

# การประกวด โครงงาน

IoT



SCHOOL OF  
SCIENCE

KMITL  
FIGHT

SCHOOL OF ENGINEERING KMITL

AIoT  
Artificial Intelligence  
School of Engineering KMITL

IMAKE

## เครื่องคัดแยกขยะใช้เคล

1. นายนรุตม์ กังสัมฤทธิ์
2. นายธีรเมธ ตันเล่ง
3. นายวีรศ สถิตเม่น

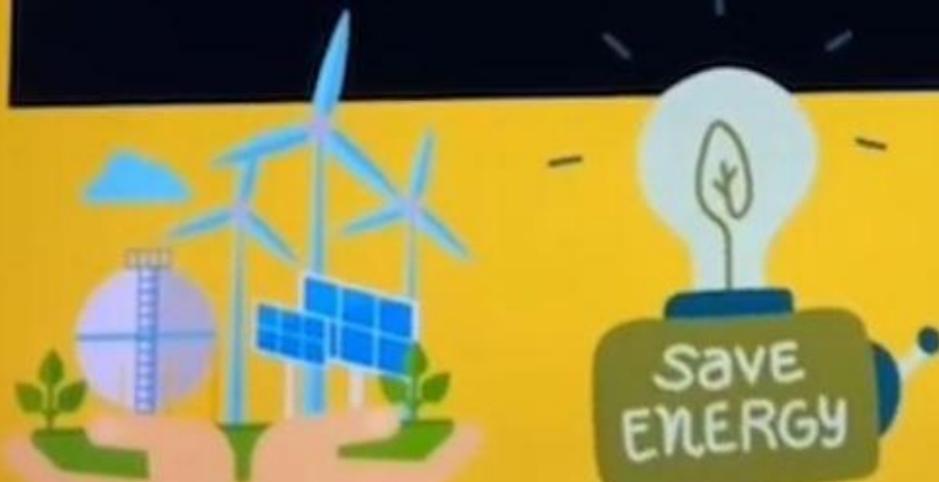
โรงเรียนสาธิตมหาวิทยาลัยศรีนครินทรวิโรฒ ฝ่ายมัธยม

# PGT Automation Project:

Plastic, Glass, and Tin (Aluminum Can)



การแปรรูปขยะเป็นผลิตภัณฑ์อีน  
หรือนำกลับมาใช้ซ้ำ เป็นการอนุรักษ์  
พลังงานและสิ่งแวดล้อม





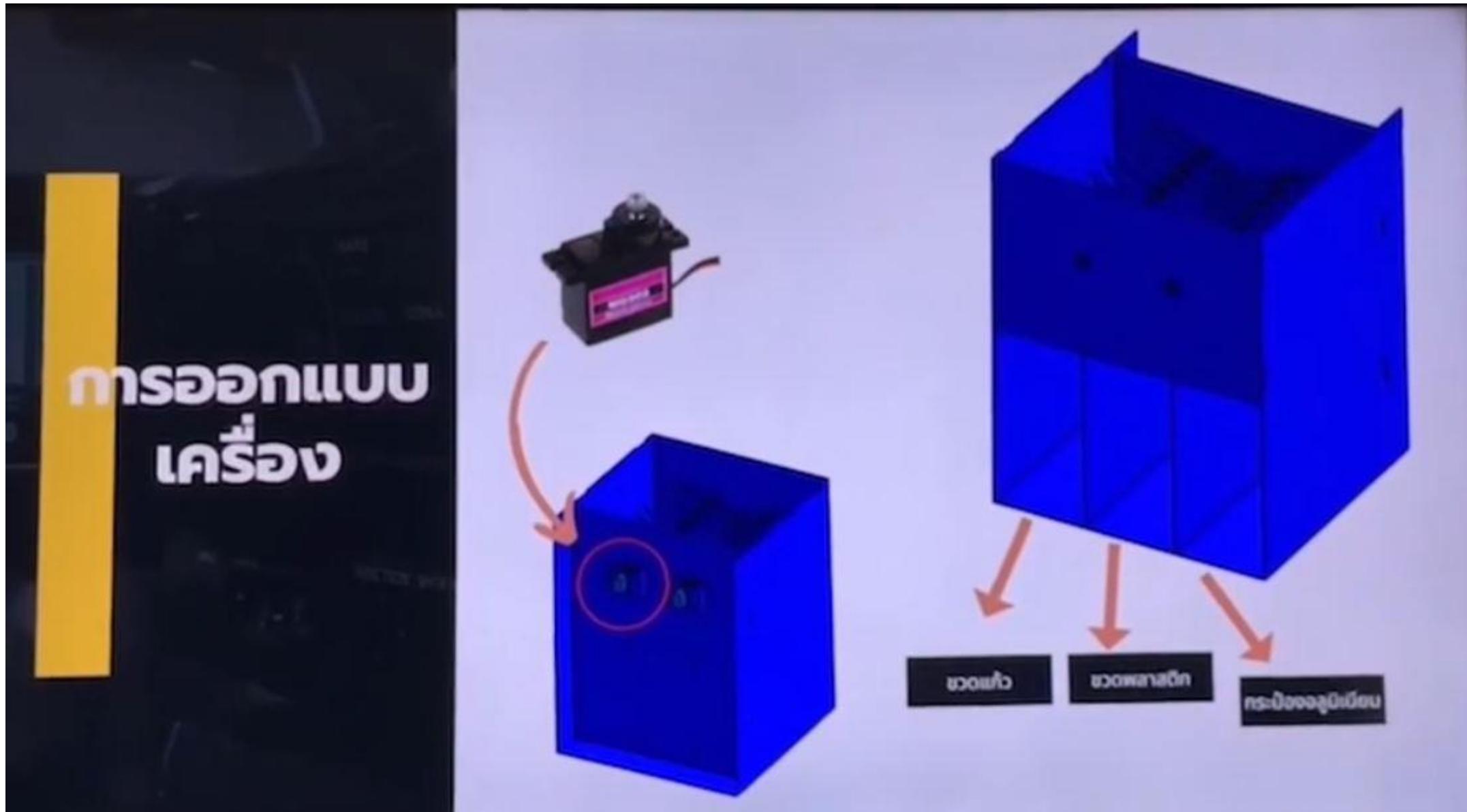
# PGT Automation Project:



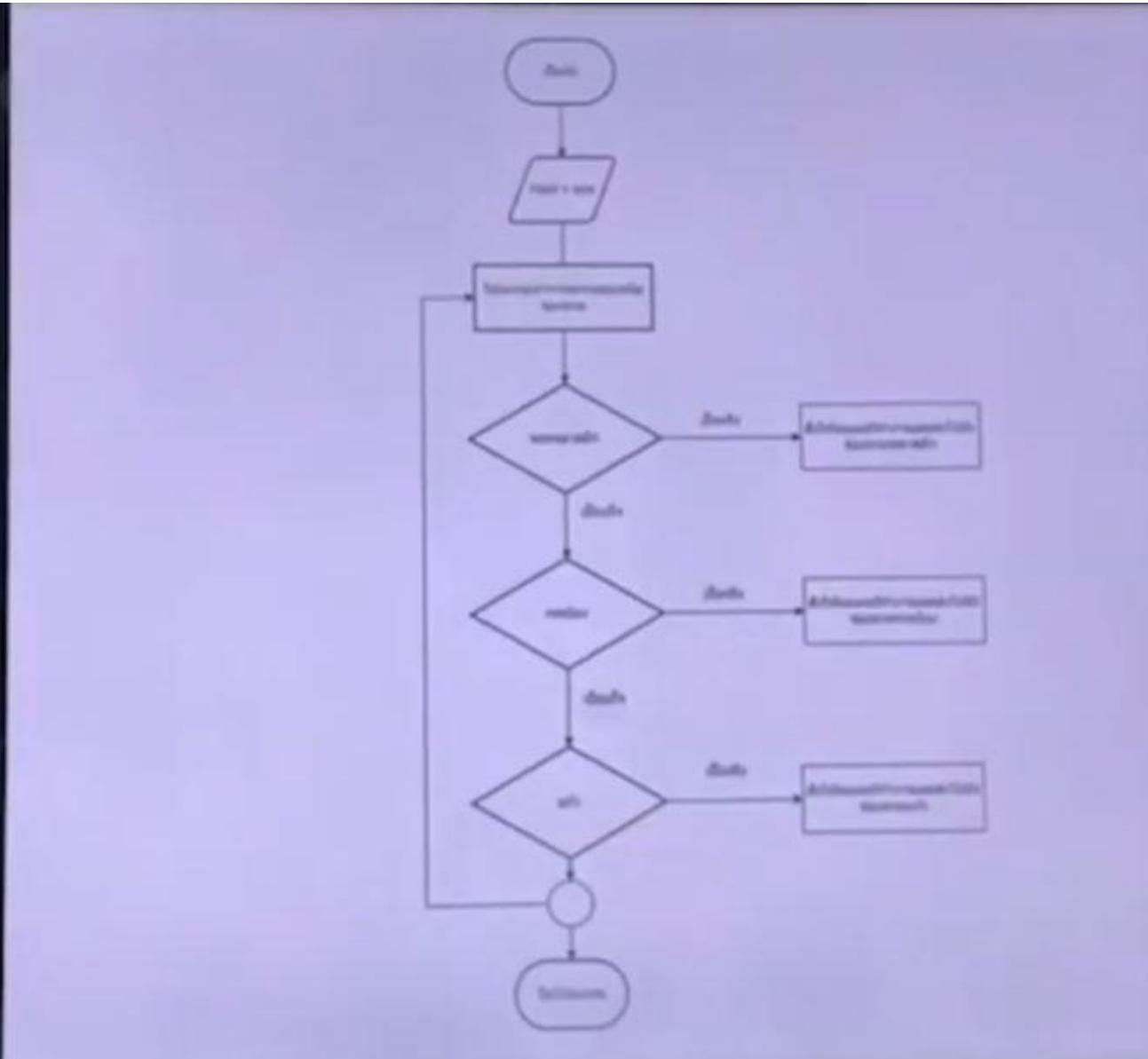
## เครื่องคัดแยกขวด



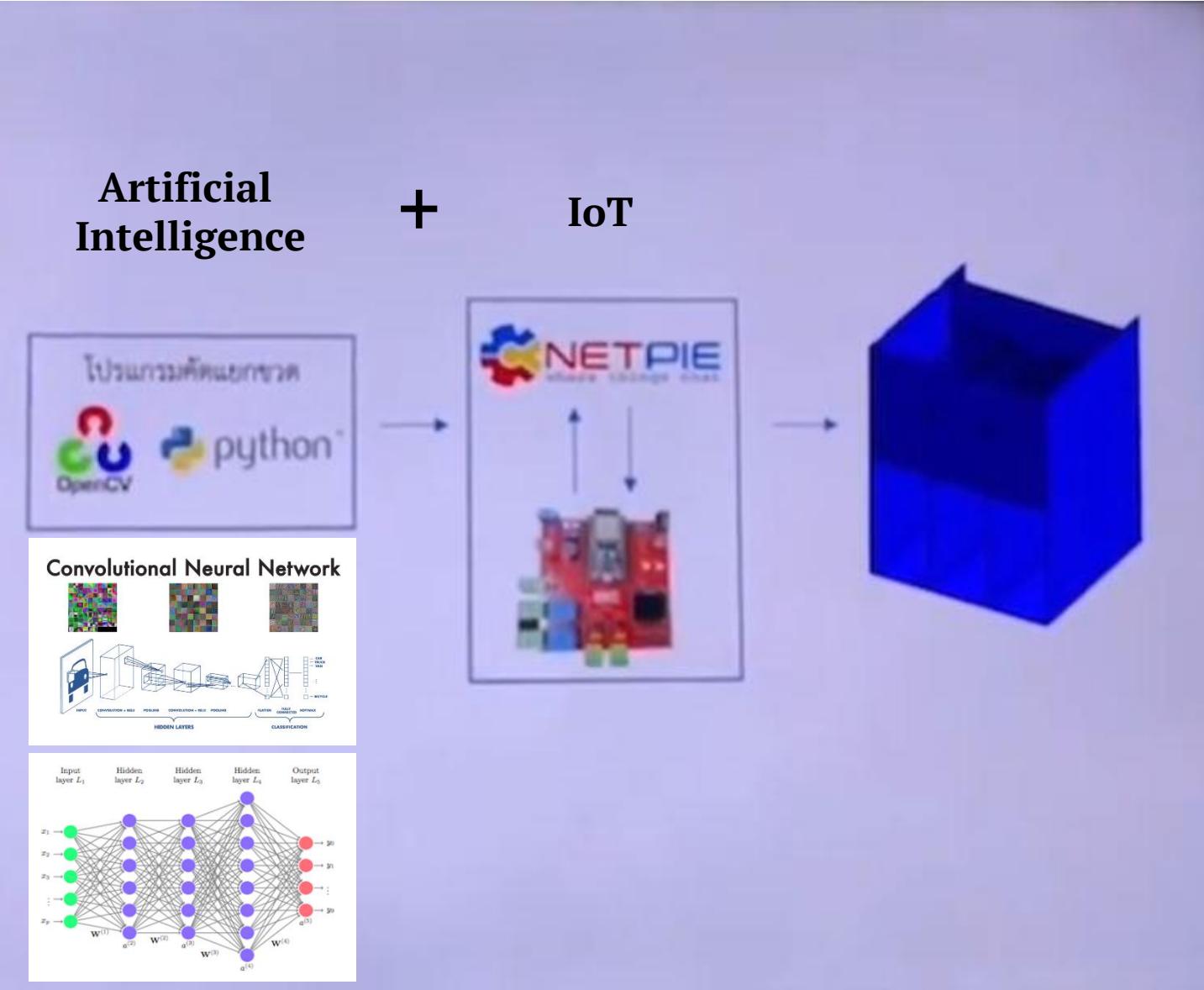
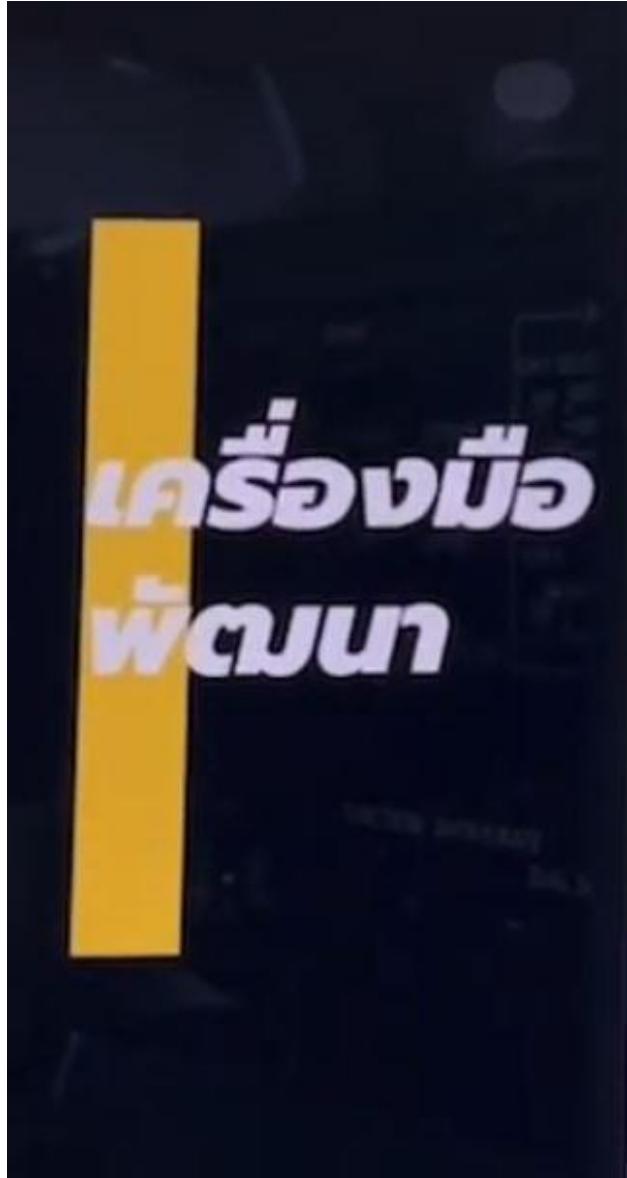
# PGT Automation Project:



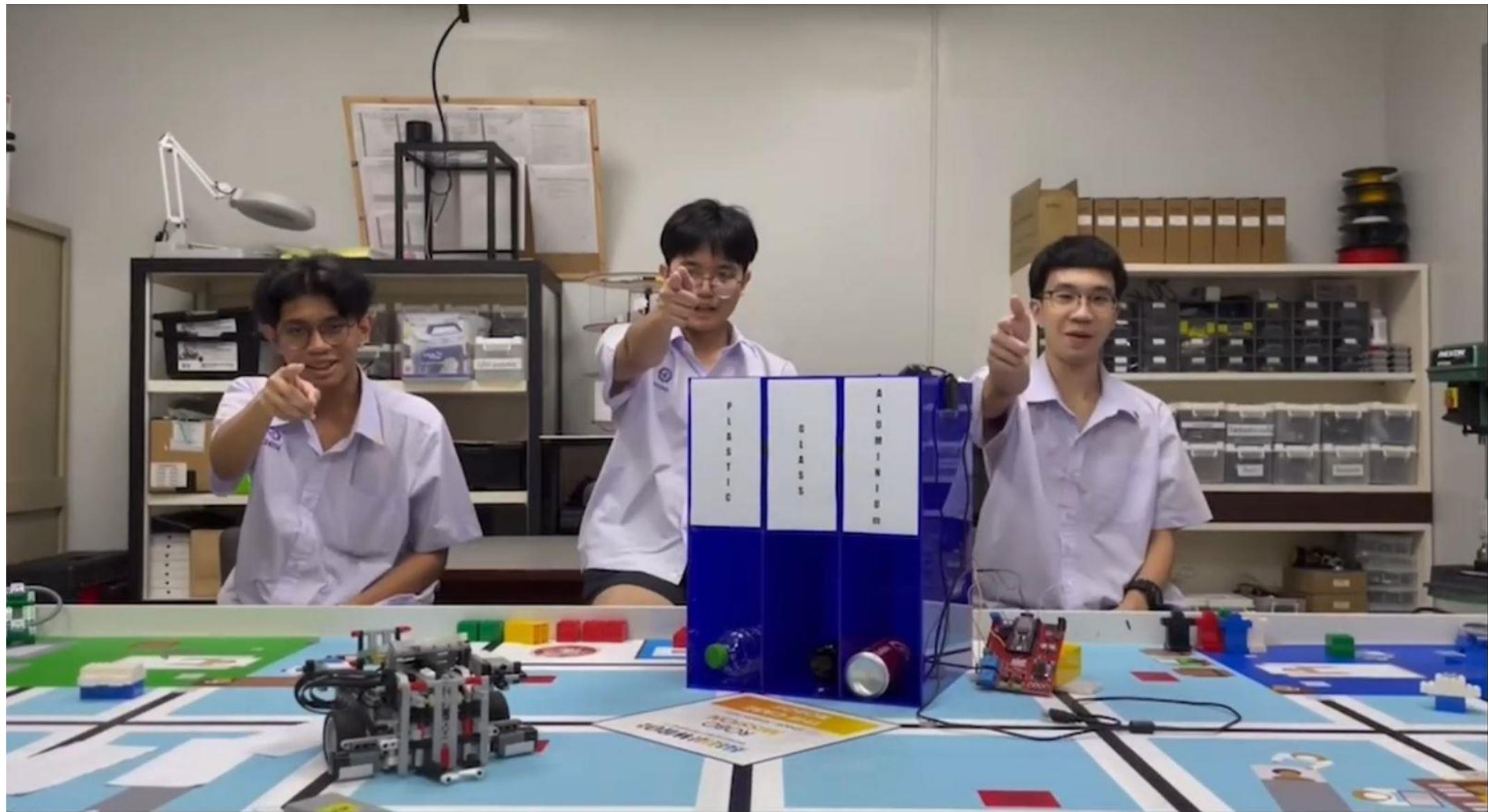
# PGT Automation Project:



# PGT Automation Project:



# PGT Automation Project:



ชื่อโครงการ (Project Charter):

## เครื่องคัดแยกขยะรีไซเคิล (อัจฉริยะ)

พัฒนาโดย ...

- นาย นรุตม์ กังสัมฤทธิ์
- นาย ธีรเมธ ตันเลง
- นาย วีรศ สถิตมั่น

โรงเรียนสาธิตมหาวิทยาลัยศรีนครินทรวิโรฒ ฝ่ายมัธยม

## ขอบเขตของโครงการ (Scope of Project):

โครงการระบบแยกขยะรีไซเคิลอัจฉริยะแบบผสมผสาน **AIoT** ถูกพัฒนาขึ้นโดยใช้การเรียนรู้ของคอมพิวเตอร์ในเชิงลึก (**Deep Learning/CNN**) ผสมผสานกับการนำเอาเทคโนโลยี **IoT** มาพัฒนาและประยุกต์เพิ่มเติม เพื่อให้ระบบแยกแยะรีไซเคิล มีความฉลาดและสามารถแยกแยะรีไซเคิล 3 ชนิด (ที่กำหนด) โดยอัตโนมัติได้อย่างประสิทธิภาพและแม่นยำสูงถึง 95% โดยไม่ต้องอาศัยมนุษย์ในการแยกแยะ

ระบบจะสามารถแยกแยะได้ดังนี้:

1. พลาสติก **Plastic** (ขวดพลาสติก หรือบรรจุภัณฑ์ใส่ของพลาสติก)
2. แก้ว **Glass** (แก้ว ขวดแก้ว หรือ ผลิตภัณฑ์บรรจุแบบแก้ว)
3. กระป๋องอลูมิเนียม **Aluminum Can**

**AIoT:** บัญญาประดิษฐ์ *Artificial Intelligence + Internet Of Things*

**Deep Learning:** เทคโนโลยีการเรียนรู้เชิงลึก (เป็นส่วนหนึ่งของ **Machine Learning**)

**CNN:** *Convolutional Neural Network Model* (เป็นรูปแบบการเรียนรู้ของ **Deep Learning** ประเภทหนึ่ง)

# PGT Automation Project:

## วัตถุประสงค์ของโครงการ (Objectives):

เพื่อพัฒนาระบบแยกขยายชีวิตรีไซเคิลอัจฉริยะแบบผสมผสาน AIoT เพื่อให้ระบบมีความชาญฉลาดและสามารถแยกขยายชีวิตรีไซเคิล 3 ชนิด (ที่กำหนด) โดยอัตโนมัติได้อย่างประสิทธิภาพและแม่นยำสูงถึง 95% โดยไม่ต้องอาศัยมนุษย์ในการแยกและ

## ผลลัพธ์ที่คาดหวัง (Expected Benefits):

ระบบแยกขยายชีวิตรีไซเคิลอัจฉริยะแบบผสมผสาน AIoT สามารถแยกขยายชีวิตรีไซเคิล 3 ชนิด (ที่กำหนด) โดยอัตโนมัติได้อย่างประสิทธิภาพและแม่นยำสูงอย่างน้อย 95% และสามารถนำไปเป็นต้นแบบในการประยุกต์การเรียนรู้การแยกขยายชีวิตรีไซเคิลที่หลากหลายชนิด (มากกว่า 3 ชนิด) เพื่อให้สามารถนำไปใช้งานในอุตสาหกรรมการบริหารจัดการขยะได้จริง

## แนวทางการดำเนินงานโครงการ (Approach):

1. ค้นหาแนวทางในการพัฒนาโครงการ ตลอดจนเทคโนโลยี อุปกรณ์และเครื่องมือที่ต้องใช้ (*Python, CNN and models, Development Platform – Cloud/local computer, web cam, computer, IoT and Arduino, devices and connections, etc.*)
2. แหล่งข้อมูลภาพ และการเพิ่มข้อมูลด้วย **Data Augmentation (rotate, zoom, flip, etc.)**
3. จัดเตรียมข้อมูลภาพ สำหรับการ **train, validate และ test CNN models** ในการแยกขยะรีไซเคิลแบบอัจฉริยะ
4. การอ่านข้อมูลภาพ (**load image data**)
5. การพัฒนา **CNN model** แยกแยกขยะรีไซเคิล 3 ชนิด ด้วยภาษา **Python** และการเลือก **model** ที่มีอยู่ มาทำการ **train, validate และ test (Transfer Learning)**
6. การนำ **model** ไปใช้งาน (แปลง **model** เป็นโมเดลไฟล์ “.h5” และ/หรือ “.TFLite” (**TensorFlow Lite**))
7. การพัฒนา **IoT application (Netpie)** เพื่อต่อเชื่อมกับระบบ **CNN model** ด้วยภาษา **Python**
8. การปรับปรุงระบบ **hardware** เพื่อให้มีโครงสร้างที่แข็งแรงมากขึ้นเพื่อรับรองการใช้งานแยกขยะรีไซเคิล

# PGT Automation Project:

## 1. แนวทางการพัฒนาโครงการ:

จากแนวความคิดจนนำไปสู่ต้นแบบ

- การพัฒนา **CNN model** ใช้ **Google Colab environment** (เพราะประสิทธิภาพ CPU, GPU สูง และทำให้การ **training model** ใช้ระยะเวลาสั้นลงมากกว่าบน **local computer**)
- การนำ **CNN model** ที่ **train** แล้ว มาใช้งานและพัฒนาต่อเชื่อมกับ **IoT application** บน **local computer** เพื่อให้ **response time** ดีกว่า และใช้งานได้จริง

สามารถถูกได้จาก วิดีโอระบบตันแบบการแยกขยะอัจฉริยะ ประกอบในโครงการ

## 2. แหล่งข้อมูลภาพ และการเพิ่มข้อมูลด้วย Data Augmentation :

Kaggle.com, Image-net.org, ics.uci.edu และ website ต่างๆ

จำนวนภาพ

	AluCan	Glass	Plastic
Train set	894	1,110	1,273
Validate set	227	277	319

# PGT Automation Project:

## 3. จัดเตรียมข้อมูล (ภาพ):

**Train set (80%) , validate set (20%) และ test set (optional)**



AluCan\_0053.jpg



AluCan\_0054.jpg



AluCan\_0055.jpg



AluCan\_0061.jpg



AluCan\_0062.jpg



AluCan\_0063.jpg



AluCan\_0069.jpg



AluCan\_0070.jpg



AluCan\_0071.jpg



AluCan\_0077.jpg



AluCan\_0078.jpg



AluCan\_0079.jpg



Glass\_0006.jpg



Glass\_0007.jpg



Glass\_0008.jpg



Glass\_0014.jpg



Glass\_0015.jpg



Glass\_0016.jpg



Glass\_0022.jpg



Glass\_0023.jpg



Glass\_0024.jpg



Glass\_0030.jpg



Glass\_0031.jpg



Glass\_0032.jpg



Plastic\_0001.jpg



Plastic\_0009.jpg



Plastic\_0017.jpg



Plastic\_0025.jpg



Plastic\_0002.jpg



Plastic\_0010.jpg



Plastic\_0018.jpg



Plastic\_0026.jpg



Plastic\_0003.jpg



Plastic\_0011.jpg



Plastic\_0019.jpg



Plastic\_0027.jpg

# PGT Automation Project:

## 4. การอ่านภาพข้อมูล:

- นำข้อมูลมาจัดเตรียม แยกโฟล์เดอร์เป็น **train\_set**, **validate\_set**, (**test\_set** เป็น optional)
- Upload** ข้อมูลไว้ที่ **Google Drive** และทำการเขียนโปรแกรมเพื่อ **import** ข้อมูลเข้าไว้ใน **Colab environment**

The screenshot shows a Google Colab notebook titled "PGT\_Classify\_v1.3.ipynb". The left sidebar displays a file tree with directories like "drive", "sample\_data", "test\_set", "train\_set" (which contains "AluCan", "Glass", "Plastic"), "val\_set" (which contains "AluCan", "Glass", "Plastic"), and "best\_model.h5". The main workspace contains the following code:

```
[3] from google.colab import drive  
drive.mount('/content/drive')  
  
[4] #Import the libraries  
import zipfile  
import os  
  
zip_ref = zipfile.ZipFile('/content/drive/MyDrive/MyData/wastedata.zip', 'r') #Opens the zip file in read mode  
zip_ref.extractall('/content') #Extracts the files into the /tmp folder  
zip_ref.close()
```

The status bar at the bottom indicates "Disk 68.24 GB available", "38s completed at 11:52 PM", and a green progress bar.

# PGT Automation Project:

## 5. การพัฒนา CNN model

- เขียนโปรแกรม และนำ CNN model “MobileNet\_v2” มาทำการ train กับ data ที่เตรียมไว้เพิ่มเติม เพื่อให้มีความแม่นยำสูงโดยไม่ต้องใช้ข้อมูลจำนวนมาก ๆ และใช้ระยะเวลาไม่มาก

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** PGT\_Classify\_v1.3.ipynb
- File Tree:** The left sidebar shows a file tree with directories: .., drive, sample\_data, test\_set, train\_set (containing AluCan, Glass, Plastic), val\_set (containing AluCan, Glass, Plastic), and best\_model.h5.
- Code Cells:** Two code cells are visible:
  - [5] Import statements for numpy, matplotlib.pyplot, ImageDataGenerator, image, MobileNetV2, preprocess\_input, decode\_predictions, Sequential, Dense, Flatten, Dropout, and EarlyStopping, ModelCheckpoint from tensorflow.keras modules.
  - [6] Code to set target\_img\_shape to (128, 128), define train\_dir as './train\_set', val\_dir as './val\_set', and create a train\_datagen object using ImageDataGenerator with preprocess\_function=preprocess\_input.
- Runtime Status:** RAM Disk usage is shown as 0s. The status bar at the bottom indicates the cell completed at 11:52 PM.

หมายเหตุ: การสร้าง CNN model ใหม่ และทำการ train model ให้มีความแม่นยำสูง ต้องใช้ข้อมูลจำนวนมาก เช่น 30,000-50,000 ภาพขึ้นไป และใช้ระยะเวลา train และปรับปรุงนานมาก

# PGT Automation Project:

## 5. การพัฒนา CNN model

- Show ข้อมูล Label มี 3 class: AluCan, Plastic, Glass

```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[6] target_img_shape=(128, 128)

train_dir = './train_set'
val_dir = './val_set'

train_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_set = train_datagen.flow_from_directory(train_dir,
                                              target_size=target_img_shape,
                                              batch_size=32,
                                              class_mode='sparse')

val_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

val_set = val_datagen.flow_from_directory(val_dir,
                                              target_size=target_img_shape,
                                              batch_size=32,
                                              class_mode='sparse')

Found 3277 images belonging to 3 classes.
Found 823 images belonging to 3 classes.

[7] print(train_set.class_indices)
```

RAM Disk Editing

Disk 68.24 GB available

38s completed at 11:52 PM

# PGT Automation Project:

## 5. การพัฒนา CNN model

- Show ข้อมูล Label มี 3 class: AluCan, Plastic, Glass เพื่อตรวจสอบข้อมูลก่อนการ train

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a file browser showing a directory structure with 'train\_set' and 'val\_set' containing 'AluCan', 'Glass', and 'Plastic' subfolders, along with a 'best\_model.h5' file. The main area contains a code cell [7] with the following Python code:

```
[7] for image_batch, labels_batch in train_set:  
    print(image_batch.shape)  
    print(labels_batch.shape)  
  
    img = image_batch[0] - image_batch[0].min()  
    img /= 2.0  
  
    print(img.min(), img.max())  
    plt.imshow(img)  
  
    print('class:', labels_batch[0])  
    break
```

The output of the code cell is:

```
{'AluCan': 0, 'Glass': 1, 'Plastic': 2}  
(32, 128, 128, 3)  
(32,)  
0.0 0.99607843  
class: 2.0
```

Below the output, there is a small image plot showing a green textured background with a white plastic bottle in the center. The plot has axes ranging from 0 to 120.

At the bottom of the screen, a status bar shows "Disk 68.24 GB available", a progress bar, and the text "38s completed at 11:52 PM".

# PGT Automation Project:

## 5. การพัฒนา CNN model

- Show ข้อมูล Label มี 3 class: AluCan, Plastic, Glass เพื่อตรวจสอบข้อมูลก่อนการ train

The screenshot shows a Jupyter Notebook interface with a sidebar labeled "Files". The "train\_set" folder contains subfolders "AluCan", "Glass", and "Plastic", and a file "best\_model.h5". The main area displays the following Python code:

```
[8] ids, counts = np.unique(train_set.classes, return_counts=True)
print(ids)
print(counts)

[9] labels = (train_set.class_indices)
labels = dict((v,k) for k,v in labels.items())
labels

for i in ids:
    print('{:>8} = {}'.format(labels[i], counts[i]))

    AluCan = 894
    Glass = 1110
    Plastic = 1273

[10] label_names = [k for k in train_set.class_indices]
label_names

['AluCan', 'Glass', 'Plastic']

[11] import pandas as pd

df_train_labels = pd.DataFrame({'Label':label_names, 'Count':counts})
df_train_labels.set_index('Label', inplace=True)
df_train_labels
```

The code prints the unique labels and their counts from the training set. The output shows three classes: AluCan (894), Glass (1110), and Plastic (1273). The final cell imports pandas and creates a DataFrame for training labels.

# PGT Automation Project:

## 5. การพัฒนา CNN model

- Show ข้อมูล Label มี 3 class: AluCan, Plastic, Glass เพื่อตรวจสอบข้อมูลก่อนการ train

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a file browser titled 'Files' showing a directory structure with 'train\_set' and 'val\_set' containing 'AluCan', 'Glass', and 'Plastic' subfolders. A file 'best\_model.h5' is also listed. The main area contains two code cells and a plot.

**Code Cell 11:**

```
[11] df_train_labels = pd.DataFrame({'Label':label_names, 'Count':counts})  
df_train_labels.set_index('Label', inplace=True)  
df_train_labels
```

**Output 11:**

Label	Count
AluCan	894
Glass	1110
Plastic	1273

**Code Cell 12:**

```
[12] df_train_labels.plot.bar()  
plt.show()
```

**Output 12:**

A bar chart titled 'df\_train\_labels.plot.bar()' showing the count of each label. The x-axis is labeled 'Label' and has three categories: 'AluCan', 'Glass', and 'Plastic'. The y-axis is labeled 'Count' and ranges from 0 to 1200. The bars are blue.

Label	Count
AluCan	894
Glass	1110
Plastic	1273

At the bottom of the notebook, it says '38s completed at 11:52 PM'.

## 5. การพัฒนา CNN model

- การสร้าง model โดยการนำ MobileNet\_v2 model มาทำการ train เพิ่ม (Transfer Learning)

The screenshot shows a Jupyter Notebook environment with a dark theme. On the left is a file browser titled 'Files' showing a directory structure with folders like 'drive', 'sample\_data', 'test\_set', 'train\_set' (containing 'AluCan', 'Glass', 'Plastic' subfolders), 'val\_set' (containing 'AluCan', 'Glass', 'Plastic' subfolders), and a file 'best\_model.h5'. The main area contains a code cell with the following Python code:

```
[13] ids, counts = np.unique(val_set.classes, return_counts=True)
      print(ids)
      print(counts)
      [0 1 2]
      [227 277 319]

[14] _, train_count = np.unique(train_set.classes, return_counts=True)
      _, val_count = np.unique(val_set.classes, return_counts=True)

      print('Ratio Validation/Training set:', val_count/train_count * 100)
      Ratio Validation/Training set: [25.39149888 24.95495495 25.05891595]

[15] in_shape = (target_img_shape[0], target_img_shape[1], 3)

      base_model = MobileNetV2(include_top=False, weights='imagenet',
                               input_shape=in_shape)

      model = Sequential()

      model.add(base_model)
      model.add(Flatten())
      model.add(Dense(256, activation='relu'))
      model.add(Dropout(0.4))
      model.add(Dense(128, activation='relu'))
      model.add(Dropout(0.4))
      model.add(Dense(3, activation='softmax'))
```

The code cell has three execution steps: Step 13 takes 0s, Step 14 takes 0s, and Step 15 takes 2s. The status bar at the bottom indicates the total time was 38s and the process completed at 11:52 PM.

# PGT Automation Project:

## 5. การพัฒนา CNN model

- การสร้าง model โดยการนำ MobileNet\_v2 model มาทำการ train เพิ่ม (Transfer Learning)

The screenshot shows a terminal window with a dark theme. On the left is a file browser pane titled "Files" showing a directory structure under "{x}": .., drive, sample\_data, test\_set, train\_set (containing AluCan, Glass, Plastic), val\_set (containing AluCan, Glass, Plastic), and best\_model.h5. The main pane contains Python code for building a model:

```
[15] in_shape = (target_img_shape[0], target_img_shape[1], 3)

base_model = MobileNetV2(include_top=False, weights='imagenet',
                         input_shape=in_shape)

model = Sequential()

model.add(base_model)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(3, activation='softmax'))

model.summary()

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_orderi9412608/9406464 [=====] - 0s 0us/step
9420800/9406464 [=====] - 0s 0us/step
Model: "sequential"
```

Below the code, a table shows the model's layers and their characteristics:

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_128 (Functional)	(None, 4, 4, 1280)	2257984
flatten (Flatten)	(None, 20480)	0
dense (Dense)	(None, 256)	5243136

At the bottom, it says "38s completed at 11:52 PM".

# PGT Automation Project:

## 5. การพัฒนา CNN model

- การสร้าง model โดยการนำ MobileNet\_v2 model มาทำการ train เพิ่ม (Transfer Learning)

The screenshot shows a Jupyter Notebook environment with a dark theme. On the left is a file browser titled 'Files' showing a directory structure with folders like 'drive', 'sample\_data', 'test\_set', 'train\_set', 'val\_set', and a file 'best\_model.h5'. The main area contains Python code in a code cell:

```
[16] base_model.trainable = False
[17] print("Trainable..\n---")
      for variable in model.trainable_variables:
          print(variable.name)

Trainable..
---
dense/kernel:0
dense/bias:0
dense_1/kernel:0
dense_1/bias:0
dense_2/kernel:0
dense_2/bias:0

[18] model.summary()

Model: "sequential"
-----
```

Layer (type)	Output Shape	Param #
mobilnetv2_1.00_128 (Functional)	(None, 4, 4, 1280)	2257984
flatten (Flatten)	(None, 20480)	0
dense (Dense)	(None, 256)	5243136
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
dropout_1 (Dropout)	(None, 128)	0

At the bottom of the code cell, it says '6s completed at 12:53 AM'.

# PGT Automation Project:

## 5. การพัฒนา CNN model

- การสร้าง model โดยการนำ MobileNet\_v2 model มาทำการ train เพิ่ม (Transfer Learning)

The screenshot shows a Jupyter Notebook environment with a dark theme. On the left is a file browser pane titled 'Files' showing a directory structure with folders like 'drive', 'sample\_data', 'test\_set', 'train\_set', 'val\_set', and a file 'best\_model.h5'. The main pane contains Python code for training a model using the MobileNet\_v2 architecture. The code imports time, defines the model compilation, sets up callbacks for early stopping and saving the best model, fits the model to the training set with validation data, and prints the total time taken. Below the code, the notebook displays the output of the training process, showing epochs from 1 to 5/20, loss values, accuracy values, and validation metrics. The status bar at the bottom indicates the execution took 6 seconds and completed at 12:53 AM.

```
import time

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
               metrics=['accuracy'])

es = EarlyStopping(monitor='val_loss', verbose=1, patience=5) #
mc = ModelCheckpoint('best_model.h5', monitor='val_accuracy', verbose=1, save_best_only=True)

start = time.time()
history = model.fit(train_set,
                     validation_data=val_set,
                     epochs=20, verbose=1, callbacks=[es, mc])

end = time.time()
print("Time Taken: {:.2f} minutes".format((end - start)/60))

Epoch 1/20
103/103 [=====] - ETA: 0s - loss: 1.4117 - accuracy: 0.6955
Epoch 1: val_accuracy improved from -inf to 0.90158, saving model to best_model.h5
103/103 [=====] - 77s 692ms/step - loss: 1.4117 - accuracy: 0.6955 - val_loss: 0.2818 - val_accuracy: 0.9
Epoch 2/20
103/103 [=====] - ETA: 0s - loss: 0.4199 - accuracy: 0.8572
Epoch 2: val_accuracy improved from 0.90158 to 0.92953, saving model to best_model.h5
103/103 [=====] - 66s 641ms/step - loss: 0.4199 - accuracy: 0.8572 - val_loss: 0.2023 - val_accuracy: 0.9
Epoch 3/20
103/103 [=====] - ETA: 0s - loss: 0.3091 - accuracy: 0.8853
Epoch 3: val_accuracy improved from 0.92953 to 0.94168, saving model to best_model.h5
103/103 [=====] - 66s 632ms/step - loss: 0.3091 - accuracy: 0.8853 - val_loss: 0.1952 - val_accuracy: 0.9
Epoch 4/20
103/103 [=====] - ETA: 0s - loss: 0.2412 - accuracy: 0.9228
Epoch 4: val_accuracy did not improve from 0.94168
103/103 [=====] - 66s 639ms/step - loss: 0.2412 - accuracy: 0.9228 - val_loss: 0.1849 - val_accuracy: 0.9
Epoch 5/20
```

6s completed at 12:53 AM

# PGT Automation Project:

## 5. การพัฒนา CNN model

- การสร้าง model โดยการนำ MobileNet\_v2 model มาทำการ train เพิ่ม (Transfer Learning)

The screenshot shows a terminal window with a dark theme. On the left is a file browser pane titled 'Files' showing a directory structure with 'drive', 'sample\_data', 'test\_set', 'train\_set', 'val\_set', and 'best\_model.h5'. The main pane displays the command-line output of a training session. The output shows the progress of 103 epochs, with metrics like loss, accuracy, and validation loss/accuracy. It indicates several epochs where validation accuracy did not improve, leading to early stopping after epoch 11. The total time taken for the training is 14.03 minutes.

```
[19] Epoch 4: val_accuracy did not improve from 0.94168
103/103 [=====] - ETA: 0s - loss: 0.2412 - accuracy: 0.9228 - val_loss: 0.1849 - val_accuracy: 0.9228
Epoch 5/20
103/103 [=====] - ETA: 0s - loss: 0.2124 - accuracy: 0.9292
Epoch 5: val_accuracy did not improve from 0.94168
103/103 [=====] - ETA: 0s - loss: 0.2124 - accuracy: 0.9292 - val_loss: 0.1908 - val_accuracy: 0.9292
Epoch 6/20
103/103 [=====] - ETA: 0s - loss: 0.1961 - accuracy: 0.9393
Epoch 6: val_accuracy improved from 0.94168 to 0.95018, saving model to best_model.h5
103/103 [=====] - 70s 681ms/step - loss: 0.1961 - accuracy: 0.9393 - val_loss: 0.1464 - val_accuracy: 0.9393
Epoch 7/20
103/103 [=====] - ETA: 0s - loss: 0.1381 - accuracy: 0.9518
Epoch 7: val_accuracy improved from 0.95018 to 0.95626, saving model to best_model.h5
103/103 [=====] - 67s 653ms/step - loss: 0.1381 - accuracy: 0.9518 - val_loss: 0.1496 - val_accuracy: 0.9518
Epoch 8/20
103/103 [=====] - ETA: 0s - loss: 0.1107 - accuracy: 0.9597
Epoch 8: val_accuracy did not improve from 0.95626
103/103 [=====] - 71s 683ms/step - loss: 0.1107 - accuracy: 0.9597 - val_loss: 0.1617 - val_accuracy: 0.9597
Epoch 9/20
103/103 [=====] - ETA: 0s - loss: 0.1088 - accuracy: 0.9637
Epoch 9: val_accuracy did not improve from 0.95626
103/103 [=====] - 71s 684ms/step - loss: 0.1088 - accuracy: 0.9637 - val_loss: 0.1674 - val_accuracy: 0.9637
Epoch 10/20
103/103 [=====] - ETA: 0s - loss: 0.0869 - accuracy: 0.9710
Epoch 10: val_accuracy improved from 0.95626 to 0.96233, saving model to best_model.h5
103/103 [=====] - 69s 673ms/step - loss: 0.0869 - accuracy: 0.9710 - val_loss: 0.1551 - val_accuracy: 0.9710
Epoch 11/20
103/103 [=====] - ETA: 0s - loss: 0.0828 - accuracy: 0.9716
Epoch 11: val_accuracy did not improve from 0.96233
103/103 [=====] - 73s 710ms/step - loss: 0.0828 - accuracy: 0.9716 - val_loss: 0.1487 - val_accuracy: 0.9716
Epoch 11: early stopping
Time Taken: 14.03 minutes
```

# PGT Automation Project:

## 5. การพัฒนา CNN model

- กราฟแสดงความแม่นยำของ model ในการทำนายแยกและ เมื่อทำการ train แล้ว



# PGT Automation Project:

## 5. การพัฒนา CNN model

- เขียนโค้ดเพิ่มสำหรับการทำนาย predict

The screenshot shows a Jupyter Notebook environment with a sidebar labeled "Files" containing directory structures for "drive", "sample\_data", "test\_set", "train\_set", and "val\_set", along with a file named "best\_model.h5". The main area displays Python code for model evaluation and prediction.

```
[22] print(len(val_set))
acc = model.evaluate(val_set, verbose=1)
print('score = {:.3f}'.format(acc[1]))

26
26/26 [=====] - 13s 503ms/step - loss: 0.1487 - accuracy: 0.9453
score = 0.945

Predict

[23] from tensorflow.keras.preprocessing.image import load_img, img_to_array

def predict(img_fname):
    img = load_img(img_fname, target_size=target_img_shape)
    plt.imshow(img)
    img = img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)

    pred = model.predict(img)
    pred_cls = labels[np.argmax(pred, -1)[0]]
    print('Prediction:', pred_cls, pred[0].round(3))
    return(pred_cls, pred)

[24] labels = (train_set.class_indices)
labels = dict((v,k) for k,v in labels.items())
labels

{0: 'AluCan', 1: 'Glass', 2: 'Plastic'}
```

RAM Disk

6s completed at 12:53 AM

# PGT Automation Project:

## 5. การพัฒนา CNN model

- ทดสอบผลการทำงาน (แม่นยำ) ทีละภาพ

```
+ Code + Text  
2s [25] _,pred_result = predict('./val_set/AluCan/AluCan_9004.jpg')  
labels[np.argmax(pred_result, -1)[0]]  
  
Prediction: AluCan [0.629 0.139 0.232]  
'AluCan'  
  
0 20 40 60 80 100 120  
0 20 40 60 80 100 120  
  
[26] _,pred_result = predict('./val_set/Glass/Glass_9002.jpg')  
labels[np.argmax(pred_result, -1)[0]]  
  
Prediction: Glass [0.092 0.872 0.036]  
'Glass'  
  
0 20 40 60  
0 20 40 60  
  
Disk 68.24 GB available  
6s completed at 12:53 AM
```

### 5. การพัฒนา CNN model

- เขียนโค้ดเพิ่มสำหรับการทำนาย predict ใน test\_set (จำนวนหลายภาพ)

The screenshot shows a Jupyter Notebook environment with a dark theme. On the left is a file browser titled 'Files' showing a directory structure with folders like 'drive', 'sample\_data', 'test\_set', 'train\_set', 'val\_set', and a file 'best\_model.h5'. The main area has two code cells and their outputs.

**Code Cell 1:**

```
[27] pred_result = predict('./test_set/0011.jpg')
labels[np.argmax(pred_result, -1)[0]]
```

Prediction: Glass [0. 1. 0.]  
'Glass'

**Code Cell 2:**

```
[28] import glob
```

```
mytestlist = [f for f in glob.glob('./test_set/*')]
for testimg in mytestlist:
    index = mytestlist.index(testimg)
    mypredlist,pred_result = predict(mytestlist[index])
    print(labels[np.argmax(pred_result, -1)[0]])
    plt.show()
```

6s completed at 12:53 AM

At the bottom, there are status indicators for RAM and Disk usage, along with other standard notebook controls.

### 5. การพัฒนา CNN model

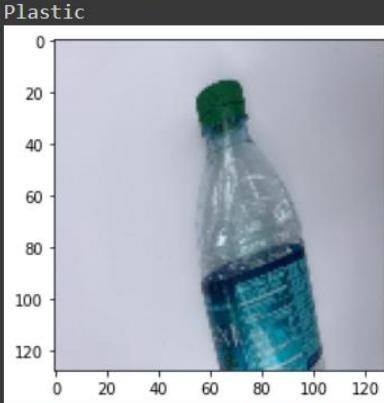
- ทดสอบผลการทำงาน (ความแม่นยำ 95%)

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a file browser pane titled 'Files' showing a directory structure with folders like 'drive', 'sample\_data', 'test\_set', 'train\_set', 'val\_set', and a file 'best\_model.h5'. The main area contains a code cell and a results section.

**Code Cell:**

```
for testing in mytestlist:  
    index = mytestlist.index(testing)  
    mypredlist,pred_result = predict(mystestlist[index])  
    print(labels[np.argmax(pred_result, -1)[0]])  
    plt.show()
```

**Results:**

Prediction: Plastic [0. 0. 1.]  
Plastic  
  
Prediction: Glass [0.005 0.965 0.03 ]  
Glass  


RAM: 68.24 GB available

6s completed at 12:53 AM

# PGT Automation Project:

## 6. การนำ model ไปใช้งาน

- “mypgt.h5” file

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a file browser titled "Files" showing a directory structure under "{x}":

- ..
- drive
- mypgtmodel
- sample\_data
- test\_set
- train\_set
  - AluCan
    - Glass
    - Plastic
  - val\_set
    - AluCan
    - Glass
    - Plastic
- best\_model.h5
- pgt\_model.h5

The main area contains several code cells:

- Save Model in .h5 for Use**

```
[30] model.save('./pgt_model.h5')
```
- Save Model in folder for Use**

```
[31] model.save('./mypgtmodel')
```

WARNING:absl:Function `wrapped\_model` contains input name(s) mobilenetv2\_1.00\_128\_input with unsupported characters which will be INFO:tensorflow:Assets written to: ./mypgtmodel/assets  
INFO:tensorflow:Assets written to: ./mypgtmodel/assets
- Save class config to decode prediction result to class name (Plastic, Glass, Alucan)**

```
[32] import pickle  
file_name = 'classname.pkl'  
open_file = open(file_name, "wb")  
pickle.dump(labels, open_file)  
open_file.close()
```
- Load h5 for Use**

```
from tensorflow.keras.models import load_model # lib load model  
my_model = load_model('./pgt_model.h5') # load model h5
```

At the bottom, a progress bar indicates the last cell completed at 1:07 AM.

# PGT Automation Project:

## 5. การพัฒนา CNN model

- เขียนโค้ดเพิ่มสำหรับการใช้ “mypgt.h5” เพื่อทำนาย predict

The screenshot shows a terminal window with a dark theme. On the left is a file explorer sidebar titled "Files" showing a directory structure with folders like "drive", "mypgtmodel", "sample\_data", "test\_set", "train\_set" (containing "AluCan", "Glass", "Plastic"), "val\_set" (containing "AluCan", "Glass", "Plastic"), and files "best\_model.h5", "classname.pkl", and "pgt\_model.h5". The main pane contains Python code for loading a model and performing predictions on an image from a camera device.

```
[35] import pickle
file_name = "classname.pkl"
open_file = open(file_name, "rb")
class_names = pickle.load(open_file)
open_file.close()
class_names
{0: 'AluCan', 1: 'Glass', 2: 'Plastic'}
```

```
from keras.models import load_model
from keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input, decode_predictions
import numpy as np
import matplotlib.pyplot as plt

# insert code to take a picture snapshot of image from camera device
# and save the image file and input the image file
input_image = './test_set/0074.jpg'
img = image.load_img(input_image, target_size=(128, 128))
plt.figure(figsize=(2, 2))
plt.imshow(img)

pred_image = image.img_to_array(img)
pred_image = np.expand_dims(pred_image, axis=0)
pred_image = preprocess_input(pred_image)

my_model = load_model('./pgt_model.h5') # load model h5
pred_result = my_model.predict(pred_image)
print(pred_result)
class_names[np.argmax(pred_result, -1)[0]]
```

RAM Disk completed at 1:08 AM

# PGT Automation Project:

## 5. การพัฒนา CNN model

- เขียนโค้ดเพิ่มสำหรับการใช้ “mypgt.h5” เพื่อทำนาย predict

The screenshot shows a Jupyter Notebook environment with a dark theme. On the left is a file browser pane titled "Files" showing a directory structure with folders like {x}, mypgtmodel, sample\_data, test\_set, train\_set, val\_set, and files best\_model.h5, classname.pkl, and pgt\_model.h5. The main pane contains two code cells and their outputs.

**Code Cell 36:**

```
[36] [[1.5285565e-08 1.0000000e+00 2.9671103e-09]]  
'Glass'  

```

**Code Cell 37:**

```
[37] from tensorflow.keras.models import load_model  
model = load_model('./pgt_model.h5') #  
  
model.summary()  
model.get_config()['layers'][0]
```

**Output of Cell 37:**

```
Model: "sequential"  
=====  
Layer (type) Output Shape Param #  
=====  
mobilennetv2_1.00_128 (Functional) (None, 4, 4, 1280) 2257984  
flatten (Flatten) (None, 20480) 0  
dense (Dense) (None, 256) 5243136  
dropout (Dropout) (None, 256) 0  
=====
```

At the bottom, a progress bar indicates "0s completed at 1:10 AM".

## 5. การพัฒนา CNN model

- เขียนโค้ดเพิ่มสำหรับการแปลง “.h5” เป็น ”.tflite” เป็นสำหรับทางเลือกในการนำไปใช้กับ IoT, Mobile, etc. เพราะมีขนาดไฟล์เล็กกว่า

The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer (Left):** Shows a directory structure:
  - ..
  - drive
  - mypgtmodel
  - sample\_data
  - test\_set
  - train\_set
    - AluCan
    - Glass
    - Plastic
  - val\_set
    - AluCan
    - Glass
  - Plastic
    - best\_model.h5
    - classname.pkl
    - pgt\_model.h5
- Code Cell (Top Right):** Displays the summary of a Keras model:

```
+ Code + Text  
0s 46s  
dense_1 (Dense) (None, 128) 32896  
dropout_1 (Dropout) (None, 128) 0  
dense_2 (Dense) (None, 3) 387  
=====  
Total params: 7,534,403  
Trainable params: 5,276,419  
Non-trainable params: 2,257,984
```
- Code Cell (Bottom Right):** Shows Python code for converting a Keras model to TensorFlow Lite and calculating its size.

```
import tensorflow as tf  
converter = tf.lite.TFLiteConverter.from_keras_model(model)  
tflite_pgt_model = converter.convert()  
  
f = open('pgt_model.tflite', "wb")  
f.write(tflite_pgt_model)  
f.close()  
  
regular_model_size = len(tflite_pgt_model) / 1024  
print('Regular model size = %d KBs.' % regular_model_size)
```

Output of the code cell:

```
WARNING:absl:Function `__wrapped_model` contains input name(s) mobilenetv2_1.00_128_input with unsupported characters which will be ignored.  
INFO:tensorflow:Assets written to: /tmp/tmpptzhriu_6/assets  
INFO:tensorflow:Assets written to: /tmp/tmpptzhriu_6/assets  
INFO:tensorflow:Assets written to: /tmp/tmpptzhriu_6/assets
```

Execution status: 46s completed at 1:12 AM.

## 5. การพัฒนา CNN model

- เขียนโค้ดเพิ่มสำหรับ optimize ".tflite" เพื่อบีบอัดให้ไฟล์มีขนาดเล็กลงไปอีก หมายเหตุ IoT, Mobile แต่อาจการทำงานอาจช้ากว่า ".h5" และ ".tflite" ทั่วไป

The screenshot shows a terminal window with the following content:

**Convert .h5 to TensorFlow Lite (Optimized)**

```
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_optimized_model = converter.convert()

optimized_model_size = len(tflite_optimized_model) / 1024
print('Optimized model size = %d KBs,' % optimized_model_size)
print('which is about %d%% of the regular model size.'\
      % (optimized_model_size * 100 / regular_model_size))

f = open('opt_pgt_model.tflite', "wb")
f.write(tflite_optimized_model)
f.close()
```

WARNING:absl:Function `\_\_wrapped\_model` contains input name(s) mobilenetv2\_1.00\_128\_input with unsupported characters which will be  
INFO:tensorflow:Assets written to: /tmp/tmp5flqluqb/assets  
INFO:tensorflow:Assets written to: /tmp/tmp5flqluqb/assets  
WARNING:absl:Buffer deduplication procedure will be skipped when flatbuffer library is not properly loaded  
Optimized model size = 7741 KBs,  
which is about 26% of the regular model size.

**Use TensorFlow Lite to Predict**

```
[1] import tensorflow as tf
from time import time

tflite_path = 'pgt_model.tflite'
```

Disk 68.06 GB available

40s completed at 1:13 AM

# PGT Automation Project:

## 5. การพัฒนา CNN model

- เขียนโค้ดเพิ่มสำหรับการใช้ “pgt\_model.tflite” เพื่อทำนาย predict

```
+ Code + Text
Use TensorFlow Lite to Predict
import tensorflow as tf
from time import time

tflite_path = 'pgt_model.tflite'

interpreter = tf.lite.Interpreter(model_path=tflite_path)
interpreter.allocate_tensors()

input = interpreter.get_input_details()
input_shape = input[0]['shape']
print('input shape:', input_shape)

input_tensor_index = input[0]["index"]
output = interpreter.tensor(interpreter.get_output_details()[0]["index"])

interpreter.set_tensor(input_tensor_index, pred_image) #

time_start = time()
interpreter.invoke()

time_end = time()
total_tflite_time = time_end - time_start
print("Total prediction time: ", total_tflite_time)

digit = np.argmax(output()[0])
#print(digit)
print(class_names[digit])
```

Disk 68.06 GB available

0s completed at 1:14 AM

## 5. การพัฒนา CNN model

- เขียนโค้ดเพิ่มสำหรับการใช้ “pgt\_model.tflite” เพื่อทำนาย predict

The screenshot shows a Jupyter Notebook environment with a dark theme. On the left is a file browser pane showing a directory structure with folders like 'drive', 'mypygtmodel', 'sample\_data', 'test\_set', 'train\_set' (containing 'AluCan', 'Glass', 'Plastic' subfolders), 'val\_set' (containing 'AluCan', 'Glass', 'Plastic' subfolders), and files 'best\_model.h5', 'classname.pkl', 'opt\_pgt\_model.tflite', 'pgt\_model.h5', and 'pgt\_model.tflite'. The main pane contains a code cell with the following Python script:

```
interpreter.set_tensor(interpreter.get_input_details()[0]['index'], pred_image) #  
time_start = time()  
interpreter.invoke()  
  
time_end = time()  
total_tflite_time = time_end - time_start  
print("Total prediction time: ", total_tflite_time)  
  
digit = np.argmax(output()[0])  
#print(digit)  
print(class_names[digit])  
  
plt.figure(figsize=(2, 2))  
plt.imshow(pred_image[0], cmap=plt.cm.gray_r)  
plt.title('Predicted:{}'.format(digit))  
plt.xticks([]); plt.yticks([])  
plt.show()
```

The output pane shows the results of the execution:

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Input shape: [ 1 128 128  3]  
Total prediction time:  0.010280609130859375  
Glass  
Predicted:1
```

Below the image, a thumbnail preview shows a dark bottle against a white background.

# PGT Automation Project:

## 6. การพัฒนา IoT application

- เขียนโค้ดเพิ่มและใช้ OpenCV (Image Processing Library) เพื่อต่อเชื่อมกับ web cam, IoT board, NETPIE และ CNN model
- Web cam ทำหน้าที่ถ่ายภาพวัตถุ (ขยะ) อย่างต่อเนื่อง หรือถ่ายภาพตามเวลาที่กำหนดทุก ๆ x วินาที หรือทุกครั้งเมื่อกด spacebar key
- นำภาพที่ได้เป็น input ส่งต่อให้กับ “mypgt.h5” model เพื่อคำนวณผล
- ส่งผลลัพธ์ที่ได้จากการคำนวณผ่านตัวแปรที่เชื่อมต่อผ่าน NETPIE และ IoT board เพื่อส่งคำสั่งให้ hardware ทำการแยกขยะลงในช่องที่กำหนด

```
1 import cv2
2 import os, time
3 from datetime import datetime
4
5 from keras import models
6 from PIL import Image
7 from keras.models import load_model
8 from keras.preprocessing import image
9 from tensorflow.keras.preprocessing import image
10 from tensorflow.keras.applications.mobilenet_v2 import preprocess_input, decode_predictions
11
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pickle
15
16 import microgear.client as client
17
18 def saveimgfile(frame):
19     #In Windows, ':' is invalid character for a file name.
20     filename = 'img' + datetime.now().strftime("%Y_%m_%d-%I-%M-%S_%p")
21     print('filename',filename)
22     if not cv2.imwrite(saveimg_path + "/" + filename + ".png",frame):
23         raise Exception("Could not write image")
24
25 def wastepredict_h5(frame):
26     # =====
27     # * * * Predict image class from captured image * * *
28     # 0: AluCan, 1: Glass, 2: Plastic
29     # =====
30     #Convert the captured frame (BGR) into RGB (convert a NumPy array to an image)
31     im = Image.fromarray(frame, 'RGB')
32
33     #Resizing into 128x128 because we trained the model with this image size.
34     im = im.resize((128,128))
35
36     #Changing dimension 128x128x3
37     img_array = np.array(im)
```

# PGT Automation Project:

## 6. การพัฒนา IoT application

- เขียนโค้ดการเชื่อมต่อ กับ NETPIE ผ่าน “MQTT” message protocol

```
75     interpreter = tf.lite.Interpreter(model_path=tflite_path)
76     interpreter.allocate_tensors()
77
78     input = interpreter.get_input_details()
79     input_shape = input[0]['shape']
80
81     input_tensor_index = input[0]["index"]
82     output = interpreter.tensor(interpreter.get_output_details()[0]["index"])
83
84     interpreter.set_tensor(input_tensor_index, img_array)
85
86     time_start = time.time()
87     interpreter.invoke()
88
89     time_end = time.time()
90     total_tflite_time = time_end - time_start
91     print("Total prediction time: ", total_tflite_time)
92
93     digit = np.argmax(output()[0])
94     #print(digit)
95     waste_type = class_names[digit]
96     print(waste_type)
97
98     return(waste_type)
99
100
101 def on_connect(client, userdata, flags, rc):
102     print("Connected with result code "+str(rc))
103     # Subscribing in on_connect() means that if we lose the connection and
104     # reconnect then subscriptions will be renewed.
105     client.subscribe("@msg/data")
106
107 def on_message(client, userdata, msg):
108     print(msg.topic+" "+str(msg.payload))
109     client_id = "aae1efd-41db-4773-9e14-2827bab6abf2"
110     token = "2U7u7Lycj5DC9Vub4oD627LaYSo3SsTt"
111     secret = "Aen$azB1LA4CZ$cgmapYy)Vb)JdbQSEJ"
```

## 6. การพัฒนา IoT application

- เขียนโค้ดนำข้อมูลภาพจาก web cam และจัดเตรียมข้อมูลสำหรับการทำนาย

```
39      #Our keras model used a 4D tensor, (images x height x width x channel)
40      #So changing dimension 128x128x3 into 1x128x128x3
41      img_array = np.expand_dims(img_array, axis=0)
42
43      #Prepare input array to predict
44      img_array = preprocess_input(img_array)
45
46      # OR we can use the following code section
47      pred_result = my_model.predict(img_array)
48      waste_type = class_names[np.argmax(pred_result, -1)[0]]
49      print(waste_type)
50      return(waste_type)
51
52  def wastepredict_tflite(frame):
53      # =====
54      # * * * Predict image class from captured image * * *
55      # Use TensorFlow Lite to Predict
56      # 0: AluCan, 1: Glass, 2: Plastic
57      # =====
58      #Convert the captured frame (BGR) into RGB (convert a NumPy array to an image)
59      im = Image.fromarray(frame, 'RGB')
60
61      #Resizing into 128x128 because we trained the model with this image size
62      im = im.resize((128,128))
63
64      #Changing dimension 128x128x3
65      img_array = np.array(im)
66
67      #Our keras model used a 4D tensor, (images x height x width x channel)
68      #So changing dimension 128x128x3 into 1x128x128x3
69      img_array = np.expand_dims(img_array, axis=0)
70
71      #Prepare input array to predict
72      img_array = preprocess_input(img_array)
73
74      #Calling the predict method on model to predict on the image
75      interpreter = tf.lite.Interpreter(model_path=tflite_path)
```

# PGT Automation Project:

## 6. การพัฒนา IoT application

- Load “mypgt.h5” (CNN model) สำหรับใช้ในการทำนายผลจากภาพที่ถ่ายมาจากการ cam

```
112     broker = "mqtt.netpie.io"
113     port = 1883
114     client = mqtt.Client(client_id)
115     client.username_pw_set(token, secret)
116     client.on_connect = on_connect
117     client.on_message = on_message
118     client.connect(broker, port)
119
120     def garbage_publishing(prediction):
121         if prediction == "Plastic":
122             client.publish("@msg/data","Plastic")
123         elif prediction == "Glass":
124             client.publish("@msg/data","Glass")
125         else:
126             client.publish("@msg/data","AluCan")
127
128     if __name__ == '__main__':
129         # Check current working directory.
130         # retval = os.getcwd()
131         # print("Current working directory: ",retval)
132
133         # Set output directory to save image snapshot if it does not exist
134         saveimg_path = 'd:/saveimage'
135         if not os.path.exists(saveimg_path):
136             os.makedirs(saveimg_path)
137
138         # Load class configuration (0:Alucan, 1:Glass, 2:Plastic)
139         file_name = "d:/IoT/classname.pkl"
140         open_file = open(file_name, "rb")
141         class_names = pickle.load(open_file)
142         open_file.close()
143         class_names
144         #print(class_names)
145
146         # Load pgt h5 model (to predict)
147         my_model = load_model('d:/IoT/pgt_model.h5') # load model h5
```

# PGT Automation Project:

## 6. การพัฒนา IoT application

- เขียนโค้ดเพิ่มวน loop เพื่อให้สามารถรับภาพข้อมูลจาก web cam ทุก x วินาทีโดยอัตโนมัติ หรือทุกครั้งที่กดปุ่ม spacebar หรือออกจากโปรแกรมเท่ากับกดปุ่ม “q”
- นำภาพที่ได้เป็น input ส่งต่อให้กับ “mypgt.h5” model เพื่อทำนายผล

```
149     # Save a picture snapshot manually or automaticall every defined period of time
150     sec = 0
151
152     ans = input ("Do you also want to save a picture? (y/n) ")
153     if ans.upper() == "Y":
154         sec = int(input("Save a picture at every ... seconds [0: every second - elapsed time] "))
155
156     # Open the device at the ID 0
157     #cap = cv2.VideoCapture(0) #Camera Channel 0
158     cap = cv2.VideoCapture(1) #Camera channel 1
159
160     # Check whether user selected camera is opened successfully.
161     if not (cap.isOpened()):
162         print("Could not open video device")
163
164     cap.set(3,640/2) #width=640/2
165     cap.set(4,480/2) #height=480/2
166
167     # Set up automatic timer to wait in seconds between each object prediction
168     wait_sec = 5
169     t1_time = datetime.now()
170     timeout_wait = True
171
172     while(True):
173         start_time = time.time() # for saved image file name purpose
174         # Capture frame-by-frame: turn video frame into numpy ndarray
175         ret, frame = cap.read()
176         # Display the resulting frame
177         cv2.imshow('preview',frame)
178
179         # Set up the waitKey (keyboard) for checking multiple times
180         keyboard = cv2.waitKey(1) & 0xFF
181
182         if ans.upper() == "Y": # save a picture manually
183             if sec == 0: # save a picture automatically by default
184                 end_time = time.time()
185                 time.sleep(1.0 - (end_time - start_time))
```

# PGT Automation Project:

## 6. การพัฒนา IoT application

- ส่งผลลัพธ์ที่ได้จากการทำงานของผ่านตัวแปรเชื่อมต่อผ่าน NETPIE และ IoT board เพื่อส่งคำสั่งให้ hardware ทำการแยกขยะลงในช่องที่กำหนด

```
185         time.sleep(1.0 - (end_time - start_time))
186         saveimgfile(frame)
187         #time.sleep(1.0 - time.time() + start_time) # Sleep for 1 second minus elapsed time
188     else: # save a picture automatically at every x seconds
189         time.sleep(sec - time.time() + start_time) # Sleep for x second minus elapsed time
190         saveimgfile(frame)
191     # Waits for a user input to save the image
192     elif keyboard == ord('y'):
193         saveimgfile(frame)
194
195     # Waits for a user input to quit the application
196     if keyboard == ord('q'):
197         break
198     if keyboard == ord('p'): #pause program
199         cont = input("Pause, do you want to continue (Y/N)? ")
200         if not (cont.upper() == "Y"):
201             break
202     elif keyboard == 32: # if space bar, then predicting object immediately without delay
203         timeout_wait = True
204
205     # Next improvement: add sensor to detect object existed,
206     # and execute the prediction only if the object exists (timeout_wait = True).
207
208     if timeout_wait: # space bar or wait for 4 seconds
209         # choose between h5 and tflite model
210         waste_category = wastepredict_h5(frame)          #large size predict model
211         #waste_category = wastepredict_tflite(frame)    #small size predict model
212
213         # waste_category can be used for further action (e.g., IoT application, robotic programming, etc.)
214         garbage_publishing(waste_category)
215         t1_time = datetime.now()
216         timeout_wait = False
217
218     # Calculate the wait time (seconds) after previously predicting the object
219     t2_time = datetime.now()
220     #print((t2_time - t1_time).total_seconds())
221     if ((t2_time - t1_time).total_seconds() > wait_sec):
```

# PGT Automation Project:

## 6. การพัฒนา IoT application

- ส่งผลลัพธ์ที่ได้จากการทำงานของผ่านตัวแปรเชื่อมต่อผ่าน **NETPIE** และ **IoT board** เพื่อส่งคำสั่งให้ **hardware** ทำการแยกขยะลงในช่องที่กำหนด

```
218     # Calculate the wait time (seconds) after previously predicting the object
219     t2_time = datetime.now()
220     #print((t2_time - t1_time).total_seconds())
221     if ((t2_time - t1_time).total_seconds() >= wait_sec):
222         t1_time = t2_time
223         timeout_wait = True
224         #print("wait %d seconds done " % wait_sec)
225
226     # When everything done, release the capture
227     cap.release()
228     cv2.destroyAllWindows()
```

## สรุปผลโครงการ (Summary):

- ระบบนี้สามารถทำงานได้อย่างแม่นยำตามวัตถุประสงค์ของโครงการ ความแม่นย้ำโดยเฉลี่ยประมาณ 95%
- ตำแหน่งกล้อง **web cam** และมุมแสงตาก และมุมแสงสะท้อนของวัตถุ อาจมีผลต่อการทำนายแยกแยะวัตถุผิดพลาดได้ในบางครั้ง หากมีการปรับตำแหน่งและมุมที่เหมาะสม การทำนายแยกแยะจะมีความแม่นยำสูงมาก
- การพัฒนาโครงการนี้ยังมีข้อจำกัดบางประการที่อธิบายรายละเอียดใน slide ถัดไป



## ข้อจำกัดของโครงการ Limitation:

- ระยะเวลาในการจัดเตรียมข้อมูลไม่เพียงพอ การจัดเตรียมข้อมูลเป็นขั้นตอนที่ยากที่สุดและใช้เวลานานกว่าส่วนอื่น นอกจากนี้การค้นหาแหล่งข้อมูล และข้อมูลภาพที่เหมาะสมมีข้อจำกัดอยู่มาก อีกทั้งใช้เวลามาก
- ระบบถูกพัฒนาสามารถใช้ทำงานได้หลายแบบและแยกแยะรีไซเคิลได้เพียง 3 ชนิดเท่านั้น
- หากภาพข้อมูลไม่มีอยู่ในการเรียนรู้ของ model ผลการทำงานจะเป็นค่า 0 โดย default ซึ่งคือ AluCan

## Further Improvement

- จัดเตรียมข้อมูล (ภาพ) ขยายที่หลากหลายชนิดมากขึ้น
- นำระบบต้นแบบในการประยุกต์การเรียนรู้การแยกแยะรีไซเคิลที่หลากหลายชนิด (มากกว่า 3 ชนิด) เพื่อให้สามารถนำไปใช้งานในอุตสาหกรรมการบริหารจัดการขยะได้จริง
- ออกแบบและปรับปรุงประสิทธิภาพโครงสร้างของ hardware ให้สามารถรองรับวัตถุที่มีขนาดใหญ่และน้ำหนักมาก

# Q & A



SCHOOL OF  
**SCIENCE**

KMITL  
**FIGHT**

SCHOOL  
OF ENGINEERING

**KMITL**

AIoT  
By IoT & Information Engineering  
School of Engineering KMITL

IMAKE