

Colab Research Interface showing a Jupyter Notebook titled "Lecture5-4.ipynb". The code cell [4] defines a 2D convolution operation:

```
[4] N, n_H, n_W, n_C = 1, 28, 28, 5
    n_filter = 1
    k_size = 3

    images = tf.random.uniform(minval=0, maxval=1,
                               shape=(N, n_H, n_W, n_C))

    conv = Conv2D(filters=n_filter, kernel_size=k_size)

    y = conv(images)

    W, B = conv.get_weights()

    print(images.shape)
    print(W.shape)
    print(B.shape)
    print(y.shape)
```

Handwritten annotations illustrate the process:

- A 3D cube representing the input image with dimensions $28 \times 28 \times 5$.
- A 3D cube representing the kernel with dimensions $3 \times 3 \times 1$.
- A 2D square representing the output feature map with dimensions $26 \times 26 \times 1$.

The output shapes are printed as:

```
(1, 28, 28, 5)
(3, 3, 5, 1)
(1, 1)
(1, 26, 26, 1)
```

Code cell [5] is titled "Code.5-1-2: Correlation Calculation" and is currently empty.

Code cell [6] is titled "Code.5-1-3: Correlation with n-channel" and is currently empty.

window가 28×28 이미지 에서 앞에서부터 쪽 연산을하고 다 끝나면 한장의 이미지가 나온다

Colab Research Interface showing a Jupyter Notebook titled "Lecture5-4.ipynb". The code cell [5] defines a 2D convolution operation with 10 filters:

```
[5] import tensorflow as tf

    from tensorflow.keras.layers import Conv2D

    N, n_H, n_W, n_C = 1, 28, 28, 5
    n_filter = 10
    k_size = 3

    images = tf.random.uniform(minval=0, maxval=1,
                               shape=(N, n_H, n_W, n_C))

    conv = Conv2D(filters=n_filter, kernel_size=k_size)

    y = conv(images)

    W, B = conv.get_weights()

    print(images.shape)
    print(W.shape)
    print(B.shape)
    print(y.shape)
```

Handwritten annotations illustrate the process:

- A 3D cube representing the input image with dimensions $28 \times 28 \times 5$.
- A 3D cube representing the kernel with dimensions $3 \times 3 \times 5$.
- A 2D square representing the output feature map with dimensions $26 \times 26 \times 10$.

The output shapes are printed as:

```
(1, 28, 28, 5)
(3, 3, 5, 10)
(10, 1)
(1, 26, 26, 10)
```

Code cell [6] is titled "Code.5-1-2: Correlation Calculation" and is currently empty.

kernel size가 10 이라고 가정하자 28×28 이미지와 filter size 10개가 만들어지고 커널하나당 bias를 가지게 되고 각 filter당 한장의 이미지가 연산되어 output으로 나오게되고 최종 10장의 이미지가 나온다

Chrome File Edit View History Bookmarks People Tab Window Help

colab.research.google.com/drive/7nxesn5Dty7ge28BjssCFXu2yrcHzf#scrollTo=d-EyG1TJ6f

Lecture5-4.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

```
import tensorflow as tf

from tensorflow.keras.layers import Conv2D

N, n_H, n_W, n_C = 1, 5, 5, 3
n_filter = 1
k_size = 3

images = tf.random.uniform(minval=0, maxval=1,
                           shape=(N, n_H, n_W, n_C))

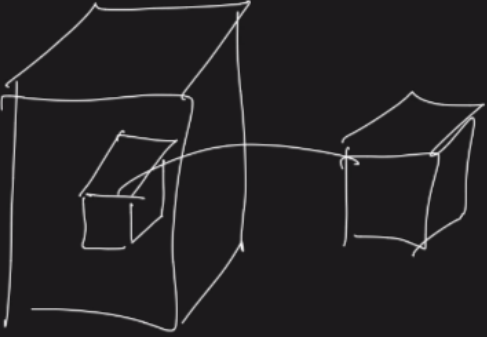
conv = Conv2D(filters=n_filter, kernel_size=k_size)

y = conv(images)
print("Y(Tensorflow): \n", y.numpy().squeeze())
W, B = conv.get_weights()

#####
images = images.numpy().squeeze()
W = W.squeeze()

y_man = np.zeros(shape=(n_H - k_size + 1, n_W - k_size + 1))
for i in range(n_H - k_size + 1):
    for j in range(n_W - k_size + 1):
        window = images[i : i+k_size, j : j+k_size]
        y_man[i, j] = np.sum(window*W) + B

print("Y(Manual): \n", y_man)
```



RAM 디스크 수정 가능