**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH**

Thực hiện: Nhóm 13

20C12007 – Trần Đình Lâm

20C11035 – Trương Thế Kiệt

20C11040 – Đặng Nhật Minh

20C11034 – Nguyễn Trung Kiên

BÁO CÁO ĐỒ ÁN CÀI ĐẶT

NĂM HỌC 2020-2021

**TRÍ TUỆ NHÂN TẠO nâng cao Phân loại biển báo giao thông**

**ĐẠI HỌC QUỐC GIA TPHCM**

**BẢNG THÔNG TIN CHI TIẾT NHÓM**

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã nhóm:** | **13** | | |
| **Tên nhóm:** | **K2014** | | |
| **Số lượng:** | **4** | | |
| **MSSV** | **Họ tên** | **Email** | **Điện thoại** |
| 20C12007 | Trần Đình Lâm | tdlam123@gmail.com | 0383522356 |
| 20C11035 | Trương Thế Kiệt | truongthekiet709@gmail.com |  |
| 20C11040 | Đặng Nhật Minh | minhdangnhat685@gmail.com |  |
| 20C11034 | Nguyễn Trung Kiên | ngkien1530@gmail.com |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **BẢNG PHÂN CÔNG & ĐÁNH GIÁ HOÀN THÀNH CÔNG VIỆC** | | | |
| **Người thực hiện** | **Công việc thực hiện** | **Mức độ hoàn thành** | **Đánh giá của nhóm** |
| 20C12007  Trần Đình Lâm | Phân công các công việc cần thực hiện | 90% | 9/10 |
| Cài đặt model classification đã tham khảo, tìm hiểu và giải thích mô hình mạng CNN được đề xuất |
| Train và test model với tỉ lệ 0.8/0.2, Thử nghiệm chạy train bằng GPU |
| Viết báo cáo phần so sánh kết quả phân loại với một số state-of-the-art |
| 20C11035  Trương Thế Kiệt | So sánh kết quả phân loại của mạng với các kết quả của cuộc thi | 80% | 8/10 |
| Mô tả cấu trúc dataset GTSRB |
|  |
| 20C11040  Đặng Nhật Minh | Chạy Train lại model với thay đổi là tỉ lệ DropOut thành 0.2 thay vì 0.5 | 80% | 8/10 |
| Viết báo cáo mô tả bài toán |
| Viết báo cáo mô tả tiêu chí so sánh với các mạng khác |
| 20C11034  Nguyễn Trung Kiên | Chạy Train lại model và đánh giá thử với một số thay đổi, lưu lại model vào đo thời gian thực thi | 80% | 8/10 |
| Viết báo cáo Mô tả các bước khi chạy model |
| Viết báo cáo kết quả so sánh với các giải pháp khác |

MỤC LỤC

[I. TỔNG QUAN 2](#_Toc69046027)

[1. Mô tả bài toán 2](#_Toc69046028)

[2. Các thư viện sử dụng trong project 2](#_Toc69046029)

[II. PHÂN TÍCH BỘ DỮ LIỆU GTSRB 3](#_Toc69046030)

[1. Cấu trúc Dataset GTSRB 3](#_Toc69046031)

[2. Chuẩn hóa data 4](#_Toc69046032)

[3. Load data 4](#_Toc69046033)

[III. CÀI ĐẶT MÔ HÌNH MẠNG 5](#_Toc69046034)

[1. Giới thiệu về Keras 5](#_Toc69046035)

[2. Cấu trúc mạng nhóm đã cài đặt 6](#_Toc69046036)

[IV. CHẠY CHƯƠNG TRÌNH VÀ ĐÁNH GIÁ KẾT QUẢ 10](#_Toc69046037)

[1. Các bước cần thực hiện 10](#_Toc69046038)

[2. Kiểm thử và đo kết quả 11](#_Toc69046039)

[V. SO SÁNH VỚI CÁC MÔ HÌNH KHÁC 13](#_Toc69046040)

[1. Tiêu chí so sánh 13](#_Toc69046041)

[2. Kết quả so sánh 13](#_Toc69046042)

[VI. KẾT LUẬN 15](#_Toc69046043)

[TÀI LIỆU THAM KHẢO 16](#_Toc69046044)

# TỔNG QUAN

## Mô tả bài toán

Ý tưởng về mạng nơ-ron tích chập (Convolutional Neural Network) được giới thiệu vào những năm 1950 và 1960 bởi Hubel và Wiesel và được cài đặt lần đầu tiên năm 1990 bởi Yann LeCun để áp dụng và nhạn dạng chữ viết tay, và hiện tại là một mô hình được ứng dụng rất thành công trong việc nhận dạng và phân loại hình ảnh

Trong đồ án này, nhóm sẽ tìm hiểu về cấu trúc, cách hoạt động và xây dựng một mạng nơ-ron tích chập để ứng dụng trong việc phân loại các biển báo giao thông. Sau đó bằng thực nghiệm so sánh kết quả đạt được với các mô hình khác

Tập dữ liệu được sử dụng trong quá trình huấn luyện và benchmark là GTSRB

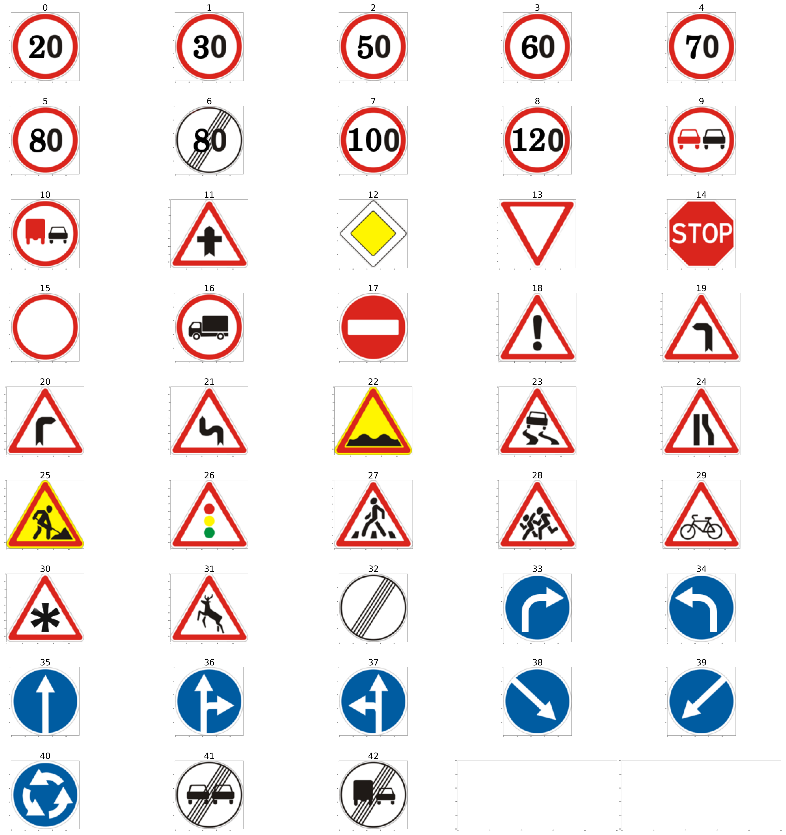
## Các thư viện sử dụng trong project

|  |  |
| --- | --- |
| **Tên thư viện** | **Mục đích sử dụng** |
| os | làm việc với các tập tin và thư mục |
| pandas | làm việc với dataset |
| numpy | tính toán |
| matplotlib và seaborn | trực quan hóa dữ liệu bằng các dạng biểu đồ |
| PIL | thao tác trên hình ảnh |
| Sciki-learn sklearn | cung cấp cài đặt của các thuật toán thường dùng trong machine learning |
| Tensorflow và Keras | xây dựng mô hình neural network |

# PHÂN TÍCH BỘ DỮ LIỆU GTSRB

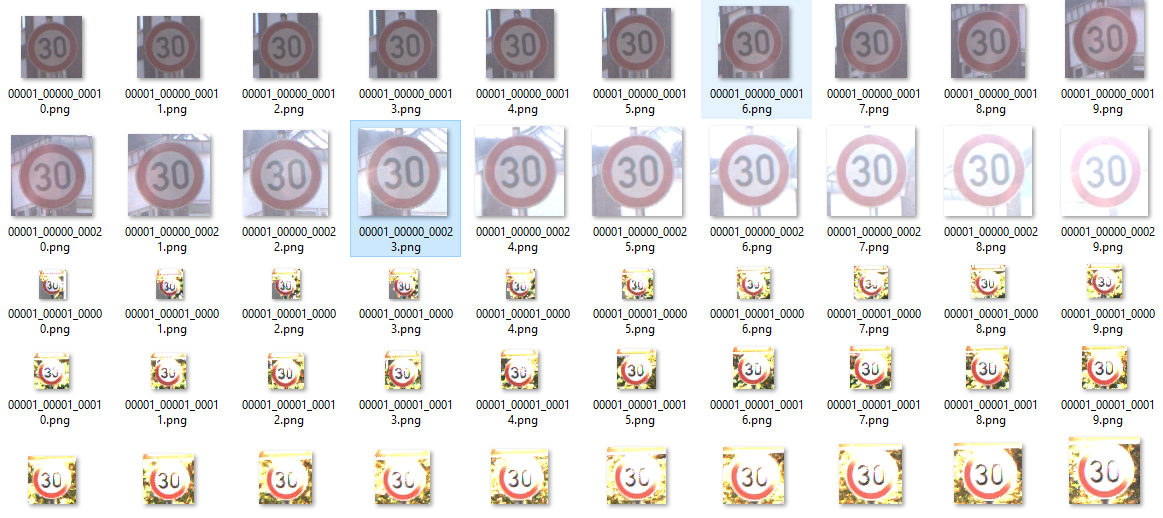
## Cấu trúc Dataset GTSRB

* **German Traffic Sign Recognition Benchmark(GTSRG)** là bộ data được sử dụng để trainning và test trong khảo sát này. Bộ data này được phần chia thành 43 loại với tổng số lượng hình ảnh cho bộ dữ liệu train lên đến 39209 hình.
* Các loại biển báo trong bộ data **GTSRG**:



Hình ‑: Các loại biển báo trong GTSRB

* Hình ảnh trong mỗi loại biển báo rất đa dạng về độ lớn, độ tương phản, noise và blurred.



Hình ‑b: Độ đa dạng hình ảnh trong từ loại

## Chuẩn hóa data

* Do hình ảnh với size khác nhau, nên mỗi tấm ảnh từ nguyên gốc với size w x h x 3 sẽ được đưa về dạng được resize về 50x50x3.
* Sau khi chuẩn hóa xong thì dùng numpy để lưu lại 2 file vào folder "numpy", để không phải scan lại tập train.
* Lúc này ta có được tập train kích thước (39209, 50, 50, 3)

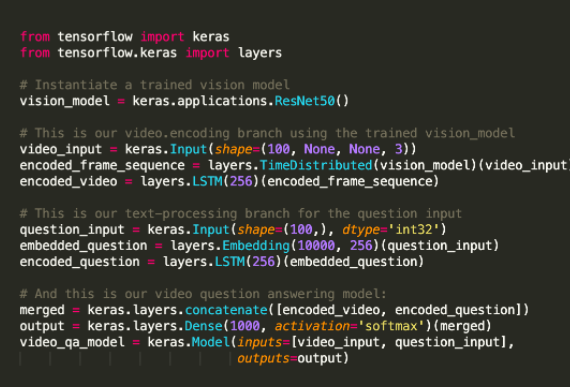
## Load data

* Tập dữ liệu input sau khi được chuẩn hóa xong sẽ được chia làm hai bộ là bộ train và bộ test.
* Bốn biến output tương ứng sau khi load data:
* x\_train có x\_val tương ứng
* y\_train có y\_val tương ứng

# CÀI ĐẶT MÔ HÌNH MẠNG

## Giới thiệu về Keras

**Keras** [1] là một thư viện open-source phổ biến, cung cấp môi trường và các công cụ trên Python hỗ trợ cài đặt mạng neural nhân tạo. Keras hoạt động dựa trên nền TensorFlow do Google phát triển. Keras chứa nhiều thành phần phổ biến, và nhiều cài đặt cụ thể, đa dạng phong phú các thành phần cấu thành trong một mạng neural như các layer, objective, các activation function, optimizer,…



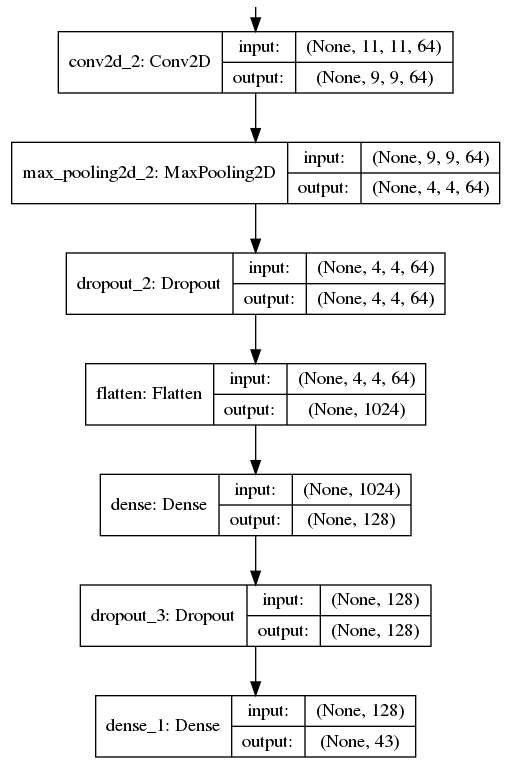
Hình ‑: Ví dụ cài đặt sử dụng Keras (Nguồn: https://keras.io/)

Keras thường được sử dụng phổ biến trong các bài toán xử lý ảnh và text, giúp dễ dàng lập trình hơn, đặc biệt Keras hỗ trợ rất tốt trong việc xây dựng các mạng CNN và RNN, do có cài đặt đa dạng và đầy đủ các utility cần thiết như dropout, batch normalization, pooling, và còn cho phép chạy model trên các nền phần cứng từ CPU, CPU tới TPU.

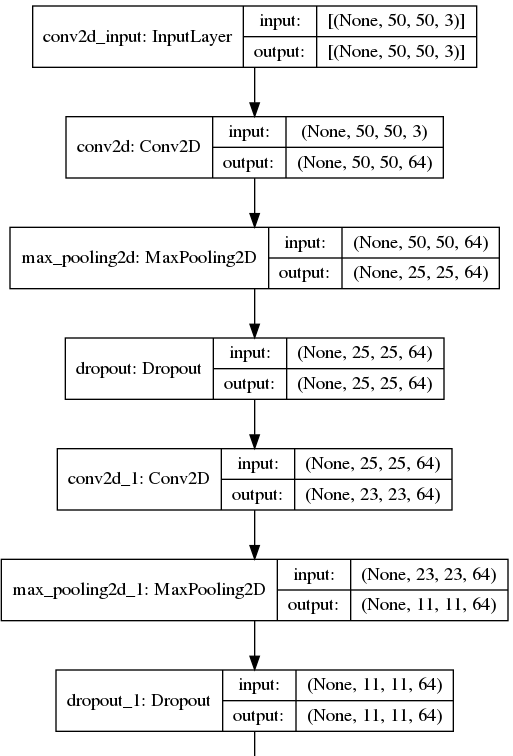
Các core module [2] của Keras có thể kể đến như sau:

* Activation module: Các hàm kích hoạt cài đặt sẵn
* Backend module: Keras backend API
* Callback module: Các hàm callback được gọi trong quá trình train model
* Datasets module: Các dataset mặc định phổ biến
* Layer module: Các layer API dựng sẵn
* Lossed module: Các hàm loss dựng sẵn
* Metrics module
* Models module
* Optimizers module
* Preprocessing module

## Cấu trúc mạng nhóm đã cài đặt

 Dựa theo cấu trúc một số mạng CNN được sử dụng tương ứng cho tập dữ liệu GTSRB trên Kaggle [3], cùng một số điều chỉnh mới, nhóm đã xây dựng mô hình mạng CNN như hình *III‑b* và *III‑c*:

Hình ‑c Cấu trúc 4 layer sau của mạng

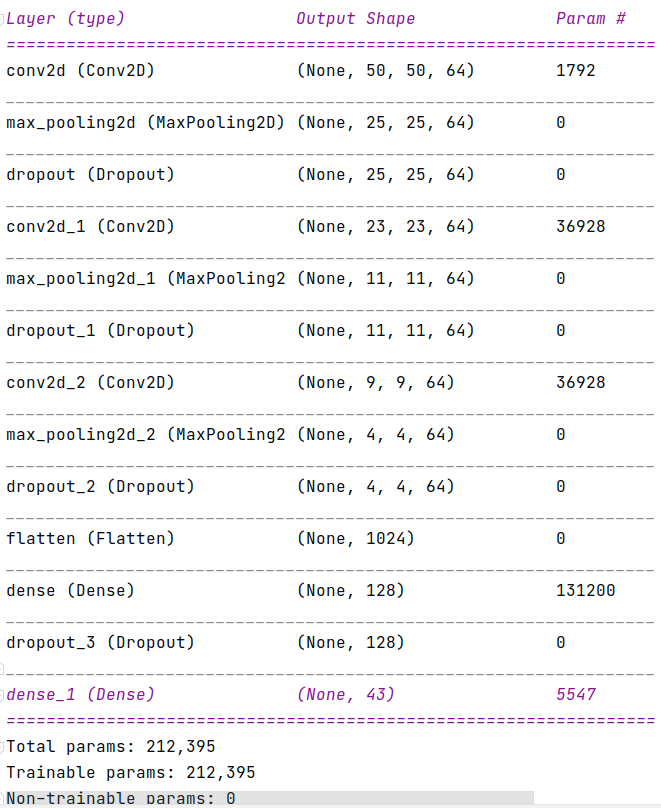


Hình ‑b Cấu trúc 2 layer đầu của mạng

Theo đó, cấu trúc mạng bao gồm 6 layer chính:

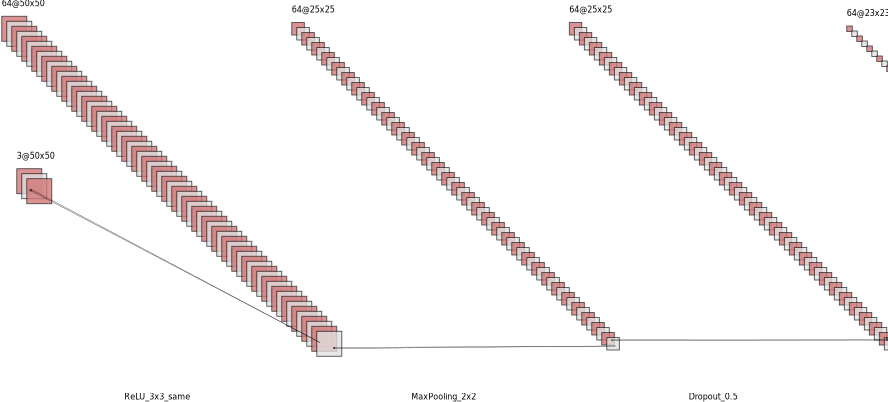
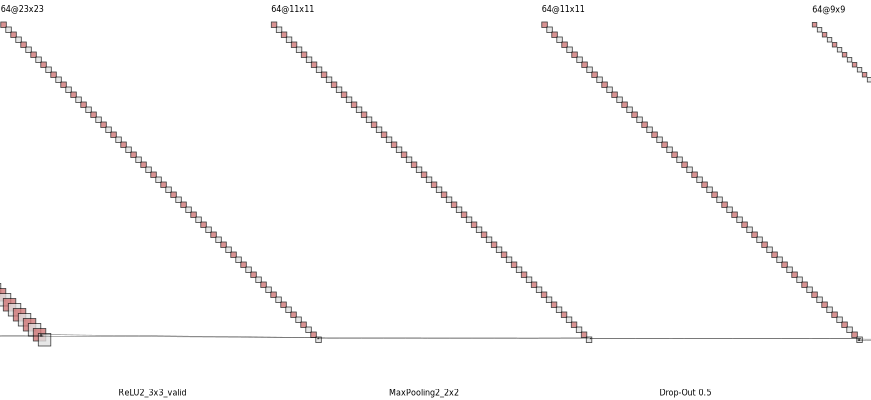
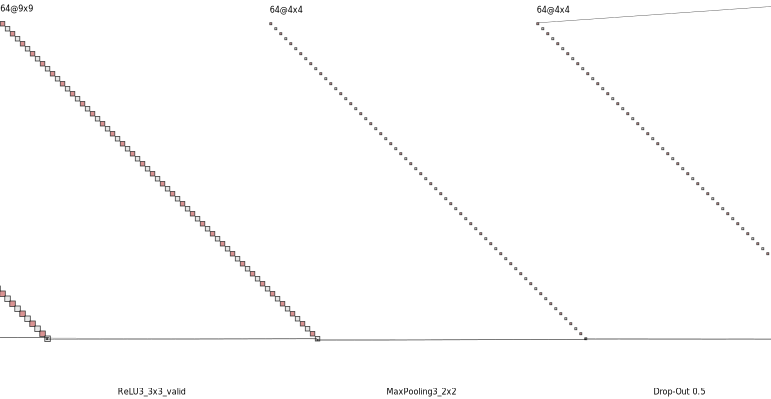
* Layer 1 đến layer 3: Convolutional với kernel size 3x3, sau đó kết hợp max pooling 2x2 và dropout 50%
* Layer 4: Là một lớp Flatten để dàn phẳng output của layer 3
* Layer 5 và Layer 6: Fully-connected layer
* Đầu ra cuối cùng là vector 43 chiều, biểu thị 43 loại biển báo giao thông cần phân loại.

Sau khi cài đặt, các thông số mô tả mô hình biểu thị như hình *III‑d* sau:



Hình ‑d Các thông số của model cài đặt bằng Keras

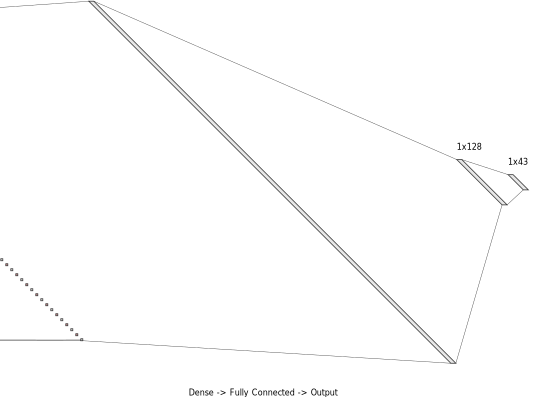
Một cách thể hiện khác của mô hình, vẽ bằng công cụ [4]:



Hình ‑h Layer 3

Hình ‑g Layer 2

Hình ‑e Layer 1



Hình ‑i Layer 4,5,6

# CHẠY CHƯƠNG TRÌNH VÀ ĐÁNH GIÁ KẾT QUẢ

## Các bước cần thực hiện

* Cấu hình tham số và đường dẫn  
  
* Trực quan hóa dữ liệu  
  
* Tìm giá trị trung bình  
  
* Chuẩn hóa dữ liệu  
  
* Bắt đầu thực hiện quá trình huấn luyện  
  

**Lưu ý:** Sau khi quá trình huấn luyện hoàn tất, ta nhận được tập tin model h5 được lưu trong thư mục model. Đây là quá trình đánh giá mô hình (Model Evaluation), tập tin này lưu lại kết quả history của model. Mục đích của bước này là vẽ ra sự biến thiên của hàm loss và đo độ chính xác của mạng.

## Kiểm thử và đo kết quả

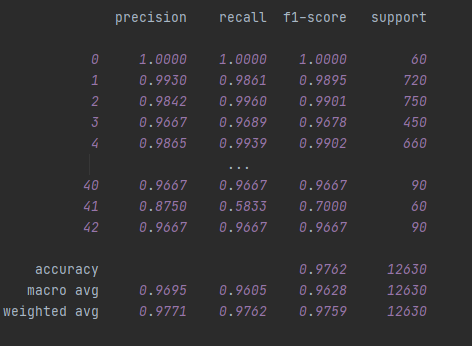
* Tiến hành load model đã lưu và chạy test  
  
* Những môi trường nhóm đã thực hiện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Môi trường | Tham số | Thời gian chạy |
| 1 | Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz (4 CPUs) | train=0.8  test=0.2  batch\_size=64  standard\_size=50,50  filters\_layer\_1: 64  filters\_layer\_2: 64  filters\_layer\_3: 64 | 2796 giây |
| 2 | Intel(R) Core(TM) i5 8400 @ 2.800GHz (6 CPUs) | train=0.9  test=0.1  batch\_size=64  standard\_size=60,60  filters\_layer\_1: 64  filters\_layer\_2: 64  filters\_layer\_3: 64 | 1479 giây |
| 3 | Google Colab GPU Tesla K30 | train=0.8  test=0.2  batch\_size=64  standard\_size=50,50  filters\_layer\_1: 64  filters\_layer\_2: 64  filters\_layer\_3: 64 | 134 giây |
| 4 | NVIDIA GeForce GTX 1070 | train=0.8  test=0.2  batch\_size=64  standard\_size=50,50  filters\_layer\_1: 64  filters\_layer\_2: 96  filters\_layer\_3: 128 | 130 giây |

* Sau khi thực hiện với nhiều tham số khác nhau, nhóm đã đúc kết được 1 model tốt nhất với bộ tham số:



* Kết quả đạt được với độ chính xác tốt nhất là **97.71%**



Hình IV‑i Kết quả mô hình tốt nhất

# SO SÁNH VỚI CÁC MÔ HÌNH KHÁC

## Tiêu chí so sánh

Ta cần mô tả sơ lược về các mô hình được đem ra so sánh với mô hình của nhóm dựa trên các tiêu chí sau:

* Tiêu chí 1: Tổng Accuracy
* Tiêu chí 2: Từng loại Traffic-sign nhỏ:

Trong tài liệu tham khảo, các tác giả chia 43 loại Traffic-sign thành 6 loại con bao gồm:

* + "Blue": 8 loại 33, ...,40
  + “Danger”: 15 loại: 11, 18, ..., 31
  + “End-of”: 4 loại 6, 32, 41, 42
  + “Speed”: 8 loại 0, 1, 2, 3, 4, 5, 7, 8
  + "Red-other”: 4 loại 9, 10, 15, 16
  + “Spezial”: 4 loại 12, 13, 14, 17

## Kết quả so sánh

Nhóm sẽ chọn ra model tốt nhất trong các model đã huấn luyện được để so sánh với các model khác trong cuộc thi

* 1. **Bảng so sánh tổng Accuracy:**

|  |  |
| --- | --- |
| **Method** [5] | **Accuracy** |
| Novel DNN Selection | 99.92% |
| Haloi | 99.81% |
| Committee of CNNs | 99.46% |
| Human Performance | 98.84% |
| Multi-Scale CNNs | 98.31% |
| **Của nhóm** | **98.00%** (tốt nhất hiện tại của nhóm) |
| Random Forests | 96.14% |

* 1. **Bảng so sánh accuracy theo từng loại:**
* Lớp con đầu tiên được gọi là “Blue” và chứa 8 lớp bao gồm lớp 33 đến lớp 40.
* Lớp thứ hai được gọi là "Danger" và có 15 lớp bao gồm lớp 18 đến lớp 31 và 11.
* Lớp thứ ba được gọi là "End-of" và bao gồm 4 lớp bao gồm lớp 6, 32, 41, 42.
* Chiếc thứ tư được gọi là "Danger" và có 8 lớp bao gồm lớp 0 đến lớp 5 và lớp 7, 8.
* Lớp thứ năm được gọi là "Red-other" và chứa 4 lớp bao gồm lớp 9, 10, 15, 16.
* Lớp thứ sáu được gọi là "Spezial" và chứa 4 lớp bao gồm lớp 17, 12, 13, 14.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Method** [5] | **Blue** | **Danger** | **End-of** | **Speed** | **Red other** | **Spezial** |
| Deep network 1 | 99.52% | 99.93% | 99.98% | 99.03% | 99.89% | 99.94% |
| Deep network 2 | 99.96% | 99.17% | 97.19% | 99.92% | 99.95% | 98.27% |
| Haloi | 99.72% | 99.89% | 99.95% | 99.86% | 100% | 99.87% |
| Committee of CNNs | 99.89% | 98.03% | 94.44% | 98.61% | 99.87% | 98.63 |
| Human performance | 99.72% | 98.67% | 98.89% | 97.63% | 99.93% | 100% |
| **Our method** | **98.64%** | **96.31%** | **90.27%** | **98.74%** | **97.70%** | **98.70%** |
| Random Forests | 95.95% | 99.13% | 87.50% | 99.27% | 92.08% | 98.73% |

# KẾT LUẬN

* Với mục tiêu tìm hiểu, cài đặt và thực hiện train một model mới cho việc phân loại biển báo giao thông thì nhóm đã đạt mục tiêu đặt ra ban đầu.
* Kết quả thực nghiệm cho thấy tính hiệu quả của mô hình nhóm đưa ra có kết quả tốt hơn một vài mô hình trước đó, nhưng không thể đạt được cao hơn so với những mô hình học sâu (độ chính xác lên đến 99,9xx%).
* Ngoài ra dựa vào kết quả chạy trên các môi trường khác nhau cho thấy tốc độ train của một model cũng phụ thuộc vào phần cứng.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | "Keras - From Wikipedia, the free encyclopedia," [Online]. Available: https://en.wikipedia.org/wiki/Keras. |
| [2] | "Module: tf.keras | TensorFlow Core v2.4.1," [Online]. Available: https://www.tensorflow.org/api\_docs/python/tf/keras?hl=tr. |
| [3] | Kaggle, "Traffic sign Classification using CNN," [Online]. Available: view-source:https://www.kaggle.com/pritamaich/traffic-sign-classification-using-cnn. |
| [4] | "NN-SVG," [Online]. Available: http://alexlenail.me/NN-SVG/LeNet.html. |
| [5] | S. I. M. K. M. &. T. S. Saha, "An Efficient Traffic Sign Recognition Approach Using a Novel Deep Neural Network Selection Architecture," 2019. |
| [6] | V. H. Tiệp, "Bài 36. Giới thiệu về Keras," machinelearningcoban.com, [Online]. Available: https://machinelearningcoban.com/2018/07/06/deeplearning/. |