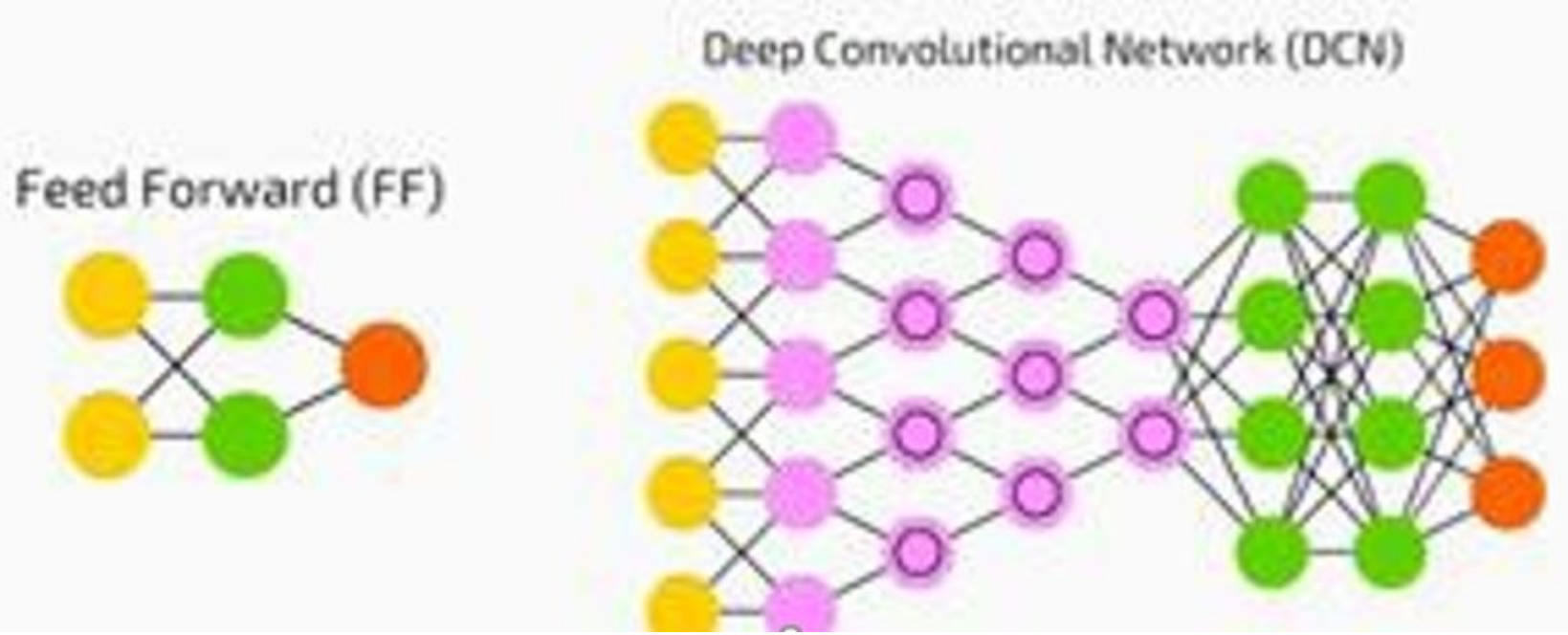
Day 4

CNN

CNN (Convolutional Neural Network)
합성곱 신경망

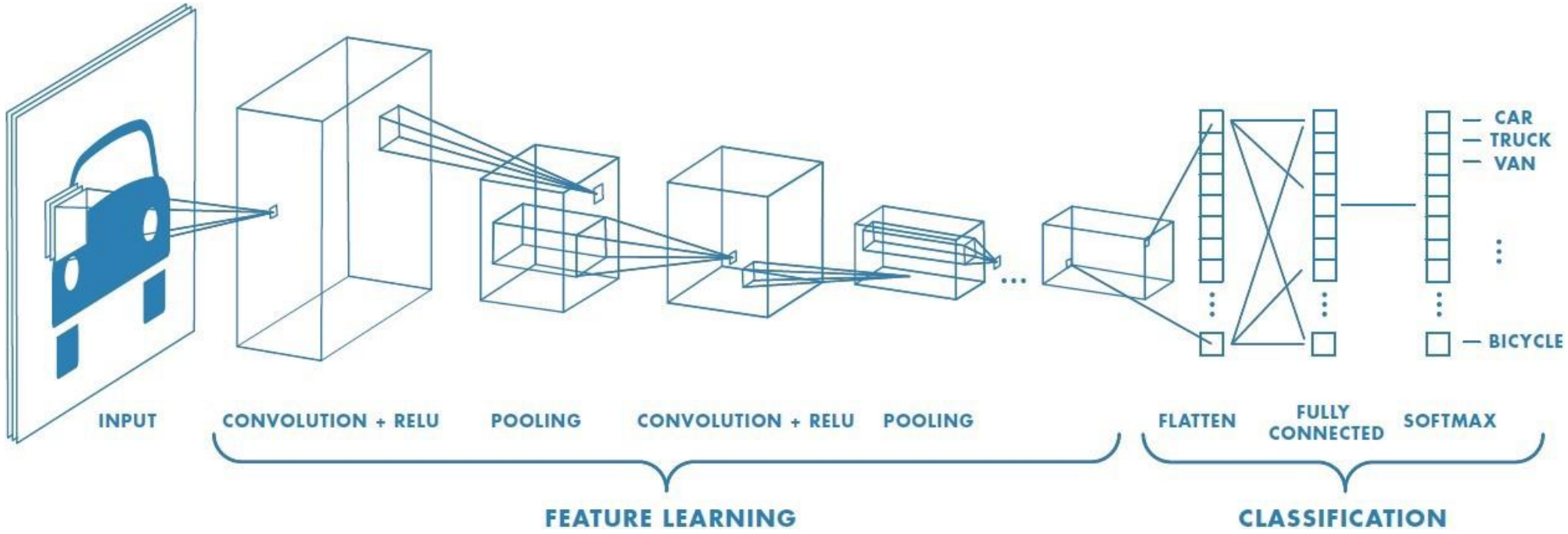
- A neural network with some convolutional layers (and some other layers), e.g., Image processing
  시각적 이미지를 분석하는 데 사용되는 깊고 피드-포워드적인 인공신경망



Convolutional Layers

Responsible for the convolutional operation in which feature maps identifies features in the images

이미지안에 있는 특성을 찾아내기 위해 컨볼루션 계산을 하는 층



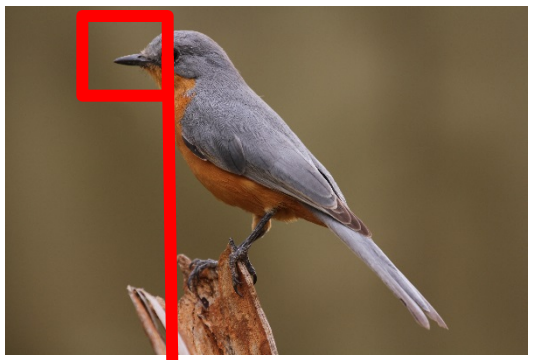
CNN Applications

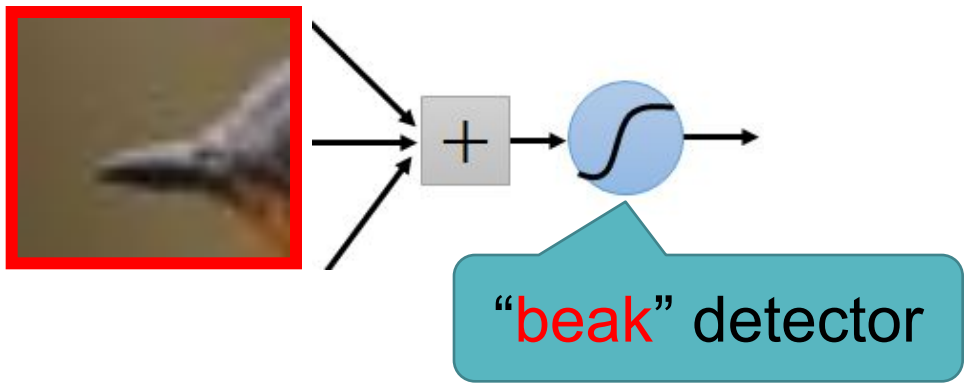
- Detection – tv or monitor1, 2, 3, person, remote control1,2,3
- Scene parsing – sky, building, road, sidewalk, etc.
- Indoor semantic labeling – books, chair, furniture, sofa, object, TV, etc.
- Action detection
- Image processing

Consider learning an image:

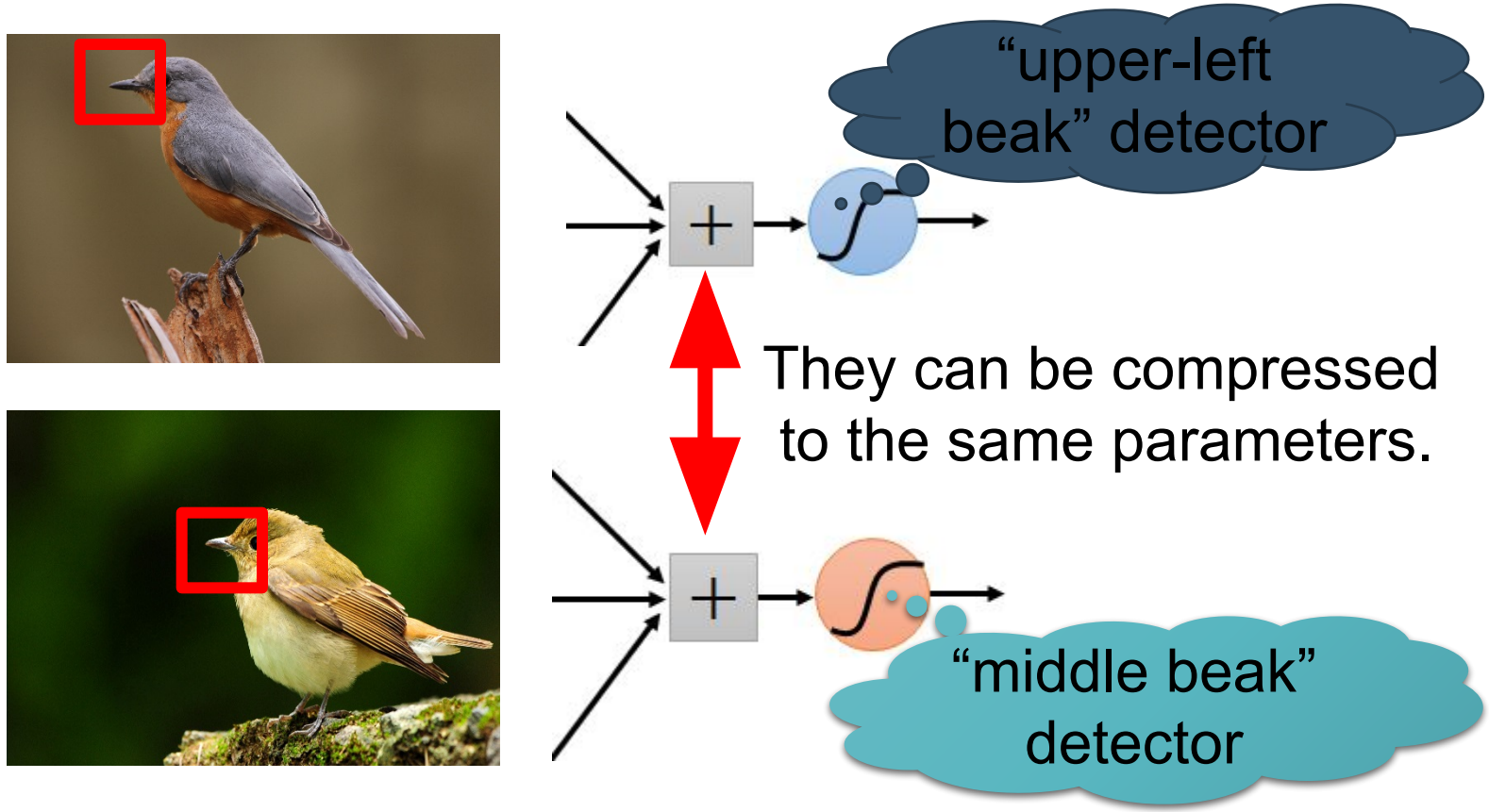
- Some patterns are much smaller than the whole image 전체이미지보다 작은 패턴

Can represent a small region with fewer parameters



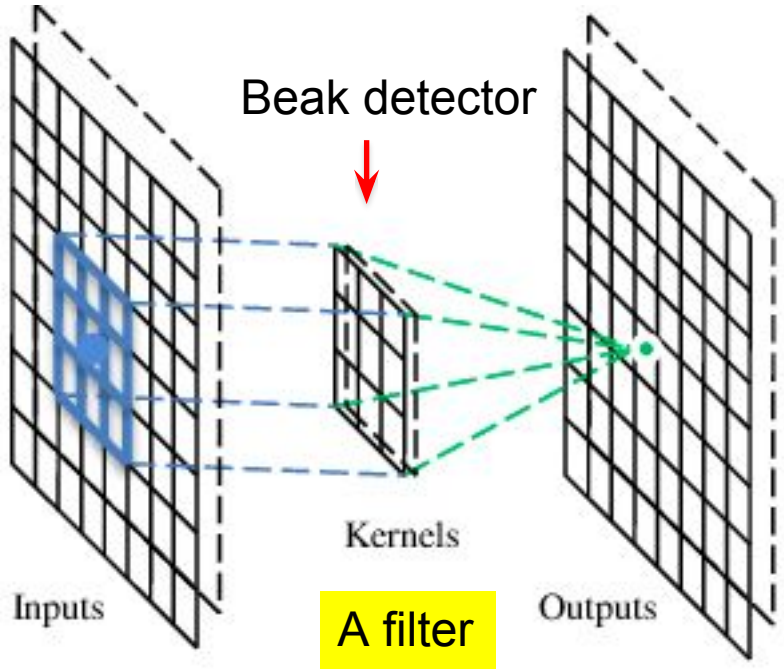


Same pattern appears in different places:
They can be compressed! 같은 패턴이 다른 위치에 나타남
What about training a lot of such "small" detectors
and each detector must "move around". 여러개의 디텍터들을 만든 다음 움직이면서 검사



Convolutional Layer

- A convolutional layer has a number of filters that does convolutional operation. **합성곱연산을 하는 여러개의 필터들**



Convolution

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

These are the network parameters to be learned.

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Dot product → 3 -1

6 x 6 image

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

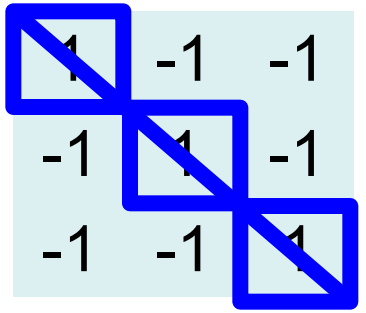
If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

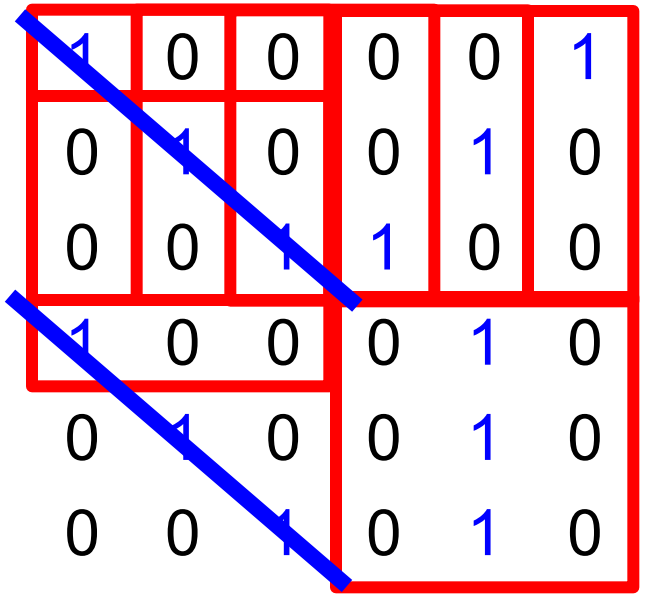
3 -3

Convolution

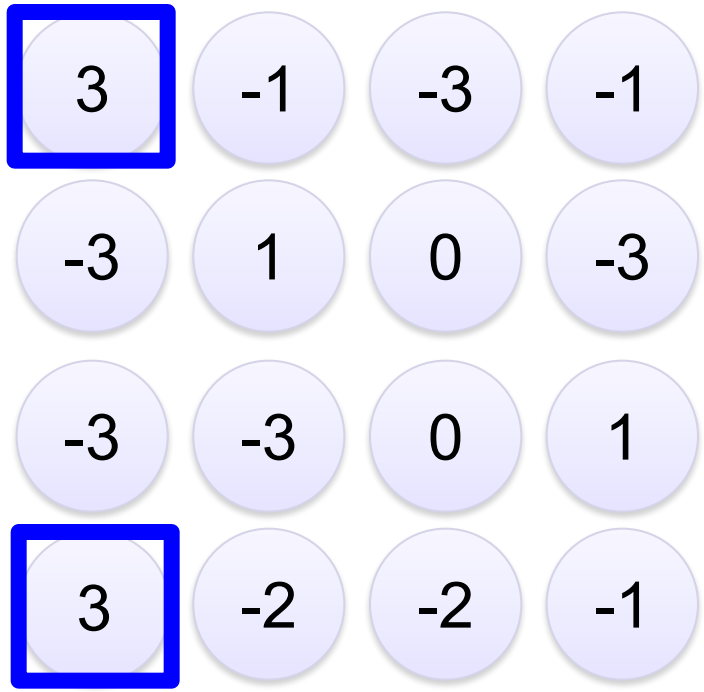


Filter 1

stride=1



6 x 6 image



Convolution

-1	1	-1
-1	1	-1
-1	1	-1

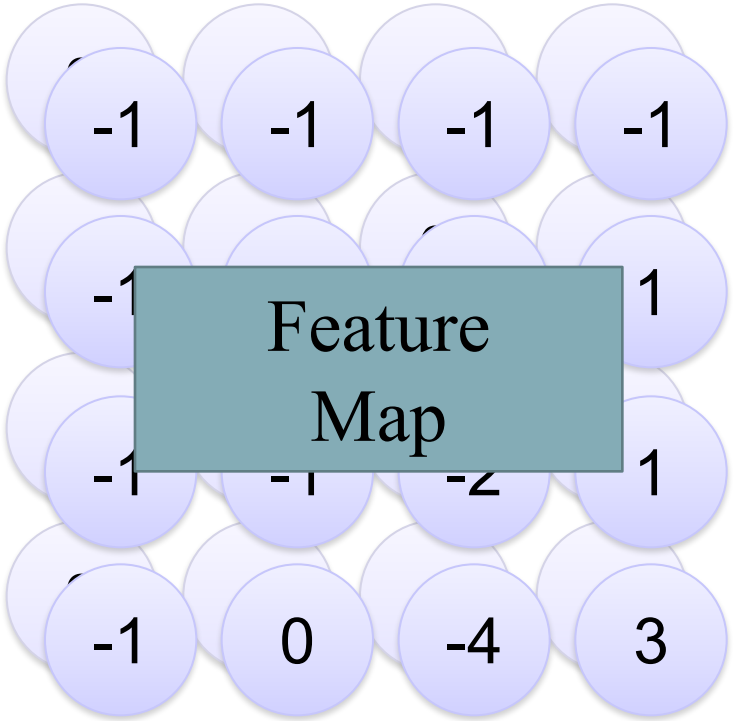
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Repeat this for each filter



Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB 3 channels

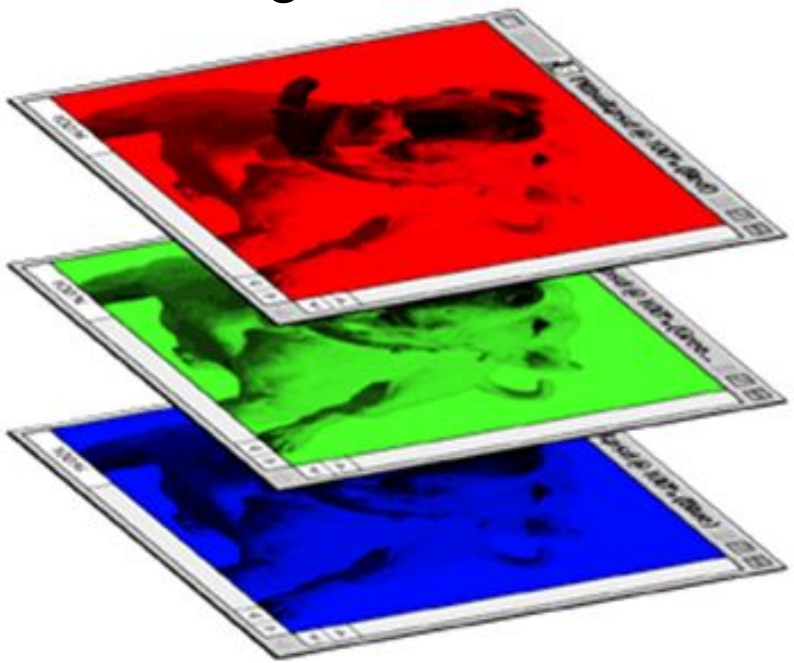
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

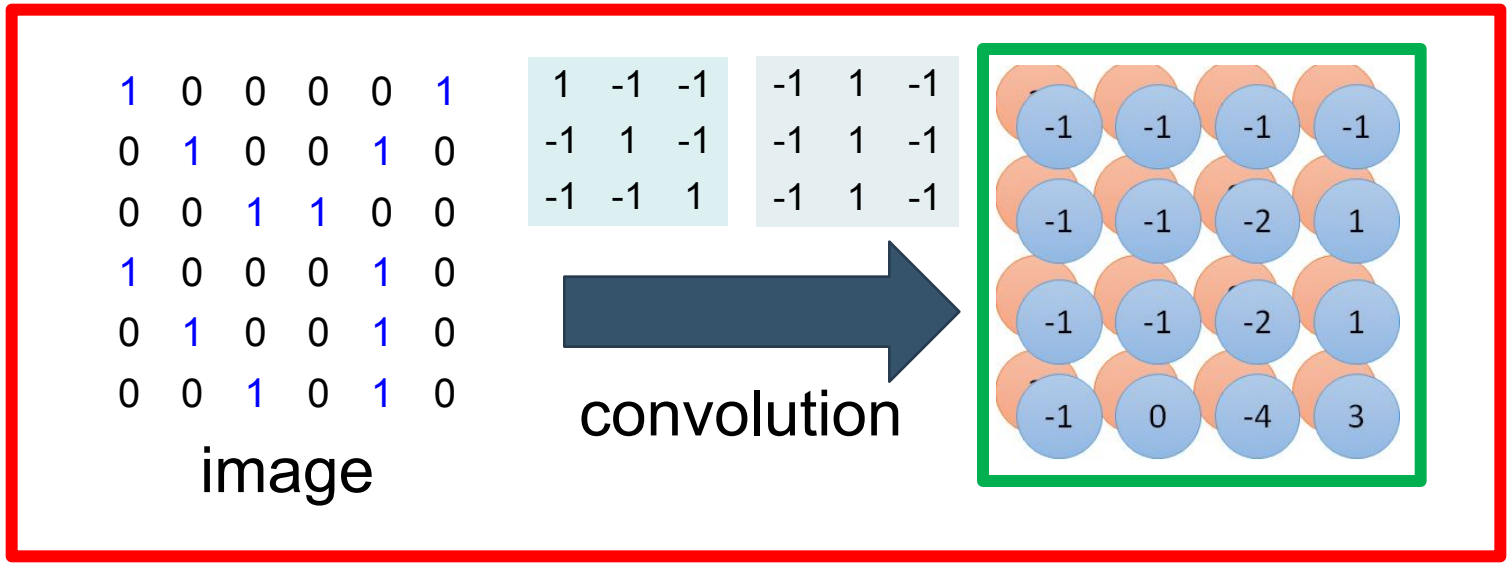
Filter 2

Color image

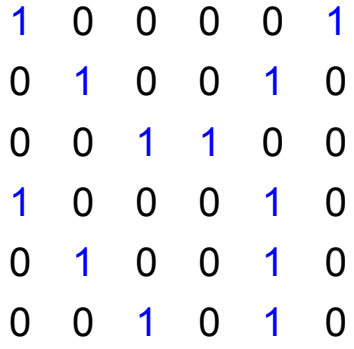


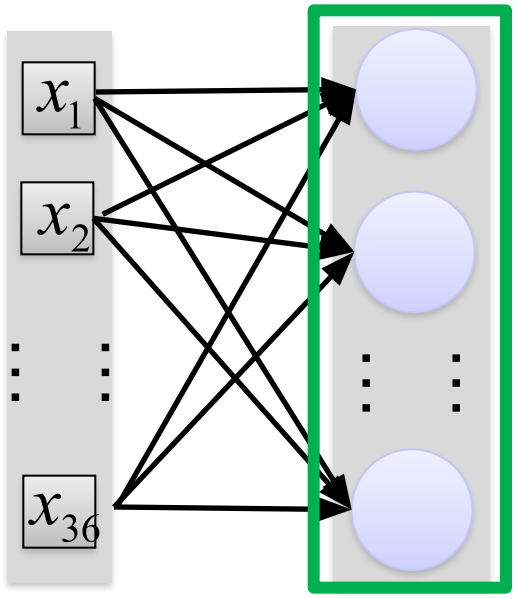
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

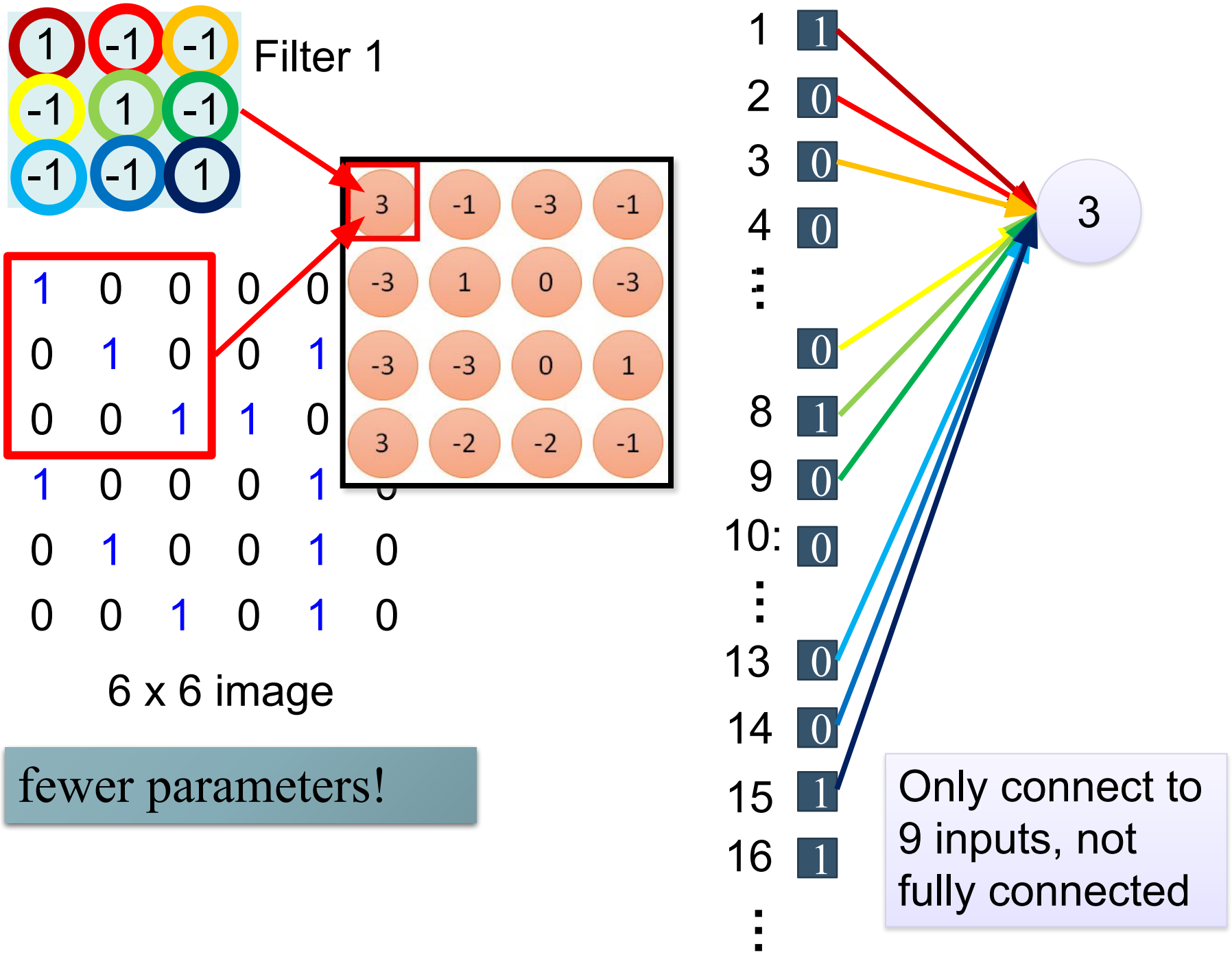
Convolution vs Fully Connected

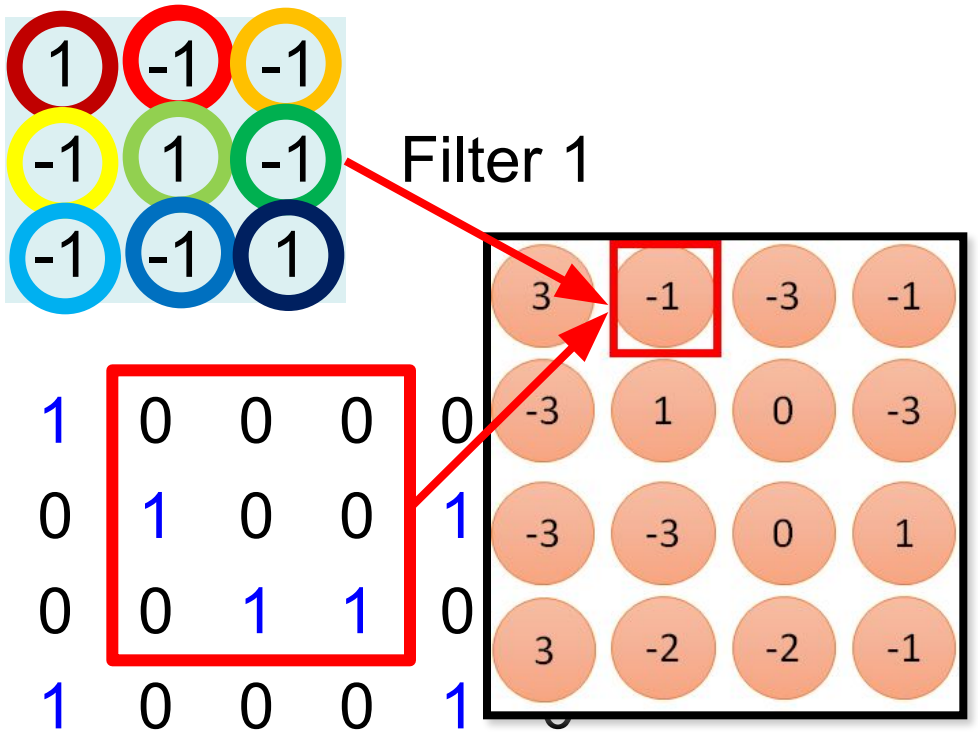


Fully-connected



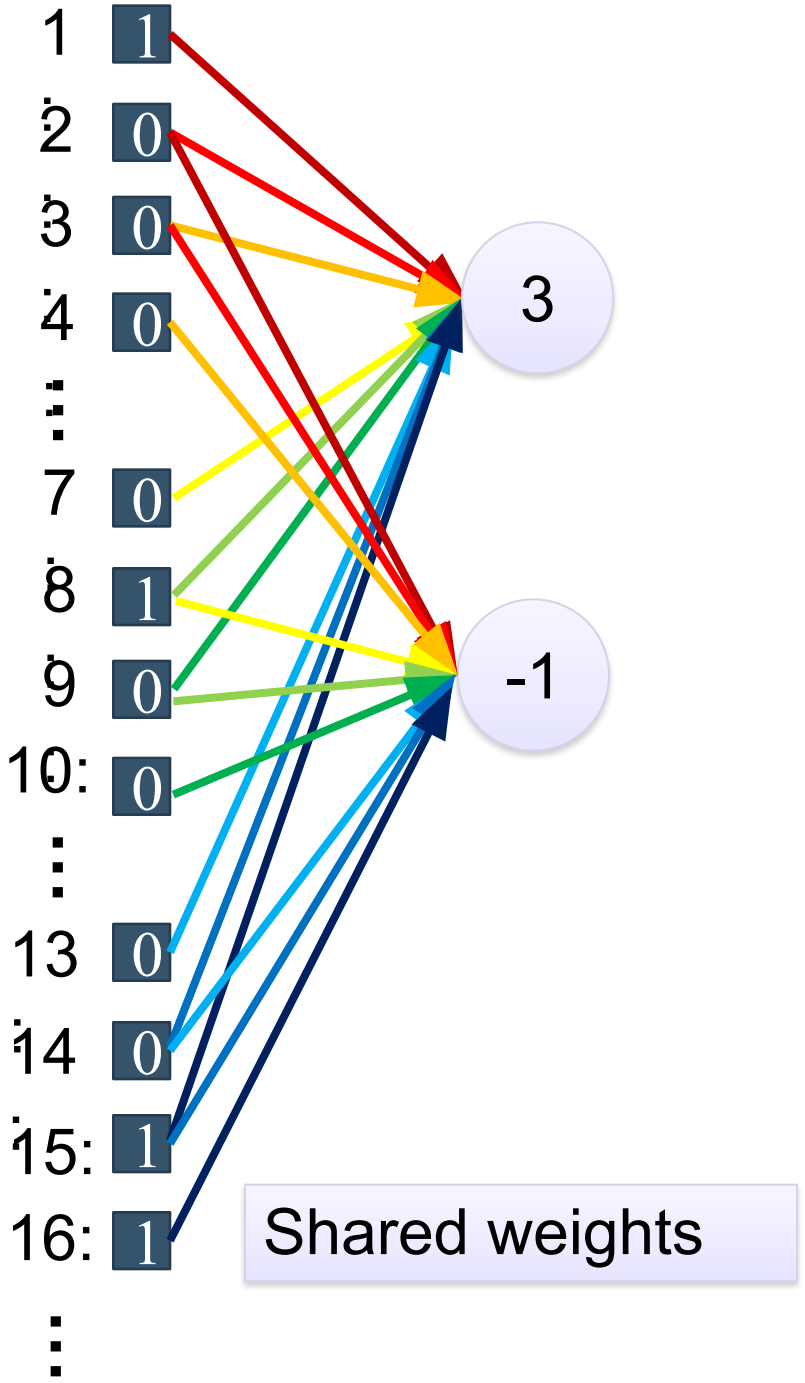



1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
fewer parameters!
1 1
2 0
3 0
4 0
0
8 1
9 0
10: 0
13 0
14 0
15 1
16 1
3
Only connect to 9 inputs, not fully connected

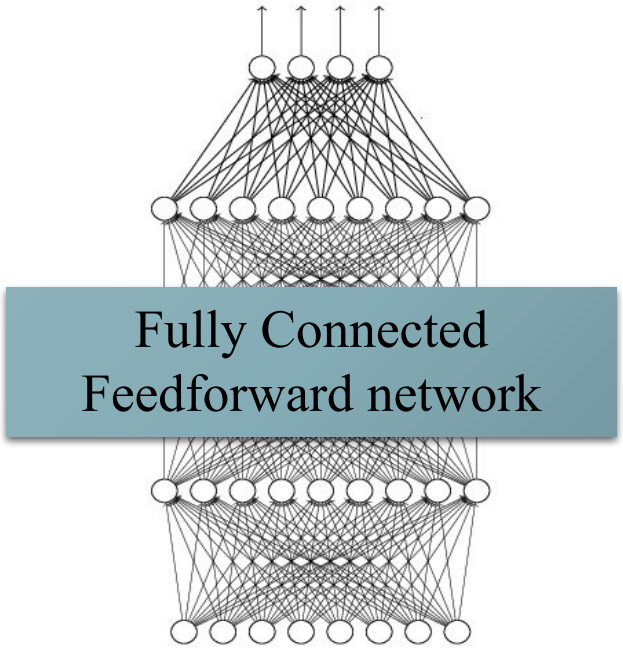


Fewer parameters

Even fewer parameters



The whole CNN



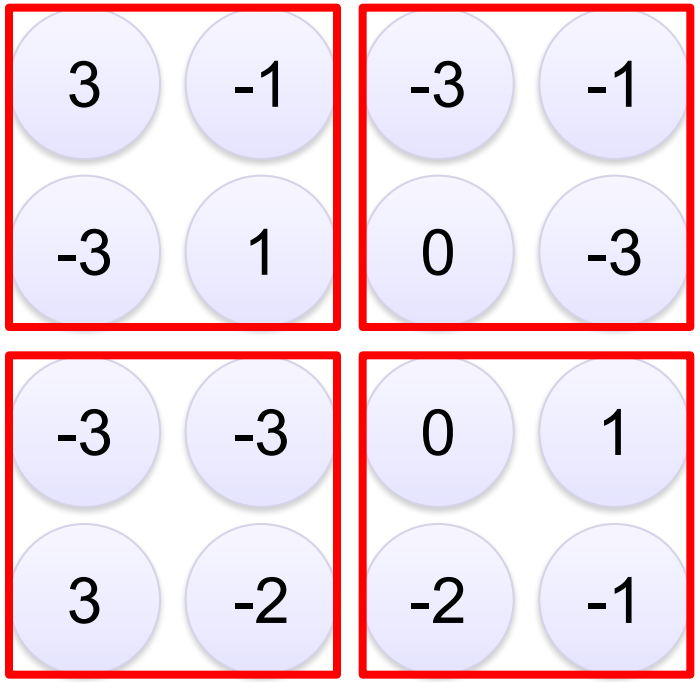
Max Pooling

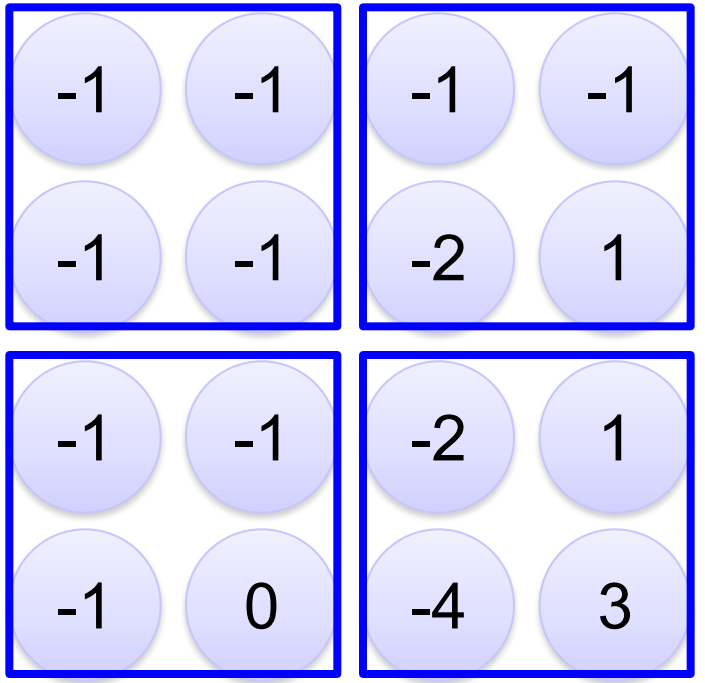
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2





Why Pooling

- Subsampling pixels will not change the object 서브샘플링으로 이미지를 바꿈

bird

Subsampling

bird

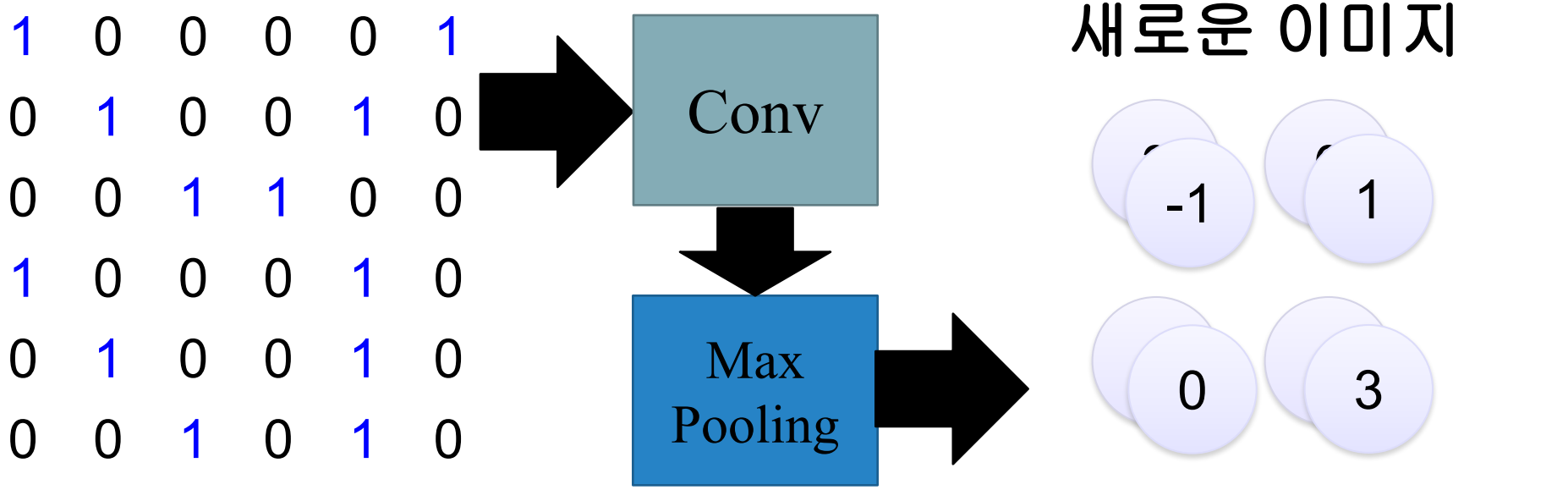
We can subsample the pixels to make image smaller 이미지를 작게

→ fewer parameters to characterize the image 계수를 더 작게 만듬

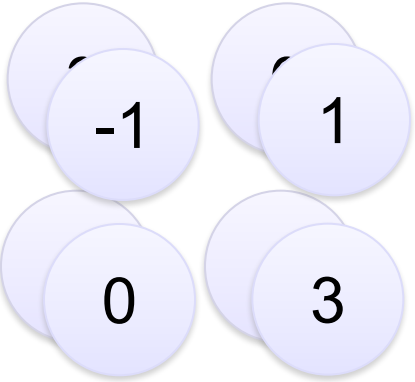
A CNN compresses a fully connected network in two ways: CNN에서 풀리컨넥티드된 네트워크를 압축하는 방법

- Reducing number of connections 컨넥션의 수를 줄임
- Shared weights on the edges 가중치를 공유
- Max pooling further reduces the complexity 맥스풀링으로 복잡도를 줄임

Max Pooling


1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
Conv
Max Pooling
New image
but smaller 훨씬 작은
새로운 이미지
-1
1
0
3
2 x 2 image
Each filter
is a channel 필터가
하나의 채널임

The whole CNN



A new image

Convolution

Max Pooling

Convolution

Max Pooling

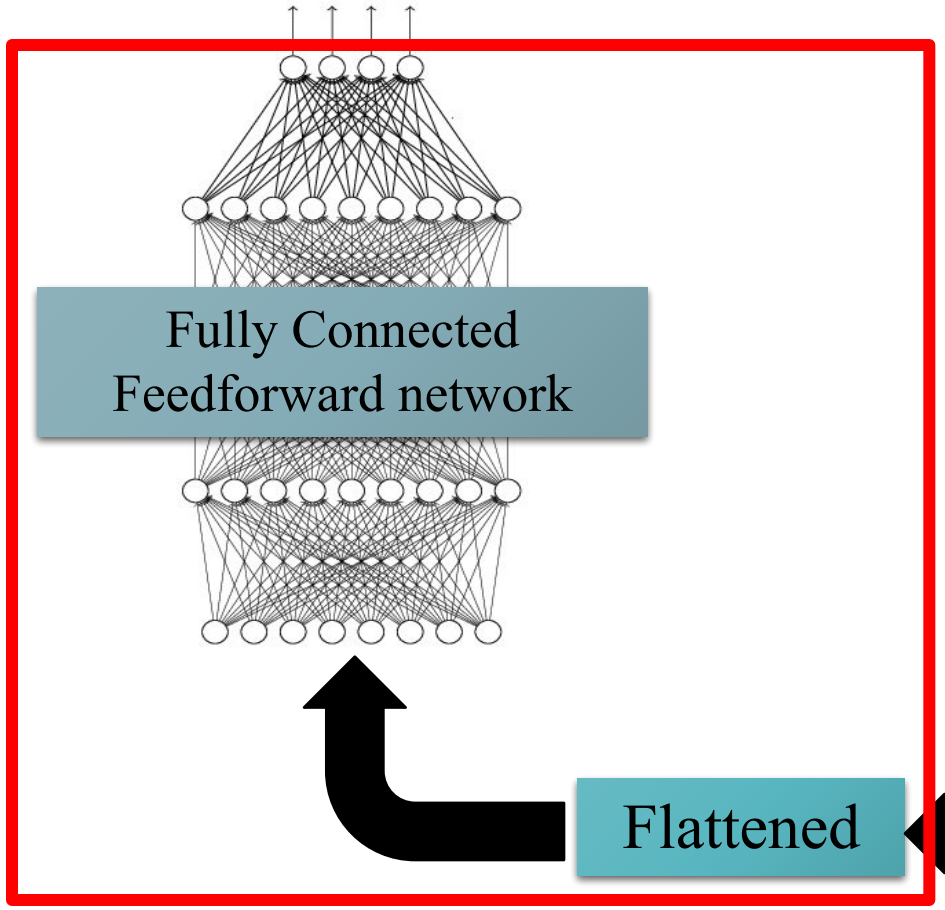
Can repeat many times

Smaller than the original image 원래 이미지보다 작음

The number of channels is the number of filters 필터의 수가 채널의 숫자

The whole CNN

cat dog ……



Convolution

Max Pooling

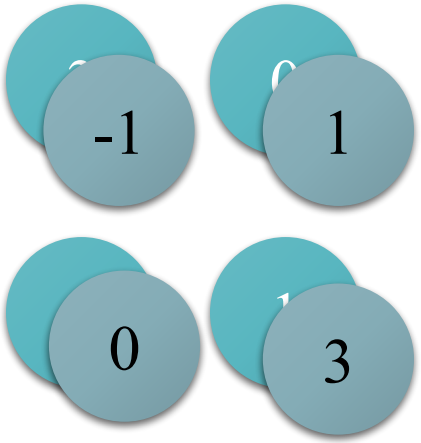
A new image

Convolution

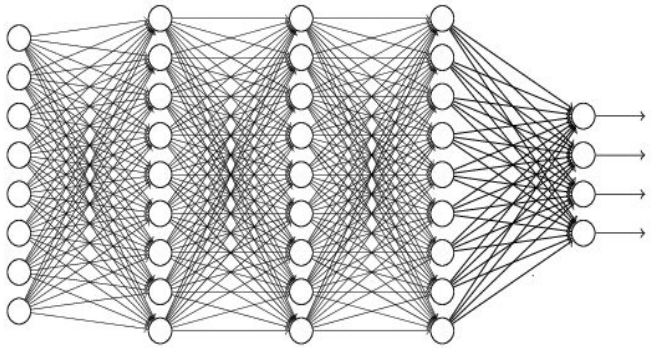
Max Pooling

A new image

Flattening



Flattened

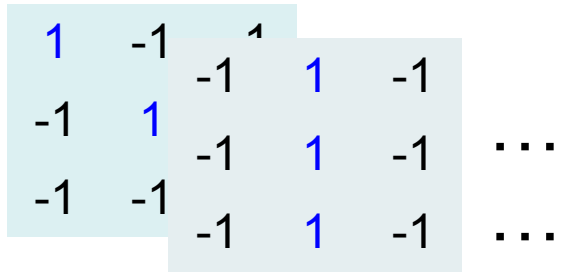


Fully Connected
Feedforward network

CNN in Keras

Only modified the *network structure* and *input format (vector -> 3-D tensor)* ***입력은벡터로, shape은 3D로***

```
model2.add( Convolution2D( 25,3,3,
                input_shape=(28,28,1)) )
```



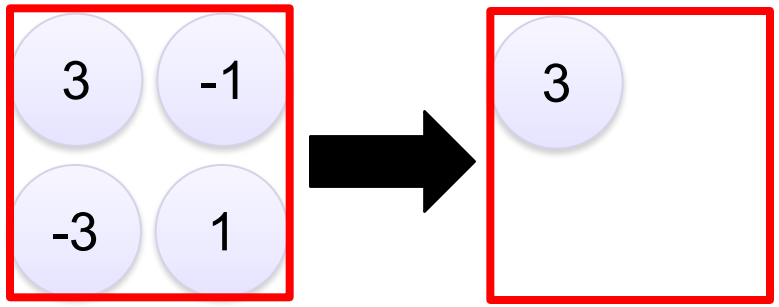
There are 25 3x3 filters.

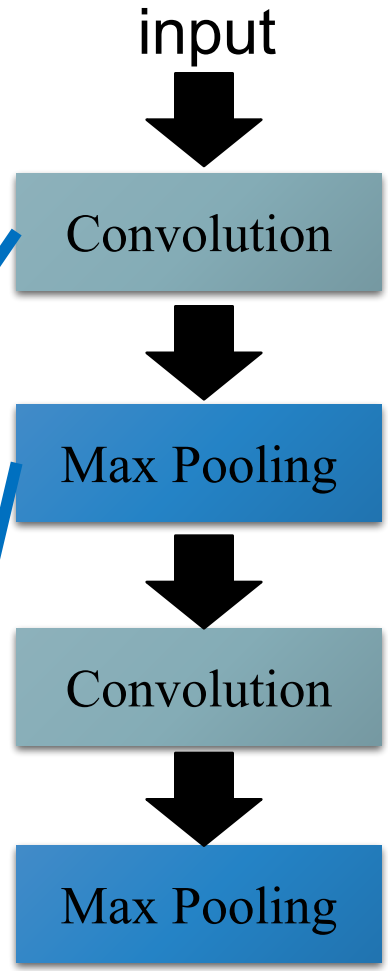
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

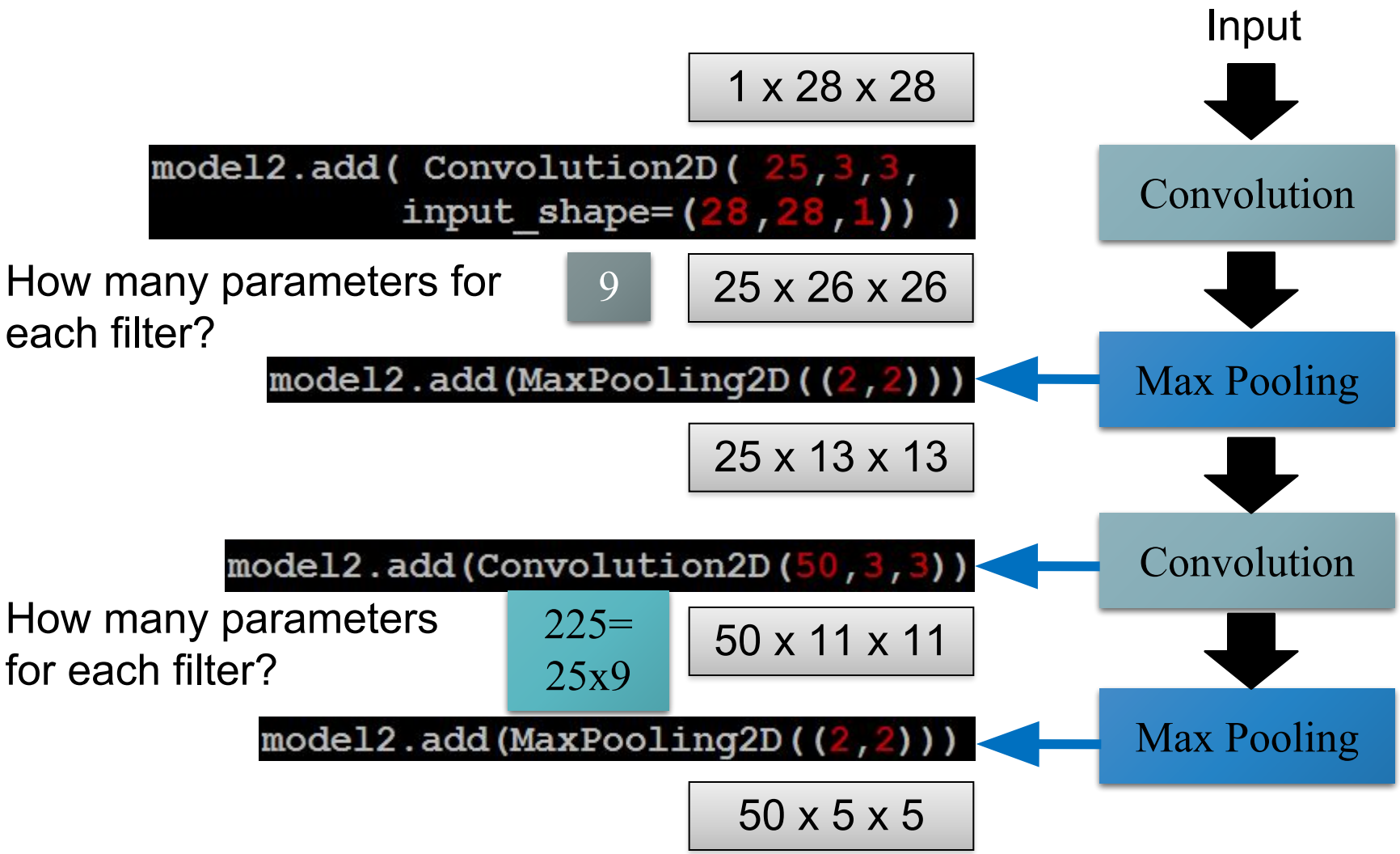
```
model2.add(MaxPooling2D((2,2)))
```





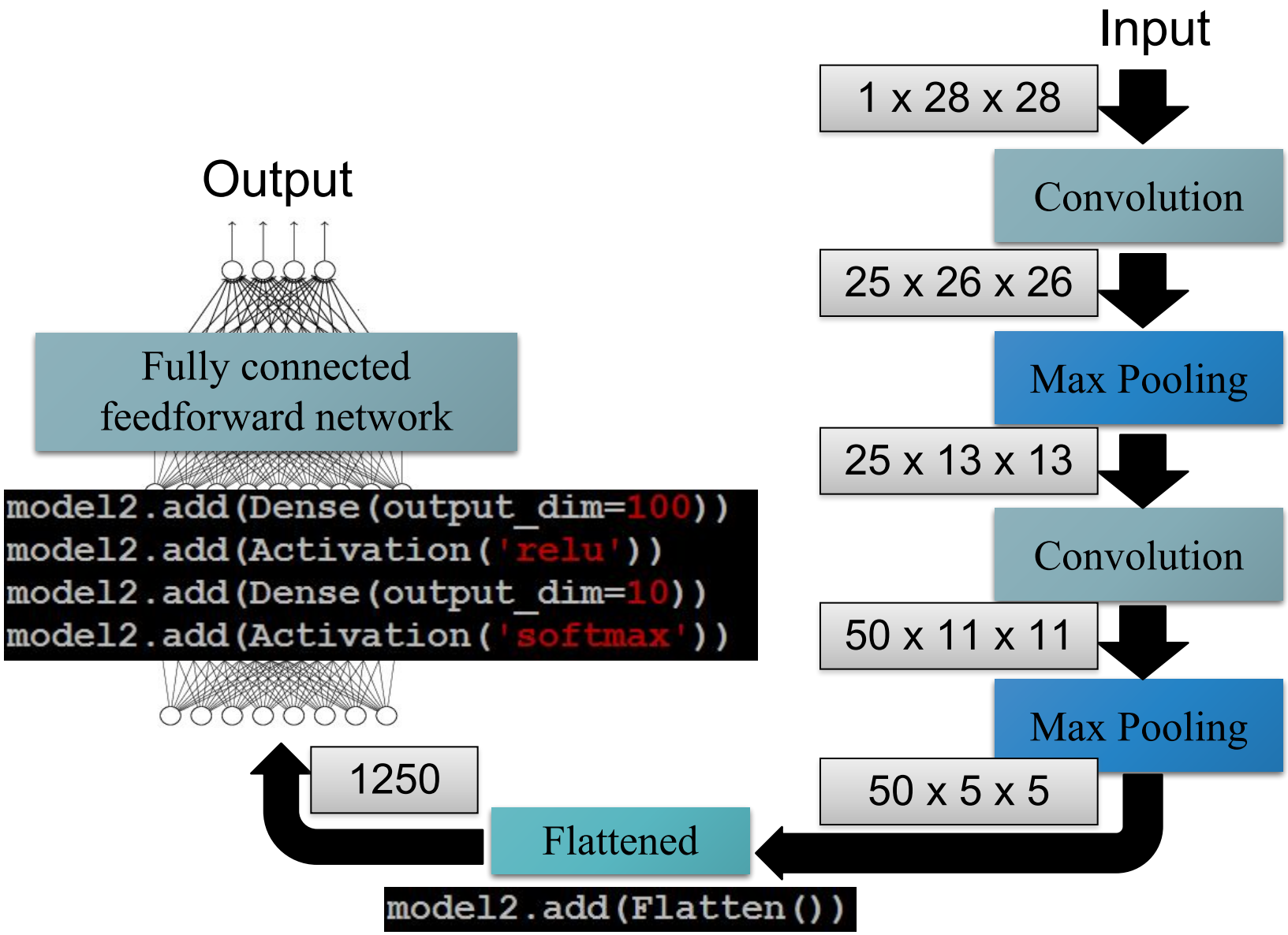
CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



Exercise #4

- Complete the Keras tutorial on https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5
  - Evaluate the model (model.evaluate(x_train, y_train))
  - Save the model and load again to predict the second test data (cnn_model.model)
  - Tensorboard

Loading Dataset

- Download mnist data and split into train and test sets

```
from keras.datasets import mnist
(X_train, y_train), (X_test, y_test)
= mnist.load_data()
```

Exploratory Data Analysis

```
import matplotlib.pyplot as plt
#plot the first image in the dataset
plt.imshow(X_train[0])
#check image shape
X_train[0].shape
```

Data Pre-Processing

- `#reshape data to fit model`
- `X_train = X_train.reshape(60000,28,28,1)`
- `X_test = X_test.reshape(10000,28,28,1)`

Data Pre-Processing

```
from keras.utils import to_categorical
#one-hot encode target column
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
y_train[0]
```

Libraries for CNN

```
from keras.models import
Sequential
from keras.layers import Dense,
Conv2D, Flatten
```

Sequential Model

- Sequential is the easiest way to build a model in Keras.
  캐라스로 가장 쉽게 만들수 있는 모델
- It allows you to build a model layer by layer.
  층층으로 모델을 만들수 있게 함

```
model = Sequential()
```

Convolutional Layer

- Contains a set of filters that deal with input images, 2-dimensional matrices. **2차 메트릭스의 이미지를 처리하는 필터들을 포함**

```
model.add(Conv2D(64, kernel_size=3,
activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(32, kernel_size=3,
activation='relu'))
```

Convolutional Layer

- 64 and 32 - the number of nodes in each layer, which can be adjusted depending on the size of the dataset 데이터셋의 사이즈에 따라 각층에 있는 노드의 갯수를 조정
- Kernel size - the size of the filter matrix for the convolution (i.e., 3 is 3x3 filter matrix) 컨볼루션을 위한 필더행렬의 크기
- Activation function – ReLU (Rectified Linear Activation)
- Input shape – only for the first layer. 1 indicates that the images are greyscale. 첫번째 레이어에서만 지정. 1은 이미지가 흑백이라는 의미

Flatten Layer

- Serves as a connection between the convolution and dense layers. **컨볼루션과 덴스레이어 사이를 연결**

```
model.add(Flatten())
```

Dense Layer

- Layer type used in output layer **출력레이어에 사용**
- 10 nodes in output layer (possible outcome between 0 and 9) **10개의 노드는 0에서 9까지를 출력**
- Softmax - output sum up to 1 as probabilities. Can make prediction based on the probability **출력의 확률의 합은 1. 확률을 기초로 예측**

```
model.add(Dense(10, activation='softmax'))
```

Compile

- Optimizer - controls the learning rate. The adam optimizer adjusts the learning rate throughout training. A smaller learning rate may lead to more accurate weights, but it takes longer time **옵티마이저가 학습률을 조절. 적은 학습률일수록 더 정확한 가중치를 찾아내지만, 오래 걸림. 아담옵티마이저는 훈련하는 동안 학습률을 조절함**
- Loss function - a lower score indicates that the model is performing better. Categorical_crossentropy is the most common choice for classification. **점수가 낮을수록 모델의 성과가 좋음. 분류문제에 가장 좋은 categorical_crossentropy**

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

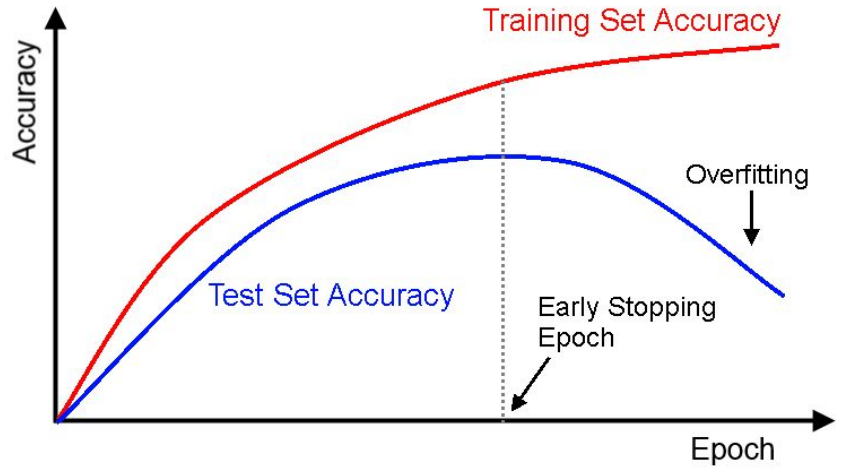
Training Model

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=3)

model.fit(x, y, batch_size=32, validation_split=.1)

model.fit(x_train, y_train, batch_size=128, epochs=12,verbose=1,validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/3
60000/60000 [==============================] - 22s 363us/step - loss: 1.3991 - acc: 0.8830 - val_loss: 0.0882 - val_acc: 0.9738
Epoch 2/3
60000/60000 [==============================] - 20s 334us/step - loss: 0.0712 - acc: 0.9790 - val_loss: 0.0874 - val_acc: 0.9729
Epoch 3/3
60000/60000 [==============================] - 20s 334us/step - loss: 0.0484 - acc: 0.9854 - val_loss: 0.0898 - val_acc: 0.9757
<keras.callbacks.History at 0x7fc38442e240>
```

epochs

The number of times the model will cycle through the data. The more epochs we run, the more the model will improve, up to a certain point. After that point, the model will stop improving during each epoch. **데이터를 가지고 한번 포워드, 백워드 계산을 하는 수. 이팍횟수가 많아질수록 모델이 향상됨. 어떤 지점에 이르면 값이 수렴되며 더이상 향상되지 않음**

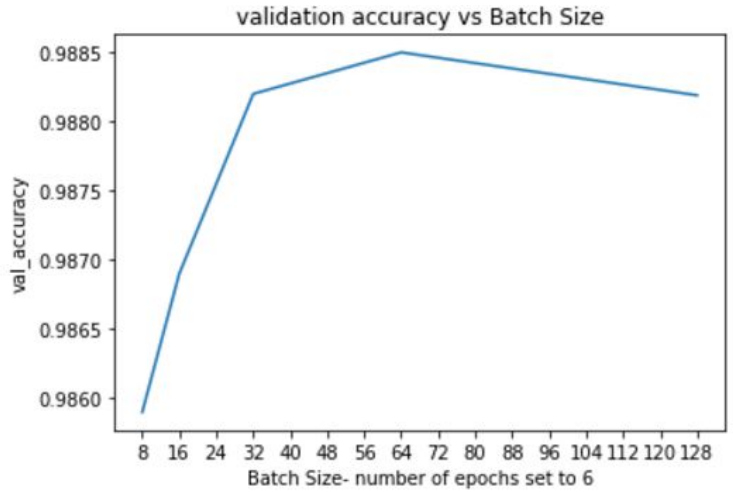


No right numbers of epochs
It is related to how diverse your data is
정답이 없음. 데이터의 다양성에 따라 효과적인 이팍의 횟수가 달라짐

batch size

- The number of training examples utilized in one iteration
  한번 훈련할때 사용하는 데이터의 갯수
- For example, you can divide the dataset of 2000 examples into batches of 500 then it will take 4 iterations to complete 1 epoch.
  2000개의 데이터를 가지고 500개의 배치로 나눠서 모델을 만드는 경우 한번의 이팍에 4번을 계산을 수행하게 됨

In general, batch size of 32 is a good starting point, and then try 64, 128, and 256. **32로 시작해서 64, 128, 256로 증가시킴**



verbose

- How you want to see the training progress for each epoch.
  훈련과정을 어떻게 보고 싶은지 결정
  - verbose=0 will show nothing 아무것도 안보임
  - verbose=1 will show an animated progress bar 진행바를 보여줌

```
[==============================]
```

  - verbose=2 will show the number of epoch 진행수를 보여줌

```
Epoch 1/10
```

Make Predictions

```
model.predict(X_test[:4])
model.predict(X_test)
```

```
array([[1.6117248e-09, 8.6684462e-16, 6.8095707e-10, 1.5486043e-08,
        6.2878847e-14, 1.2934288e-15, 1.1453808e-16, 9.9999928e-01,
        1.0626109e-08, 6.9729606e-07],
       [1.3555871e-07, 2.6465393e-06, 9.9999511e-01, 2.0351818e-08,
        1.9796262e-11, 1.6996018e-12, 2.1163373e-06, 1.2008194e-17,
        4.8792381e-10, 2.6086671e-13],
       [6.7238901e-08, 9.9785548e-01, 1.9031411e-04, 3.9194603e-08,
        1.2894072e-04, 1.5791730e-06, 1.2754040e-06, 4.1349044e-09,
        1.8221687e-03, 5.5910935e-08],
       [9.9999356e-01, 1.6909821e-12, 8.2496926e-10, 1.7359107e-11,
        1.7359230e-12, 1.8865266e-13, 6.4659162e-06, 2.3738855e-11,
        1.1319052e-08, 2.6948474e-08]], dtype=float32)
```

Predicted 7, 2, 1 and 0 for the first four images. **처음 네개의 이미지를 7,2,1,0으로 예측**

- Actual results for first 4 images in test set
  테스트셋에서 처음 4개 이미지에 대한 실제결과

```
y_test[:4]
```

```
array([[0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

Evaluate Model

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Save and Load Model

```
model.save('cnn_model.model')
```

#나중에 사용할때

```
from keras.models import load_model
new_model = load_model('cnn_model.model')
pred = new_model.predict(X_test)
pred
pred[0]
```

Tensorboard

```
from tensorflow.python.keras.callbacks import TensorBoard
from time import time
tensorboard =
TensorBoard(log_dir="logs/{}".format(time()))
model.fit(x_train, y_train, validation_data=(x_test,
y_test), epochs=3, callbacks=[tensorboard])
```

- Command line

```
tensorboard --logdir=logs/
```

Keras Testing

https://www.tensorflow.org/tutorials/quickstart/beginner

```
import tensorflow as tf

mnist =
tf.keras.datasets.mnist

(x_train, y_train),
(x_test, y_test) =
mnist.load_data()

x_train, x_test = x_train
/ 255.0, x_test / 255.0
```

```
model =
tf.keras.models.Sequential([

tf.keras.layers.Flatten(inpu
t_shape=(28, 28)),

  tf.keras.layers.Dense(128,
activation='relu'),

tf.keras.layers.Dropout(0.2)
,

  tf.keras.layers.Dense(10,
activation='softmax')

])
```

Keras Testing

https://www.tensorflow.org/tutorials/quickstart/beginner

```
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy'
,              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test,  y_test,
verbose=2)
```

Pre-Processing

- Normalize the data values to the range [0, 1]. **0과 1사이의 값으로 정규화시킴**

```
x_train = x_train/255
x_test = x_test/255
```

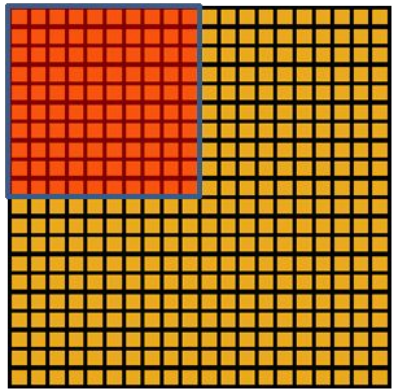
Dropout Layer

- Randomly switches off some neurons in the network which forces the data to find new paths. Therefore, this reduces overfitting
  무작위로 뉴론을 꺼버림으로써 데이터가 새로운 경로를 찾게 하여 오버피팅을 방지

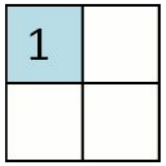
```
from keras.layers import Dropout
model.add(Dropout(0.25))
```

Pooling Layer

- Used to downsample the image as reducing the number of parameters
  계수의 수를 줄여서 이미지를 줄여나감
- Reduces the computation and avoids overfitting. **계산도 줄고 오버피팅도 방지**
  - Max Pooling—Selecting the maximum value
    사이즈를 정해서 그안에서 가장 큰값을 가지고 옴
  - Average Pooling—Sum all of the values and dividing it by the total number of values **전체를 다 더해서 갯수를 나눔**



Convolved feature



Pooled feature

```
from keras.layers import MaxPooling2D
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Loss

```
loss='categorical_crossentropy'
loss='sparse_categorical_crossentropy'
loss='binary_crossentropy'
loss='squared_hinge'
```

- Regression Loss Functions
  - Mean Squared Error Loss
  - Mean Squared Logarithmic Error Loss
  - Mean Absolute Error Loss
- Binary Classification Loss Functions
  - Binary Cross-Entropy
  - Hinge Loss
  - Squared Hinge Loss
- Multi-Class Classification Loss Functions
  - Multi-Class Cross-Entropy Loss
  - Sparse Multiclass Cross-Entropy Loss
  - Kullback Leibler Divergence Loss

categorical_crossentropy vs sparse_categorical_crossentropy

- For one-hot encoded y, use categorical_crossentropy

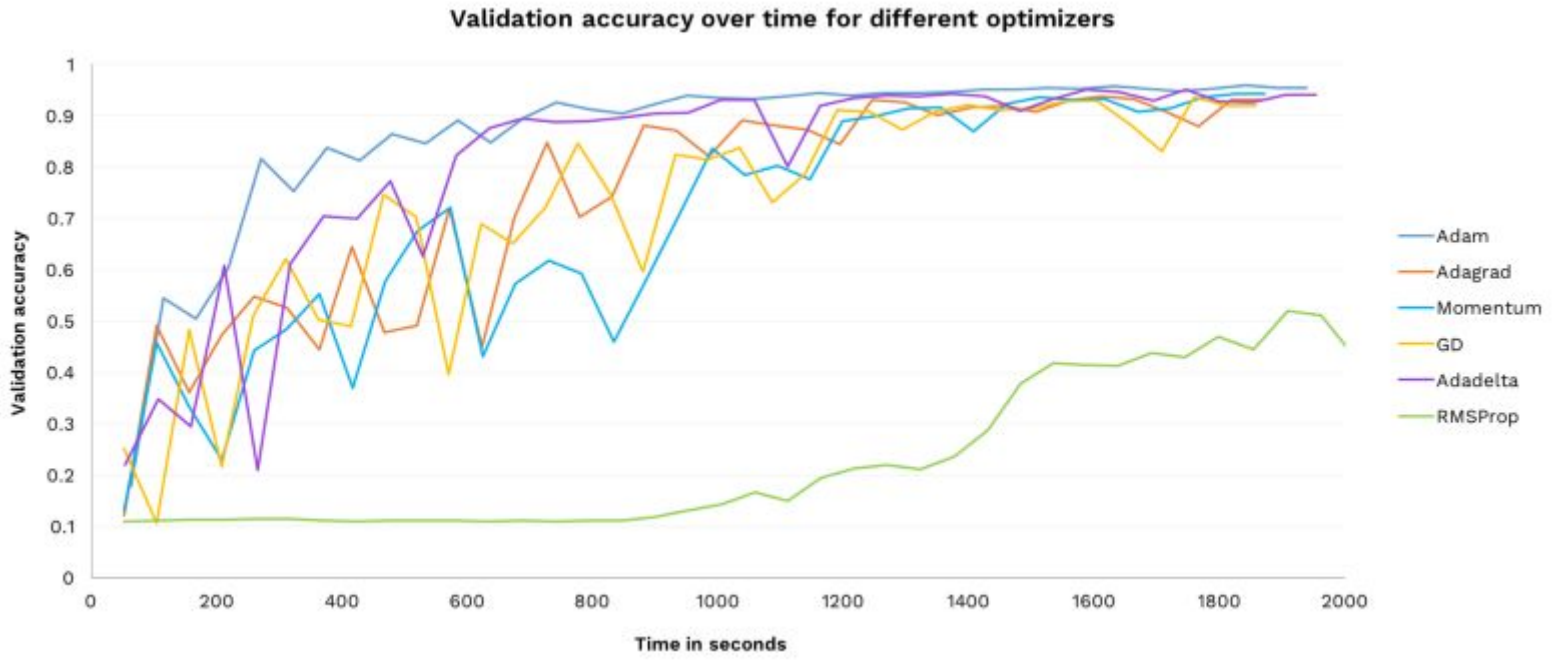
[1,0,0]
[0,1,0]
[0,0,1]

- For integer y, use sparse_categorical_crossentropy.

1
2
3

optimizers

```
optimizer='Adam'

from keras.optimizers import SGD
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```



Stochastic Gradient Descent (SGD)

Adam

Adagrad

Adadelta

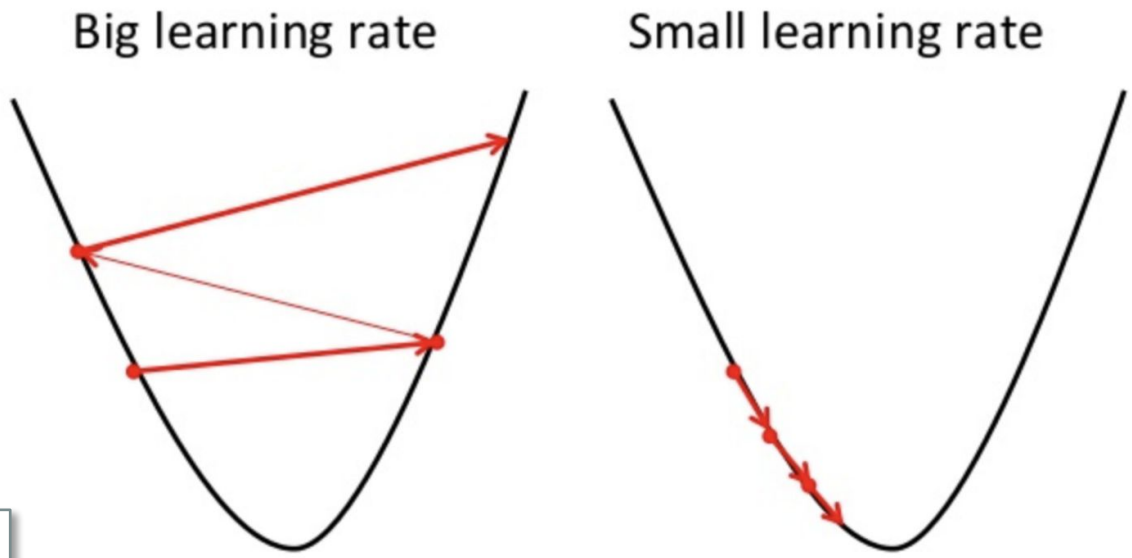
RMS Prop

Momentum.

learning rate

- The amount that the weights are updated during training
  트레이닝할때 가중치가 업데이트되는 정도
- Referred as the step size
  스텝사이즈라고도 함

Start with a large value like 0.1, then try exponentially lower values: 0.01, 0.001, etc.
0.1에서 시작해서 0.01, 0.001을 시도

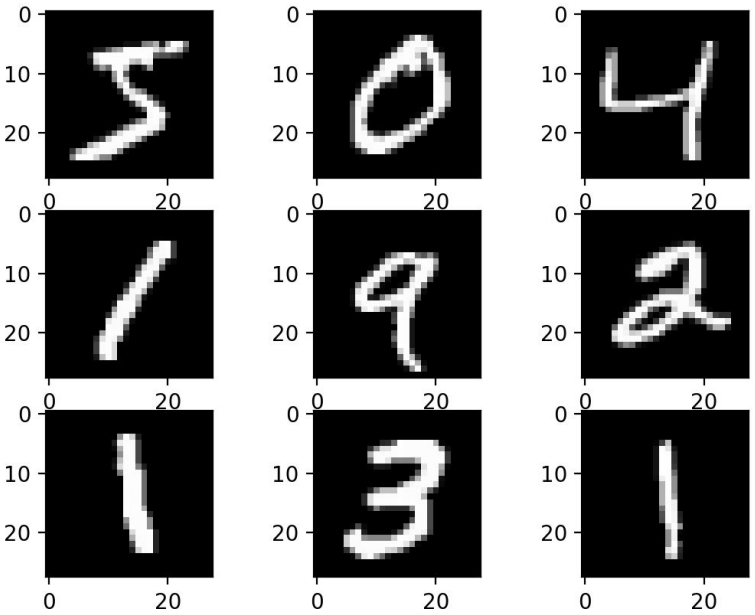


Exercise #4

- Improve Keras Testing Example with loss history graph (https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/) and tensorboard

mnist Dataset

- This dataset consists of 70,000 images of handwritten digits from 0–9. **7만개의 손으로 쓴 숫자이미지를 포함**
- Out of the 70,000 images provided in the dataset, 60,000 are given for training and 10,000 are given for testing. **6만개는 훈련용으로 만개는 테스트용으로 사용**



Exercise #4

- Use mnist to complete the following tasks (keras.datasets.mnist)
  - train and test split
  - Exploratory data analysis (EDA) – imshow, shape
  - Scaling – keras.utils.normalize(x_train, axis=1)
  - Sequential layers – 1 flatten layer, dense with 128 nodes , 128 nodes using ReLU, dense layer with 10 output using softmax)
  - Compile – adam optimizer, sparse_categorical_crossentropy (without y encoding), accuracy metrics
  - Fit – 3 epochs
  - Prediction
  - Evaluation (use evaluate and np.mean to compare y_test and y_pred)
  - Save and load the model to predict the third test data

Exercise #4

- Open mnist image files (from keras.datasets import mnist)
  - Train and test split (mnist.load_data())
  - Exploration (imshow, shape)
  - Preprocessing (x_train and y_train reshape, divided by 255 to normalize, y encoding using to_categorical)

Exercise #4

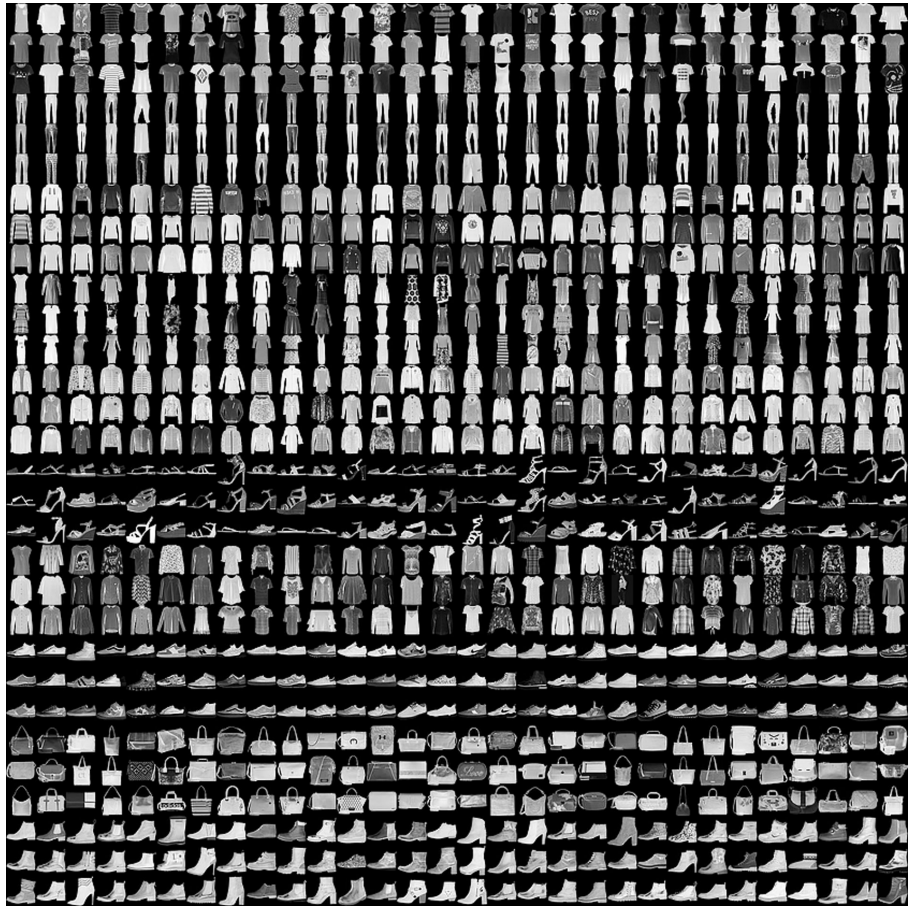
- Build a CNN model
  - conv2d with 32 nodes, 3x3 filters, relu activation function
  - conv2d with 32 nodes, 3x3 filters, relu activation function
  - Maxpooling with 2x2 filers
  - Dropout .25
  - Flatten
  - Dense with 128 nodes with relu activation
  - Dropout .25
  - Dense with softmax function

Exercise #4

- Compile the model
  - loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy']
- Fit the moel
  - batch_size=32, epoch=10, verbose=1
- Evaluation
  - Cnn.evaluate(x_test, y_test)
  - np.argmax
- Save and load the model to predict the first 5 image of test dataset

Fashion-MNIST dataset

- A dataset of Zalando's article images, with 28x28 grayscale images of 70,000 fashion products from 10 categories, and 7,000 images per category. **10개의 분류가 있고, 각각에 7천개의 이미지를 가짐. 각각은 28x28의 회색이미지**
- The training set has 60,000 images, and the test set has 10,000 images. **훈련셋은 6만개의 이미지, 테스트셋은 만개의 이미지**



Exercise #4

- Build a CNN model using Fashion-MNIST dataset on https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python

Exercise #4

- Load fashion_mnist dataset and split it into train and test set (keras.datasets.fashion_mnist)
- Exploratory data analysis (imshow of 9 images of train and the 9 images of test, shape of train and test, and number of classes and names of classes)
- Proprocessing (reshape, divided by 255 to normalize, encoding y)

Exercise #4

- Build a CNN model
  - conv2d with 32 nodes, 3x3 filters, relu activation function
  - Maxpooling with 2x2 filers
  - Dropout .25
  - conv2d with 64 nodes, 3x3 filters, relu activation function
  - Maxpooling with 2x2 filers
  - Dropout .25
  - conv2d with 128 nodes, 3x3 filters, relu activation function
  - Maxpooling with 2x2 filers
  - Dropout .25
  - Flatten
  - Dense with 128 nodes with relu activation
  - Dropout .3
  - Dense with softmax function

Exercise #4

- Compile the model
  - loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy']
- Fit the model
  - batch_size=64,epochs=5,verbose=1
- Prediction
  - y_pred = cnn.predict(x_test)

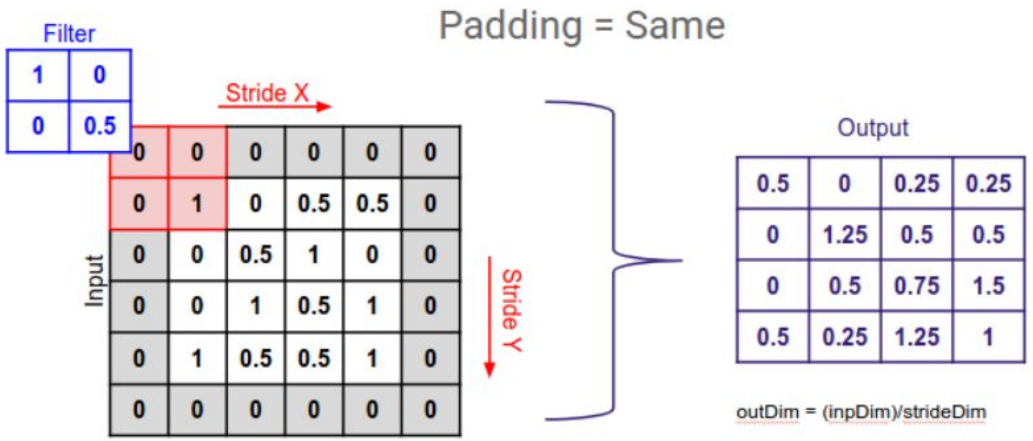
Exercise #4

- Evaluate the model
  - Loss and accuracy score (cnn.evaluate(x_test, y_test)
  - Classification report (classification_report(y_test, y_pred)
  - Training and validation accuracy plot and Training and validation loss plot

Plot Accuracy and Loss

```
accuracy = fashion_train_dropout.history['acc']
val_accuracy = fashion_train_dropout.history['val_acc']
loss = fashion_train_dropout.history['loss']
val_loss = fashion_train_dropout.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
```

Padding



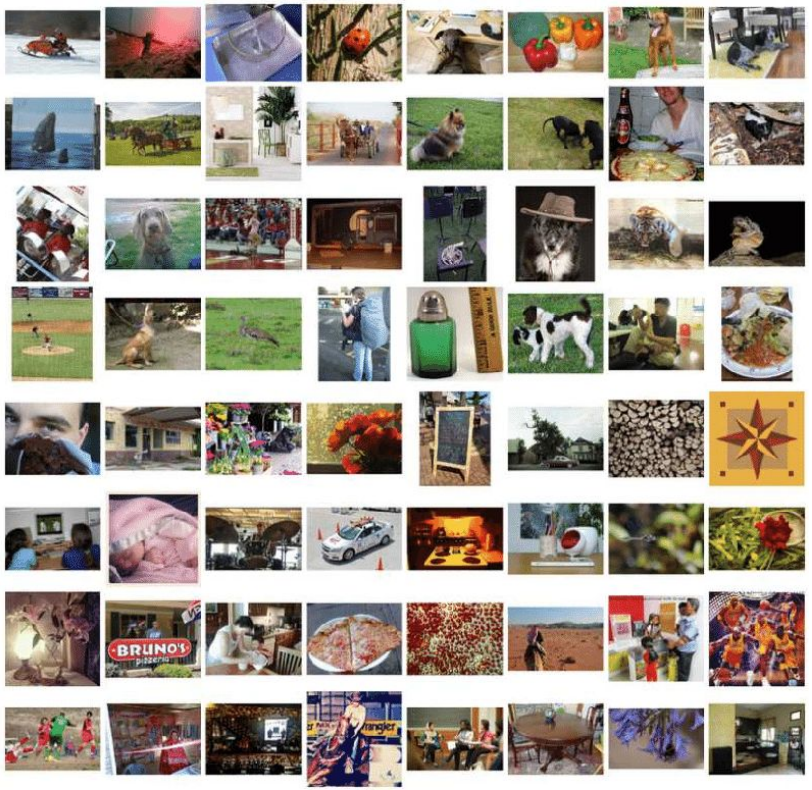
- An additional layer that added to the border of an image to overcome shrinking outputs and loosing information on corners of the image

레이어하나를 더해서 컨볼루션, 맥스풀링시 데이터가 손실되는 것 또는 코너의 데이터를 잃는 것을 방지

Imagenet

http://www.image-net.org/

- A repository of millions of digital images to classify a dataset into categories like cats and dogs. 고양이와 개를 구분하기위한 이미지들의 저장소
- DL nets are increasingly used for dynamic images apart from static ones and for time series and text analysis. 움직이는 이미지나 시계열, 텍스트분석에 사용됨



Exercise #4

- Build a CNN model using Cats and Dogs Dataset on https://www.microsoft.com/en-us/download/details.aspx?id=54765

Exercise #4

- Build a prediction model using CNN
  - College Dataset
  - Boston Housing Dataset
  - AirPassanger Dataset
  - SECOM Dataset

College Dataset

Use ISLR's built in College Data Set which has several features of a college and a categorical column and build a predictive model to determine whether or not the School is Public or Private

컬리지 데이터셋에는 대학에 관한 여러가지 특징들과 범주형 변수들이 포함돼 있습니다. 그 대학이 공립인지 사립인지 결정하는 예측모델을 구축하시요.

```
library(ISLR)
data(College)
```

Boston Housing Dataset

The Boston dataset is a collection of data about housing values in the suburbs of Boston. Use the Boston dataset in the MASS package and predict the median value of owner-occupied homes (medv) using all the other continuous variables available.

보스톤 데이터셋에는 보스톤 교외에 있는 집의 가치에 대한 데이터를 포함합니다. 포함되어 있는 모든 연속변수를 사용하여 집의 중위값을 예측하시요.

```
library(MASS)
data <- Boston
```

AirPassengers

AirPassengers dataset includes monthly Airline Passenger Numbers 1949-1960. Build a prediction model for Airline Passenger Number. 에어패씬저 데이터셋에는 1949년에서 1960년까지의 월별 승객수에 대한 데이터를 포함하고 있습니다. 승객수를 예측하는 모델을 만드시요.

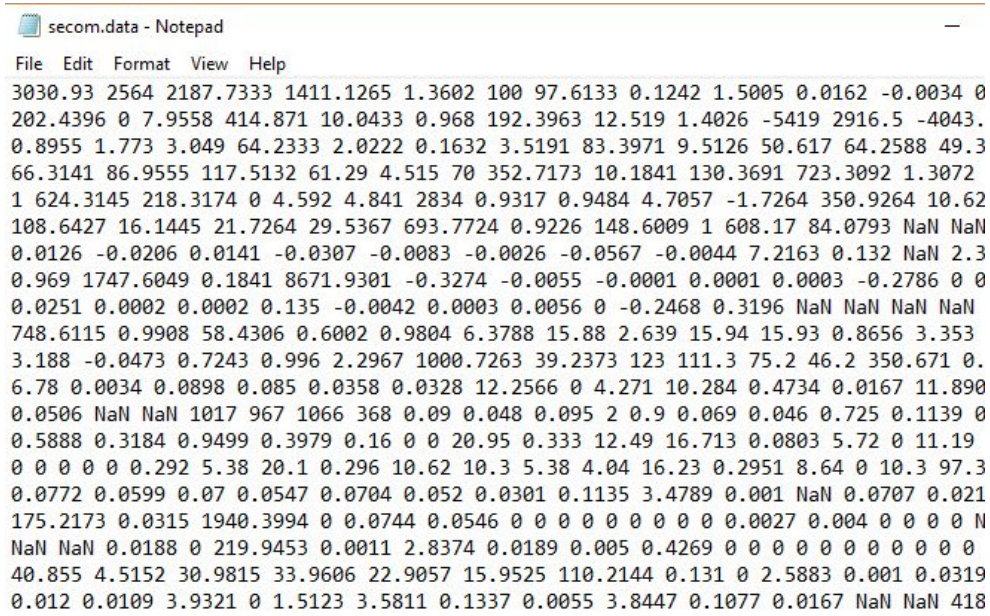
```
Data("AirPassengers")
```

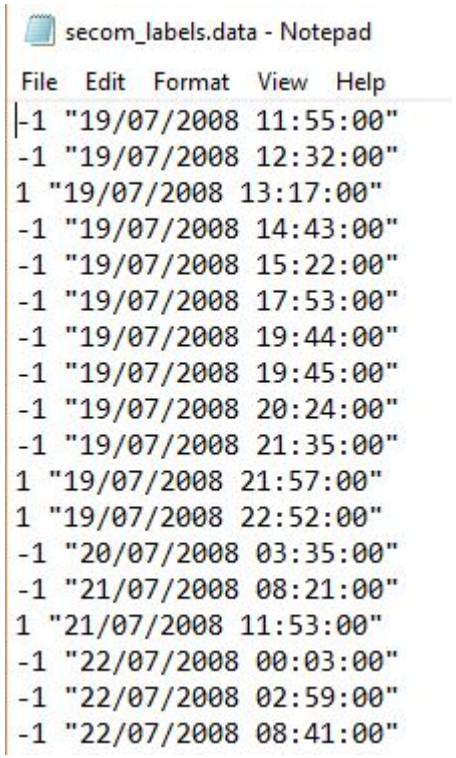
SECOM Dataset

Semiconductor Manufacturing (SECOM) dataset consisting of manufacturing operation data (secom.data) and the semiconductor quality data (secom_labels.data). It includes 590 sensor measurements, a label of pass/fail test, and date. 1567 observations taken from a wafer fabrication production line and 104 fail cases (1 is fail and -1 is pass). Build a predictive model to determine whether a semiconductor is failed or passed. **반도체제조 데이터셋은 제조과정데이터와 품질데이터로 구성되어 있습니다. 590개의 센서의 측정치와 성공/실패 결과, 날짜를 포함합니다. 와퍼제고 생산라인에서 수집한 1567개의 관측치중 104개가 실패했습니다. 반도체가 실패인지 통과인지 결정하는 예측모델을 만드시요.**

SECOM Dataset

http://archive.ics.uci.edu/ml/datasets/secom





secom.data
1567 x 590

secom_labels.data
1567 x 2