**NHẬN DẠNG HÌNH ẢNH VẼ TAY DỰA TRÊN MẠNG NƠ-RON TÍCH CHẬP SÂU**

Bùi Văn Tân, Lương Minh Tùng, Trần Huy Hoàng

Khoa Công nghệ thông tin, trường Đại học Kinh tế - Kỹ thuật Công nghiệp.

**HAND-DRAWN IMAGE RECOGNITION BASED ON DEEP CONVOLUTIONAL NEURAL NETWORK**

| **Tóm tắt:** | Thị giác máy tính là một lĩnh vực nghiên cứu quan trọng trong khoa học máy tính, lĩnh vực này nghiên cứu những mô hình xử lý, phân tích hình ảnh để hiểu và trích xuất thông tin từ chúng. Đối với bài toán nhận dạng hình ảnh, việc nhận dạng hình ảnh vẽ tay là một bài toán khó hơn so với ảnh chụp. Trong bài báo này, chúng tôi đề xuất một mô hình nhận dạng hình ảnh vẽ tay sử dụng mạng nơ-ron tích chập học sâu (Hand-Drawn Image Recognition - HDIR). Mô hình HDIR được thiết kế dựa trên các kiến trúc mạng phổ biến như VGG và ResNet, tuy nhiên, nó được tinh chỉnh để phù hợp với dữ liệu hình ảnh vẽ tay. Để đánh giá hiệu năng của mô hình HDIR, chúng tôi đã thực nghiệm trên một tập dữ liệu lớn bao gồm các hình ảnh vẽ tay thuộc nhiều chủ đề khác nhau. Kết quả thực nghiệm cho thấy rằng mô hình HDIR có khả năng nhận dạng các hình ảnh vẽ tay với độ chính xác cao. Nodejs Nodejs |  |
| --- | --- | --- |
| **Từ khóa:** | Mạng nơ-ron tích chập, thị giác máy tính, nhận dạng hình ảnh, nhận dạng hình ảnh vec tay. |  |
| **Abstract:** | Computer vision is an important area of study in computer science, which studies processing patterns and analyzes images to understand and extract information from them. For the image recognition problem, the hand-drawn image recognition is a more difficult problem than the photo. In this paper, we propose a hand-drawn image recognition model using a deep learning convolutional neural network (Hand-Drawn Image Recognition - HDIR). The HDIR model is designed based on popular network architectures such as VGG and ResNet, however, it is fine-tuned to fit hand-drawn image data. To evaluate the performance of the HDIR model, we experimented on a large data set consisting of hand-drawn images of many different topics. Experimental results show that the HDIR model is capable of recognizing hand-drawn images with high accuracy. |  |
| **Keywords:** | CNN, Computer vision, Image Recognition, Hand-Drawn Image Recognition. | |

**1. GIỚI THIỆU.**

Trên không gian mạng, mỗi ngày có hàng triệu hình ảnh số được tạo ra và chúng cần được phân loại để dễ dàng sắp xếp, truy cập, phân tích và xử lý với tốc độ nhanh hơn. Phân loại hình ảnh là một quá trình phân loại hoặc tách các hình ảnh thành các danh mục khác nhau, dựa trên những đặc trưng của chúng. Phân loại ảnh là một bài toán quan trọng và có nhiều ứng dụng. Cho đến thời điểm hiện tại, phân loại chính xác ảnh số vẫn còn là một thách thức bởi lẽ hình ảnh có thể bị nhiễu, mờ, nền lộn xộn hoặc ảnh chất lượng thấp. Các kỹ thuật phân loại ảnh chủ yếu được chia thành ba loại: Phân loại có giám sát; Phân loại không giám sát; Phân loại bán giám sát. Quá trình phân loại hình ảnh được thực hiện theo các bước như sau:

• Tiền xử lý hình ảnh: Bước này được sử dụng để cải thiện chất lượng hình ảnh bằng cách loại bỏ các biến dạng không mong muốn và làm nổi bật các đặc trưng quan trọng. Các bước tiền xử lý hình ảnh bao gồm đọc hình ảnh, thay đổi kích thước hình ảnh và tăng cường dữ liệu (tỷ lệ xám, phản chiếu, làm mờ gaussian, biểu đồ, cân bằng, xoay và dịch).

• Phát hiện đối tượng: Bước này nhắm đến phân đoạn ảnh và xác định vị trí của đối tượng được quan tâm trong ảnh.

• Trích xuất đặc trưng và huấn luyện: Bước này triển khai các phương pháp thống kê hoặc học sâu để xác định các mẫu và đặc điểm thú vị nhất của hình ảnh có thể là duy nhất cho một lớp cụ thể, sau này giúp mô hình phân biệt giữa các lớp khác nhau. Quá trình này, trong đó mô hình học các tính năng từ tập dữ liệu được gọi là đào tạo mô hình..

• Phân loại đối tượng: Bước này phân loại các đối tượng được phát hiện thành các lớp được xác định trước bằng cách sử dụng một kỹ thuật phân loại phù hợp để so sánh các mẫu hình ảnh với các mẫu mục tiêu.

Trong bài báo này, chúng tôi trình bày tổng quan về bài toán phân loại hình ảnh vẽ tay dựa trên kiến trúc mạng nơ-ron tích chập học sâu. Trong đó phần II trình bày mô hình phân lớp ảnh sử dụng kiến trúc mạng nơron tích chập. Phần III bài viết đễ xuất mô hình HDIR cho bài toán nhận dạng hình ảnh vẽ tay. Phần IV trình bày các thực nghiệm và đánh giá. Cuối cùng, kết luận và hướng phát triển được trình bày trong phần V của bài viết.

**2.** **PHÂN LỚP HÌNH ẢNH DỰA TRÊN MẠNG NƠ-RON TÍCH CHẬP (Hoàng)**

Phân loại ảnh là một quá trình phức tạp và tốn nhiều thời gian nếu thực hiện thủ công, với việc thực hiện các phương pháp phân loại ảnh khác nhau, quá trình này có thể được tự động hóa để thu được kết quả có độ chính xác cao. Mạng nơ-ron tích chập gồm nhiều lớp với hai thành phần quan quan trọng gồm: lớp tích chập và các lớp tổng hợp.

Rathi và các cộng sự [1] trình bày quy trình phân loại loài cá dưới nước tự động với độ chính xác 96,29% trái ngược với các phương pháp tiếp cận truyền thống nhấn mạnh vào việc phân loại cá bên ngoài nước vì việc phân loại cá dưới nước gặp một số thách thức như tiếng ồn xung quanh, biến dạng, phân biệt đối tượng , chất lượng hình ảnh và tắc nghẽn. Phương pháp đề xuất đã thực hiện việc loại bỏ nhiễu trong tập dữ liệu thông qua việc thực hiện xử lý ảnh trước bước đào tạo giúp loại bỏ các chướng ngại vật dưới nước, bụi bẩn và các vật thể không phải cá khỏi ảnh. Bước tiếp theo sử dụng phương pháp học sâu bằng cách triển khai mạng nơ ron tích chập để phân loại các loài cá. Bài báo cũng cung cấp sự hiểu biết về hiệu suất của các chức năng kích hoạt khác nhau với việc so sánh ReLU, SoftMax và tanh và chức năng kích hoạt ReLU được cho là có độ chính xác cao.

Sanghvi và các cộng sự [2] đề xuất phương pháp phân loại tự động ảnh động vật sử dụng mạng nơ ron tích chập trên bộ dữ liệu Animal-10 với độ chính xác 91,84%. Nó đề cập đến việc triển khai mạng thần kinh tích chập với kiến trúc VGG16 và chức năng kích hoạt Leaky ReLU để phân loại hình ảnh động vật. Mạng nơ-ron được đào tạo để phân loại hình ảnh của động vật và giúp xác định lớp động vật. Việc phân loại hình ảnh động vật được đề xuất sử dụng mạng nơ-ron tích chập có thể được sử dụng rộng rãi để phân loại hình ảnh động vật, điều này sẽ hỗ trợ các nhà sinh thái học và nhà nghiên cứu nghiên cứu sâu hơn và/hoặc cải thiện môi trường sống, mô hình môi trường và sự tuyệt chủng.

Shaha và các cộng sự [3] sử dụng học chuyển giao để tinh chỉnh các tham số mạng được đào tạo trước để phân loại hình ảnh. Các tác giả đã thực hiện hai thử nghiệm trên hai cơ sở dữ liệu hiện đại, GHIM10K và CalTech256. Thử nghiệm đầu tiên phân tích hiệu suất của kiến trúc VGG19 cho nhiệm vụ phân loại hình ảnh. Cùng với VGG19, phân tích hiệu suất của AlexNet và VGG16 trên cơ sở dữ liệu GBHIM10K đã được so sánh. Thử nghiệm thứ hai bao gồm việc sử dụng cơ sở dữ liệu CalTech256 để đánh giá hiệu suất của kiến trúc VGG19 cho tác vụ phân loại hình ảnh. Hiệu suất của mô hình AlexNet, VGG16 và VGG19 được so sánh cùng với Recall, Precision và F1-Score và với kết quả của cả hai thử nghiệm, VGG19 được cho là chính xác nhất.

**3. ĐỀ XUẤT MÔ HÌNH**

**3.1. Mô hình phân lớp ảnh sử dụng mạng nơ-ron tích chập**

Ba lớp cơ bản được sử dụng trong một kiến trúc mạng nơ-ron tích chập gồm: Lớp tích chập (Convolution layer- CONV) , Lớp gộp (Pooling layer - POOL), Lớp liên kết đầy đủ (Fully connected - FC).

Một lớp tích chập là khối xây dựng chính của mạng tích chập (CNN). Nó bao gồm một tập hợp các bộ lọc (hoặc kernel), các tham số của chúng được học trong quá trình huấn luyện. Kích thước của bộ lọc thường nhỏ hơn kích thước thực tế của hình ảnh. Mỗi bộ lọc tích chập với hình ảnh và tạo ra một bản đồ kích hoạt. Để tích chập, bộ lọc trượt qua chiều cao và chiều rộng của hình ảnh và tính tích vô hướng giữa mỗi phần tử của bộ lọc và đầu vào tại mỗi vị trí không gian. Bản đồ kích hoạt được tạo ra bằng cách lặp lại quá trình này cho mỗi phần tử của hình ảnh đầu vào. Thể tích đầu ra của lớp tích chập được tạo ra bằng cách xếp chồng các bản đồ kích hoạt của mỗi bộ lọc theo chiều sâu. [4]

Lớp gộp làm giảm kích thước không gian, đồng thời giảm số lượng tham số và lượng tính toán trong mạng bằng cách giữ lại thông tin quan trọng nhất. Các lớp gộp hoạt động trên mỗi bản đồ kích hoạt một cách riêng biệt. Để gộp, đầu vào được chia thành các vùng con hình chữ nhật không giao nhau, có kích thước bằng kích thước của bộ lọc, và thông tin được trích xuất từ mỗi vùng con. Phương pháp thường được sử dụng là MaxPooling. [5]

Một tầng kết nối đầy đủ có thể được coi như một mạng perceptron đa tầng truyền thống. Kết nối đầy đủ có nghĩa là tất cả các nút trong tầng trước đều được kết nối với tất cả các nút trong tầng tiếp theo. Tầng kết nối đầy đủ là một cách khác để học kết hợp phi tuyến giữa các đặc trưng có độ sâu khác nhau. Nó có thể được sử dụng làm tầng đầu ra, trong đó bản đồ đặc trưng cuối cùng sẽ được chuyển đổi thành kết quả phân loại với kết nối đầy đủ. Trong trường hợp này, hàm kích hoạt softmax thường được sử dụng. [6]

Mô hình mạng nơ-ron phân lớp ảnh bằng được định nghĩa như sau:

*model = Sequential()*

*model.add(Conv2D(filters = 32, kernel\_size = (5,5),padding = 'Same',activation ='relu', input\_shape = (150,150,3)))*

*model.add(MaxPooling2D(pool\_size=(2,2)))*

*model.add(Conv2D(filters = 64, kernel\_size = (3,3),padding = 'Same',activation ='relu'))*

*model.add(MaxPooling2D(pool\_size=(2,2), strides=(2,2)))*

*model.add(Conv2D(filters =96, kernel\_size = (3,3),padding = 'Same',activation ='relu'))*

*model.add(MaxPooling2D(pool\_size=(2,2), strides=(2,2)))*

*model.add(Conv2D(filters = 96, kernel\_size = (3,3),padding = 'Same',activation ='relu'))*

*model.add(MaxPooling2D(pool\_size=(2,2), strides=(2,2)))*

*model.add(Flatten())*

*model.add(Dense(512))*

*model.add(Activation('relu'))*

*model.add(Dense(5, activation = "softmax"))*

Đầu vào của mô hình là một ảnh có kích thước 150x150 và 3 kênh màu (RGB), sau đó được đưa qua các lớp tích chập (Conv2D) với các bộ lọc có kích thước khác nhau. Lớp pooling (MaxPooling2D) với kích thước ma trận 2x2 để giảm kích thước đầu ra của lớp tích chập và giảm số lượng tham số cần tối ưu. Quá trình giảm kích thước sẽ thực hiện bằng cách chọn giá trị lớn nhất (max) trong vùng được phủ bởi bộ lọc và đưa giá trị đó vào đầu ra của lớp. Tham số strides = (2,2) chỉ định bước nhảy (stride) của bộ lọc giảm kích thước. Lớp fully connected (Dense) với 512 đơn vị là số lượng nơ-ron trong lớp Dense, được áp dụng để kết nối các đặc trưng học được từ các lớp tích chập. Một lớp fully connected cuối cùng, đầu ra của lớp này là một vector có độ dài bằng số lượng lớp cần phân loại.

**3.2. Mô hình HDIR**

*model = keras.Sequential()*

*model.add(Conv2D(16, (3, 3), padding='same', input\_shape=x\_train.shape[1:], activation='relu'))*

*model.add(MaxPooling2D(pool\_size=(2, 2)))*

*model.add(Conv2D(32, (3, 3), padding='same', activation= 'relu'))*

*model.add(MaxPooling2D(pool\_size=(2, 2)))*

*model.add(Conv2D(64, (3, 3), padding='same', activation= 'relu'))*

*model.add(MaxPooling2D(pool\_size =(2,2)))*

*model.add(Flatten())*

*model.add(Dense(128, activation='relu'))*

*model.add(Dense(100, activation='softmax'))*

*# Train model*

*adam = tf.optimizers.Adam()*

*model.compile(loss='categorical\_crossentropy', optimizer=adam, metrics=['accuracy'])*

*print(model.summary()*

*Model: "sequential\_1"*

- Hai lớp Convolution2D lần lượt có các tham số đầu vào: Filers(32-64), kernel\_size(5,5), hàm activation ReLu.

- Với padding = 'Same' thì đầu ra của lớp tích chập sẽ có cùng kích thước với đầu vào bằng cách thêm các giá trị zero-padding vào xung quanh ảnh đầu vào. Khi đó, bộ lọc có thể trượt trên các cạnh của ảnh đầu vào mà không bị mất thông tin.

- Hai lớp MaxPooling2D gồm pooling\_size(2,2) và strides(2,2), padding: same (tức là thêm các số 0 vào cạnh rìa của input, sử dụng padding sẽ làm features map có kích thước [I/S] để thuận tiện cho kích thước đầu ra của bộ lọc với I là độ dài kích thước đầu vào, S là độ trượt).

-Flatten() dùng để chuyển dữ liệu về mảng 1 chiều, hàm densen() dùng để trả về số unit (neuron) trong mô hình và tham số thứ 2 là hàm kích hoạt được sử dụng (ReLU). Có thể thấy trong ví dụ này số neuron được sử dụng lần lượt là 512 và 126.

-Sau khi khởi tạo, ta cần tối ưu hóa các hàm mất mát, số liệu, … Trong ví dụ ta sử dụng hàm mất mát categoical\_crossentropy, trình tối ưu hóa adam, và chỉ số độ chính xác metrics.

-Hàm ReLU được sử dụng để cải thiện độ chính xác (ngoài ra có thể sử dụng hàm Sigmoid hay tanh)

-Hàm softmax được tính bằng cách lấy một vector đầu vào và biến đổi nó thành một vector đầu ra có cùng số chiều, trong đó mỗi phần tử trong vector đầu ra là giá trị xác suất của phần tử tương ứng trong vector đầu vào. Cụ thể, giá trị xác suất của phần tử thứ i trong vector đầu vào được tính bằng công thức:



| Kí hiệu | Ý nghĩa |
| --- | --- |
|  | Giá trị vector nhập vào cho hàm softmax, (từ 20 đến 2k) |
| i | Tất cả các giá trị z đều là giá trị vector nhập cho hàm softmax. Chúng có thể là bất cứ số thực nào, số dương, số âm hay số 0. Ví dụ một mạng thần kinh nhân tạo có thể có giá trị vector ra là (-0.62, 8.12, 2.53). Đây không phải là phân phối xác suất đúng. Đó là vì sao ta cần đến hàm softmax. |
|  | Hàm lũy thừa tiêu chuẩn được áp dụng cho mỗi giá trị nhập. Nó sẽ đưa ra một giá trị dương lớn hơn 0. Giá trị này sẽ rất nhỏ nếu giá trị nhập là âm, và rất lớn nếu giả trị nhập dương. Tuy nhiên, nó sẽ không cố định trong khoảng (0:1]. Đây là yêu cầu của một xác suất. |
|  | Dòng phía dưới của công thức là một cụm chuẩn hóa. Nó đảm bảo rằng tổng của các giá trị ra sẽ luôn bằng 1 và nằm trong khoảng (0:1]. Như vậy, sẽ xuất hiện - phân phối xác suất chính xác. |
|  | Số class trong một phân loại nhiều class. |

- Tác dụng chính của hàm softmax là biến đổi đầu vào thành một phân phối xác suất, cho phép ta dễ dàng áp dụng các phương pháp phân loại dựa trên xác suất, chẳng hạn như lựa chọn lớp có xác suất cao nhất làm đầu ra của mô hình.

**IV. Thực nghiệm và đánh giá**

1) Thực nghiệm phân loại hoa

1.1.Bộ dữ liệu

Bộ dữ liệu này chứa 5000 hình ảnh của hoa[[1]](#footnote-0). Các hình ảnh được chia thành năm lớp: hoa cúc, hoa tulip, hoa hồng, hoa hướng dương và hoa sen. Đối với mỗi lớp có 1000 ảnh. Ảnh không được giảm về kích thước đơn lẻ, chúng có tỷ lệ khác nhau.

1.2. Thiết lập siêu tham số cho mô hình

batch\_size: xác định số lượng mẫu sử dụng trong mỗi batch trong quá trình huấn luyện.

epochs: xác định số lần tập dữ liệu huấn luyện sẽ được lặp lại trong quá trình huấn luyện.

steps\_per\_epoch: xác định số lượng batch được xử lý cho mỗi epoch. Thông thường, nó được đặt là số lượng mẫu huấn luyện chia cho kích thước batch.

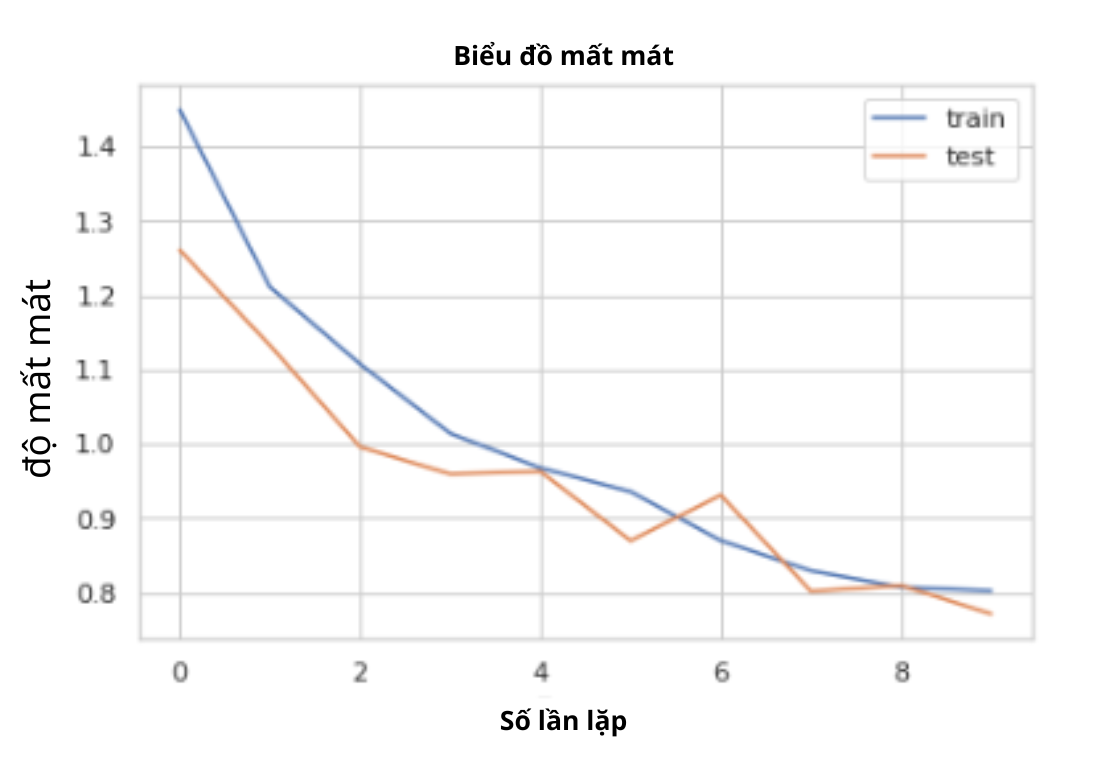
datagen: là một đối tượng sinh dữ liệu (data generator object) có thể thực hiện các hoạt động tăng cường dữ liệu như xoay ngẫu nhiên, dịch chuyển, phóng to / thu nhỏ, vv. trên dữ liệu huấn luyện để tăng kích thước bộ dữ liệu và cải thiện hiệu suất của mô hình.

verbose: xác định mức độ cộng tác trong quá trình huấn luyện. Giá trị 1 có nghĩa là thanh tiến trình (progress bar) trên màn hình, hiển thị phần trăm hoàn thành của quá trình huấn luyện.

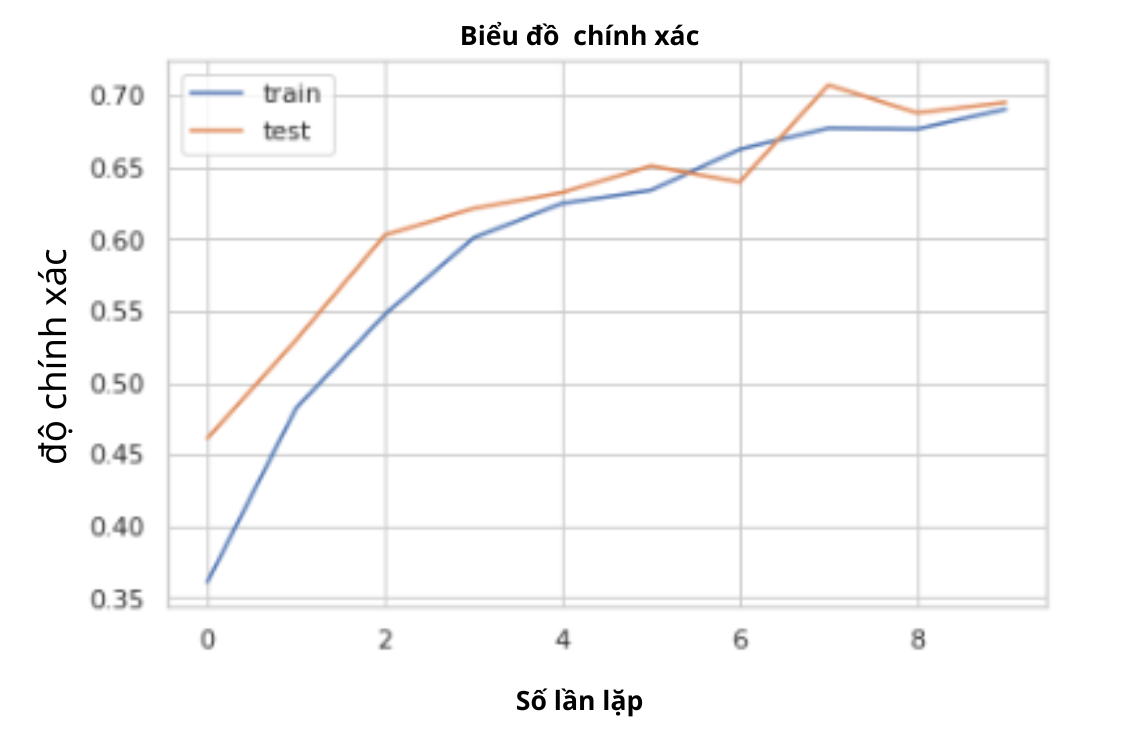
validation\_data: là một bộ đôi chứa dữ liệu xác thực được sử dụng trong quá trình huấn luyện. Nó bao gồm hình ảnh xác thực và các nhãn tương ứng.

1.3. kết quả thực nghiệm

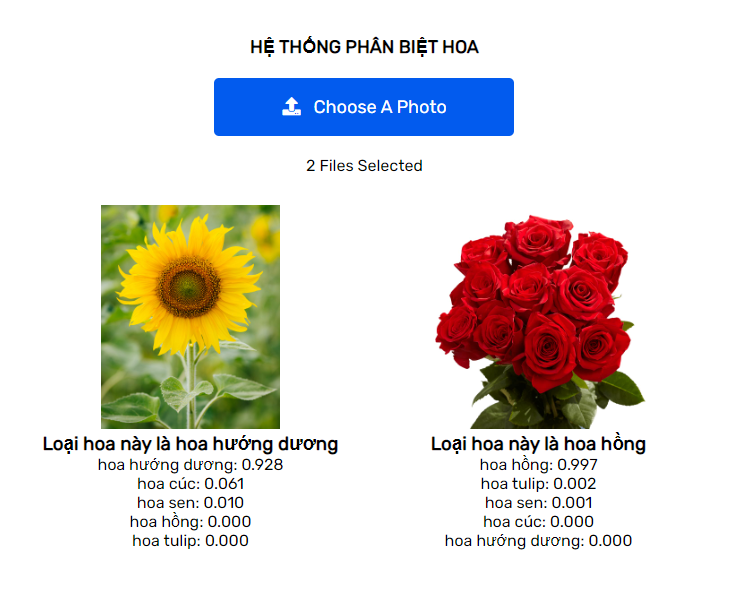
Mô hình được huấn luyện với số lần lặp là 25, cho thấy giá trị của hàm mất mát giảm và độ chính xác của mô hình được tăng lên sau mỗi lần lặp.



Đây là biểu đồ đánh giá hàm mất mát của mô hình theo chiều đi xuống của tập dữ liệu train và test.



Đây là biểu đồ đánh giá tỷ lệ Accuracy của mô hình đang theo chiều đi lên của tập dữ liệu train và test. Chứng tỏ mô hình hoạt động tốt.



1. **Thực nghiệm nhận dạng hình ảnh vẽ tay**

2.1.Bộ dữ liệu

* Dữ liệu là các tệp hình vẽ tay được Google thu thập từ 50 triệu người dùng trên toàn thế giới.
* Các tập dữ liệu này đã được dán nhãn và làm đẹp (đưa về mảng 1 chiều 784) vậy nên khi huấn luyện cần lưu ý reshape dữ liệu về 28x28 để thuận tiện.

VD: ảnh về 1 chiếc đồng hồ sau khi reshape

* Các tệp dữ liệu này có thể tải miễn phí tại trang Github:

<https://github.com/googlecreativelab/quickdraw-dataset>

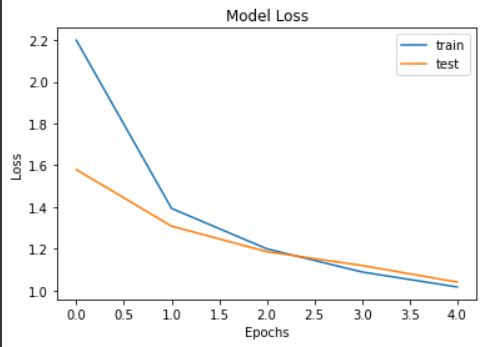
* Trong mô hình đang nghiên cứu ta sử dụng các tệp “Numpy bitmap files” (.npy). Tải và sử dụng tại:

<https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/numpy_bitmap;tab=objects?prefix=&forceOnObjectsSortingFiltering=false>

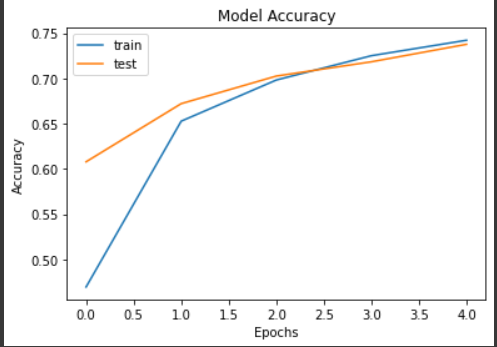
2.2.Kết quả thực nghiệm

* Model qua 10 epochs:

*History = model.fit(x = x\_train, y = y\_train, validation\_data=(x\_test,y\_test), batch\_size = 512, verbose=1, epochs=5)*



Biểu đồ của Loss và Val\_loss



Biểu đồ của Accuracy và Val\_Accuracy

2.3.Hyper-paramter

* random\_state = 0, test\_size = 0.1, epochs = 10, batch\_size = 512, learning\_rate=0.01, verbose=1, validation\_data=(x\_test,y\_test) 🡪 có thể sử dụng validation\_split=0.1

V. Kết luận

Trong bài viết này, chúng tôi đã đề xuất một mô hình có khả năng phân loại chính xác các hình ảnh vẽ tay. Mô hình đã thực nghiệm trên một tập dữ liệu lớn bao gồm các hình ảnh vẽ tay với chủ đề đa dạng. Kết quả thực nghiệm cho thấy rằng mô hình của chúng tôi đạt được độ chính xác cao hơn so với các mô hình cơ sở được so sánh.

**TÀI LIỆU THAM KHẢO**

[1] Rathi, D., Jain, S., & Indu, S. (2017, December). Underwater fish species classification using convolutional neural network and deep learning. In *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)* (pp. 1-6). IEEE.

[2]. Sanghvi, K., Aralkar, A., Sanghvi, S., & Saha, I. (2020). Fauna Image Classification using Convolutional Neural Network. *International Journal of Future Generation Communication and Networking*, *13*(1s), 08-16.

[3] Shaha, M., & Pawar, M. (2018, March). Transfer learning for image classification. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 656-660). IEEE.

[4] Sakib Mostafa, Fang-Xiang Wu. (2021). Neural Engineering Techniques for Autism Spectrum Disorder.

[5] Jason Brownlee. (2019, July). A Gentle Introduction to Pooling Layers for Convolutional Neural Networks.

[6] Keming Mao, Duo Lu, Dazhi E and Zhenhua Tan. (2018, June). A Case Study on Attribute Recognition of Heated Metal Mark Image Using Deep Convolutional Neural Networks.

1. <https://www.kaggle.com/datasets/hoangkaggle/flower-images> [↑](#footnote-ref-0)