

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CƠ KHÍ CHẾ TẠO MÁY**

**BỘ MÔN CƠ ĐIỆN TỬ**



**HCMUTE**

**BÁO CÁO CUỐI KỲ**

**Môn: Trí tuệ nhân tạo**

**Đề tài**

**Nhận Dạng Thức Ăn Đường Phố Việt Nam Và  
Tính Chất Từng Món**

**GVHD: PGS. TS Nguyễn Trường Thịnh**

**SVTH: Nguyễn Văn Tâm**

**MSSV: 19146384**

**TP. Hồ Chí Minh, Tháng 6/2022**

## MỤC LỤC

<b>A. Mở đầu .....</b>	<b>2</b>
1. Giới thiệu đề tài.....	3
2. Mục tiêu đề tài.....	3
3. Phạm vi .....	3
<b>B. Nội Dung .....</b>	<b>3</b>
1. Tổng quan về đề tài .....	3
1.1. Deep learning.....	3
1.1.1. Deep learning là gì? .....	3
1.1.2. Cách thức hoạt động .....	5
1.2. Tổng quan về thư viện Tensorflow .....	5
1.2.1. Tensorflow là gì? .....	5
1.2.2. Cách thức hoạt động .....	6
2. Tính chất của các món ăn .....	7
3. Mô hình CNN- Convolutional Network.....	8
3.1. Convolutional là gì? .....	8
3.2. Cấu trúc của mạng CNN .....	10
3.2.1. Convolutional layer .....	10
3.2.2. Pooling layer .....	11
3.2.3. Fully connected layer.....	12
3.3. Tìm hiểu về keras.....	13
4. Các bước huấn luyện mô hình .....	14
5. Huấn luyện mô hình Google Colab.....	14
5.1. Tiến lập trình lập.....	15
6. Kết quả.....	23
7. Kết Luận .....	24

## DANH MỤC HÌNH ẢNH

Hình 1: Mạng lưới kết nối dữ liệu .....	4
Hình 2: Convolution.....	9
Hình 3: Convolved .....	10
Hình 4: Convolutional layer .....	11
Hình 5: Pooling layer .....	11
Hình 6: Max pooling và average pooling .....	12
Hình 7: Fully connected layer .....	13
Hình 8: Deep learning framework .....	14
Hình 9: Độ chính xác của quá trình training .....	23
Hình 10: Nhận diện thức ăn bằng hình ảnh.....	23
Hình 11: Nhận diện realtime bằng colab.....	24
Hình 12: Giao diện load dữ liệu.....	24
Hình 13: Giao diện app .....	24

## **A. Mở đầu**

### **1. Giới thiệu đề tài**

Thức ăn đường phố là các loại thức ăn, đồ uống được chế biến sẵn, các món này được phục vụ tại chỗ và là thức ăn nhanh, chi phí rẻ hơn một bữa ăn trong các nhà hàng và nhanh chóng, tiện lợi, giá cả phải chăng. Theo Tổ chức Lương thực và Nông nghiệp Liên Hợp Quốc (FAO) thì khoảng 2,5 tỷ người ăn thức ăn đường phố mỗi ngày. Thức ăn đường phố có mối liên hệ mật thiết Take-out, đồ ăn vặt (hàng rong, quà vặt), đồ ăn nhẹ (snack), thức ăn nhanh, nó được phân biệt bởi hương vị địa phương và được mua trên đường phố, mà không cần nhập bất kỳ trụ sở hay công trình xây dựng gì.

Trên thế giới đã có một số công trình nghiên cứu về cứu nhận dạng các loại thức ăn bằng cách vận dụng kỹ thuật xử lý ảnh và nhận dạng đã đạt được một số kết quả khả quan. Các hệ thống nhận dạng được nhiều loại thức ăn và đồ uống, từ đó dự đoán và gợi ý về chế độ dinh dưỡng cho người dùng, bảo đảm cung cấp đủ chất dinh dưỡng cho người dùng.

Nhận dạng thức ăn thông qua xử lý ảnh bằng thuật toán CNN, để nhận diện đúng loại và tính chất của món ăn, từ đó có thể đưa ra dự đoán chính xác về thành phần dinh dưỡng cung cấp cho người dùng. Vì vậy, em đã chọn nghiên cứu về đề tài “Nhận dạng thức ăn đường phố Việt Nam” một ứng dụng nhận dạng dựa vào hình ảnh mang tính chất của món ăn đó.

### **2. Mục tiêu đề tài**

Tìm hiểu về các loại thức ăn đường phố đặt trưng ở Việt Nam và xây dựng nhận dạng các loại thức ăn thông qua kỹ thuật xử lý ảnh. Từ đó đưa ra các tính chất của từng món.

### **3. Phạm vi**

Phạm vi thực hiện dựa trên những kiến thức thu được từ những môn học trong quá trình học tập tại trường và tham khảo thêm một số tài liệu bên ngoài để bổ sung thêm kiến thức nhằm hoàn thiện tốt hơn đáp ứng được mục tiêu của đề ra.

## **B. Nội Dung**

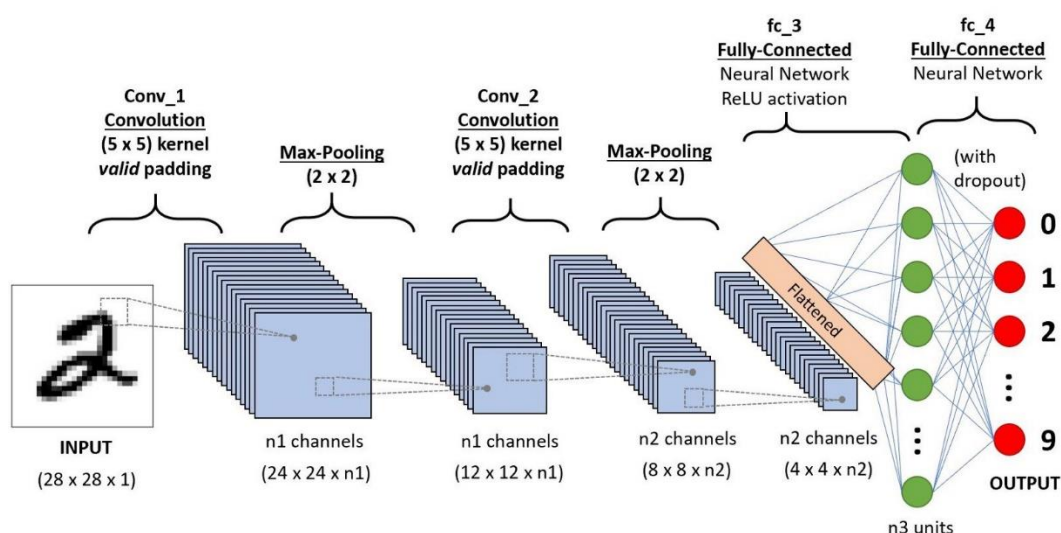
### **1. Tổng quan về đề tài**

#### **1.1. Deep learning**

##### **1.1.1. Deep learning là gì?**

Deep learning được bắt nguồn từ thuật toán **Neural network** vốn xuất phát chỉ là một ngành nhỏ của Machine Learning. Deep Learning là một chi của ngành máy học dựa trên một tập hợp các thuật toán để cố gắng mô hình dữ liệu trừu tượng hóa ở mức cao bằng cách sử dụng nhiều lớp xử lý với cấu trúc phức tạp, hoặc bằng cách khác bao gồm nhiều biến đổi phi tuyến.

Tương tự như cách chúng ta học hỏi từ kinh nghiệm thuật toán, deep learning sẽ thực hiện một nhiệm vụ nhiều lần mỗi lần tinh chỉnh nhiệm vụ một chút để cải thiện kết quả. Deep Learning chỉ đơn giản là kết nối dữ liệu giữa tất cả các tế bào thần kinh nhân tạo và điều chỉnh chúng theo dữ liệu mẫu.



Hình 1: Mạng lưới kết nối dữ liệu

Càng có nhiều tế bào thần kinh được thêm vào thì kích thước của dữ liệu sẽ càng lớn. Nó tự động có tính năng học tập ở nhiều cấp độ trừu tượng. Qua đó cho phép một hệ thống học hàm ánh xạ phức tạp mà không phụ thuộc vào bất kỳ thuật toán cụ thể nào. Không ai thực sự biết những gì xảy ra trong một mạng lưới thần kinh nhân tạo. Vì vậy, hiện tại bạn có thể gọi Deep Learning là một cái hộp đen.

Deep Learning đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 15 năm trước: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú

thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn, phim, ảnh, âm nhạc.

### **1.1.2. Cách thức hoạt động**

Cách thức hoạt động của thuật toán Deep Learning diễn ra như sau: Các dòng thông tin sẽ được trải qua nhiều lớp cho đến lớp sau cùng. Lấy quy trình học của con người làm ví dụ cụ thể. Qua các lớp đầu tiên sẽ tập trung vào việc học các khái niệm cụ thể hơn trong khi các lớp sâu hơn sẽ sử dụng thông tin đã học để nghiên cứu và phân tích sâu hơn trong các khái niệm trừu tượng. Quy trình xây dựng biểu diễn dữ liệu này được gọi là trích xuất tính năng.

Để có thể dễ hình dung về Deep Learning chúng ta sẽ tìm hiểu cách nó hoạt động thông qua một số ví dụ sau. Hãy bắt đầu với một ví dụ đơn giản về Deep Learning ở cấp độ khái niệm. Hãy cùng suy nghĩ làm thế nào để chúng ta có thể nhận biết được một hình nào đó là hình vuông.

Có thể đầu tiên bạn sẽ kiểm tra xem hình đó có 4 cạnh hay không, nếu nó đúng chúng ta sẽ kiểm tra tiếp 4 cạnh này có được kết nối với nhau thành 1 hình tứ giác hay không, nếu đúng chúng ta sẽ kiểm tra tiếp 4 cạnh này có vuông góc với nhau không và chúng có kích thước bằng nhau không. Nếu tất cả đều đúng thì kết quả nó là hình vuông

Nhìn chung thì cũng không có gì phức tạp cả nó chỉ là 1 hệ thống phân cấp các khái niệm. Chẳng hạn như ví dụ ở trên chúng ta đã chia nhiệm vụ xác định hình vuông thành những nhiệm vụ nhỏ và đơn giản hơn. Deep Learning cũng hoạt động tương tự như vậy nhưng ở quy mô lớn hơn.

## **1.2. Tổng quan về thư viện Tensorflow**

### **1.2.1. Tensorflow là gì?**

Với sự bùng nổ của lĩnh vực Trí Tuệ Nhân Tạo – A.I. trong thập kỷ vừa qua, machine learning và deep learning rõ ràng cũng phát triển theo cùng. Và ở thời điểm

hiện tại, TensorFlow chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

Các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model.

Kiến trúc TensorFlow hoạt động được chia thành 3 phần:

- Tiền xử lý dữ liệu
- Dựng model
- Train và ước tính model

Tensor đại diện cho các loại dữ liệu được đưa vào trong Tensorflow. Mỗi thuộc tính trong tensor sẽ có những đặc điểm và tính năng khác nhau. Để giúp bạn hiểu hơn về Tensorflow, Bizfly sẽ giới thiệu ngay đến bạn những thuộc tính cơ bản sau đây:

- Rank: Trong các cấu trúc dữ liệu, thuộc tính bậc được hiểu một cách đơn giản chính là sự phân cấp bậc và là căn cứ cho việc phân loại các tensor. Mỗi tensor khi được phân bậc sẽ có tên gọi khác nhau, cụ thể bậc 0 là Scalar, bậc 1 là Vector bậc 2 là Matrix, các bậc cao hơn nữa sẽ được gọi là n-tensor.
- Shape: Đây là thuộc tính chiều của tensor các cấu trúc dữ liệu.
- Type: Kiểu dữ liệu của các element và là thuộc tính type duy nhất có trong tensor. Một tensor chỉ có một loại type duy nhất cho toàn bộ các element có trong tensor. Vì vậy mà cấu trúc dữ liệu có tính thống nhất.

### **1.2.2. Cách thức hoạt động**

TensorFlow cho phép các lập trình viên tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 sê-ri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học, và mỗi kết nối hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là ‘tensor’.

TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python. Vì Python khá dễ học và làm việc, ngoài ra còn cung cấp nhiều cách tiện lợi để ta hiểu được làm thế nào các high-level abstractions có thể kết hợp cùng nhau. Node và tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow bản thân chúng cũng là các ứng dụng Python.

Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau. Các ứng dụng TensorFlow có thể chạy trên hầu hết mọi mục tiêu thuận tiện: máy cục bộ, cụm trong đám mây, thiết bị iOS và Android, CPU hoặc GPU. Nếu bạn sử dụng đám mây của riêng Google, bạn có thể chạy TensorFlow trên silicon Đơn vị xử lý TensorFlow (TPU) tùy chỉnh của Google để tăng tốc hơn nữa. Tuy nhiên, các mô hình kết quả được tạo bởi TensorFlow, có thể được triển khai trên hầu hết mọi thiết bị nơi chúng sẽ được sử dụng để phục vụ dự đoán. TensorFlow 2.0, được ra mắt vào tháng 10 năm 2019, cải tiến framework theo nhiều cách dựa trên phản hồi của người dùng, để dễ dàng và hiệu quả hơn khi làm việc cùng nó (ví dụ: bằng cách sử dụng các Keras API liên quan đơn giản cho việc train model). Train phân tán dễ chạy hơn nhờ vào API mới và sự hỗ trợ cho TensorFlow Lite cho phép triển khai các mô hình trên khá nhiều nền tảng khác nhau. Tuy nhiên, nếu đã viết code trên các phiên bản trước đó của TensorFlow thì bạn phải viết lại, đôi lúc 1 ít, đôi lúc cũng khá đáng kể, để tận dụng tối đa các tính năng mới của TensorFlow 2.0.

## **2. Tính chất của các món ăn**



Bánh khoai mỡ chiên, có lớp vỏ vàng giòn, bên trong mềm dẻo thơm mùi khoai mỡ. Bánh thơm, vị béo béo ngọt ngọt vừa ăn.

Bánh tiêu là bánh thường có hình tròn, được làm từ bột và xung quanh có mè, khi chiên lên có màu vàng, khi ăn có vị ngọt và giòn giòn của mè.

Bánh Tráng Nướng là món nổi tiếng của các tỉnh đồ khi đi Đà Lạt, bánh tráng làm bánh có hình tròn và thường có trứng, xúc xích, chả, ... thường có màu sắc rất đẹp

Bánh ướt là món làm từ bột được hấp chín, với vị nướng mỡ đặt trung thường có ăn chung với chả lụa

Bánh Xèo là món quen thuộc với người dân Việt Nam nhưng mỗi miền có các kích thước khác nhau thường được làm từ bột xay pha với nước và được đổ lên khuôn và có một chút thịt băm và rau.

Bò né là có vẻ như là món ở nước ngoài nhưng nó được chế biến theo đúng khẩu vị của người VN, nó có trứng, xúc xích, được để trên một khay nóng và thường ăn chung với lẩu salad

Bún Xào là món ăn làm bún xào được nêm nếm và có thêm các carot được cắt mỏng.

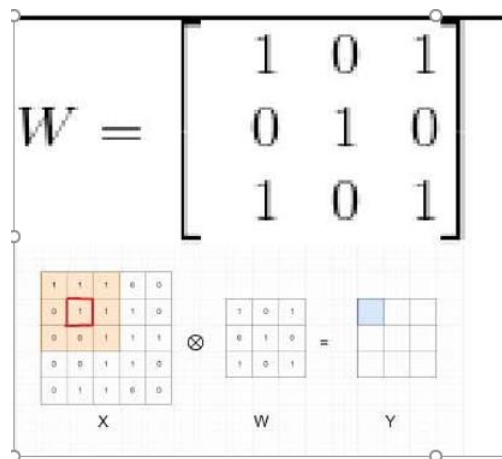
### **3. Mô hình CNN- Convolutional Network**

#### **3.1. Convolutional là gì?**

Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature.

Trong hình ảnh ví dụ trên, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước  $5 \times 5$  và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểm của dòng và cột.

Convolution hay tích chập là nhân từng phần tử trong ma trận 3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là  $3 \times 3$ .



*Hình 2: Convolution*

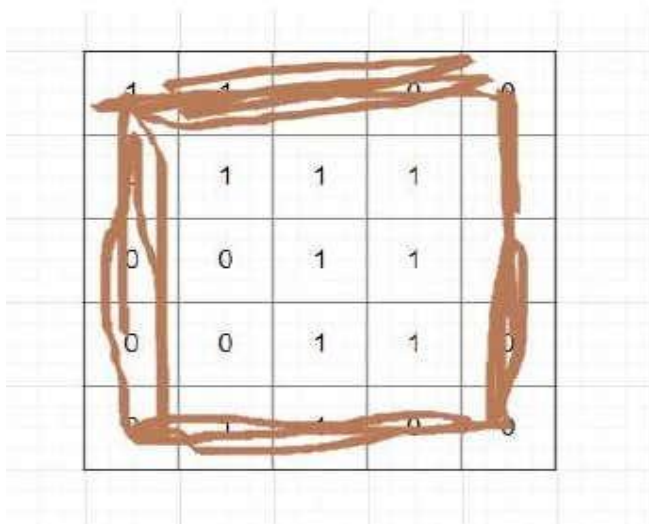
Convolution hay tích chập là nhân từng phần tử bên trong ma trận 3×3 với ma trận bên trái. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhân ma trận Filter với ma trận ảnh 5×5 bên trái.

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

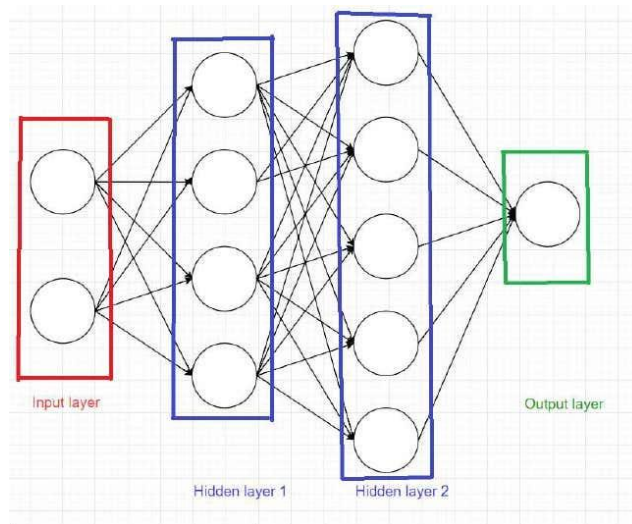


Hình 3: Convolved

Các bước thực hiện phép tính convolution cho ma trận X với kernel K ở trên

## 3.2. Cấu trúc của mạng CNN

### 3.2.1. Convolutional layer

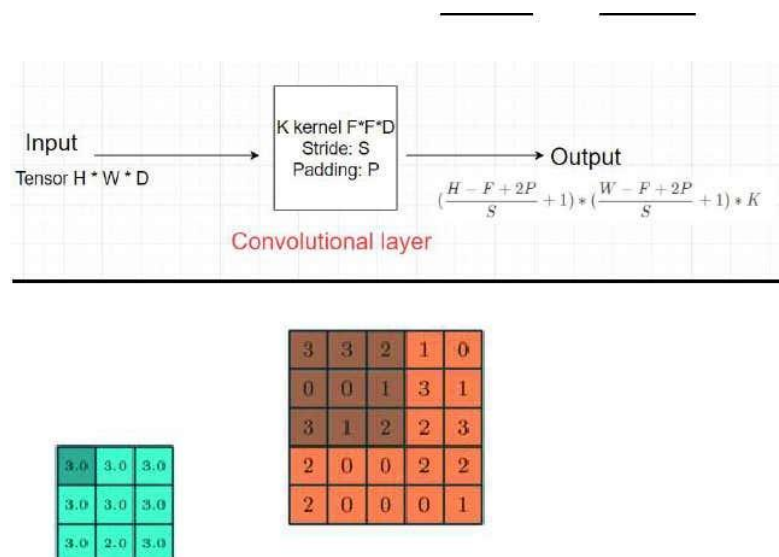


Hình 4: Convolutional layer

Mô hình neural network

Mỗi hidden layer được gọi là fully connected layer, tên gọi theo đúng ý nghĩa, mỗi node trong hidden layer được kết nối với tất cả các node trong layer trước. Cả mô hình được gọi là fully connected neural network (FCN).

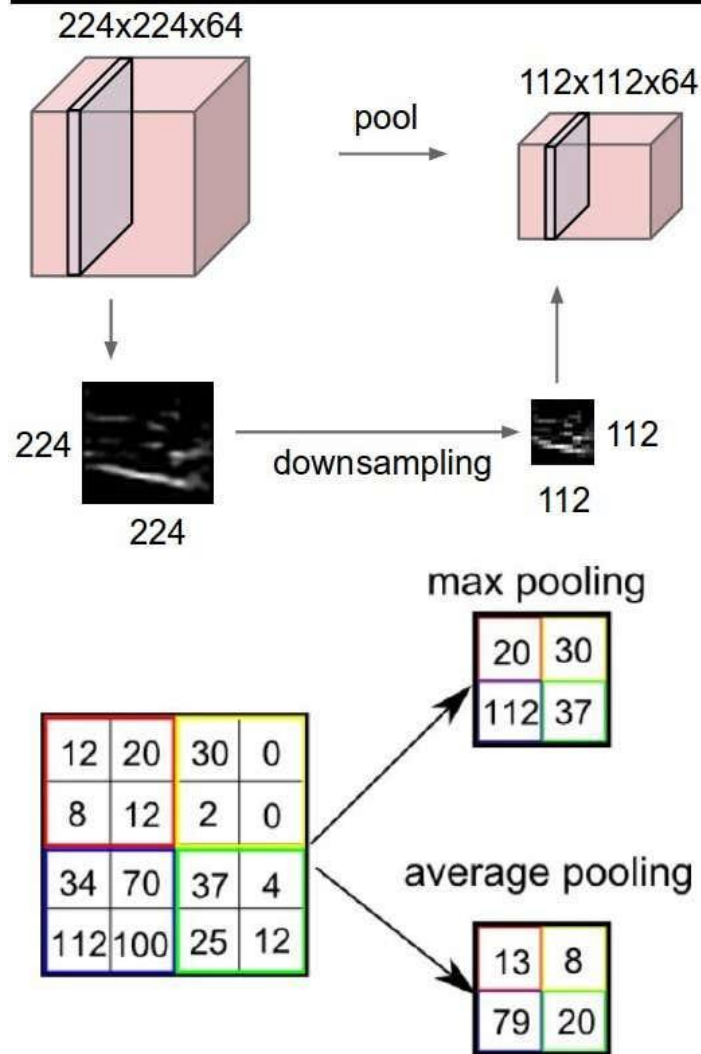
### 3.2.2. Pooling layer



Hình 5: Pooling layer

Max pooling layer với size= (3,3), stride=1, padding=0

Có 2 loại pooling layer phổ biến là: max pooling và average pooling

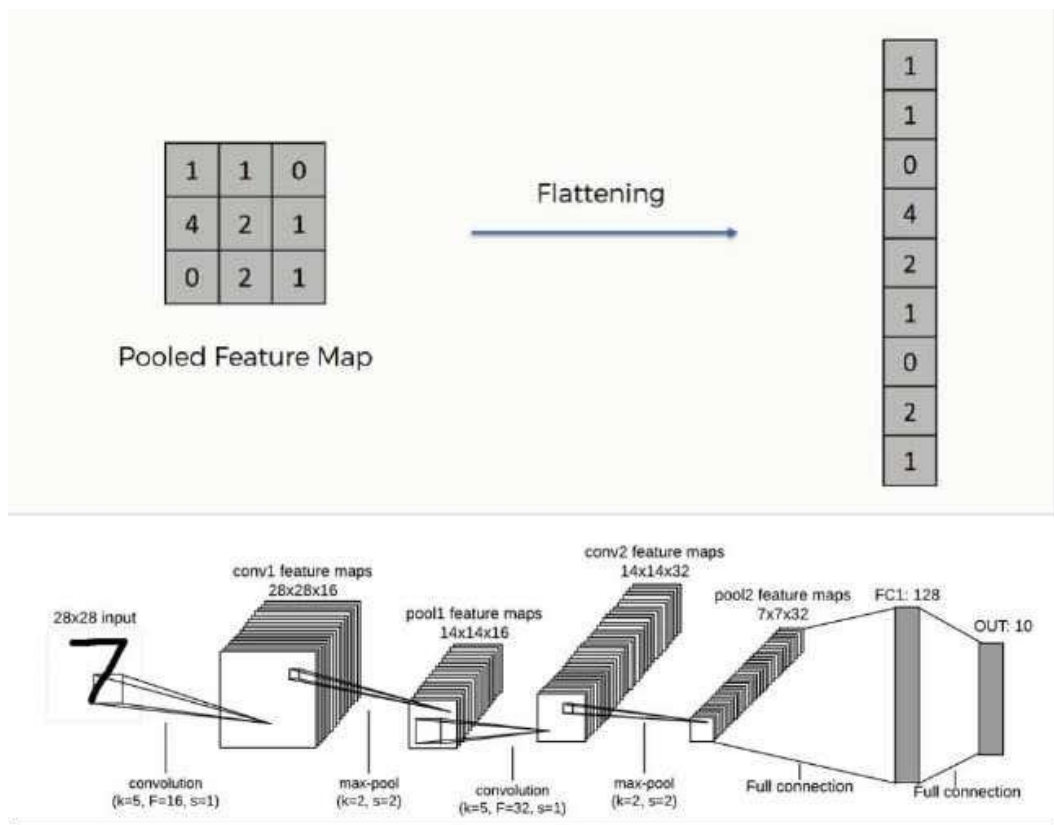


Hình 6: Max pooling và average pooling

### 3.2.3. Fully connected layer

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được

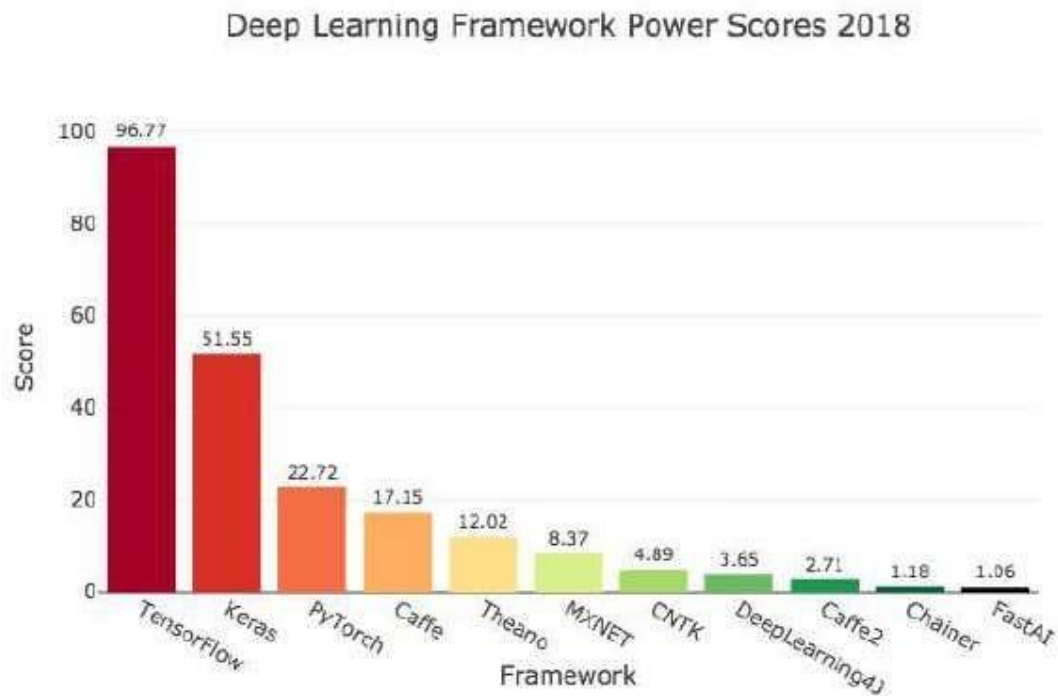
Tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,...) thì tensor của output của layer cuối cùng, kích thước  $H \times W \times D$ , sẽ được chuyển về 1 vector kích thước  $(H \times W \times D)$ .



Hình 7: Fully connected layer

### 3.3. Tìm hiểu về keras

Keras là một framework mã nguồn mở cho deep learning được viết bằng Python. Nó có thể chạy trên nền của các deep learning framework khác như: tensorflow, theano, CNTK. Với các API bậc cao, dễ sử dụng, dễ mở rộng, keras giúp người dùng xây dựng các deep learning model một cách đơn giản.



*Hình 8: Deep learning framework*

#### 4. Các bước huấn luyện mô hình

1. Xây dựng bài toán
2. Chuẩn bị dữ liệu (dataset)
3. Xây dựng model
4. Định nghĩa loss function
5. Thực hiện backpropagation và áp dụng gradient descent để tìm các parameter gồm weight và bias để tối ưu loss function.
6. Dự đoán dữ liệu mới bằng model với các hệ số tìm được ở trên
7. Realtime bằng webcam

#### 5. Huấn luyện mô hình Google Colab

Huấn luyện (hay còn gọi là train) một mô hình Deep Learning cần xử lý lượng phép tính lớn hơn nhiều so với các mô hình Machine Learning khác. Để cải thiện tốc độ tính toán, người ta dùng GPU (Graphics Processing Unit) thay cho CPU (Central Processing Unit) vì với 1 GPU cho phép xử lý phép tính song song với rất nhiều core sẽ nhanh hơn nhiều so với CPU. Tuy nhiên giá của GPU thì khá đắt đỏ để mua hoặc thuê server có

GPU. Thế nên Google đã cung cấp Google Colab miễn phí có GPU để chạy code python (deep learning) cho mục đích nghiên cứu.

Ở trên môi trường Colab có cài sẵn các thư viện Deep Learning phổ biến như PyTorch, TensorFlow, Keras, ... Ngoài ra bạn cũng có thể cài thêm thư viện để chạy nếu cần. Thêm vào đó thì bạn cũng có thể liên kết Colab với google drive và đọc, lưu dữ liệu lên google drive nên rất tiện để sử dụng. Mặc dù Ở trên Colab chỉ hỗ trợ 2 version Python là 2.7 và 3.6 và chưa hỗ trợ ngôn ngữ R và Scala, thì Google Colab vẫn là môi trường tuyệt vời để học và thực hành với deep learning.

### 5.1. Tiến lập trình lập

Import các thư viện:

```
✓ 2s #1. Thêm các thư viện cần thiết
import matplotlib.pyplot as plt
from keras.utils import np_utils
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import SGD, RMSprop
from keras.layers import Dense, Flatten, Dropout
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
```

Xử lý dữ liệu

```
✓ 0s # xử lý dữ liệu
train = ImageDataGenerator(rescale = 1./255,
                           width_shift_range=0.2,
                           rotation_range=40,
                           shear_range=0.2,
                           height_shift_range=0.2,
                           zoom_range=0.2,
                           fill_mode='nearest',
                           horizontal_flip=True)
validation = ImageDataGenerator(rescale = 1./255)
```

Load dữ liệu



```

✓ [5] # load dữ liệu
3s train_data = train.flow_from_directory('/content/drive/MyDrive/DataFood/Training',
                                         target_size = (150,150),
                                         class_mode = 'categorical')

validation_dataset = train.flow_from_directory('/content/drive/MyDrive/DataFood/Validation',
                                                target_size = (150,150),
                                                class_mode = 'categorical')

```

Found 1050 images belonging to 15 classes.  
Found 300 images belonging to 15 classes.

## Tạo model


```

✓ #Định nghĩa model
4s model = Sequential()
# Thêm Convolutional layer với 32 kernel, kích thước kernel 3*3
# dùng hàm sigmoid làm activation và chỉ rõ input_shape cho layer đầu tiên
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding = 'same',input_shape=(150,150,3)))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding = 'same'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
model.add(Dense(64,activation='relu',kernel_initializer='he_uniform'))
model.add(Dropout(0.2))
model.add(Dense(15,activation='softmax'))
model.summary()

```




Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 32)	896
max_pooling2d (MaxPooling2D)	(None, 75, 75, 32)	0
conv2d_1 (Conv2D)	(None, 75, 75, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 37, 37, 32)	0
flatten (Flatten)	(None, 43808)	0
dense (Dense)	(None, 64)	2803776
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 15)	975

Total params: 2,814,895  
Trainable params: 2,814,895  
Non-trainable params: 0

## Compile model



```

# Compile model, chỉ rõ hàm loss_function nào được sử dụng, phương thức
# dùng để tối ưu hàm loss function
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
# Thực hiện train model với data
history = model.fit(train_data,epochs=20,batch_size=128,validation_data=validation_dataset,verbose=1)

```

```

/20
=====] - 369s 11s/step - loss: 3.7801 - accuracy: 0.0810 - val_loss: 2.6713 - val_accuracy: 0.0667
/20
=====] - 9s 279ms/step - loss: 2.5238 - accuracy: 0.1400 - val_loss: 2.2537 - val_accuracy: 0.1400
/20
=====] - 9s 276ms/step - loss: 2.0240 - accuracy: 0.2610 - val_loss: 1.7206 - val_accuracy: 0.4000
/20
=====] - 9s 278ms/step - loss: 1.8025 - accuracy: 0.3733 - val_loss: 1.4490 - val_accuracy: 0.5167
/20
=====] - 9s 277ms/step - loss: 1.4903 - accuracy: 0.5076 - val_loss: 1.2237 - val_accuracy: 0.6833
/20
=====] - 9s 277ms/step - loss: 1.3264 - accuracy: 0.5638 - val_loss: 1.0202 - val_accuracy: 0.6933
/20
=====] - 9s 278ms/step - loss: 1.2062 - accuracy: 0.5943 - val_loss: 0.9073 - val_accuracy: 0.7033
/20
=====] - 9s 277ms/step - loss: 1.1043 - accuracy: 0.6190 - val_loss: 0.7635 - val_accuracy: 0.7633

```

Realtime:

```

}

function onAnimationFrame() {
  if (!shutdown) {
    window.requestAnimationFrame(onAnimationFrame);
  }
  if (pendingResolve) {
    var result = "";
    if (!shutdown) {
      captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);
      result = captureCanvas.toDataURL('image/jpeg', 0.8)
    }
    var lp = pendingResolve;
    pendingResolve = null;
    lp(result);
  }
}

async function createDom() {
  if (div !== null) {
    return stream;
  }
}

```

```

from IPython.display import display, Javascript, Image
# JavaScript to properly create our live video stream using our webcam as input

def video_stream():
  js = Javascript("""
    var video;
    var div = null;
    var stream;
    var captureCanvas;
    var imgElement;
    var labelElement;

    var pendingResolve = null;
    var shutdown = false;

    function removeDom() {
      stream.getVideoTracks()[0].stop();
      video.remove();
      div.remove();
      video = null;
      div = null;
      stream = null;
      imgElement = null;
      captureCanvas = null;
      labelElement = null;
    }
  """)

```

```

div = document.createElement('div');
div.style.border = '2px solid black';
div.style.padding = '3px';
div.style.width = '100%';
div.style.maxWidth = '600px';
document.body.appendChild(div);

const modelOut = document.createElement('div');
modelOut.innerHTML = "<span>Status:</span>";
labelElement = document.createElement('span');
labelElement.innerText = 'No data';
labelElement.style.fontWeight = 'bold';
modelOut.appendChild(labelElement);
div.appendChild(modelOut);

video = document.createElement('video');
video.style.display = 'block';
video.width = div.clientWidth - 6;
video.setAttribute('playsinline', '');
video.onclick = () => { shutdown = true; };
stream = await navigator.mediaDevices.getUserMedia(
  {video: { facingMode: "environment"}});
div.appendChild(video);

```

```

const instruction = document.createElement('div');
instruction.innerHTML =
  '<span style="color: red; font-weight: bold;">' +
  'Bấm vào video để dừng</span>';
div.appendChild(instruction);
instruction.onclick = () => { shutdown = true; };

video.srcObject = stream;
await video.play();

captureCanvas = document.createElement('canvas');
captureCanvas.width = 640; //video.videoWidth;
captureCanvas.height = 480; //video.videoHeight;
window.requestAnimationFrame(onAnimationFrame);

return stream;
}

```

```

    },
    async function stream_frame(label, imgData) {
        if (shutdown) {
            removeDom();
            shutdown = false;
            return '';
        }

        var preCreate = Date.now();
        stream = await createDom();

        var preShow = Date.now();
        if (label != "") {
            labelElement.innerHTML = label;
        }

        if (imgData != "") {
            var videoRect = video.getClientRects()[0];
            imgElement.style.top = videoRect.top + "px";
            imgElement.style.left = videoRect.left + "px";
            imgElement.style.width = videoRect.width + "px";
            imgElement.style.height = videoRect.height + "px";
            imgElement.src = imgData;
        }
    }

```

```

        var preCapture = Date.now();
        var result = await new Promise(function(resolve, reject) {
            pendingResolve = resolve;
        });
        shutdown = false;

        return {
            'create': preShow - preCreate,
            'show': preCapture - preShow,
            'capture': Date.now() - preCapture,
            'img': result
        };
    }
    ...
}

```

display(js)

```

def video_frame(label, bbox):
    data = eval_js('stream_frame("{}","{}").format(label, bbox)')
    return data

```

```

%cd /content
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from base64 import b64decode, b64encode
import numpy as np
import PIL
import io
import cv2
from keras.models import load_model

# start streaming video from webcam
video_stream()
# label for video
label_html = 'Đang lấy hình ảnh...'
# initialize bounding box to empty
bbox = ''
count = 0

# Load model Nhận diện tiền
model_file_path = "/content/tam.h5"
vggmodel = load_model(model_file_path)

```

```

# Vẽ lên một ảnh để tạo nửa overlay

# create transparent overlay for bounding box
bbox_array = np.zeros([480,640,4], dtype=np.uint8)

bbox_array = cv2.putText(bbox_array, "{}".format(class_name),
                        (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                        (0, 255,0), 2)

bbox_array[:, :, 3] = (bbox_array.max(axis = 2) > 0 ).astype(int) * 255
# convert overlay of bbox into bytes
bbox_bytes = bbox_to_bytes(bbox_array)
# update bbox so next frame gets new overlay
bbox = bbox_bytes

```

```

classes = ['BanhKhoaiMo', 'BanhTieu', 'BanhTrangNuong', 'BanhUot', 'BanhXeo', 'BoNe', 'BunXao', 'CaVienChier

while True:
    # Đọc ảnh trả về từ JS
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break

    # convert JS response to OpenCV Image
    frame = js_to_image(js_reply["img"])

    # Resize để đưa vào model
    frame_p = cv2.resize(frame, dsize=(150,150))
    tensor = np.expand_dims(frame_p, axis=0)

    # Feed vào mạng
    pred = vggmodel.predict(tensor)
    class_id = np.argmax(pred)
    class_name = classes[class_id]

```

```

# function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
    """
    Params:
        js_reply: JavaScript object containing image from webcam
    Returns:
        img: OpenCV BGR image
    """
    # decode base64 image
    image_bytes = b64decode(js_reply.split(',')[1])
    # convert bytes to numpy array
    jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
    # decode numpy array into OpenCV BGR image
    img = cv2.imdecode(jpg_as_np, flags=1)

    return img

```

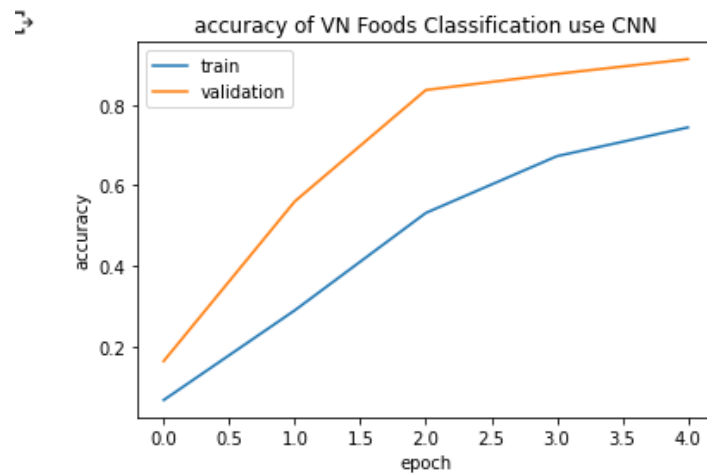
```

# function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
    """
    Params:
        bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
    Returns:
        bytes: Base64 image byte string
    """
    # convert array into PIL image
    bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
    iobuf = io.BytesIO()
    # format bbox into png for return
    bbox_PIL.save(iobuf, format='png')
    # format return string
    bbox_bytes = 'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue())), 'utf-8'))

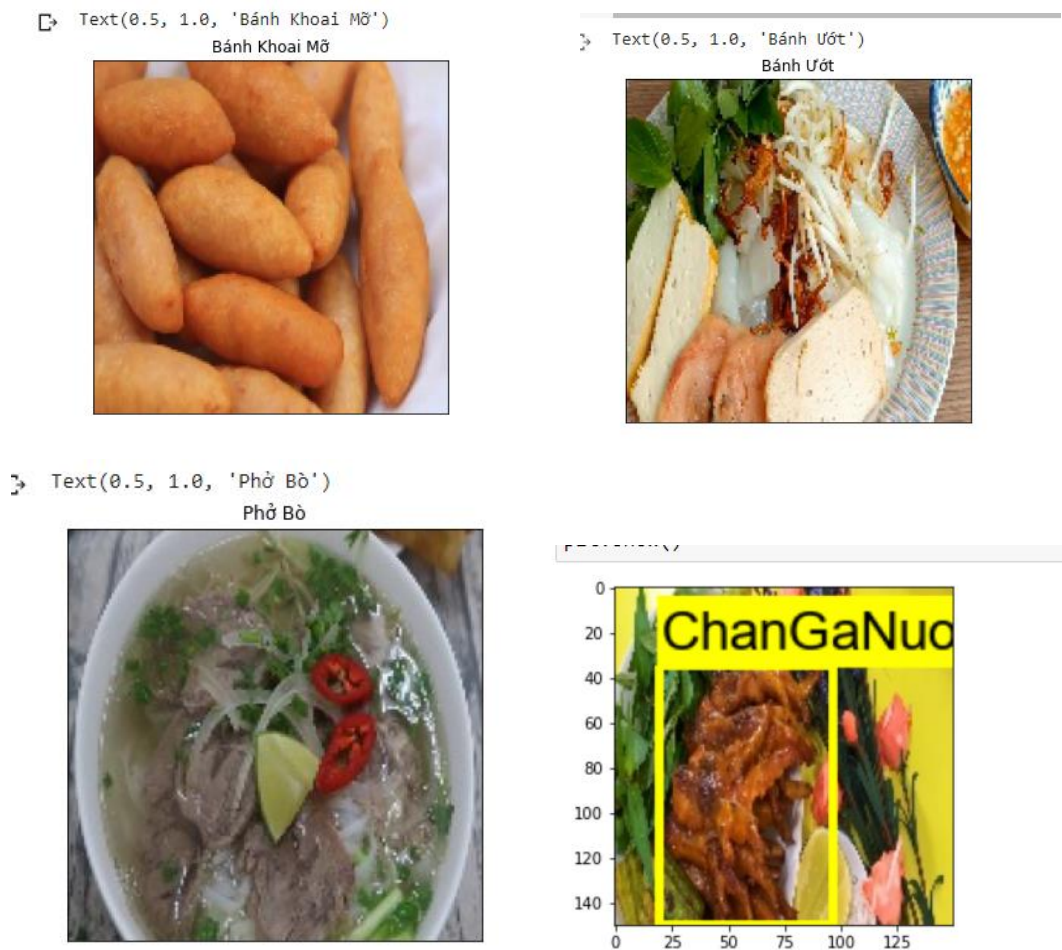
    return bbox_bytes

```

## 6. Kết quả

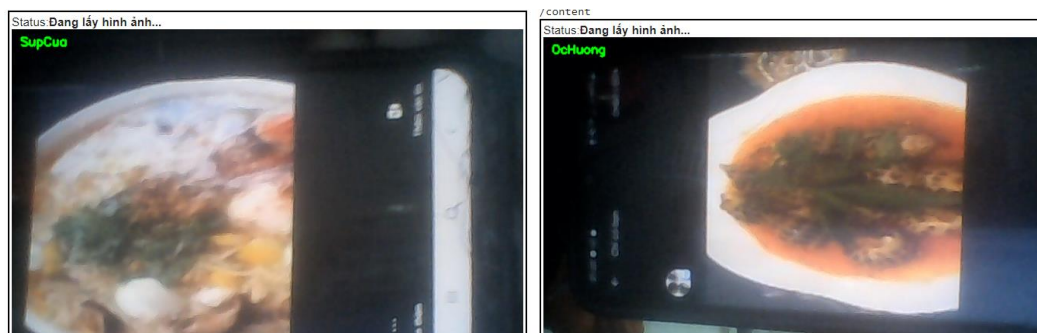


Hình 9: Độ chính xác của quá trình training



Hình 10: Nhận diện thức ăn bằng hình ảnh

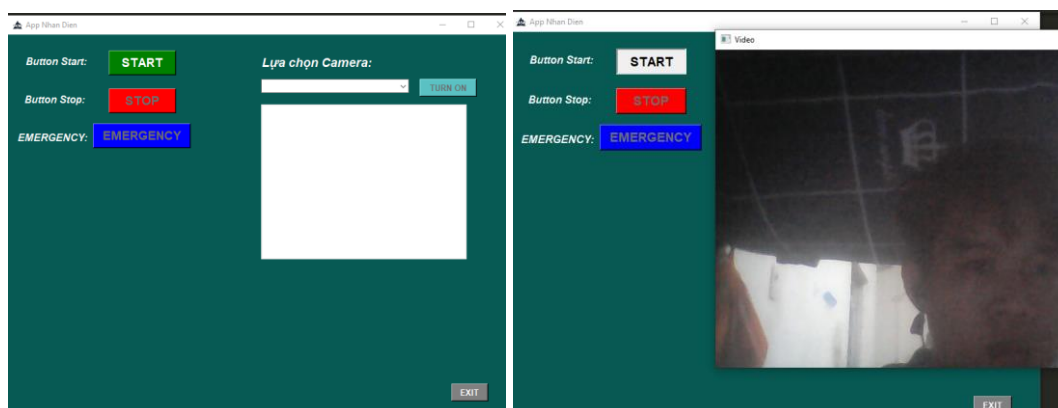




Hình 11: Nhận diện realtime bằng colab



Hình 12: Giao diện load dữ liệu



Hình 13: Giao diện app

## 7. Kết Luận

Đề tài này cho thấy rằng mạng nơ-ron tích chập có thể giải quyết thành công các vấn đề phân loại hình ảnh thức ăn với số lượng lớp tương đối nhỏ. Việc phân loại thành nhiều lớp hơn đòi hỏi nhiều kiến trúc phức tạp hơn. Bên cạnh độ phức tạp của một mô hình, việc chọn các chức năng tối ưu hóa và các tham số phù hợp cũng rất quan trọng. Với bộ dữ liệu đã cho cho 15 loại thức ăn: độ chính xác cuối cùng của mô hình đạt 90%.

## **TÀI LIỆU THAM KHẢO**

- [1] Lowe, D.G. Object Recognition from Local Scale Invariant Features. In Proceedings of the ICCV'99, Corfu, Greece, 20–21 September 1999.
- [2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
- [3] David Poole, “Artificial intelligence – Foundations of computational agents”. 2010 [Online]. Available: [http://artint.info/html/ArtInt\\_180.html](http://artint.info/html/ArtInt_180.html) [Accessed: Sept. 28, 2015].
- [4] BLVC, “The toolkit for the CNN – Caffe” 2014. [Online]. Available: <http://caffe.berkeleyvision.org/> [Accessed: Sept. 28, 2015]
- [5] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).