường hợp văn bản viết tay lộn xộn.

Mô hình CNN thiếu kiểm soát lỗi và chưa xử lý ngoại lệ

Hệ thống giả định tất cả contour đều là ký tự có thể phân loại được, nhưng không có bước kiểm tra hậu kỳ hoặc đánh giá độ tin cậy (confidence threshold). Điều này có thể dẫn đến nhận dạng sai hoặc nhiễu, đặc biệt khi ảnh chứa đường nét, vết mực thừa hoặc ký hiệu đặc biệt mà mô hình chưa từng thấy. Ngoài ra, nếu mô hình CNN không hoạt động đúng (ví dụ lỗi nạp model, lỗi shape ảnh...), hệ thống chỉ in cảnh báo chứ không chuyển đổi linh hoạt.

3. Hướng cải thiện

Tích hợp mô hình CRNN với CTC Loss để nhận diện chuỗi

Thay vì trích xuất từng ký tự riêng lẻ, có thể chuyển sang cách tiếp cận nhận diện toàn dòng bằng CRNN (Convolutional Recurrent Neural Network) kết hợp với CTC Loss. Phương pháp này sẽ cho phép hệ thống nhận diện trực tiếp toàn bộ dòng văn bản mà không cần tách ký tự, từ đó giảm sai sót khi phân đoạn và duy trì mối quan hệ ngữ cảnh.

Bổ sung bước căn chỉnh ảnh (deskewing + perspective correction)

Để cải thiện độ chính xác trong trường hợp ảnh bị nghiêng hoặc chụp lệch góc, có thể thêm bước tự động căn chỉnh ảnh bằng cách sử dụng Hough Transform hoặc phát hiện các đường thẳng chính trong ảnh để hiệu chỉnh phối cảnh.

Áp dụng kỹ thuật tăng cường dữ liệu trong huấn luyện mô hình

Nếu sử dụng mô hình CNN tùy chỉnh để nhận diện ký tự, nên áp dụng các kỹ thuật data augmentation như xoay nhẹ, co giãn, nhiễu Gaussian hoặc thay đổi độ tương phản để tăng độ bền của mô hình trước các biến dạng ảnh thực tế.

--HẾT--

IV. Danh mục tài liệu tham khảo

1. Alex Graves – "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks"
   https://www.cs.toronto.edu/~graves/icdar_2009.pdf
2. Shi, Bai, Yao – "An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition" (CRNN)
   https://arxiv.org/abs/1507.05717
3. Baek et al. – "What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis"
   https://arxiv.org/abs/1904.01906
4. Zhong et al. – "Handwritten Chinese Text Recognition using Semi-supervised Learning with Spatial Transformer Network"
   https://arxiv.org/abs/1910.12006