Image Processing : Activity 10 Reports

10.1 Load Base Model and Add New Layers

```python
import numpy as np
import cv2 as cv2
from matplotlib import pyplot as plt
import keras as keras
from keras.preprocessing import image
from keras.layers import Dense, GlobalAveragePooling2D
from keras.applications.mobilenet import preprocess_input, MobileNet
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
import sys
import pandas as pd
from sklearn.metrics import confusion_matrix, classification_report

import seaborn as sns
```

✓ 0.2s Python

```python
# Load base model
base_model = MobileNet(weights='imagenet', include_top=False)

# Add new layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
# we add dense layers so that the model can learn more complex functions and classif
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)  # dense layer 2
x = Dense(512, activation='relu')(x)  # dense layer 3
preds = Dense(3, activation='softmax')(x)

# Assign transfer base model + new layers to model
model = Model(inputs=base_model.input, outputs=preds)
model.summary()

# Assign Trainable Layers and freeze layer -> ลองเปลี่ยน ช่วง layer ในการ trainable Tru
for layer in model.layers[:20]:
    layer.trainable = False
for layer in model.layers[20:]:
    layer.trainable = True
```

✓ 0.6s Python

Result :

```
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, None, None, 3)]   0

 conv1 (Conv2D)              (None, None, None, 32)    864

 conv1_bn (BatchNormalizatio  (None, None, None, 32)   128
 n)

 conv1_relu (ReLU)           (None, None, None, 32)    0

 conv_dw_1 (DepthwiseConv2D)  (None, None, None, 32)   288

 conv_dw_1_bn (BatchNormaliz  (None, None, None, 32)   128
 ation)

 conv_dw_1_relu (ReLU)       (None, None, None, 32)    0

 conv_pw_1 (Conv2D)          (None, None, None, 64)    2048

 conv_pw_1_bn (BatchNormaliz  (None, None, None, 64)   256
 ation)
...
Total params: 5,854,403
Trainable params: 5,832,515
Non-trainable params: 21,888
```

เป็นการโหลด base model และเพิ่ม layer ใหม่ ที่ต้องการจะ train โดยนำ output ของ base model ไปทำการ reshape จาก 3D ไปเป็น 1D ด้วยฟังก์ชัน GlobalAveragePooling2D() จากนั้นเพิ่ม Dense เข้าไป 3 layers ตามโหนดที่กำหนด (1024, 1024, 512) และฟังก์ชันการตัดสินใจเป็น activation='relu' แล้วกำหนด output layer ที่จะมีจำนวนโหลดเท่ากับจำนวน class ผลลัพท์หรือจำนวน dense layers ที่เพิ่มเข้ามา กำหนด activation='softmax' เพื่อที่จะได้รู้ว่ามีโอกาสเป็น class ผลลัพท์ class ละกี่เปอร์เซ็นต์ และแสดง

class ผลลัพท์ ที่มีโอกาสมากที่สุดออกมา (preds) แล้วสร้าง model โดยกำหนด input เป็น base model, output เป็น preds ดังกล่าว สุดท้ายกำหนดการ train โดย trainable เป็น False คือการปิดการ train ให้ใช้ weight เก่าใน model เมื่อเป็น layer ที่ < 20 และให้ layer >= 20 คือ layer ที่ต้องการจะ train เพิ่มให้ trainable เป็น True

10.2 Model Training and Validation

10.2.1 Create ImageDataGenerator (Train, Validation)

```
# Create DataGeneartor Object
seed_value = 0
datagen = ImageDataGenerator(rescale=1./255, rotation_range=30, zoom_range=0.5,
                             width_shift_range=0.2, height_shift_range=0.2,
                             shear_range=0.15, horizontal_flip=True,
                             fill_mode="nearest")

# Create Train Image generator
train_generator = datagen.flow_from_directory('./Ship/train/',
                                              target_size=(224, 224), color_mode='rgb',
                                              batch_size=32,
                                              class_mode='categorical', seed=seed_value,
                                              shuffle=True)

# Create Validation Image generator
val_generator = datagen.flow_from_directory('./Ship/validate',
                                            target_size=(224, 224), color_mode='rgb',
                                            batch_size=23,
                                            class_mode='categorical', seed=seed_value,
                                            shuffle=True)
```

✓ 0.4s Python

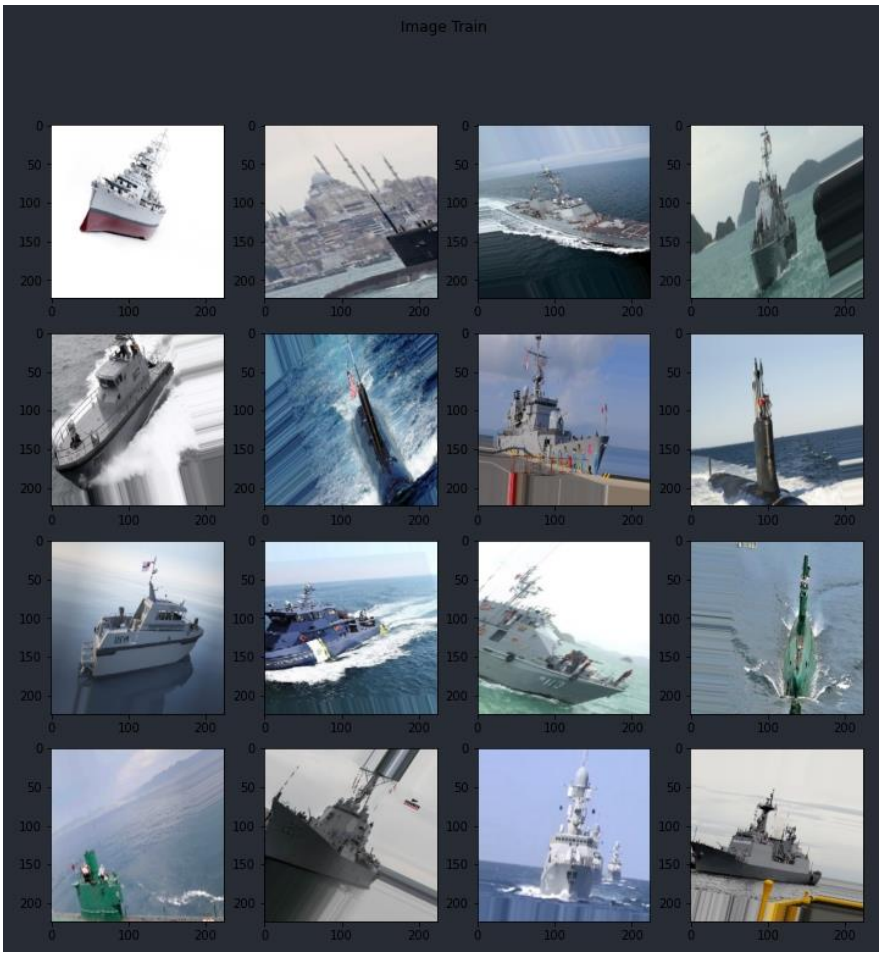
```
Found 97 images belonging to 3 classes.
Found 69 images belonging to 3 classes.
```

สร้าง ImageDataGenerator สำหรับการทำ geometric transform ให้ dataset กำหนด seed = 0 แล้วทำการสร้าง train image generator และ validate image generator โดยกำหนด Batch size ไว้ที่ 32 และ 23 ซึ่ง Batch size ของ validate image generator เป็นค่าที่ปรับหลังจากการกำหนดจำนวน step ที่จะใช้ train ในข้อ 10.2.2 แล้วทำการ plot ภาพออกมาเพื่อแสดงผล

```python
batch1 = train_generator.next()
img_train = (batch1[0]*255)
batch2 = val_generator.next()
img_val = (batch2[0]*255)

plt.figure()
f1, ax1 = plt.subplots(4, 4, figsize=(12, 12))
f2, ax2 = plt.subplots(4, 4, figsize=(12, 12))
f1.suptitle('Image Train')
f2.suptitle('Image Val')
index = 0
for i in range(4):
    for j in range(4):
        ax1[i][j].imshow(img_train[index].astype(np.uint8))
        ax2[i][j].imshow(img_val[index].astype(np.uint8))
        # ax1[i][j].title.set_text('img_train')
        # ax2[i][j].title.set_text('img_val')
        index += 1
```

✓ 3.1s Python

Result :



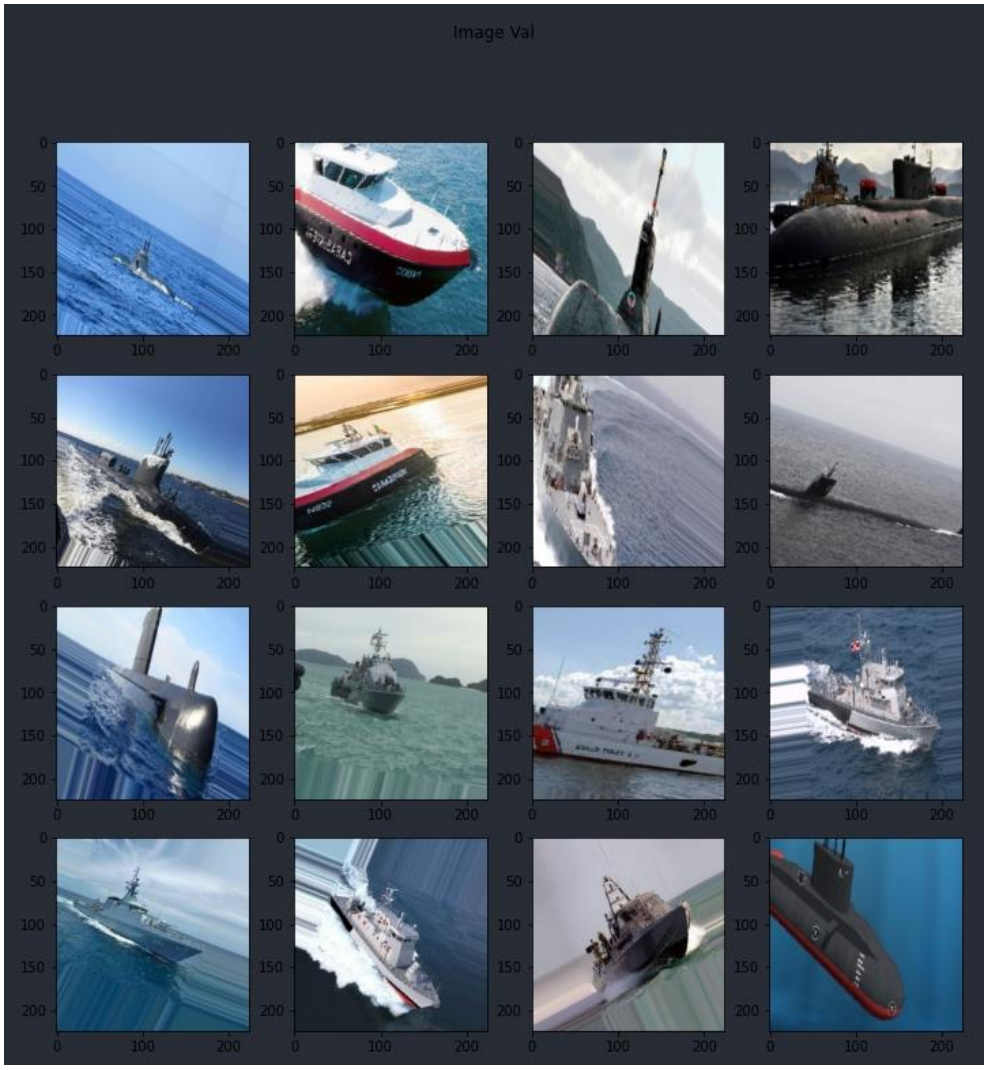


Image Val

10.2.2 Create Optimizer, parameters

```
model.compile(optimizer='Adam', loss='categorical_crossentropy',
              metrics=['accuracy'])

step_size_train = train_generator.n//train_generator.batch_size
step_size_val = val_generator.n//val_generator.batch_size

if step_size_train == step_size_val:
    print('equal')
else:
    print(train_generator.n)
    print(train_generator.batch_size)
    print(val_generator.n)
    print(val_generator.batch_size)
```

✓ 0.5s Python

ประกาศ optimizer เป็น adam โดยจะดูที่ค่า accuracy แล้วกำหนดจำนวน step ที่จะใช้ train แล้ว check ว่า step ของ train,validate เท่ากันหรือไม่ แล้วทำการปรับค่าให้ตรงกัน

10.2.3 Training, Validation, plot accuracy, loss

```
epoch = 60
history = model.fit(train_generator,
                    steps_per_epoch=step_size_train,
                    validation_data=val_generator,
                    validation_steps=step_size_val,
                    epochs=epoch,
                    verbose=1)
```

✓ 8m 30.8s Python

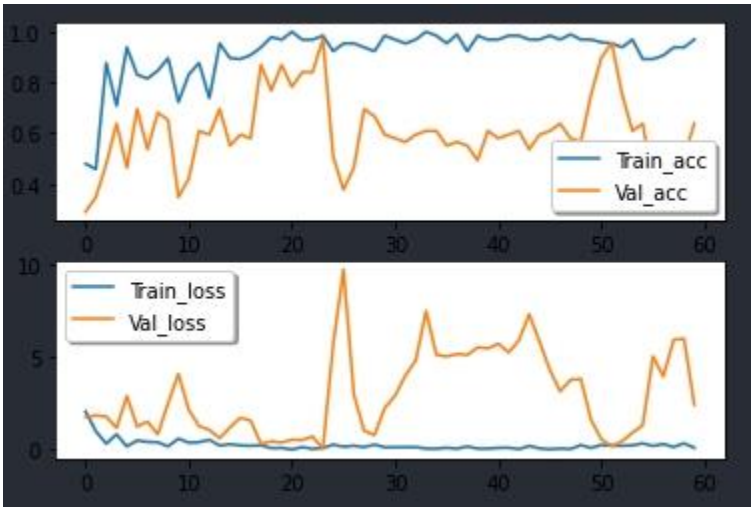
กำหนด epoch ที่ 60 รอบแล้วทำการ train model ด้วย parameter ที่เตรียมมาในข้อก่อนหน้านี้ทั้งหมด

```
Epoch 1/60
3/3 [==============================] - 9s 3s/step - loss: 2.0679 - accuracy: 0.4792 - val_loss:
1.7700 - val_accuracy: 0.2899
Epoch 2/60
3/3 [==============================] - 15s 6s/step - loss: 0.9921 - accuracy: 0.4583 - val_loss:
1.8436 - val_accuracy: 0.3478
Epoch 3/60
3/3 [==============================] - 10s 3s/step - loss: 0.3327 - accuracy: 0.8769 - val_loss:
1.7944 - val_accuracy: 0.4783
Epoch 4/60
3/3 [==============================] - 10s 3s/step - loss: 0.8341 - accuracy: 0.7083 - val_loss:
1.1730 - val_accuracy: 0.6377
Epoch 5/60
3/3 [==============================] - 8s 2s/step - loss: 0.1799 - accuracy: 0.9385 - val_loss:
2.8969 - val_accuracy: 0.4638
Epoch 6/60
3/3 [==============================] - 8s 4s/step - loss: 0.4994 - accuracy: 0.8308 - val_loss:
1.2483 - val_accuracy: 0.6957
Epoch 7/60
3/3 [==============================] - 8s 3s/step - loss: 0.4276 - accuracy: 0.8154 - val_loss:
1.5181 - val_accuracy: 0.5362
Epoch 8/60
3/3 [==============================] - 8s 3s/step - loss: 0.4022 - accuracy: 0.8462 - val_loss:
0.8321 - val_accuracy: 0.6812
Epoch 9/60
3/3 [==============================] - 10s 3s/step - loss: 0.1794 - accuracy: 0.8958 - val_loss:
2.5155 - val_accuracy: 0.6522
Epoch 10/60
3/3 [==============================] - 8s 4s/step - loss: 0.6005 - accuracy: 0.7231 - val_loss:
4.0992 - val_accuracy: 0.3478
Epoch 11/60
3/3 [==============================] - 7s 2s/step - loss: 0.3823 - accuracy: 0.8308 - val_loss:
2.1518 - val_accuracy: 0.4203
Epoch 12/60
3/3 [==============================] - 7s 3s/step - loss: 0.4073 - accuracy: 0.8769 - val_loss:
1.2799 - val_accuracy: 0.6087
Epoch 13/60
...
```

```python
f3, ax1 = plt.subplots(2)

ax1[0].plot(history.history['accuracy'], label='Train_acc')
ax1[0].plot(history.history['val_accuracy'], label='Val_acc')
ax1[0].legend(shadow=True, fancybox=True)

ax1[1].plot(history.history['loss'], label='Train_loss')
ax1[1].plot(history.history['val_loss'], label='Val_loss')
ax1[1].legend(shadow=True, fancybox=True)
```

✓ 0.5s Python

Result :



นำผลที่ได้จากการ train มาแสดงผลเพื่อดู accuracy และ loss ของ train dateset และ validate dataset โดยจากกราฟที่เห็นจะเห็นว่า accuracy และ loss ของ train dataset มีความเหวี่ยงของค่าค่อนข้างน้อย ต่างจาก Validate dataset ที่มีความคงที่น้อยทั้ง accuracy และ loss

10.3 Model Testing and Performance Visualize

10.3.1 Create ImageDataGenerator (Test) and Predict Results

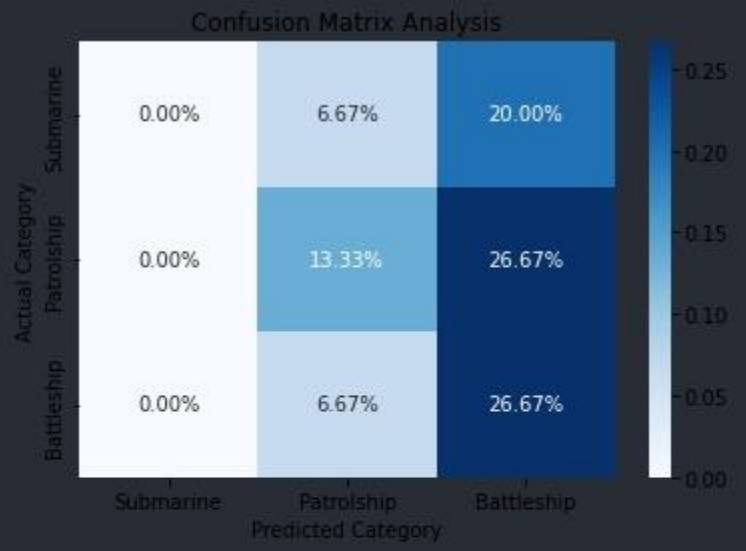
```python
test_datagen = ImageDataGenerator( rescale = 1./255)
test_generator = test_datagen.flow_from_directory('./Ship/test',
target_size = (244,244),
color_mode = 'rgb',
class_mode = 'categorical',
shuffle=False,
seed = seed_value,
batch_size = 1
)

y_true= test_generator.classes

test_generator.reset()
pred_prob = []

for i in range(len(y_true)):
    pred =model.predict(test_generator.next()[0],verbose=0)
    pred_prob.append(np.array(pred[0]))

df_pred = pd.DataFrame(pred_prob)
df_class = df_pred.idxmax(axis=1)

cf_matrix = confusion_matrix(y_true, df_class)
axx = sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt = '.2%', cmap='Blues')

axx.set_title('Confusion Matrix Analysis')
axx.set_ylabel('Actual Category')
axx.set_xlabel('Predicted Category')

axx.xaxis.set_ticklabels(['Submarine','Patrolship', 'Battleship'])
axx.yaxis.set_ticklabels(['Submarine','Patrolship', 'Battleship'])

plt.show()
print(classification_report(y_true, df_class, zero_division = 1))
```

✓ 2.1s Python

เป็นการสร้าง test generator ขึ้นมา และนำ class ของรูปภาพที่ต้องการจะ test ใส่ไว้ใน y_true เพื่อที่จะได้ class label ที่ถูกต้องมา และทำ model ทำนายรูปทีละรูปด้วย model.predict_generator() ด้วย test_ generator.next ให้ generate รูปมาทีละรูป เพื่อให้ได้ pred_prob จากนั้นหา prediction class ID โดย เริ่มจากวัดประสิทธิภาพด้วย confusion_matrix() กับ classification_report() เทียบระหว่าง class label ที่ถูกต้องตามจริง (y_true) กับ class ที่ predict มา (df_class)

Result :

```
Found 15 images belonging to 3 classes.
```



	precision	recall	f1-score	support
0	1.00	0.00	0.00	4
1	0.50	0.33	0.40	6
2	0.36	0.80	0.50	5
accuracy			0.40	15
macro avg	0.62	0.38	0.30	15
weighted avg	0.59	0.40	0.33	15

Question

Q1: Number of training, validation, test images?

- จำนวนข้อมูลในแต่ละ class มีปริมาณใกล้เคียงกันหรือไม่

  ตอบ จำนวนข้อมูลในแต่ละ class มีปริมาณใกล้เคียงกัน

- จำนวนข้อมูลทั้งหมด มีมากพอจะให้เข้าใจความแตกต่างของ class หรือไม่

  ตอบ จำนวนข้อมูลทั้งหมด มีมากพอจะให้เข้าใจความแตกต่างของ class

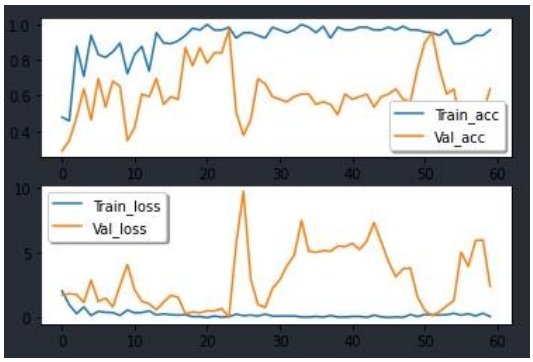
Q2: Which pre-train layers are set trainable?

- ปรับแล้ว มีผลต่อ accuracy มากน้อย อย่างไร

  ตอบ มีผลต่อ accuracy โดยที่หากให้มี layer ที่ทำการ train เพิ่มขึ้น accuracy ก็จะสูงขึ้น

Q3: What is the maximum accuracies of train, validation, test(predict)?

เมื่อ training ผ่านไปในแต่ละ Epoch



- Training accuracy เป็นอย่างไร เมื่อเทียบกับ Validation accuracy

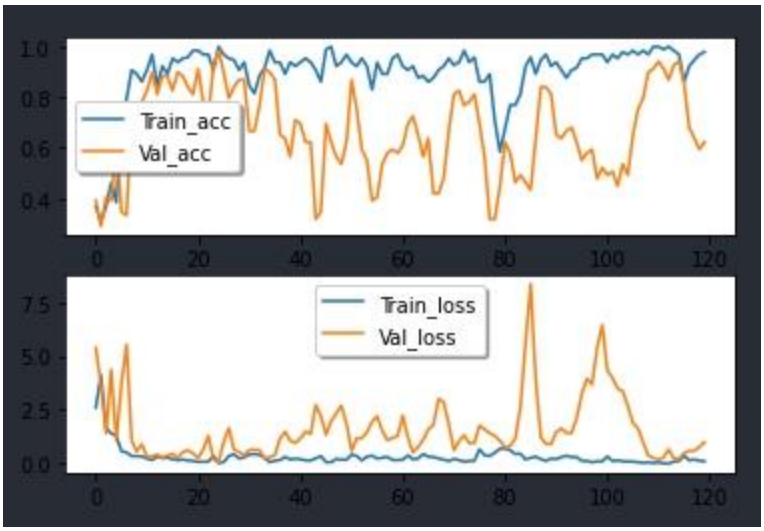
  ตอบ ค่อนข้างต่าง Training accuracy = 1.0 , Validation accuracy = 0.6 อาจจะเป็น overfit

- Training Loss เป็นอย่างไร เมื่อเทียบกับ Validation loss

  ตอบ ไม่ต่างกันมาก Training Loss = 0.1 , Validation Loss = 0.25

- เพิ่มจำนวน Epoch แล้ว มีผลอย่างไรกับค่า accuracy, loss

  ตอบ Epoch = 120, Accuracy อยู่ในช่วง 80%-100% เพิ่มขึ้น และ loss ใกล้ 0 มากกว่าเดิม



Q4: จาก confusion matrix ดูอย่างไรว่า class ไหน ทำนายผิด และ ที่ทำนายผิด ทำนายไปเป็น class ใด

ตอบ มี class Battleship ที่มีการทำนายผิดน้อยที่สุด คือทำนายผิดจาก class Battleship ไปเป็น class Patrolship และ Patrolship กับ Submarine ที่มี %Confusion เท่ากัน โดยไม่สามารถทำนาย class Submarine ถูกเลย และ class Patrolship มีการทำนายผิดเป็น class Battleship