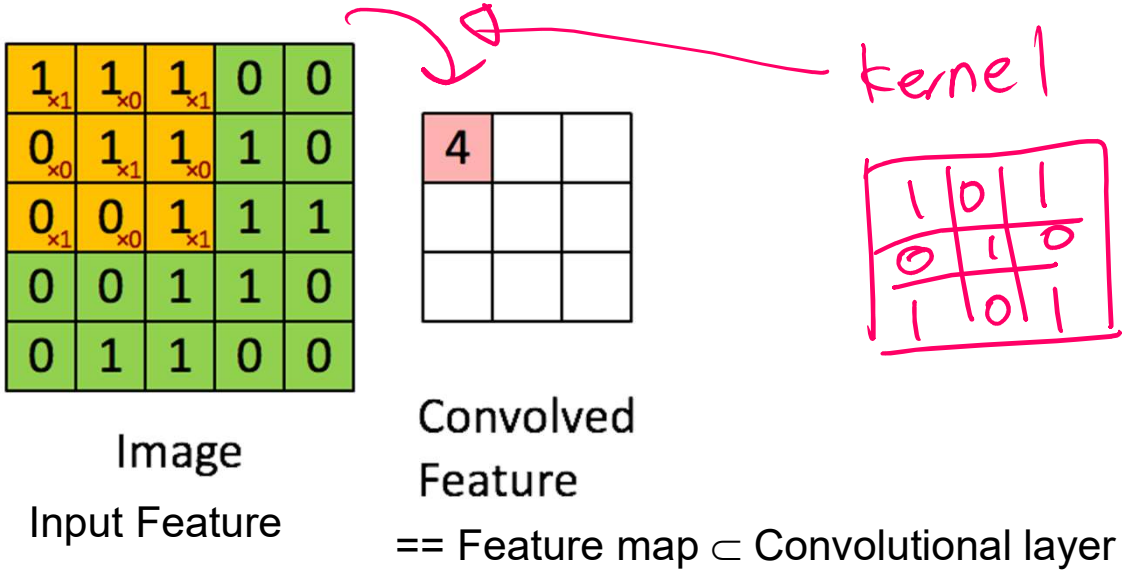
Deep Computer Vision Using Convolutional Neural Networks (CNN)

Saehwa Kim

Information and Communications Engineering
Hankuk University of Foreign Studies

Contents

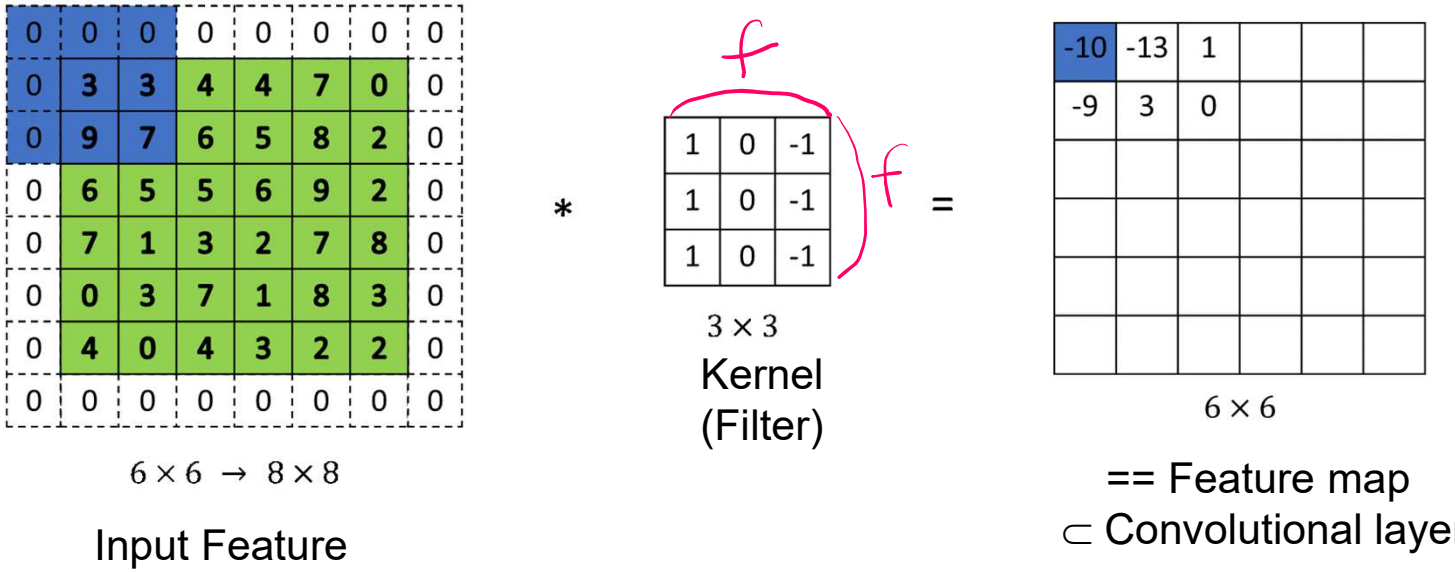
Convolution of Features with Kernels



출처: http://engineering.flipboard.com/2015/05/scaling-convnets

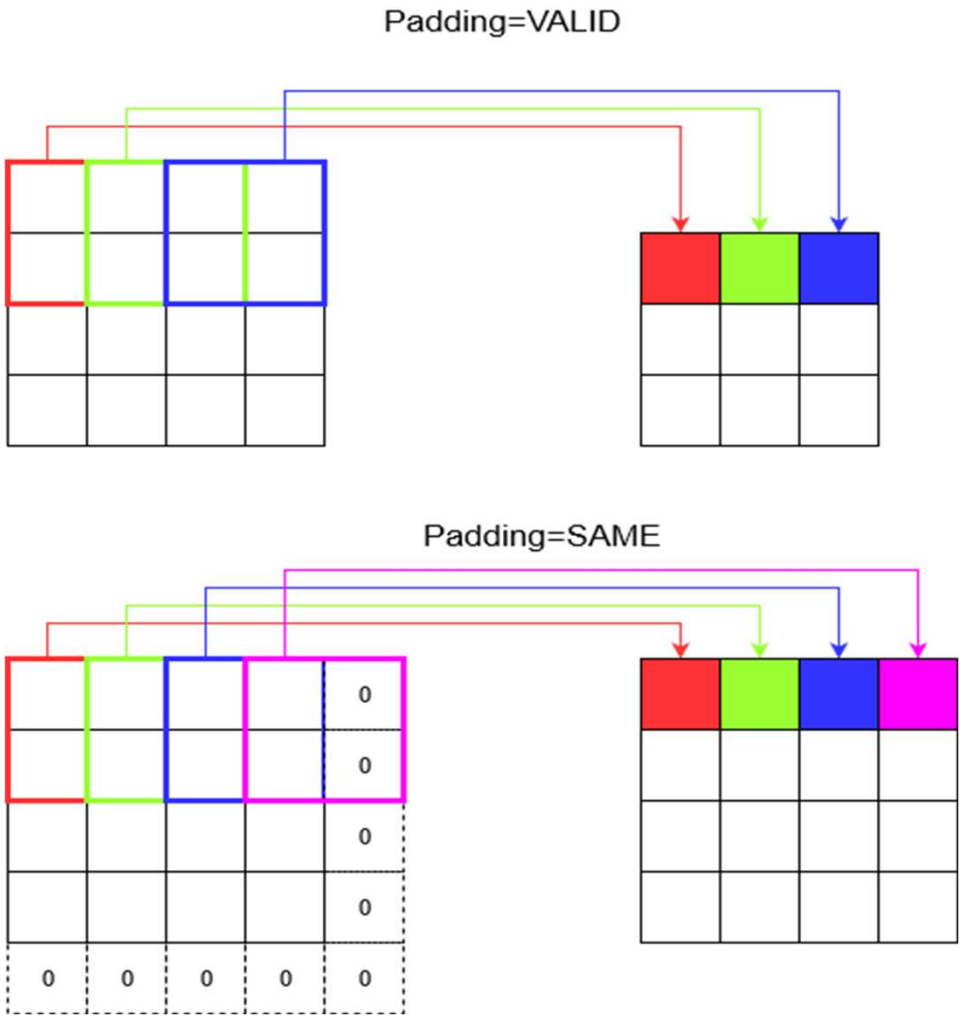
Padding (여백 채우기)

- Valid padding (유효 채우기)
  - 줄어들 게 놔두는 것
- Half padding (== Same padding, Zero padding) → 이게 가장 일반적
  - Notation: kernel size = f x f
  - Input feature의 각 가장 자리에 (f – 1)/2 픽셀 두께로 픽셀을 채운 뒤(값은 0) convolution 수행
    - Convolution에 의해 f -1 만큼 줄어들기 때문임



출처: http://datahacker.rs/what-is-padding-cnn/

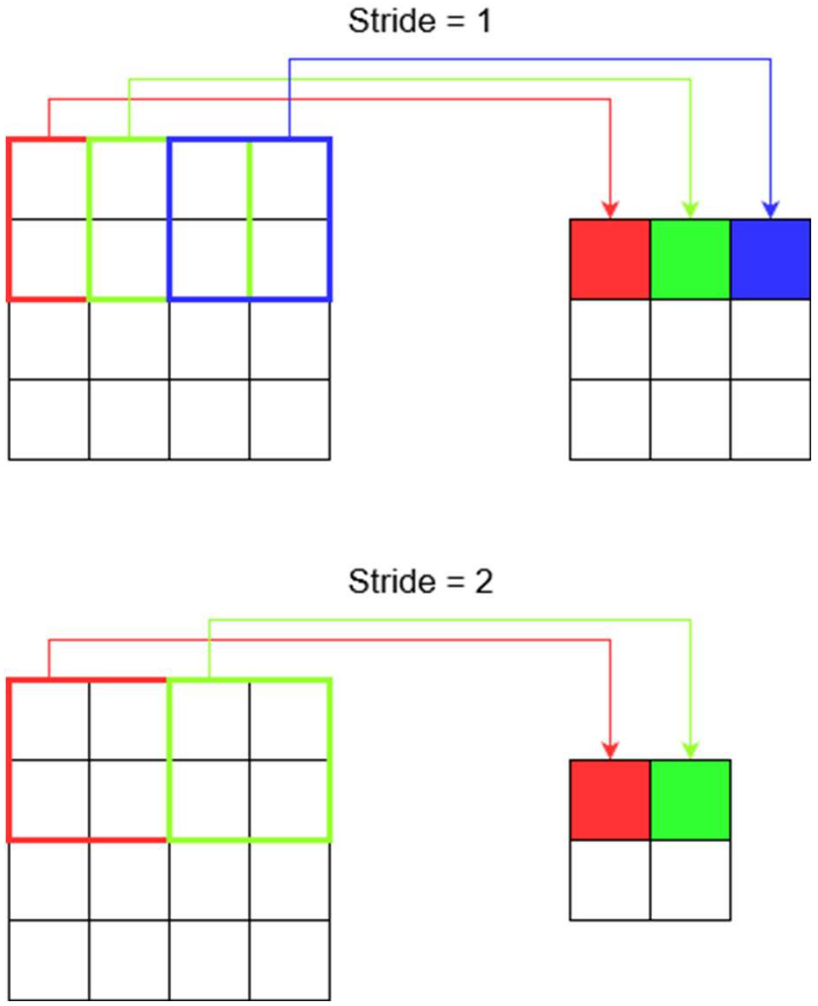
Padding (여백 채우기): Filter 길이가 짝수인 경우



출처: https://stackoverflow.com/questions/42883547/intuitive-understanding-of-1d-2d-and-3d-convolutions-in-convolutional-neural-n

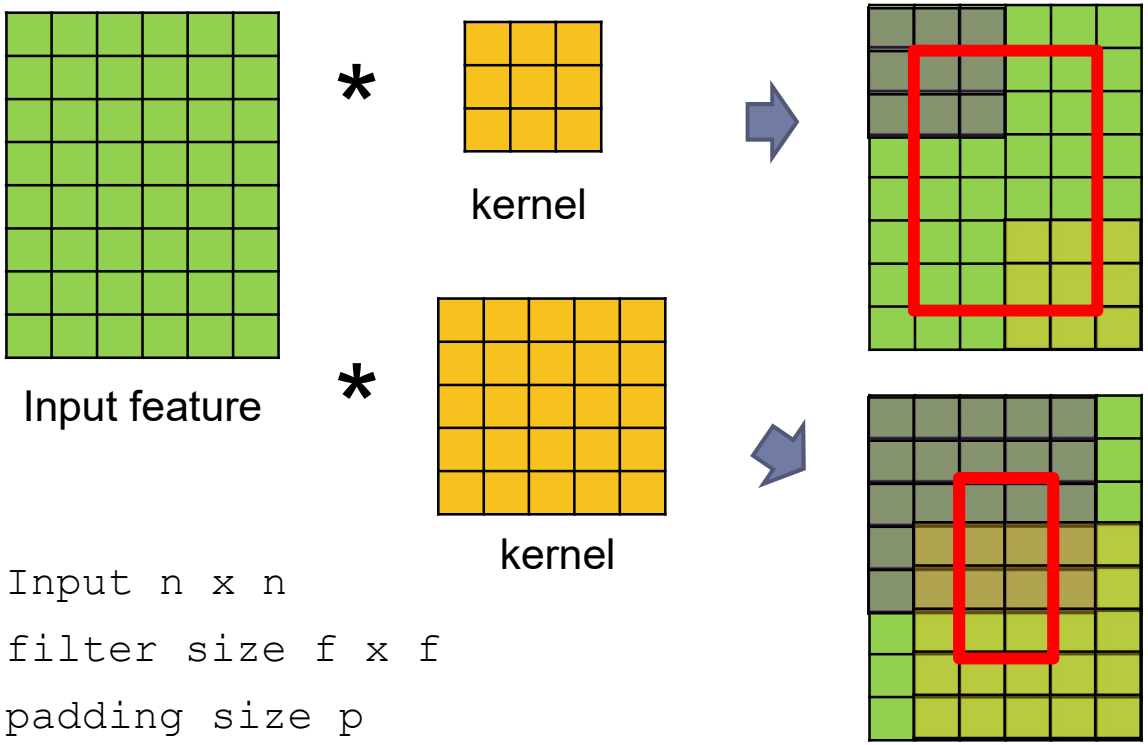
Stride (보폭)

- 연속적인 Convolution 연산에서 kernel이 이동하는 칸 수
- 가로 보폭 x 세로 보폭
  - 일반적으로 두 개가 같음
- 일반적으로 1을 가장 많이 사용 (1x1)



출처: https://stackoverflow.com/questions/42883547/intuitive-understanding-of-1d-2d-and-3d-convolutions-in-convolutional-neural-n

Output Size after Convolution



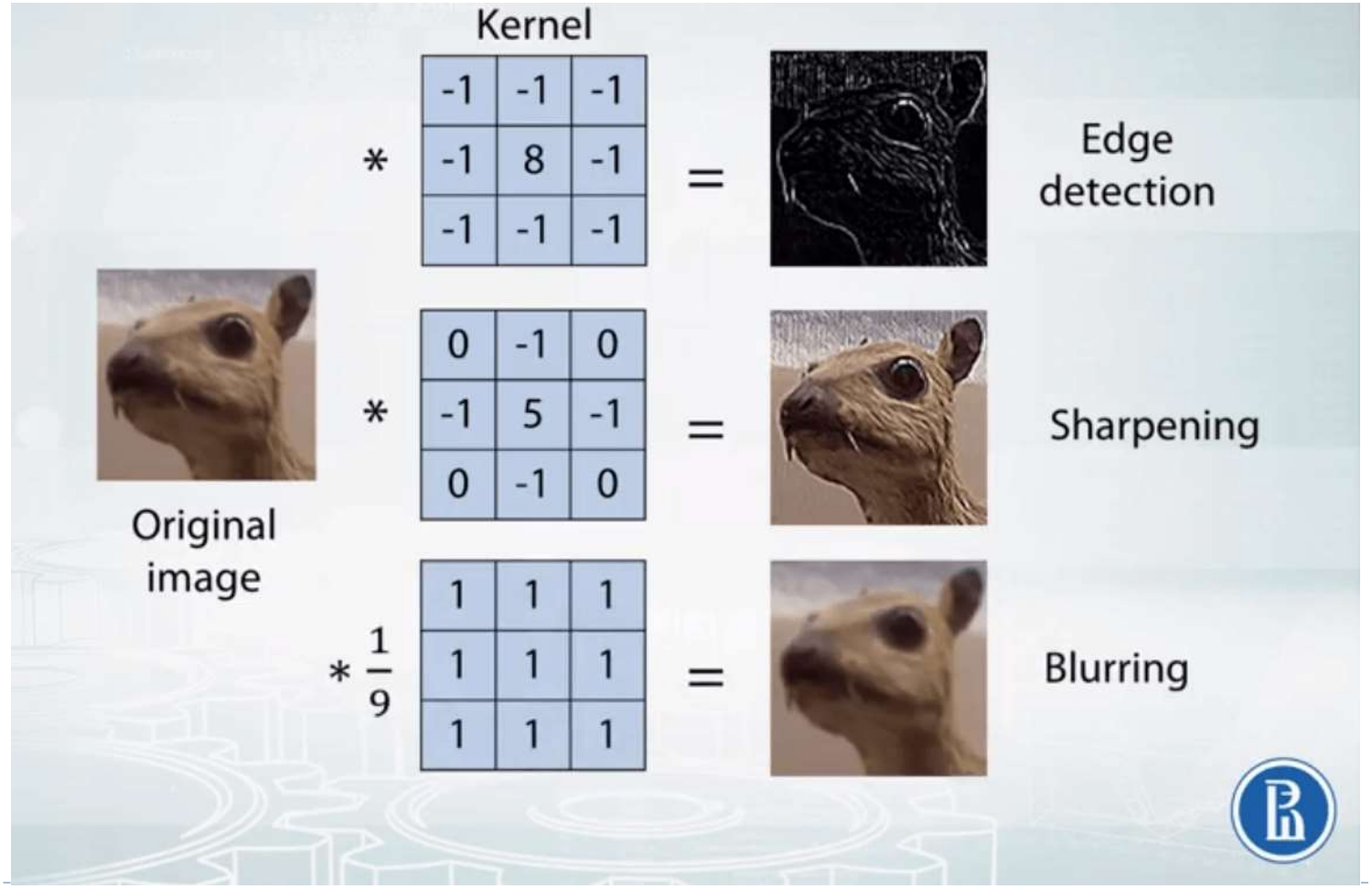
빨간 윤곽선이 output feature map

(convolution 한번에 하나의 값 나옴: kernel의 정 중앙 위치에 대응)

- Input n x n
- filter size f x f
- padding size p
- stride s x s
- → Output size is {(n+2p-f)/s +1}^2
- How many padding for the same output size as the input?
  - (n + 2p - f)/s + 1 = n
  - p = {(n - 1)*s + f - n}/2 = (f - 1)/2 # if s == 1

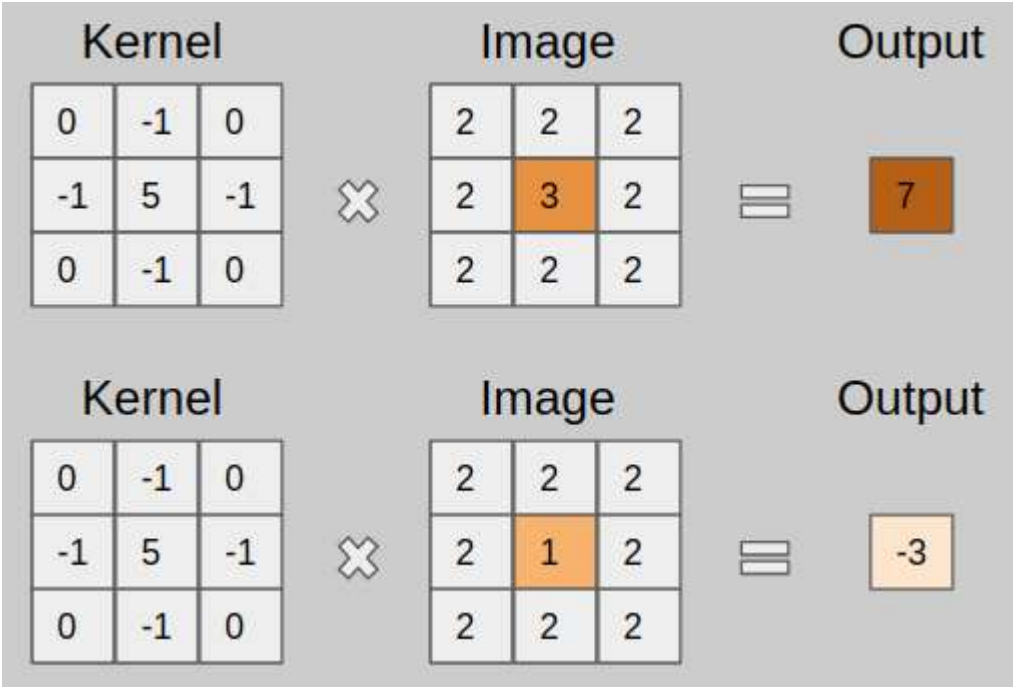
Various Kernels

- 출처: https://www.theyoungtechie.com/cm/neural-network-and-deep-learning/courses/advancedmachinelearningspecialization/introduction-to-deeplearning-aml_week3/introduction-to-cnn/



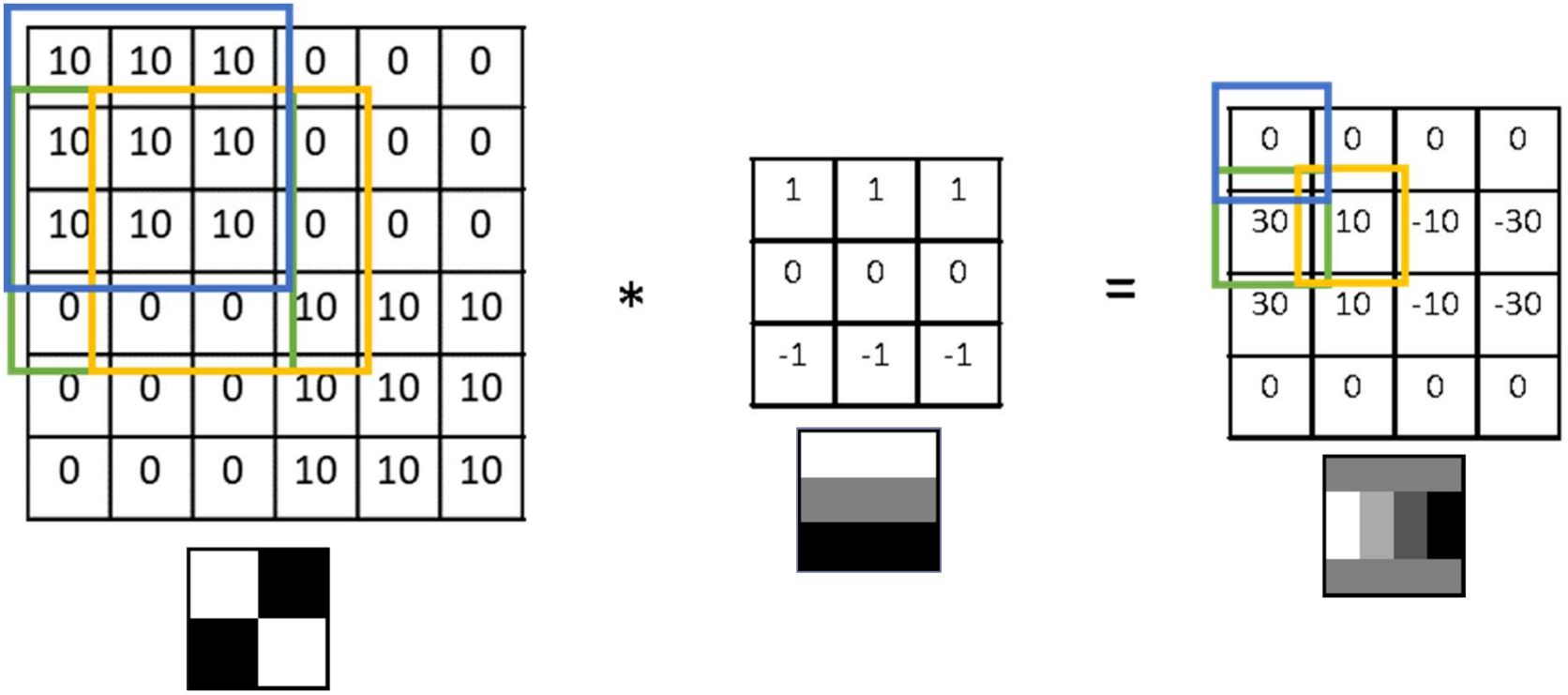
How a Sharpening Kernel Works

- 출처: https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37
  - The value 3 increased to 7
  - The value 1 decreased to -3



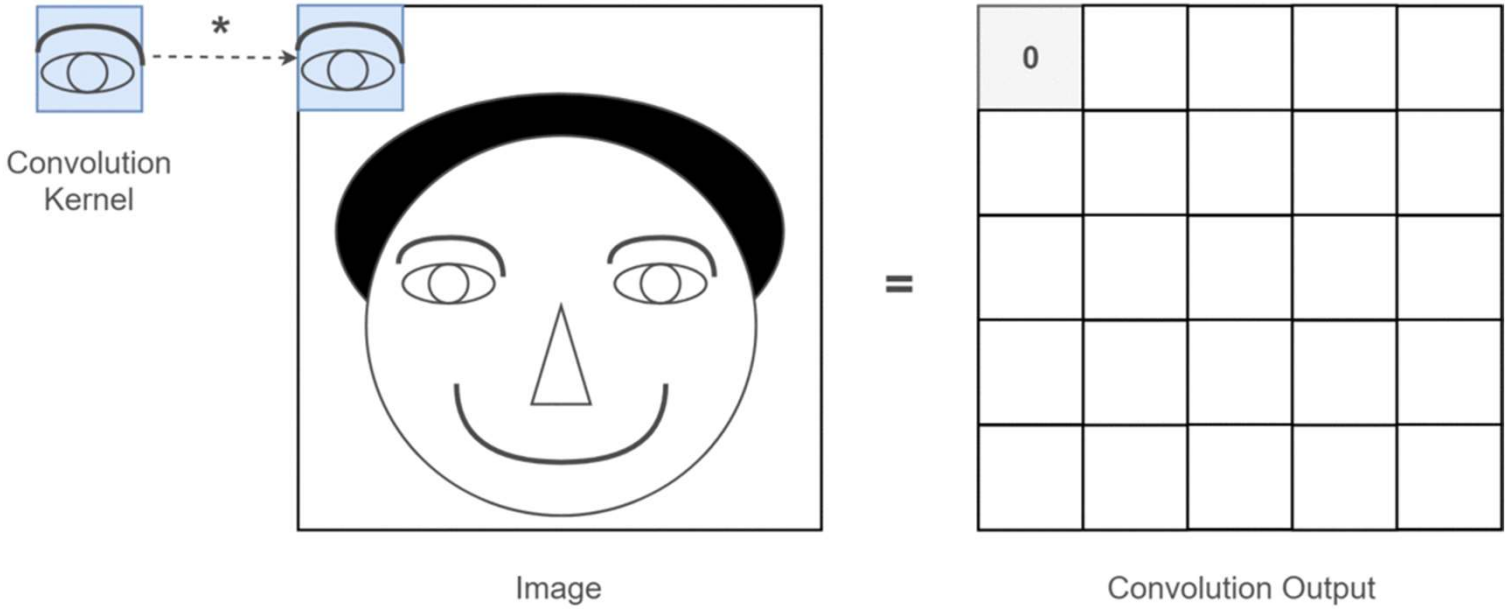
How a Horizontal Edge Detection Kernel Works

- http://datahacker.rs/edge-detection-extended/



Another Example

- 출처: https://stackoverflow.com/questions/42883547/intuitive-understanding-of-1d-2d-and-3d-convolutions-in-convolutional-neural-n



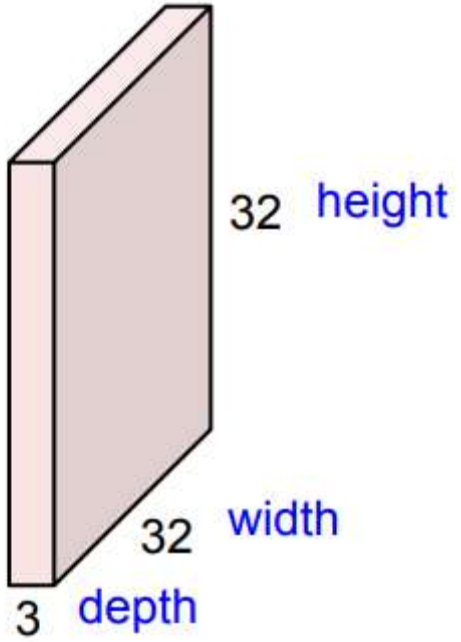
[ksaehwa] depth: channel

Convolutional Layer: Filter

Filters always extend the full depth of the input volume

32x32x**3** input (image)

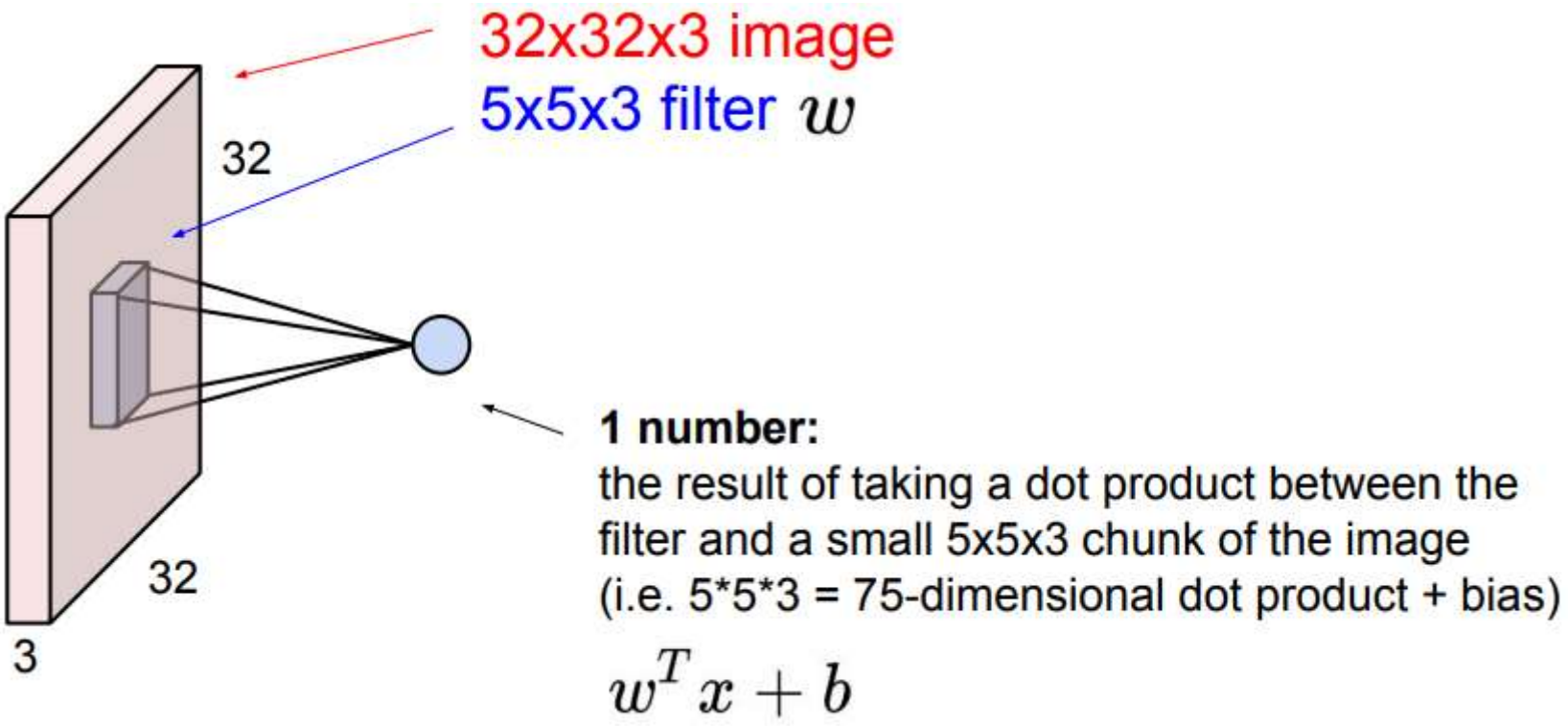
5x5x**3** filter



Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

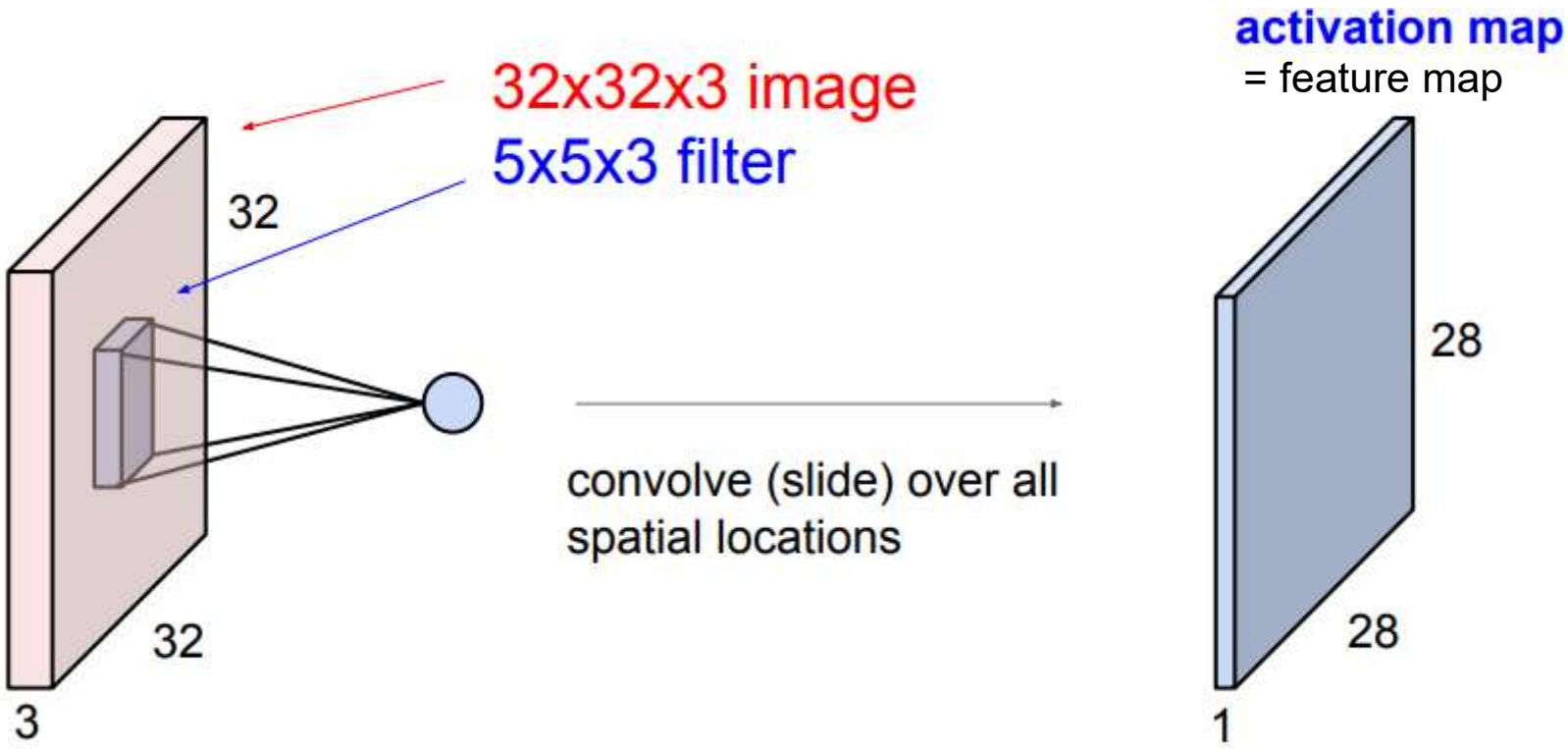
출처: https://web.stanford.edu/class/cs20si/2017/lectures/slides_06.pdf

Convolutional Layer



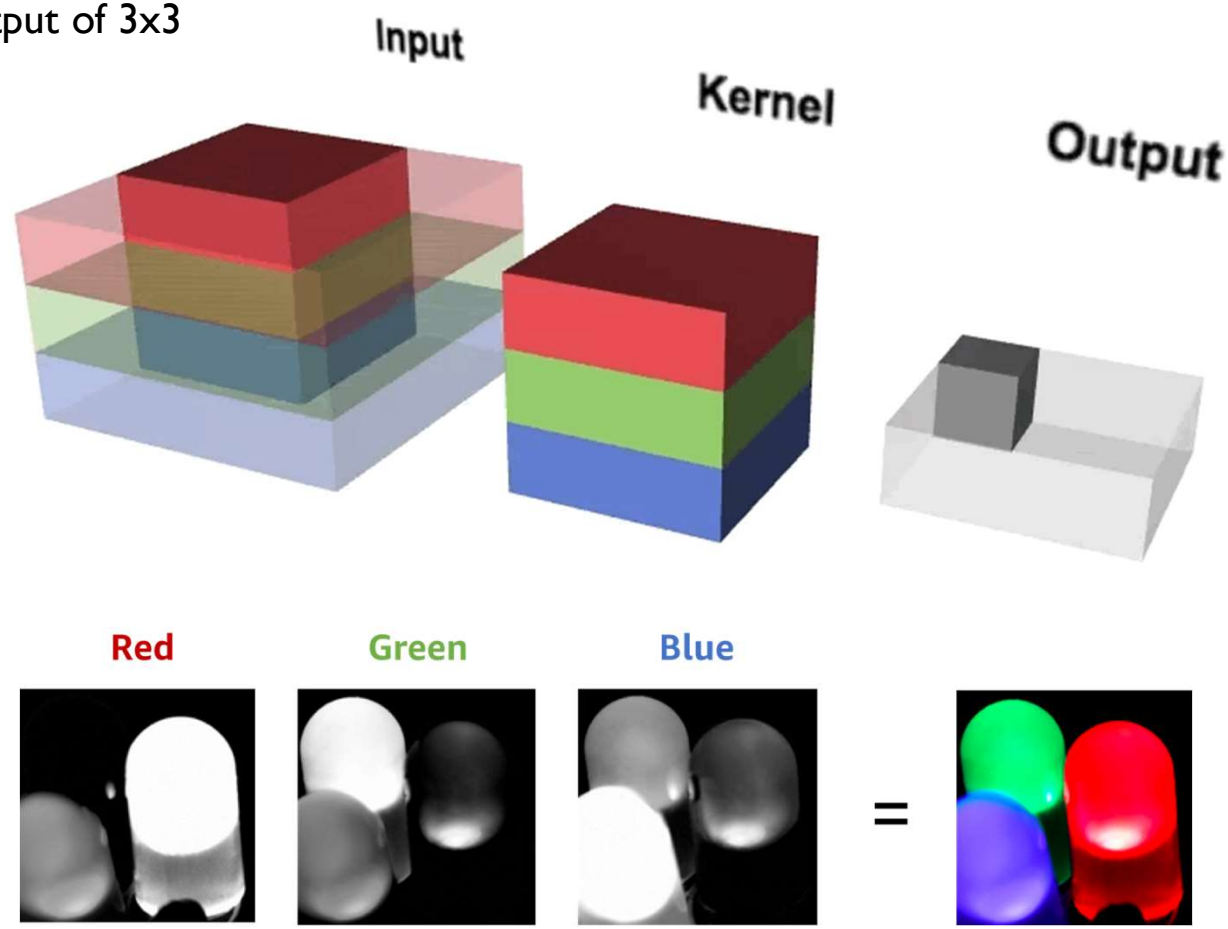
출처: https://web.stanford.edu/class/cs20si/2017/lectures/slides_06.pdf

Convolutional Layer



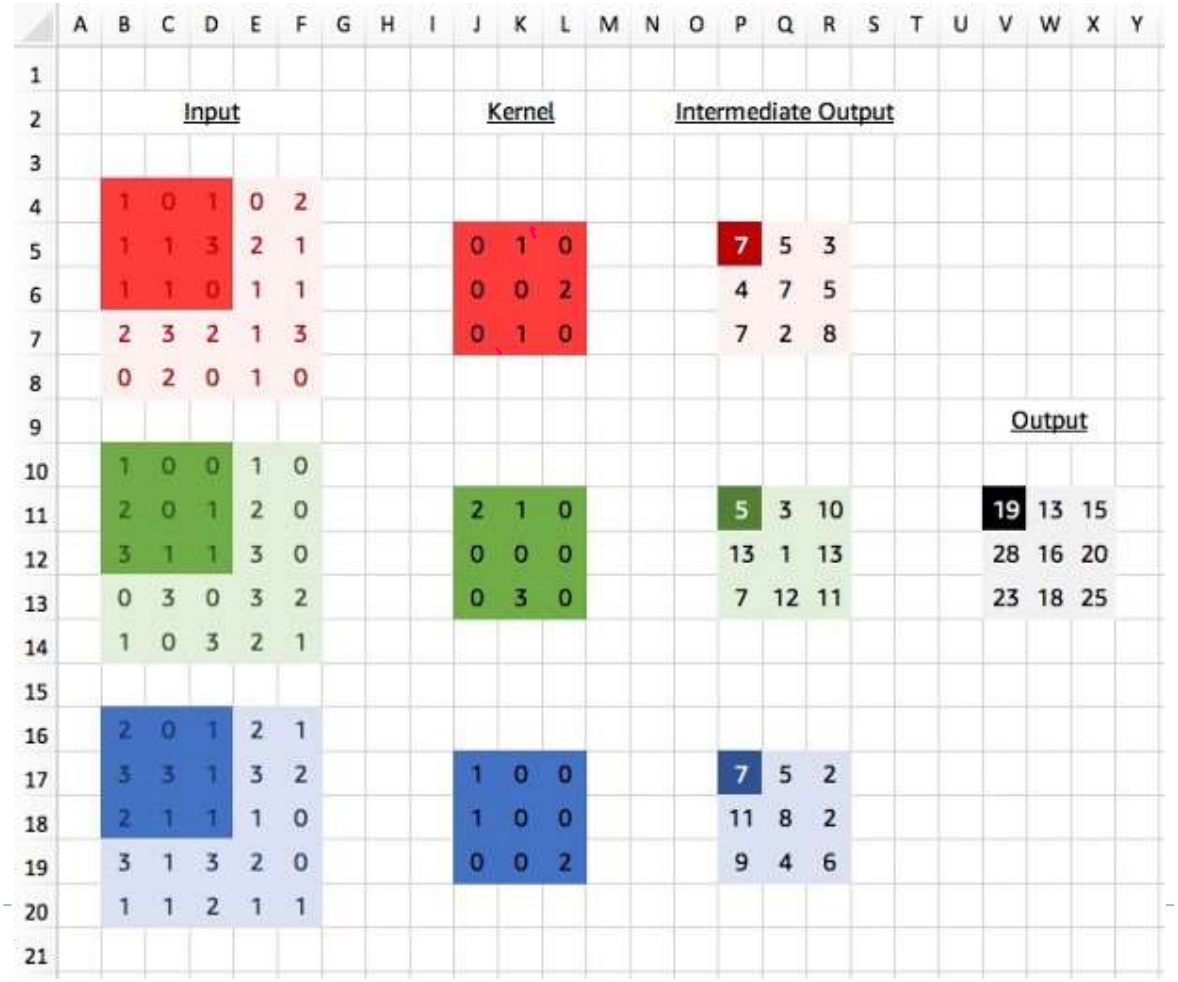
Convolutional Layer

- 출처: https://medium.com/apache-mxnet/multi-channel-convolutions-explained-with-ms-excel-9bbf8eb77108
- A 2D convolution with a 3x3 kernel applied to a 3 channel RGB input of size 5x5 to give output of 3x3



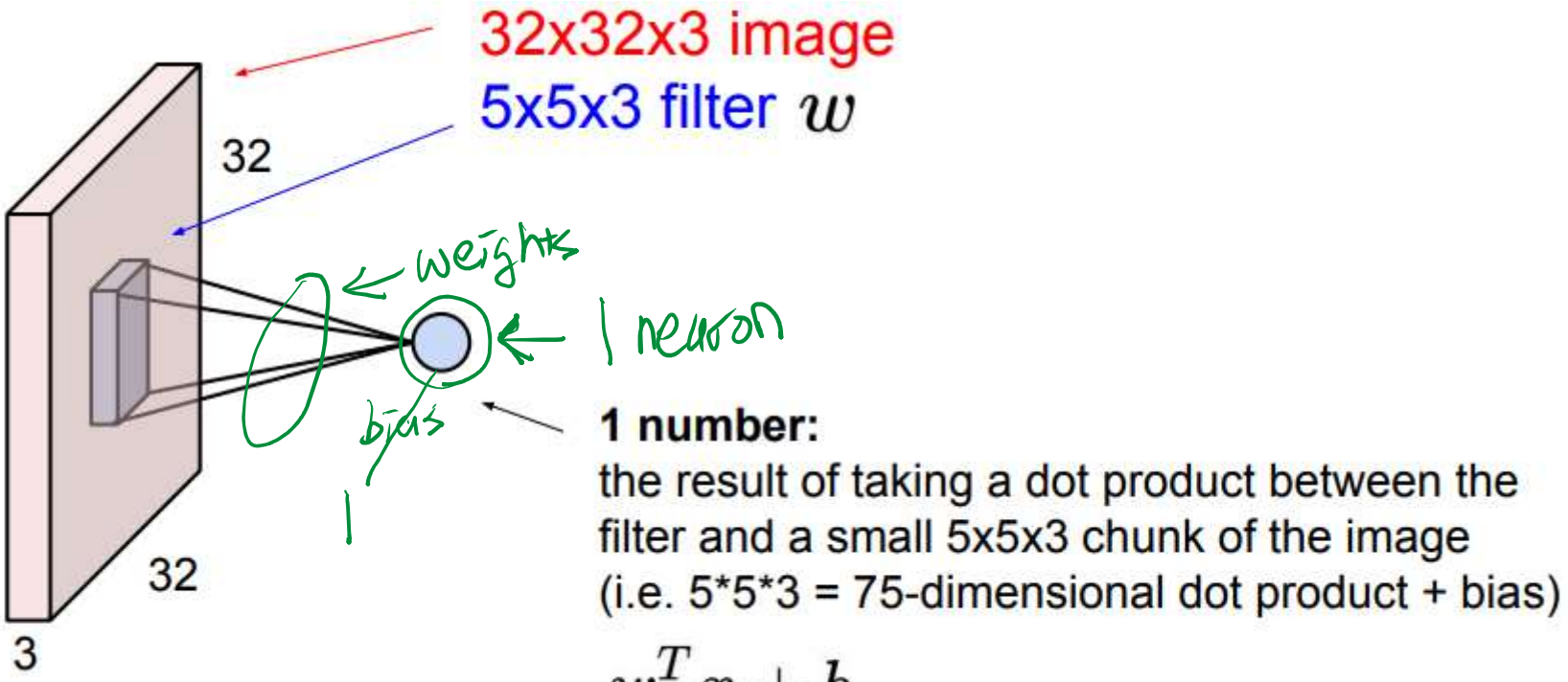
Convolutional Layer

- 출처: https://medium.com/apache-mxnet/multi-channel-convolutions-explained-with-ms-excel-9bbf8eb77108
- A 2D convolution with a 3x3 kernel applied to a 3 channel RGB input of size 5x5 to give output of 3x3



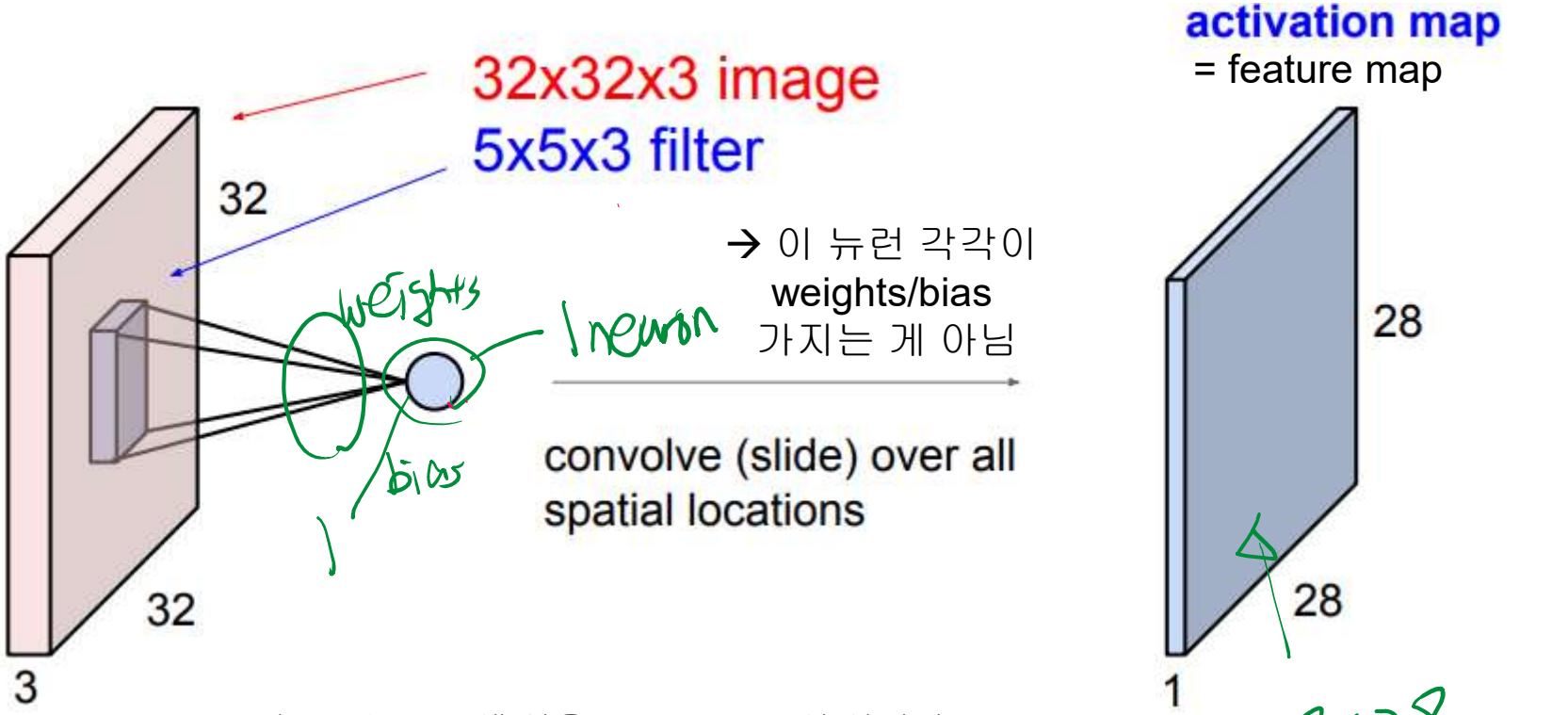
출처: https://web.stanford.edu/class/cs20si/2017/lectures/slides_06.pdf

Convolutional Layer



출처: https://web.stanford.edu/class/cs20si/2017/lectures/slides_06.pdf

Convolutional Layer



→ 이 뉴런 각각이 weights/bias 가지는 게 아님

Weights와 bias는 filter에 있음 (feature map이 아니라)

→ 한 feature map의 모든 뉴런(이 경우에는 28x28개)이 weight/bias 공유:
이 경우에는 3x5x5+1개의 trainable parameters

← Parameter가 (32x32+1)x(28x28) 개가 아님(Dense layer와 다름)

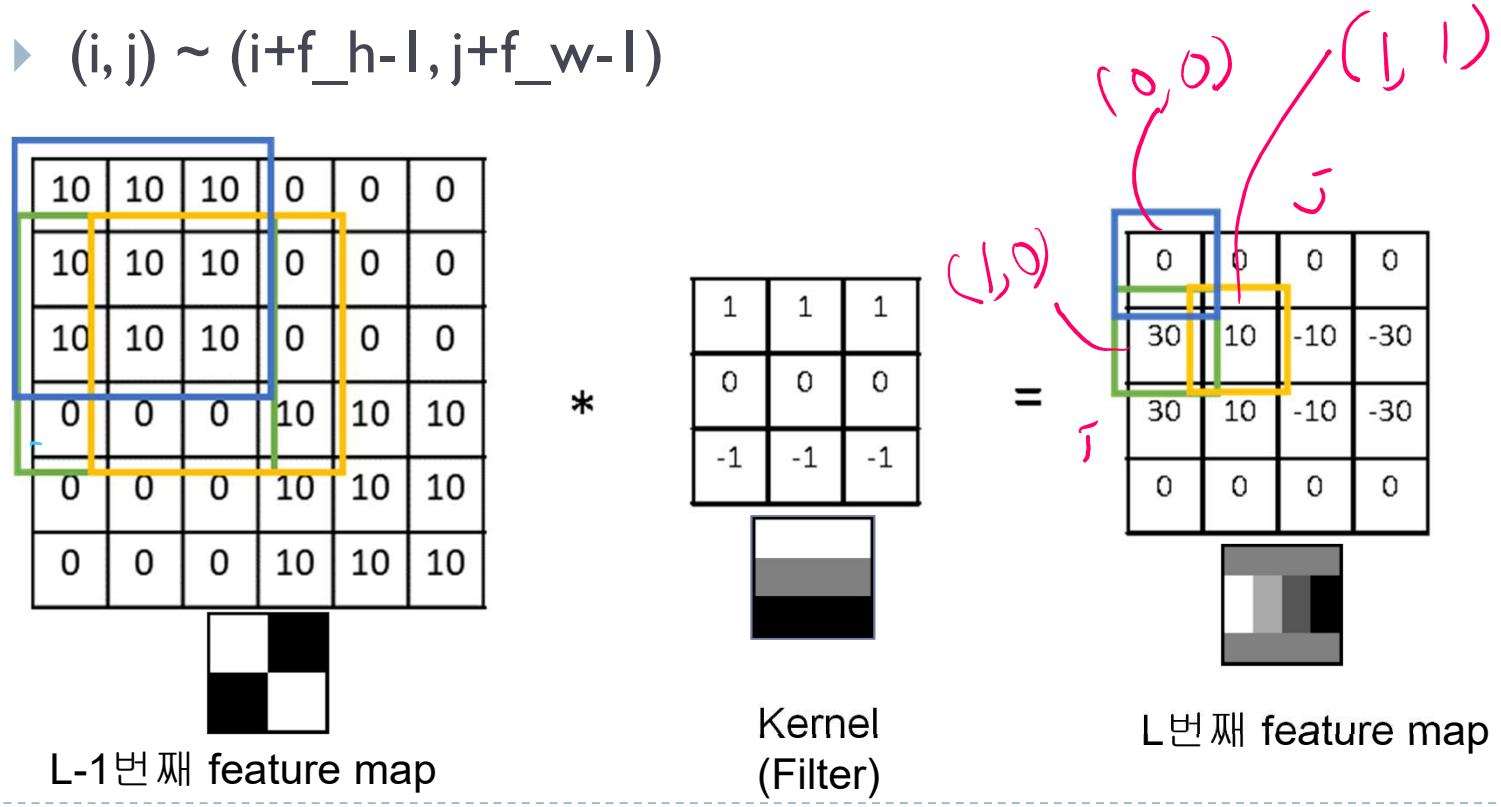
* Convolutional layer에서는 뉴런 별로 weights를 가지는 게 아님:
(유사 예) input layer, softmax layer (아예 weights가 없음)

- Convolutional layer가 다음 layer의 input layer가 됨

→ 이 뉴런 각각이 weights/bias 가지는 게 아님

Feature Map's Index with Stride 1

- Notation: kernel이 f_h x f_w 크기
- L번째 feature map에서의 (i, j) 뉴런이 연결되는 L-1번째 feature map의 뉴런은?
  - (i, j) ~ (i+f_h-1, j+f_w-1)



L-1번째 feature map

Kernel (Filter)

L번째 feature map

그림 출처: http://datahacker.rs/edge-detection-extended/

Feature Map's Index with Stride > 1

- Notation: kernel이 f_h x f_w 크기, strides s_h x s_w
- L번째 feature map에서의 (i, j) 뉴런이 연결되는 L-1번째 feature map의 뉴런은?
  - (i x s_h, j x s_w) ~ (i x s_h+f_h-1, j x s_w + f_w-1)

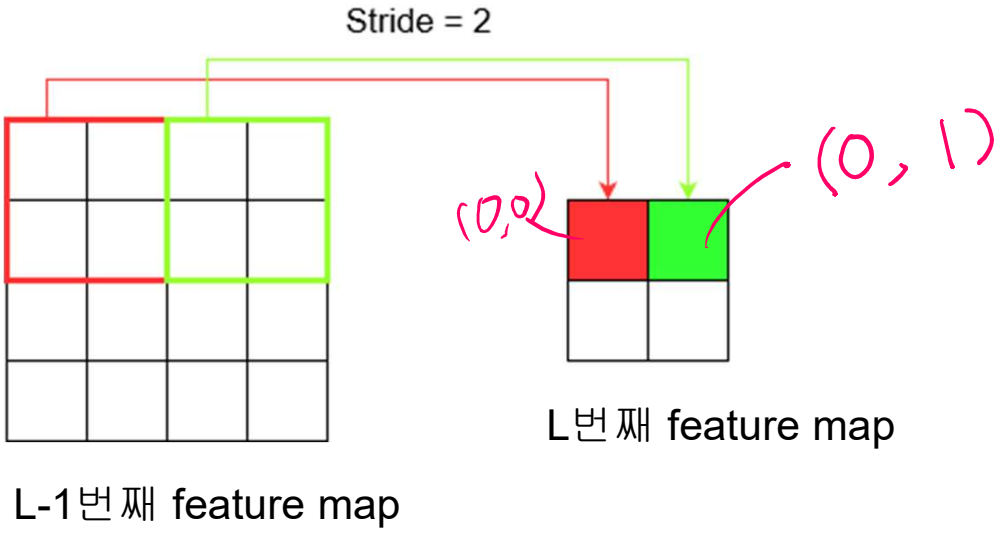
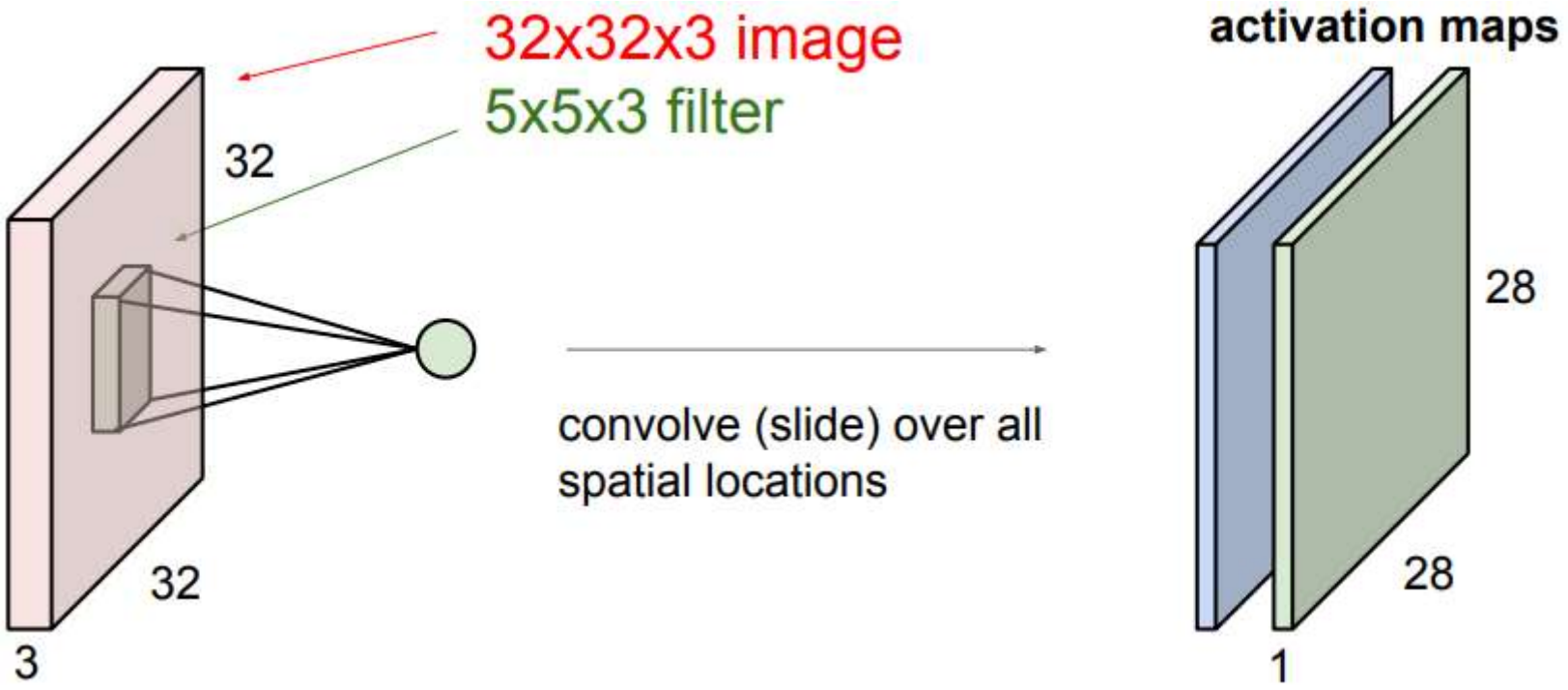


그림 출처: http://datahacker.rs/edge-detection-extended/

출처: https://web.stanford.edu/class/cs20si/2017/lectures/slides_06.pdf

Convolutional Layer

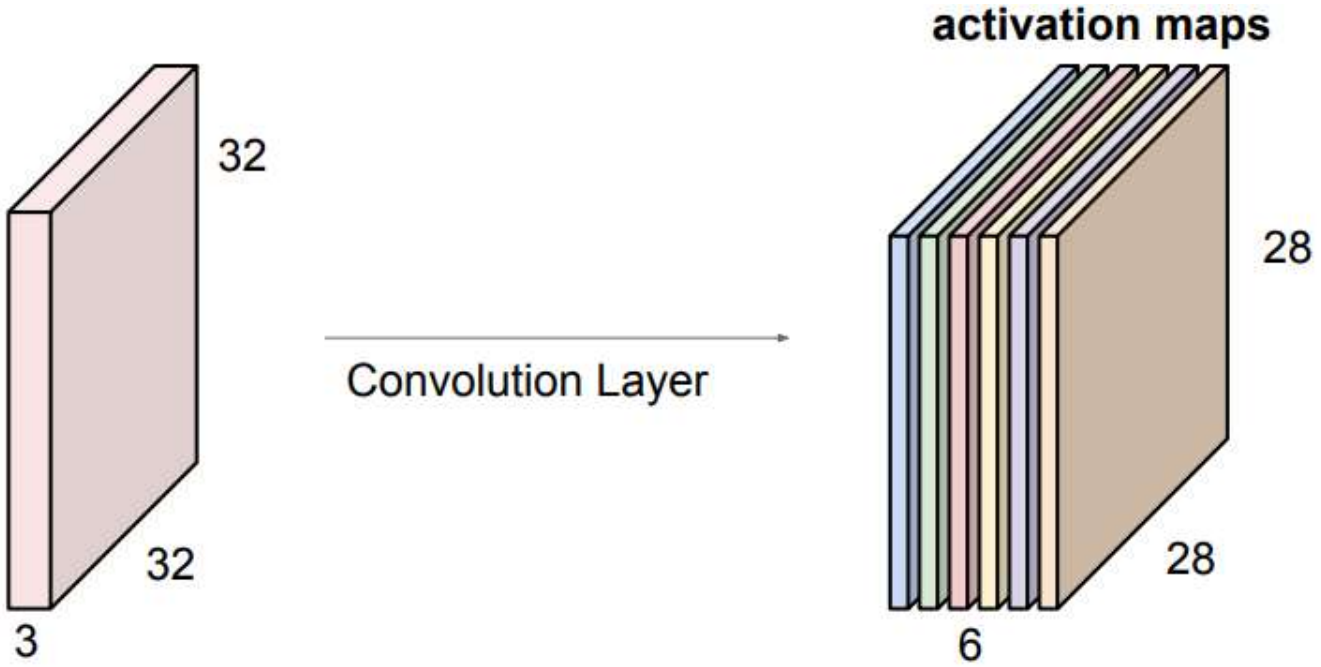
- Consider a second, **green** filter



출처: https://web.stanford.edu/class/cs20si/2017/lectures/slides_06.pdf

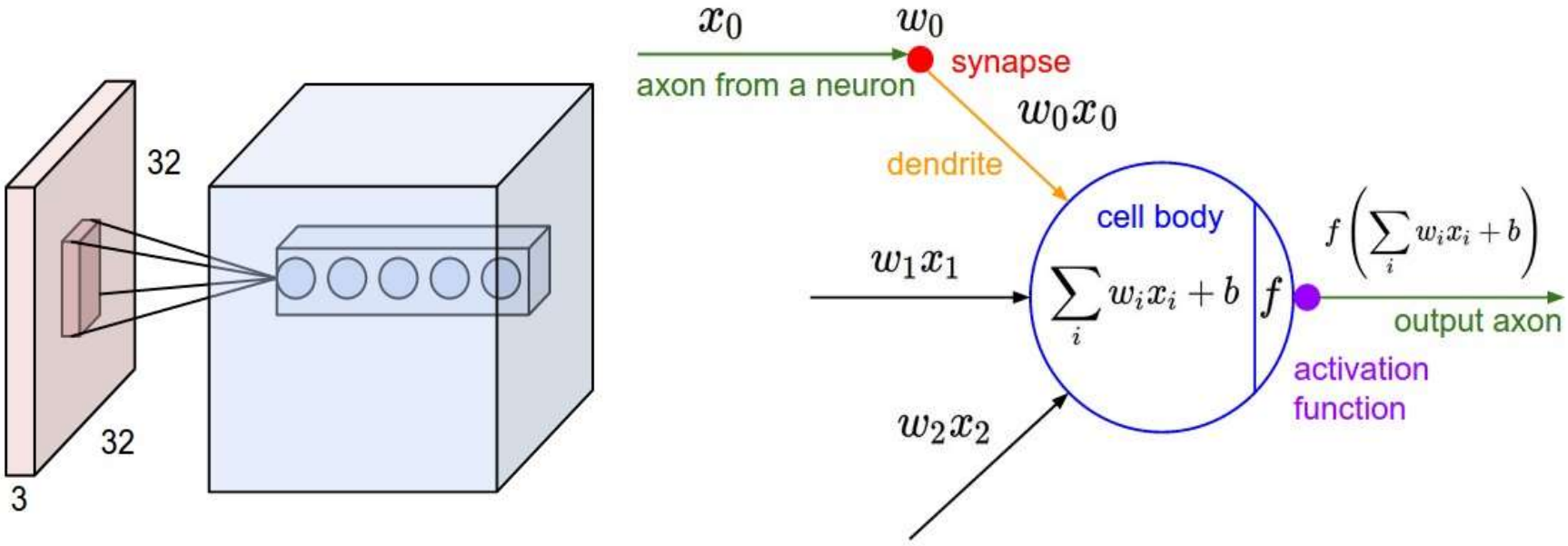
Convolutional Layer

- For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:
  - We stack these up to get a "new image" of size 28x28x6!



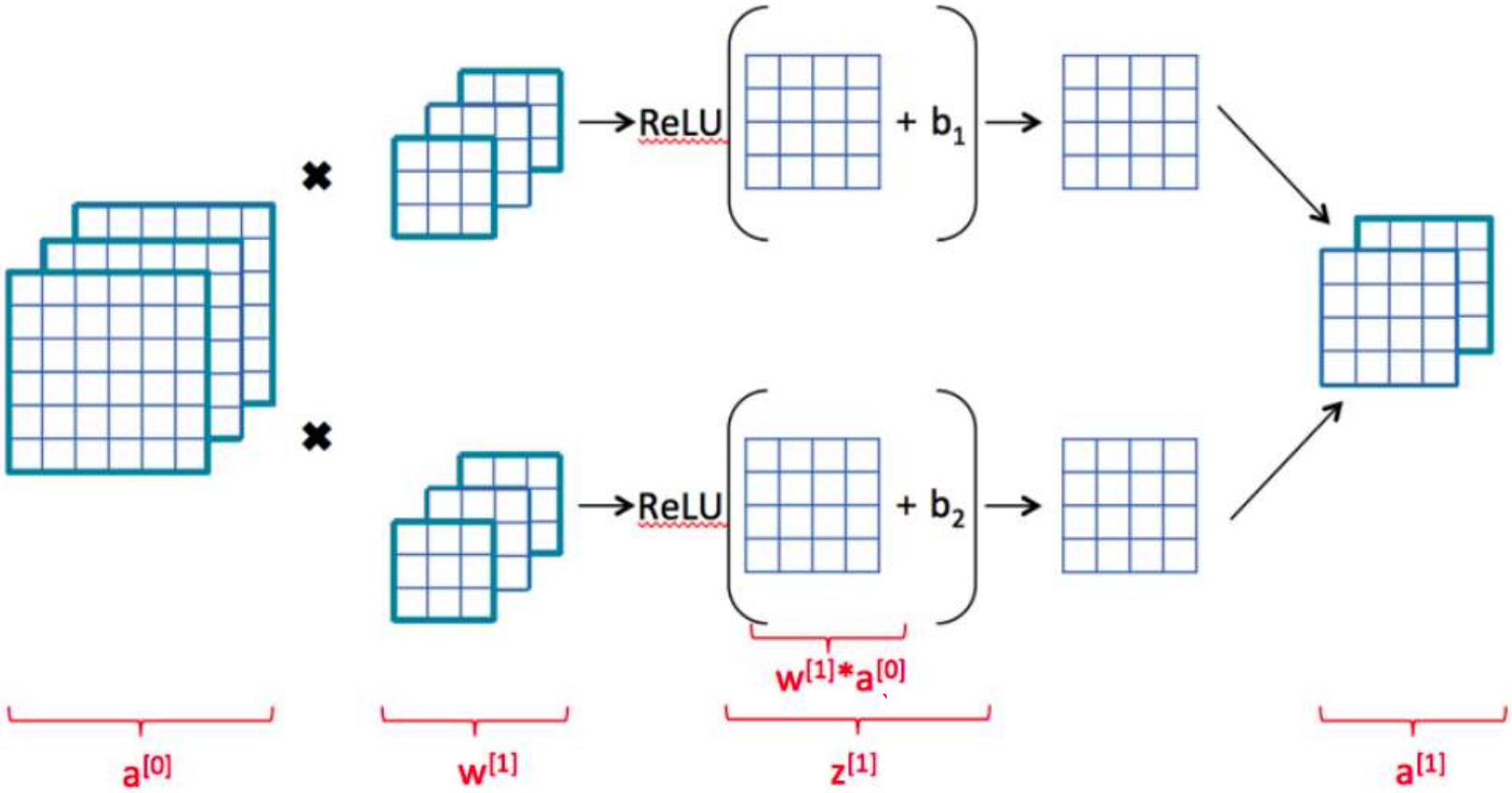
Convolutional Layer: Feed Forward Propagation

- https://cs231n.github.io/convolutional-networks/
- There are multiple neurons (5 in this example) along the depth, all looking at the same region in the input
- The neurons work the same as in the fully connected layer
  - They still compute a dot product of their weights with the input followed by a non-linearity.
  - But their connectivity is now restricted to be local spatially.



Convolutional Layer: Feed Forward Propagation

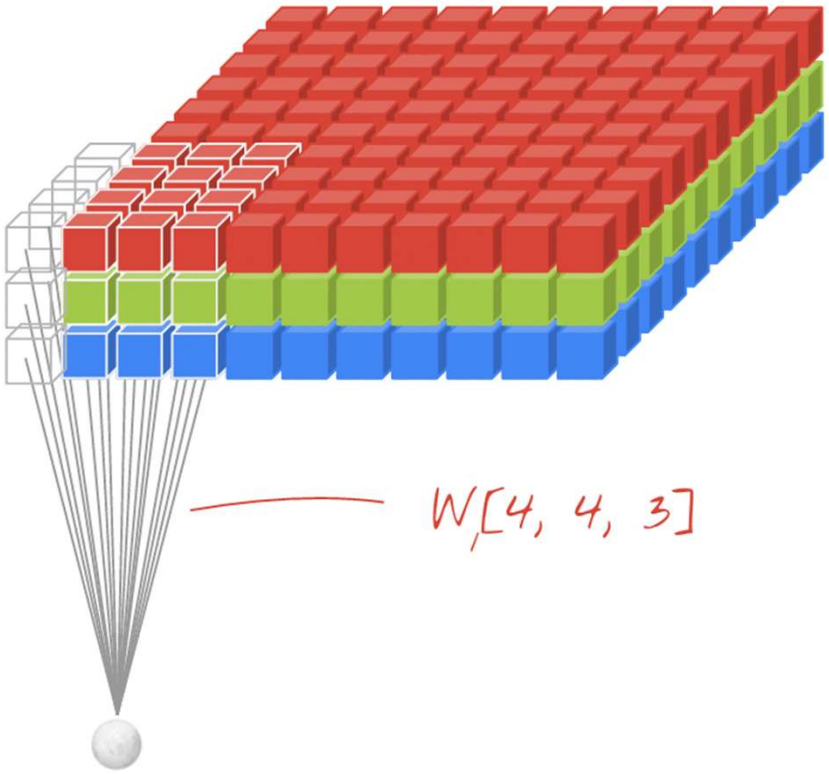
- https://www.kaggle.com/viroviro/introduction-to-cnns



출처: https://web.stanford.edu/class/cs20si/2017/lectures/slides_06.pdf

Convolutional Layer

- 3D inputs to 2D Conv
- 출처: https://sites.google.com/site/nttrungmtwiki/home/it/data-science---python/tensorflow/tensorflow-and-deep-learning-part-3?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1



Output of a Neuron in a Convolutional Layer

- Notations:
  - The current convolutional layer: l
  - The previous convolutional (or input) layer: l-1
  - z_(i, j, k): the output of the neuron located in (i, j, k) of the convolutional layer l
    - Of feature map k in the convolutional layer l
  - z'_(i', j', k'): the output of the neuron located in (i', j', k') of the convolutional layer l-1
  - s_h, s_w, f_h, f_w: height/width of stride/filter
  - f_d':# of feature maps (or depth) of the convolutional (or input) layer l-1
  - b_k: the bias of the feature map k in the layer l
    - Feature map k의 brightness(밝기)를 조절하는 knob(다이얼)로 생각할 수 있음
  - w_(u, v, k', k): the weight between any neuron in feature map k of the layer l and its input located at (u, v, k') in the previous layer l-1

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{d'}-1} z'_{i \times s_h + u,\ j \times s_w + v,\ k'} \times w_{u,v,k',k}$$

출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

TensorFlow Shapes in CNN

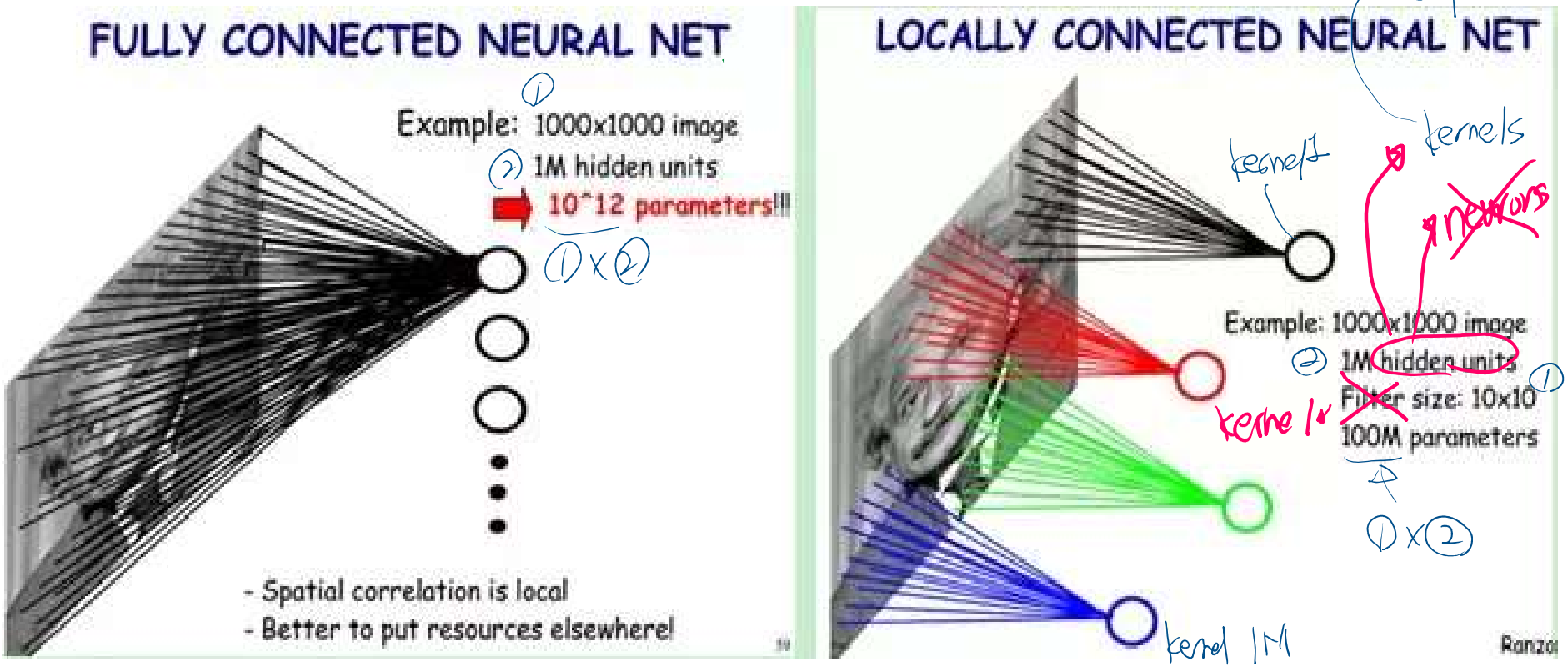
- Each input image:
  - [height, width, channels]
- Each mini-batch:
  - [mini-batch_size, height, width, channels]
- Weights of convolutional layer l:
  - [f_h, f_w, f_d', f_d]
- The bias terms of a convolutional layer l:
  - [f_d]

Kernels vs. Filters

- Kernel
  - A matrix of weights which are multiplied with the input to extract relevant features
  - Each kernel is assigned to a particular channel of the input
- Filter
  - A concatenation of multiple kernels (# input channels)
  - Always a one dimension more than the kernels
- A CNN layer with kernel dimensions h * w and input channels k
  - The filter dimensions are h * w * k

Fully Connected vs. Locally Connected NN

- https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/
  - 이 예에서의 parameter 개수는 bias 제외

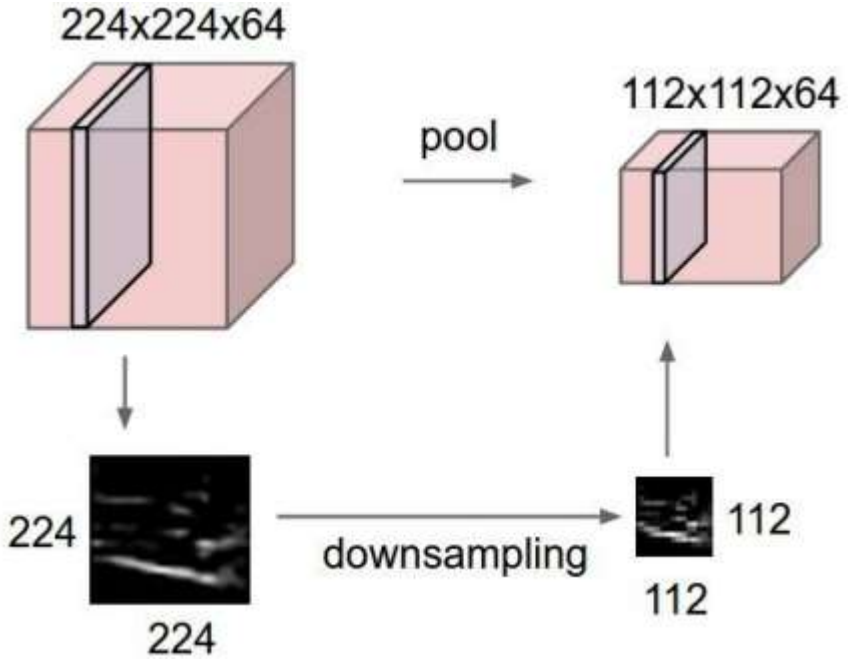



```
# https://stackoverflow.com/questions/58991594/understanding-number-of-parameters-in-keras-conv2d-layer
number_parameters = out_channels * (in_channels * kernel_h * kernel_w + 1)  # 1 for bias
```

출처: https://web.stanford.edu/class/cs20si/2017/lectures/slides_06.pdf

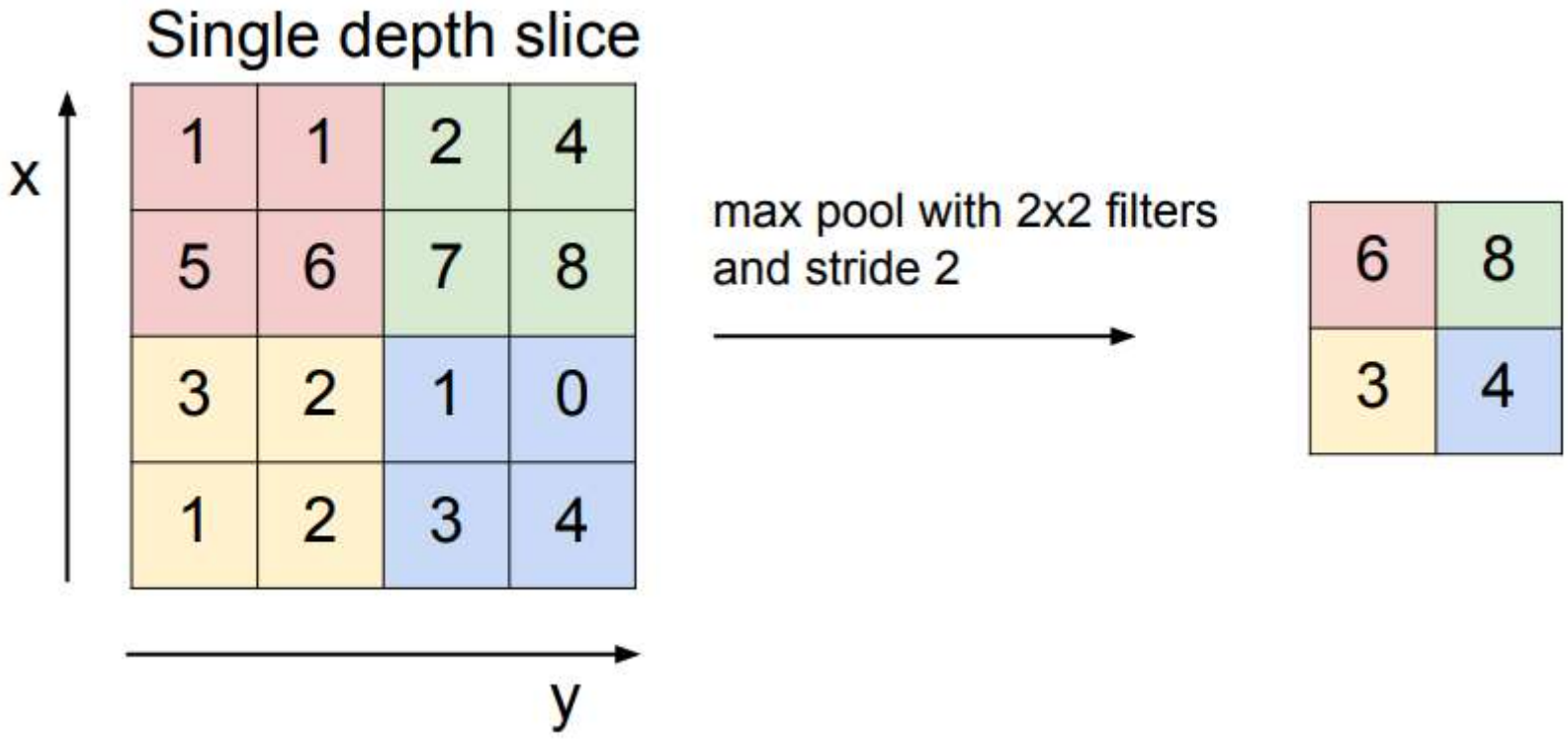
Pooling Layer

- Makes the representations smaller and more manageable
  - Operates over each activation map independently:



출처: https://web.stanford.edu/class/cs20si/2017/lectures/slides_06.pdf

Max Pooling



출처: 차루 C. 아가르왈 저/류광 역, 신경망과 심층 학습, 제이펍, 2019. 9.

Typical CNN Configuration

- Stride (보폭) = 1
  - 2를 사용하는 경우도 있으나, 3 이상은 잘 사용되지 않음
- Padding (여백 채우기)
  - Half padding(절반 채우기)이 일반적으로 사용됨
- 입력 W = H
- 필터 개수는 2의 거듭제곱
- 필터의 높이와 너비는 3이나 5가 잘 사용됨

Matplotlib plt.imshow()

```
matplotlib.pyplot.imshow(X, cmap=None, norm=None, aspect=None,
    interpolation=None, alpha=None, vmin=None, vmax=None, origin=None,
    extent=None, *, filternorm=True, filterrad=4.0, resample=None,
    url=None, data=None, **kwargs)
```

- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html

X: array-like or PIL image data
Supported array shapes are:

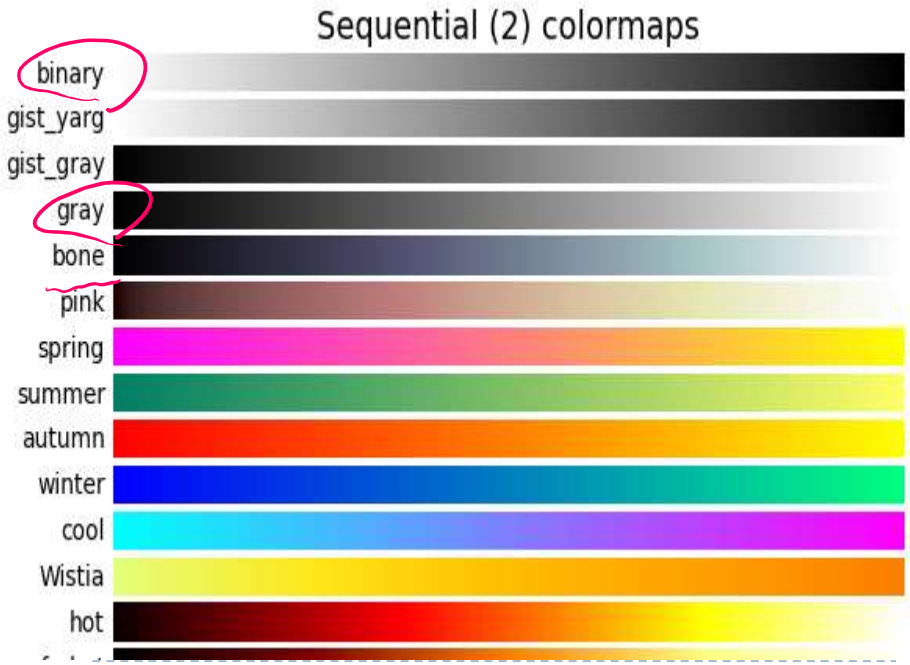
- (M, N): an image with scalar data. The values are mapped to colors using normalization and a colormap. See parameters *norm*, *cmap*, *vmin*, *vmax*.
- (M, N, 3): an image with RGB values (0-1 float or 0-255 int).
- (M, N, 4): an image with RGBA values (0-1 float or 0-255 int), i.e. including transparency.

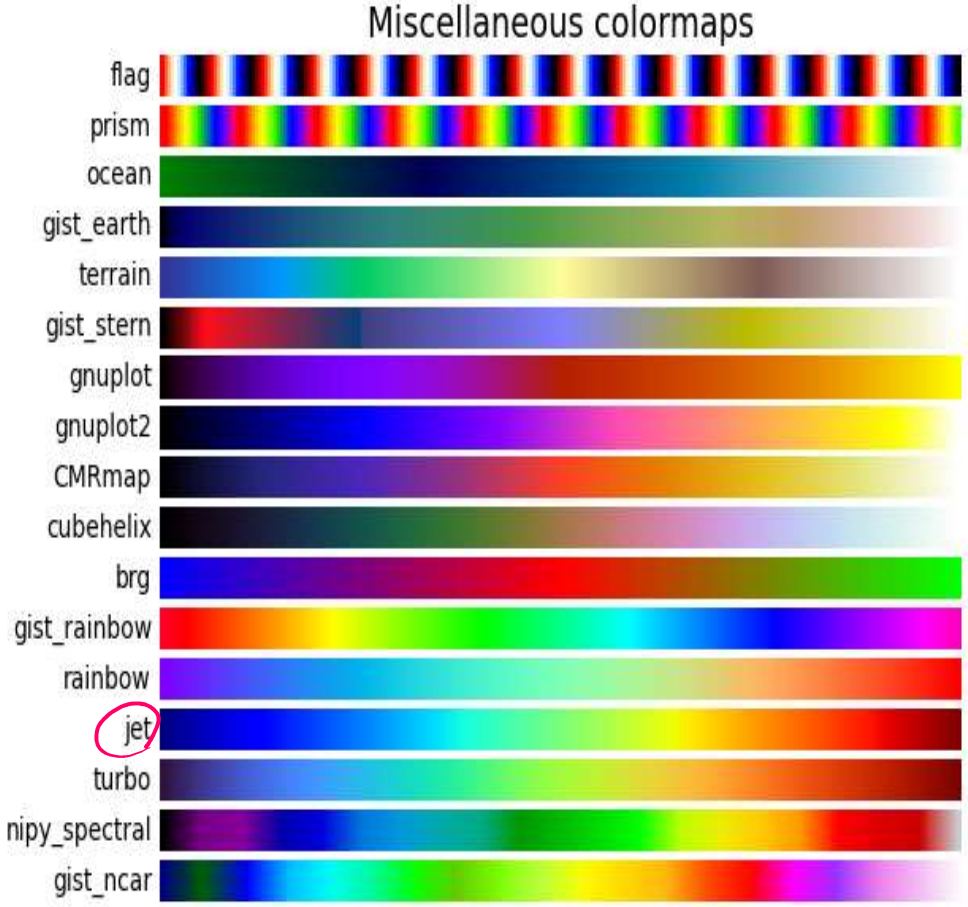
The first two dimensions (M, N) define the rows and columns of the image.
Out-of-range RGB(A) values are clipped.

cmap: str or Colormap, default: rcParams["image.cmap"] (default: 'viridis') The Colormap instance or registered colormap name used to map scalar data to colors. This parameter is ignored for RGB(A) data.

Matplotlib plt.imshow() Colormaps (cmap)

- Default for one channel: viridis (Latin어로 green의 의미)





TF Keras CNN API: keras.layers.Conv2D

```
conv = keras.layers.Conv2D(
            filters=32,
            kernel_size=3,
            strides=1,
            padding="SAME",
            activation="relu")
```

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

```
tf.keras.layers.Conv2D(
    filters, kernel_size, strides=(1, 1), padding='valid',
    data_format=None, dilation_rate=(1, 1), groups=1, activation=None,
    use_bias=True, kernel_initializer='glorot_uniform',
    bias_initializer='zeros', kernel_regularizer=None,
    bias_regularizer=None, activity_regularizer=None, kernel_constraint=None,
    bias_constraint=None, **kwargs
)
```

tf.keras.layers.Conv2D

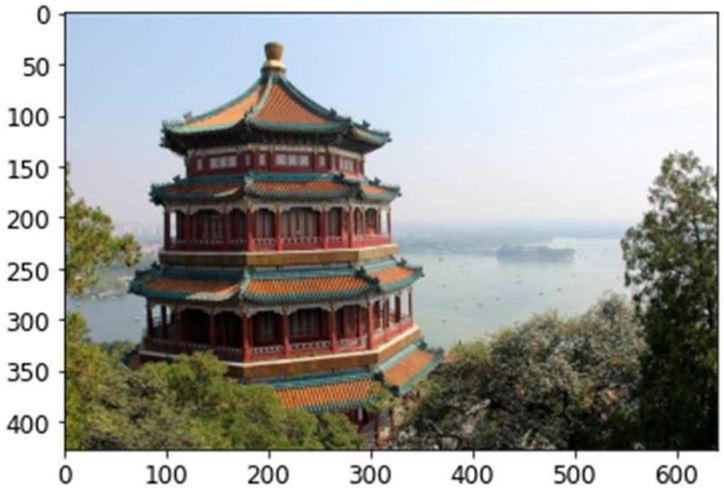
- https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

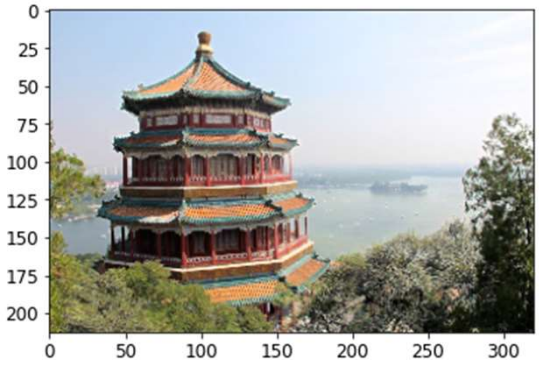
filters	Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
kernel_size	An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single intege r to specify the same value for all spatial dimensions.
strides	An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single in teger to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying an y dilation_rate value != 1.
padding	one of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding with zeros evenly to the left/ right or up/down of the input such that output has the same height/width dimension as the input.
data_format	A string, one of channels_last (default) or channels_first. The ordering of the dimensions in the inputs. channels_last corres ponds to inputs with shape (batch_size, height, width, channels) while channels_first corresponds to inputs with shape (bat ch_size, channels,height, width). It defaults to the image_data_format value found in your Keras config file at ~/.keras/keras. json. If you never set it, then it will be channels_last.
dilation_rate	an integer or tuple/list of 2 integers, specifying the dilation rate to use for dilated convolution. Can be a single integer to sp ecify the same value for all spatial dimensions. Currently, specifying any dilation_rate value != 1 is incompatible with specify ing any stride value != 1.
groups	A positive integer specifying the number of groups in which the input is split along the channel axis. Each group is convolve d separately with filters / groups filters. The output is the concatenation of all the groups results along the channel axis. Inp ut channels and filters must both be divisible by groups.
activation	Activation function to use. If you don't specify anything, no activation is applied (see keras.activations).
use_bias	Boolean, whether the layer uses a bias vector.

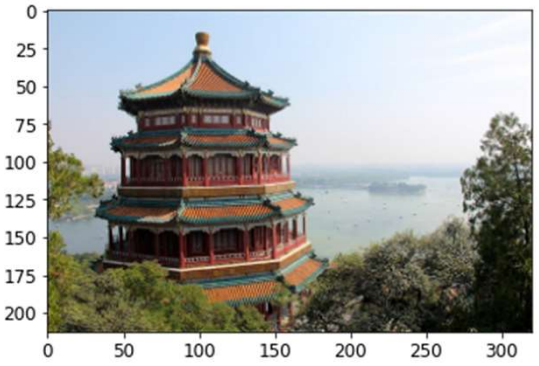
출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

TF Keras Max/Avg Pooling API

```
max_pool = keras.layers.MaxPool2D(pool_size=2)
avg_pool = keras.layers.AvgPool2D(pool_size=2)
output = max_pool(images)
output2 = avg_pool(images)
plt.imshow(images[0]); plt.show()
plt.imshow(output[0]); plt.show()
plt.imshow(output2[0]); plt.show()
```







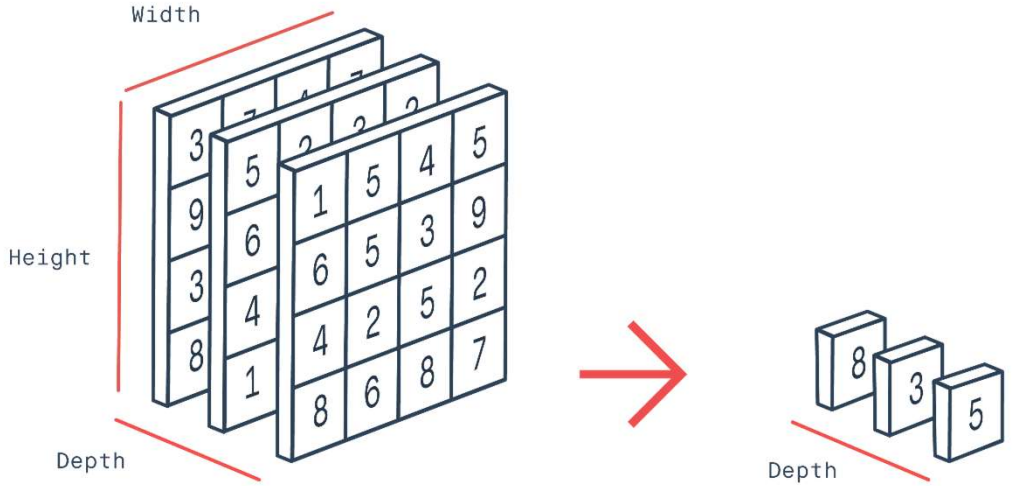
- 과거에는 사람들이 average pooling을 대부분 사용했었음(정보를 거의 안 잃으므로)
  - 현재는 사람들이 average pooling은 거의 사용하지 않음
- 현재는 max pooling만 사용: 학습이 더 잘 되기 때문
  - 평균 정보보다 최대 정보가 학습에 더 나음: 속도도 더 빠름

출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

Global Average Pooling Layer

- Computes the mean of each entire feature map
  - Outputs a single number per feature map/instance
- Has been employed by GoogleNet, Xception, SENet, …

```
>>> global_avg_pool = keras.layers.GlobalAvgPool2D()
>>> global_avg_pool(cropped_images)
<tf.Tensor: shape=(2, 3), dtype=float32, numpy=
array([[0.2788801 , 0.22507527, 0.20967631],
       [0.51287866, 0.4595188 , 0.3342377 ]], dtype=float32)>

# the same as the followings
>>> output_global_avg2 = keras.layers.Lambda(lambda X: tf.reduce_mean(X, axis=[1, 2]))
```



이미지 출처:
https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-global-average-pooling

Height x Width x Depth

1 x 1 x Depth

- https://www.tensorflow.org/api_docs/python/tf/math/reduce_mean

tf.reduce_mean()

```
y == <tf.Tensor: shape=(2, 3, 4), dtype=float64, numpy=
array([[[1., 1., 1., 1.],
        [2., 2., 2., 2.],
        [3., 3., 3., 3.]],
       [[4., 4., 4., 4.],
        [5., 5., 5., 5.],
        [6., 6., 6., 6.]]])
```

tf.math.reduce_mean(input_tensor, axis=None,
keepdims=False, name=None)

```
>>> xx = np.array([[1., 1., 1., 1.], [2., 2., 2., 2.], [3., 3., 3., 3.]])
>>> y = tf.constant([xx, xx+3])
>>> tf.reduce_mean(y, axis=-1)
<tf.Tensor: shape=(2, 3), dtype=float64, numpy= array([[1., 2., 3.], [4., 5., 6.]])>
>>> tf.reduce_mean(y, axis=0)
<tf.Tensor: shape=(3, 4), dtype=float64, numpy=
   array([[2.5, 2.5, 2.5, 2.5], [3.5, 3.5, 3.5, 3.5], [4.5, 4.5, 4.5, 4.5]])>
>>> tf.reduce_mean(y, axis=1)
<tf.Tensor: shape=(2, 4), dtype=float64, numpy= array([[2., 2., 2., 2.], [5., 5., 5., 5.]])>
>>> tf.reduce_mean(y, axis=(0, 1, 2))
<tf.Tensor: shape=(), dtype=float64, numpy=3.5>
>>> tf.reduce_mean(y, axis=[0, 1]) # the same as axis=(1, 0)
<tf.Tensor: shape=(4,), dtype=float64, numpy=array([3.5, 3.5, 3.5, 3.5])>
>>> tf.reduce_mean(y, axis=[1, 2])  # the same as axis=[2, 1]
<tf.Tensor: shape=(2,), dtype=float64, numpy=array([2, 5])>
>>> tf.reduce_mean(y, axis=(0, 2)) # the same as axis=(-1, 0)
<tf.Tensor: shape=(3,), dtype=float64, numpy=array([2.5, 3.5, 4.5])>
>>> tf.reduce_mean(y[:, :, 1])
<tf.Tensor: shape=(), dtype=float64, numpy=3.5>
```

input_tensor	The tensor to reduce. Should have numeric type.
axis	The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input_tensor), rank(input_tensor)].
keepdims	If true, retains reduced dimensions with length 1.
Name	A name for the operation (optional).

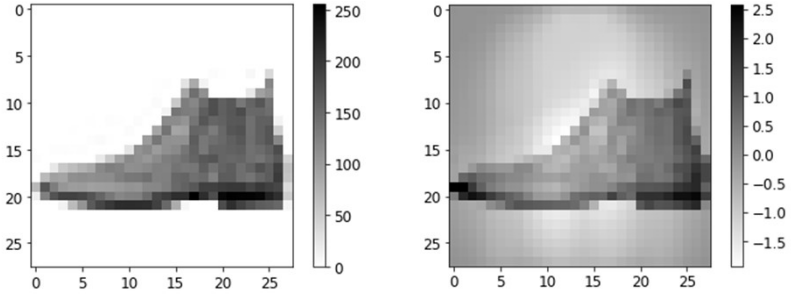
출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

CNN Example for Fashion MNIST (1)

```
(X_train_full, y_train_full), (X_test, y_test) = keras.datasets.fashion_mnist.load_data()
X_train, X_valid = X_train_full[:-5000], X_train_full[-5000:]
y_train, y_valid = y_train_full[:-5000], y_train_full[-5000:]

X_test_org = X_test[:] # ksaehwa

X_mean = X_train.mean(axis=0, keepdims=True)
X_std = X_train.std(axis=0, keepdims=True) + 1e-7

X_train = (X_train - X_mean) / X_std # X_train.shape: (55000, 28, 28)
X_valid = (X_valid - X_mean) / X_std
X_test = (X_test - X_mean) / X_std

X_train = X_train[..., np.newaxis] # X_train.shape:(55000, 28, 28, 1)
X_valid = X_valid[..., np.newaxis]
X_test = X_test[..., np.newaxis]
```

```
>>> a = np.array([[1., 1., 1., 1.],
[2., 2., 2., 2.], [3., 3., 3., 3.]])
>> a.shape
(3, 4)
>>> b = a[…, np.newaxis]
>>> b
 array([[[1.], [1.], [1.], [1.]],
        [[2.], [2.], [2.], [2.]],
        [[3.], [3.], [3.], [3.]]])
>>> b.shape
(3, 4, 1)
```

```
>>> plt.imshow(X_test_org[0][:, :, 0], cmap='binary')
>>> plt.colorbar(); plt.show()
>>> plt.imshow(X_test[0][:, :, 0], cmap='binary')
>>> plt.colorbar(); plt.show()
```

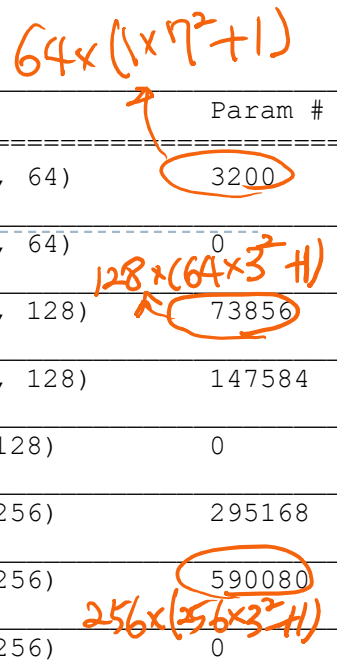


출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

CNN Example for Fashion MNIST (2)

```
from functools import partial

DefaultConv2D = partial(
     keras.layers.Conv2D,
     kernel_size=3,
     activation='relu', padding="SAME")

model = keras.models.Sequential([
    DefaultConv2D(filters=64, kernel_size=7,
       input_shape=[28, 28, 1], name='conv1'),
    keras.layers.MaxPooling2D(pool_size=2,
            name='pool1'),
    DefaultConv2D(filters=128, name='conv2'),
    DefaultConv2D(filters=128, name='conv3'),
    keras.layers.MaxPooling2D(pool_size=2,
            name='pool2'),
    DefaultConv2D(filters=256, name='conv4'),
    DefaultConv2D(filters=256, name='conv5'),
    keras.layers.MaxPooling2D(pool_size=2,
            name='pool3'),
    keras.layers.Flatten(name='fltn1'),
    keras.layers.Dense(units=128,
            activation='relu', name='dens1'),
    keras.layers.Dropout(0.5, name='drop1'),
    keras.layers.Dense(units=64,
            activation='relu', name='dens2'),
    keras.layers.Dropout(0.5, name='drop2'),
    keras.layers.Dense(units=10,
            activation='softmax', name='dens3'),
])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 28, 28, 64)        3200
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 14, 14, 128)       73856
_________________________________________________________________
conv3 (Conv2D)               (None, 14, 14, 128)       147584
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 7, 7, 128)         0
_________________________________________________________________
conv4 (Conv2D)               (None, 7, 7, 256)         295168
_________________________________________________________________
conv5 (Conv2D)               (None, 7, 7, 256)         590080
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 3, 3, 256)         0
_________________________________________________________________
fltn1 (Flatten)              (None, 2304)              0
_________________________________________________________________
dens1 (Dense)                (None, 128)               295040
_________________________________________________________________
drop1 (Dropout)              (None, 128)               0
_________________________________________________________________
dens2 (Dense)                (None, 64)                8256
_________________________________________________________________
drop2 (Dropout)              (None, 64)                0
_________________________________________________________________
dens3 (Dense)                (None, 10)                650
=================================================================
Total params: 1,413,834
Trainable params: 1,413,834
Non-trainable params: 0
```




```
# https://stackoverflow.com/questions/58991594/understanding-number-of-parameters-in-keras-conv2d-layer
number_parameters = out_channels * (in_channels * kernel_h * kernel_w + 1)  # 1 for bias
```

출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

CNN Example for Fashion MNIST (3)

```
model.compile(loss="sparse_categorical_crossentropy", optimizer="nadam", metrics=["accuracy"])
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_valid, y_valid))
score = model.evaluate(X_test, y_test)
X_new = X_test[:10] # pretend we have new images
y_pred = model.predict(X_new)
```

```
Epoch 1/10
1719/1719 [==============================] - 11s 6ms/step - loss: 0.6986 - accuracy: 0.7568 -
val_loss: 0.3509 - val_accuracy: 0.8758
Epoch 2/10
1719/1719 [==============================] - 11s 6ms/step - loss: 0.4115 - accuracy: 0.8633 -
val_loss: 0.3233 - val_accuracy: 0.8842
Epoch 3/10
1719/1719 [==============================] - 11s 6ms/step - loss: 0.3478 - accuracy: 0.8835 -
val_loss: 0.3025 - val_accuracy: 0.8962
Epoch 4/10
1719/1719 [==============================] - 11s 6ms/step - loss: 0.3163 - accuracy: 0.8941 -
val_loss: 0.2962 - val_accuracy: 0.8938
…
Epoch 8/10
1719/1719 [==============================] - 11s 6ms/step - loss: 0.2594 - accuracy: 0.9116 -
val_loss: 0.2772 - val_accuracy: 0.9028
Epoch 9/10
1719/1719 [==============================] - 11s 6ms/step - loss: 0.2436 - accuracy: 0.9175 -
val_loss: 0.2976 - val_accuracy: 0.9040
Epoch 10/10
1719/1719 [==============================] - 11s 6ms/step - loss: 0.2441 - accuracy: 0.9190 -
val_loss: 0.2816 - val_accuracy: 0.9066
313/313 [==============================] - 1s 3ms/step - loss: 0.3051 - accuracy: 0.9038
```

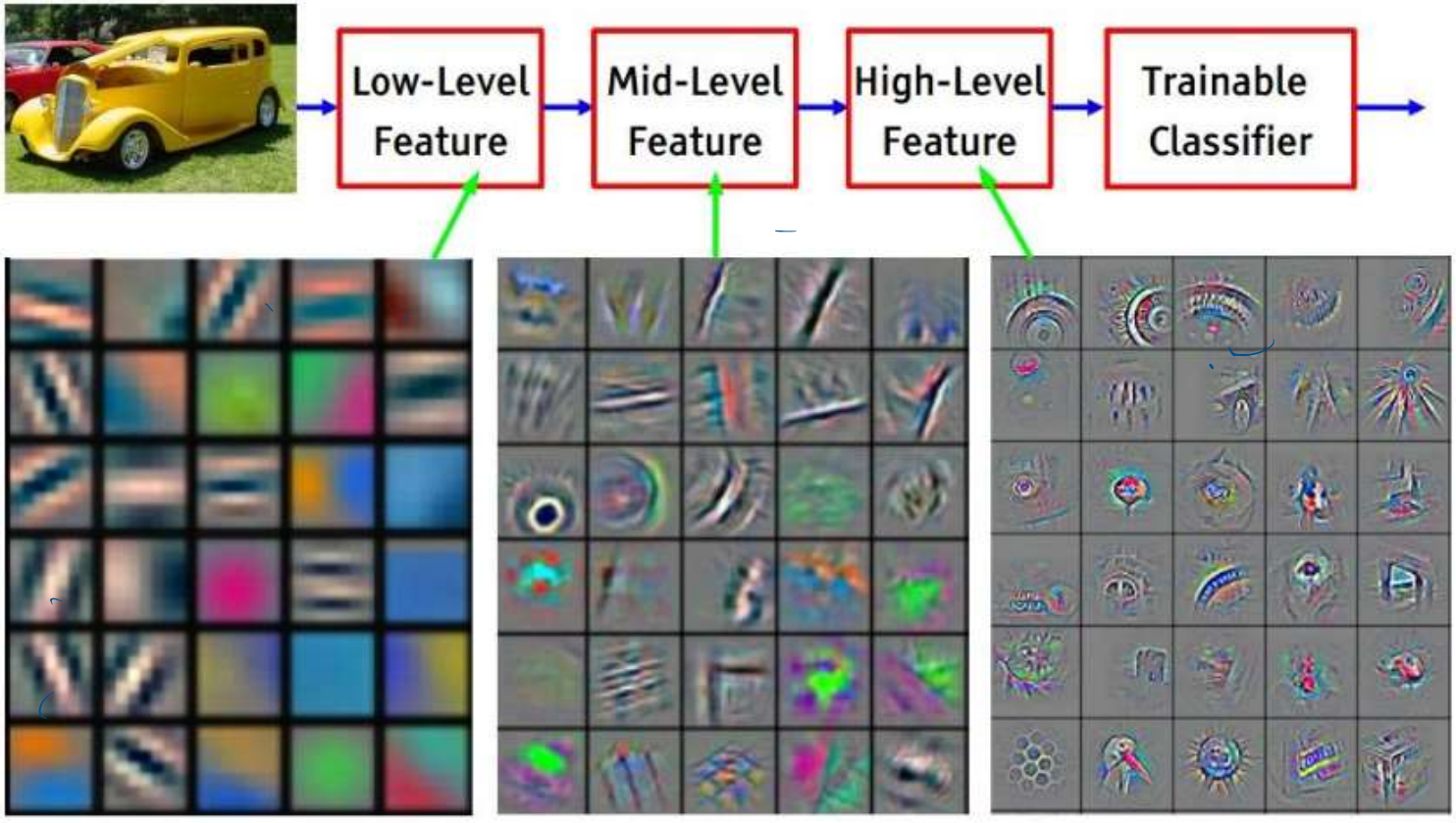
출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

CNN Example for Fashion MNIST (4)

	Non-CNN (Ch. 10)	CNN (Ch. 14)
Hidden Dense Units	(Input: 28x28=784) 300, 100	(Input: 3x3x256=2304) 128, 64
Trainable Parameters	266,610	1,413,834
Epoch 별 학습 시간	6s 3ms/step	11s 6ms/step
Training Epochs	30	10
Evaluation (10개 input prediction) 시간	1s 2m/step	1s 3m/step
Evaluation Accuracy	0.8827	0.9038

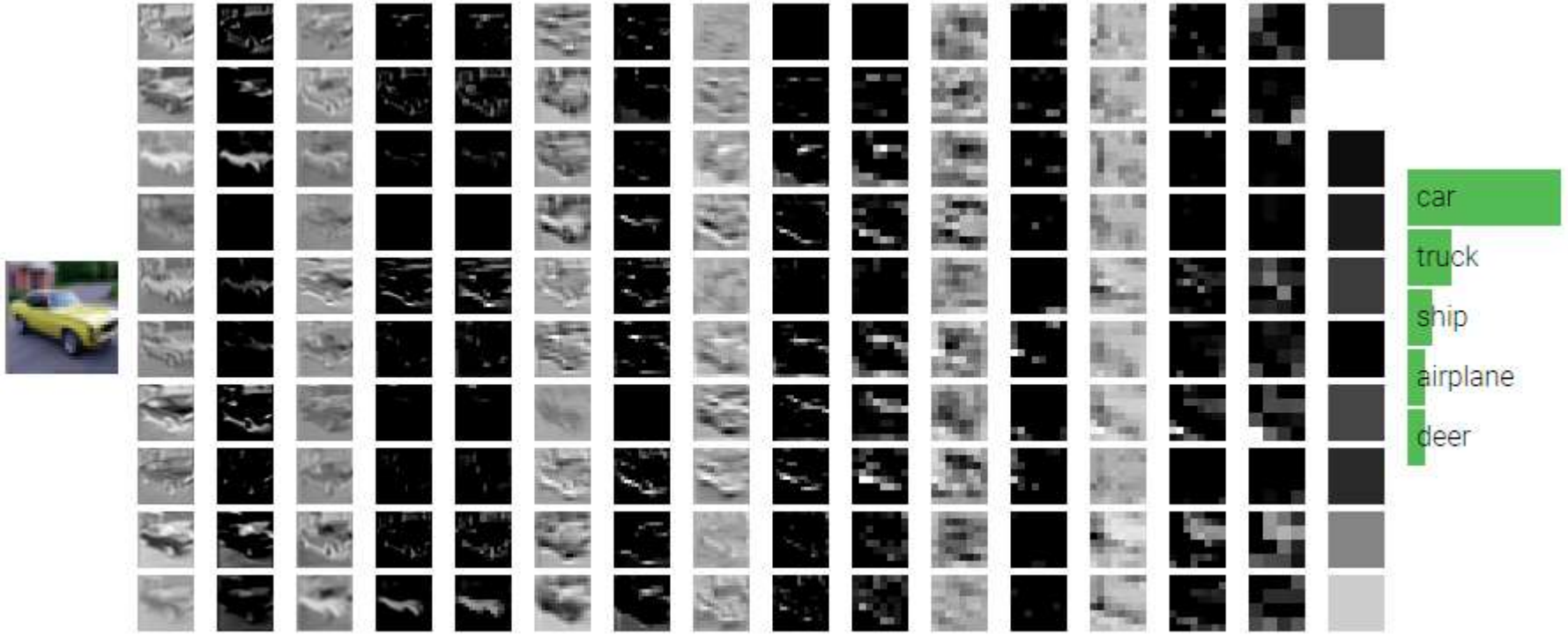
Example CNN Trained (Learned) Filters

- http://cs231n.stanford.edu/slides/winter1516_lecture7.pdf



Example CNN Outputs

- http://cs231n.stanford.edu/

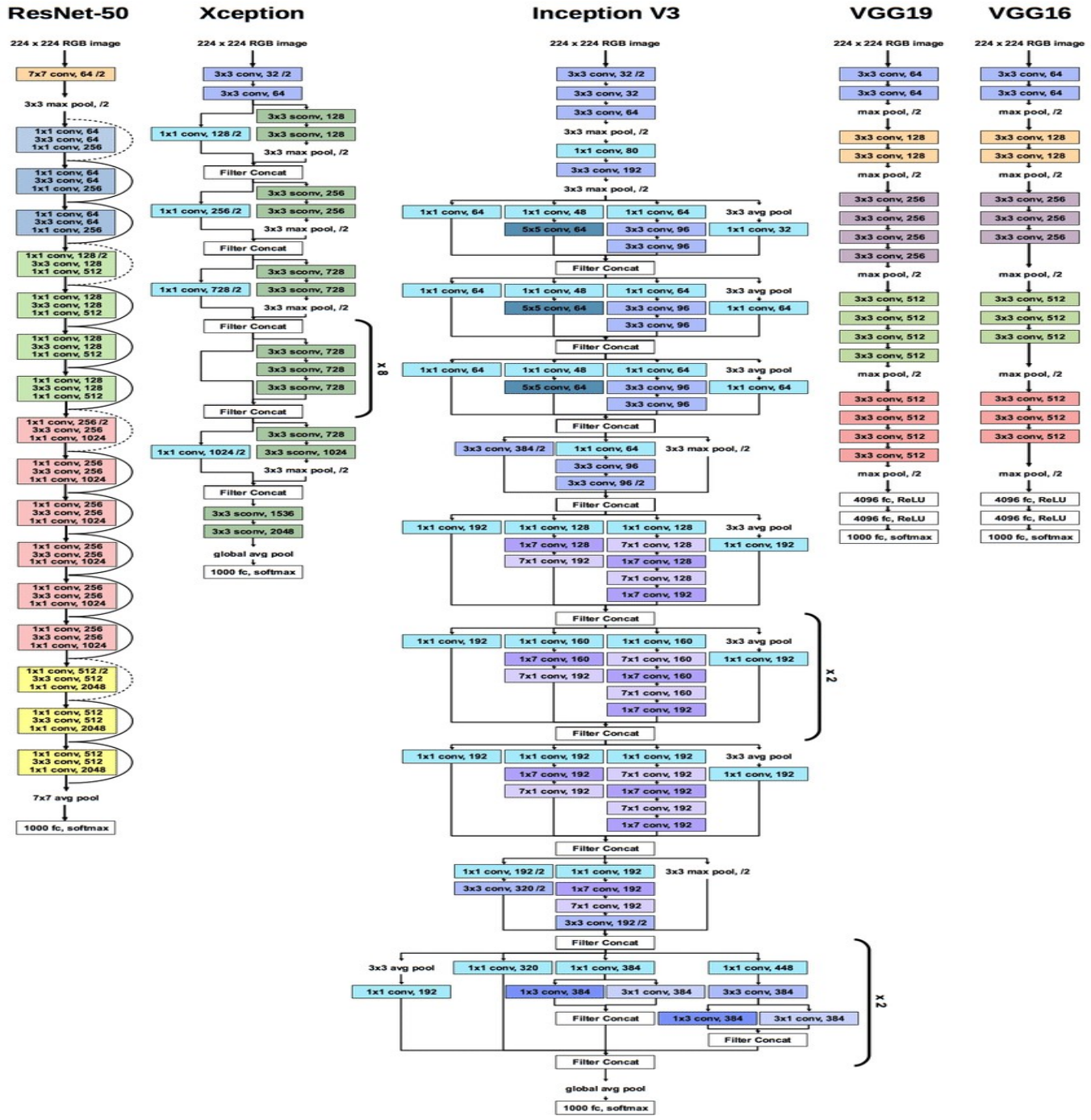


출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

CNN Architectures

- LeNet-5 1998
  - http://yann.lecun.com/exdb/lenet/
- AlexNet 2012
  - Data augmentation (데이터 증식)
- GoogLeNet 2014 → Inception v3
  - Inception module
- VGG 2014
- ResNet (Residual Network) 2015
  - 인간 수준 능가
  - Skip connections
- Xception 2016
  - Francois Chollet
- SENet 2017

- https://www.researchgate.net/figure/VGG16-VGG19-Inception-V3-Xception-and-ResNet-50-architectures_fig1_330478807



TensorFlow Convolution Operations

- keras.layers.Conv1D
  - For RNN
- keras.layers.Conv3D
- dilation_rate
  - Hyperparameter of any convolutional layer
  - Setting of 2 or more creates holes in a convolutional layer
  - Lets the convolutional layer have a larger receptive field at no computational price (and parameters)
- tf.nn.depthwise_conv2d()

출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

Pretrained Models for Transfer Learning

```
import tensorflow_datasets as tfds
dataset, info = tfds.load("tf_flowers", as_supervised=True, with_info=True)
class_names = info.features["label"].names
# ['dandelion', 'daisy', 'tulips', 'sunflowers', 'roses']
n_classes = info.features["label"].num_classes # 5
dataset_size = info.splits["train"].num_examples # 3670
```

```
>>> info.splits
{'train': <SplitInfo num_examples=3670, num_shards=2>}
```

```
test_set_raw, valid_set_raw, train_set_raw = tfds.load("tf_flowers", as_supervised=True,
    split=["train[:10%]", "train[10%:25%]", "train[25%:]"])

def preprocess(image, label):
    resized_image = tf.image.resize(image, [224, 224])
    final_image = keras.applications.xception.preprocess_input(resized_image)
    return final_image, label

batch_size = 32
train_set = train_set_raw.shuffle(1000).repeat()
#train_set = train_set.map(partial(preprocess, randomize=True)).batch(batch_size).prefetch(1)
train_set = train_set.map(preprocess).batch(batch_size).prefetch(1)
valid_set = valid_set_raw.map(preprocess).batch(batch_size).prefetch(1)
test_set = test_set_raw.map(preprocess).batch(batch_size).prefetch(1)
```

출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

Pretrained Models for Transfer Learning

```
base_model = keras.applications.xception.Xception(weights="imagenet", include_top=False)
avg = keras.layers.GlobalAveragePooling2D()(base_model.output)
output = keras.layers.Dense(n_classes, activation="softmax")(avg)
model = keras.models.Model(inputs=base_model.input, outputs=output)

for layer in base_model.layers:
    layer.trainable = False

optimizer = keras.optimizers.SGD(lr=0.2, momentum=0.9, decay=0.01)
model.compile(loss="sparse_categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
history = model.fit(train_set, epochs=5, validation_data=valid_set,
                    steps_per_epoch=int(0.75 * dataset_size / batch_size),
                    validation_steps=int(0.15 * dataset_size / batch_size))
# 이 결과 accuracy는 0.68에서 0.95로, val_accuracy는 0.84에서 0.88로 증가

for layer in base_model.layers:
    layer.trainable = True

optimizer = keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=True, decay=0.001)
model.compile(loss="sparse_categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
history = model.fit(train_set, epochs=40, validation_data=valid_set,
                    steps_per_epoch=int(0.75 * dataset_size / batch_size),
                    validation_steps=int(0.15 * dataset_size / batch_size))
# 이 결과 accuracy는 0.86에서 0.9985로, val_accuracy는 0.86에서 0.94로 증가
```

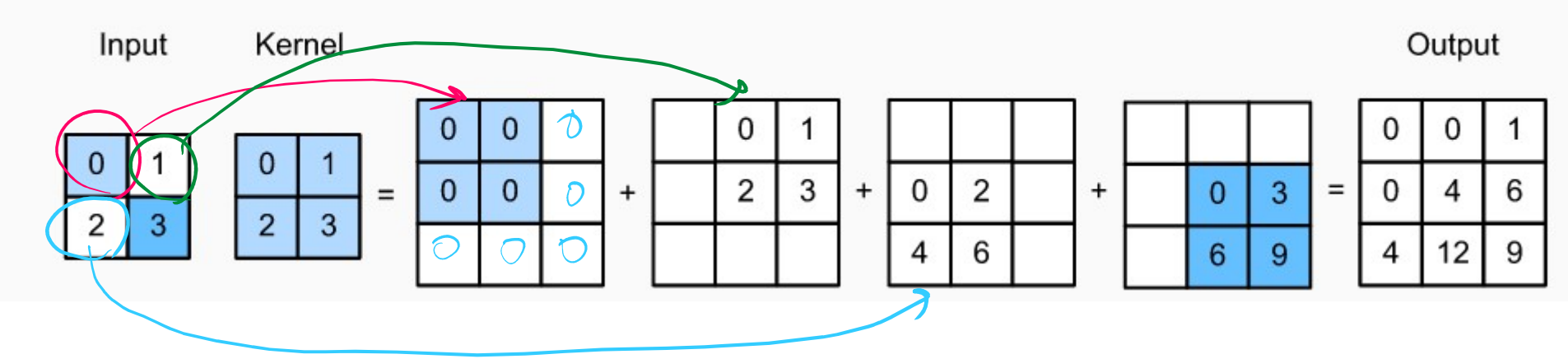
출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

Object Detection

- A (classic) straightforward method
  - Sliding across all 3x3 regions, 4x4 regions, and so on.
  - Cons
    - It will detect the same object multiple times
    - Post-processing to get rid of all the unnecessary bounding boxes needed, which is called "non-max suppression"
    - Slow
- Fully convolutional network (FCN)
  - Jonathan Long et al., "Fully Convolutional Networks for Semantic Segmentation," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015): 3431– 3440.

Upsampling with Transposed Convolution

- 여러 가지 설명 방법:
  - 이미지에 0으로 찬 빈 행과 열을 넣고 출력 크기에 맞추어 stride하면서 convolution 연산을 하는 것
  - Fractional (예: 1/2) strides
  - https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba



- tf.keras API: Conv2DTranspose

Super-Resolution via Transposed Convolution

- https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba

Ground Truth

¼ Sized Input

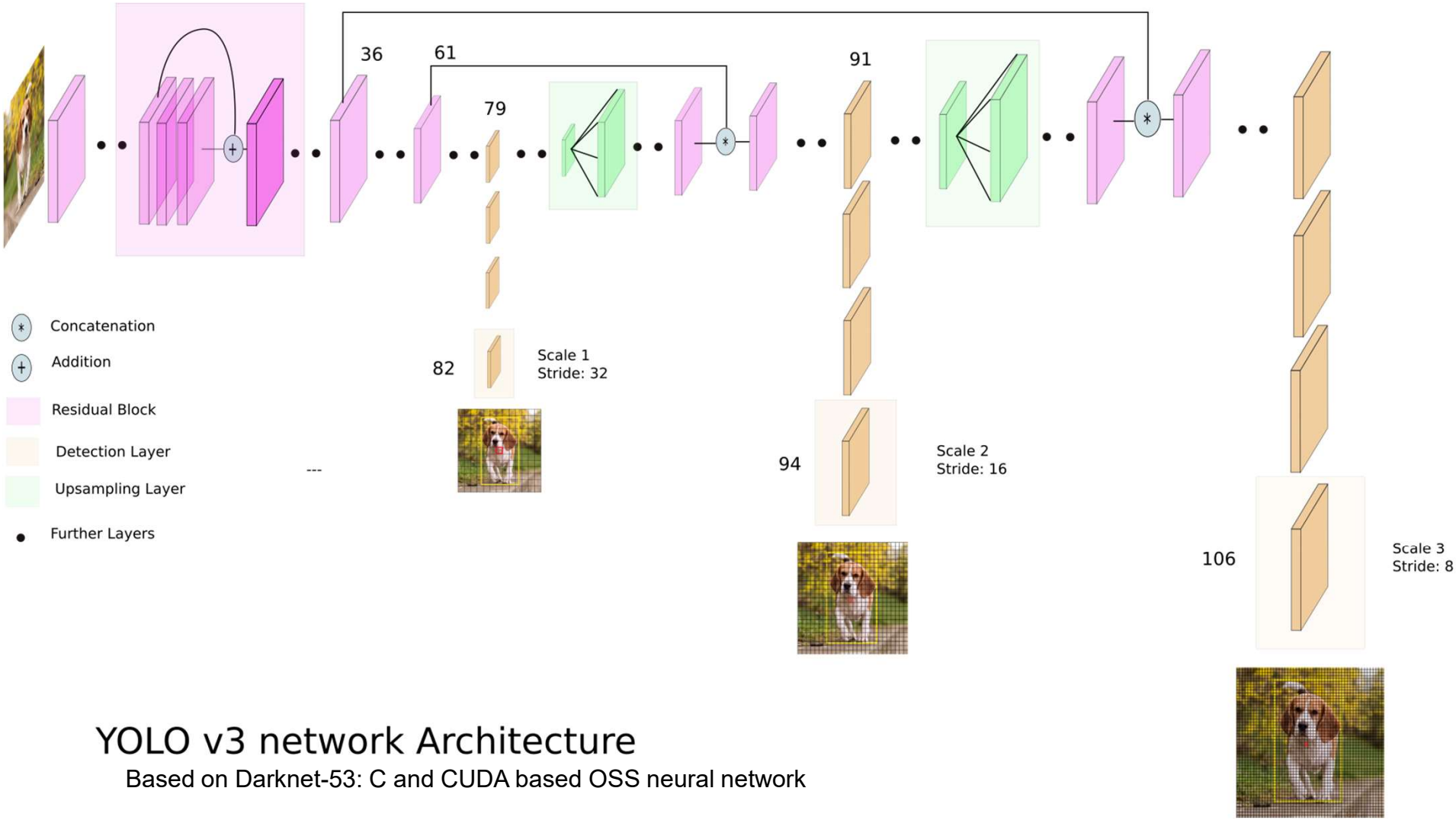
Bicubic

Super Resolution Network

You Only Look Once (YOLO)

- Object Detection
- Joseph Redmon et al., "You Only Look Once: Unified, Real-Time **Object Detection**," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016): 779– 788.
- Joseph Redmon and Ali Farhadi, "YOLO9000: Better, Faster, Stronger," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017): 6517– 6525.
- Ali Farhadi and Joseph Redmon, "YOLOv3: An Incremental Improvement," Vision and Pattern Recognition, 2018.
- Fully Convolutional Networks

YOLOv3 (1)



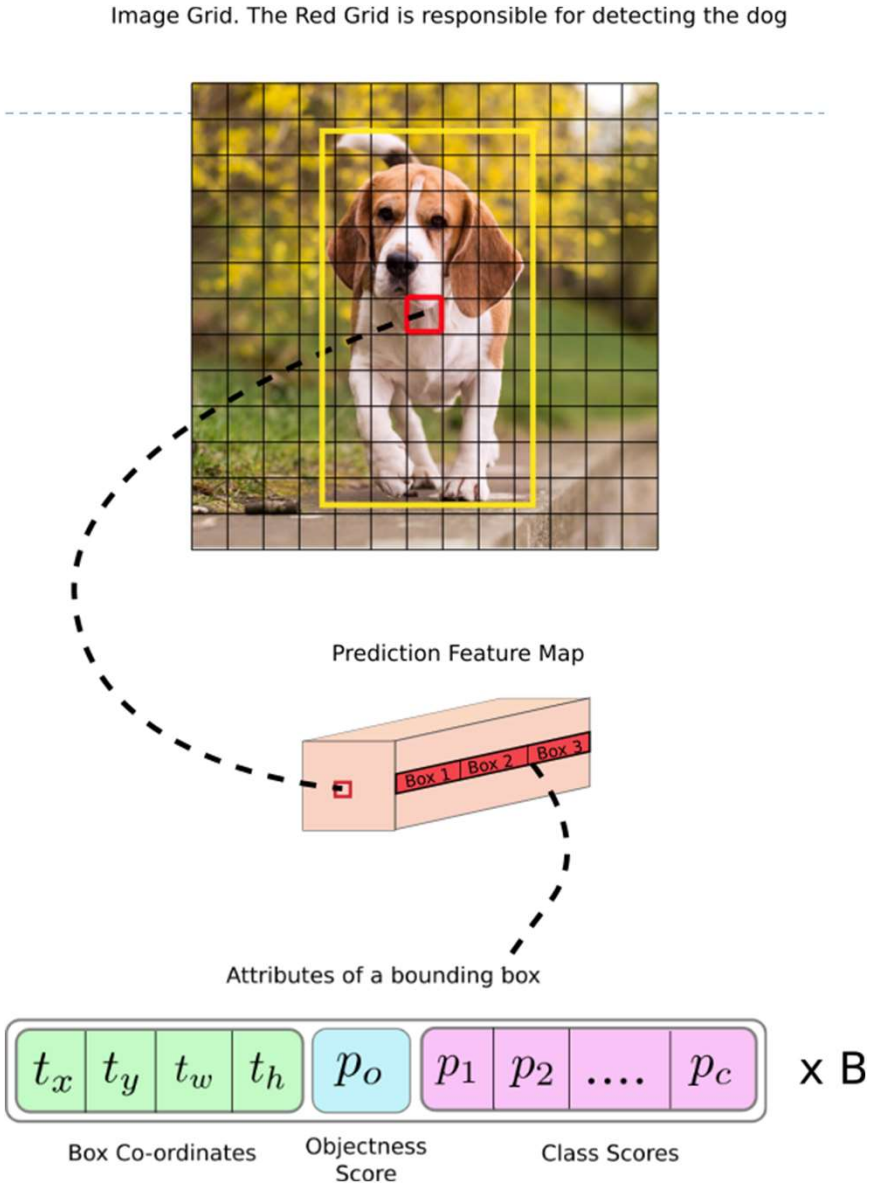
YOLO v3 network Architecture

Based on Darknet-53: C and CUDA based OSS neural network

이미지 출처: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

YOLOv3 (2)

- 3 scales
  - 1 x 1 detection kernels
  - on feature maps of 3 sizes
  - at 3 places in the network
- The shape of the detection kernel
  - 1 x 1 x (B x (5 + C))
  - B: # of bounding boxes (ex: 3)
  - C: # of classes (ex: 80)



https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

YOLOv3 (3)

- No more softmaxing the classes
  - Softmaxing classes rests on the assumption that classes are mutually exclusive.
- YOLOv2's loss function

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

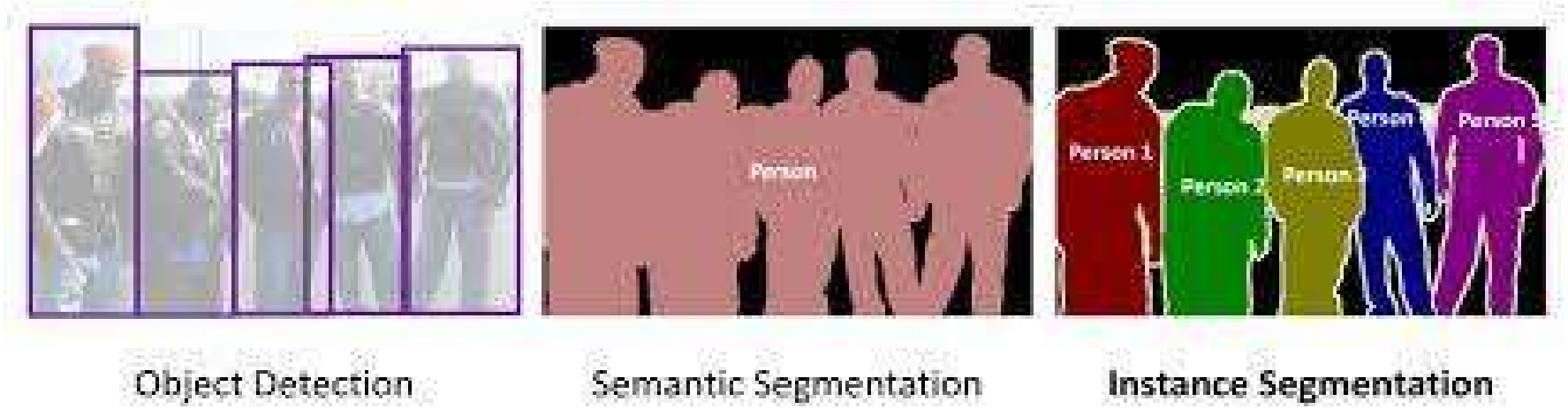
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2 \quad (3)$$

cross-entropy in YOLOv3

Image credits: https://pjreddie.com/media/files/papers/yolo_1.pdf

Instance Segmentation

- 클래스 별 분류가 아니라 객체 별 분할
- 텐서플로 모델 프로젝트에도 포함되어 있음
  - 2017년 Mask R-CNN paper
  - Kaiming He et al., "Mask R-CNN," arXiv preprint arXiv: 1703.06870 (2017).



(이미지 출처: http://www.gisdeveloper.co.kr/?p=8160)