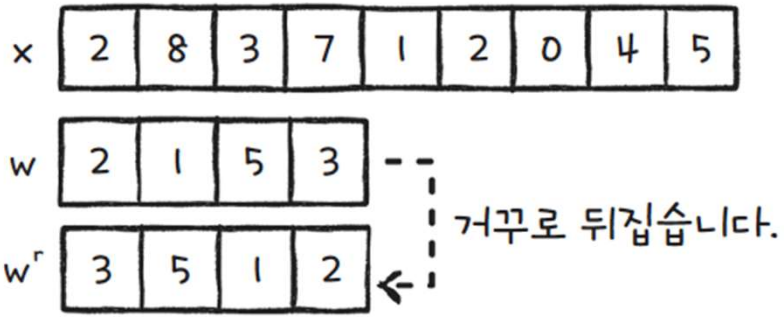
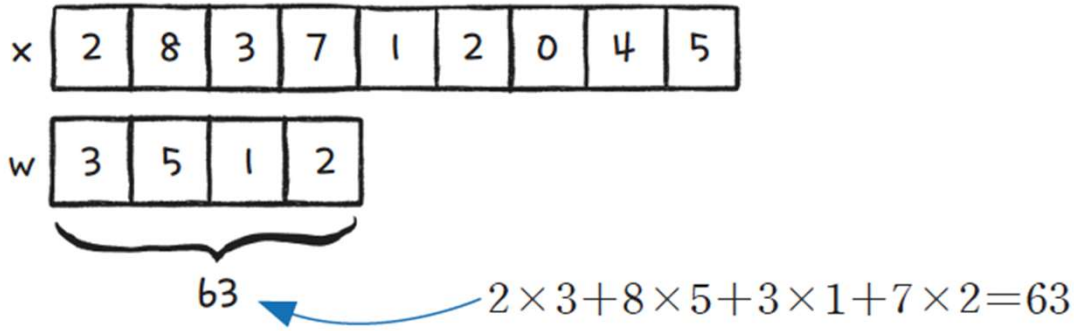
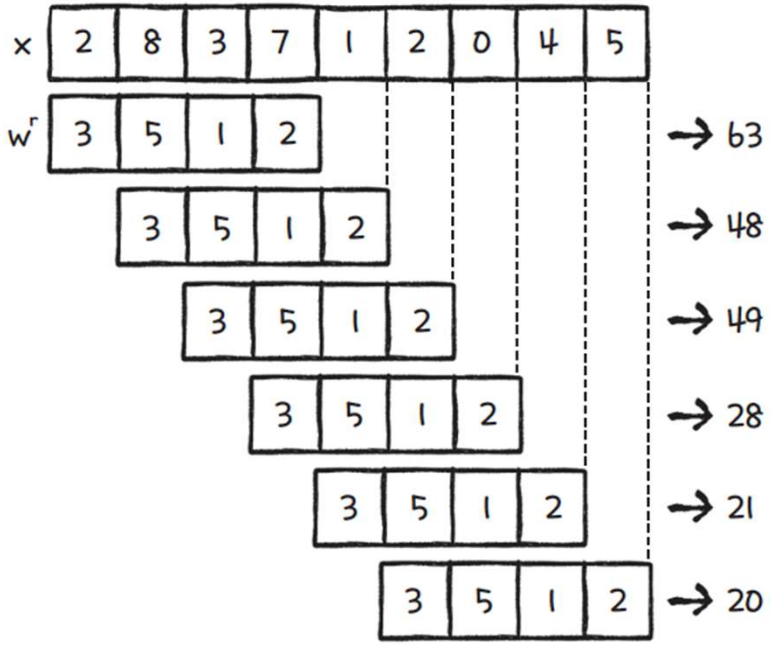
08 이미지를 분류합니다

- 합성곱 신경망

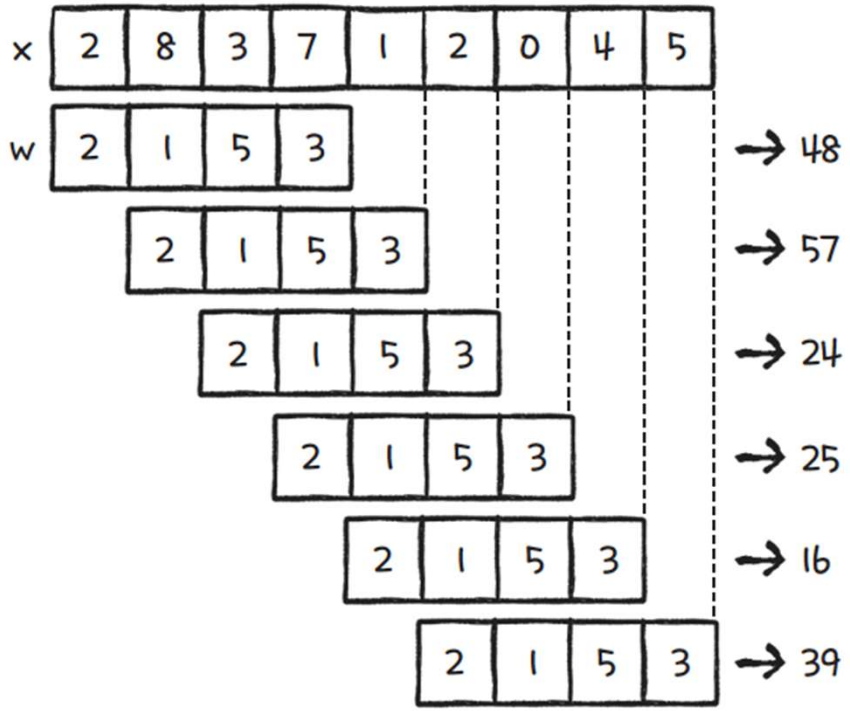
합성곱 연산


x
2 8 3 7 1 2 0 4 5
w
2 1 5 3
w^r
3 5 1 2
거꾸로 뒤집습니다.

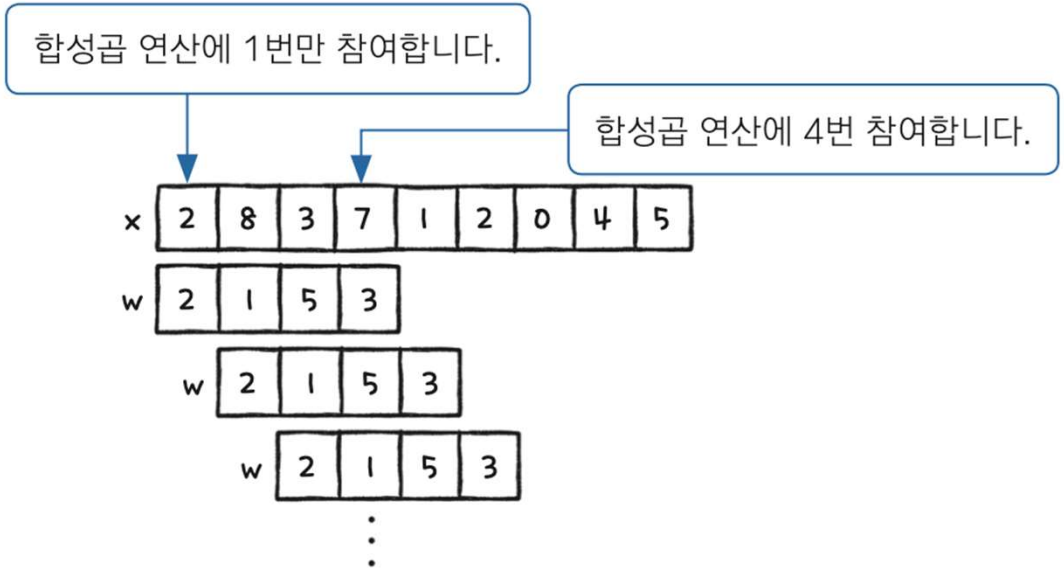

x
2 8 3 7 1 2 0 4 5
w
3 5 1 2
63
2×3+8×5+3×1+7×2=63


x
2 8 3 7 1 2 0 4 5
w^r
3 5 1 2
→ 63
3 5 1 2
→ 48
3 5 1 2
→ 49
3 5 1 2
→ 28
3 5 1 2
→ 21
3 5 1 2
→ 20

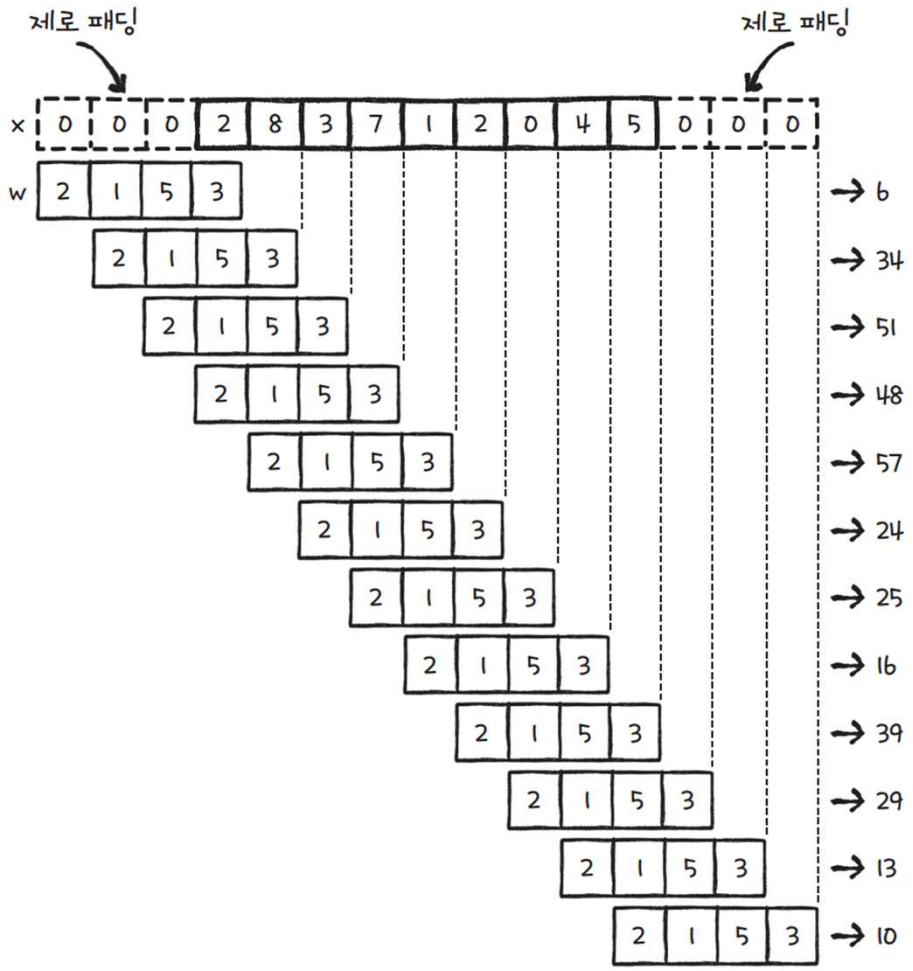
교차 상관 연산



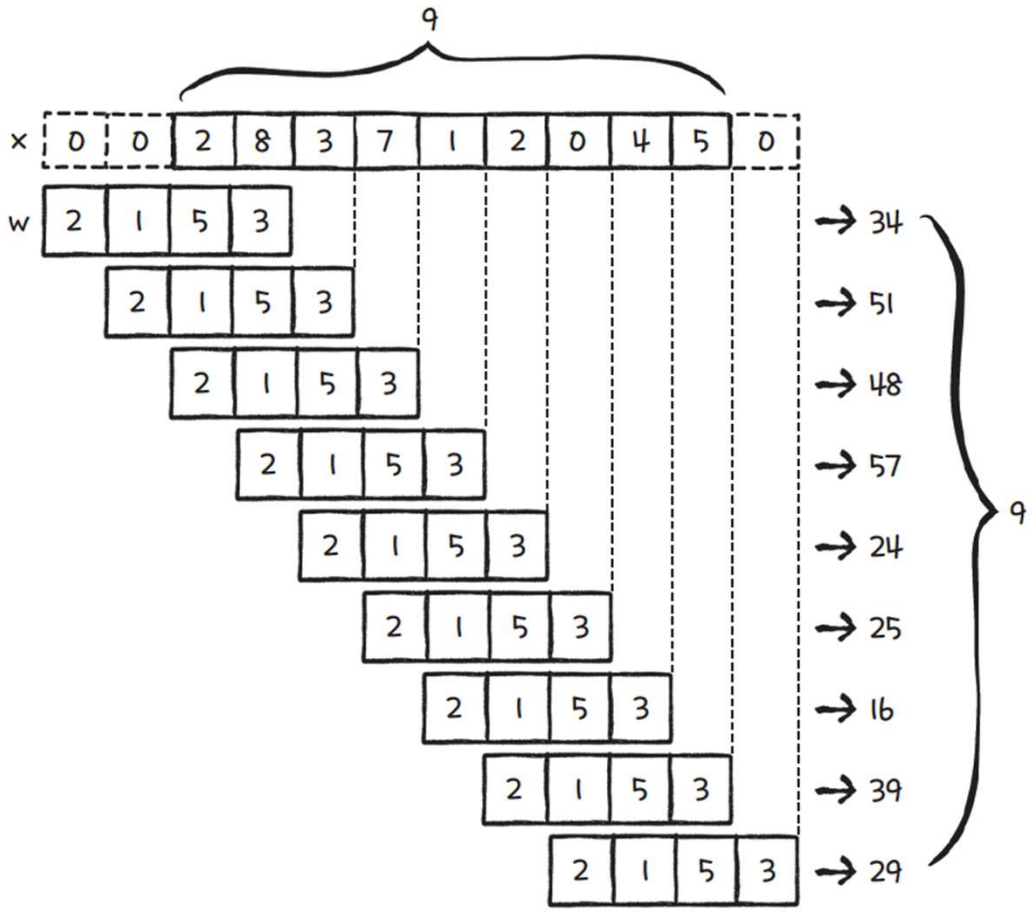
밸리드 패딩


합성곱 연산에 1번만 참여합니다.
합성곱 연산에 4번 참여합니다.
x
2 8 3 7 1 2 0 4 5
w
2 1 5 3
w
2 1 5 3
w
2 1 5 3

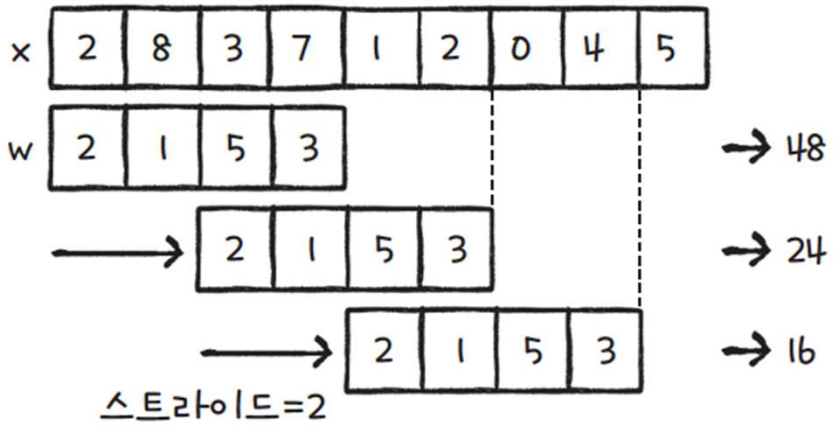
풀패딩


제로 패딩
제로 패딩
x 0 0 0 2 8 3 7 1 2 0 4 5 0 0 0
w 2 1 5 3
→ 6
2 1 5 3
→ 34
2 1 5 3
→ 51
2 1 5 3
→ 48
2 1 5 3
→ 57
2 1 5 3
→ 24
2 1 5 3
→ 25
2 1 5 3
→ 16
2 1 5 3
→ 39
2 1 5 3
→ 29
2 1 5 3
→ 13
2 1 5 3
→ 10

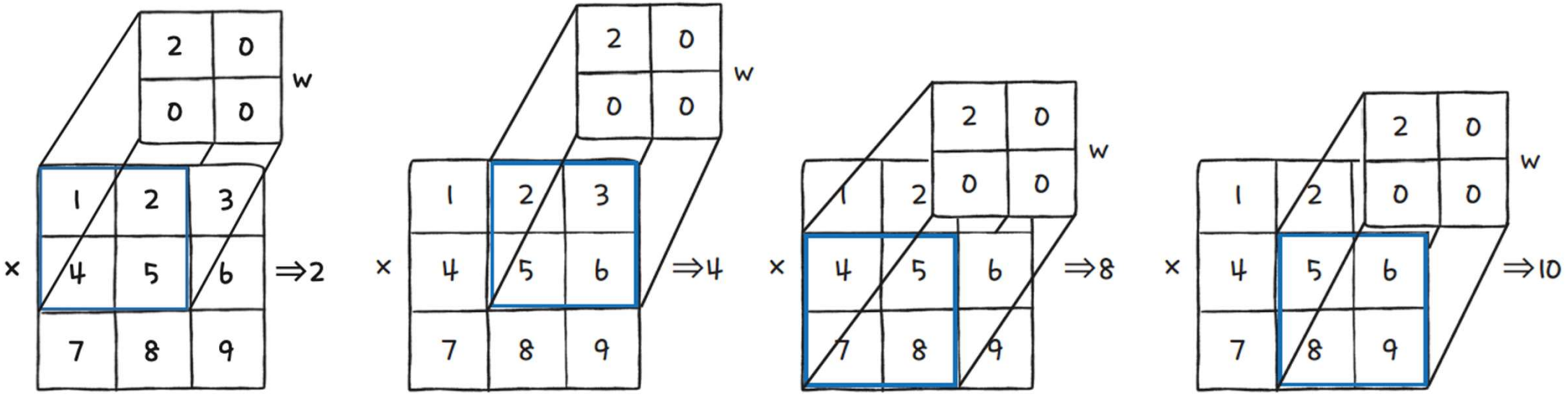
세임 패딩



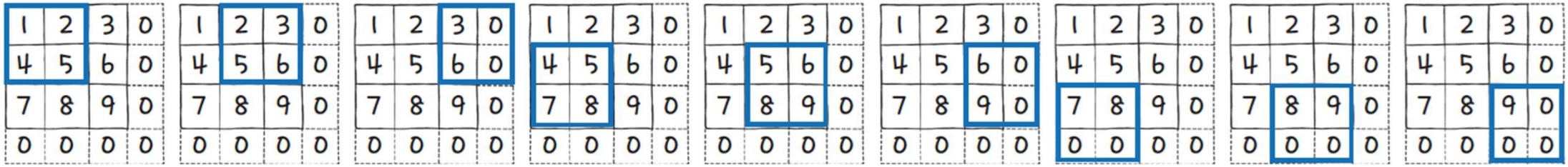
스트라이드



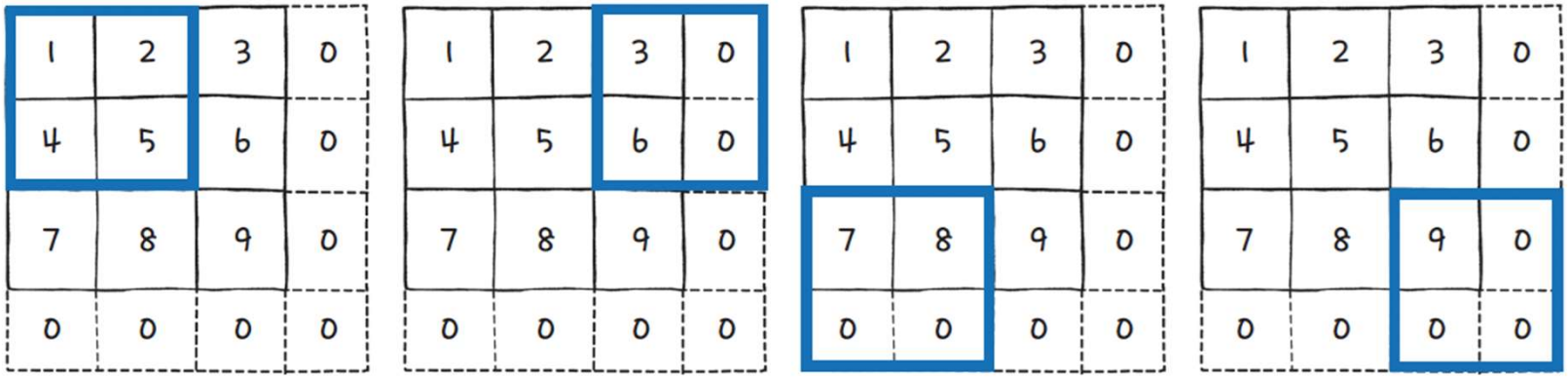
2차원 배열의 합성곱



2차원 배열의 세임 패딩

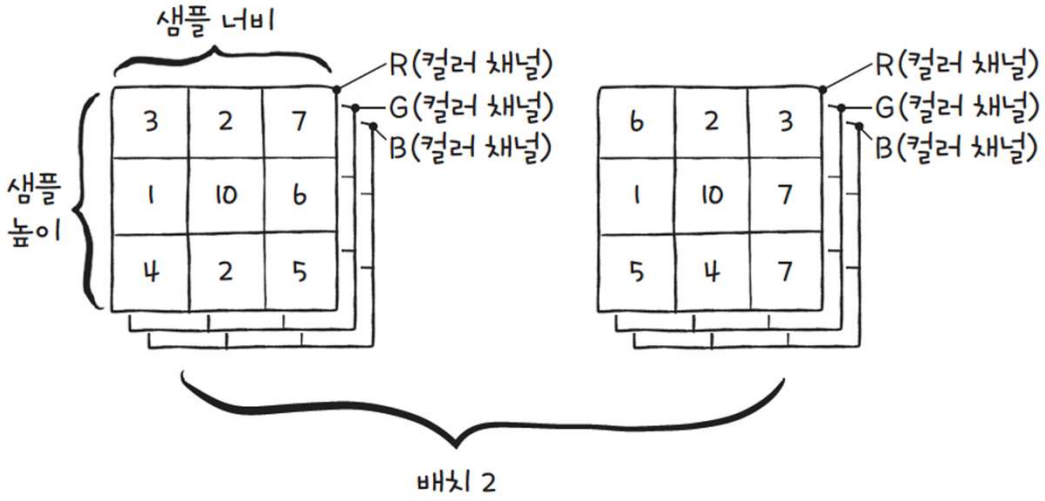


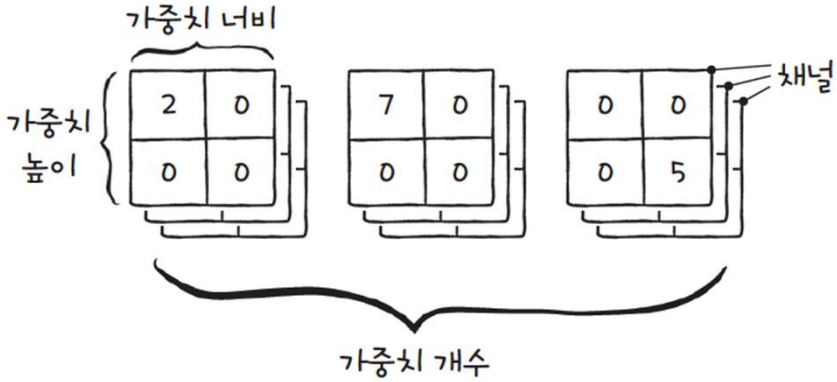
2차원 배열의 스트라이드



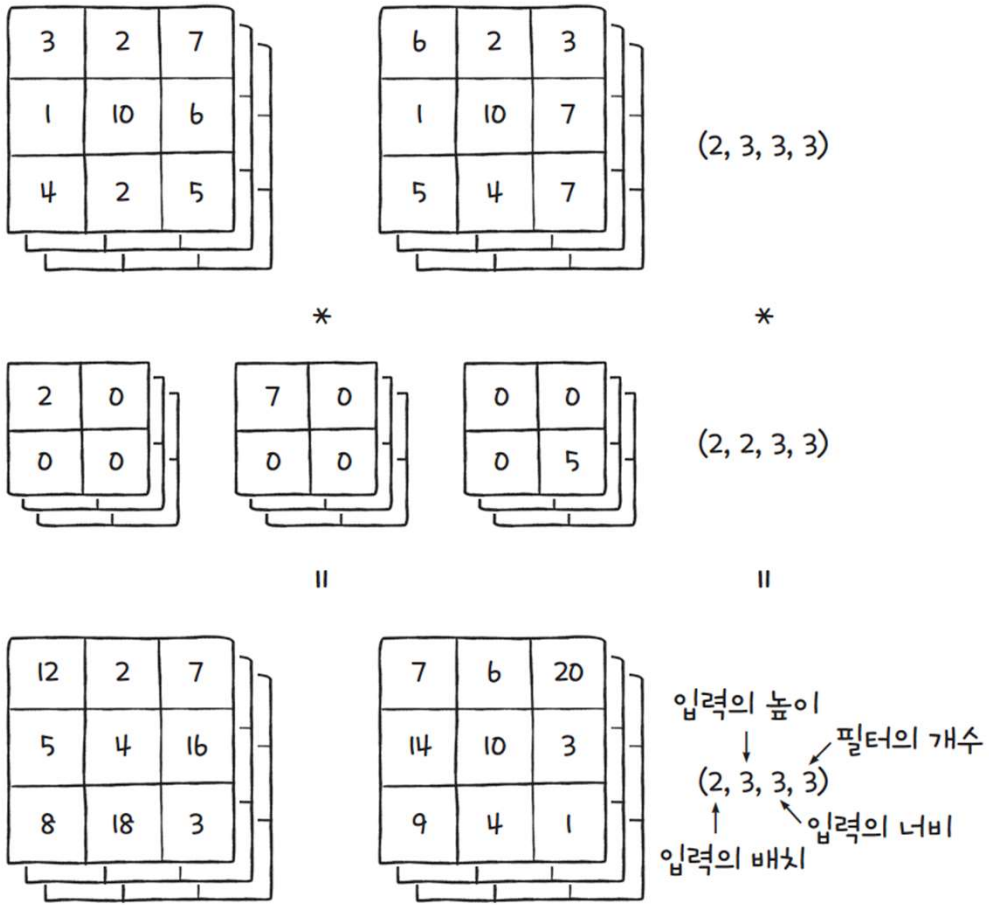
이미지 데이터와 커널

4차원 배열 사용



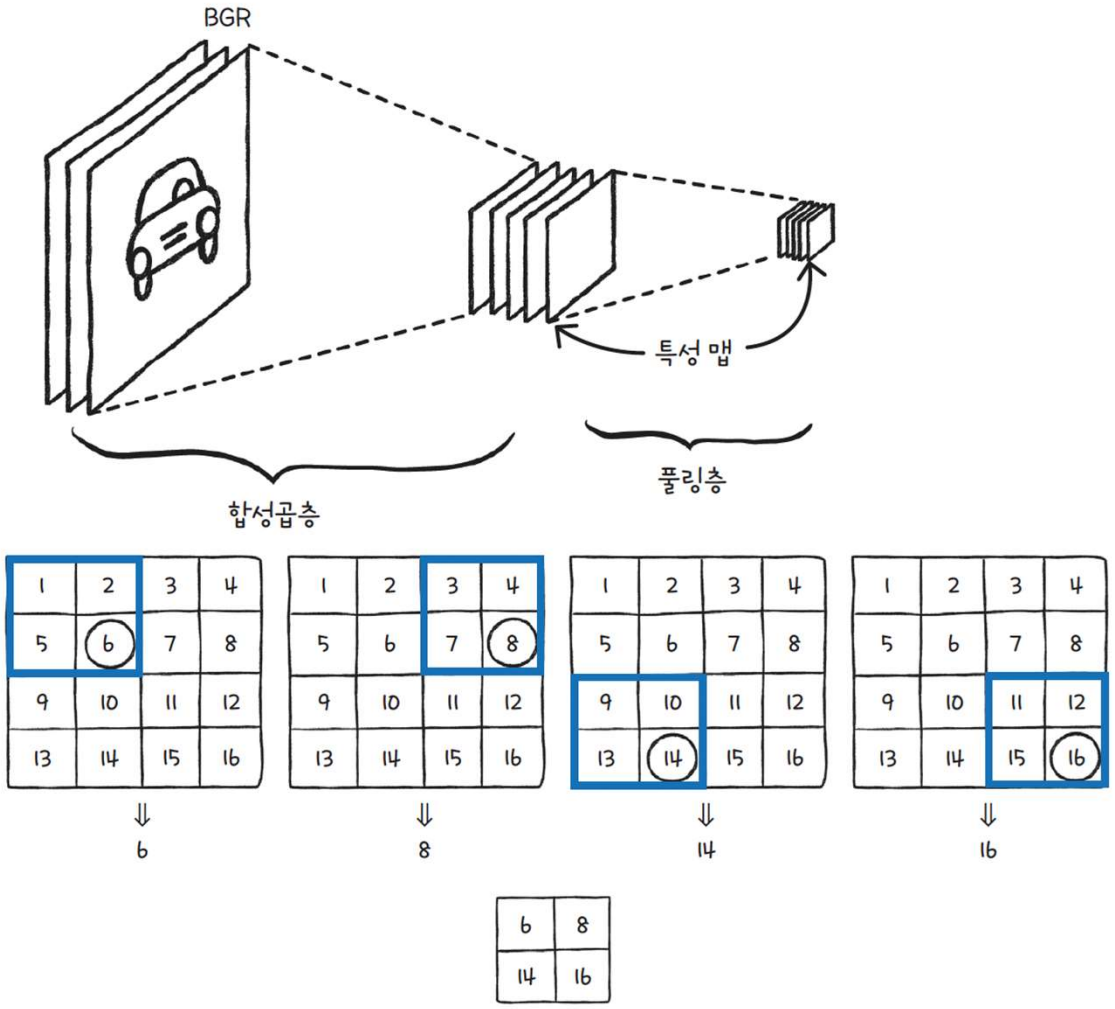


이미지 데이터의 합성곱 연산

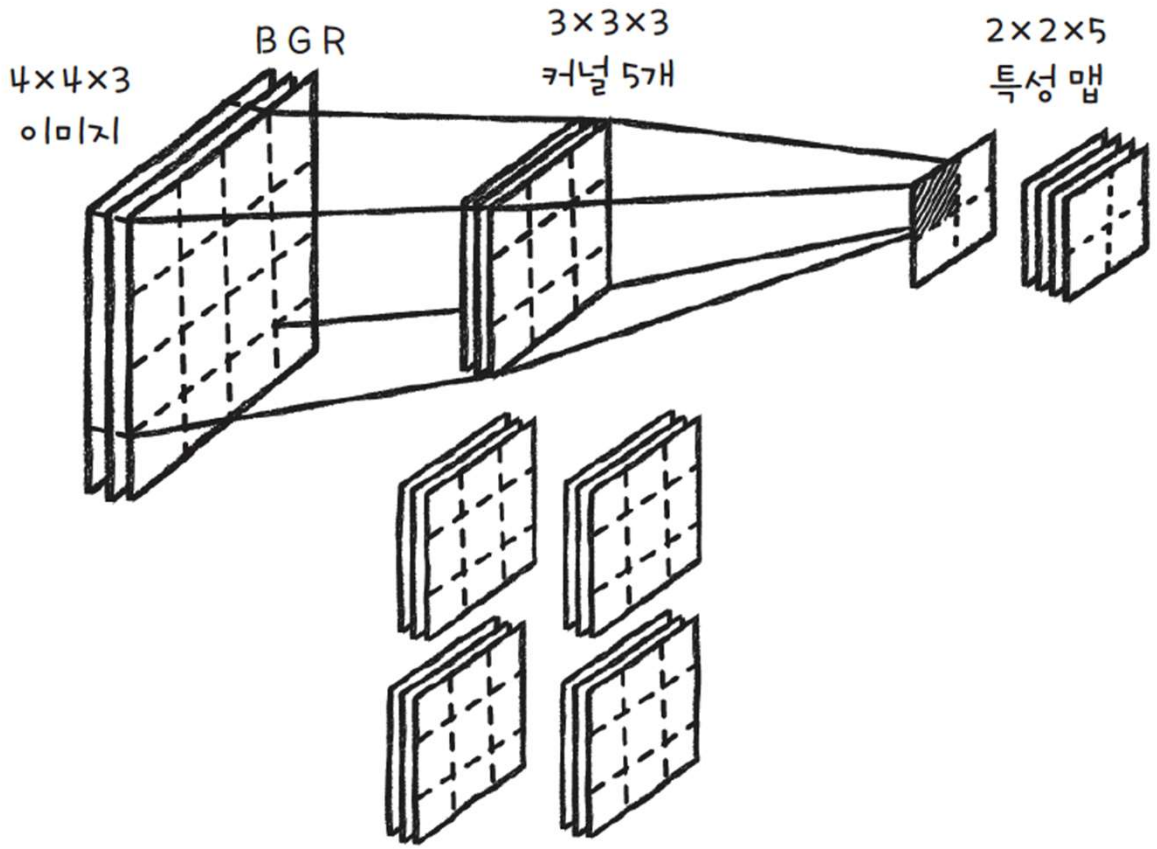


풀링

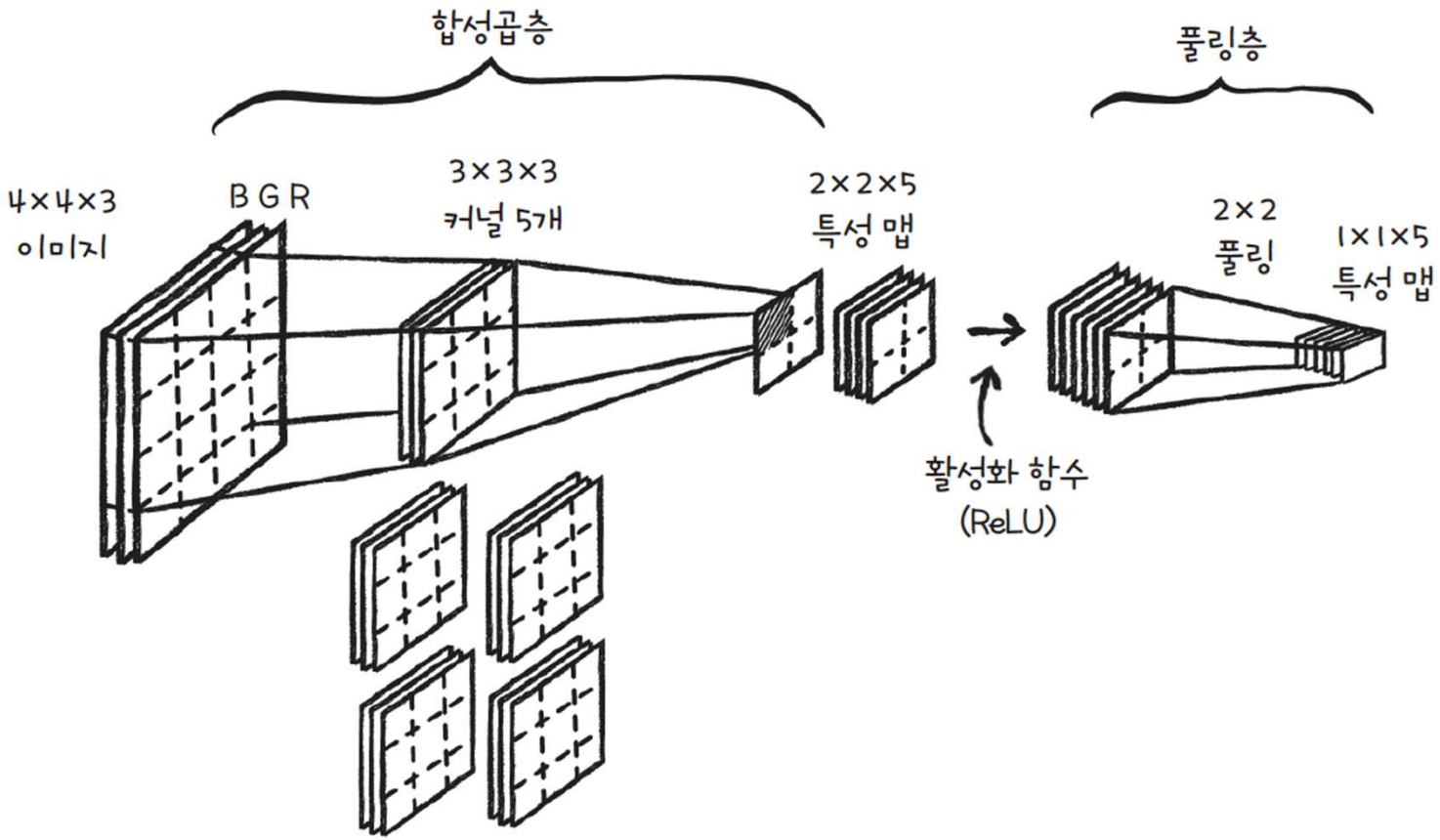
최대 풀링, 평균 풀링


BGR
특성 맵
합성곱층
풀링층
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
6
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
8
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
14
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
16
6 8
14 16

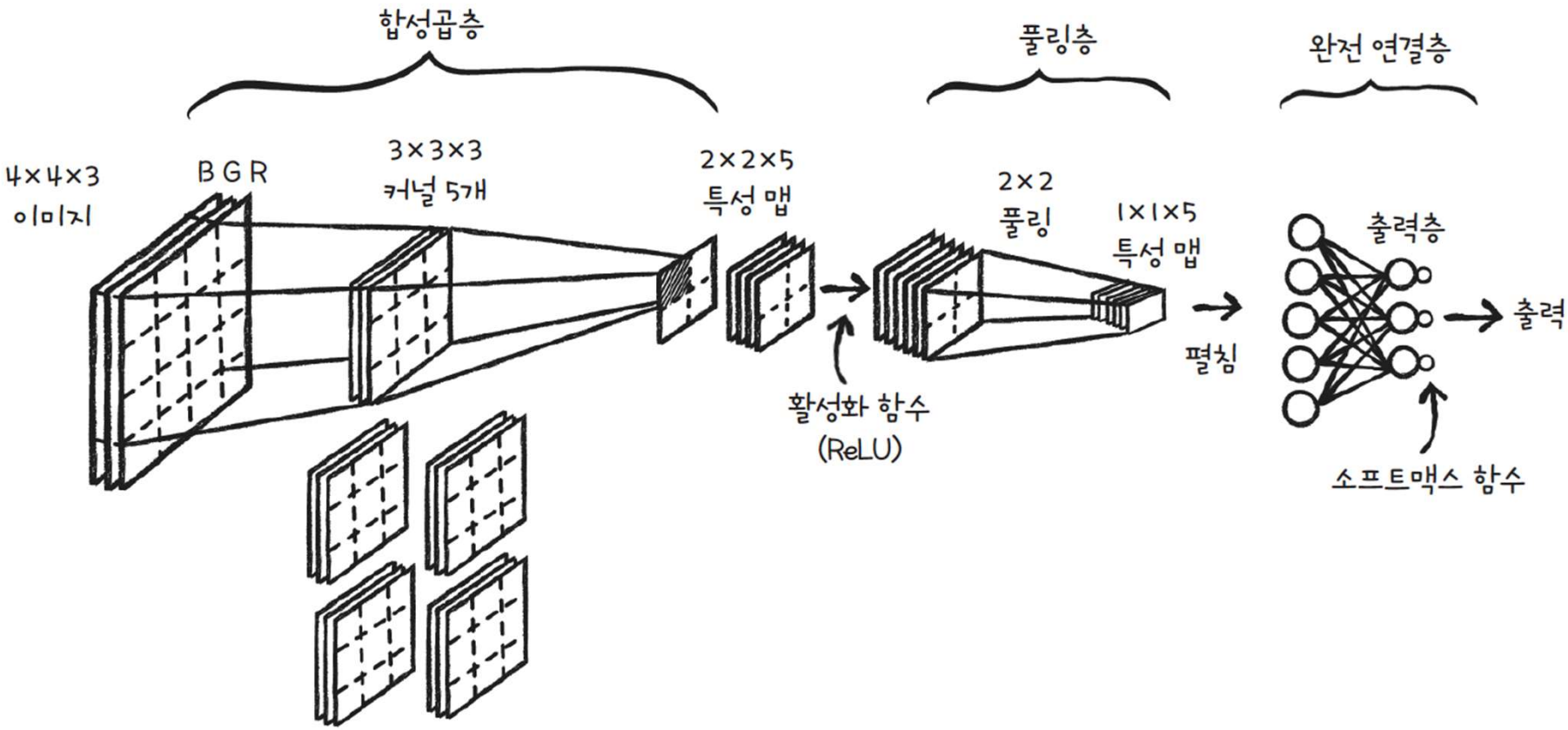
합성곱 층



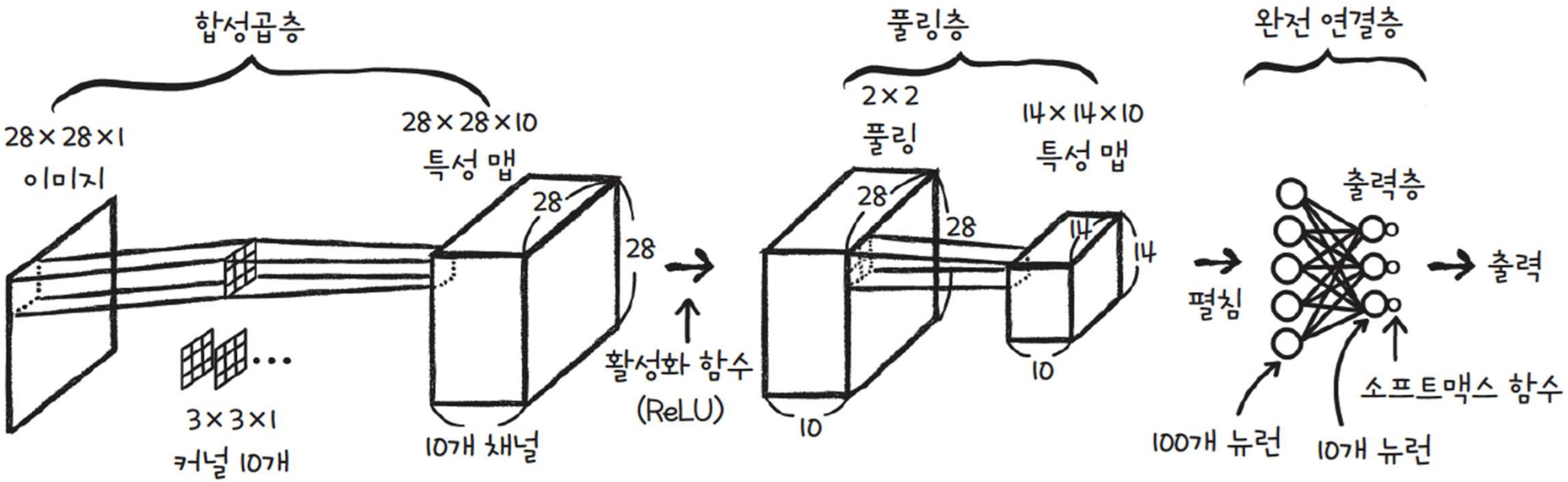
풀링 층



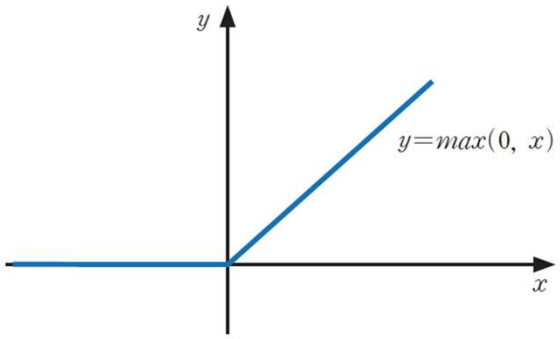
합성곱 신경망


합성곱층
풀링층
완전 연결층
4×4×3
이미지
B G R
3×3×3
커널 5개
2×2×5
특성 맵
2×2
풀링
1×1×5
특성 맵
출력층
출력
펼침
활성화 함수
(ReLU)
소프트맥스 함수

08-4 합성곱 신경망을 만들고 훈련합니다



정방향 계산




```
def forpass(self, x):
    # 3×3 합성곱 연산을 수행합니다.
    c_out = tf.nn.conv2d(x, self.conv_w, strides=1, padding='SAME') + self.conv_b
    # 렐루 활성화 함수를 적용합니다.
    r_out = tf.nn.relu(c_out)
    # 2×2 최대 풀링을 적용합니다.
    p_out = tf.nn.max_pool2d(r_out, ksize=2, strides=2, padding='VALID')
    # 첫 번째 배치 차원을 제외하고 출력을 일렬로 펼칩니다.
    f_out = tf.reshape(p_out, [x.shape[0], -1])
    z1 = tf.matmul(f_out, self.w1) + self.b1      # 첫 번째 층의 선형식을 계산합니다.
    a1 = tf.nn.relu(z1)                           # 활성화 함수를 적용합니다.
    z2 = tf.matmul(a1, self.w2) + self.b2         # 두 번째 층의 선형식을 계산합니다.
    return z2
```

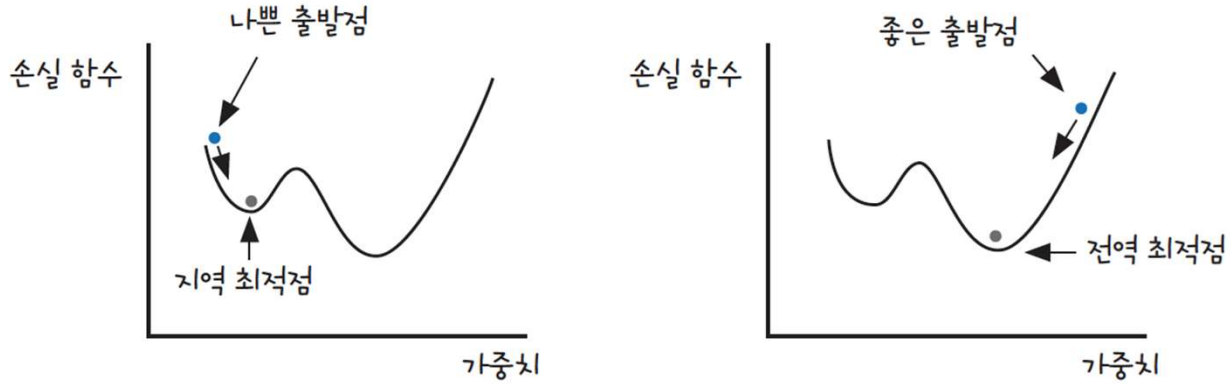
역방향 계산

```
def training(self, x, y):
    m = len(x)                                  # 샘플 개수를 저장합니다.
    with tf.GradientTape( ) as tape:
        z = self.forpass(x)                     # 정방향 계산을 수행합니다.
        # 손실을 계산합니다.
        loss = tf.nn.softmax_cross_entropy_with_logits(y, z)
        loss = tf.reduce_mean(loss)

    weights_list = [self.conv_w, self.conv_b,
                    self.w1, self.b1, self.w2, self.b2]
    # 가중치에 대한 그레이디언트를 계산합니다.
    grads = tape.gradient(loss, weights_list)
    # 가중치를 업데이트합니다.
    self.optimizer.apply_gradients(zip(grads, weights_list))
```

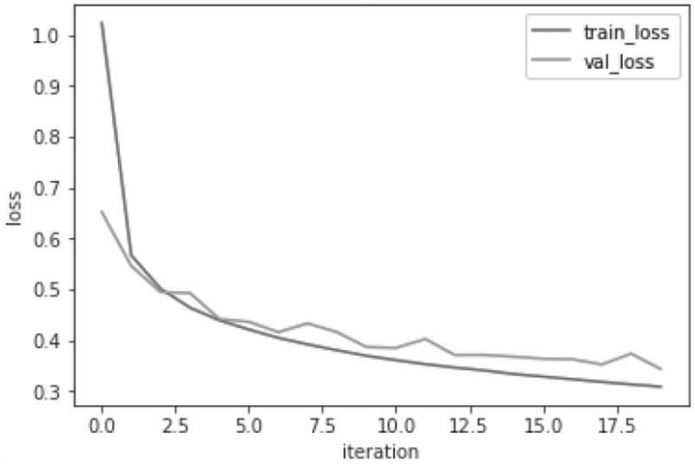
가중치 초기화

```
def init_weights(self, input_shape, n_classes):
    g = tf.initializers.glorot_uniform( )
    self.conv_w = tf.Variable(g((3, 3, 1, self.n_kernels)))
    self.conv_b = tf.Variable(np.zeros(self.n_kernels), dtype=float)
    n_features = 14 * 14 * self.n_kernels
    self.w1 = tf.Variable(g((n_features, self.units)))        # (특성 개수, 은닉층의 크기)
    self.b1 = tf.Variable(np.zeros(self.units), dtype=float) # 은닉층의 크기
    self.w2 = tf.Variable(g((self.units, n_classes)))         # (은닉층의 크기, 클래스 개수)
    self.b2 = tf.Variable(np.zeros(n_classes), dtype=float)  # 클래스 개수
```



ConvolutionNetowrk 훈련

```
cn = ConvolutionNetwork(n_kernels=10, units=100, batch_size=128, learning_rate=0.01)
cn.fit(x_train, y_train_encoded, x_val=x_val, y_val=y_val_encoded, epochs=20)
에포크 0 ..................................
에포크 1 ..................................
.
.
.
에포크 19 .................................
```



케라스로 합성곱 신경망 만들기

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
```

```
conv1 = tf.keras.Sequential()
conv1.add(Conv2D(10, (3, 3), activation='relu', padding='same', input_shape=(28, 28, 1)))
conv1.add(MaxPooling2D((2, 2)))
conv1.add(Flatten())
conv1.add(Dense(100, activation='relu'))
conv1.add(Dense(10, activation='softmax'))
```

```
conv1.summary()
```

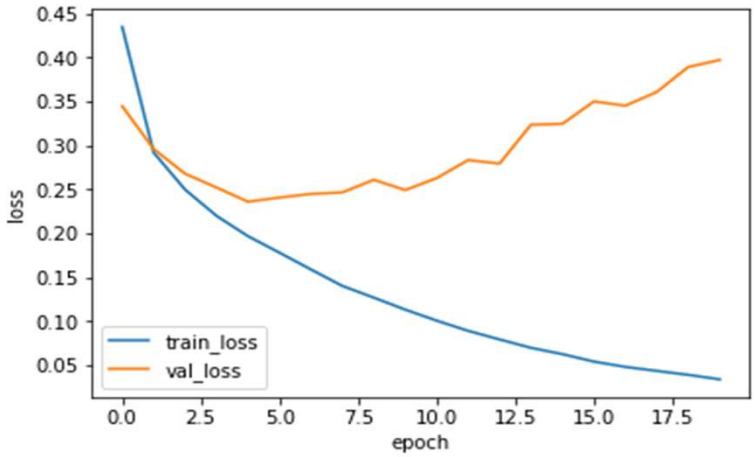
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 10)        100
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 10)        0
_________________________________________________________________
flatten (Flatten)            (None, 1960)              0
_________________________________________________________________
dense (Dense)                (None, 100)               196100
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1010
=================================================================
Total params: 197,210
Trainable params: 197,210
Non-trainable params: 0
_________________________________________________________________
```

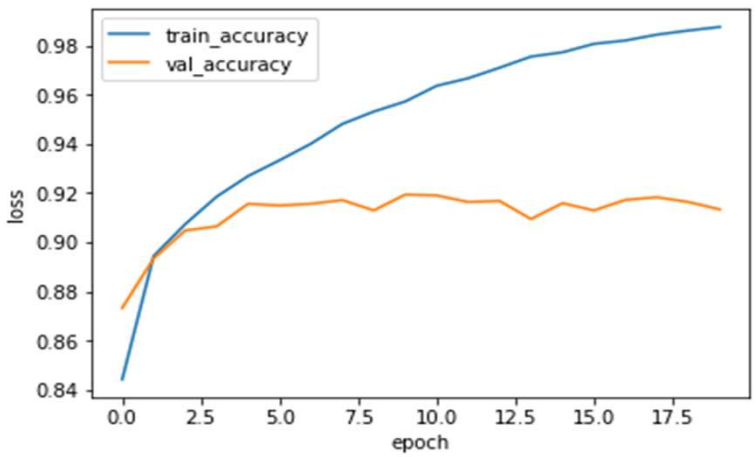
케라스 모델 훈련하기

```
conv1.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])
```

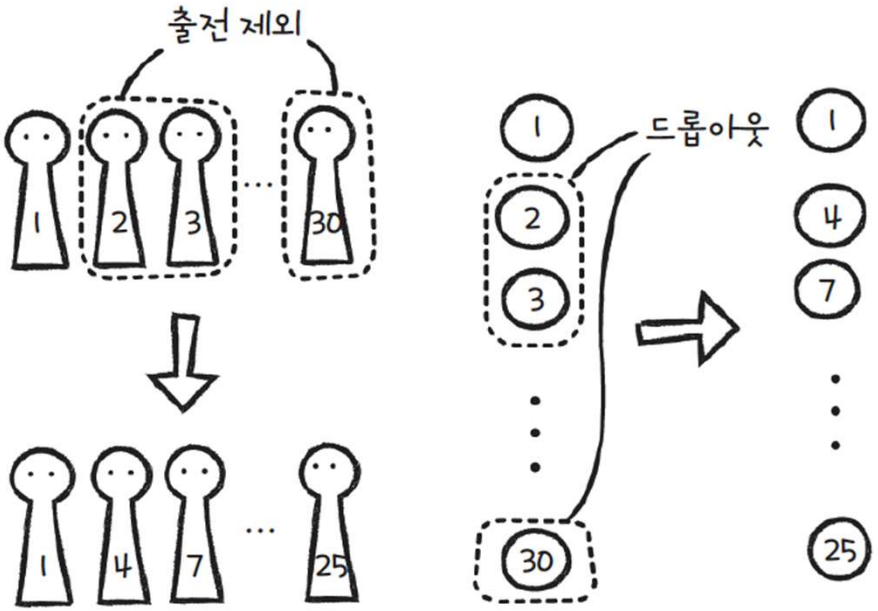
```
history = conv1.fit(x_train, y_train_encoded, epochs=20,
                    validation_data=(x_val, y_val_encoded))
```

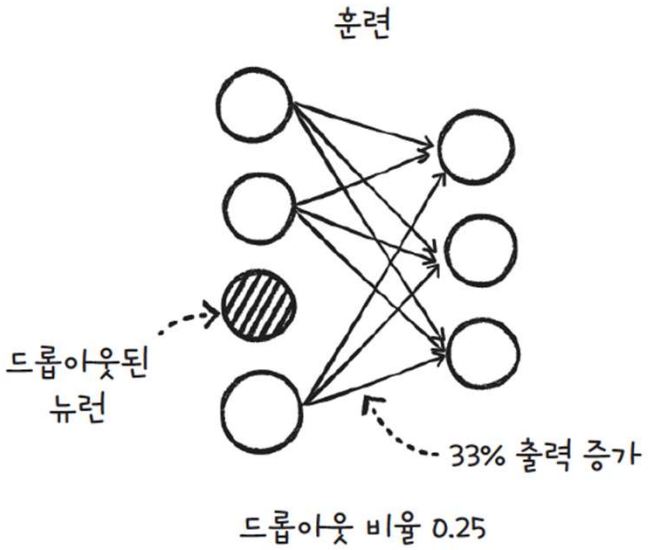
```
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 6s 120us/sample - loss: 0.4349 - accuracy: 0.84
Epoch 2/20
48000/48000 [==============================] - 5s 100us/sample - loss: 0.2920 - accuracy: 0.89
```

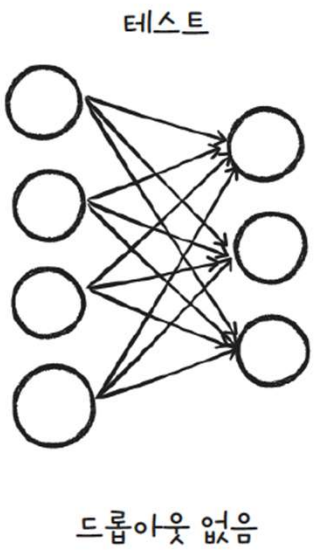




드롭아웃


출전 제외
1
2
3
30
1
4
7
25
1
2
3
30
드롭아웃
1
4
7
25


훈련
드롭아웃된
뉴런
33% 출력 증가
드롭아웃 비율 0.25


테스트
드롭아웃 없음

신경망에 드롭아웃 추가하기

```
conv2 = tf.keras.Sequential()
conv2.add(Conv2D(10, (3, 3), activation='relu', padding='same', input_shape=(28, 28, 1)))
conv2.add(MaxPooling2D((2, 2)))
conv2.add(Flatten())
conv2.add(Dropout(0.5))
conv2.add(Dense(100, activation='relu'))
conv2.add(Dense(10, activation='softmax'))
```

```
conv2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 10)        100
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 10)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1960)              0
_________________________________________________________________
dropout (Dropout)            (None, 1960)              0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               196100
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1010
=================================================================
Total params: 197,210
Trainable params: 197,210
Non-trainable params: 0
_________________________________________________________________
```

신경망에 드롭아웃 추가하기