ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

MÔN THỊ GIÁC MÁY TÍNH NÂNG CAO

Đề tài: Nhận diện làn đường

GVHD: ThS. Mai Tiến Dũng

Nhóm sinh viên thực hiện:

1. Nguyễn Phi Long MSSV: 19521791

Tp. Hồ Chí Minh, 12/2022

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

……, ngày…….....tháng……năm 20…

Người nhận xét

(Ký tên và ghi rõ họ tên)

MỤC LỤC

I. GIỚI THIỆU BÀI TOÁN

Phát hiện làn đường là một thành phần quan trọng của hệ thống xe tự hành. Các giải pháp cảnh báo đi chệch làn đường trên đường cao tốc đã xuất hiện trên thị trường từ giữa những năm 1990. Tuy nhiên, việc cải thiện và tổng quát hóa các khả năng phát hiện làn đường dựa trên thị giác máy tính vẫn là một nhiệm vụ đầy khó khăn thách thức cho đến thời gian gần đây.

Quá trình phát hiện làn đường (lane detection) trong hệ thống hỗ trợ lái xe được thiết kế để cung cấp vị trí và hướng rẽ của chiếc xe trong phạm vi làn đường xe tham gia và cung cấp chi tiết về cấu trúc đường (kích thước đường hiện tại, độ cong..) và từ đó có thể suy ra một hệ thống tham khảo để giúp định vị xe hoặc chướng ngại vật khác trong đường đi của chiếc xe đó. Mặc dù một số hệ thống đã được thiết kế và phát triển để làm việc trên những con đường hoàn toàn khác nhau, phát hiện làn đường nói chung đã được giảm đến mức các tính năng cụ thể, chẳng hạn như đánh dấu làn đường và vẽ trên mặt đường xuất ra.

Ô tô tự hành (AVG) là một dạng máy thông minh có đủ khả năng hiểu biết để tự xác định trạng thái chuyển động của mình dựa trên những điều kiện môi trường. Một dạng tiêu biểu của ô tô hay robot tự hành là khả năng vận hành và chuyển động không cần sự điều khiển của con người. Các kỹ thuật dựa trên học tập sâu đã được sử dụng trong phát hiện làn đường trong thập kỷ qua. Kết quả thu được ít phải điều chỉnh thông số và những thứ thủ công so với các kĩ thuật thị giác máy tính thông thường, và phương pháp học sâu phổ biến để phát hiện làn đường là mạng nơ ron tích chập (CNN).

1. Mạng nơ-ron tích chập (CNN)

CNN được viết tắt của Convolutional Neural Network hay còn được gọi là mạng nơ-ron tích chập, là một trong những mô hình Deep learning tiên tiến và được sử dụng rộng rãi trong các bài toán xử lý hình ảnh và video. CNN được thiết kế để xử lý các biến thể khác nhau của cùng một đối tượng nhất định, ví dụ như những đối tượng có thể xuất hiện ở các vị trí khác nhau trong hình ảnh. Điều này có nghĩa là CNN có khả

năng phát hiện được các đối tượng trong hình ảnh một cách chính xác hơn so với các mô hình khác.

Ngoài ra, CNN còn có khả năng giảm thiểu số lượng tham số cần học trong quá trình huấn luyện mô hình bằng cách sử dụng các lớp tích chập và pooling. Điều này giúp giảm thiểu đáng kể việc overfitting và tăng tốc độ huấn luyện mô hình.

II. CÁCH TIẾP CẬN BÀI TOÁN

1. Khái quát

1.1 Xác định yêu cầu của bài toán

- Nhận diện làn đường là một bài toán phân loại, vì mô hình trong bài toán này cần phải xác định vị trí của làn đường trên hình ảnh hoặc video, đánh dấu nó bằng một nhãn. Tuy nhiên, cũng có thể coi là một bài toán định vị đối tượng, vì mô hình cần phải xác định vị trí cụ thể của làn đường trên hình ảnh.

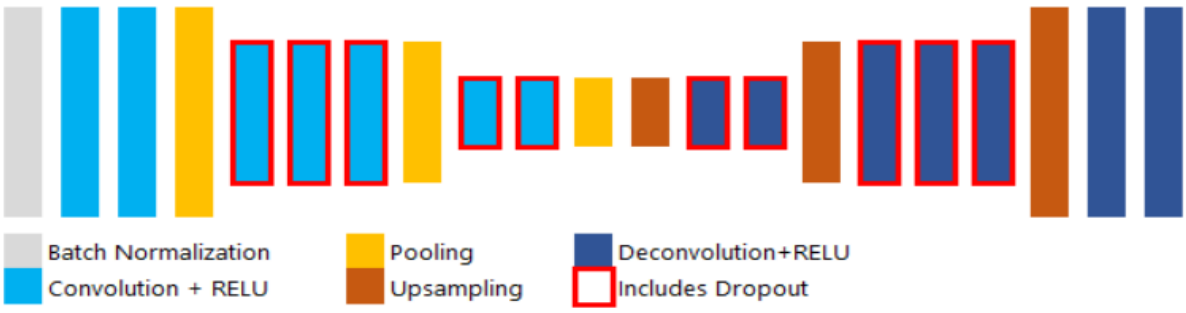
- Đối tượng trong bài toán là làn đường trên hình ảnh hoặc video.

- Đầu vào của mô hình là một hình ảnh có kích thước (80, 160, 3), đầu ra là một binary mask (80, 160, 1) cho biết vị trí của làn đường trong ảnh đầu vào.

- Để đánh giá hiệu suất của mô hình nhóm em sử dụng các độ đo như : Accuracy, precision, recall, f1, IoU, MSE.

- Mục đích của đồ án là xây dựng một mô hình máy học để phát hiện và định vị làn đường trên hình ảnh đầu vào.

1.2 Kiến trúc mạng



Hình 1. Kiến trúc mạng đề xuất.

Hình 2. Ảnh đầu vào.

Đầu vào của mạng là frame với kích thước (80, 160, 3) trong video quay từ camera hành trình trên xe ô tô. Sau đó, trong giai đoạn đầu nó sẽ được tính toán bởi 2 lớp tích chập (con2d, con2d_1) và 1 lớp tổng hợp tối đa (max_pooling2d). Kích thước của bộ lọc trong lớp tích chập là 3 x 3, bước nhảy là 1 và số lượng bộ lọc là 16. Hàm kích hoạt sử dụng trong lớp tích chập là ReLU. Để giảm thiểu số lượng tính toán và hiện tượng overfitting, lớp tổng hợp tối đa được sử dụng với cửa sổ trượt có kích thước 2 x 2. Do đó kích thước của feature map sẽ giảm đi một nửa khi đi qua lớp tổng hợp tối đa. Đầu ra của giai đoạn một sẽ được sử dụng cho đầu vào của giai đoạn hai. Sự khác biệt giữa hai giai đoạn này là số lượng bộ lọc của giai đoạn hai là 32 và lớp tích chập tăng lên 3. Sau đó tiếp tục tính toán cho giai đoạn ba và bốn. Số lượng bộ lọc của lớp tích chập thay đổi từ 64, 128.

Sau khi đi qua các lớp tích chập và các lớp tổng hợp tối đa, kích thước của ảnh đặc trưng được trích xuất từ ảnh làn đường ban đầu đã nhỏ đi rất nhiều và để có thể hiển thị ảnh đặc trưng được trích xuất, chúng ta sẽ cần đến lớp giải chập. Đầu tiên cho ảnh đặc trưng qua Upsampling 2D layer, lớp này tương tự như hàm resize với kích thước lớn hơn ảnh input trong opencv bằng cách copy các giá trị pixel liền kề theo các window size. Sau đó nó sẽ đi qua lớp tích chập chuyển vị (Transposed Convolution), có thể coi tích chập chuyển vị là một quá trình ngược của tích chập thông thường khi mỗi một đặc trưng (feature) được mapping sang các pixels ảnh thay vì ngược lại từ các pixels sang đặc trưng (feature). Lớp Dropout được thêm vào để giảm thiểu vấn đề overfitting bằng cách loại bỏ ngẫu nhiên một số nơ ron trong mạng, trong quá trình huấn luyện thì lớp Dropout sẽ đặt 20% giá trị của tensor đầu vào thành 0 với dropout_rate = 0.2, ngoài ra

các giá trị được đặt thành 0 là ngẫu nhiên. Đầu ra của mạng là một binary mask cho biết vị trí của làn đường được phát hiện trong ảnh với kích thước là (80, 160, 1).

2. Chi tiết

2.1. Tập dữ liệu

- Nhóm em sử dụng tập dữ liệu và tập nhãn đi kèm với mô hình pre-trained của tác giả Michael Virgo.

- Có tổng cộng 12,764 ảnh, trong đó:

* Theo thời tiết:

+ Trời có mây vào buổi chiều: 8424 ảnh.

+ Trời có mưa vào buổi sáng: 2042 ảnh.

+ Trời quang vào buổi tối: 2298 ảnh.

* Theo độ cong của đường:

+ Đường thẳng: 3446 ảnh.

+ Đường cong vừa phải: 3829 ảnh.

+ Đường cong nhiều: 5489 ảnh.

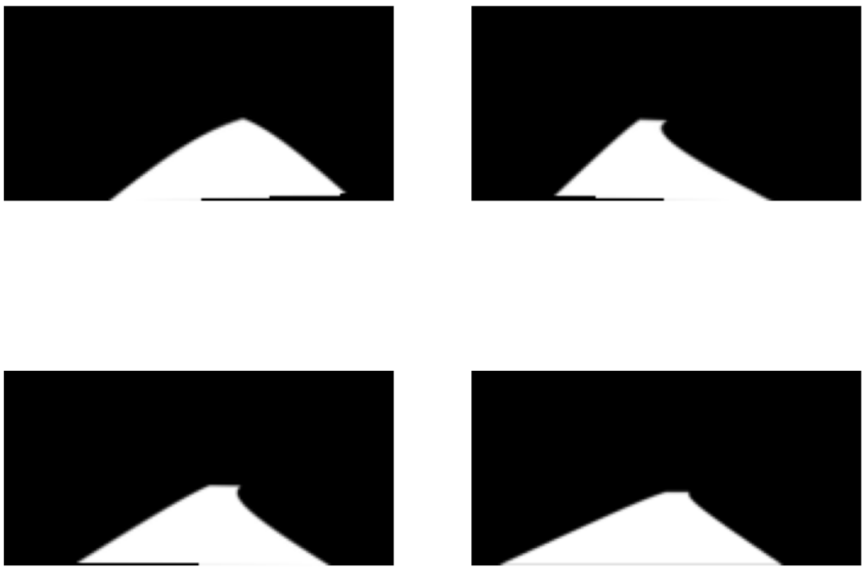
Hình 3. Một vài hình ảnh từ tập dữ liệu.

Hình 4. Thống kê tập dữ liệu (Trích từ báo cáo của tác giả).

2.2 Tập nhãn

- Tập nhãn là các binary mask (80, 160, 1), gồm có 12,764 nhãn tương ứng 12,764 ảnh.

- Vùng màu trắng là các điểm dữ liệu được gán nhãn là đối tượng (đối tượng được quan tâm ở đây chính là làn đường), còn vùng màu đen là các điểm được gán nhãn không phải là đối tượng.



Hình 5. Một vài hình ảnh trong tập nhãn.

- Để huấn luyện mô hình, nhóm em chia tập dữ liệu thành hai phần trainning và validation, training 90% (11487 ảnh) và validation 10% (1277 ảnh).

2.3 Tiền xử lý

- Tập dữ liệu đã được giảm kích thước về (80, 60, 3) vì kích thước ảnh gốc to dẫn đến quá trình xử lý trong mạng lâu và mất nhiều chi phí tính toán.

2.4 Các lớp trong mô hình

2.4.1 Lớp tích chập (Convolution layer)

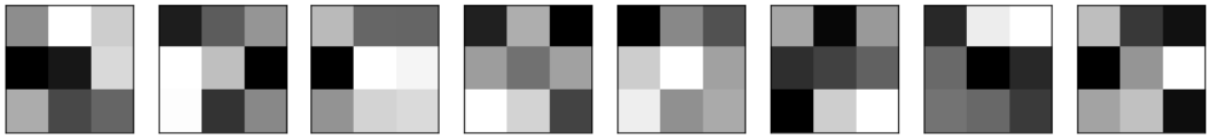
- Đầu vào của lớp tích chập là tensor, đầu ra là feature map, mục đích của lớp này là trích xuất đặc trưng của ảnh.

- Trong đó, tensor là 1 khái niệm chung cho một mảng số có nhiều chiều. Nó có thể hiểu như là ma trận nhiều chiều, và có thể được sử dụng để biểu diễn các dữ liệu có cấu trúc khác nhau. Trong mô hình này tensor có kích thước (11487, 80, 160, 3), 1 tập hợp ảnh có kích thước 80 x 160 với 3 kênh màu và có 11,487 ảnh trong tập hợp đó.

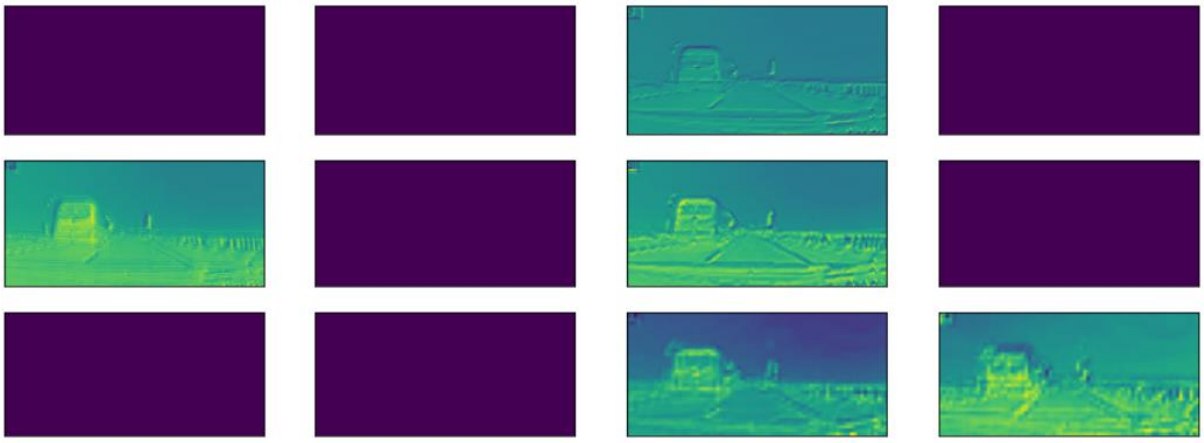
- Feature map là một dạng đặc biệt của tensor biểu diễn các đặc trưng của dữ liệu đầu vào. Feature map có nhiều kênh, mỗi kênh sẽ biểu diễn một đặc trưng khác nhau. Ví dụ feature map (77, 157, 16) là một tập hợp các đặc trưng có kích thước 77 x 157 với 16 đặc trưng khác nhau.

- Sau khi nhân tích chập dữ liệu đầu vào với các bộ lọc, các giá trị sẽ đi qua hàm kích hoạt ReLU để loại bỏ các giá trị nhỏ hơn 0 (để tránh vấn đề vanishing gradient) và được mapping sang feature map.

Hình 6. Ảnh đầu vào.



Hình 7. Một số bộ lọc.



Hình 8. Một số feature map.

- Kích thước của ảnh đầu vào là (80, 160, 3), sau khi đi qua 2 lớp tích chập kích thước giảm xuống (74, 154, 16). Kích thước bị giảm xuống là vì lớp tích chập sử dụng các bộ lọc (còn được gọi là kernel) để tích chập với dữ liệu đầu vào. Mỗi lần tích chập, bộ lọc sẽ được áp dụng trên một phần nhỏ của dữ liệu đầu vào, và sẽ trả về một giá trị duy nhất cho phần đó. Kích thước của dữ liệu sau khi đi qua lớp tích chập sẽ giảm xuống so với kích thước ban đầu vì số lượng giá trị trả về sẽ ít hơn số lượng giá trị đầu vào.

- Điều này có thể dễ hiểu hơn bằng cách xem lớp tích chập như một bộ lọc cho dữ liệu đầu vào. Bộ lọc này sẽ loại bỏ một số thông tin không cần thiết hoặc không quan trọng từ dữ liệu đầu vào, và chỉ giữ lại những thông tin quan trọng. Do đó, dữ liệu sau khi đi qua lớp tích chập sẽ ít hơn dữ liệu ban đầu.

- Kích thước của feature map sẽ được tính như sau:

Output_shape = ((input_shape – filter_size + 2 * padding) / stride + 1)

Trong đó:

+ Output_shape: Kích thước đầu ra

+ Input_shape: Kích thước đầu vào

+ filter_size: kích thước của bộ lọc

+ padding: kích thước của viền

+ stride: Bước nhảy

2.4.2 Lớp tổng hợp tối đa (MaxPooling 2D)

- Đầu vào là 1 feature map hoặc là 1 tensor, đầu ra là 1 tensor, lớp này có nhiệm vụ giảm kích thước của dữ liệu đầu vào nhưng vẫn giữ được các đặc trưng.

- Nó hoạt động bằng cách trượt một của sổ trên dữ liệu đầu vào và lấy giá trị lớn nhất trong cửa sổ đó làm đầu ra cho vị trí hiện tại. Kích thước của dữ liệu sau khi đi qua lớp tổng hợp tối đa sẽ giảm xuống so với kích thước ban đầu vì số lượng giá trị trả về sẽ ít hơn số lượng giá trị đầu vào.

- Mục đích của việc giảm chiều dữ liệu là giảm độ phức tạp tính toán và cải thiện khả năng khái quát hóa, việc giảm kích thước của dữ liệu đầu vào có thể giảm số lượng tham số trong mô hình và lượng tính toán cần thiết để xử lý dữ liệu, ngoài ra còn giảm hiện tượng overfitting và mô hình có thể tìm hiểu các mẫu chung hơn trong dữ liệu có thể áp dụng cho nhiều tình huống hơn. Kích thước dữ liệu đầu vào là (74, 154, 16), tensor đầu ra có kích thước (37, 77, 16).

- Kích thước của dữ liệu đầu ra sẽ được tính như sau:

Output_shape = ((input_shape – pool_size + 2 * padding) / stride + 1) (*Nguồn tham khảo*)

Trong đó:

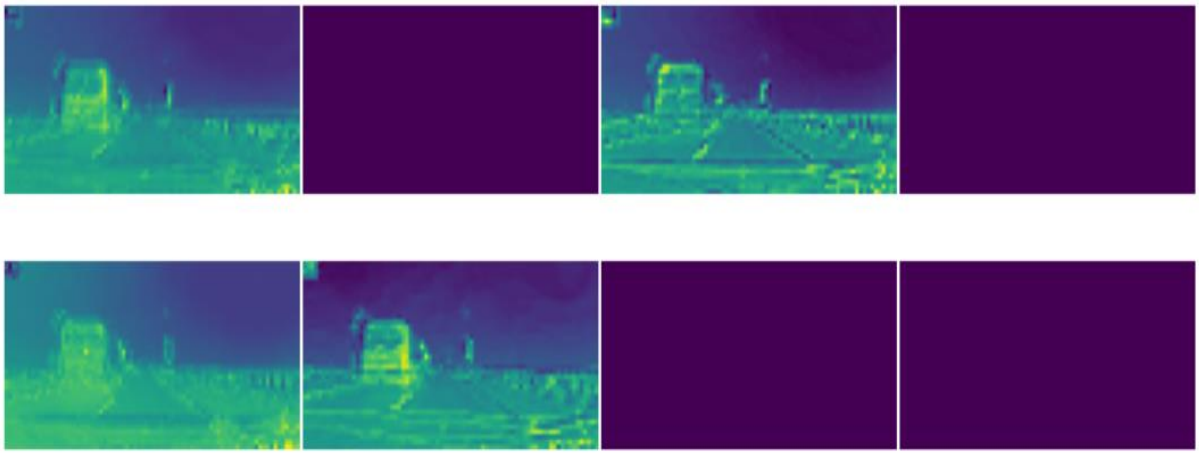
+ Output_shape: Kích thước đầu ra

+ Input_shape: Kích thước đầu vào

+ pool_size: kích thước của bộ lọc

+ padding: kích thước của viền

+ stride: bước nhảy ở đây sẽ bằng pool_size để cửa sổ trượt không bị trùng lên nhau.



Hình 9. Kết quả đầu ra của lớp tổng hợp tối đa.

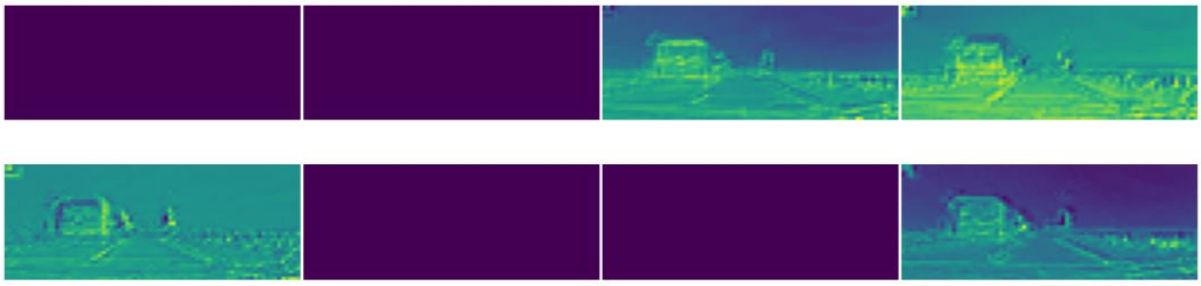
2.4.3 Lớp bỏ qua (Dropout layer)

- Đầu vào là 1 tensor, đầu ra là 1 tensor có cùng kích thước và số chiều với dữ liệu đầu vào.

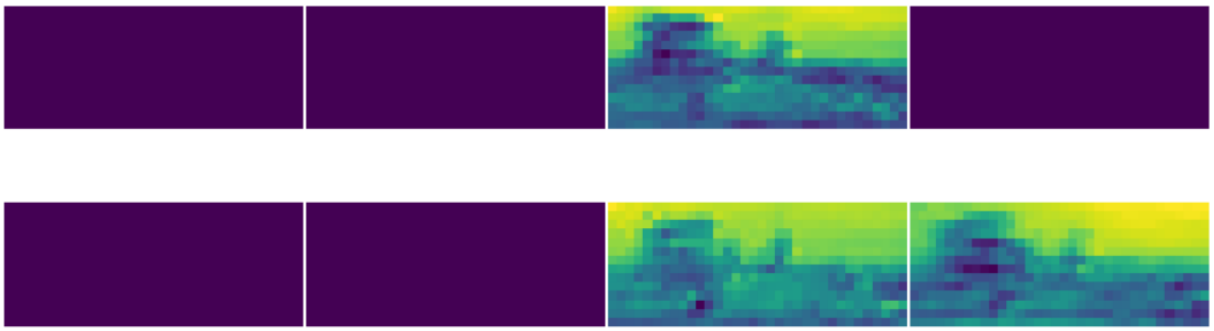
- Lớp Dropout được thêm vào để giảm thiểu vấn đề overfitting bằng cách loại bỏ ngẫu nhiên một số nơ ron trong mạng, trong quá trình huấn luyện thì lớp Dropout sẽ đặt 1 nhóm các giá trị đầu vào thành 0, tỉ lệ các nhóm giá trị đặt thành 0 sẽ dựa vào dropout_rate.

- Overfitting là hiện tượng xảy ra khi mô hình được huấn luyện quá tốt với tập huấn luyện nhưng không tốt trên tập kiểm thử hoặc tập dữ liệu thực tế. Hiện tượng này xảy ra khi mô hình quá phức tạp và có quá nhiều trọng số, dẫn đến việc mô hình học quá nhiều chi tiết về tập huấn luyện mà không có khả năng khái quát hóa tốt cho các dữ liệu mới.

- Trong quá trình huấn luyện mô hình, nhóm em đặt dropout_rate là 0.2, nghĩa là 20% các nhóm giá trị đầu vào sẽ trở thành 0.



Hình 10. Kết quả đầu ra của lớp dropout.



Hình 11. Kết quả đầu ra của lớp dropout.

2.4.4 Lớp tăng mẫu (Upsample layer)

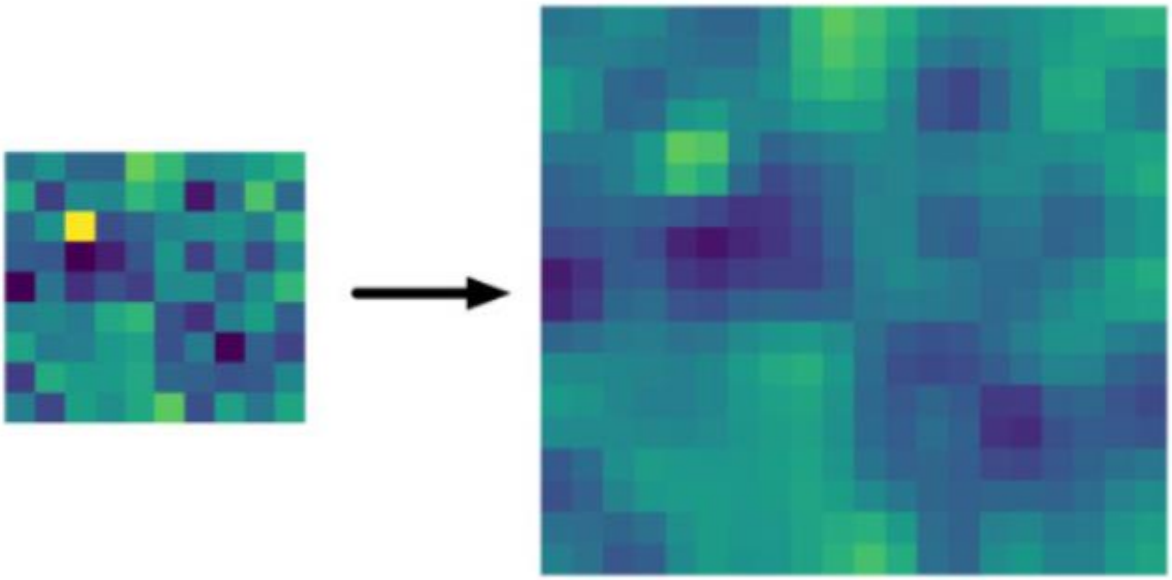
- Đầu vào là 1 tensor có kích thước (6, 16, 64), đầu ra là 1 tensor có kích thước (12, 32, 64).

- Lớp tăng mẫu được sử dụng để tăng kích thước của dữ liệu đầu vào của mô hình,sau khi mô hình đã tổng hợp được các đặc trưng của hình ảnh bằng cách sử dụng các lớp tích chập và tổng hợp tối đa.

- Lớp Upsample sử dụng một quá trình ngược lại của Max Pooling để tăng kích thước của ma trận đầu vào. Nó lặp lại số dòng và số cột của dữ liệu đầu vào dựa trên pool_size.

Hình 12. Đầu vào của lớp Upsample với kích thước (6, 16, 64).

Hình 13. Đầu ra của lớp Upsample với kích thước (12, 32, 64).



Hình 14. Ví dụ minh họa (Nguồn: Tại đây).

2.4.5 Lớp tích chập chuyển vị (Conv2Dtranspose layer)

- Ngoài lớp tăng mẫu (Upsample) ra nhóm em còn sử dụng thêm lớp tích chập chuyển vị để làm tăng kích thước của ma trận đầu vào.

- Lớp này hoạt động giống như lớp tích chập thông thường nhưng thay vì các pixels ảnh mapping sang các feature map thì các feature map sẽ mapping sang các pixels ảnh. Trong lớp tích chập thông thường, bộ lọc (còn gọi là kernel) được áp dụng trên một phần nhỏ của dữ liệu đầu vào, và sẽ trả về một giá trị duy nhất cho phần đó. Tuy nhiên, trong lớp tích chập chuyển vị, bộ lọc được áp dụng trên toàn bộ dữ liệu đầu vào, và sẽ trả về một ma trận có kích thước khác so với dữ liệu đầu vào. Để tích chập với dữ liệu đầu vào, lớp tích chập chuyển vị sử dụng một bộ lọc chuyển vị (transposed convolutional kernel). Bộ lọc này được xây dựng từ một bộ lọc thông thường bằng cách chuyển vị ma trận của bộ lọc đó. Đầu vào là 1 tensor với kích thước (14, 34, 64), đầu ra là 1 tensor với kích thước (16, 36, 64).

- Cách tính kích thước của dữ liệu đầu ra:

```
new_rows = ((rows - 1) * strides[0] + kernel_size[0] - 2 * padding[0] +
output_padding[0])
new_cols = ((cols - 1) * strides[1] + kernel_size[1] - 2 * padding[1] +
output_padding[1])
```

Hình 15. Công thức tính (Nguồn).

Trong đó:

+ new_rows: Chiều cao của output tensor

+ new_cols: Chiều rộng của output tensor

+ rows: Chiều cao của input tensor

+ cols: Chiều rộng của input tensor

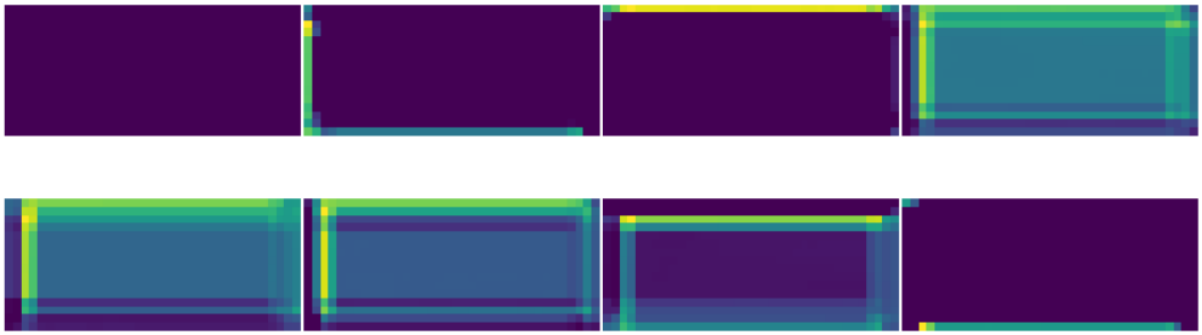
+ stride: Bước nhảy

+ kernel_size: Kích thước của bộ lọc

+ padding: Kích thước viền đầu vào

+ output_padding: Kích thước viền đầu ra

Hình 16. Tensor đầu ra.



Hình 17. Tensor đầu ra.

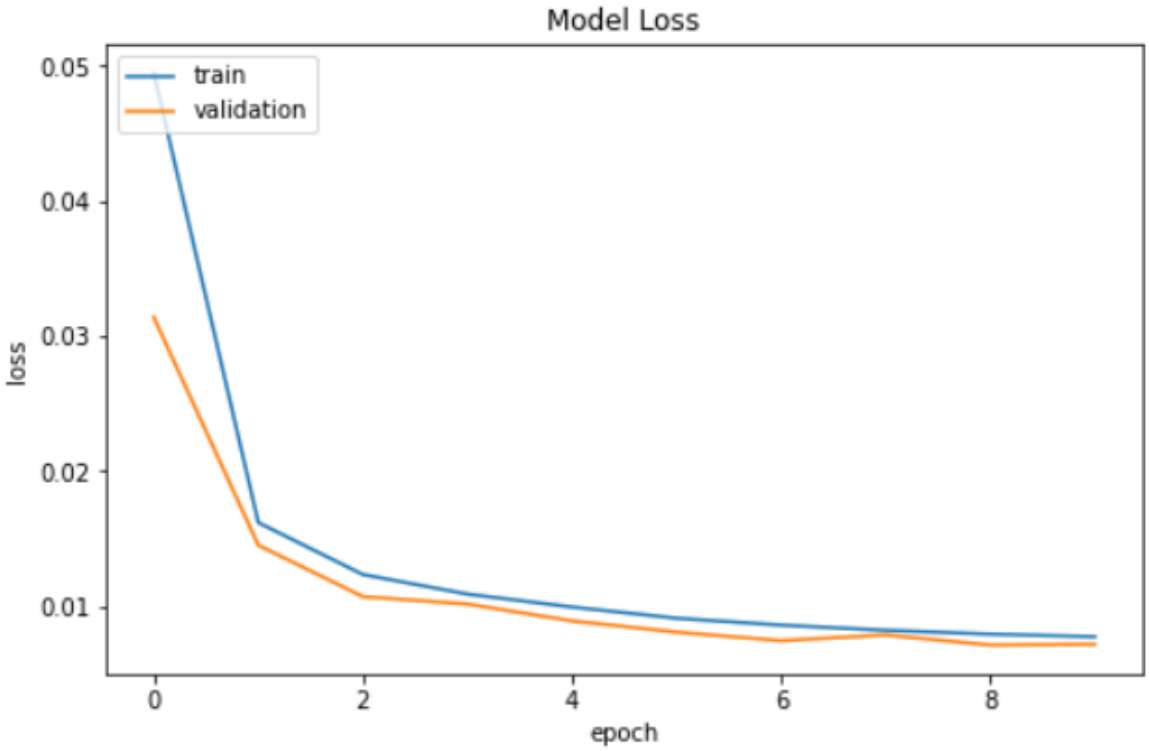
2.5 Huấn luyện mô hình

- Kích thước của dữ liệu đưa vào mô hình (80, 160, 3), số lượng mẫu trong mỗi lần huấn luyện là 128 (batch size), mạng được huấn luyện 10 lần (epochs) và kích thước của cửa sổ trượt trong lớp tổng hợp tối đa là (2 x 2) (pool size). Ngoài ra nhóm còn sử dụng thư viện ImageDataGenerator giúp tải và gắn nhãn các tập dữ liệu hình ảnh.

2.6 Đánh giá các độ đo

2.6.1. Các độ đo

- Sau 10 lần huấn luyện với batch size 128, mô hình kết thúc với MSE cho training là 0.0077 và validation là 0.0072.



Hình 18. Biểu đồ sự thay đổi của Loss function.

Hình 19. Kết quả.

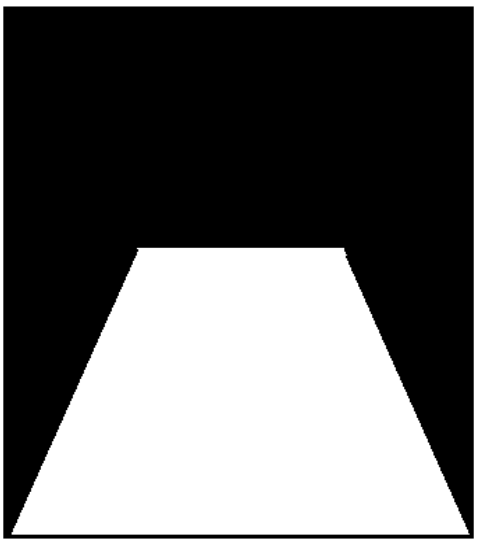
Nhận xét: MSE cho training và validation tại các epochs đầu khá cao, nhưng nó giảm dần theo các epochs và kết quả thu được khá tốt.

- Trên tập validation, độ chính xác (Accuracy) của mô hình lên đến 97.79%, độ đo precision đạt 0.98, độ đo f1 đạt 0.93, độ đo recall và độ đo IoU không được cao, chỉ đạt 0.89 cho recall và 0.87 cho IoU.

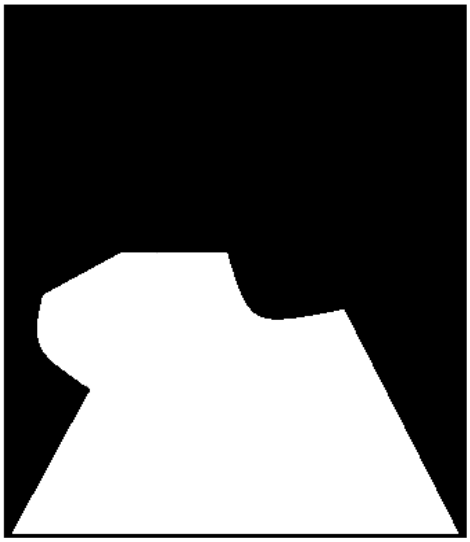
- Để có thể tính các độ đo trên, nhóm em sử dụng confusion matrix dưới đây:

		Predicted	
		Lane	No Lane
Ground-truth	Lane	TP	FN
	No Lane	FP	TN

Bảng 1. Confusion matrix.



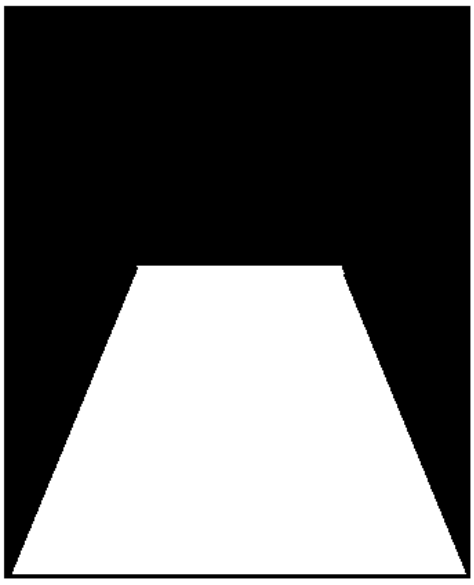
Hình 20. Nhãn thật.



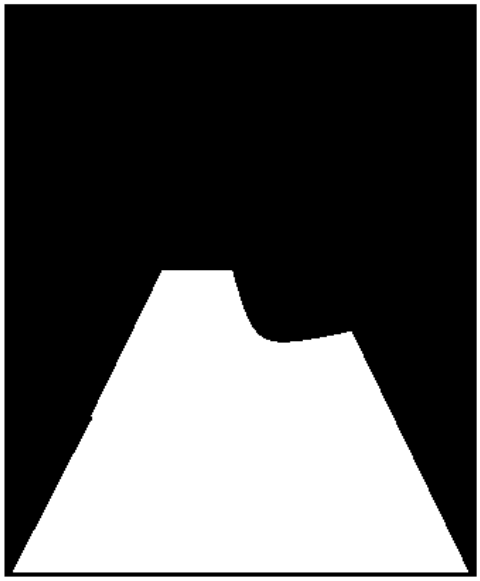
Hình 21. Nhãn dự đoán.

Trong đó:

+ TP: Tổng số lượng điểm dữ liệu được gán nhãn là đối tượng mà thực sự là đối tượng.

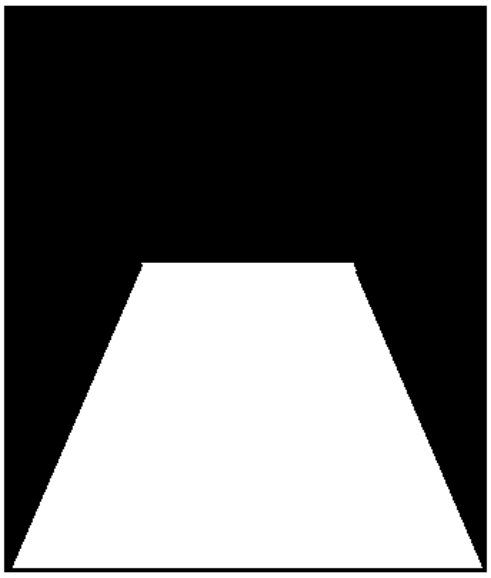


Hình 22. Nhãn thật.

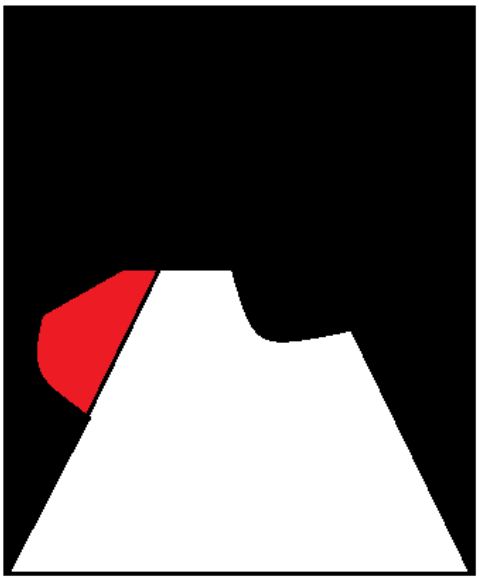


Hình 23. Nhãn dự đoán.

+ FP: Tổng số lượng điểm dữ liệu được gán nhãn là đối tượng nhưng thực sự nó không phải đối tượng.

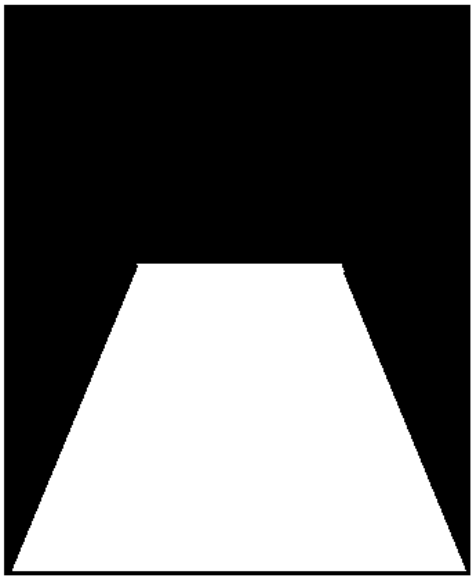


Hình 24. Nhãn thật.

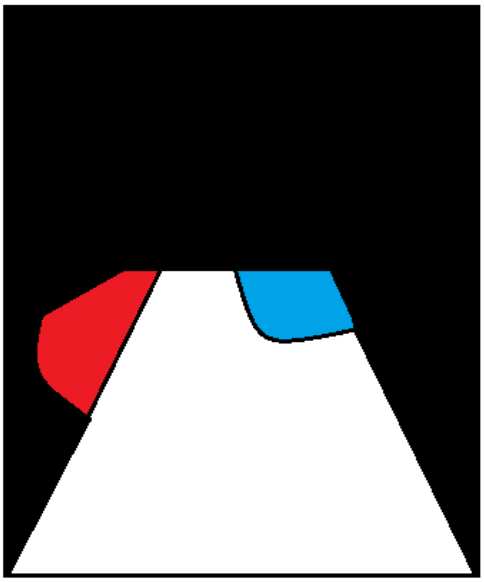


Hình 25. Nhãn dự đoán.

+ FN: Tổng số lượng điểm dữ liệu được gán nhãn không phải là đối tượng nhưng thực sự nó là đối tượng.



Hình 26 Nhãn thật.

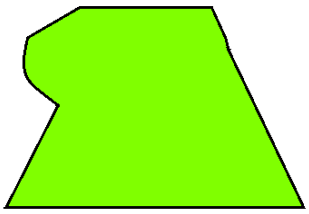


Hình 27. Nhãn dự đoán.

+ TN: Tổng số lượng điểm dữ liệu được gán nhãn không phải là đối tượng và thực sự nó không phải là đối tượng.

- Độ đo IoU:

Intersection



Union

$$IoU = \frac{Intersection}{Union}$$

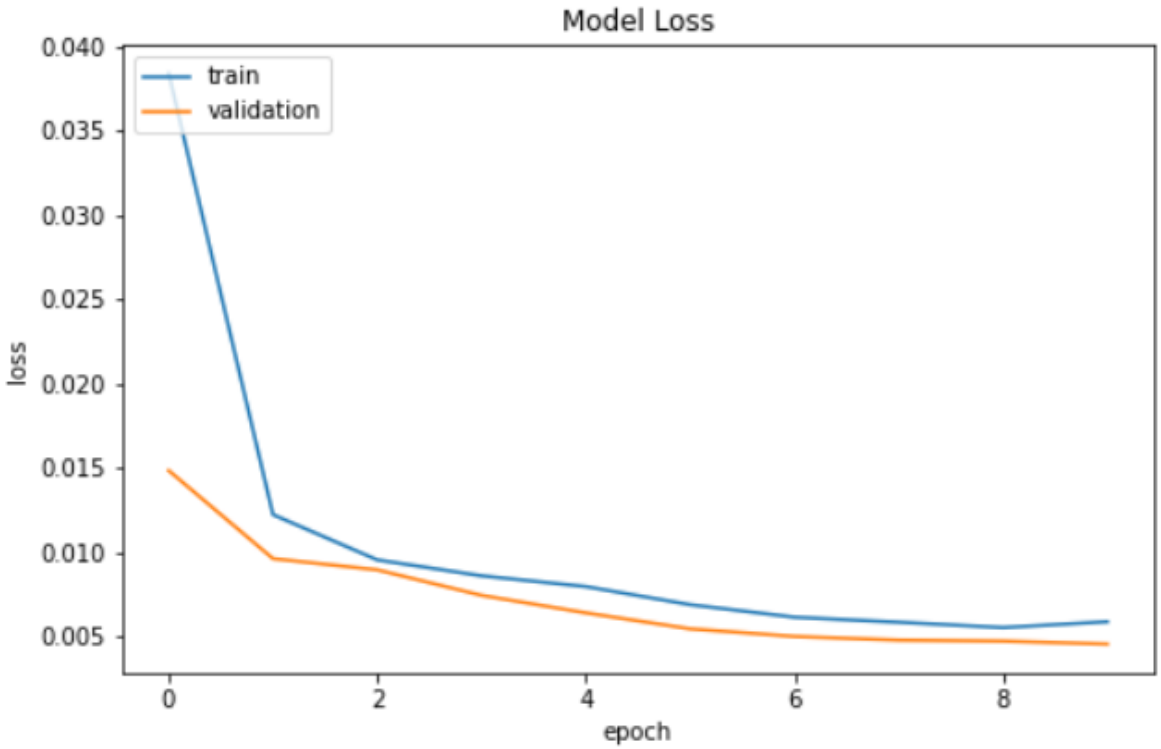
2.6.2 Tuning hyperparameter

- Một vài siêu tham số có thể kể đến như: batch size, kernel size, pool size, stride, dropout rate, epochs, padding.

2.6.2.1 Batch size

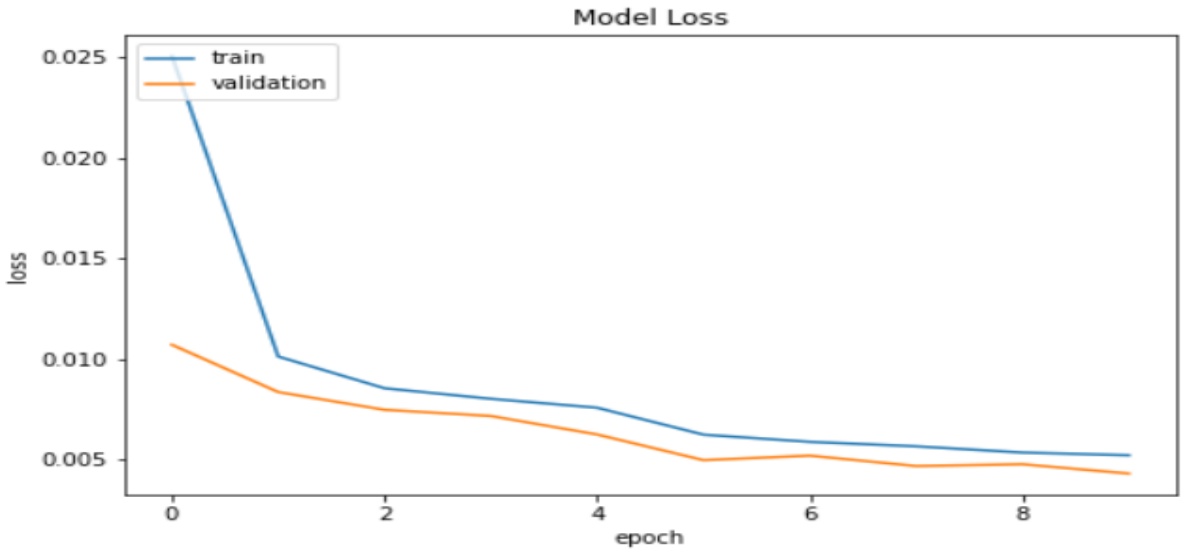
- Batch size ban đầu là 128, MSE cho training và validation: 0.0077 và 0.0072.

- Sau đó nhóm giảm batch size xuống còn 64, MSE cho training và validation: 0.0059 và validation là 0.0045.



Hình 28. Biểu đồ sự thay đổi của Loss function.

- Sau đó nhóm tiếp tục giảm batch size xuống 32, MSE cho training là 0.0052 và validation là 0.0043.



Hình 29. Biểu đồ sự thay đổi của Loss function.

- Như vậy với batch size 128, 64, 32 thì tại batch size 32 cho kết quả tốt nhất.

2.6.2.2 Kernel size

- Kernel size ban đầu là (3 x 3), MSE cho training và validation: 0.0077 và 0.0072

- Đầu tiên nhóm giảm kích thước bộ lọc xuống (2 x 2), với batch size 32, epochs 10, pool size (2 x 2), thu được MSE cho training và validation sau 10 epochs là 0.0072, 0.0064.

Hình 30. Kết quả thu được.

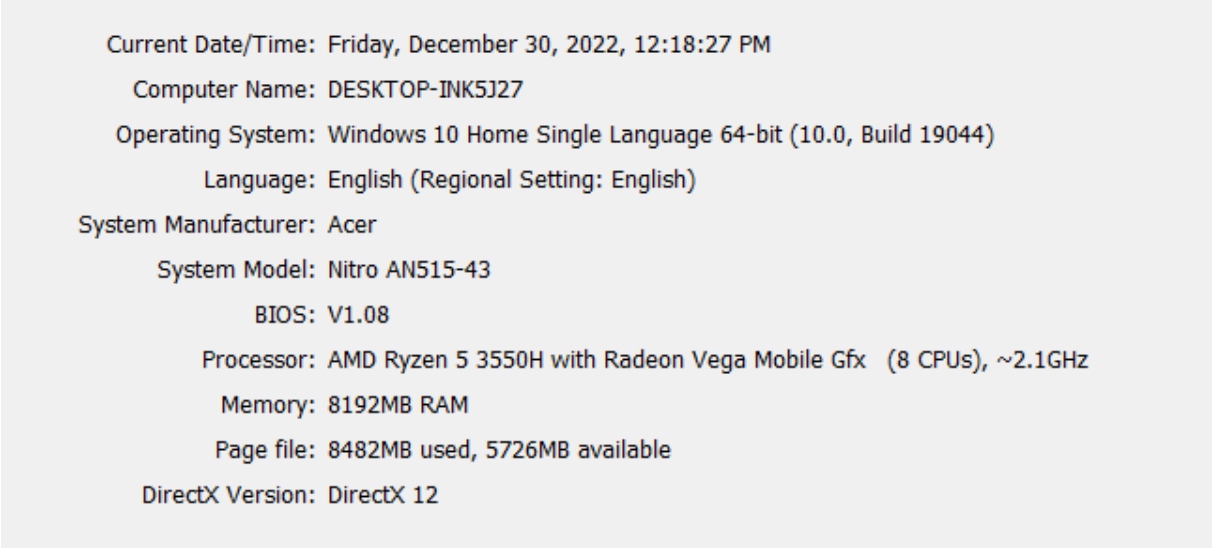
- Tiếp theo nhóm tăng kích thước bộ lọc lên (4 x 4), với batch size 32, epochs 10, pool size (2 x 2), thu được MSE cho training và validation sau 10 epochs là 0.0042, 0.0037.

Hình 31. Kết quả thu được.

- Tiếp theo nhóm tăng kích thước bộ lọc lên (5 x 5), với batch size 32, epochs 10, pool size (2 x 2), thu được MSE cho training và validation sau 10 epochs là 0.0046, 0.0036.

Hình 32. Kết quả thu được.

- Ram của google colab giới hạn 12.68 GB, nhóm em sử dụng run time type là GPU, với tài khoản đầu tiên nhóm em huấn luyện tập dữ liệu được 2 lần sau đó thì bị crash session. Với tài khoản thứ hai nhóm em huấn luyện tập dữ liệu được thêm 2 lần thì bị crash session, đến tài khoản thứ 3 thì chỉ huấn luyện được thêm 1 lần nữa rồi crash session, đến tài khoản thứ 4 thì chỉ huấn luyện được tới epochs thứ 6 sau đó cũng crash session và không thể kết nối đến GPU được nữa. Nhóm quyết định sẽ huấn luyện tiếp trong jupyter notebook. Dưới đây là cấu hình máy để huấn luyện:



Hình 33. Cấu hình máy.

Do quá trình huấn luyện mất quá nhiều thời gian nên nhóm quyết định dừng quá trình điều chỉnh siêu tham số tại kernel size.

III KẾT LUẬN

1. Nhận xét

Qua thuật toán tìm kiếm làn đường trên, bước đầu ta đã có thể phát hiện làn đường một cách nhanh chóng và thuận tiện trong cài đặt, nhưng đôi khi thuật toán vẫn gặp nhiễu, đặc biệt là những làn đường xấu ở Việt Nam và tốc độ thực thi cần phải được cải thiện thêm. Trong thực tế bài toán phát hiện làn đường là một bài toán khó gồm nhiều yếu tố, môi trường có thể ảnh hưởng đến kết quả khác nhau, ví dụ như tùy thuộc vào cách thiết kế, chất lượng đường hay mật độ phương tiện. Thông qua đồ án, nhóm em đã có cái nhìn sơ bộ về việc làm cách nào để phát hiện làn đường, cũng như những kiến thức liên quan đến mạng nơ ron tích chập, sử dụng các kĩ thuật đã được

học vào một chương trình thực tế.

2. Một số hướng phát triển

Trong thời gian tới, nhóm sẽ cố gắng để mô hình có thể hoạt động tốt hơn trong các điều kiện thời tiết khác nhau, đặc biệt là trong điều kiện ánh sáng yếu hoặc là trời mưa. Ngoài ra còn cải thiện tốc độ xử lý để có thể chạy thời gian thực.

TÀI LIỆU THAM KHẢO

(1) mvirgo/MLND-Capstone: Lane Detection with Deep Learning - My Capstone project for Udacity's ML Nanodegree (github.com)

(2) Source code

(3) (dergipark.org.tr)

(4) CS231n Convolutional Neural Networks for Visual Recognition

(5) What is the size of the output of a maxpool layer in a CNN? - Quora

(6) Upsampling in Core ML (machinethink.net)

(7) tf.keras.layers.Conv2DTranspose | TensorFlow v2.11.0

(8) (99+) Lane marking detection using simple encode decode deep learning technique: SegNet | International Journal of Electrical and Computer Engineering (IJECE) - Academia.edu

(9) [Deep Learning] Tìm hiểu về mạng tích chập (CNN) (viblo.asia)

(10) Khoa hoc dữ liệu (phamdinhkhanh.github.io)

(11) Keras Convolutional Layers - w3seo tìm hiểu cơ bản đến nâng cao (websitehcm.com)

(12) How to use the UpSampling2D and Conv2DTranspose Layers in Keras - MachineLearningMastery.com

(13) Ouput_Shape of Feature map