ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN-ĐIỆN TỬ

Ứng dụng mô hình Resnet vào bài toán UAV tránh vật cản trong nhà

Giáo viên hướng dẫn: PGS.TS Đỗ Trọng Tuấn
Sinh viên thực hiện: Đặng Hữu Hiếu
MSSV: 20203416

Mô tả nhiệm vụ

STT	Nhiệm vụ	Mô tả	Kết quả đạt được
1	Mô hình resnet	Tìm hiểu về kiến trúc của mô hình resnet v8	Hoàn thành và hiểu rõ được kiến trúc resnet8
2	Training custom dataset	Training lại bộ dataset của UAV trong môi trường airsim bằng mô hình resnet 8	Training, test và bay thử thành công
3	Cải tiến mô hình	Thay thế mô hình resnet8 bằng mô hình resnet50	Training, test và bay thử thành công So sánh với mô hình resnet8

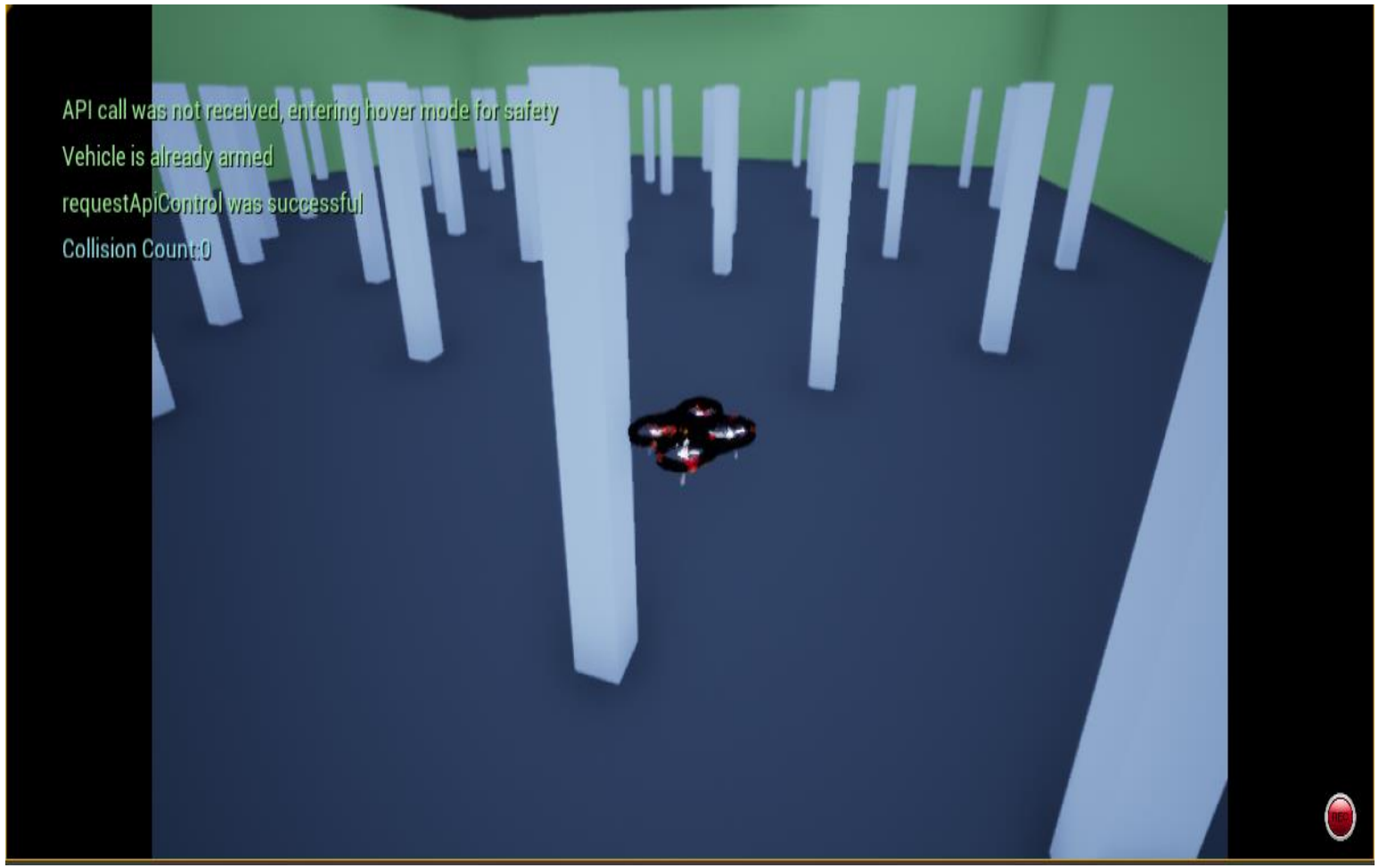
Yêu cầu hệ thống

Máy	Cấu hình				Ghi chú
	Tối thiểu	Thực tế	Tối thiểu	Thực tế	
Laptop Msi Bravo	Chip i5, RAM 4GB, tốc độ 1.6-1.8 GHz, Window OS, Linux OS	Chip R5 4700H, RAM 8GB, 1.6-1.8 GHz, Window OS	Visual Studio Code 2018 trở lên	Visual Studio Code 2018	
			Môi trường Python Anaconda 3.6 trở lên	Môi trường Python Anaconda 3.7.10	
			Anaconda 2018 trở lên	Anaconda 2020	

A. Bài toán đặt ra

Môi trường: Môi trường mô phỏng của Airsim trên phần mềm Unreal Engine
Yêu cầu: UAV tự động tránh vật cản trong môi trường và đi đến điểm đích
Mô tả hành động: UAV thực hiện các hành động:

- Đi thẳng - 0
- Rẽ trái 1 góc pi/18 - 1
- Rẽ phải 1 góc pi/18 – 2
- Lùi lại nếu va chạm



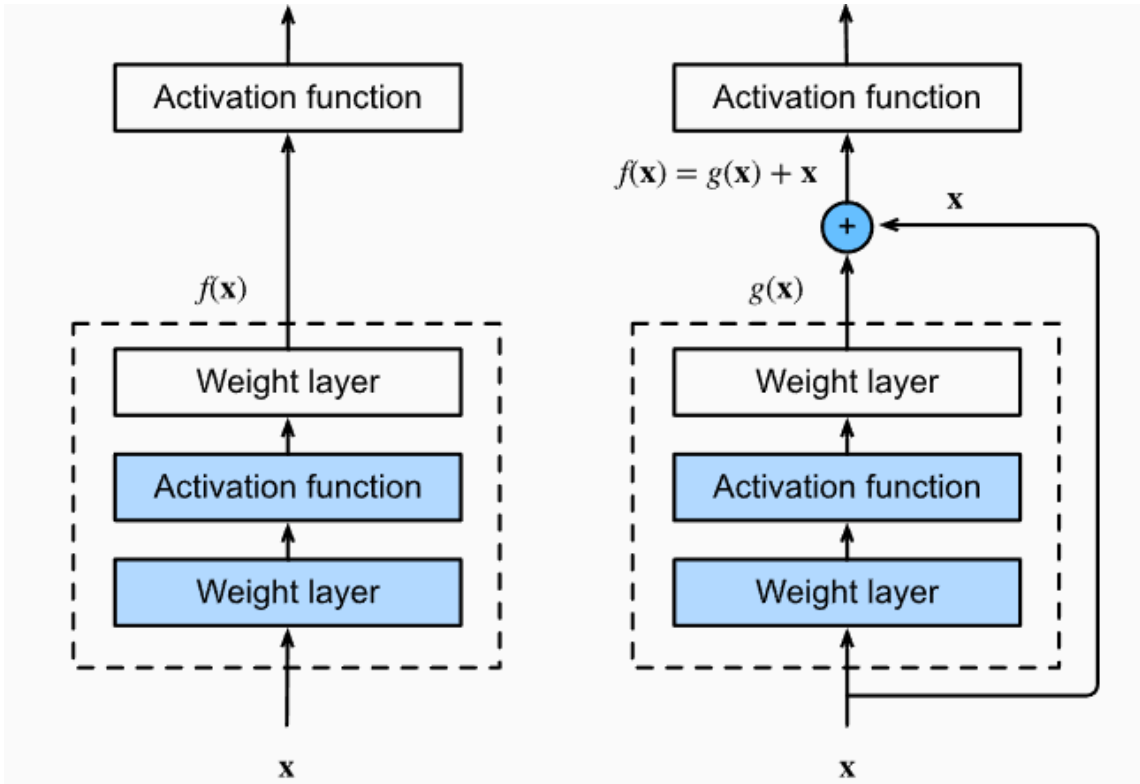
1) Residual Neural Network

Yêu cầu: Tăng độ chính xác cho mạng CNN => Tăng số lượng layers => Vanishing/ Exploding Gradient => Gradient bằng 0 / Gradient quá lớn

➡ Mô hình Residual Neural Network

Residual Neural Network (Resnet)

- ❑ Cấu tạo mô hình tương tự với CNN: convolutional layers, pooling layers, fully connected layers, hàm activation
- ❑ Kĩ thuật Skip Connection: Kết nối 1 layers với các layers ở xa hơn, bỏ qua 1 số layers ở giữa

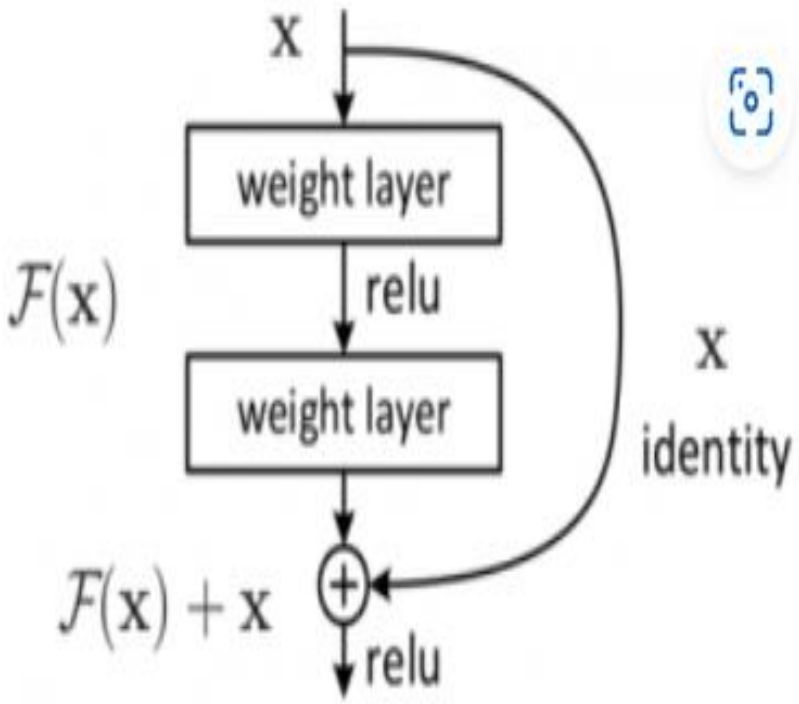


Mô hình NN thông thường

Mô hình Resnet cơ bản

B. Cơ sở lý thuyết

1) Residual Neural Network



Mô hình cơ bản của Resnet

Mô tả:

- Bổ sung input X và đầu ra của layers

=> Gradient tiến chậm về 0 hơn

➡ ~~Vanishing/ Exloding~~ Gradient

- Do cộng thêm X nên layers sau đảm bảo có hiệu suất ít nhất là bằng layers trước

=> Cải thiện hiệu suất mô hình

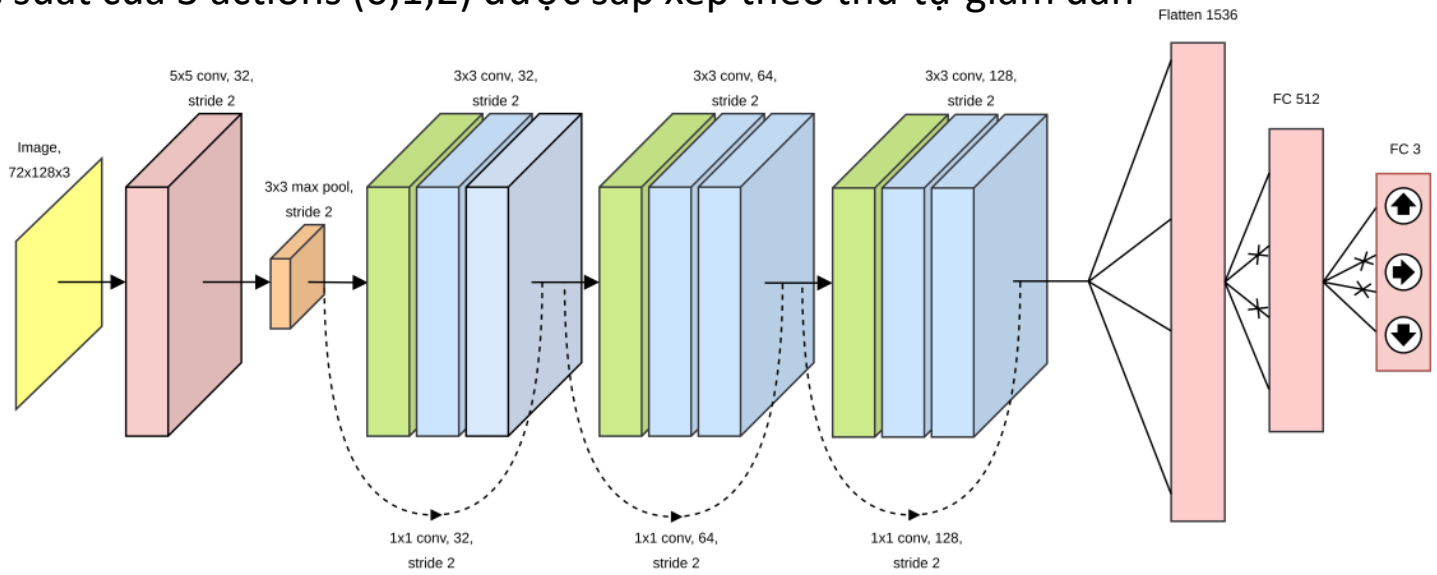
- Kĩ thuật này tạo nên các Residual Blocks. Sắp xếp các Resiual Blocks tạo nên Resnet
- Cấu tạo mô hình tương tự với CNN: convolutional layers, pooling layers, fully connected layers, hàm activation

❖ F(X) + X là 1 phép cộng ma trận nên X sẽ được qua 1 lớp conv(1,1) để điều chỉnh lại số chiều cho bằng với F(X)

2) Resnet 8

Mô tả kiến trúc:

- Input: Ảnh chụp trực tiếp từ UAV (72x128x3)
- Sau đó Input qua 32 lớp conv 5x5 đầu tiền, đầu ra thu được qua max pooling layers, activation function là hàm Relu
- Tiếp tục cho đầu ra thu được ở trên đi qua 3 khối residual block, trong đó:
  +) Đầu tiên là lớp BatchNormalization nhằm chuẩn hóa Input
  +) Sau đó là qua hàm activation Relu
  +) Cuối cùng là qua lớp tích chập Conv2D(32,(3,3))
- Đầu ra thu được sẽ được qua Flatten layer sau đó qua lớp fully connected layer
- Output: vector (3,1) với xác suất của 3 actions (0;1;2) được sắp xếp theo thứ tự giảm dần



3) Tóm tắt quy trình

Tóm tắt quy trình hệ thống

C. Training model Resnet 8

1) Dataset

Custom dataset gồm 2 file:

- File imgs chứa ảnh bao gồm 1018 ảnh png 144x256
- File label: 1 file excel chứa nhãn của ảnh

Sử dụng kĩ thuật data augmentation để tăng số lượng data từ 1018 lên 11198

Data sẽ được chia 8-2 để làm data train và test

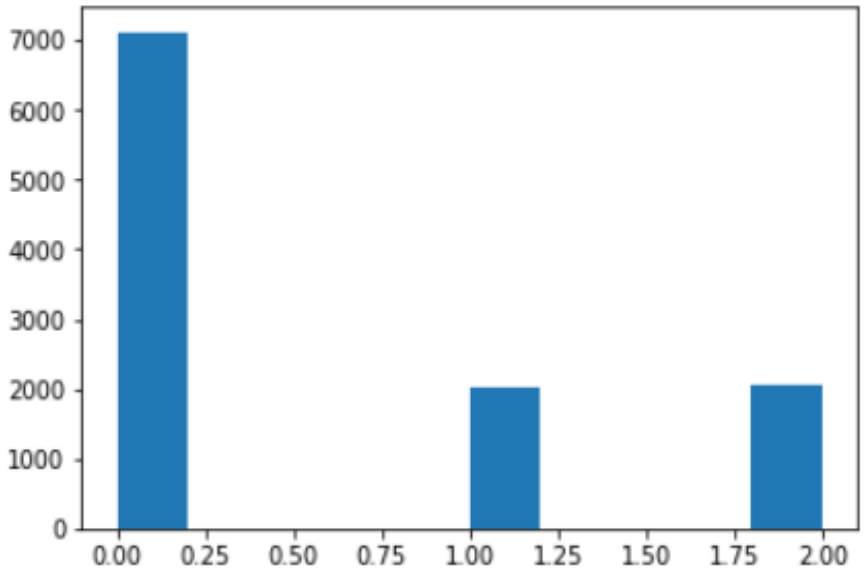
2) Training model Resnet 8

2.1. Tiền xử lý data:

```
IMAGE_HEIGHT, IMAGE_WIDTH, IMAGE_CHANNELS = 64, 64, 3

#1 ham sua nhan ve 0: di thang; 1: re trai; 2: re phai
def fix_label (y):
  for i in range(0, len(y)):
    if y[i] == 0:
      y[i] = 0
    elif y[i] < 0:
      y[i] = 1
    else :
      y[i] = 2
  return y
```

Sửa nhãn của ảnh thành các action (0;1;2)

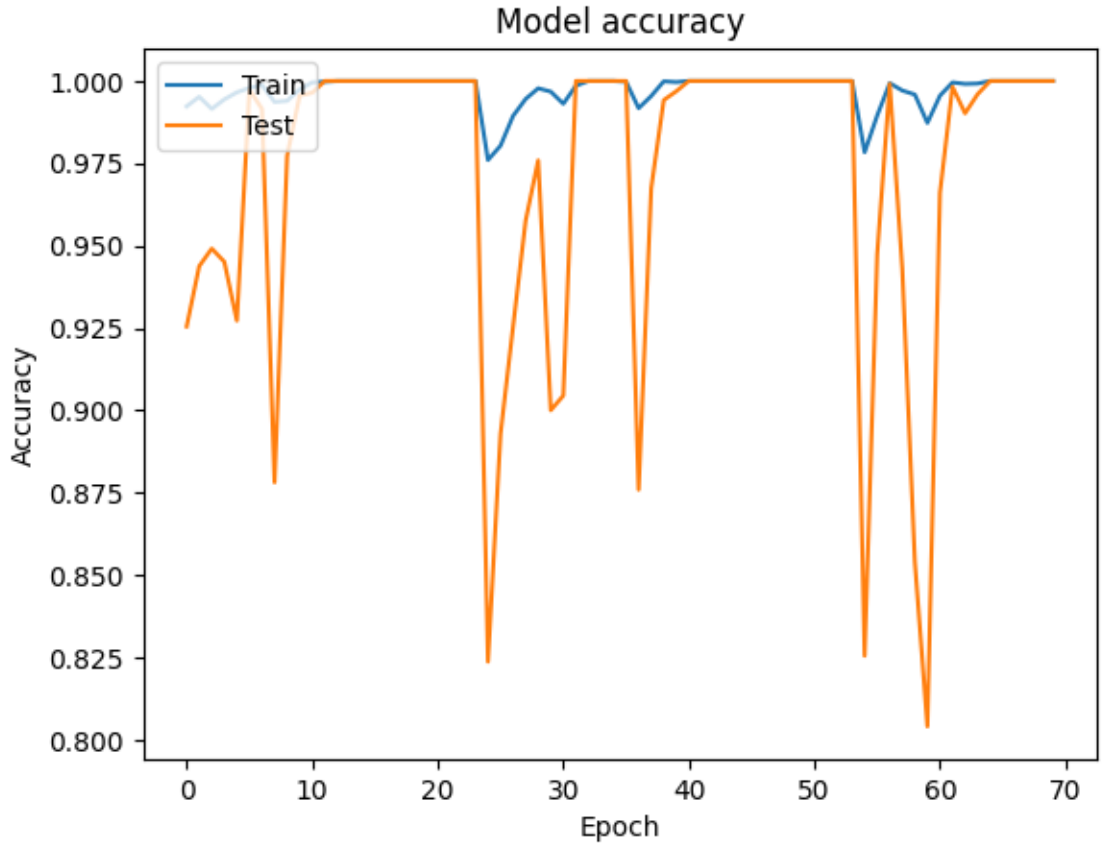


Đồ thị biểu thị các action tương ứng với nhãn sau khi được sửa

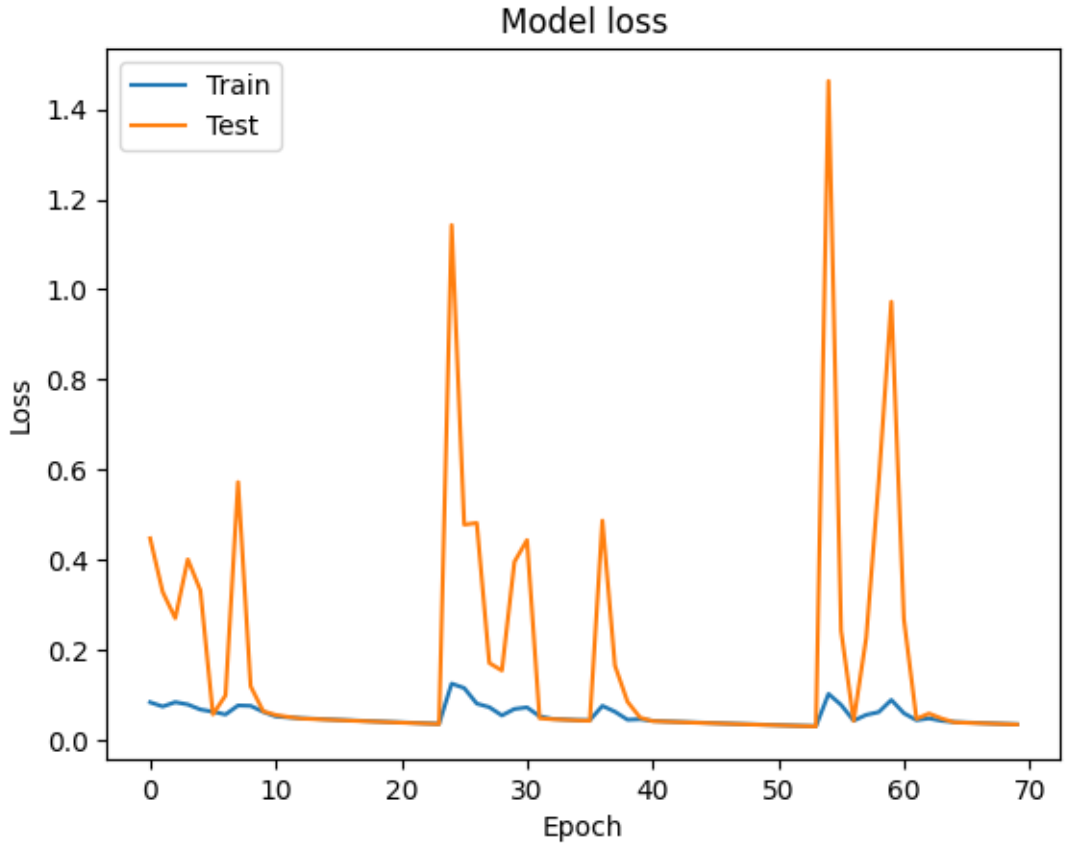
C. Training model Resnet 8

2.2. Tiến hành training

Data được train với batch_size=128, epoch=70. Kết quả train với độ chính xác từ 0.97 ->0.99



Đồ thị Accuracy

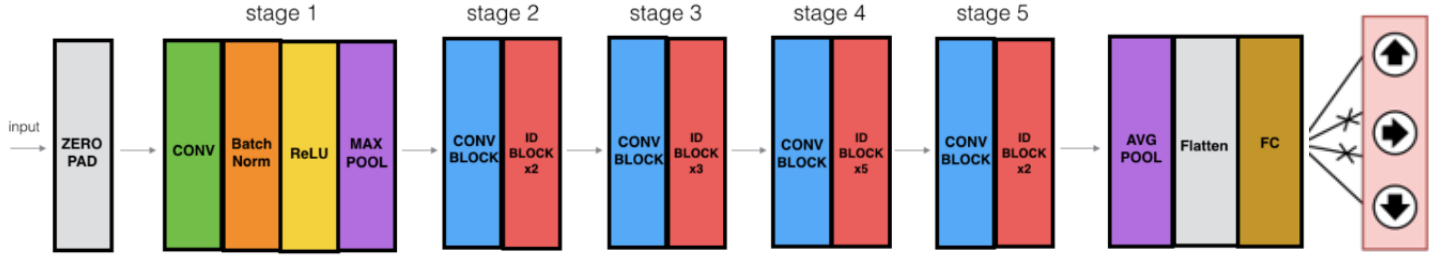


Đồ thị hàm loss

D. Thay thế mô hình

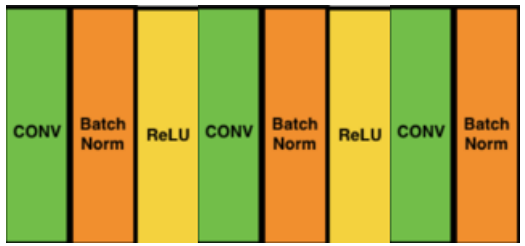
Mục tiêu: thay thế mô hình resnet 8 bằng mô hình resnet50 và đưa ra so sánh

1. Kiến trúc mô hình



Điểm khác biệt:

- Đầu tiên input được đưa qua 1 lớp zeropadding, có nhiệm vụ thêm số 0 vào các các góc của ảnh, nhằm đảm bảo output qua mỗi lớp tích chập sẽ không bị thất thoát
- Mô hình có 4 khối Residual Block, trong đó cấu tạo của mỗi Block gồm các khối ConvBlock và IDBlock
- IDBlock là các khối không có lớp tích chập ở shortcut, trong khi đó ConvBlock bổ sung them 1 conv layer ở shortcut. 2 khối này đều có cấu tạo cơ bản như sau:

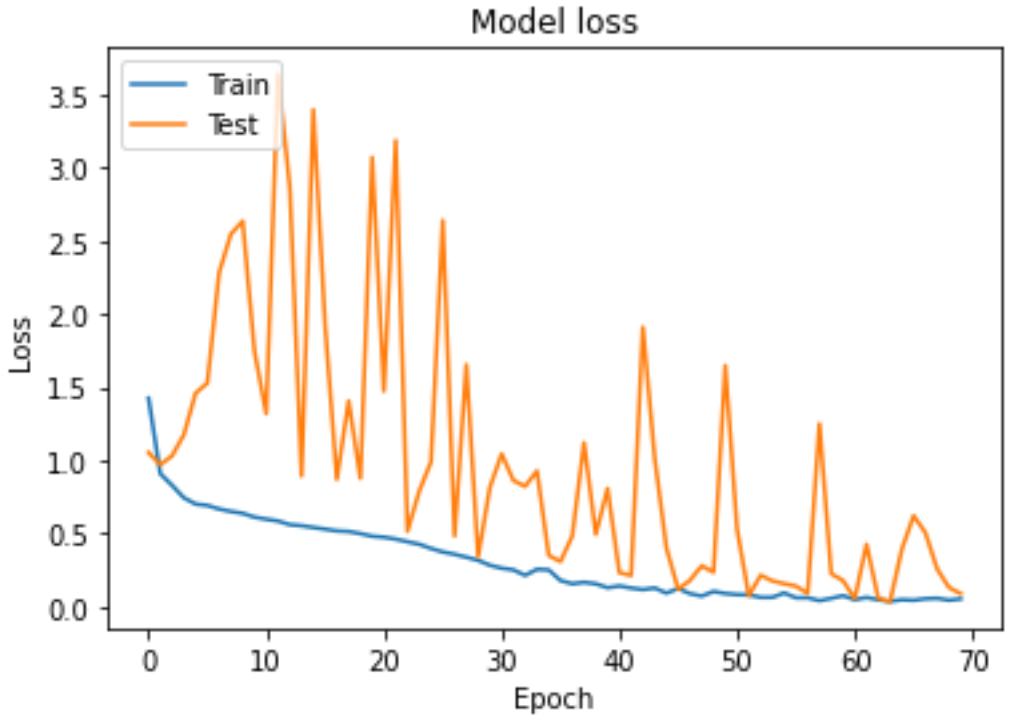


- Mỗi lớp Conv trong mô hình đều có số layer nhiều hơn đáng kể so với Resnet8

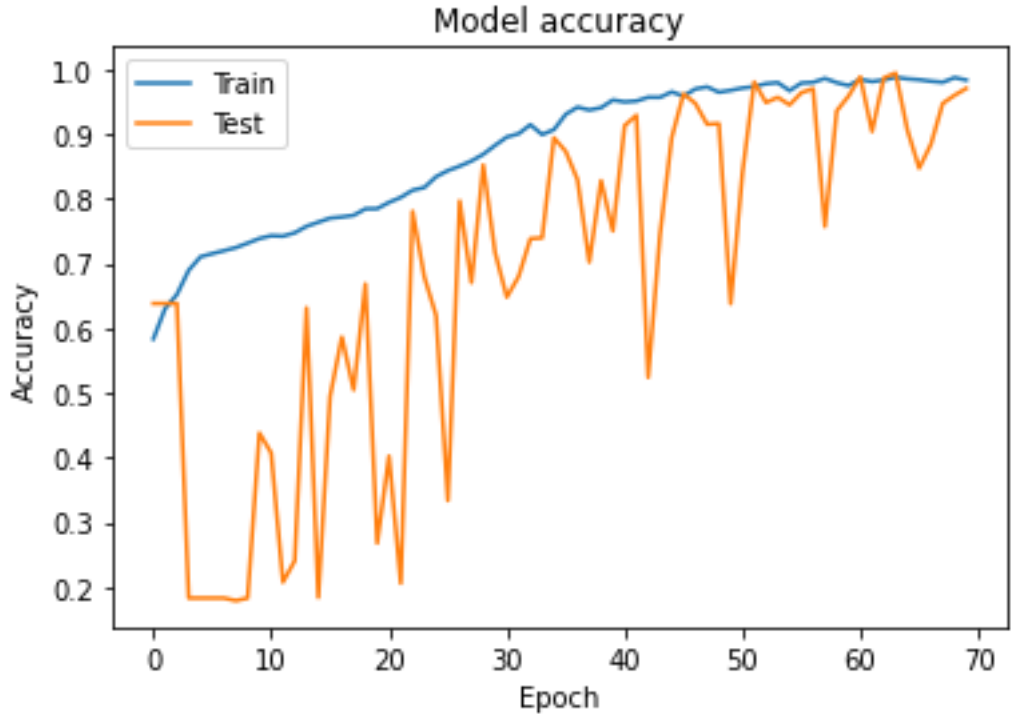
D. Cải tiến mô hình

2. Quá trình training:

Về cơ bản quá trình training mô hình Resnet50 gần như tương tự với mô hình Resnet8. Mô hình được train với epoch=70 và batchsize=64



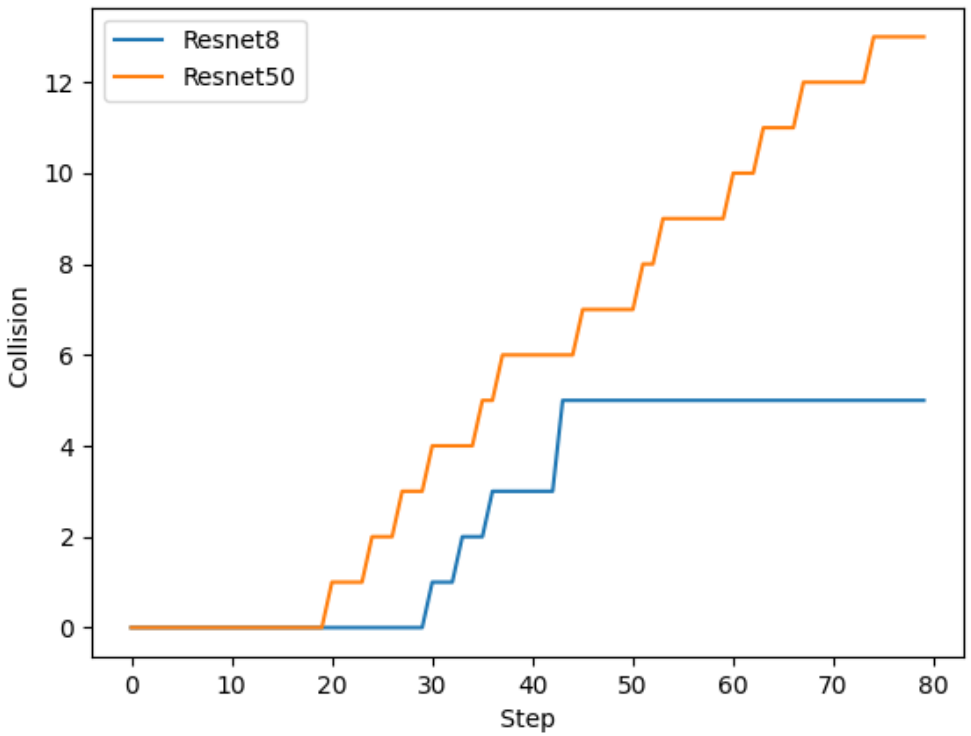
Đồ thị hàm loss



Đồ thị hàm accuracy

D. Cải tiến mô hình

3. So sánh 2 mô hình:



Biểu đồ va chạm của 2 mô hình qua 80 bước

Nhìn vào biểu đồ biểu thị va chạm của 2 mô hình Resnet8 và Renset50 được tính trung bình qua 10 lần đo ta thấy Resnet8 va chạm ít hơn 1 nửa so với Resnet50

Nguyên nhân đề xuất:

- Mô hình Resnet8 xây dựng tối ưu hơn
- Mô hình Resnet50 quá phức tạp so với yêu cầu bài toán với số lượng layers lớn hơn nhiều dẫn đến quá trình kết quả training chưa tốt, khi vào bài toán nhận dạng kém chính xác hơn

Kết luận: Mô hình Resnet8 hoạt động tốt hơn trong bài toán này

E. Định hướng phát triển

Các phương án phát triển dự án: Mục tiêu là đưa số va chạm về 0 và tăng tốc đô xử lý

- Thay đổi bộ data đa dạng hơn
- Nâng cao code, đào sâu hơn các tính năng của airsim
- Ứng dụng thuật toán tối ưu hơn, kết hợp thêm các thuật toán khác
- Xây dựng thêm môi trường để test model

HUST

hust.edu.vn fb.com/dhbkhn

Thanks for watching