



Classification Image

ผู้จัดทำ

นายชวลิต จันทรอินตา 6004062630078

นายณภัทร ปิยะวงศ์ 6004062630124

รายงานนี้เป็นส่วนหนึ่งของวิชา Neural Network (040613462)

ภาคเรียนที่ 2 ปีการศึกษา 2564

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

สารบัญ

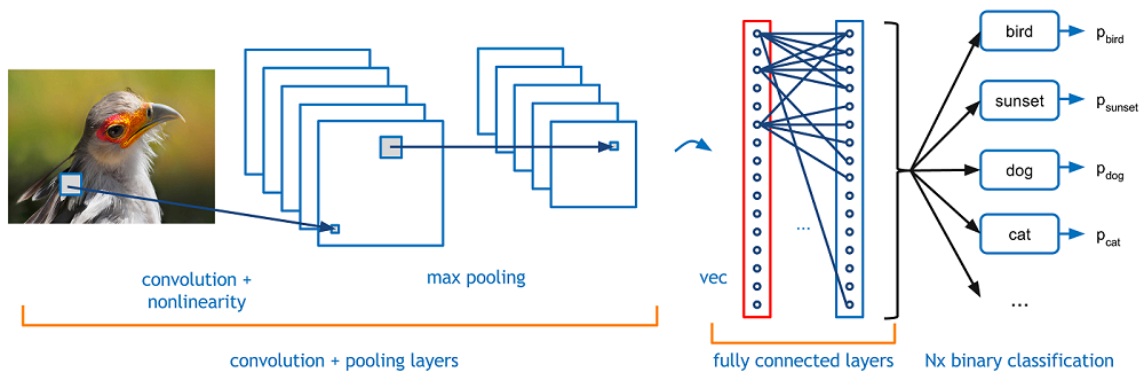
ที่มาและความสำคัญ	3
Architecture ของ Model ที่ใช้	3
Hardware ที่ใช้ในการ Train , Run	4
Software ที่ใช้ในการ Label , Train , Test	4
Library ที่ใช้ในการ Label , Train , Test	5
API ที่ใช้ในการ Label , Train , Test	6
วิธีการ Label	7
วิธีการ Preprocess	8
วิธีการ Train	9
วิธีการ Test	10
วิธีเปรียบเทียบประสิทธิภาพของโมเดล	15
Reference	19

1. ที่มาและความสำคัญ

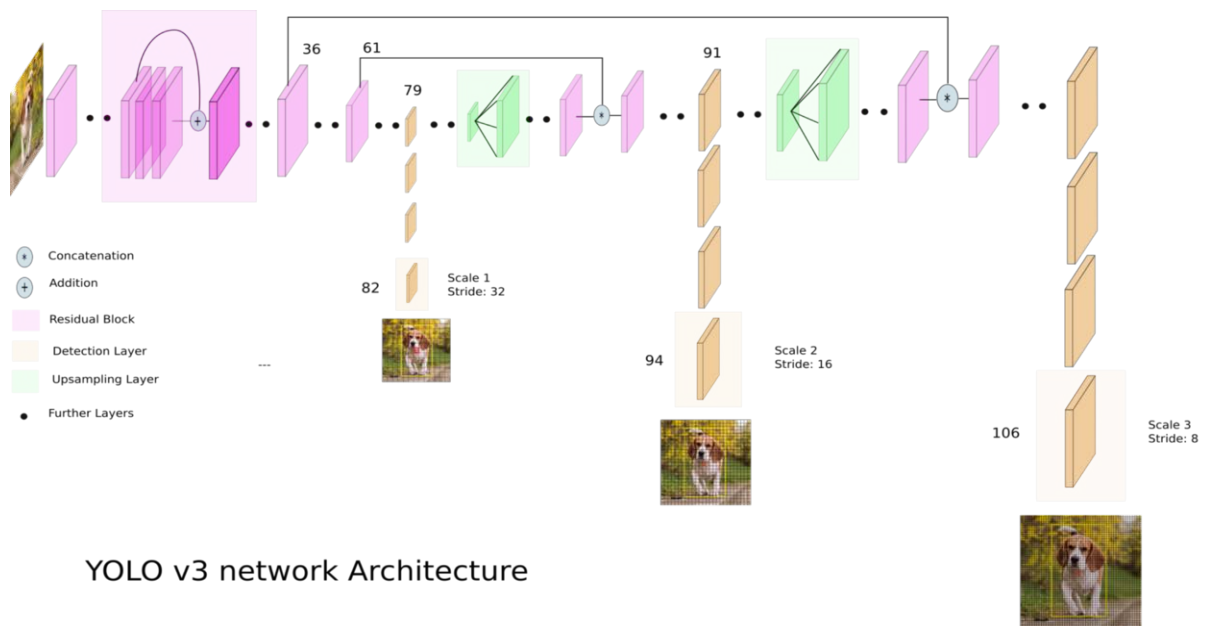
ระบบ Autopilot คือการเพิ่มความปลอดภัยบนถนน เพราะด้วยการใช้ระบบช่วยขับนี้จะทำให้ลดความผิดพลาดของผู้ขับที่อาจพลั้งเผลอหรือเหม่อ และช่วยในการเรื่องของการความสะดวกสบาย ในการขับขี่ทั้งหมดของ Autopilot ล้วนมีพื้นฐานมาจาก CNN ที่ช่วยในการแยกแยะวัตถุสิ่งต่างๆ ที่อยู่รอบข้างรถบวกกับใช้ความสามารถของ LiDAR ประกอบเข้าไปด้วย

2. Architecture ของ Model ที่ใช้

Convolutional Neural Network



Yolo Version 3

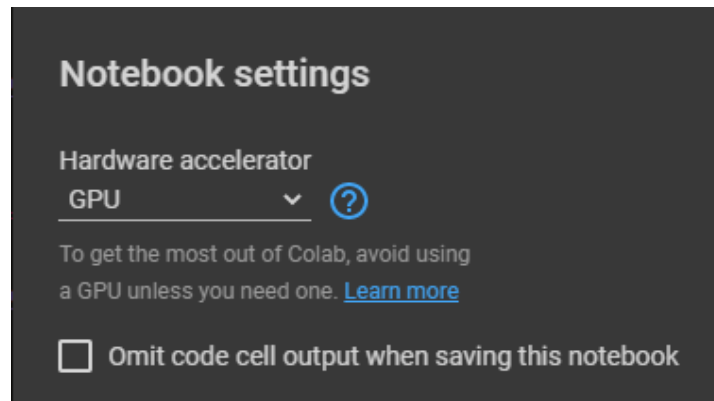


YOLO v3 network Architecture

3. Hardware ที่ใช้ในการ Train , Run

Convolutional Neural Network และ Yolo Version 3

- GPU ของ Colab



4. Software ที่ใช้ในการ Label , Train , Test

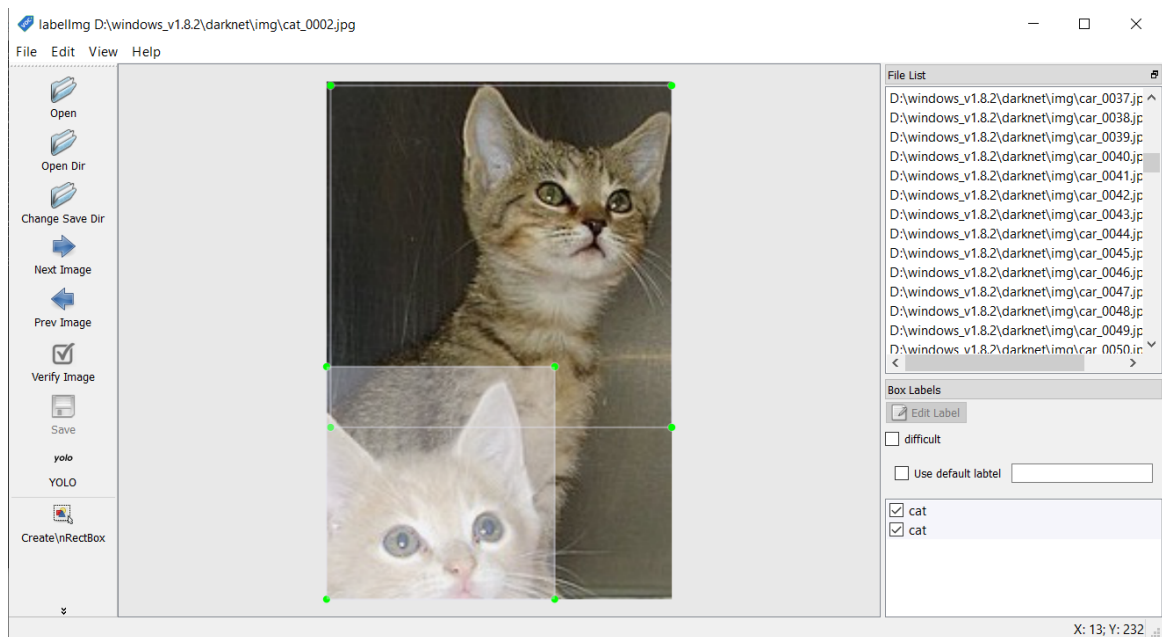
Convolutional Neural Network และ Yolo Version 3

- Colab



Yolo Version 3

- โปรแกรม LabelImg



5. Library ที่ใช้ในการ Label , Train , Test

Convolutional Neural Network

```
import os

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from keras import optimizers, metrics, models
from keras.layers import Dense, Conv2D, Flatten,
    Activation, MaxPooling2D, Dropout
from keras.utils.np_utils import to_categorical
from google.colab import files
from keras.utils import plot_model
from keras.callbacks import EarlyStopping
from sklearn.metrics import classification_report
```

Yolo Version 3

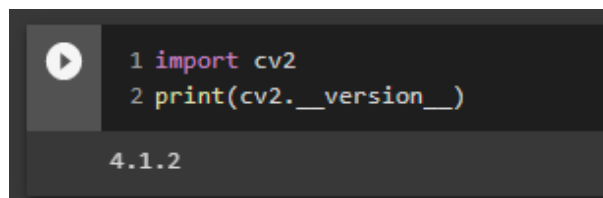
ใช้ในการ Train

- Darknet YOLO: Real-Time Object Detection Version 3

```
!./darknet detector train "/content/gdrive/My Drive/darknet/obj.data"  
"/content/gdrive/My Drive/darknet/cfg/yolov3.cfg" "/content/gdrive/My Drive/darknet/backup/yolov  
3_last.weights" -dont_show
```

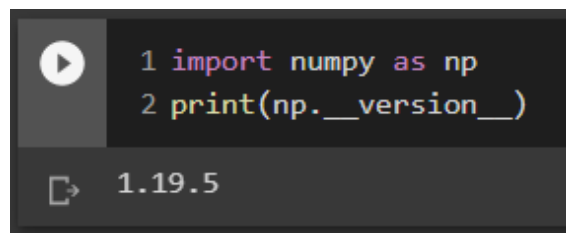
ใช้ในการ Test

- opencv version 4.1.2



```
1 import cv2  
2 print(cv2.__version__)  
  
4.1.2
```

- numpy version 1.19.5



```
1 import numpy as np  
2 print(np.__version__)  
  
1.19.5
```

```
import cv2  
import numpy as np
```

6. API ที่ใช้ในการ Label , Train , Test

Convolutional Neural Network

ใช้ในการเรียกใช้ Dataset จาก <https://www.kaggle.com/prasunroy/natural-images>

```
! pip install -q kaggle
```

7. วิธีการ Label

Convolutional Neural Network

วนลูปเก็บข้อมูลไว้ในตัวแปร ft และเก็บ Label หรือผลเฉลยของแต่ละรูปไว้ในตัวแปร lb

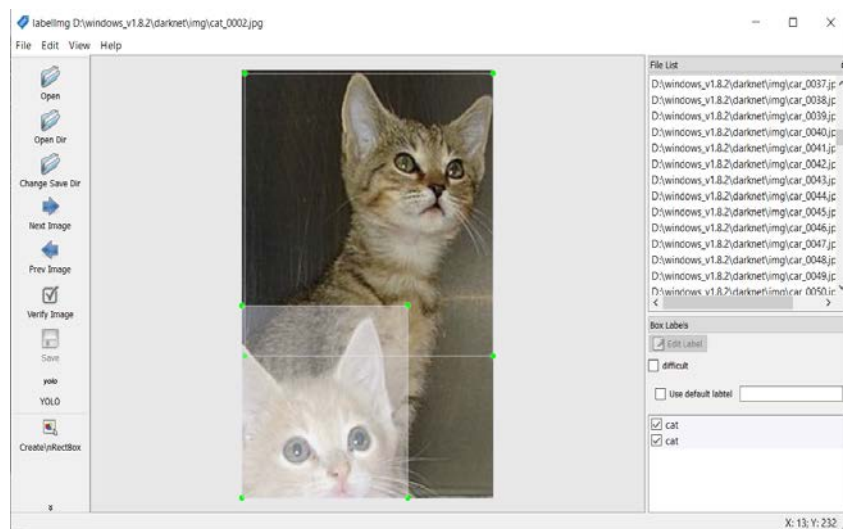
```
# Extracting label string and index
for i, k in enumerate(os.listdir(dir)):



















    if k in determine_label:
        labels[k] = r
        r += 1

# Appending features and labels
for i in determine_label:
    subdir = os.path.join(dir, i)
    for j in tqdm(os.listdir(subdir)):
        im = cv2.imread(os.path.join(subdir, j), 0)
        im = cv2.resize(im, (70, 70))
        ft.append(im)
        lb.append(labels[i])
```

Yolo Version 3

โปรแกรม labelImg



Name	Date modified	Type	Size
 cat_0000.jpg	2019/10/03 14:31	JPG File	39 KB
 cat_0001.jpg	2019/10/03 14:31	JPG File	36 KB
 cat_0002.jpg	2019/10/03 14:31	JPG File	21 KB
 cat_0003.jpg	2019/10/03 14:31	JPG File	30 KB
 cat_0004.jpg	2019/10/03 14:31	JPG File	48 KB
 cat_0005.jpg	2019/10/03 14:31	JPG File	24 KB
 cat_0006.jpg	2019/10/03 14:31	JPG File	30 KB
 cat_0007.jpg	2019/10/03 14:31	JPG File	36 KB
 cat_0008.jpg	2019/10/03 14:31	JPG File	43 KB
 cat_0009.jpg	2019/10/03 14:31	JPG File	37 KB
 cat_0010.jpg	2019/10/03 14:31	JPG File	67 KB
 cat_0011.jpg	2019/10/03 14:31	JPG File	7 KB
 cat_0012.jpg	2019/10/03 14:31	JPG File	39 KB
 cat_0013.jpg	2019/10/03 14:31	JPG File	21 KB
 cat_0014.jpg	2019/10/03 14:31	JPG File	45 KB
 cat_0015.jpg	2019/10/03 14:31	JPG File	12 KB
 cat_0016.jpg	2019/10/03 14:31	JPG File	9 KB
 cat_0017.jpg	2019/10/03 14:31	JPG File	53 KB

8. วิธีการ Preprocess

Convolutional Neural Network

- เก็บข้อมูลและ Label ในรูปแบบ Array โดยใช้ np.array
- ตัวแปร X เก็บข้อมูลโดยทำการ reshape ให้ channel = 1 หรือ gray channel ทำให้รูปมีสีเทา และถูก Normalize ด้วยการหารด้วย 255
- ตัวแปร Y ฟังก์ชัน `to_categorical()` แปลง label ให้อยู่ในรูป One-hot Encoding
- แบ่งข้อมูล Train Test เป็น 70:30 ตาม Default ของ `train_test_split(X, Y)`

```
ft = np.array(ft)
lb = np.array(lb).reshape(len(ft), 1)
X = ft.reshape(-1, 70, 70, 1)
Y = to_categorical(lb)

# Normalizing feautures values
X = X/255.0

# Spliting into train and test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y)
```


Yolo Version 3

เป็นการเรียกใช้ฟังก์ชันของ openCV ในการ

- ปรับ scalefactor เป็น 0.00392
- ปรับขนาดภาพ เป็น (608, 608)
- สลับค่าสี Red กับ Blue

```
blob = cv2.dnn.blobFromImage(image, 0.00392 , (608, 608), swapRB=True, crop=False)
```

9. วิธีการ Train

Convolutional Neural Network

- กำหนด epoch = 180 ซึ่งเป็นจำนวนรอบในการ Train
- เรียกใช้ callbacks=[early_stopping_monitor] เพื่อป้องกันการเกิด Overfitting

```
# Compiling the model
model.compile(optimizer='sgd',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Importing a callback
from keras.callbacks import EarlyStopping
early_stopping_monitor = EarlyStopping(patience=5)

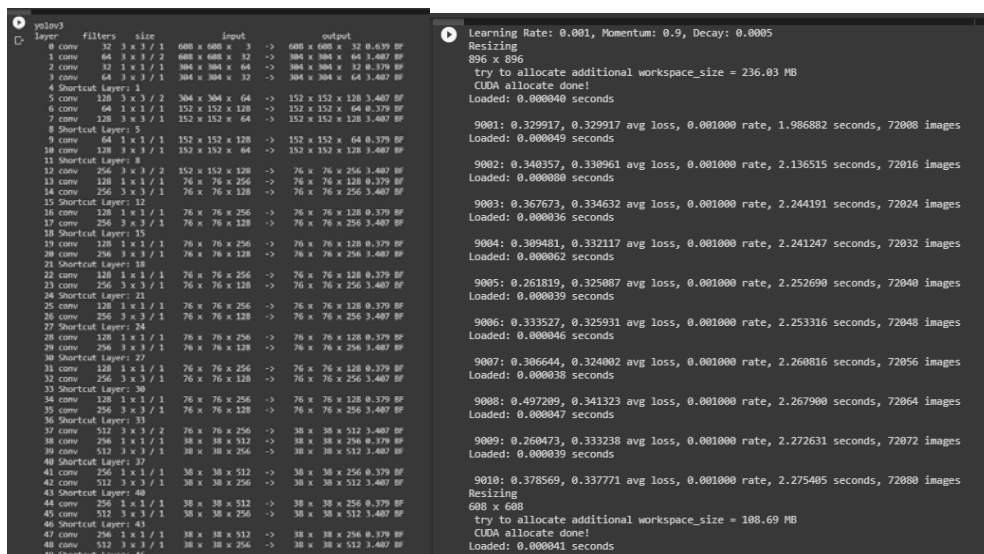
hist = model.fit(x_train, y_train, epochs=180,
                callbacks=[early_stopping_monitor],
                validation_data=(x_test, y_test))
```

Yolo Version 3

- ใช้คำสั่งของ Yolo Darknet Version 3
- Train ไปทั้งหมด 9000

```
./darknet detector train "/content/gdrive/My Drive/darknet/obj.data"  
"/content/gdrive/My Drive/darknet/cfg/yolov3.cfg" "/content/gdrive/My Drive/darknet/backup/yolov  
3_last.weights" -dont_show
```

ตัวอย่างการ Train



```
yolov3  
layer  filters  size  input  output  
0 conv  32 3 x 3 / 1  608 x 608 x 3  -> 608 x 608 x 32 0.639 BF  
1 conv  64 3 x 3 / 2  608 x 608 x 32  -> 304 x 304 x 64 1.687 BF  
2 conv  32 1 x 1 / 1  304 x 304 x 64  -> 304 x 304 x 32 0.378 BF  
3 conv  64 3 x 3 / 1  304 x 304 x 32  -> 304 x 304 x 64 1.687 BF  
4 Shortcut Layer: 1  
5 conv  128 3 x 3 / 2  304 x 304 x 64  -> 152 x 152 x 128 3.467 BF  
6 conv  64 1 x 1 / 1  152 x 152 x 128  -> 152 x 152 x 64 0.379 BF  
7 conv  128 3 x 3 / 1  152 x 152 x 64  -> 152 x 152 x 128 3.467 BF  
8 Shortcut Layer: 5  
9 conv  64 1 x 1 / 1  152 x 152 x 128  -> 152 x 152 x 64 0.379 BF  
10 conv  128 3 x 3 / 1  152 x 152 x 64  -> 152 x 152 x 128 3.467 BF  
11 Shortcut Layer: 8  
12 conv  256 3 x 3 / 2  152 x 152 x 128  -> 76 x 76 x 256 1.487 BF  
13 conv  128 1 x 1 / 1  76 x 76 x 256  -> 76 x 76 x 128 0.379 BF  
14 conv  256 3 x 3 / 1  76 x 76 x 128  -> 76 x 76 x 256 3.467 BF  
15 Shortcut Layer: 12  
16 conv  128 1 x 1 / 1  76 x 76 x 256  -> 76 x 76 x 128 0.379 BF  
17 conv  256 3 x 3 / 1  76 x 76 x 128  -> 76 x 76 x 256 3.467 BF  
18 Shortcut Layer: 15  
19 conv  128 1 x 1 / 1  76 x 76 x 256  -> 76 x 76 x 128 0.379 BF  
20 conv  256 3 x 3 / 1  76 x 76 x 128  -> 76 x 76 x 256 3.467 BF  
21 Shortcut Layer: 18  
22 conv  128 1 x 1 / 1  76 x 76 x 256  -> 76 x 76 x 128 0.379 BF  
23 conv  256 3 x 3 / 1  76 x 76 x 128  -> 76 x 76 x 256 3.467 BF  
24 Shortcut Layer: 21  
25 conv  128 1 x 1 / 1  76 x 76 x 256  -> 76 x 76 x 128 0.379 BF  
26 conv  256 3 x 3 / 1  76 x 76 x 128  -> 76 x 76 x 256 3.467 BF  
27 Shortcut Layer: 24  
28 conv  128 1 x 1 / 1  76 x 76 x 256  -> 76 x 76 x 128 0.379 BF  
29 conv  256 3 x 3 / 1  76 x 76 x 128  -> 76 x 76 x 256 3.467 BF  
30 Shortcut Layer: 27  
31 conv  128 1 x 1 / 1  76 x 76 x 256  -> 76 x 76 x 128 0.379 BF  
32 conv  256 3 x 3 / 1  76 x 76 x 128  -> 76 x 76 x 256 3.467 BF  
33 Shortcut Layer: 30  
34 conv  128 1 x 1 / 1  76 x 76 x 256  -> 76 x 76 x 128 0.379 BF  
35 conv  256 3 x 3 / 1  76 x 76 x 128  -> 76 x 76 x 256 3.467 BF  
36 Shortcut Layer: 33  
37 conv  512 3 x 3 / 2  76 x 76 x 256  -> 38 x 38 x 512 1.487 BF  
38 conv  256 1 x 1 / 1  38 x 38 x 512  -> 38 x 38 x 256 0.379 BF  
39 conv  512 3 x 3 / 1  38 x 38 x 256  -> 38 x 38 x 512 1.487 BF  
40 Shortcut Layer: 37  
41 conv  256 1 x 1 / 1  38 x 38 x 512  -> 38 x 38 x 256 0.379 BF  
42 conv  512 3 x 3 / 1  38 x 38 x 256  -> 38 x 38 x 512 1.487 BF  
43 Shortcut Layer: 40  
44 conv  256 1 x 1 / 1  38 x 38 x 512  -> 38 x 38 x 256 0.379 BF  
45 conv  512 3 x 3 / 1  38 x 38 x 256  -> 38 x 38 x 512 1.487 BF  
46 Shortcut Layer: 43  
47 conv  256 1 x 1 / 1  38 x 38 x 512  -> 38 x 38 x 256 0.379 BF  
48 conv  512 3 x 3 / 1  38 x 38 x 256  -> 38 x 38 x 512 1.487 BF  
49 Shortcut Layer: 46  
  
Learning Rate: 0.001, Momentum: 0.9, Decay: 0.0005  
Resizing  
608 x 608  
try to allocate additional workspace_size = 236.83 MB  
CUDA allocate done!  
Loaded: 0.000040 seconds  
9001: 0.329917, 0.329917 avg loss, 0.001000 rate, 1.986882 seconds, 72008 images  
Loaded: 0.000049 seconds  
9002: 0.340357, 0.330961 avg loss, 0.001000 rate, 2.136515 seconds, 72016 images  
Loaded: 0.000080 seconds  
9003: 0.367673, 0.334632 avg loss, 0.001000 rate, 2.244191 seconds, 72024 images  
Loaded: 0.000036 seconds  
9004: 0.309481, 0.332117 avg loss, 0.001000 rate, 2.241247 seconds, 72032 images  
Loaded: 0.000062 seconds  
9005: 0.261819, 0.325087 avg loss, 0.001000 rate, 2.252690 seconds, 72040 images  
Loaded: 0.000039 seconds  
9006: 0.333527, 0.325931 avg loss, 0.001000 rate, 2.253316 seconds, 72048 images  
Loaded: 0.000046 seconds  
9007: 0.306644, 0.324002 avg loss, 0.001000 rate, 2.268816 seconds, 72056 images  
Loaded: 0.000030 seconds  
9008: 0.497209, 0.341323 avg loss, 0.001000 rate, 2.267900 seconds, 72064 images  
Loaded: 0.000047 seconds  
9009: 0.260473, 0.333238 avg loss, 0.001000 rate, 2.272631 seconds, 72072 images  
Loaded: 0.000039 seconds  
9010: 0.378569, 0.337771 avg loss, 0.001000 rate, 2.275405 seconds, 72080 images  
Resizing  
608 x 608  
try to allocate additional workspace_size = 188.69 MB  
CUDA allocate done!  
Loaded: 0.000041 seconds
```

10. วิธีการ Test

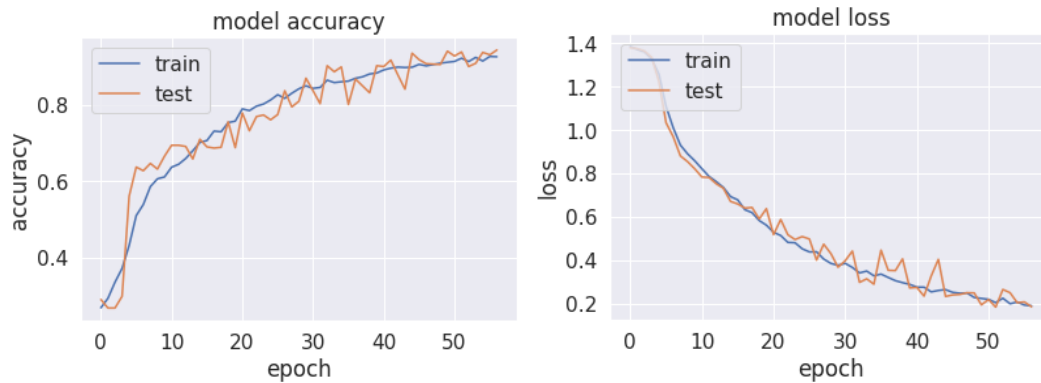
Convolutional Neural Network

```
print('Training Accuracy: {0:.2%}'  
      .format(float(model.evaluate(x_train,y_train)[1])))  
print('Testing Accuracy: {0:.2%}'  
      .format(float(model.evaluate(x_test,y_test)[1])))
```

- ผลลัพธ์จากโค้ด ที่บอกถึง Accuracy ของ Train และ Test

```
89/89 [=====] - 1s 6ms/step - loss: 0.1676 - accuracy: 0.9417
Training Accuracy: 94.17%
30/30 [=====] - 0s 6ms/step - loss: 0.1863 - accuracy: 0.9421
Testing Accuracy: 94.21%
```

- กราฟของ model ที่บอกถึงค่า Accuracy และ Loss ของ Train และ Test ระหว่าง Train model



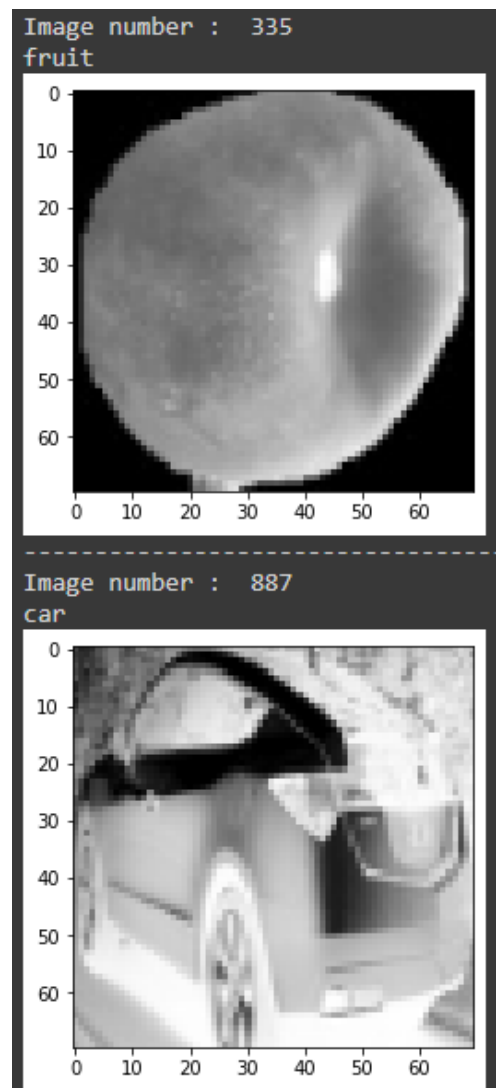
- เป็นโค้ดสำหรับใช้ Predict ภาพจาก model

```
# Prediction
import random
def predict(i):
    pre_img = i
    print('Image number : ',pre_img)

    print(rev_labels[np.argmax(
        model.predict([x_test[pre_img:pre_img+1]])))
    plt.imshow(x_test[pre_img:pre_img+1]
        .reshape(70,70), cmap='binary')
    plt.show()
    print('-----')

for i in range(5):
    predict(i+random.randint(0,x_test.shape[0]))
```

- ผลลัพธ์จากโค้ด



Yolo Version 3

- เป็นโค้ดสำหรับใช้ Predict ภาพจาก model

```
import cv2
import numpy as np

def get_output_layers(net):
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
    return output_layers
```

```

def draw_prediction(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
    label = str(classes[class_id])
    print(label)
    color = COLORS[class_id]
    cv2.rectangle(img, (x,y), (x_plus_w,y_plus_h), color, 1)
    cv2.putText(img, label, (x-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

## read input image
image = cv2.imread("/content/gdrive/My Drive/darknet/img/dog_0051.jpg")
Width = image.shape[1]
Height = image.shape[0]

## read object class name
classes = None
with open("/content/gdrive/My Drive/darknet/obj.names", 'r') as f:
    classes = [line.strip() for line in f.readlines()]

## random boundingbox colors for each object label
COLORS = np.random.uniform(0, 255, size=(len(classes), 3))

## read yolo's weights
net = cv2.dnn.readNet("/content/gdrive/My Drive/darknet/backup/yolov3_last.weights", "/content/gdrive/My Drive/darknet/cfg/yolov3.cfg")

## preprocessing image
blob = cv2.dnn.blobFromImage(image, 0.00392, (608, 608), True, crop=False)
net.setInput(blob)

## object detection process
outs = net.forward(get_output_layers(net))
class_ids = []
confidences = []
boxes = []
conf_threshold = 0.5          ### confidence threshold
nms_threshold = 0.4          ### non maximum supression threshold

```

```

for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > conf_threshold:
            center_x = int(detection[0] * Width)
            center_y = int(detection[1] * Height)
            w = int(detection[2] * Width)
            h = int(detection[3] * Height)
            x = center_x - w / 2
            y = center_y - h / 2
            class_ids.append(class_id)
            confidences.append(float(confidence))
            boxes.append([x, y, w, h])

indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)

for i in indices:
    i = i[0]
    box = boxes[i]
    x = box[0]
    y = box[1]
    w = box[2]
    h = box[3]
    draw_prediction(image, class_ids[i], confidences[i], round(x), round(y), round(x+w), round(y+h))

##### detected image #####

from google.colab.patches import cv2_imshow
cv2_imshow(image)

```

- ผลลัพธ์จากโค้ด จะทำการสร้างกรอบล้อมรอบสิ่งที่เป็นคำตอบและบอกว่าสิ่งนั้นคืออะไร



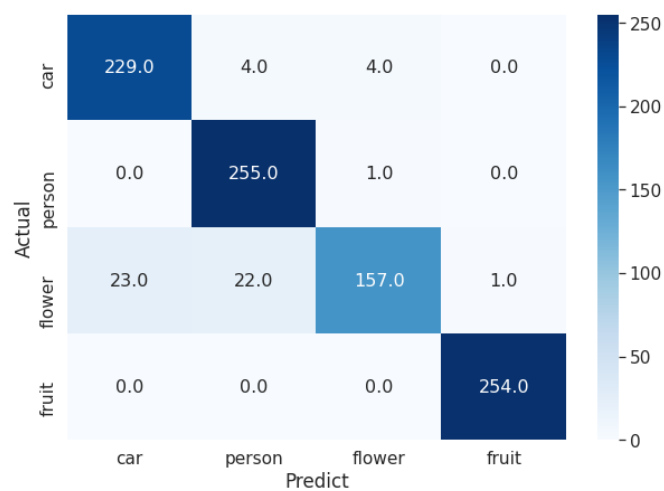
11. วิธีเปรียบเทียบประสิทธิภาพของโมเดล

Convolutional Neural Network

ผลลัพธ์จาก Convolutional Neural Network

	precision	recall	f1-score	support
0	0.91	0.97	0.94	237
1	0.91	1.00	0.95	256
2	0.97	0.77	0.86	203
3	1.00	1.00	1.00	254
accuracy			0.94	950
macro avg	0.95	0.93	0.94	950
weighted avg	0.94	0.94	0.94	950

ค่า Accuracy \approx 94% หรือ 0.94



วิธีการคำนวณ

Car	Actual			
Predict	TP	FP	Accuracy	$(229 + 690) / (229 + 690 + 8 + 23) = 0.97$
	229	23	Precision	$229 / (229 + 23) = 0.91$
	8	690	Recall	$229 / (229 + 8) = 0.97$
	FN	TN	F1-Score	$2 * ((0.91 * 0.97) / (0.91 + 0.97)) = 0.94$

Person	Actual			
Predict	TP	FP	Accuracy	$(255 + 668) / (255 + 668 + 1 + 26) = 0.97$
	255	26	Precision	$255 / (255 + 26) = 0.91$
	1	668	Recall	$255 / (255 + 1) = 1.00$
	FN	TN	F1-Score	$2 * ((0.91 * 1.00) / (0.91 + 1.00)) = 0.95$

Flower	Actual			
Predict	TP	FP	Accuracy	$(157 + 742) / (157 + 742 + 5 + 46) = 0.95$
	157	5	Precision	$157 / (157 + 5) = 0.97$
	46	742	Recall	$157 / (157 + 46) = 0.77$
	FN	TN	F1-Score	$2 * ((0.97 * 0.77) / (0.97 + 0.77)) = 0.86$

Fruit	Actual			
Predict	TP	FP	Accuracy	$(254 + 695) / (254 + 695 + 0 + 1) = 1.00$
	254	1	Precision	$254 / (254 + 1) = 1.00$
	0	695	Recall	$255 / (255 + 1) = 1.00$
	FN	TN	F1-Score	$2 * ((0.97 * 0.77) / (0.97 + 0.77)) = 1.00$

Pooled	Actual			Micro average
Predict	TP	FP	Accuracy	$(895 + 2795) / (895 + 2795 + 55 + 55) = 0.97$
	895	55	Precision	$895 / (895 + 55) = 0.94$
	55	2795	Recall	$895 / (895 + 55) = 0.94$
	FN	TN	F1-Score	$2 * ((0.94 * 0.94) / (0.94 + 0.94)) = 0.94$

Macro average

Precision	$(0.91 + 0.91 + 0.97 + 1.00) / 4 = 0.95$
Recall	$(0.97 + 1.00 + 0.77 + 1.00) / 4 = 0.93$
F1-Score	$(0.94 + 0.95 + 0.86 + 1.00) / 4 = 0.94$

Yolo Version 3

ผลลัพธ์จาก Yolo Version 3

```

detections_count = 126, unique_truth_count = 44
class_id = 0, name = airplane, ap = 100.00% (TP = 5, FP = 0)
class_id = 1, name = car, ap = 100.00% (TP = 5, FP = 2)
class_id = 2, name = cat, ap = 91.92% (TP = 4, FP = 0)
class_id = 3, name = dog, ap = 94.81% (TP = 5, FP = 2)
class_id = 4, name = flower, ap = 77.32% (TP = 6, FP = 0)
class_id = 5, name = fruit, ap = 100.00% (TP = 5, FP = 1)
class_id = 6, name = motorbike, ap = 100.00% (TP = 5, FP = 2)
class_id = 7, name = person, ap = 100.00% (TP = 5, FP = 0)

for thresh = 0.25, precision = 0.85, recall = 0.91, F1-score = 0.88
for thresh = 0.25, TP = 40, FP = 7, FN = 4, average IoU = 70.51 %

IoU threshold = 50 %, used 11 Recall-points
mean average precision (mAP@0.50) = 0.955054, or 95.51 %
Total Detection Time: 17.000000 Seconds

```

วิธีการคำนวณ

Accuracy

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

$$\text{Accuracy} = (40 + 0) / (40 + 0 + 7 + 4) = 0.78$$

Precision

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

$$\text{Precision} = 40 / (40 + 7) = 0.85$$

Recall

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{Recall} = 40 / (40 + 4) = 0.91$$

F1-Score

$$\text{F1-Score} = 2 * ((\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}))$$

$$\text{F1-Score} = 2 * ((0.85 * 0.91) / (0.85 + 0.91)) = 0.88$$

ตารางเปรียบเทียบประสิทธิภาพของโมเดล

	Convolutional Neural Network	Yolo Version 3
Accuracy	0.94	0.78
Precision	0.95	0.85
Recall	0.93	0.91
F1-Score	0.94	0.88

สรุป

Model **Convolutional Neural Network** มีค่า **Accuracy** มากกว่า Model **Yolo Version 3**

Model **Convolutional Neural Network** มีค่า **Precision** มากกว่า Model **Yolo Version 3**

Model **Convolutional Neural Network** มีค่า **Recall** มากกว่า Model **Yolo Version 3**

Model **Convolutional Neural Network** มีค่า **F1-Score** มากกว่า Model **Yolo Version 3**

12. Reference

Convolutional Neural Network

Image Data Set

<https://www.kaggle.com/prasunroy/natural-images>

Code Reference

<https://www.kaggle.com/err0r90/natural-images-with-cnn>

Yolo Version 3

Image Data Set

<https://www.kaggle.com/prasunroy/natural-images>

Code Reference

[https://colab.research.google.com/drive/1JRbOMBzcJVvRuxLHDj9v1g6N96](https://colab.research.google.com/drive/1JRbOMBzcJVvRuxLHDj9v1g6N96W5-3le)

[W5-3le](#)