

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**



**Báo cáo cuối kỳ**

**Đề tài: Car and bike classification using CNN**

**Môn học** : Python cho Khoa học Dữ liệu  
**Giảng viên giảng dạy** : ThS. Hà Văn Thảo  
**Lớp** : 20KDL  
**Nhóm thực hiện** : Nhóm 25

**Thành viên:**

1. Nguyễn Quốc Tiến	20280098
2. Dương Vi Doanh	20280018
3. Nguyễn Nhật Hào	20280029
4. Phù Chí Đạt	20280015
5. Trần Nguyễn Nhất Duy	20280024

# Mục lục

<b>I. Mô hình CNN</b>	<b>3</b>
1. Giới thiệu	3
2. Lớp tích chập (Convolution Layer)	4
3. Hàm kích hoạt (Activation Function)	5
4. Lớp gộp (Pooling Layer)	6
5. Lớp được kết nối đầy đủ (Fully Connected Layer)	6
6. Hàm mất mát (Loss Function)	7
<b>II. Tổng quan, xử lý dữ liệu</b>	<b>8</b>
1. Tổng quan	8
2. Xử lý dữ liệu	8
3. Train mô hình	11
4. Demo	12
<b>III. Kết luận</b>	<b>13</b>
<b>IV. Nguồn tham khảo</b>	<b>14</b>

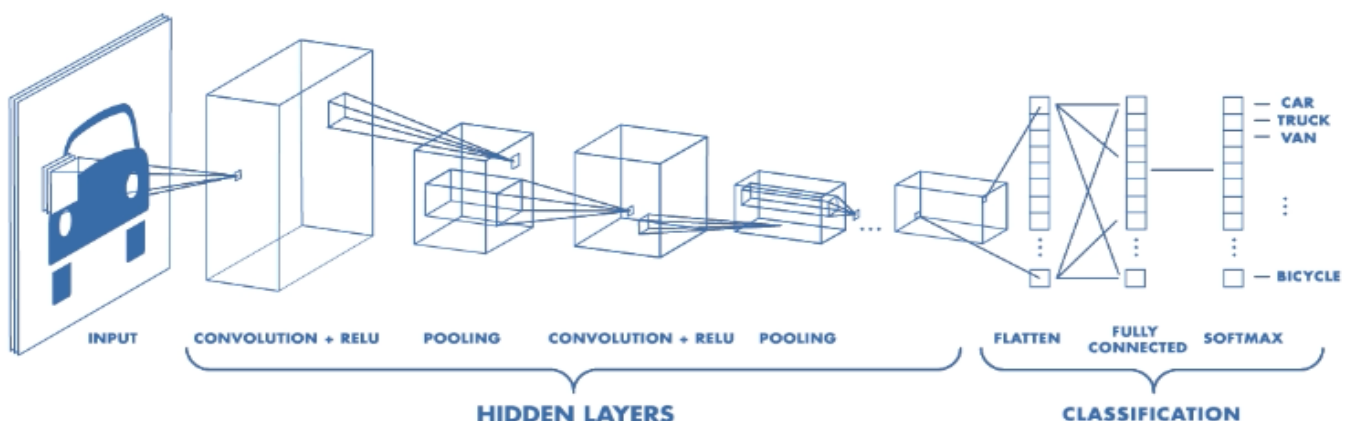
# I. Mô hình CNN

## 1. Giới thiệu

CNN (Convolutional Neural Network hay Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning hiện đại, tiên tiến, được ứng dụng rộng rãi trong việc phân loại, nhận dạng hình ảnh là các đối tượng hoặc khuôn mặt. CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các nhãn đã được cho trước (VD: xe máy, oto, xe đạp, chó, mèo...).

Những hình ảnh đầu vào sẽ được lần lượt chuyển qua nhiều lớp tích chập (Convolution Layer) bao gồm các bộ lọc (Kernels), mỗi lớp sử dụng các hàm kích hoạt như ReLU để tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo, sau đó được đưa tới các lớp được kết nối đầy đủ (Fully Connected layers) để chuyển hình ảnh ban đầu đã qua xử lý thành các phiếu bầu và cuối cùng áp dụng một hàm toán học để phân loại đối tượng có giá trị xác suất từ 0 đến 1.

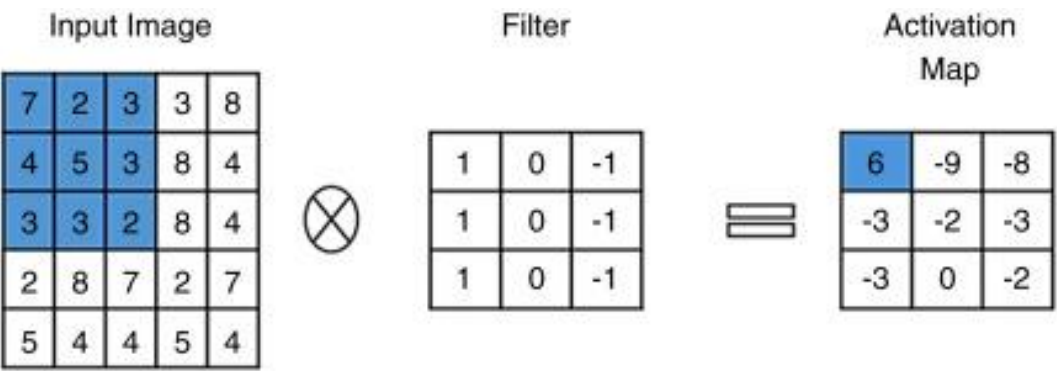
Hình ảnh minh họa một luồng CNN:










## 2. Lớp tích chập (Convolution Layer)

Là phần chính của CNN, bao gồm nhiều ma trận filters (hoặc kernels). Những ma trận này thường có kích thước nhỏ hơn ma trận ảnh. Tích của mỗi ma trận filter với ma trận ảnh sẽ tạo ra những activation map. Ở những Convolution Layer đầu sẽ trích ra những feature đơn giản của ảnh như đường thẳng, ở những lớp sau sẽ trích ra những feature phức tạp hơn như hình dáng, độ sắc nét, màu sắc, đối tượng cụ thể.

Hình ảnh minh họa:



Hình ảnh minh họa lớp tích chập rút trích những feature của ảnh.

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

### 3. Hàm kích hoạt (Activation Function)

Hàm kích hoạt là những hàm phi tuyến được áp dụng để chỉnh sửa lại các ma trận hình ảnh sau khi qua một lớp tích chập, giúp mô hình học được các quan hệ phi tuyến có trong dữ liệu. Nếu như không có hàm kích hoạt, mạng CNN sẽ không thể phát hiện ra những quan hệ phức tạp của dữ liệu dẫn đến làm giảm khả năng dự đoán của mô hình.

Một số hàm kích hoạt phổ biến:

Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

Hàm Sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (0;1). Đầu vào là số thực âm rất nhỏ sẽ cho đầu ra một số tiệm cận với 0, ngược lại, nếu đầu vào là một số thực dương rất lớn sẽ cho đầu ra là một số tiệm cận với 1.

Rectified Linear Unit (ReLU)

$$f(x) = \max(0, x)$$

Hàm ReLU chuyển các giá trị âm trong ma trận thành 0. Đây là hàm được sử dụng rất phổ biến trong mô hình CNN, giúp quá trình train mô hình được thực hiện nhanh hơn rất nhiều. Nhược điểm của hàm này là mô hình sẽ không học được gì từ các giá trị âm của ma trận.

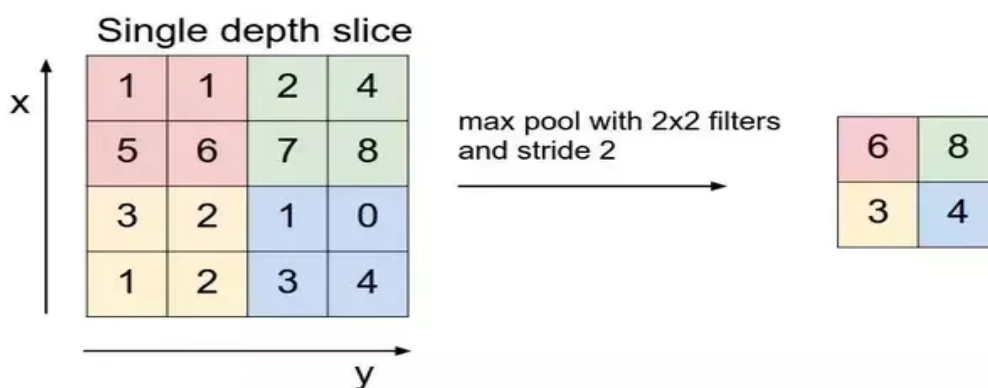
## 4. Lớp gộp (Pooling Layer)

Là phương pháp làm giảm kích thước (số chiều) của những hình ảnh (ma trận) lớn lại nhưng vẫn giữ được những thông tin quan trọng của ảnh. Có nhiều phương pháp pooling khác nhau như:

Max pooling: lấy phần tử lớn nhất.

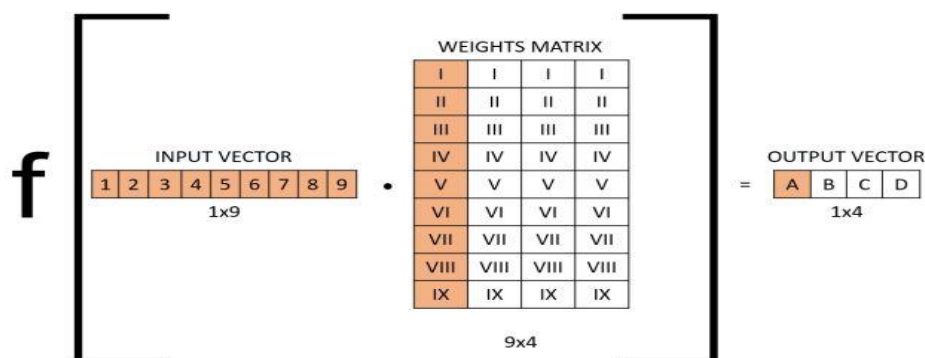
Average pooling: lấy trung bình tổng các phần tử.

Sum pooling: lấy tổng các phần tử.



## 5. Lớp được kết nối đầy đủ (Fully Connected Layer)

Thực hiện việc chuyển các hình ảnh đã qua xử lý thành các phiếu bầu bằng cách áp dụng biến đổi tuyến tính, đưa các ma trận hình ảnh thành các vector có giá trị tương ứng với các nhãn phân loại.



## 6. Hàm mất mát (Loss Function)

Loss function trả về một giá trị không âm thể hiện sự chênh lệch giữa hai đại lượng: số dự đoán đúng ( $y$ ) so với tổng số dự đoán ( $\hat{y}$ ).

Hàm loss nhóm sử dụng: Binary Cross Entropy Loss

$$L_{BCE} = -\frac{1}{n} \sum_{i=1}^n (Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

Hàm này được sử dụng rất phổ biến khi mô hình dùng để phân loại 2 class. Nó còn phản ánh sự chắc chắn trong việc dự đoán của mô hình. Nếu mô hình cho ra kết quả đúng nhưng độ chắc chắn không cao thì hàm mất mát sẽ có giá trị lớn.

## II. Tổng quan, xử lý dữ liệu

### 1. Tổng quan

Mục tiêu của bài báo cáo là train mô hình để phân loại hình ảnh đưa vào là xe oto hay xe máy. Nhóm sử dụng dataset gồm 2000 ảnh xe oto và 2000 ảnh xe máy khác loại. Do xe oto, xe máy rất đa dạng về dòng, đời sản phẩm khác nhau nên việc train mô hình cần tập trung làm sao để mô hình nhận dạng được sự khác biệt cơ bản của xe oto và xe máy, từ đó đưa ra kết quả chính xác.

### 2. Xử lý dữ liệu

#### Lọc dữ liệu:

Mô hình chỉ có thể nhận diện được các định dạng ảnh jpeg, jpg, bmp, png nên nhóm viết hàm `remove_img` để lọc ra các file có định dạng khác và không phù hợp, chỉ chừa lại các ảnh phù hợp để train.

Một số ảnh bị lọc ra:



Bike (3)



Bike (17)



Bike (26)



Bike (132)



Bike (136)



Bike (138)



Bike (184)



Car (2)



Car (15)

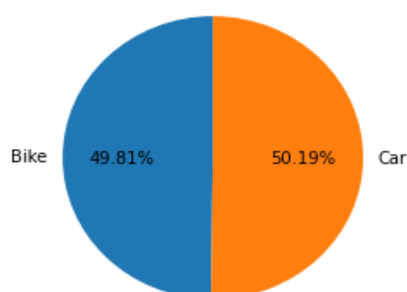


Sau khi lọc, dataset còn lại 1983 ảnh xe máy và 1998 ảnh xe oto.

```
[ ] len_dir_last = [len(os.listdir(os.path.join(data_dir, i))) for i in name_dir ]  
    print(len_dir_last)
```

```
[1983, 1998]
```

```
plt.pie(len_dir , labels = name_dir , autopct='%1.2f%%', startangle = 90)  
plt.show()
```



## Đóng gói các ảnh thành batch:

Chương trình không thể thực hiện việc lấy tất cả ảnh lên để xử lý cùng lúc, vì vậy các ảnh cần phải được đóng gói thành các batch rồi lần lượt đưa vào mô hình. Mỗi batch sẽ bao gồm 32 ảnh, mỗi ảnh được điều chỉnh lại thành mảng ma trận 60x60x3 (3 là giá trị RGB). Đồng thời việc xử lý ảnh theo batch còn giúp tăng tốc độ train mô hình.

```
[ ] IMAGE_SIZE = 60  
    BATCH_SIZE = 32  
    CHANNELS = 3
```

```
[ ] Data = tf.keras.preprocessing.image_dataset_from_directory(  
    'Car-Bike-Dataset',  
    batch_size = 32,  
    image_size = (IMAGE_SIZE , IMAGE_SIZE),  
    shuffle = True  
)
```

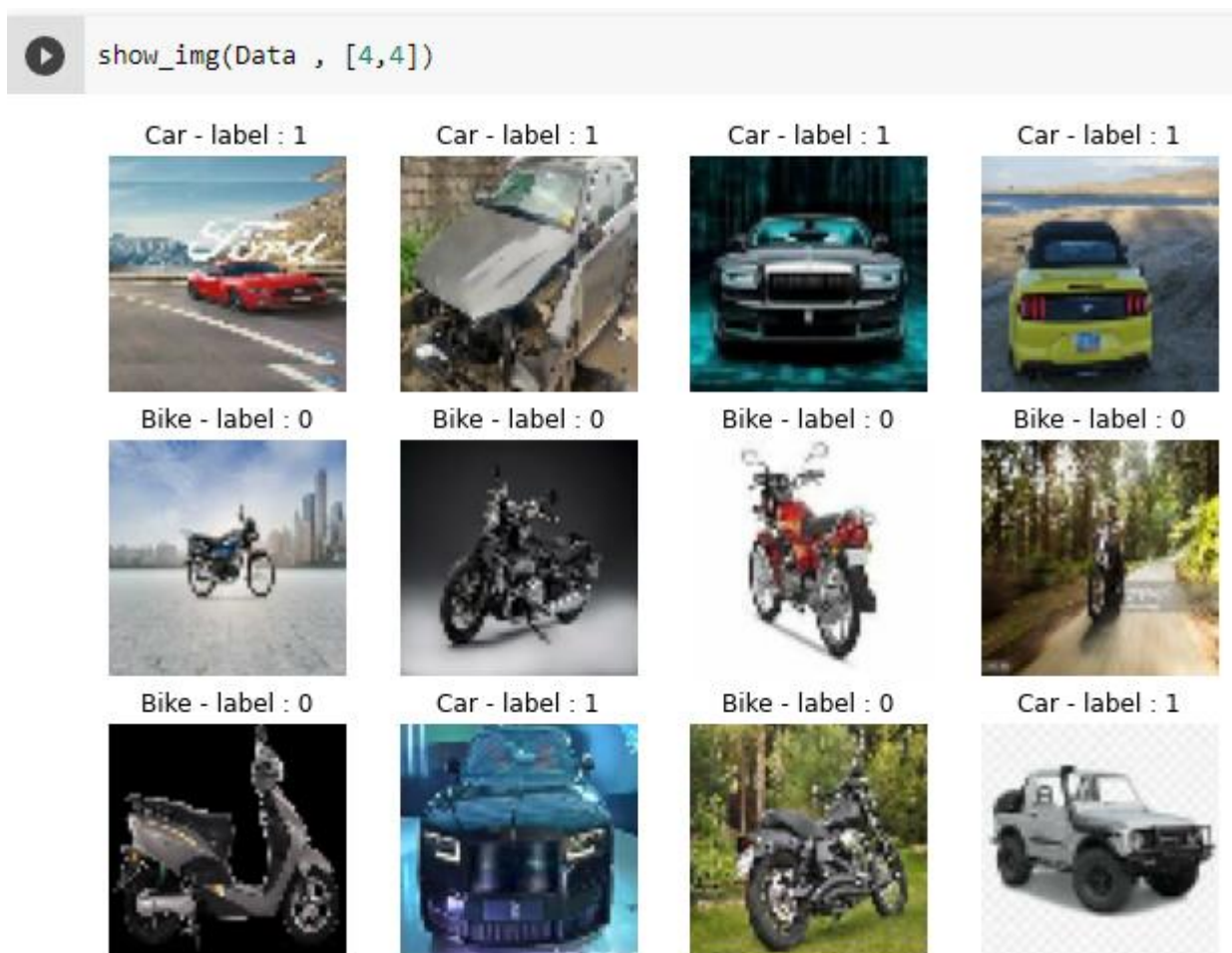
```
Found 3981 files belonging to 2 classes.
```

```
[ ] print('Number Batch : {}'.format(len(Data)))
```

```
Number Batch : 125
```

Sau đó sẽ gán nhãn dữ liệu cho từng ảnh, 0 ứng với xe máy và 1 ứng với xe oto.

Dữ liệu sau khi được gán nhãn:



### Chia dữ liệu:

Dữ liệu được chia ngẫu nhiên thành 3 bộ: train chiếm 80% (100 batch), validate chiếm 10% (12 batch) và test chiếm 10% (13 batch).

```
[ ] print('Number Train : {}'.format(len(Train)))  
    print('Number Val : {}'.format(len(Val)))  
    print('Number Test : {}'.format(len(Test)))
```

```
Number Train : 100  
Number Val : 12  
Number Test : 13
```

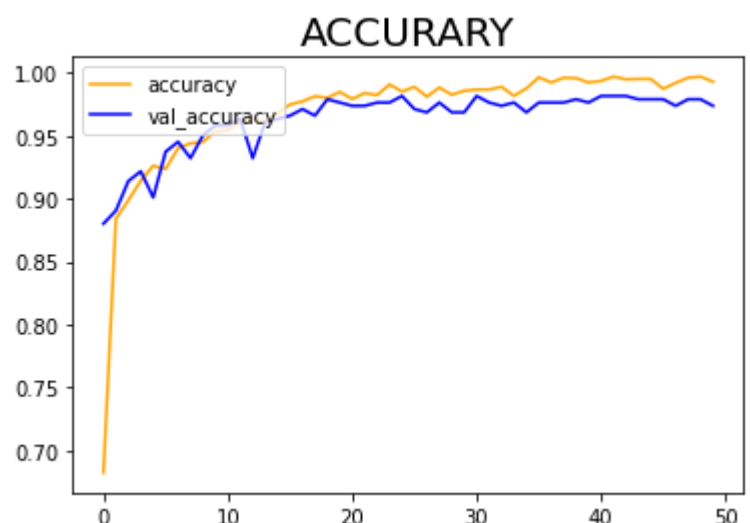
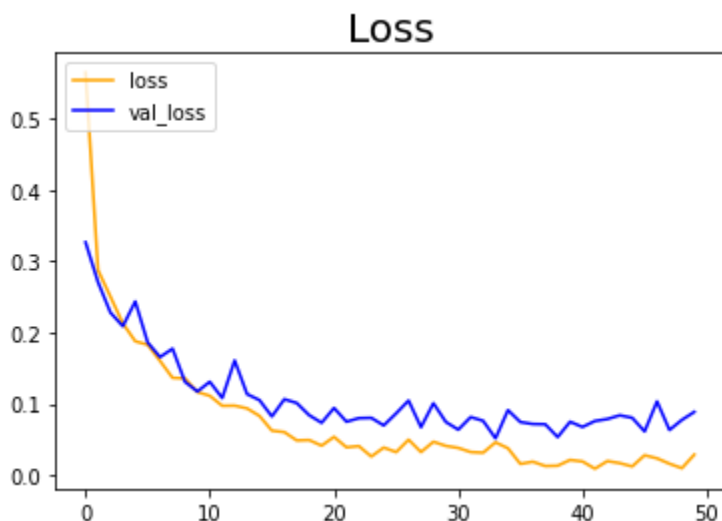
### 3. Train mô hình

Sử dụng mạng CNN gồm 3 Convolution Layers. 2 Layer đầu sử dụng 32 filters, size 3x3. Lớp layer thứ 3 sử dụng 64 filters, size 3x3. Hàm kích hoạt dùng trong các Convolution Layer là ReLU. Sau mỗi lớp Convolution Layers đều sử dụng MaxPooling size 2x2 để giảm kích thước ma trận ảnh lại.

Độ chính xác sau lần train đầu tiên là 68%, sau 50 lần, độ chính xác tăng lên 99.31%. Thời gian thực hiện 50 lần train tốn 30 phút. Thực hiện việc lưu lại mô hình đã train để tái sử dụng.

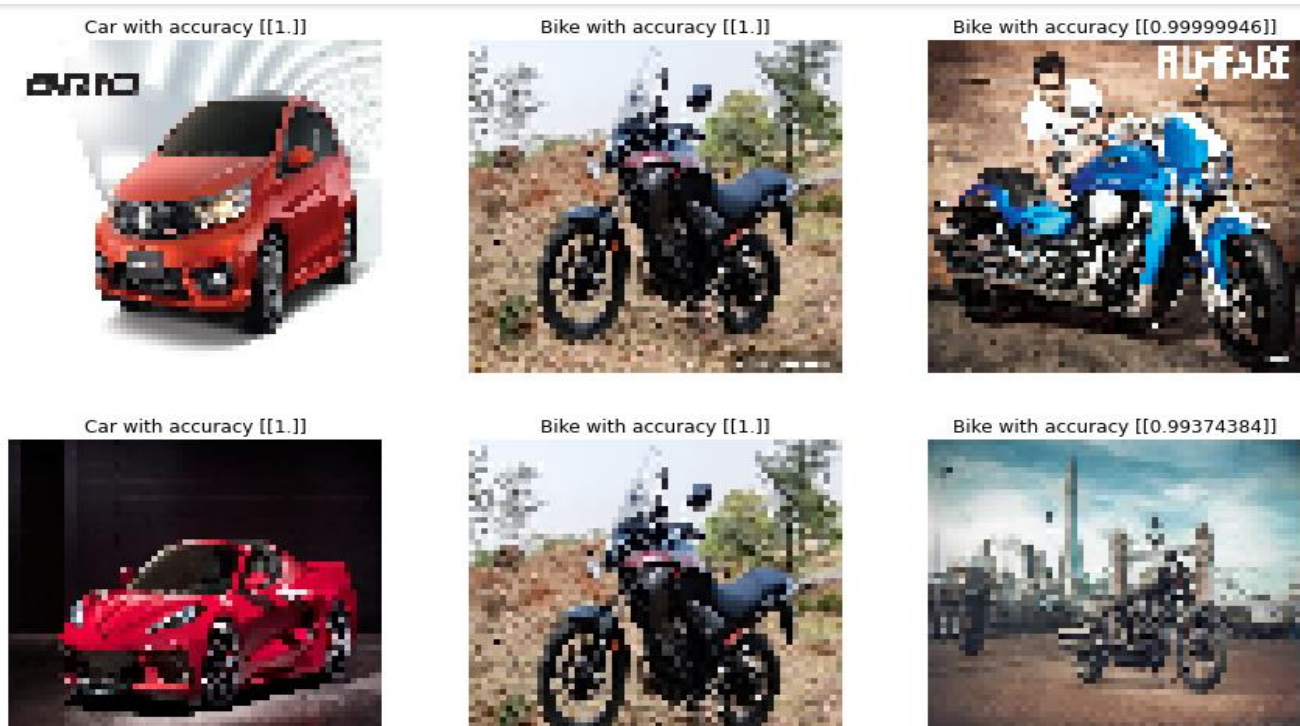
```
Epoch 50/50  
100/100 [=====] - ETA: 0s - loss: 0.0287 - binary_accuracy: 0.9931  
Epoch 50: saving model to training_1\cp.ckpt  
100/100 [=====] - 13s 128ms/step - loss: 0.0287 - binary_accuracy: 0.9931 - val_loss: 0.0887 - val_binary_accuracy: 0.9740
```

Hệ số loss và độ chính xác qua 50 lần train:



## 4. Demo

Mô hình thực hiện dự đoán các ảnh được lấy từ internet:



Sau khi train xong, nhóm em label một bộ dataset khác (được lưu trong folder img) gồm 31 ảnh xe máy và 35 ảnh xe oto được lấy ngẫu nhiên từ internet để kiểm tra độ hiệu quả của mô hình. Độ chính xác khi cho mô hình dự đoán dataset này là 95.45%

```
Data_Test = tf.keras.preprocessing.image_dataset_from_directory(  
    'img',  
    batch_size = 32,  
    image_size = (IMAGE_SIZE , IMAGE_SIZE),  
    shuffle = True  
)
```

Found 66 files belonging to 2 classes.

```
# test tu cac file anh dow tu ben ngoai ve  
new_model.evaluate(Data_Test)  
# do chinh xac sau khi test bo anh 95.45%
```

```
3/3 [=====] - 1s 26ms/step - loss: 0.3117 - binary_accuracy: 0.9545  
[0.3117428719997406, 0.9545454382896423]
```

### III. Kết luận

Qua thử nghiệm sử dụng CNN để phân loại ảnh xe oto và xe máy, chúng ta dễ dàng nhận thấy CNN giúp cho việc nhận dạng, đặc biệt là phân loại ảnh được thực hiện một cách tương đối dễ dàng cùng với độ chính xác rất cao (99% sau 50 lần train, sử dụng dataset có sẵn và 95% sử dụng bộ dataset khác do nhóm tự label) mà không cần phải qua quá nhiều bước xử lý. Tuy nhiên, CNN có một điểm yếu chí mạng là thời gian train mô hình tương đối lâu đồng thời đòi hỏi phải có GPU đủ mạnh để duyệt và rút trích ra những feature đặc biệt của ảnh. Ngoài ra CNN còn phụ thuộc khá nhiều vào lượng dữ liệu đầu vào, nếu lượng dữ liệu đầu vào không đủ lớn và không đủ độ đa dạng thì CNN sẽ không cho ra được các mô hình dự đoán với độ chính xác cao. Hơn nữa, việc hiểu và sử dụng các lớp trong mô hình CNN sao cho đúng, hiệu quả và tiết kiệm thời gian không phải là điều dễ dàng, cần phải hiểu rõ và kết hợp tốt các kỹ thuật nhân ma trận, hàm kích hoạt, pooling và các kỹ thuật khác như đóng gói và xử lý ảnh theo batch để tối ưu hóa thời gian và tài nguyên. Nhóm sử dụng CPU i5 6200U và card đồ họa tích hợp iGPU Intel HD Graphic 530 để train mô hình, tốn 30 phút cho 50 lần train. Thời gian train mô hình phần lớn sẽ phụ thuộc vào sức mạnh của GPU.

## IV. Nguồn tham khảo

[Convolutional Layer - an overview | ScienceDirect Topics](#)

[CNN | Introduction to Pooling Layer - GeeksforGeeks](#)

[Fully Connected Layer vs Convolutional Layer: Explained | Built In](#)

[Binary Cross Entropy/Log Loss for Binary Classification \(analyticsvidhya.com\)](#)

[\[Deep Learning\] Tìm hiểu về mạng tích chập \(CNN\) \(viblo.asia\)](#)

[Mạng nơ-ron tích chập \(P1\) \(viblo.asia\)](#)

[Mạng nơ-ron tích chập \(P2-hết\) \(viblo.asia\)](#)

**Cảm ơn thầy và các bạn đã xem qua bài thuyết trình.**

**Chúc mọi người có một ngày vui vẻ.**

**-Hết-**