3. 합성망 모형

Hosik Choi

Dept. of Applied Statistics, Kyonggi Univ.

June, 2019

Outline

- 이미지 학습을 위한 NN
  - 동물의 시각 인지 과정을 모방한 모형으로 이미지 분류 문제에 특히 높은 성능을 나타냄
  - convolutional layer, pooling layer, convolutional layer, pooling layer, ..., fully-connected layer, output layer의 순서로 네트워크가 이루어짐
  - 일반적으로 역전파 알고리즘을 이용하여 모수 추정
  - Convolution: dimension reduction
  - 고차원 입력벡터의 차원을 축소
  - 예. gray scale의 image 자료의 경우 $224^2 = 50176$

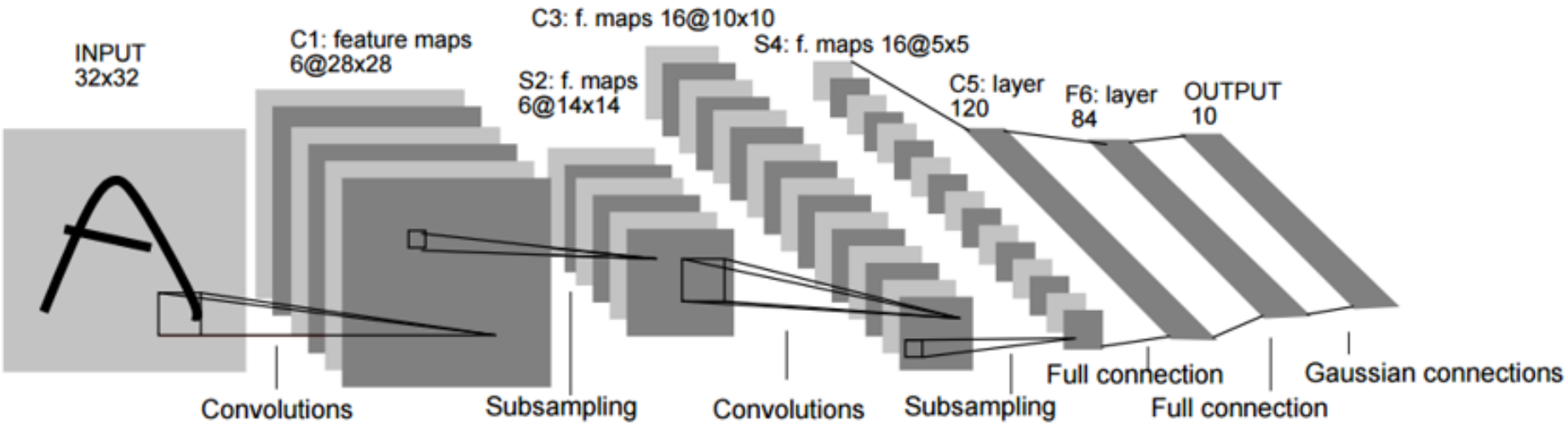


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

CNN의 간단한 역사

- 1983년에 CNN의 시초라 할 수 있는 모델인 Neocognitron (K.Fukushima, 1983)이 발표되었음.
- 1998년, Y.LeCun이 CNN의 표준이 되는 모델인 LeNet-5를 발표. (Y. LeCun et al., 1998)
- 2012년에는 ILSVRC(ImageNet Large-Scale Visual Recognition Challenge)-2012에 A.Krizhevsky et al. (2012)가 CNN을 이용한 딥러닝 모형(Alex net)을 개발하여 압도적인 성능으로 우승을 함에 따라 CNN에 대한 연구가 폭발적으로 증가.
- 이후로 Zeiler net (M.D.Zeiler and R.Fergus, 2014), GoogLenet (C.Szegedy et al., 2015), Resnet (K.He et al., 2016) 등 우수한 성능을 가진 CNN 모형들이 개발.

AlexNet(A. Krizhevsky et. al., 2012) I

- A deep CNN to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes.
- It uses ReLU, dropout, local response normalization and data augmentation techniques.
- It achieved top-I and top-5 error rates of 37.5% and 15% which is considerably better than the previous state-of-the-art.

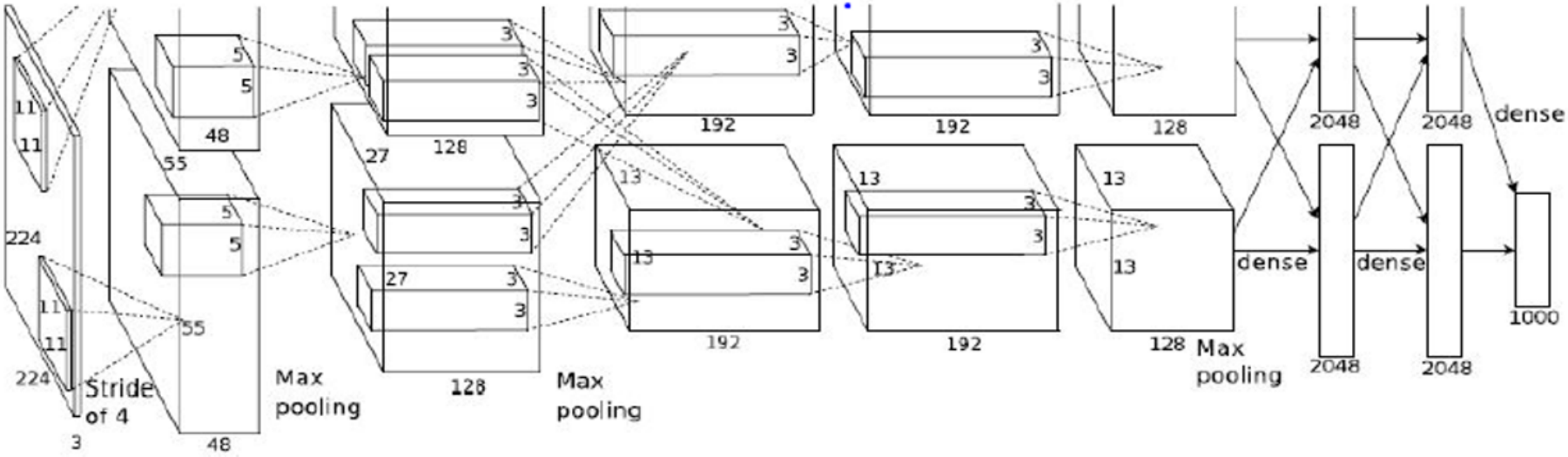

224
224
11
11
3
Stride
of 4
55
55
5
5
48
48
Max
pooling
27
27
128
128
Max
pooling
13
13
192
192
13
13
192
192
13
13
128
128
Max
pooling
dense
dense
2048
2048
2048
2048
dense
1000

Table: 역대 ILSVRC 우승 모델 정보

연도	모델	계층 수	Top-5 오류율
2012	AlexNet	8	15.3%
2013	ZF Net	8	14.8%
2014	GoogLeNet	22	6.67%
2015	ResNet	152	3.57%

- 층 깊이 따른 추출 정보
  - 계층이 깊어질수록 강하게 반응하는 뉴런은 더 추상화됨
- AlexNet
  - 층1: 단순 엣지 층2: 줄무늬 층3: 복잡한 사물의 일부

CNN 구조 I

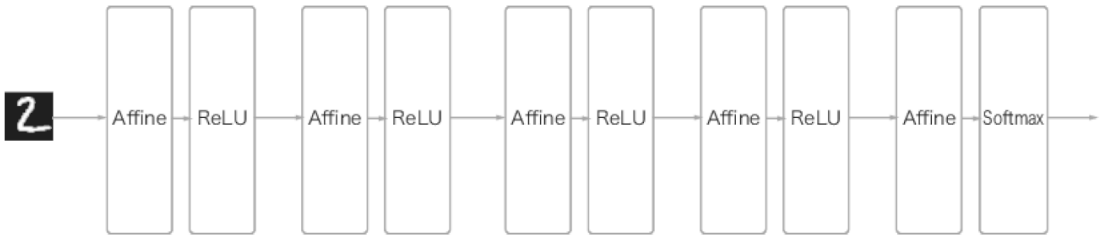


Figure: 완전연결(affine) 층이 활용 DNN

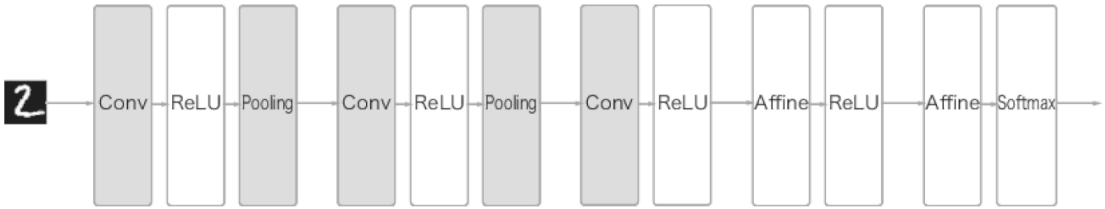


Figure: CNN: 합성곱과 풀링층이 추가(회색)

- 완전연결계층의 문제점
  - 이미지의 경우 데이터의 형상이 무시(상관구조 고려안됨)
  - 3차원 이미지를 1차원 데이터로 평탄화 ($28 \times 28 \longrightarrow 784$)
  - 인접한 공간의 픽셀값의 관련성 반영할 필요 있음
- 합성곱 계층은 입력자료의 형상을 유지함.
  - 합성곱 계층의 입출력 데이터를 특징맵(feature map)이라 부름.

합성곱(convolution) I

CNN의 두요소:

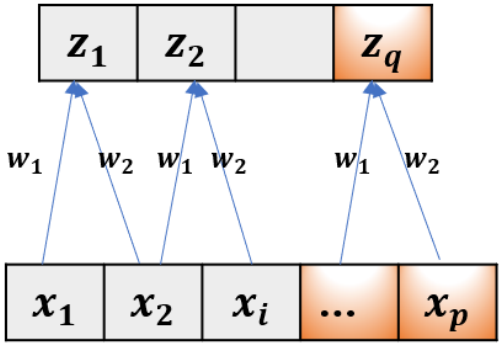
1. 합성곱: 특징 추출(feature map)
   - 합성곱층(convolution layer)
   - 풀링층(pooling layer)
2. 완전연결 신경망

합성곱(컨볼루션)의 정의 (https://brunch.co.kr/@chris-song/24)

$$(f \cdot g)(j) = \sum_i f(i)g(j-i) = \sum_i f(j-i)g(j)$$

국소연결가중치 공유 신경망

$$\begin{bmatrix} z_1 = w_1 x_1 + w_2 x_2 \\ z_2 = w_1 x_2 + w_2 x_3 \\ \vdots \\ z_{p-1} = w_1 x_{p-1} + w_2 x_p \end{bmatrix}$$



그러므로 다음이 성립.

$$\begin{aligned} z_j &= w_1 x_j + w_2 x_{j+1} \\ &= \sum_{i=1}^{2} w_i x_{j+i-1} \\ &= \tilde{w}_2 x_j + \tilde{w}_1 x_{j+1} \\ &= \sum_{i=1}^{2} \tilde{w}_{j-i+1} x_{j+i-1}, j = p - 2 + 1 \end{aligned}$$

여기서, $\tilde{\mathbf{w}} = (\tilde{w}_1, \tilde{w}_2) = (w_2, w_1)$.

- 크기가 l인 필터 적용할 경우

$$\begin{aligned} z_j &= \sum_{i=1}^{l} w_{j-i+1} x_{j+i-1} \\ &= (\mathbf{x} \circ \tilde{w})(j), j = 1, \cdots, p - l + 1 \end{aligned}$$

- CNN은 DNN에서 weight들이 sparse하고 weight의 값을 공유하는 architecture로 이해할 수 있다.

2차원 합성곱 연산 I

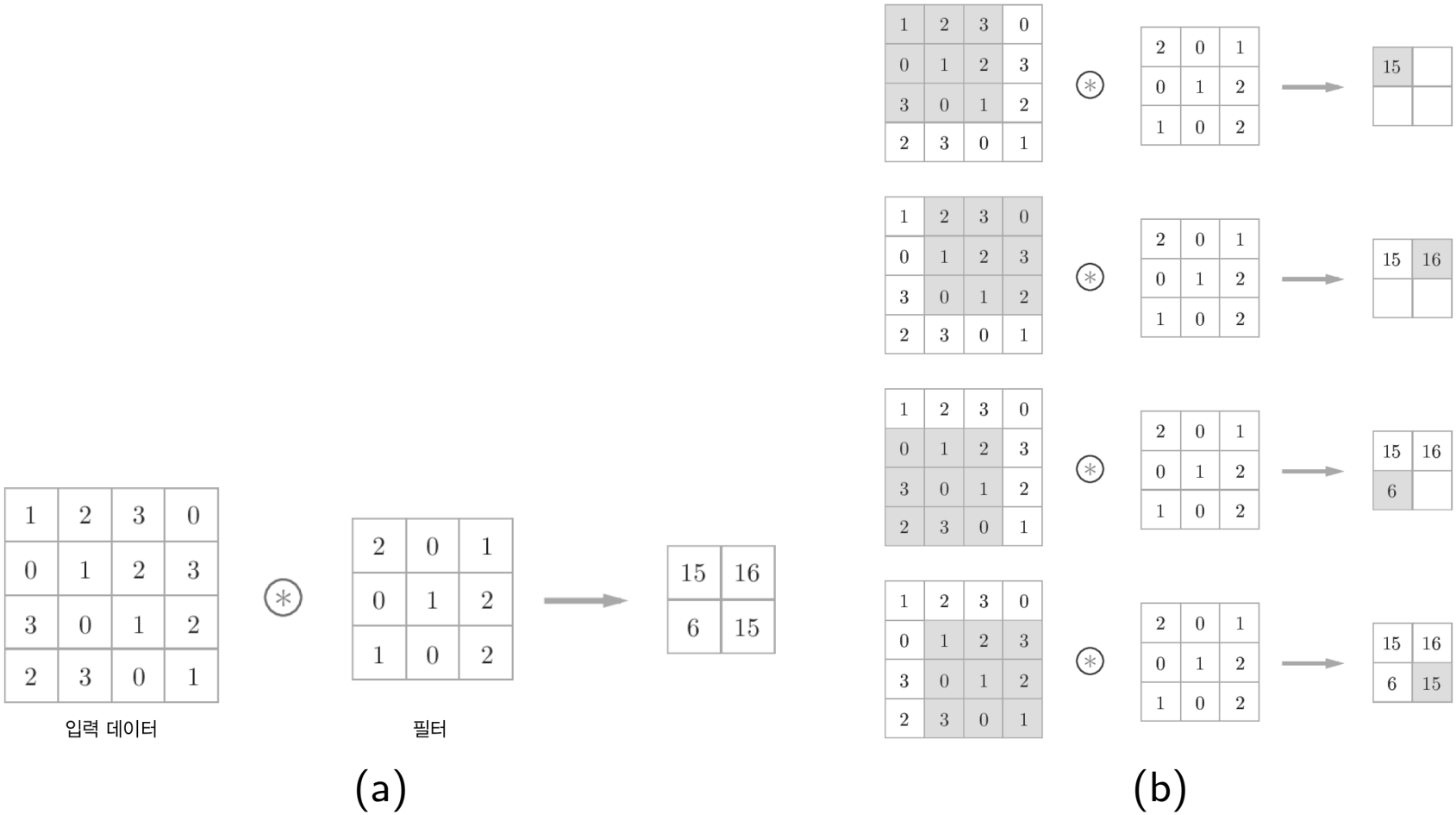


Figure: https://github.com/WegraLee/deep-learning-from-scratch. 참고: http://cs231n.github.io/convolutional-networks/

패딩(padding)

- (4,4) 입력자료에 (3,3) 필터 적용시 출력은 입력보다 각각 2가 줄어든 (2,2)가 됨.
- 합성곱 반복 적용시 차원이 작아서 합성곱 연산잉 적용할 수 없게 됨.
- 원래 차수를 가급적 유지하기 위한 방안으로 입력자료의 외곽에 0값을 가진 가상의 자료를 패딩함.

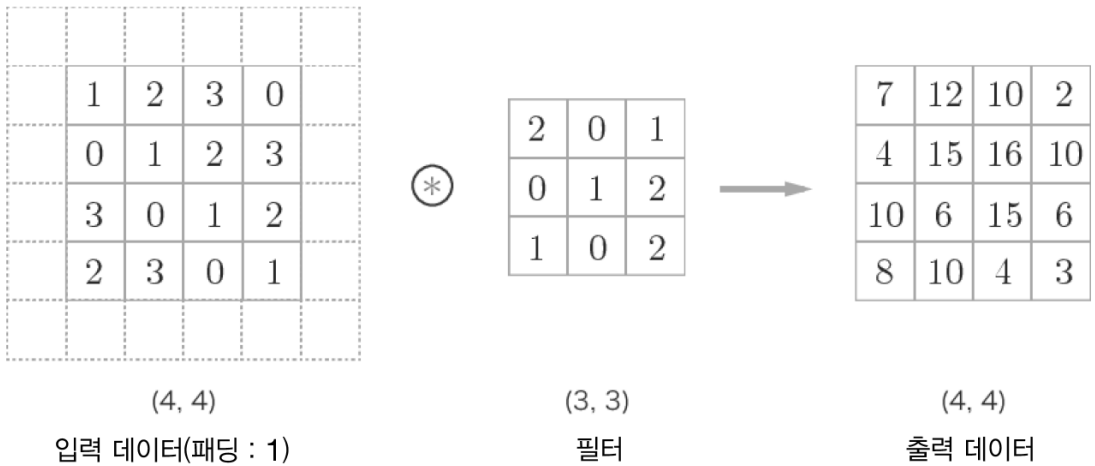
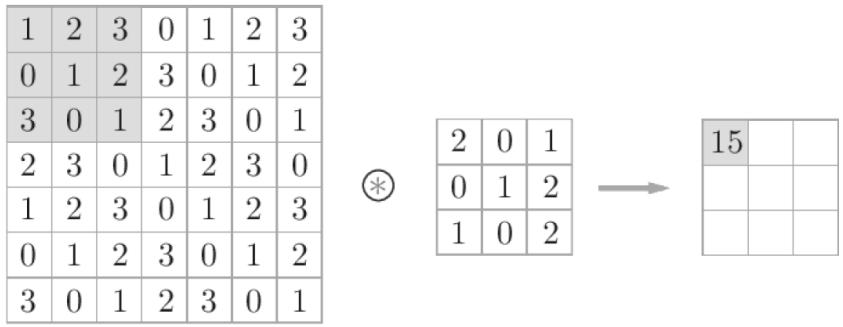


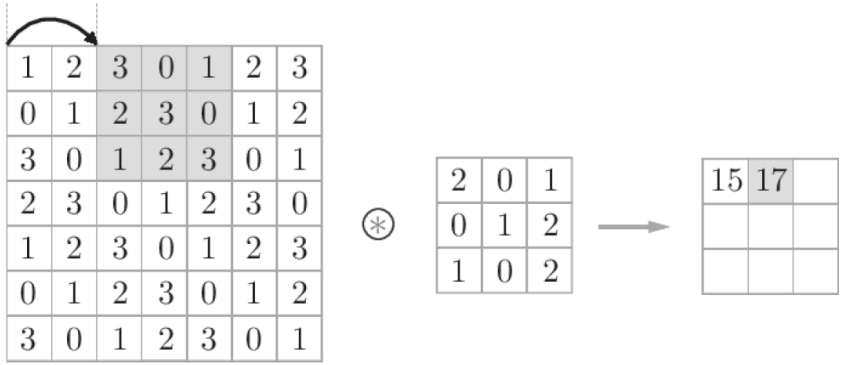
Figure: https://github.com/WegraLee/deep-learning-from-scratch

스트라이드(stride)

필터적용 간격 조정



스트라이드 : 2



- 합성곱 필터의 특징에 따른 효과
  - https://docs.gimp.org/2.10/ko/gimp-filter-convolution-matrix.html

풀링 연산

- 풀링(pooling)은 세로, 가로방향의 공간을 줄이는 연산을 의미함.
- 특징맵을 하향 표본추출(downsampling)한다고 표현
- 평균풀링, 최대풀링 등이 있음.
  - 학습해야할 매개변수가 없음.
  - 차원(예. 채널수) 불편
  - 입력의 변화에 영향을 적게 받음.

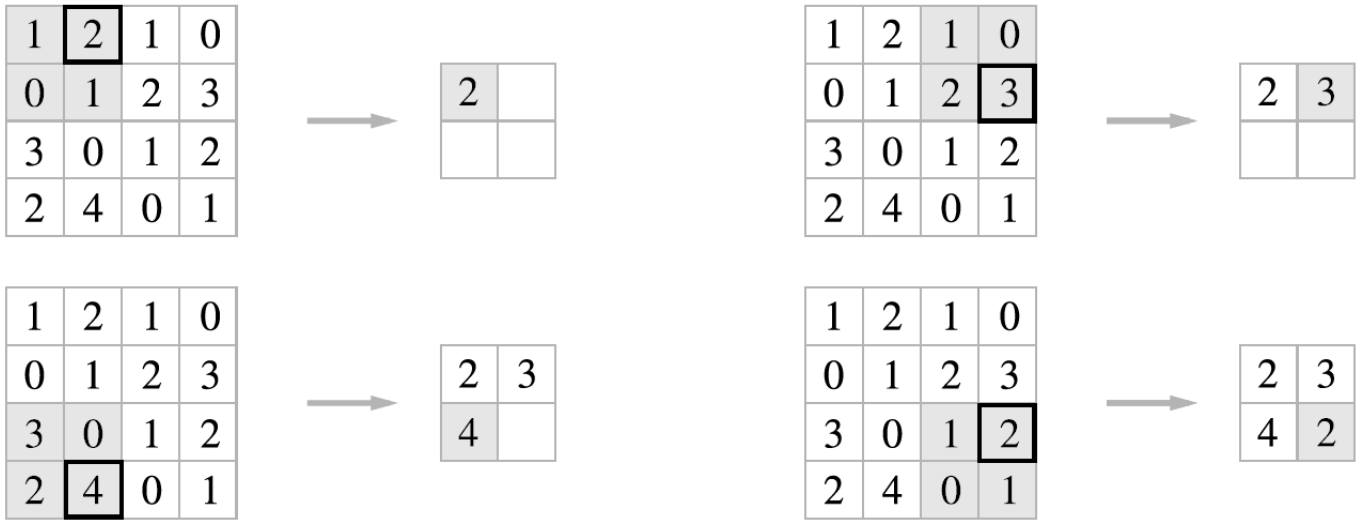


Figure: max pooling, `https://github.com/WegraLee/deep-learning-from-scratch`

실습: 우편번호 다중클래스 분류

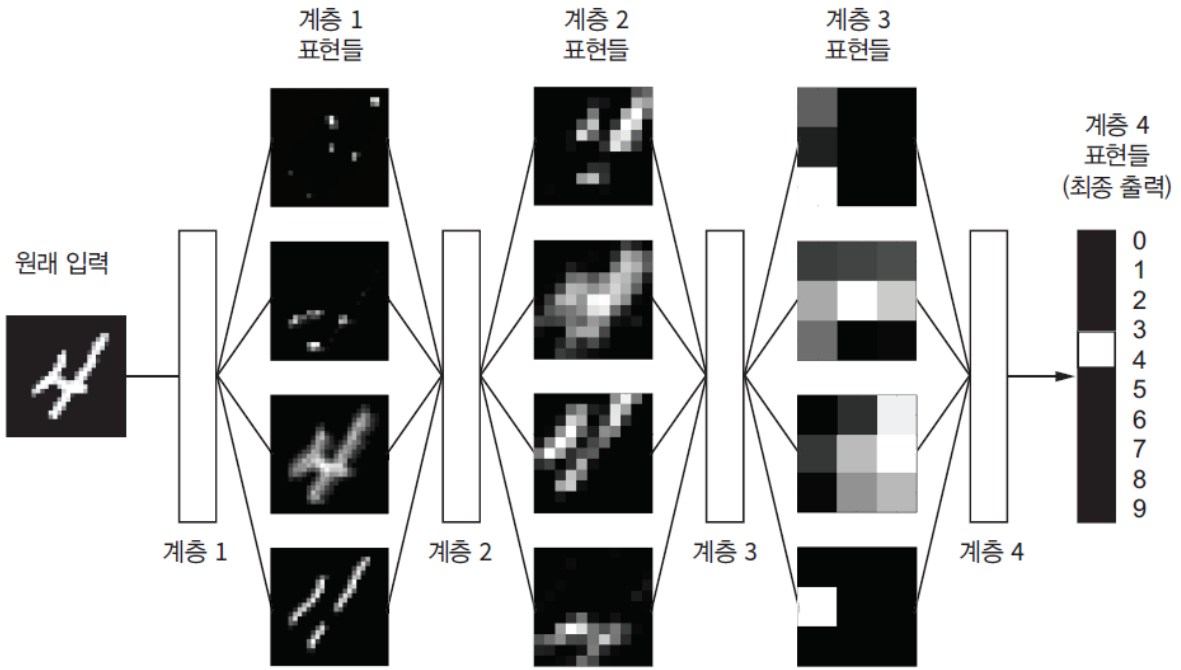


Figure: MNIST, `https://github.com/WegraLee/deep-learning-from-scratch`

- 0-9까지 총 10가지의 수 중 하나를 손으로 직접 쓴 이미지 데이터
- 60000개의 훈련자료와 10000개의 시험자료

자료 준비

```
> mnist <- dataset_mnist()
> c(c(train_images, train_labels), c(test_images, test_labels))
  %<-% mnist

> train_images <- array_reshape(train_images, c(60000, 28, 28, 1))
> train_images <- train_images / 255

> test_images <- array_reshape(test_images, c(10000, 28, 28, 1))
> test_images <- test_images / 255

> train_labels <- to_categorical(train_labels)
> test_labels <- to_categorical(test_labels)
```

- 'layer_conv_2d()' and 'layer_max_pooling_2d()'
- 텐서의 shape 은 (높이, 넓이, 채널)= '(height, width, channels)'인 3D으로 배치의 크기는 포함하지 않음.
- inputs of size '(28, 28, 1)'은 'input_shape = c(28, 28, 1)' 로 처리

망 설정

```
> library(keras)
> model <- keras_model_sequential() %>%
  layer_conv_2d(filters=32, kernel_size=c(3, 3), activation="relu",
                input_shape=c(28, 28, 1)) %>%
  layer_max_pooling_2d(pool_size=c(2, 2)) %>%
  layer_conv_2d(filters=64, kernel_size=c(3, 3),
                activation="relu") %>%
  layer_max_pooling_2d(pool_size=c(2, 2)) %>%
  layer_conv_2d(filters=64, kernel_size=c(3, 3), activation="relu")

> summary(model)
```

- 출력 3D 텐서 '(3, 3, 64)'를 1D 조밀 층에 연결함

```
> model <- model %>%   layer_flatten() %>%
   layer_dense(units = 64, activation = "relu") %>%
   layer_dense(units = 10, activation = "softmax")
```

```
Model: "sequential_15"
_____________________________________________________________________________________________
Layer (type)                                 Output Shape                            Param #
=============================================================================================
conv2d_9 (Conv2D)                            (None, 26, 26, 32)                      320
_____________________________________________________________________________________________
max_pooling2d_6 (MaxPooling2D)               (None, 13, 13, 32)                      0
_____________________________________________________________________________________________
conv2d_10 (Conv2D)                           (None, 11, 11, 64)                      18496
_____________________________________________________________________________________________
max_pooling2d_7 (MaxPooling2D)               (None, 5, 5, 64)                        0
_____________________________________________________________________________________________
conv2d_11 (Conv2D)                           (None, 3, 3, 64)                        36928
_____________________________________________________________________________________________
flatten (Flatten)                            (None, 576)                             0
_____________________________________________________________________________________________
dense_30 (Dense)                             (None, 64)                              36928
_____________________________________________________________________________________________
dense_31 (Dense)                             (None, 10)                              650
=============================================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

참고

- 예를 들어 1층 모수의 수는 kernel의 크기 (3,3)의 9개와 절편 1개로 총 10개이며, units의 수가 32개이므로 총 320개가 됨.
- 입력크기 (h, w), 필터크기 (h_f, w_f), 패딩 p, 스트라이드 s라 할 때,

  합성곱망을 통과한, 출력크기 (h_o, w_o)는

$$h_o = \frac{h + 2p - h_f}{s} + 1, w_o = \frac{w + 2p - h_w}{s} + 1$$

- padding='valid'는 패딩 $p = 0$로 디폴트임, 만약 padding='same'로 설정하면 입력차원과 같은 차원을 가지는 p값로 설정됨
- 예를 들어 1층에서 입력 (28,28), 패딩 0, 스트라이드 1, 필터 (3,3)이므로

$$h_o = \frac{28 + 2 \times 0 - 3}{1} + 1 = 26, w_o = \frac{28 + 2 \times 0 - 3}{1} + 1 = 26$$

망 컴파일

```
> model %>% compile( optimizer = "rmsprop",
  loss = "categorical_crossentropy",  metrics = c("accuracy"))
```

망 컴파일, 적합, 평가

```
# 망 컴파일
> model %>% compile( optimizer = "rmsprop",
   loss = "categorical_crossentropy",   metrics = c("accuracy"))

> history <- model %>% fit(
   train_images, train_labels,
   epochs = 5, batch_size=64, validation_split = 0.1)

# 평가
> results <- model %>% evaluate(test_images, test_labels)
> listviewer::jsonedit(history, model="view")
```

모형 수렴

```
> plot(history$metrics$loss, main="Model Loss",
    xlab = "epoch", ylab="loss", type="l")
> lines(history$metrics$val_loss, col=2, lty=2)
> legend("topright", c("train","test"), col=c(1, 2), lty=c(1,2))
```

모형 정확성

```
> plot(history$metrics$acc, main="Model Accuracy",
    xlab="epoch", ylab="accuracy", type="l", ylim=c(0.8,1))
> lines(history$metrics$val_acc, col=2, lty=2)
> legend("bottomright", c("train","test"), col=c(1,2), lty=c(1,2))
```

평가표본을 통한 평가

```
> pred_mat <- model %>% predict(test_images)
> pred <- apply(pred_mat,1,which.max)

# 오차 행렬(Confusion Matrix)
> table(test_labels_c, pred)
```

Table: Test_labels vs. pred

	1	2	3	4	5	6	7	8	9	10
0	947	0	4	1	3	2	2	2	5	14
1	0	1115	7	3	0	2	1	1	4	2
2	0	0	1027	0	0	0	0	3	2	0
3	0	0	3	998	0	7	0	0	2	0
4	0	0	3	0	966	0	0	0	2	11
5	0	0	0	3	0	888	1	0	0	0
6	1	2	1	1	2	6	941	0	4	0
7	0	0	4	2	0	0	0	999	1	22
8	0	0	4	0	0	1	0	0	967	2
9	0	0	0	0	2	3	0	0	1	1003

참고

- 이진분류 평가 측도: recall, precision, f1-score, ROC, AUC etc.

정리

- CNN은 지금까지의 완전연결 계층 네트워크에 합성곱 계층과 풀링 계층을 새로 추가함
  - 계층이 깊어질수록 고급정보가 추출되는 것으로 알려짐
  - 다단식 정보 증류 연산(information-distillation operation)으로 이해
  - 정보가 연속적인 증류기를 거치면서 점차 순수해짐
- 대표적 CNN: LeNet, AlexNet, ResNet(150 layers)
  - 그 외 VGG, GoogLeNet 등