class Mod(tf.keras.models.Model):

def \_\_init\_\_(self,out\_dim,\*\*kwargs):

super().\_\_init\_\_(\*\*kwargs)

self.h1=tf.keras.layers.Conv2D(32,(3,3),activation='relu',kernel\_initializer='he\_normal')

self.h2=tf.keras.layers.BatchNormalization()

self.h3=tf.keras.layers.MaxPooling2D(2,2)

self.h4=tf.keras.layers.Conv2D(64,(3,3),activation='relu',kernel\_initializer='he\_normal')

self.h5=tf.keras.layers.BatchNormalization()

self.h6=tf.keras.layers.MaxPooling2D(2,2)

self.h7=tf.keras.layers.Conv2D(128,(3,3),activation='selu',kernel\_initializer='lecun\_normal')

self.h8=tf.keras.layers.BatchNormalization()

self.h9=tf.keras.layers.Dropout(0.3)

self.flatten=tf.keras.layers.Flatten()

self.dense=tf.keras.layers.Dense(256,activation='relu',kernel\_initializer='he\_normal')

self.out=tf.keras.layers.Dense(out\_dim,activation='softmax',kernel\_regularizer=l1(0.1))

def call(self,inputs):

z=inputs

z=self.h1(z)

……….

return self.out(z)

model=Mod(10)

model.compile(optimizer='adam',loss='categorical\_crossentropy',metrics=['accuracy'])

results=model.fit(x\_train,y\_train,epochs=10,batch\_size=32,validation\_data=(x\_test,y\_test))

Epoch 5/10

1875/1875 [==============================] - 137s 73ms/step - loss: 0.3716 - accuracy: 0.9077 - val\_loss: 0.4352 - val\_accuracy: 0.8762

Epoch 6/10

1875/1875 [==============================] - 165s 88ms/step - loss: 0.3495 - accuracy: 0.9128 - val\_loss: 0.4384 - val\_accuracy: 0.8784

Epoch 7/10

1875/1875 [==============================] - 158s 85ms/step - loss: 0.3433 - accuracy: 0.9151 - val\_loss: 0.3896 - val\_accuracy: 0.8980

Epoch 8/10

1875/1875 [==============================] - 152s 81ms/step - loss: 0.3336 - accuracy: 0.9192 - val\_loss: 0.3939 - val\_accuracy: 0.8967

Epoch 9/10

1875/1875 [==============================] - 154s 82ms/step - loss: 0.3265 - accuracy: 0.9195 - val\_loss: 0.3686 - val\_accuracy: 0.9050

Epoch 10/10

1875/1875 [==============================] - 143s 76ms/step - loss: 0.3207 - accuracy: 0.9222 - val\_loss: 0.3564 - val\_accuracy: 0.9053

CNN 모형을 사용하면서 각 층마다 batch normalization을 수행하였고, 출력층에서 L1 regularization을 수행하였다. 또한 CNN 마지막 layer에서 0.3만큼의 dropout을 수행하여 과대적합을 막았다. 또한 CNN의 채널을 줄임으로써 추정해야 할 모수를 줄여 과대적합을 방지했다. 마지막으로 Flatten을 통해 MLP 모형에 적합할 수 있도록 바꿨고, 출력층 전에 노드가 256인 MLP층을 하나 추가해 모델의 정확도를 높였다.

Fit 결과를 보면, validation loss가 지속적으로 감소하고 있고, validation accuracy가 수렴하고 있기에 과대적합문제가 발생하지 않았다고 볼 수 있다. 오히려 에폭을 조금 더 크게 잡아 정밀도를 높일 필요가 있어 보인다.