

Deep Learning

Convolutional Neural Networks

Han Zhang

2018-10-04

Table of Contents

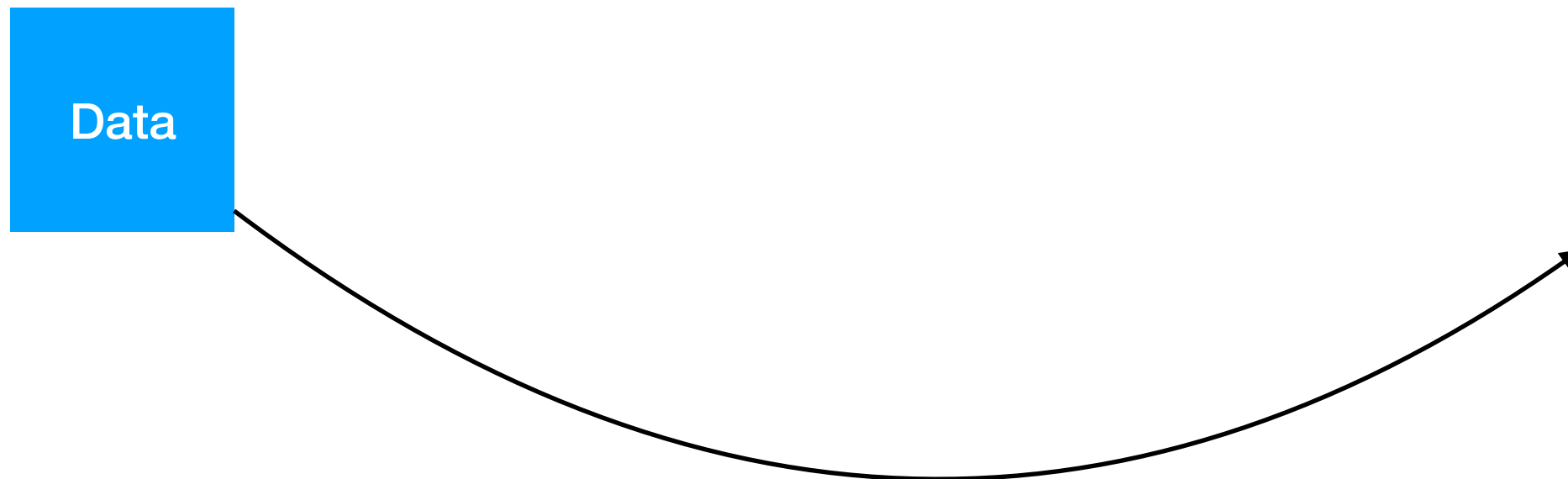
- Representation Learning
- Neural Networks
- Convolutional Neural Networks

Conventional Machine Learning

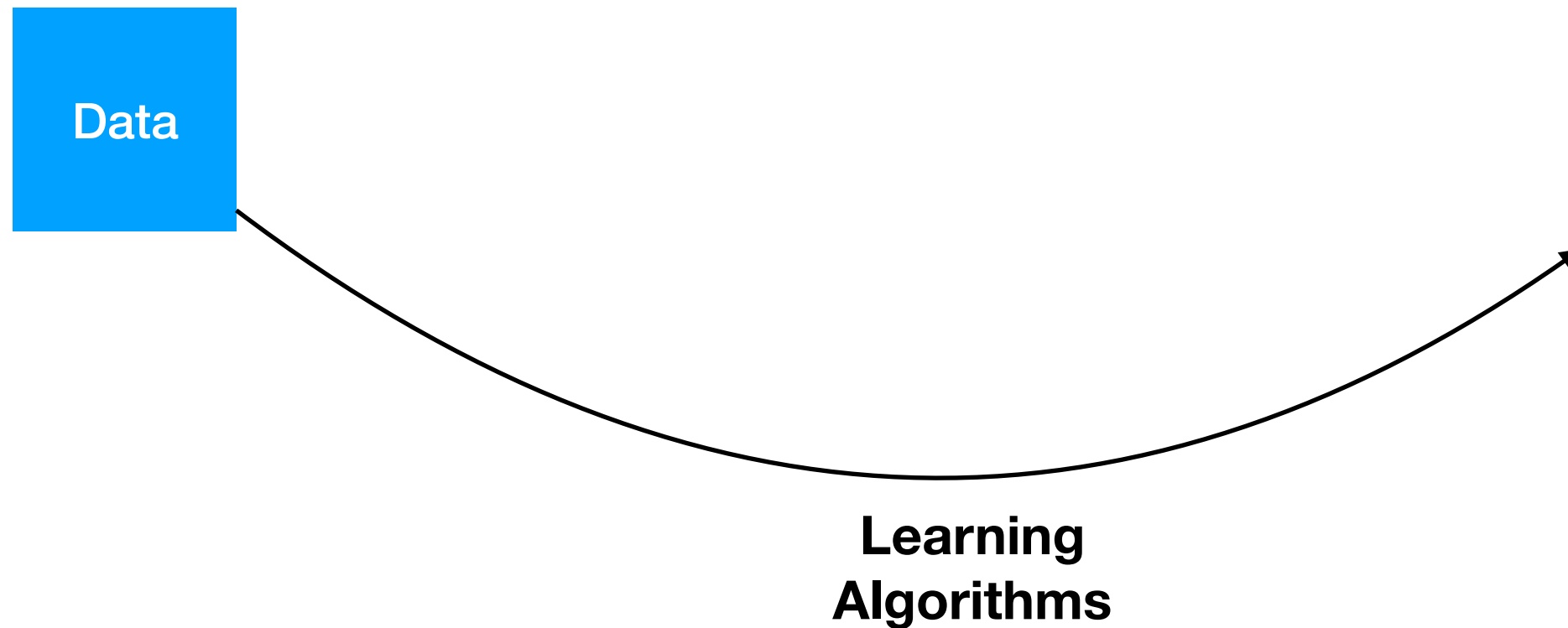


Data

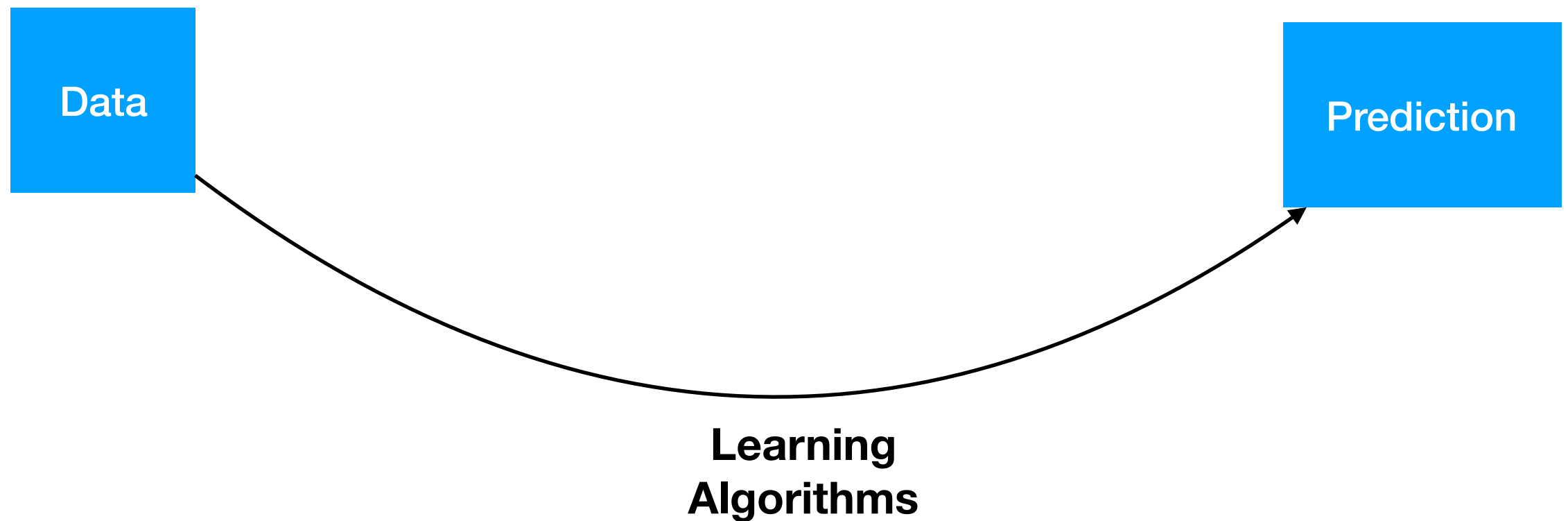
Conventional Machine Learning



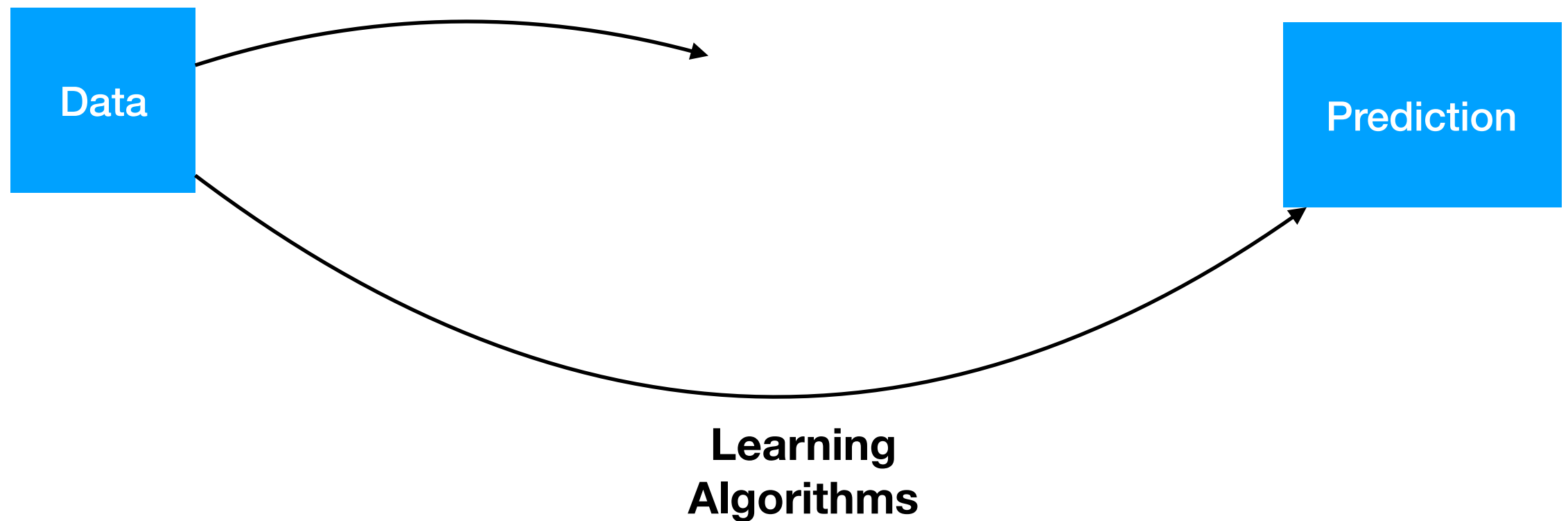
Conventional Machine Learning



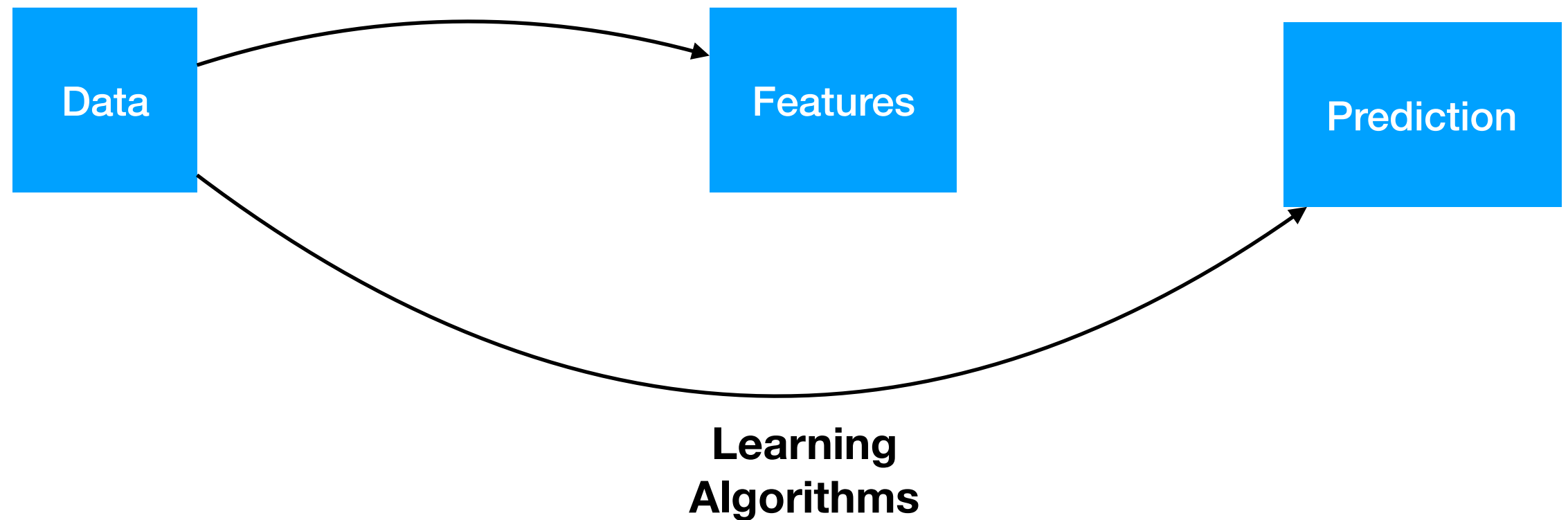
Conventional Machine Learning



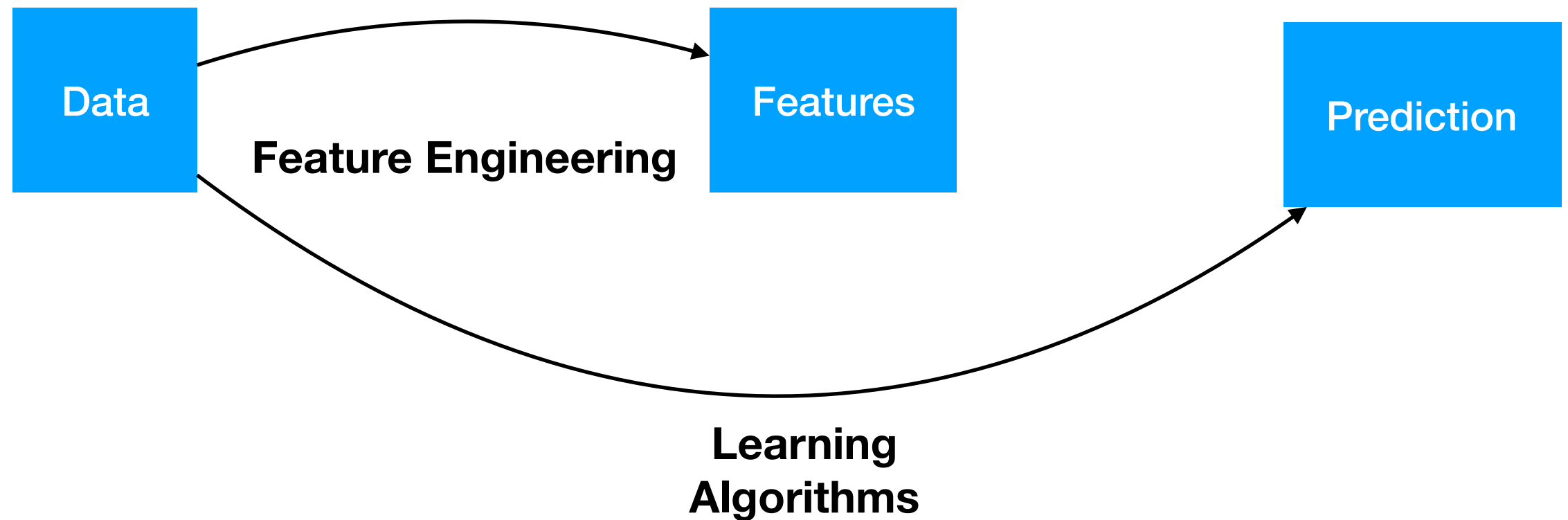
Conventional Machine Learning



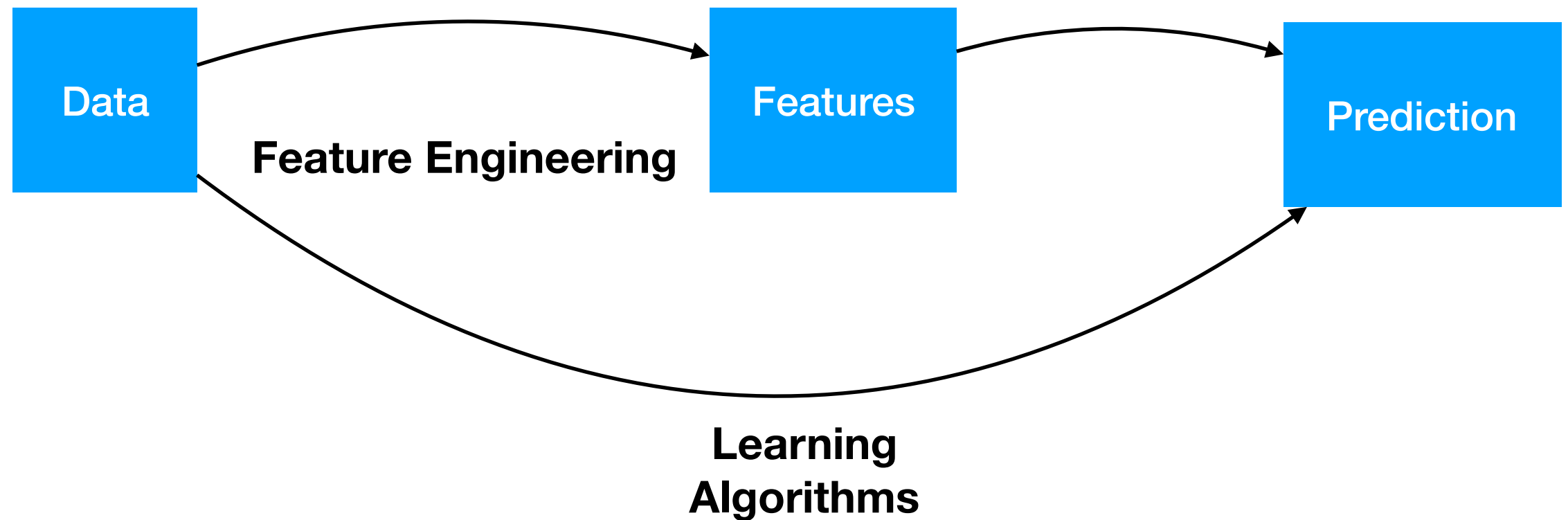
Conventional Machine Learning



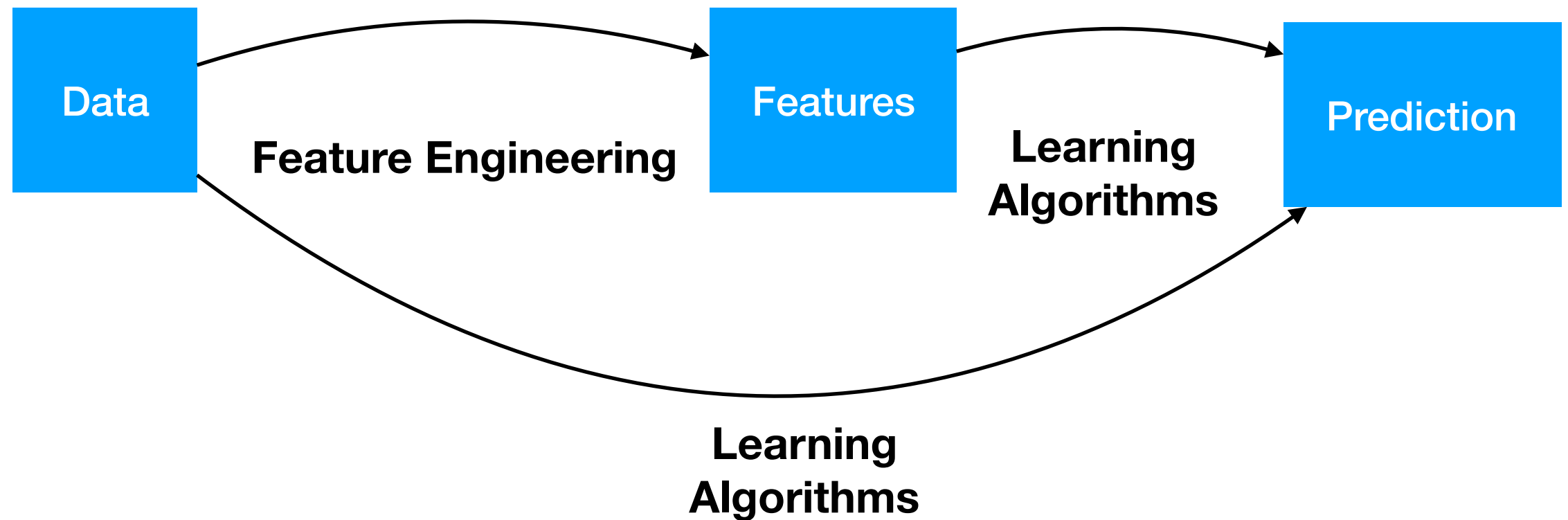
Conventional Machine Learning



Conventional Machine Learning



Conventional Machine Learning



Example: Facial Recognition



Face detection



Prediction

Stalin

Internal Representation

RGB Representation

					165	187	209	58	7
					14	125	233	201	98
253	144	120	251	41	147	204			
67	100	32	241	23	165	30			
209	118	124	27	59	201	79			
210	236	105	169	19	218	156			
35	178	199	197	4	14	218			
115	104	34	111	19	196				
32	69	231	203	74					

Sub-matrix that contains faces

124	27	59
105	169	19
199	197	4

Predicted Label

Stalin

Feature engineering is hard

Lightening



<https://alitarhini.wordpress.com/2010/12/05/face-recognition-an-introduction/>

Feature engineering is hard

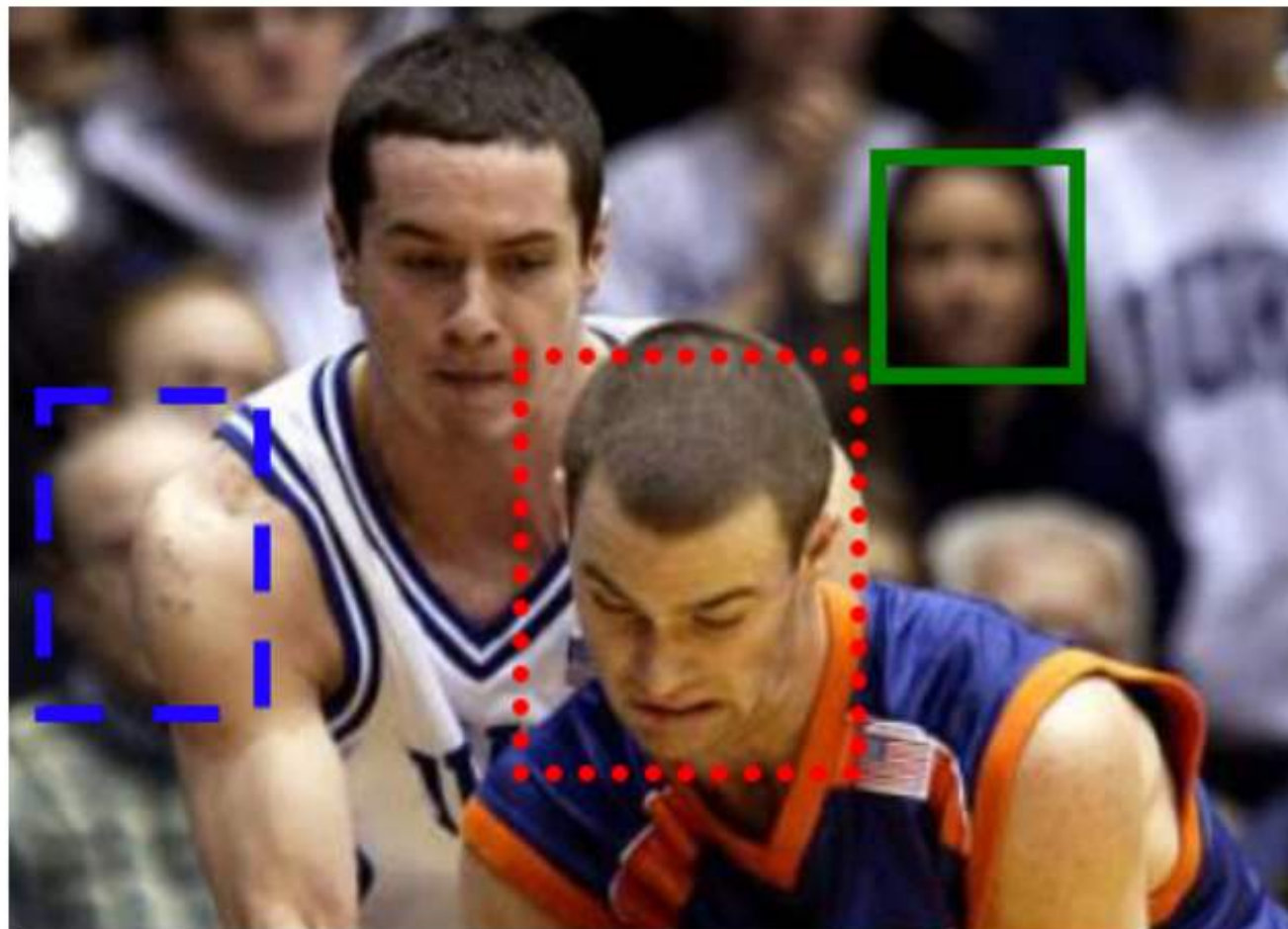
Angle



<https://alitarhini.wordpress.com/2010/12/05/face-recognition-an-introduction/>

Feature engineering is hard

Background



Feature engineering is hard

multiple faces/ no faces



<https://alitarhini.wordpress.com/2010/12/05/face-recognition-an-introduction/>

Feature engineering is hard

- Feature engineering is hard
 - Requires domain-knowledge;
 - Hard to find a **general** feature extractor
- Feature engineering and prediction are two separate steps.

Representation Learning

- Learning representations of the raw data that makes it easier to extract useful information when building prediction models.

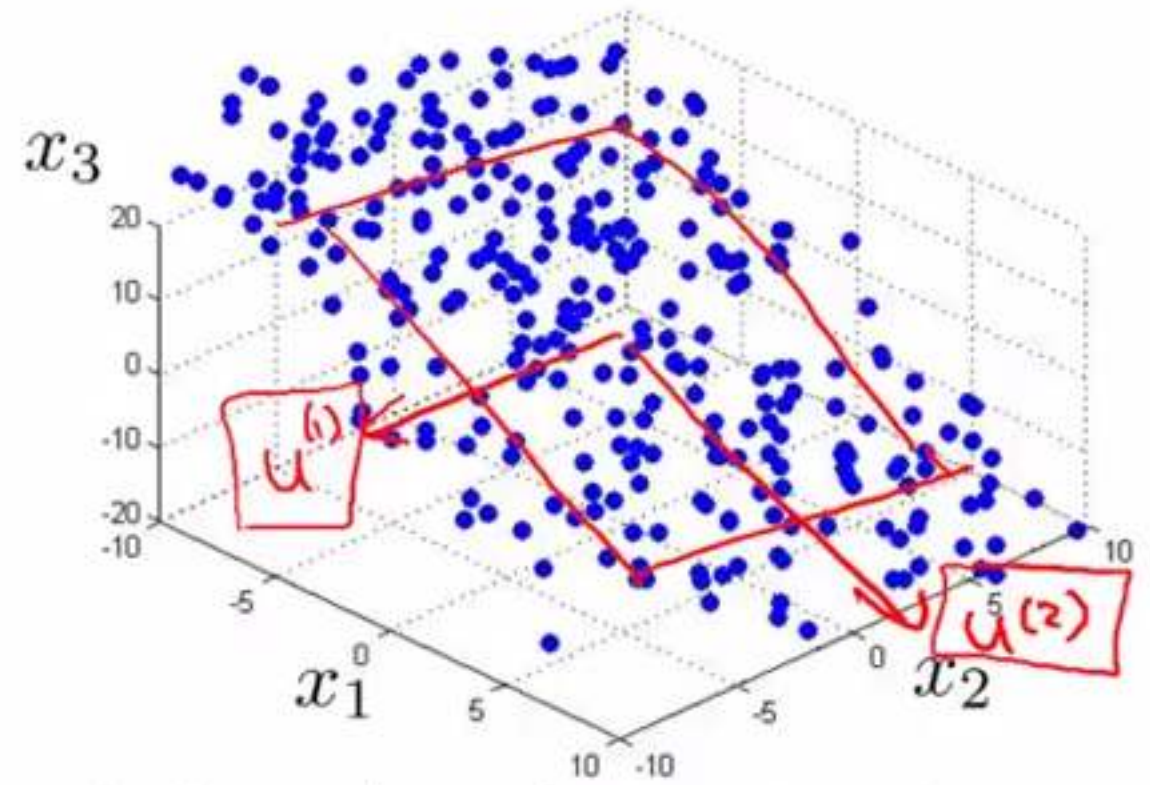
Some existing representation examples

- PCA:
 - reduce the dimension of the raw data.
 - does not work when data does not line on linear manifold

Roweis, Sam T., and
Lawrence K. Saul.
"Nonlinear dimensionality
reduction by locally linear
embedding." *Science*, no.
5500 (2000): 2323-2326.

Some existing representation examples

- PCA:
 - reduce the dimension of the raw data.
 - does not work when data does not line on linear manifold



Roweis, Sam T., and Lawrence K. Saul.
"Nonlinear dimensionality reduction by locally linear embedding." *Science*, no. 5500 (2000): 2323-2326.

Some existing representation examples

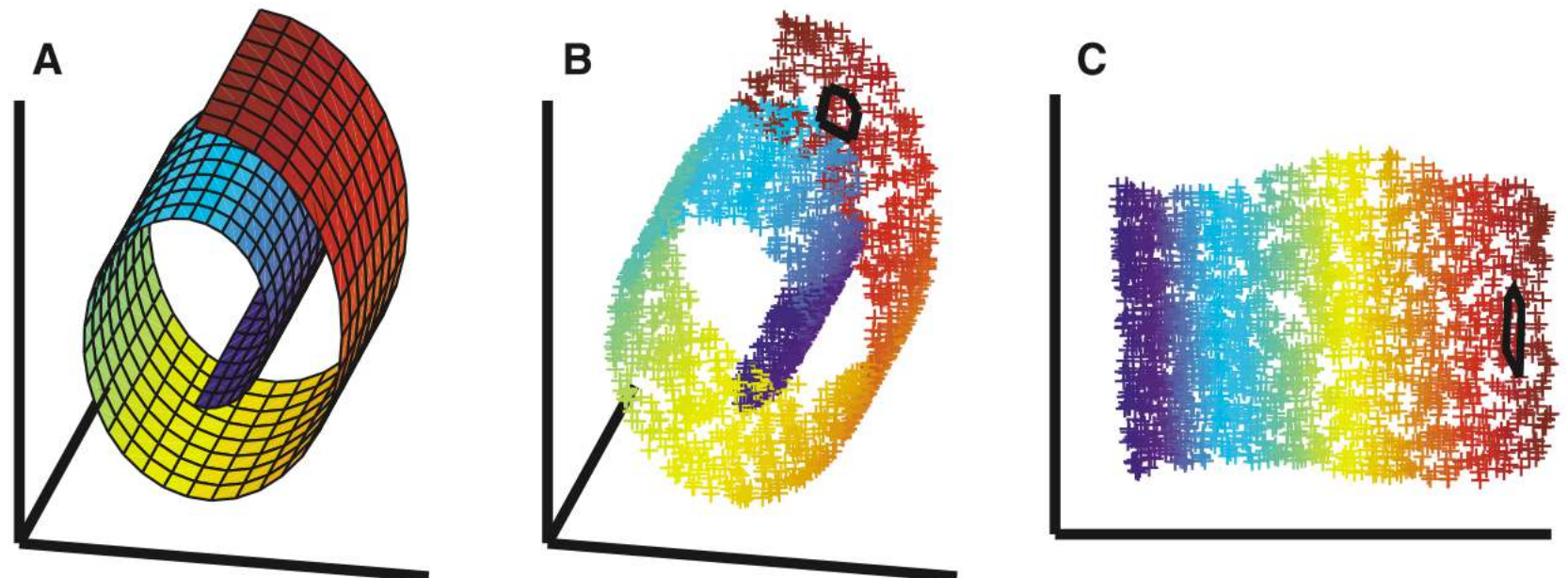
- PCA:
 - reduce the dimension of the raw data.
 - does not work when data does not line on linear manifold

Roweis, Sam T., and
Lawrence K. Saul.
"Nonlinear dimensionality
reduction by locally linear
embedding." *Science*, no.
5500 (2000): 2323-2326.

Some existing representation examples

- PCA:
 - reduce the dimension of the raw data.
 - does not work when data does not line on linear manifold

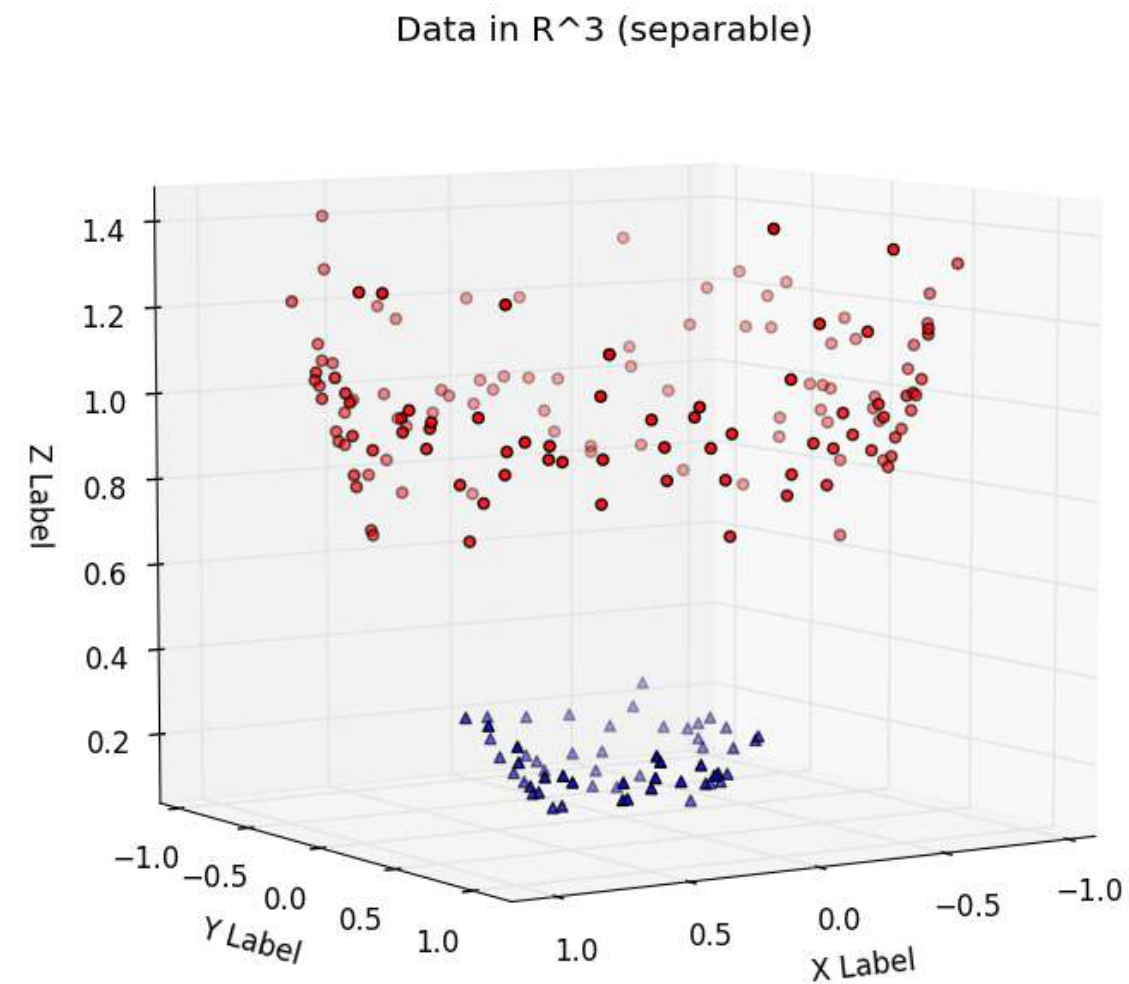
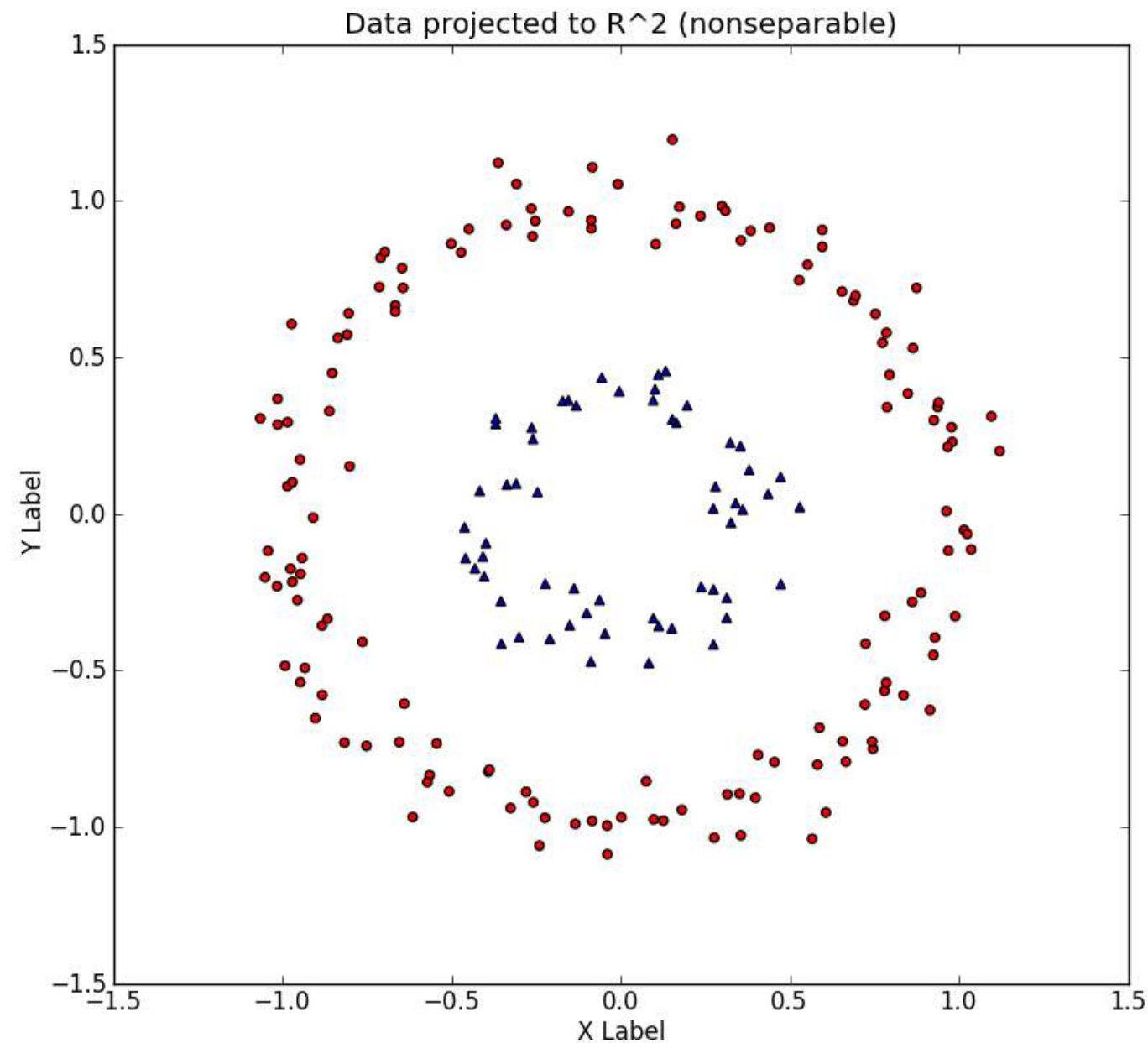
Roweis, Sam T., and Lawrence K. Saul.
"Nonlinear dimensionality reduction by locally linear embedding." *Science*, no. 5500 (2000): 2323-2326.



Some existing representation examples

- Kernel tricks:
 - project the data to higher dimensions to make them linear separable.
 - does not explicitly learn representations.

Some existing representation examples



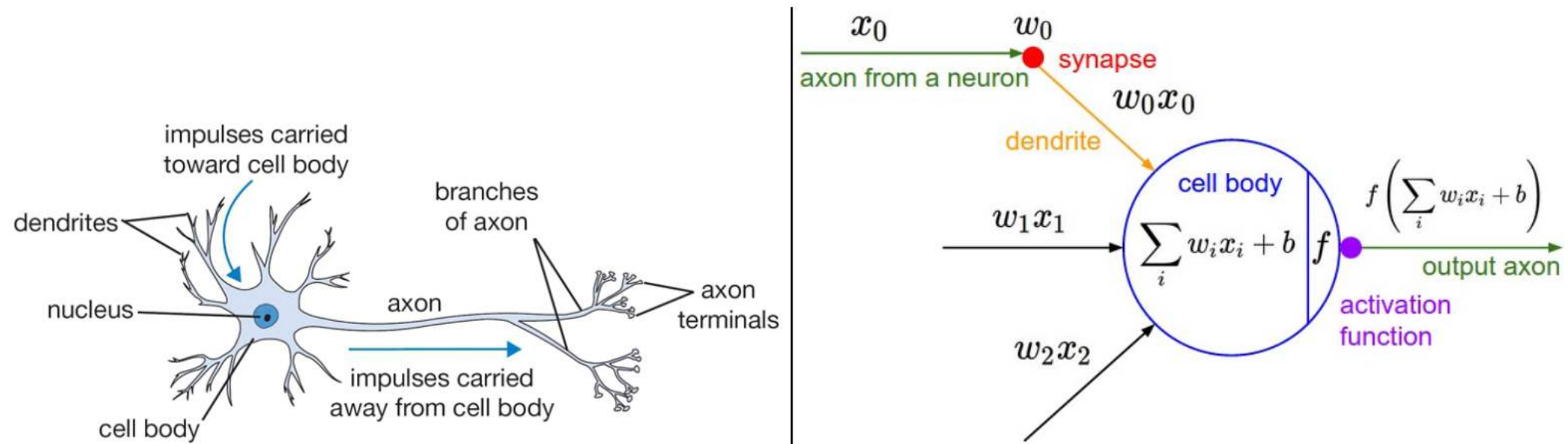
Some existing representation examples

- Kernel tricks:
 - project the data to higher dimensions to make them linear separable.
 - does not explicitly learn representations.

Deep Learning

- Deep Neural Networks
 - **General** feature extractor, with **multiple levels** of representation.
 - Automatically discover the representations needed for subsequent prediction tasks.

Neurons

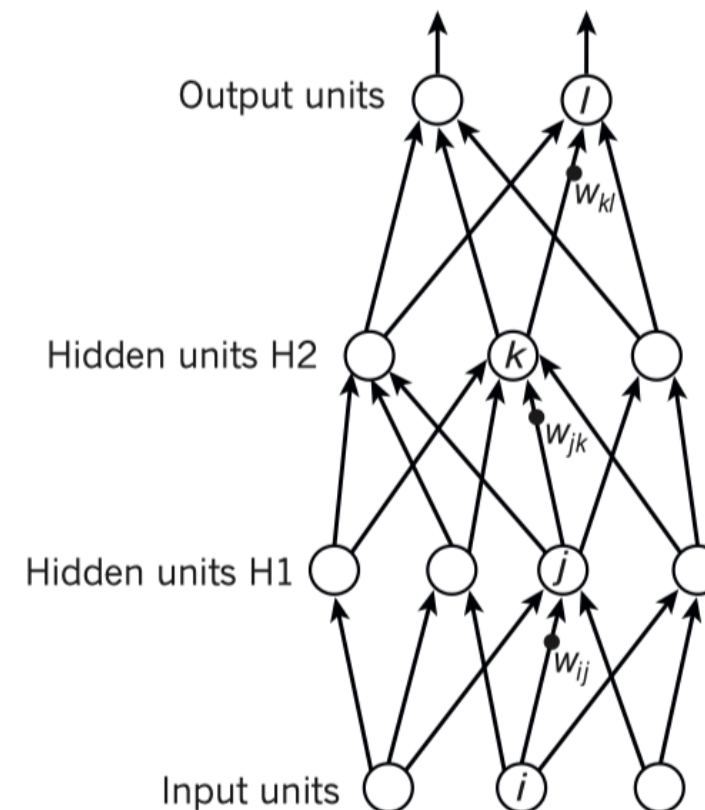


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Multi-layer Neural Networks

- $f()$: **activation** function
 - Activation function are usually **nonlinear**
 - When they are linear, reduces to linear models.
- w_{ij} : **weights**
- z_j, z_k : **activations**, weighted sums of previous layer's units
- y_j, y_k : **hidden units**
- Each unit is obtained by applying the activation function on activations

c



$$y_l = f(z_l)$$
$$z_l = \sum_{k \in H2} w_{kl} y_k$$

$$y_k = f(z_k)$$
$$z_k = \sum_{j \in H1} w_{jk} y_j$$

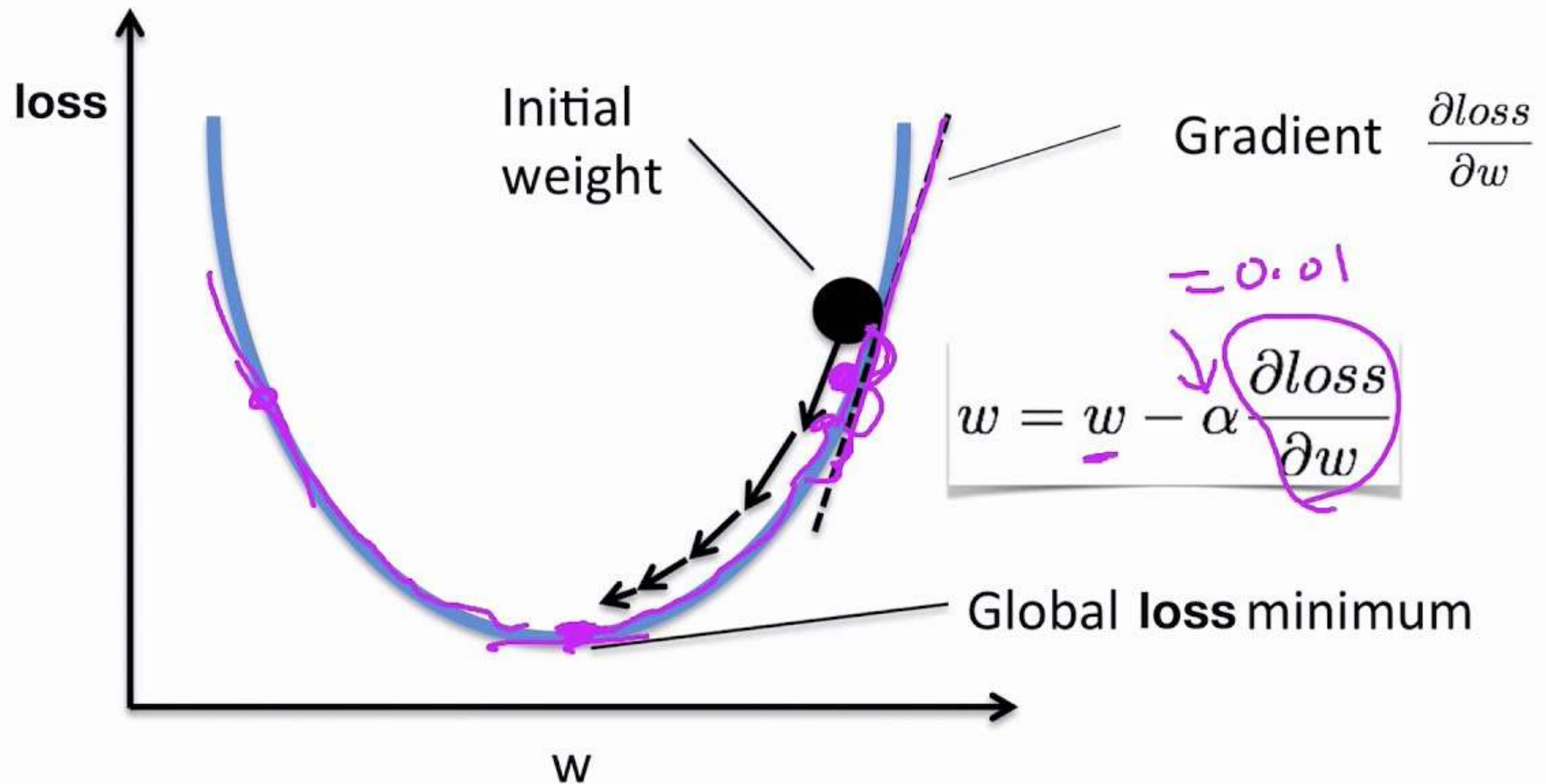
$$y_j = f(z_j)$$
$$z_j = \sum_{i \in \text{Input}} w_{ij} x_i$$

Learning

- Goal: with inputs and outputs observed, **learn the weights.**
- Algorithm:
 - Gradient descent
 - Error Back propagation

<https://www.youtube.com/watch?v=b4Vyma9wPHo>

Gradient descent algorithm



<https://www.youtube.com/watch?v=b4Vyma9wPHo>

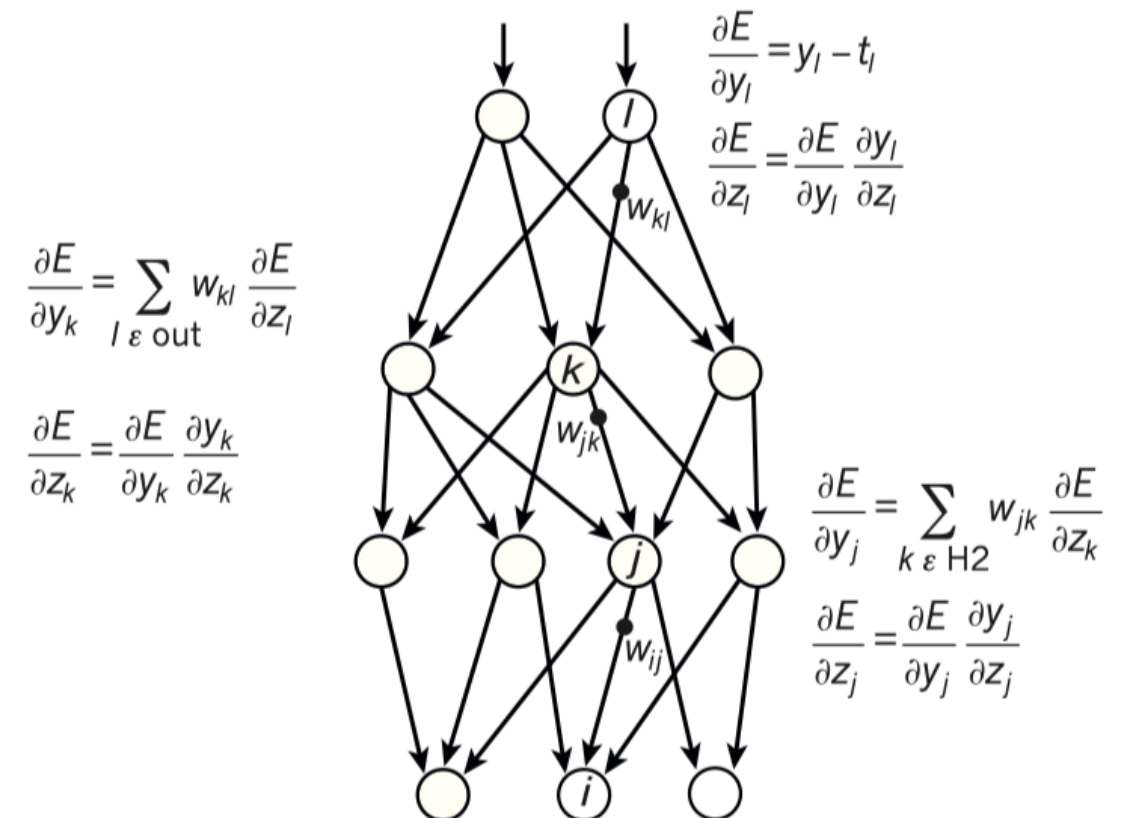
Error Back Propagation

- To use gradient descent, we need to know the value of $\frac{\partial E_l}{\partial w_{kl}}$

- $\frac{\partial E_l}{\partial w_{kl}} = \frac{\partial E_l}{\partial z_l} \frac{\partial z_l}{\partial w_{kl}}$
- $\frac{\partial E_l}{\partial z_l} = \frac{\partial E_l}{\partial y_l} \frac{\partial y_l}{\partial z_l} = \frac{\partial E_l}{\partial y_l} \frac{\partial f(z_l)}{\partial z_l}$
- $\frac{\partial E_l}{\partial y_l} = \frac{\partial \frac{1}{2}(y_l - t_l)^2}{\partial z_l} = y_l - t_l$ **error**
- $\frac{\partial z_l}{\partial w_{kl}} = \frac{\partial \sum_k w_{kl} y_k}{\partial w_{kl}} = y_k$

d

Compare outputs with correct answer to get error derivatives



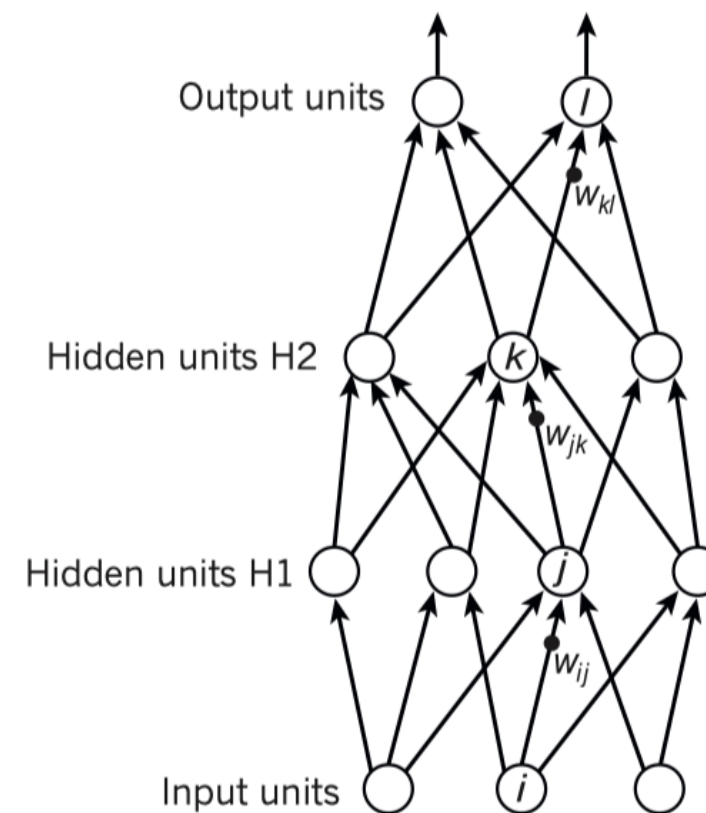
Neural Network: learning

- Initialize: randomly assign some weights (often small random values around 0).
1. **Forward pass**: take some input units X and calculate activations of all layers
 2. **Back propagation**: obtain partial derivatives of weights using back prop
 3. **Update weights** using gradient descent
 4. Repeat 1 - 3 until convergence.

Neural Network: learning

- Initialize: randomly assign some weights (often small random values around 0).
1. **Forward pass**: take some input units X and calculate activations of all layers
 2. **Back propagation**: obtain partial derivatives of weights using back prop
 3. **Update weights** using gradient descent
 4. Repeat 1 - 3 until convergence.

c



$$y_l = f(z_l)$$
$$z_l = \sum_{k \in H2} w_{kl} y_k$$

$$y_k = f(z_k)$$
$$z_k = \sum_{j \in H1} w_{jk} y_j$$

$$y_j = f(z_j)$$
$$z_j = \sum_{i \in \text{Input}} w_{ij} x_i$$

Neural Network: learning

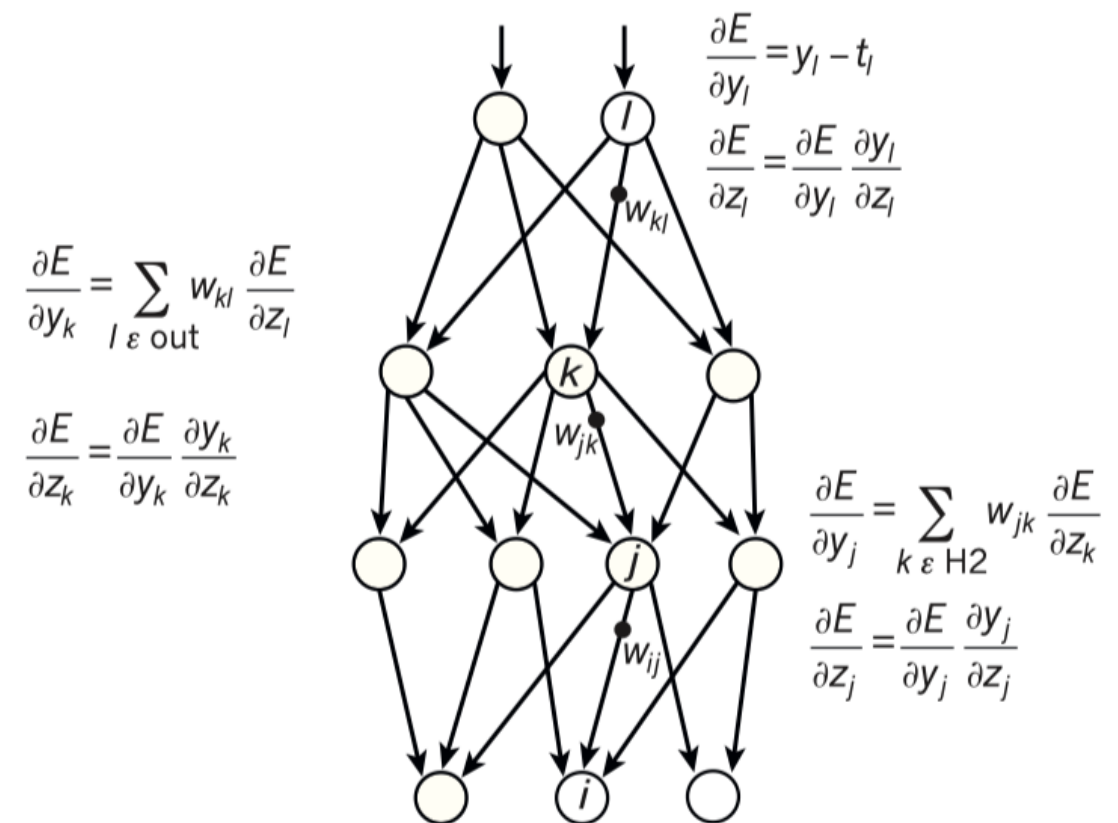
- Initialize: randomly assign some weights (often small random values around 0).
1. **Forward pass**: take some input units X and calculate activations of all layers
 2. **Back propagation**: obtain partial derivatives of weights using back prop
 3. **Update weights** using gradient descent
 4. Repeat 1 - 3 until convergence.

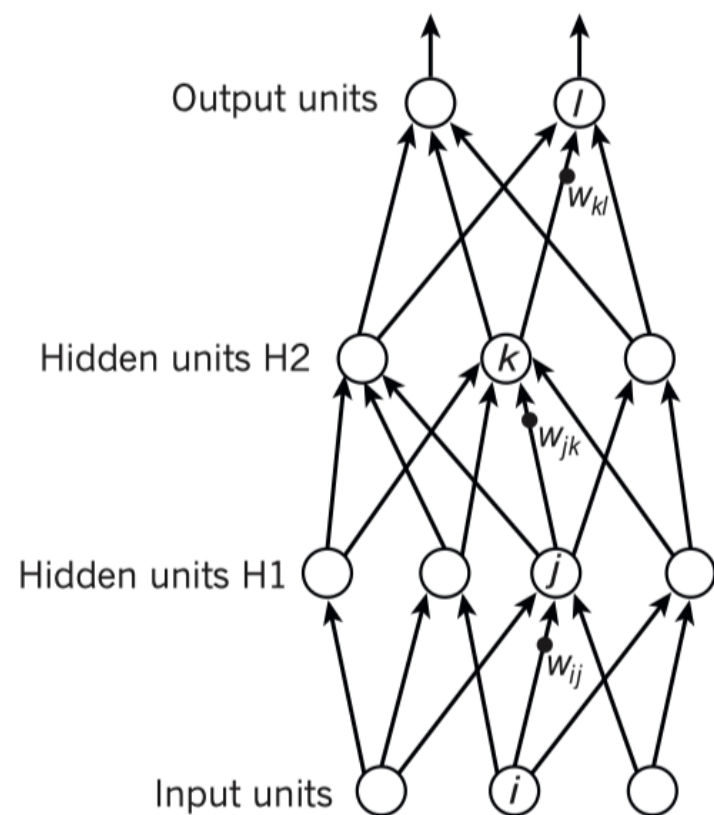
Neural Network: learning

- Initialize: randomly assign some weights (often small random values around 0).
1. **Forward pass**: take some input units X and calculate activations of all layers
 2. **Back propagation**: obtain partial derivatives of weights using back prop
 3. **Update weights** using gradient descent
 4. Repeat 1 - 3 until convergence.

d

Compare outputs with correct answer to get error derivatives



c

$$y_l = f(z_l)$$

$$z_l = \sum_{k \in H2} w_{kl} y_k$$

$$y_k = f(z_k)$$

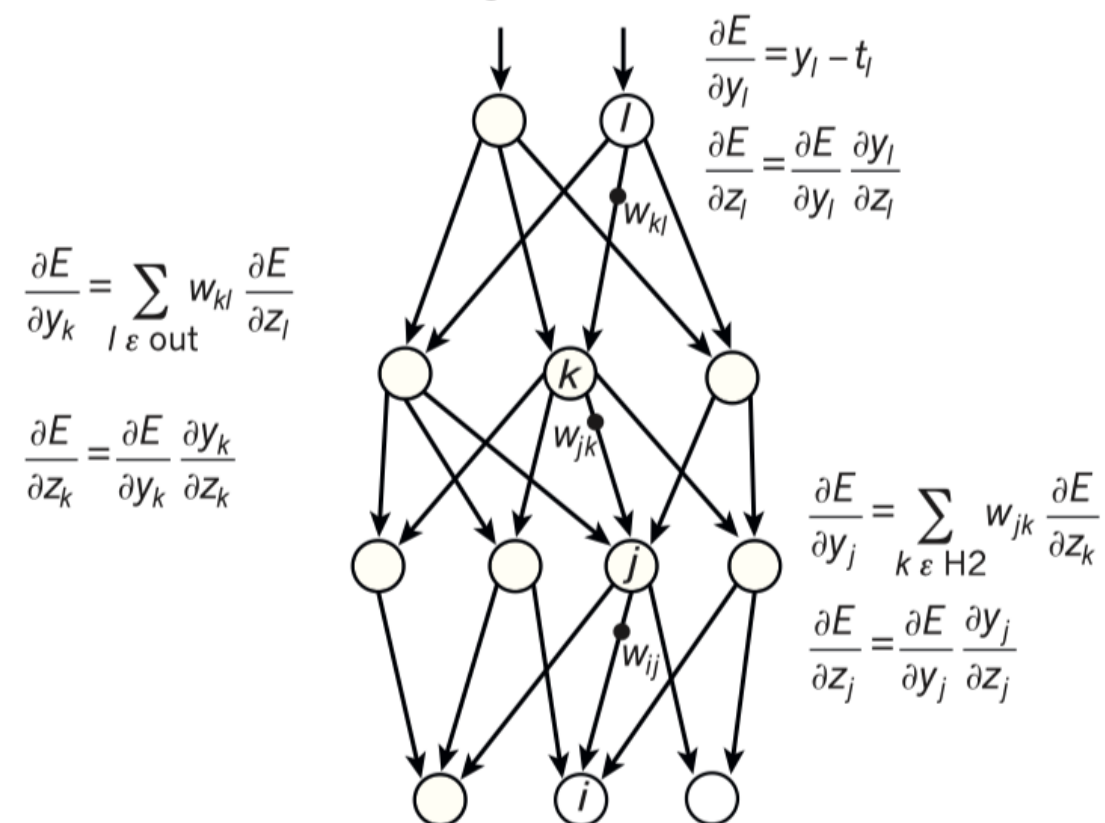
$$z_k = \sum_{j \in H1} w_{jk} y_j$$

$$y_j = f(z_j)$$

$$z_j = \sum_{i \in \text{Input}} w_{ij} x_i$$

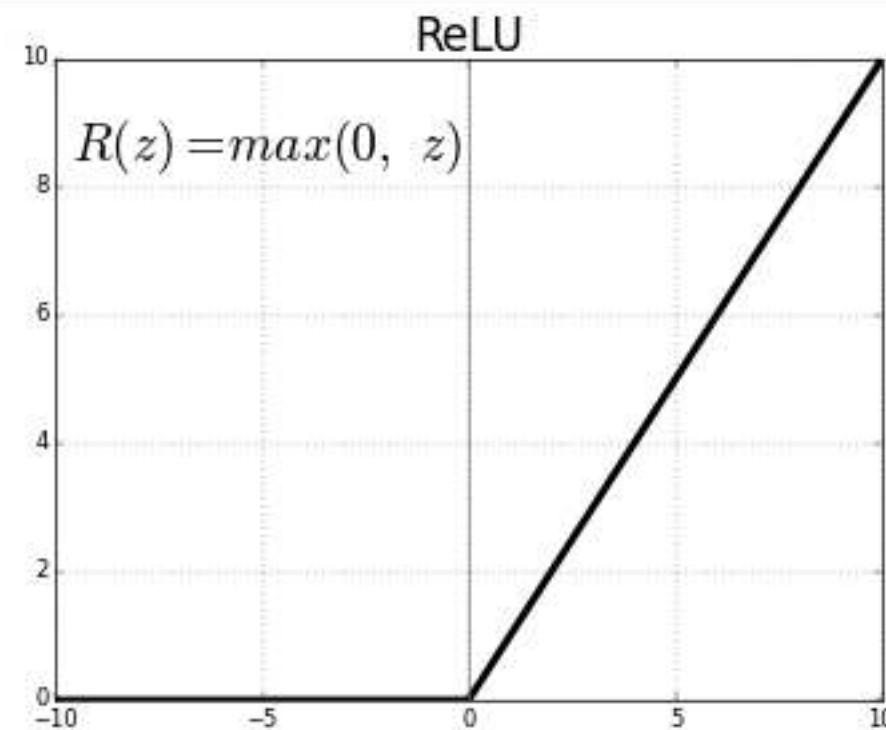
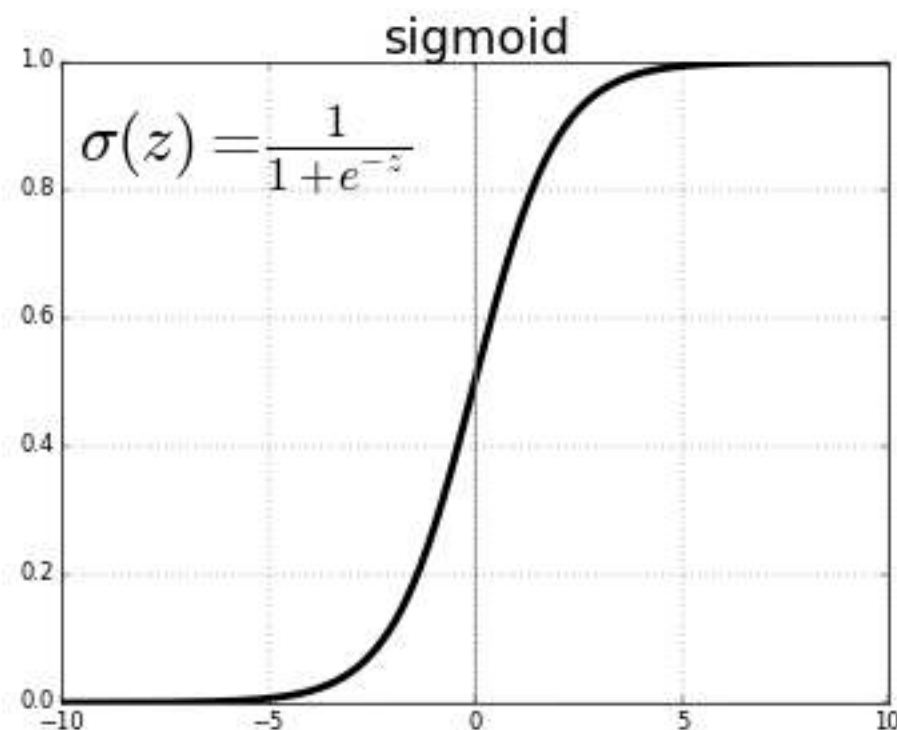
d

Compare outputs with correct answer to get error derivatives



Choice of activation function

- Activation function adds non-linearity to the linear weighted sum of hidden units.
- Common choices: Sigmoid and ReLu (Rectified Linear Unit)
- Sigmoid has the “vanishing gradient” problem: $\frac{df(x)}{d(x)} = f(x) \cdot (1 - f(x))$



Basic neural networks

- Free parameters:
 - the number of hidden layers
 - the number of units per layer.
 - # of hidden units increase ->
 - model complexity increases
 - more likely to overfit.

Convolutional Neural Networks

- **Sparse connectivity** (local connections): each unit depends on only on local regions of the previous layer.
- rationale: local groups of values are often highly correlated
 - n-grams in 1D texts and speeches;
 - subregions in 2D photos
 - video clip from a longer video (3D).

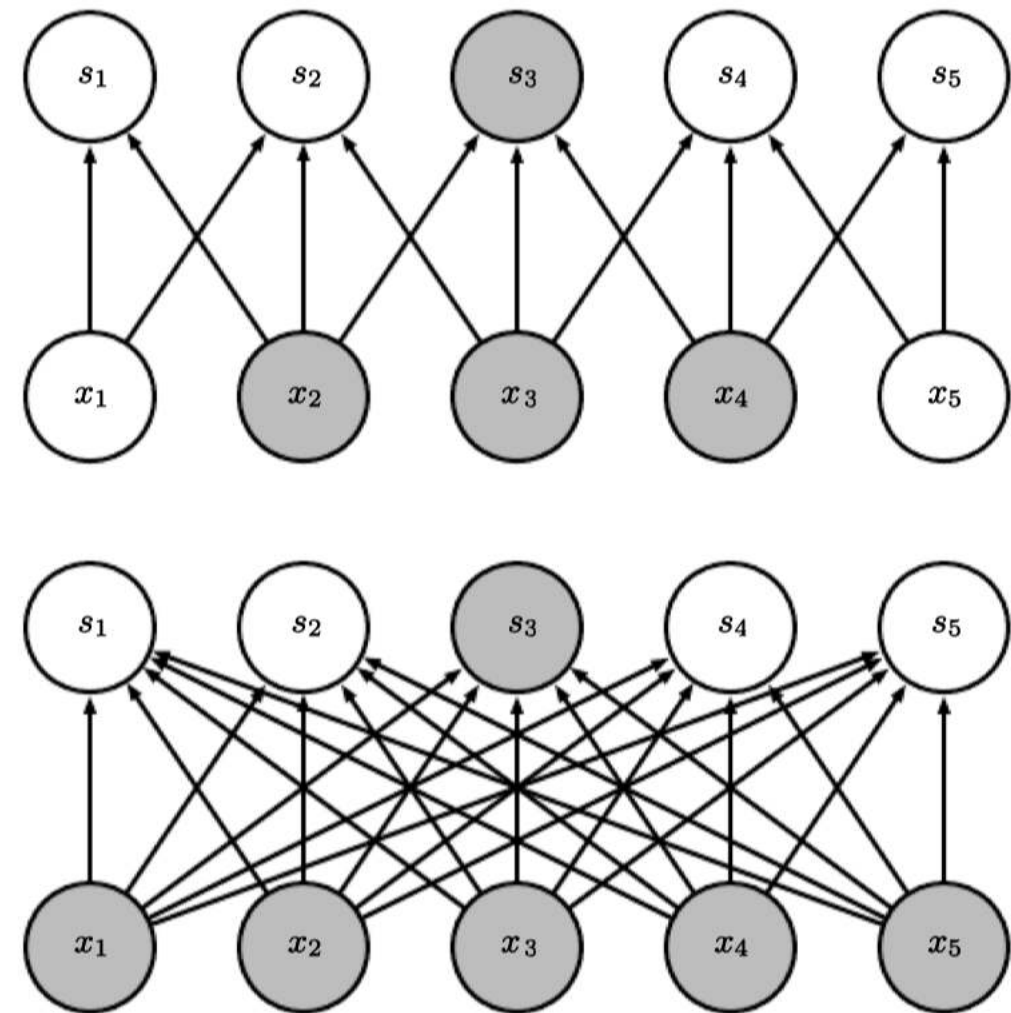
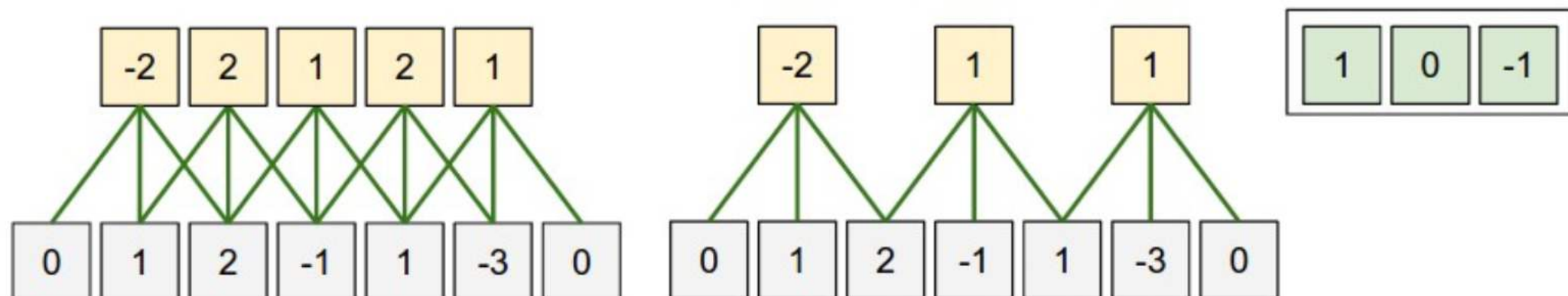


Figure 9.2, Goodfellow et al., 2015

Convolutional Neural Networks

- Parameter sharing
 - rationale: a particular layer fulfills some feature extraction tasks.
 - This task should be invariant to the location of subregion.



Example of Convolution

Kernel is also known as filter

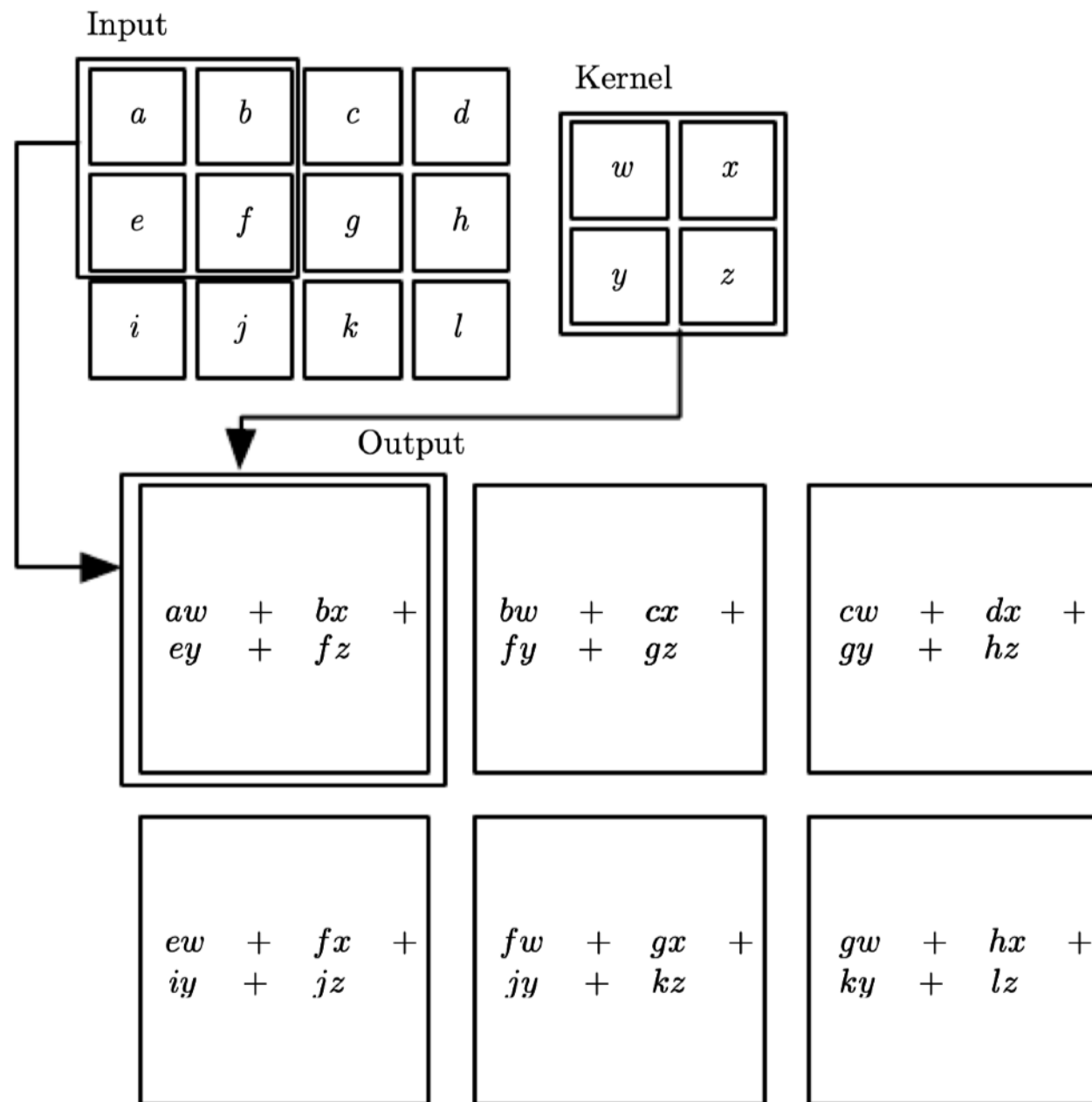
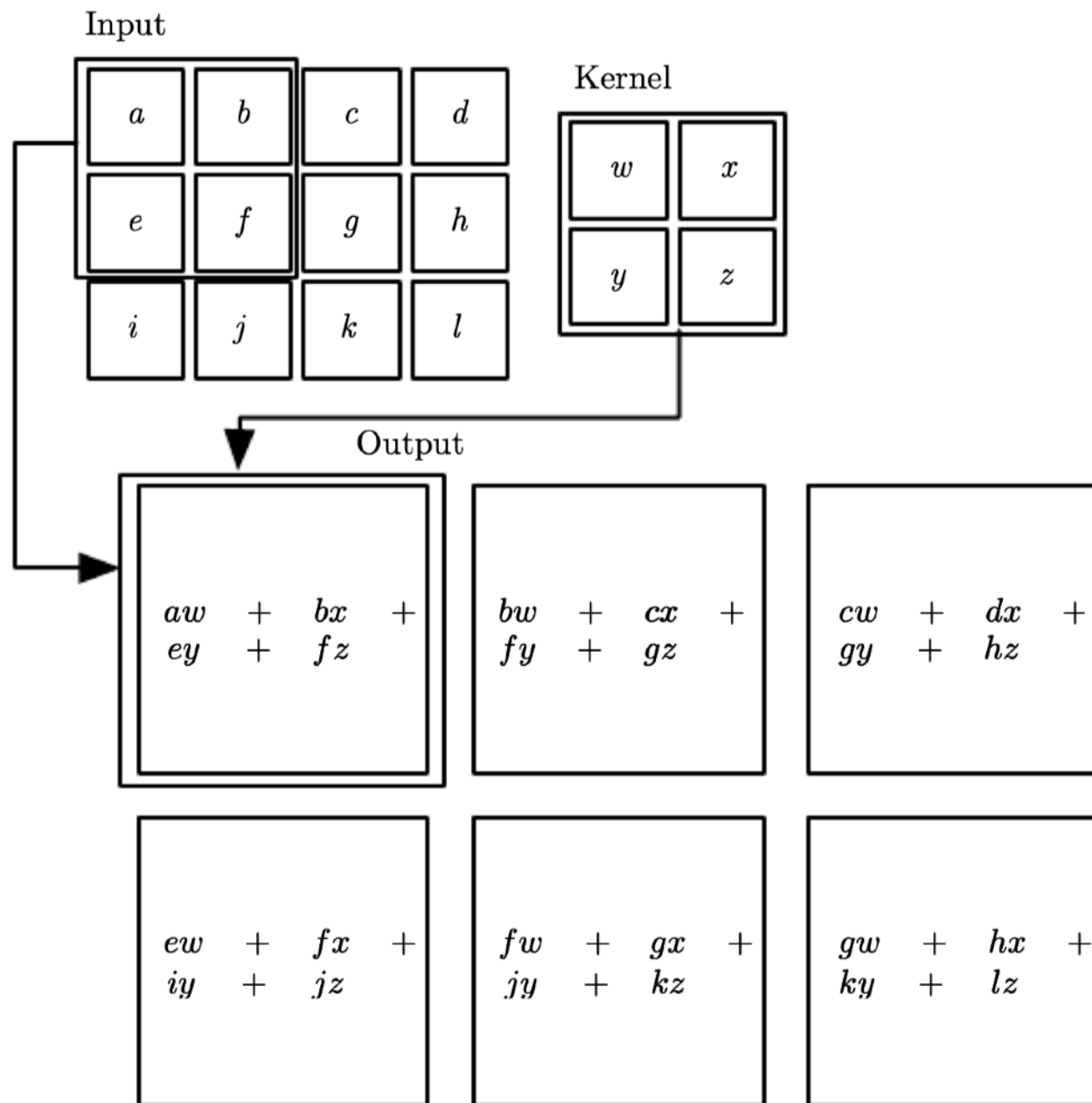


Figure 9.1, Goodfellow et al., 2015

Example of Convolution

Kernel is also known as filter



1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Figure 9.1, Goodfellow et al., 2015

Example of Convolution

Kernel is also known as filter

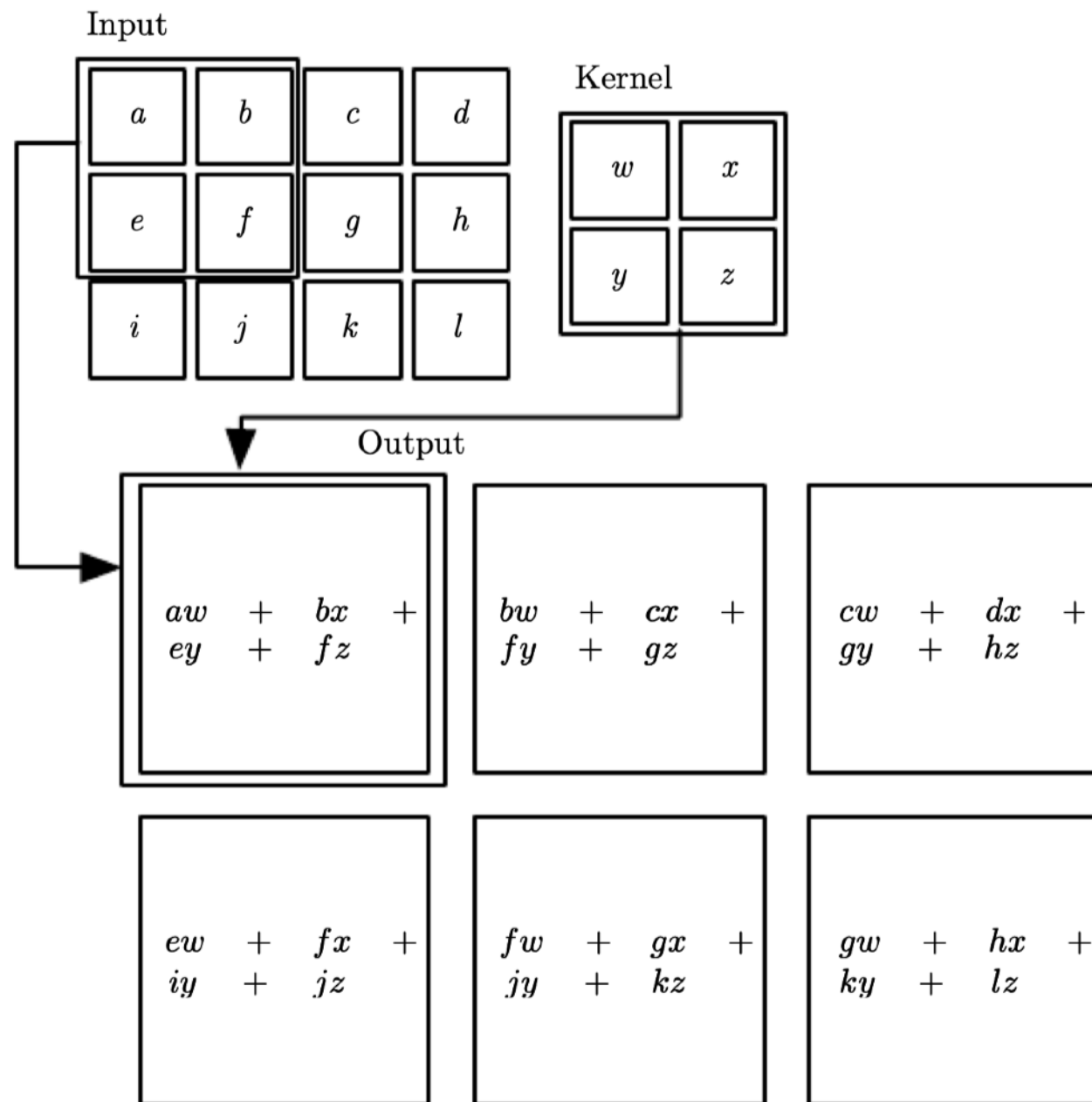
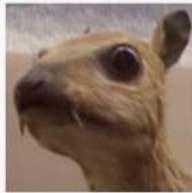
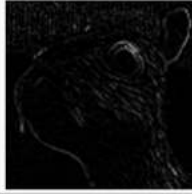
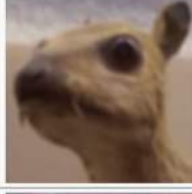


Figure 9.1, Goodfellow et al., 2015

Filter Examples

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Feature Maps

- Each convolution extract one type of feature; it is also called a **feature map**
- Weights of a feature map are the same (parameter sharing)
- There are often multiple feature maps; each aims to capture a different feature (e.g., edge, circle, lines).

One convolutional Layer

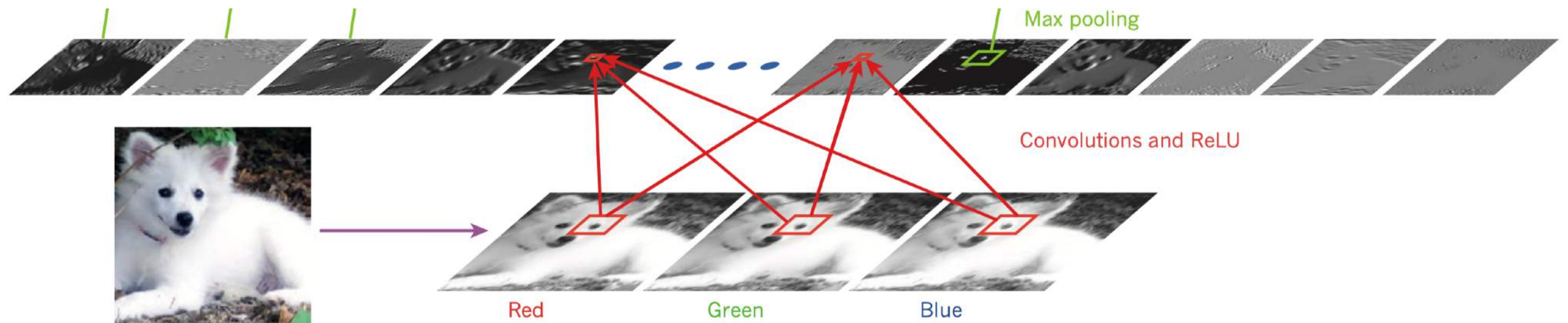
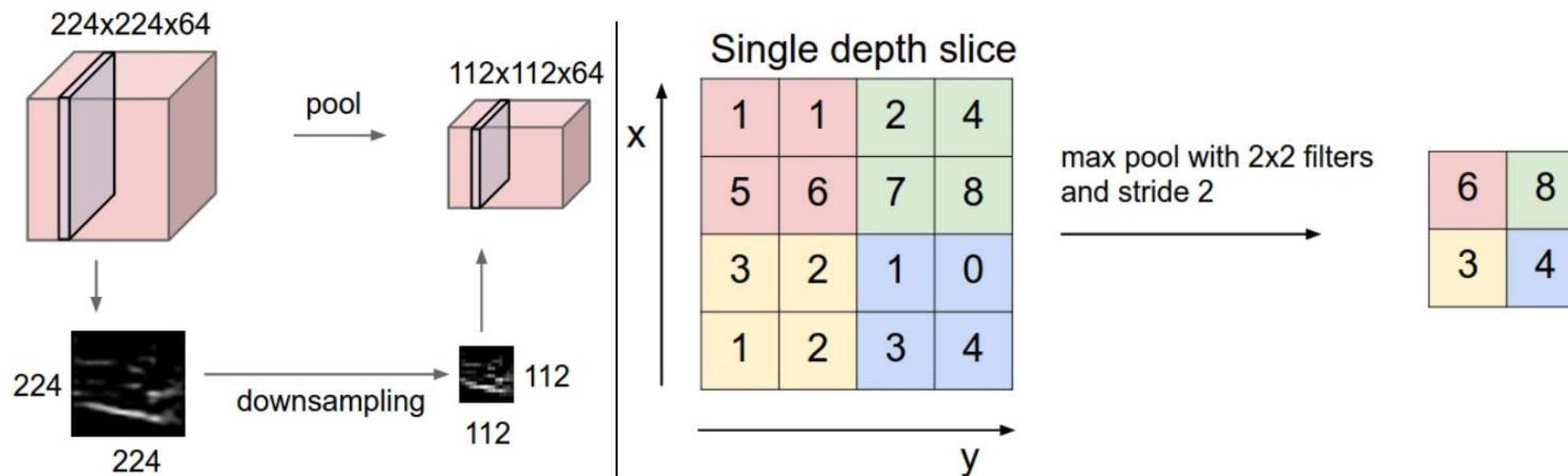


Figure 2, LeCun et al., 2015

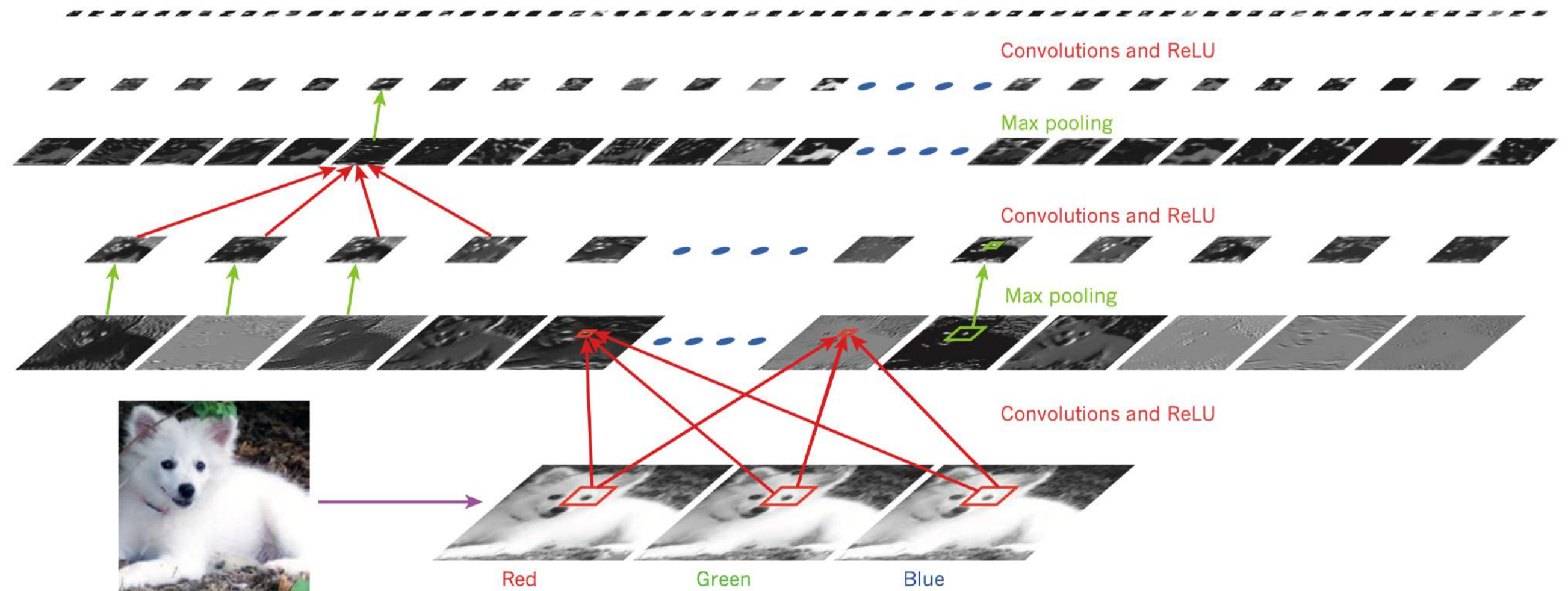
Pooling

- why we need pooling:
 - translation invariant: small changes in feature location does not matter.
 - down sampling; reduce complexity.
- max-pooling is popular, but there are other types (e.g., average pooling).



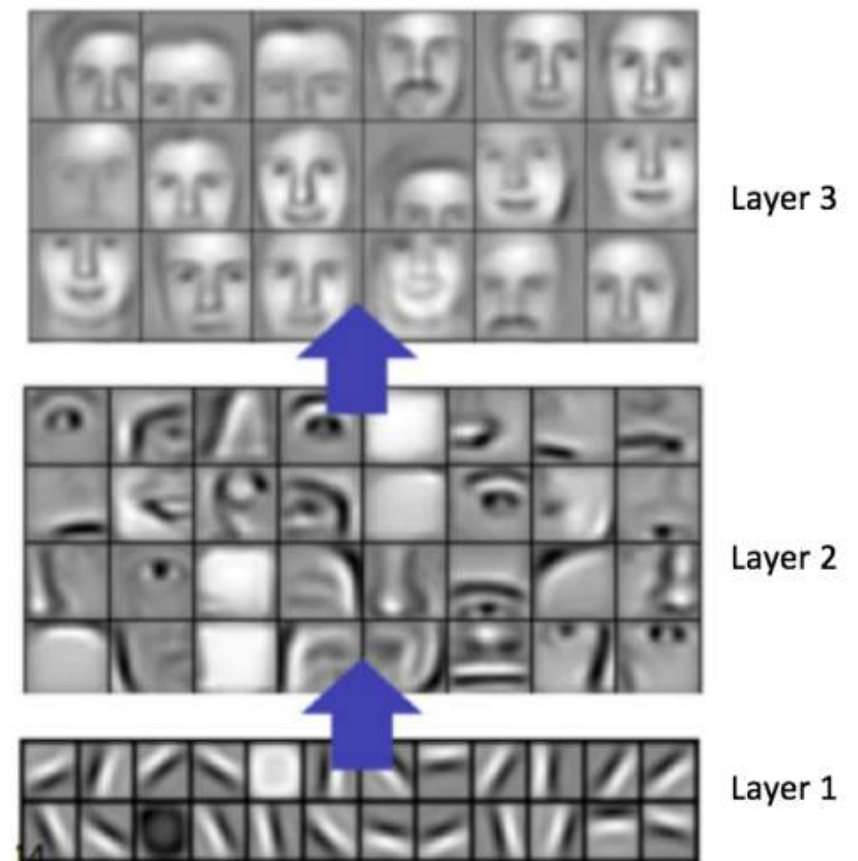
Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size $[224 \times 224 \times 64]$ is pooled with filter size 2, stride 2 into output volume of size $[112 \times 112 \times 64]$. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2×2 square).

Convolutional Layer + Pooling Layer



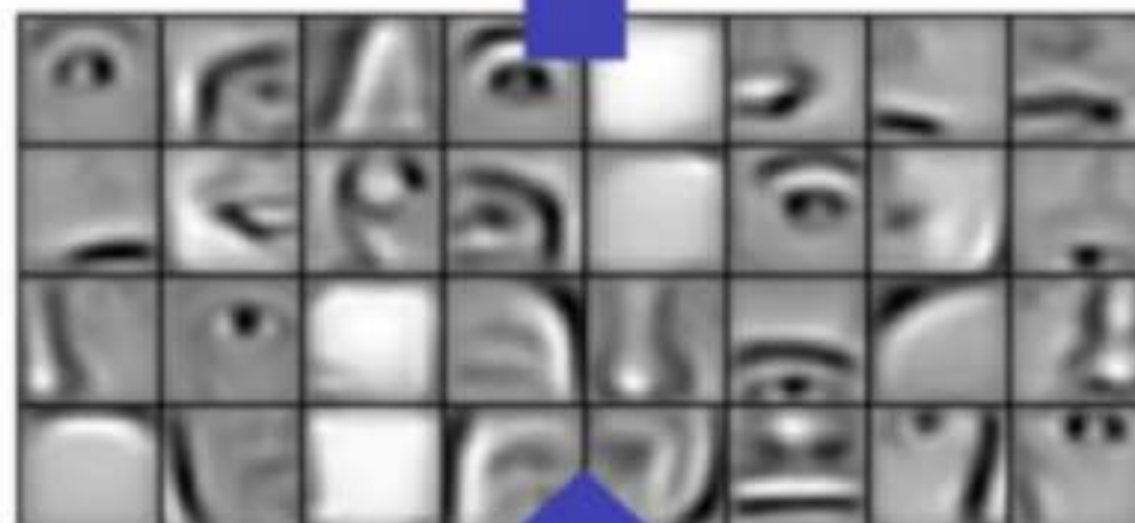
Why multiple layers

- Multiple levels of representation
 - lower layers capture basic motifs such as edges and circles.
 - Upper layers capture combinations of motifs.





Layer 3



Layer 2



Layer 1

Neural Networks vs Conv Neural Networks

Neural Networks vs Conv Neural Networks

- Neural networks are wide
- Fully connected
- Weights between layers are all different

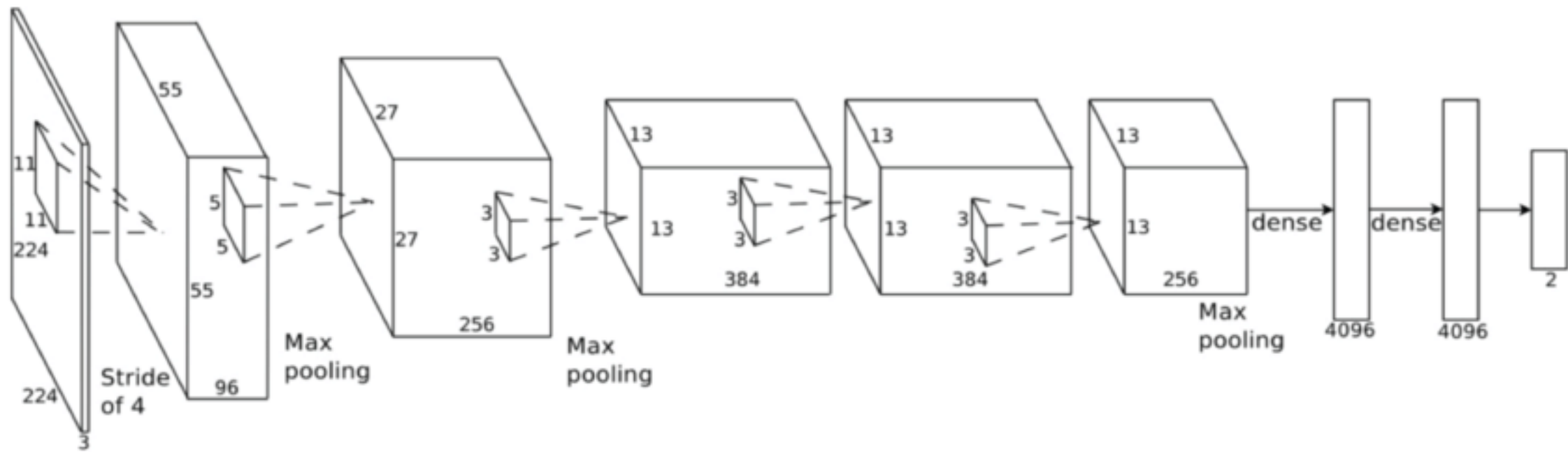
Neural Networks vs Conv Neural Networks

- Neural networks are wide
- Fully connected
- Weights between layers are all different
- Conv Nets are deep
- Sparsely connected
- Weights in each feature map is the same: extract the same type of feature well.
- weights across feature maps are different: extract many different types of features.

Last step: fully connected layers

- After several rounds of (Conv Layers + pooling layers), the last (or last several) layers produce outputs.
- Without the feature extraction part (Conv Layers + pooling layers), the network is equivalent to basic neural networks.
- We can skip the last fully connected layers, and use the extracted features as input for other algorithms (e.g., SVM or logistic regression).

AlexNet, 2012

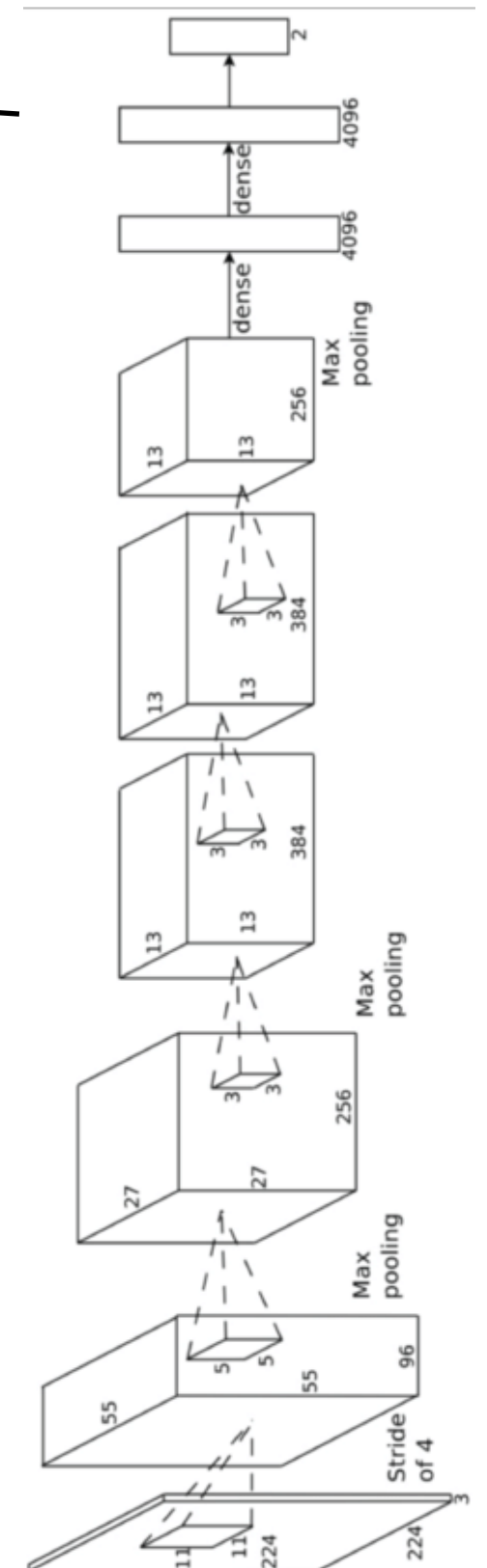


Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097-1105. 2012.

CNN as a feature extractor

- Visualize the last layer: should contain representation of the object itself.
- A good representation of feature should naturally put similar objects in similar positions.

Test image L2 Nearest neighbors in feature space



Regularization

- traditional: add regularizer terms, to the loss function

$$||w||^2$$

- early stopping: a kind of cross-validation strategy.
 - split the data into train, validation and test
 - stop when test accuracy begins to drop
- dropout: memorizing the past is not always useful
 - randomly remove some nodes in the network

Regularization

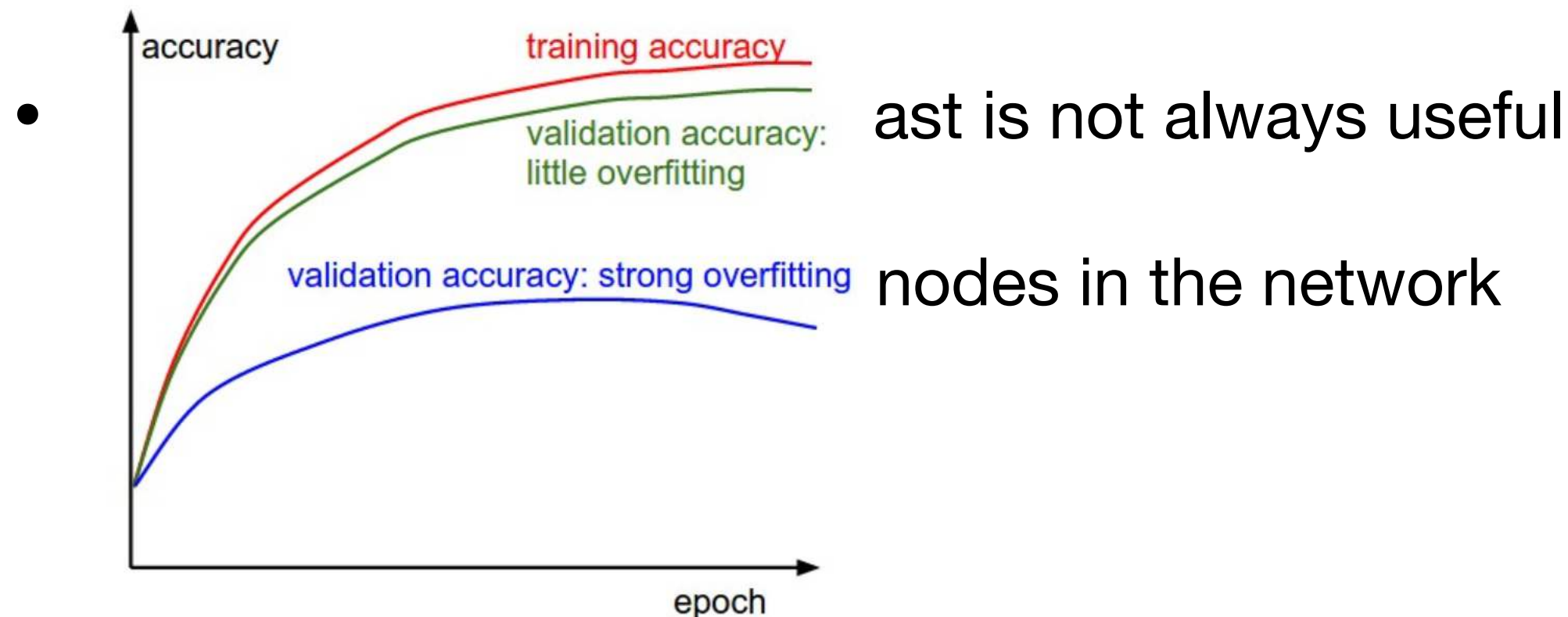
- traditional: add regularizer terms, to the loss function

$$||w||^2$$

- early stopping: a kind of cross-validation strategy.

- split the data into train, validation and test

- stop when test accuracy begins to drop



Regularization

- traditional: add regularizer terms, to the loss function

$$||w||^2$$

- early stopping: a kind of cross-validation strategy.
 - split the data into train, validation and test
 - stop when test accuracy begins to drop
- dropout: memorizing the past is not always useful
 - randomly remove some nodes in the network

Regularization

- traditional: add regularizer terms, to the loss function

$$||w||^2$$

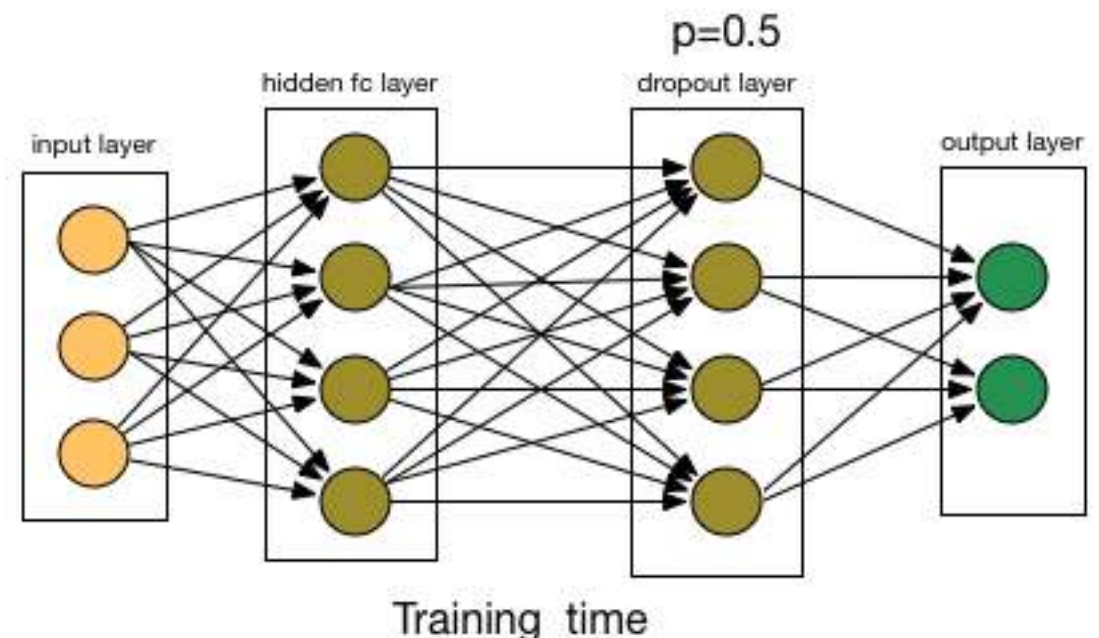
- early stopping: a kind of cross-validation strategy.

- split the data into train, validation and test

- stop when test accuracy begins to drop

- dropout: memorizing the past i

- randomly remove some nodes



Lots of parameter choices

- Number of layers:
 - More layers means better representation ability
 - may overfit; increase computational complexity.
- Number of feature maps: usually double by each convolutional layers.
- Filter size, stride size.
- Learning rates
- Batch size: how many data points the learning algorithm sees each time
- Number of epochs: # of full pass of data.

Some history

Some history

- Rosenblatt, F. (1958, 1962): basic neural networks

Some history

- Rosenblatt, F. (1958, 1962): basic neural networks
- Rumelhart, Hinton, and Ronald(1986): Back Prop

Some history

- Rosenblatt, F. (1958, 1962): basic neural networks
- Rumelhart, Hinton, and Ronald(1986): Back Prop
- Le Cun et al., (1989): Conv Net.

Some history

- Rosenblatt, F. (1958, 1962): basic neural networks
- Rumelhart, Hinton, and Ronald(1986): Back Prop
- Le Cun et al., (1989): Conv Net.
- ImageNet: 14 million images organized according to the **WordNet** hierarchy of nouns

Some history

- Rosenblatt, F. (1958, 1962): basic neural networks
- Rumelhart, Hinton, and Ronald(1986): Back Prop
- Le Cun et al., (1989): Conv Net.
- ImageNet: 14 million images organized according to the WordNet hierarchy of nouns
 - <http://image-net.org/about-overview>

Some history

- Rosenblatt, F. (1958, 1962): basic neural networks
- Rumelhart, Hinton, and Ronald (1986): Back Prop
- Le Cun et al., (1989): Conv Net.
- ImageNet: 14 million images organized according to the **WordNet** hierarchy of nouns
 - <http://image-net.org/about-overview>
 - J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR), 2009.

Some history

- Rosenblatt, F. (1958, 1962): basic neural networks
- Rumelhart, Hinton, and Ronald (1986): Back Prop
- Le Cun et al., (1989): Conv Net.
- ImageNet: 14 million images organized according to the **WordNet** hierarchy of nouns
 - <http://image-net.org/about-overview>
 - J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR), 2009.
- ImageNet Challenge: 1M images and 1000 categories.

Some history

- Rosenblatt, F. (1958, 1962): basic neural networks
- Rumelhart, Hinton, and Ronald (1986): Back Prop
- Le Cun et al., (1989): Conv Net.
- ImageNet: 14 million images organized according to the **WordNet** hierarchy of nouns
 - <http://image-net.org/about-overview>
 - J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR), 2009.
- ImageNet Challenge: 1M images and 1000 categories.
 - Image classification

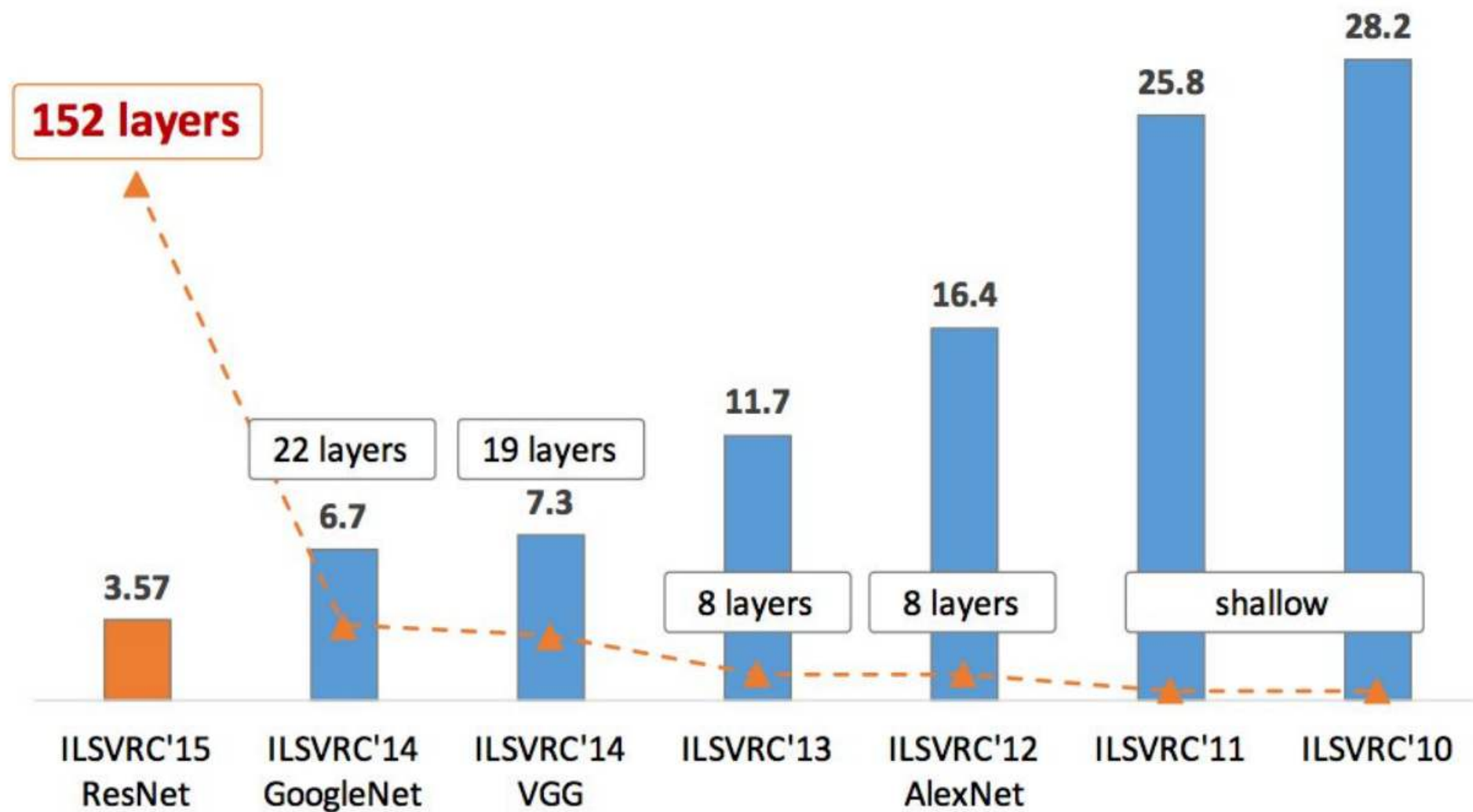
Some history

- Rosenblatt, F. (1958, 1962): basic neural networks
- Rumelhart, Hinton, and Ronald (1986): Back Prop
- Le Cun et al., (1989): Conv Net.
- ImageNet: 14 million images organized according to the **WordNet** hierarchy of nouns
 - <http://image-net.org/about-overview>
 - J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR), 2009.
- ImageNet Challenge: 1M images and 1000 categories.
 - Image classification
 - Single-object localization: try to find a bounding box indicating the position and scale of one instance of each object category

Some history

- Rosenblatt, F. (1958, 1962): basic neural networks
- Rumelhart, Hinton, and Ronald (1986): Back Prop
- Le Cun et al., (1989): Conv Net.
- ImageNet: 14 million images organized according to the **WordNet** hierarchy of nouns
 - <http://image-net.org/about-overview>
 - J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR), 2009.
- ImageNet Challenge: 1M images and 1000 categories.
 - Image classification
 - Single-object localization: try to find a bounding box indicating the position and scale of one instance of each object category
 - Object detection: find a list of object categories present in the image

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

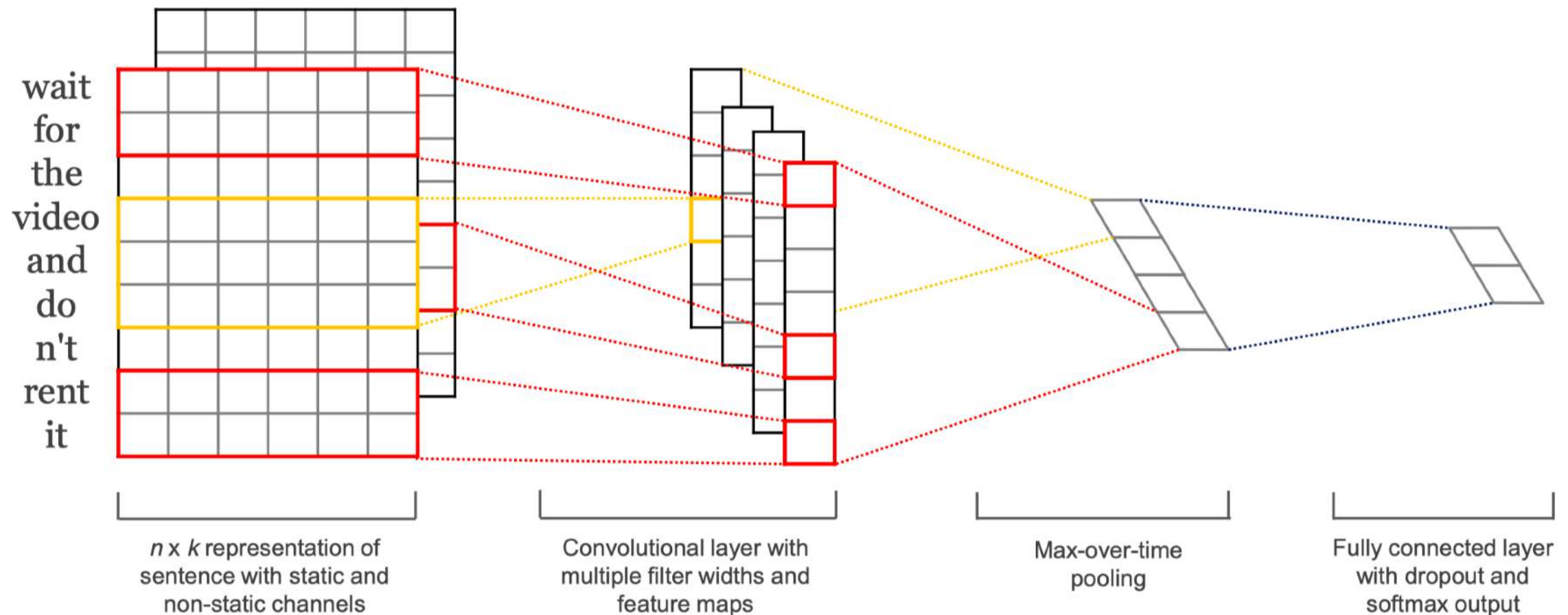


What if you do not have many data

- Transfer Learning
 - ConvNet as a **feature extractor**: feed images into existing models, and obtain the output before the last fully connected layer.
 - work if your images is similar to that of ImageNet
 - **Fine-tuning** ConvNet:
 - Freeze the weights of beginning layers of existing models
 - Train the weights of later layers

CNN in other contexts

- Text Classification



Kim, . 2014. "Convolutional Neural Networks for Sentence Classification." ArXiv: 1408.5882, August. <http://arxiv.org/abs/1408.5882>.

Summary

- Learning representation itself is important
- Deep learning provides general feature extractor
- Extracted features can be used for prediction tasks