BỘ XÂY DỰNG
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH

Viện công nghệ thông tin và Điện, điện tử

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

UTH
UNIVERSITY
OF TRANSPORT
HOCHIMINH CITY

Final Report Lab

Lecturer	:	PhD. Nguyễn Thị Khánh Tiên
Course	:	Deep Learning
Course Code	:	010112710502
Group	:	08
Link Repo	:	Link Repo

TP. HỒ CHÍ MINH, 2026

Đóng góp của các thành viên			
STT	Tên thành viên	MSSV	Nội dung công việc
1	Lê Văn An	056205001827	- Code practice 2 - Nội dung báo cáo practice 2: Lab 2 - Nội dung báo cáo chương 4: Tổng kết
2	Phạm Gia Bảo	093205005065	- Code practice 1 - Nội dung báo cáo chương 1
3	Dương Hưng	086205003910	- Code practice 1 - Nội dung báo cáo chương 1
4	Huỳnh Hậu	066205010571	- Code practice 3 - Nội dung báo cáo chương 3
5	Huỳnh Thị Cẩm Giang	051305006893	- Code practice 3 - Nội dung báo cáo chương 3

Tóm tắt

Báo cáo tổng kết các bài lab của học phần Deep Learning trình bày một cách hệ thống quá trình tiếp cận và thực hành các kỹ thuật học sâu từ cơ bản đến nâng cao. Nội dung Lab 1 tập trung vào việc xây dựng và huấn luyện mô hình CNN bằng PyTorch cho bài toán phân loại ảnh FashionMNIST, giúp người học nắm vững quy trình chuẩn gồm chuẩn bị dữ liệu, thiết kế kiến trúc mạng, huấn luyện theo vòng lặp forward–backward, đánh giá bằng accuracy và phân tích kết quả thông qua biểu đồ, confusion matrix và dự đoán trực quan.

Lab 2 mở rộng sang hướng sử dụng mô hình CNN tiền huấn luyện và transfer learning trên tập CIFAR-10, qua đó làm rõ lợi ích của việc tận dụng trọng số từ ImageNet, so sánh hiệu năng giữa VGG11, ResNet18 và DenseNet121, đồng thời rèn luyện kỹ năng đóng băng tầng, thay đổi head phân loại và theo dõi huấn luyện bằng TensorBoard.

Lab 3 chuyển sang lĩnh vực xử lý ngôn ngữ tự nhiên với Hugging Face Transformers, bao gồm cả inference nhanh bằng pretrained model cho bài toán sentiment analysis và fine-tuning DistilBERT cho phân loại văn bản nhị phân, nhấn mạnh vai trò của tokenizer, pipeline huấn luyện với Trainer, cũng như đánh giá bằng Accuracy và F1-score.

Phần tổng kết khẳng định sinh viên đã thu được kiến thức nền tảng về backpropagation, các kiến trúc CNN, RNN/LSTM và Transformer, kỹ năng chuẩn bị dữ liệu, sử dụng pretrained model, theo dõi huấn luyện và làm việc nhóm, đồng thời định hướng tiếp tục áp dụng các kiến thức này vào các bài toán thị giác máy tính, NLP và dữ liệu chuỗi trong tương lai.

Mục lục

Chương 1. Lab 1 – PyTorch FashionMNIST Classification

1.1 Mục tiêu & mô tả bài toán

Lab 1 được thiết kế nhằm giúp chúng ta nắm được quy trình cơ bản khi xây dựng và huấn luyện một mô hình học sâu bằng PyTorch cho bài toán phân loại ảnh. Thông qua bài lab, chúng ta thực hành đầy đủ các bước quan trọng gồm: chuẩn bị dữ liệu đầu vào (dataset, data loader, transform), xây dựng mô hình mạng nơ-ron (neural network), huấn luyện mô hình theo vòng lặp chuẩn (forward – loss – backward – optimize), đánh giá hiệu năng bằng độ chính xác (accuracy) và cuối cùng là lưu/khôi phục mô hình để sử dụng lại.

Bài toán được sử dụng trong lab là phân loại ảnh FashionMNIST, một tập dữ liệu gồm ảnh xám kích thước 28×28 thuộc 10 nhóm sản phẩm thời trang. Mỗi ảnh chỉ thuộc đúng một lớp (multi-class classification), do đó mô hình sẽ dự đoán nhãn bằng cách xuất ra logits cho 10 lớp và chọn lớp có xác suất cao nhất. Bài toán này phù hợp cho việc luyện tập vì dữ liệu vừa đủ đơn giản để huấn luyện nhanh, nhưng vẫn có độ khó nhất định do một số lớp có đặc điểm hình ảnh khá giống nhau (ví dụ: T-shirt/top, Shirt, Pullover, Coat).

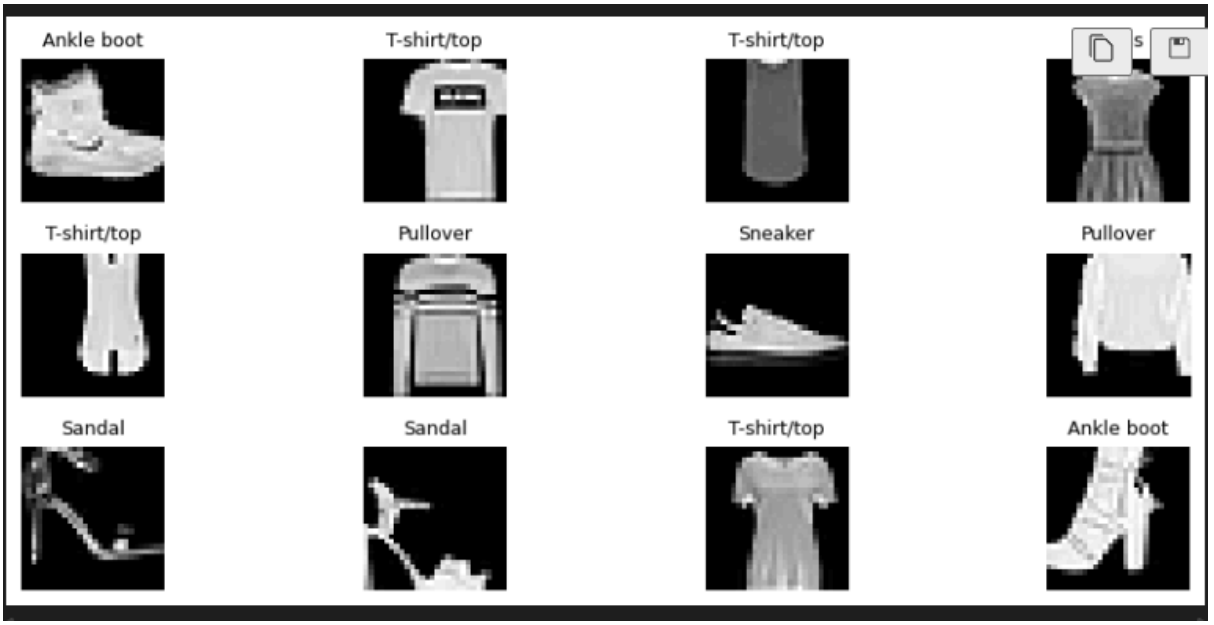
1.2 Chuẩn bị dữ liệu (Dataset, DataLoader, Transforms)

Trong notebook thực hành, dữ liệu được tải từ torchvision.datasets.FashionMNIST và lưu trong thư mục ./data. Tập dữ liệu gồm hai phần: tập huấn luyện (train) và tập kiểm tra (test), trong đó tập train có 60.000 ảnh và tập test có 10.000 ảnh. Việc tách sẵn train/test giúp đánh giá khả năng tổng quát hóa của mô hình một cách khách quan, tránh nhầm lẫn giữa học thuộc dữ liệu và học được quy luật thực sự.

Về tiền xử lý, notebook xây dựng hai luồng transform riêng cho train và test. Đối với tập train, dữ liệu được tăng cường (augmentation) bằng các phép biến đổi như RandomCrop(28, padding=4) và RandomHorizontalFlip(p=0.5) nhằm tạo thêm sự đa dạng trong dữ liệu, giúp mô hình ít bị phụ thuộc vào một vị trí hay hướng cố định của vật thể. Sau đó ảnh được chuyển về tensor và chuẩn hóa bằng Normalize((0.5,), (0.5,)) để đưa phân phối pixel về vùng ổn định hơn cho quá trình tối ưu. Đối với tập test, notebook chỉ áp dụng ToTensor và Normalize, không dùng augmentation để đảm bảo kết quả đánh giá phản ánh đúng dữ liệu gốc.

Dữ liệu sau khi chuẩn bị được đưa vào DataLoader với kích thước batch là 128. Tập train được shuffle để giảm hiện tượng mô hình học theo thứ tự dữ liệu, còn tập test không

shuffle nhằm giữ tính ổn định khi đánh giá. Ngoài ra notebook có phần hiển thị ảnh mẫu để kiểm tra trực quan dữ liệu đã được tải đúng và nhãn tương ứng hợp lý trước khi tiến hành huấn luyện.



1.3 Xây mô hình neural network

Mô hình sử dụng trong lab là một CNN có tên FashionCNNPro, được thiết kế theo hướng mạnh hơn mô hình CNN cơ bản nhờ sử dụng các block chuẩn hóa và dropout. Kiến trúc mô hình được xây dựng theo dạng các stage, trong đó mỗi stage gồm block Conv – BatchNorm – ReLU lặp 2 lần nhằm tăng khả năng trích xuất đặc trưng. Cụ thể, stage 1 chuyển từ 1 kênh ảnh đầu vào sang 32 kênh, stage 2 nâng lên 64 kênh và stage 3 nâng lên 128 kênh. Sau stage 1 và stage 2, mô hình dùng MaxPooling 2×2 để giảm kích thước feature map theo hướng 28×28 → 14×14 → 7×7, giúp giảm tải tính toán và tăng tính khái quát.

Một điểm đáng chú ý trong thiết kế là mô hình có sử dụng dropout theo từng giai đoạn với các mức Dropout2d(0.10) và Dropout2d(0.15) sau pooling, đồng thời sử dụng Dropout(0.25) trước lớp fully-connected. Việc này giúp giảm overfitting, đặc biệt khi mô hình bắt đầu học được các đặc trưng mạnh. Ngoài ra, stage cuối sử dụng AdaptiveAvgPool2d((1,1)) (Global Average Pooling) để gom toàn bộ đặc trưng về kích thước 1×1, từ đó giảm số tham số so với việc flatten feature map lớn, giúp mô hình gọn hơn và ổn định hơn khi huấn luyện. Cuối cùng, mô hình dùng lớp tuyến tính Linear(128 → 10) để đưa ra logits cho 10 lớp FashionMNIST.

1.4 Huấn luyện (forward–loss–backward–optimize)

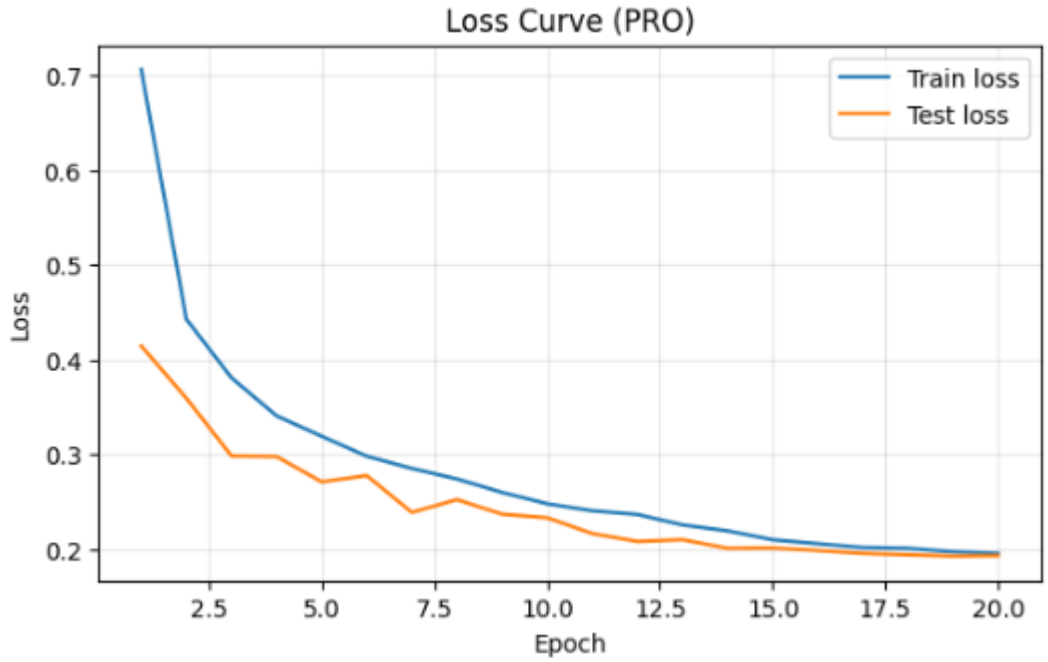
Trong quá trình huấn luyện, notebook sử dụng hàm mất mát CrossEntropyLoss, phù hợp cho phân loại đa lớp. Bộ tối ưu được lựa chọn là AdamW với learning rate 1e-3 và weight decay 1e-4, trong đó AdamW có ưu điểm kết hợp động lượng thích nghi theo tham số và regularization tốt hơn so với Adam thông thường. Ngoài ra, notebook sử dụng scheduler CosineAnnealingLR để giảm learning rate theo dạng cosine theo tiến trình epoch, giúp mô hình học nhanh ở giai đoạn đầu và tinh chỉnh tốt hơn về cuối.

Vòng lặp huấn luyện được triển khai theo đúng chuẩn của PyTorch. Mỗi batch dữ liệu sẽ được đưa qua mô hình để tính logits (forward), sau đó tính loss bằng CrossEntropy, tiếp đến xóa gradient cũ, thực hiện lan truyền ngược (backward) và cập nhật trọng số (optimizer.step). Loss và accuracy được cộng dồn theo số lượng mẫu để tính giá trị trung bình toàn epoch, đảm bảo việc thống kê phản ánh đúng dữ liệu thay vì phụ thuộc vào số batch.

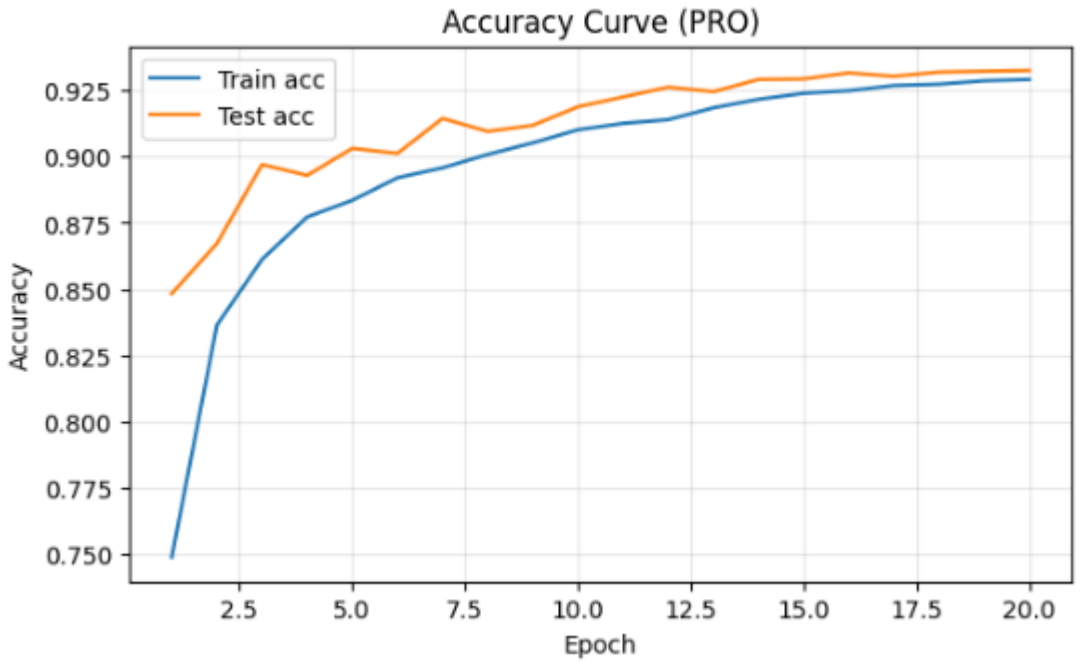
Trong notebook, tham số EPOCHS được đặt là 20 và có thiết kế cơ chế theo dõi mô hình tốt nhất thông qua test accuracy. Mỗi epoch sau khi đánh giá trên tập test, nếu accuracy cải thiện, mô hình sẽ được lưu checkpoint vào file best_fashioncnn_pro.pth nhằm đảm bảo giữ lại phiên bản tốt nhất.

1.5 Đánh giá (accuracy)

Sau mỗi epoch, mô hình được chuyển sang chế độ eval() và đánh giá trên tập test để tính test loss và test accuracy. Accuracy được tính bằng cách lấy nhãn dự đoán là chỉ số có logits lớn nhất và so sánh với nhãn thật. Kết quả huấn luyện trong notebook cho thấy mô hình cải thiện rõ rệt qua từng epoch: test accuracy tăng mạnh từ giai đoạn đầu và dần ổn định về sau khi learning rate giảm.



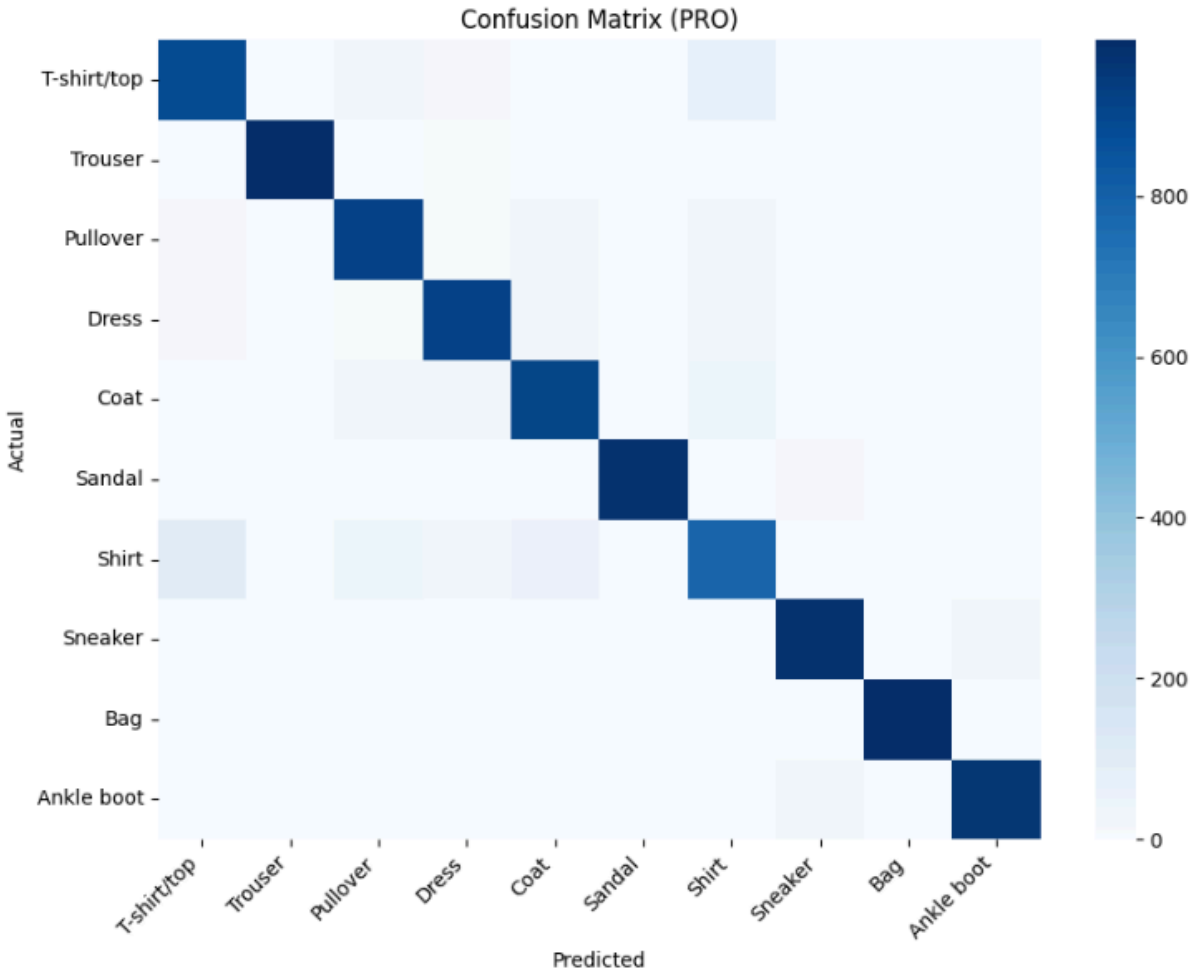
Trong lần chạy được ghi nhận, mô hình đạt best test accuracy = 93.16% tại epoch 16, và các epoch sau đó dao động quanh mức 93%. Điều này cho thấy mô hình học tốt và có khả năng tổng quát hóa ổn định trên dữ liệu kiểm tra, phù hợp với kỳ vọng của bài toán FashionMNIST.



1.6 Thử nghiệm: hyperparameters + visualize loss + predicted vs actual

Bên cạnh việc huấn luyện và đánh giá bằng số liệu, notebook còn trực quan hóa quá trình học thông qua biểu đồ loss và accuracy theo epoch. Loss curve cho thấy loss giảm dần theo thời gian, đồng thời accuracy curve cho thấy độ chính xác tăng và dần đạt trạng thái ổn định. Việc quan sát song song train và test curves giúp nhận biết dấu hiệu overfitting/underfitting. Trong bài làm này, đường test accuracy tăng và duy trì tốt, cho thấy mô hình không bị overfitting nặng.

Để phân tích sâu hơn, notebook in Classification Report và vẽ Confusion Matrix trên tập test. Kết quả classification report cho thấy đa số lớp có precision/recall rất cao, tuy nhiên lớp Shirt có chỉ số thấp hơn đáng kể (precision khoảng 0.81, recall khoảng 0.79), phản ánh đúng đặc thù dữ liệu FashionMNIST khi các loại áo dễ bị nhầm lẫn do hình dáng gần giống nhau ở ảnh xám 28×28. Confusion matrix hỗ trợ quan sát trực quan các cặp lớp hay bị nhầm, giúp định hướng cải thiện mô hình (tăng augmentation, thử kiến trúc khác hoặc tinh chỉnh dropout/lr).



Cuối cùng, notebook có phần hiển thị Predicted vs Actual bằng cách lấy ngẫu nhiên một số ảnh trong tập test và hiển thị nhãn dự đoán so với nhãn thật. Cách kiểm tra này giúp xác nhận rằng mô hình không chỉ đẹp điểm số mà còn dự đoán hợp lý trên các ví dụ trực quan, đồng thời chỉ ra những trường hợp sai để phân tích nguyên nhân.

Chương 2. Lab 2 – Pre-trained CNN & Transfer Learning

2.1 Giới thiệu và mục tiêu

Bài tập này được thiết kế nhằm giúp sinh viên làm quen với việc lập trình trong lĩnh vực học sâu, đồng thời tiếp cận quy trình sử dụng các mô hình đã được huấn luyện sẵn (pre-trained models). Nội dung bài tập tập trung vào việc sử dụng tập dữ liệu CIFAR-10 cho quá trình huấn luyện mô hình, bao gồm các bước chia tập dữ liệu, xây dựng DataLoader và thiết lập các tham số liên quan trong quá trình nạp dữ liệu.

Về phía mô hình, sinh viên thực hành việc gọi các kiến trúc mạng nơ-ron sâu từ thư viện có sẵn với trọng số mặc định, khởi tạo hàm mất mát, bộ tối ưu và xây dựng vòng lặp huấn luyện hoàn chỉnh. Trong quá trình huấn luyện, TensorBoard được sử dụng để theo dõi trực quan các chỉ số như loss và độ chính xác, qua đó hỗ trợ việc phân tích quá trình học và đánh giá mức độ hội tụ của mô hình. Cuối cùng, mô hình được đánh giá thông qua các chỉ số đo lường phù hợp nhằm phân tích hiệu năng tổng thể sau huấn luyện.

2.2 Mô hình tiền huấn luyện sử dụng

Trong Practice 2, ba mô hình mạng nơ-ron tích chập tiền huấn luyện phổ biến là VGG11, ResNet18 và DenseNet121 được sử dụng nhằm phục vụ cho các bài toán xử lý ảnh. Đây đều là các kiến trúc CNN tiêu biểu, được thiết kế chuyên biệt cho dữ liệu không gian hai chiều và đã chứng minh hiệu quả cao trong các nhiệm vụ thị giác máy tính.

Các mô hình này được huấn luyện sẵn trên tập dữ liệu ImageNet và được tải trực tiếp từ thư viện torchvision.models với trọng số mặc định. Việc sử dụng các mô hình tiền huấn luyện giúp tận dụng tri thức đã học từ dữ liệu lớn, đồng thời giảm đáng kể chi phí huấn luyện so với việc xây dựng và huấn luyện mô hình từ đầu. Thông qua việc triển khai và so sánh các kiến trúc khác nhau, sinh viên có thể quan sát sự khác biệt về cấu trúc mạng, số lượng tham số và hiệu năng mô hình trong cùng một điều kiện huấn luyện.

2.3 Chuẩn bị dữ liệu và thiết lập huấn luyện

2.3.1. Chuẩn bị dữ liệu

Trong bài thực hành này, tập dữ liệu CIFAR-10 được sử dụng cho quá trình huấn luyện và đánh giá mô hình. CIFAR-10 bao gồm tổng cộng 60.000 ảnh màu kích thước 32×32, trong đó 50.000 ảnh được sử dụng cho tập huấn luyện và 10.000 ảnh cho tập kiểm tra, thuộc 10 lớp đối tượng gồm airplane, automobile, bird, cat, deer, dog, frog, horse, ship và truck.

Tập huấn luyện ban đầu tiếp tục được chia theo tỷ lệ 80/20 để tạo tập huấn luyện và tập validation, tương ứng với 40.000 ảnh cho huấn luyện và 10.000 ảnh cho đánh giá trong quá trình training. Trước khi đưa vào mô hình, dữ liệu ảnh được tiền xử lý bằng cách chuyển đổi sang tensor, thay đổi kích thước ảnh về 224×224 nhằm phù hợp với các mô hình tiền huấn luyện trên ImageNet, đồng thời chuẩn hóa giá trị pixel theo các tham số mean và standard deviation tương ứng. Sau bước tiền xử lý, dữ liệu được đưa vào Data Loader để phục vụ cho quá trình huấn luyện theo từng batch.

2.3.2. Thiết lập huấn luyện

Đối với tập huấn luyện, batch size được thiết lập là 16 và dữ liệu được xáo trộn ngẫu nhiên sau mỗi epoch nhằm tránh hiện tượng mô hình học theo thứ tự cố định của dữ liệu và cải thiện khả năng tổng quát hóa. Đối với tập validation và tập test, batch size được thiết lập là 32 và không áp dụng shuffle nhằm tăng tốc độ suy luận và đảm bảo tính nhất quán trong quá trình đánh giá.

Các mô hình tiền huấn luyện được tải với trọng số mặc định và áp dụng chiến lược đóng băng (freeze) các lớp trích xuất đặc trưng, chỉ huấn luyện phần phân loại cuối. Do các mô hình tiền huấn luyện ban đầu được thiết kế cho bài toán phân loại 1.000 lớp trên ImageNet, lớp phân loại cuối được thay thế để phù hợp với bài toán CIFAR-10 gồm 10 lớp. Quá trình huấn luyện được thực hiện bằng hàm mất mát CrossEntropyLoss và bộ tối ưu SGD với learning rate phù hợp. Trong suốt quá trình huấn luyện, các chỉ số loss và accuracy được theo dõi trên cả tập huấn luyện và tập đánh giá, đồng thời được trực quan hóa bằng

TensorBoard nhằm hỗ trợ phân tích quá trình hội tụ và so sánh hiệu năng giữa các mô hình tiền huấn luyện khác nhau.

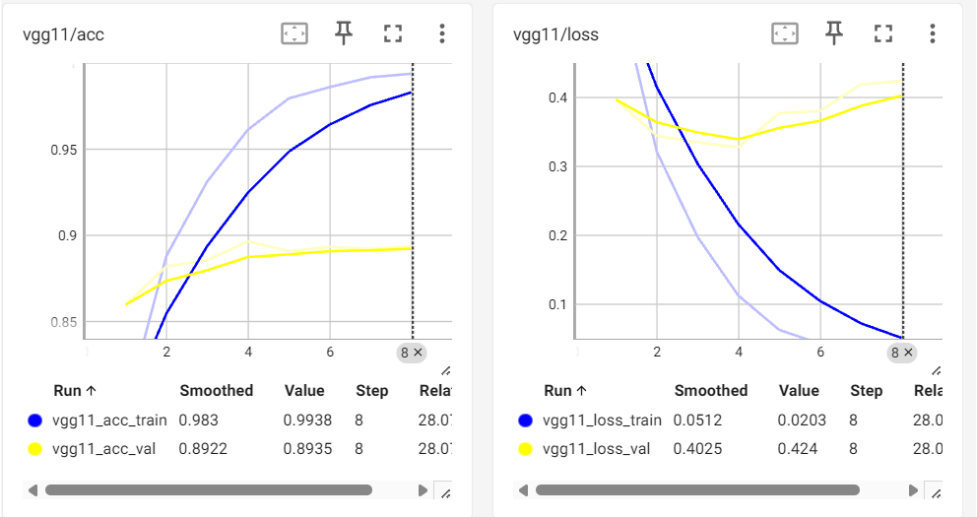
2.4 Huấn luyện và tinh chỉnh mô hình

Quá trình huấn luyện mô hình được thực hiện trong 8 epoch nhằm giúp sinh viên làm quen với toàn bộ quy trình huấn luyện và xây dựng mô hình baseline trong phạm vi bài thực hành. Số lượng epoch này chưa hướng tới việc tối ưu hiệu năng tối đa; để đạt được kết quả tốt hơn, mô hình cần được huấn luyện với số epoch lớn hơn cùng các bước tinh chỉnh bổ sung.

Trong mỗi epoch, quá trình huấn luyện được thực hiện thông qua hàm train_one_epoch. Ở giai đoạn này, dữ liệu từ tập huấn luyện được đưa vào mô hình theo từng batch để thực hiện suy luận, tính toán hàm mất mát giữa kết quả dự đoán và nhãn thực tế, sau đó thực hiện lan truyền ngược nhằm cập nhật trọng số của mô hình thông qua bộ tối ưu đã thiết lập. Sau khi hoàn tất giai đoạn huấn luyện, mô hình được đánh giá trên tập validation để theo dõi các chỉ số loss và accuracy, qua đó hỗ trợ phân tích quá trình hội tụ và khả năng tổng quát hóa của mô hình. Kết quả của train/loss, val/loss; train/acc, val/acc đều được ghi vào log để có thể trực quan trên Tensorboard.

2.5 Đánh giá và nhận xét kết quả

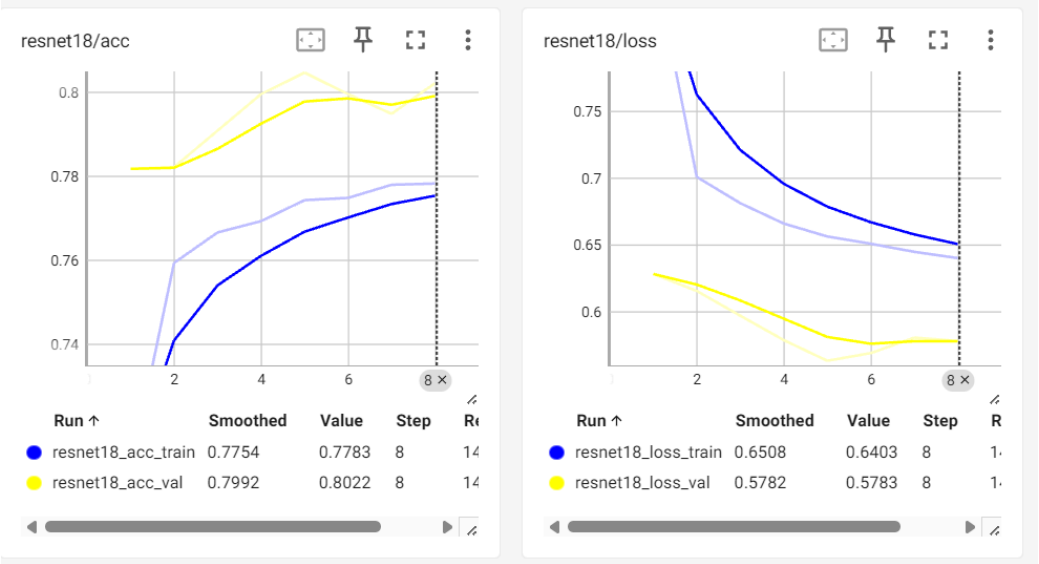
2.5.1. Mô hình VGG11



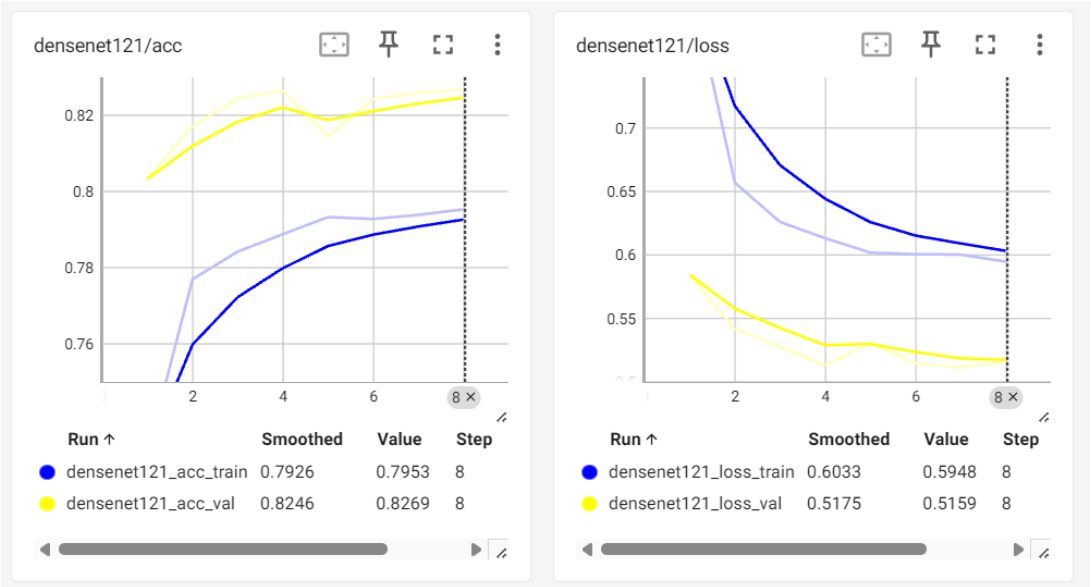
Kết quả với các chỉ số precision, recall và f1 đều đạt ở mức tốt (được đánh giá trên tập test). Tuy nhiên theo biểu đồ loss thì model đã bị overfit.

2.5.2. Mô hình Resnet18

Mô hình ResNet18 cho thấy quá trình huấn luyện ổn định trong 8 epoch, với accuracy trên tập huấn luyện đạt khoảng 0,78 và trên tập validation đạt khoảng 0,80, cho thấy khả năng tổng quát hóa tương đối tốt và chưa xuất hiện overfitting. Hàm mất mát trên cả hai tập giảm đều và ổn định, tuy nhiên các chỉ số có xu hướng chững lại ở các epoch cuối, phản ánh mô hình mới đạt mức baseline và cần thêm huấn luyện hoặc tinh chỉnh để cải thiện hiệu năng.



2.5.3. Mô hình Densenet121



Mô hình DenseNet121 cho thấy hiệu năng tốt hơn so với ResNet18 trong cùng điều kiện huấn luyện. Độ chính xác trên tập huấn luyện đạt khoảng 0,80, trong khi accuracy trên

tập validation đạt xấp xỉ 0,83, cho thấy khả năng tổng quát hóa tốt và không xuất hiện dấu hiệu overfitting.

Chương 3. Lab 3 – Hugging Face Transformers

3.1. Giới thiệu và mục tiêu

Lab 3 tập trung vào việc làm quen với hệ sinh thái Hugging Face Transformers trong xử lý ngôn ngữ tự nhiên (NLP). Nội dung của lab được triển khai theo hai hướng thường gặp trong thực tế. Hướng thứ nhất là sử dụng mô hình đã được huấn luyện sẵn (pretrained model)

để thực hiện suy luận nhanh cho bài toán phân tích cảm xúc. Hướng thứ hai là fine-tuning một mô hình pretrained trên dữ liệu cụ thể để giải quyết bài toán phân loại văn bản nhị phân. Việc triển khai song song hai hướng này giúp nhóm hiểu rõ sự khác biệt giữa "dùng model có sẵn để dự đoán" và "huấn luyện lại model để phù hợp bài toán".

Mục tiêu chính của Lab 3 là nắm được quy trình làm việc tiêu chuẩn với Transformers, bao gồm cài đặt các thư viện cần thiết, tải model và tokenizer từ Hugging Face Hub, thực hiện tokenize dữ liệu đầu vào và hiểu rõ các thành phần mà mô hình sử dụng khi suy luận. Bên cạnh đó, nhóm cần thực hiện inference sentiment analysis, lưu kết quả để phục vụ báo cáo, đồng thời xây dựng pipeline fine-tuning bằng Trainer, đánh giá mô hình bằng các chỉ số cơ bản như Accuracy và F1-score, và kiểm tra khả năng dự đoán trên tập test trong trường hợp đề bài yêu cầu xuất kết quả theo định dạng submission.

3.2. Môi trường và thư viện sử dụng

Nhóm sử dụng môi trường Python với ba thư viện chính là transformers, datasets và evaluate. Thư viện transformers đóng vai trò trung tâm vì cung cấp các lớp để tải mô hình và tokenizer, cũng như cung cấp Trainer hỗ trợ quá trình huấn luyện và đánh giá theo quy trình chuẩn. Thư viện datasets được dùng để đọc dữ liệu từ các tệp CSV, tạo tập train/validation, và thực hiện tokenize theo batch thông qua hàm map giúp tăng tốc xử lý. Thư viện evaluate được sử dụng để tính toán các chỉ số đánh giá, đảm bảo cách tính metrics thống nhất và dễ so sánh giữa các lần chạy. Ngoài ra, nhóm tổ chức thư mục src/common/ để gom các tiện ích dùng chung. Thư mục này hỗ trợ đọc cấu hình YAML nhằm tách phần cấu hình khỏi phần code, giúp thay đổi model hoặc đường dẫn input/output mà không cần sửa logic chương trình. Đồng thời, src/common/ còn đảm nhiệm việc tạo thư mục output tự động, ghi kết quả inference theo định dạng JSONL để tiện theo dõi theo từng dòng, và thiết lập seed để tăng khả năng tái lập kết quả khi chạy lại trong các điều kiện tương tự.

3.3. Bài 1 – Sentiment Analysis với Hugging Face (Pretrained Model)

3.3.1. Mục tiêu bài 1

Mục tiêu của bài 1 là hoàn thành một pipeline inference cơ bản bằng pretrained model cho bài toán phân tích cảm xúc. Nhóm cần thực hiện cài đặt transformers, tải mô hình sentiment từ Hugging Face Hub, tokenizer một câu mẫu để quan sát dữ liệu đầu vào mà

model nhận, sau đó chạy dự đoán để thu được nhãn cảm xúc và độ tin cậy. Cuối cùng, kết quả cần được lưu ra output để phục vụ việc tổng hợp và trình bày trong báo cáo.

3.3.2. Các bước thực hiện

Trước hết, nhóm cài đặt thư viện transformers để có thể sử dụng các thành phần quan trọng như AutoTokenizer, AutoModelForSequenceClassification và pipeline. Sau khi môi trường sẵn sàng, nhóm chọn pretrained model distilbert-base-uncased-finetuned-sst-2-english. Đây là mô hình DistilBERT đã fine-tune trên SST-2, phù hợp cho phân loại cảm xúc nhị phân, vì vậy kết quả dự đoán sẽ trả về nhãn POSITIVE hoặc NEGATIVE kèm theo score thể hiện mức độ tin cậy.

Để hiểu rõ cơ chế đầu vào của Transformers, nhóm xây dựng file src/task_1_sentiment/tokenize_demo.py nhằm minh hoạ quá trình tokenize. Tokenizer sẽ biến văn bản thành tokens theo vocabulary của model, sau đó mã hoá thành input_ids là dãy số nguyên. Đồng thời attention_mask được tạo ra để mô hình phân biệt phần dữ liệu thật và phần padding khi cần. Việc quan sát tokens, input_ids và attention_mask giúp nhóm nắm rõ cách dữ liệu tự nhiên được chuyển thành dạng số trước khi đưa vào mạng Transformer. Ở bước inference, nhóm triển khai hai cách để vừa đảm bảo tính thực hành nhanh vừa đảm bảo hiểu sâu quy trình. Cách thứ nhất dùng pipeline("sentiment-analysis") để thực hiện dự đoán trực tiếp, nhanh và ít code, trả về label và score. Cách thứ hai thực hiện forward trực tiếp qua AutoModelForSequenceClassification để lấy logits, sau đó áp dụng softmax để chuyển logits thành xác suất và chọn nhãn có xác suất cao nhất. Việc làm theo cả hai cách giúp nhóm đối chiếu kết quả và hiểu rõ pipeline bên trong của mô hình phân loại.

Sau khi dự đoán, nhóm ghi kết quả vào outputs/task_1_inference/samples.jsonl. Định dạng JSONL giúp lưu nhiều mẫu dự đoán theo từng dòng, dễ kiểm tra và thuận tiện khi cần tổng hợp. Các tham số cấu hình như tên model, danh sách câu test và đường dẫn output được quản lý trong configs/task_1_sentiment.yaml, giúp việc thay đổi dữ liệu hoặc model trở nên linh hoạt mà không ảnh hưởng đến code.

3.3.3. Kết quả đạt được

Nhóm hoàn thành đầy đủ quy trình pretrained inference, bao gồm tải model và tokenizer, minh hoạ tokenize với tokens, input_ids, attention_mask, chạy dự đoán sentiment theo hai cách khác nhau và lưu kết quả ra JSONL. Kết quả cho thấy mô hình trả về nhãn cảm

xúc và score hợp lý với các câu thử nghiệm, đồng thời nhóm hiểu rõ cách dữ liệu được chuẩn hoá trước khi đi vào mô hình.

3.4. Bài 2 – Fine-tuning Pretrained Model cho Binary Text Classification

3.4.1. Mục tiêu bài 2

Mục tiêu của bài 2 là xây dựng quy trình fine-tuning một mô hình pretrained cho bài toán phân loại văn bản nhị phân. Nhóm cần thực hiện đầy đủ các bước gồm nạp dữ liệu, tiền xử lý dữ liệu để làm sạch và chuẩn hoá nhãn, tokenize dữ liệu theo đúng tokenizer của mô hình, huấn luyện bằng Trainer với TrainingArguments phù hợp, đánh giá mô hình bằng Accuracy và F1-score, và cuối cùng có thể dự đoán trên tập test theo yêu cầu của đề bài nếu cần tạo file submission.

3.4.2. Tiền xử lý dữ liệu

Trước khi huấn luyện, nhóm chú trọng làm sạch dữ liệu để tránh lỗi khi tokenize và hạn chế nhiễu trong quá trình học. Vì vậy nhóm viết hai script trong src/task_2_finetune/script/. Script prepare_dataset.py xử lý dữ liệu train bằng cách loại bỏ các dòng text rỗng hoặc null, chuẩn hoá khoảng trắng, đồng thời đưa nhãn về dạng nhị phân 0/1 để mô hình học nhất quán. Một điểm quan trọng trong bước này là xử lý dữ liệu trùng lặp và trường hợp xung đột nhãn, tức cùng một nội dung nhưng bị gán nhãn khác nhau. Nếu giữ lại các mẫu xung đột, mô hình sẽ học không ổn định và metric giảm, vì vậy nhóm cần loại bỏ hoặc xử lý phù hợp trước khi train. Sau khi hoàn tất, script xuất file train_clean.csv theo cấu trúc text,label để sẵn sàng cho datasets load. Script prepare_test.py làm sạch dữ liệu test theo hướng tương tự, đồng thời chuẩn hoá các giá trị null-like để đảm bảo pipeline dự đoán không gặp lỗi, sau đó xuất test_clean.csv để dùng ở bước inference sau fine-tune.

3.4.3. Huấn luyện và đánh giá bằng Trainer

Nhóm triển khai fine-tuning chủ yếu trong notebooks/02_data_validations_eda.ipynb để thuận tiện kiểm tra dữ liệu và theo dõi quá trình huấn luyện. Trước hết, nhóm dùng datasets để đọc train_clean.csv và tạo train/validation split nhằm đánh giá mô hình trong quá trình train. Tiếp theo, nhóm tải pretrained checkpoint distilbert-base-uncased và tokenizer tương ứng. Mô hình được cấu hình cho bài toán phân loại nhị phân với 2 nhãn.

Sau đó nhóm tokenize dữ liệu theo batch để tạo input_ids và attention_mask. Việc chọn max_length được dựa trên khảo sát độ dài câu để đảm bảo không cắt quá nhiều thông tin nhưng vẫn tối ưu thời gian và bộ nhớ. Khi dữ liệu đã ở định dạng phù hợp, nhóm thiết lập TrainingArguments để điều khiển các tham số quan trọng như số epoch, batch size, learning rate, cách đánh giá theo từng giai đoạn, logging và chiến lược lưu checkpoint. Từ các cấu hình này, nhóm sử dụng Trainer để huấn luyện mô hình theo quy trình chuẩn.

Ở bước đánh giá, nhóm sử dụng evaluate để tính Accuracy và F1-score trên validation. Accuracy phản ánh tỷ lệ dự đoán đúng tổng thể, trong khi F1-score đặc biệt hữu ích khi dữ liệu có thể mất cân bằng nhãn vì nó cân bằng giữa Precision và Recall. Sau khi hoàn tất huấn luyện và đánh giá, nhóm có thể tokenize tập test tương tự train và chạy dự đoán để xuất kết quả theo định dạng yêu cầu của bài, phục vụ mục tiêu kiểm tra hoặc tạo submission nếu đề bài yêu cầu.

3.4.4. Kết quả đạt được

Nhóm xây dựng được pipeline fine-tune đầy đủ từ khâu chuẩn hoá dữ liệu bằng script, load dữ liệu bằng datasets, tokenize bằng tokenizer của mô hình, huấn luyện bằng Trainer và đánh giá bằng Accuracy/F1. Quy trình này thể hiện cách triển khai chuẩn của Transformers trong bài toán phân loại văn bản nhị phân và có thể tái sử dụng cho các bộ dữ liệu khác với thay đổi tối thiểu ở phần cấu hình và preprocessing.

3.5. Kết luận

Thông qua Lab 3, nhóm nắm được hai cách áp dụng Hugging Face trong NLP. Bài 1 giúp nhóm hiểu rõ quy trình inference nhanh với pretrained model và làm rõ cơ chế tokenize của Transformers thông qua các thành phần tokens, input_ids và attention_mask. Bài 2 giúp nhóm thực hành quy trình huấn luyện lại mô hình bằng Trainer trên dữ liệu đã chuẩn hoá, đồng thời đánh giá mô hình bằng các metrics cơ bản như Accuracy và F1-score. Kết quả của lab tạo nền tảng vững chắc để nhóm tiếp tục triển khai các bài toán NLP nâng cao hơn trong các phần tiếp theo.

Chương 4. Tổng kết

4.1 Nhận xét và đánh giá

Thông qua chuỗi bài lab và practice, sinh viên đã có cơ hội tiếp cận trực tiếp với các thao tác cơ bản trong Deep Learning, từ chuẩn bị dữ liệu, xây dựng mô hình đến huấn luyện

và đánh giá kết quả. Các bài thực hành được thiết kế theo hướng từng bước, giúp sinh viên làm quen với quy trình triển khai một mô hình học sâu trong thực tế.

Bên cạnh đó, các thành viên trong nhóm đã phối hợp tốt, phân công nhiệm vụ rõ ràng và hoàn thành đầy đủ các bài được giao đúng tiến độ. Quá trình làm việc nhóm giúp sinh viên rèn luyện kỹ năng trao đổi, thảo luận và hỗ trợ lẫn nhau trong việc giải quyết các vấn đề kỹ thuật phát sinh. Nhìn chung, nhóm đã thực hiện nghiêm túc các yêu cầu của bài lab, đạt được mục tiêu đề ra và tạo nền tảng thuận lợi cho các học phần chuyên sâu hơn trong tương lai.

4.2. Kiến thức và kỹ năng thu được

Thông qua học phần và quá trình thực hiện bài tiểu luận, sinh viên đã tiếp thu được các kiến thức nền tảng về Deep Learning, đặc biệt là nguyên lý hoạt động của mạng nơ-ron sâu và cơ chế backpropagation trong việc cập nhật trọng số nhằm tối ưu mô hình. Việc hiểu rõ quá trình lan truyền thuận và lan truyền ngược giúp sinh viên nắm được bản chất của quá trình huấn luyện thay vì chỉ dừng lại ở mức sử dụng mô hình có sẵn.

Bên cạnh đó, sinh viên đã làm quen với các kiến trúc mạng phổ biến như CNN cho bài toán xử lý ảnh, RNN/LSTM cho dữ liệu chuỗi và Transformer với cơ chế attention hiện đại. Sinh viên cũng nắm được quy trình chuẩn bị dữ liệu, xây dựng Dataset và Data Loader, cũng như cách áp dụng pretrained model để tận dụng tri thức có sẵn thông qua transfer learning.

Ngoài ra, sinh viên đã rèn luyện kỹ năng xây dựng và tùy chỉnh mô hình Deep Learning, bao gồm thiết kế kiến trúc mạng, freeze các tầng mô hình, thay đổi số node đầu ra cho phù hợp với từng bài toán cụ thể. Việc sử dụng TensorBoard giúp sinh viên theo dõi và đánh giá quá trình huấn luyện một cách trực quan, từ đó nâng cao khả năng phân tích và cải thiện hiệu quả mô hình. Những kiến thức và kỹ năng này tạo nền tảng quan trọng cho việc học tập và nghiên cứu sâu hơn trong lĩnh vực trí tuệ nhân tạo.

4.3. Hướng phát triển trong tương lai

Trong thời gian tới, sinh viên sẽ tiếp tục vận dụng các kiến thức đã học về Deep Learning vào các học phần chuyên sâu tiếp theo. Cụ thể, các hiểu biết về mô hình và cơ chế huấn luyện mạng nơ-ron sẽ được áp dụng trong Xử lý ảnh và Thị giác máy tính, Xử lý ngôn ngữ tự nhiên và Phân tích dữ liệu chuỗi thời gian. Thông qua đó, sinh viên có thể nâng cao kỹ năng giải quyết các bài toán thực tế, đồng thời xây dựng nền tảng vững chắc cho quá trình học tập và nghiên cứu trong lĩnh vực trí tuệ nhân tạo.