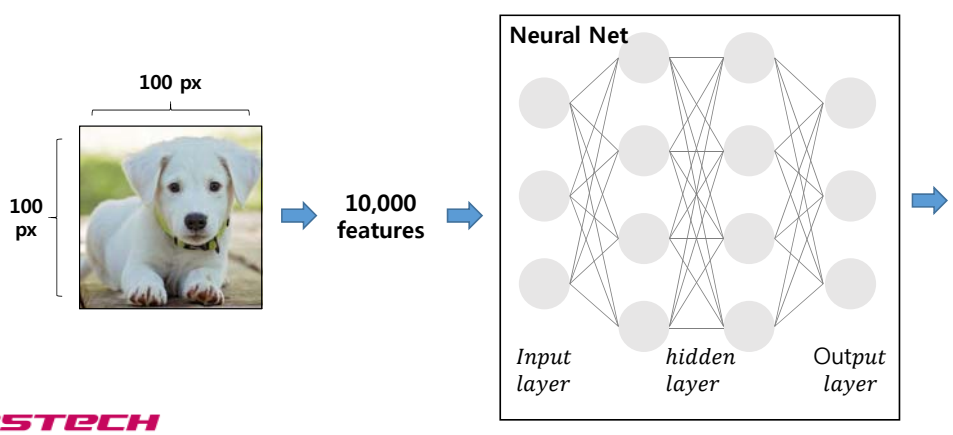
Wk16-2 : Convolutional Neural Network

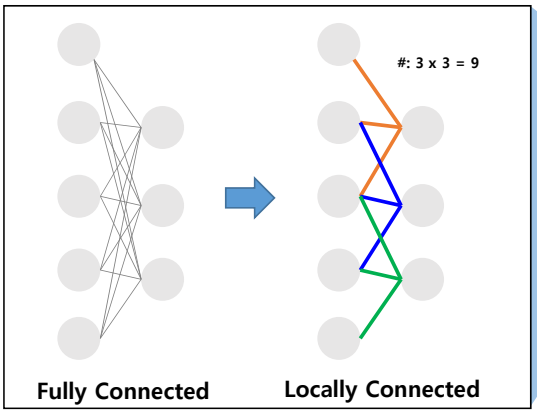
1. Features

- 신경망 모델(Neural Net)은 입력값으로 객체의 특성(feature)을 받고,
  - 출력된 값과 실제 값을 비교하는 과정을 거침 (지도학습; Supervised Learning)
- 하나의 이미지는 수많은 픽셀들이 모여 형성하고 있으며, 특정 색에 해당하는 특정 값을 가짐
  - 따라서, 이미지의 모든 픽셀값들을 입력값으로 갖는 신경망 모델을 만들 수 있음



2. Intuitions

- 하지만, 고해상도 이미지의 경우 특성feature의 수가 너무 많아지므로
  - 모든 뉴런들이 모든 픽셀들과 모두 연결되어 있을 경우 (fully connected) 모델 학습에 큰 어려움이 있음
  - 따라서, 각 뉴런들이 이미지의 일부의 특성feature만 연결될 수 있는 구조가 더 적합함
  - Convolution operation을 통해 이를 구현할 수 있음.



Feed forward Network: x_i^n을 구한다

Convolution

Max Pooling

Activation function

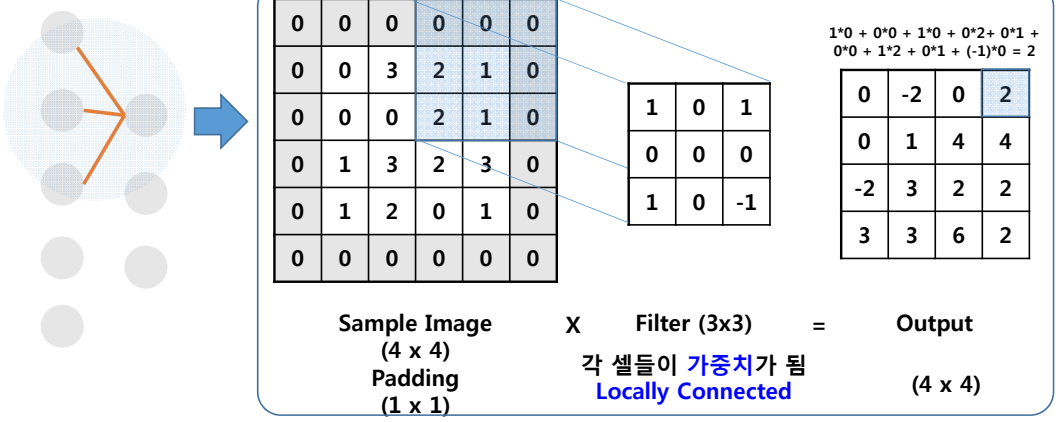
Backpropagation: Error 최소화함

Convolutional Neural Network

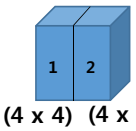
3. Convolution Operation

- 임의의 값으로 설정된 filter가 전체 이미지 중 일부의 선형 결합을 계산함
  - 각각의 결과값은 하나의 Neuron이 되며, filter는 해당 Neuron의 가중치가 됨
  - 결과값의 사이즈를 정하기 위해선 Stride, Padding 그리고 Depth을 고려해야함

$$Output = \frac{W-F+2P}{S} + 1$$



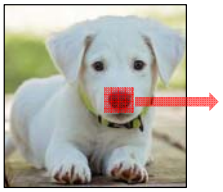
- **Stride**
  - filter를 몇 칸 이동할지를 결정. 왼쪽 예에서는 stride 1임
- **Padding**
  - input 주변에 0으로 padding을 삽입. 왼쪽 예에서 padding은 (1,1)임
- **Depth(number of filter)**
  - 3차원 상의 neuron의 깊이를 결정. 만약 filter (3x3) 가 두 개가 있다면 다음과 같이 Output이 형성될 것



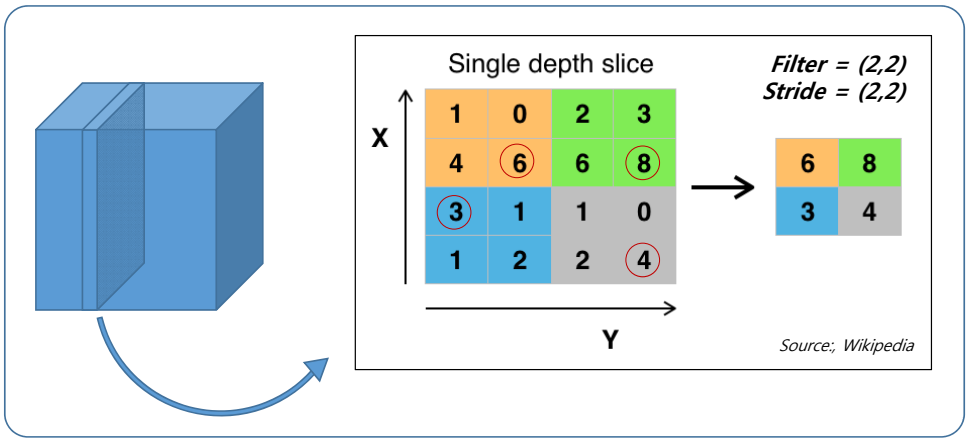
4. Pooling

- Convolutional Layer 사이에 Pooling Layer를 넣어주는 방법이 많이 사용됨
  - 추출해낸 이미지에서 지역적인 부분의 특징만을 뽑아 다음 layer로 넘겨줌
  - 이를 통해, 1) 가중치들의 수를 줄일 수 있으며 2) 과적합(overfitting)을 방지함
  - 대표적으로 가장 큰 값(Local Maxima)만을 뽑아내는 Max Pooling이 많이 사용됨

$$Output = \frac{W-F}{S} + 1$$



코를 나타내는 픽셀들은 모두 검정색으로 되어 있으니, 검정색 하나의 값만 있으면 됨



5. MNIST

- 손으로 쓴 숫자들을 인식하기 위해 사용되는 데이터
  - 28x28 pixel의 흑백 이미지(0~255)들이 있음
  - 0부터 9까지 총 70,000개의 손글씨 이미지들이 있음
  - 상대적으로 신경망 모델을 학습하기에 작은 사이즈를 가지고 있음
  - 출처(http://yann.lecun.com/exdb/mnist)

MNIST (Modified National Institute of Standards and Technology)
데이터 : 손으로 쓴 숫자들로 이루어진 대형 데이터베이스

Source:, Wikipedia

6. CNN in R

- 신경망 모델 생성을 위한 패키지: mxnet

```
# lec16_2_cnn.r
# Convolutional Neural Network
# Require mxnet package
# install.packages("https://github.com/jeremiedb/mxnet_winbin/raw/master/
library(mxnet)

# If you have Error message "no package called XML or DiagrmmeR", then ir
#install.packages("XML")
#install.packages("DiagrammeR")
```

mxnet 라이브러리 설정시 아래와 같은 Error메세지가 나오면, DiagrmmeR 패키지 설치. XML도 설치필요할 수 있음

```
> library(mxnet)
Error: package or namespace load failed for 'mxnet' 
], c(lib.loc, .libPaths()), versionCheck = vI[[j]]):
 there is no package called 'DiagrammeR'
In addition: Warning message:
package 'mxnet' was built under R version 3.4.4
```


6. CNN in R

- MNIST 데이터 불러오기

```
# Load MNIST mn1
# 28*28, 1 channel images
mn1 <- read.csv("mini_mnist.csv")
set.seed(123)
N<-nrow(mn1)
tr.idx<-sample(1:N, size=N*2/3, replace=FALSE)

# split train data and test data
train_data<-data.matrix(mn1[tr.idx,])
test_data<-data.matrix(mn1[-tr.idx,])

test<-t(test_data[,-1]/255)
features<-t(train_data[,-1]/255)
labels<-train_data[,1]

# data preprocession
features_array <- features
dim(features_array) <- c(28,28,1,ncol(features))
test_array <- test
dim(test_array) <- c(28,28,1,ncol(test))

ncol(features)
table(labels)
```

학습 데이터: 2/3
테스트 데이터: 1/3

0과 1사이에 분포하도록(Normalized)
(0: 검정색/ 255: 흰색)

입력 데이터의 차원을 설정 (픽셀 * 객체 개수)
ncol(features): 학습 데이터 수(866)

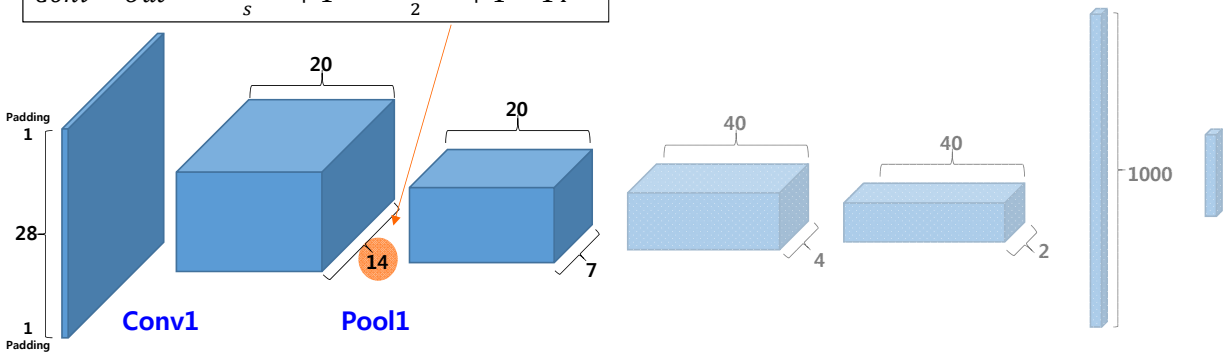
```
> ncol(features)
[1] 866
> table(labels)
labels
  0   1   2
282 307 277
```

6. CNN in R

• **Convolutional Layer 구성**

```
# Build cnn model
# first conv layers
my_input = mx.symbol.Variable('data')
conv1 = mx.symbol.Convolution(data=my_input, kernel=c(4,4), stride=c(2,2), pad=c(1,1), num.filter = 20, name='conv1')
relu1 = mx.symbol.Activation(data=conv1, act.type='relu', name='relu1')
mp1 = mx.symbol.Pooling(data=relu1, kernel=c(2,2), stride=c(2,2), pool.type='max', name='pool1')
```

$$Conv - Out = \frac{W-F+2P}{s} + 1 = \frac{28-4+2*1}{2} + 1 = 14$$

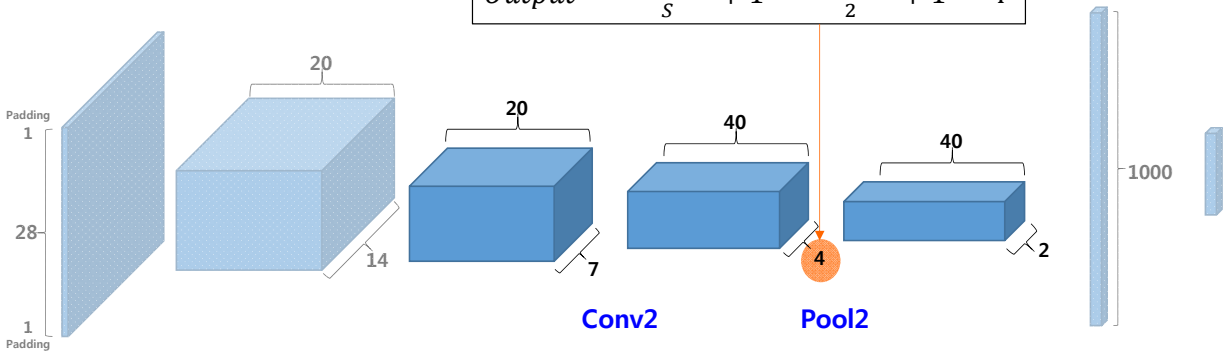


6. CNN in R

• **Convolutional Layer 구성**

```
# second conv layers
conv2 = mx.symbol.Convolution(data=mp1, kernel=c(3,3), stride=c(2,2), pad=c(1,1), num.filter = 40, name='conv2')
relu2 = mx.symbol.Activation(data=conv2, act.type='relu', name='relu2')
mp2 = mx.symbol.Pooling(data=relu2, kernel=c(2,2), stride=c(2,2), pool.type='max', name='pool2')
```

$$Output = \frac{W-F+2P}{S} + 1 = \frac{7-3+2*1}{2} + 1 = 4$$



6. CNN in R

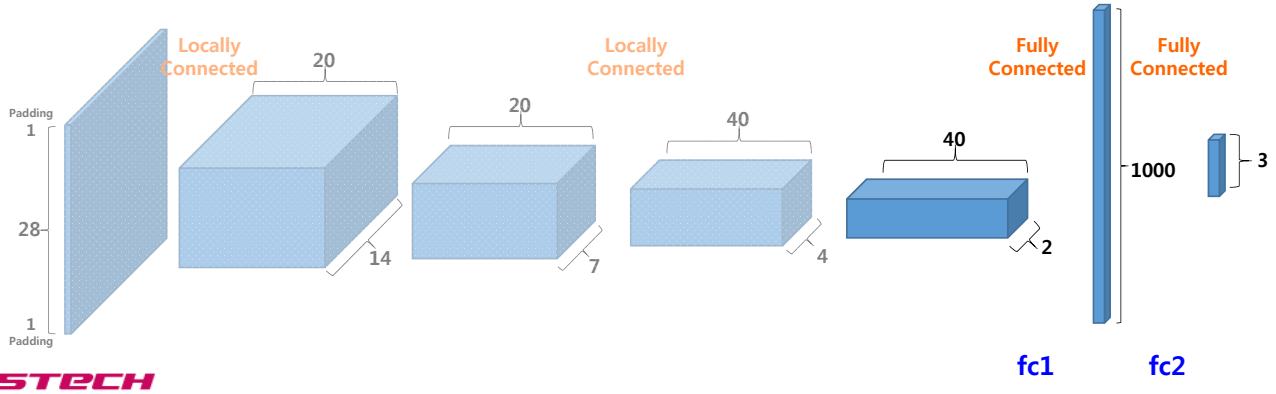
- **Convolutional Layer 구성**

```
# fully connected
fc1 = mx.symbol.FullyConnected(data=mp2, num.hidden = 1000, name='fc1')
relu3 = mx.symbol.Activation(data=fc1, act.type='relu', name='relu3')
fc2 = mx.symbol.FullyConnected(data=relu3, num.hidden = 3, name='fc2')

# softmax
sm = mx.symbol.SoftmaxOutput(data=fc2, name='sm')
```

1000개의 뉴런들이 모두 연결되어 있음

3개의 뉴런들 중 가장 확률이 높은 값이 0~2 중 하나를 가리킴



6. CNN in R

- **모델 훈련**

```
# training
mx.set.seed(100)
device <- mx.cpu()
model <- mx.model.FeedForward.create(symbol=sm,
                                     optimizer = "sgd",
                                     array.batch.size=30,
                                     num.round = 70, learning.rate=0.1,
                                     X=features_array, y=labels, ctx=device,
                                     eval.metric = mx.metric.accuracy,
                                     epoch.end.callback=mx.callback.log.train.metric(100))
graph.viz(model$symbol)
```

Stochastic Gradient Descent
batch size = 30 (총 29개 그룹)

Iteration(epoch): 70
Learning Step: 0.1

```
[59] Train-accuracy=0.973563218390804
[60] Train-accuracy=0.973563218390804
[61] Train-accuracy=0.975862068965517
[62] Train-accuracy=0.975862068965517
[63] Train-accuracy=0.977011494252873
[64] Train-accuracy=0.97816091954023
[65] Train-accuracy=0.981609195402299
[66] Train-accuracy=0.981609195402299
[67] Train-accuracy=0.982758620689655
[68] Train-accuracy=0.982758620689655
[69] Train-accuracy=0.983908045977011
[70] Train-accuracy=0.982758620689655
```

6. CNN in R

- 모델 테스트

```
# test
predict_probs <- predict(model, test_array)
predicted_labels <- max.col(t(predict_probs)) - 1
table(test_data[, 1], predicted_labels)
sum(diag(table(test_data[, 1], predicted_labels)))/length(predicted_labels)
```

```
> table(test_data[, 1], predicted_labels)
   predicted_labels
      0   1   2
  0 129   0   2
  1   0 161   0
  2   3   1 137
> sum(diag(table(test_data[, 1], predicted_labels)))/length(predicted_labels)
[1] 0.9861432
```


6. CNN in R

- 네트워크 시각화 함수 : graph.viz(model$symbol)

```
# training
mx.set.seed(100)
device <- mx.cpu()
model <- mx.model.FeedForward.create(symbol=s
                                     optimize
                                     array.ba
                                     num.roun
                                     X=featur
                                     eval.met
                                     epoch.en

graph.viz(model$symbol)
```