Week 4

ML/DL General 안태영

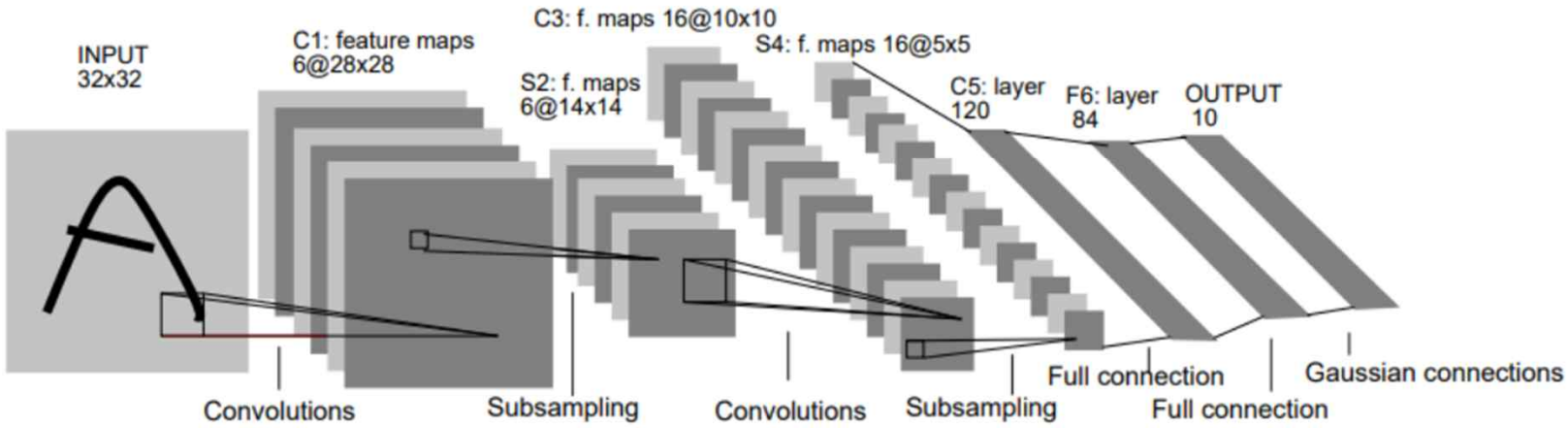
Lecture 11-1: ConvNet의 Conv layer 만들기

History

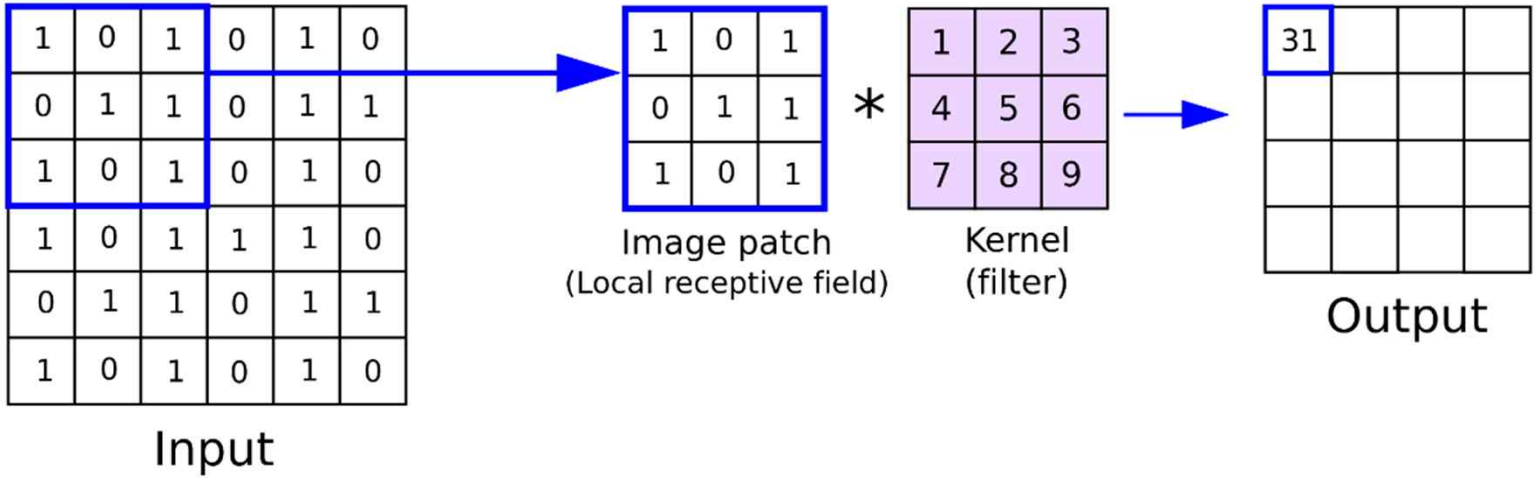
→ 고양이가 사물을 인식할 때 시각 뉴런이 부분적으로 사물을 파악하는 과정에서 착안하여 만들어짐

Convolutional Neural Network (CNN)

- Conv(+ReLU) layer → Pooling layer → Fully Connection 과정으로 진행됨



Convolution Layer

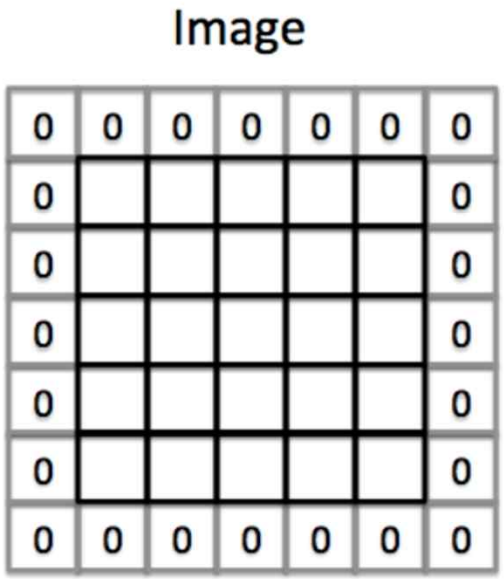


- weigth를 가진 3 X 3 filter를 이동시키면서 하나의 number로 계산시켜 부분적으 학습하는 layer
- $Wx + b$에 ReLU를 바로 적용해서 $ReLU(Wx + b)$로 바로 적용하기도 한다

한 번 연산시, 얻을 수 있는 number 의 개수

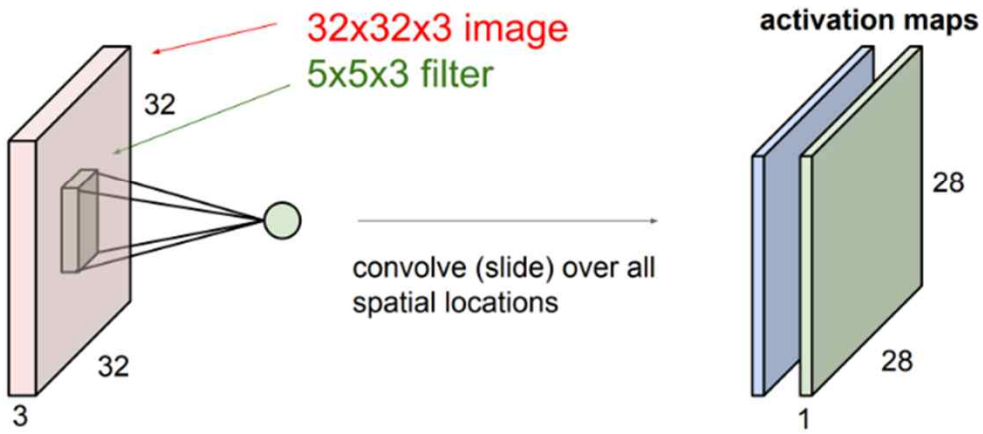
- 위의 예제를 보면 6 X 6 이미지에서 3 X 3 filter을 적용했다
- stride(filter를 움직일때, 몇 칸을 움직이는지에 대한 변수)가 1이다
- (6 - 3)/1 + 1 = 4
- 즉, N X N 형태의 이미지를 F X F 형태의 filter로 stride만큼 이동할때 $\frac{(N-F)}{stride} + 1$이다

Padding



- 계속해서 CONV filter를 적용할 경우 이미지의 크기가 작아진다
- 이를 방지하기 위해서 기존 이미지 테두리에 0을 넣은 다음 연산을 진행시키는 과정을 padding이라고 한다
- ex) 7 X 7 이미지를 1px padding을 시키면 9 X 9 이미지가 출력된다

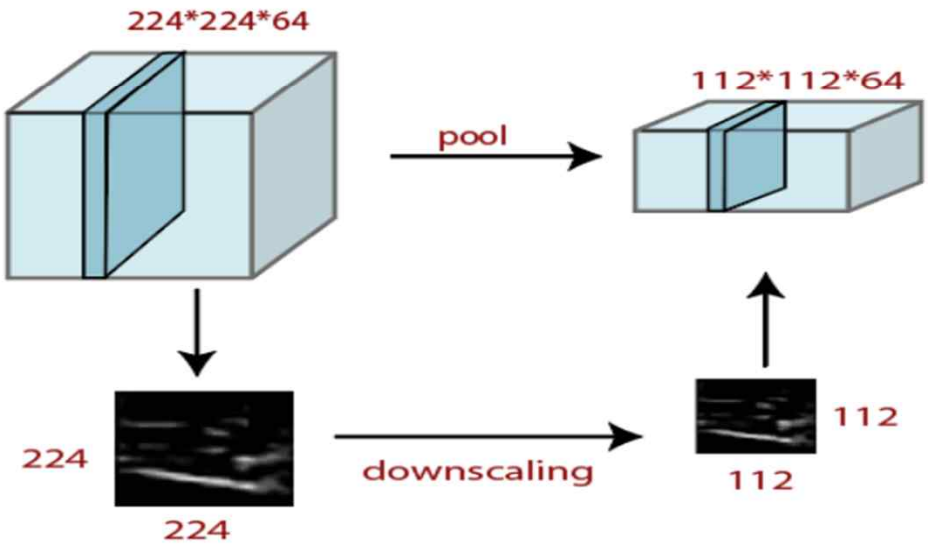
Activation maps



- 여러개의 filter를 만들어서 하나의 이미지에 여러가지 filter를 적용시킨 후 나온 결과 이미지의 집합을 activation map이라고 한다
- 위 예제에서 filter의 개수가 2개이므로 차원은 28 X 28 X 2가 된다
- activation map에서 마지막 차원은 filter의 개수로 결정된다
- Weight 값의 개수는 5 * 5 * 3 * 2 가 된다

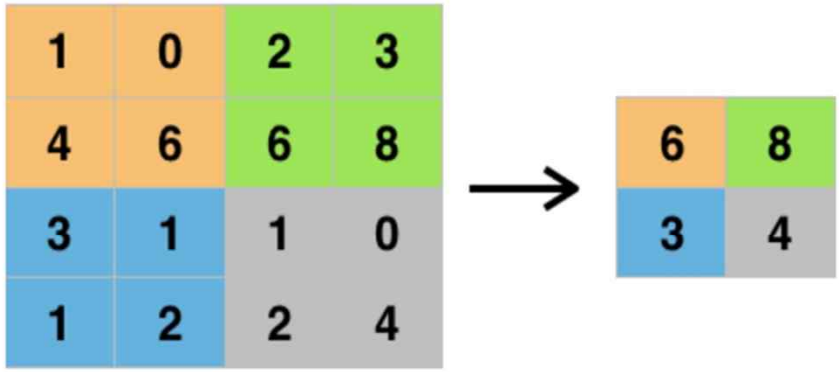
Lecture 11-2: Conv Net Max Pooling과 Fullnetwork

Pooling Layer (Sampling)



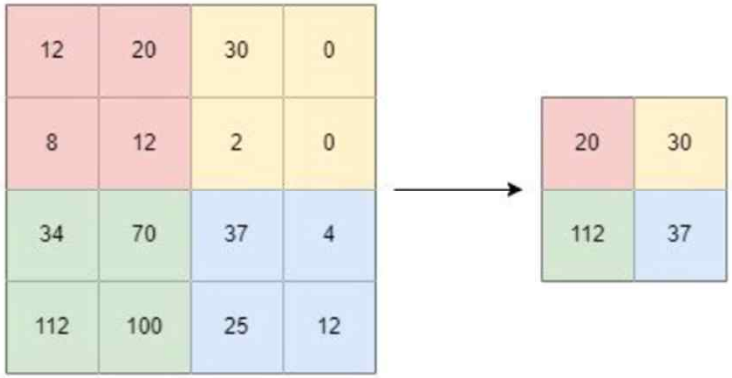
- convolutional layer에서 하나의 layer를 추출해서 resize하고 다시 합치는 과정
- pooling layer를 Sampling 이라고 한다

Max Pooling



- 만약 pooling layer 크기가 2 X 2라고 하면 근처 4개의 값 중에서 가장 큰 값을 추출하는 과정을 max pooling이라고 한다
- 노이즈가 감소하고 영상 분별력이 좋아진다
- 탐색 속도를 높이고자 했지만 이미지 구성 요소간의 공간관계에 대한 정보를 잃는다

Global Average Pooling



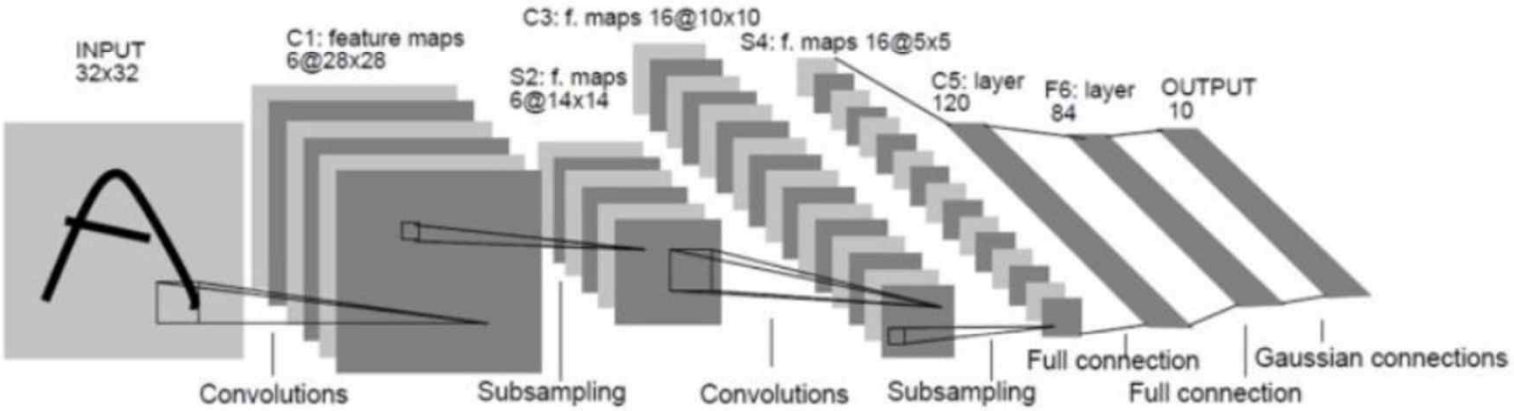
- 각 node의 평균 값을 feature 값으로 취한다
- 계산 속도가 느리다

왜 Min Pooling은 없을까?

- 일반적으로 Activation function은 0이하 값들을 0으로 취급하기 때문에 의미있는 정보만을 남겨서 처리할 수 없게 된다

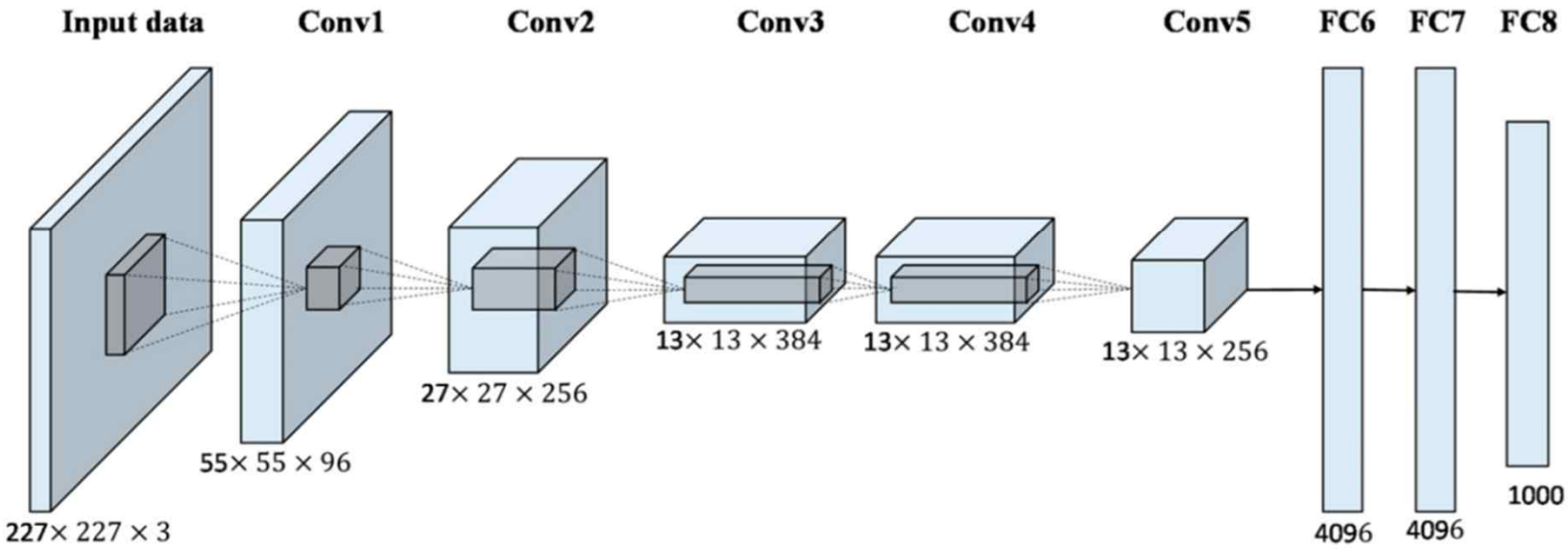
Lectrue 11-3: ConvNet의 활용 예

LeNet-5



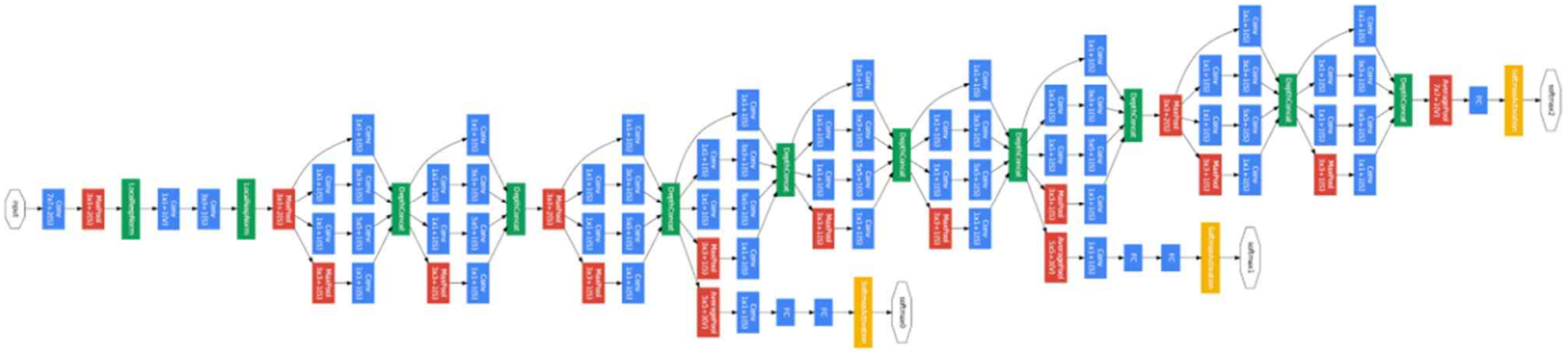
- Lecan: 초기 CNN 창시자
- Conv filter 5 X 5, stride 1
- Subscaling(sampling) 2 X 2, stride 2

AlexNet(2012)



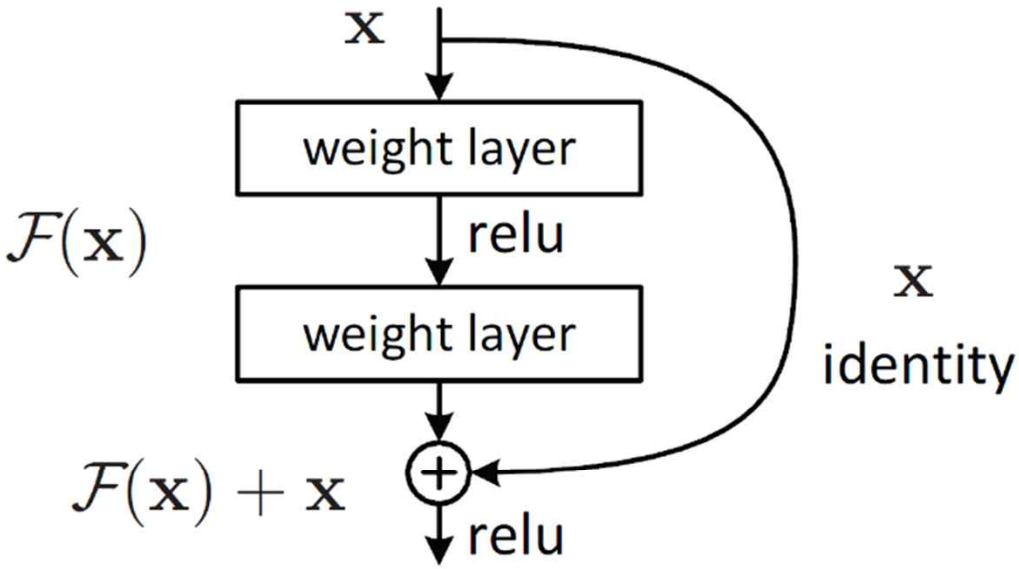
- Input: 227 x 227 X 3
- Conv filter: 96개 11 X 11 X 3, stride 4
- Pooling filter: 3 X 3 filters, stride 2
- Normalization이 사용되었지만 큰 의미는 주지 않았다
- 처음으로 ReLU함수가 개발되었다
- dropout 0.5 , batch 128
- 7 CNN Asssemble

GoogleNet



- 중산중간 convolution layer와 pooling을 한 사진을 다시 합성하여 여러가지 조합

ResNet



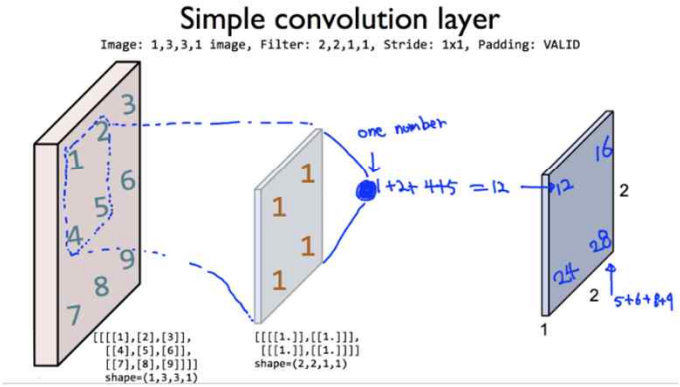
- 중간에 어떤 것은 띄어넘고 연산을 진행해서 연산속도를 증진 시키고 152개 layer를 두면서 정확도를 높였다

Lab 11-0: CNN Basic

Convolution

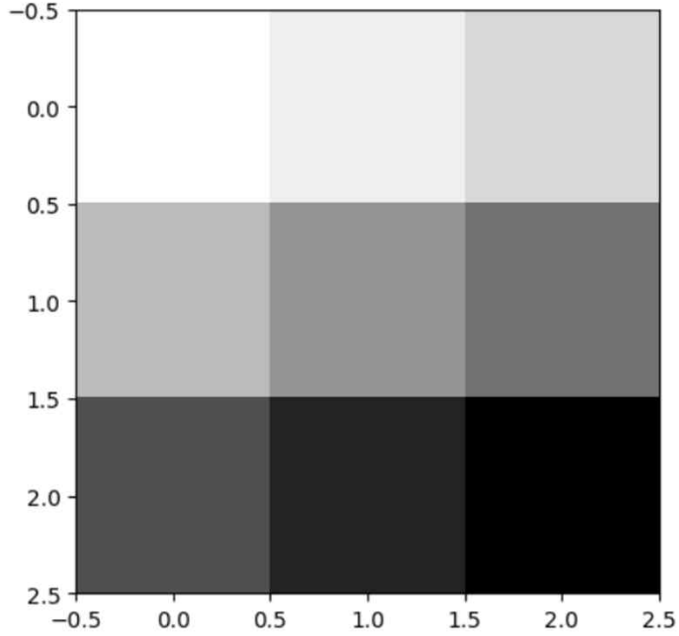
- **conv2d = keras.layers.Conv2D(filters=1, kernel_size=2, padding='VALID',**

```
                kernel_initializer=weight_init)(image)<br><br>
```



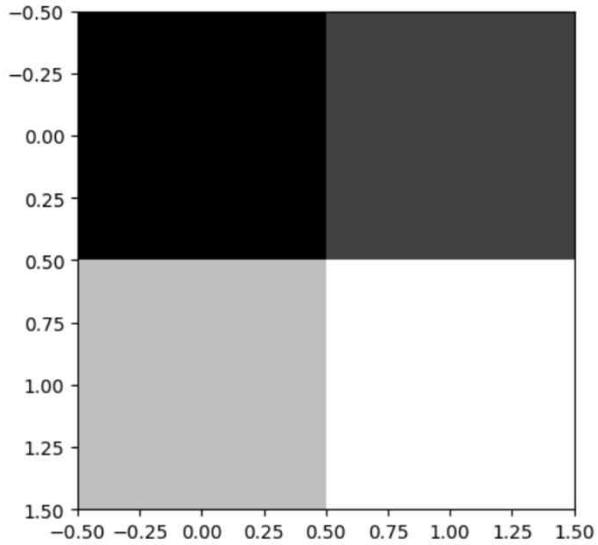

```
image = tf.constant([[[[1],[2],[3]],
                      [[4],[5],[6]],
                      [[7],[8],[9]]]], dtype=np.float32)
print(image.shape)
plt.imshow(image.numpy().reshape(3,3), cmap='Greys')
plt.show()
print("image.shape", image.shape)
weight = np.array([[[[1.]],[[1.]]],
                   [[[1.]],[[1.]]]])
print("weight.shape", weight.shape)
weight_init = tf.constant_initializer(weight)
conv2d = keras.layers.Conv2D(filters=1, kernel_size=2, padding='VALID',
                             kernel_initializer=weight_init)(image)
print("conv2d.shape", conv2d.shape)
print(conv2d.numpy().reshape(2,2))
plt.imshow(conv2d.numpy().reshape(2,2), cmap='gray')
plt.show()
```

(1, 3, 3, 1)




```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 1)
conv2d.shape (1, 2, 2, 1)
[[12. 16.]
 [24. 28.]]
```

```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 1)
conv2d.shape (1, 2, 2, 1)
[[12. 16.]
 [24. 28.]]
```

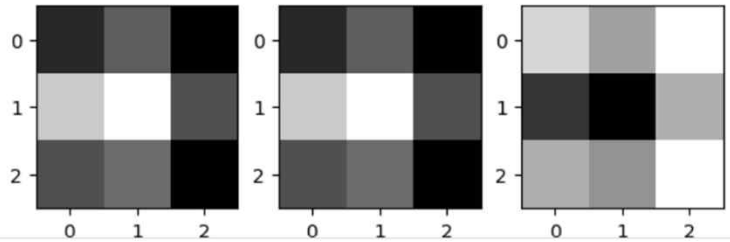


3 filters

- **conv2d = keras.layers.Conv2D(filters=3, kernel_size=2, padding='SAME', kernel_initializer=weight_init)**

```
[ ]  1 # print("imag:₩n", image)
     2 print("image.shape", image.shape)
     3
     4 weight = np.array([[[[1.,10.,-1.]],[[1.,10.,-1.]]],
     5                   [[[1.,10.,-1.]],[[1.,10.,-1.]]]])
     6 print("weight.shape", weight.shape)
     7 weight_init = tf.constant_initializer(weight)
     8 conv2d = keras.layers.Conv2D(filters=3, kernel_size=2, padding='SAME',
     9                              kernel_initializer=weight_init)(image)
    10 print("conv2d.shape", conv2d.shape)
    11 feature_maps = np.swapaxes(conv2d, 0, 3)
    12 for i, feature_map in enumerate(feature_maps):
    13     print(feature_map.reshape(3,3))
    14     plt.subplot(1,3,i+1), plt.imshow(feature_map.reshape(3,3), cmap='gray')
    15 plt.show()
```

```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 3)
conv2d.shape (1, 3, 3, 3)
[[12. 16.  9.]
 [24. 28. 15.]
 [15. 17.  9.]]
[[120. 160.  90.]
 [240. 280. 150.]
 [150. 170.  90.]]
[[-12. -16.  -9.]
 [-24. -28. -15.]
 [-15. -17.  -9.]]
```



Maxpooling

keras.layers.MaxPool2D(pool_size=(2,2), strides=1, padding='VALID')(image)

```
[ ]  1 image = tf.constant([[[[4],[3]],
     2                      [[2],[1]]]], dtype=np.float32)
     3 pool = keras.layers.MaxPool2D(pool_size=(2,2), strides=1, padding='VALID')(image)
     4 print(pool.shape)
     5 print(pool.numpy())
```

```
(1, 1, 1, 1)
[[[[4.]]]]
```

0 Padding

- **pool = keras.layers.MaxPool2D(pool_size=(2,2), strides=1, padding='SAME')**

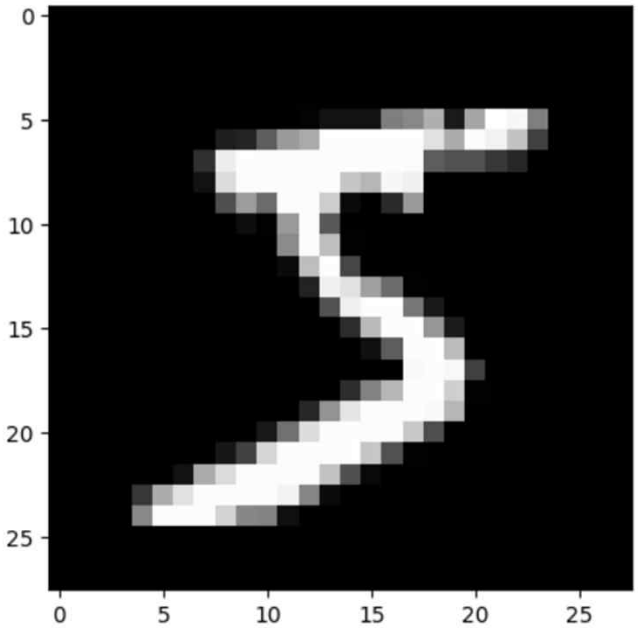
```
[ ]  1 image = tf.constant([[[[4],[3]],
     2                      [[2],[1]]]], dtype=np.float32)
     3 pool = keras.layers.MaxPool2D(pool_size=(2,2), strides=1, padding='SAME')(image)
     4 print(pool.shape)
     5 print(pool.numpy())
```

```
(1, 2, 2, 1)
[[[[4.]
   [3.]]

  [[2.]
   [1.]]]]
```

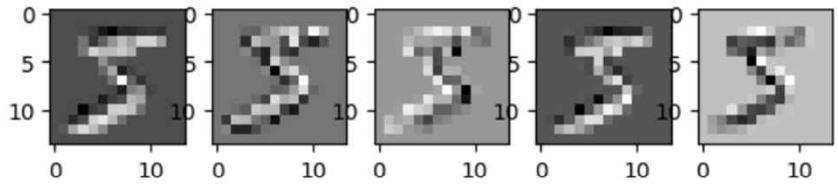
```
[ ]  1 mnist = keras.datasets.mnist
     2 class_names = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
     3 #mnist = keras.datasets.fashion_mnist
     4 #class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
     5 (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
     6 train_images = train_images.astype(np.float32) / 255.
     7 test_images = test_images.astype(np.float32) / 255.
     8 img = train_images[0]
     9 plt.imshow(img, cmap='gray')
    10 plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```



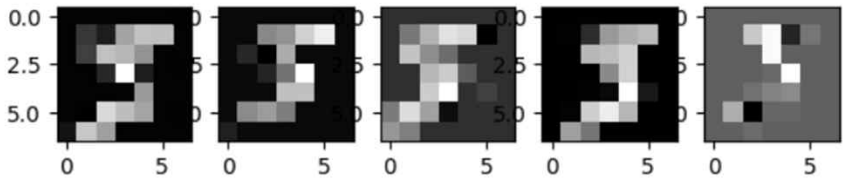
```
[ ]  1 img = img.reshape(-1,28,28,1)
     2 img = tf.convert_to_tensor(img)
     3 weight_init = keras.initializers.RandomNormal(stddev=0.01)
     4 conv2d = keras.layers.Conv2D(filters=5, kernel_size=3, strides=(2, 2), padding='SAME',
     5                              kernel_initializer=weight_init)(img)
     6 print(conv2d.shape)
     7 feature_maps = np.swapaxes(conv2d, 0, 3)
     8 for i, feature_map in enumerate(feature_maps):
     9     plt.subplot(1,5,i+1), plt.imshow(feature_map.reshape(14,14), cmap='gray')
    10 plt.show()
```

```
(1, 14, 14, 5)
```



```
[ ]  1 pool = keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='SAME')(conv2d)
     2 print(pool.shape)
     3
     4 feature_maps = np.swapaxes(pool, 0, 3)
     5 for i, feature_map in enumerate(feature_maps):
     6     plt.subplot(1,5,i+1), plt.imshow(feature_map.reshape(7, 7), cmap='gray')
     7 plt.show()
```

```
(1, 7, 7, 5)
```



Lab 11-1: Mnist cnn keras sequential eager

→ 기존과 달라진건 함수 CNN 정의만 달라졌다

```
[ ]
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.utils import to_categorical
import numpy as np
import matplotlib.pyplot as plt
import os

```

```
[ ]
learning_rate = 0.001
training_epochs = 15
batch_size = 100

tf.random.set_seed(777)
cur_dir = os.getcwd()
ckpt_dir_name = 'checkpoints'
model_dir_name = 'minst_cnn_seq'

checkpoint_dir = os.path.join(cur_dir, ckpt_dir_name, model_dir_name)
os.makedirs(checkpoint_dir, exist_ok=True)

checkpoint_prefix = os.path.join(checkpoint_dir, model_dir_name)
```

```python
mnist = keras.datasets.mnist
class_names = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

train_images = train_images.astype(np.float32) / 255.
test_images = test_images.astype(np.float32) / 255.
train_images = np.expand_dims(train_images, axis=-1)
test_images = np.expand_dims(test_images, axis=-1)

train_labels = to_categorical(train_labels, 10)
test_labels = to_categorical(test_labels, 10)

train_dataset = tf.data.Dataset.from_tensor_slices((train_images, train_labels)).shuffle(
                buffer_size=100000).batch(batch_size)
test_dataset = tf.data.Dataset.from_tensor_slices((test_images, test_labels)).batch(batch_size)
```

```python
def create_model():
    model = keras.Sequential()
    model.add(keras.layers.Conv2D(filters=32, kernel_size=3, activation=tf.nn.relu, padding='SAME',
                                  input_shape=(28, 28, 1)))
    model.add(keras.layers.MaxPool2D(padding='SAME'))
    model.add(keras.layers.Conv2D(filters=64, kernel_size=3, activation=tf.nn.relu, padding='SAME'))
    model.add(keras.layers.MaxPool2D(padding='SAME'))
    model.add(keras.layers.Conv2D(filters=128, kernel_size=3, activation=tf.nn.relu, padding='SAME'))
    model.add(keras.layers.MaxPool2D(padding='SAME'))
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(256, activation=tf.nn.relu))
    model.add(keras.layers.Dropout(0.4))
    model.add(keras.layers.Dense(10))
    return model

model = create_model()
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 28, 28, 32)        320

 max_pooling2d_3 (MaxPoolin  (None, 14, 14, 32)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_4 (MaxPoolin  (None, 7, 7, 64)          0
 g2D)

 conv2d_5 (Conv2D)           (None, 7, 7, 128)         73856

 max_pooling2d_5 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 256)               524544

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 619786 (2.36 MB)
Trainable params: 619786 (2.36 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Lab 11-2: Mnist model function eager

→ conv와 pooling layer를 직접 만들고 층을 쌓는 부분만 다르다

```
[ ]
def create_model():
    inputs = keras.Input(shape=(28, 28, 1))
    conv1 = keras.layers.Conv2D(filters=32, kernel_size=[3, 3], padding='SAME', activation=tf.nn.relu)(inputs)
    pool1 = keras.layers.MaxPool2D(padding='SAME')(conv1)
    conv2 = keras.layers.Conv2D(filters=64, kernel_size=[3, 3], padding='SAME', activation=tf.nn.relu)(pool1)
    pool2 = keras.layers.MaxPool2D(padding='SAME')(conv2)
    conv3 = keras.layers.Conv2D(filters=128, kernel_size=[3, 3], padding='SAME', activation=tf.nn.relu)(pool2)
    pool3 = keras.layers.MaxPool2D(padding='SAME')(conv3)
    pool3_flat = keras.layers.Flatten()(pool3)
    dense4 = keras.layers.Dense(units=256, activation=tf.nn.relu)(pool3_flat)
    drop4 = keras.layers.Dropout(rate=0.4)(dense4)
    logits = keras.layers.Dense(units=10)(drop4)
    return keras.Model(inputs=inputs, outputs=logits)
```

Lab 11-3: mnist cnn keras subclassing eager

→ Model subscaling으로 class를 정의해서 사용한다

```
[ ]
class MNISTModel(tf.keras.Model):
    def __init__(self):
        super(MNISTModel, self).__init__()
        self.conv1 = keras.layers.Conv2D(filters=32, kernel_size=[3, 3], padding='SAME', activation=tf.nn.relu)
        self.pool1 = keras.layers.MaxPool2D(padding='SAME')
        self.conv2 = keras.layers.Conv2D(filters=64, kernel_size=[3, 3], padding='SAME', activation=tf.nn.relu)
        self.pool2 = keras.layers.MaxPool2D(padding='SAME')
        self.conv3 = keras.layers.Conv2D(filters=128, kernel_size=[3, 3], padding='SAME', activation=tf.nn.relu)
        self.pool3 = keras.layers.MaxPool2D(padding='SAME')
        self.pool3_flat = keras.layers.Flatten()
        self.dense4 = keras.layers.Dense(units=256, activation=tf.nn.relu)
        self.drop4 = keras.layers.Dropout(rate=0.4)
        self.dense5 = keras.layers.Dense(units=10)
    def call(self, inputs, training=False):
        net = self.conv1(inputs)
        net = self.pool1(net)
        net = self.conv2(net)
        net = self.pool2(net)
        net = self.conv3(net)
        net = self.pool3(net)
        net = self.pool3_flat(net)
        net = self.dense4(net)
        net = self.drop4(net)
        net = self.dense5(net)
        return net
```

Lab 11-4: mnist cnn keras ensemble eager

→ model을 여러개 만들어서 하나의 결과물을 출력해주는 과정

→ modeling과 evluation 하는 부분만 바뀐다

```
[ ]
class MNISTModel(tf.keras.Model):
    def __init__(self):
        super(MNISTModel, self).__init__()
        self.conv1 = keras.layers.Conv2D(filters=32, kernel_size=[3, 3], padding='SAME', activation=tf.nn.relu)
        self.pool1 = keras.layers.MaxPool2D(padding='SAME')
        self.conv2 = keras.layers.Conv2D(filters=64, kernel_size=[3, 3], padding='SAME', activation=tf.nn.relu)
        self.pool2 = keras.layers.MaxPool2D(padding='SAME')
        self.conv3 = keras.layers.Conv2D(filters=128, kernel_size=[3, 3], padding='SAME', activation=tf.nn.relu)
        self.pool3 = keras.layers.MaxPool2D(padding='SAME')
        self.pool3_flat = keras.layers.Flatten()
        self.dense4 = keras.layers.Dense(units=256, activation=tf.nn.relu)
        self.drop4 = keras.layers.Dropout(rate=0.4)
        self.dense5 = keras.layers.Dense(units=10)
    def call(self, inputs, training=False):
        net = self.conv1(inputs)
        net = self.pool1(net)
        net = self.conv2(net)
        net = self.pool2(net)
        net = self.conv3(net)
        net = self.pool3(net)
        net = self.pool3_flat(net)
        net = self.dense4(net)
        net = self.drop4(net)
        net = self.dense5(net)
        return net
models = []
num_models = 3
for m in range(num_models):
    models.append(MNISTModel())


```

```
def evaluate(models, images, labels):
    predictions = np.zeros_like(labels)
    for model in models:
        logits = model(images, training=False)
        predictions += logits
    correct_prediction = tf.equal(tf.argmax(predictions, 1), tf.argmax(labels, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    return accuracy
```

Lab 11-5: mnist cnn best keras eager

> **Data Argumentation**
>
> - 주어진 data를 가지고 rotate, shift 등을 이용해서 data 개수를 늘리는 방법

```
[ ]
```

```python
mnist = keras.datasets.mnist
class_names = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
def data_augmentation(images, labels):
    aug_images = []
    aug_labels = []

    for x, y in zip(images, labels):
        aug_images.append(x)
        aug_labels.append(y)

        bg_value = np.median(x)

        for _ in range(4):
            angle = np.random.randint(-15, 15, 1)
            ###rotate
            rot_img = ndimage.rotate(x, angle[0], reshape=False, cval=bg_value)

            ###shift
            shift = np.random.randint(-2, 2, 2)
            shift_img = ndimage.shift(rot_img, shift, cval=bg_value)

            aug_images.append(shift_img)
            aug_labels.append(y)
    aug_images = np.array(aug_images)
    aug_labels = np.array(aug_labels)
    return aug_images, aug_labels
```

Batch Normalization

→ dense와 그냥 conv layer를 나중에 합치는 형태로 진행하기 때문에 따로 class를 정의한다

```
]  1 class DenseBNRelu(tf.keras.Model):
   2     def __init__(self, units):
   3         super(DenseBNRelu, self).__init__()
   4         self.dense = keras.layers.Dense(units=units, kernel_initializer='glorot_normal')
   5         self.batchnorm = tf.keras.layers.BatchNormalization()
   6     def call(self, inputs, training=False):
   7         layer = self.dense(inputs)
   8         layer = self.batchnorm(layer)
   9         layer = tf.nn.relu(layer)
  10         return layer
```

```
]  1 class ConvBNRelu(tf.keras.Model):
   2     def __init__(self, filters, kernel_size=3, strides=1, padding='SAME'):
   3         super(ConvBNRelu, self).__init__()
   4         self.conv = keras.layers.Conv2D(filters=filters, kernel_size=kernel_size, strides=strides,
   5                                         padding=padding, kernel_initializer='glorot_normal')
   6         self.batchnorm = tf.keras.layers.BatchNormalization()
   7     def call(self, inputs, training=False):
   8         layer = self.conv(inputs)
   9         layer = self.batchnorm(layer)
  10         layer = tf.nn.relu(layer)
  11         return layer
```