64050194 ภูริดา กำเหนิด

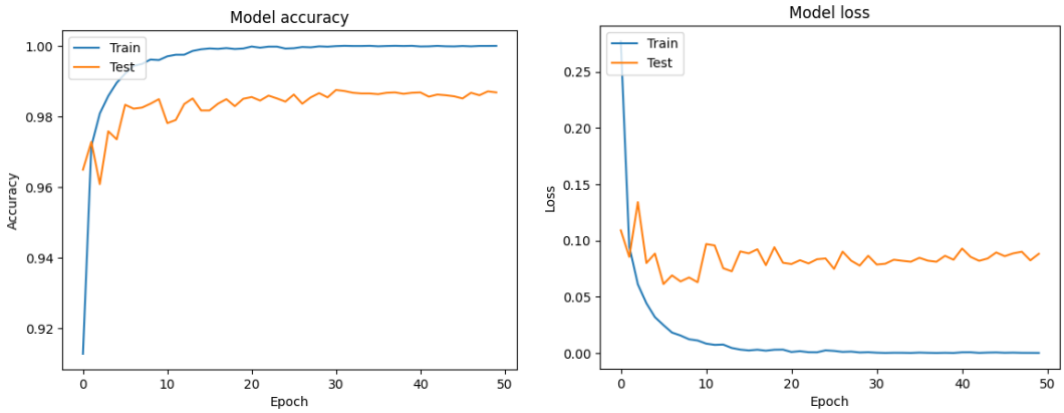
การแก้ปัญหาการ Overfitting

โดยเลือกใช้ Convolutional Neural Network (CNN) และใช้ dataset MNIST ซึ่งเป็นชุดข้อมูลที่ใช้ในการศึกษาและทดสอบการทำนายตัวเลขที่เขียนด้วยลายมือ ซึ่งเป็นชุดข้อมูลมาตรฐานสำหรับปัญหาการจำแนกหมวดหมู่ในศาสตร์ข้อมูลและการเรียนรู้เชิงลึก โดยมีลักษณะของชุดข้อมูลประกอบด้วยภาพขนาด 28x28 พิกเซลของตัวเลข 0-9 ที่เขียนด้วยลายมือ ซึ่งมีจำนวนรูปภาพทั้งหมด 70,000 รูป (60,000 รูปภาพใช้สำหรับการฝึกและ 10,000 รูปภาพใช้สำหรับการทดสอบ) เป็นชุดข้อมูลที่นิยมในการทดสอบและเปรียบเทียบประสิทธิภาพของโมเดลการเรียนรู้ของเครื่องในการจำแนกและระบุตัวเลข โดยมี Code เป็นดังนี้

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import RMSprop
import numpy as np
import matplotlib.pyplot as plt

# Load MNIST data
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Preprocess the data
x_train = x_train.reshape(60000, 784).astype('float32') / 255
x_test = x_test.reshape(10000, 784).astype('float32') / 255
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)

model = Sequential()
model.add(Dense(1024, activation='relu', input_shape=(784,)))
model.add(Dropout(0.1))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(10, activation='softmax'))

model.summary()

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

history = model.fit(x_train, y_train,
                    batch_size=256,
                    epochs=50,
```

```
                    verbose=1,
                    validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

Output ที่ได้รับ คือ



จากกราฟของ Model accuracy จะเห็นได้ว่า เส้นของ train และ test มีความแตกต่างกันเป็นอย่างมาก ซึ่งแสดงให้เห็นการถึงปัญหา Overfitting และในขณะเดียวกัน กราฟของ Model loss จะสามารถเห็นได้ว่าเส้นของ train มีการลดลงอย่างต่อเนื่อง ซึ่งตรงกันข้ามกันเส้นของ test ซึ่งแสดงให้เห็นการถึงปัญหา Overfitting เช่นเดียวกัน

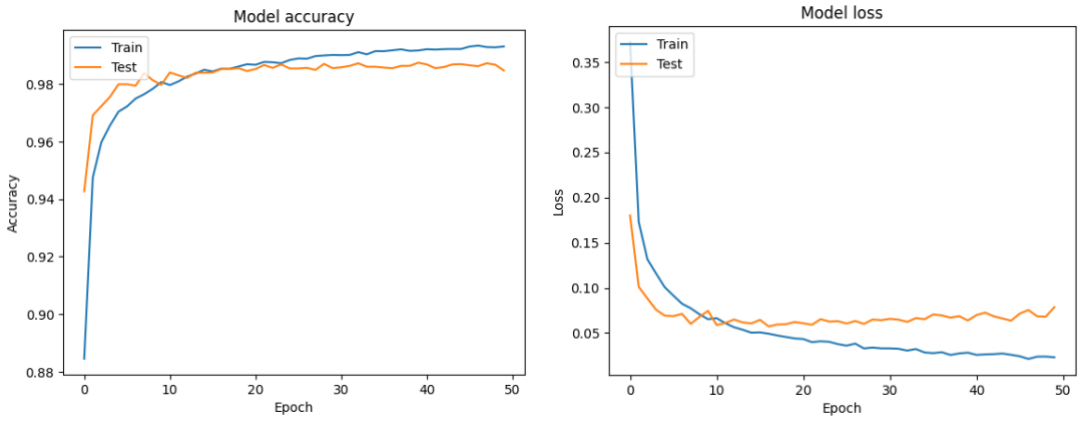
วิธีการแก้ปัญหา overfit

วิธีการที่ 1 : เพิ่ม Dropout Rate

การเพิ่ม dropout rate เป็นวิธีหนึ่งในการป้องกัน overfitting โดย dropout จะปิดการทำงานของบางจำนวนของ units ใน layer ก่อนหน้านั้น ซึ่งช่วยลดความเชื่อมโยงระหว่าง units และช่วยลดการติดอยู่กับข้อมูลของข้อมูลการเรียนรู้ที่เกินไป โดยในที่นี้จะใช้ dropout rate 0.6 ซึ่งหมายถึงจะปิดการทำงานของ units ใน layer ดังกล่าวในระดับ 60% โดยมีการแก้ไขและเพิ่มเติม code เป็นดังนี้

```
model = Sequential()
model.add(Dense(1024, activation='relu', input_shape=(784,)))
# เพิ่ม Dropout rate เป็น 0.6
model.add(Dropout(0.6))
model.add(Dense(1024, activation='relu'))
# เพิ่ม Dropout rate เป็น 0.6
model.add(Dropout(0.6))
model.add(Dense(10, activation='softmax'))
```

Output ที่ได้รับ คือ



จากกราฟของ Model accuracy จะเห็นได้ว่า เส้นของ train และ test มีความแตกต่างลดลงจากกราฟก่อนหน้าซึ่งมีการใช้ dropout rate เพียง 0.1 ซึ่งแสดงให้เห็นว่าการเพิ่ม dropout rate สามารถช่วยลดปัญหาการเกิด Overfitting ได้ และในขณะเดียวกัน กราฟของ Model loss จะสามารถเห็นได้ว่าเส้นของ test ก็มีการลดลงจากกราฟก่อนหน้าซึ่งมีการใช้ dropout rate เพียง 0.1 เช่นกัน ซึ่งแสดงให้เห็นว่าการเพิ่ม dropout rate สามารถช่วยลดปัญหา การเกิด Overfitting ได้

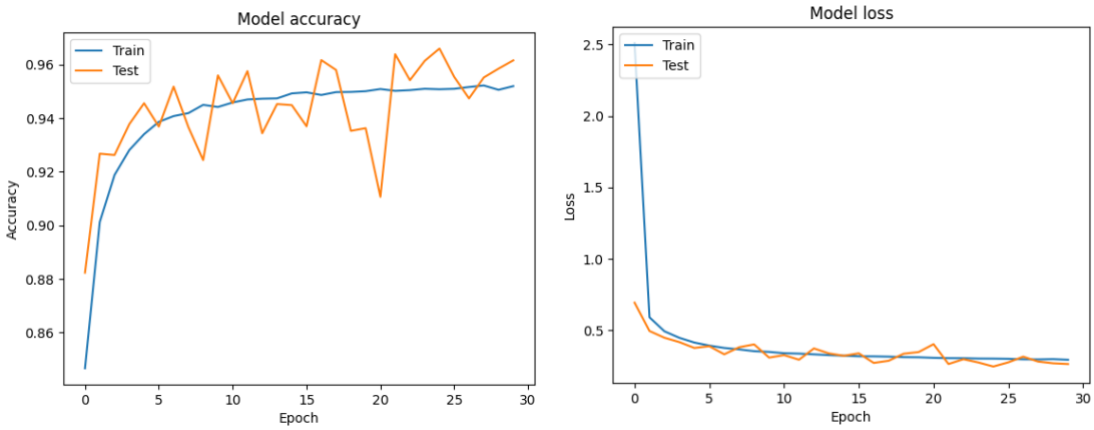
วิธีการที่ 2 : Regularization L2

การใช้ Regularization L2 เพื่อลดความเสี่ยงที่จะเกิด Overfitting โดยการเพิ่มค่า penalty term ในฟังก์ชัน loss ซึ่งจะช่วยลดค่าของน้ำหนักในโมเดล และทำให้โมเดลมีความเรียบง่ายมากขึ้น โดยมีการกำหนดให้ Regularization L2 มีค่าเท่ากับ 0.01 โดยมีการแก้ไขและเพิ่มเติม code เป็นดังนี้

```
model = Sequential()
model.add(Dense(1024, activation='relu', kernel_regularizer=regularizers.l2(0.01), input_shape=(784,)))
```

```
model.add(Dropout(0.1))
model.add(Dense(1024, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dropout(0.1))
model.add(Dense(10, activation='softmax'))
```

Output ที่ได้รับ คือ



จากกราฟของ Model accuracy จะเห็นได้ว่า เส้นของ train และ test มีความแตกต่างกัน โดยจะเห็นได้ว่าเส้นของ test มีแนวโน้มที่ไม่แน่นอนมีการขึ้น ๆ ลง ๆ และในขณะเดียวกัน กราฟของ Model loss จะสามารถเห็นได้ว่าเส้นของ train มีการลดลงอย่างต่อเนื่อง และเส้นของ test จะมีค่าที่ค่อนข้างคงที่อยู่ที่มีการขึ้นลงบ้างเล็กน้อยแต่ไม่มาก

วิธีการที่ 3 : Data augmentation

Data Augmentation เป็นวิธีที่มีประสิทธิภาพในการป้องกัน overfitting และเพิ่มความสามารถในการทำนายของโมเดล โดย Data Augmentation จะทำการปรับเปลี่ยนข้อมูลภาพอันเดียวกันให้เป็นรูปแบบที่แตกต่างกันเล็กน้อย ซึ่งทำให้โมเดลเรียนรู้และทดสอบกับข้อมูลที่หลากหลายมากขึ้น โดยมีการปรับเปลี่ยนรูปภาพโดยการกำหนดค่าต่าง ๆ ดังนี้

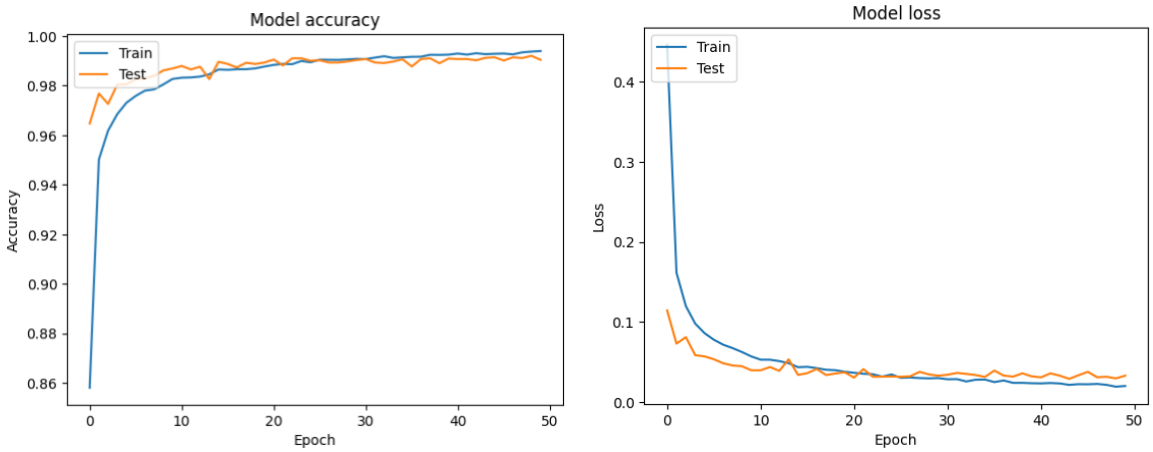
1. rotation_range กำหนดช่วงการหมุนภาพ (องศา) ที่จะถูกใช้ในการสร้างข้อมูลภาพใหม่ ค่าที่ใช้ในที่นี้คือ 2 องศา
2. width_shift_range กำหนดช่วงการเลื่อนตำแหน่งภาพในแนวนอน ที่จะถูกใช้ในการสร้างข้อมูลภาพใหม่ ค่าที่ใช้ในที่นี้คือ 0.1 (โดยมีค่าอยู่ในช่วง -0.1 ถึง +0.1)
3. height_shift_range กำหนดช่วงการเลื่อนตำแหน่งภาพในแนวตั้ง ที่จะถูกใช้ในการสร้างข้อมูลภาพใหม่ ค่าที่ใช้ในที่นี้คือ 0.05 (โดยมีค่าอยู่ในช่วง -0.05 ถึง +0.05).
4. shear_range กำหนดช่วงการเบน ที่จะถูกใช้ในการสร้างข้อมูลภาพใหม่ ค่าที่ใช้ในที่นี้คือ 0.03
5. zoom_range กำหนดช่วงการซูมภาพ ที่จะถูกใช้ในการสร้างข้อมูลภาพใหม่ ค่าที่ใช้ในที่นี้คือ 0.05 (โดยมีค่าอยู่ในช่วง 0.95 ถึง 1.05).

6. horizontal_flip กำหนดว่าจะใช้การพลิกภาพแนวนอนหรือไม่ โดยมีค่าเป็น True หรือ False เพื่อสร้างข้อมูลภาพใหม่ ค่าที่ใช้ในที่นี้กำหนดเป็น False หมายถึง ไม่มีการพลิกภาพแนวนอน
7. fill_mode กำหนดวิธีการเติมพื้นที่ที่ว่างหรือเกิดขึ้นในกรณีที่มีการเปลี่ยนขนาดภาพ ค่าที่ใช้ในที่นี้คือ 'nearest' ที่ใช้วิธีการสำหรับการเติมพื้นที่ที่ใกล้ที่สุดกับค่าที่มีอยู่

โดยมีการแก้ไขและเพิ่มเติม code เป็นดังนี้

```
# Data augmentation
datagen = ImageDataGenerator(
    rotation_range=2,
    width_shift_range=0.1,
    height_shift_range=0.05,
    shear_range=0.03,
    zoom_range=0.05,
    horizontal_flip=False,
    fill_mode='nearest'
)
datagen.fit(x_train)
```

Output ที่ได้รับ คือ



จากกราฟของ Model accuracy จะเห็นได้ว่า เส้นของ train และ test มีค่า accuracy ที่สูงใกล้เคียงกัน ซึ่งเส้นของ test มีค่าที่สูงกว่ากราฟก่อนหน้าที่ยังไม่มีการแก้ไขปัญหา Overfitting และในตอนเริ่มต้นเส้นของ test มีค่า accuracy ที่สูงกว่าเส้นของ train และในขณะเดียวกัน กราฟของ Model loss จะสามารถเห็นได้ว่า เส้นของ train ค่า loss มีการลดลงอย่างต่อเนื่อง และเส้นของ test ค่า loss มีค่าที่ค่อนข้างต่ำกว่ากราฟก่อนหน้าที่ยังไม่มีการแก้ไขปัญหา Overfitting เช่นเดียวกัน ซึ่งแสดงให้เห็นการถึงปัญหา Overfitting