HW6

Data

```
x_train shape: (1000, 28, 28, 1)
y_train shape: (1000, 10)
large_x_train shape: (2000, 28, 28, 1)
large_y_train shape: (2000, 10)
```

이상적인 train,test의 비율은 train:test = 8:2정도 이지만, Overfitting을 유도하기 위해 train:test = 1:10 (x_train), train:test = 1:5 (large_x_train)으로 설정하였습니다.

학습 data가 test data보다 매우 적은것으로 보아 accuracy가 많이 낮을것을 예상할 수 있습니다.

Base Line

```
batch_size = 28
num_classes = 10
epochs = 50
```

```
### START CODE HERE ###
model = tf.keras.models.Sequential([
                                    tf.keras.layers.Conv2D(32,(3,3),activation = 'relu', input_shape = (28,28,1)),
                                    #activation 포함 (relu)
                                    tf.keras.layers.MaxPool2D(pool_size=(2,2)),
                                    tf.keras.layers.Conv2D(32,(3,3),activation = 'relu', input_shape = (28,28,1)),
                                    #activation 포함 (relu)
                                    tf.keras.layers.MaxPool2D(pool_size=(2,2)),
                                    tf.keras.layers.Flatten(),
                                    tf.keras.layers.Dense(10,activation='softmax')
                                    #마지막 dense에 acitvation은 softmax

])
### END CODE HERE ###
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_20 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 11, 11, 32)        9248
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 5, 5, 32)          0
_________________________________________________________________
flatten_10 (Flatten)         (None, 800)               0
_________________________________________________________________
dense_10 (Dense)             (None, 10)                8010
=================================================================
Total params: 17,578
Trainable params: 17,578
Non-trainable params: 0
_________________________________________________________________
```

```
Test loss: 2.3198959827423096
Test accuracy: 0.13289999961853027
```

주어진 parameter를 맞추기 위해 convolution layer의 kernel size와 max pooling의 parameter를 다음과 같이 설정하였습니다.

Activation function은 conv2D 함수에 포함시켰습니다.

예상한대로 test accuracy가 0.13로 낮게 나온 것을 확인할 수 있었습니다.

Regulization

Lage Data

```
72/72 [==============================] - 4s 56ms
Largemodel Test loss: 2.2684571743011475
Largemodel Test accuracy: 0.19580000638961792
```

Train data를 1000개에서 2000개로 늘려보았습니다.

Test accuracy가 0.06증가하였습니다.

하지만 여전히 train:test = 1:5로 비이상적인 비율이기 때문에 accuracy가 0.196로 매우 낮습니다.

Drop out

```
dropout_model = tf.keras.models.Sequential([
                                    tf.keras.layers.Conv2D(32,(3,3),activation = 'relu', input_shape = (28,28,1)),
                                    #activation 포함 (relu)
                                    tf.keras.layers.MaxPool2D(pool_size=(2,2)),
                                    tf.keras.layers.Conv2D(32,(3,3),activation = 'relu', input_shape = (28,28,1)),
                                    #activation 포함 (relu)
                                    tf.keras.layers.MaxPool2D(pool_size=(2,2)),
                                    tf.keras.layers.Dropout(0.2),

                                    tf.keras.layers.Flatten(),
                                    tf.keras.layers.Dropout(0.2),
                                    tf.keras.layers.Dense(10,activation='softmax')
                                    #마지막 dense에 acitvation은 softmax

])
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 11, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 32)          0
_________________________________________________________________
dropout (Dropout)            (None, 5, 5, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 800)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 800)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                8010
=================================================================
Total params: 17,578
Trainable params: 17,578
Non-trainable params: 0
_________________________________________________________________
```

```
Dropout model Test loss: 2.270078420639038
Dropout model Test accuracy: 0.1941000074148178
```

Overfitting을 방지하는 regulization의 방법 중 drop out을 사용하였습니다. 학습을 하다보면 weight들이 서로 동조하는 현상이 발생할 수 있는데, 이를 의도적으로 무시하면서 동조화현상을 피할 수 있습니다. 단, 학습시간이 증가합니다.Parameter는 0.2로 설정하였습니다.

Test accuracy가 Base line(0.15)보다 0.04증가하였습니다. (0.194)

Drop out rate를 0.2에서 0.5로 변경하였습니다.

즉 node를 끊는 비율을 20%에서 50%로 늘렸다.

```
Dropout 0.8 model Test loss: 2.289858102798462
Dropout 0.8model Test accuracy: 0.09440000355243683
```

Data의 수가 적기 때문에 drop out rate를 0.5로 늘리게 되면 test accuracy가 매우 낮아진다.

Batch Normaliztion

```
bn_model = tf.keras.models.Sequential([
                                    tf.keras.layers.Conv2D(32,(3,3),activation = 'relu', input_shape = (28,28,1)),
                                    #activation 포함 (relu)
                                    tf.keras.layers.BatchNormalization(),
                                    tf.keras.layers.MaxPool2D(pool_size=(2,2)),

                                    tf.keras.layers.Conv2D(32,(3,3),activation = 'relu', input_shape = (28,28,1)),
                                    #activation 포함 (relu)
                                    tf.keras.layers.BatchNormalization(),
                                    tf.keras.layers.MaxPool2D(pool_size=(2,2)),

                                    tf.keras.layers.Flatten(),
                                    tf.keras.layers.Dense(10,activation='softmax')
                                    #마지막 dense에 acitvation은 softmax

])
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
batch_normalization_6 (Batch (None, 26, 26, 32)        128
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 11, 11, 32)        9248
_________________________________________________________________
batch_normalization_7 (Batch (None, 11, 11, 32)        128
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 5, 5, 32)          0
_________________________________________________________________
flatten_12 (Flatten)         (None, 800)               0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                8010
=================================================================
Total params: 17,834
Trainable params: 17,706
Non-trainable params: 128
_________________________________________________________________
```

```
BN model Test loss: 2.7836060523986816
BN model Test accuracy: 0.18279999494552612
```

Final Model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_26 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
batch_normalization_8 (Batch (None, 26, 26, 32)        128
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 11, 11, 32)        9248
_________________________________________________________________
batch_normalization_9 (Batch (None, 11, 11, 32)        128
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 5, 5, 32)          0
_________________________________________________________________
dropout_18 (Dropout)         (None, 5, 5, 32)          0
_________________________________________________________________
flatten_13 (Flatten)         (None, 800)               0
_________________________________________________________________
dropout_19 (Dropout)         (None, 800)               0
_________________________________________________________________
dense_13 (Dense)             (None, 10)                8010
=================================================================
Total params: 17,834
Trainable params: 17,706
Non-trainable params: 128
_________________________________________________________________


Final model Test loss: 2.11065673828125
Final model Test accuracy: 0.29100000858306885
```

Image Augumentation

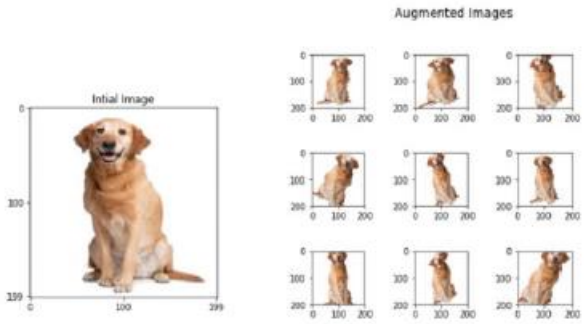
```
datagen  = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")

datagen.fit(x_train)
```

Keras의 ImageDataGenerator를 사용하여 data augumentation을 진행하였습니다.

(공식 튜토리알 : https://keras.io/api/preprocessing/image/)

Data augumentation은 기존 데이터에서 새로운 데이터를 생성해 인공적으로 train set의 크기를 늘리는 방법입니다.

이미지에선 Rotation, shift, 밝기조절, zoom등 다양하게 image data를 증대할 수 있습니다.




```
Final model Test loss: 2.1502387523651123
Final model Test accuracy: 0.30820000171661377
```

Final model에 적용하였을 때, test accuracy가 0.3으로 기존 x_train으로 진행한 final model보다 높게 나온것을 확인할 수 있었습니다.

(회전 , 이동 , 확대 등의 Affine Transform 이외에도 여러가지 방법의 Augmentation 이 있습니다

가우시안 분포를 가진 난수를 더하거나 Elastic Transform 을 통해 형태를 변화시킬 수 있습니다 .)

X_train보다 2배 많은 Large_x_train을 사용하여 image data augumentation을 진행하였습니다. (증대 방법 동일, data만 다르게 증대)

```
history_datagen =final_model.fit(datagen.flow(large_x_train, large_y_train, batch_size=batch_size),
          steps_per_epoch=len(large_x_train) / batch_size, epochs=epochs)

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

```
Final model Test loss: 1.9738690853118896
Final model Test accuracy: 0.34200000762939453
```

```
for e in range(100):
    print('Epoch', e)
    batches = 0
    for x_batch, y_batch in datagen.flow(large_x_train, large_y_train, batch_size=batch_size):
        final_model.fit(x_batch, y_batch)
        batches += 1
        if batches >= len(large_x_train) / batch_size:
            # we need to break the loop by hand because
            # the generator loops indefinitely
            break
```

(imagedatageneration 학습 수동으로 진행)

Epoch을 늘려 학습을 진행하였을 때, 보다 높은 accuracy를 얻을 수 있었습니다.

결과적으로 regulization을 모두 사용하고, image augumentation을 하였을 때, 가장 높은 accuracy를 얻을 수 있었습니다.

(Epoch을 늘리고 batch size를 크게하는것도 test accuracy를 높일 수 있는 방법중 하나입니다.)