Digital Image Processing ECE 566

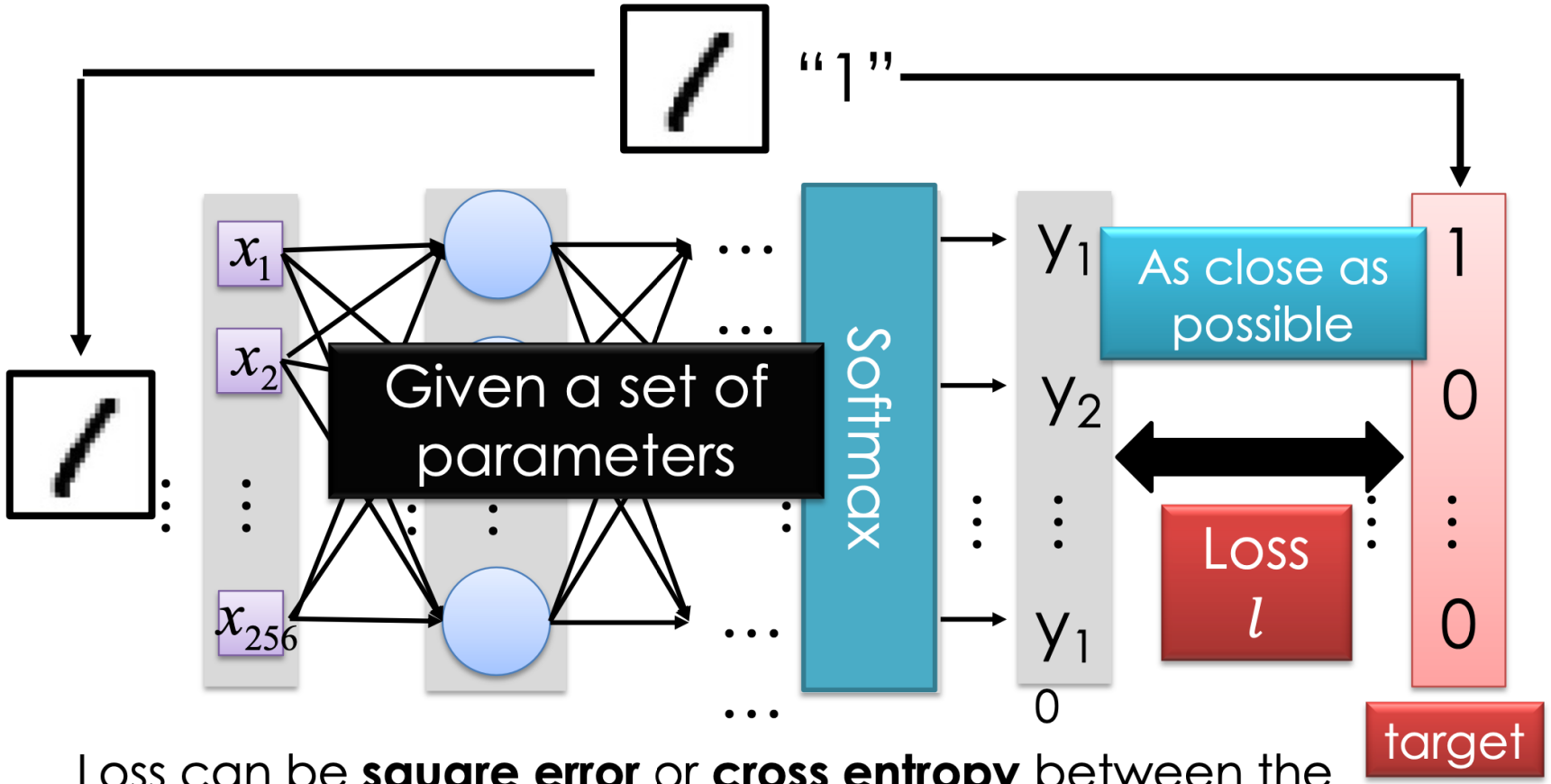
Ahmad Ghasemi

Department of Electrical and Computer Engineering

University of Massachusetts Amherst

Loss

A good function should make the loss of all examples as small as possible.

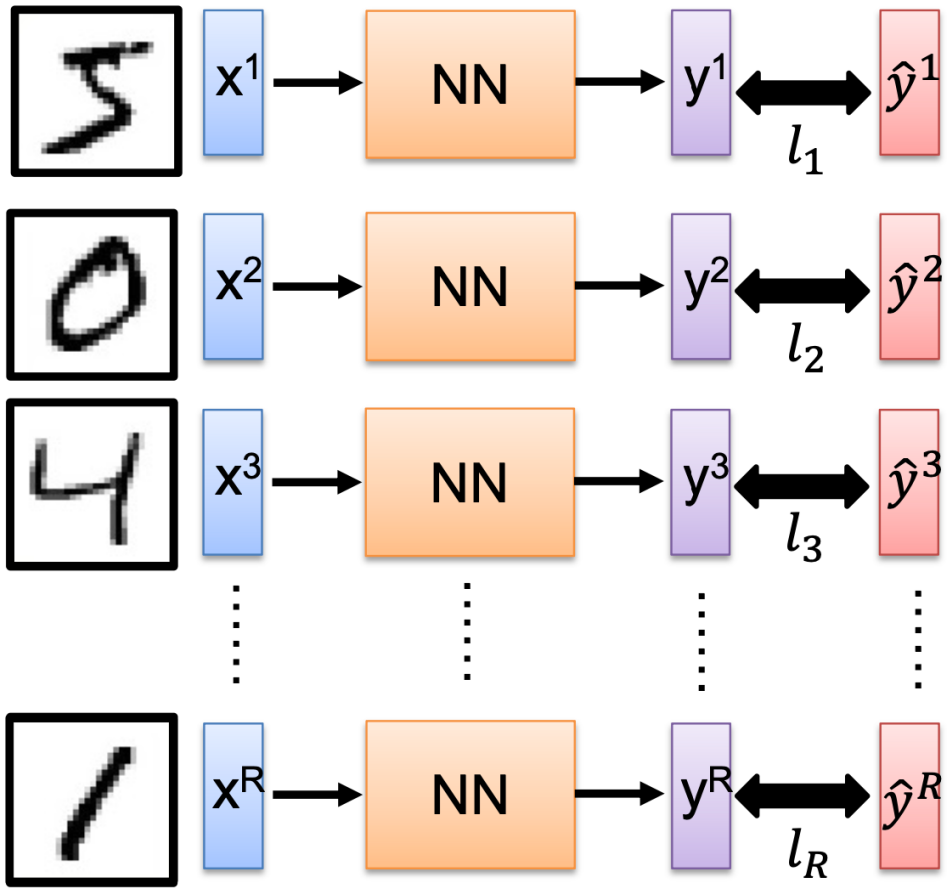


Loss can be **square error** or **cross entropy** between the network output and target

Credit: Hung-yi Lee, Andrew, Stanford course

Total Loss

For all training data …



Total Loss: $L = \sum_{r=1}^{R} l_r$

As small as possible

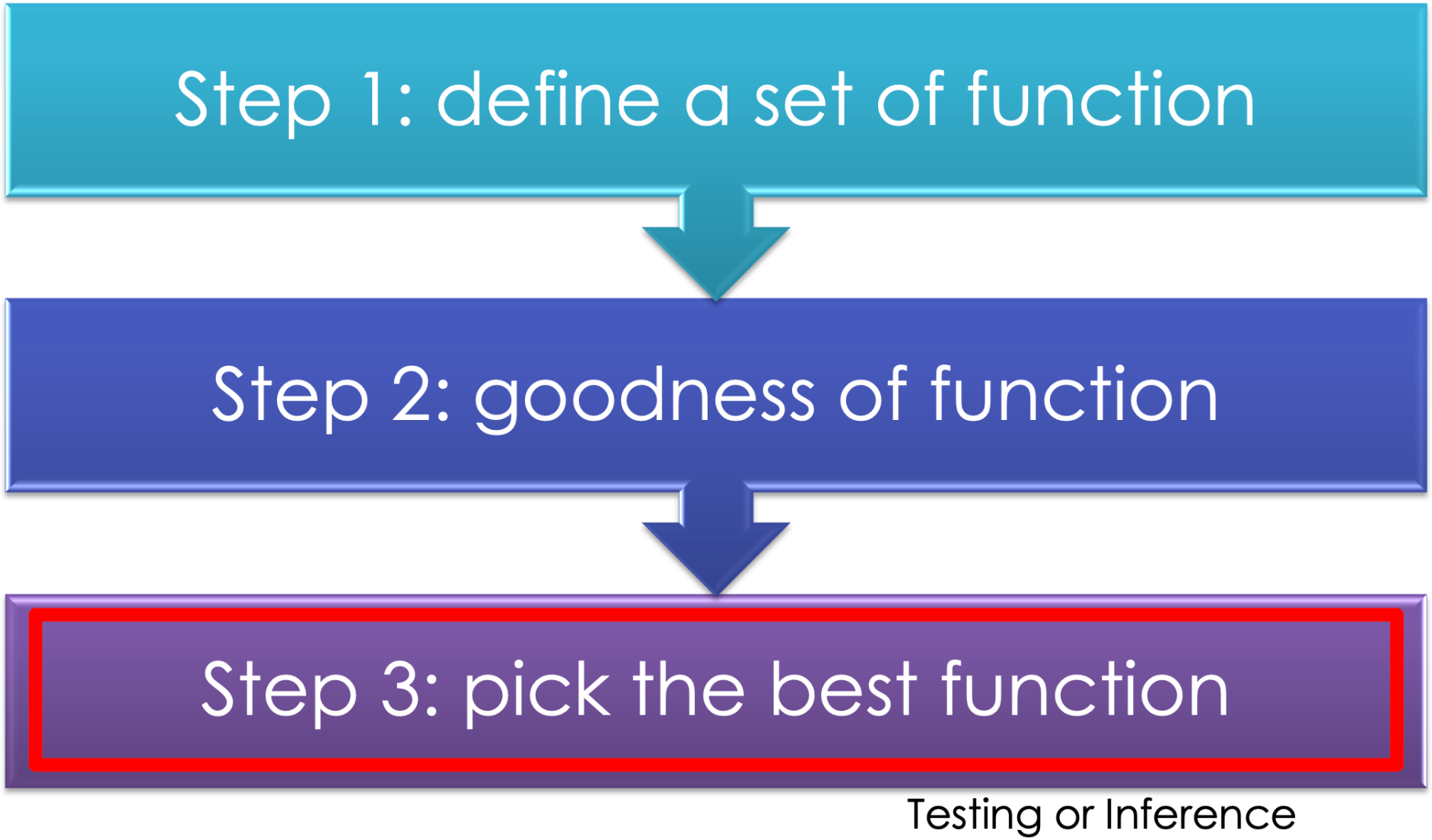
Find ***a function in function set*** that minimizes total loss L

Find ***the network parameters θ**** that minimize total loss L

Credit: Hung-yi Lee, Andrew, Stanford course

University of Massachusetts Amherst

Three steps for Deep Learning



Credit: Hung-yi Lee, Andrew, Stanford course

Convolutional Neural Network (CNN)

Widely used in image processing and computer vision

Credit: Hung-yi Lee, Andrew, Stanford course

Why CNN for Images?

- Some patterns are much smaller than the whole image

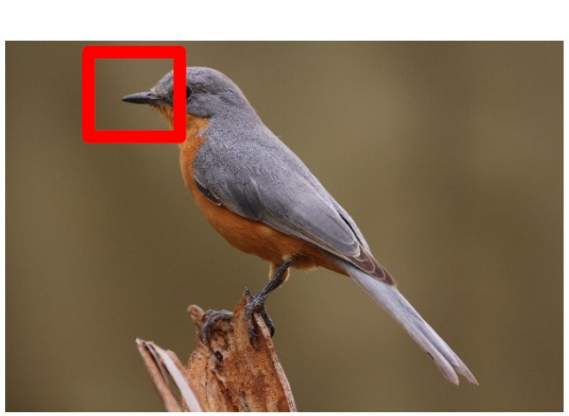
A neuron does not have to see the whole image to discover the pattern.

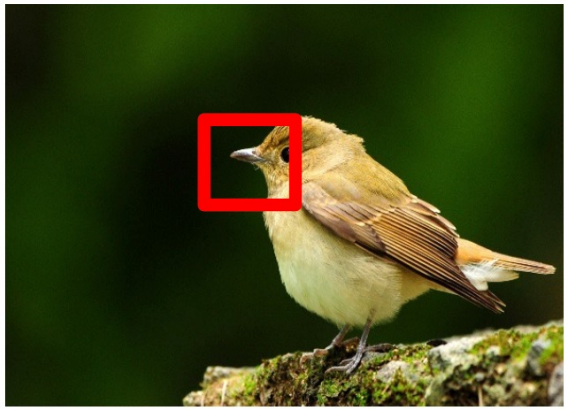
Connecting to small region with less parameters

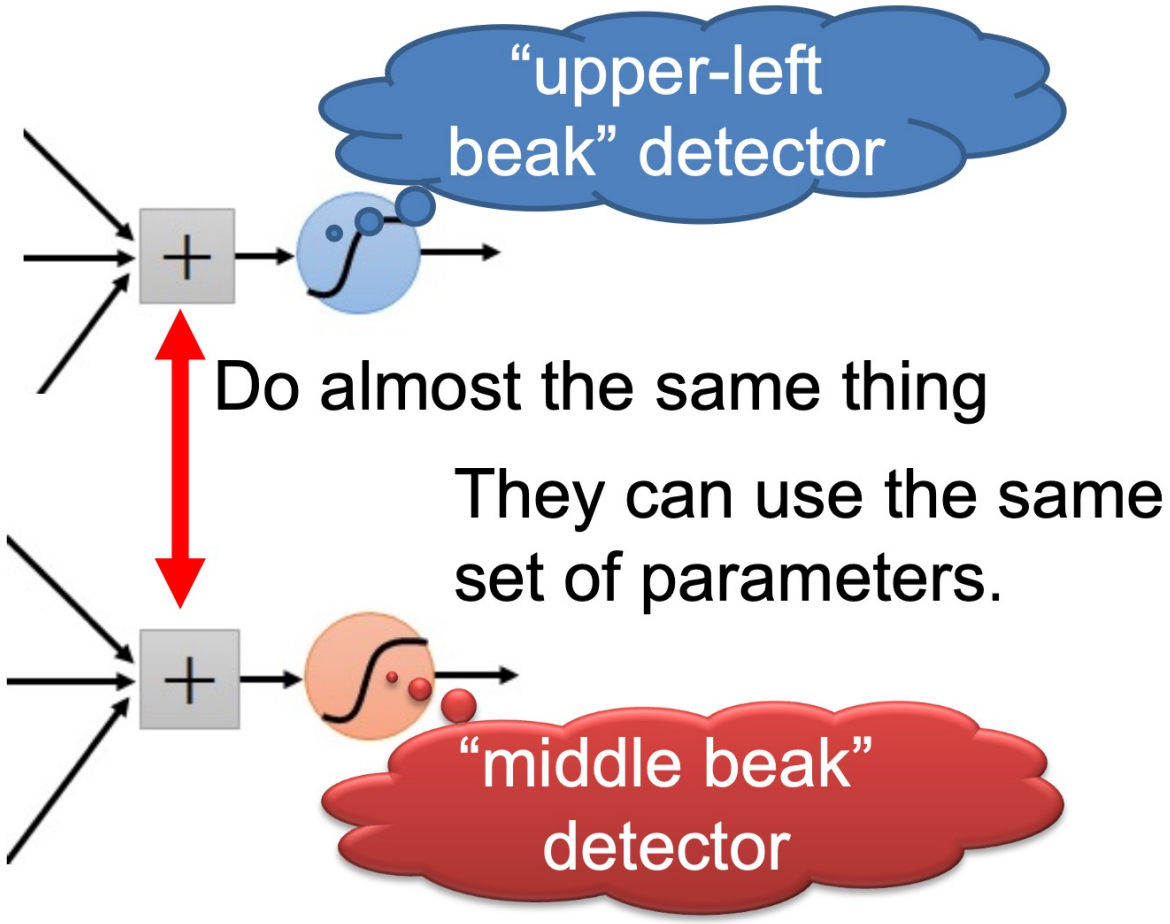
Credit: Hung-yi Lee, Andrew, Stanford course

Why CNN for Images?

- The same patterns appear in different regions.







Credit: Hung-yi Lee, Andrew, Stanford course

Why CNN for Images?

- Subsampling the pixels will not change the object

bird

subsampling

bird

We can subsample the pixels to make image smaller

→ Less parameters for the network to process the image

Credit: Hung-yi Lee, Andrew, Stanford course

University of Massachusetts Amherst

Why CNN for Images?

Property 1

- Some patterns are much smaller than the whole

Property 2

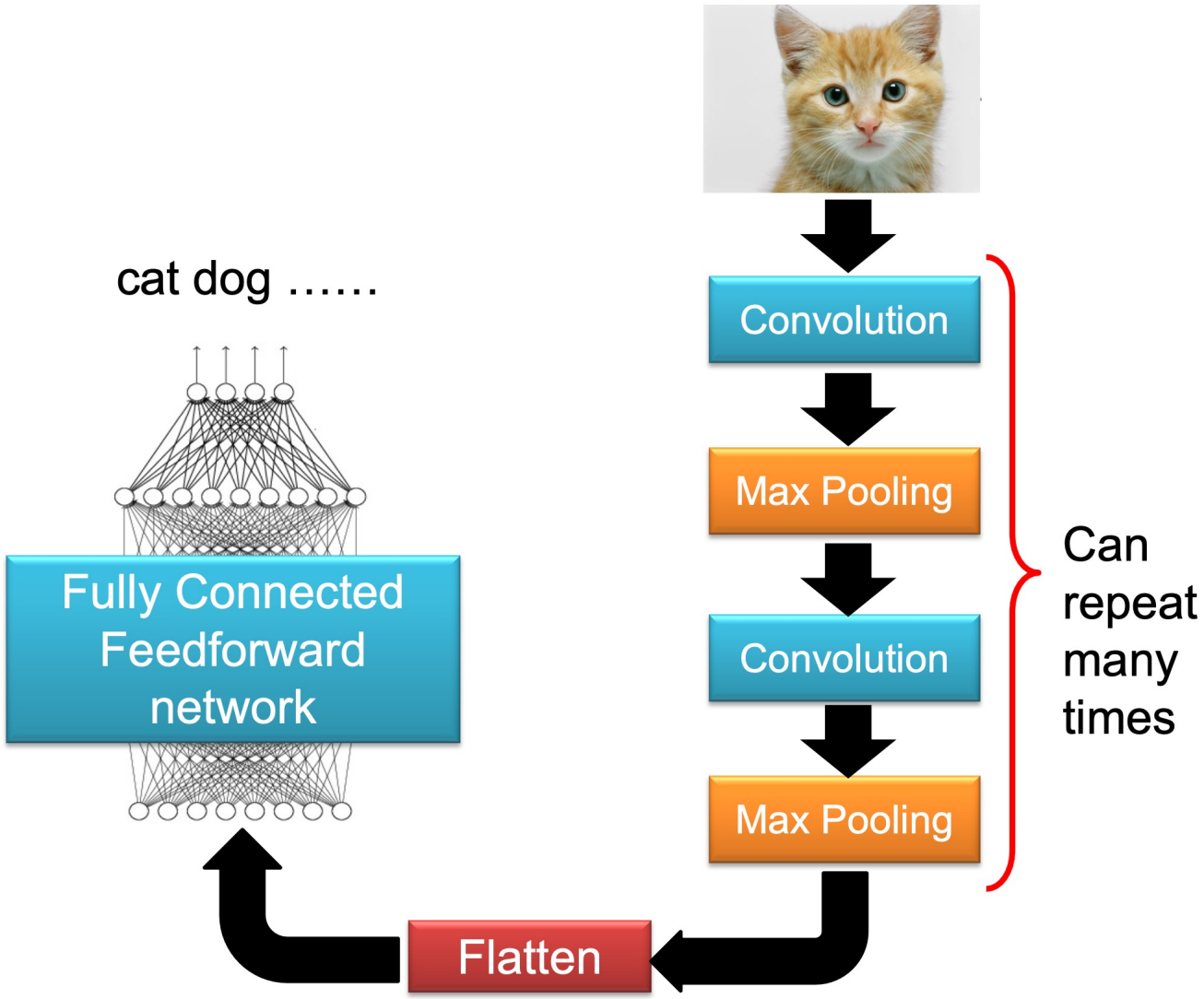
- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Credit: Hung-yi Lee, Andrew, Stanford course

The Whole CNN



Credit: Hung-yi Lee, Andrew, Stanford course

The Whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

Credit: Hung-yi Lee, Andrew, Stanford course

CNN: Convolution

Those are the network parameters to be learned.

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

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

Filter/kernel 1

Matrix

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

Filter 2

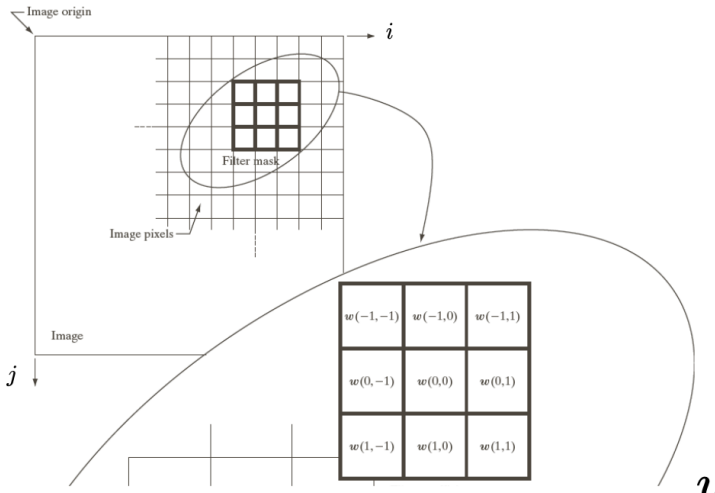
Matrix

⋮ ⋮

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

University of Massachusetts Amherst

CNN: Convolution



Filter/Kernel 1

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

filter previous level

$$u_{i,j}^{L} = \sum_{m,n} w_{i,j,m,n}^{L} a_{i-m,j-n}^{L-1}$$

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

3 -1

If stride=2

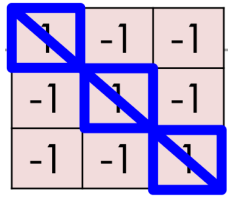
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

3 -3

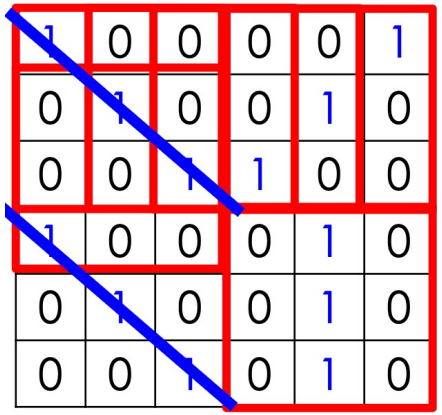
We set stride=1 below

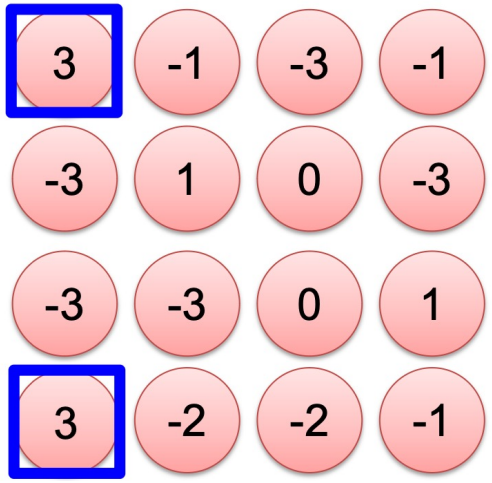
Credit: Hung-yi Lee, Andrew, Stanford course

CNN: Convolution



stride=1





Credit: Hung-yi Lee, Andrew, Stanford course

CNN: Convolution

Filter 2

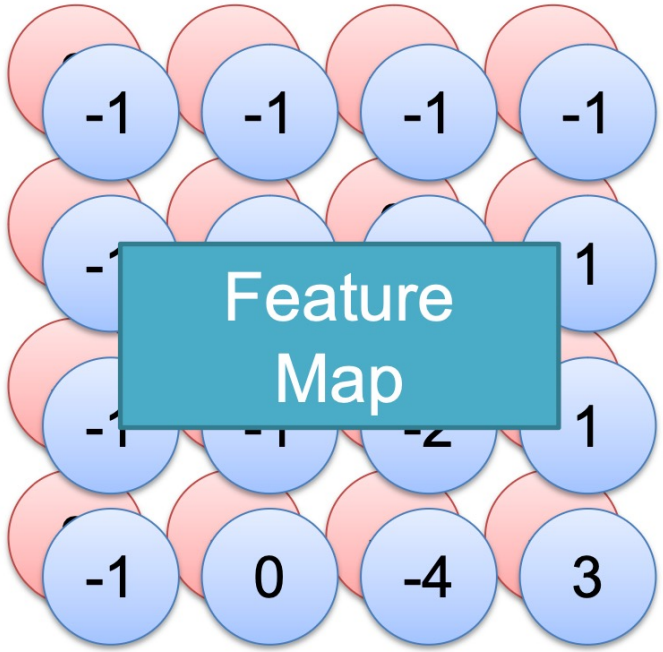
-1	1	-1
-1	1	-1
-1	1	-1

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

6 x 6 image

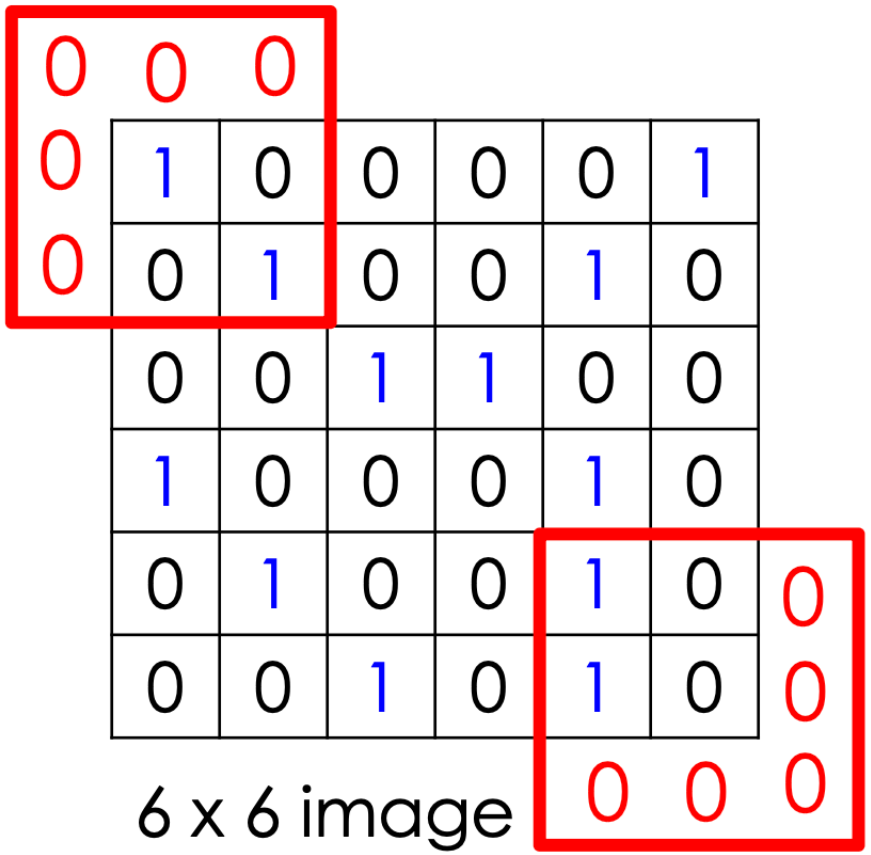
Do the same process for every filter



4 x 4 image (reduced size)

Credit: Hung-yi Lee, Andrew, Stanford course

CNN: Zero Padding

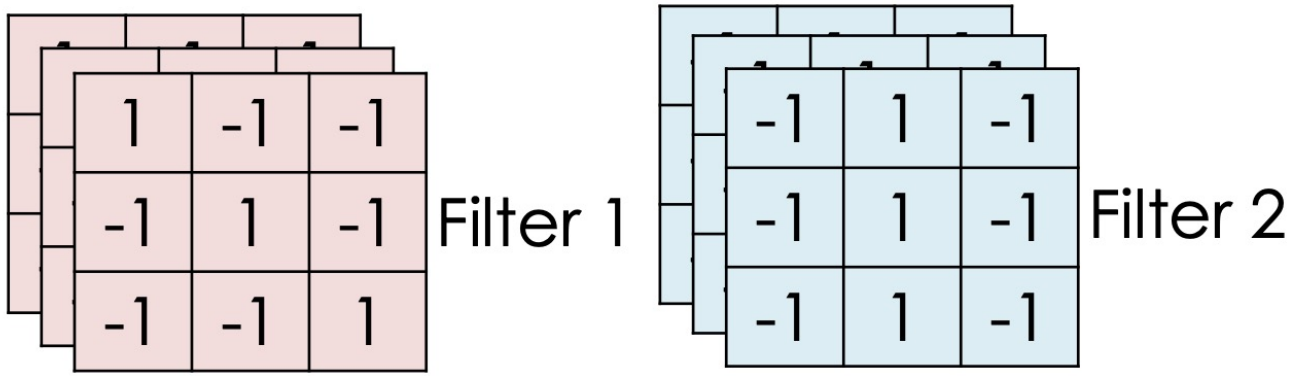


You will get another 6 x 6 image in this way

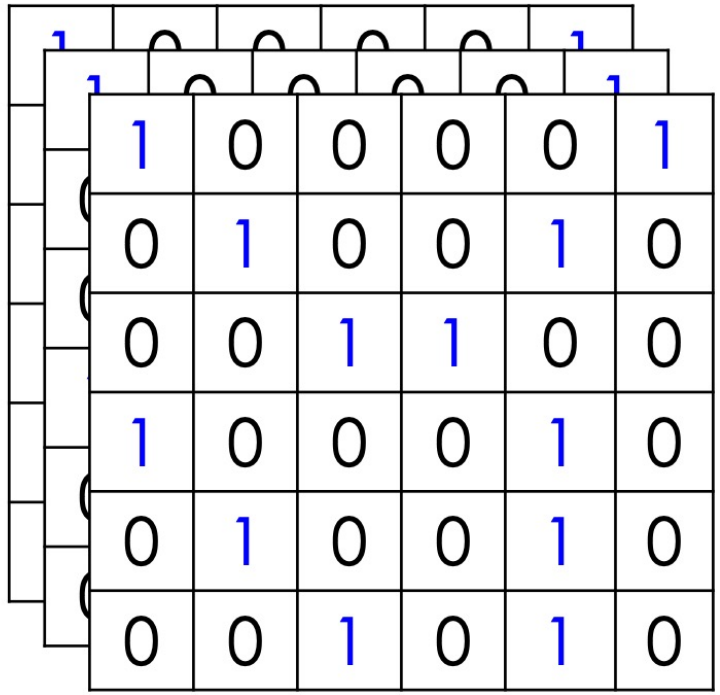
Zero padding

Credit: Hung-yi Lee, Andrew, Stanford course

CNN: Color Image



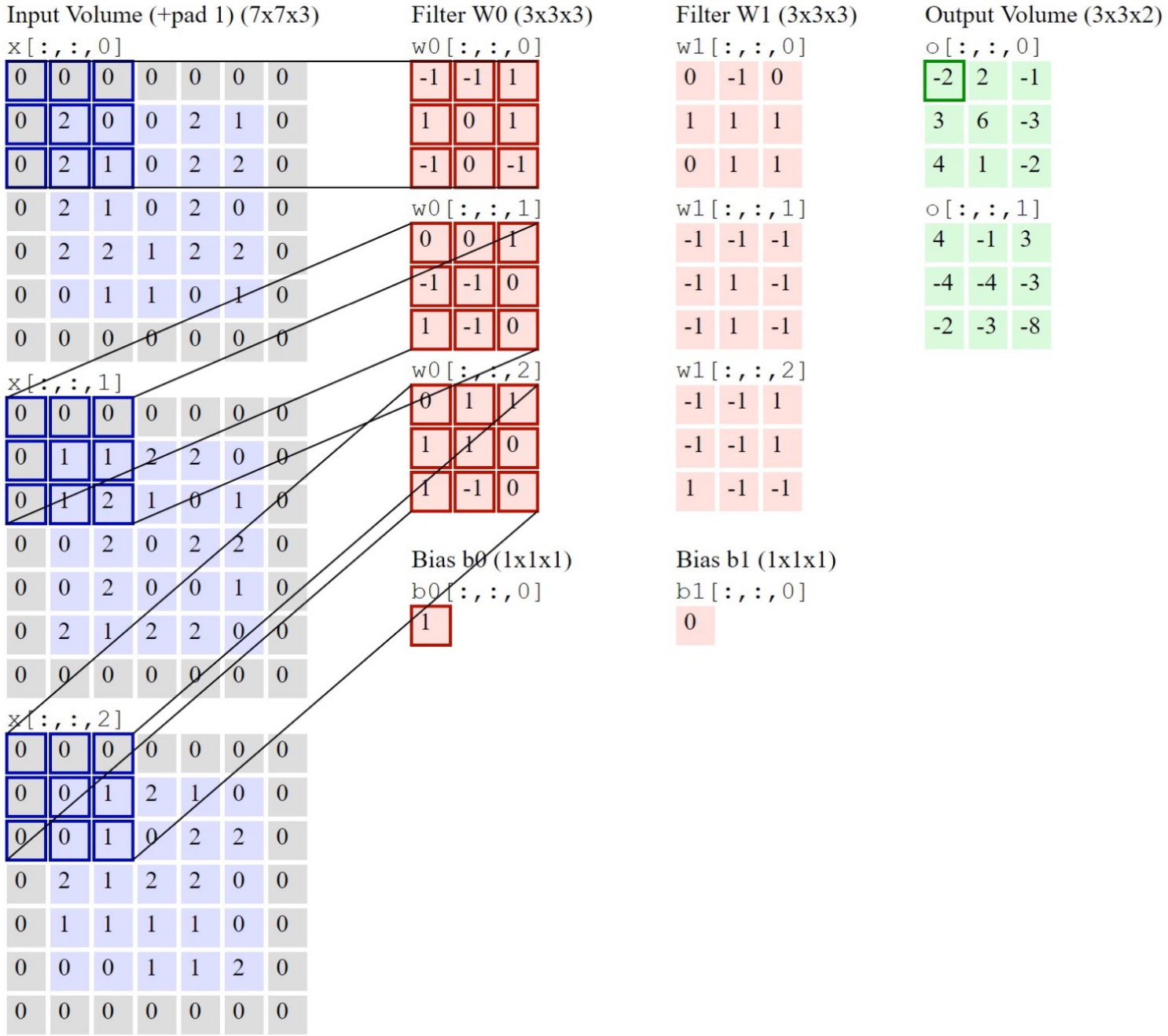
Color image



Credit: Hung-yi Lee, Andrew, Stanford course

University of Massachusetts Amherst

CNN: Color Image

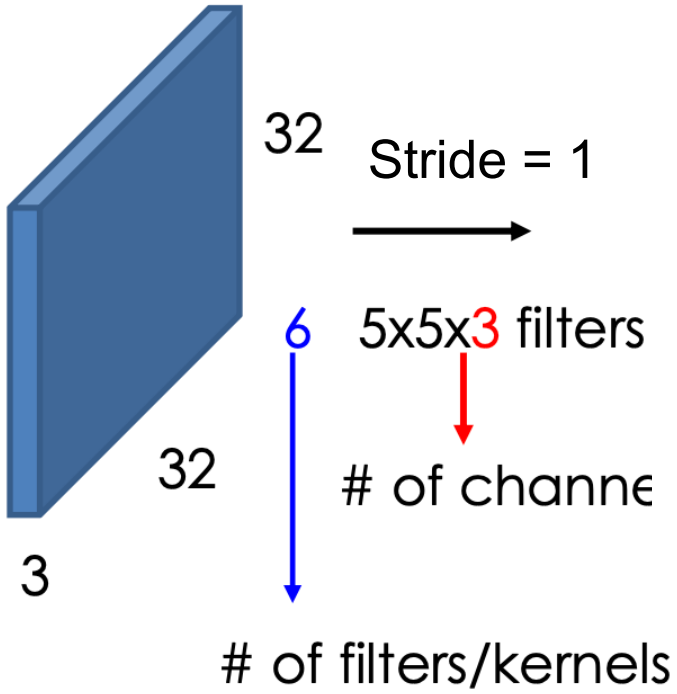


http://cs231n.github.io/convolutional-networks/

Convolutional Network: Example

- Convolution network is a sequence of these layers

$$\text{Output size} = \frac{\text{Input size} - \text{Filter size}}{\text{Stride}} + 1$$

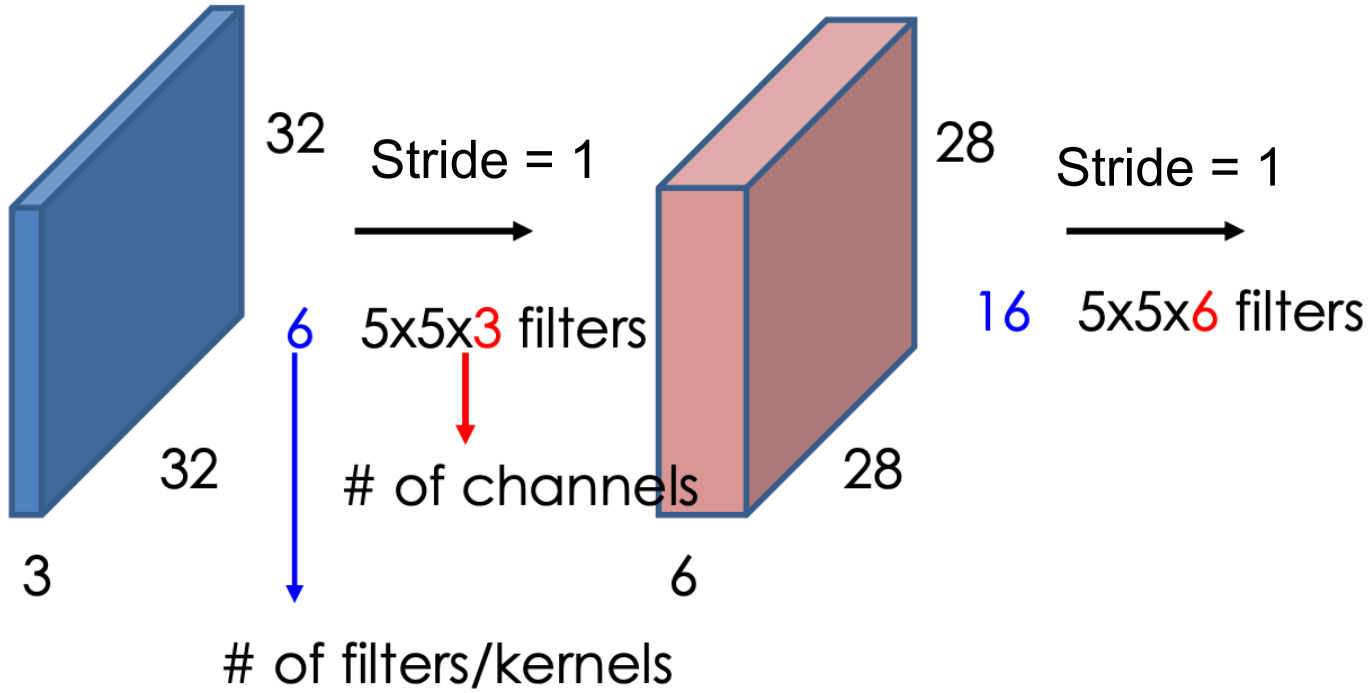


Credit: Hung-yi Lee, Andrew, Stanford course

Convolutional Network: Example

- Convolution network is a sequence of these layers

$$\text{Output size} = \frac{\text{Input size} - \text{Filter size}}{\text{Stride}} + 1$$

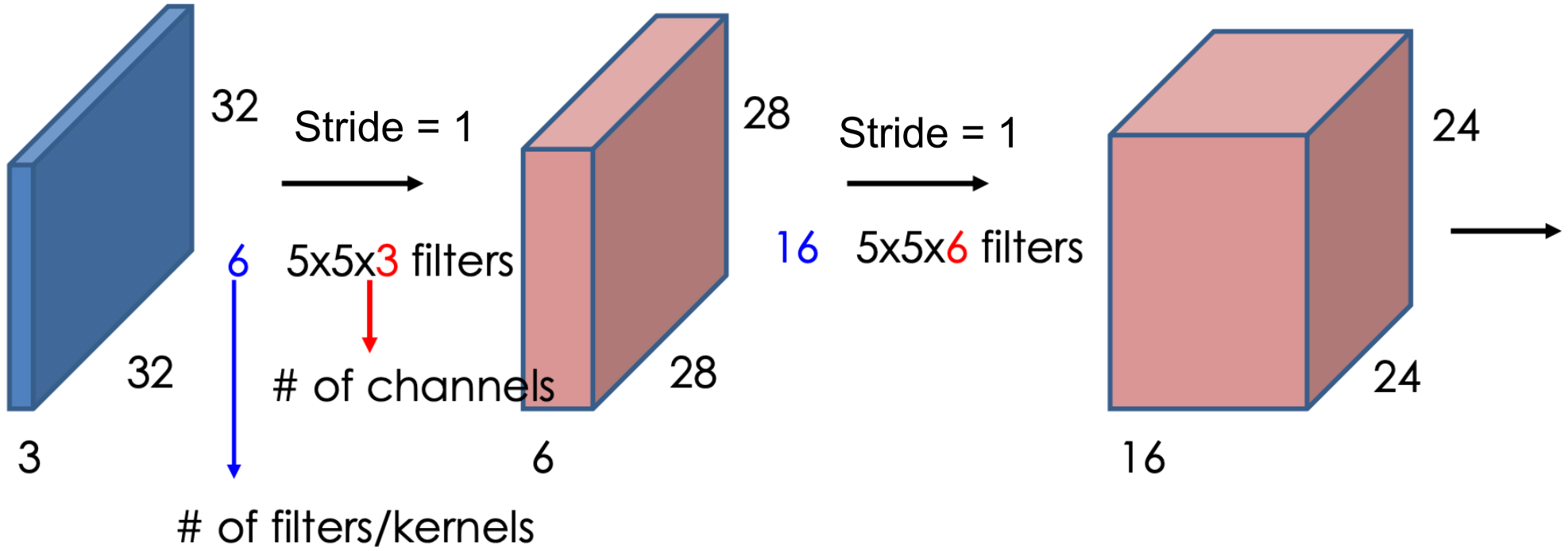


Credit: Hung-yi Lee, Andrew, Stanford course

Convolutional Network: Example

- Convolution network is a sequence of these layers

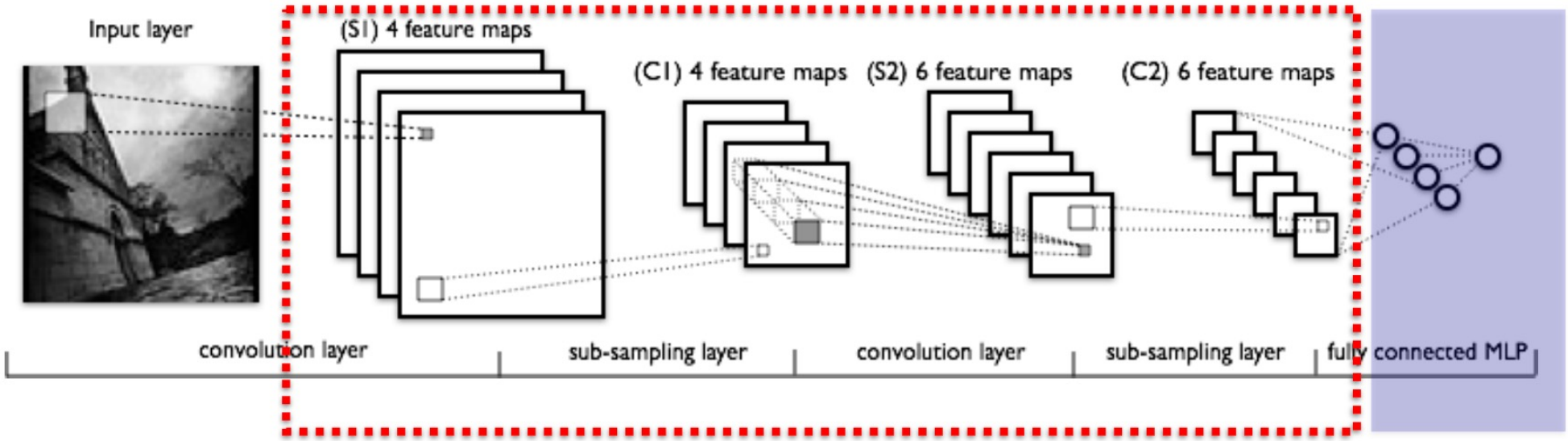
$$\text{Output size} = \frac{\text{Input size} - \text{Filter size}}{\text{Stride}} + 1$$



Credit: Hung-yi Lee, Andrew, Stanford course

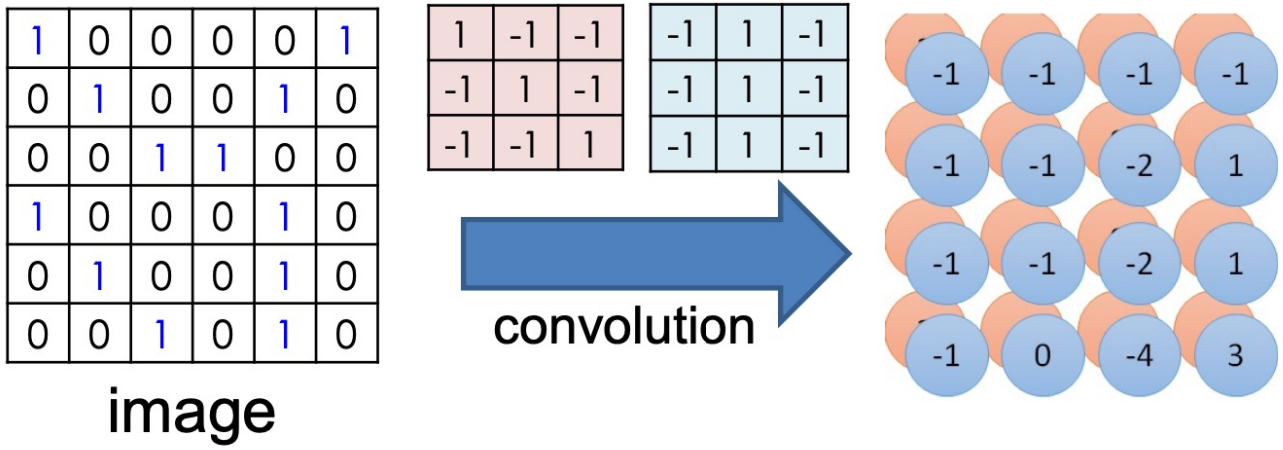
University of Massachusetts Amherst

Convolutional Network

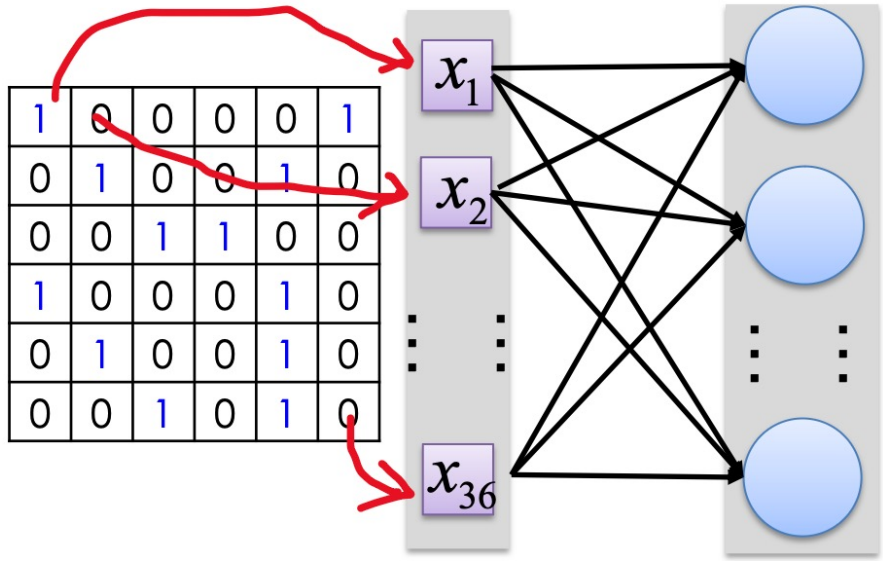


Feature extraction using convolutional layers

Convolution vs. Fully Connected

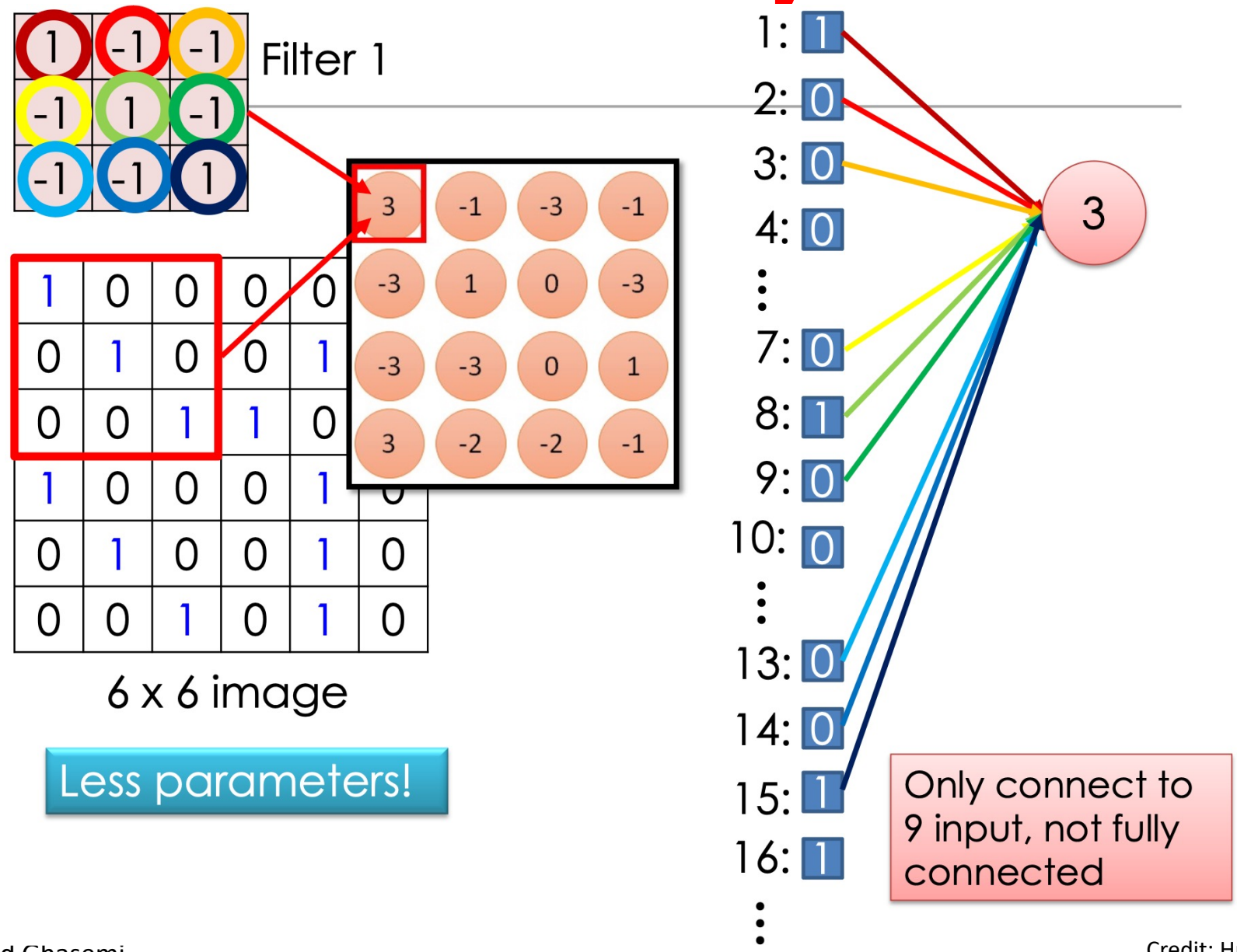


Fully-connected



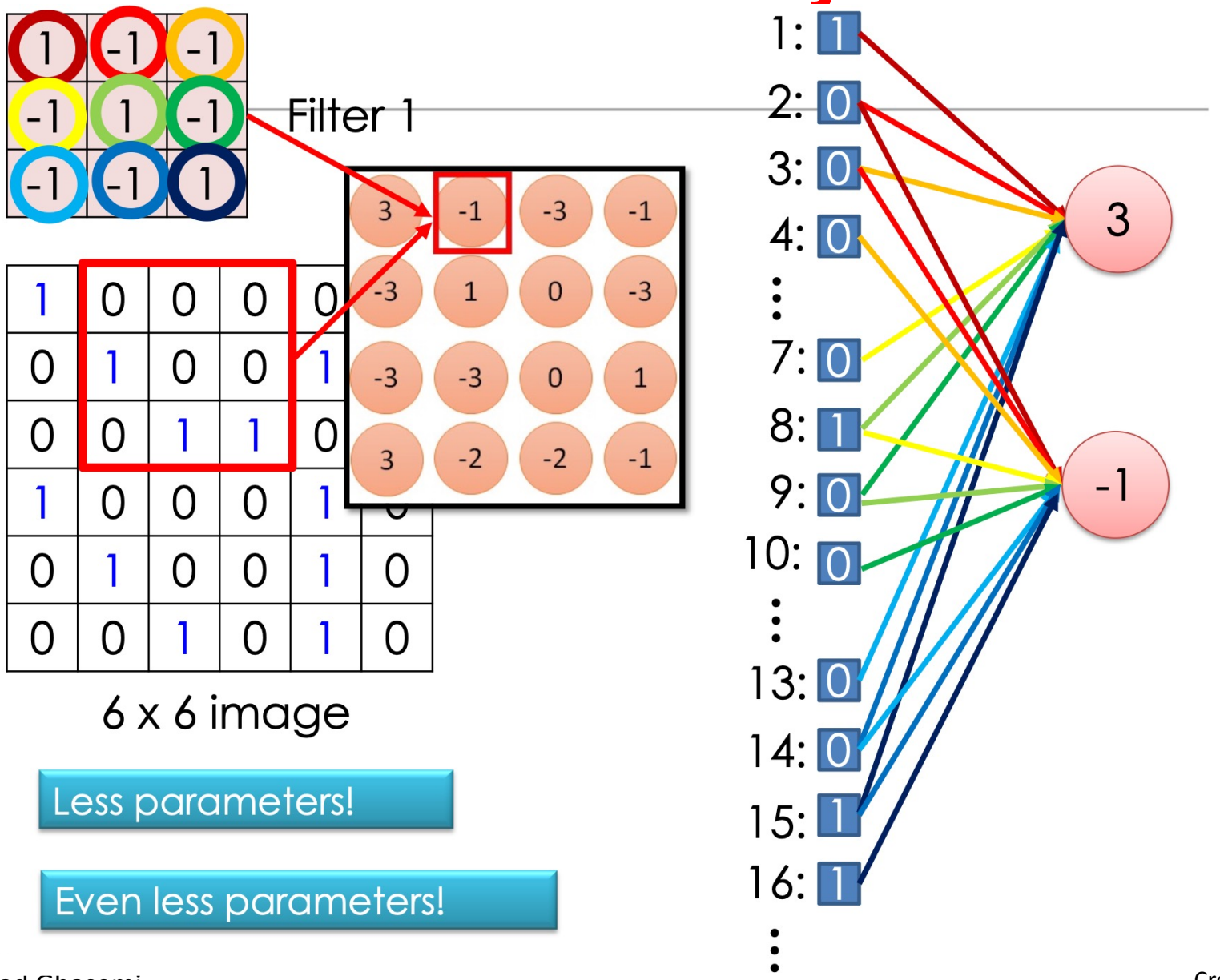
Credit: Hung-yi Lee, Andrew, Stanford course

Convolution vs. Fully Connected



Credit: Hung-yi Lee, Andrew, Stanford course

Convolution vs. Fully Connected



Less parameters!

Even less parameters!

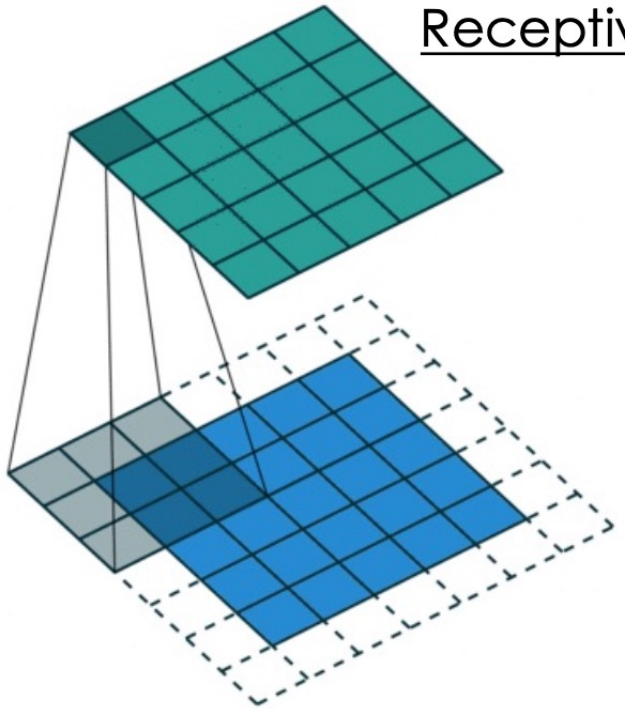
Credit: Hung-yi Lee, Andrew, Stanford course

Variants of Convolution Operation

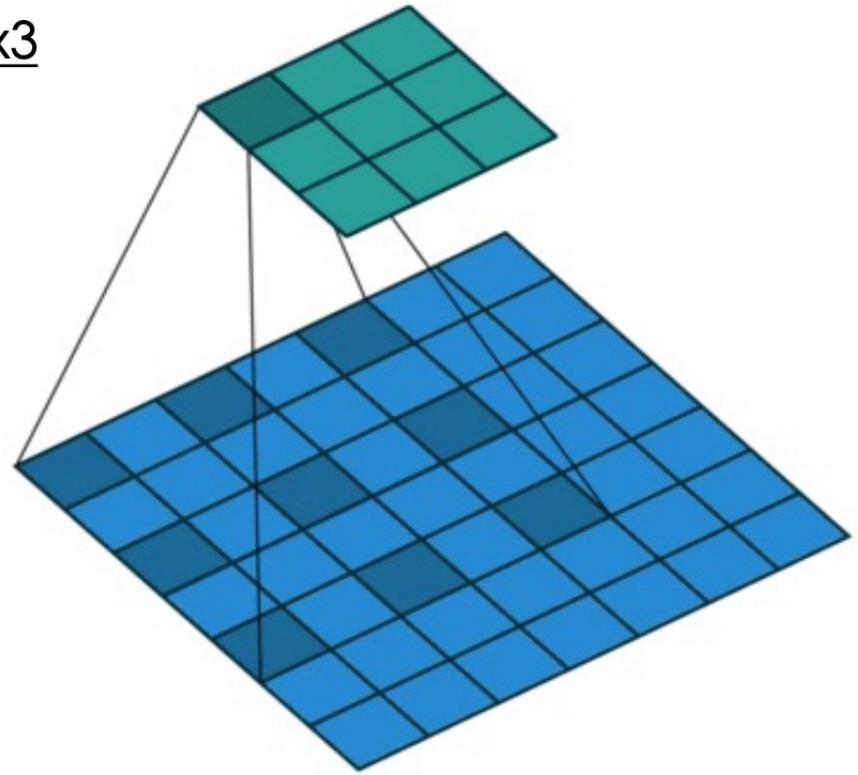
- Dilated/Atrous Convolution

Dilation rate = 2
Receptive filed = 5x5
With only 9 parameters

Receptive filed = 3x3



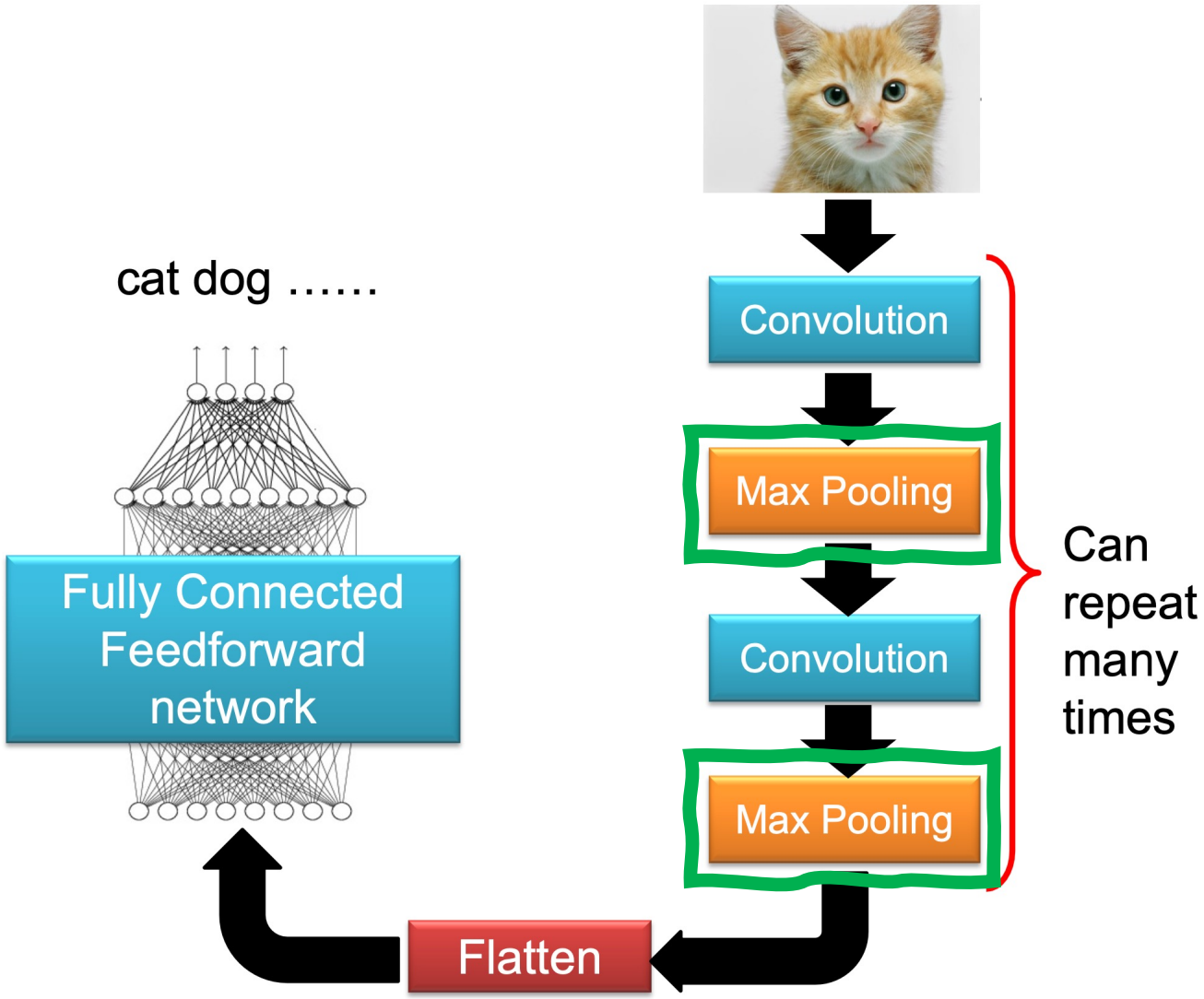
Normal convolution



Effective in segmentation task

Credit: Hung-yi Lee, Andrew, Stanford course

The Whole CNN



Credit: Hung-yi Lee, Andrew, Stanford course

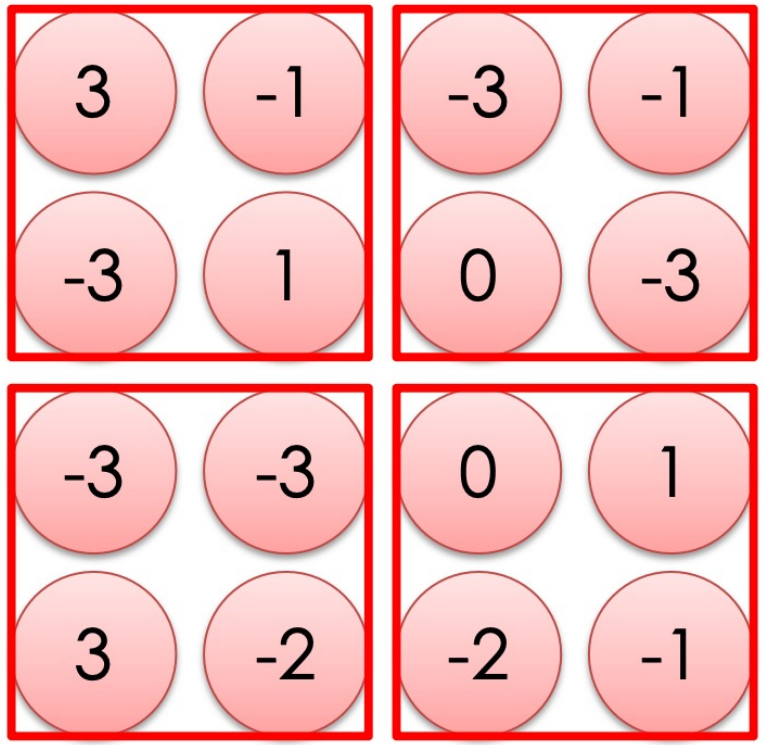
CNN: Max Pooling

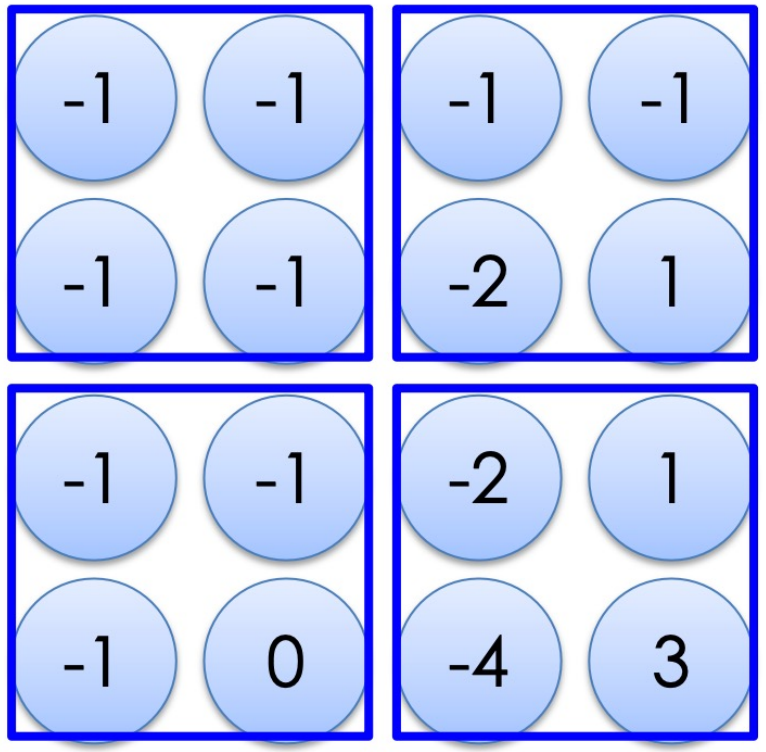
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Filter 2





Credit: Hung-yi Lee, Andrew, Stanford course

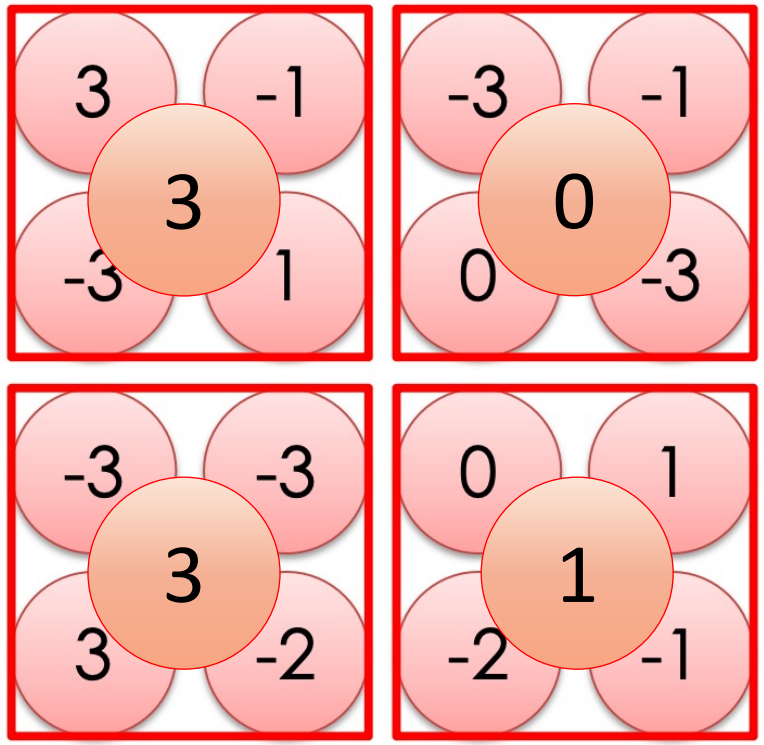
CNN: Max Pooling

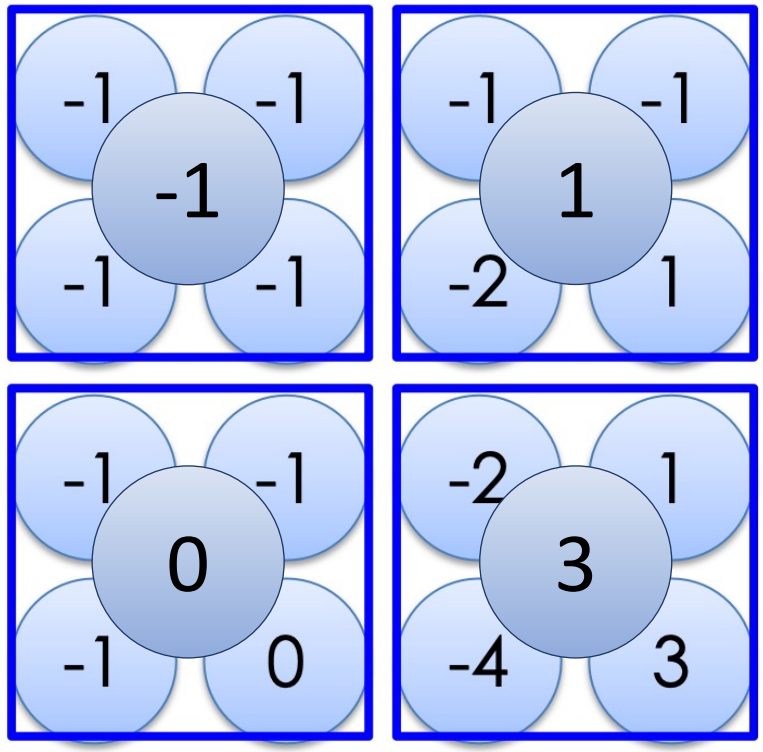
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

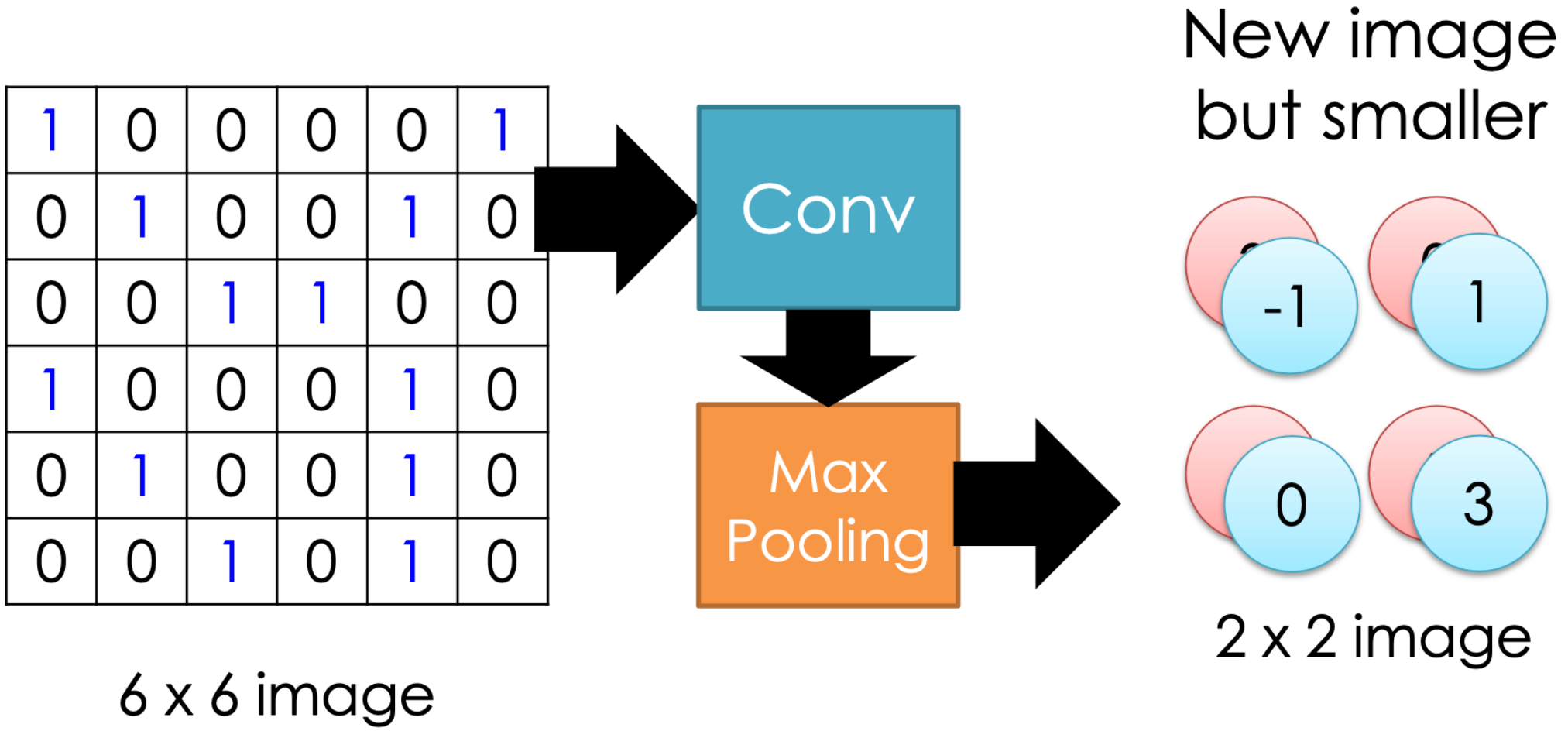
Filter 2





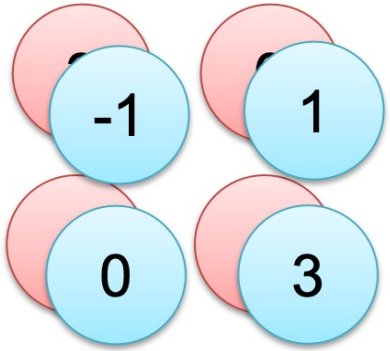
Credit: Hung-yi Lee, Andrew, Stanford course

CNN: Max Pooling



Credit: Hung-yi Lee, Andrew, Stanford course

The Whole CNN



Convolution

Max Pooling

A new image

Convolution

Max Pooling

Can repeat many times

Smaller than the original image

The number of the channel is the number of filters

Credit: Hung-yi Lee, Andrew, Stanford course

Why Pooling?

- It progressively reduce the spatial size of the representation

  ➔ reduce the number of parameters and computation in the network.

Why Pooling?

- It progressively reduce the spatial size of the representation

  ➔ reduce the number of parameters and computation in the network.

- Another way to reduce feature map size?

  ➔ convolution with stride, e.g., 2.

Why Pooling?

- It progressively reduce the spatial size of the representation

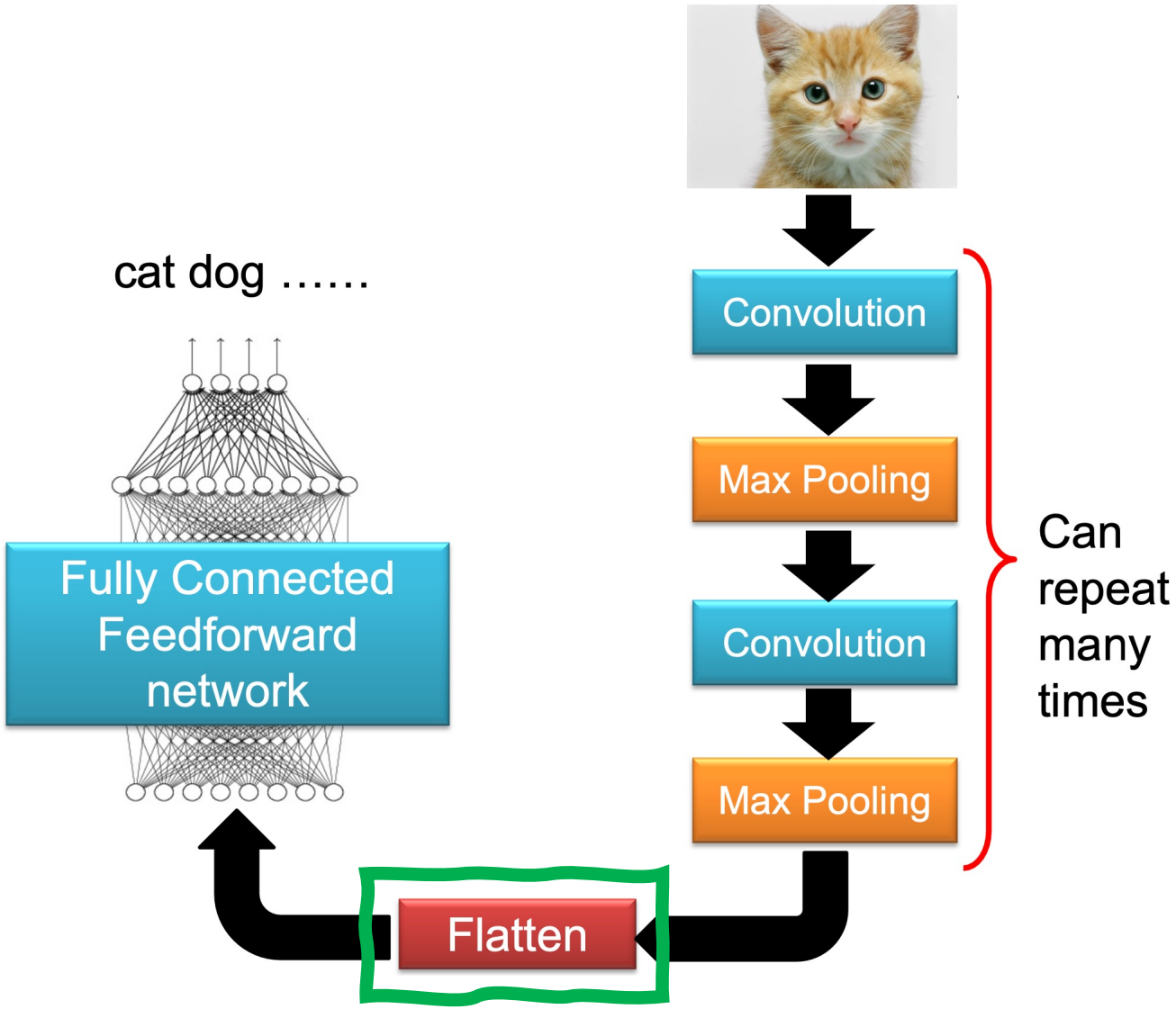
  ➔ reduce the number of parameters and computation in the network.

- Another way to reduce feature map size?

  ➔ convolution with stride, e.g., 2.

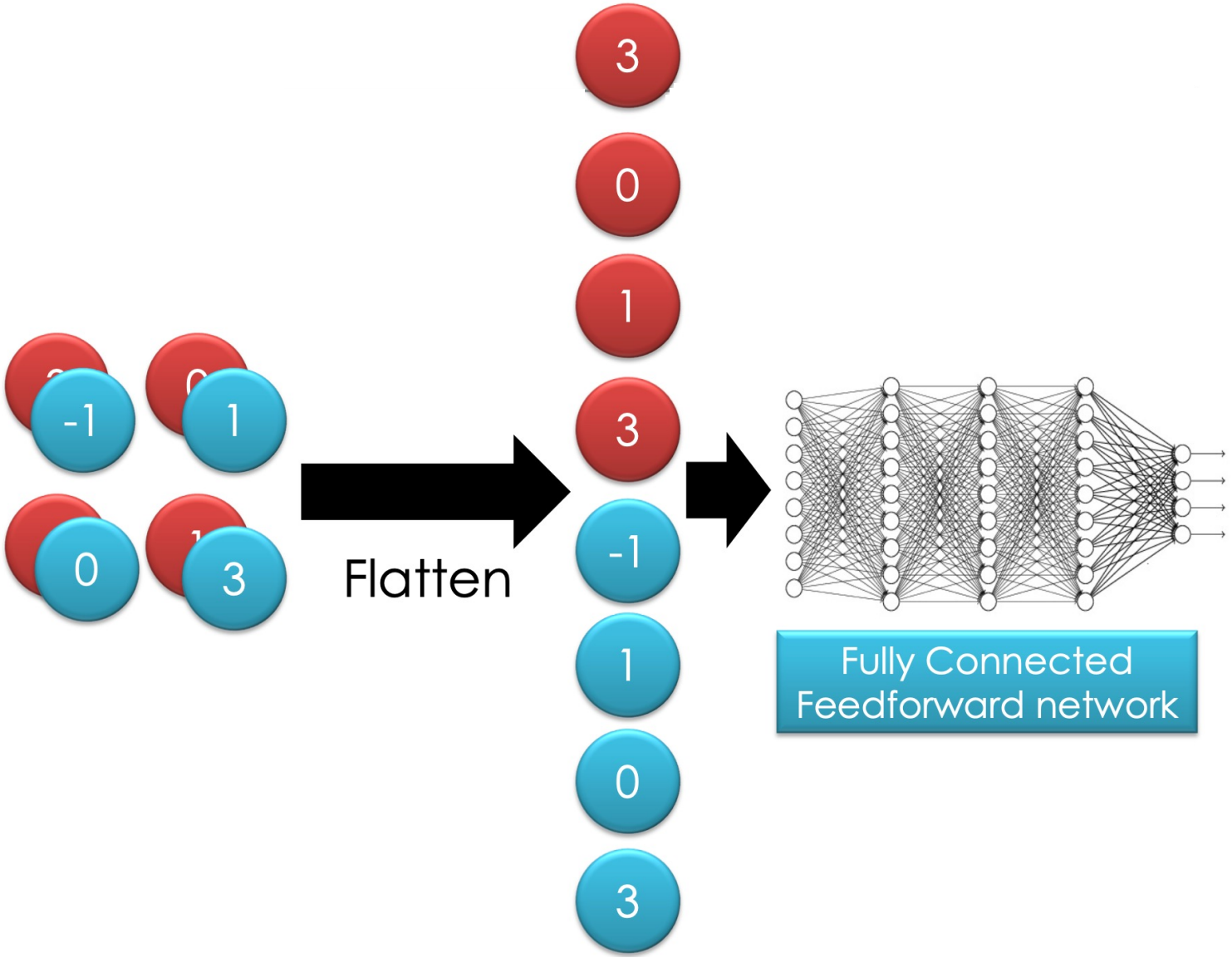
- If instead of taking the maximum, using the average will be the average pooling.

The Whole CNN



Credit: Hung-yi Lee, Andrew, Stanford course

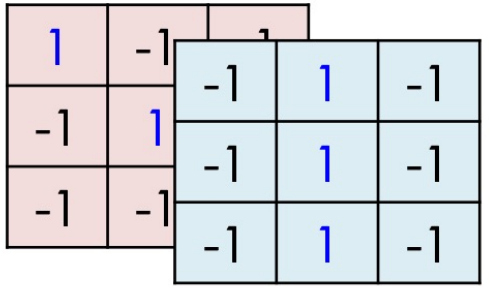
Flatten



Credit: Hung-yi Lee, Andrew, Stanford course

CNN in Keras

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



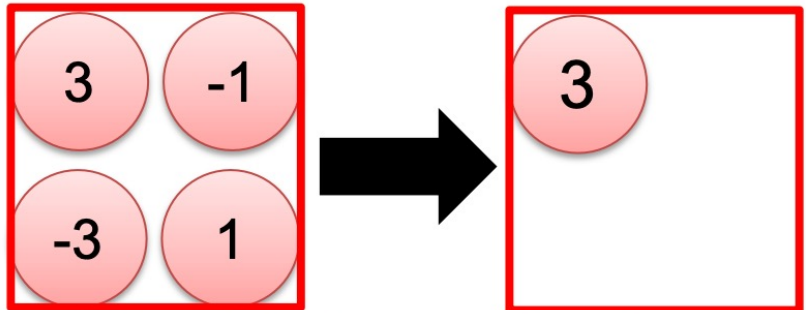
There are 25 3x3 filters.

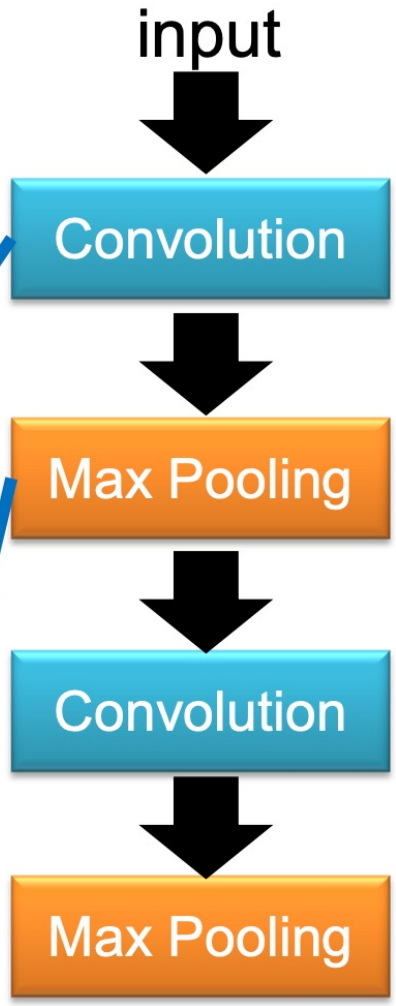
Input_shape = (1 , 28 , 28)

1: grayscale, 3: RGB

28 x 28 pixels

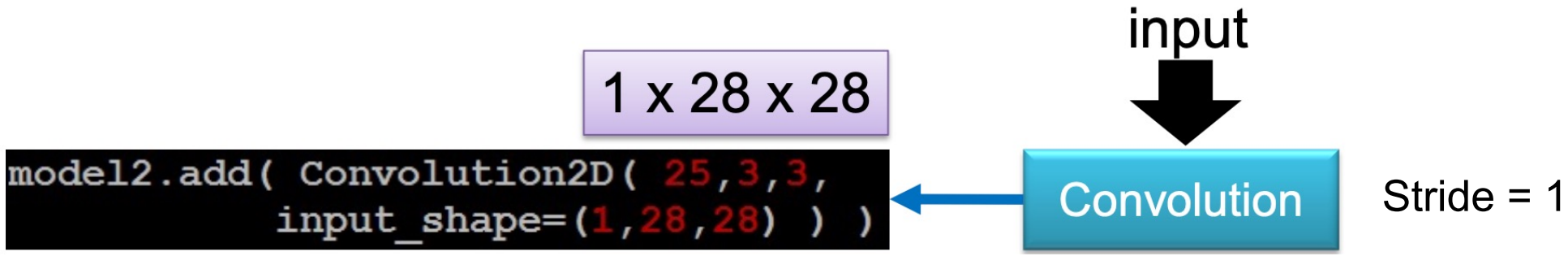
```
model2.add(MaxPooling2D((2,2)))
```





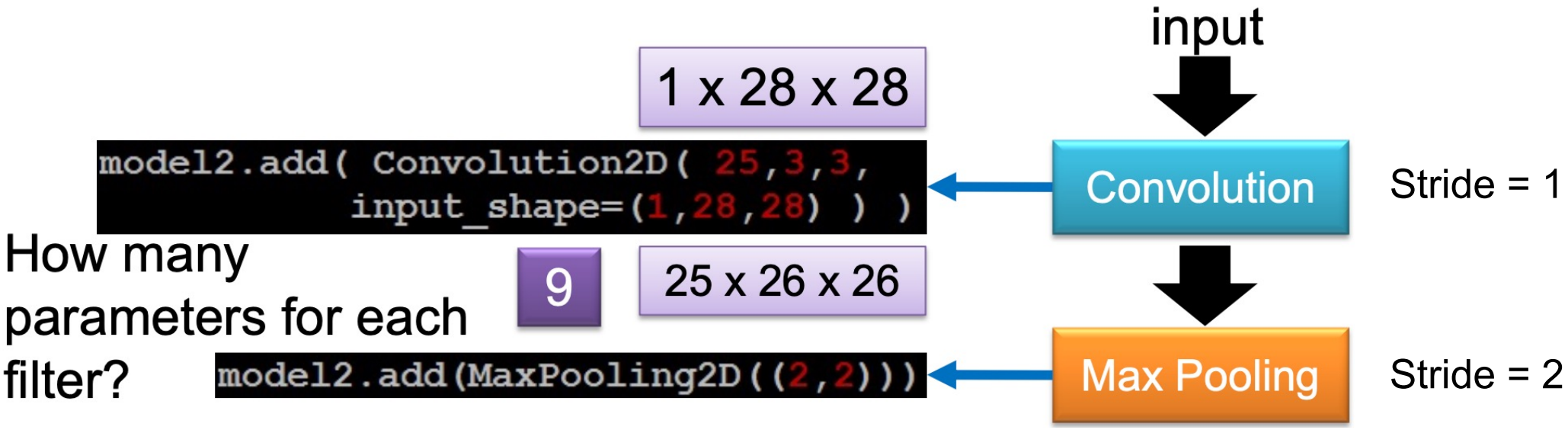
Credit: Hung-yi Lee, Andrew, Stanford course

CNN in Keras



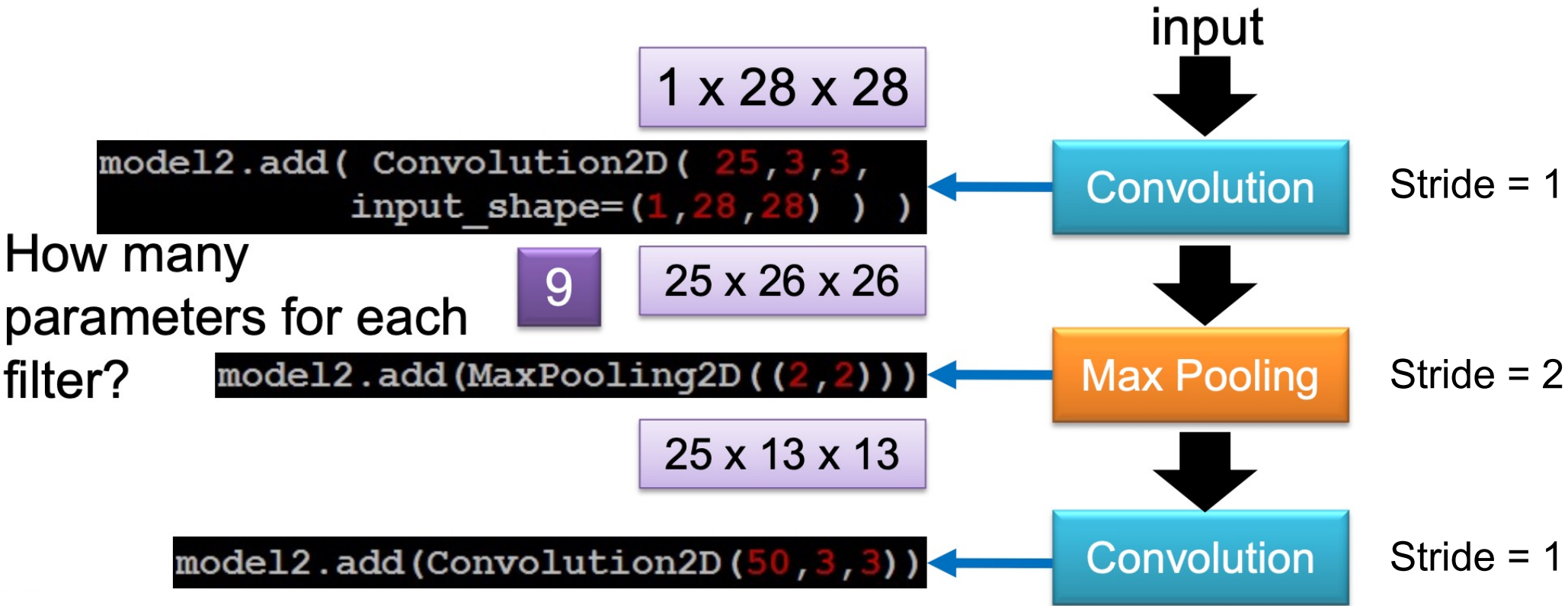
Credit: Hung-yi Lee, Andrew, Stanford course

CNN in Keras



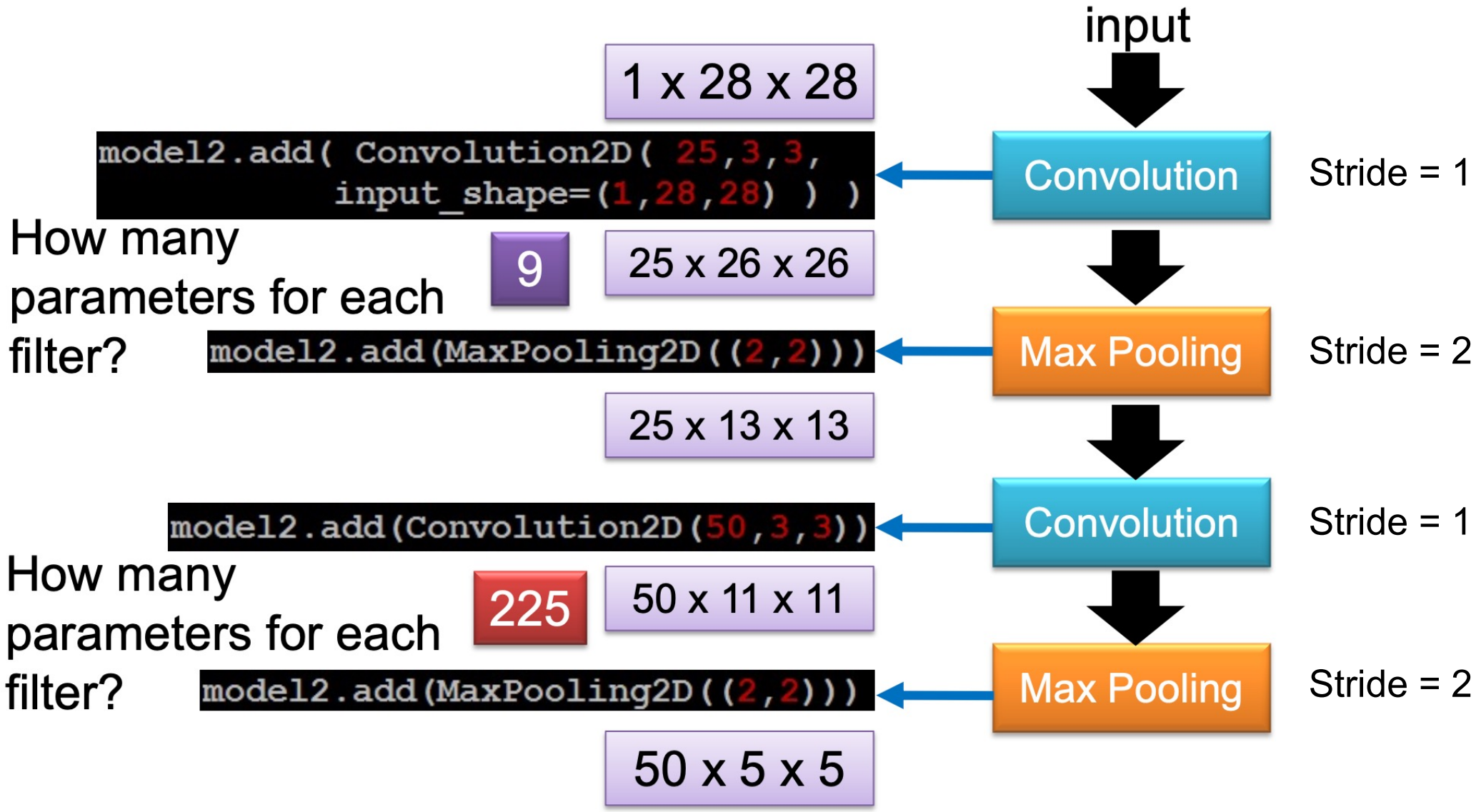
Credit: Hung-yi Lee, Andrew, Stanford course

CNN in Keras



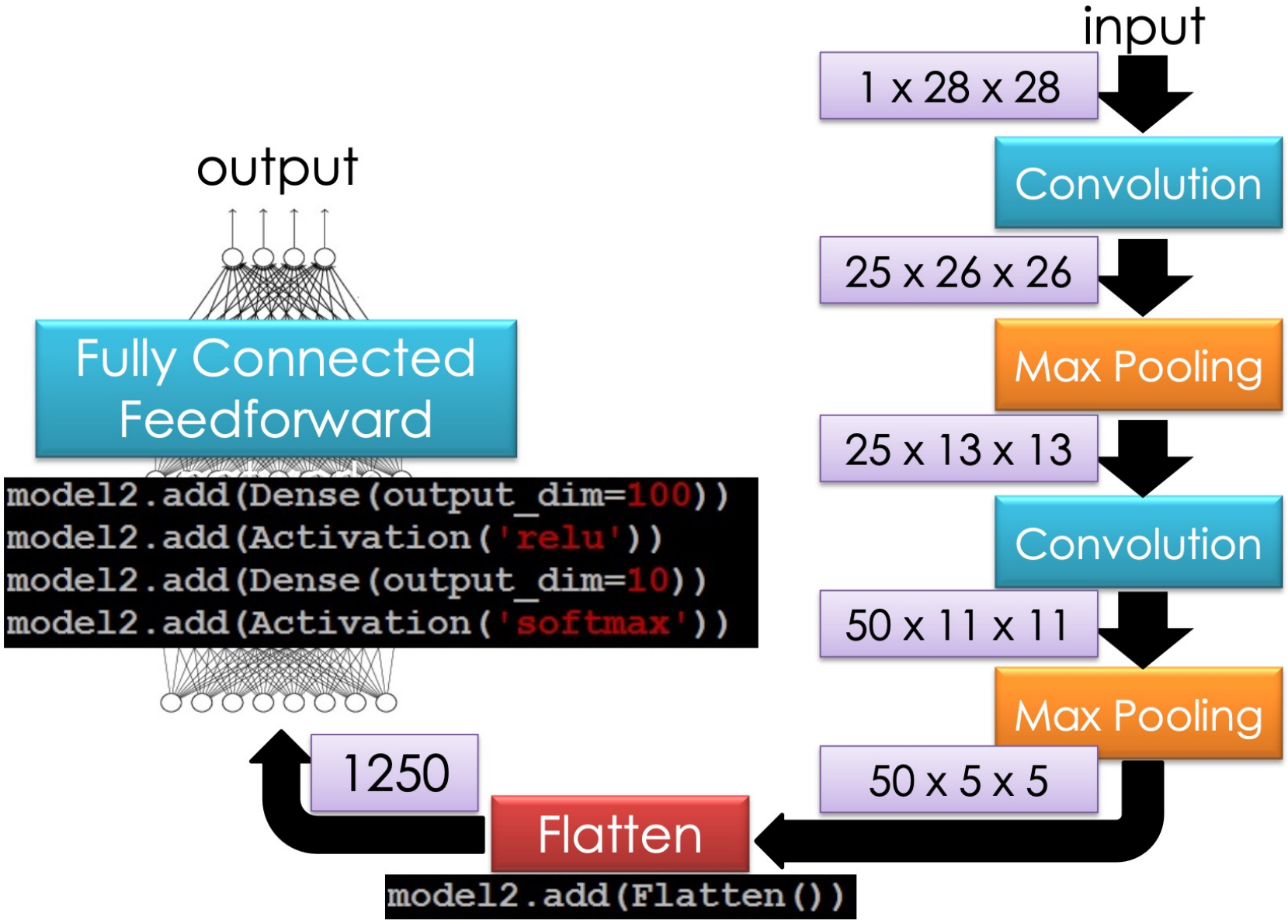
Credit: Hung-yi Lee, Andrew, Stanford course

CNN in Keras



Credit: Hung-yi Lee, Andrew, Stanford course

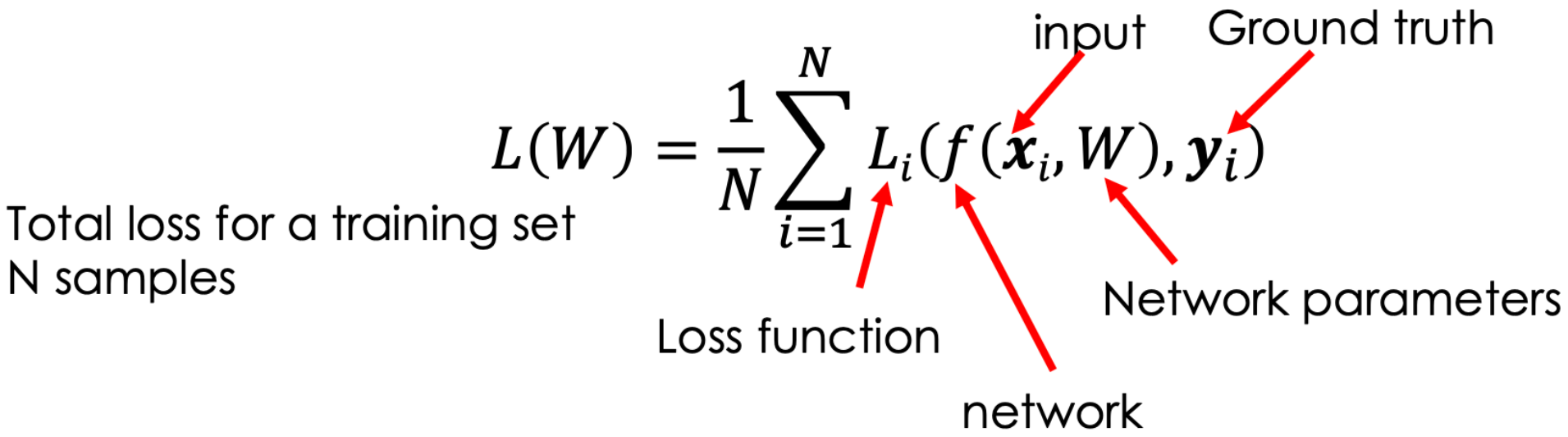
CNN in Keras



Credit: Hung-yi Lee, Andrew, Stanford course

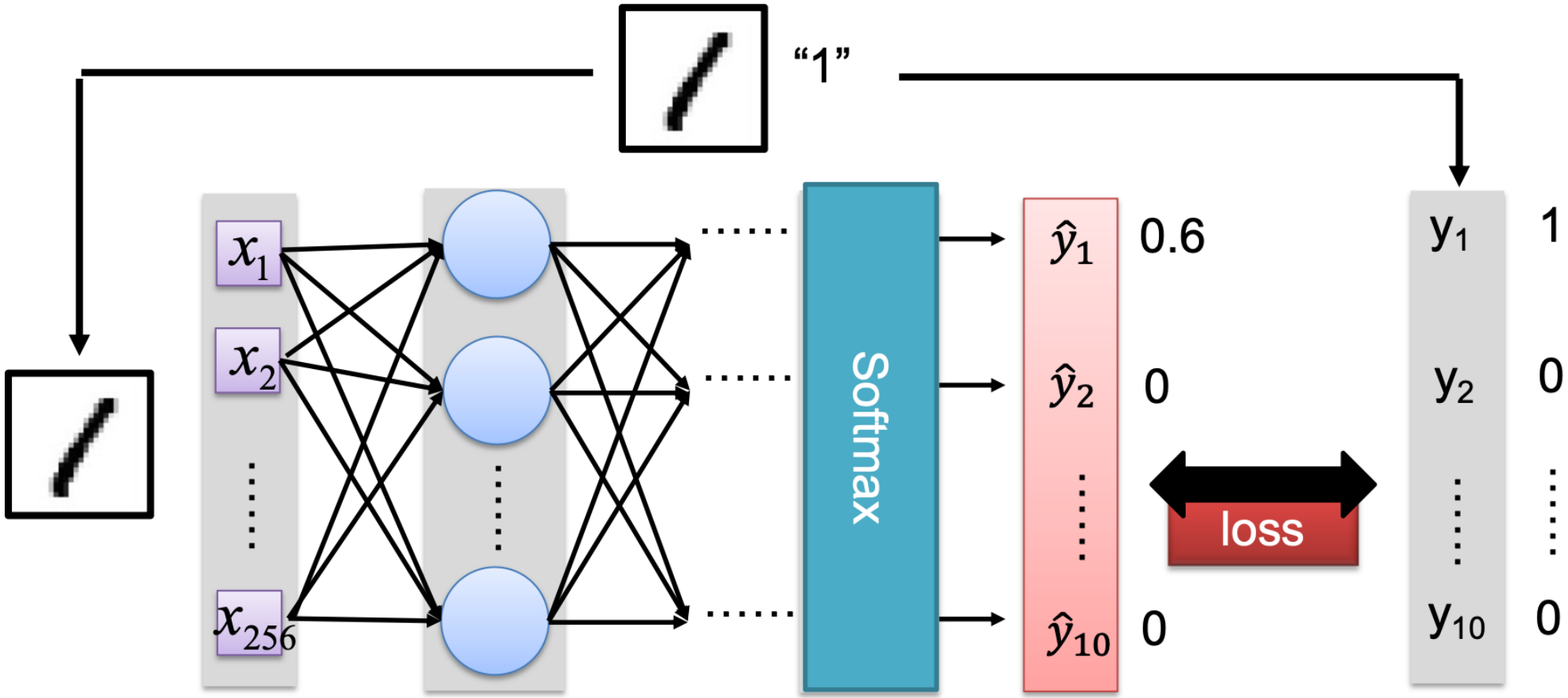
Loss Function

- Way to define how good the network is performing
  - In terms of prediction
- Network training (Optimization)
  - Find the best network parameters to minimize the loss



Credit: Hung-yi Lee, Andrew, Stanford course

Proper Loss Selection



Square Error

$$\sum_{i=1}^{10}(y_i - \hat{y}_i)^2$$

Target (one hot encoding)

Credit: Hung-yi Lee, Andrew, Stanford course

Cross-entropy Loss

- Binary case (we know this from logistic regression)

$$-(ylog(\hat{y}) + (1-y)lo\,g(1-\hat{y}))$$

- General form

K classes

$$-\sum_{i=1}^{K} y_i log(\hat{y}_i)$$

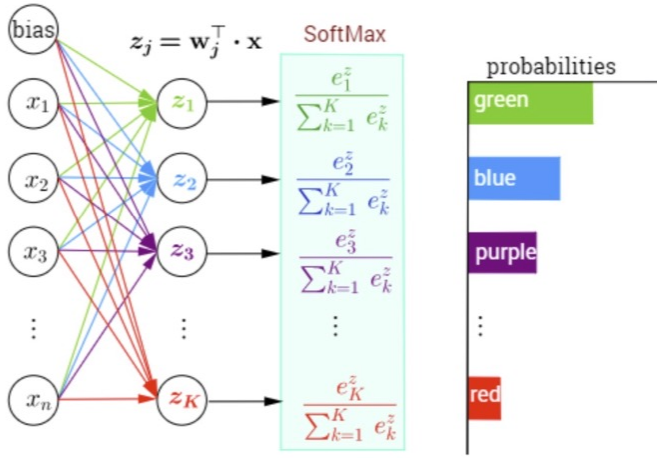
Label $\boldsymbol{y}$ is a one-hot vector

$y_i \in [0,1]$

Cross-entropy Loss

- Multi-class classification

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



Cross-Entropy loss for one sample

$$-\sum_{i=1}^{K} y_i log(\hat{y}_i)$$

Label $\boldsymbol{y}$ is a one-hot vector

$(y_i) \in [0,1]$

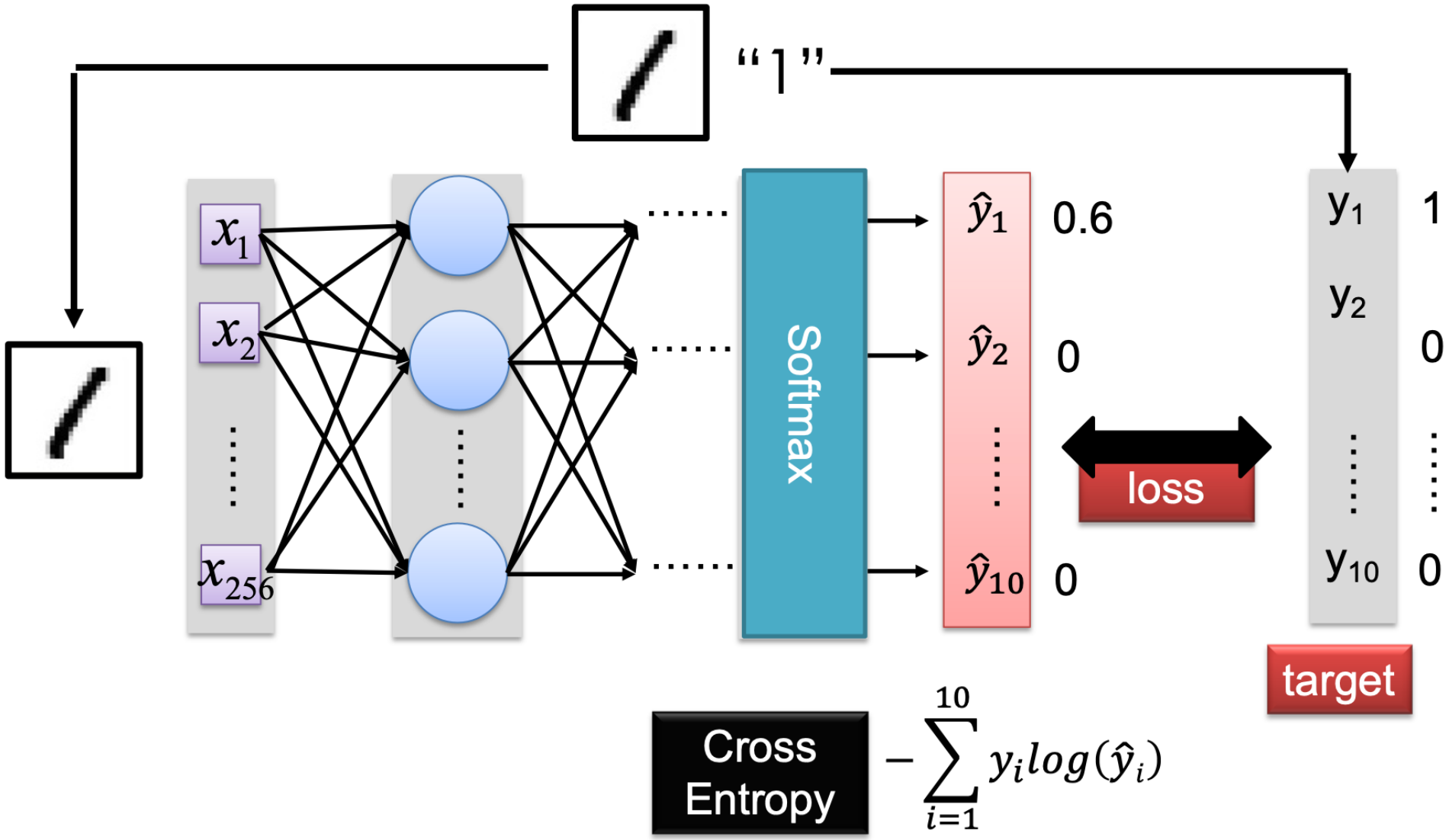
Softmax

$$\sigma(j) = \frac{\exp(\mathbf{w}_j^\top \mathbf{x})}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^\top \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)}$$

computed	targets	correct?
0.3 0.3 0.4	0 0 1 (democrat)	yes
0.3 0.4 0.3	0 1 0 (republican)	yes
0.1 0.2 0.7	1 0 0 (other)	no

$$-((\log(0.3) \times 0) + (\log(0.3) \times 0) + (\log(0.4) \times 1)) = -\log(0.4)$$

Cross-entropy Loss



Credit: Hung-yi Lee, Andrew, Stanford course

Proper Loss Selection

Square Error

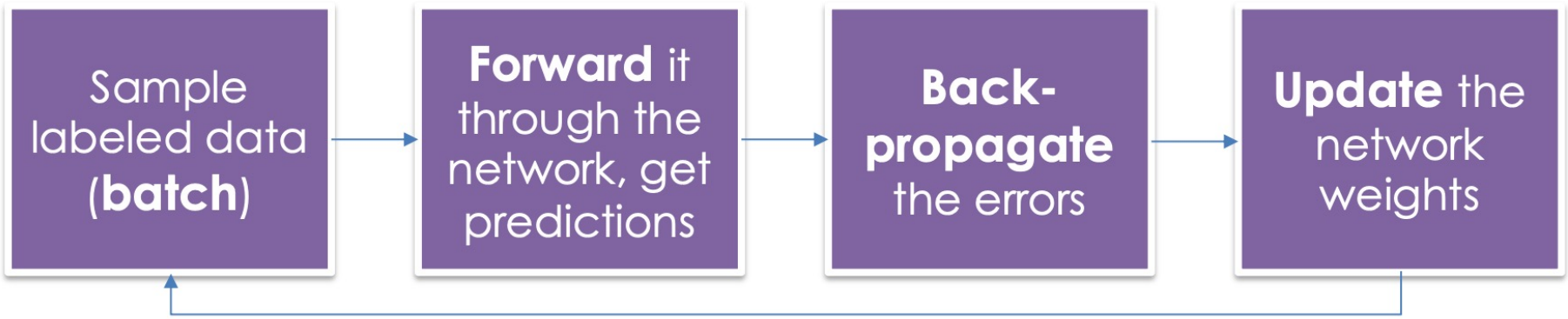
```
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

Cross Entropy

```
model.compile(loss='categorical_crossentropy',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

Several alternatives: https://keras.io/objectives/

Training



Optimize (min. or max.) **objective/cost function** $J(\theta)$

Generate **error signal** that measures difference between predictions and target values

Use error signal to change the **weights** and get more accurate predictions

Subtracting a fraction of the **gradient** moves you towards the **(local) minimum of the cost function**

https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descend-optimization-algorithm-4106a6702d39

University of Massachusetts Amherst

Gradient Descent

Optimize (min. or max.) **objective/cost function $J(\theta)$**

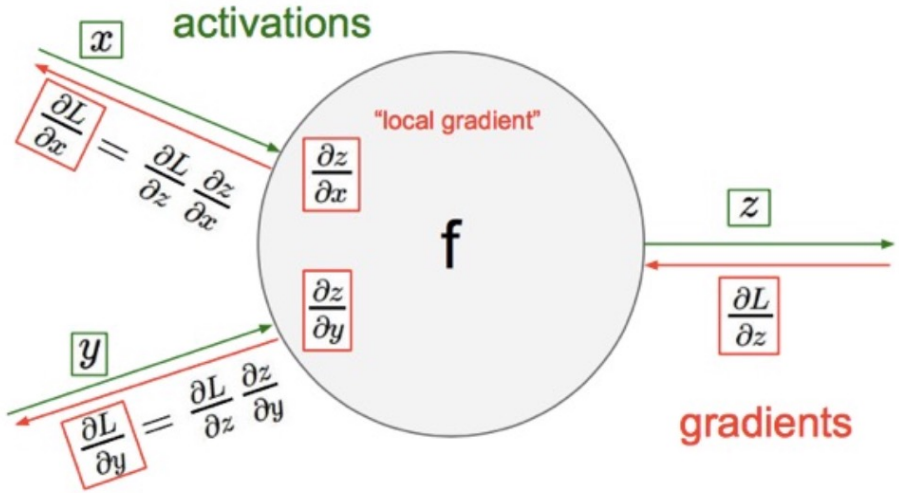
Use error signal to change the **weights** and get more accurate predictions

Subtracting a fraction of the **gradient** moves you towards the **(local) minimum of the cost function**

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} - \alpha \frac{d}{d\theta_j^{\text{old}}} J(\theta)$$ Update each element of θ

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} - \alpha \nabla_{\boldsymbol{\theta}} J(\theta)$$ Matrix notation for all parameters

learning rate



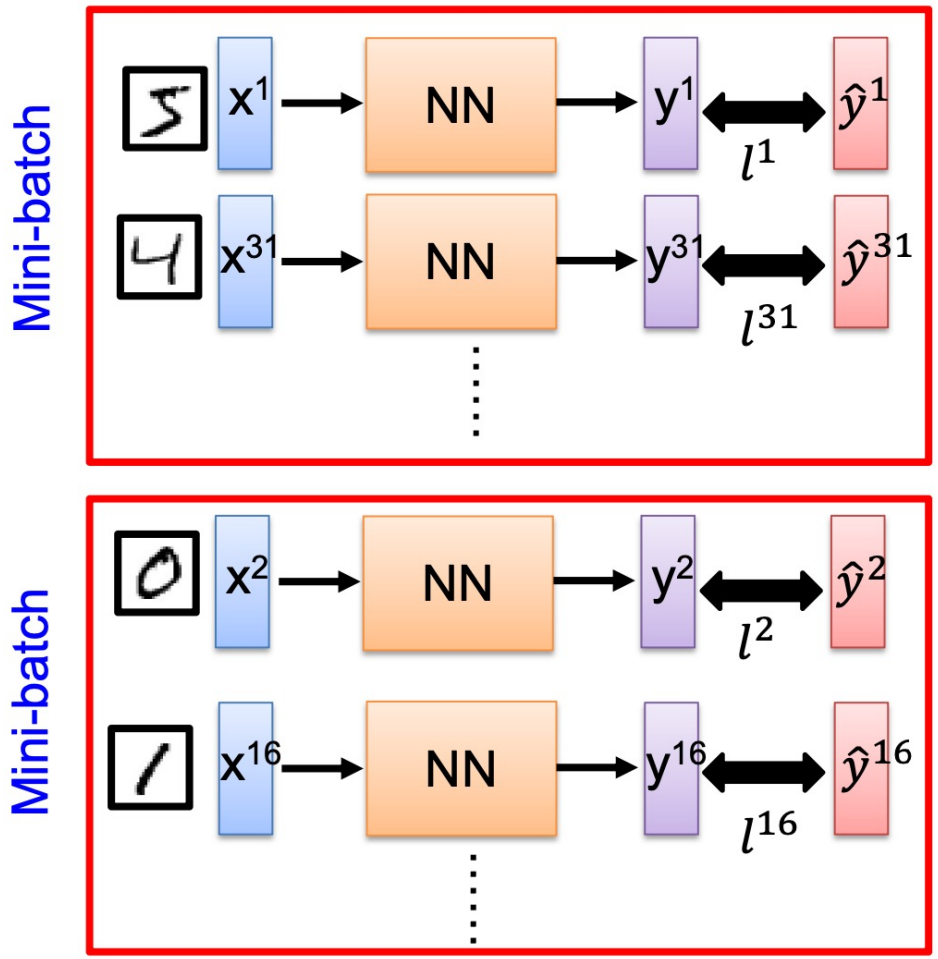
Recursively apply **chain rule** though each node

https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descend-optimization-algorithm-4106a6702d39

Stochastic Gradient Descent

- Gradient over entire dataset is impractical
- Better to take quick, noisy steps
- Estimate gradient over a mini-batch of examples

Mini Batch



- Randomly initialize network parameters
- Pick the 1st batch

  $L' = l^1 + l^{31} + \cdots$

  Update parameters once
- Pick the 2nd batch

  $L'' = l^2 + l^{16} + \cdots$

  Update parameters once

  ⋮
- Until all mini-batches have been picked

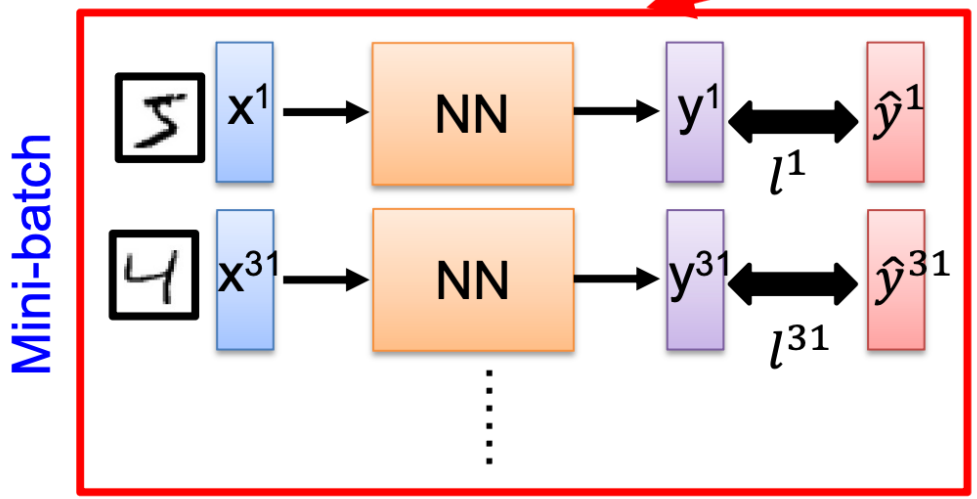
one epoch

Repeat the above process

We do not really minimize total loss!

Mini Batch

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```



100 examples in a mini-batch

Repeat 20 times

- Pick the 1st batch

  $L' = l^1 + l^{31} + \cdots$

  Update parameters once

- Pick the 2nd batch

  $L'' = l^2 + l^{16} + \cdots$

  Update parameters once

  ⋮

- Until all mini-batches have been picked

one epoch

Other Optimizers

Adagrad [John Duchi, JMLR'11]

RMSprop https://www.youtube.com/watch?v=O3sxAc4hxZU

Adadelta [Matthew D. Zeiler, arXiv'12]

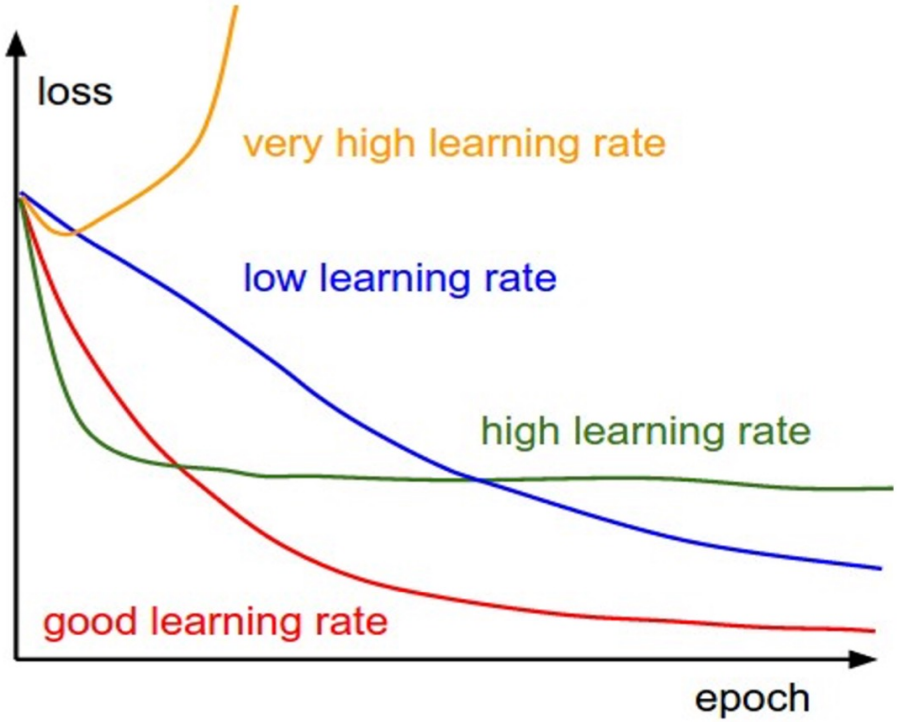
"No more pesky learning rates" [Tom Schaul, arXiv'12]

AdaSecant [Caglar Gulcehre, arXiv'14]

Nadam http://cs229.stanford.edu/proj2015/054_report.pdf

Best Learning Rate?

- Too big = diverge, too small = slow convergence
- No “one learning rate to rule them all”
- Start from a high value and keep cutting by half if model diverges
- Learning rate schedule: decay learning rate over time



http://cs231n.github.io/assets/nn3/learningrates.jpeg

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

Batch Normalization

```
# example of batch normalization for an cnn
from keras.layers import Dense
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import BatchNormalization
...
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D())
model.add(Dense(1))
...
```