ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC
MÁY HỌC - CS114

ĐỀ TÀI : NHẬN DẠNG CHỮ SỐ VIẾT TAY VÀ DỰ ĐOÁN ĐIỂM IT001

Giảng viên hướng dẫn : ThS. Phạm Nguyễn Trường An
PGS.TS. Lê Đình Duy

Lớp : CS114.P21

Sinh viên thực hiện :

Đỗ Minh Hội	23520550
Nguyễn Hoàng Kha	23520667
Nguyễn Hải Thiện	23521481

LinkRepository:
https://github.com/adamwhite625/CS114.P21_Final_Project.git

Thành phố Hồ Chí Minh, tháng 06 năm 2025

MỤC LỤC

LỜI NÓI ĐẦU

Trong khuôn khổ môn học Máy học - CS114, giảng viên đã giao cho toàn bộ lớp hai đề tài nhằm thực hành và vận dụng các kiến thức đã học vào các bài toán thực tiễn. Hai đề tài đó là: (1) Nhận diện chữ số viết tay từ ảnh và (2) Dự đoán điểm môn học IT001 dựa trên dữ liệu học tập. Đây đều là những bài toán có tính ứng dụng cao và giúp sinh viên hiểu sâu hơn về cách triển khai mô hình học máy trong thực tế.

Bài toán nhận diện chữ số viết tay là một ví dụ kinh điển trong lĩnh vực học sâu, thường được dùng để đánh giá khả năng nhận diện ảnh, đặc biệt trong các hệ thống OCR (nhận diện ký tự quang học). Việc giải quyết bài toán này giúp nhóm sinh viên chúng em làm quen với các mô hình học sâu như CNN và transfer learning, cũng như các bước xử lý dữ liệu ảnh từ đầu đến cuối.

Bài toán thứ hai – dự đoán điểm môn học IT001 – mang ý nghĩa giáo dục rõ rệt. Dựa vào các đặc trưng đầu vào như điểm bài Wecode, thời gian nộp bài,… mô hình có thể dự đoán các cột điểm của sinh viên. Bài toán này không chỉ giúp rèn luyện kỹ năng tiền xử lý và huấn luyện mô hình hồi quy, mà còn giúp sinh viên hiểu được cách khai thác dữ liệu giáo dục để hỗ trợ công tác giảng dạy và học tập.

Thông qua hai đề tài này, nhóm chúng em không chỉ áp dụng được kiến thức đã học về học máy, mà còn hiểu rõ hơn về quy trình xây dựng một hệ thống từ thu thập, xử lý dữ liệu, huấn luyện mô hình cho đến đánh giá kết quả.

Chúng em xin chân thành cảm ơn ThS. Phạm Nguyễn Trường An đã tận tình giảng dạy và giao những đề tài có ý nghĩa, giúp chúng em có cơ hội cọ xát với bài toán thực tế. Nhờ đó, chúng em đã nâng cao được tư duy kỹ thuật, khả năng lập trình cũng như tinh thần làm việc nhóm.

Mặc dù đã cố gắng hoàn thành đồ án một cách nghiêm túc, nhưng không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý của thầy để đồ án được hoàn thiện hơn.

Thành phố Hồ Chí Minh, tháng 6 năm 2025

PHẦN A: ĐỒ ÁN NHẬN DIỆN CHỮ SỐ VIẾT TAY

CHƯƠNG 0: CÁC THAY ĐỔI SO VỚI KHI VẤN ĐÁP

Trong buổi vấn đáp ban đầu, nhóm đã trình bày kết quả của 2 model đầu tiên (Model 1,Model 2) với độ chính xác lần lượt dưới 5% và 55% trên tập kiểm tra. Sau khi nhận được phản hồi từ thầy, nhóm đã thực hiện các cải tiến sau:

- Thêm hai model mới sử dụng kỹ thuật transfer learning với kiến trúc pretrained: MobileNetV3-Large (xem Chương 3, mục 3.4) và ResNet50 (xem Chương 3, mục 3.5), đạt độ chính xác lần lượt 82% và 79%.
- Tối ưu hóa pipeline xử lý dữ liệu bằng cách bổ sung các phép biến đổi dữ liệu như RandomHorizontalFlip và RandomRotation để tăng cường độ đa dạng của tập huấn luyện (xem Chương 2).
- Cải thiện quy trình huấn luyện bằng cách áp dụng early stopping và lưu model tốt nhất dựa trên loss trên tập kiểm tra.
- Nâng cao quá trình đánh giá bằng cách vẽ biểu đồ loss và accuracy qua từng epoch, giúp theo dõi hiệu suất chi tiết của từng model (xem Chương 3 và Chương 4).

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN VÀ DỮ LIỆU

1. Bối cảnh và ý nghĩa

Nhận diện chữ số viết tay là một bài toán kinh điển trong lĩnh vực học sâu, với ứng dụng thực tiễn trong các hệ thống như nhận diện mã bưu điện, xử lý biểu mẫu, và giao diện người-máy. Bài toán này thách thức do sự đa dạng trong phong cách viết tay và chất lượng hình ảnh.

2. Nguồn dữ liệu

Nhóm đã đóng góp dữ liệu vào kho GitHub, bao gồm các hình ảnh chữ số viết tay ở định dạng .jpg, .png, và .heic, được tổ chức theo mã sinh viên (ví dụ: ??52????) và nhãn từ 0 đến 9. Tổng số ảnh đóng góp là 300 ảnh, giúp tăng cường độ đa dạng của tập dữ liệu.

CHƯƠNG 2: DATA PROCESSING

Dữ liệu được tải từ GitHub của lớp, bao gồm các hình ảnh chữ số viết tay ở các định dạng .jpg, .png, và .heic. Quá trình xử lý dữ liệu được thực hiện chi tiết như sau:

1. Kiểm tra và lọc dữ liệu

Dữ liệu được tải từ GitHub của lớp, bao gồm hàng nghìn hình ảnh chữ số viết tay. Nhóm đã thực hiện các bước kiểm tra như sau:

- Sử dụng ThreadPoolExecutor để kiểm tra tính hợp lệ của các file ảnh, đảm bảo chỉ giữ lại các file thuộc định dạng .jpg, .jpeg, .png, và .heic.
- Loại bỏ các file lỗi hoặc không đọc được. Kết quả, có tổng cộng 6947 ảnh hợp lệ.

2. Chia tập dữ liệu

Dữ liệu được chia thành:

- Tập huấn luyện: 80% tổng số ảnh, tương ứng với int(0.8 * len(valid_image_lists)).
- Tập kiểm tra: 20% còn lại.

Việc chia dữ liệu được thực hiện ngẫu nhiên với random.seed để đảm bảo tính tái lập.

3. Biến đổi dữ liệu

Các phép biến đổi dữ liệu được áp dụng để tăng cường tính tổng quát của mô hình:

- Tập huấn luyện:
  - Chuyển thành tensor với ToTenser().
  - Thay đổi kích thước thành 200x200 pixel.
  - Lật ngang ngẫu nhiên với xác suất 0.5 RandomHorizontalFlip.
  - Xoay ngẫu nhiên tối đa 10 độ RandomRotation.
  - Chuẩn hóa với mean=0.5, std=0.5 cho từng kênh màu.
- Tập kiểm tra: Chỉ áp dụng chuyển thành tensor, thay đổi kích thước, và chuẩn hóa tương tự.

4. Tạo Data Loader

Nhóm sử dụng DataLoader với các tham số:

- Batch size: 32 cho tập huấn luyện, 16 cho tập kiểm tra (đối với ResNet50).
- Shuffle: Bật cho tập huấn luyện, tắt cho tập kiểm tra.
- num_workers = 2 để tăng tốc độ tải dữ liệu.

5. Cân bằng lớp

Để xử lý mất cân bằng lớp, nhóm tính trọng số lớp với công thức:

$$w_i = \frac{1}{\text{số mẫu lớp } i + \epsilon}$$

Trong đó e = 10^-6 để tránh chia cho 0. Trọng số được chuẩn hóa để tổng bằng 10 và sử dụng trong hàm loss.

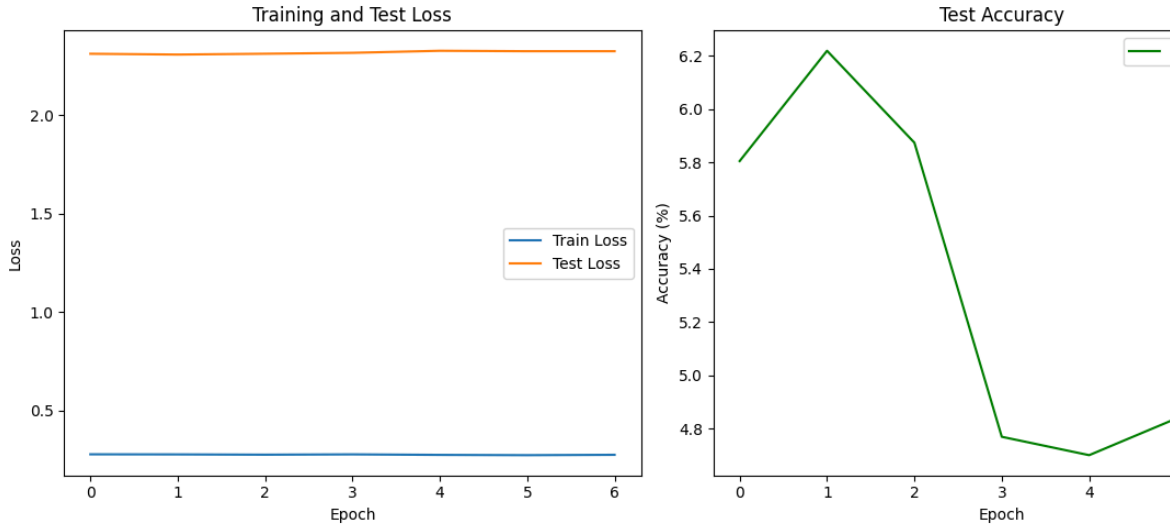
CHƯƠNG 3: HUẤN LUYỆN

1. Tổng quan quá trình huấn luyện

Nhóm đã thử nghiệm năm mô hình với các kiến trúc và tham số khác nhau, từ mạng CNN tự xây dựng đến các mô hình pretrained. Quá trình huấn luyện được thực hiện trên GPU NVIDIA Tesla T4 với framework PyTorch.

2. Model 1: CNN cơ bản

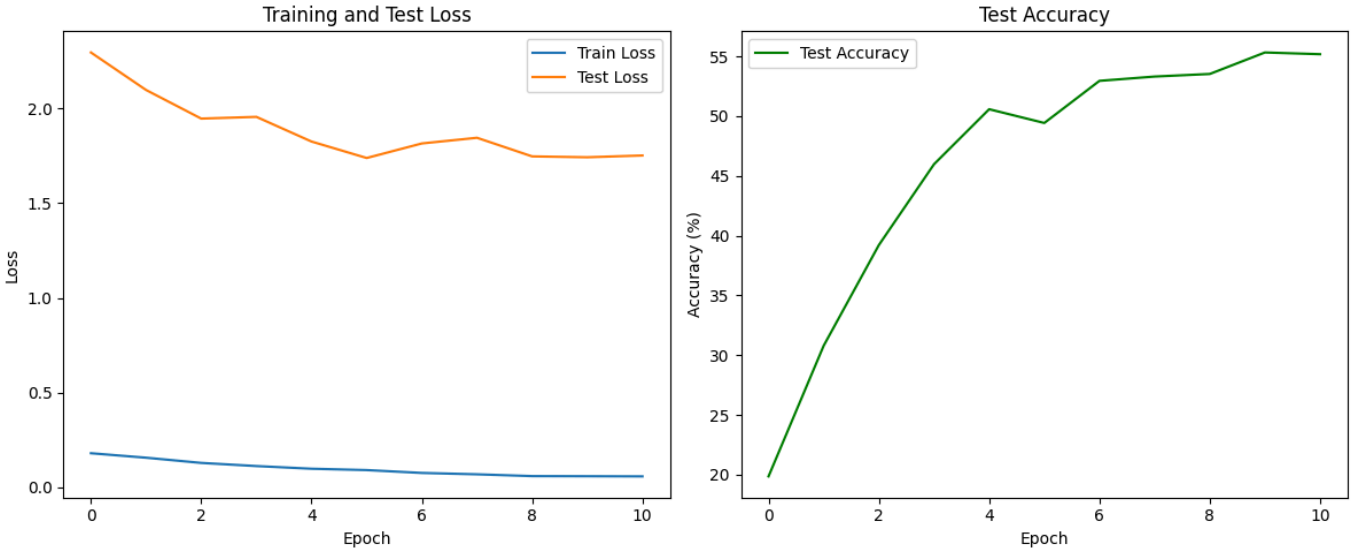
- Kiến trúc mô hình: Mạng CNN gồm 2 tầng tích chập Conv2d với số lượng kernel lần lượt là 16 và 32, kernel size 3, padding 1. Mỗi tầng tích chập đi kèm hàm kích hoạt ReLU và MaxPooling (2x2). Hai lớp fully connected: fc1 (đầu vào 32x56x56, đầu ra 256), fc2 (đầu ra 10 lớp). Có dropout (0.3) để chống overfitting.
- Dữ liệu và xử lý: Ảnh huấn luyện được resize thành 224x224. Các augmentation áp dụng: Grayscale, ColorJitter, RandomHorizontalFlip, RandomRotation.
- Quá trình huấn luyện: Epochs: 20 (có early stopping nếu không cải thiện sau 5 epochs). Nhóm dùng Optimizer: Adam và hàm tính Loss là CrossEntropyLoss. Learning rate: 0.001, giảm StepLR(gamma=0.1) sau mỗi 5 epoch. Batch size: 64.
- Kết quả: Early stopping tại epoch 7, độ chính xác cao nhất đạt 6.2%, mặc dù mô hình đã sâu hơn mô hình cơ bản. Loss dao động quanh 2.3 trong suốt quá trình huấn luyện, không giảm đáng kể.
- Nhận xét: Mô hình hoạt động **rất kém**, cho thấy không học được đặc trưng nào từ dữ liệu. Các nguyên nhân có thể bao gồm: kiến trúc quá đơn giản, Preprocessing chưa đủ tốt, mặc dù đã có augmentation nhưng mô hình không tận dụng được, Batch size lớn (64) với một mô hình nhỏ có thể gây khó khăn trong việc cập nhật gradient hiệu quả. Do đó, Model 1 chỉ đóng vai trò như một baseline để so sánh với các mô hình tốt hơn phía sau.



Hình 1: Biểu đồ loss và accuracy của Model 1 qua các epoch.

3. Model 2: CNN cải tiến

- Kiến trúc mô hình: Mạng CNN tự xây dựng gồm 3 tầng tích chập Conv2d, số kênh: 16 → 32 → 64. Mỗi tầng đi kèm ReLU và MaxPooling. Fully connected: fc1 (đầu vào 64×21×21, đầu ra 512), fc2 (128 neurons), fc3 (10 đầu ra tương ứng với các chữ số). Dropout 0.5 ở fc1.
- Dữ liệu và xử lý: Ảnh resize 200x200. Augmentation: Resize, RandomHorizontalFlip, RandomRotation. Normalize: mean = std = 0.5 (cho cả 3 kênh RGB). Trọng số lớp class_weights được tính toán và áp dụng trong loss.
- Quá trình huấn luyện: Có 20 epochs, Batch size là 32, Optimizer: Adam. Loss function: CrossEntropyLoss với class_weights. Early stopping nếu không cải thiện loss sau 5 epoch. Lưu mô hình tốt nhất (best_model.pth).
- Kết quả: Early stopping ở epoch 11, độ chính xác đạt khoảng 55%. Loss giảm từ 2.3 xuống còn 1.7, nhưng tốc độ cải thiện chậm. Biểu đồ loss và accuracy có xu hướng dừng lại sớm, cho thấy chưa học đủ.
- Nhận xét: Dù model đơn giản hơn model 1 nhưng xử lý dữ liệu và regularization tốt hơn. Mô hình có khả năng cải thiện nếu tăng số epoch hoặc fine-tune lại kiến trúc. Thích hợp làm baseline cho việc chuyển sang model pretrained.



Hình 2: Biểu đồ loss và accuracy của Model 2 qua các epoch.

4. Model 3: MobileNetV3-Large

- Kiến trúc mô hình: Mô hình sử dụng kiến trúc **MobileNetV3-Large** được huấn luyện sẵn (pretrained) trên tập dữ liệu ImageNet. Toàn bộ phần backbone của mô hình được giữ nguyên, chỉ thay thế lớp phân loại cuối cùng bằng một tầng Linear(1280, 10) để phân loại 10 chữ số viết tay. Trong quá trình huấn luyện, chỉ phần classifier được fine-tuned trong 9 epoch (các tầng trước đó được freeze để tiết kiệm tài nguyên và tăng tốc độ huấn luyện).
- Dữ liệu và xử lý ảnh: Ảnh được resize về kích thước 200x200, chuyển sang ảnh grayscale 3 kênh. Áp dụng các kỹ thuật biến đổi ảnh: RandomHorizontalFlip(p=0.5), RandomRotation(10), Normalize theo mean/std lấy từ tập huấn luyện. Tập huấn luyện và kiểm tra được chia theo tỷ lệ 80/20.
- Quá trình huấn luyện: Số epoch: 9 (dừng sớm với early stopping khi loss không giảm). Batch size: 32, Optimizer: Adam, Learning rate: 0.001, Loss function: CrossEntropyLoss có sử dụng class weights để cân bằng các lớp. Sử dụng GPU để tăng tốc quá trình huấn luyện.
- Kết quả: Loss ban đầu khoảng 1.7 → giảm xuống còn 0.61 tại epoch 4. Độ chính xác trên tập kiểm tra: Epoch 1 (71.4%), Epoch 2 (82.2%), Epoch 5 (82.7%), Epoch 7 đạt **83.5%.** Huấn luyện kết thúc sớm tại epoch 9 do không còn cải thiện loss.
- Nhận xét: Mô hình cho kết quả tốt nhất trong tất cả các mô hình đã thử nghiệm. Việc sử dụng mô hình pretrained giúp rút ngắn thời gian huấn luyện và tăng hiệu quả tổng quát hóa. MobileNetV3-Large là kiến trúc nhẹ, rất phù

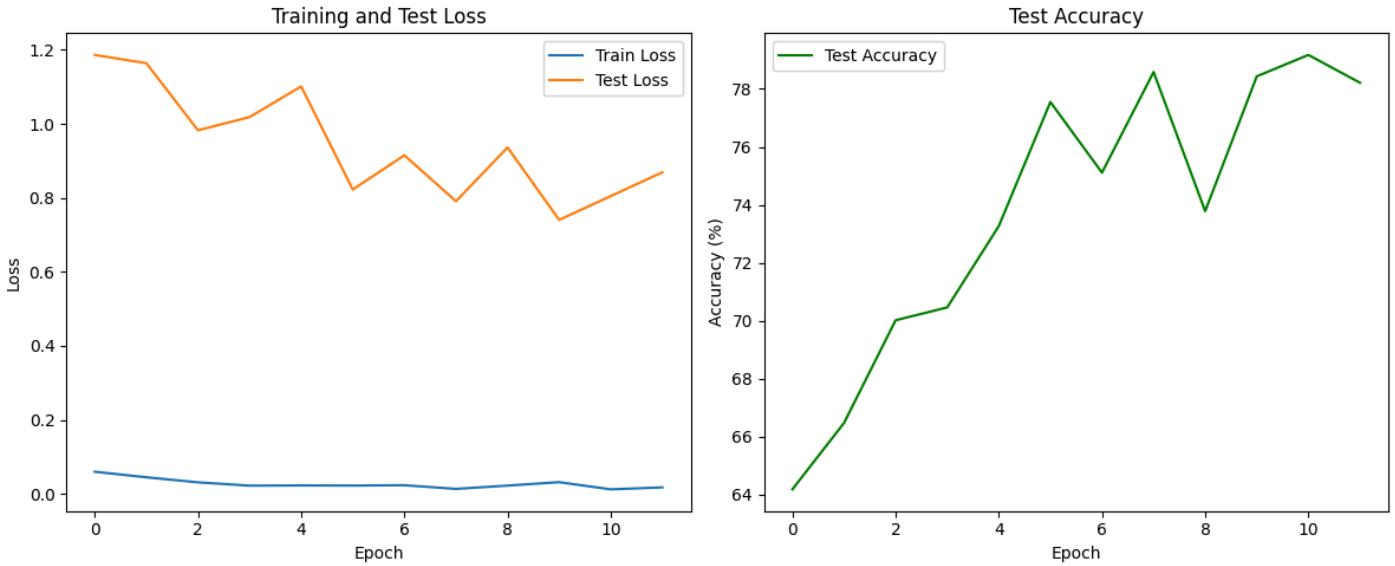
hợp cho các bài toán nhận diện ký tự hoặc triển khai thực tế trên thiết bị có cấu hình vừa phải.

Hình 3: Biểu đồ loss và accuracy của Model 3: MobileNetV3-Large qua các epoch

5. Model 4: ResNet50

- Kiến trúc mô hình: Mô hình sử dụng kiến trúc **ResNet50** pretrained trên ImageNet. Thay thế lớp phân loại cuối cùng bằng Linear(2048, 10) để phân loại 10 lớp đầu ra. Không freeze phần backbone, mô hình được fine-tune toàn bộ, cho phép học sâu hơn nhưng tốn nhiều thời gian và tài nguyên hơn.
- Dữ liệu và xử lý ảnh: Ảnh đầu vào được resize về kích thước 200x200, chuyển thành grayscale 3 kênh. Áp dụng augmentation giống Model 4: HorizontalFlip, Rotation, Normalization. Sử dụng cùng tập dữ liệu, pipeline tiền xử lý và chia tập như Model 3 để đảm bảo công bằng khi so sánh.
- Quá trình huấn luyện: Huấn luyện trong 12 epoch. Batch size: 32. Optimizer: Adam. Learning rate: 0.0003. Loss function: CrossEntropyLoss (không dùng class weights).
- Kết quả: Loss giảm từ 1.18 xuống còn 0.74 tại epoch 10. Độ chính xác tăng dần qua các epoch: Epoch 1 (64.2%), Epoch 6 (77.5%), Epoch 8 (78.6%), Epoch 11 (cao nhất: 79.2%). Không bị overfit rõ rệt, tuy nhiên kết quả vẫn thấp hơn MobileNetV3.

- Nhận xét: ResNet50 là mô hình mạnh và có khả năng học đặc trưng tốt, nhưng do số lượng tham số lớn nên việc huấn luyện chậm hơn. Dù có độ chính xác cao, nhưng ResNet50 lại yêu cầu tài nguyên nhiều hơn so với MobileNetV3, nên không phù hợp lắm với môi trường hạn chế tính toán. Mô hình thể hiện tốt, nhưng không vượt qua được hiệu quả tổng thể của MobileNetV3 trong bài toán này.



Hình 4: Biểu đồ loss và accuracy của Model 3: ResNet50 qua các epoch

CHƯƠNG 4: ĐÁNH GIÁ

1. Phương pháp đánh giá

Việc đánh giá mô hình được thực hiện trên hai nguồn dữ liệu:

- Tập kiểm tra (20%): được tách ra từ tập dữ liệu ban đầu, không trùng với dữ liệu huấn luyện.
- Tập ảnh trên nền tảng Wecode: do thầy cung cấp riêng để kiểm tra khả năng tổng quát hóa của mô hình.

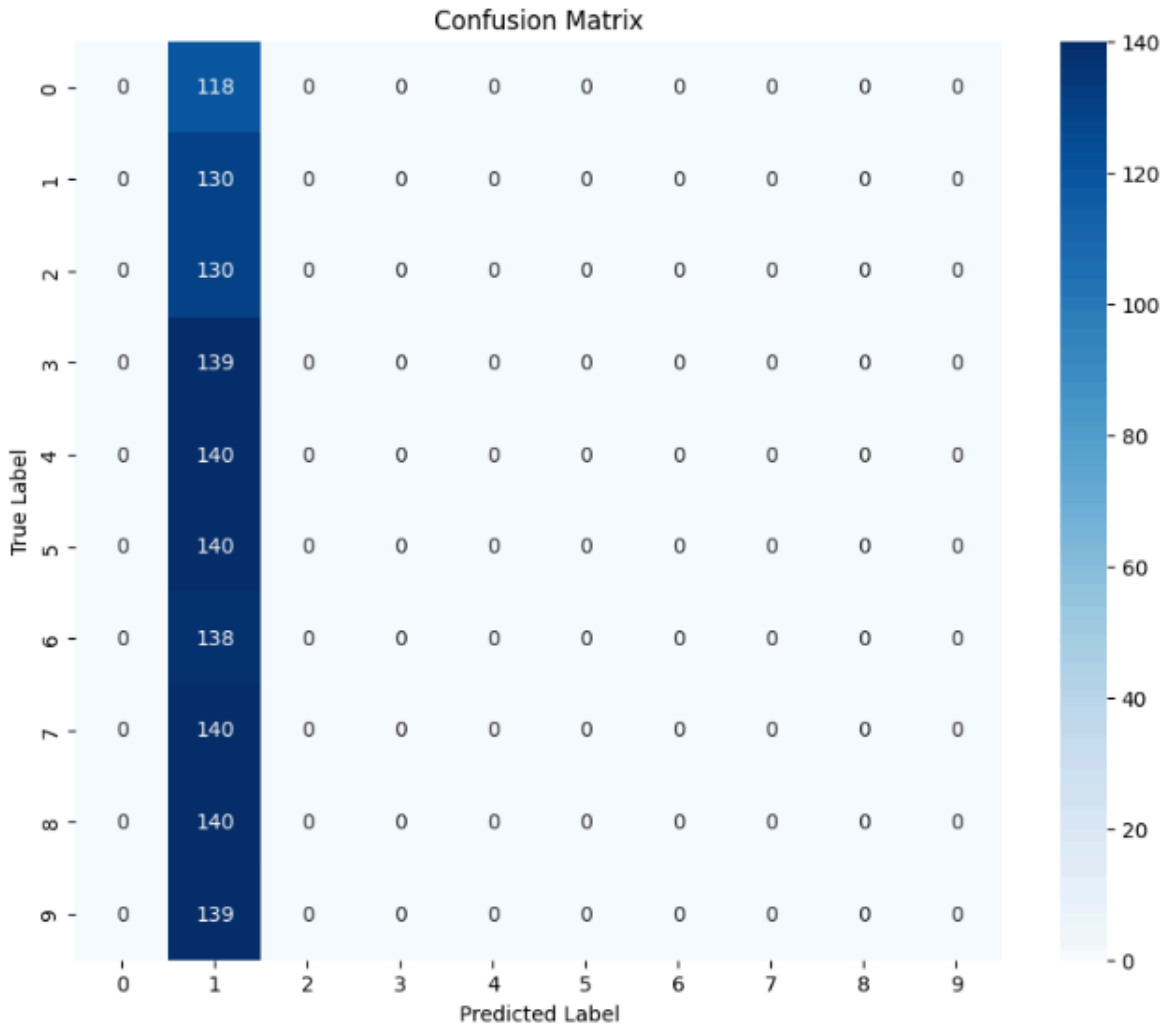
Các chỉ số đánh giá bao gồm:

- Độ chính xác (Accuracy): phần trăm dự đoán đúng trên tổng số mẫu.
- Ma trận nhầm lẫn (Confusion Matrix): thể hiện mức độ nhầm lẫn giữa các lớp.
- Precision, Recall, F1 Score: đo độ chính xác và độ phủ của mô hình theo từng lớp.
- Biểu đồ Loss và Accuracy theo từng epoch: cho thấy tiến trình học và hiệu quả huấn luyện.

2. Kết quả đánh giá và ma trận nhầm lẫn

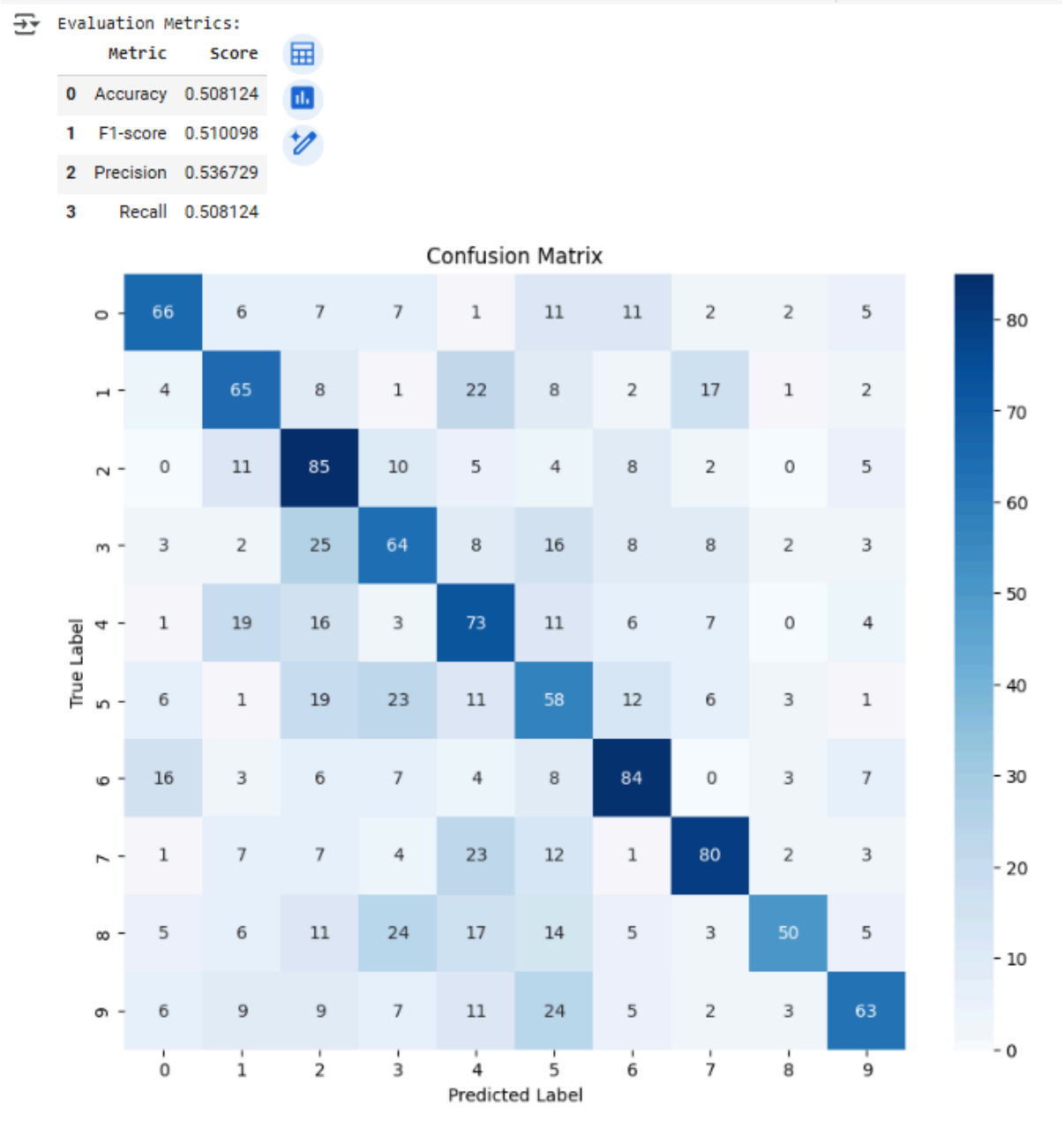
- Model 1 (CNN đơn giản): Độ chính xác rất thấp: chỉ đạt **6.2%**, gần như dự đoán ngẫu nhiên. Không học được đặc trưng nào đáng kể từ dữ liệu. Loss không giảm trong suốt quá trình huấn luyện (khoảng 2.3).

	Metric	Score
0	Accuracy	0.096012
1	F1-score	0.016821
2	Precision	0.009218
3	Recall	0.096012



Hình 5: Accuracy, F1-score, Precision, Recall và Confusion matrix của Model 1

- Model 2 (CNN cải tiến): Cải thiện nhẹ so với Model 1, nhưng độ chính xác vẫn dưới 60%. Loss giảm từ 2.3 xuống 1.8, nhưng chưa đủ tốt.

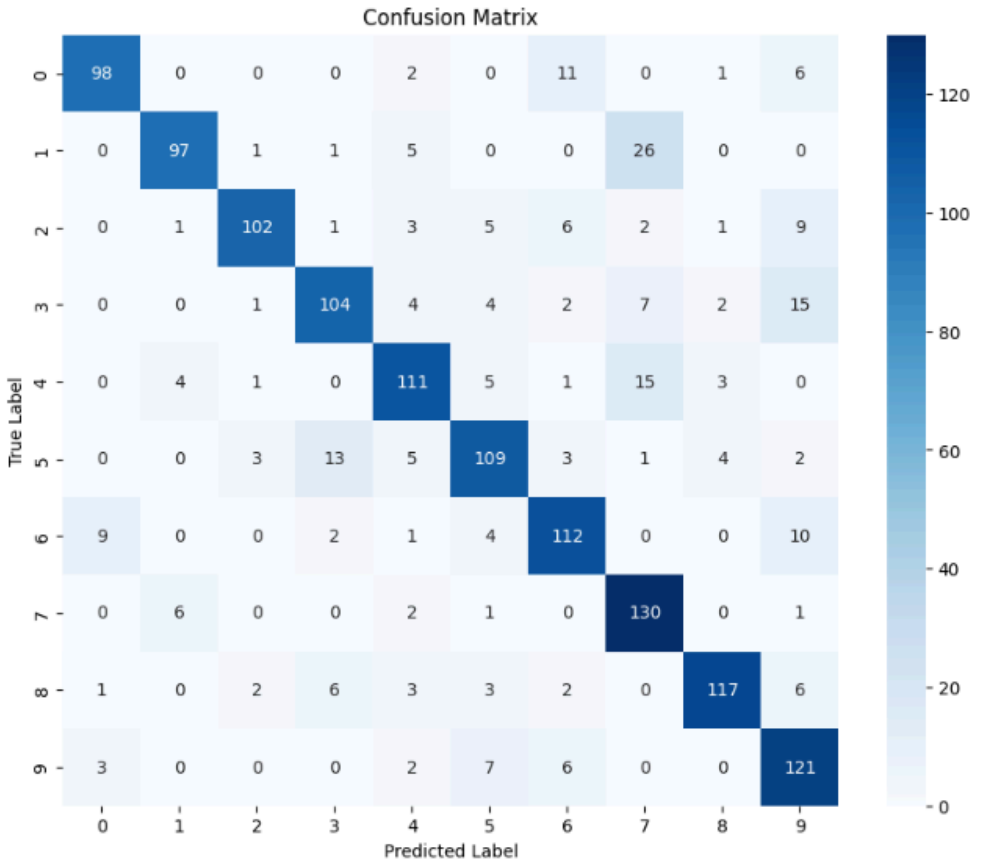


Hình 6: Accuracy, F1-score, Precision, Recall và Confusion matrix của Model 2

- Model 3 (MobileNetV3-Large): Hiệu suất cao nhất: đạt 83.5% trên tập kiểm tra, trung bình đạt 80%. Nhận diện tốt hầu hết chữ số, ngoại trừ một số ảnh chất lượng thấp. Độ ổn định cao, không overfit, dừng sớm ở epoch 9.

Evaluation Metrics:

	Metric	Score
0	Accuracy	0.813146
1	F1-score	0.814061
2	Precision	0.822777
3	Recall	0.813146

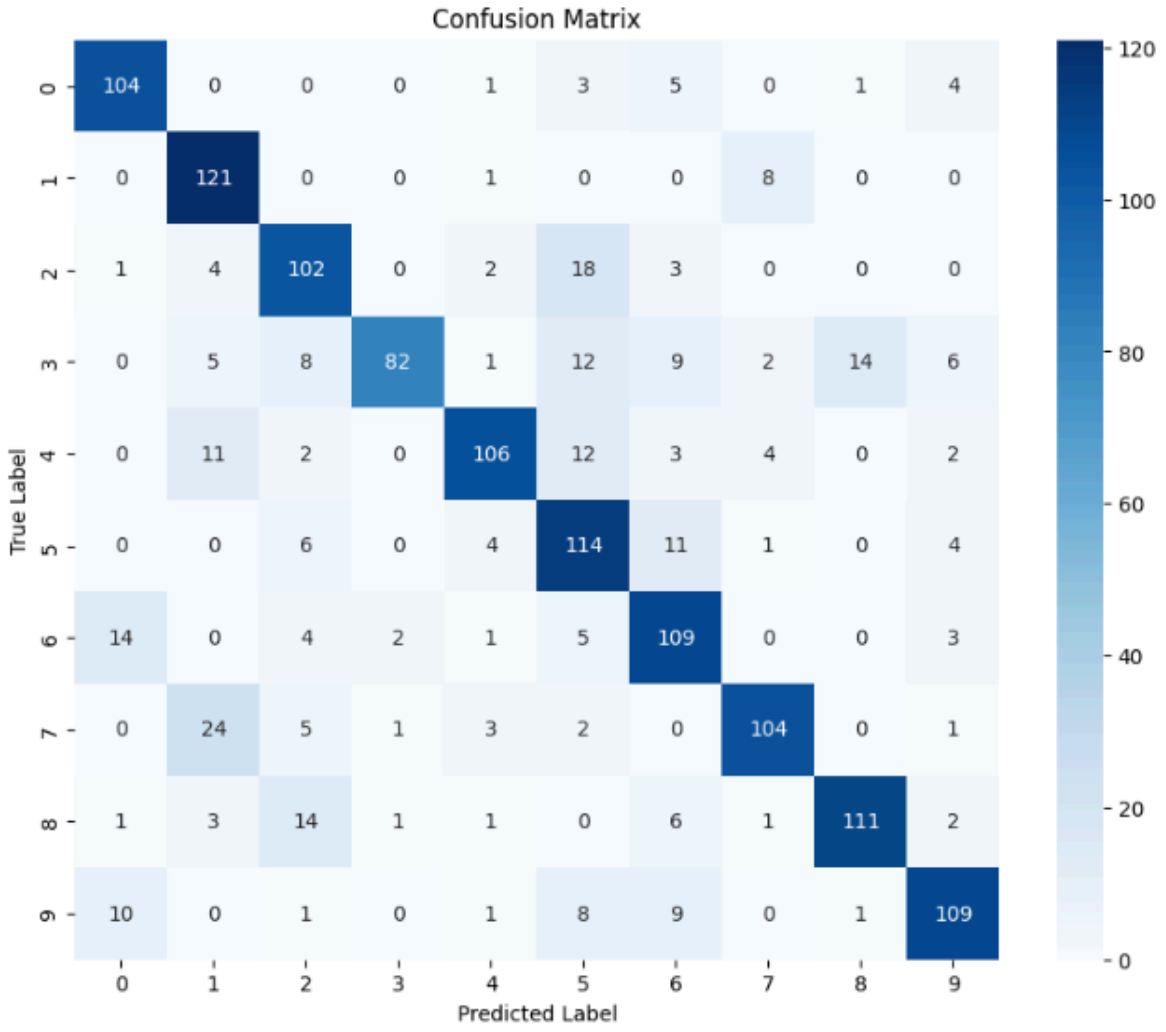


Hình 7: Accuracy, F1-score, Precision, Recall và Confusion matrix của Model 3

- Model 4 (ResNet50): Đạt độ chính xác 79.2%, rất gần với MobileNetV3-Large. Ổn định hơn với ảnh độ phân giải cao nhưng yêu cầu nhiều tài nguyên tính toán.

Evaluation Metrics:

	Metric	Score
0	Accuracy	0.784343
1	F1-score	0.784369
2	Precision	0.801117
3	Recall	0.784343



Hình 8: Accuracy, F1-score, Precision, Recall và Confusion matrix của Model 4

3. So sánh hiệu suất các mô hình

Mô hình	Độ chính xác tập kiểm tra	Độ chính xác Wecode

Model 1	~6.2%	998/9987 ảnh (score: 9)
Model 2	~55%	2138/2939 ảnh (score: 73)
Model 3	~83.5%	8993/9987 ảnh (score: 90)
Model 4	~79.2%	8332/9987 ảnh (score: 83)

CHƯƠNG 5: KẾT LUẬN

1. Tóm tắt kết quả

Nhóm đã triển khai và đánh giá 4 mô hình khác nhau cho bài toán nhận diện chữ số viết tay. Trong đó:

- MobileNetV3-Large đạt hiệu suất cao nhất với độ chính xác lên đến 83.5%.
- Các mô hình sử dụng kỹ thuật transfer learning đều vượt trội rõ rệt so với các mô hình CNN tự xây dựng.
- Kết quả cho thấy việc tận dụng mô hình pretrained trên tập ImageNet mang lại hiệu quả rất lớn, kể cả với bài toán có dữ liệu nhỏ hơn như nhận diện chữ số.

2. Bài học kinh nghiệm

Qua quá trình thực hiện đồ án, nhóm đã rút ra được nhiều bài học quan trọng:

- Tầm quan trọng của **xử lý dữ liệu đầu vào và augmentation** trong việc cải thiện hiệu suất mô hình.
- **Transfer learning** giúp tiết kiệm thời gian huấn luyện và đạt kết quả cao hơn.
- Việc theo dõi loss, accuracy và ma trận nhầm lẫn giúp nhóm điều chỉnh mô hình hiệu quả hơn.
- Nhóm đã rèn luyện được kỹ năng làm việc nhóm, phân công rõ ràng, và quản lý tiến độ tốt.

PHẦN B: ĐỒ ÁN DỰ ĐOÁN ĐIỂM MÔN HỌC TỪ KẾT QUẢ NỘP BÀI TRÊN WECODE

CHƯƠNG 0: CÁC THAY ĐỔI SO VỚI KHI VẤN ĐÁP

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN VÀ DỮ LIỆU

1. Bối cảnh và ý nghĩa

Dự đoán điểm quá trình, thực hành, cuối kỳ và trung bình của sinh viên từ dữ liệu nộp bài trên Wecode là một bài toán ứng dụng trong lĩnh vực học máy giáo dục. Thông qua các thông tin như số lần nộp bài, thời gian, tiến độ cải thiện, ta có thể khai thác hành vi học tập để xây dựng mô hình dự đoán kết quả.

Bài toán này không chỉ giúp giảng viên theo dõi và đánh giá sinh viên một cách khách quan, mà còn hỗ trợ cảnh báo sớm các trường hợp học yếu. Đồng thời, nó góp phần thúc đẩy cá nhân hóa học tập và nâng cao hiệu quả giảng dạy trong môi trường giáo dục số.

2. Nguồn dữ liệu

Nguồn dữ liệu được thầy cung cấp chi tiết bao gồm dữ liệu cho bài toán dự đoán điểm IT001 từ kết quả nộp bài lên wecode. Trong file .zip đã đưa lên google classroom, sẽ có file annonimized.csv gồm các cột: assignment_id, problem_id, mã số sinh viên (đã xóa thông tin định danh), lần nộp này có tính điểm không, code có chạy không, hệ số nộp bài trễ, % testcase đúng làm tròn 2 chữ số và đổi sang số nguyên, ngày giờ nộp (không có năm), ngày giờ chấm, kết quả chấm (thời gian và memory mỗi test case sử dụng, số lượng testcase sai).

Và kèm theo sẽ được cung cấp kết quả đi của khoảng 800 sinh viên trong các tbtl-public,csv, qt-public.csv, v.v... và sẽ đi dự đoán điểm cho các sinh viên còn lại tương tự cho các cột điểm khác.

CHƯƠNG 2: DATA PROCESSING

1. Kiểm tra và lọc dữ liệu

Dữ liệu được tải từ folder annonimized.csv và file điểm .csv do thầy cung cấp kiểm tra file dữ liệu kiểm tra xem kích thước của file thông tin và thông tin các cột đã đúng và chính xác hay chưa

- Sử dụng raise ValueError để thông báo lỗi nếu 2 file dữ liệu cần kiểm tra bị rỗng không hợp lệ
- Dùng .shape để kiểm tra kích thước chính xác của dữ liệu

2. Chia tập dữ liệu

Dữ liệu được chia thành:

- Tập huấn luyện: chiếm 80% tổng số dòng dữ liệu hợp lệ.
- Tập kiểm tra: 20% còn lại.

Việc chia được thực hiện ngẫu nhiên bằng hàm train_test_split với random_state = 42 nhằm đảm bảo tính tái lập trong quá trình huấn luyện và đánh giá mô hình

3. Biến đổi dữ liệu

Các phép biến đổi dữ liệu được áp dụng để các dữ liệu đầu vào của mô hình được đúng cũng như việc train dữ liệu sẽ đạt hiệu quả tốt hơn.

- Xử lý và làm sạch dữ liệu
    - Đổi tên các cột thông tin sao cho phù hợp.
    - Chuẩn hóa lại cột điểm và chuyển sang kiểu dữ liệu phù hợp.
    - Xử lý cột thời gian và chuẩn hóa sao cho hợp lệ.
    - Viết hàm trích xuất số testcase sai từ phần cột đánh giá để tạo features.
- Tạo các features để huấn luyện mô hình

Các features được chia thành các nhóm features để dễ quản lý cũng như tăng logic cho bài làm.

- Tạo nhóm các features cơ bản để phản ánh cường độ và chất lượng làm bài tổng thể.
- Tạo nhóm các features về thời gian cho thấy sinh viên có phân bổ thời gian hợp lý và đều đặn hay không.
- Tạo nhóm các features về chất lượng làm bài tập phản ánh năng lực và mức độ hoàn thành các bài tập.
- Tạo nhóm các features về quá trình học tập để đo mức độ cố gắng và khả năng học hỏi trong quá trình làm bài.
- Tạo nhóm các features về chỉ số tổng hợp làm nổi bật năng suất và sự chuyên cần của người học.

CHƯƠNG 3: HUẤN LUYỆN

1. Tổng quan quá trình huấn luyện

Nhóm đã thử nghiệm 2 mô hình phổ biến và đạt thông số cao cho bài tập huấn luyện mô hình tuyến tính là LightGBM và mô hình XGBoost với các tham số khác nhau để huấn luyện cho 3 bài tập dự đoán điểm gồm điểm quá trình, điểm thực hành và điểm cuối kỳ.

2. Model 1: LightGBM

- Kiến trúc mô hình: LightGBM là một mô hình gradient boosting tree-based, tối ưu hóa theo bài toán hồi quy với hàm mất mát RMSE. Mô hình được thiết lập với 100 leaf nodes mỗi cây, học với learning rate = 0.03, sử dụng chiến lược bagging (80% dữ liệu, tần suất 3 vòng) và random feature selection (90%). Mô hình sử dụng 500 vòng lặp tối đa và dừng sớm nếu không cải thiện sau 100 vòng.
- Dữ liệu và xử lý: Dữ liệu được xử lý từ file log nộp bài của sinh viên, gồm các đặc trưng liên quan đến thời gian, chất lượng nộp bài, quá trình học tập và chỉ số tổng hợp. Tập dữ liệu được chia thành 80% để huấn luyện và 20% để kiểm tra, đảm bảo tính ngẫu nhiên và tái lập (random_state = 42).
- Quá trình huấn luyện: Mô hình được huấn luyện trong vòng tối đa 500 lượt, sử dụng early stopping sau 100 vòng không cải thiện.Sau khi huấn luyện xong thì lưu file cho model file.txt.
- Kết quả:

	Bài toán dự đoán điểm thực hành:	Bài toán dự đoán điểm quá trình:	Bài toán dự đoán điểm cuối kỳ:
R^2 Score	0.4537	0.2518	0.3413
RMSE	1.6652	1.3309	1.6339

3. Model 2: XGBoost

- Kiến trúc mô hình: Mô hình sử dụng XGBoost – một thuật toán boosting mạnh mẽ cho bài toán hồi quy. Cây được xây dựng với độ sâu tối đa là 6, learning rate = 0.03. Mô hình sử dụng 90% đặc trưng mỗi cây (colsample_bytree = 0.9) và 80% dữ liệu huấn luyện mỗi vòng (subsample = 0.8). Mô hình được huấn luyện tối đa 500 vòng và dừng sớm nếu không cải thiện sau 100 vòng.

- Dữ liệu và xử lý: Dữ liệu là đặc trưng từ quá trình làm bài của sinh viên (đã qua xử lý), chia thành 80% huấn luyện và 20% kiểm tra với random_state = 42. Dữ liệu đầu vào được chuyển thành DMatrix – định dạng tối ưu cho XGBoost.
- Quá trình huấn luyện: Mô hình được huấn luyện trong tối đa 500 vòng, sử dụng early stopping sau 100 vòng không cải thiện. Tập kiểm tra (validation) được dùng để theo dõi và đánh giá tiến trình huấn luyện.Sau khi huấn luyện xong thì lưu file cho model file.json.
- Kết quả:

	Bài toán dự đoán điểm thực hành:	Bài toán dự đoán điểm quá trình:	Bài toán dự đoán điểm cuối kỳ:
R^2 Score	0.4578	0.1866	0.2870
RMSE	1.6590	1.3878	1.6999

4. Nhận xét từ kết quả train model từ LightGBM và XGBoost

- Bài toán dự đoán điểm thực hành
  - Đây là bài toán có kết quả tốt nhất trong cả ba mô hình, với R^2 gần 0.46 ở cả LightGBM và XGBoost, cho thấy mô hình đã học được mối liên hệ khá rõ giữa các đặc trưng và điểm thực hành.
  - RMSE ~1.66 là chấp nhận được với thang điểm 10.
  - XGBoost nhỉnh hơn một chút so với LightGBM, tuy nhiên sự chênh lệch là không đáng kể.
  - Có thể hiểu rằng các đặc trưng liên quan đến tần suất làm bài, tỷ lệ điểm cao, số lượng bài nộp… phản ánh khá tốt hiệu suất thực hành.
- Bài toán dự đoán điểm quá trình
  - Đây là bài toán khó hơn, với R^2 thấp (~0.25 cho LightGBM và ~0.18 cho XGBoost), nghĩa là mô hình chỉ giải thích được khoảng 18–25% phương sai của dữ liệu.
  - Dù RMSE thấp hơn so với bài toán thực hành (vì điểm quá trình ít dao động hơn), R^2 thấp cho thấy mô hình chưa khai thác được rõ ràng các đặc trưng phù hợp với điểm quá trình.
  - LightGBM hoạt động rõ ràng tốt hơn XGBoost ở bài toán này.
  - Nguyên nhân có thể do điểm quá trình bị ảnh hưởng bởi các yếu tố ngoài dữ liệu Wecode (như điểm chuyên cần, tham gia lớp, tương tác nhóm…), nên khó dự đoán chính xác.

- Bài toán dự đoán điểm cuối kỳ
    - Kết quả ở mức trung bình, tốt hơn dự đoán điểm quá trình nhưng kém hơn điểm thực hành.
    - R^2 ~0.28–0.34 cho thấy mô hình đã khai thác được một phần đặc trưng phản ánh năng lực tổng thể của sinh viên, nhưng vẫn còn nhiều yếu tố khác chưa nắm bắt được (ví dụ: kỹ năng làm bài thi cuối kỳ, tâm lý thi cử...).
    - LightGBM tiếp tục vượt trội hơn XGBoost ở bài toán này.
    - Có thể cải thiện thêm bằng cách tích hợp các đặc trưng về tiến độ học theo thời gian (time series features) hoặc lịch sử cải thiện điểm cụ thể cho từng dạng bài tập.

CHƯƠNG 4: ĐÁNH GIÁ

1. Phương pháp đánh giá

Việc đánh giá mô hình được thực hiện trên hai nguồn dữ liệu:

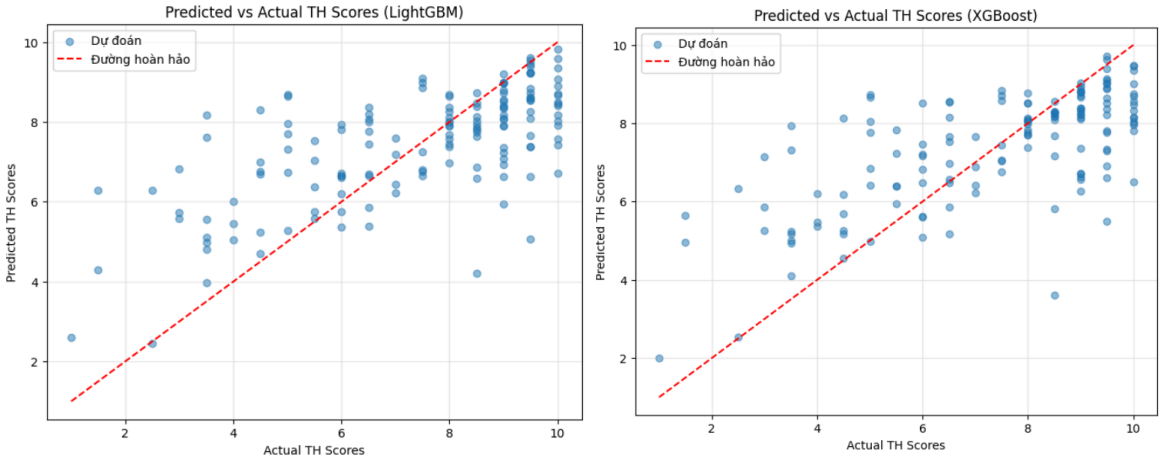
- Tập kiểm tra (20%): Được tách từ dữ liệu ban đầu bằng train_test_split, đảm bảo không trùng lặp với dữ liệu huấn luyện. Dùng để kiểm tra khả năng mô hình dự đoán trên dữ liệu chưa thấy.
- Tập dữ liệu điểm thực tế: Là tập dữ liệu điểm thật của sinh viên (thực hành, quá trình, cuối kỳ), dùng để kiểm chứng khả năng mô hình tổng quát hóa trong môi trường thực tế học tập.

Các chỉ số đánh giá bao gồm:

- R^2 Score (Hệ số xác định): Đo lường mức độ giải thích của mô hình đối với phương sai của biến mục tiêu. Giá trị càng gần 1, mô hình càng tốt.
- RMSE (Root Mean Squared Error): Đo sai số trung bình giữa giá trị dự đoán và giá trị thực tế. Giá trị càng nhỏ, mô hình càng chính xác
- Biểu đồ phân phối sai số: Được sử dụng để kiểm tra xem mô hình có sai lệch hệ thống không (ví dụ: thiên lệch hoặc underfit).

2. Biểu đồ thể hiện kết quả dự đoán và kết quả thực

Hình 9: Biểu đồ thể hiện kết quả dự đoán điểm Thực hành và kết quả điểm Thực hành thực tế



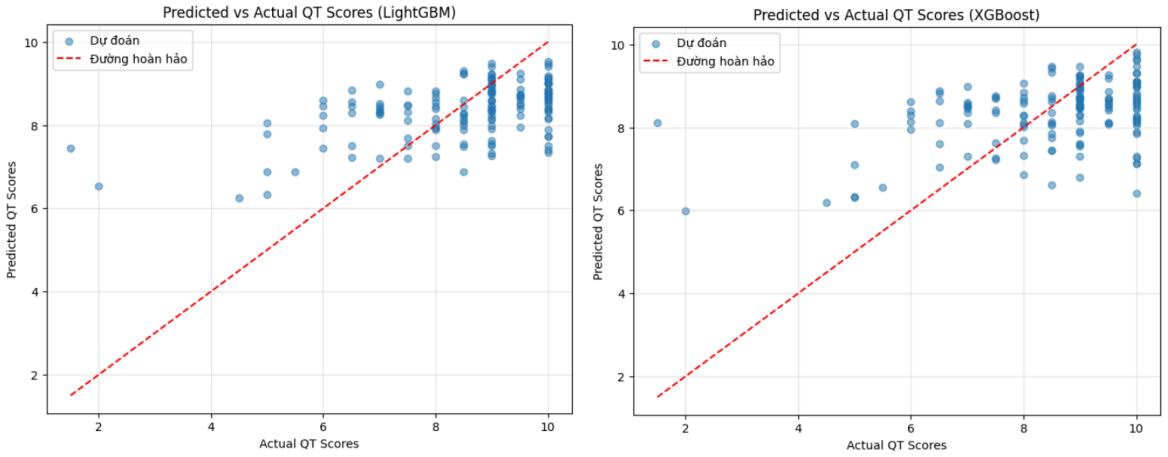
Nhận xét mô hình LightGBM:

- Các điểm phân bố tương đối gần với đường chéo lý tưởng, đặc biệt trong khoảng điểm từ 6 đến 10.
- Cho thấy mô hình hoạt động khá tốt với các sinh viên có điểm thực hành trung bình – cao, dự đoán tương đối chính xác.
- Tuy nhiên, vẫn còn hiện tượng dự đoán sai lệch với sinh viên có điểm thấp, khi mô hình dự đoán cao hơn thực tế (các điểm nằm phía trên đường chéo ở vùng điểm <5).
- Mô hình có độ chính xác khá tốt trong bài toán này ($R^2 \sim 0.45$).

Nhận xét mô hình XGBoost:

- Phân bố điểm dự đoán gần đường lý tưởng, đặc biệt đồng đều hơn ở toàn bộ dải điểm so với LightGBM.
- Các điểm dự đoán ít bị outlier mạnh, thể hiện mức ổn định cao hơn, đặc biệt ở vùng điểm thấp.
- Một số điểm vẫn bị dự đoán sai (lệch trên/dưới), nhưng số lượng ít và không ảnh hưởng đáng kể đến tổng thể.
  Đây là bài toán mà XGBoost hoạt động tốt nhất trong ba bài toán, với R^2 cao nhất.

Hình 10: Biểu đồ thể hiện kết quả dự đoán điểm Quá trình và kết quả điểm Quá trình thực tế



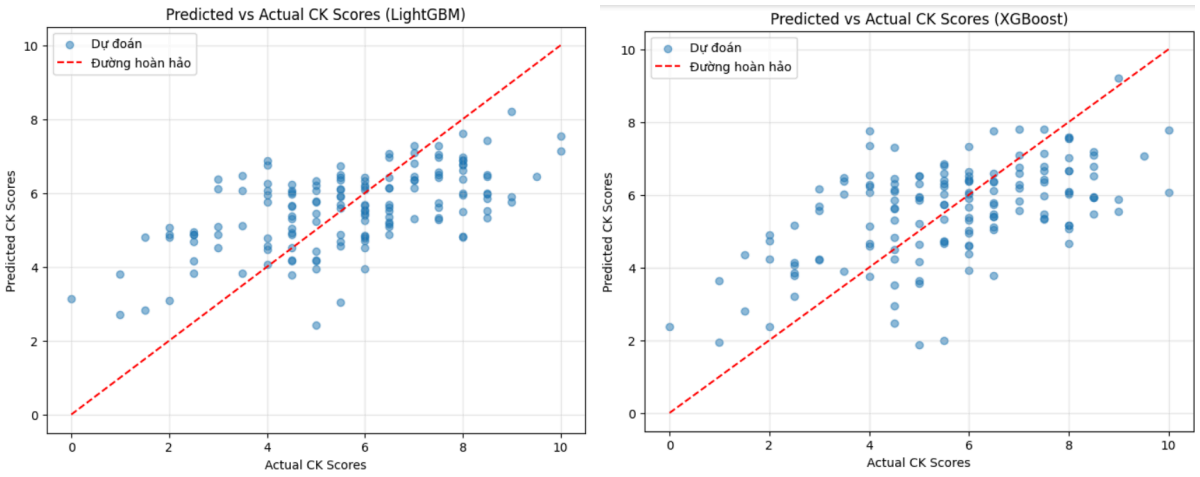
Nhận xét mô hình LightGBM**:**

- Các điểm dự đoán phần lớn tập trung phía trên đường chéo, cho thấy mô hình **có** xu hướng dự đoán cao hơn thực tế đối với các điểm thấp.
- Với các điểm cao (8–10), dự đoán khá gần đường lý tưởng, cho thấy mô hình học tốt hơn với nhóm điểm cao.
- Tuy nhiên, vẫn còn khá nhiều nhiễu, nhất là trong khoảng điểm từ 5–8.

Nhận xét mô hình XGBoost:

- Tương tự LightGBM, nhưng nhiễu lớn hơn. Các điểm dự đoán phân tán nhiều hơn, đặc biệt ở vùng điểm cao.
- Mô hình chưa bắt được xu hướng tuyến tính rõ ràng, thể hiện qua nhiều điểm cách xa đường chéo.
- Cho thấy hiệu suất dự đoán của XGBoost thấp hơn LightGBM ở bài toán này.

Hình 11: Biểu đồ thể hiện kết quả dự đoán điểm Cuối kỳ và kết quả điểm Cuối kỳ thực tế



Nhận xét mô hình LightGBM:

- Các điểm dự đoán khá phân tán, nhất là ở vùng điểm thấp (0–4), cho thấy mô hình khó khăn trong việc dự đoán chính xác học sinh yếu.
- Ở vùng điểm trung bình–khá (5–8), mô hình có xu hướng "nén" điểm dự đoán về khoảng giữa, tức là dự đoán thiếu với điểm cao và dư với điểm thấp.
- Điều này phản ánh hiện tượng underfitting nhẹ, tức mô hình chưa học được mối liên hệ phức tạp giữa đặc trưng và điểm số.

Nhận xét mô hình XGBoost:

- Phân bố điểm dự đoán có phần đều hơn LightGBM, tuy nhiên nhiều điểm lệch xa đường chéo.
- Mô hình có vẻ tuyến tính hơn nhưng kém chính xác, với R^2 thấp hơn LightGBM.
- Một số điểm quá cao hoặc quá thấp bị dự đoán sai nghiêm trọng (trên 2 điểm chênh lệch).

3. So sánh hiệu suất 2 mô hình

- Ta sẽ so sánh độ chính xác của 2 mô hình và lấy kết quả cao hơn để nộp cho wecode
- Điểm TBTL sẽ không có tập dữ liệu để train model ta sẽ dùng 3 cột điểm đã dự đoán sẽ tính toán theo công thức hệ số điểm để đưa ra cột điểm TBTL

  TBTL = 20%*Điểm quá trình + 40%Điểm thực hành + 40%Điểm cuối kỳ

	Độ chính xác (R^2 score) trên tập kiểm tra		Độ chính xác(R^2 score) trên wecode
	LightGBM	XGBoost	
Điểm quá trình	0.2518	0.1866	36/100
Điểm cuối kỳ	0.3413	0.2870	15/100
Điểm thực hành	0.4537	0.4578	34/100
Điểm TBTL			-175

CHƯƠNG 5: KẾT LUẬN

1. Tóm tắt kết quả

Trong đồ án này, nhóm đã triển khai và đánh giá hai mô hình hồi quy hiện đại là LightGBM và XGBoost nhằm giải quyết bài toán dự đoán điểm số sinh viên

(bao gồm: điểm Thực hành, điểm Quá trình và điểm Cuối kỳ) và từ đó dự đoán điểm TBTL dựa trên dữ liệu nộp bài từ nền tảng Wecode.

Từ kết quả thực nghiệm cho thấy

- LightGBM nhìn chung đạt hiệu suất cao và ổn định hơn XGBoost trong hầu hết các bài toán.
- Bài toán dự đoán điểm Thực hành là khả thi nhất, với R^2 lên đến ~0.45 và RMSE khoảng 1.66.
- Bài toán điểm Quá trình và điểm Cuối kỳ có độ chính xác thấp hơn, cho thấy còn nhiều yếu tố ngoài dữ liệu Wecode ảnh hưởng đến điểm thực tế.
- Dù không đạt độ chính xác quá cao, mô hình vẫn thể hiện được khả năng học được xu hướng chung trong kết quả học tập của sinh viên.

2. Bài học kinh nghiệm

Qua quá trình thực hiện đồ án, nhóm đã rút ra được nhiều bài học quan trọng:

- Tiền xử lý dữ liệu và chọn lọc đặc trưng (feature engineering) đóng vai trò then chốt. Việc nhóm các đặc trưng thành 5 nhóm (cơ bản, thời gian, chất lượng, quá trình học, chỉ số tổng hợp) giúp mô hình học tốt hơn.
- Tách tập huấn luyện – kiểm tra đúng cách, đảm bảo đánh giá khách quan hiệu quả mô hình.
- Các chỉ số đánh giá hồi quy như R^2 và RMSE giúp nhóm hiểu rõ hơn hiệu suất và độ sai lệch của mô hình.
- XGBoost và LightGBM tuy tương đồng, nhưng hiệu quả khác nhau tùy theo bài toán cụ thể.

3. Định hướng phát triển

- Mở rộng dữ liệu: Kết hợp thêm dữ liệu chuyên cần, điểm danh, đánh giá từ giáo viên… để tăng độ chính xác mô hình.
- Thử nghiệm thêm các mô hình phức tạp hơn như Random Forest, hoặc MLP hồi quy để so sánh.
- Tích hợp mô hình vào hệ thống cảnh báo học tập sớm, giúp sinh viên cải thiện điểm số kịp thời.

NGUỒN THAM KHẢO

1. Giáo trình môn học CS114 : Máy học
2. Huấn luyện CNN với PyTorch từ cơ bản (CIFAR10 tutorial)

   https://docs.pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
3. Hướng dẫn Transfer Learning với PyTorch

   https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
4. Thư viện các mô hình có sẵn trong torchvision (MobileNet, ResNet,...)

   https://docs.pytorch.org/vision/master/models.html
5. Thư viện các câu lệnh, cách dùng và ý nghĩa các câu lệnh thư viện pandas https://pandas.pydata.org/pandas-docs/stable/reference/index.html
6. Mô hình LightGBM(cấu hình tham số,huấn luyện, dự đoán đánh giá,...) https://lightgbm.readthedocs.io/en/stable/
7. Mô hình XGBoost(cấu hình tham số,huấn luyện, dự đoán đánh giá,...) https://xgboost.readthedocs.io/en/stable/
8. Metrics and scoring: quantifying the quality of predictions https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics