ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO MÔN HỌC

NHẬP MÔN THỊ GIÁC MÁY TÍNH

CS231.M21.KHCL

ĐỀ TÀI: XE TỰ LÁI TRONG GAME BẰNG VIỆC SỬ DỤNG DEEP LEARNING

GIẢNG VIÊN HƯỚNG DẪN: MAI TIẾN DŨNG

SINH VIÊN THỰC HIỆN: NGUYỄN VŨ DƯƠNG

MSSV: 20520465

TP. HỒ CHÍ MINH, 05/2022

LỜI CẢM ƠN

Sau quá trình học tập và rèn luyện tại Trường Đại học Công Nghệ Thông Tin, em đã được trang bị các kiến thức cơ bản, các kỹ năng thực tế để có thể hoàn thành đồ án môn học của mình.
Em xin gửi lời cảm ơn chân thành đến thầy TS. Mai Tiến Dũng – Giảng viên phụ trách lớp CS231.M21.KHCL– Môn Nhập môn Thị giác máy tính đã tận tâm hướng dẫn, truyền đạt những kiến thức cũng như kinh nghiệm cho em trong suốt thời gian học tập.
Trong quá trình làm đồ án môn học, mặc dù em đã cố gắng nhưng chắc chắn sẽ không tránh được những sai sót không đáng có. Mong nhận được sự góp ý cũng như kinh nghiệm quý báu của các các thầy và các bạn sinh viên để được hoàn thiện hơn và rút kinh nghiệm cho những môn học sau.
Em xin chân thành cảm ơn!

TP. Hồ Chí Minh, tháng 5 năm 2022.

MỤC LỤC

DANH MỤC HÌNH

I. Tổng quan về đề tài

1.1. Giới thiệu

- Xe ô tô tự lái (hay còn gọi là xe không người lái) - là một phương tiện ô tô có khả năng cảm nhận môi trường xung quanh và hoạt động mà chỉ cần ít hoặc dường như không cần bất kỳ sự tham gia nào của con người trong việc điều khiển chúng.
- Có rất nhiều cách thức vận hành xe tự lái được chia thành 5 mức độ :
    + Cấp độ 0: Hoàn toàn thủ công.
    + Cấp độ 1: Hỗ trợ lái xe.
    + Cấp độ 2: Tự động hóa một phần lái xe.
    + Cấp độ 3: Tự động hóa có điều kiện.
    + Cấp độ 4: Tự động hóa hoàn toàn trong môi trường kiểm soát.
    + Cấp độ 5: Tự động hóa hoàn toàn.
- Trong đồ án này thì em sử dụng cách vận hành xe tự lái theo cấp độ thứ 3.

1.2. Công dụng

- Việc sử dụng xe tự lái sẽ giúp giảm thiệu đáng kể tai nạn giao thông diễn ra.
- Các phương tiện giao thông tự lái cũng sẽ tuân thủ các quy định về giới hạn tốc độ và biển báo giao thông.

1.3. Mục tiêu và phương pháp

- Vì do xe tự hành là một đề tài khá khó và có nhiều vấn đề và em chỉ muốn làm quen về xe tự lái do đó em sẽ giới hạn lại là làm xe tự vận hành nhưng sẽ không có nhận diện các biển báo giao thông và đèn giao thông và áp dụng nó vào trong game .
- Phương pháp: Em chỉ sử dụng phương pháp phân loại hình ảnh để train model có thể đưa ra quyết định đi.
- Hướng phát triển : Có thể thêm nhận diện vật thể để xe tự hành có thể phát triển lên cấp độ 4 thậm chí là 5.

1.4. Input-Ouput

Em sẽ chia nhỏ đồ án này thành nhiều bài toán nhỏ

- Thứ nhất bài toán grabscreen: dùng để lấy khung hình tọa độ đưa vào.
  + Input : Tọa độ khu vực.
  +Output: Bức hình ở tọa độ đó.
- Thứ hai bài toán get_key : dùng để kiểm tra người dùng nhập phím.

- Thứ ba bài toán direct_key: giúp cho python có thể dùng các phím trong bàn phím để điều khiển game.
- Thứ 4 bài toán xác định vị trí của tựa game
  + Input : Toàn màn hình máy tính
  + Output: Vị trí của tựa game mình cần điều khiển.
- Thứ 5 Tạo dataset
  + Input: Gồm khung hình và sự lựa chọn của người lấy dataset
  + Output: Tập các dữ liệu gồm khung hình + sự lựa chọn của người dùng
- Thứ 6 bài toán cân bằng dữ liệu
  + Input: Tập được lấy ở bài toán thứ 5
  + Output: Tập dataset đã được cân bằng
- Thứ 7 bài toán thành lập model
  + Input : Lượng dataset đã được cân bằng
  + Output: Một model hoàn chỉnh mà có thể giúp cho xe tự vận hành được
- Thứ 8 bài toán giúp cho xe từ model có thể chạy 1 cách trơn tru nhất

Các quá trình sẽ được trình bày cụ thể ở phần sau.

II. Phương pháp từng bài toán

2.1 Bài toán grabscreen

- Bài toán grabscreen này dùng mục đích là dùng để chụp hình ảnh từ vị trí quy định của người dùng.

Các thư viện hỗ trợ : win32gui , win32ui , win32con, win32api,cv2 và numpy.

2.2 Bài toán getkey

- Bài toán getKey giúp cho em có thể kiểm tra được các nút trong bàn phím có nằm trong phím mình đã quy định hay không.

Thư viện hỗ trợ : win32api

Trong đoạn code em sử dụng GetAsyncKeyState() nhận trạng thái khóa không đồng bộ , tức là, không cần chờ đợi bất cứ điều gì, tức là nó sẽ nhận trạng thái của khóa hiện hành đang nhập

2.3 Bài toán direct_key

- Bài toán này dùng với mục đích là em code ra một hàm hỗ trợ cho python có thể điều khiển các nút một cách dễ dàng.

- Tại sao em lại code một hàm ra như vậy mà trong khi đó em có thể sử dụng thư viện có sẵn là PyAutoGUI. Nhưng sau khi em thử và tìm hiểu thì em phát hiện ra thư viện này không phải gửi bàn phím như các trò chơi mong muốn mà là thứ mà trò chơi cần đó là “Trực tiếp”.

- Sau khi tìm hiểu thêm cách thức làm sao code thì em đã tìm ra một phương pháp là sử dụng là tái định dạng lại cấu trúc của C trong python để giúp hoạt động trong game một cách trực tiếp.

- Trong directKey , để tiện cho việc làm python có thể điều khiển game thì em đã thêm 2 hàm đó chính là PressKey() và ReleaseKey(), Trong đó PressKey() dùng để nhấn phím và ReleaseKey() dùng để dừng nhấn phím đó .

Thư viện hỗ trợ: ctypes, time.

2.4 Bài toán xác định vị trí

- Để có thể xác định được vị trí của một cửa sổ nào đó , Em nghĩ đã nghĩ ra 1 phương pháp đó chính là chụp hình màn hình và sau đó sử dùng selectROI trong OpenCV để có được các trọng số x0,y0,w,h .

Trong đó:

+ x0 là vị trí bắt đầu theo trục x.

+ y0 là vị trí bắt đầu theo trục y, w

+ w là độ dài theo trục x của chỗ vị trí mình quan tâm.

+ h là độ cao theo trục y của chỗ vị trí mình quan tâm.

2.5 Bài toán tạo dataset

- Dataset trong bài toán này trong bài này bao gồm : Feature là hình ảnh, Label của dataset là phím nhấn của người dùng.

- Label của bài toán sẽ là gồm 3 kiểu đầu ra đó là các phím ‘A’,’W’,’D’ . Để có thể biểu diễn label của bài toán cho model có thể hiểu thì em sử dụng phương pháp one-hot encoding.

- Em sử dụng 3 phím ‘A’ , ‘W’, ‘D’ là label vì các phím là các phím dùng để điều khiển xe trong game và sẽ thầy thắc mắc tại sao em lại không sử dụng thêm phím ‘S’ là do phím này xuất hiện rất ít trong quá trình em tạo dataset và nếu thêm vào nó sẽ làm cho model sẽ không cân bằng do đó để giải quyết cho vấn đề phím S em sẽ trình bày cách thức để phím ‘S’ hoạt động trong phần testmodel.

+ **One-hot encoding** là quá trình biến đổi từng giá trị thành các đặc trưng nhị phân chỉ chứa giá trị 1 hoặc 0. Mỗi mẫu trong đặc trưng phân loại sẽ được biến đổi thành một vector có kích thước m chỉ với một trong các giá trị là 1 (biểu thị nó là active).

VD : Với Label của tập dataset thì nó gồm 3 loại label nên vector on-hot sẽ có kích thước là 3.

+ Với phím ‘A’ là kết quả thì vector one-hot có dạng là [1,0,0].

+ Với phím ‘W’ là kết quả thì vector one-hot có dạng là [0,1,0].

+ Với phím ‘D’ là kết quả thì vector one-hot có dạng là [0,1,0].

- Đối với feature của bài toán thì bằng việc sử dụng hàm grabscreen (em đã trình bày ở trên) thì em dễ dàng có thể đưa được hình ảnh của tựa game em muốn train vào dataset thế nhưng em phải đưa bức ảnh về dạng trắng đen và resize về kích thước(160,120) vì do quá trình 1 phần là em train model trên collab để collab tránh bị tràn ram và GPU miễn phí do collab cung cấp.
- Để tránh tràn ram trong quá trình lấy dataset thì cứ mỗi 500 cặp feature – label thì em sẽ lưu lại .
- Để tạm tạm dừng quá trình lấy dataset thì nhấn phím ‘T’ và dừng hẳn quá trình lấy dataset thì dùng phím ‘E’.

2.6 Bài toán cân bằng dữ liệu

- Vì trong quá trình lấy dataset từ file lấy dataset ở bài toán 5 , em nhận thấy lượng dataset của bài bị mất cân bằng rất nhiều .

```
Before: Counter({'[0, 0, 1]': 35432, '[1, 0, 0]': 58760, '[0, 1, 0]': 870467})
```

Hình 2.1: Dataset trong quá trình thu nhập.

- Ta có thể thấy được rằng lượng dataset thu nhập được gần 1 triệu bức hình thế nhưng lượng dataset lại bị nghiêng hẳn về phím ‘W’ điều đó là 1 điều không hề tốt cho việc huấn luyện model do đó giải pháp của em đó chính là cân bằng lại lượng dataset này lại sao cho dữ liệu train cho 3 label được cân bằng với nhau.

```
Counter({'[0, 1, 0]': 33110, '[1, 0, 0]': 33110, '[0, 0, 1]': 33110})
```

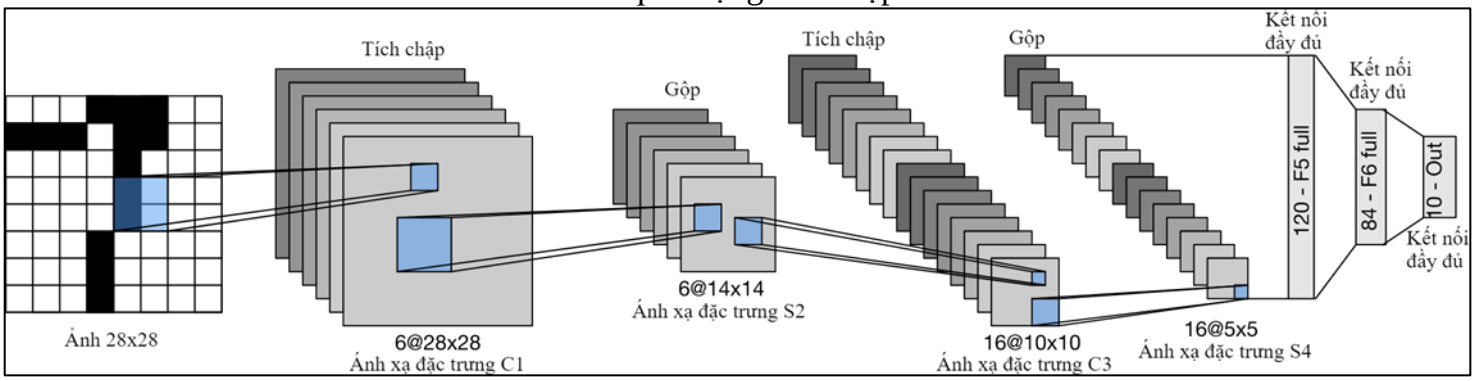
Hình 2.2: Dataset sau khi được cân bằng.

- Một câu hỏi đặt ở đây thì tại sao lượng dataset lại mất cân bằng như thế bởi vì một phần trong game việc lái xe thì để xe luôn đi thẳng thì ta bắt buộc nhấn phím ‘W’ rất nhiều do đó lượng dataset thu nhập từ ‘W’ là rất nhiều .
- Thế nhưng sau khi cân bằng thì ta có thể thấy thì gần 900000 dataset nhấn phím ‘W’ bị biến thành 33110 dataset phím ‘W’ thì liệu dataset có hoạt động

ổn định hay không ? Câu trả lời sẽ là có thì cách khắc phục điều này em sẽ nói trong phần testmodel .

2.7 Bài toán tạo model

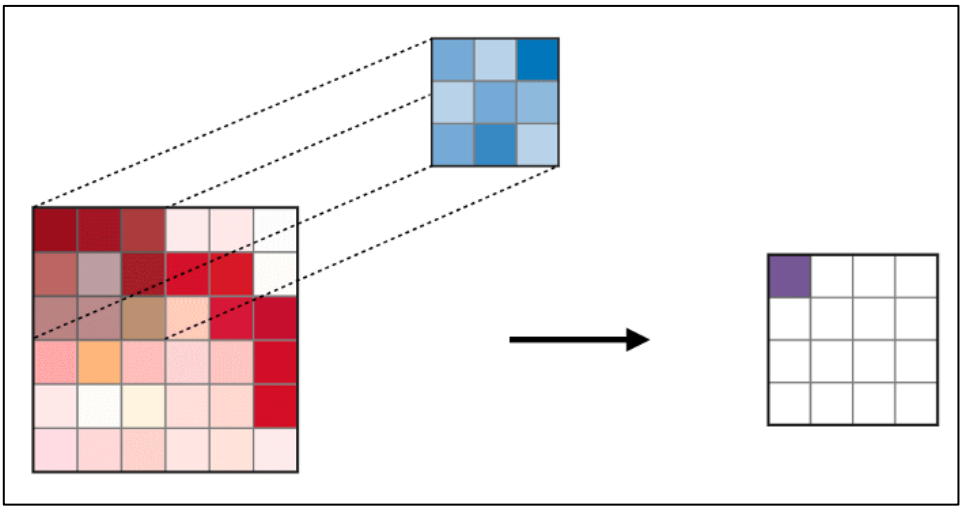
- Vì dataset train trong bài này là gồm 2 thành phần là : bức ảnh là feature để train và label là phím chọn . Với bức ảnh train trong bài này là một bức ảnh chứa rất nhiều thứ trong đó VD: ngôi nhà , con đường, bầu trời , con người , những xe xung quanh , …
- Vì có nhiều thứ xung quanh bức hình do đó không thể sử dụng các phương án thông thường như K-nearest neighbors , Histogram of Oriented Gradient,SVM ,hay cả linear Regression vì độ hiệu quả của nó là rất thấp .
- Do đó phương án em hướng tới đó chính là áp dụng kĩ thuật học sâu (Deep Learning) . Deep Learning là là một thuật toán dựa trên các ý tưởng đến từ não bộ thông qua việc tiếp thu rất nhiều tầng biểu đạt hay trừu tượng để có thể làm rõ nghĩa của các loại dữ liệu.
- Với một lượng dataset lớn và trong đó còn chứa rất nhiều đối tượng trong mỗi bức hình thì em nghĩ ngay phương án sẽ sử dụng ngay Mạng nơ-ron tích chập(CNN- Convolutional Neural Network) vì mạng tích chập xây dựng được những hệ thống thông minh với độ chính xác cao.
- Đầu tiên ta sẽ tìm hiểu sơ qua mạng tích chập :



Hình 2.3: Cấu trúc của mạng tích chập gồm rất nhiều lớp.

+ **Lớp Convolutional** Là một cửa sổ trượt (Sliding Windows) trên một ma trận.

Cách thứ hoạt động của lớp Convolutional:



Hình 2.4: Minh họa cách thức hoạt động của lớp Convolutional.

- Hình bên trái là một bức ảnh đầu vào (đây là ảnh nhị phân gồm 2 màu trắng và đen) được biểu diễn dưới ma trận m*n.
- Hình ở giữa , cửa sổ trượt (sliding window) là một ma trận có kích thước là k*k trong đó k là số lẻ.
- Kết quả đầu ra của một tích chập là một ma trận từ việc trượt ma trận và thực hiện cùng lúc trên toàn bộ ma trận ảnh.

+ **Lớp gộp (Pooling)** sử dụng một cửa sổ trượt quét qua toàn bộ ảnh dữ liệu, mỗi lần trượt theo một số bước trượt (stride) cho trước. Các phương thức phổ biến trong lớp Pooling là Max Pooling (Lấy giá trị lớn nhất) , Min Pooling(Lấy giá trị nhỏ nhất) và và Average Pooling (lấy giá trị trung bình).

Hình 2.5: Minh họa cách thức hoạt động của lớp pooling.

+ **Lớp Local Response Normalization** là lớp chuẩn hoá dựa trên các neuron "hàng xóm" tức thuộc cùng vị trí nhưng ở các kernel lân cận.

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

Hình 2.6: Công thức chuẩn hóa của lớp Local Response Normalization

Trong đó:

n: số output lân cận - đây là một hyper param.

N: số lượng filter của layer này.

aix,y: là neuron output ở vị trí (x,y) sau khi áp filter thứ i k,α,β: là các hyperparam để điều chỉnh quá trình normalize này.

+ **Dense layer hay Fully-connected layer** là một lớp cổ điển trong mạng nơ ron nhân tạo. Mỗi nơ ron nhận đầu vào từ tất cả nơ ron lớp trước đó.

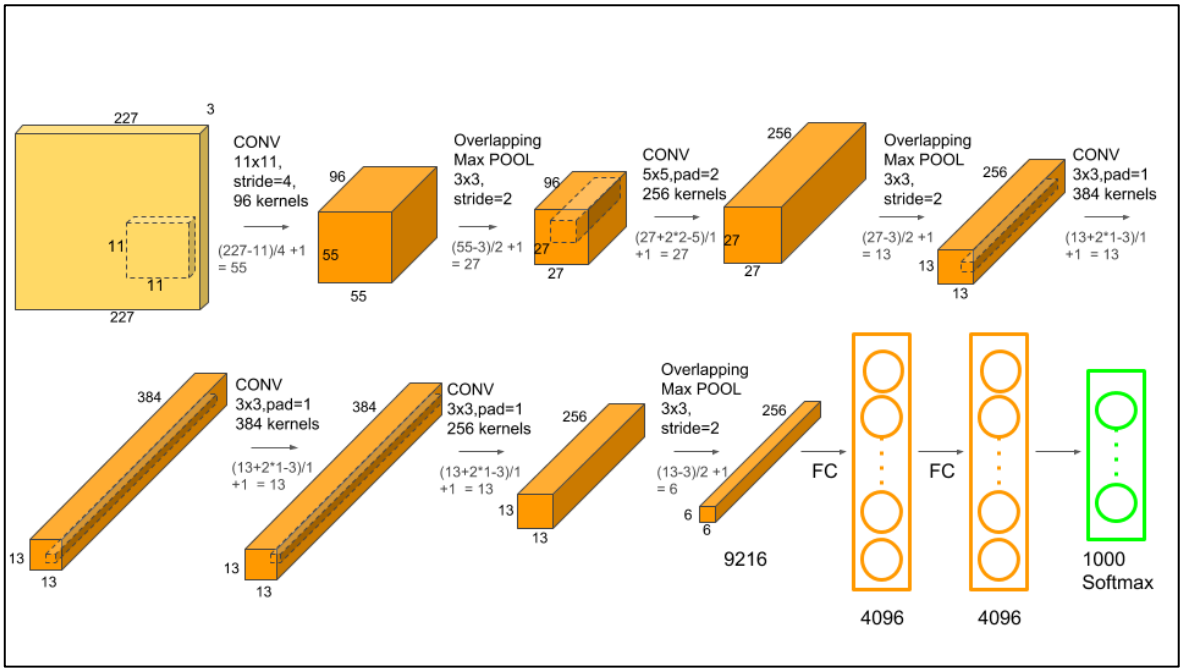
+ **Lớp Dropout** : Hiểu 1 cách đơn giản thì Dropout là việc bỏ qua các đơn vị (tức là 1 nút mạng) trong quá trình đào tạo 1 cách ngẫu nhiên. Bằng việc bỏ qua này thì đơn vị đó sẽ không được xem xét trong quá trình forward và backward. Theo đó, p được gọi là xác suất giữ lại 1 nút mạng trong mỗi giai đoạn huấn luyện, vì thế xác suất nó bị loại bỏ là (1 - p).

Tại sao chúng ta cần lớp Dropout vì để tránh model tránh trường hợp rơi vào trường hợp overfitting(học tủ)

- Để có thể tao ra 1 model từ CNN cho bài toán của em thì với lượng kiến thức em được học thì em chưa đủ khả năng để tạo ra một model nào phù hợp với lượng input đầu vào của em do đó sau khi em tìm hiểu các model nổi tiếng thì em tìm ra một model rất nổi tiếng đó chính là model "alexnet".
- Mạng AlexNet được giới thiệu vào năm 2012, được đặt theo tên của Alex Krizhevsky, tác giả thứ nhất của bài báo đột phá trong phân loại ImageNet Mạng AlexNet bao gồm 8 tầng mạng nơ-ron tích chập, đã chiến thắng cuộc thi ImageNet Large Scale Visual Recognition Challenge năm 2012 với cách

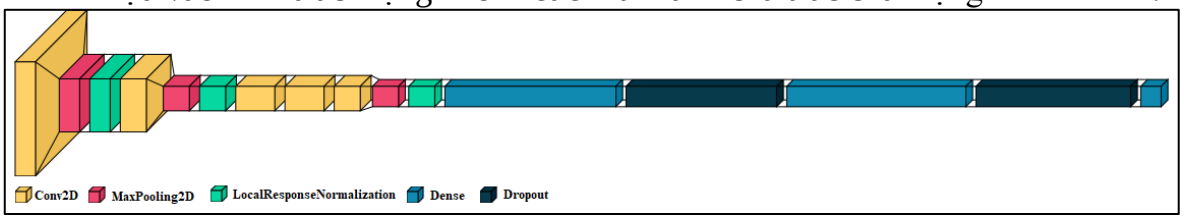
biệt không tưởng. AlexNet lần đầu tiên đã chứng minh được rằng các đặc trưng thu được bởi việc học có thể vượt qua các đặc trưng được thiết kế thủ công, phá vỡ định kiến trước đây trong nghiên cứu thị giác máy tính. AlexNet được giới thiệu là " ông hoàng" trong phân loại hình ảnh.

-

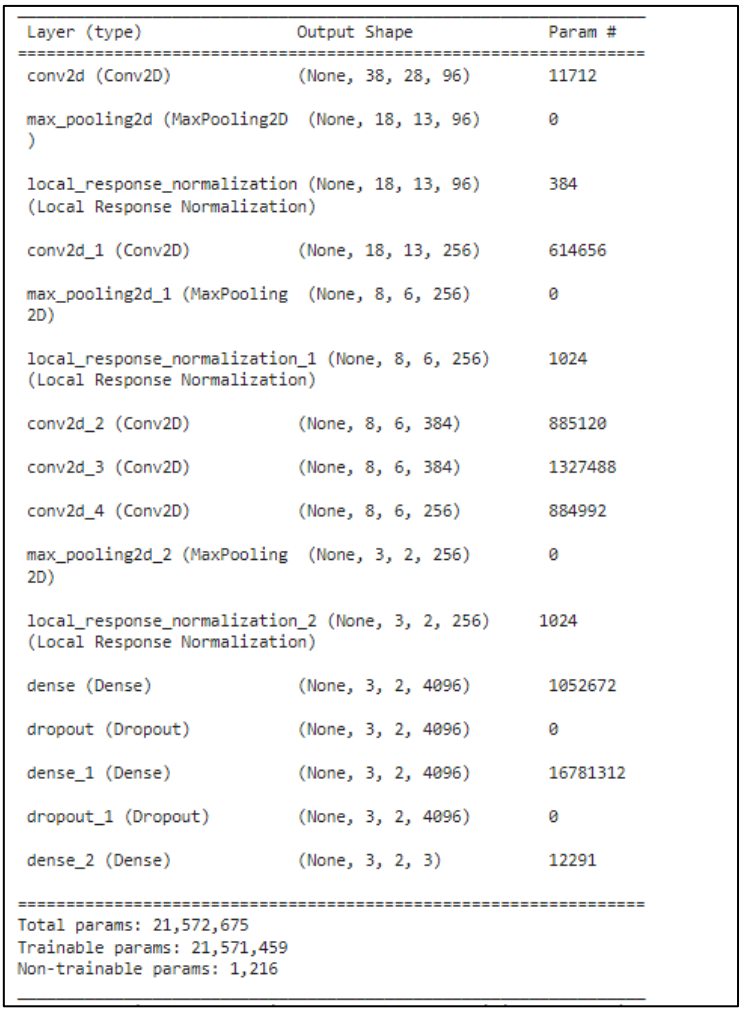


Hình 2.7: Kiến trúc mạng AlexNet

- Dựa vào kiến trúc mạng AlexNet em thiết kế cấu trúc của mạng em như sau:



Hình 2.8: Cấu trúc mạng áp dụng cho đồ án



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 38, 28, 96)	11712
max_pooling2d (MaxPooling2D)	(None, 18, 13, 96)	0
local_response_normalization (Local Response Normalization)	(None, 18, 13, 96)	384
conv2d_1 (Conv2D)	(None, 18, 13, 256)	614656
max_pooling2d_1 (MaxPooling2D)	(None, 8, 6, 256)	0
local_response_normalization_1 (Local Response Normalization)	(None, 8, 6, 256)	1024
conv2d_2 (Conv2D)	(None, 8, 6, 384)	885120
conv2d_3 (Conv2D)	(None, 8, 6, 384)	1327488
conv2d_4 (Conv2D)	(None, 8, 6, 256)	884992
max_pooling2d_2 (MaxPooling2D)	(None, 3, 2, 256)	0
local_response_normalization_2 (Local Response Normalization)	(None, 3, 2, 256)	1024
dense (Dense)	(None, 3, 2, 4096)	1052672
dropout (Dropout)	(None, 3, 2, 4096)	0
dense_1 (Dense)	(None, 3, 2, 4096)	16781312
dropout_1 (Dropout)	(None, 3, 2, 4096)	0
dense_2 (Dense)	(None, 3, 2, 3)	12291

Total params: 21,572,675
Trainable params: 21,571,459
Non-trainable params: 1,216

Hình 2.9: Summary model

2.8 Bài toán tạo run_model

- Với bài toán tạo file để thực thi model nhất cách mượt mà nhất thì em phải giải quyết một số vấn đề sau:
  - 1: Đưa ra có số liệu về thời gian nhấn phím để phù hợp với phần cứng máy tính.
  - 2: Giải quyết về vấn đề phím ‘W’ đề cập ở phần cân bằng dữ liệu.
  - 3: Giải quyết được về hàm tính toán sao cho có thể lùi xe khi đụng vào tường.
  - 4: Đưa ra thông số phù hợp để chăn dưới cho kết quả dự đoán của model.
- Với vấn đề 1: Vì máy tính em sử dụng có không sử dụng card đồ họa rời của nvidia nên không có nhân cuda nên model của em chạy trên CPU do đó để nhẹ máy tính thì em thiết kê như sau :
  - Thời gian chạy của “W” là khoảng 0.012s.
  - Thời gian chạy phím “A”,”D” là 0.034s.

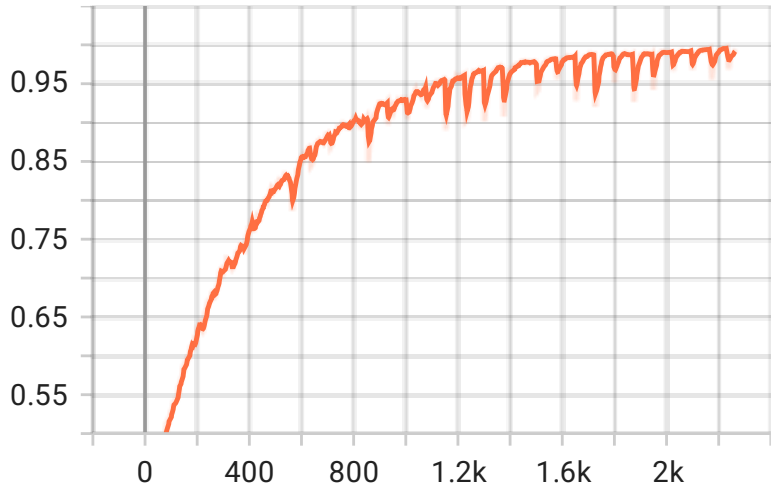
Những thông số em có được như trên là trong quá trình em cho model chạy thử và em rút ra được thông số phù hợp.

- Với vấn đề 2 : Để giải quyết về lượng dataset ‘W’ bị mất em sẽ sử dụng 1 biến toàn cục có tên là AUTO_STRAIGHT , biến này sẽ kết hợp với hai phím là “A” và “D” hay nói 1 cách dễ hiểu là khi biến AUTO_STRAIGHT == True thì nếu model dự đoán là phím “A” thì nó sẽ nhấn cùng lúc 2 phím “W” và phím “A” tại sao em lại sử dụng cách thức này vì do quá trình em thu nhập dataset thì khi em rẽ sang trái hoặc sang phải trong game thì em luôn hay kết hợp cả phím ‘W’ vào nữa và để hạn chế cho chiếc xe bị chạy nhanh quá thì cứ sau 60 khung hình thì em sẽ cho AUTO_STRAIGHT = NOT AUTO_STRAIGHT . Thêm 1 trường hợp em tìm ra được là trong trường hợp AUTO_STRAIGHT = False và số lượng phím “A” , “D” được nhấn lớn hơn 16 (Không có phím “W” được nhấn nếu “W” được nhấn thì số lượng phím đó sẽ dc gán lại = 0) thì em sẽ gán AUTO_STRAIGHT = True.
- Về vấn đề thứ 3: Thì em sẽ tạo ra 1 hàm tính dùng để đo chuyển động . Nó sẽ làm nổi bật bất kỳ chuyển động nào mà nó nhìn thấy.Nó cũng phát hiện số lương. Nó cũng phát hiện số lượng tương đối của chuyển động và tĩnh và chỉ ra những thay đổi đáng kể. Trong thời gian di chuyển, các khung hình riêng lẻ sẽ được lưu lại. Dựa vào đó và đưa ra thông số phù hợp thì ta có thể làm cho model mình lùi khi gặp chướng ngại vật hoặc tường.
- Vấn đề 4: Trong quá trình thực nghiệm sau khi quan sát các thông số dự đoán thì em thấy được với phím ‘W’ thì nếu prediction[‘W’] < 0.6 thì nó không được đúng lắm nên em tạo ra biến chặn dưới có tên là straight_thresh và gán bằng 0.6. Tương tự với 2 phím ‘A’ và ‘D’ thì em cho turn_right_thresh = 0.65 và turn_left_thresh = 0.65.

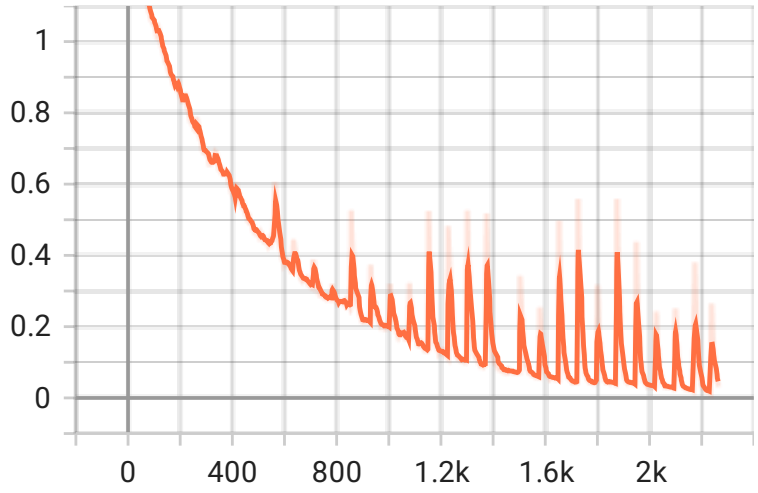
III. Kết quả

- Kết quả này là em dựa trên quá trình train trên tập dataset mà em đã cân bằng.
- Tập dataset bao gồm tất cả 99603 bức ảnh.
  + Tập train gồm 94603 bức ảnh.
  + Tập test gồm 5000 bức ảnh.

3.1 Trên tập train

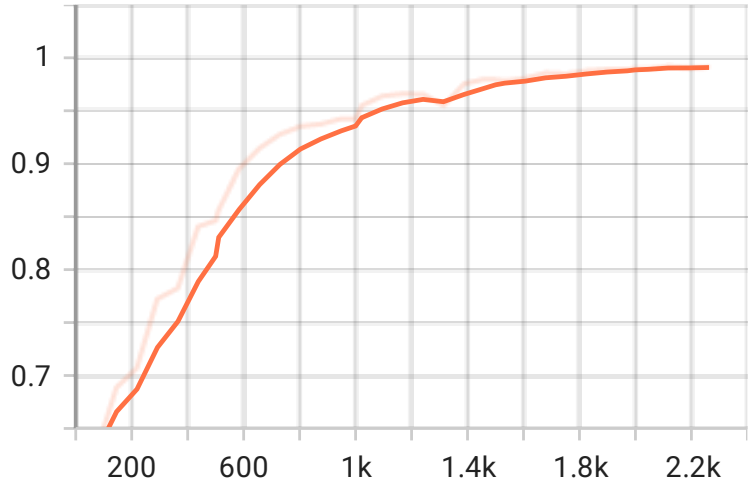


Hình 3.1 Accuracy trên tập train

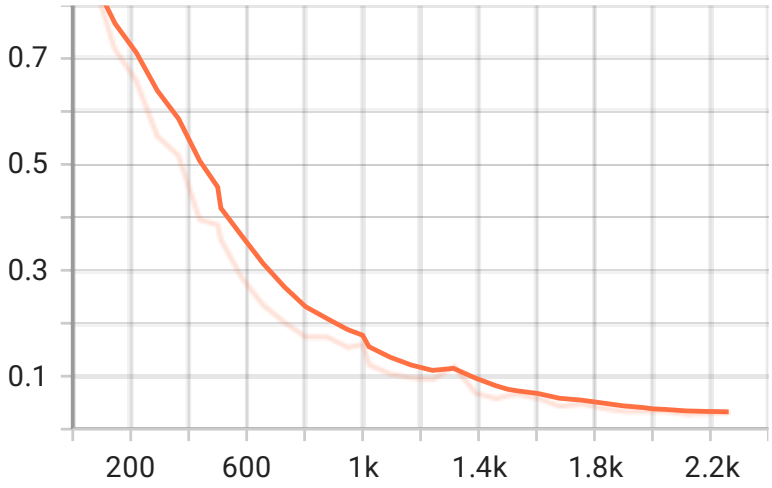


Hình 3.2 Loss trên tập train

3.2 Trên tập test



Hình 3.3 Accuracy trên tập train



Hình 3.4 Loss trên tập test

3.3 Video demo

Link:https://drive.google.com/file/d/19SgXlglk46QH53tF0rwgEEnwlkpmkLU5/view?usp=sharing

Link:https://drive.google.com/file/d/1mlrn2Ay5dCIjNhKZXXK0m1xpuBCXjT1-/view?usp=sharing

IV. Kết luận và Hướng phát triển

4.1 Kết luận

- Tự động lái xe bằng phương pháp phân loại thì model hoạt động ở mức ổn với đường xá ít đông đúc còn đối với đường xá nhiều xe cộ thì nó chỉ hoạt động ở mức độ tạm ổn. Nhưng đối với tự động xe như ngoài đời thì model chưa đạt được vì model chưa biết dừng đèn đỏ hay các tín hiệu giao thông trên đường. Và hơn thế nữa model vẫn chưa biết được lane đường đi của mình có sai hay không.

4.2 Hướng phát triển

- Em sẽ phát triển thêm như nhận diện các vật thể , nhận diện vạch kẻ đường, đi theo chỉ dẫn GPS để model có thể giống với tự động lái xe như ngoài đời . Đồng thời thêm những ý tưởng mới nhằm nâng cao hiệu quả hiệu suất của chương trình.

V. Tài Liệu Tham Khảo

- Simulate Python keypresses for controlling a game (Link: https://tinyurl.com/33kbx44n) [Truy cập 26/03/2022].

- DirectInput keyboard scan codes (Link: https://tinyurl.com/ms38wvym) [Truy cập 26/03/2022].

- Thư viện hỗ trợ trong cv2 (Link: https://opencv.org/) [Truy cập 26/03/2022].

- Tìm hiểu về CNN (Link: https://tinyurl.com/2p9fd33p) [Truy cập 28/03/2022].

- AlexNet Paper (Link: https://tinyurl.com/2p8779fu) [Truy cập 04/04/2022].

- Movement detector (Link: https://tinyurl.com/4jkpvhfx) [Truy cập 10/04/2022].