

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN



Mai Hoàng Hiệp  
Trần Trung Hiếu  
Nguyễn Duy Đạt

Báo cáo  
Project: Quick, Draw!  
Doodle Recognition Challenge

CHƯƠNG TRÌNH CHÍNH QUY

Tp. Thủ Đức, tháng 11/2024

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN

Mai Hoàng Hiệp - 24122015

Trần Trung Hiếu - 24122033

Nguyễn Duy Đạt - 24122014

Báo cáo

Project: Quick, Draw!

Doodle Recognition Challenge

CHƯƠNG TRÌNH CHÍNH QUY

GIÁO VIÊN HƯỚNG DẪN

ThS. Nguyễn Trần Duy Minh

Tp. Thủ Đức, tháng 11/2024

# Lời cam đoan

Tôi xin cam đoan đây là công trình nghiên cứu của riêng nhóm chúng tôi. Các số liệu và kết quả nghiên cứu trong luận văn này là trung thực và không trùng lặp với các đề tài khác.

# Lời cảm ơn

Tôi xin chân thành cảm ơn ThS. Nguyễn Trần Duy Minh là giảng viên môn Giới thiệu ngành Trí tuệ nhân tạo - TTNT2024 phần thực hành đã hướng dẫn và cung cấp tài liệu tham khảo quý giá giúp nhóm chúng tôi hoàn thành báo cáo này. Đồng thời gửi lời cảm ơn sâu sắc đến các tác giả của những tài liệu nguồn tham khảo mà chúng tôi đã dùng trong bài báo cáo lần này. Cảm ơn những thành viên trong nhóm: Mai Hoàng Hiệp, Trần Trung Hiếu, Nguyễn Duy Đạt đã hoàn thành tốt nhiệm vụ của mình để làm nên bài báo cáo của nhóm.

# Mục lục

# Đánh giá thành viên

- Mai Hoàng Hiệp hoàn thành tốt nhiệm vụ được giao đúng hạn: training model, chạy được chương trình, up code và dataset lên github, làm báo cáo latex phần nội dung, thuyết trình. Có tinh thần trách nhiệm đối với nhóm, luôn theo dõi quan sát thông tin có trên nhóm và họp đầy đủ các buổi họp nhóm.
- Trần Trung Hiếu hoàn thành tốt nhiệm vụ được giao đúng hạn: training model, chạy được chương trình, up code và dataset lên github, làm báo cáo latex phần nội dung, thuyết trình. Có tinh thần trách nhiệm đối với nhóm, luôn theo dõi quan sát thông tin có trên nhóm và họp đầy đủ các buổi họp nhóm.
- Nguyễn Duy Đạt hoàn thành tốt nhiệm vụ được giao đúng hạn: làm báo cáo latex, phân chia công việc, làm canva thuyết trình, video thuyết trình, thuyết trình + demo chương trình. Có tinh thần trách nhiệm đối với nhóm, luôn theo dõi quan sát thông tin có trên nhóm và họp đầy đủ các buổi họp nhóm.

# Đánh giá mức độ hoàn thành cho từng yêu cầu

1. Report 100/100%
2. Video 100/100%
3. Slide 100/100%
4. Source Code 100/100%
5. Phân công công việc 100/100%

# Chương 1

## Ngôn ngữ

Ngôn ngữ để viết và trình bày báo cáo khóa luận tốt nghiệp, đồ án tốt nghiệp, thực tập tốt nghiệp (sau đây gọi chung là báo cáo) là tiếng Việt hoặc tiếng Anh. Trường hợp chọn ngôn ngữ tiếng Anh để viết và trình bày báo cáo, sinh viên cần có đơn đề nghị, được cán bộ hướng dẫn (CBHD) đồng ý và nộp cho bộ phận Giáo vụ của Khoa vào thời điểm đăng ký đề tài để xin ý kiến. Báo cáo viết và trình bày bằng tiếng Anh phải có bản tóm tắt viết bằng tiếng Việt.

Tóm tắt luận văn được trình bày nhiều nhất trong 24 trang in trên hai mặt giấy, cỡ chữ Times New Roman 11 của hệ soạn thảo Winword hoặc phần mềm soạn thảo Latex đối với các chuyên ngành thuộc ngành Toán.

Mật độ chữ bình thường, không được nén hoặc kéo giãn khoảng cách giữa các chữ. Chế độ dẫn dòng là Exactly 17pt. Lề trên, lề dưới, lề trái, lề phải đều là 1.5 cm. Các bảng biểu trình bày theo chiều ngang khổ giấy thì đầu bảng là lề trái của trang. Tóm tắt luận án phải phản ánh trung thực kết cấu, bố cục và nội dung của luận án, phải ghi đầy đủ toàn văn kết luận của luận án. Mẫu trình bày trang bìa của tóm tắt luận văn (phụ lục 1).



## Chương 2

# Nội dung chính của báo cáo

### 2.1 Giới thiệu bài toán

Quick, Draw! [wiki] là một trò chơi đoán trực tuyến do Google phát triển và phát hành, thách thức người chơi vẽ một bức tranh về một vật thể hoặc ý tưởng, sau đó sử dụng trí tuệ nhân tạo mạng nơ-ron để đoán xem các bức vẽ đó đại diện cho điều gì. AI học hỏi từ mỗi bức vẽ, cải thiện khả năng đoán đúng trong tương lai. Trò chơi này tương tự như Pictionary ở chỗ người chơi chỉ có thời gian giới hạn để vẽ (20 giây). Các khái niệm mà nó đoán có thể đơn giản, như 'nước', hoặc phức tạp hơn, như 'ngụy trang'.

Chạy một ứng dụng mà bạn có thể vẽ trước máy ảnh (Nếu bạn sử dụng máy tính xách tay, webcam của bạn sẽ được sử dụng theo mặc định)



Hình 2.1: Quick, Draw! by Google

## 2.2 Giải thích phần xử lý dữ liệu, giới thiệu mô hình, giải thích mô hình

### 2.2.1 Xây dựng tập dữ liệu

Quick,Draw thu thập dữ liệu thông qua việc yêu cầu người dùng vẽ các đồ vật cụ thể, chẳng hạn như "ngôi sao", "lá cây", "tia chớp", v.v. Dữ liệu được thu thập trong những lần vẽ của người dùng thông qua giao diện vẽ trực tuyến, và mỗi bản vẽ sẽ được gán nhãn dựa trên tên đồ vật mà người dùng đang vẽ.



Hình 2.2: Xây dựng tập dữ liệu

Dữ liệu vẽ [DataSet] được thu thập từ hàng triệu người dùng trên toàn thế giới, với mỗi người vẽ một đối tượng trong một khoảng thời gian cố định (thường là 20 giây). Những bản vẽ này được ghi lại dưới dạng tọa độ của các điểm vẽ, từ đó có thể tạo ra một đường đi hoặc hình dạng của đồ vật.

Chúng ta sẽ sử dụng dữ liệu đã được Google chuẩn bị sẵn, là những

bộ dữ liệu đã trải qua quá trình tiền xử lý cẩn thận. Dữ liệu này đã được chuẩn hóa tọa độ, chuyển đổi thành các vector đặc trưng, và được biểu diễn dưới dạng ma trận, đảm bảo tính đồng nhất và sẵn sàng cho quá trình huấn luyện. Thay vì tự tay thực hiện tất cả các bước tiền xử lý phức tạp, chúng ta sẽ tải trực tiếp dữ liệu đã được xử lý sẵn từ liên kết mà Google cung cấp. Điều này giúp tiết kiệm thời gian và tài nguyên, đồng thời đảm bảo rằng dữ liệu đầu vào của chúng ta luôn đạt chất lượng cao, hoàn hảo cho việc huấn luyện các mô hình học máy.

Sau khi chuẩn bị xong, chúng ta sẽ tiến hành phân chia dữ liệu theo tỷ lệ 80:20, với 80% dùng cho việc huấn luyện mô hình và 20% còn lại dành cho việc kiểm tra và đánh giá hiệu quả của mô hình. Đồng thời, chúng ta cũng sẽ mã hóa nhãn để đơn giản hóa quá trình xử lý, ví dụ như gán số 0 cho 'quả táo' và số 1 cho 'cuốn sách'. Việc mã hóa nhãn giúp mô hình dễ dàng nhận diện và phân loại các đối tượng, đồng thời tạo sự thuận tiện trong việc huấn luyện và đánh giá kết quả.

## 2.2.2 Giới thiệu Mô hình

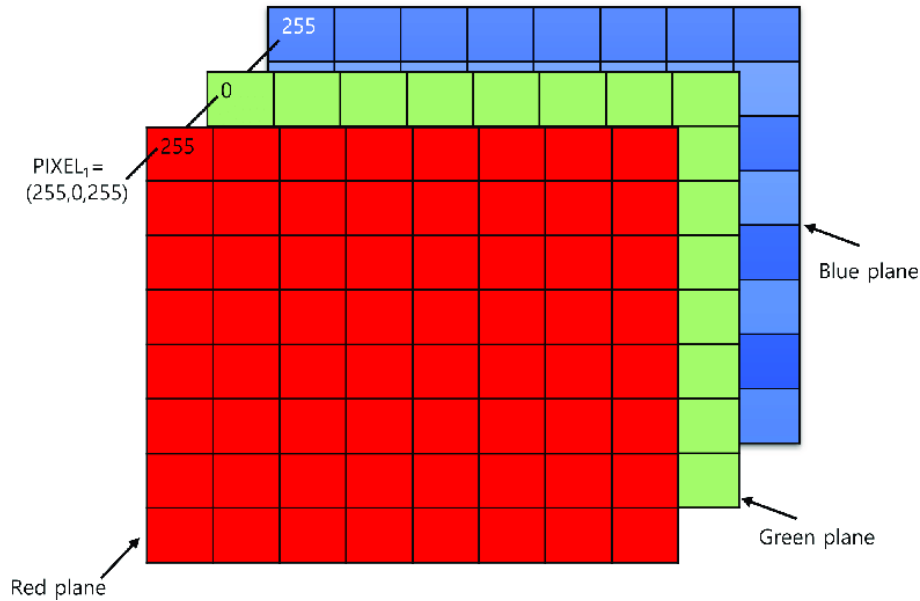
Đoạn mã bạn đưa ra là một mô hình **QuickDraw** [kaggle] được xây dựng bằng PyTorch, sử dụng kiến trúc **Convolutional Neural Network (CNN)** để giải quyết bài toán phân loại hình vẽ tay. Mô hình này bao gồm các lớp **Convolutional Layers (Conv2d)**, **Fully Connected Layers (Linear)**, và các lớp **Dropout** để tránh overfitting.

### CNN là gì?

Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN) [CNN] là một trong những mô hình học sâu phổ biến nhất được sử dụng trong nhận dạng và phân loại hình ảnh. CNN đặc biệt hiệu quả trong việc nhận diện các đối tượng và nhận dạng khuôn mặt, hai lĩnh vực mà mô hình này được áp dụng rộng rãi. Với khả năng xử lý và phân tích các đặc trưng hình ảnh một cách tự động, CNN đã trở thành công cụ mạnh mẽ trong nhiều ứng dụng thực tế liên quan đến thị giác máy tính.

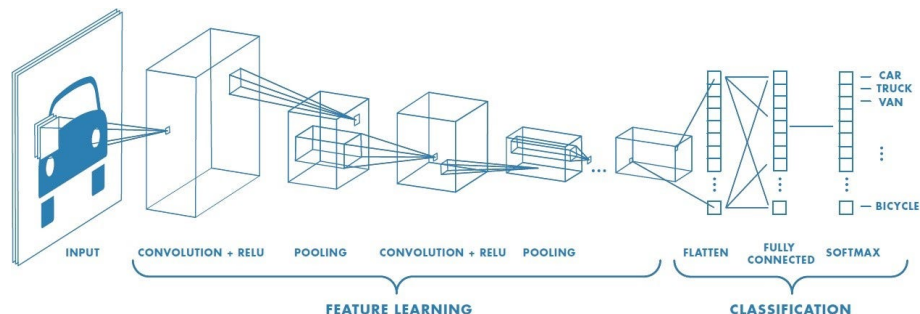
CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy

tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy  $H \times W \times D$  (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB  $6 \times 6 \times 3$  (3 ở đây là giá trị RGB).



Hình 2.3: RGB layers

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernels), tổng hợp lại các lớp được kết nối đầy đủ (Fully Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.

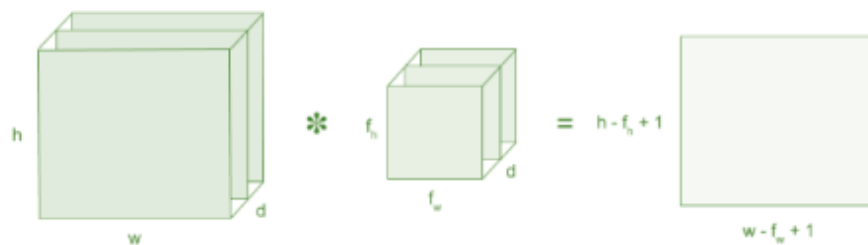


Hình 2.4: Convolutional Neural Networks - CNN

## Lớp tích chập - Convolution Layer

Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vuông nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f<sub>h</sub> x f<sub>w</sub> x d)**
- Outputs a volume dimension **(h - f<sub>h</sub> + 1) x (w - f<sub>w</sub> + 1) x 1**



Xem xét 1 ma trận 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như hình bên dưới.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

**5 x 5 – Image Matrix**






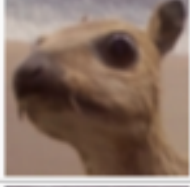



1	0	1
0	1	0
1	0	1

**3 x 3 – Filter Matrix**

Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map'

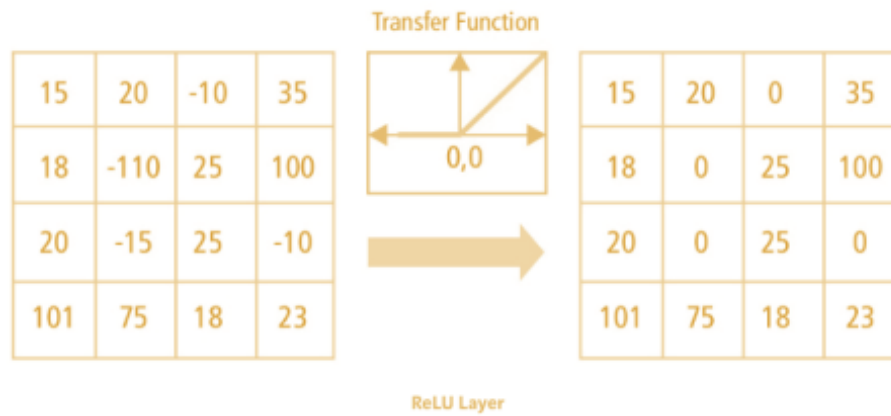
Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau.

Operation	Filter	Convolved Image
<b>Identity</b>	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>Edge detection</b>	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
<b>Sharpen</b>	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
<b>Box blur</b> (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
<b>Gaussian blur</b> (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

## Hàm phi tuyến - ReLU

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là:  $f(x) = \max(0, x)$ .

Tại sao ReLU lại quan trọng: ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm.

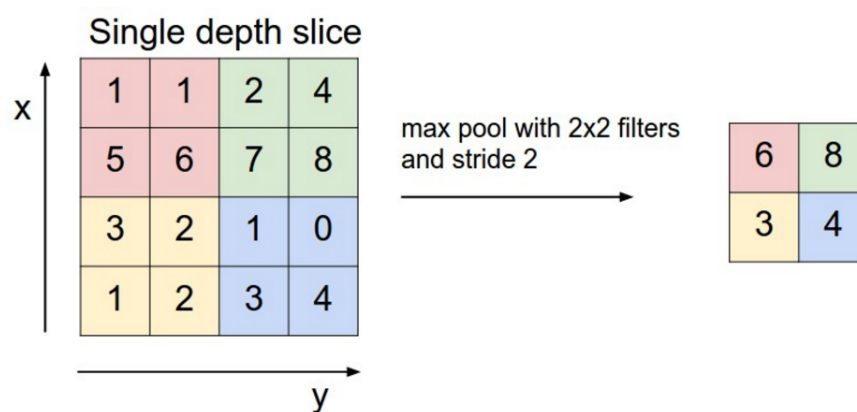


Có 1 số hàm phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.

## Lớp gộp - Pooling Layer

Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau: Max Pooling, Average Pooling, Sum Pooling.

Max pooling lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling.



## Dropout:

Dropout là một kỹ thuật **regularization** (chuẩn hóa) được sử dụng để ngẫu nhiên loại bỏ (tắt) một số nơ-ron trong mạng nơ-ron trong quá trình

huấn luyện. Khi sử dụng dropout, một tỷ lệ phần trăm của các nơ-ron trong mỗi lớp sẽ bị **tắt (drop)** ở mỗi lần huấn luyện, có nghĩa là những nơ-ron này không tham gia vào quá trình tính toán forward và không cập nhật trọng số trong quá trình backward.

- **Cách hoạt động của Dropout:**

- **Trong quá trình huấn luyện:** Khi mạng nơ-ron được huấn luyện, dropout sẽ ngẫu nhiên loại bỏ một số nơ-ron khỏi mạng với xác suất nhất định, thường được gọi là tỷ lệ dropout. Ví dụ, nếu tỷ lệ dropout là 0.5, thì mỗi nơ-ron trong lớp sẽ có xác suất 50% bị tắt đi trong mỗi lần cập nhật trọng số.
- **Trong quá trình dự đoán (inference):** Khi mô hình đã được huấn luyện xong và thực hiện dự đoán trên dữ liệu mới, tất cả các nơ-ron sẽ được kích hoạt và sử dụng đầy đủ. Tuy nhiên, để bù đắp cho việc giảm số lượng nơ-ron trong quá trình huấn luyện, các trọng số của các nơ-ron được nhân với tỷ lệ giữ lại (1 - dropout rate).

- **Mục tiêu của Dropout:**

- **Giảm Overfitting:** Khi huấn luyện mạng nơ-ron, mô hình có thể dễ dàng "học thuộc" dữ liệu huấn luyện, tức là nó có thể ghi nhớ quá mức các đặc trưng cụ thể trong bộ dữ liệu và không thể tổng quát tốt khi gặp dữ liệu mới (dẫn đến overfitting). Dropout giúp giảm thiểu tình trạng này bằng cách ngăn mô hình phụ thuộc quá nhiều vào bất kỳ nơ-ron cụ thể nào, do đó tăng khả năng tổng quát của mô hình.
- **Tạo ra các mạng con ngẫu nhiên:** Dropout có tác dụng như một phương pháp tạo ra các "mạng con" ngẫu nhiên từ mạng lớn, giúp mô hình học được nhiều tính năng từ các đặc trưng khác nhau và không dựa vào những mối quan hệ quá chặt chẽ giữa các nơ-ron.



### 2.2.3 Giải thích Mô hình

#### Các phép toán trong mô hình:

- **Convolution (CNN):** Trong CNN, các lớp convolution giúp nhận diện các đặc trưng cơ bản của hình ảnh (như cạnh, góc, kết cấu) thông qua các bộ lọc (filters).
- **Activation function:** Các hàm kích hoạt (như ReLU, Sigmoid, Tanh) giúp thêm tính phi tuyến vào mô hình, từ đó cho phép mô hình học được các mối quan hệ phức tạp hơn.
- **Pooling layers (CNN):** Giúp giảm kích thước đầu ra của các lớp convolution, tăng tính tổng quát và giảm sự phụ thuộc vào chi tiết không quan trọng.
- **LSTM/GRU (RNN):** Các tế bào này giúp mô hình ghi nhớ thông tin trong các chuỗi thời gian dài hạn, nhờ vào cơ chế quên và ghi nhớ.

#### Khởi tạo mô hình

```
1 def __init__(self, input_size=28, num_classes=15):  
2     super(QuickDraw, self).__init__()
```

**input\_size:** Kích thước của hình ảnh đầu vào. Mặc định là 28, có nghĩa là hình ảnh đầu vào có kích thước 28x28 pixel (ví dụ như hình ảnh MNIST).

**num\_classes:** Số lượng lớp phân loại đầu ra (mặc định là 15). Điều này có nghĩa là mô hình sẽ phân loại hình ảnh vào một trong 15 lớp.

#### Quá trình huấn luyện

##### a. Lớp Convolution đầu tiên (conv1):

```
1     self.conv1 = nn.Sequential(  
2         nn.Conv2d(1, 32, 5, bias=False),  
3         nn.ReLU(inplace=True),  
4         nn.MaxPool2d(2, 2)  
5     )
```

- `nn.Conv2d(1, 32, 5, bias=False)`: Đây là một lớp **convolution** 2D.
  - **1**: Số kênh đầu vào (1 kênh, tức là hình ảnh đen trắng).
  - **32**: Số kênh đầu ra (số lượng bộ lọc, nghĩa là 32 bộ lọc sẽ được áp dụng để trích xuất các đặc trưng từ hình ảnh).
  - **5**: Kích thước của kernel (lọc), tức là mỗi bộ lọc sẽ có kích thước 5x5.
  - **bias=False**: Không sử dụng thông số bias trong phép tính.
- `nn.ReLU(inplace=True)`: Hàm kích hoạt **ReLU** (Rectified Linear Unit), giúp thêm tính phi tuyến vào mô hình. **inplace=True** có nghĩa là các phép toán sẽ được thực hiện trực tiếp trên tensor đầu vào mà không tạo ra bản sao mới.
- `nn.MaxPool2d(2, 2)`: Lớp **MaxPooling** 2D, giúp giảm kích thước không gian của đầu ra từ lớp convolution.
  - **2, 2** có nghĩa là pooling sẽ lấy giá trị tối đa từ mỗi khu vực 2x2 của ma trận đầu vào, làm giảm chiều rộng và chiều cao của bản đồ đặc trưng xuống một nửa.

## b. Lớp Convolution thứ hai (conv2):

```

1 self.conv2 = nn.Sequential(
2     nn.Conv2d(32, 64, 5, bias=False),
3     nn.ReLU(inplace=True),
4     nn.MaxPool2d(2, 2)
5 )

```

- `nn.Conv2d(32, 64, 5, bias=False)`: Lớp convolution thứ hai. Sau lớp `conv1`, đầu vào của lớp này có 32 kênh (từ 32 bộ lọc), và lớp này sẽ áp dụng 64 bộ lọc để trích xuất các đặc trưng cao hơn.
- Các tham số khác tương tự như lớp đầu tiên.
- `nn.ReLU(inplace=True)` và `nn.MaxPool2d(2, 2)` vẫn giữ nguyên tác dụng như lớp đầu tiên, tạo tính phi tuyến và giảm kích thước đặc trưng.

## Lớp Fully Connected:

```
1 dimension = int(64 * pow(input_size/4 - 3, 2))
2 self.fc1 = nn.Sequential(
3     nn.Linear(dimension, 512),
4     nn.Dropout(0.5)
5 )
```

- **dimension**: Tính toán kích thước của đầu vào cho lớp **fc1** (fully connected layer). Sau hai lần pooling (mỗi lần giảm kích thước của hình ảnh xuống một nửa), chiều rộng và chiều cao của hình ảnh sẽ bị giảm xuống còn một phần tư của kích thước ban đầu. Sau đó, chiều dài của đầu vào sẽ được tính bằng cách lấy số lượng kênh đầu ra của lớp **conv2** (64) nhân với diện tích của bản đồ đặc trưng (chiều cao và chiều rộng).
  - **input\_size / 4 - 3**: Sau hai lần pooling, chiều cao và chiều rộng của hình ảnh giảm xuống một nửa mỗi lần. Sau đó, trừ đi 3 vì mỗi lớp convolution có kernel 5x5 và lớp pooling 2x2 làm giảm thêm chiều kích thước.
- **nn.Linear(dimension, 512)**: Lớp **fully connected** (mạng nơ-ron đầy đủ) đầu tiên, có 512 neuron. Đây là lớp dùng để kết nối các đặc trưng trích xuất từ các lớp convolution với các lớp đầu ra.
  - **dimension**: Số lượng đặc trưng đầu vào cho lớp này (được tính ở trên).
  - **512**: Số lượng neuron trong lớp **fc1**.
- **nn.Dropout(0.5)**: Lớp **Dropout** được sử dụng để giảm overfitting. Trong quá trình huấn luyện, nó sẽ "tắt" ngẫu nhiên 50% (tỷ lệ dropout là 0.5) của các neuron trong lớp này, giúp mô hình không quá phụ thuộc vào một số đặc trưng nhất định.

### c. Lớp Fully Connected thứ hai (fc2):

```

1     self.fc2 = nn.Sequential(
2         nn.Linear(512, 128),
3         nn.Dropout(0.5)
4     )

```

- `nn.Linear(512, 128)`: Lớp **fully connected** thứ hai, với 128 neuron. Đầu vào của lớp này là 512 đặc trưng từ lớp `fc1`.
- `nn.Dropout(0.5)`: Giống như lớp trước, lớp Dropout giúp giảm overfitting.

#### d. Lớp Fully Connected cuối cùng (fc3):

```

1     self.fc3 = nn.Sequential(
2         nn.Linear(128, num_classes)
3     )

```

- `nn.Linear(128, num_classes)`: Lớp **fully connected** cuối cùng để tạo ra đầu ra. Số lượng neuron trong lớp này là `num_classes`, tương ứng với số lượng lớp cần phân loại (ví dụ, nếu `num_classes` = 15, đầu ra sẽ là 15 giá trị).

#### Hàm forward:

```

1     def forward(self, input):
2         output = self.conv1(input)
3         output = self.conv2(output)
4         output = output.view(output.size(0), -1)
5         output = self.fc1(output)
6         output = self.fc2(output)
7         output = self.fc3(output)
8         return output

```

- `self.conv1(input)` và `self.conv2(output)`: Dữ liệu đầu vào sẽ đi qua các lớp convolutional (conv1 và conv2) để trích xuất các đặc trưng từ hình ảnh.
- `self.conv1(input)` và `self.conv2(output)`: Dữ liệu đầu vào sẽ đi qua các lớp convolutional (conv1 và conv2) để trích xuất các đặc trưng từ hình ảnh.

- `output.view(output.size(0), -1)`: Sau khi qua các lớp convolutional, dữ liệu có dạng (batch\_size, số kênh, chiều cao, chiều rộng). Dòng này sẽ làm phẳng (flatten) đặc trưng để đưa chúng vào các lớp fully connected. `output.size(0)` là kích thước batch, và `-1` giúp PyTorch tự động tính toán chiều còn lại.
- `self.fc1`, `self.fc2`, `self.fc3`: Dữ liệu sau khi được làm phẳng sẽ đi qua các lớp fully connected để tạo ra đầu ra cuối cùng.

#### 2.2.4 Cách thức hoạt động:

- **Thư viện OpenCV**

**OpenCV (Open Source Computer Vision Library)** là một thư viện mã nguồn mở mạnh mẽ và phổ biến được sử dụng để xử lý ảnh và video trong các ứng dụng thị giác máy tính. Thư viện này cung cấp một loạt các công cụ và thuật toán để thực hiện các nhiệm vụ như nhận diện đối tượng, phân tích hình ảnh, nhận dạng khuôn mặt, xử lý video, và nhiều ứng dụng khác liên quan đến thị giác máy tính.



Hình 2.5: Logo OpenCV [image]

- **Thư viện MediaPipe:**

**MediaPipe** là một thư viện mã nguồn mở được phát triển bởi Google, chuyên cung cấp các công cụ và thuật toán mạnh mẽ để xử lý và phân tích hình ảnh, video, âm thanh trong thời gian

thực. **MediaPipe** chủ yếu được thiết kế để xây dựng các ứng dụng và mô hình thị giác máy tính, học máy, và các tác vụ nhận dạng, phân tích hình ảnh/video với hiệu suất cao và đáp ứng thời gian thực.

- **Thư viện hand:**

**Hand** là một thư viện trong **MediaPipe** được phát triển để nhận diện và theo dõi các cử chỉ của bàn tay trong thời gian thực. Thư viện này giúp phát hiện và theo dõi các điểm đặc trưng của bàn tay, với mục tiêu phục vụ cho các ứng dụng như giao diện người dùng không chạm, thực tế tăng cường (AR), và các hệ thống nhận diện cử chỉ.

- **Cách thức vẽ**

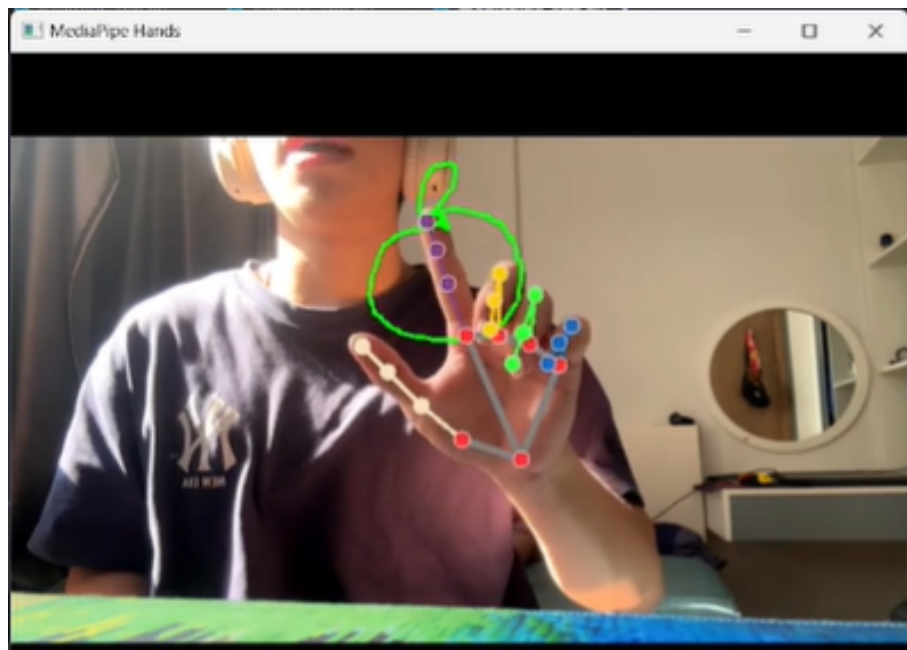
- **Bằng camera\_app:**

- \* Để nhận diện các vật thể có màu đỏ làm trung tâm, ta sẽ sử dụng màu sắc đặc trưng này để xác định vị trí và nét vẽ của người dùng. Quá trình này sẽ bắt đầu bằng việc xác định các vật thể có màu đỏ trong khung hình, từ đó xác định vùng vẽ mà người dùng sẽ thực hiện. Khi người dùng bắt đầu vẽ, họ chỉ cần nhấn phím **backspace** để khởi động quá trình vẽ, hệ thống sẽ nhận diện và theo dõi các nét vẽ trong thời gian thực. Sau khi hoàn tất việc vẽ, người dùng sẽ tiếp tục nhấn lại phím **backspace** để kết thúc quá trình vẽ.
    - \* Khi việc vẽ đã được hoàn tất và người dùng nhấn **backspace**, hệ thống sẽ sử dụng các nét vẽ đã được xác định để chuyển sang giai đoạn dự đoán. Lúc này, mô hình sẽ phân tích các nét vẽ và tiến hành dự đoán kết quả dựa trên các thông tin đã được thu thập. Phương pháp này giúp tạo ra một quá trình đơn giản và mượt mà, nơi người dùng chỉ cần thao tác cơ bản để tương tác với hệ thống, đồng thời giảm thiểu sự phức tạp trong việc sử dụng và điều khiển phần mềm.

- \* Khi người dùng muốn kết thúc chương trình, họ chỉ cần nhấn phím **Esc** để thoát khỏi ứng dụng. Việc này giúp đơn giản hóa quá trình đóng chương trình, không cần phải thực hiện nhiều thao tác phức tạp, chỉ cần một cú nhấn phím duy nhất để chương trình dừng lại và thoát hoàn toàn.

#### – Bằng Hand\_app:

- \* Bằng cách sử dụng thư viện **Hand** để nhận diện các khớp tay của con người, chúng ta có thể tận dụng những chuyển động và vị trí của các ngón tay để tạo ra các nét vẽ. Cụ thể, khi người dùng giơ ngón trỏ lên và các ngón tay còn lại chụm lại, hệ thống sẽ nhận diện hành động này như là việc cầm cây bút. Khi người dùng di chuyển ngón trỏ, hệ thống sẽ hiểu đó là hành động vẽ, và các nét vẽ sẽ được tạo ra tương ứng với chuyển động của ngón tay.



Hình 2.6: Hand<sub>drawing</sub>

- \* Khi người dùng hoàn tất việc vẽ và muốn máy bắt đầu dự đoán kết quả, chỉ cần dang tay ra, mở rộng các ngón tay, và hệ thống sẽ nhận diện điều này như một dấu hiệu để kết thúc quá trình vẽ và chuyển sang giai đoạn dự đoán. Cách thức này giúp tạo ra một tương tác tự nhiên và mượt mà

giữa người dùng và hệ thống mà không cần sử dụng bất kỳ thiết bị ngoại vi nào.

- \* Khi bạn muốn thoát khỏi chương trình, chỉ cần nhấn phím **Esc**.

– **Bằng `drawing_app`:**

- \* Bạn có thể sử dụng chuột để trực tiếp làm công cụ vẽ. Khi bắt đầu, chỉ cần nhấn phím **backspace** và bắt đầu vẽ trên bức tranh của mình. Khi hoàn tất việc vẽ và muốn hệ thống dự đoán, bạn chỉ cần nhấn lại phím **backspace** để kích hoạt quá trình dự đoán. Tương tự, để thoát khỏi chương trình, bạn chỉ cần nhấn phím **Esc**, giúp kết thúc ứng dụng một cách nhanh chóng và dễ dàng.

## 2.3 Experiment

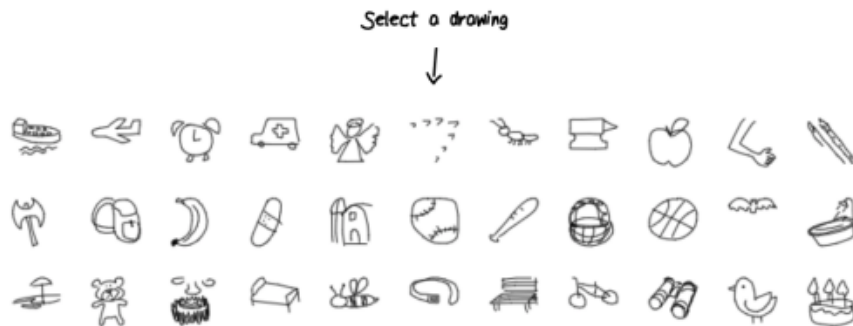
### 2.3.1 Giới thiệu dataset, mô tả dataset( gồm bao nhiêu mẫu, có những gì trong dataset)

Dữ liệu của Quickdraw [**experiment**] được thu thập từ một nguồn do Google cung cấp, nơi mà người dùng trực tiếp vẽ các hình ảnh dựa trên các chủ đề đã được xác định trước. Những hình ảnh này không chỉ phản ánh khả năng vẽ của người dùng mà còn chứa đựng thông tin về cách thức mỗi cá nhân thể hiện một ý tưởng hoặc một đối tượng qua nét vẽ. Sau khi thu thập, các hình ảnh này được lưu trữ trong các tệp dữ liệu dưới dạng các mảng numpy (.npy), một định dạng phổ biến trong lĩnh vực học máy, giúp bảo toàn dữ liệu một cách hiệu quả và dễ dàng trong quá trình xử lý. Thay vì phải thu thập và chuẩn bị dữ liệu từ đầu, chúng ta có thể tận dụng các tệp dữ liệu này, được Google tạo sẵn, để thực hiện các nghiên cứu, phân tích hoặc xây dựng mô hình học máy mà không cần phải lo lắng về việc xử lý dữ liệu thô. Các tệp numpy chứa các mảng với thông tin hình ảnh đã được mã hóa sẵn, do đó chúng ta chỉ cần tải về và sử dụng chúng.



## What do 50 million drawings look like?

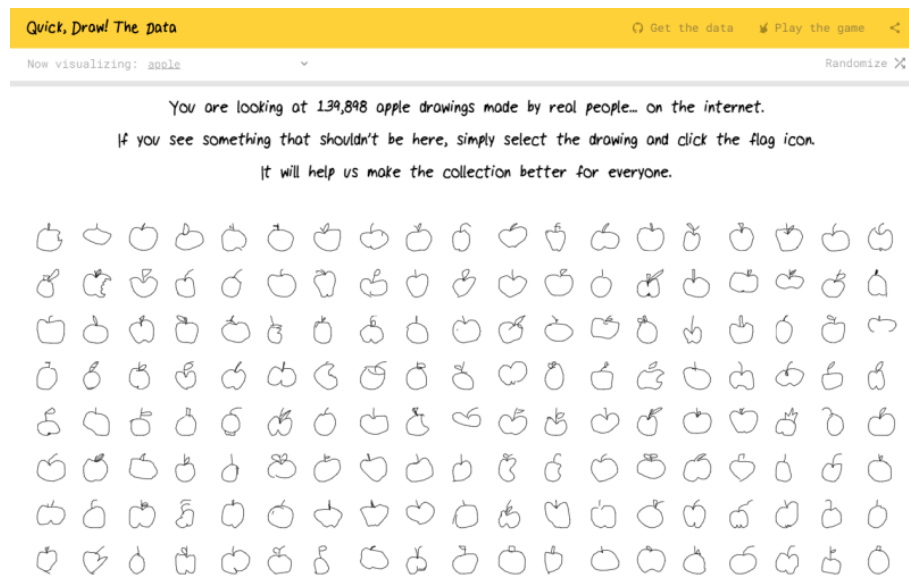
Over 15 million players have contributed millions of drawings playing [Quick, Draw!](#) These doodles are a unique data set that can help developers train new neural networks, help researchers see patterns in how people around the world draw, and help artists create things we haven't begun to think of. That's why [we're open-sourcing them](#), for anyone to play with.



Hình 2.7: Quick, Draw! data

Vì giới hạn về dung lượng bộ nhớ và thời gian xử lý, chúng ta chỉ có thể sử dụng tối đa 20 bộ dữ liệu liên quan đến các đồ vật quen thuộc và dễ nhận diện trong đời sống hàng ngày. Những đồ vật này có thể bao gồm những hình ảnh đơn giản nhưng đầy tính biểu tượng như ngôi sao sáng rực trên bầu trời, chiếc lá cây mỏng manh đang đung đưa trong làn gió nhẹ, hay tia chớp lóe sáng rạch ngang bầu trời đen tối trong những cơn mưa giông. Mặc dù chỉ có 20 bộ dữ liệu, nhưng mỗi bộ chứa đựng những đặc điểm nổi bật và dễ nhận diện, mang lại giá trị trong việc xử lý và phân tích hình ảnh.

Việc lựa chọn những đồ vật quen thuộc này không chỉ giúp giảm thiểu độ phức tạp của bài toán mà còn tối ưu hóa khả năng tính toán, giúp tiết kiệm thời gian và tài nguyên trong quá trình thực hiện. Điều này đặc biệt quan trọng trong môi trường có giới hạn về dung lượng bộ nhớ hoặc khi thời gian xử lý là yếu tố quyết định. Dẫu vậy, với số lượng dữ liệu hạn chế, những đồ vật này vẫn đủ để giúp chúng ta xây dựng các mô hình nhận diện hoặc phân loại cơ bản, đáp ứng yêu cầu của bài toán mà không gặp phải sự quá tải về tài nguyên.

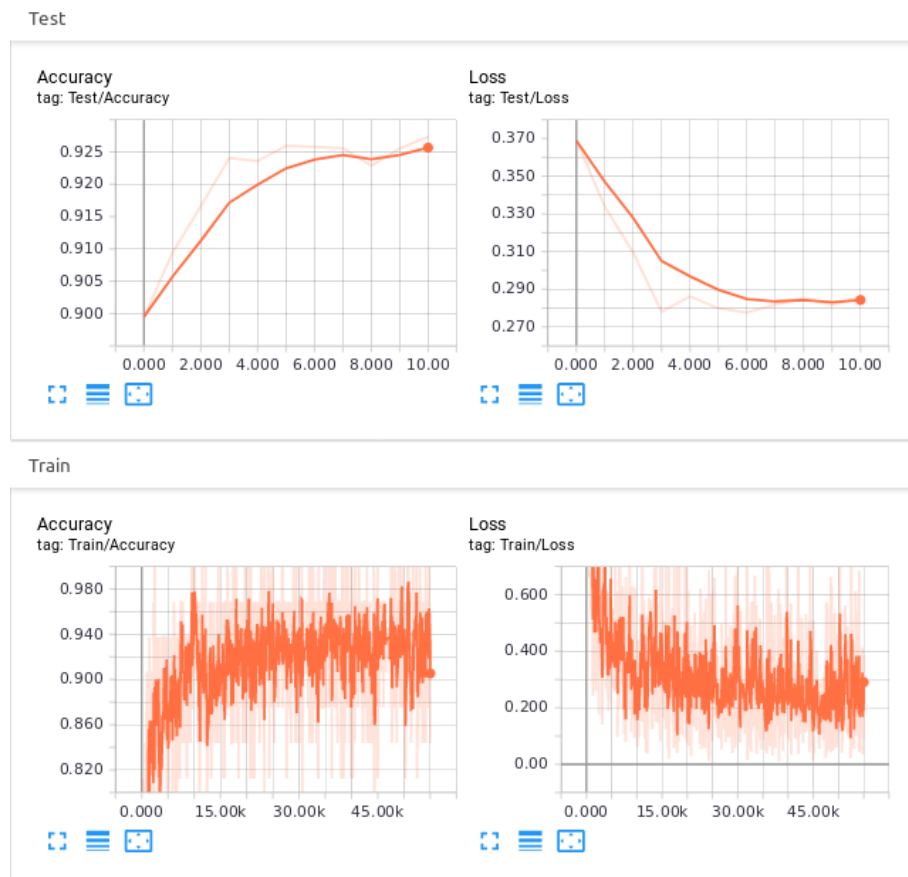


### 2.3.2 Cấu hình chạy thí nghiệm

Để triển khai các bài toán nhận dạng và phân tích hình ảnh trong Python 3.6, chúng ta có thể sử dụng một số thư viện mạnh mẽ như **OpenCV (cv2)**, **PyTorch**, **NumPy**, **Hand**, và **MediaPipe**. **OpenCV (cv2)** là thư viện phổ biến dùng để xử lý hình ảnh và video, với khả năng hỗ trợ nhận diện đối tượng, các phép biến đổi hình ảnh, cũng như các kỹ thuật xử lý ảnh thời gian thực. **PyTorch** là một thư viện học sâu mạnh mẽ, giúp xây dựng và huấn luyện các mô hình học máy, hỗ trợ tính toán tensor và GPU, rất phù hợp cho các tác vụ nhận dạng hình ảnh hoặc học sâu. **NumPy** là thư viện số học cơ bản, cung cấp các cấu trúc dữ liệu mảng mạnh mẽ và các phép toán số học hiệu quả, rất cần thiết khi làm việc với dữ liệu số hoặc hình ảnh. **Hand** và **MediaPipe** là hai thư viện đặc biệt mạnh trong việc xử lý và nhận dạng cử chỉ tay. **MediaPipe** cung cấp các mô hình đã được huấn luyện để phát hiện và theo dõi các điểm quan trọng trên tay người, trong khi **Hand** có thể hỗ trợ phân tích và nhận dạng các cử chỉ tay trong không gian 3D. Việc kết hợp các thư viện này tạo nên một cấu hình mạnh mẽ, hỗ trợ phát triển các ứng dụng nhận diện hình ảnh và phân tích cử chỉ tay, phục vụ cho các hệ thống nhận diện cử chỉ người dùng, tương tác bằng tay, và nhiều ứng dụng khác trong lĩnh vực thị giác máy tính.

### 2.3.3 Kết quả chạy thí nghiệm

Nhìn chung, mô hình [github] có khả năng dự đoán khá chính xác khi nhận diện các vật thể đơn giản. Tuy nhiên, khi gặp những vật thể phức tạp hoặc khó nhận diện, mô hình vẫn gặp phải một số sai sót và nhầm lẫn trong việc dự đoán.



## 2.4 Kết luận

Quick, Draw vẫn là một công cụ rất ấn tượng và thú vị, mang lại trải nghiệm giải trí hấp dẫn và hiệu quả trong việc giải tỏa căng thẳng. Mặc dù mô hình của nó khá đơn giản, nhưng sự cuốn hút từ cách thức người dùng tương tác và sự thú vị trong việc thử thách khả năng vẽ khiến nó trở thành một công cụ giải trí tuyệt vời. Với khả năng nhận diện hình ảnh dựa trên các nét vẽ đơn giản, Quick, Draw vẫn giữ được sự hấp dẫn và dễ tiếp cận, làm cho người dùng không thể rời mắt khỏi trò chơi này.

Mặc dù mô hình có thể đưa ra những dự đoán tương đối chính xác trong hầu hết các trường hợp, nhưng đôi khi kết quả dự đoán vẫn có thể bị sai lệch. Một trong những nguyên nhân chính của sự sai lệch này có thể là do chất lượng của các nét vẽ không được hoàn hảo. Người dùng có thể vẽ một cách vội vàng, không rõ ràng hoặc thậm chí là không đúng với hình dạng chuẩn mà mô hình đã được huấn luyện. Bên cạnh đó, các yếu tố khác như độ mờ của nét vẽ, sự khác biệt trong cách vẽ của từng người hay các biến thể trong cách thể hiện các đối tượng có thể gây khó khăn cho mô hình khi nhận diện và phân loại hình ảnh.

Ngoài ra, mặc dù bộ dữ liệu của Quickdraw bao gồm hàng trăm ngàn mẫu hình ảnh, nhưng không thể tránh khỏi sự tồn tại của một số sai sót, chẳng hạn như những nét vẽ sai hoặc không chính xác từ người dùng. Những sai sót này có thể xảy ra do nhiều yếu tố như người dùng vẽ vội, không chú ý, hoặc thiếu kỹ năng vẽ. Khi các dữ liệu này được sử dụng trong quá trình huấn luyện mô hình, chúng có thể gây ra một số sai lệch nhỏ trong kết quả dự đoán. Những "lỗi" này, dù là nhỏ, vẫn có thể ảnh hưởng đến độ chính xác của mô hình trong một số trường hợp. Tuy nhiên, đây là một vấn đề không thể hoàn toàn tránh khỏi khi làm việc với dữ liệu thực tế, đặc biệt khi dữ liệu được thu thập từ nhiều người dùng khác nhau với các mức độ kỹ năng vẽ và cách thể hiện rất khác nhau.

## Chương 3

# Báo cáo làm việc hàng tuần

### 3.1 Họp lần 1

(a) Ngày họp: 6/12/2024

(b) Thành viên tham dự:

- Mai Hoàng Hiệp
- Trần Trung Hiếu
- Nguyễn Duy Đạt

(c) Phân công:

- Nội dung phân công:
  - cài đặt và chạy được mô hình sau khi tham khảo [1, 2, 3]
  - up source code lên github [2]
  - làm nội dung ghi vào Latex và slide [1, 3]
  - làm báo cáo bằng Latex [3]
  - làm slide [1, 2, 3]
  - làm video thuyết trình [1, 2]
- Thời gian làm: 7/12/2024 - 18/12/2024
- Người được phân công:
  - Mai Hoàng Hiệp [1], Trần Trung Hiếu [2], Nguyễn Duy Đạt [3]

(d) Tiến độ làm tuần trước:

- Nội dung phân công: Không
- Người được phân công: Không

- Tiến độ làm: Mai Hoàng Hiệp đã tham khảo được các nguồn tài liệu, video hướng dẫn về Quick, Draw!.
- Công việc đã làm: tạo github của nhóm để up source code, tham khảo các tài liệu liên quan.
- Công việc chưa làm: training và chạy được mô hình, làm nội dung ghi vào Latex và Slide, làm báo cáo bằng Latex, làm slide thuyết trình, video thuyết trình + demo chương trình, up source code lên github.

## 3.2 Họp lần 2

(a) Ngày họp: 11/12/2024

(b) Thành viên tham dự:

- Mai Hoàng Hiệp
- Trần Trung Hiếu
- Nguyễn Duy Đạt

(c) Phân công:

- Nội dung phân công:
  - đọc hiểu lại từng phần code [1, 2, 3]
  - hoàn thiện nội dung của báo cáo bằng Latex [1, 3]
  - làm slide thuyết trình [1, 2, 3]
  - làm video thuyết trình + demo chương trình [1, 2]
- Thời gian làm: 11/12/2024-18/12/2024
- Người được phân công:
  - Mai Hoàng Hiệp [1], Trần Trung Hiếu [2], Nguyễn Duy Đạt [3]

(d) Tiến độ làm tuần trước:

- Nội dung phân công: hoàn tất các bước của chương trình yêu cầu, hoàn thành quá trình training, viết tiếp báo cáo Latex, viết nội dung trình chiếu, làm video, làm slide, up source code lên github.

- Người được phân công: Mai Hoàng Hiệp, Trần Trung Hiếu, Nguyễn Duy Đạt.
- Tiến độ làm: làm tiếp Latex, hoàn thành source code, hoàn thành quá trình training, chạy được mô hình hoạt động ổn định.
- Công việc đã làm: Nội dung thêm vào Latex (từ đầu đến chapter 1), source code up lên github.
- Công việc chưa làm: Latex (các phần chapter 2, chapter 3 và references), slide thuyết trình, Video.

### 3.3 Họp lần 3

(a) Ngày họp: 17/12/2024

(b) Thành viên tham dự:

- Mai Hoàng Hiệp
- Trần Trung Hiếu
- Nguyễn Duy Đạt

(c) Phân công:

- Nội dung phân công:
  - Làm video thuyết trình bằng cách ghi âm chèn vào slide [1, 2, 3]
  - hoàn thiện slide [1, 3]
  - hoàn thiện Latex [2, 3]
- Thời gian làm: 17/12/2024 - 18/12/2024
- Người được phân công:
  - Mai Hoàng Hiệp [1], Trần Trung Hiếu [2], Nguyễn Duy Đạt [3]

(d) Tiến độ làm tuần trước:

- Nội dung phân công: hoàn thiện source code, hoàn thiện Latex, Slide, Video.
- Người được phân công: Mai Hoàng Hiệp, Trần Trung Hiếu, Nguyễn Duy Đạt.

- Tiến độ làm: đang làm tiếp Latex, hoàn thiện slide thuyết trình.
- Công việc đã làm: Latex (chapter 2 và references), source code, slide.
- Công việc chưa làm: Video, Latex (chapter 3).



# Tài liệu tham khảo