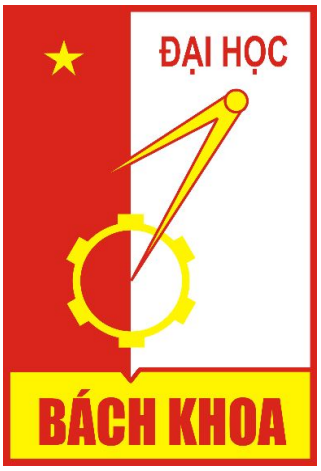
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO MÔN HỌC

HỌC SÂU VÀ ỨNG DỤNG

Đề tài:

SỬ DỤNG MẠNG HỌC SÂU ĐỂ NHẬN DIỆN BIỂN BÁO GIAO THÔNG

Giáo viên hướng dẫn: TS. Trịnh Anh Phúc

Nhóm sinh viên thực hiện: 1. Nguyễn Trí Quân - 20173312

2. Phạm Viết Bằng - 20172965

3. Mai Văn Hòa - 20173122

4. Phan Bá Hoàng - 20173141

5. Lê Công Nguyên - 20173290

6. Kiều Minh Hiếu - 20173111

Hà Nội, tháng 11 năm 2020

Lời mở đầu

Trong thời đại thông tin bùng nổ như hiện nay, từ khóa tự động hóa, trí tuệ nhân tạo đã không còn quá xa lạ đối với con người. AI đang được áp dụng rộng rãi vào mọi mặt của đời sống như xử lý các hoạt động tài chính, tiền đầu tư và cổ phiếu, quản lý các tài sản khác nhau, chăm sóc sức khỏe cá nhân, làm các công việc nặng nhọc, nguy hiểm thay con người... Trước đây những robot tự động giúp việc chỉ có trong truyện Doraemon, hay những cỗ máy định danh người, xác định hành động cụ thể chỉ có trong phim viễn tưởng. Tuy nhiên những năm trở lại đây với sự giúp đỡ của AI và những bộ vi xử lý mạnh mẽ thì những điều đó đều thành hiện thực bằng cách áp dụng trí tuệ nhân tạo vào trong robot, camera…

Khi bắt đầu bước vào chuyên ngành, chúng em, những sinh viên năm 4, đã thấy học máy là một mảng lĩnh vực hết sức thú vị và có tiềm năng phát triển rất tốt. Chính vì vậy, kỳ học vừa rồi chúng em đã chủ động đăng ký môn "Học sâu và ứng dụng" nhằm tìm hiểu, trau dồi những kiến thức cơ bản về học máy, đặc biệt là học sâu. Qua đó, tạo ra một sản phẩm đơn giản là phân loại biển báo giao thông để không những củng cố kiến thức mà còn thể hiện những gì mình được học để áp dụng vào thực tế.

Hà Nội, tháng 11 năm 2020.

\

1. TỔNG QUAN BÀI TOÁN

1.1. Giới thiệu bài toán.

Bài toán nhận diện biển báo giao thông từ tập dữ liệu "**German Traffic Sign Dataset**" là một bài toán học có giám sát, khá thú vị giúp cho sinh viên, đặc biệt là những bạn mới học về Deep Learning, có thể thực hành và rèn luyện khả năng lập trình cũng như kiến thức nền tảng cần nắm vững.

Tập dữ liệu gồm các *single-image*, tức ảnh chỉ có 1 biển báo và không có nhiều objections, và là một bài toán phân loại nhiều lớp (*multi-class classification problem)*.

1.2. Ý tưởng tiến hành

Mạng VGG là một mạng CNN (Convolutional Neural Network) có kiến trúc đơn giản nhưng đem lại hiệu quả cao trong bài toán phân loại ảnh .Vì vậy nhóm chúng em quyết định xây dựng mô hình dựa trên kiến trúc mạng VGG, tuy nhiên, do kích thước ảnh đầu vào nhỏ (32x32x3) và số lượng nhãn lớp ít (43 lớp) không cần thiết phải sử dụng một mạng quá sâu, quá phức tạp nên chúng em đã sửa lại cấu trúc mạng cho đơn giản hơn. Ngoài ra chúng em còn thêm vào các tầng **batch normalization** để tăng tốc quá trình huấn luyện và **dropout** để tăng tính tổng quát hóa cho mạng

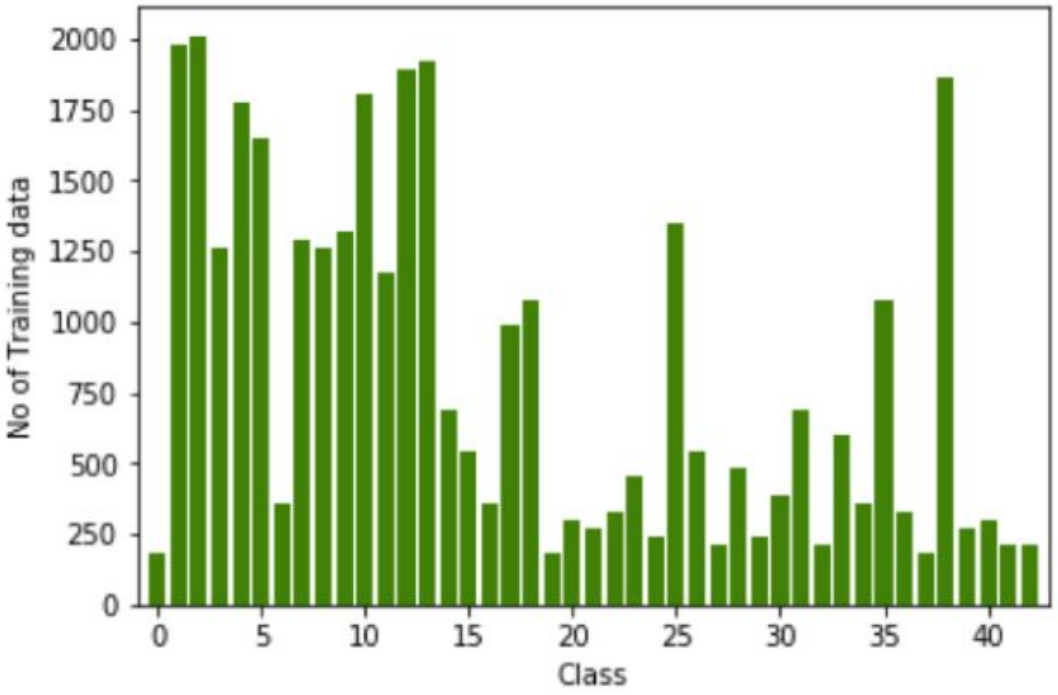
2. Tiến hành

2.1. Dữ liệu

Download dữ liệu trên trang chủ:

http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

Tập dữ liệu có **43 lớp** phân bố trên **hơn 50000 ảnh**. Cụ thể: tập **training** có **34799 ảnh**, tập **test** gồm **12630 ảnh** và tập **validation** gồm **4410 ảnh**.



Hình 1. Phân bố dữ liệu trên tập học

Ảnh đầu vào có kích thước 32x32x3, mỗi điểm ảnh có giá trị trong khoảng từ 0 đến 255.

Hình 2. Ví dụ 25 ảnh đầu vào

2.2. Chuẩn hóa dữ liệu

Chuẩn hóa những features hay các giá trị điểm ảnh bằng việc giảm khoảng giá trị từ [0;255] xuống còn [0;1] từ đó giảm độ chênh lệch ảnh hưởng giữa các điểm lên quá trình học, sự ảnh hưởng của từng data point lên kết quả training sẽ tương đồng nhau nhiều hơn, tránh bị thiên vị đối với những điểm có giá trị lớn.

2.3. Xây dựng mô hình

Dựa trên ý tưởng của mạng VGG bọn em sử dụng 4 tầng convolutional, qua 2 tầng convolutional liên tiếp sẽ có 1 tầng maxpool để giảm chiều. Trong mỗi tầng convolutional sẽ sử dụng kernel kích thước 3x3, stride là 1 và padding là 1 giữ nguyên kích thước ảnh. Tầng Maxpool có stride là 2 và kernel kích thước 2x2. Sau mỗi lần maxpool kích thước channel của tầng convolutional sẽ được tăng gấp đôi. Cuối cùng sẽ là 2 tầng fully connected được dùng cho phân loại với đầu ra là 43 tượng trưng cho số nhãn lớp.

Số lượng channel khởi đầu của tầng convolutional và số lượng neural trong mỗi tầng fully connected được coi như các siêu tham số cùng với tốc độ học và kích thước batch và sẽ được chúng em lựa chọn sao cho phù hợp ở phần sau.

Chúng em còn thêm vào sau mỗi tầng convolutional và tầng linear đầu tiên các tầng batch normalization giúp cải thiện luồng gradient và cho phép mô hình học nhanh hơn. Chúng em không để batch normalization ở tầng linear cuối cùng vì nó sẽ làm cho output có mean = 0 gây ảnh hưởng xấu tới kết quả phân loại

Chúng em còn sử dụng thêm chiến lược dropout để làm giảm overfit. Ở đây đầu ra của mạng linear thứ nhất sẽ được dropout với xác suất 0.5. Sở dĩ chúng em lựa chọn dropout ở đây bởi vì đây là tầng có số lượng tham số nhiều nhất vì vậy sẽ giúp làm giảm khối lượng tính toán và tránh hàm học được quá phức tạp.

Và sau đây là thông tin tổng quan các tầng, số lượng params sử dụng tương ứng của mô hình với các tham số tối ưu là batch size = 128, số lượng channel = 48, số neural tầng ẩn = 256:

```
      Layer (type)               Output Shape         Param #
================================================================
           Conv2d-1          [128, 48, 32, 32]           1,344
      BatchNorm2d-2          [128, 48, 32, 32]              96
```

```
              ReLU-3          [128, 48, 32, 32]               0
            Conv2d-4          [128, 48, 32, 32]          20,784
         MaxPool2d-5          [128, 48, 16, 16]               0
       BatchNorm2d-6          [128, 48, 16, 16]              96
              ReLU-7          [128, 48, 16, 16]               0
            Conv2d-8          [128, 96, 16, 16]          41,568
       BatchNorm2d-9          [128, 96, 16, 16]             192
             ReLU-10          [128, 96, 16, 16]               0
           Conv2d-11          [128, 96, 16, 16]          83,040
        MaxPool2d-12            [128, 96, 8, 8]               0
      BatchNorm2d-13            [128, 96, 8, 8]             192
             ReLU-14            [128, 96, 8, 8]               0
           Linear-15                 [128, 256]       1,573,120
          Dropout-16                 [128, 256]               0
      BatchNorm1d-17                 [128, 256]             512
             ReLU-18                 [128, 256]               0
           Linear-19                  [128, 43]          11,051
          Softmax-20                  [128, 43]               0
================================================================
Total params: 1,731,995
Trainable params: 1,731,995
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.50
Forward/backward pass size (MB): 343.08
Params size (MB): 6.61
Estimated Total Size (MB): 351.19
```

2.4. Quá trình huấn luyện

Chúng em sử dụng phương pháp tối ưu SGD với momentum = 0.9. Cùng với đó là chiến lược điều chỉnh tốc độ học như sau: giảm tốc độ học xuống 1/3 khi lỗi trên tập valid không được cải thiện trong 5 epoch liên tiếp. Các con số trên được chúng em lựa chọn dựa trên thực nghiệm và thấy sau khi điều chỉnh tốc độ học như vậy mô hình có thể thoát khỏi tối ưu cục bộ và tìm được phương án tốt hơn. Vì đây là bài toán phân loại với nhiều nhãn lớp nên chúng em dùng

hàm loss là cross entropy. Mô hình được huấn luyện trong 100 epoch, mỗi epoch thời gian huấn luyện là 18.5s, thời gian đánh giá là 2.7s. Độ đo chúng em sử dụng để đánh giá ở đây là Accuracy, được tính bằng số lần dự đoán đúng chia cho tổng số lần dự đoán.

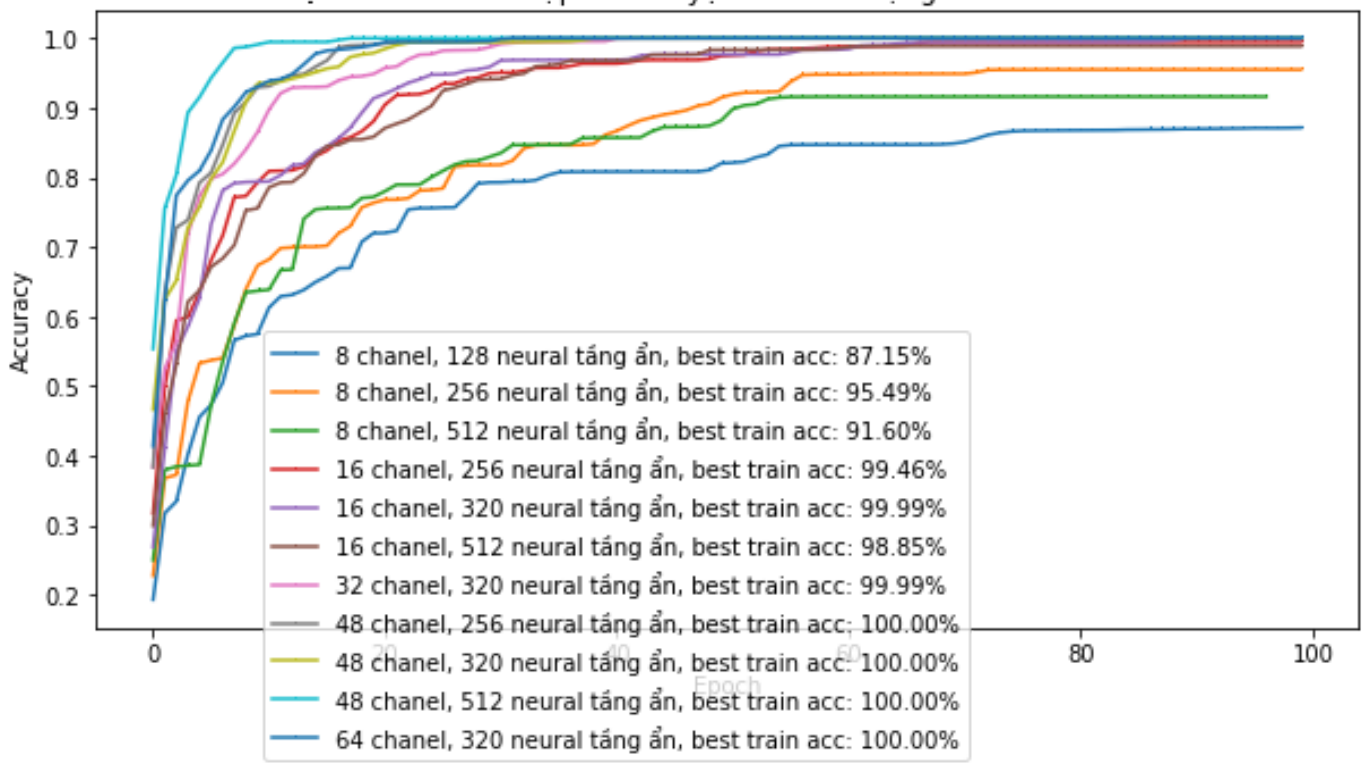
Cấu hình google colab: RAM: 12.72GB, DISK: 64.40GB, GPU: Tesla T4 bộ nhớ 15079 MiB, CUDA version 10.1.

2.5. Tối ưu các siêu tham số

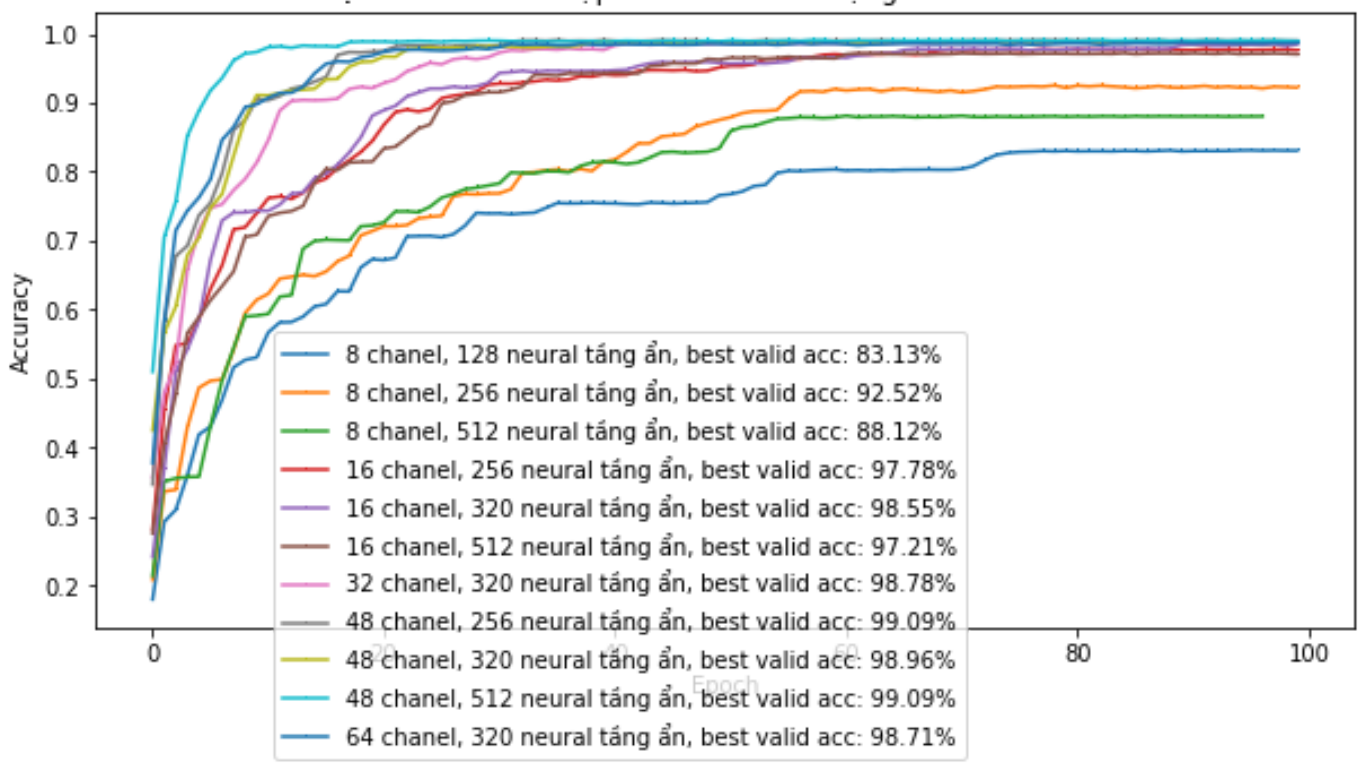
2.5.1. Tối ưu kích thước mạng:

Cố định learning rate là 0.01, batch size là 128, cho chạy thử mô hình với các giá trị của số channel là (8, 16, 32, 48, 64) và các giá trị của số neural tầng ẩn là (128, 256, 320, 512). Kết quả thu được như ở hình 3 và hình 4.

Hình 3: Độ chính xác trên tập huấn luyện của các mạng có kích thước khác nhau



Hình 4: Độ chính xác trên tập tối ưu của các mạng có kích thước khác nhau



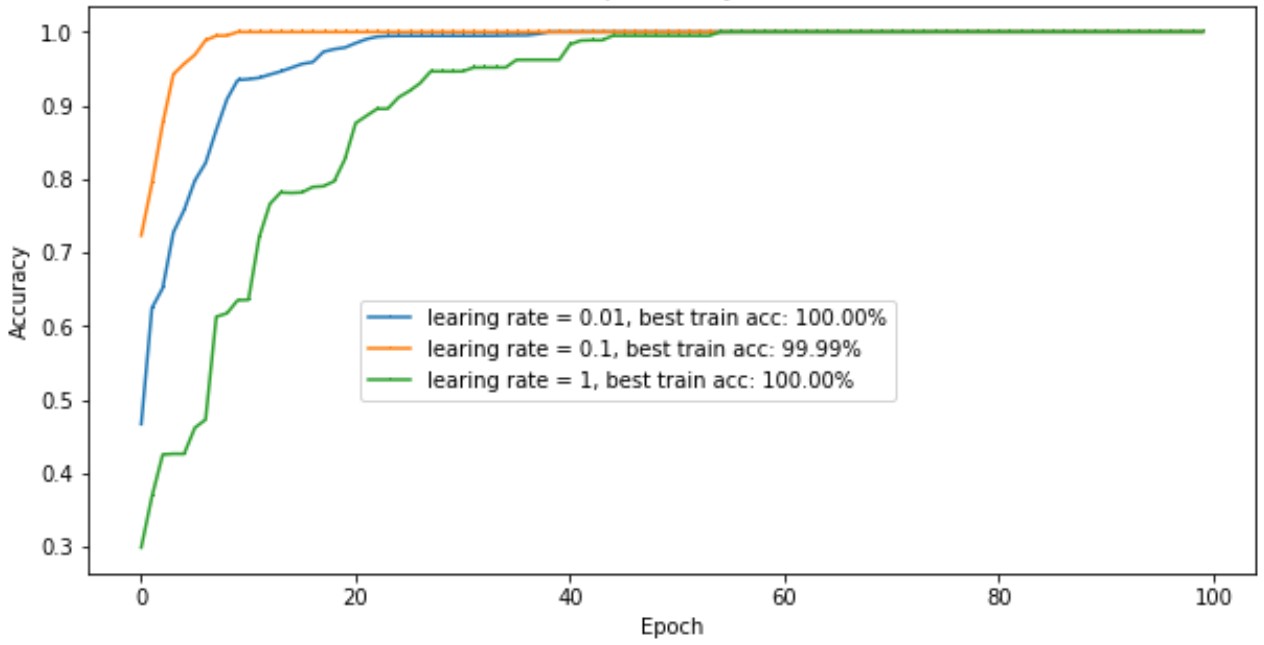
Với số lượng channel ít (8), số lượng đặc trưng được trích xuất ra quá nhỏ không đủ để biểu diễn ảnh. Quá nhiều (512) hoặc quá ít (128) neural tầng ẩn cũng làm cho hiệu quả học của mạng giảm.

Lựa chọn tốt nhất của chúng em đó là 48 channel và 256 neural tầng ẩn

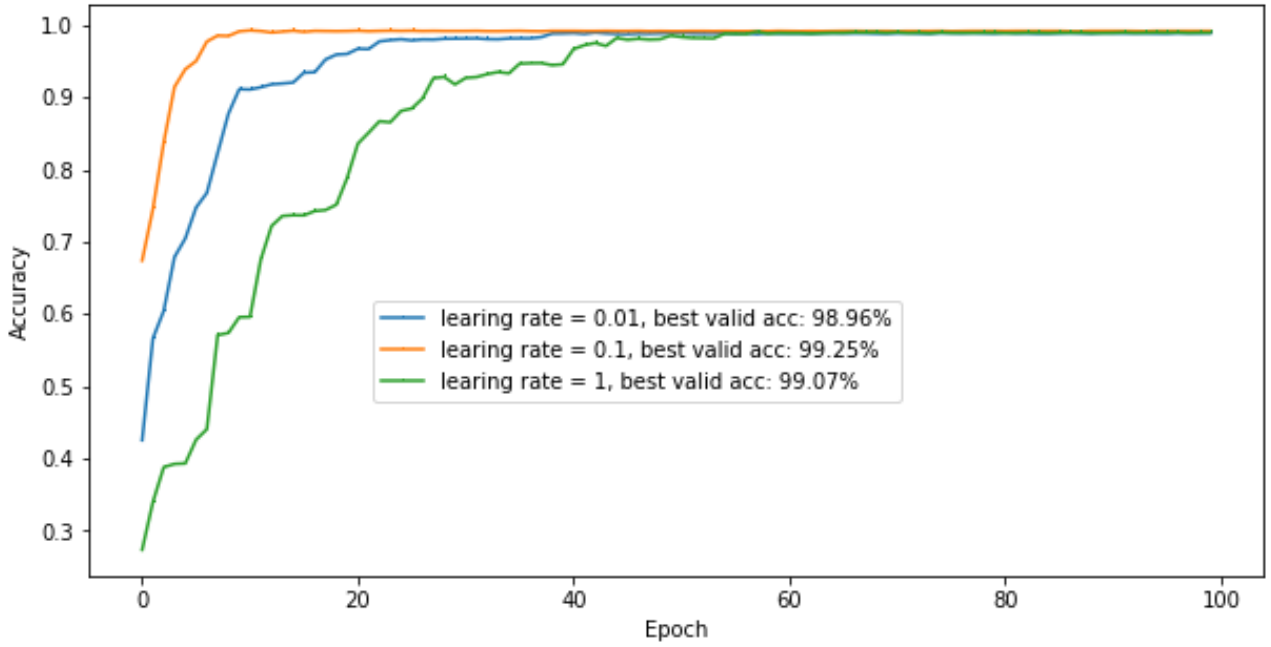
2.5.2. Tối ưu tốc độ học:

Cố định số channel là 48, số neural tầng ẩn là 320, batch size là 128. Chạy thử mô hình với các giá trị tốc độ học là (0.01, 0.1, 1). Kết quả thu được như ở hình 5, hình 6.

Hình 5: Độ chính xác trên tập huấn luyện với tốc độ học khác nhau



Hình 6: Độ chính xác trên tập tối ưu với tốc độ học khác nhau



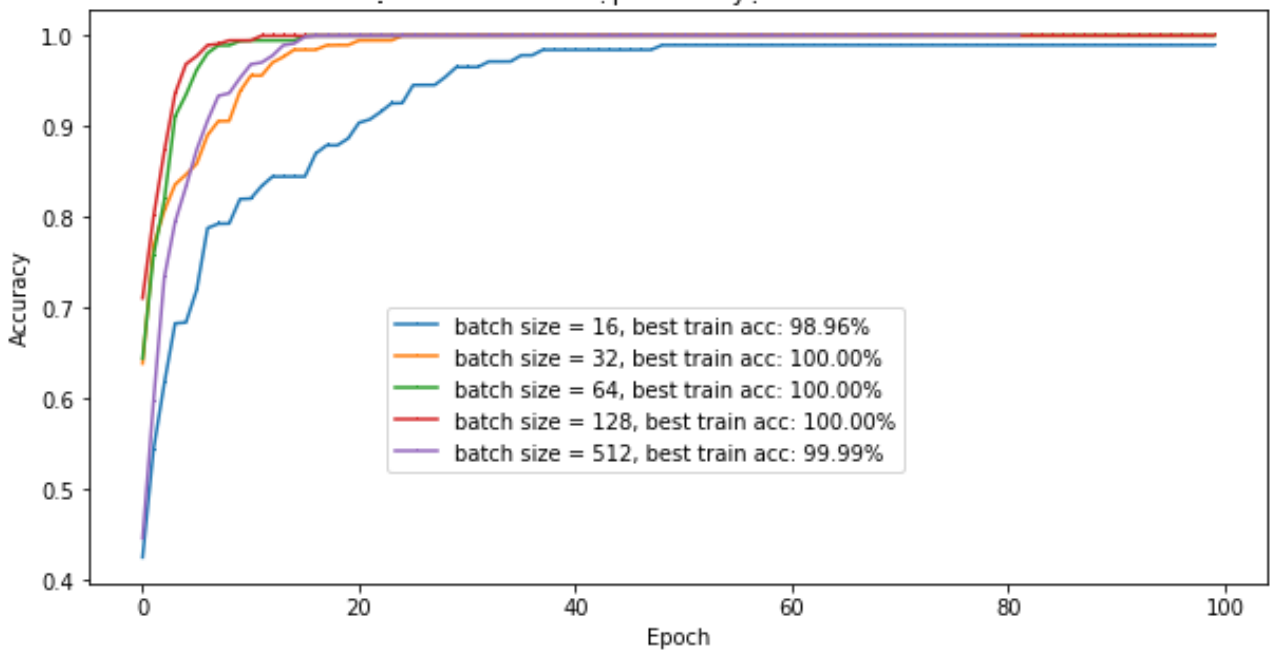
Khi tốc độ học nhỏ mạng hội tụ chậm. Đối với tốc độ học lớn, mạng khó học hơn nhưng nhờ có các tầng batch normalization mạng vẫn có thể hội tụ tới một nghiệm tương đối tốt.

Chọn một tốc độ học hợp lý có thể giúp cải thiện độ chính xác trên tập tối ưu và giúp mạng có thể hội tụ nhanh hơn.

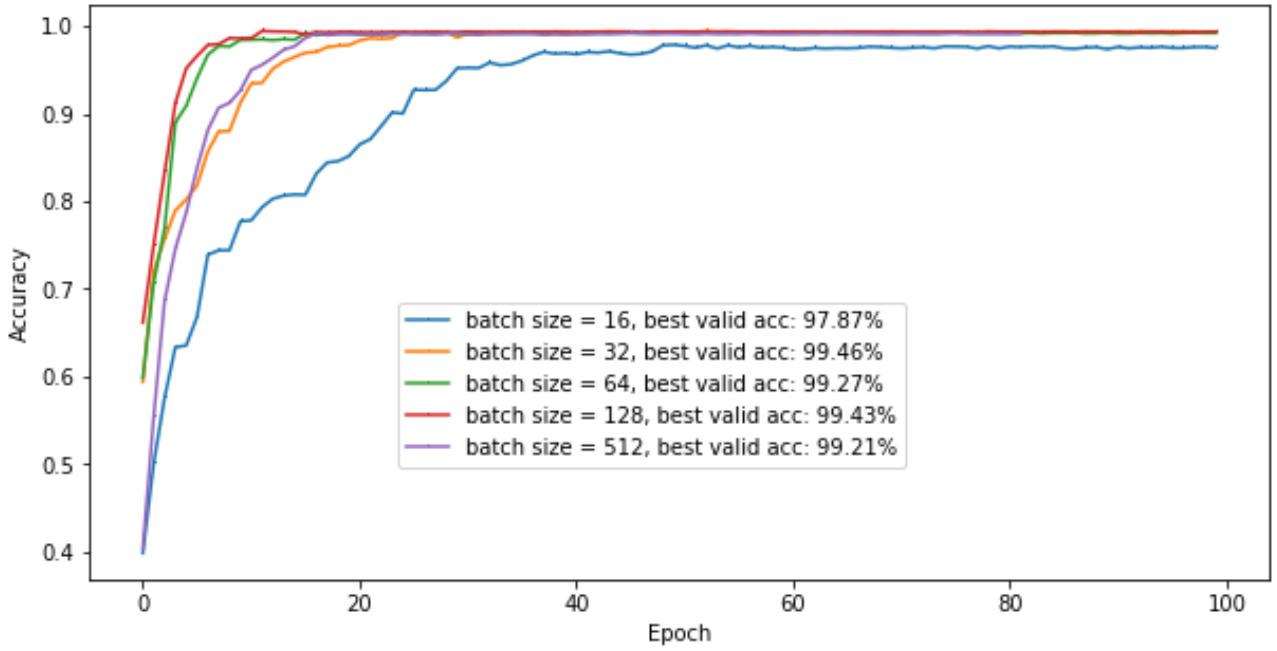
Vậy nên chúng em chọn tốc độ học là 0.1, với tốc độ học này cho phép mạng hội tụ sau 10 epoch mà vẫn cho kết quả độ chính xác tốt hơn.

2.5.2. Tối ưu kích thước batch:

Hình 7: Độ chính xác trên tập huấn luyện với batch size khác nhau



Hình 8: Độ chính xác trên tập tối ưu với batch size khác nhau



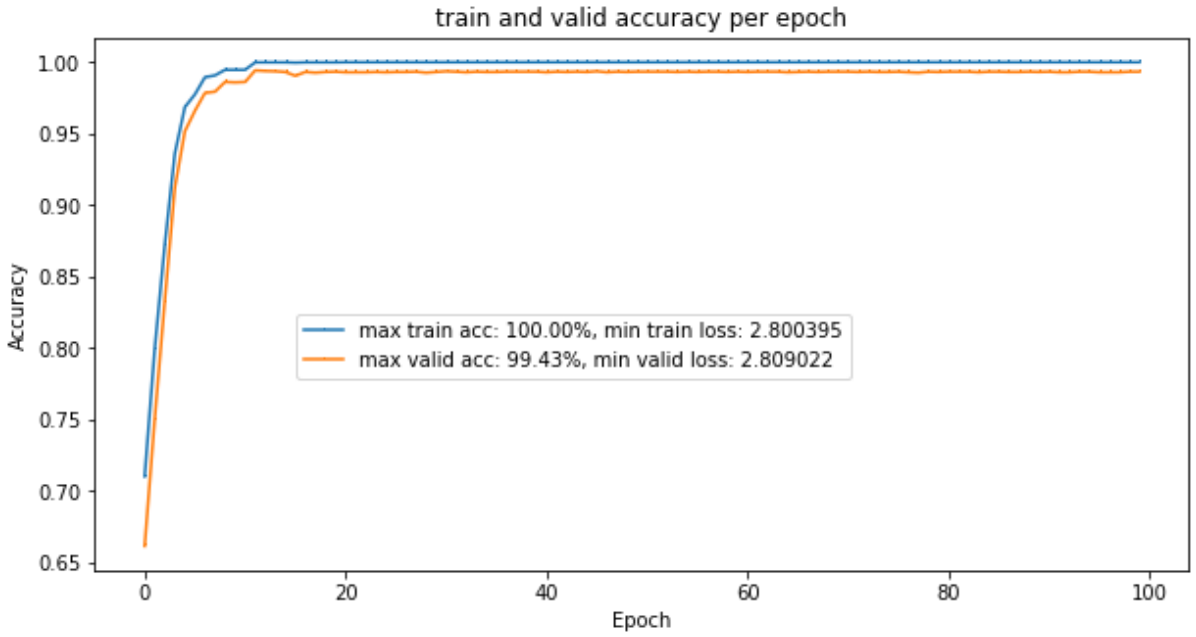
Cố định số channel là 48, số neural tầng ẩn là 256, tốc độ học là 0.1. Chạy thử mô hình với các giá trị batch size là (16, 32, 64, 128, 256). Kết quả thu được như ở hình 5, hình 6.

Tuy rằng batch size = 32 cho ra độ chính xác cuối cùng trên tập tối ưu là cao nhất nhưng batch size = 128 lại có tốc độ hội tụ nhanh nhất(hội tụ sau 10 epoch thay vì 25 epoch với batch size = 32). Chúng em quyết định chọn batch

size = 128 vì có thể huấn luyện mô hình nhanh chóng mà độ chính xác vẫn tương đối tốt.

2.6. Kết quả cuối cùng

Sau đây là kết quả huấn luyện mô hình tốt nhất với learning rate = 0.1, batch size = 128, số channel = 48, số neural tầng ẩn là 256, huấn luyện trong 100 epoch:



Thời gian huấn luyện trong 1 epoch là 18.4s và thời gian đánh giá lại trên tập train và valid là 2.7s. Mô hình hội tụ chỉ sau 12 epoch, tức là chỉ cần huấn luyện trong 253s.

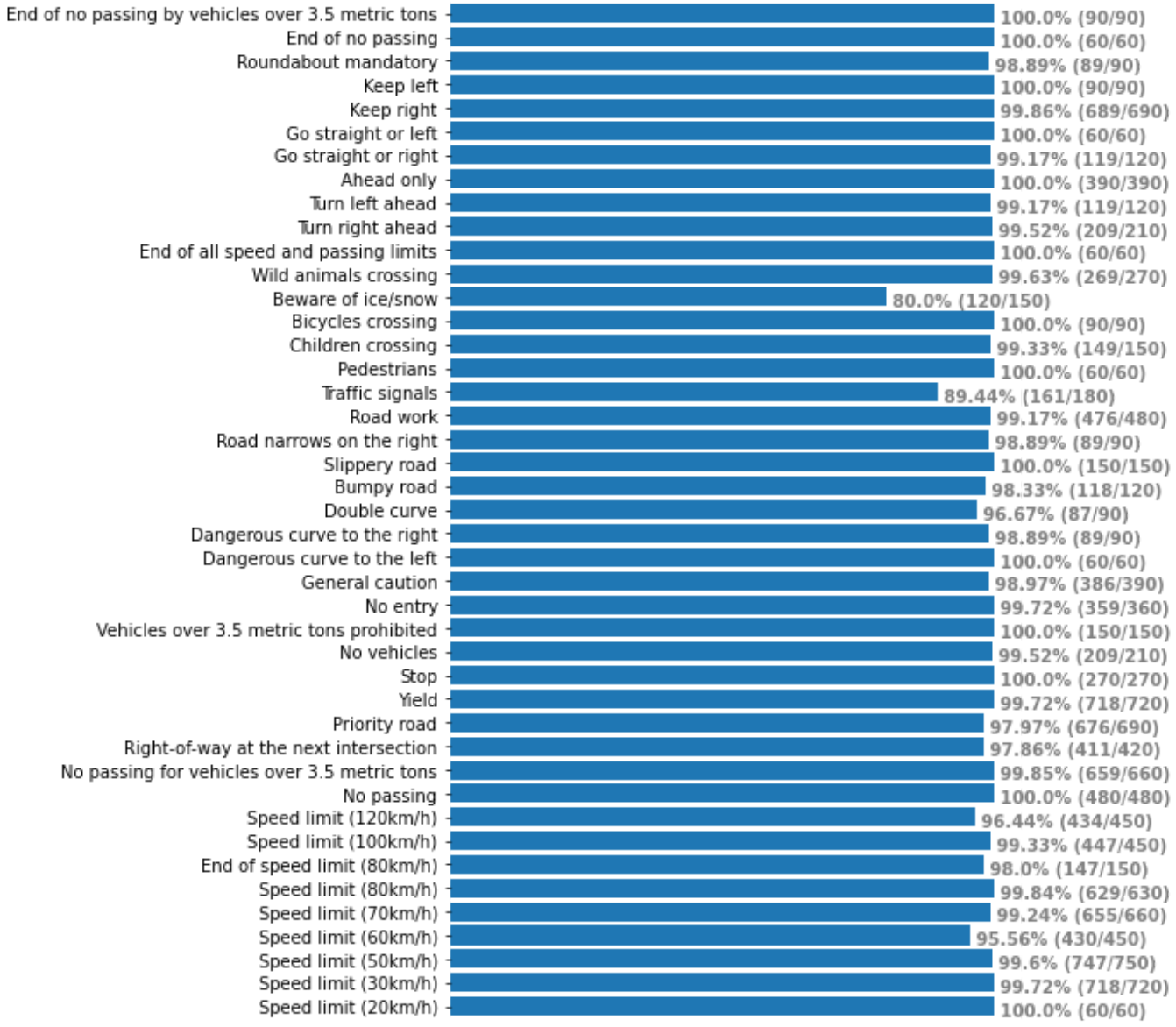
Đánh giá trên tập thử nghiệm thu được độ chính xác và lỗi trung bình là:

test acc = 98.80%

test loss = 2.818

Thời gian đánh giá : 0.85s

Độ chính xác khi nhận diện từng loại biển báo như sau:



3. Tổng kết

Trên đây là toàn bộ quá trình nhóm em đã áp dụng hầu hết những kiến thức học được trên lớp vào việc giải bài toán thực tế là nhận diện biển báo giao thông. Mặc dù kết quả đạt được chưa được như ý muốn nhưng có thể nói nhóm em đã khá thành công trong việc giải quyết bài toán. Và hướng phát triển tiếp theo nhóm em muốn hướng tới đó là tiếp tục nghiên cứu, tìm hiểu một số model khác giải quyết bài toán thu được kết quả tốt hơn cũng như một số phương pháp học sâu cải tiến model đã áp dụng. Ngoài ra, nhóm em có thể tích hợp vào trong một application, nhúng trong các thiết bị điện tử để có thể áp dụng vào trong thực tế.

Cuối cùng, nhóm chúng em xin cảm ơn thầy **Trịnh Anh Phúc**, giảng viên bộ môn Học sâu và ứng dụng, đã tạo điều kiện cho nhóm em được nghiên cứu, tìm tòi và phát triển những kiến thức nền tảng của môn học.