합성곱 신경망의 시각화

구글 코랩에서 실행하기

가중치 시각화

In [1]:
```
from tensorflow import keras
```

In [2]:
```
# 코랩에서 실행하는 경우에는 다음 명령을 실행하여 best-cnn-model.h5 파일을 다운로드받아 사용하세요.
!wget https://github.com/rickiepark/hg-mldl/raw/master/best-cnn-model.h5
```

```
--2023-12-08 12:32:29--  https://github.com/rickiepark/hg-mldl/raw/master/best-cnn-model.h5
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/rickiepark/hg-mldl/master/best-cnn-model.h5 [following]
--2023-12-08 12:32:30--  https://raw.githubusercontent.com/rickiepark/hg-mldl/master/best-cnn-model.h5
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4046712 (3.9M) [application/octet-stream]
Saving to: 'best-cnn-model.h5'

best-cnn-model.h5   100%[===================>]   3.86M  --.-KB/s    in 0.08s

2023-12-08 12:32:30 (46.3 MB/s) - 'best-cnn-model.h5' saved [4046712/4046712]
```

In [3]:
```
model = keras.models.load_model('best-cnn-model.h5') # 이전 생성된 모델의 가중치를 제외 모든게 파일을 읽어서 model에 저장
```

In [4]:
```
model.layers
```

Out[4]:
```
[<keras.src.layers.convolutional.conv2d.Conv2D at 0x7fb92756c8b0>,
 <keras.src.layers.pooling.max_pooling2d.MaxPooling2D at 0x7fb92756db40>,
 <keras.src.layers.convolutional.conv2d.Conv2D at 0x7fb92756eaa0>,
 <keras.src.layers.pooling.max_pooling2d.MaxPooling2D at 0x7fb92756d6f0>,
 <keras.src.layers.reshaping.flatten.Flatten at 0x7fb92758dcd0>,
 <keras.src.layers.core.dense.Dense at 0x7fb92758f100>,
 <keras.src.layers.regularization.dropout.Dropout at 0x7fb92756e8f0>,
 <keras.src.layers.core.dense.Dense at 0x7fb916518a60>]
```

In [5]:
```
conv = model.layers[0] # 첫 번째 합성곱 층을 conv에 저장

print(conv.weights[0].shape, conv.weights[1].shape)
```

```
(3, 3, 1, 32) (32,)
```

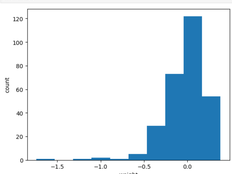
In [6]:
```
conv_weights = conv.weights[0].numpy()
# Tensor클래스의 객체인 weights 속성을 Numpy()로 넘파이 배열 형태로 변형

print(conv_weights.mean(), conv_weights.std())
# 가중치 배열의 평균과 표준편차 출력
```

```
-0.02494116 0.24951957
```

In [7]:
```
import matplotlib.pyplot as plt
```

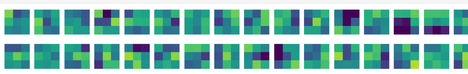
In [8]:
```
plt.hist(conv_weights.reshape(-1, 1))
# 히스토그램으로 하기 위해서 1차원 배열로 하기 위해 reshape(-1,1) 실행
plt.xlabel('weight')
plt.ylabel('count')
plt.show()
```



In [9]:
```
fig, axs = plt.subplots(2, 16, figsize=(15,2))

for i in range(2):
    for j in range(16):
        axs[i, j].imshow(conv_weights[:,:,0,i*16 + j], vmin=-0.5, vmax=0.5)
        axs[i, j].axis('off')

plt.show()
# 32개의 커널을 16개씩 두 줄에 출력
```



In [10]:
```
no_training_model = keras.Sequential()
# 훈련되지 않은 sequential 클래스 객체 no_training_model todtjd

no_training_model.add(keras.layers.Conv2D(32, kernel_size=3, activation='relu',
                                          padding='same', input_shape=(28,28,1)))
# no_training_model에 합성곱 층 Conv2Dmf cnrk
```

In [11]:
```
no_training_conv = no_training_model.layers[0]

print(no_training_conv.weights[0].shape)
# 가중치의 크기 출력
```

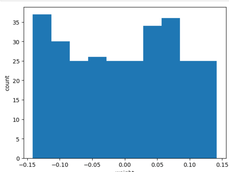
```
(3, 3, 1, 32)
```

In [12]:
```
no_training_weights = no_training_conv.weights[0].numpy()

print(no_training_weights.mean(), no_training_weights.std())
```

```
-0.0031503653 0.08355746
```

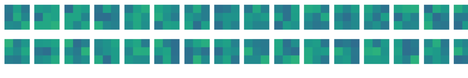
In [13]:
```
plt.hist(no_training_weights.reshape(-1, 1))
plt.xlabel('weight')
plt.ylabel('count')
plt.show()
```



In [14]:
```
fig, axs = plt.subplots(2, 16, figsize=(15,2))

for i in range(2):
    for j in range(16):
        axs[i, j].imshow(no_training_weights[:,:,0,i*16 + j], vmin=-0.5, vmax=0.5)
        axs[i, j].axis('off')

plt.show()
```



함수형 API

In [15]:
```
print(model.input)
```

```
KerasTensor(type_spec=TensorSpec(shape=(None, 28, 28, 1), dtype=tf.float32, name='conv2d_input'), name='conv2d_input', description="created by layer 'conv2d_input'")
```

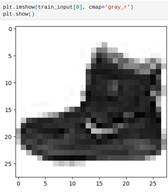
In [16]:
```
conv_acti = keras.Model(model.input, model.layers[0].output) # model.input과 model.layers[0].output을 연결하는 새로운 모델 conv_acti생성
```

특성 맵 시각화

In [17]:
```
(train_input, train_target), (test_input, test_target) = keras.datasets.fashion_mnist.load_data()
# 패션 MNIST 데이터셋을 다운
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

In [18]:
```
plt.imshow(train_input[0], cmap='gray_r')
plt.show()
```



In [19]:
```
inputs = train_input[0:1].reshape(-1, 28, 28, 1)/255.0
# inputs을 슬라이싱 연산자를 사용해 샘플을 사적 주, 크기를 (28,28,1)로 변경하여 255로 나눠서 정규화함

feature_maps = conv_acti.predict(inputs)
# predict()에 inputs을 넣어서 결과를 feature_maps에 저장
```

```
1/1 [==============================] - 7s 7s/step
```

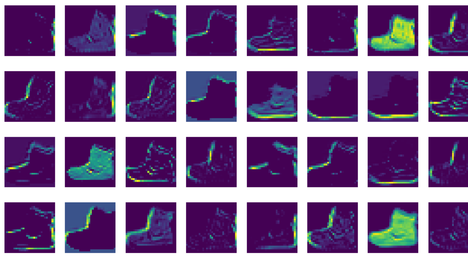
In [20]:
```
print(feature_maps.shape)
```

```
(1, 28, 28, 32)
```

In [21]:
```
fig, axs = plt.subplots(4, 8, figsize=(15,8))

for i in range(4):
    for j in range(8):
        axs[i, j].imshow(feature_maps[0,:,:,i*8 + j])
        axs[i, j].axis('off')

plt.show()
```



In [22]:
```
conv2_acti = keras.Model(model.input, model.layers[2].output)
# model.input과 model.layers[2].output을 연결하는 새로운 모델 conv2_acti 생성
```

In [23]:
```
feature_maps = conv2_acti.predict(train_input[0:1].reshape(-1, 28, 28, 1)/255.0)
```

```
1/1 [==============================] - 0s 122ms/step
```

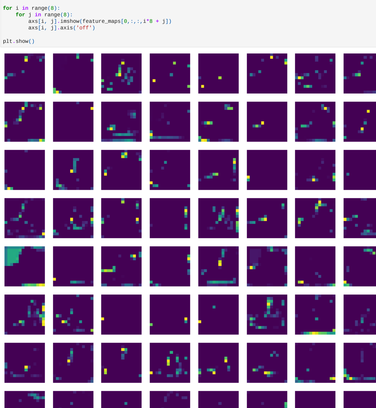
In [24]:
```
print(feature_maps.shape)
```

```
(1, 14, 14, 64)
```

In [25]:
```
fig, axs = plt.subplots(8, 8, figsize=(12,12))

for i in range(8):
    for j in range(8):
        axs[i, j].imshow(feature_maps[0,:,:,i*8 + j])
        axs[i, j].axis('off')

plt.show()
```



In [ ]:
```

```