

**KHOA CƠ KHÍ CHẾ TẠO MÁY**

**BỘ MÔN CƠ ĐIỆN TỬ**

**🙡🙢✪🙠🙣**

**BÁO CÁO CUỐI KÌ**

**Insect Detection**

**GVHD: PGS.TS NGUYỄN TRƯỜNG THỊNH**

**SVTH: HUỲNH ANH KIỆT**

**MSSV: 20146499**

**LỚP: Chiều thứ 2**

**Tp.Hồ Chí Minh, Tháng 05 Năm 2023**

**NHẬN XÉT CỦA GIẢNG VIÊN**

**.....................................................................................................................................**

**.....................................................................................................................................**

**.....................................................................................................................................**

**.....................................................................................................................................**

**.....................................................................................................................................**

**.....................................................................................................................................**

**.....................................................................................................................................**

**.....................................................................................................................................**

**.....................................................................................................................................**

**.....................................................................................................................................**

**.....................................................................................................................................**

**Điểm: ……………………………..**

**KÝ TÊN**

**MỤC LỤC**

[ABSTRACT 1](#_Toc135555331)

[CHƯƠNG 1: GIỚI THIỆU 2](#_Toc135555332)

[CHƯƠNG 2: PHƯƠNG PHÁP LUẬN 3](#_Toc135555333)

[2.1. Thuật toán CNN (Convolutional Neural Network) 3](#_Toc135555334)

[2.1.1. CNN (Convolutional Neural Network) là gì 3](#_Toc135555335)

[2.1.2. Cấu trúc của mạng CNN (Convolutional Neural Network) 3](#_Toc135555336)

[2.2. Thư viện Keras 6](#_Toc135555337)

[2.3. Thư viện Tkinter 7](#_Toc135555338)

[2.4. Thư viện Tensorflow 7](#_Toc135555339)

[CHƯƠNG 3: THỰC HIỆN MÔ HÌNH TRAINING 9](#_Toc135555340)

[3.1. Khái báo các thư viện đê xây dựng mô hình 9](#_Toc135555341)

[3.2. Lấy dữ liệu và mô tả dữ liệu 9](#_Toc135555342)

[3.2.1. Lấy dữ liệu và mô tả dữ liệu train 9](#_Toc135555343)

[3.2.2. Lấy dữ liệu và mô tả dữ liệu test 11](#_Toc135555344)

[3.3. Xây dựng mô hình CNN 14](#_Toc135555345)

[3.4. Train mô hình CNN 17](#_Toc135555346)

[3.5. Độ chính xác của mô hình và đồ thị theo số lần học 18](#_Toc135555347)

[3.6. Lưu dữ liệu train 18](#_Toc135555348)

[CHƯƠNG 4: KẾT QUẢ 19](#_Toc135555349)

[CHƯƠNG 5: KẾT LUẬN 20](#_Toc135555350)

[TÀI LIỆU THAM KHẢO 21](#_Toc135555351)

[PHỤ LỤC 22](#_Toc135555352)

**ABSTRACT**

Trên toàn cầu, có rất nhiều loại côn trùng khác nhau và chúng có thể tác động tốt hoặc xấu cho tự nhiên và cuộc sống của chúng ta. Với những cải tiến trong công nghệ trí tuệ nhân tạo, một số ứng dụng đã xuất hiện trong bối cảnh nông nghiệp, bao gồm tự động phát hiện, giám sát và nhận dạng côn trùng. Mục đích là để cung cấp cho người đọc một cái nhìn tổng quan về những phát triển chính trong lĩnh vực này. Độ chính xác của phân loại bị ảnh hưởng nhiều nhất bởi kích thước tập dữ liệu, trong khi khả năng phát hiện bị ảnh hưởng đáng kể bởi số lượng lớp và kích thước tập dữ liệu.

**CHƯƠNG 1: GIỚI THIỆU**

Hiện nay AI(Artificial Intelligence) được áp dụng rộng rãi vào trong đời sống của con người. Một trong những ứng của AI đó là nhận diện các loại côn trùng. Mặc dù hiện nay đã có rất nhiều ứng dụng và thiết bị nhận diện nhận diện các loại côn trùng xuất hiện trên thị trường, chúng vẫn còn bị hạn chế trong phạm vi sử dụng. Trong bài này sẽ ta cố gắng nghiên cứu xây dựng ra một hệ thống nhận diện côn trùng có thể linh hoạt đáp ứng được với mọi nhu cầu của người sử dụng và áp dụng vào được trong nhiều nghành nghề lĩnh vực, nền tảng sử dụng khác nhau.

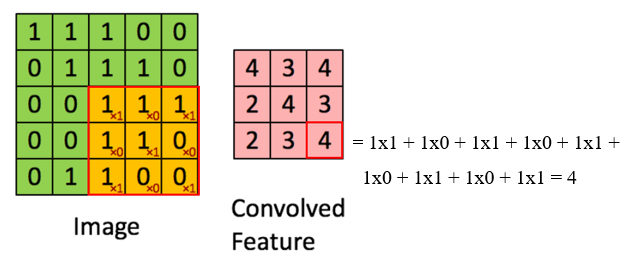
Mô hình nhận diện côn trùng dựa vào các đặc điểm nổi bật của từng loài và màu sắc của loài. Để nhận diện côn trùng bằng cách dùng hình ảnh dựa thuật toán CNN cho việc huấn luyện mô hình. Sau khi huấn luyện ta có thể nhận diện các loài côn trùng thường gặp trong cuộc sống.

**CHƯƠNG 2: PHƯƠNG PHÁP LUẬN**

* 1. **Thuật toán CNN (Convolutional Neural Network)**
     1. **CNN (Convolutional Neural Network) là gì**

CNN là Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống những mô hình để nhận dạng và phân loại hình ảnh thông minh với độ chính xác cao như hiện nay.

Convolution Layer (Lớp tích chập) 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.

Convolutional là một cửa sổ trượt (Sliding Windows) trên một ma trận như mô tả hình dưới:

**Hình 1**. Ma trận Convolutional

* + 1. **Cấu trúc của mạng CNN (Convolutional Neural Network)**

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

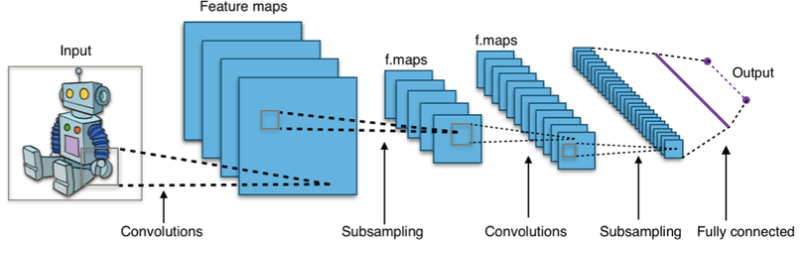
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.

******Hình 2:** Cấu trúc mạng CNN

Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter.

Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

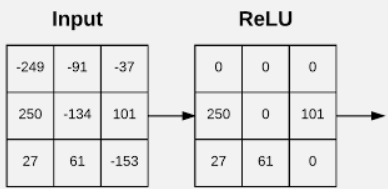
+ Đường viên – Padding

Khi Kernel không phù hợp với hình ảnh đầu vào thì ta thể: chèn thêm các số 0 vào 4 đường biên của hình ảnh (padding) hoặc cắt bớt hình ảnh tại những điểm không phù hợp với Kernel.

+ Hàm phi tuyến – ReLU

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

Hàm phi tuyến ReLU có tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm.

**Hình 3:** ReLU Layer

+ 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: lấy phần tử lớn nhất từ ma trận đối tượng

+Average Pooling: lấy tổng trung bình phần tử trong ma trận đối tượng

+Sum Pooling: lấy tổng tất cả các phần tử trong map

* 1. **Thư viện Keras**

Keras chạy trên các thư viện máy mã nguồn mở như TensorFlow, Theano và bộ công cụng nhận thức (CNTK) Theano là một thư viện python được sử dụng cho các tác vụ tính toán số nhanh. TensorFlow là thư viện toán học biểu tượng nổi tiếng nhất được sử dụng để tạo mạng nơ-ron và mô hình học sâu. TensorFlow rất linh hoạt và lợi ích chính là tính toán phân tán. CNTK là khung học sâu được phát triển bởi Microsoft. Nó sử dụng các thư viện như Python, C #, C ++ hoặc các bộ công cụ học máy độc lập. Chúng đều là những thư viện rất mạnh nhưng cũng khó hiểu để tạo mạng nơ-ron.

Keras là một API cấp cao được thiết kế cho Python để triển khai mạng nơ-ron dễ dàng hơn. Nó được phát triển bởi Google.

Keras tận dụng các kỹ thuật tối ưu hóa khác nhau để làm cho API mạng thần kinh cấp cao dễ dàng hơn và hiệu quả hơn. Nó hỗ trợ các tính năng sau:

* API nhất quán, đơn giản và có thể mở rộng.
* Cấu trúc tối thiểu - dễ dàng đạt được kết quả mà không cần rườm rà.
* Hỗ trợ nhiều nền tảng và backend.
* Thân thiện với người dùng chạy trên cả CPU và GPU.
* Khả năng mở rộng tính toán cao.
* Keras năng động, mạnh mẽ và có những ưu điểm sau
* Cộng động lớn hỗ trợ
* Dễ dàng để kiểm tra.
* Mạng nơ-ron Keras được viết bằng Python giúp mọi thứ đơn giản hơn.

Keras hỗ trợ cả mạng convolution và recurrent. Keras là một thư viện tương đối dễ sử dụng đối với người mới bắt đầu. Nó cung cấp các hàm số cần thiết với cú pháp đơn giản.

* 1. **Thư viện Tkinter**

Tkinter là thư viện GUI tiêu chuẩn cho Python. Tkinter trong Python cung cấp một cách nhanh chóng và dễ dàng để tạo các ứng dụng GUI. Tkinter cung cấp giao diện hướng đối tượng cho bộ công cụ Tk GUI.

Việc sử dụng Tkinter rất là đơn giản vì Tkinter sử dụng các cú pháp rất đơn giản và dễ hiểu.

Tkinter cũng cung cấp cho chúng ta rất nhiều bảng điều khiển khác nhau để sử dụng trong việc tạo các giao diện như là nút, nhãn, hôp kiểm,… Các bange điểu khiển này được gọi là các widget. Hiện tại thì Tkinter có tới 15 kiểu widget cho chúng ta.

Nên Tkinter rất phù hợp cho cả người mới học và người có kinh nghiệm sử dụng trong lập trình giao diện.

* 1. **Thư viện Tensorflow**

Tensorflow 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.

Nhờ có Tensorflow, người dùng có thể đơn giản hóa toán học thông qua các đồ thị luồng dữ liệu tổng hợp. Ngoài ra, không thể không kể đến các ứng dụng của Tensorflow như:

* Tích hợp sẵn rất nhiều các thư viện machine learning
* Có khả năng tương thích và mở rộng tốt. Được Google phát triển cho machine learning phục vụ cả nghiên cứu lẫn xây dựng các ứng dụng thực tế

Sử dụng TensorFlow sẽ mang lại nhiều lợi ích quan trọng cho việc lập trình machine learning chính là abstraction.

**CHƯƠNG 3: THỰC HIỆN MÔ HÌNH TRAINING**

## **3.1. Khái báo các thư viện đê xây dựng mô hình**

|  |
| --- |
| from os import listdir  from numpy import asarray, save  from keras.utils import load\_img, img\_to\_array  from keras.utils import to\_categorical  from keras.layers import  Dense,Dropout,Flatten,Conv2D,MaxPooling2D,Normalization  from keras.models import Sequential,Model  from keras import Input  from keras.layers import LeakyReLU  from keras.optimizers import Adam  import matplotlib.pyplot as plt  import numpy as np |

## **3.2. Lấy dữ liệu và mô tả dữ liệu**

**3.2.1. Lấy dữ liệu và mô tả dữ liệu train**

|  |
| --- |
| folder = '/content/drive/MyDrive/Final\_Project/Train/'  photos, labels = list(), list()  for file in listdir(folder):  output = 0.0  if file.startswith('boCanhCungKhongLo '):  output = 1.0  if file.startswith('boCanhCungTiger '):  output = 2.0  if file.startswith('boCap '):  output = 3.0  if file.startswith('boHung '):  output = 4.0  if file.startswith('boNguaPhongLan '):  output = 5.0  if file.startswith('boNgua '):  output = 6.0  if file.startswith('boRua '):  output = 7.0  if file.startswith('buomDem '):  output = 8.0  if file.startswith('buom '):  output = 9.0  if file.startswith('caoCao '):  output = 10.0  if file.startswith('chauChau '):  output = 11.0  if file.startswith('chuonChuon '):  output = 12.0  if file.startswith('de '):  output = 13.0  if file.startswith('deTrui '):  output = 14.0  if file.startswith('domDom '):  output = 15.0  if file.startswith('gian '):  output = 16.0  if file.startswith('kienBaKhoang '):  output = 17.0  if file.startswith('kienDen '):  output = 18.0  if file.startswith('kienVang '):  output = 19.0  if file.startswith('motCaoCo '):  output = 20.0  if file.startswith('muoi '):  output = 21.0  if file.startswith('ong '):  output = 22.0  if file.startswith('ongBapCay '):  output = 23.0  if file.startswith('ret '):  output = 24.0  if file.startswith('ruoi '):  output = 25.0  if file.startswith('sau '):  output = 26.0  if file.startswith('veSau '):  output = 27.0  if file.startswith('nhen '):  output = 28.0  photo = load\_img(folder + file, target\_size =(128,128))  photo = img\_to\_array(photo)  photos.append(photo)  labels.append(output)  # Train  photos\_train = asarray(photos)  labels\_train = asarray(labels)  print(photos\_train.shape,labels\_train.shape) |
| (5117, 128, 128, 3) (5117,) |

|  |
| --- |
| # Reshape the sets and rescale within the byte range  photos\_train = photos\_train.reshape(5117,128,128,3)  photos\_train = photos\_train.astype('float32')/255  # One-hot encode the target classes  labels\_train = to\_categorical(labels\_train)  (photos\_train.shape, labels\_train.shape) |
| (5117, 128, 128, 3), (5117, 29)) |

**3.2.2. Lấy dữ liệu và mô tả dữ liệu test**

|  |
| --- |
| folder = '/content/drive/MyDrive/Final\_Project/Test/'  photos, labels = list(), list()  for file in listdir(folder):  output = 0.0  if file.startswith('boCanhCungKhongLo '):  output = 1.0  if file.startswith('boCanhCungTiger '):  output = 2.0  if file.startswith('boCap '):  output = 3.0  if file.startswith('boHung '):  output = 4.0  if file.startswith('boNguaPhongLan '):  output = 5.0  if file.startswith('boNgua '):  output = 6.0  if file.startswith('boRua '):  output = 7.0  if file.startswith('buomDem '):  output = 8.0  if file.startswith('buom '):  output = 9.0  if file.startswith('caoCao '):  output = 10.0  if file.startswith('chauChau '):  output = 11.0  if file.startswith('chuonChuon '):  output = 12.0  if file.startswith('de '):  output = 13.0  if file.startswith('deTrui '):  output = 14.0  if file.startswith('domDom '):  output = 15.0  if file.startswith('gian '):  output = 16.0  if file.startswith('kienBaKhoang '):  output = 17.0  if file.startswith('kienDen '):  output = 18.0  if file.startswith('kienVang '):  output = 19.0  if file.startswith('motCaoCo '):  output = 20.0  if file.startswith('muoi '):  output = 21.0  if file.startswith('ong '):  output = 22.0  if file.startswith('ongBapCay '):  output = 23.0  if file.startswith('ret '):  output = 24.0  if file.startswith('ruoi '):  output = 25.0  if file.startswith('sau '):  output = 26.0  if file.startswith('veSau '):  output = 27.0  if file.startswith('nhen '):  output = 28.0  photo = load\_img(folder + file, target\_size =(128,128))  photo = img\_to\_array(photo)  photos.append(photo)  labels.append(output)  # Test  photos\_test = asarray(photos)  labels\_test = asarray(labels)  print(photos\_test.shape,labels\_test.shape) |
| (1928, 128, 128, 3) (1928,) |

|  |
| --- |
| # Reshape the sets and rescale within the byte range  photos\_test = photos\_test.reshape(1928, 128, 128, 3)  photos\_test = photos\_test.astype('float32')/255  # One-hot encode the target classes  labels\_test = to\_categorical(labels\_test)  (photos\_test.shape, labels\_test.shape) |
| ((1928, 128, 128, 3), (1928, 29)) |

|  |
| --- |
| class\_Insect = {1: 'boCanhCungKhongLo',2: 'boCanhCungTiger',3: 'boCap',  4: 'boHung',5: 'boNguaPhongLan',  6: 'boNgua',7: 'boRua',8: 'buomDem',9: 'buom',10: 'caoCao',  11: 'chauChau',12: 'chuonChuon',13: 'de',14: 'deTrui',  15: 'domDom', 16: 'gian',17: 'kienBaKhoang',18: 'kienDen',  19: 'kienVang',20: 'motCaoCo', 21: 'muoi',22: 'ong',  23: 'ongBapCay',24: 'ret', 25: 'ruoi',26: 'sau',  27: 'veSau',28: 'nhen'}  # Tên của các loại côn trùng có trong Folder |

## **3.3. Xây dựng mô hình CNN**

|  |
| --- |
| model = Sequential()  # CNN  model.add(Conv2D(32,kernel\_size=(3,3), kernel\_initializer='he\_uniform',activation='relu',input\_shape=(128,128,3),padding='same'))  model.add(LeakyReLU(alpha=0.1))  model.add(MaxPooling2D((2,2),padding = 'same'))  #Lớp đầu tiên là lớp Convolution2D với dữ liệu đầu vào có kích thước là (128x128x3), sử dụng 32 filter. Lớp này có kích thước cửa sổ trượt (Kernel\_size) là (3,3) , ma trận trong số hạt nhân ‘he\_uniform’ và bước trượt là (2,2)  # Lớp Maxpooling2D, kích thước (pool\_size) là (2,2), làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng.  #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  model.add(Conv2D(64,(3,3), kernel\_initializer='he\_uniform',activation = 'relu',padding = 'same'))  model.add(LeakyReLU(alpha = 0.1))  model.add(MaxPooling2D((2,2), padding='same'))  #Lớp đầu tiên là lớp Convolution2D với dữ liệu đầu vào có kích thước là (128x128x3), sử dụng 64 filter. Lớp này có kích thước cửa sổ trượt (Kernel\_size) là (3,3) , ma trận trong số hạt nhân ‘he\_uniform’ và bước trượt là (2,2)  # Lớp Maxpooling2D, kích thước (pool\_size) là (2,2), làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng.  #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  model.add(Conv2D(128,(3,3), kernel\_initializer='he\_uniform',activation = 'relu', padding = 'same'))  model.add(LeakyReLU(alpha = 0.1))  model.add(MaxPooling2D((2,2),padding= 'same'))  #Lớp đầu tiên là lớp Convolution2D với dữ liệu đầu vào có kích thước là (128x128x3), sử dụng 128 filter. Lớp này có kích thước cửa sổ trượt (Kernel\_size) là (3,3) , ma trận trong số hạt nhân ‘he\_uniform’ và bước trượt là (2,2)  # Lớp Maxpooling2D, kích thước (pool\_size) là (2,2), làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng.  #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  model.add(Conv2D(256,(3,3), kernel\_initializer='he\_uniform',activation = 'relu', padding = 'same'))  model.add(LeakyReLU(alpha = 0.1))  model.add(MaxPooling2D((2,2),padding= 'same'))  #Lớp đầu tiên là lớp Convolution2D với dữ liệu đầu vào có kích thước là (128x128x3), sử dụng 256 filter. Lớp này có kích thước cửa sổ trượt (Kernel\_size) là (3,3) , ma trận trong số hạt nhân ‘he\_uniform’ và bước trượt là (2,2)  # Lớp Maxpooling2D, kích thước (pool\_size) là (2,2), làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng.  #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  model.add(Conv2D(512,(3,3), kernel\_initializer='he\_uniform',activation = 'relu', padding = 'same'))  model.add(LeakyReLU(alpha = 0.1))  model.add(MaxPooling2D((2,2),padding= 'same'))  #Lớp đầu tiên là lớp Convolution2D với dữ liệu đầu vào có kích thước là (128x128x3), sử dụng 512 filter. Lớp này có kích thước cửa sổ trượt (Kernel\_size) là (3,3) , ma trận trong số hạt nhân ‘he\_uniform’ và bước trượt là (2,2)  # Lớp Maxpooling2D, kích thước (pool\_size) là (2,2), làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng.  model.add(Flatten())  #Lớp Flatten là lớp sẽ có chức năng reshape cụ thể trong đó tất cả các axes được làm phẳng ( smooshed) hoặc ghép lại (squashed ) với nhau, hay có thể hiểu là dữ liệu khi qua lớp này sẽ được trải phẳng và có dạng một chiều  model.add(Dense(512,activation = 'relu'))  model.add(LeakyReLU(alpha = 0.1))  model.add(Dense(256,activation = 'relu'))  model.add(LeakyReLU(alpha = 0.1))  model.add(Dense(128,activation = 'relu'))  model.add(LeakyReLU(alpha = 0.1))  model.add(Dense(64,activation = 'relu'))  model.add(LeakyReLU(alpha = 0.1))  model.add(Dense(classes ,activation = 'softmax'))  model.summary() |
| Model: "sequential"  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =================================================================  conv2d\_5 (Conv2D) (None, 128, 128, 32) 896  leaky\_re\_lu\_9 (LeakyReLU) (None, 128, 128, 32) 0  max\_pooling2d\_5 (MaxPooling (None, 64, 64, 32) 0  2D)  conv2d\_6 (Conv2D) (None, 64, 64, 64) 18496  leaky\_re\_lu\_10 (LeakyReLU) (None, 64, 64, 64) 0  max\_pooling2d\_6 (MaxPooling (None, 32, 32, 64) 0  2D)  conv2d\_7 (Conv2D) (None, 32, 32, 128) 73856  leaky\_re\_lu\_11 (LeakyReLU) (None, 32, 32, 128) 0  max\_pooling2d\_7 (MaxPooling (None, 16, 16, 128) 0  2D)  conv2d\_8 (Conv2D) (None, 16, 16, 256) 295168  leaky\_re\_lu\_12 (LeakyReLU) (None, 16, 16, 256) 0  max\_pooling2d\_8 (MaxPooling (None, 8, 8, 256) 0  2D)  conv2d\_9 (Conv2D) (None, 8, 8, 512) 1180160  leaky\_re\_lu\_13 (LeakyReLU) (None, 8, 8, 512) 0  max\_pooling2d\_9 (MaxPooling (None, 4, 4, 512) 0  2D)  flatten\_1 (Flatten) (None, 8192) 0  dense\_5 (Dense) (None, 512) 4194816  leaky\_re\_lu\_14 (LeakyReLU) (None, 512) 0  dense\_6 (Dense) (None, 256) 131328  leaky\_re\_lu\_15 (LeakyReLU) (None, 256) 0  dense\_7 (Dense) (None, 128) 32896  leaky\_re\_lu\_16 (LeakyReLU) (None, 128) 0  dense\_8 (Dense) (None, 64) 8256  leaky\_re\_lu\_17 (LeakyReLU) (None, 64) 0  dense\_9 (Dense) (None, 29) 1885  =================================================================  Total params: 5,937,757  Trainable params: 5,937,757  Non-trainable params: 0 |

## **3.4. Train mô hình CNN**

|  |
| --- |
| batch\_size = 32 # mỗi lần học là 32 dữ liệu  epochs = 50 # 50 lần học  classes = 29  # Compile: Biên dịch  model.compile(optimizer = Adam(), loss = 'categorical\_crossentropy', metrics = ['accuracy'])  train = model.fit(photos\_train, labels\_train, batch\_size = batch\_size ,epochs = epochs, verbose = 1) |
| Epoch 1/50  160/160 [==============================] - 15s 32ms/step - loss: 3.3039 - accuracy: 0.0606  Epoch 2/50  160/160 [==============================] - 4s 26ms/step - loss: 3.0778 - accuracy: 0.1130  Epoch 3/50  160/160 [==============================] - 4s 26ms/step - loss: 2.8082 - accuracy: 0.1915  --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  Epoch 48/50  160/160 [==============================] - 4s 26ms/step - loss: 0.0236 - accuracy: 0.9928  Epoch 49/50  160/160 [==============================] - 4s 26ms/step - loss: 0.0415 - accuracy: 0.9896  Epoch 50/50  160/160 [==============================] - 4s 27ms/step - loss: 0.0562 - accuracy: 0.9850  61/61 [==============================] - 1s 14ms/step - loss: 3.6145 - accuracy: 0.5322 |

## **3.5. Độ chính xác của mô hình và đồ thị theo số lần học**

|  |
| --- |
| test\_loss, test\_acc = model.evaluate(photos\_test, labels\_test)  print('Do chinh xac cua ham', test\_acc)  print('Ham mat mat', test\_loss) |
| Do chinh xac cua ham 0.5321576595306396  Ham mat mat 3.614483118057251 |

|  |
| --- |
| # Đồ thị độ chính xác và hàm mất mác  accuracy = train.history['accuracy']  loss = train.history['loss']  epochs = range(len(accuracy))  plt.plot(epochs, accuracy, 'r-', label = 'Training accuracy')  plt.plot(epochs, loss, 'b', label = 'Training loss')  plt.legend(['Train accuracy', 'Train loss'])  plt.show() |
|  |

## **3.6. Lưu dữ liệu train**

|  |
| --- |
| model.save('/content/drive/MyDrive/Final\_Project/Final\_Project/model\_insect.h5') |

# **CHƯƠNG 4: KẾT QUẢ**

|  |
| --- |
| Predicted: nhen |

|  |
| --- |
| Predicted: sau |

**CHƯƠNG 5: KẾT LUẬN**

Mô hình CNN nhận diện côn trùng có độ chính xác không được cao vì vẫn có những sai sót khi nhận diện các loại côn trùng(kích thức của các côn trùng tương đối nhỏ, các loại côn trùng có màu sắc khá trùng với môi trường chúng sinh sống,…)

Hướng phát triển của mô hình:

Cần thêm dữ liệu về nhiều loại côn trùng khác nhau trên thế giới như là tên của loài côn trùng, có nguy hại hay không, các biện pháp phòng chống loài côn trùng đó,.. Để nhận diện được nhiều loài con trùng giúp chúng ta có thêm hiểu biết về các loại côn trùng,

Mô hình CNN này kết hợp nhận côn trùng theo thời gian thực để chúng ta nhận biết nhanh chóng loại côn trùng.

# **TÀI LIỆU THAM KHẢO**

1. Thuật toán CNN là gì? Cấu trúc mạng Convolutional Neural Network, Link truy cập: <https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/>
2. Thuật toán CNN là gì? Cấu trúc mạng Convolutional Neural Network, Link truy cập: <https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2>
3. Tổng quan về Keras cho người mới bắt đầu, Link truy cập: <https://irender.vn/tong-quan-ve-keras-cho-nguoi-moi-bat-dau/>
4. A Systematic Review on Automatic Insect Detection Using Deep Learning, Link truy cập: <https://www.mdpi.com/2077-0472/13/3/713>
5. Tensorflow là gì? Kiến thức cơ bản cần biết về TensorFlow, Link truy cập: <https://itnavi.com.vn/blog/tensorflow-la-gi>
6. PYTHON TKINTER - LẬP TRÌNH GUI BẰNG TKINTER TRONG PYTHON, Link truy cập: <https://t3h.edu.vn/tin-tuc/python-tkinter-lap-trinh-gui-bang-tkinter-trong-python#:~:text=Tkinter%20l%C3%A0%20th%C6%B0%20vi%E1%BB%87n%20GUI,b%E1%BB%99%20c%C3%B4ng%20c%E1%BB%A5%20Tk%20GUI>.

# **PHỤ LỤC**

1. Link GitHub: <https://github.com/KietHuynh02/Final_Project>
2. Link Data: <https://drive.google.com/drive/folders/1rnrlt_06PPxQEeE_PSO6_CwKnsLMRCYG?usp=sharing>
3. Link Youtube: <https://youtu.be/Fa9EAwJjgmI>