ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

Tel. (84-511) 3736949, Fax. (84-511) 3842771

Website: itf.dut.udn.vn, E-mail: cntt@dut.udn.vn

BÁO CÁO ĐỒ ÁN

MÔN HỌC XỬ LÝ ẢNH SỐ

ĐỀ TÀI

HỆ THỐNG NHẬN DIỆN CÁC LOÀI HOA

SINH VIÊN THỰC HIỆN:

Nguyễn Văn Mạnh	LỚP: 20T1	NHÓM: 20.10
Nguyễn Văn Hoàng Phúc	LỚP: 20T1	NHÓM: 20.10
Nguyễn Công Cường	LỚP: 20T1	NHÓM: 20.10

GIẢNG VIÊN HƯỚNG DẪN: TS. Huỳnh Hữu Hưng

Đà Nẵng 05-2024

TÓM TẮT ĐỒ ÁN

Đề tài nhận diện các loài hoa là một trong những lĩnh vực quan trọng của thị giác máy tính và trí tuệ nhân tạo. Mục tiêu của đề tài này là xây dựng mô hình tự động có khả năng phân loại loài hoa từ hình ảnh, giúp giảm bớt công sức và thời gian cho việc nhận diện và phân loại các loài hoa trong các ứng dụng thực tiễn. Chính vì vậy nhóm chúng tôi chọn đề tài "Hệ thống nhận diện các loài hoa" nhằm tạo ra một hệ thống giúp người dùng dễ dàng nhận biết được các loài hoa mà người đó có thể bắt gặp nhưng lại không biết tên.

Đây là đề tài phục vụ cho môn học "Xử lý ảnh số" nên chúng tôi thực hành thử nghiệm trên 2 pha. Pha thứ nhất, chúng tôi tìm hiểu, sử dụng và xây dựng các kiến trúc CNN từ các kiến trúc ra đời trước đến các kiến trúc xuất hiện sau này. Pha thứ hai, chúng tôi phát triển một kiến trúc tân tiến hơn là Vision Transformers, xây dựng và custom để đạt hiệu quả tốt nhất có thể. Các kết quả ở pha một là tốt so với mong đợi, các model tự xây dựng đạt kết quả cao nhất là gần 90% trên tập dữ liệu Oxford 102 Flower (lấy mẫu 15 loài hoa). Đối với pha hai cũng đạt được những cải tiến đáng kể so với kiến trúc ban đầu tự xây dựng theo bài báo gốc, tuy nhiên kết quả chưa hiệu quả so với các model ở pha thứ nhất. Chúng tôi có kết hợp các model có hiệu suất tốt nhất vào một website, giúp người dùng có thể tương tác trực tiếp và thực hiện nhận diện loài hoa mong muốn.

Bảng 1.1. Bảng phân công nhiệm vụ

Sinh viên	Nhiệm vụ	Hoàn thành
Nguyễn Văn Mạnh	Phân công công việc, đảm bảo tiến độ đồ án	x
	Tìm kiếm và chọn bộ dữ liệu	x
	Transfer learning EfficientNetB6	x
	Đề xuất tham số cho mymodel 1	x
	Xây dựng và cải thiện Vision Transformers	x
	Hỗ trợ xây dựng Server	x
	Ghép nối và thử nghiệm sản phẩm	x
	Viết báo cáo	x
	Làm slide	x
Nguyễn Văn Hoàng Phúc	Tìm kiếm và chọn bộ dữ liệu	x
	Xây dựng và cải thiện model LeNet-5	x
	Xây dựng và cải thiện model GoogLeNet	x
	Đề xuất tham số cho mymodel 2	x
	Xây dựng và cải thiện Vision Transformers	x
	Hỗ trợ xây dựng Server	x
	Ghép nối và thử nghiệm sản phẩm	x
	Viết báo cáo	x
	Làm slide	x
Nguyễn Công Cường	Tìm kiếm và chọn bộ dữ liệu	x
	Tiền xử lý dữ liệu	x
	Xây dựng và cải thiện model AlexNet	x
	Đề xuất tham số cho mymodel 3	x
	Xây dựng và cải thiện Vision Transformers	x
	Hỗ trợ xây dựng Server	x
	Ghép nối và thử nghiệm sản phẩm	x
	Viết báo cáo	x
	Làm slide	x

MỤC LỤC

MỤC LỤC HÌNH ẢNH

MỤC LỤC BẢNG

Chương 1. Giới thiệu

1.1. Thực trạng sản phẩm

Đề tài nhận diện các loài hoa là một trong những lĩnh vực quan trọng của thị giác máy tính và trí tuệ nhân tạo. Mục tiêu của đề tài này là xây dựng mô hình tự động có khả năng phân loại loài hoa từ hình ảnh, giúp giảm bớt công sức và thời gian cho việc nhận diện và phân loại các loài hoa trong các ứng dụng thực tiễn.

Hiện nay, trên thị trường có nhiều ứng dụng và hệ thống nhận diện loài hoa được phát triển và sử dụng rộng rãi. Những sản phẩm này thường sử dụng công nghệ trí tuệ nhân tạo (AI) và học máy (machine learning) để phân tích hình ảnh hoa và đưa ra kết quả nhận diện. Dưới đây là một số thực trạng chính của các sản phẩm hiện tại:

- Sự phong phú về ứng dụng: Xuất hiện trên nhiều nền tảng khác nhau, từ điện thoại di động đến các hệ thống máy tính chuyên dụng. Một số ứng dụng phổ biến như PlantSnap, PictureThis, và FlowerChecker,...
- Độ chính xác và tốc độ nhận diện: Độ chính xác khá cao nhờ vào việc sử dụng các thuật toán học sâu (deep learning). Tuy nhiên, độ chính xác có thể khác nhau tùy thuộc vào chất lượng hình ảnh và điều kiện ánh sáng khi chụp ảnh hoa.
- Cơ sở dữ liệu đa dạng: Bao gồm hàng ngàn loài hoa khác nhau từ khắp nơi trên thế giới và liên tục được cập nhật và mở rộng để cải thiện khả năng nhận diện.
- Thân thiện với người dùng: Giao diện thiết kế đơn giản và dễ sử dụng. Người dùng chỉ cần chụp ảnh hoa hoặc tải lên một hình ảnh có sẵn, hệ thống sẽ tự động phân tích và cung cấp thông tin chi tiết về loài hoa đó.
- Hạn chế và thách thức: Khó khăn trong việc nhận diện các loài hoa có hình dáng và màu sắc tương tự nhau, hay bị ảnh hưởng bởi các yếu tố ngoại cảnh như ánh sáng, góc chụp. Ngoài ra, việc cập nhật và duy trì cơ sở dữ liệu đòi hỏi nhiều nguồn lực và công sức.

Nhìn chung, các sản phẩm nhận diện loài hoa đang ngày càng hoàn thiện và trở thành công cụ hữu ích cho nhiều đối tượng người dùng. Tuy nhiên, để đạt được hiệu quả cao nhất, cần tiếp tục nghiên cứu và phát triển nhằm khắc phục các hạn chế hiện tại và đáp ứng tốt hơn nhu cầu của người dùng.

1.2. Các vấn đề cần giải quyết

Hiện nay các ứng dụng nhận diện loài hoa thường gặp các vấn đề sau:

- Độ chính xác trong điều kiện thực tế.
- Phân biệt các loài hoa tương tự nhau.
- Cập nhật và mở rộng cơ sở dữ liệu.
- Tính khả dụng và tốc độ xử lý.

Giải quyết các vấn đề này sẽ giúp cải thiện đáng kể chất lượng và hiệu quả của các hệ thống nhận diện loài hoa, đáp ứng tốt hơn nhu cầu của người dùng và góp phần thúc đẩy sự phát triển của công nghệ nhận diện hình ảnh trong tương lai.

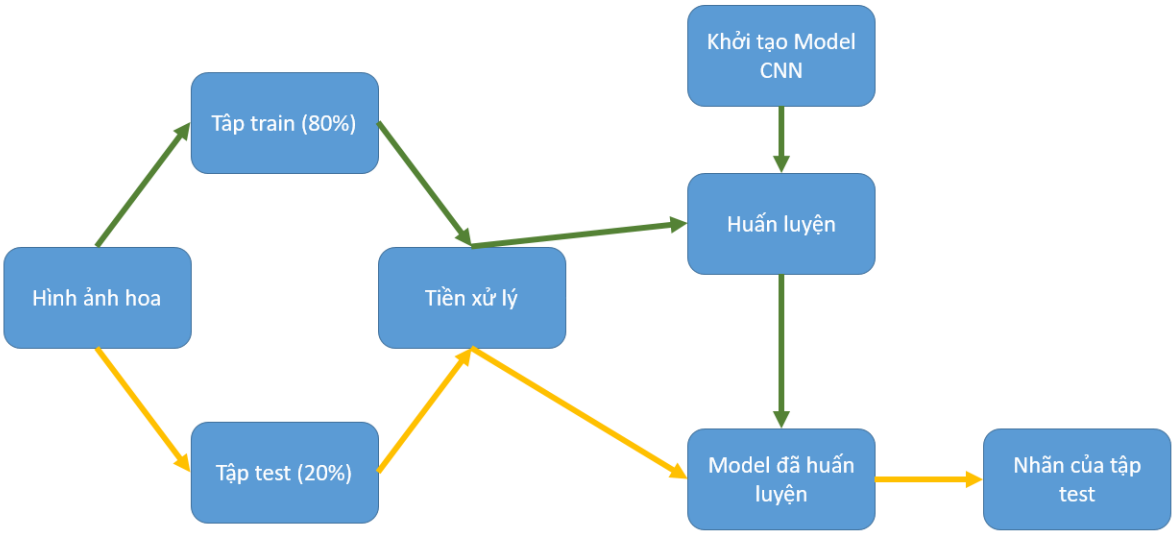
1.3. Đề xuất giải pháp tổng quan

Để giải quyết các vấn đề đã nêu và nâng cao hiệu quả của hệ thống nhận diện loài hoa, cần có những giải pháp tổng quan, bao gồm các khía cạnh công nghệ, cơ sở dữ liệu. Ở báo cáo này chúng tôi chủ yếu tập trung vào khía cạnh công nghệ, dưới đây là những đề xuất giải pháp chính:

- Sử dụng các thuật toán CNN cơ bản và các biến thể;
- Sử dụng kĩ thuật học chuyển giao (transfer learning);
- Sử dụng mô hình vision transformer, transformer kĩ thuật bên NLP áp dụng vào computer vision.

Chương 2. Giải pháp

2.1. Sử dụng các thuật toán CNN cơ bản và các biến thể



Hình 2.1. Pipeline model CNN và các biến thể

2.1.1. Giải pháp tiền xử lý:

a) Tăng cường dữ liệu

Tăng cường dữ liệu chỉ thực hiện trên tập train, tuân theo nguyên tắt dữ liệu, tập test là tập cố định không thay đổi, bị ẩn đi cho đến khi thực hiện dự đoán.

Được thực hiện với class ImageDataGenerator có các thuộc tính sau:

- *zoom_range:* thực hiện zoom ngẫu nhiên trong một phạm vi nào đó;
- *width_shift_range:* dịch theo chiều ngang ngẫu nhiên trong một phạm vi nào đó;
- *height_shift_range:* dịch ảnh theo chiều dọc trong một phạm vi nào đó;
- *brightness_range*: tăng cường độ sáng của ảnh trong một phạm vi nào đó;
- *vertical_flip:* lật ảnh ngẫu nhiên theo chiều dọc;
- *rotation_range:* xoay ảnh góc tối đa là 45 độ;
- *shear_range:* Làm méo ảnh.

b) Chuẩn hoá dữ liệu:

Cả tập train và test đều được chuẩn hoá trước khi đưa qua model huấn luyện và nhận diện. Bước chuẩn hoá là thực hiện chuyển các giá trị màu sắc từ 0-255 thành số thực từ 0-1 bằng cách chia tất cả các giá trị cho 255. Điều này giúp model học được những đặc trưng gần nhau hơn, ít chi phối bởi các đặc trưng quá khác biệt.

2.1.2. Giới thiệu về CNN model:

Convolutional Neural Networks (CNN) là một loại mạng nơ-ron nhân tạo đặc biệt mạnh mẽ trong việc xử lý dữ liệu dạng lưới, chẳng hạn như hình ảnh. CNN đã trở thành tiêu chuẩn vàng trong các nhiệm vụ liên quan đến thị giác máy tính.

a) Cấu trúc chung của CNN:

Thông thường trong kiến trúc của một mô hình CNN sẽ bao gồm các lớp sau:

- Convolutional Layers (Lớp tích chập):

Mỗi lớp tích chập sử dụng các bộ lọc (filter) để quét qua toàn bộ hình ảnh và phát hiện ra các đặc trưng như cạnh, góc, và các mẫu phức tạp hơn trong những lớp sau.

- Pooling Layers (Lớp lấy mẫu):

Các lớp này giảm kích thước không gian của dữ liệu (hình ảnh), giúp giảm số lượng tham số và tính toán trong mạng. Phổ biến nhất là Max Pooling và Average Pooling.

- Fully Connected Layers (Lớp kết nối đầy đủ):

Sau các lớp tích chập và lấy mẫu, dữ liệu sẽ được làm phẳng (flatten) và đưa qua các lớp kết nối đầy đủ. Lớp này tương tự như mạng nơ-ron thông thường và thực hiện các tác vụ phân loại dựa trên các đặc trưng đã trích xuất.

- Activation Functions (Hàm kích hoạt):

Thường sử dụng hàm ReLU (Rectified Linear Unit) sau mỗi lớp tích chập để giới thiệu tính phi tuyến vào mô hình.

b) Các mô hình CNN phổ biến:

Các mô hình phổ biến gồm: LeNet-5, AlexNet, VGGNet, GoogLeNet (Inception), ResNet (Residual Networks)…

2.1.3. Các kiến trúc model CNN tái xây dựng sử dụng:

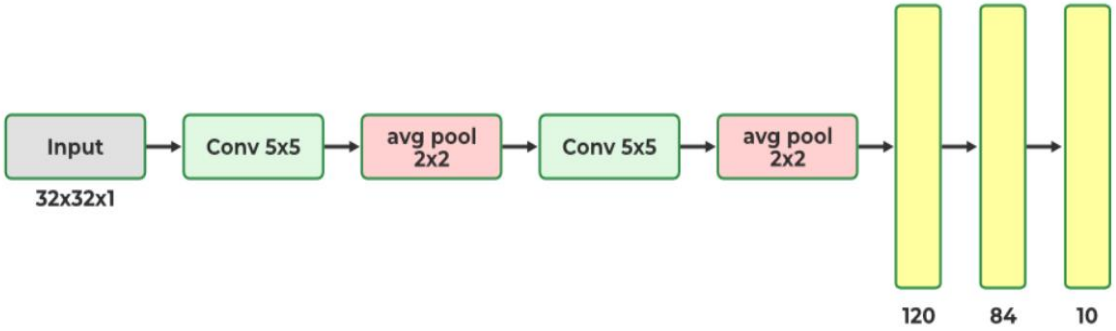
Trong báo cáo này chúng tôi tái xây dựng một số model dựa trên kiến trúc được chia sẻ trong bài báo gốc của nó và khởi tạo một số model tự xây dựng sau:

- *LeNet-5:* Một trong những mô hình CNN đầu tiên, được Yann LeCun giới thiệu vào những năm 1990. LeNet-5 được sử dụng chủ yếu cho việc nhận diện chữ số viết tay (MNIST dataset).
- *AlexNet:* Được phát triển bởi Alex Krizhevsky và nhóm nghiên cứu của ông vào năm 2012, AlexNet đã chiến thắng cuộc thi ImageNet và mang lại sự chú ý lớn cho CNN. Mô hình này có 8 lớp học sâu và sử dụng ReLU, Dropout, và các kỹ thuật tăng cường dữ liệu (data augmentation).
- *GoogLeNet:* Được phát triển bởi Google, mô hình này sử dụng các khối Inception để tăng cường độ sâu và rộng của mạng mà không làm tăng số lượng

tham số quá mức. GoogLeNet đã chiến thắng cuộc thi ImageNet 2014.

- *CNN tự xây dựng:* Các model CNN cơ bản tương tự với các lớp của model LeNet, thực hiện tăng số lượng lớp, tuỳ chỉnh các tham số, bổ sung lớp chuẩn hoá và dropout nhằm rút ra kết luận về các tham số ảnh hưởng đến hiệu suất của các kiến trúc.

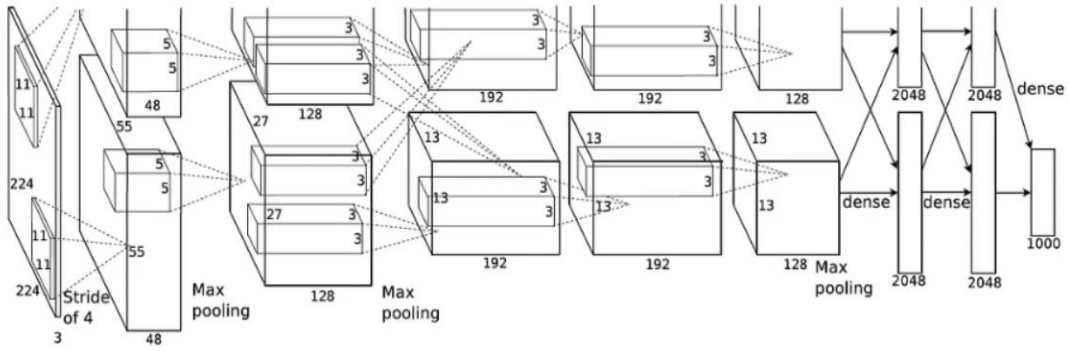
2.1.4. Kiến trúc model LeNet-5 tái xây dựng:



Hình 2.2 Kiến trúc model LeNet-5 gốc

Model LeNet-5 bao gồm 2 lớp CNN, cuối mỗi lớp CNN sẽ có các lớp Average Pooling để giảm số đặc trưng, cuối cùng là 3 lớp Dense với các kernel nhỏ dần và output là số chiều tương ứng với số loài hoa.

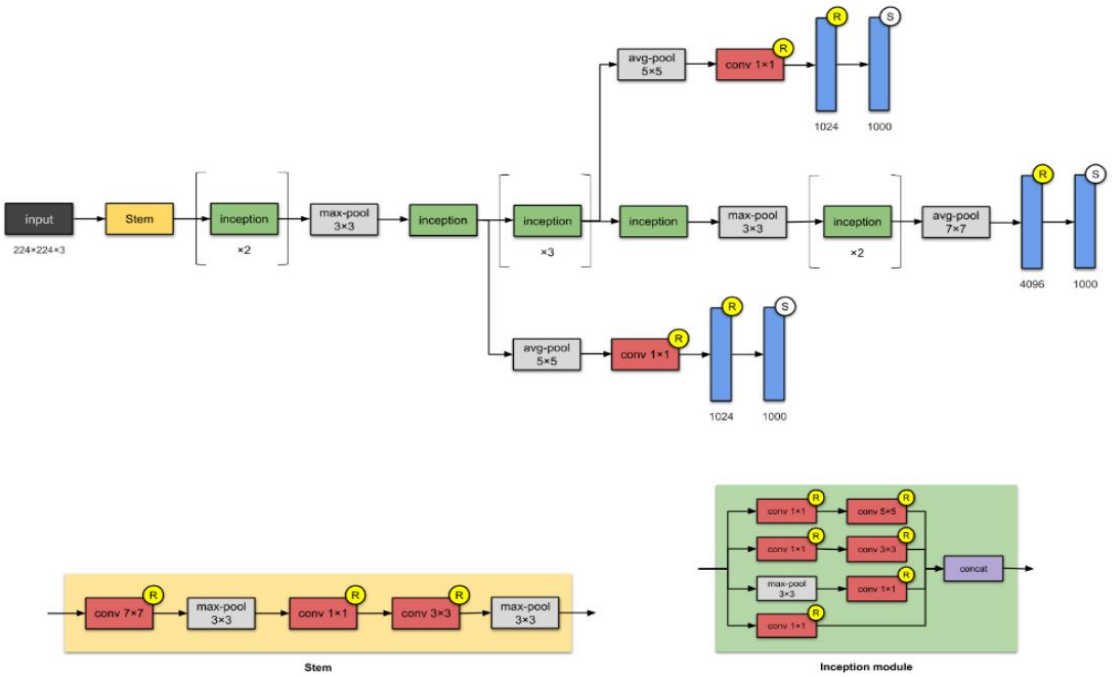
2.1.5. Kiến trúc model AlexNet tái xây dựng:



Hình 2.3 Kiến trúc model AlexNet gốc

Kiến trúc AlexNet bao gồm 5 lớp Conv, với 2 lớp đầu thì thực hiện max pooling sau mỗi lớp, 3 lớp sau sẽ thực hiện liên tiếp mà không có các lớp max pooling xen giữa, và kết thúc bằng một lớp max pooling ở cuối 3 lớp này. Đoạn cuối mô hình vẫn là Flatten và các lớp fully connected có cùng số chiều là 4096, và lớp fully connected cuối có số chiều là số loài hoa cần nhận diện.

2.1.6. Kiến trúc model GoogLeNet tái xây dựng:

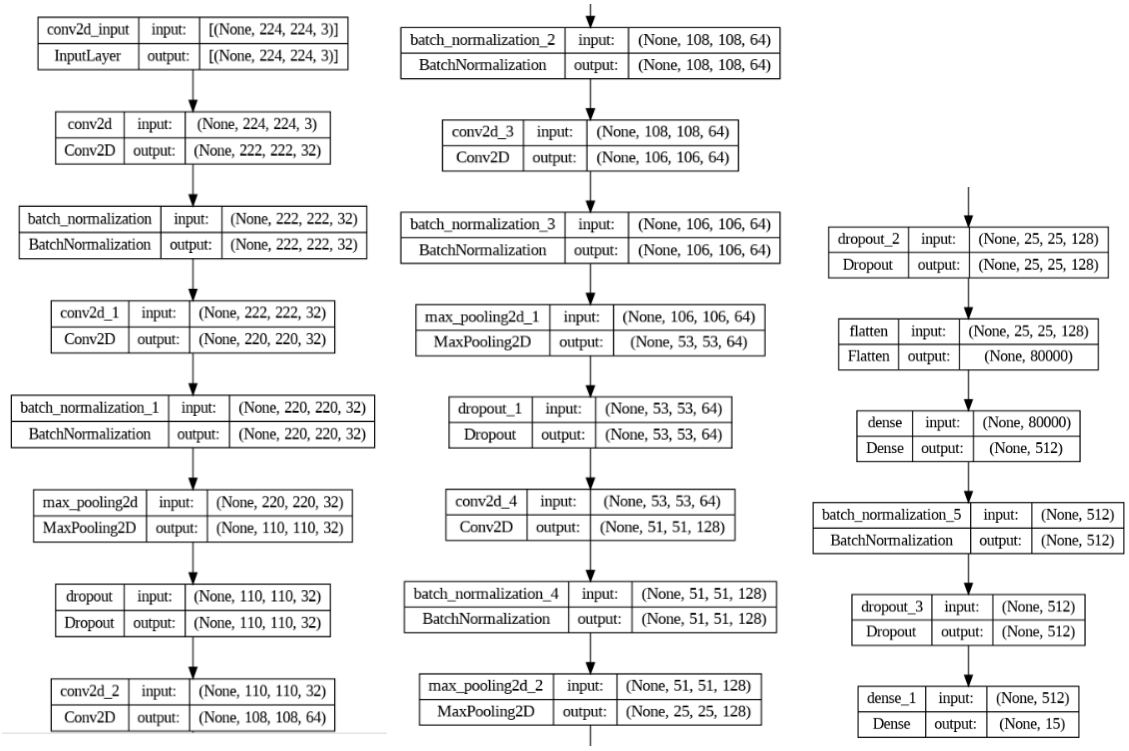


Hình 2.4 Kiến trúc model GoogLeNet gốc

Kiến trúc model GoogLeNet đặc trưng bởi các khối inception module và không phải là cấu trúc một hàng mà có thực hiện rẽ thành 3 nhánh để học các đặc trưng ở mức thấp, mức trung và mức cao.

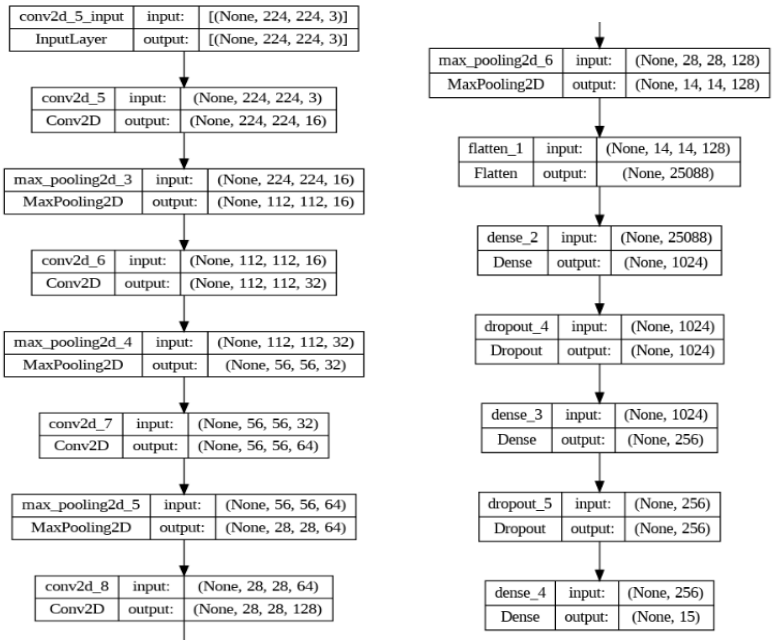
Các khối inception sẽ được chia thành 4 nhánh tương ứng với 4 kernel khác nhau là 5*5, 3*3, 1*1 có max pooling và 1*1 không có max pooling. Từ đó đặc trưng tại mỗi bước nhỏ sẽ được học trên vùng xa nhau (5*5) đến những vùng nhỏ nhất (1*1). Đó chính là ưu điểm của kiến trúc GoogLeNet.

2.1.7. Kiến trúc model tự xây dựng:



Hình 2.5 Kiến trúc my model 1 tự xây dựng

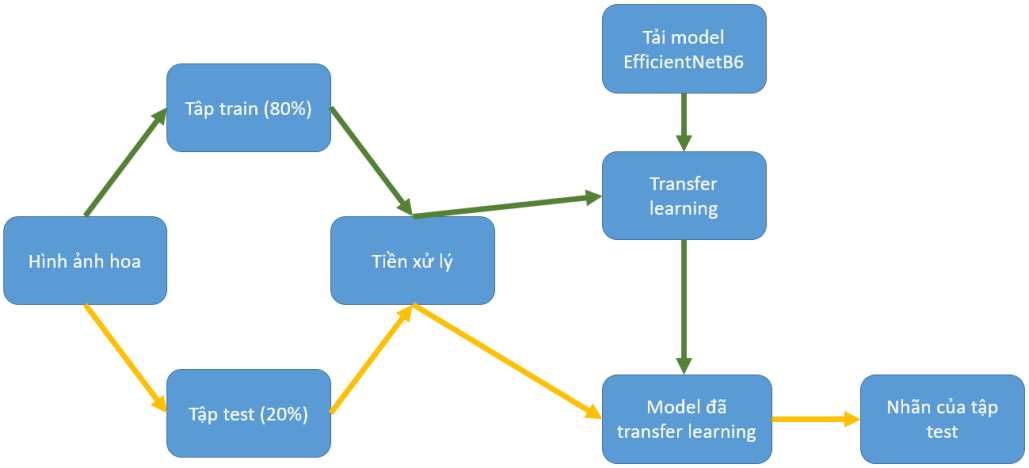
Mục đích xây dựng model: Model sử dụng 5 lớp conv, với số lượng tham số lên đến hơn 51 triệu tham số. Chúng tôi muốn xem xét có phải số lượng tham số quyết định chính đến kết quả của mô hình hay không.



Hình 2.6 Kiến trúc my model 2 tự xây dựng

Mục đích: Thay đổi kiến trúc thành 4 lớp conv, thay đổi hàm kích hoạt.

2.2. Sử dụng kĩ thuật học chuyển giao (transfer learning)

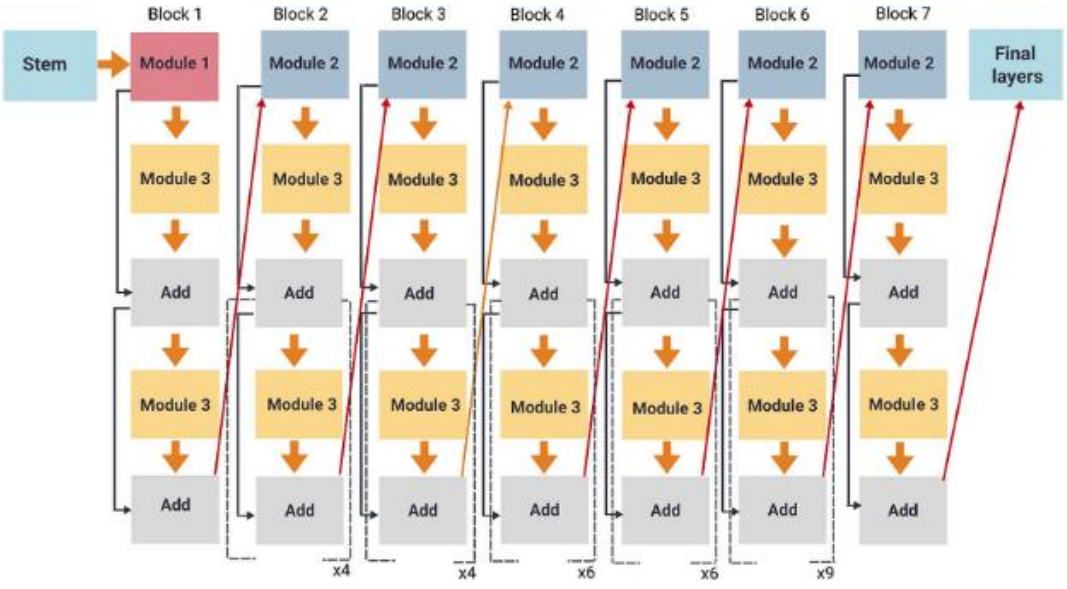


Hình 2.7 Pipeline của giải pháp sử dụng kĩ thuật học chuyển giao

2.2.1. Giới thiệu EfficientNetB6 model

EfficientNetB6 là một trong những mô hình thuộc dòng EfficientNet, được phát triển bởi nhóm nghiên cứu của Google nhằm tối ưu hóa hiệu quả và hiệu suất của mạng neural. EfficientNet sử dụng một phương pháp gọi là "compound scaling" để đồng thời mở rộng độ sâu, độ rộng và độ phân giải của mạng, nhằm đạt được sự cân bằng tốt hơn giữa hiệu suất và độ chính xác.

a) Kiến trúc của EfficientNet



Hình 2.8 Kiến trúc của EfficientNetB6

Mô hình EfficientNetB6 dựa trên kiến trúc EfficientNet, với sự cải tiến và tối ưu hóa thông qua việc áp dụng kỹ thuật Neural Architecture Search (NAS) - một

phương pháp tự động tìm kiếm kiến trúc mạng neural tối ưu. Điều này giúp mô hình đạt được hiệu suất cao hơn so với các mô hình truyền thống mà không tăng đáng kể số lượng tham số.

Kiến trúc EfficientNet bao gồm các khối MBConv (Mobile Inverted Bottleneck Convolution) được mở rộng theo chiều sâu *(depth)*, chiều rộng *(width)*, và độ phân giải *(resolution)*. Công thức compound scaling được sử dụng để tăng cường hiệu suất của mô hình bằng cách mở rộng các tham số này một cách cân đối. Đối với EfficientNetB6, hệ số mở rộng được điều chỉnh để đạt được độ chính xác cao trong khi vẫn duy trì tính toán hiệu quả.

b) Ưu điểm của EfficientNetB6

- Hiệu quả tính toán cao: EfficientNetB6 sử dụng ít tham số hơn so với nhiều mô hình khác nhưng vẫn đạt được độ chính xác cao, nhờ vào kiến trúc được tối ưu hóa qua phương pháp compound scaling.
- Khả năng tổng quát hóa tốt: Với việc mở rộng đồng thời độ sâu, độ rộng và độ phân giải, EfficientNetB6 có khả năng học được nhiều đặc trưng phức tạp từ dữ liệu, giúp cải thiện khả năng tổng quát hóa của mô hình.
- Tối ưu tài nguyên: Nhờ vào việc tối ưu hóa kiến trúc, EfficientNetB6 yêu cầu ít tài nguyên hơn so với các mô hình truyền thống như ResNet hay Inception, điều này giúp tiết kiệm chi phí và thời gian huấn luyện.

c) Ứng dụng trong học chuyển giao

EfficientNetB6 được sử dụng rộng rãi trong học chuyển giao (transfer learning) nhờ vào khả năng tổng quát hóa và hiệu quả tính toán. Trong học chuyển giao, mô hình EfficientNetB6 có thể được sử dụng như một mô hình cơ sở (pre-trained model) để trích xuất đặc trưng từ dữ liệu mới. Sau đó, các lớp cuối của mô hình có thể được thay thế và huấn luyện lại để phù hợp với nhiệm vụ cụ thể.

Khi áp dụng EfficientNetB6 trong học chuyển giao, chúng ta có thể tận dụng các đặc trưng đã học từ dữ liệu lớn như ImageNet và áp dụng chúng vào các bài toán cụ thể như phân loại hình ảnh, nhận diện đối tượng, hoặc các nhiệm vụ khác liên quan đến thị giác máy tính. Điều này giúp tăng tốc quá trình huấn luyện và cải thiện độ chính xác của mô hình trên các tập dữ liệu nhỏ hơn và chuyên biệt hơn.

EfficientNetB6 là một lựa chọn mạnh mẽ và hiệu quả cho các ứng dụng học chuyển giao, mang lại sự cân bằng giữa độ chính xác và hiệu quả tính toán, phù hợp với nhiều bài toán trong lĩnh vực thị giác máy tính.

2.2.2. Các bước Transfer Learning

Trong dự án này, tôi đã áp dụng phương pháp học chuyển giao (transfer learning) với mô hình EfficientNetB6 theo ba bước chính. Dưới đây là chi tiết các

bước thực hiện:

a) Bước 1: Freeze base layer + training new head classifier

Ở bước đầu tiên, toàn bộ các lớp cơ sở của mô hình EfficientNetB6 được đóng băng, nghĩa là các trọng số của chúng sẽ không được cập nhật trong quá trình huấn luyện. Một bộ phân loại mới (head classifier) được thêm vào đỉnh của mô hình và chỉ có bộ phân loại này được huấn luyện.

- *Mục đích:* Tận dụng các đặc trưng đã được học từ dữ liệu lớn như ImageNet để trích xuất các đặc trưng quan trọng từ dữ liệu mới.
- *Thực hiện:* Đóng băng tất cả các lớp của mô hình EfficientNetB6. Thêm một hoặc nhiều lớp Dense vào đỉnh của mô hình để tạo ra bộ phân loại mới. Huấn luyện chỉ các lớp mới này với dữ liệu cụ thể của dự án.

b) Bước 2: Freeze a half layers of base model and fine-turning

Sau khi bộ phân loại mới đã được huấn luyện ổn định, bước tiếp theo là mở khóa một nửa số lớp cơ sở của mô hình EfficientNetB6 và thực hiện quá trình tinh chỉnh (fine-tuning).

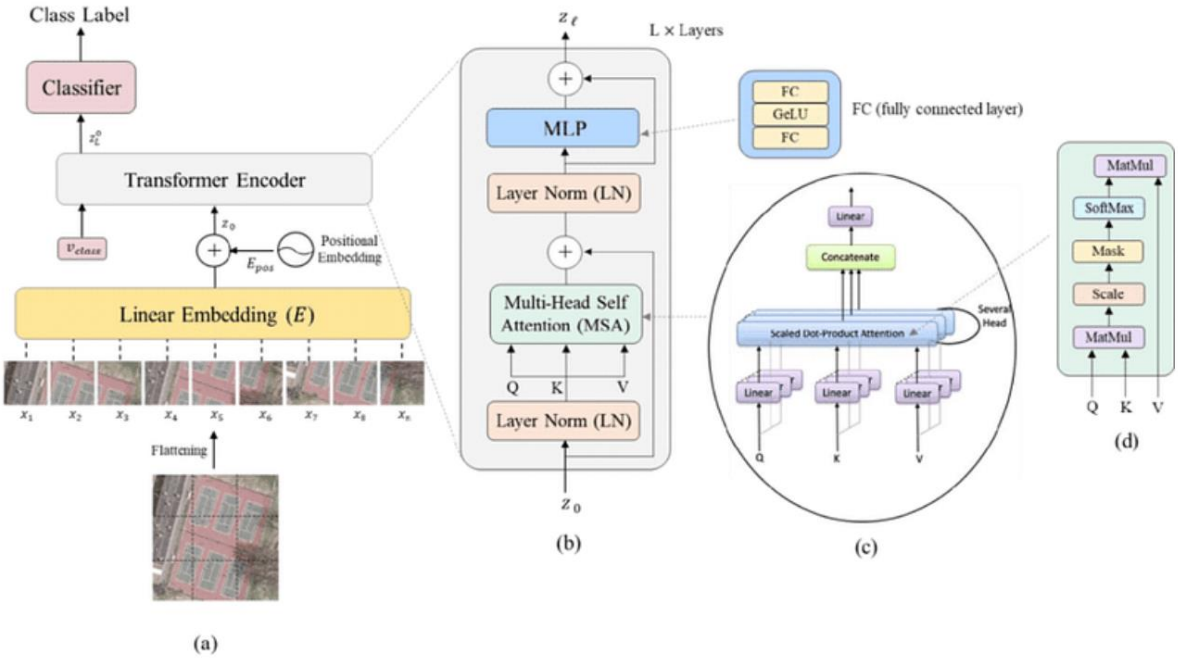
- *Mục đích:* Điều chỉnh các đặc trưng đã học từ dữ liệu nguồn để phù hợp hơn với dữ liệu đích, cải thiện hiệu suất của mô hình.
- *Thực hiện:* Giữ nguyên các lớp của bộ phân loại mới. Mở khóa một nửa số lớp cơ sở của mô hình (từ lớp giữa cho đến lớp cuối cùng). Huấn luyện mô hình với tốc độ học (learning rate) thấp hơn để tinh chỉnh các trọng số của các lớp mở khóa.

c) Bước 3: No freeze base model and full fine-turning

Bước cuối cùng là mở khóa tất cả các lớp của mô hình EfficientNetB6 và thực hiện quá trình tinh chỉnh toàn bộ mô hình.

- *Mục đích:* Tối ưu hóa toàn bộ mô hình, đảm bảo rằng cả các đặc trưng cơ bản và các lớp cao hơn đều phù hợp tối đa với dữ liệu đích.
- *Thực hiện:* Mở khóa toàn bộ các lớp của mô hình EfficientNetB6. Tiếp tục huấn luyện với tốc độ học rất thấp để tránh làm biến dạng các trọng số đã được tối ưu hóa từ các bước trước đó.

2.3. Sử dụng mô hình Vision transformer



Hình 2.9 Kiến trúc Vision transformer

2.3.1. Giới thiệu Vision transformer

Vision Transformer (ViT) là một kiến trúc mạng neural tiên tiến được thiết kế đặc biệt cho các nhiệm vụ liên quan đến thị giác máy tính, như phân loại hình ảnh, nhận diện đối tượng, và các bài toán thị giác khác. ViT đánh dấu một bước tiến quan trọng trong lĩnh vực học sâu, khi chuyển đổi từ việc sử dụng các mạng neural tích chập (CNN) truyền thống sang một mô hình dựa trên cơ chế transformer, vốn nổi tiếng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP).

2.3.2. Công thức sử dụng:

Những thay đổi trong kiến trúc vision transformer:

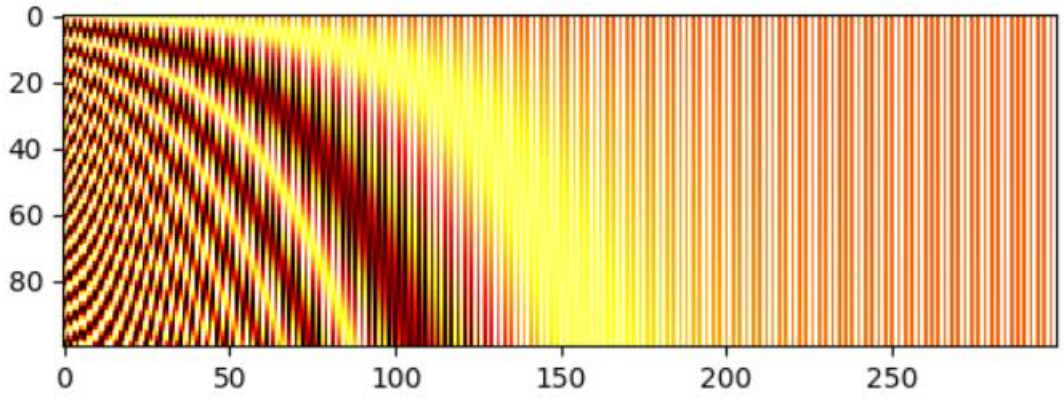
- Position Embedding:

$$p_{i,j} = \begin{cases} \sin\left(\dfrac{i}{10000^{\frac{j}{d_{emb\_dim}}}}\right) & if\ j\ is\ even \\ \cos\left(\dfrac{i}{10000^{\frac{j-1}{d_{emb\_dim}}}}\right) & if\ j\ is\ odd \end{cases}$$

Việc nhúng vị trí này là một hàm của số phần tử trong chuỗi và số chiều của từng phần tử. Vì vậy, nó luôn là một tensor 2 chiều hay còn gọi là "hình chữ nhật".

Đây là một hàm đơn giản, với số lượng mã thông báo và số thứ nguyên của

từng mã thông báo, tạo ra một ma trận trong đó mỗi tọa độ (i, j) là giá trị được thêm vào mã thông báo i trong thứ nguyên j.

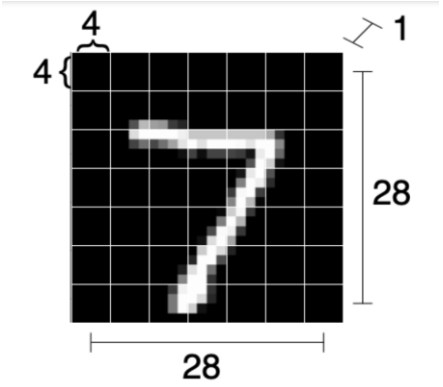


Hình 2.10 Hình chữ nhật thể hiện hàm thông tin vị trí

2.3.3. Kiến trúc mô hình

a) Step 1: Patchifying and the linear mapping

Ở bước này chúng ta thực hiện chia nhỏ hình ảnh đầu vào thành các sub-images, được gọi là patches. Mỗi patches sẽ là các hình vuông nhỏ có size xác định. Nếu input nhận vào có chiều là (N, C, H, W), thực hiện bước trên với số patch mỗi hàng cột là P sẽ có chiều đầu ra là (N, P*P, H*C/P * W*C/P).



Hình 2.11 Patchifying and the linear mapping

Áp dụng đối với hình trên thì sẽ có đầu ra là *(N, 49, 16)*

b) Step 2: Adding the classification token

Nói một cách đơn giản, đây là một token đặc biệt mà chúng tôi thêm vào mô hình của mình, có vai trò thu thập thông tin về các token khác. Điều này sẽ xảy ra với khối MSA (sau này). Khi thông tin về tất cả các token khác sẽ xuất hiện ở đây, chúng tôi sẽ có thể phân loại hình ảnh chỉ bằng cách sử dụng token đặc biệt này. Giá trị ban đầu của mã thông báo đặc biệt (giá trị được cung cấp cho transformer encoder) là một

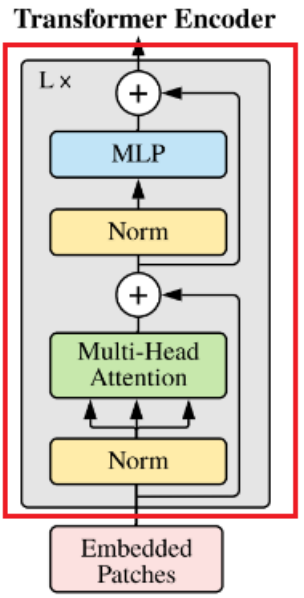
tham số của mô hình cần được tìm hiểu.

Bây giờ chúng ta có thể thêm một tham số vào mô hình của mình và chuyển đổi tensor mã thông báo *(N, 49, 8)* thành tensor *(N, 50, 8)* (chúng tôi thêm token đặc biệt vào mỗi chuỗi).

c) Step 3: Positional encoding

Chúng tôi xác định việc nhúng vị trí là một tham số của mô hình của chúng tôi (chúng tôi sẽ không cập nhật bằng cách đặt yêu cầu requires_grad của nó thành False). Lưu ý rằng trong phương thức chuyển tiếp, vì tokens có kích thước (N, 50, 8), chúng ta phải lặp lại ma trận positional encoding (50, 8) N lần.

d) Step 4: The encoder block



The encoder block được tạo thành từ một chồng các lớp transformer. Mỗi lớp biến áp chủ yếu bao gồm một feed-forward network và một mạng chuyển tiếp nguồn cấp dữ liệu. Để mở rộng mô hình tốt hơn và ổn định quá trình đào tạo, hai lớp chuẩn hóa và bỏ qua kết nối được thêm vào lớp transformer.

Hãy triển khai một lớp máy biến áp (được gọi là khối trong mã vì đây là khối xây dựng cho bộ mã hóa máy biến áp). Chúng ta sẽ bắt đầu với mạng chuyển tiếp nguồn cấp dữ liệu, là một MLP hai lớp đơn giản có kích hoạt GELU ở giữa.

e) Step 5: Classification MLP

Cuối cùng, chúng tôi chỉ có thể trích xuất token phân loại (mã thông báo đầu tiên) từ N chuỗi của mình và sử dụng từng mã thông báo để nhận N phân loại.

Vì chúng tôi đã quyết định rằng mỗi mã thông báo là một vectơ D chiều và vì chúng tôi có 15 loài hoa có thể nên chúng tôi có thể triển khai MLP phân loại dưới dạng ma trận Dx15 đơn giản, được kích hoạt bằng hàm SoftMax.

2.3.4. Tham số mô hình

Trong kiến trúc của model chúng tôi sử dụng bộ tham số sau:

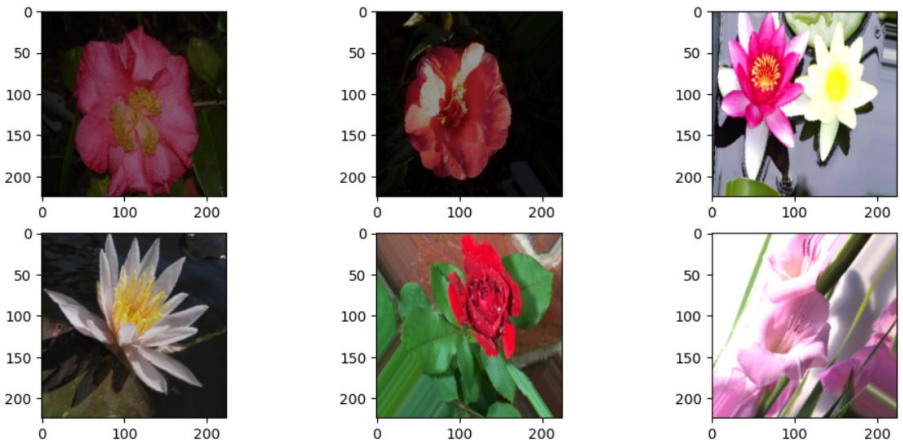
- *(C,H,W):* (3, 224, 224) là chiều sâu, chiều cao, rộng của hình ảnh đầu vào;
- *n_patches:* 16 là số lượng patches được chia ra mỗi trục dọc và trục ngang;
- *n_blocks:* 12 là số blocks encoder transformer trong kiến trúc sử dụng;
- *hidden_d:* 768 là số chiều của lớp ẩn trong kiến trúc;
- *n_heads:* 12 là số head của lớp multi-head attention;
- *out_d:* 15 là số chiều của nhãn đầu ra (15 loài hoa).

Chương 3. Kết quả

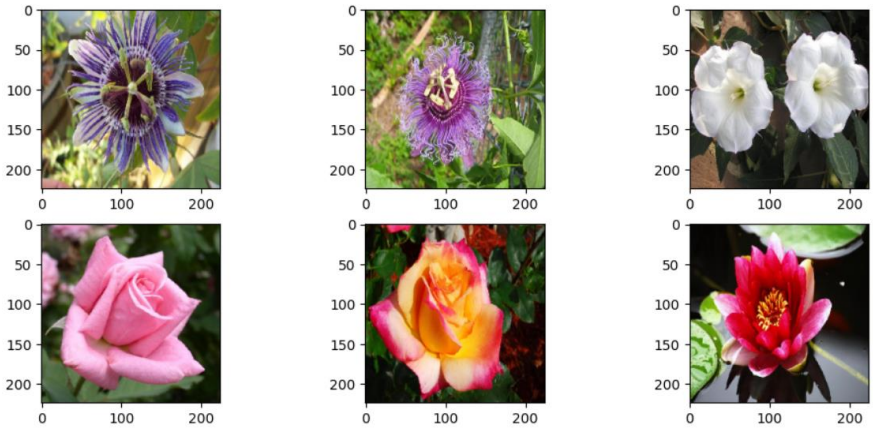
3.1. Sử dụng kiến trúc CNN – GoogLeNet:

3.1.1. Tập dữ liệu :

- Sử dụng tập dữ liệu *Oxford 102 Flower*, thực hiện cắt 15 loài hoa trong tổng số 102 loài hoa để huấn luyện.
- Số lượng ảnh : *1796*, mỗi ảnh là một bông hoa.
- Tỉ lệ *train:test = 80%:20%*, tương ứng 1436 ảnh cho tập train và 360 ảnh cho tập test
- Số loài hoa: 15
- Ảnh được giảm kích thước xuống *(224, 224)*, chuyển đổi sang định dạng RGB.
- Dữ liệu được thực hiện tăng cường, chuẩn hoá theo các giải pháp tiền xử lý nêu trên.



Hình 3.1 Tập dữ liệu train được thực hiện tăng cường



Hình 3.2 Tập dữ liệu test không được thực hiện tăng cường

Nhận xét tập dữ liệu: Có thể thấy tập train màu sắc sẽ bị mờ, hình ảnh bị méo, bị cắt từ đó có thể học được các đặt trưng phức tạp trong các điều kiện khác nhau.

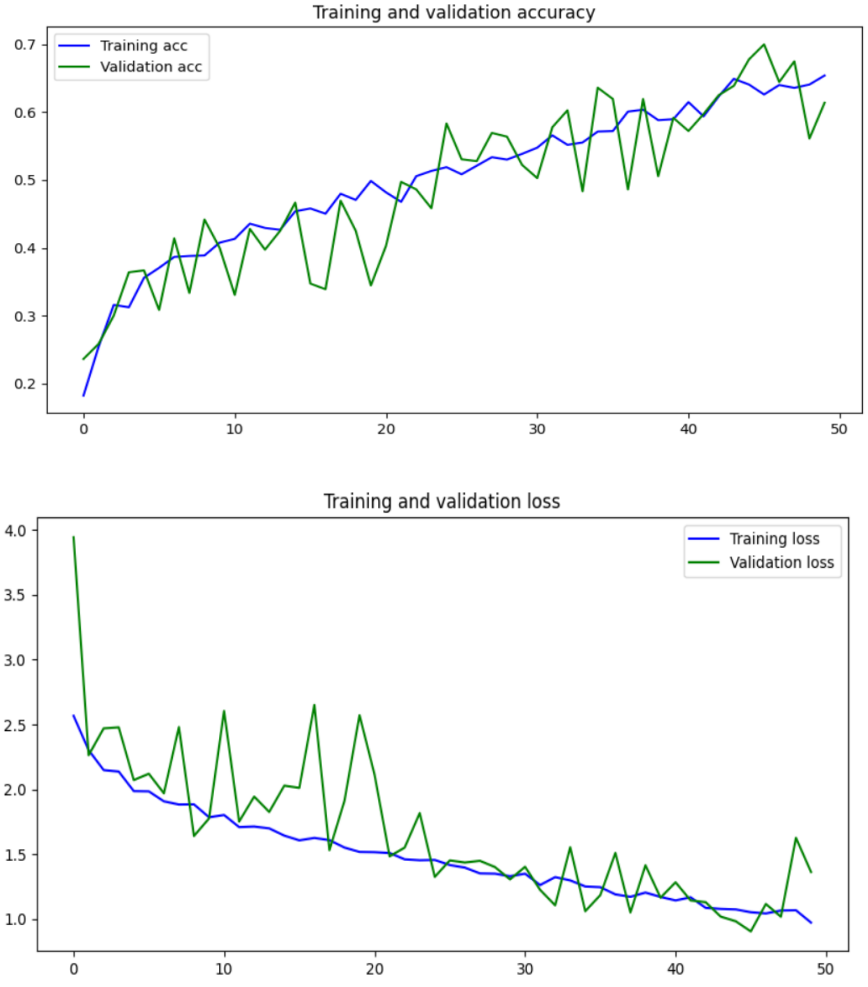
Trong khi tập test vẫn ở chế độ màu sắc tốt, kết quả đối với tập test có thể sẽ không tăng, nhưng khi thực nghiệm với nhiều cường độ sáng, góc ảnh khác nhau thì sẽ mang lại hiệu quả với ứng dụng thực tế.

3.1.2. Các tham số trong quá trình huấn luyện:

- *Metric sử dụng đánh giá:* Accuracy (= TP + TN / TP + TN + FP + FN)
- *Loss function:* Cross Entropy
- *Early stop:* Có sử dụng dừng sớm nếu như độ chính xác trên tập validation không tăng trong *10 epochs* liên tiếp.

3.1.3. Quá trình huấn luyện

- GoogLeNet

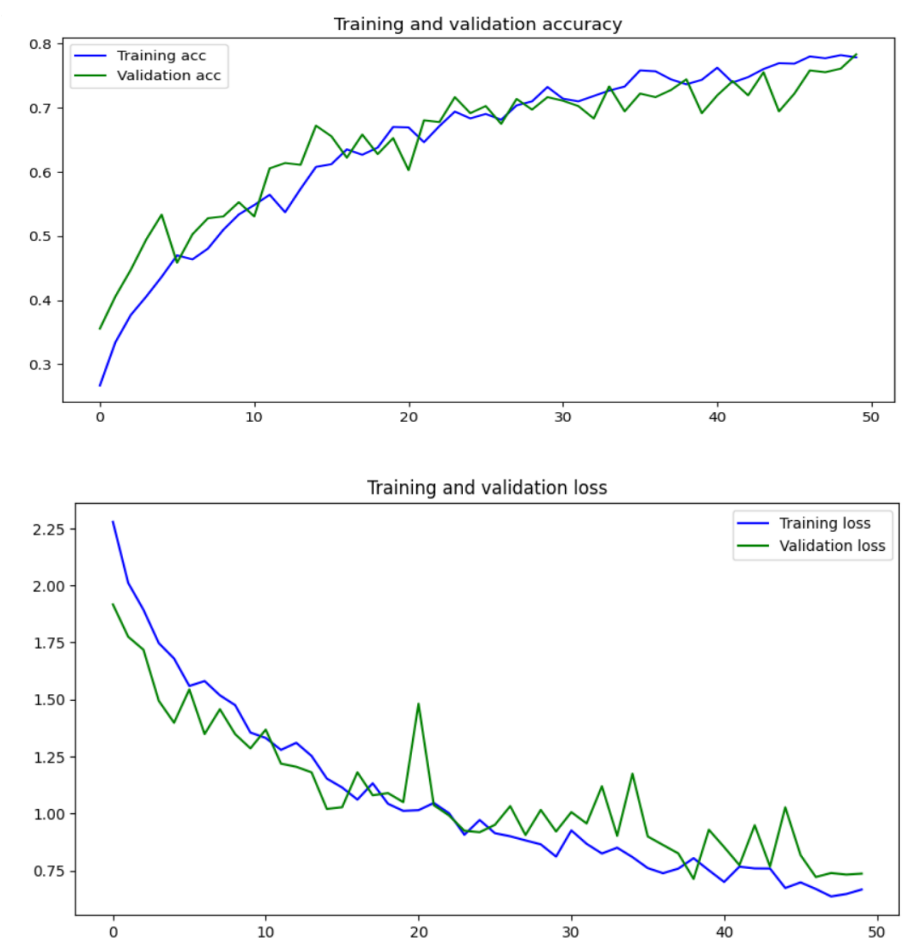


Hình 3.3 Đồ thị loss và accuray của GoogLeNet

Nhận xét: Với model GoogLeNet này thì độ dốc khá ổn định, trên tập train thì đường màu xanh dương luôn tăng đối với accuracy và luôn giảm đối với tập validation, còn trên tập validation đường màu xanh lá thì sẽ giao động quanh đường xanh dương nhưng xu hướng tăng giảm vẫn tương tự trên tập train. Đây là một đồ thì

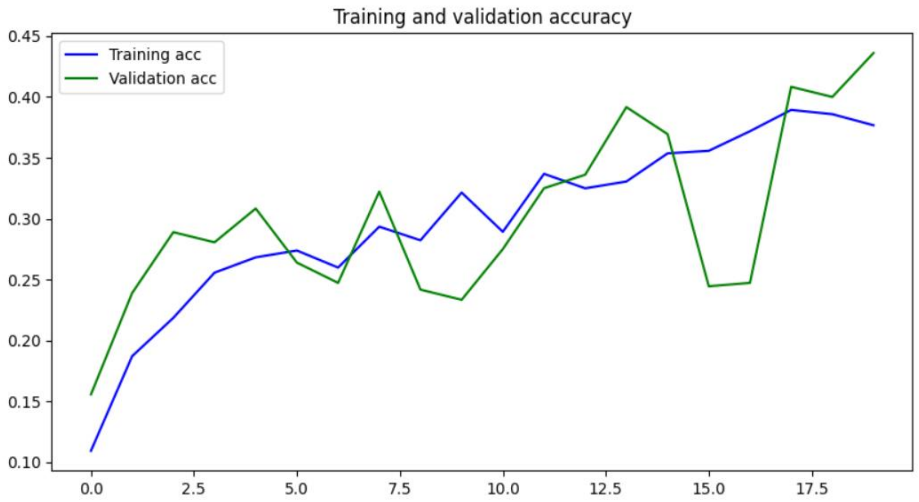
loss, accuracy như mong đợi.

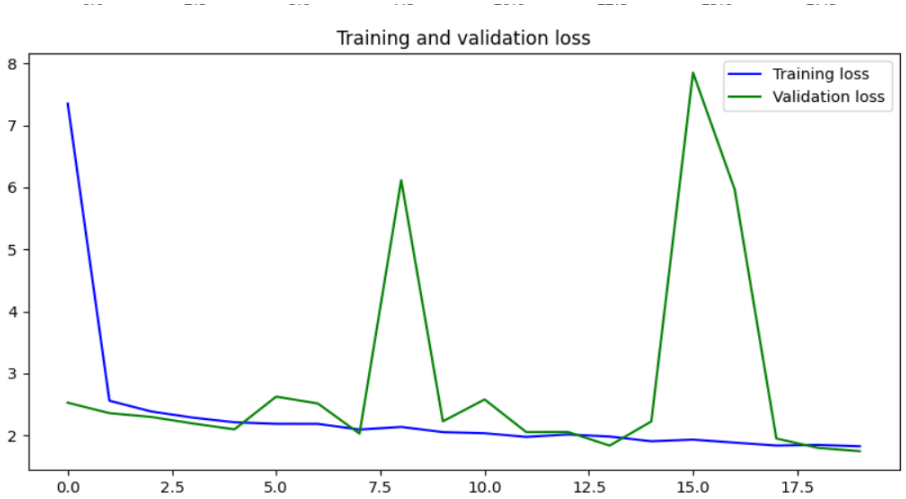
- My model 2



Hình 3.4 Đồ thị loss và accuray của my model 2

- My model 3





Hình 3.5 Đồ thị loss và accuray của my model 3

Nhận xét: Đối với my model 2, có thể thấy sự khác biệt chính là việc thêm và bỏ đi các lớp *Batch Normalizer*, nhưng phần đồ thì có phần biến động lớn hơn.

3.1.4. Kết quả

Kết quả chính là độ chính xác trên tập test (gồm 360 ảnh thuộc 15 loài hoa đã được huấn luyện. Dưới đây là bảng thống kê.

Bảng 3.1 Bảng thống kê kết quả các mô hình CNN

Model	**Accuracy**	**Loss: CrossEntropy**	**Params**
LeNet (Có tinh chỉnh)	76.11% (55e)	0.8511	30,561,199
AlexNet	49.17% (20e)	1.3552	28,878,127
GoogLeNet	89.72% (100e)	0.3849	6,260,687
My model 1 (5 Conv + 2 Dense + kernel 3*3) . Yes BatchNorm	76.42% (67e)	0.8726	51,531,183
My model 2 (4 Conv + 2 Dense + kernel 5*5) . No BatchNorm	79.17% (64e)	0.6839	30,028,719
My model 3 (4 Conv + 2 Dense + kernel 5*5) . Yes BatchNorm	78.33% (58e)	0.7257	30,029,679

a) Nhận xét

- Kết quả cao nhất của các model tái xây dựng và tự xây dựng là *89.72%* đạt

được đối với mô hình GoogLeNet tại *100 epochs*, trong khi mô hình này có số lượng tham số thấp nhất là hơn *6 triệu tham số.*

- Kết quả thấp nhất là mô hình AlexNet với độ chính xác rất thấp *29.17%* khi chỉ mới huấn luyện được *20 epochs*.
- Các model tự xây dựng và LeNet-5 thì có kết quả ít biến động hơn giao động trong khoảng *76.42%* đến *79.17%* nằm ở mức trung bình, và số lượng tham số cũng tương đương nhau gần *30 triệu tham số*, riêng có my model 1 là số lượng tham số vượt trội *hơn 51 triệu*.

b) Giải thích:

- Model GoogLeNet đạt hiệu quả cao do đây là mô hình tiên tiến nhất so với các mô hình còn lại, đặc trưng của mô hình này là sự tận dụng tối đa khả năng kết hợp các đặc trưng của hình ảnh lại với nhau đồng thời trích xuất đặc trưng tối thiểu trước khi đưa vào fully connected.
- Model AlexNet là mô hình gây tranh cãi khi nó được đánh giá cao hơn so với LeNet-5 trên tập ImageNet nhưng khi áp dụng đối với tập dữ liệu Flowers thì kết quả dừng sớm ở *20 epochs* và quá trình tiếp tục huấn luyện vẫn sẽ không tăng. Nguyên do chỉnh ở đây chỉ có thể nói là do mỗi mô hình CNN có thể sẽ phù hợp với tập dữ liệu nhất định.
- Các model tự xây dựng cùng với model LeNet-5 đều là các mô hình có cấu trúc thẳng, tức là chỉ có một trục chính duy nhất từ input đầu vào đến khi trả về nhãn dự đoán, số lượng tham số cũng ở mức tương đương. Điều đó là lý do chính các kết quả tương tự nhau, giao động nhỏ.

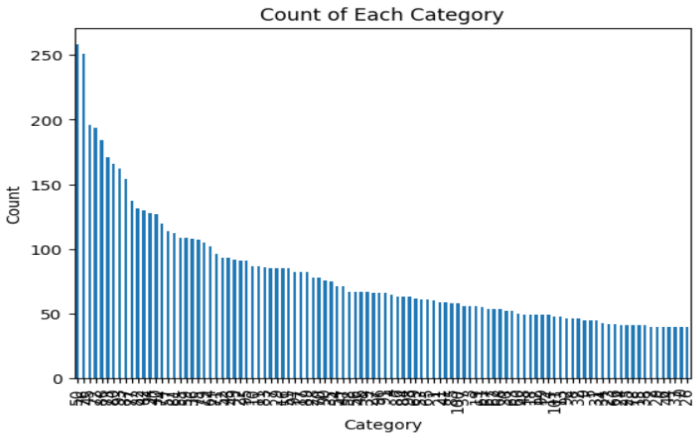
c) Kết luận rút ra trong quá trình tinh chỉnh tham số:

Kinh nghiệm tinh chỉnh tham số và mô hình:

- Các hàm activation trong các lớp tích chập không được khác nhau.
- Hàm activation trong các lớp tích chập và hàm activation trong các lớp fully connected khác nhau có thể sẽ hiệu quả hơn.
- Về vị trí của BatchNorm và Hàm phi tuyến (vd: relu, tanh), tuỳ bộ dữ liệu và trường hợp áp dụng sẽ có kết quả khác nhau. Với mỗi model và mỗi tập dữ liệu khác nhau nên thực nghiệm để xem xét kết quả.
- Các lớp dropout giảm overfitting.
- Các lớp dropout nên đặt sau các bước biến đổi quan trọng (sau các khối conv, hoặc sau khi flatten).
- Sử dụng các khối inception module để học ở các mức độ kernel khác nhau.

3.2. Sử dụng Transfer learning EfficientNetB6

3.2.1. Tập dữ liệu



Hình 3.6 Đồ thị thể hiện số lượng mỗi loài hoa

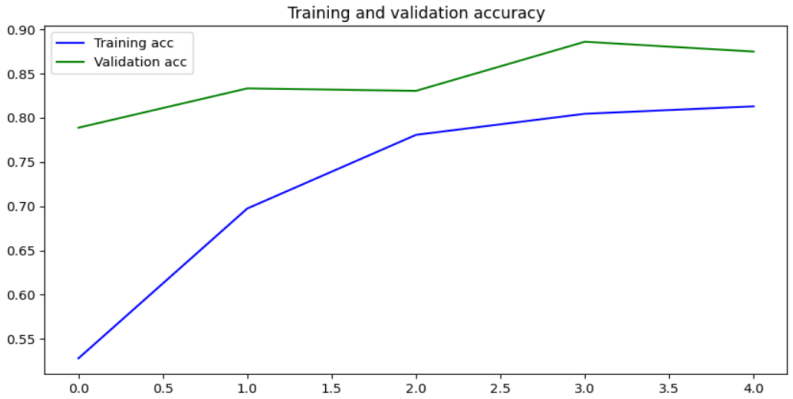
Tập dữ liệu vẫn là tập dữ liệu sử dụng ở giải pháp 2.1 tuy nhiên số lượng các loài hoa sử dùng là toàn bộ tập dữ liệu gốc 102 loài hoa với mỗi lớp bao gồm từ 40 đến 258 hình ảnh. Tổng *8190* ảnh.

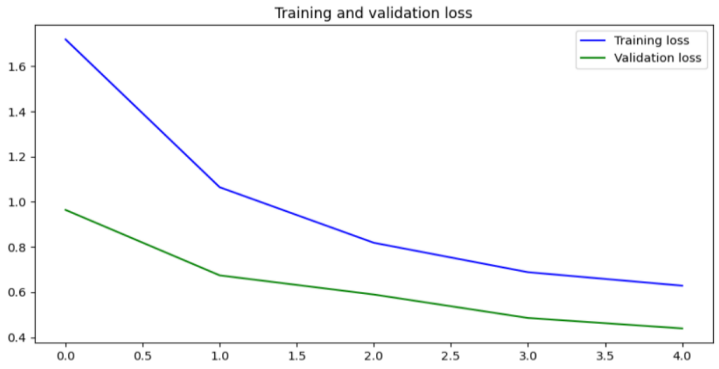
Các hình ảnh có tỷ lệ lớn, tư thế và sự thay đổi ánh sáng. Ngoài ra, có những danh mục có sự khác biệt lớn trong danh mục và một số danh mục rất giống nhau.

3.2.2. Các tham số trong quá trình huấn luyện:

- LR_step1 = 1e-3
- LR_step2 = 1e-4
- LR_step3 = 1e-5
- *Metric: Accuracy* = (TP + TN / TP + TN + FP + FN)
- *Loss function:* Categorical Cross Entropy
- *Optimizer:* Adam
- *Epochs:* 5 cho mỗi step

3.2.3. Quá trình huấn luyện





Hình 3.7 Đồ thị loss và accuray của transfer learning efficientNetB6

Nhận xét:

- Với mô hình đã được pretrain thì các đặc trưng đã được học trên bộ dữ liệu lớn, có nhiều đặt trưng khác nhau, do đó khi thực hiện transfer learning thì ban đầu độ chính xác đã ở ngưỡng *50%* là rất cao đối với epochs đầu tiên, tuy nhiên huấn luyện tiếp tục thì tốc độ tăng của những epochs sau có phần chậm lại.
- Để tránh mất đi khả năng trích xuất của bộ tham số ban đầu nên số epochs huấn luyện ít để giữa lại đặc trưng quan trọng.

3.2.4. Kết quả

Bảng 3.2 Kết quả các bước transfer learning efficientNetB6

Step	Accuracy	Số epochs
1	84.50%	5
2	92.42%	5
3	92.92%	5

a) Nhận xét

- Qua từng bước transfer learning kết quả liên tục cải thiện từ *84.50%* đến *92.92%*.
- Độ chính xác cao *92.92%* cao hơn nhiều so với độ chính xác của model GoogLeNet ở giải pháp trước. Tuy nhiên số lượng tham số cao hơn gần 40 triệu tham số.

b) Giải thích

- Mỗi bước transfer sẽ thực hiện huấn luyện tập trung vào một phần trong model do đó khi các phần lần lượt được huấn luyện sẽ mang lại hiệu quả trên tập test.
- Đây là model đã được học trên bộ dữ liệu lớn và cách trích xuất đặt trưng tốt từ trước, nên độ chính xác sẽ cao hơn so với mô hình chưa được huấn luyện trên dữ liệu lớn. Tuy nhiên model còn khá nặng *40tr tham số*, nếu như tối ưu lại kiến trúc trên tập hoa thì sẽ hiệu quả với thời gian xử lý hơn.

3.3. Sử dụng Vision transformer

Bảng 3.3 Bảng kết quả các kiến trúc Vision Transformer

Kiến trúc model	Accuracy
patchify bằng cách cắt hình + function position 2D	10.32%
patchify bằng convolution + position 1D (bài báo gốc)	15.45%
patchify bằng convolution + function position 2D	25.55%

3.3.1. Nhận xét:

- Đạt hiệu xuất cao nhất trong ViT sử dụng patchify bằng convolution + function position đề xuất là *25.55%*, tuy nhiên kết quả còn quá thấp so với các giải pháp trước.
- Cải tiến đáng kể *(10%)* nằm ở việc áp dụng đúng phương pháp patchify trong bài báo gốc + position đề xuất.

3.3.2. Giải thích

- ViT đạt hiệu quả cao trên các bộ dữ liệu lớn, do dữ liệu hoa còn khá nhỏ, dễ dẫn đến việc học các đặc trưng phức tạp hơn của mô hình ViT bị hạn chế (được xem là ưu thế của phương pháp này). Do đó các cải tiến tiếp theo cần tập trung vào tập dữ liệu.

Chương 4. Kết luận và hướng phát triển

4.1. Kết quả đạt được:

- Hiệu quả cao trên model GoogLeNet tái xây dựng là *89.72%*.
- Rút ra các kết luận và kinh nghiệm trong quá trình tinh chỉnh tham số.
- Độ chính xác cao nhất trên tập hoa khi thực hiện transfer learning model efficientNetB6 là *92.92%*.
- Tìm hiểu và tái xây dựng ViT nhưng kết quả vẫn còn hạn chế.

4.2. Hướng phát triển

- Đối với GoogLeNet thực hiện train trên toàn bộ dữ liệu hoa và ImageNet để học được các đặc trưng mới hơn.
- Tiếp tục phát triển và huấn luyện ViT trên dữ liệu lớn.
- Tìm hiểu các mô hình multi-task huấn luyện trên dữ liệu đa dạng và adaptor sang bộ dữ liệu hoa.

Danh mục tài liệu tham khảo

[1] Architecture of EfficientNetB6: https://www.researchgate.net/figure/Architecture-of-EfficientNetB6_fig4_374724674
[2] Scaling Up Visual and Vision-Language Representation Learning with NoisyText Supervision: https://arxiv.org/abs/2102.06183
[3] Searching for MobileNetV3: https://arxiv.org/abs/1905.02244
[4] MobileNetV3: A Highly Efficient Large-Scale Convolutional Neural Network: https://arxiv.org/abs/1905.02244
[5] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks: https://arxiv.org/abs/1905.11946
[6] Caling Up Visual and Vision-Language Representation Learning with NoisyText Supervision, https://arxiv.org/abs/2102.06183
[7] Searching for MobileNetV3, https://arxiv.org/abs/1905.02244
[8] MobileNetV3: A Highly Efficient Large-Scale Convolutional Neural Network, https://arxiv.org/abs/1905.02244
[9] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, https://arxiv.org/abs/1905.11946
[10] Architecture of EfficientNetB6, https://www.researchgate.net/figure/Architecture-of-EfficientNetB6_fig4_374724674
[11] Vision transformers from scratch pytorch a step by step guide, https://readmedium.com/vision-transformers-from-scratch-pytorch-a-step-by-step-guide-96c3313c2e0c
[12] Huỳnh Hữu Hưng, “Tài liệu học tập Huỳnh Hữu Hưng”.