**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**KHOA KHOA HỌC MÁY TÍNH**

**---****---**

Đồ án môn: Máy học

# Đề tài: Bài toán nhận diện chữ số viết tay và Dự đoán các cột điểm môn IT001

**Giảng viên hướng dẫn:**

**+ Phạm Nguyễn Trường An**

**Sinh viên thực hiện:**

**+ Hứa Mạnh Tân (23521396)**

**+ Nguyễn Tấn Tài (23521376)**

**+ Nguyễn Chí Cường (2352019)**

**HỒ CHÍ MINH – 2025**

# MỤC LỤC

[Đề tài: Bài toán nhận diện chữ số viết tay và Dự đoán các cột điểm môn IT001 1](#_Toc202205635)

[MỤC LỤC 2](#_Toc202205636)

[I Giới thiệu chung 3](#_Toc202205637)

[**1.1** **Công cụ thực hành** 3](#_Toc202205638)

[**1.2** **Công cụ lưu trữ và trao đổi bài làm** 3](#_Toc202205639)

[II Bài toán 3](#_Toc202205641)

[**Phần A : đồ án nhận diện chữ số viết tay (hand written digit recognition)** 3](#_Toc202205642)

[Chương 1: Các thay đổi so với vấn đáp 3](#_Toc202205643)

[Chương 2: Data loader và data processing 3](#_Toc202205644)

[Chương 3: Mô tả các model đã thử nghiệm và cách xác định các tham số để training từng model 4](#_Toc202205647)

[Chương 4: Mô tả quá trình đánh giá model qua cảm quan của nhóm cũng như kết quả trên wecode. 9](#_Toc202205648)

[Chương 5 : Cảm nghĩ của nhóm sau khi làm bài 11](#_Toc202205649)

[**Phần B: Đồ án dự đoán điểm môn học từ kết quả nộp bài trên wecode.** 13](#_Toc202205650)

[Chương 0: Liệt kê các thay đổi so với khi vấn đáp 13](#_Toc202205651)

[Chương 1: Giới thiệu bài toán và dữ liệu 13](#_Toc202205652)

[Chương 2: Data processing 14](#_Toc202205653)

[Chương 3: Training 15](#_Toc202205654)

[Chương 4: Evaluation 17](#_Toc202205655)

[Chương 5: Kết luận. 18](#_Toc202205656)

# I Giới thiệu chung

* 1. **Công cụ thực hành**

− Google colab

− Jupyter notebook

* 1. **Công cụ lưu trữ và trao đổi bài làm**

# Github :<https://github.com/huapogba/may-hoc>

# II Bài toán

## **Phần A : đồ án nhận diện chữ số viết tay (hand written digit recognition)**

### Chương 1: Các thay đổi so với vấn đáp

− Bổ sung phần chạy bài toán với ảnh chỉ resize theo kích thước đầu vào của 3 mô hình(mobinet\_v2, cnn, resnet18) rồi từ đó dự đoán trên tập dataset đã cấp

− Bổ sung phần chạy bài toán với ảnh grayscale và resize theo kích thước đầu vào của 3 mô hình (mobinet\_v2, cnn, resnet18) rồi từ đó dự đoán trên tập dataset đã cấp

− Bổ sung phần data preparation ( download ảnh từ github của các sinh viên)

### Chương 2: Data loader và data processing

## Mô tả cách tải và chuẩn bị dữ liệu trước khi đưa vào huấn luyện model

− Đầu tiên nhóm sẽ copy toàn bộ nội dung trên diễn đàn thảo luận sau đó chạy chương trình để lọc ra những github repository

− Sau đó ghi vào file hand\_written\_digit\_4 ở google drive là những file ở github repo với tên trùng với tên github\_repo

− Tiếp theo tìm kiếm các ảnh trong các thư mục theo cú pháp “f'{data\_dir}/\*/hand\_written\_digit/??52????” và tạo thành 1 image\_lists

− Cho toàn bộ danh sách image\_lists vào foler all\_images (lưu trong folder hand\_written\_digit\_4)

− Bước cuối cung là duyệt theo kí tự vị trí thứ 7 ( bắt đầu là vị trí thứ 0) để đưa vào các foler theo kí tự vị trí thứ 7 ( từ 0 đến 9) va lưu trong tập sort\_images

− Github : https://github.com/huapogba/may- hoc/blob/main/hand\_written\_digit/data\_preparation.ipynb

### Data processing

* + 1. Nhị phân + Căn giữa chữ số + Resize về pixel x pixel theo đầu vào mô hình (Hiệu quả cao nhất)

− Mô tả:

* + - * Ảnh được đọc ở chế độ grayscale.
      * Resize ban đầu về 256x256 để chuẩn hóa kích thước.
      * Áp dụng ngưỡng hóa nhị phân (Otsu) kết hợp đảo màu để làm nổi bật chữ số.
      * Tìm contour lớn nhất đại diện cho chữ số chính.
      * Cắt ảnh bao quanh contour, rồi đặt vào giữa một ảnh vuông (padding đều 2 chiều).
      * Cuối cùng resize về kích thước đầu vào theo từng mô hình ,

chuẩn hóa pixel [0,1].

− Ưu điểm:

* + - * Giúp mô hình tập trung vào vùng chứa chữ số thay vì cả nền.
      * Loại bỏ nhiễu, căn giữa và làm nổi bật đặc trưng chữ số.
      * Dễ nhận dạng chữ số hơn

− Github : [https://github.com/huapogba/may-](https://github.com/huapogba/may-hoc/tree/main/hand_written_digit/xu_li_anh_bao_cao) [hoc/tree/main/hand\_written\_digit/xu\_li\_anh\_bao\_cao](https://github.com/huapogba/may-hoc/tree/main/hand_written_digit/xu_li_anh_bao_cao)

* + 1. Resize ảnh màu về theo kích thước đầu vào mô hình

− Mô tả:

* + - * Ảnh được đọc giữ nguyên màu (BGR).
      * Resize 2 bước: từ ảnh gốc → 256x256 → theo kích thước đầu vào của từng mô hình
      * Không áp dụng nhị phân hay cắt vùng chữ số.
      * Chuẩn hóa pixel về [0,1].

− Ưu điểm:

* + - * Giữ được nhiều thông tin hơn (màu sắc).
      * Dễ triển khai, không cần xử lý contour.

− Nhược điểm:

* + - * Dễ bị ảnh hưởng bởi nhiễu nền và kích thước chữ số không đồng đều.
      * Không căn giữa, nên chữ số có thể bị lệch.

− Github : [https://github.com/huapogba/may-](https://github.com/huapogba/may-hoc/tree/main/hand_written_digit/xu_li_anh_resize) [hoc/tree/main/hand\_written\_digit/xu\_li\_anh\_resize](https://github.com/huapogba/may-hoc/tree/main/hand_written_digit/xu_li_anh_resize)

* + 1. Resize trực tiếp về kích thước đầu vào từng mô hình và chuyển ảnh xám

− Mô tả:

* + - * Ảnh được đọc ở chế độ grayscale.
      * Resize trực tiếp từ ảnh gốc về kích thước đầu vào từng mô hình
      * Không làm nhị phân, không cắt contour, không căn giữa.
      * Chuẩn hóa pixel về [0,1].

− Ưu điểm:

* + - * Nhanh, đơn giản, ít tính toán.

− Nhược điểm:

* + - * Không xử lý nhiễu, không căn giữa → chữ số có thể nhỏ, lệch, mờ.
      * Mất nhiều thông tin chi tiết nếu ảnh gốc không cân đối.

− Github : [https://github.com/huapogba/may-](https://github.com/huapogba/may-hoc/tree/main/hand_written_digit/xu_li_anh_grayscale) [hoc/tree/main/hand\_written\_digit/xu\_li\_anh\_grayscale](https://github.com/huapogba/may-hoc/tree/main/hand_written_digit/xu_li_anh_grayscale)

### Chương 3: Mô tả các model đã thử nghiệm và cách xác định các tham số để training từng model

* 1. Mô hình CNN
     1. Mô tả mô hình

|  |  |  |
| --- | --- | --- |
| **Tầng** | **Loại tầng** | **Thông tin chi tiết** |
| 1 | Conv2d(1, 32, 3) | Tích chập 2D, đầu vào 1 kênh (ảnh xám), đầu ra 32 kênh, kernel 3x3 |
| 2 | MaxPool2d(2, 2) | Tích chập 2D, đầu vào 1 kênh (ảnh xám), đầu ra 32 kênh, kernel 3x3 |
| 3 | Conv2d(32, 64, 3) | Tích chập 2D tiếp theo, đầu vào 32 kênh, đầu ra 64 kênh, kernel 3x3 |
| 4 | MaxPool2d(2, 2) | Pooling tiếp theo, giảm tiếp kích thước |
| 5 | Flatten | Chuyển tensor 3D thành vector 1D (từ [batch, 64, 5, 5] → [batch, 1600]) |
| 6 | Linear(1600, 64) | Fully Connected 1: từ 1600 node về 64 node ẩn |
| 7 | Linear(64, 10) | Fully Connected 2: từ 64 node → 10 đầu ra (tương ứng 10 chữ số 0–9) |

* + 1. Mô tả mô hình trước training

− Trước khi tiến hành huấn luyện mô hình, nhóm đã thực hiện các bước chuẩn bị như sau:

* Tiền xử lý dữ liệu đầu vào theo 3 cách đã mô tả ở chương 2 ( mục 2.2) (mỗi lần chạy tương ứng với 1 cách )
* Sau đó tiến hành chia tập dữ liệu với tập test chiếm 20% dữ liệu, tập train chiếm 80% dữ liệu
* Khởi tạo mô hình CNN để huấn luyện
* Thiết lập hàm mất mát CrossEntropyLoss()
* Thiết lập thuật toán tối ưu Adam optimizer với learning rate

=0,001

* Khởi tạo bactch size =32 để mỗi vòng lặp mô sẽ xử lí đồng thời 32 ảnh
* Khởi tạo bactch size test =64 để mỗi vòng lặp mô sẽ xử lí đồng thời 64 ảnh
* Khởi tạp epoch =20 số lần tập train được đưa vào mô hình để học
* Tạo các danh sách train\_losses, train\_accuracy, test\_losses, test\_accuracy để lưu trữ loss và accuracy sau mỗi epoch, nhằm mục đích vẽ biểu đồ phân tích sau khi huấn luyện
  + 1. Mô tả mô trong training

− Quá trình huấn luyện được theo dõi qua từng epoch, với các chỉ số:

* Train Loss : Mức độ sai lệch trung bình của mô hình trên dữ liệu huấn luyện.
* Test Loss : Mức độ sai lệch trung bình của mô hình trên dữ liệu kiểm tra
* Test accuracy : Tỷ lệ mẫu dự đoán đúng trên tập kiểm tra
* Train accuracy : Tỷ lệ mẫu dự đoán đúng trên tập huấn luyện.

− Trong quá trình chạy nhóm nhận thấy rằng :

* Giá trị train loss giảm dần qua các epoch
* Giá trị test lost giảm dần qua các epoch
* Giá trị train accuracy tăng dần qua các epoch
* Giá trị test accuracy tăng dần qua các epoch
  + 1. Mô tả mô sau training

− Phần chung nhóm sẽ chạy thử với 20 epoch và quan sát xem các chỉ số test accuracy, train accuracy, test lost, train lost có biến độ mạnh hay không nếu sự biến đổi rất nhỏ thì sẽ dùng lại . Nếu độ biến đổi còn mạnh nhóm sẽ chạy lại với 50 epoch để quan sát và lấy epoch có test accuracy, và train lost tốt nhất để đánh giá hiệu quả mô hình

* 1. Mô hình Mobinet\_v2
     1. Mô tả mô hình

|  |  |  |
| --- | --- | --- |
| **Khối chức năng** | **Cấu hình** | **Ghi chú** |
| Tầng đầu vào | Conv2d(1, 32, 3×3, stride=2, padding=1) | Tùy vào bước xử lí mà tham số đầu tiên sẽ thay đổi |
| Trích đặt trưng | 17 khối Inverted Residual | Tăng giảm kênh: 32 → 1280 |
| Global Pooling | GAP → output: [batch, 1280] | Không cần flatten |
| Tầng phân loại | Dropout(0.2) → Linear(1280, 10) | Phân loại 10 lớp |

* + 1. Mô tả mô hình trước training

− Trước khi tiến hành huấn luyện mô hình, nhóm đã thực hiện các bước chuẩn bị như sau:

* Tiền xử lý dữ liệu đầu vào theo 3 cách đã mô tả ở chương 2 ( mục 2.2) (mỗi lần chạy tương ứng với 1 cách )
* Sau đó tiến hành chia tập dữ liệu với tập test chiếm 20% dữ liệu, tập train chiếm 80% dữ liệu
* Khởi tạo mô hình Mobinet\_v2 để huấn luyện
* Thiết lập hàm mất mát CrossEntropyLoss()
* Thiết lập thuật toán tối ưu Adam optimizer với learning rate

=0,001

* Khởi tạo bactch size =32 để mỗi vòng lặp mô sẽ xử lí đồng thời 32 ảnh
* Khởi tạo bactch size test =64 để mỗi vòng lặp mô sẽ xử lí đồng thời 64 ảnh
* Khởi tạp epoch =20 số lần tập train được đưa vào mô hình để học
* Tạo các danh sách train\_losses, train\_accuracies, test\_losses, test\_accuracies để lưu trữ loss và accuracy sau mỗi epoch, nhằm mục đích vẽ biểu đồ phân tích sau khi huấn luyện
  + 1. Mô tả mô trong training

− Quá trình huấn luyện được theo dõi qua từng epoch, với các chỉ số:

* Train Loss : Mức độ sai lệch trung bình của mô hình trên dữ liệu huấn luyện.
* Test Loss : Mức độ sai lệch trung bình của mô hình trên dữ liệu kiểm tra
* Test accuracy : Tỷ lệ mẫu dự đoán đúng trên tập kiểm tra
* Train accuracy : Tỷ lệ mẫu dự đoán đúng trên tập huấn luyện.

− Trong quá trình chạy nhóm nhận thấy rằng :

* Giá trị train loss giảm dần qua các epoch
* Giá trị test lost giảm dần qua các epoch
* Giá trị train accuracy tăng dần qua các epoch
  + 1. Mô tả mô sau training

− Phần chung nhóm sẽ chạy thử với 20 epoch và quan sát xem các chỉ số test accuracy, train accuracy, test lost, train lost có biến độ mạnh hay không nếu sự biến đổi rất nhỏ thì sẽ dùng lại . Nếu độ biến đổi còn mạnh nhóm sẽ chạy lại với 50 epoch để quan sát và lấy epoch có test accuracy, và train lost tốt nhất để đánh giá hiệu quả mô hình

* 1. Mô hình Resnet18
     1. Mô tả nô hình

|  |  |  |
| --- | --- | --- |
| **Tầng** | **Loại tầng** | **Thông tin chi tiết** |
| 1 | Conv2d(1, 64, 7×7) | Tích chập đầu vào, thay đổi từ 3 kênh → 1 kênh (grayscale), stride=2, padding=3 |
| 2 | BatchNorm2d(64) | Chuẩn hóa đầu ra từ tầng tích chập đầu tiên |
| 3 | ReLU | Hàm kích hoạt phi tuyến tính |
| 4 | MaxPool2d(kernel\_size=3) | Giảm kích thước không gian (spatial size), stride=2 |
| 5  8 | Layer1 – BasicBlock ×2 | Gồm 2 block tích chập có shortcut connection, đầu ra 64 kênh |
| 9  12 | Layer2 – BasicBlock ×2 | Block residual, đầu ra 128 kênh, stride=2 (giảm kích thước ảnh) |
| 13  16 | Layer3 – BasicBlock ×2 | Block residual, đầu ra 256 kênh, stride=2 |
| 17 20 | Layer4 – BasicBlock ×2 | Block residual, đầu ra 512 kênh, stride=2 |
| 21 | AdaptiveAvgPool2d(output=1) | Pooling toàn cục để chuyển đầu ra về kích thước [batch\_size, 512] |
| 22 | Linear(512, 10) | Lớp fully connected phân loại 10 lớp đầu ra |

* + 1. Mô tả mô hình trước training

− Trước khi tiến hành huấn luyện mô hình, nhóm đã thực hiện các bước chuẩn bị như sau:

* Tiền xử lý dữ liệu đầu vào theo 3 cách đã mô tả ở chương 2 ( mục 2.2) (mỗi lần chạy tương ứng với 1 cách )
* Sau đó tiến hành chia tập dữ liệu với tập test chiếm 20% dữ liệu, tập train chiếm 80% dữ liệu
* Khởi tạo mô hình Mobinet\_v2 để huấn luyện
* Thiết lập hàm mất mát CrossEntropyLoss()
* Thiết lập thuật toán tối ưu Adam optimizer với learning rate

=0,001

* Khởi tạo bactch size =32 để mỗi vòng lặp mô sẽ xử lí đồng thời 32 ảnh
* Khởi tạo bactch size test =64 để mỗi vòng lặp mô sẽ xử lí đồng thời 64 ảnh
* Khởi tạp epoch =20 số lần tập train được đưa vào mô hình để học
* Tạo các danh sách train\_losses, train\_accuracies, test\_losses, test\_accuracies để lưu trữ loss và accuracy sau mỗi epoch, nhằm mục đích vẽ biểu đồ phân tích sau khi huấn luyện
  + 1. Mô tả mô trong training

− Quá trình huấn luyện được theo dõi qua từng epoch, với các chỉ số:

* Train Loss : Mức độ sai lệch trung bình của mô hình trên dữ liệu huấn luyện.
* Test Loss : Mức độ sai lệch trung bình của mô hình trên dữ liệu kiểm tra
* Test accuracy : Tỷ lệ mẫu dự đoán đúng trên tập kiểm tra
* Train accuracy : Tỷ lệ mẫu dự đoán đúng trên tập huấn luyện.

− Trong quá trình chạy nhóm nhận thấy rằng :

* Giá trị train loss giảm dần qua các epoch
* Giá trị test lost giảm dần qua các epoch
* Giá trị train accuracy tăng dần qua các epoch
  + 1. Mô tả mô sau training

− Phần chung nhóm sẽ chạy thử với 20 epoch và quan sát xem các chỉ số test accuracy, train accuracy, test lost, train lost có biến độ mạnh hay không nếu sự biến đổi rất nhỏ thì sẽ dùng lại . Nếu độ biến đổi còn mạnh nhóm sẽ chạy lại với 50 epoch để quan sát và lấy epoch có test accuracy, và train lost tốt nhất để đánh giá hiệu quả mô hình

* 1. Bảng biểu thống kê so sánh
     1. Trường hợp lúc báo cáo

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Số epoch** | **Batch size train** | **Optimizer** | **Learning rate** | **Train accuracy** | **Train lost** |
| CNN | 20 | 32 | Adam | 0.001 | 85.90% | 0.4550 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mobinet\_v2 | 20 | 32 | Adam | 0.001 | 91.92% | 0.2513 |
| Resnet18 | 20 | 32 | Adam | 0.001 | 96.20% | 0.1138 |

* + 1. Trường hợp lúc báo cáo chạy với 50 epoch

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Số epoch** | **Batch size train** | **Optimizer** | **Learning rate** | **Train accuracy** | **Train lost** |
| CNN | 50 | 32 | Adam | 0.001 | 95.81% | 0.1825 |
| Mobinet\_v2 | 50 | 32 | Adam | 0.001 | 96.28% | 0.0962 |
| Resnet18 | 50 | 32 | Adam | 0.001 | 94.83% | 0.1522 |

* + 1. Trường hợp ảnh được resize và graysacle

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Số epoch** | **Batch size train** | **Optimizer** | **Learning rate** | **Train accuracy** | **Train lost** |
| CNN | 20 | 32 | Adam | 0.001 | 72.17% | 0.9350 |
| Mobinet\_v2 | 20 | 32 | Adam | 0.001 | 95.83% | 0.1344 |
| Resnet18 | 20 | 32 | Adam | 0.001 | 97.09% | 0.0887 |

* + 1. Trường hợp ảnh được resize

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Số epoch** | **Batch size train** | **Optimizer** | **Learning rate** | **Train accuracy** | **Train lost** |
| CNN | 20 | 32 | Adam | 0.001 | 69.96% | 1.0020 |
| Mobinet\_v2 | 20 | 32 | Adam | 0.001 | 97.19% | 0.0897 |
| Resnet18 | 20 | 32 | Adam | 0.001 | 97.44% | 0.0795 |

### Chương 4: Mô tả quá trình đánh giá model qua cảm quan của nhóm cũng như kết quả trên wecode.

* 1. Trường hợp lúc báo cáo

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Test accuracy** | **Test Lost** | **Train accuracy** | **Train Lost** | **Epoch** | **Điểm wecode** |
| CNN | 70.67% | 1.3073 | 85.90% | 0.4550 | 20/20 | 82 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| Mobinet\_v2 | 84.98% | 0.5140 | 91.92% | 0.2513 | 15/20 | 89 |
| Resnet18 | 87.06% | 0.5766 | 96.20% | 0.1138 | 19/20 | 91 |

Nhận xét:

* CNN cho thấy overfitting nhẹ, train accuracy khá cao nhưng test accuracy thấp hơn 15%.
* MobileNetV2 và ResNet18 đều giảm gấp giữa train/test, đặc biệt ResNet18 đạt kết quả tốt nhất trên Wecode.
* MobileNetV2 hội tụ nhanh nhất (15 epoch) → thích hợp khi cần tốc độ huấn luyện.
  1. Trường hợp lúc báo cáo chạy với 50 epoch

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Test accuracy** | **Test lost** | **Train accuracy** | **Train lost** | **Epoch** | **Điểm wecode** |
| CNN | 72.15% | 1.7300 | 95.81% | 0.1825 | 44/50 | 82 |
| Mobinet\_v2 | 86.47% | 0.5399 | 96.28% | 0.0962 | 37/50 | 90 |
| Resnet18 | 86.91% | 0.5181 | 94.83% | 0.1522 | 17/50 | 91 |

Nhận xét:

* CNN vẫn không cải thiện đáng kể dù train rất lâu (44 epoch), thậm chí test loss còn tăng → overfitting rõ rệt.
* MobileNetV2 và ResNet18 vẫn giữ hiệu suất tốt, ít bị overfitting → mô hình mạnh và ổn định.
* ResNet18 chỉ cần 17 epoch để đạt hiệu năng tốt, cho thấy khả năng hội tụ nhanh và tổng quát hóa tốt.
  1. Trường hợp với ảnh được resize và graysacle

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Test accuracy** | **Test lost** | **Train accuracy** | **Train lost** | **Epoch** | **Điểm wecode** |
| CNN | 68.00% | 1.1549 | 72.17% | 0.9350 | 17/20 | 61 |
| Mobinet\_v2 | 94.14% | 0.2313 | 95.83% | 0.1344 | 16/20 | 91 |
| Resnet18 | 92.95% | 0.2479 | 97.09% | 0.0887 | 15/20 | 93 |

Nhận xét:

* CNN hoạt động rất kém trong trường hợp này, do không tận dụng tốt thông tin grayscale và kiến trúc đơn giản.
* MobileNetV2 và ResNet18 cải thiện rõ rệt so với các trường hợp trước, chứng tỏ việc chuyển ảnh về grayscale có lợi cho mạng sâu đã pretrained.
* ResNet18 cho điểm Wecode cao nhất (93) → rất phù hợp với dữ liệu xám đã tiền xử lý tốt.
  1. Trường hợp với ảnh chỉ resize theo kích thước đầu vao mô hình

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Test accuracy** | **Test lost** | **Train accuracy** | **Train lost** | **Epoch** | **Điểm wecode** |
| CNN | 69.96% | 1.1986 | 69.96% | 1.0020 | 20/20 | 75 |
| Mobinet\_v2 | 94.06% | 0.2031 | 97.19% | 0.0897 | 15/20 | 92 |
| Resnet18 | 94.28% | 0.2465 | 97.44% | 0.0795 | 14/20 | 93 |

Nhận xét:

* CNN cải thiện hơn so với 4.3 nhưng vẫn không bằng 4.1 hoặc 4.2. → mô hình này phù hợp với ảnh đơn giản hơn.
* MobileNetV2 và ResNet18 đạt kết quả cao nhất trong toàn bộ các trường hợp

→ resize ảnh + giữ nguyên kênh màu gốc (RGB) là tối ưu nhất cho mô hình pretrained.

* ResNet18 vẫn là model mạnh nhất về cả hiệu năng và điểm số Wecode.

**Kết luận**

|  |  |  |
| --- | --- | --- |
|  | **Ưu điểm** | **Nhược điểm** |
| CNN | Dễ xây dựng chạy nhanh | Dễ ovetfit , không ổn định |
| Mobinet\_v2 | Nhẹ, nhanh, hiệu quả tốt | Nhạy cảm với ảnh preprocessing |
| Resnet18 | Ổn định, chính xác cao | Mô hình lớn, thời gian huấn luyện lâu hơn |

### Chương 5 : Cảm nghĩ của nhóm sau khi làm bài

* 1. Kết luận

− Sau quá trình thực hiện đồ án, nhóm đã biết các bước để làm bài toán hand\_written\_digit như sau :

* + - Tiền xử lý ảnh đầu vào (resize, chuyển grayscale nếu cần).
    - Áp dụng và huấn luyện nhiều mô hình học sâu khác nhau: CNN tự xây, MobileNetV2, ResNet18.
    - Đánh giá hiệu năng các mô hình qua các chỉ số: train/test accuracy, loss, confusion matrix, điểm Wecode.
    - Thử nghiệm với nhiều hướng tiền xử lý để rút ra ảnh hưởng thực tế của dữ liệu đầu vào đến mô hình học sâu.

 Qua so sánh kết quả, nhóm nhận thấy mô hình ResNet18 hoạt động ổn định và mạnh nhất trong hầu hết các tình huống. Tuy nhiên, MobileNetV2 là mô hình cân bằng tốt giữa tốc độ và độ chính xác,

phù hợp với các ứng dụng thực tế cần thời gian phản hồi nhanh.

* 1. Cảm nghĩ

− Qua đồ án này, nhóm không chỉ được củng cố kiến thức về Machine learning , mà còn rèn luyện kỹ năng làm việc nhóm, xử lý dữ liệu thực tế và debug mô hình.

− Việc sử dụng thư viện torchvision, sklearn, matplotlib… giúp nhóm tiếp cận gần hơn với quy trình xây dựng mô hình thực tế trong ngành.

− Bên cạnh đó nhóm vẫn gặp một số khó khăn như: lỗi đọc ảnh, ảnh đầu vào bị mờ, chênh lệch dữ liệu giữa các lớp... đã giúp nhóm hiểu tầm quan trọng của tiền xử lý dữ liệu và phân tích lỗi mô hình.

− Nếu có cơ hội nhóm sẽ thử nghiệm thêm với dữ liệu thời gian thực hoặcgiao diện người dùng để nâng cấp đồ án thành ứng dụng thực tế.

## **Phần B: Đồ án dự đoán điểm môn học từ kết quả nộp bài trên wecode.**

### Chương 0: Liệt kê các thay đổi so với khi vấn đáp

* Ở phần xử lí dữ liệu đầu vào, các đặc trưng được suy ra từ dữ liệu log Wecode ban đầu đã được tính toán và chọn lọc kĩ lưỡng hơn để tránh việc các đặc trưng đó nó có mối liên hệ quá gần với nhau (hệ số tương quan lớn).
* Sử dụng các kĩ thuật chọn lọc đặc trưng thuộc nhóm “wrapper method" là Recursive Feature Elimination with Cross Validation (RFECV) và Sequential Feature Selection (SFS), bao gồm cả Forward Selection và Backward Selection để chọn lọc các đặc trưng đã được trích xuất ở phần xử lí dũ liệu, rồi dùng dữ liệu với tập các đặc trưng đã được chọn đó để huấn luyện các mô hình, so sánh kết quả với khi huấn luyện với dữ liệu chưa thực hiện chọn lọc đặc trưng.
* Sử dụng kĩ thuật tinh chỉnh tham số mô hình đó là Randomized Search Cross Validation để tìm ra bộ siêu tham số tương đối tối ưu cho mô hình trong khoảng thời gian không quá dài. Sau đó sẽ dùng các mô hình với bộ siêu tham số mới này để huấn luyện mô hình với cả tập dữ liệu không qua chọn lọc đặc trưng và tập dữ liệu đã được chọn lọc đặc trưng để làm cơ sở so sánh kết quả.

### Chương 1: Giới thiệu bài toán và dữ liệu

* Trong môn học Nhập môn lập trinh (IT001), sinh viên thường xuyên thực hiện các bài tập lập trình trên hệ thống Wecode. Mỗi lần nộp bài sẽ được ghi lại trong bản log chi tiết trong hệ thống, bao gồm thông tin như: mã bài tập, mã bài toán, mã sinh viên (đã được ẩn danh), trạng thái chấm điểm, hệ số trễ khi nộp bài, điểm số, ... Vậy nên, nó là dữ liệu có thể được dùng để đánh giá khả năng và trình độ của sinh viên nếu được tổng hợp và xử lí đúng cách.
* Bài toán đặt ra là: Dựa trên lịch sử nộp bài của sinh viên trên hệ thống Wecode, hãy dự đoán các loại điểm của môn học IT001, bao gồm:
* Điểm thực hành
* Điểm quá trình
* Điểm cuối kì
* Điểm trung bình tích lũy (TBTL) sau năm học đầu tiên (có thể tính toán từ các điểm đã được dự đoán ở phía trên)
* Dữ liệu được cung cấp gồm:
* \_File annonimized.csv : chứa dữ liệu nộp bài đã được ẩn danh
* Các file như th-public.csv , qt-public.csv , ck-public.csv v.v...: chứa điểm thật của khoảng 800 sinh viên
* Input: dữ liệu nộp bài của sinh viên trên hệ thống wecode
* Output: điểm số ứng với cột điểm muốn dự đoán (thực hành, quá trình, cuối kì)

### Chương 2: Data processing

* Quy trình xử lý dữ liệu được thiết kế để trích xuất các đặc trưng quan trọng từ bài nộp bài Wecode, nhằm phục vụ cho bài toán dự đoán điểm thông qua các bài tập lập trình.Toàn bộ pipeline xử lý gồm các bước chính sau:
  1. **Đọc và chuẩn hóa dữ liệu**
* Đọc dữ liệu log nộp bài từ file ‘annonimized.csv’ và dữ liệu điểm thật từ ‘th-public.csv’, ‘qt- public.csv’, ‘qt-public.csv’, ‘tbtl-public.csv’.
* Chuẩn hóa tên cột để thống nhất định danh sinh viên và bài tập.

**2.2 Xử lý định dạng thời gian**

* Chuyển các cột ‘created\_at’, ‘updated\_at’ về định dạng ‘datetime’ chuẩn để phân tích thời gian.

**2.3 Tạo đặc trưng liên quan đến điểm số**

* Tính toán điểm sơ bộ có trọng số: ‘coefficient \* pre\_score / 100’.
* Xác định bài đúng hoàn toàn (score 10000) và bài đúng cuối cùng.
* Tính tổng điểm cuối có trọng số từ các bài đúng cuối cùng.

**2.4 Tính thời gian làm bài**

* Với mỗi bài, tính thời gian từ lần nộp đầu đến lần cuối, sau đó lấy trung bình trên toàn bộ bài đã làm.

**2.5 Tỷ lệ làm đúng ngay lần đầu**

* Tính tỷ lệ số bài được làm đúng hoàn toàn ngay từ lần nộp đầu tiên.

**2.6 Tổng số bài đã giải đúng**

* Đếm số lượng problem được giải đúng (score 10000) với điều kiện ‘is\_final = 1’.

**2.7 Trích xuất đặc trưng hành vi**

* Tổng lượt nộp, số bài đã thử, điểm TB mỗi lần nộp, số lỗi biên dịch, v.v.

**2.8 Kết hợp với điểm thật và chuẩn hóa**

* Ghép với bảng điểm thật ứng với điểm muốn dự đoán: ‘th-public.csv’, ‘qt-public.csv’, ‘qt- public.csv’, ‘tbtl-public.csv’.
* Chuẩn hóa toàn bộ đặc trưng số bằng ‘StandardScaler’.

**2.9 Bảng: Tóm tắt các đặc trưng đã trích xuất**

|  |  |
| --- | --- |
| Tên đặc trưng | Mô tả |
| total\_submissions | Tổng số lượt nộp bài |
| total\_correct\_submissions | Số lượt nộp đúng hoàn toàn |
| number\_of\_tried\_problems | Số bài đã thử |
| number\_of\_tried\_assignments | Số assignment đã thử |
| total\_weighted\_final\_score | Tổng điểm có trọng số (bài đúng) |
| score\_per\_submission | Điểm trung bình mỗi lượt nộp |
| accuracy | Tỷ lệ đúng hoàn toàn |
| submission\_per\_problem | Số lượt nộp trung bình mỗi bài |
| avg\_compilation\_errors\_per\_problem | Số lỗi biên dịch trung bình mỗi bài |
| avg\_interval\_between\_first\_and\_last\_sub | Thời gian trung bình giữa lần nộp đầu và cuối (phút) |
| first\_try\_success\_rate | Tỷ lệ đúng ngay từ lần đầu |
| number\_of\_solved\_problems | Số bài đã giải đúng hoàn toàn |

### Chương 3: Training

* 1. **Mô hình thử nghiệm ban đầu với tham số mặc định**
* Các mô hình hồi quy đã được thử nghiệm trên tập dữ liệu chưa chọn lọc đặc trưng, với các tham số mặc định. Dữ liệu được chia thành tập huấn luyện (train) và kiểm tra (test). Các mô hình gồm:
* Linear Regression (LR)
* Support Vector Regressor (SVR)
* Random Forest Regressor (RFR)
* Gradient Boosting Regressor (XGB)
* Mỗi mô hình được huấn luyện và đánh giá bằng hệ số xác định R² trên tập kiểm tra.

**3.2 Lựa chọn đặc trưng (Feature Selection)**

* Ba phương pháp được áp dụng để chọn lọc đặc trưng đầu vào cho mô hình:
* RFECV (Recursive Feature Elimination with Cross-Validation)
* Sequential Feature Selection (Forward/Backward)
* Không chọn lọc (sử dụng toàn bộ)
* Mỗi phương pháp được dùng để lọc ra tập đặc trưng tốt nhất cho từng mô hình, nhằm cải thiện độ chính xác và giảm overfitting.

**3.3 Tinh chỉnh tham số mô hình (Hyperparameter Tuning)**

* Các mô hình (trừ Linear Regression) được tinh chỉnh tham số bằng RandomizedSearchCV. Sử dụng 5-fold cross-validation với thước đo R² để chọn tổ hợp tham số tốt nhất.

**3.4 Đánh giá mô hình sau tinh chỉnh**

* Sau khi tinh chỉnh, mô hình được huấn luyện lại và đánh giá trên tập kiểm tra. Kết quả R² sau khi tinh chỉnh và không được tinh chỉnh được ghi nhận.

**Bảng so sánh kết quả giá trị R2 của các mô hình trong dự đoán điểm thực hành**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tên mô hình | Đã tinh chỉnh các tham số | | | | Không tinh chỉnh các tham số | | | |
| Linear Regressi on | SVR | Random Forrest | XGBoost | Linear Regressi on | SVR | Random Forrest | XGBoost |
| RFEC V | - | - | 0.4347 | 0.4240 | 0.3790 | - | 0.4561 | 0.4525 |
| SFS | - | 0.4412 | 0.4317 | 0.4377 | 0.3863 | 0.3970 | 0.4156 | 0.4629 |
| Không chọn lọc | - | 0.4185 | 0.4380 | 0.4415 | 0.3815 | 0.3712 | 0.4508 | 0.5177 |

**Bảng so sánh kết quả giá trị R2 của các mô hình trong dự đoán điểm quá trình**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tên mô hình | Đã tinh chỉnh các tham số | | | | Không tinh chỉnh các tham số | | | |
| Linear Regressi on | SVR | Random Forrest | XGBoost | Linear Regressi on | SVR | Random Forrest | XGBoost |
| RFEC V | - | - | 0.0574 | 0.0939 | -0.0878 | - | 0.0964 | 0.0404 |
| SFS | - | 0.1275 | -0.1701 | -0.1520 | 0.0535 | 0.0765 | -0.2202 | -0.1540 |
| Không chọn lọc | - | 0.1762 | 0.0941 | 0.0981 | 0.0678 | 0.1368 | 0.0885 | 0.0431 |

**Bảng so sánh kết quả giá trị R2 của các mô hình trong dự đoán điểm cuối kì**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tên mô hình | Đã tinh chỉnh các tham số | | | | Không tinh chỉnh các tham số | | | |
| Linear Regressi on | SVR | Random Forrest | XGBoost | Linear Regressi on | SVR | Random Forrest | XGBoost |
| RFEC V | - | - | 0.2266 | 0.2354 | 0.2442 | - | 0.2336 | 0.1966 |
| SFS | - | 0.2474 | 0.2227 | 0.1972 | 0.1748 | 0.2417 | 0.1836 | 0.2082 |
| Không chọn lọc | - | 0.2585 | 0.2266 | 0.2354 | 0.2434 | 0.2159 | 0.2421 | 0.1906 |

Các mô hình sau tinh chỉnh đã cải thiện độ chính xác (ngoại trừ điểm thực hành), đặc biệt là SVR,XGBoost và Random Forest. Việc chọn lọc đặc trưng giúp rút gọn mô hình và giảm nhiễu.

### Chương 4: Evaluation

* 1. **Đánh giá**
* Sau khi tiến hành huấn luyện các mô hình với dữ liệu từ hệ thống Wecode, nhóm đã tiến hành đánh giá các mô hình qua hai khía cạnh chính: cảm quan chất lượng dự đoán và kết quả chính thức trên hệ thống chấm điểm của Wecode.
* Qua quá trình thử nghiệm, nhóm nhận thấy các mô hình tree-based như Random Forest và XGBoost có khả năng học tốt các mối quan hệ phi tuyến giữa đặc trưng và điểm thực. Mô hình Linear Regression dễ bị ảnh hưởng bởi outlier và có độ chính xác thấp nhất. Support Vector Regression (SVR) cho kết quả trung bình nhưng lại có kết quả cao với dự đoán điểm quá trình.
  1. **Đánh giá kết quả trên Wecode**
* Nhóm đã nộp kết quả từ các mô hình sau tinh chỉnh và không tinh chỉnh để thu được bảng đánh giá như sau:

**Bảng so sánh kết quả dự đoán điểm thực hành các mô hình**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tên mô | Đã tinh chỉnh các tham số | | | Không tinh chỉnh các tham số | | |
| hình | RFECV | SFS | Không chọn lọc | RFECV | SFS | Không chọn lọc |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Linear Regression | - | - | - | 1956 | 1913 | 1962 |
| SVR | - | 2782 | 2833 | - | 2366 | 2599 |
| Random Forest | 2603 | 2751 | 2814 | 2642 | 2565 | 2653 |
| XGBoost | 2844(Cao  nhất) | 2099 | 2449 | 2580 | 2294 | 2580 |

**Bảng so sánh kết quả dự đoán điểm quá trình các mô hình**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tên mô hình | Đã tinh chỉnh các tham số | | | Không tinh chỉnh các tham số | | |
| RFECV | SFS | Không chọn lọc | RFECV | SFS | Không chọn lọc |
| Linear Regression | - | - | - | 62 | 204 | 173 |
| SVR | - | -1250 | -341 | - | -2251 | -1018 |
| Random Forest | -426 | 402 | 546(Cao  nhất) | -1291 | 103 | 146 |
| XGBoost | -1243 | -347 | -277 | -2014 | -1535 | -1820 |

**Bảng so sánh kết quả dự đoán điểm cuối kì các mô hình**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tên mô hình | Đã tinh chỉnh các tham số | | | Không tinh chỉnh các tham số | | |
| RFECV | SFS | Không chọn lọc | RFECV | SFS | Không chọn lọc |
| Linear Regression | - | - | - | 1267 | 1074 | 1258 |
| SVR | - | 2036 | 2127(Cao  nhất) | - | 1641 | 1896 |
| Random Forest | 2097 | 2079 | 2109 | 1791 | 2003 | 2016 |
| XGBoost | 1999 | 1979 | 1728 | 1971 | 1719 | 1948 |

* Dựa trên bảng thống kê, XGBoost là mô hình được nhóm lựa chọn để nộp chính thức nhờ khả năng tổng quát hóa tốt và điểm R² cao nhất.

### Chương 5: Kết luận.

### 5.1 kết luận

### Việc thử nghiệm nhiều mô hình khác nhau như Linear Regression, SVR, Random Forest và XGBoost giúp nhóm có cái nhìn toàn diện về ưu - nhược điểm của từng thuật toán.

### Quy trình chọn lọc đặc trưng (feature selection) đã phần nào cải thiện độ chính xác, giúp giảm hiện tượng overfitting và giảm số chiều đầu vào cho mô hình.

### Tinh chỉnh tham số bằng RandomizedSearchCV là bước quan trọng, giúp nâng cao hiệu quả dự đoán rõ rệt cho các mô hình phi tuyến (SVR, RFR, XGB).

### Tuy nhiên, mô hình Linear Regression nhìn chung hoạt động không hiệu quả do không phù hợp với dữ liệu có quan hệ phi tuyến và nhạy cảm với ngoại lệ.

### 5.2 kết luận

### Nhóm nhận thấy việc xây dựng mô hình dự đoán là một quá trình lặp lại nhiều lần giữa huấn luyện, đánh giá và cải tiến, đòi hỏi kiến thức không chỉ về thuật toán mà còn về xử lý dữ liệu.

### Điểm số chưa cao là điều dễ hiểu, vì nhóm vẫn còn nhiều thiếu sót trong việc xử lý nhiễu, chuẩn hóa dữ liệu, hoặc khai thác thêm đặc trưng tiềm năng bên cạnh đó kết quả dư đoán có thể ảnh hưởng nhiều từ các yếu tố thưc tế

### Dù kết quả còn hạn chế, nhóm đã học được quy trình xây dựng mô hình từ A đến Z, bao gồm chọn mô hình, lọc đặc trưng, tinh chỉnh tham số, và đánh giá bằng nhiều tiêu chí khác nhau.