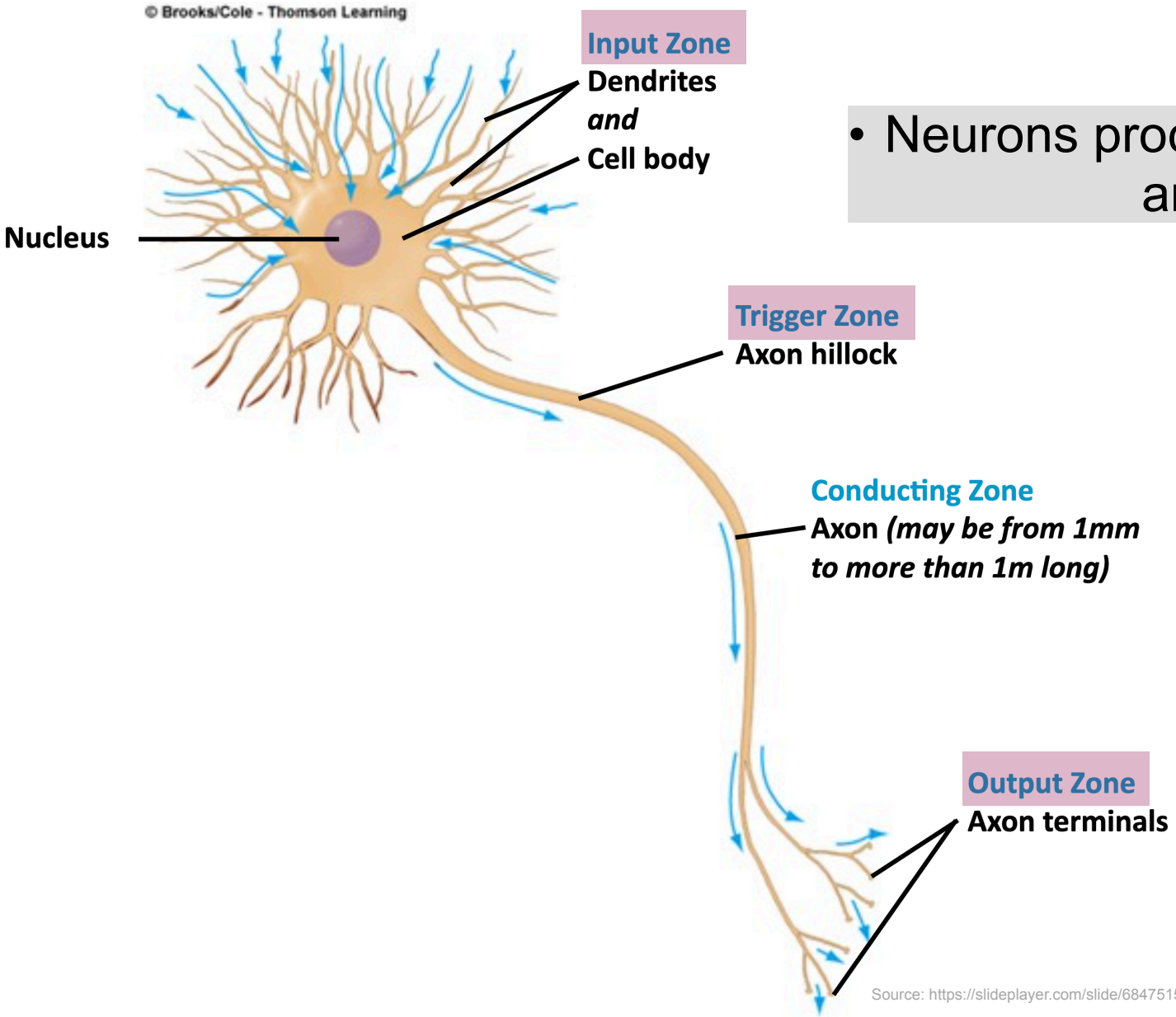
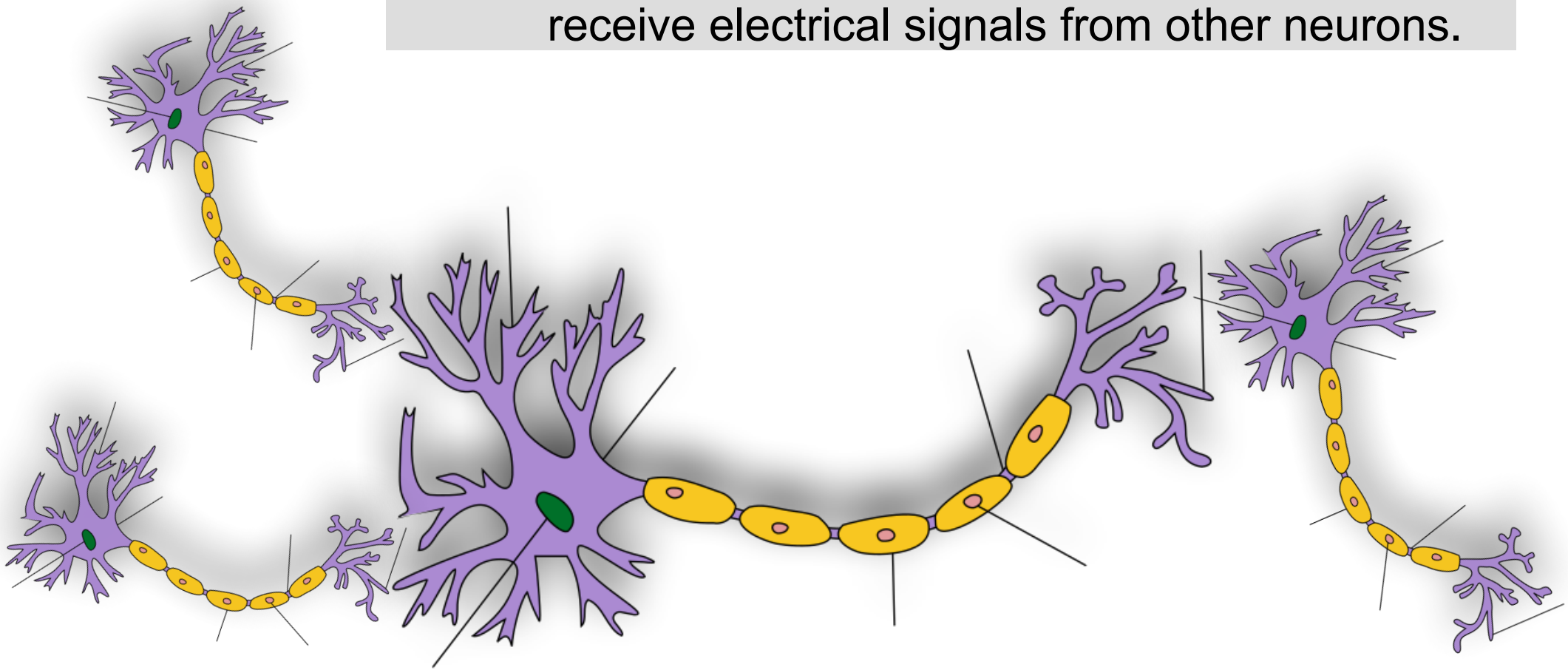
Neural Networks



- Neurons process input signals and can be activated.

Source: https://slideplayer.com/slide/6847515/

- Neurons are connected to and
  receive electrical signals from other neurons.



Source: https://upload.wikimedia.org/wikipedia/commons/thumb/b/bc/Neuron_Hand-tuned.svg/800px-Neuron_Hand-tuned.svg.png

Artificial Neural Networks

Boolean Circuit Model of Brain (McCulloch & Pitts, 1943)

- A model of artificial neurons in which each neuron is characterized as being “on” or “off”
  - “on” occurs in response to stimulation by a sufficient number of neighboring neurons
- A simple mathematical model of neurons
  - It “fires” when a linear combination of its inputs exceeds some threshold (a linear classifier)

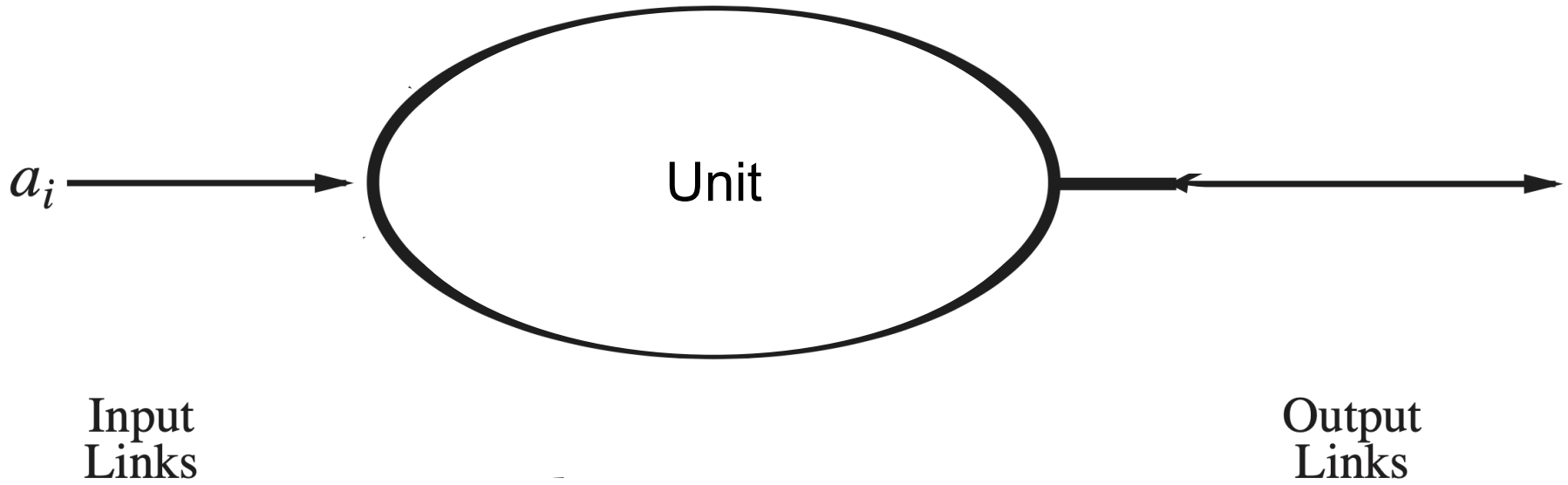
Artificial Neural Networks

- Model mathematical function from inputs to outputs based on the structure and parameters of the network
  - Input values are typically continuous
  - Outputs are continuous

- The network learns by adjusting the values of the network's **parameters** so that the networks as a whole fits the training data

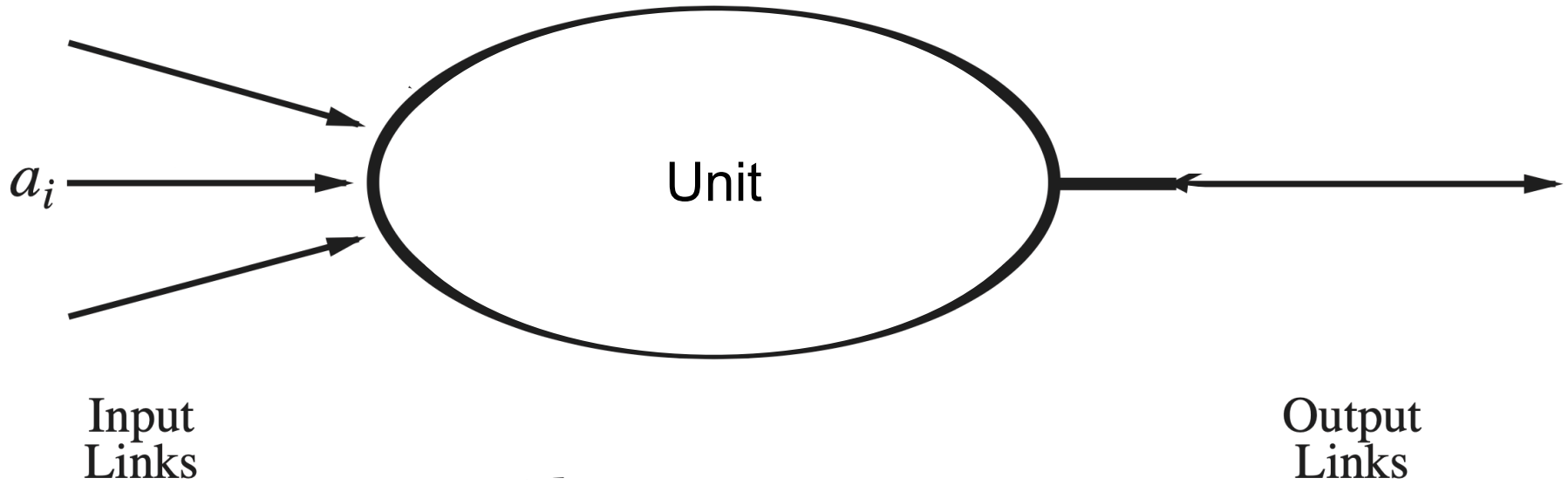
Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**



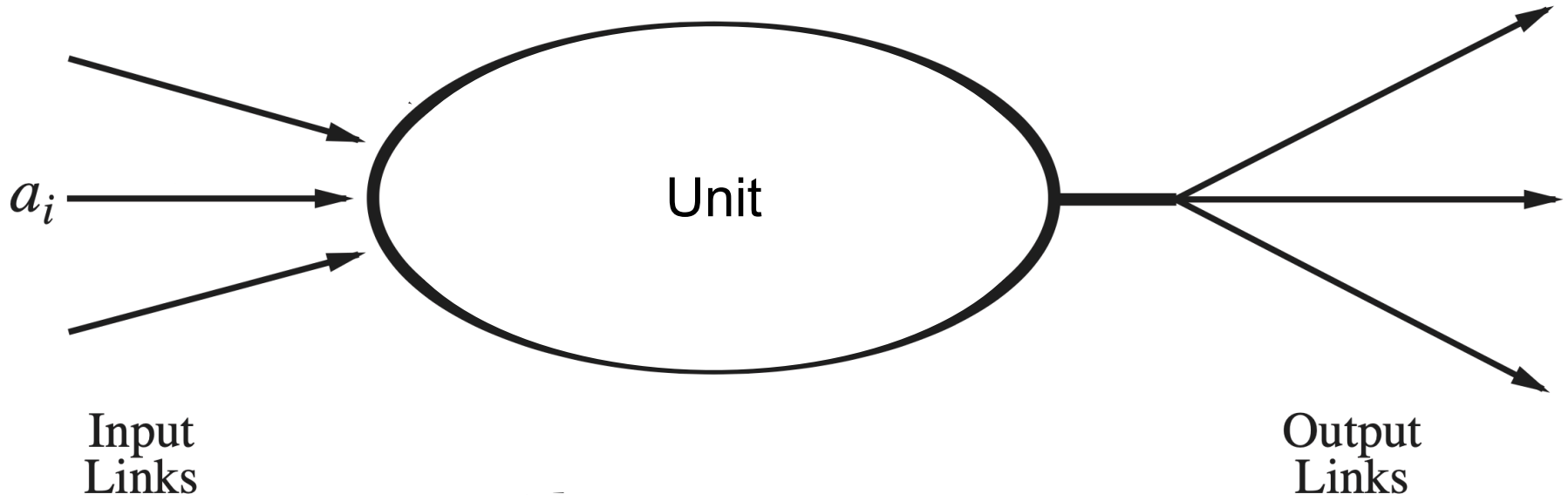
Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**



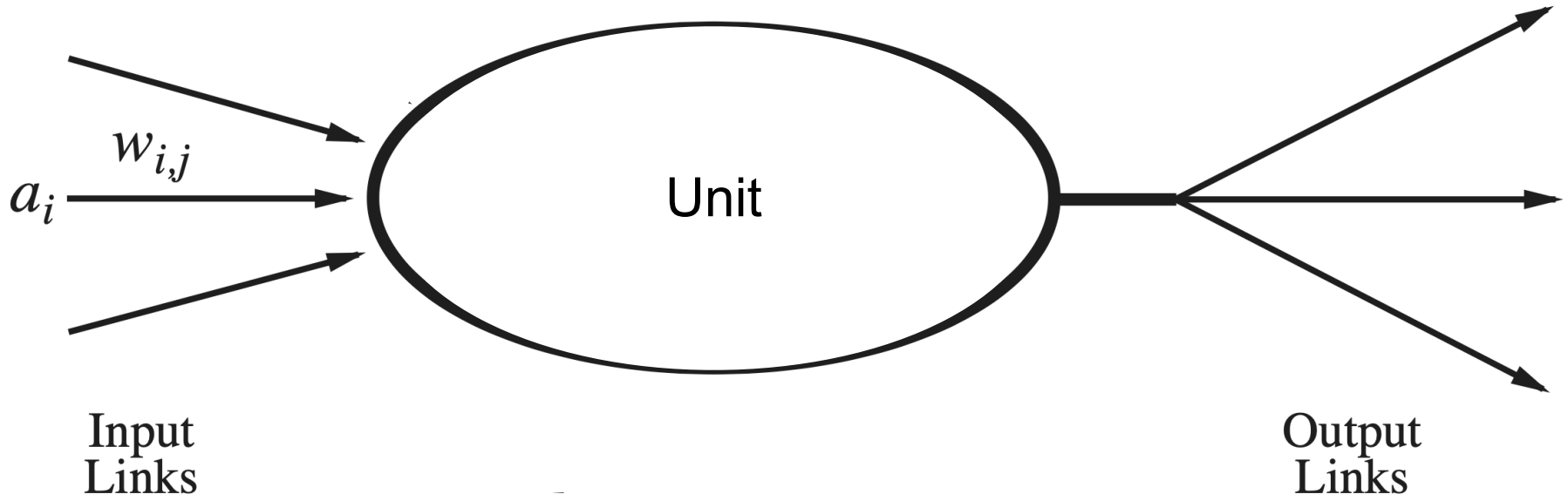
Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**



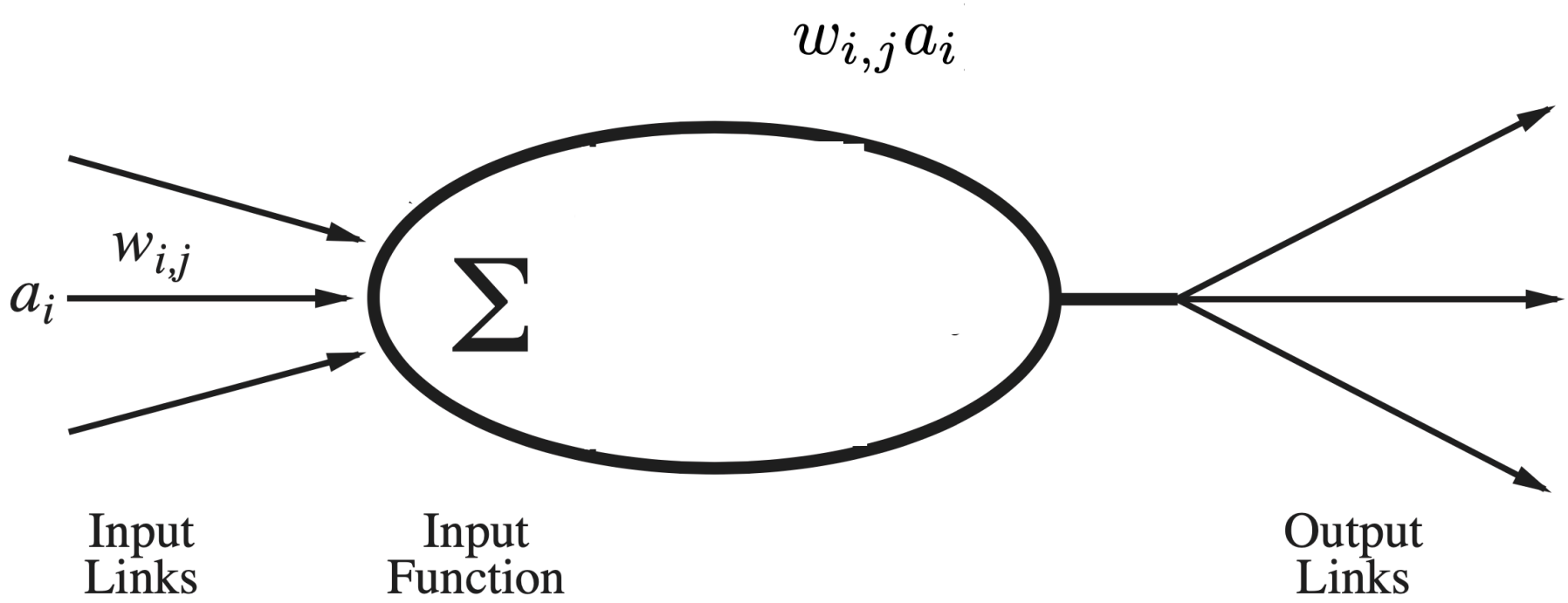
Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**



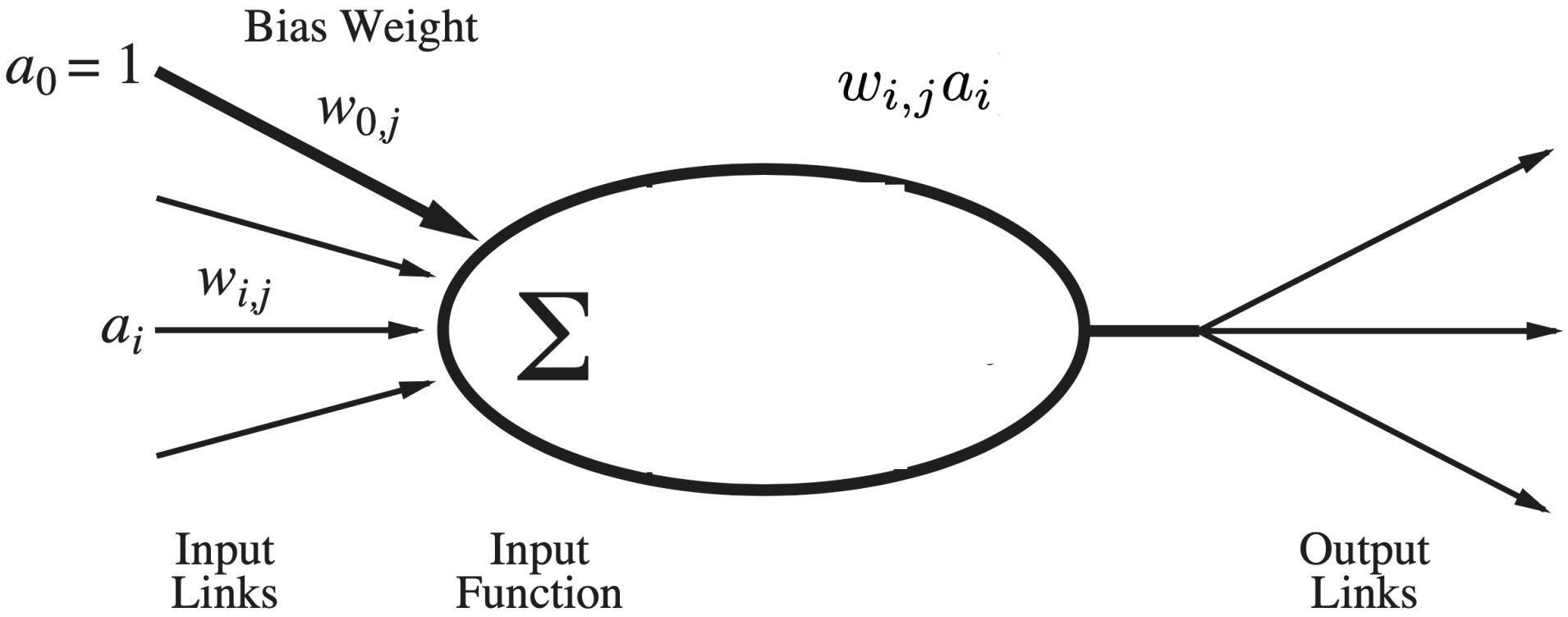
Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**



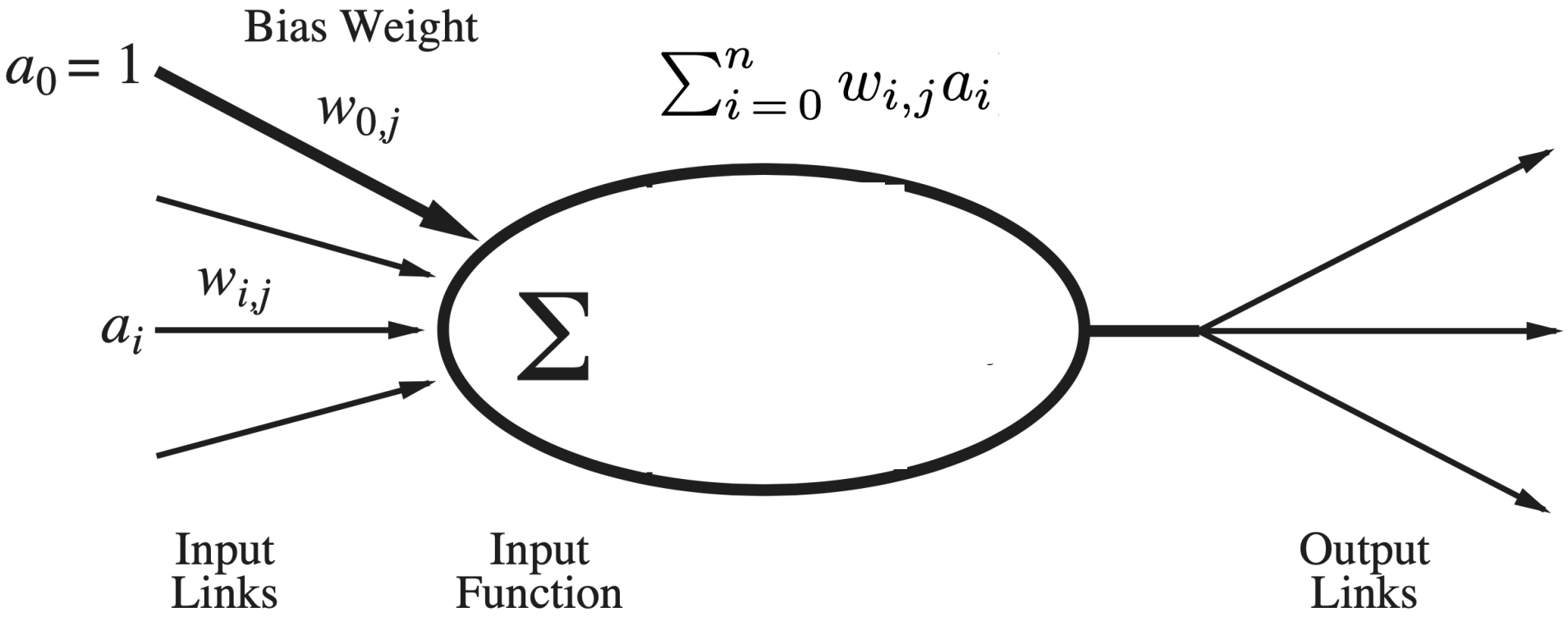
Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**
  - Each unit has an extra input from a dummy unit that is fixed to +1 and a weight $w_{0,j}$ for that input



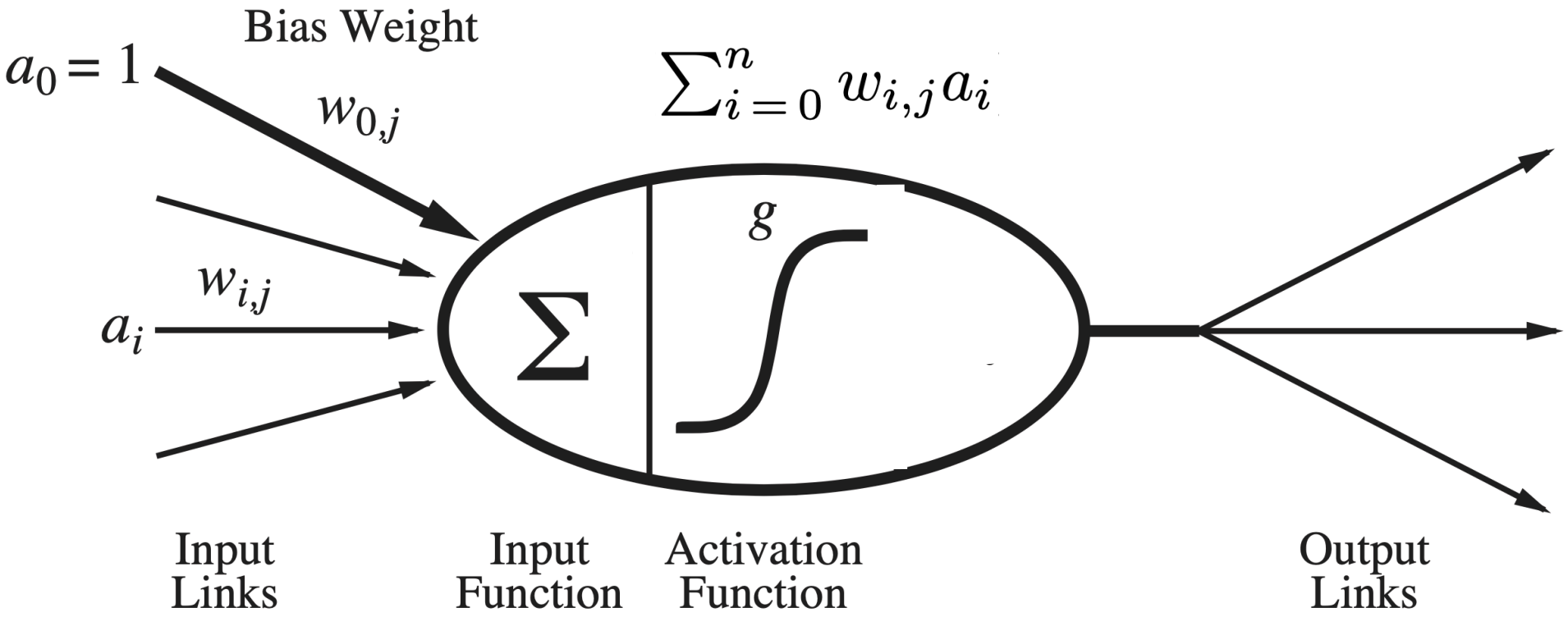
Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**
  - Each unit has an extra input from a dummy unit that is fixed to +1 and a weight $w_{0,j}$ for that input



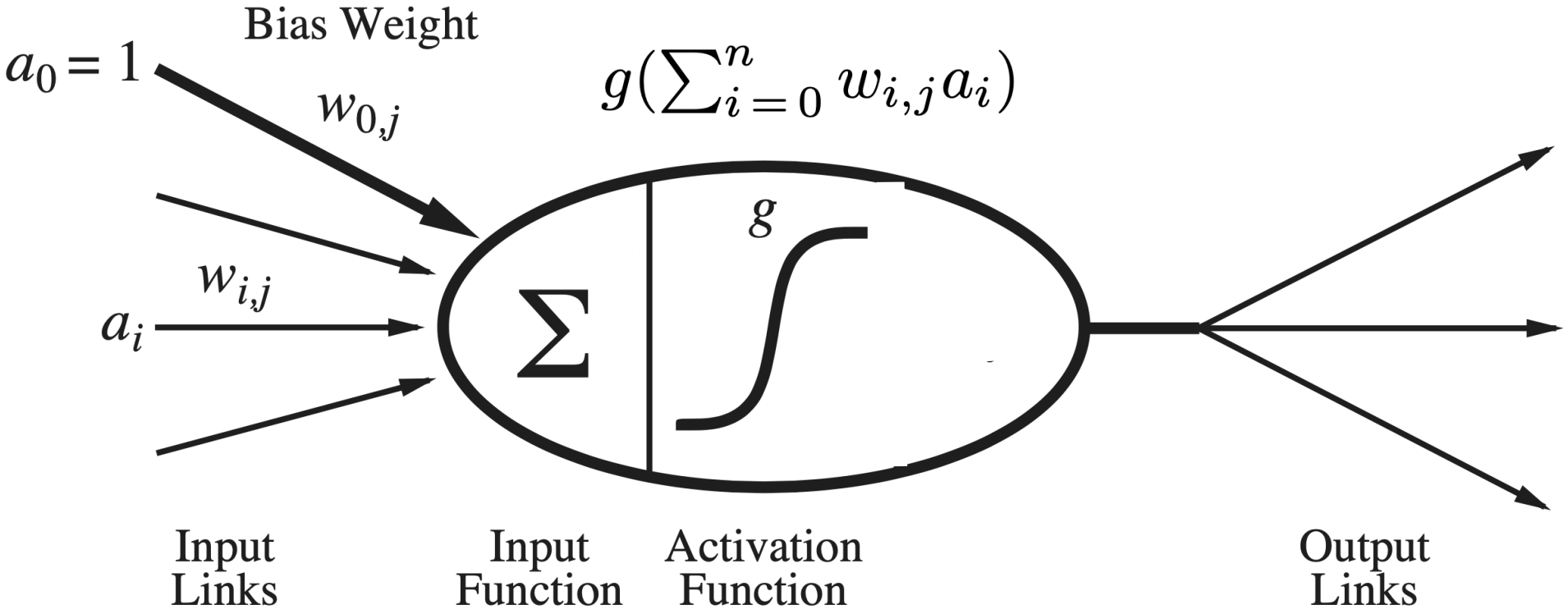
Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**
  - Each unit has an extra input from a dummy unit that is fixed to +1 and a weight $w_{0,j}$ for that input



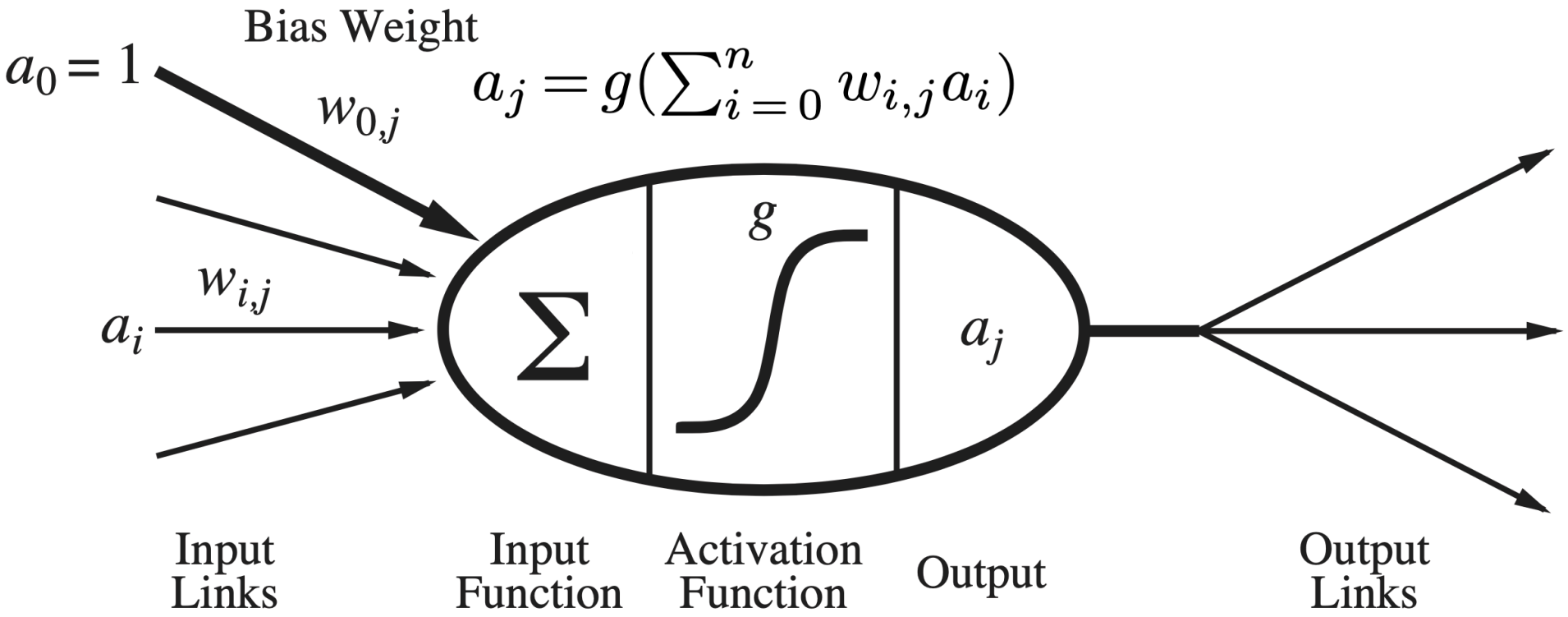
Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**
  - Each unit has an extra input from a dummy unit that is fixed to +1 and a weight $w_{0,j}$ for that input



Artificial Neural Networks (cont.)

- Each node within a network is called a **unit**
  - Each unit has an extra input from a dummy unit that is fixed to +1 and a weight $w_{0,j}$ for that input



Activation Functions

- **Nonlinear** function
  - The nonlinearity is what allows sufficiently large networks of units to represent arbitrary functions
- Universal approximation theorem
  - A network with just two layers of computational units, the first nonlinear and the second linear, can approximate any continuous function to an arbitrary degree of accuracy

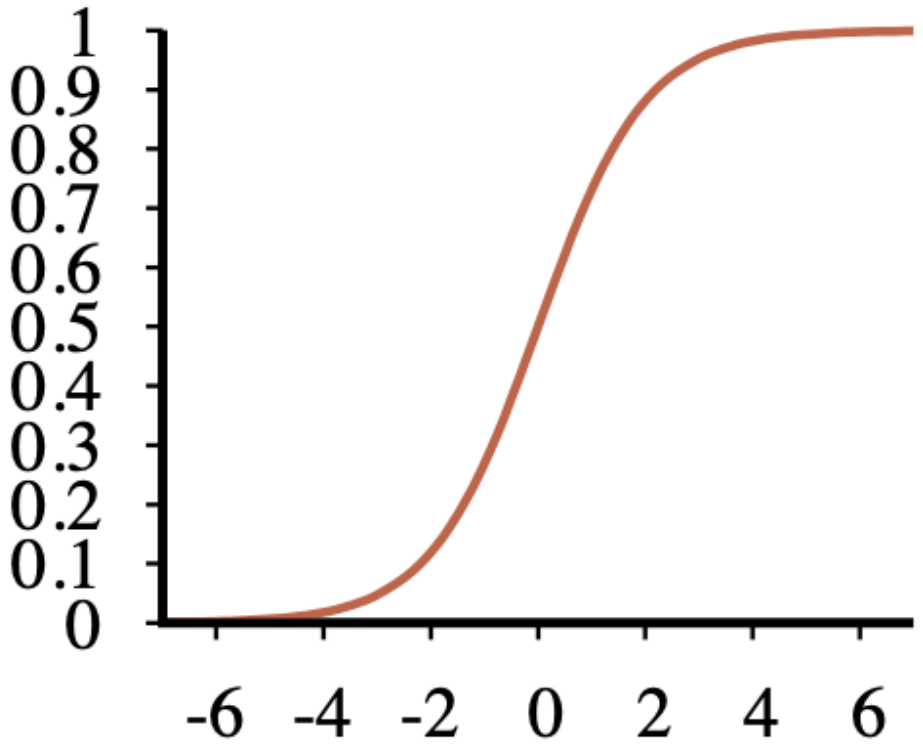
Activation Functions (cont.)

- Logistic (sigmoid) function
- Rectified linear unit (ReLU)
- Softplus function
- tanh function
- …

Activation Functions (cont.)

- Logistic (sigmoid) function

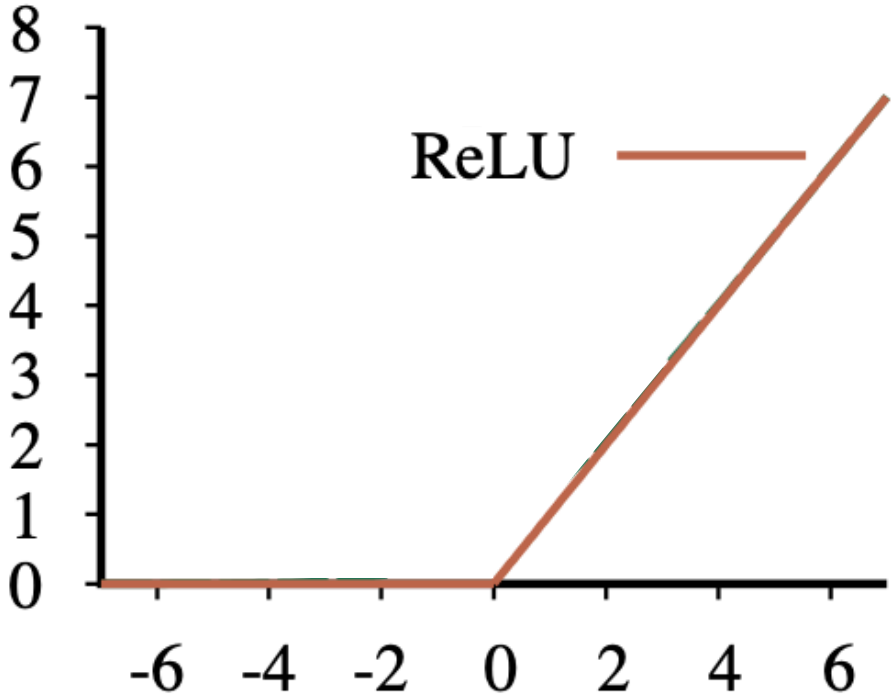
$$\sigma(x) = 1/(1 + e^{-x})$$



Activation Functions (cont.)

- Rectified linear unit (ReLU)

$$\mathrm{ReLU}(x) = \max(0, x)$$

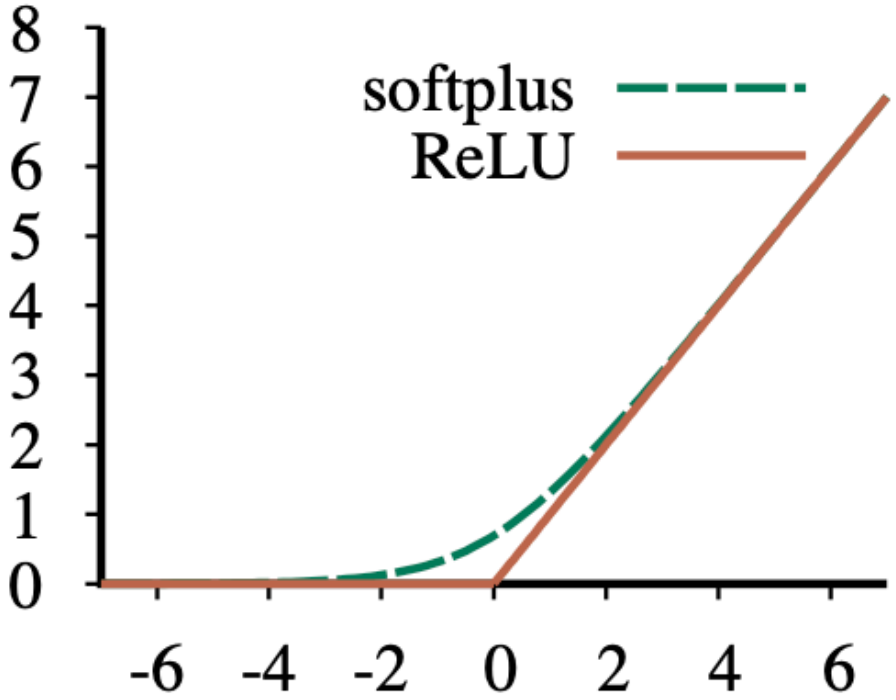


Activation Functions (cont.)

- Softplus function

$$\text{softplus}(x) = \log(1 + e^x)$$

  - A smooth version of ReLU
  - The derivative of the softplus function is the sigmoid function



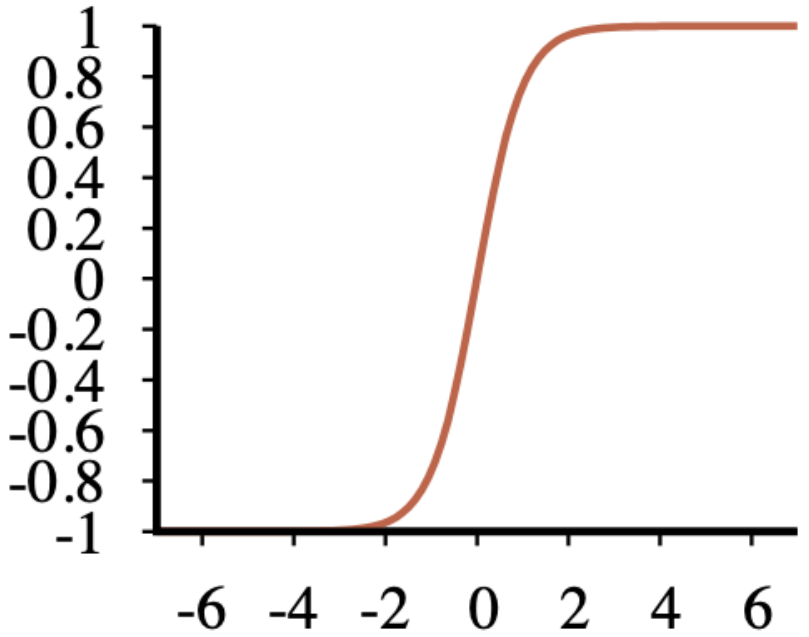
Activation Functions (cont.)

- tanh function

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

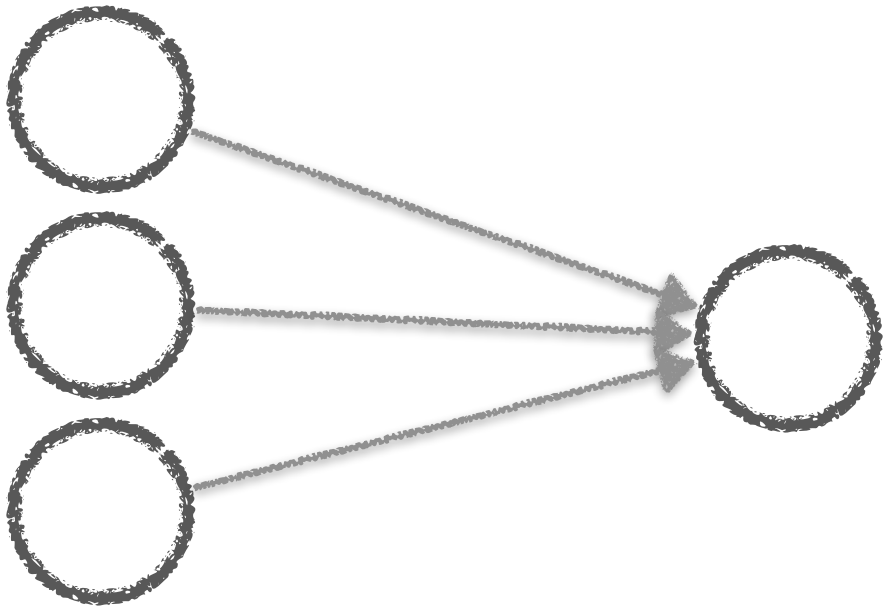
  - Range: (-1,1)
  - A scaled and shifted version of the sigmoid:

$$\tanh(x) = 2\sigma(2x) - 1$$



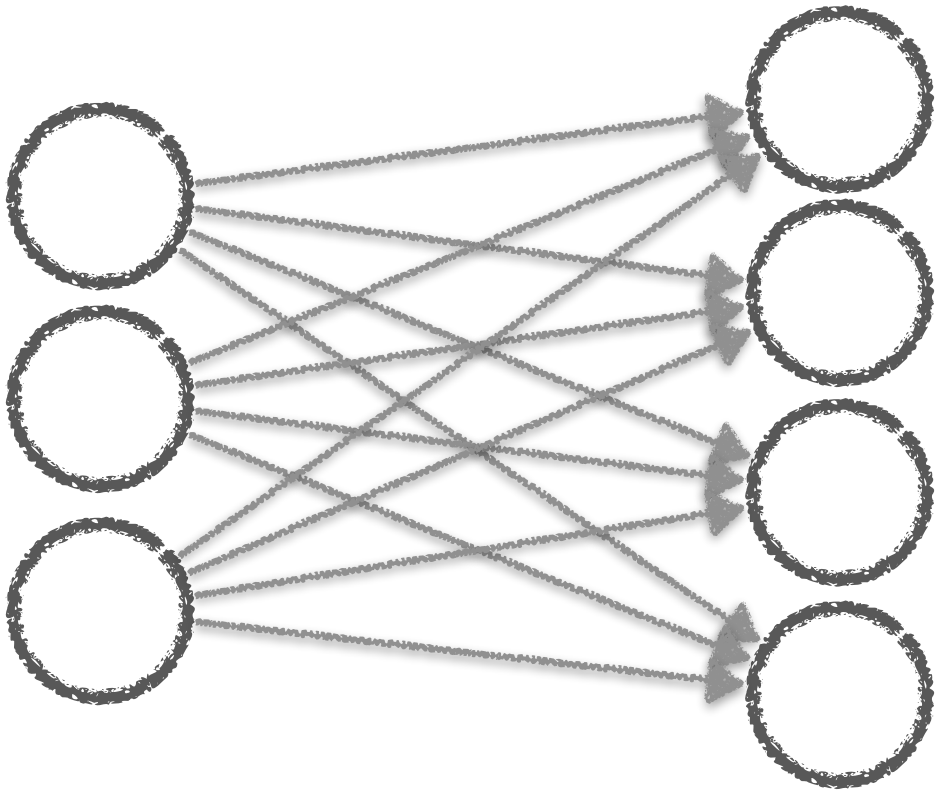
Feedforward Neural Networks

- A feedforward network has connections only in one direction
  i.e., it forms a directed acyclic graph with designated input and output nodes



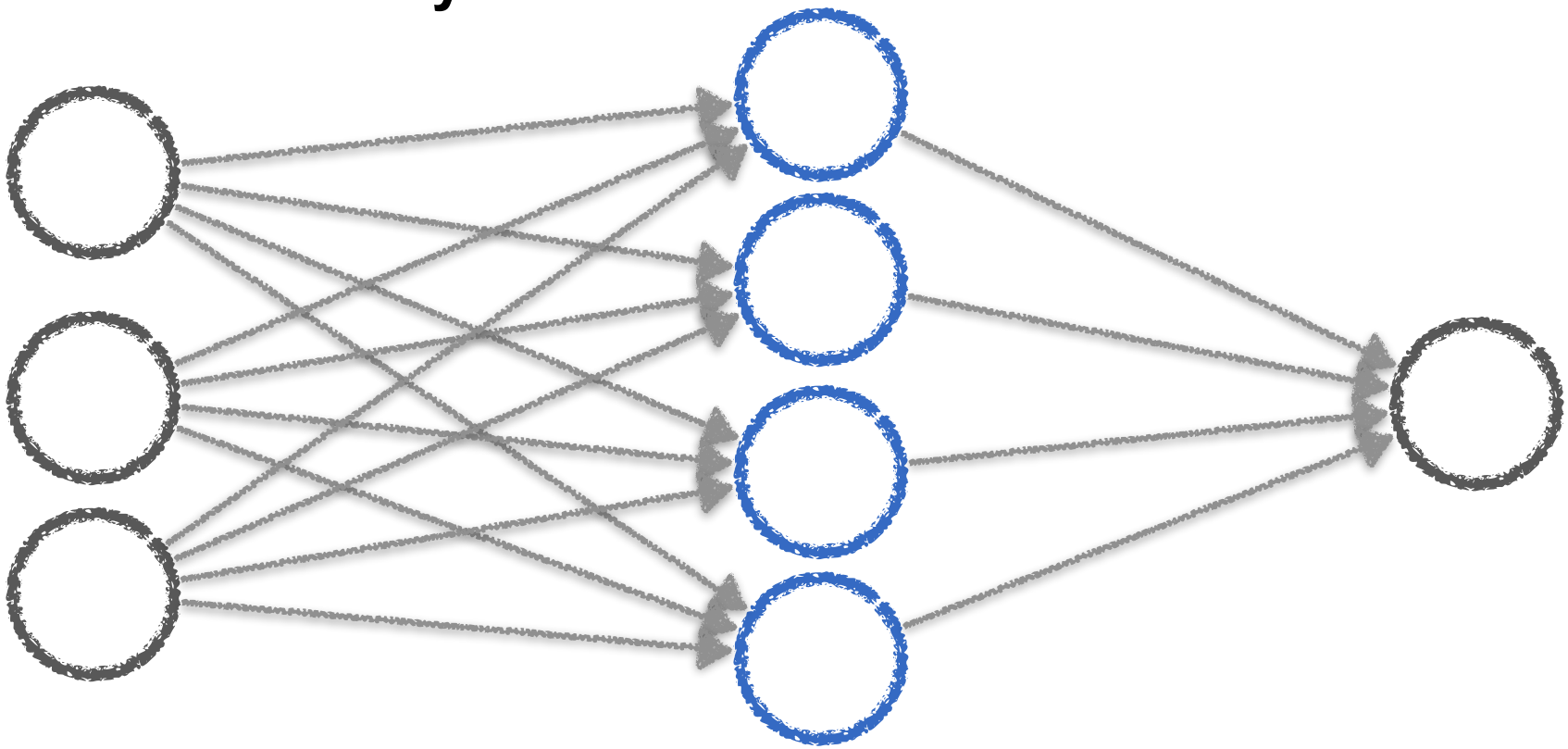
Fully Connected Neural Networks

- Every node in each layer is connected to every node in the next layer



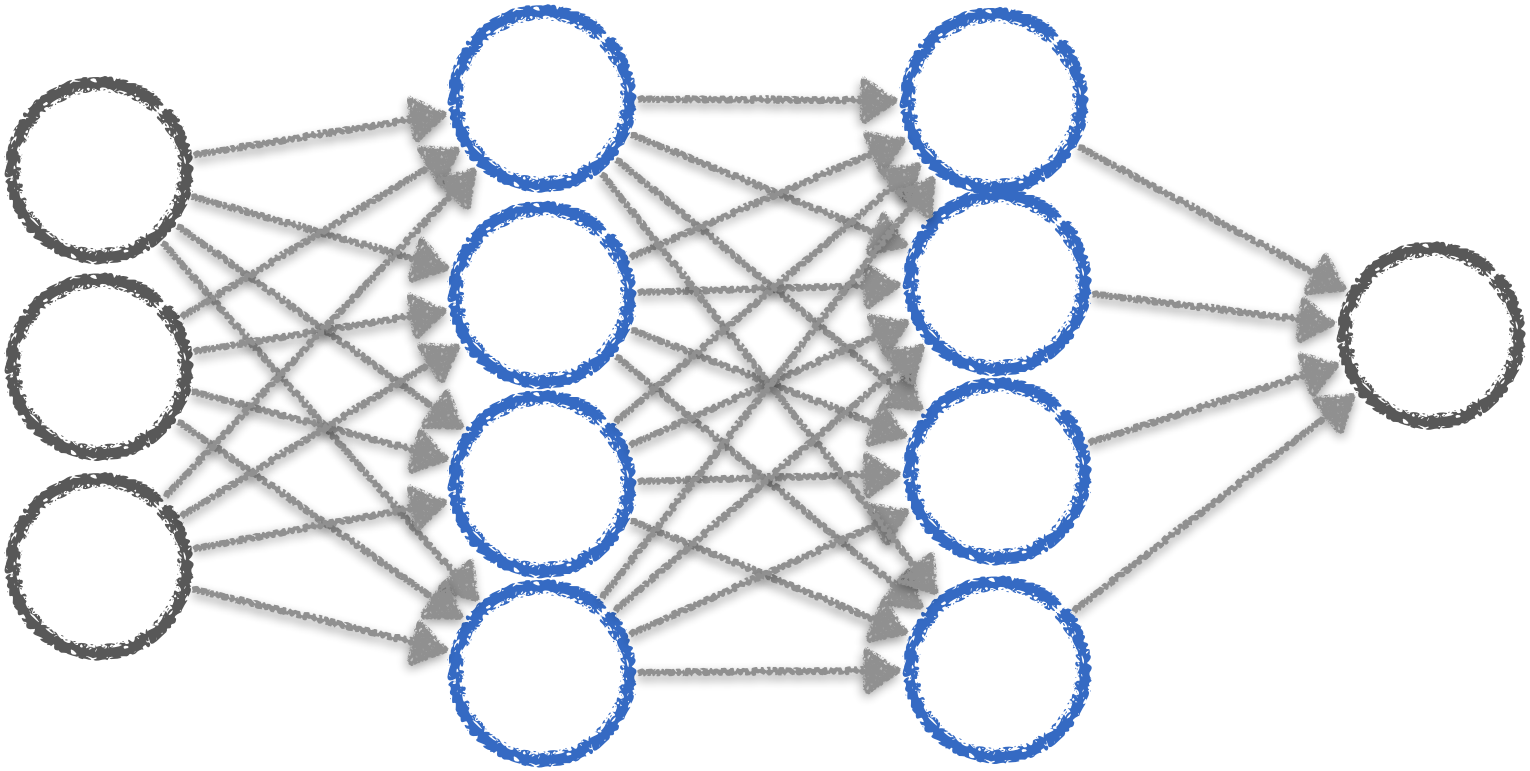
Multilayer Neural Network

- Artificial neural network with an **input layer**, an **output layer**, and at least one **hidden layer**



Deep Neural Networks

- A neural network with **multiple hidden layers**



Deep Learning

- Represent hypotheses as deep neural networks with tunable weights
- Compute the gradient of the loss function with respect to those weights in order to fit the training data

Gradient Descent for Learning Weights in NNs

- Goal: **Minimize loss** when training a neural network
- Processes
  - Start with a random choice of weights
  - Repeat:
    - Calculate the gradient of the loss function based on all data points
    - Adjust the weights along the gradient direction to reduce the loss

Gradient Descent for Learning Weights in NNs (cont.)

- Issue
  - For training, we need the derivative of the loss with respect to weights in early layers of the network
  - But loss is computed only at the very end of the network!
- Solution: back-propagation
  - Idea: backward differentiation
  - Implementation of chain rule of derivatives for efficient computing gradients

Optimization for Learning Weights in NNs

- Stochastic gradient descent (SGD)
- Mini-batch gradient descent
- Momentum
- AdaGrad
- RMSProp
- Adam
- …

Vanishing Gradient Problem

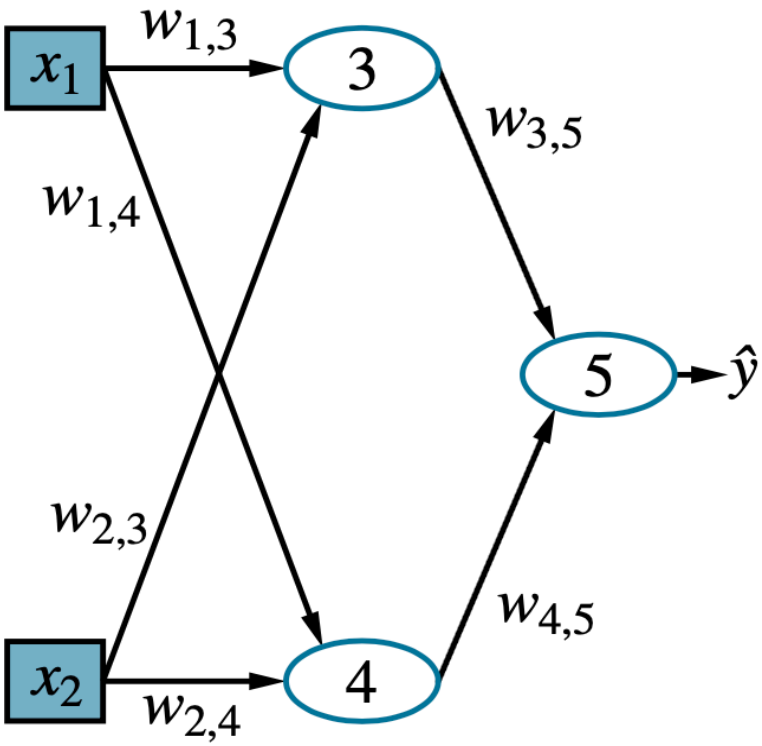
- Deep networks with many layers may suffer from a **vanishing gradient**
  - If the derivative g_j' is small or zero, changing the weights leading into unit *j* will have a negligible effect on its output

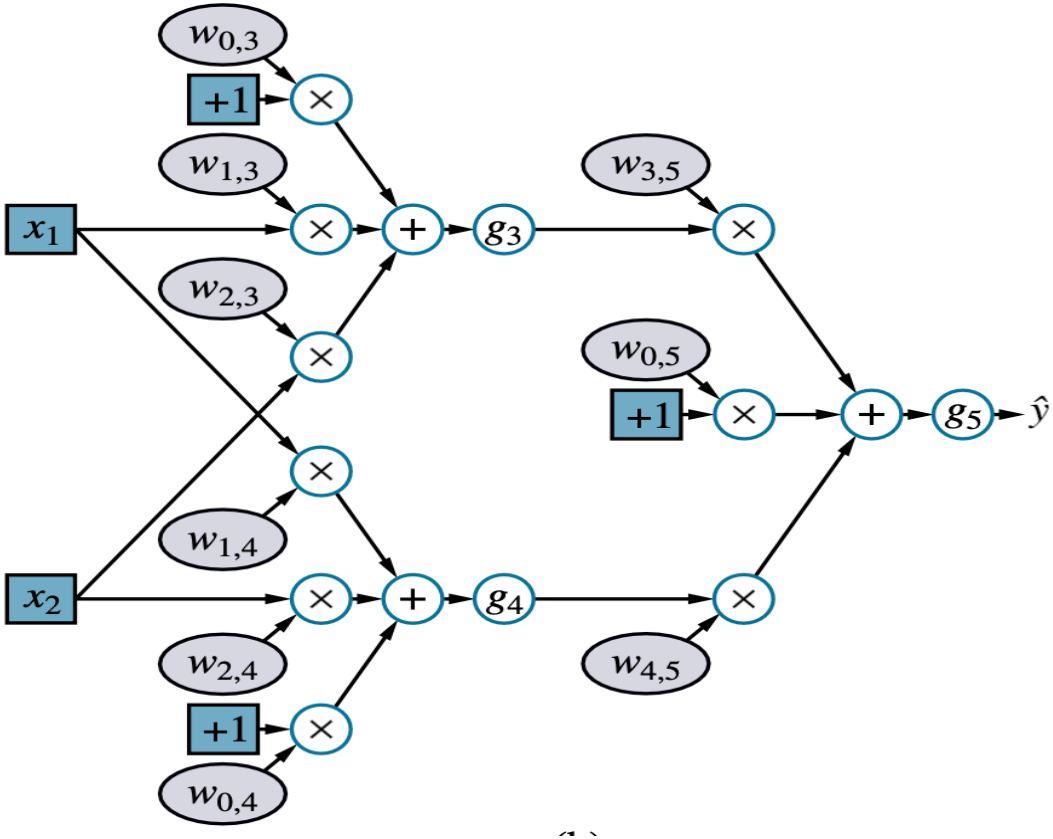
Vanishing Gradient Problem

- Deep networks with many layers may suffer from a **vanishing gradient**
  - If the derivative g_j' is small or zero, changing the weights leading into unit *j* will have a negligible effect on its output
- Solutions
  - ReLu
  - Batch normalization
  - …

Computation Graph

- The network can be unpacked into a computation graph





Computation Graph: Input Encoding

- Boolean (false/true)
  - 0/1 or -1/1
- Numeric
  - Integer or real-valued (or scaled to fit within a fixed range)
- Categorical
  - One-hot encoding, frequency encoding, target encoding etc.

Color	Target
Red	1
Blue	0
Red	0
Green	1

f_Color	Target
0.5	1
0.25	0
0.5	0
0.25	1

t_Color	Target
0.5	1
0	0
0.5	0
1	1

Computation Graph: Output Layer

- Regression problem
  - Target value y is continuous
  - Linear output layer (i.e., $\hat{y}_j = in_j$) without any activation function

Computation Graph: Output Layer (cont.)

- Binary classification
  - A sigmoid output layer
    - One output unit with a sigmoid activation function

Computation Graph: Output Layer (cont.)

- Multiclass classification
  - Goal: output a categorical distribution
    - If there are *d* possible answers, we need *d* output nodes that represent probabilities summing to 1
  - A softmax layer
    - Given a vector of input values **in** =<in_1,..,in_d>, the layer outputs a vector of *d* values
      - The *k*th element of the output vector is given by

$$\text{softmax}(\mathbf{in})_k = \frac{e^{in_k}}{\sum_{k'=1}^{d} e^{in_{k'}}}$$

Loss Functions

- Regression problem
  - e.g, mean squared error (MSE), mean absolute error (MAE), etc.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Loss Functions (cont.)

- Classification problem
  - e.g., cross-entropy:

$$H(P,Q) = -\sum_{x} P(x) log Q(x)$$

    - A measure of dissimilarity between two discrete probability distributions *P* and *Q*, where *P* is the true distribution and *Q* is the approximation
- Example
  - $P(x)$ = [0, 0, 1], $Q(x)$ = [0.1, 0.2, 0.7]
  - $H(P,Q)$ = - 0*log0.1 - 0*log0.2 - 1*log0.7 = 0.36

Overfitting

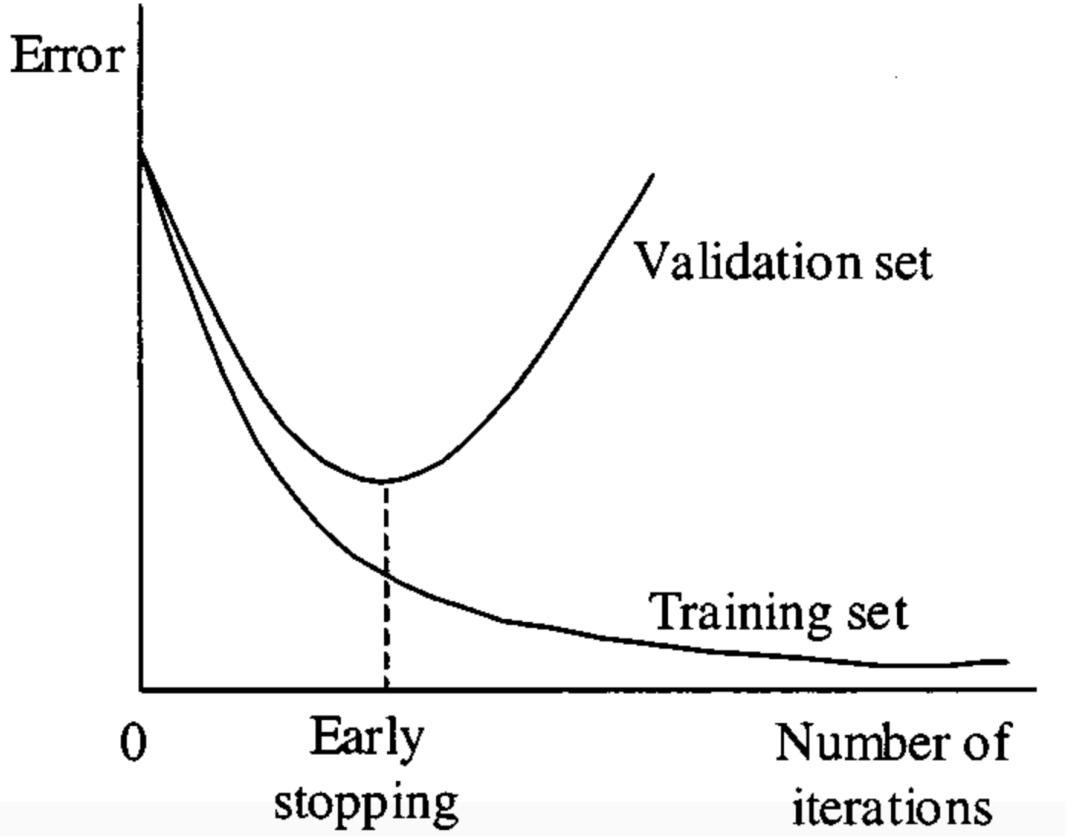
- Regularization
- Early stopping
- Dropout

Overfitting: Regularization

- Weight decay (or weight regularization)
  - Limit the complexity of a model
  - Add a penalty $\lambda \Sigma_{i,j} W_{i,j}^2$ to the loss function used to train the neural network
    - λ: a hyperparameter controlling the strength of the penalty
      - $\lambda = 10^{-4}$

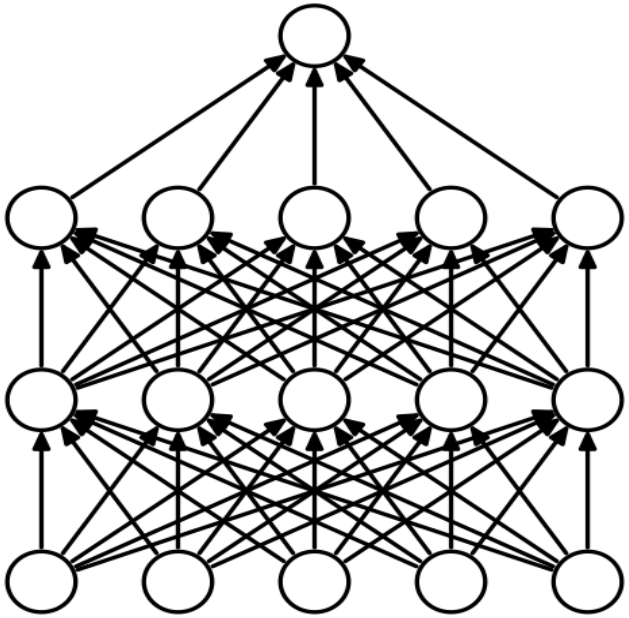
Overfitting: Early stopping

- Stop training when a monitored metric has stopped improving

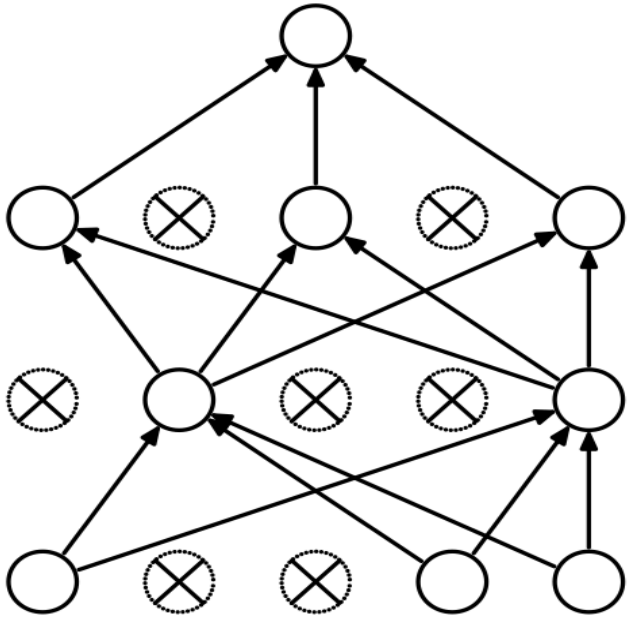


Overfitting: Dropout

- In training: temporarily removing units, selected at random, from a neural network to prevent over-reliance on certain units
- At test time, the model is run with no dropout



(a) Standard Neural Net



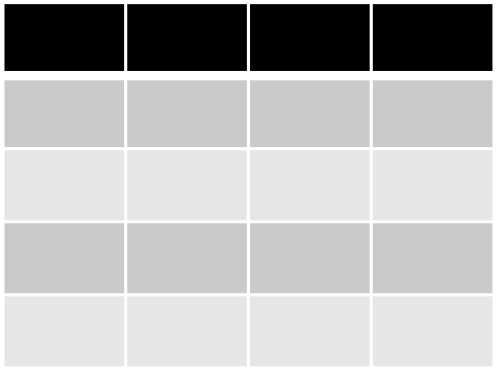
(b) After applying dropout.

rce: http://jmlr.org/papers/v15/srivastava14a.html

Overfitting: Dropout (cont.)

- Dropout algorithm
  - For every node in the network:
    - With probability p, the unit output is multiplied by a factor of $1/p$
    - Otherwise, the unit output is fixed at zero
- Hyperparameter
  - Hidden layers
    - p = 0.5 ~ 0.8, i.e.,dropout rate = 0.2 ~ 0.5
  - Input layer
    - p = 0.8, i.e., dropout rate = 0.2

Structured Data



https://media.istockphoto.com/id/1367766076/vector/database-storage-icon-vector-for-your-website-design-logo-app-ui-illustration.jpg?s=612x612&w=0&k=20&c=5-nzlAxbGQMw-lJcZc_P9QbHWXGb00ZLimhhwtr9rnc=

Unstructured Data

今天天氣很好

AAAGTCTGAC

Computer Vision

- Computational methods for analyzing and understanding digital images

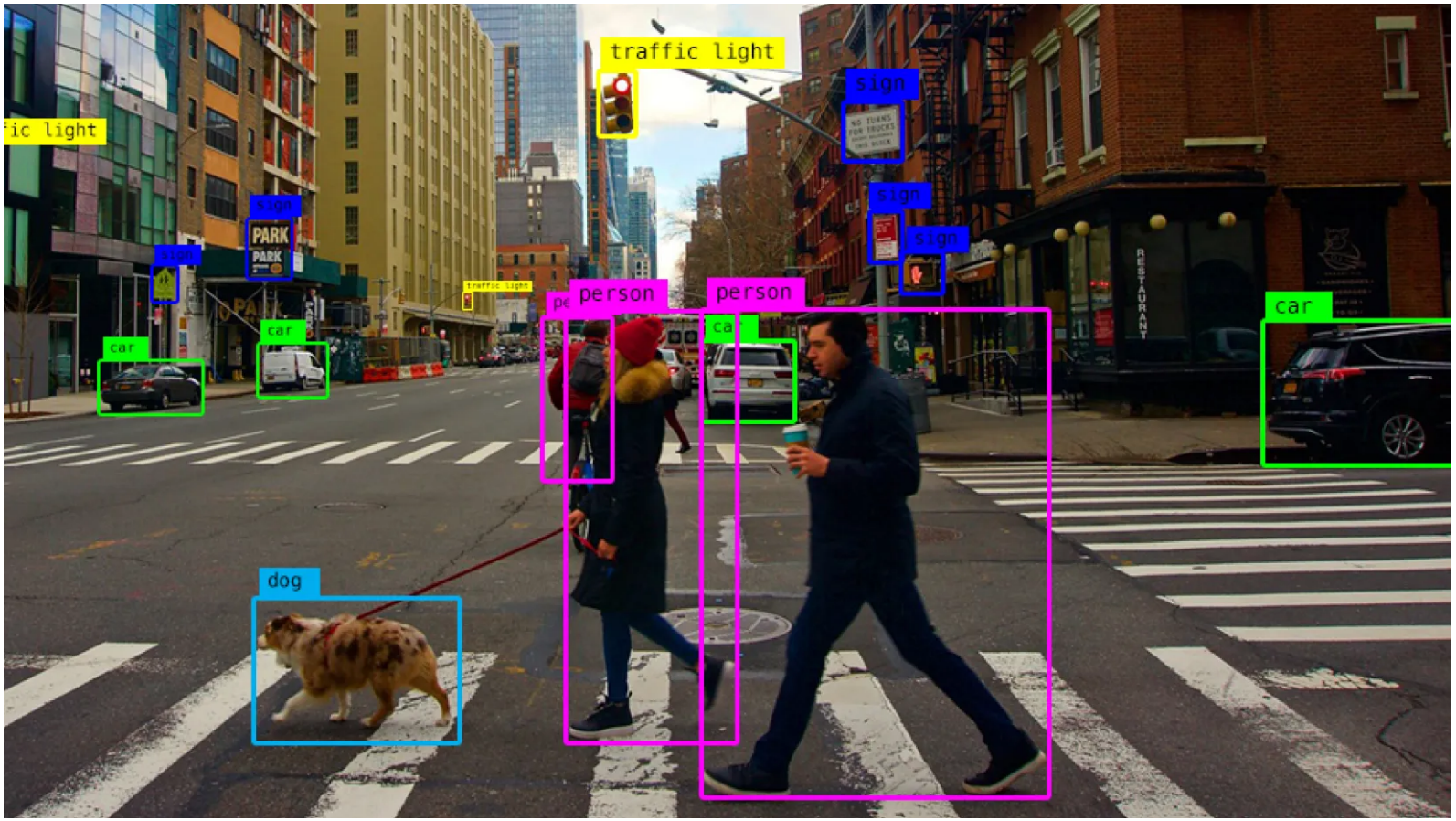
Applications in Computer Vision

- Image classification
- Object detection
- Facial recognition
- Image segmentation
- Medical imaging
- …

Applications: Image Classification

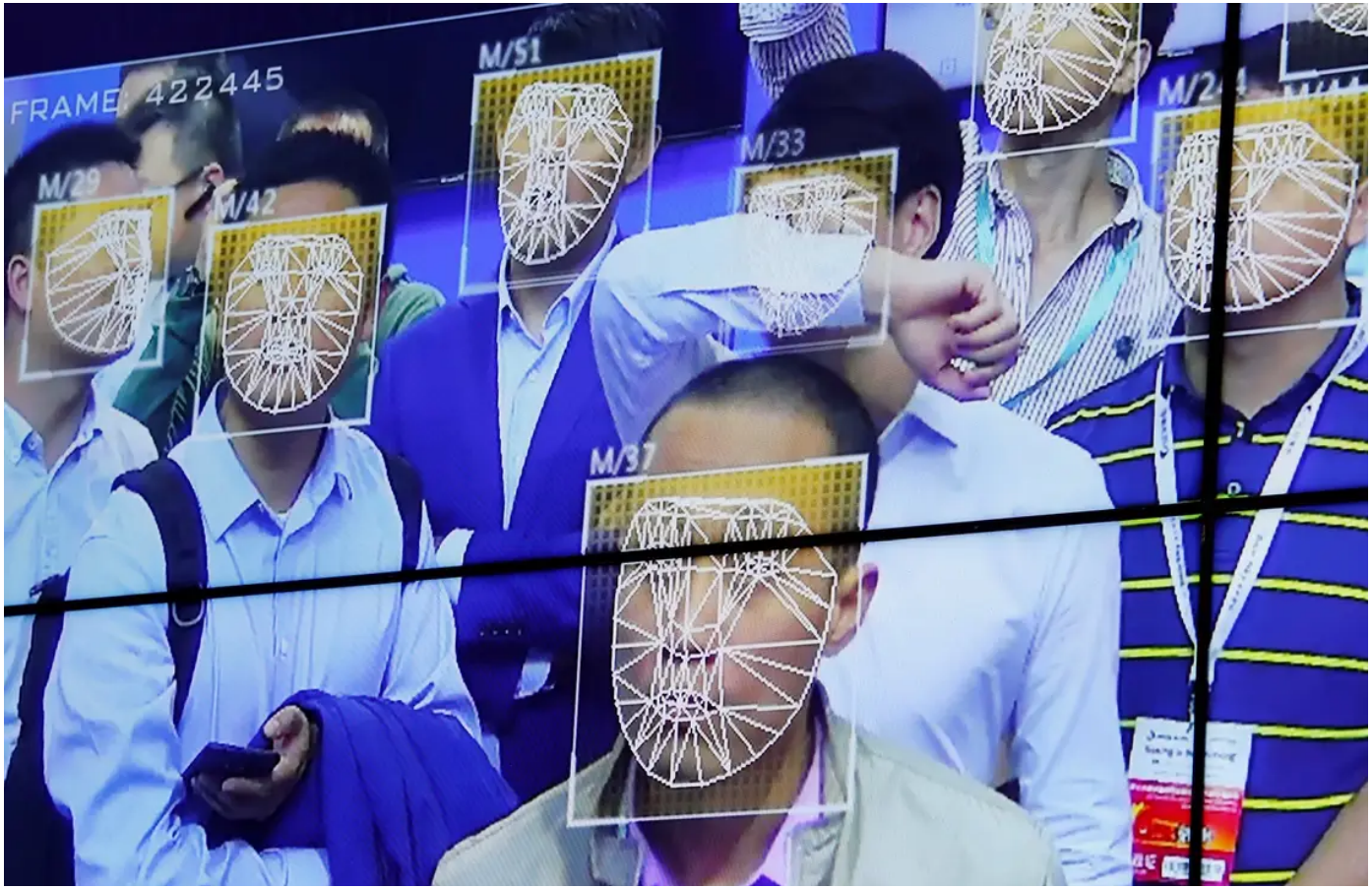
https://cdn2.ettoday.net/images/5557/d5557884.jpg

Applications: Object Detection



https://kajabi-storefronts-production.kajabi-cdn.com/kajabi-storefronts-production/file-uploads/blogs/22606/images/1446e76-f181-6047-4e73-8d8ba3c6a50e_object_detection_1.webp

Applications: Facial Recognition

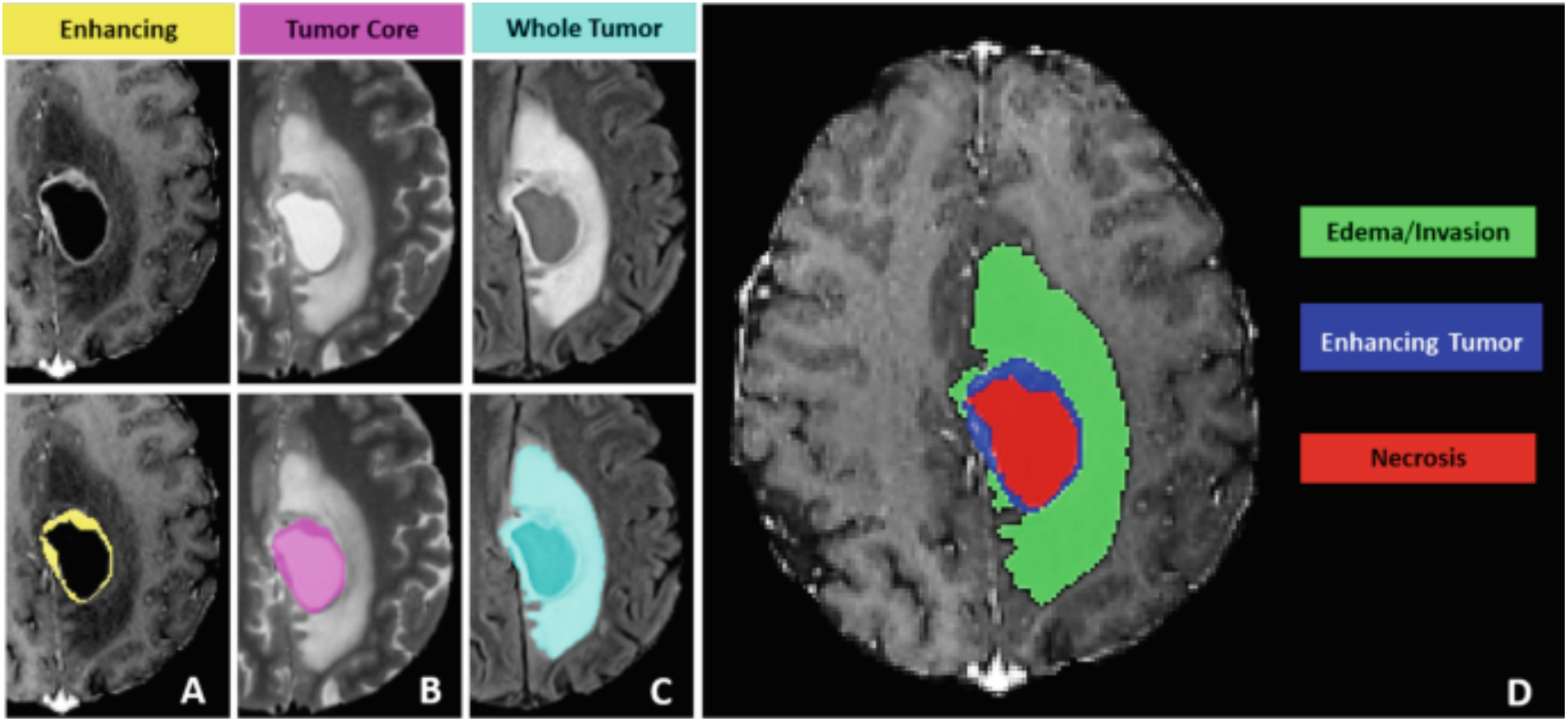


https://www.businessinsider.com/how-china-uses-facial-recognition-technology-surveillance-2018-2

Applications: Image Segmentation

https://miro.medium.com/v2/resize:fit:1400/format:webp/1*NO2OpFhjssG82z3sSrsdLQ.jpeg

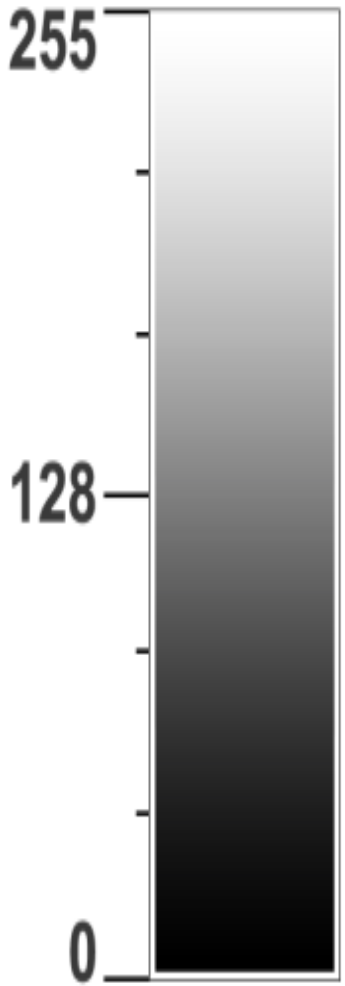
Applications: Medical Imaging



https://media.springernature.com/lw685/springer-static/image/chp%3A10.1007%2F978-1-0716-3195-9_13/MediaObjects/515045_1_En_13_Fig16_HTML.png

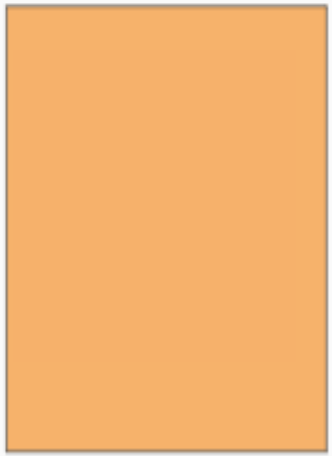
https://datasets.activeloop.ai/wp-content/uploads/2019/12/MNIST-handwritten-digits-dataset-visualized-by-Activeloop.webp

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0



https://www.analyticsvidhya.com/blog/2021/03/grayscale-and-rgb-format-for-storing-images/

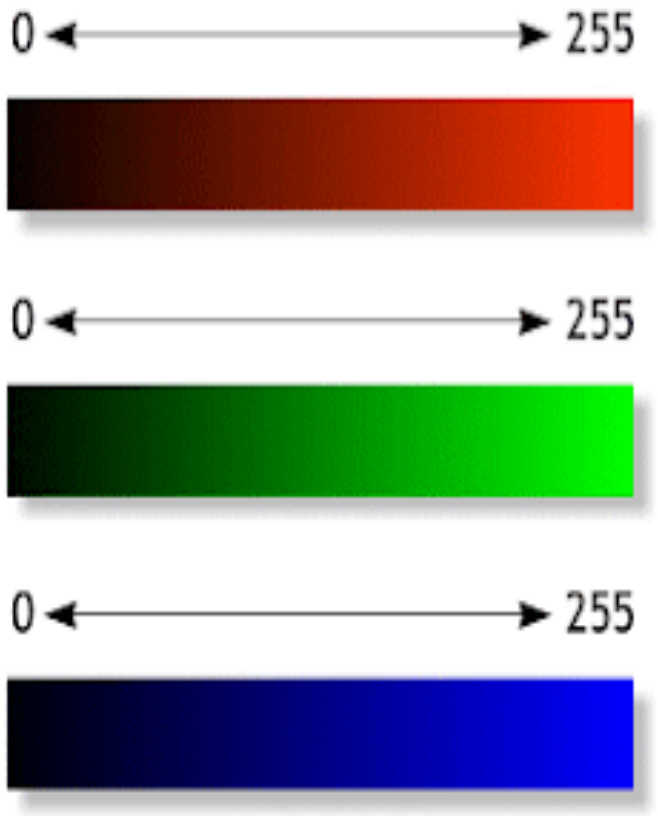
https://product.corel.com/help/Common/CDGS/540223850/Shared/CorelDRAW-06-color-manage-grayscale.png



https://medium.com/data-science-bootcamp/convolutional-neural-networks-cnn-and-computer-vision-101-fundamentals-part-1-a5d39a1faf8f

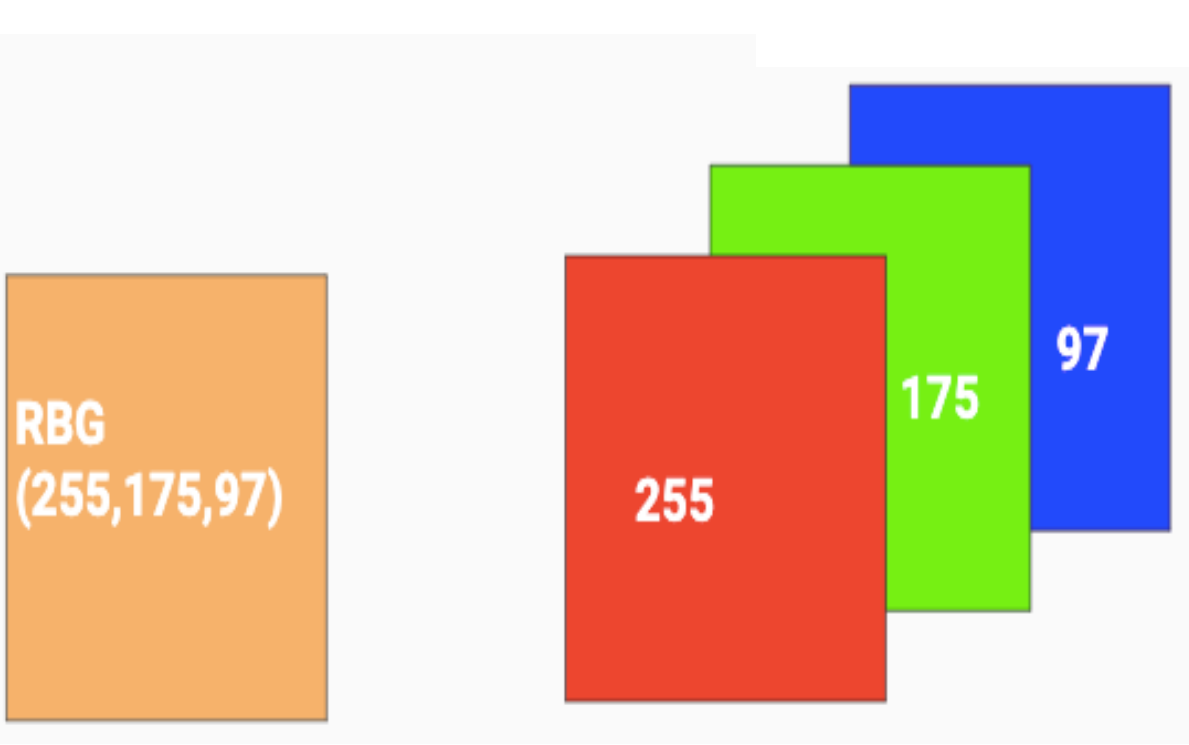
https://lh6.googleusercontent.com/proxy/fijp19Xlfs-Z39kElcW3LAdX-Q41yhSfr8bRBhAEDFs5aT70LVL0Z0wPsUbTeKzvipGfHWcub-0wZhOmuvY-FI26MuoV7O9vobh7vDVaXFE

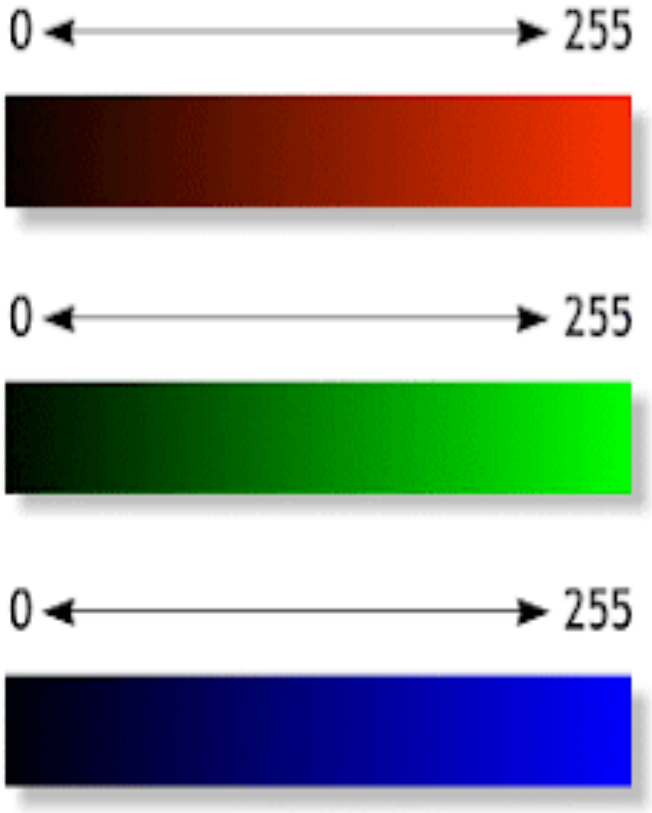
RBG
(255,175,97)



https://medium.com/data-science-bootcamp/convolutional-neural-networks-cnn-and-computer-vision-101-fundamentals-part-1-a5d39a1faf8f

https://lh6.googleusercontent.com/proxy/fijp19Xlfs-Z39kElcW3LAdX-Q41yhSfr8bRBhAEDFs5aT70LVL0Z0wPsUbTeKzvipGfHWcub-0wZhOmuvY-FI26MuoV7O9vobh7vDVaXFE

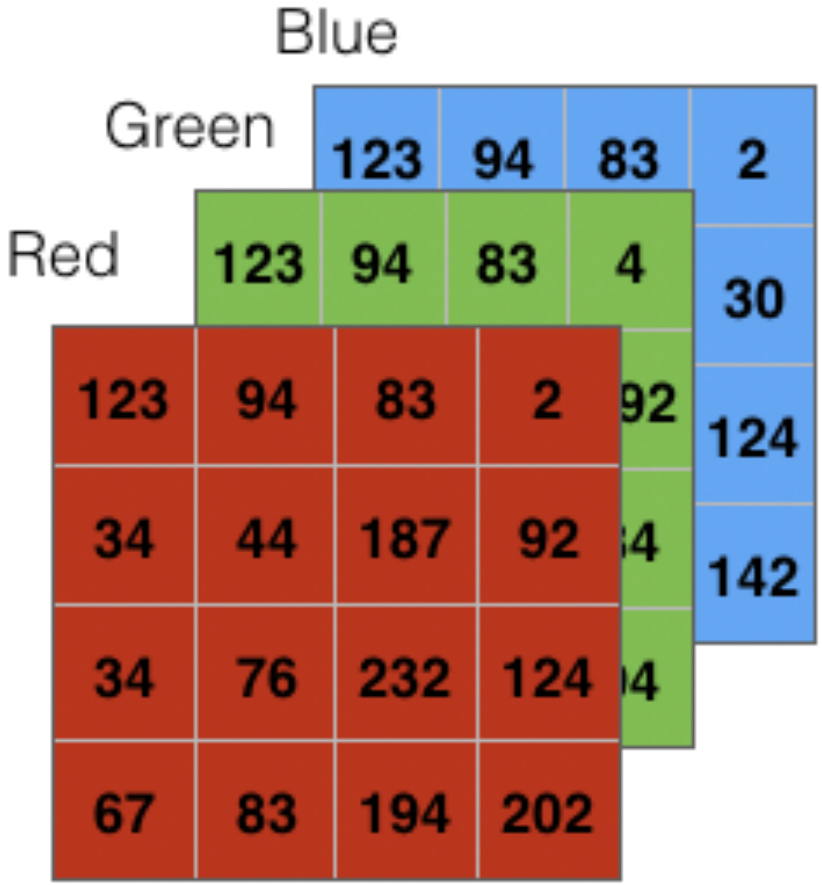




https://medium.com/data-science-bootcamp/convolutional-neural-networks-cnn-and-computer-vision-101-fundamentals-part-1-a5d39a1faf8f

https://lh6.googleusercontent.com/proxy/fijp19Xlfs-Z39kElcW3LAdX-Q41yhSfr8bRBhAEDFs5aT70LVL0Z0wPsUbTeKzvipGfHWcub-0wZhOmuvY-FI26MuoV7O9vobh7vDVaXFE

=



https://images2017.cnblogs.com/blog/864046/201711/864046-20171120230939524-128824376.png

Convolutional Neural Networks (CNNs)

Observation I

- Adjacency of pixels really matters
  - An image cannot be thought of as simple vector of input pixel values
- Performance issue
  - Fully connected network:
    - The input: n pixels
    - The first hidden layer: n units
    - Require n^2 weights
  - A vast parameter space + vast # of training images

Idea I

- Each unit in the first hidden layer receives input from only a **small**, **local region** of the image

Observation II

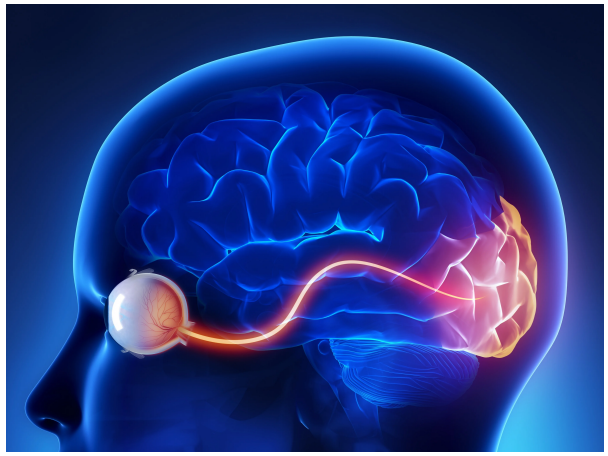
- Approximate **spatial invariance**

https://i.lmpm.com/img/lmpm-company-store-prod/6565f5c5059bb/properties/e67e660b-f317-42b3-8a75-f29bd094f14a/30farhorizondrlp-7.jpg?w=2048&h=1152&q=60

Idea II

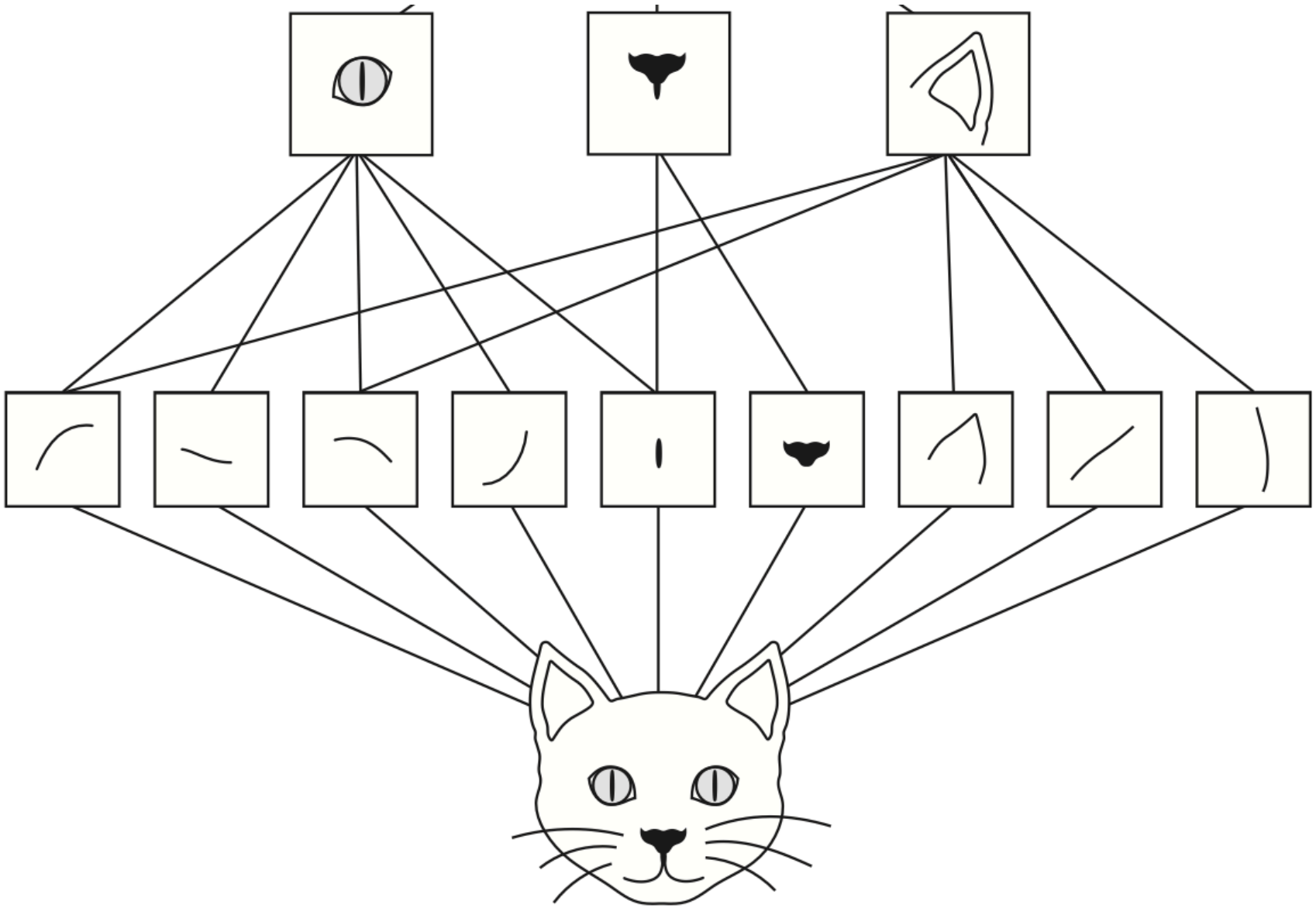
- Local spatial invariance
  - Constrain the weights connecting a local region to a unit in the hidden layer to be the same for each hidden unit

Observation III



- In the primary visual cortex
  - A simple cell
    - Respond primarily to oriented edges and gratings (bars of particular orientations)
  - A complex cell
    - Receive inputs from a number of simple cells
    - A summation and integration of the receptive fields of many input simple cells
    - Respond to patterns of light in a certain orientation within a large receptive field

https://d2jx2rerrg6sh3.cloudfront.net/image-handler/picture/2021/10/shutterstock_166610156-1.jpg



Idea III

- The first hidden layer detects many kinds of features, not just one
  - Make the hidden units into feature detectors that detect the same feature wherever it appear in the image

Convolutional Neural Networks (CNNs)

- Properties
  - Contains **spatially local** connections, at least in the early layers
  - **Kernel**/Filter
    - A pattern of weights is replicated across multiple local regions
  - **Convolution**
    - The process of applying the kernel to the pixels of the image
  - Pooling
    - Summarize a set of adjacent units from the preceding layer

Image Convolution

- A filter that adds each pixel value of an image to its neighbors, weighted according to a kernel matrix
- Example
  - One-dimensional convolution operation with kernel [+1,-1,+1]

Image Convolution

- A filter that adds each pixel value of an image to its neighbors, weighted according to a kernel matrix
- Example
  - One-dimensional convolution operation with kernel [+1,-1,+1]

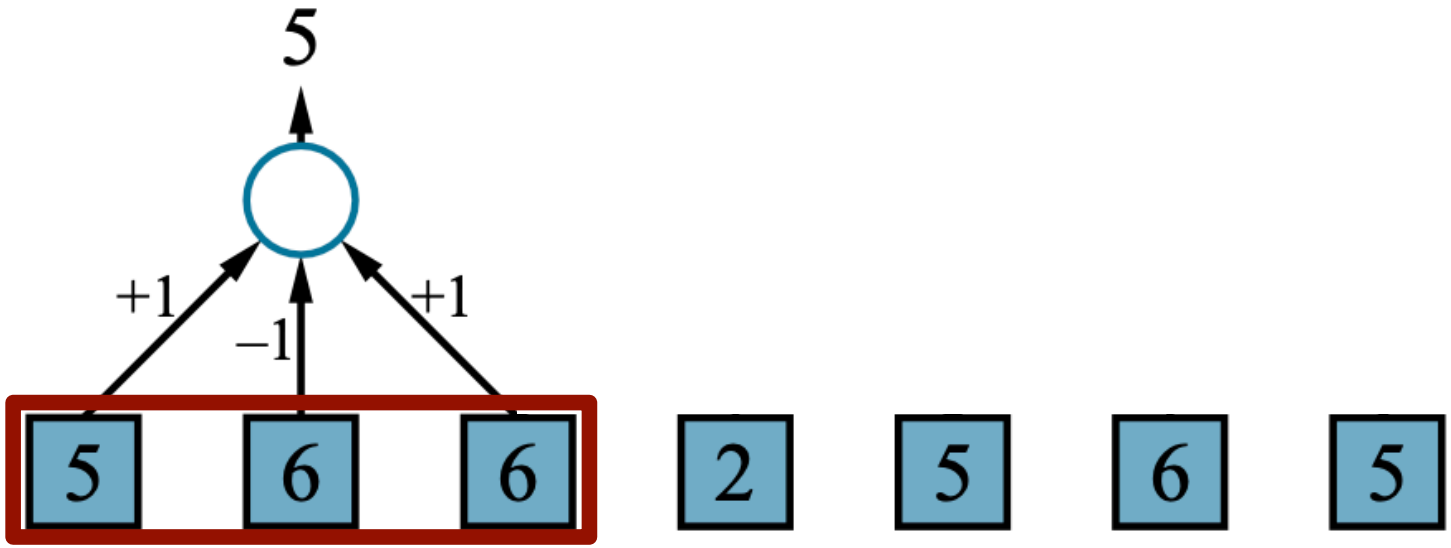


Image Convolution

- A filter that adds each pixel value of an image to its neighbors, weighted according to a kernel matrix
- Example
  - One-dimensional convolution operation with kernel [+1,-1,+1] and stride = 2

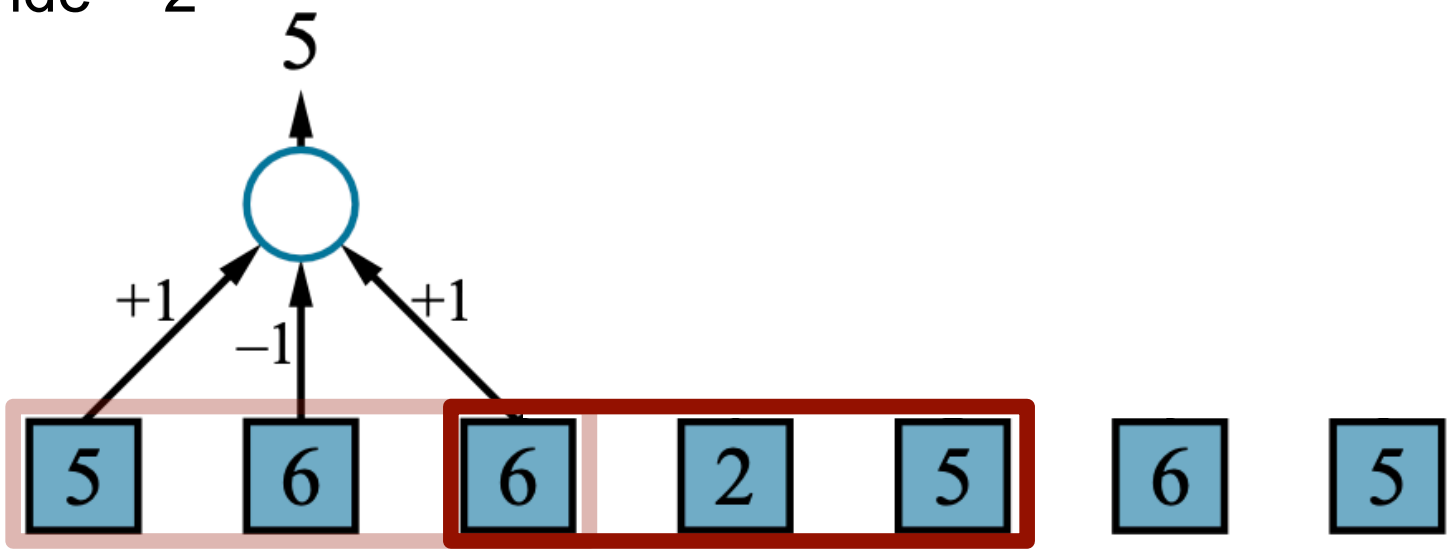


Image Convolution

- A filter that adds each pixel value of an image to its neighbors, weighted according to a kernel matrix
- Example
  - One-dimensional convolution operation with kernel [+1,-1,+1] and stride = 2

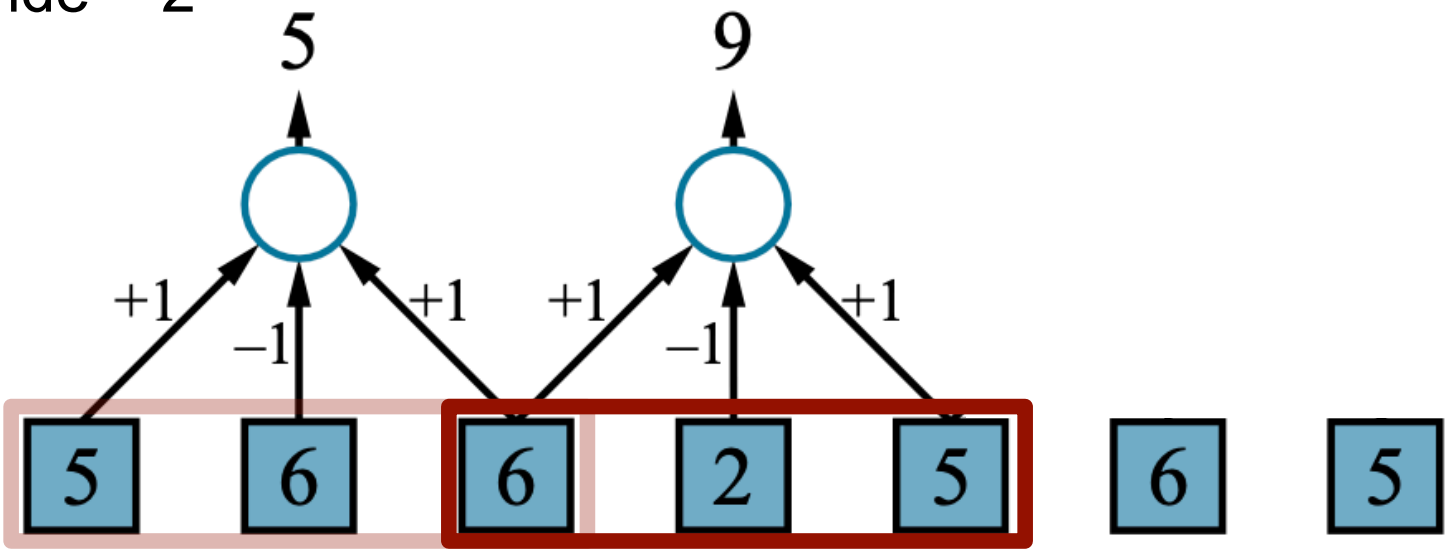


Image Convolution

- A filter that adds each pixel value of an image to its neighbors, weighted according to a kernel matrix
- Example
  - One-dimensional convolution operation with kernel [+1,-1,+1] and stride = 2

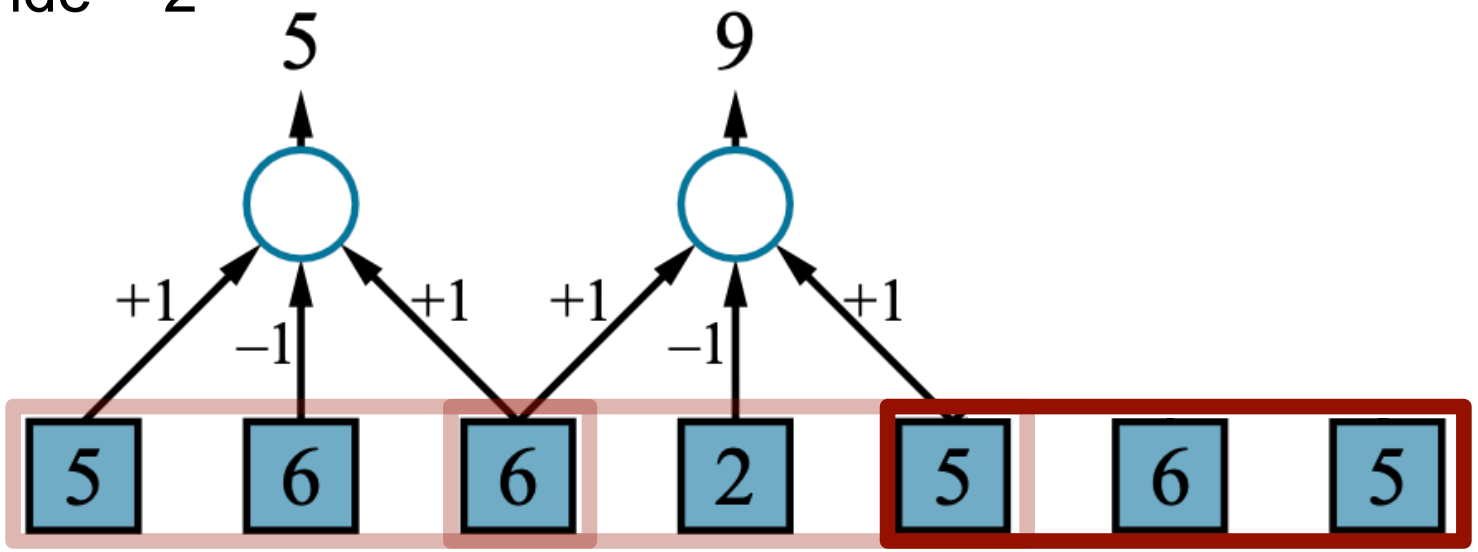
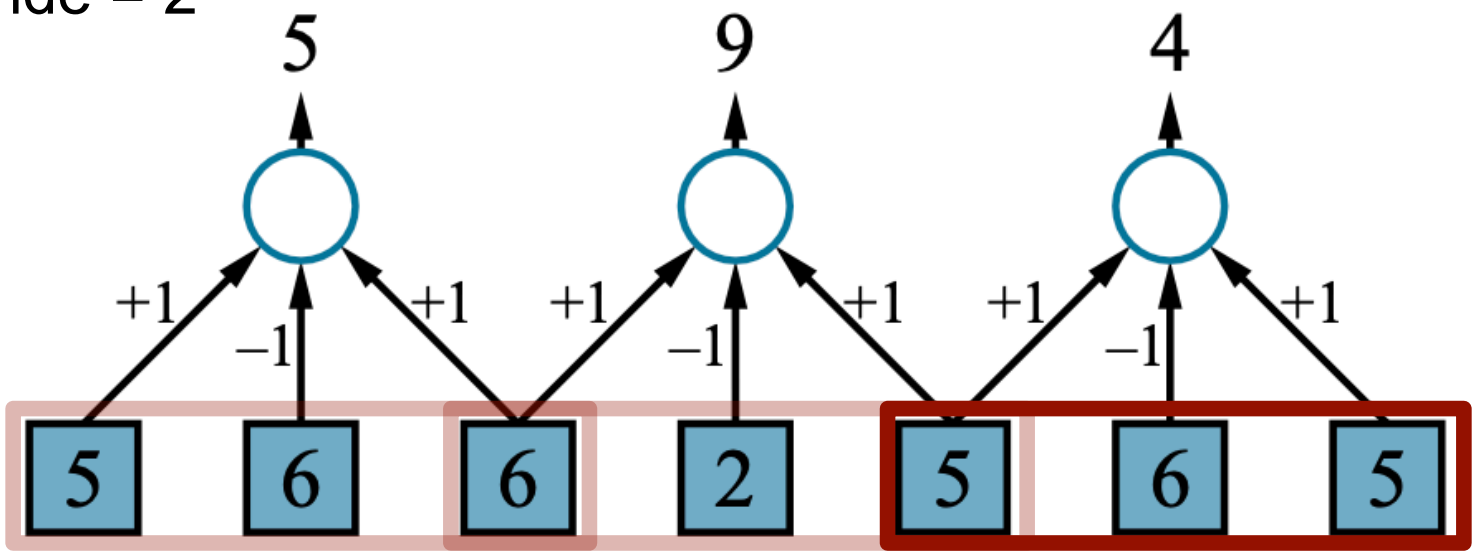
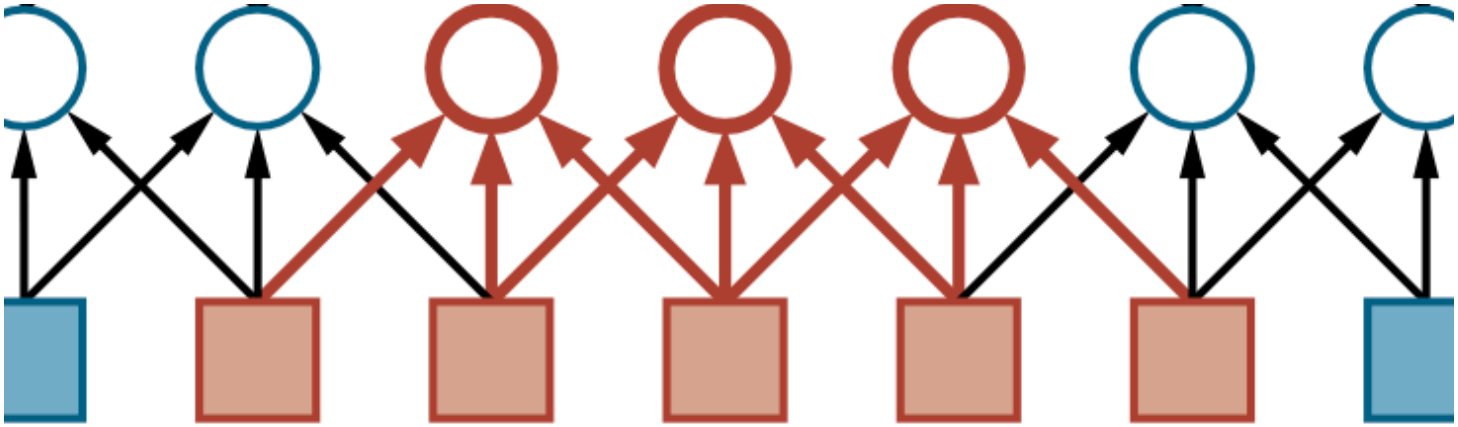


Image Convolution

- A filter that adds each pixel value of an image to its neighbors, weighted according to a kernel matrix
- Example
  - One-dimensional convolution operation with kernel [+1,-1,+1] and stride = 2

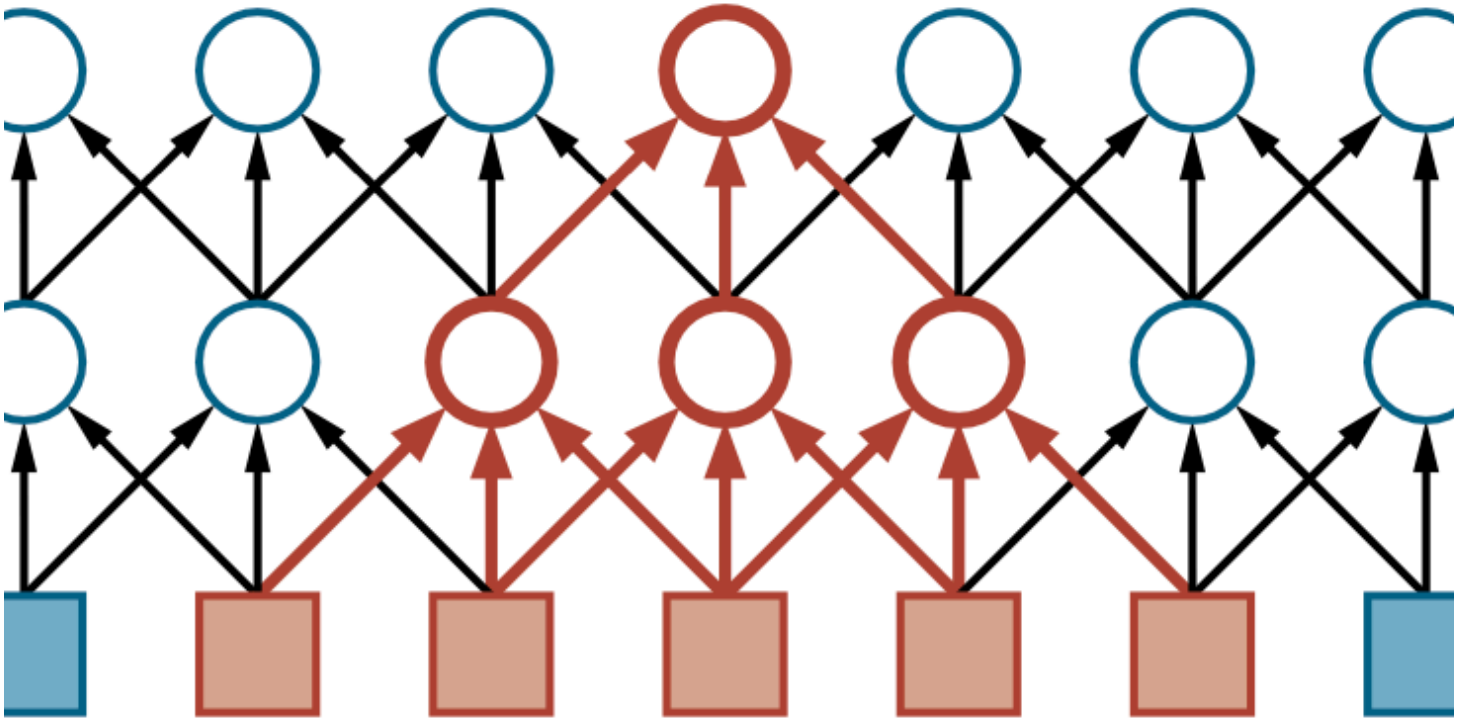


Characteristics of CNNs

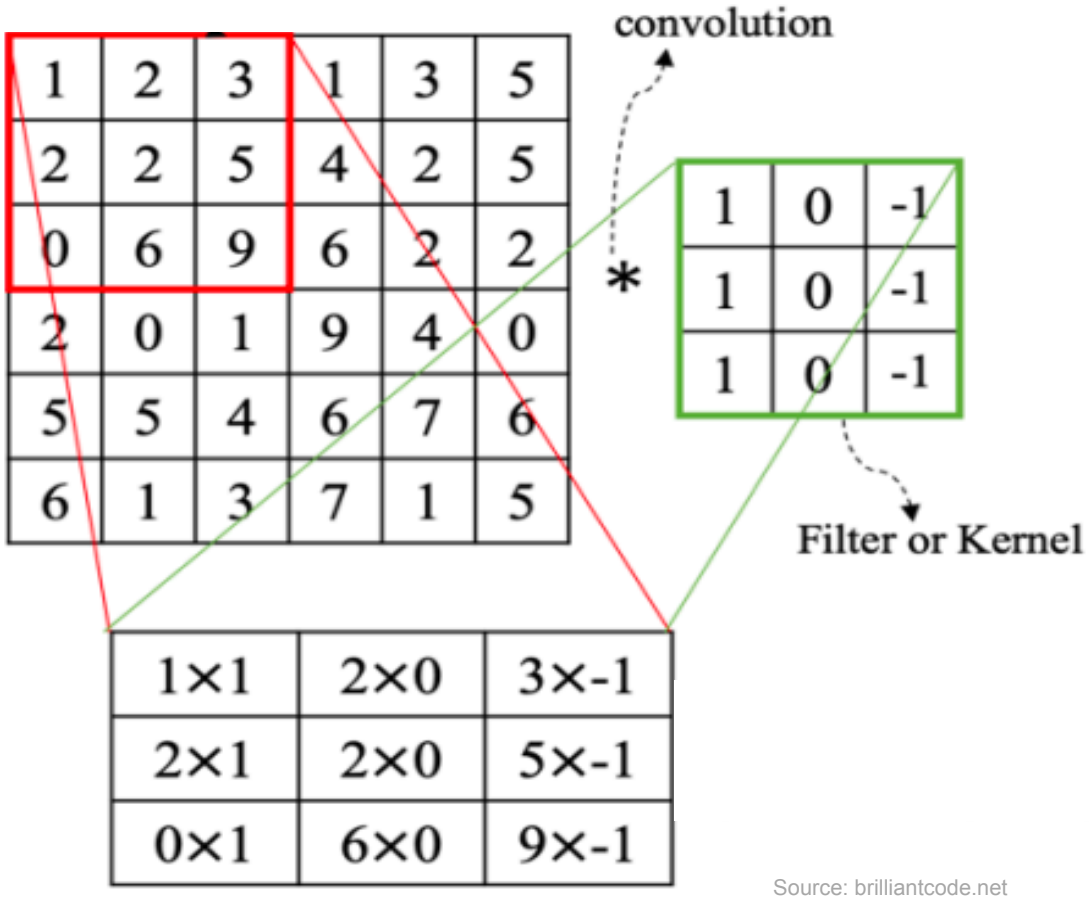


Characteristics of CNNs

- The deeper the unit, the larger the receptive field

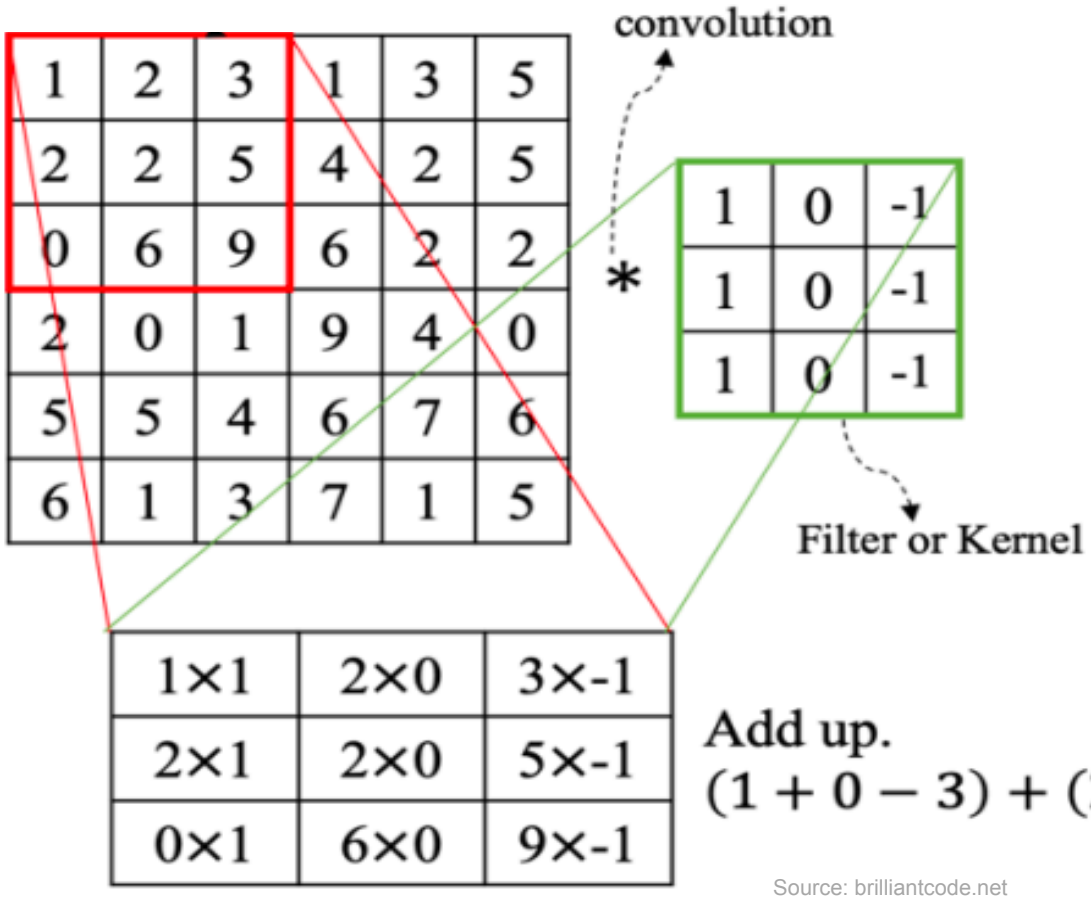


Example: Two-Dimensional Convolution



Source: brilliantcode.net

Example: Two-Dimensional Convolution



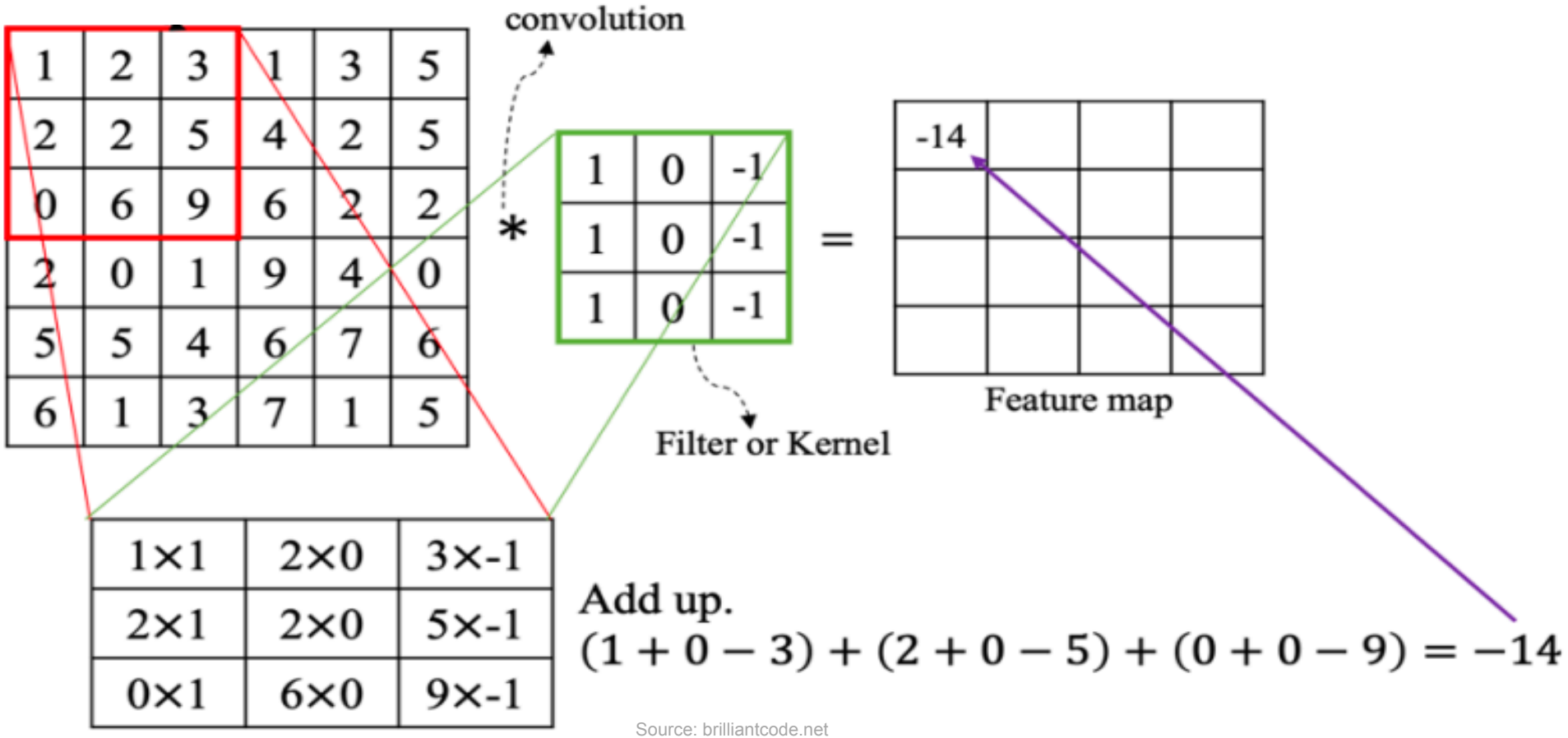
1	2	3	1	3	5
2	2	5	4	2	5
0	6	9	6	2	2
2	0	1	9	4	0
5	5	4	6	7	6
6	1	3	7	1	5

1	0	-1
1	0	-1
1	0	-1

1×1	2×0	3×-1
2×1	2×0	5×-1
0×1	6×0	9×-1

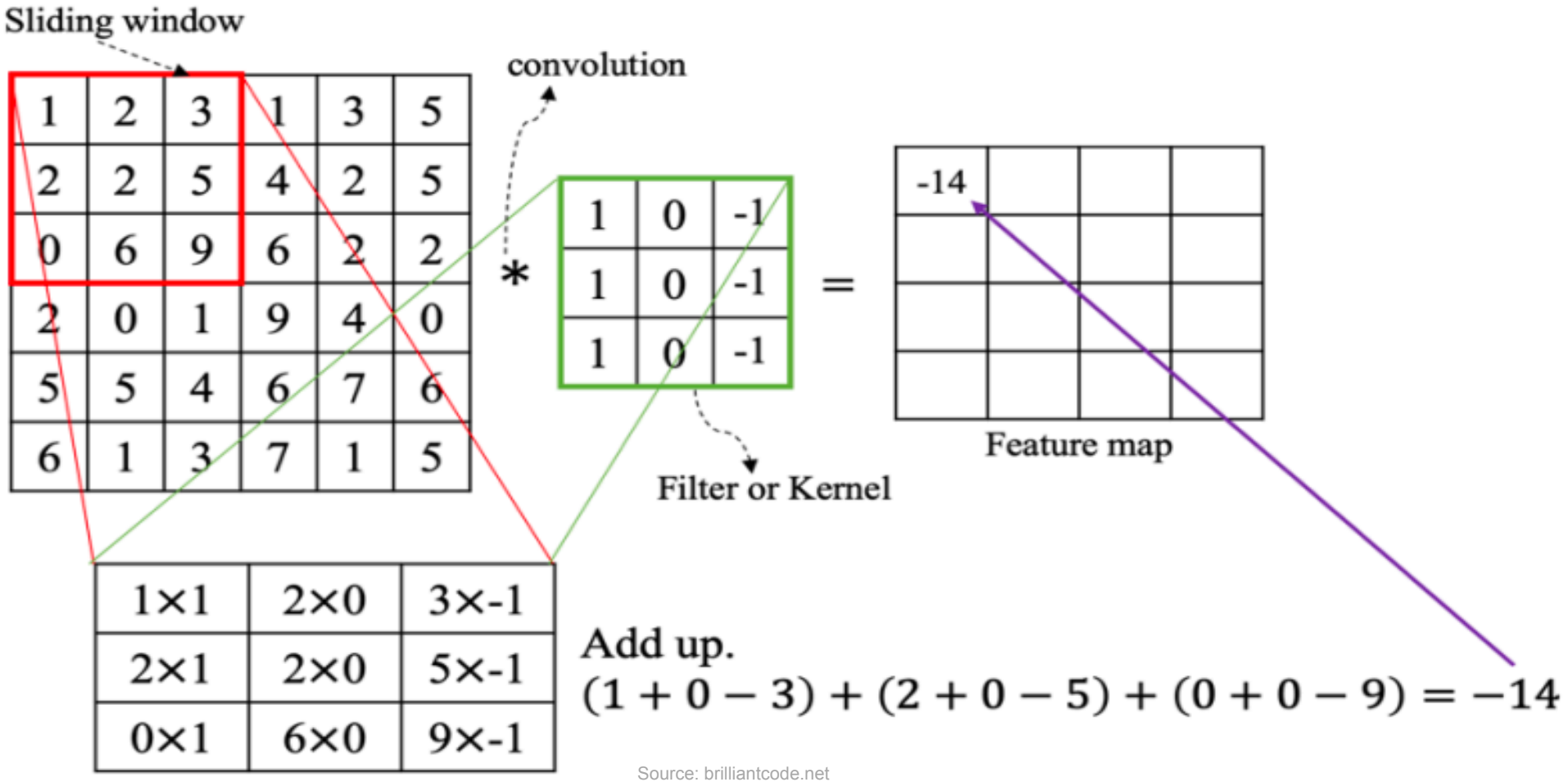
Source: brilliantcode.net

Example: Two-Dimensional Convolution



Source: brilliantcode.net

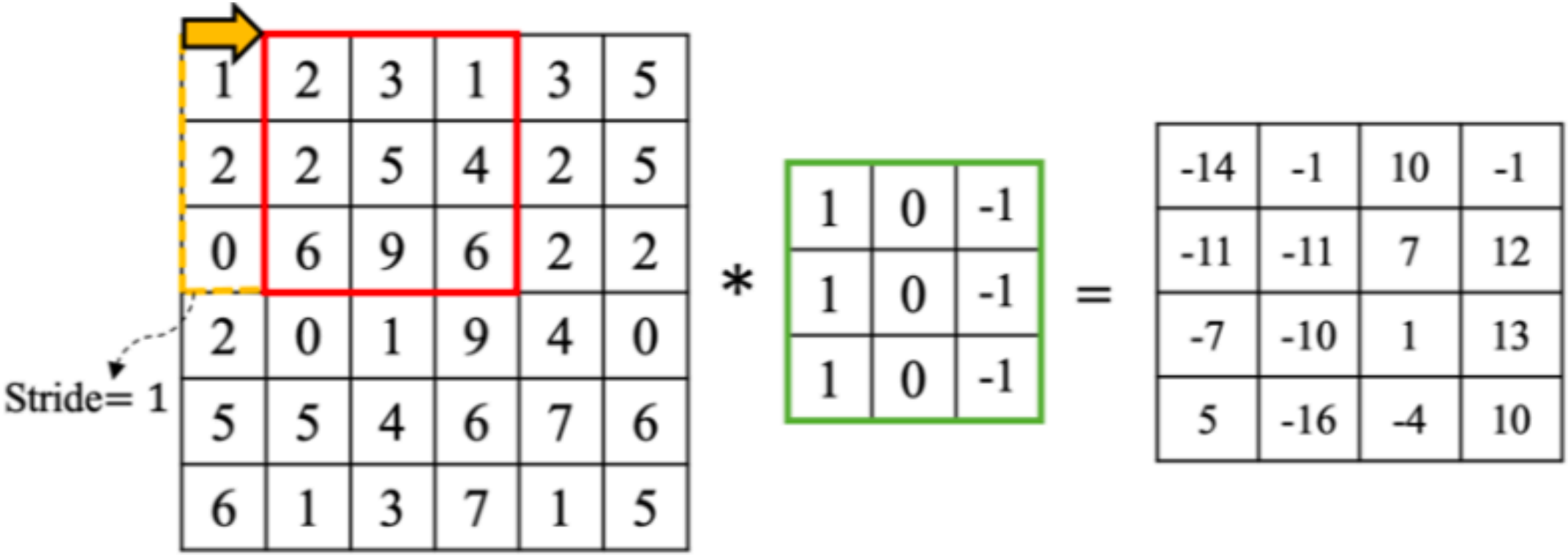
Example: Two-Dimensional Convolution



Source: brilliantcode.net

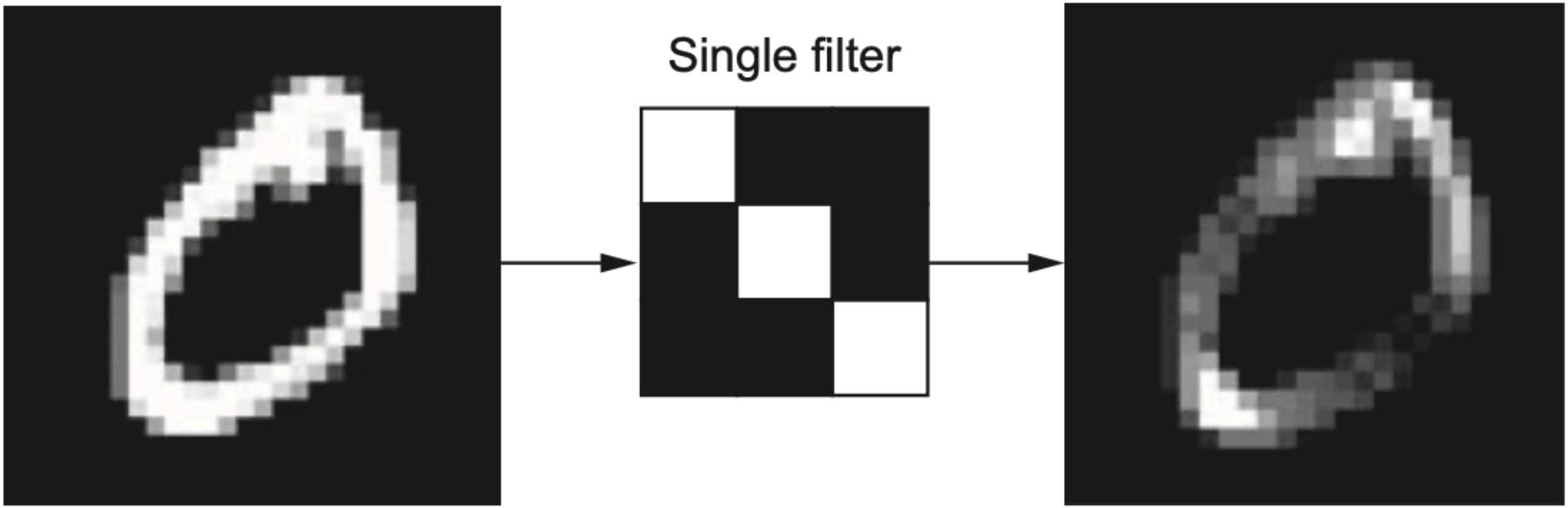
Example: Two-Dimensional Convolution (cont.)

- e.g., stride = 1



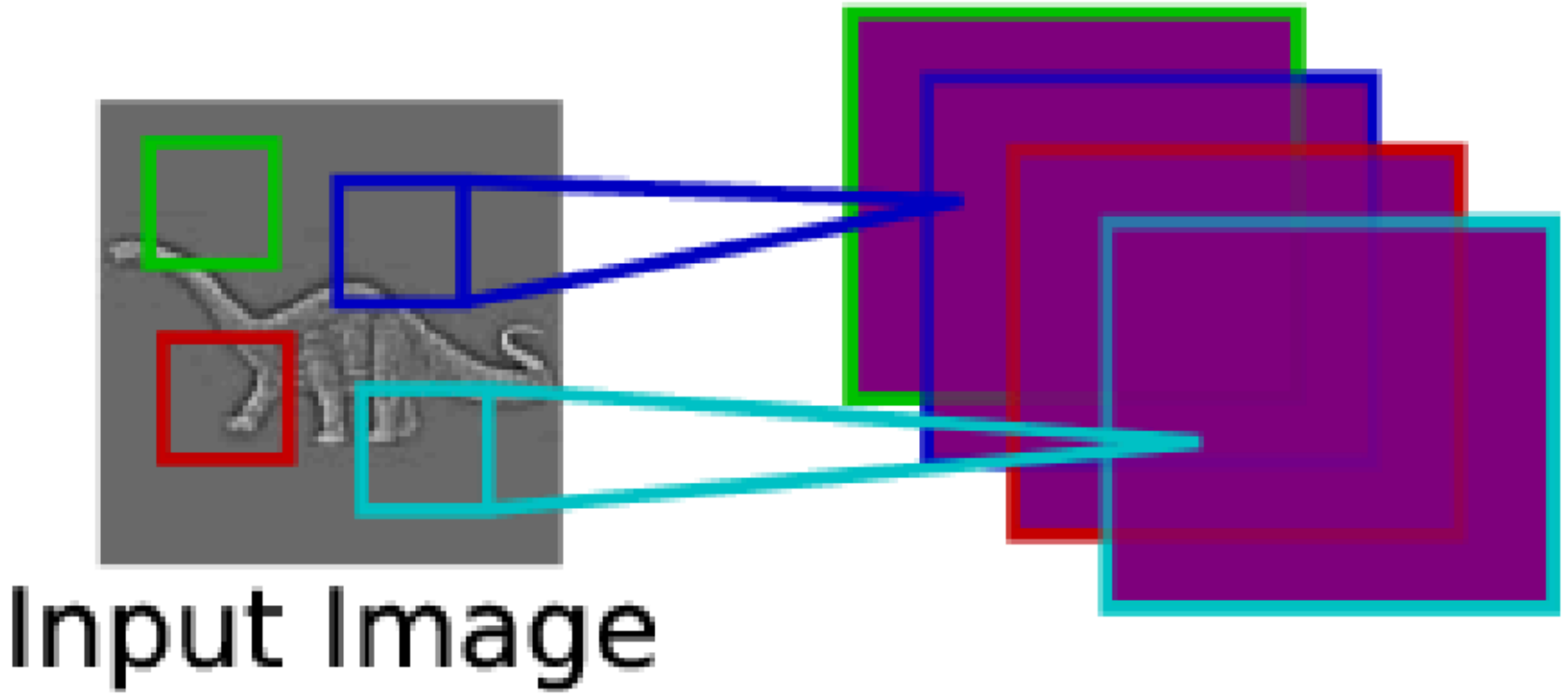
Source: brilliantcode.net

Example 1: Convolution



Example 2: Convolution

- E.g., 4 kernels

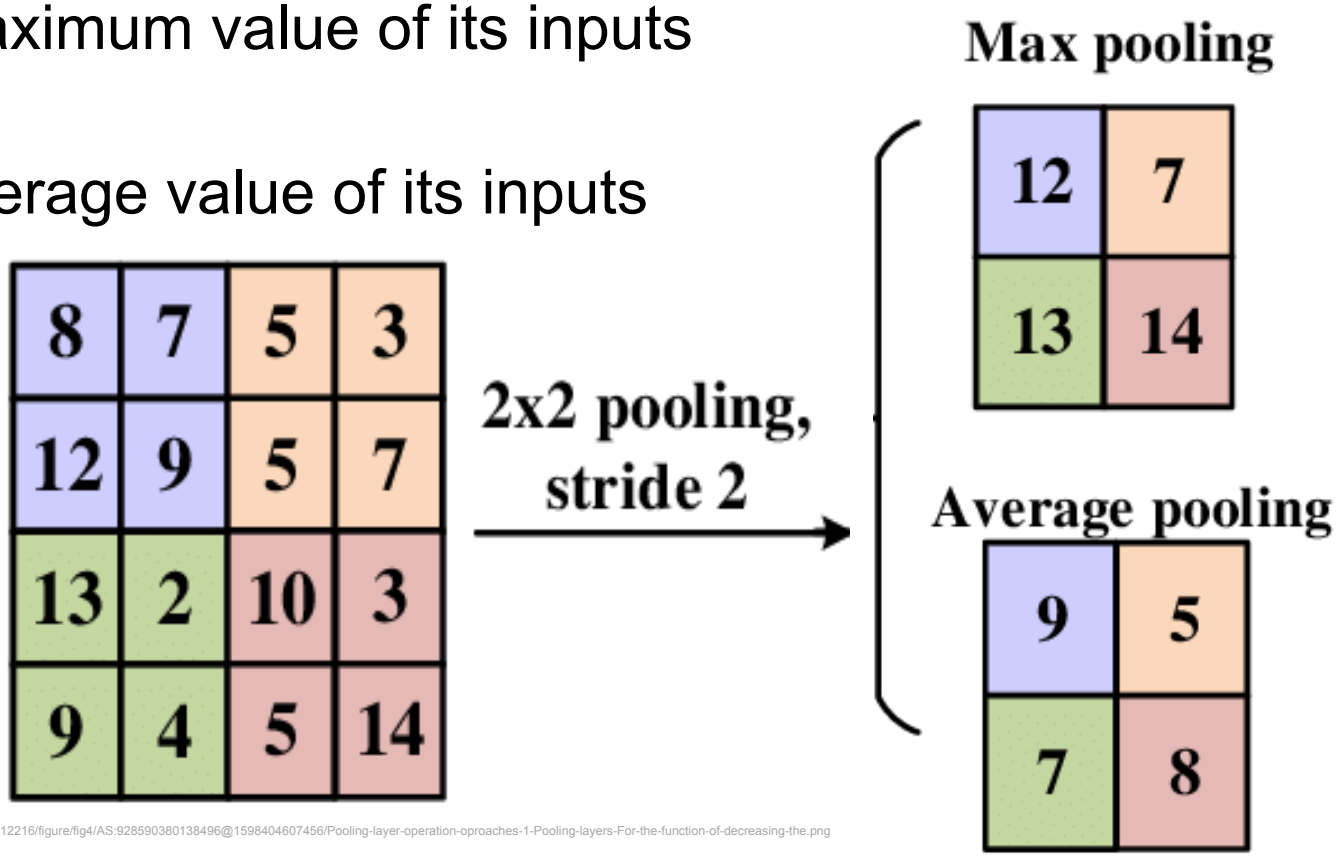


Pooling Layer

- Subsampling or Downsampling
  - Summarize a set of adjacent units from the preceding layer with a single value by a fixed kernel
- Methods
  - Max-pooling
  - Average-pooling

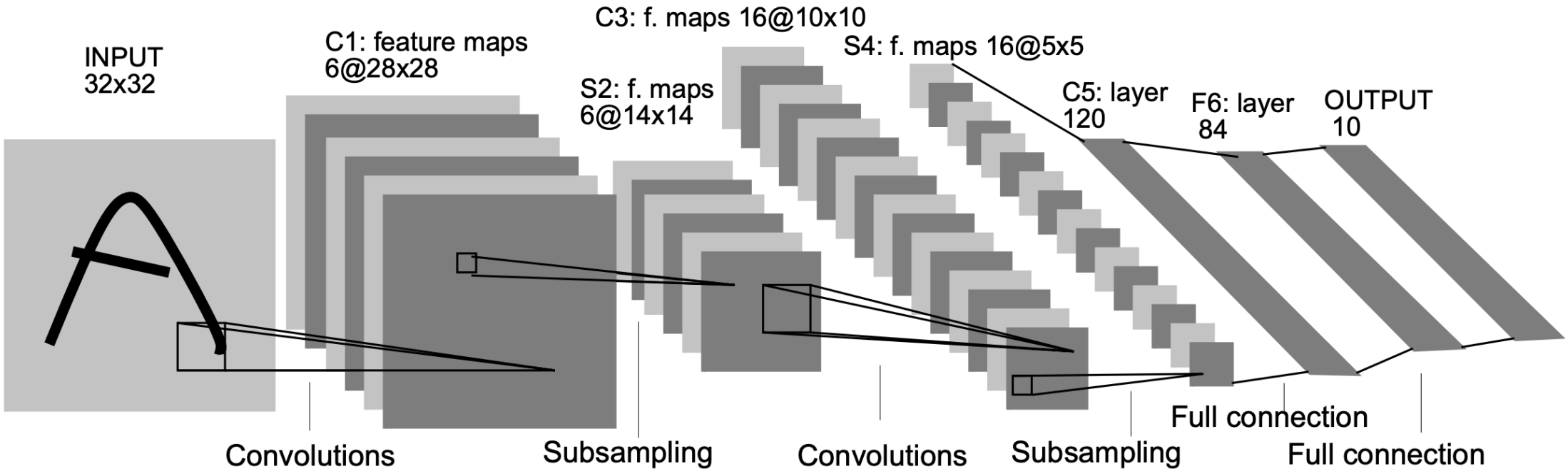
Pooling Methods

- Max-pooling
  - Compute the maximum value of its inputs
- Average-pooling
  - Compute the average value of its inputs

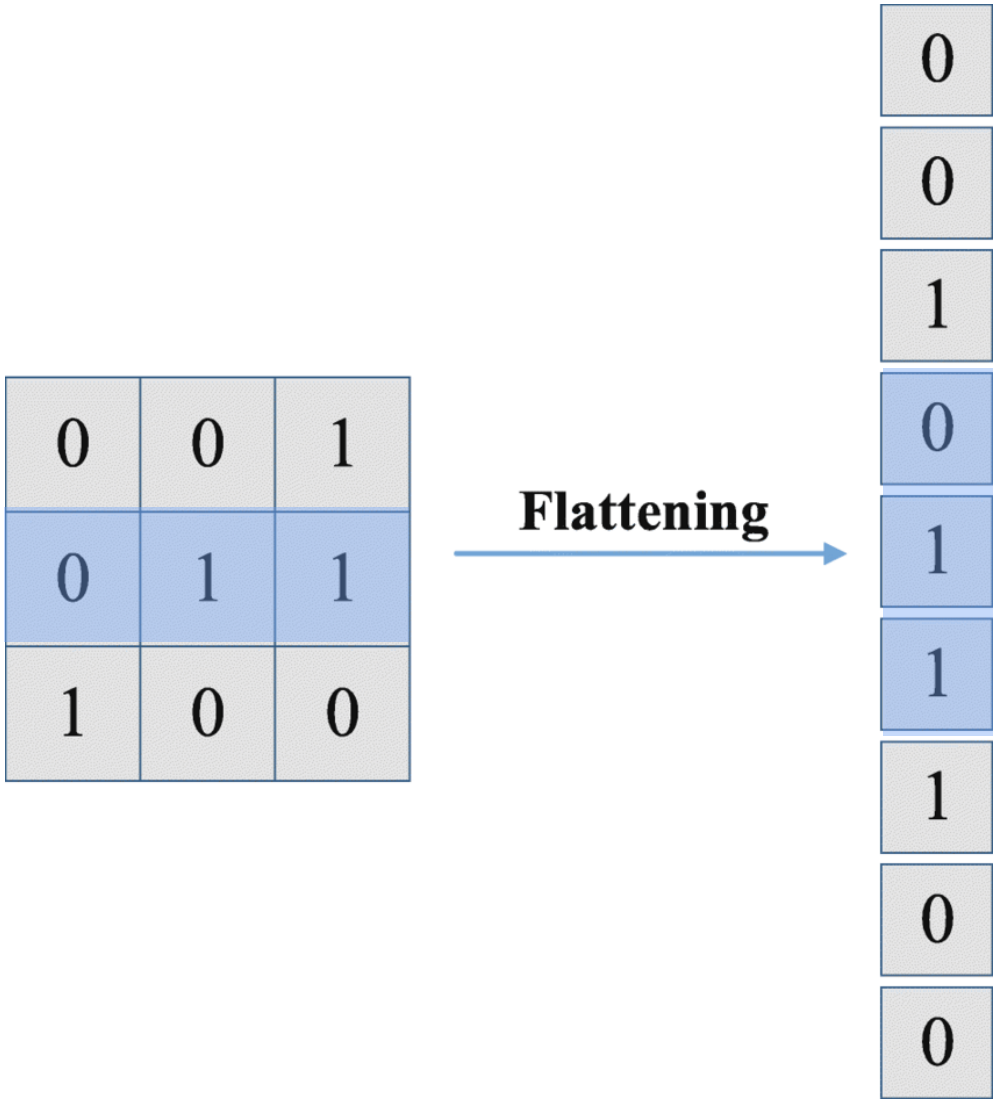


https://www.researchgate.net/publication/340812216/figure/fig4/AS:928590380138496@1598404607456/Pooling-layer-operation-oproaches-1-Pooling-layers-For-the-function-of-decreasing-the.png

CNNs Example: LeNet-5



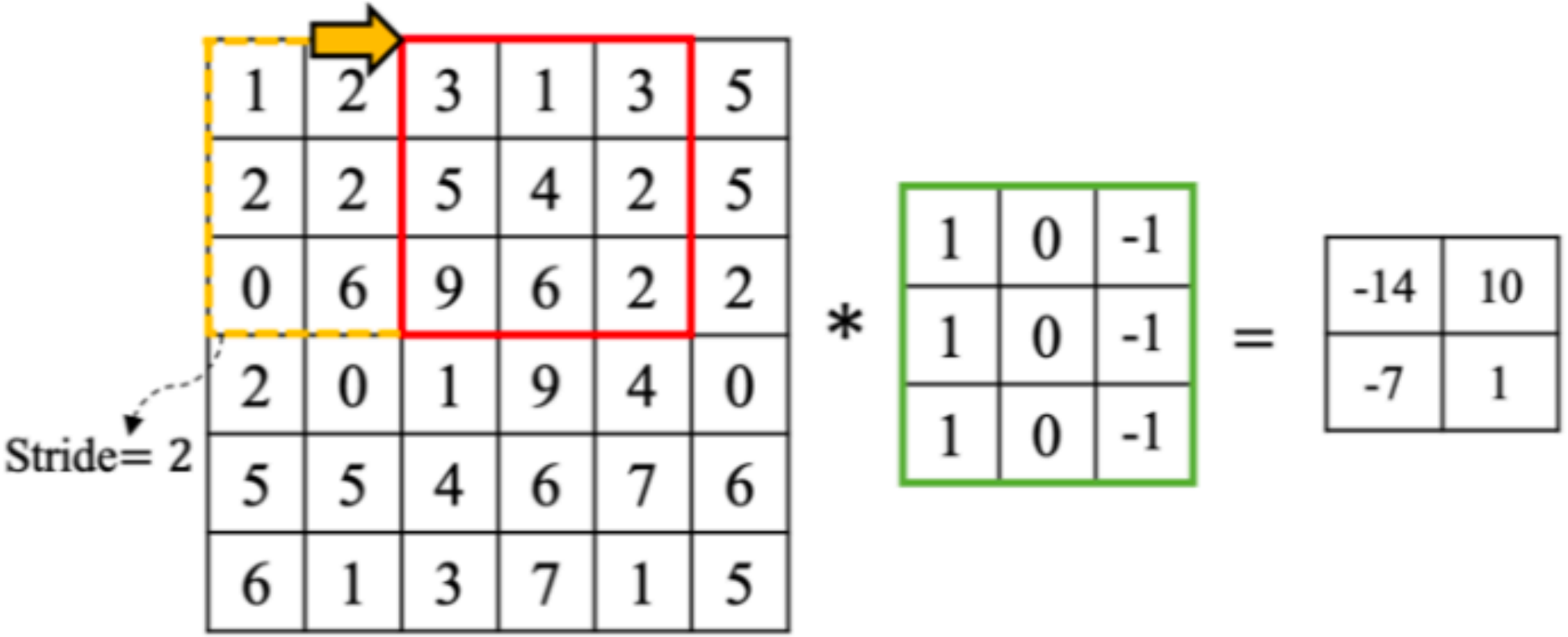
Flatten



https://www.researchgate.net/publication/343035394/figure/fig7/AS:941768455512076@1601546505544/A-simple-flattening-operation-example.png

Issue: Convolution

- e.g., stride = 2



Source: brilliantcode.net

Padding in CNNs

- Padding technique is adding extra pixels around the input image or feature map

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

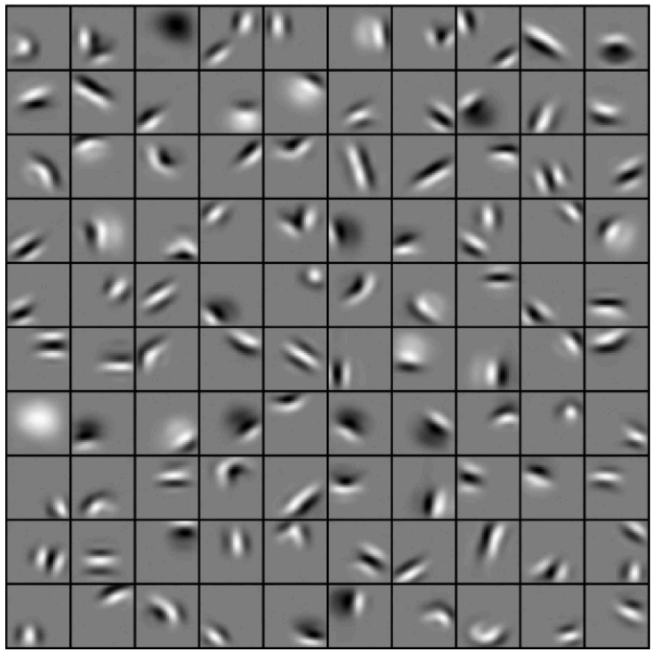
114				

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQy6lQKfi4TwX31iLCEECfMEwRF7TZLN8xn1s05a15riA&s

Padding in CNNs (cont.)

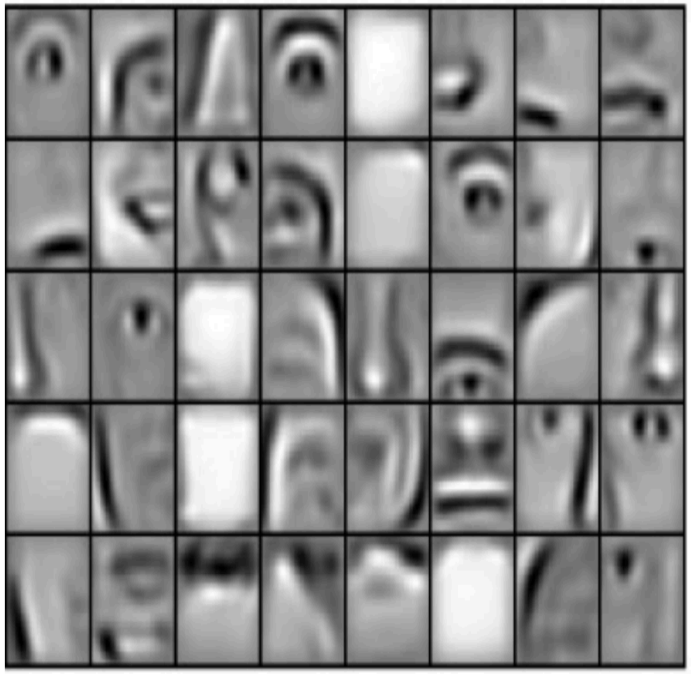
- Pros
  - Maintain spatial dimensions during the convolution operation
    - Keep the output size consistent with the input
  - Keep the spatial information intact and prevent data loss at the edges

Low Level Features



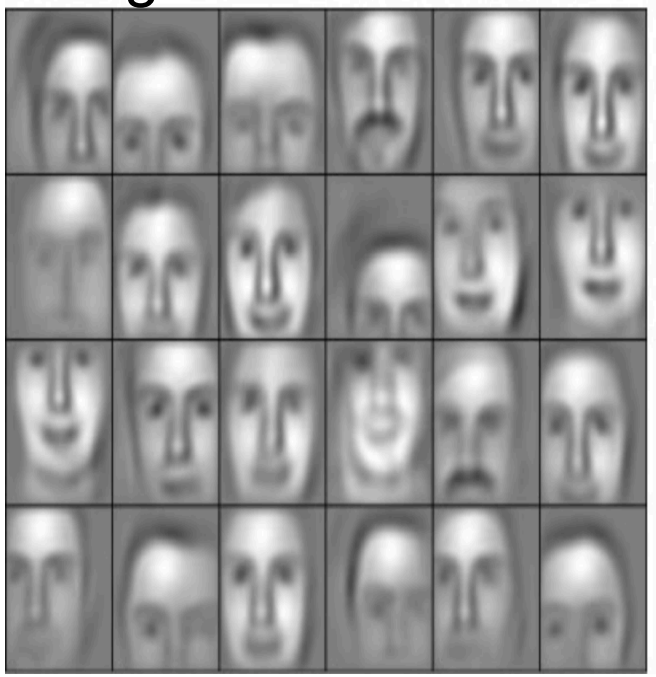
Edges, Dark Spots

Mid Level Features



Eyes, Ears, Nose

High Level Features

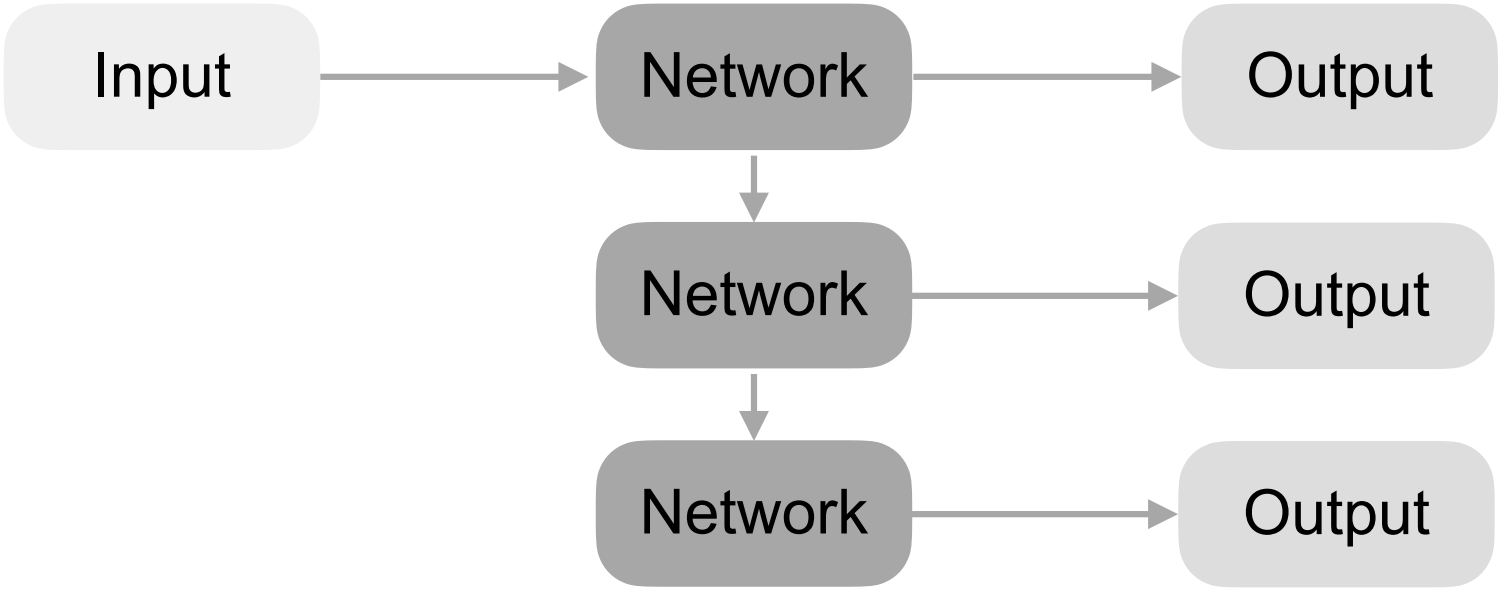


Facial Structure

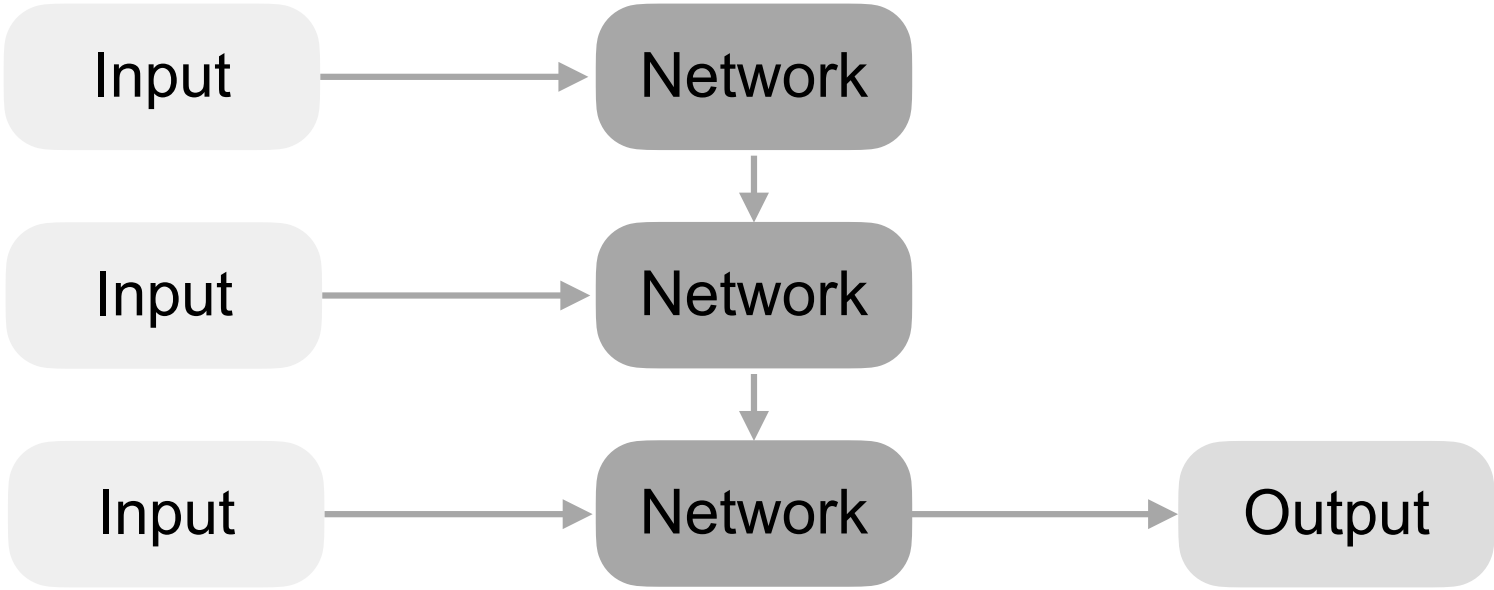
Unsupervised learning of hierarchical representations with convolutional deep belief networks,

Python: Examples

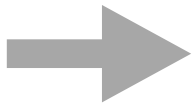

Input
Network
Output


Input
Network
Output
Network
Output
Network
Output

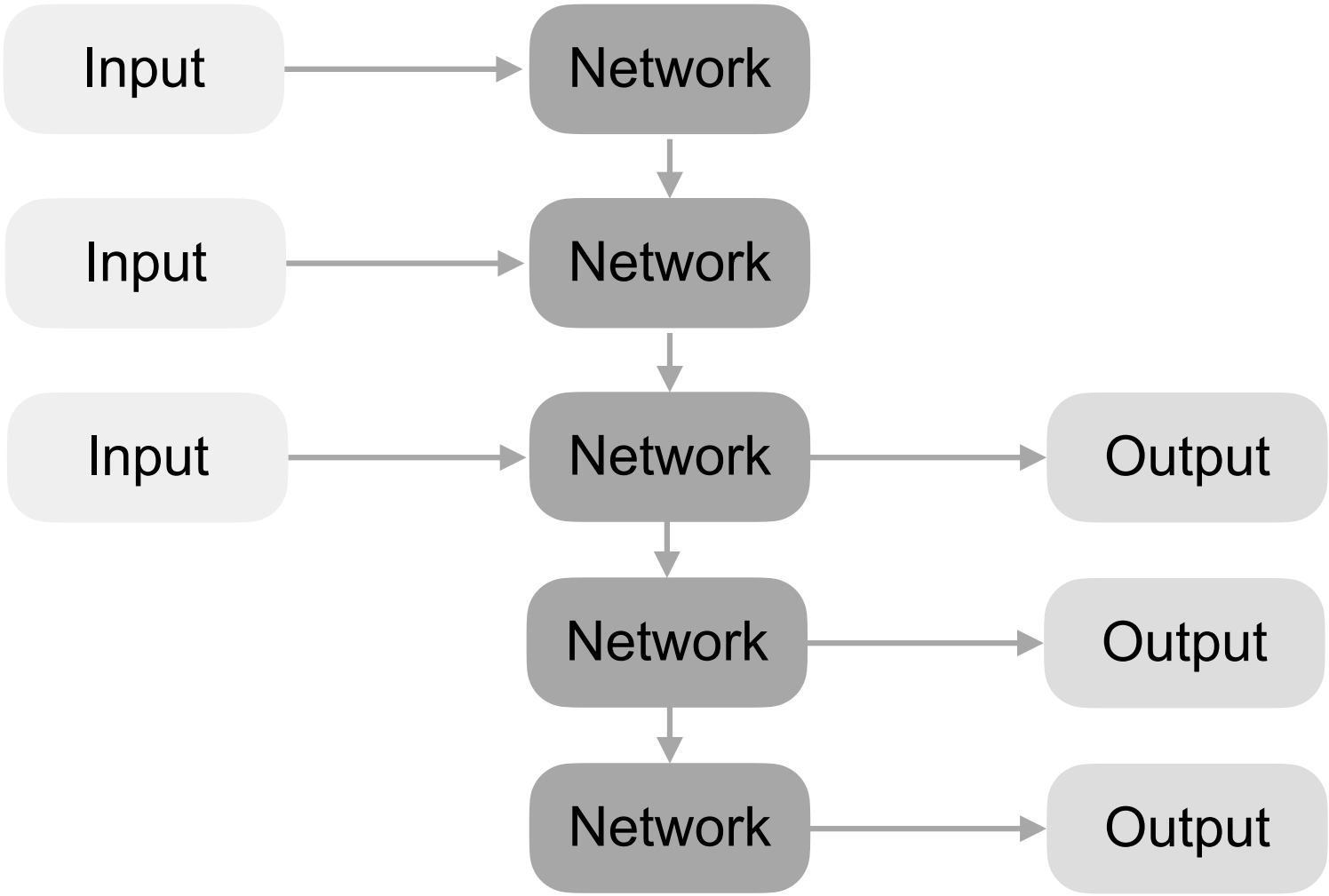
https://www.merit-times.com.tw/news_pic/20240323/883124-A9303497.JPG


Input
Network
Input
Network
Input
Network
Output

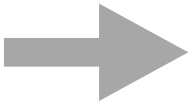
This movie was amazing



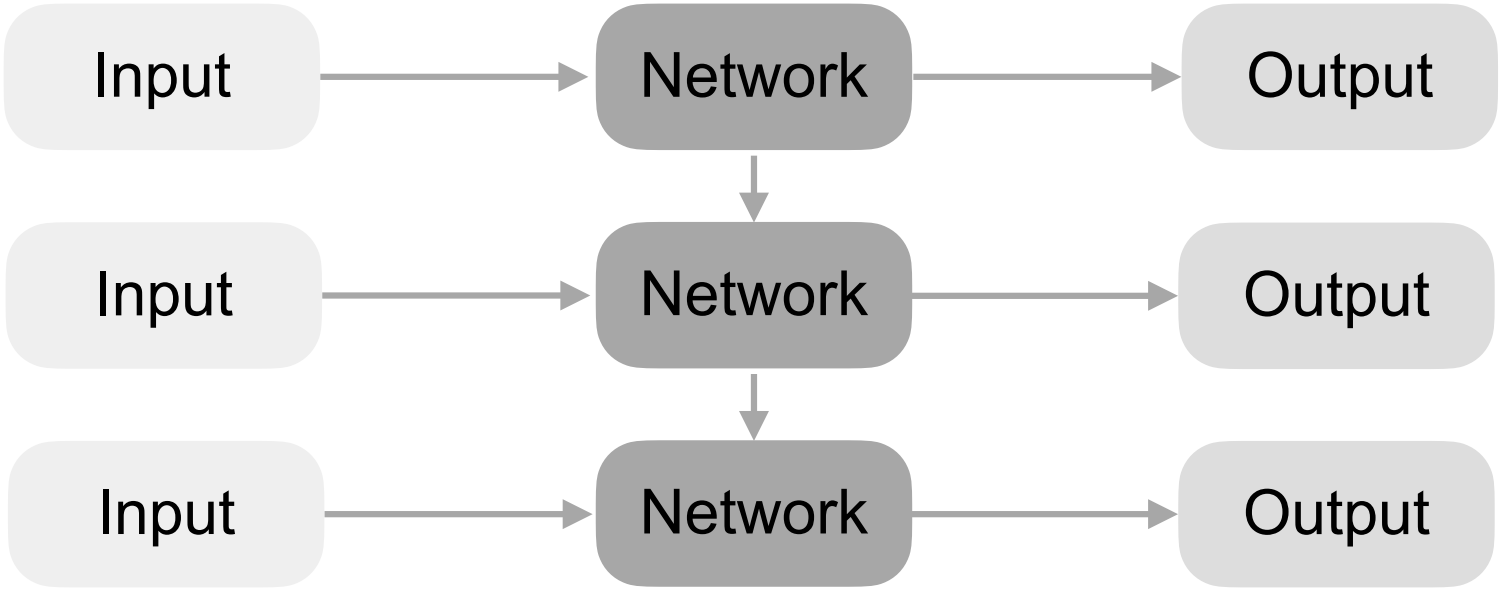
Positive

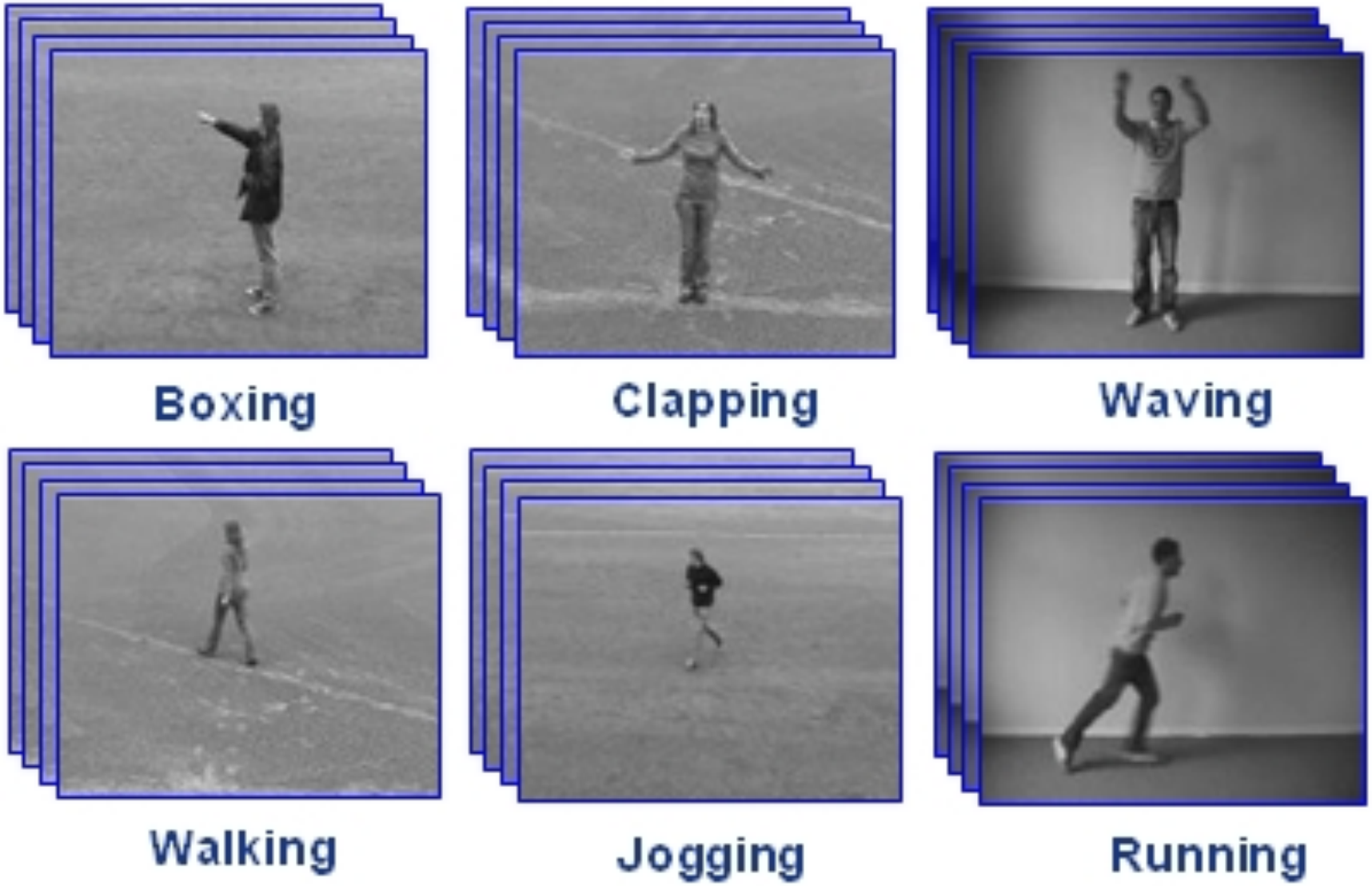

Input
Input
Input
Network
Network
Network
Network
Network
Output
Output
Output

現在正在下雨



It
is
raining
now


Input
Network
Output
Input
Network
Output
Input
Network
Output

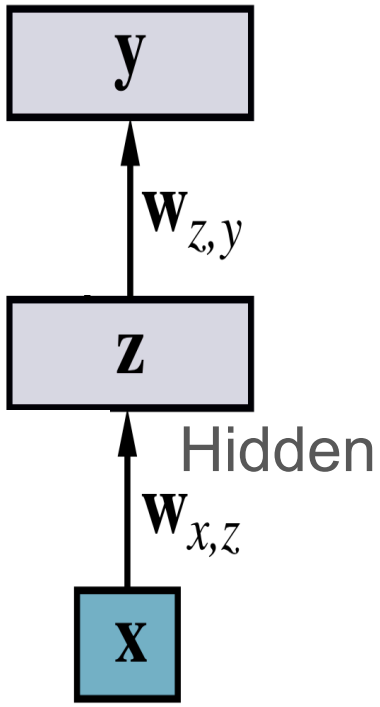


https://www.crcv.ucf.edu/projects/mmi_action/kth.jpg

Recurrent Neural Network (RNNs)

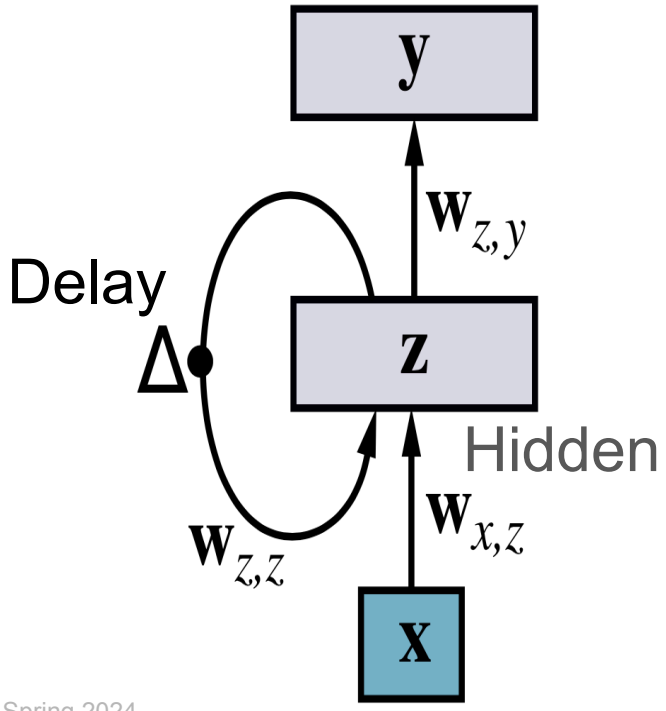
Recurrent Neural Networks (RNNs)

- A recurrent neural network feeds its intermediate or final outputs back into its own inputs



Recurrent Neural Networks (RNNs)

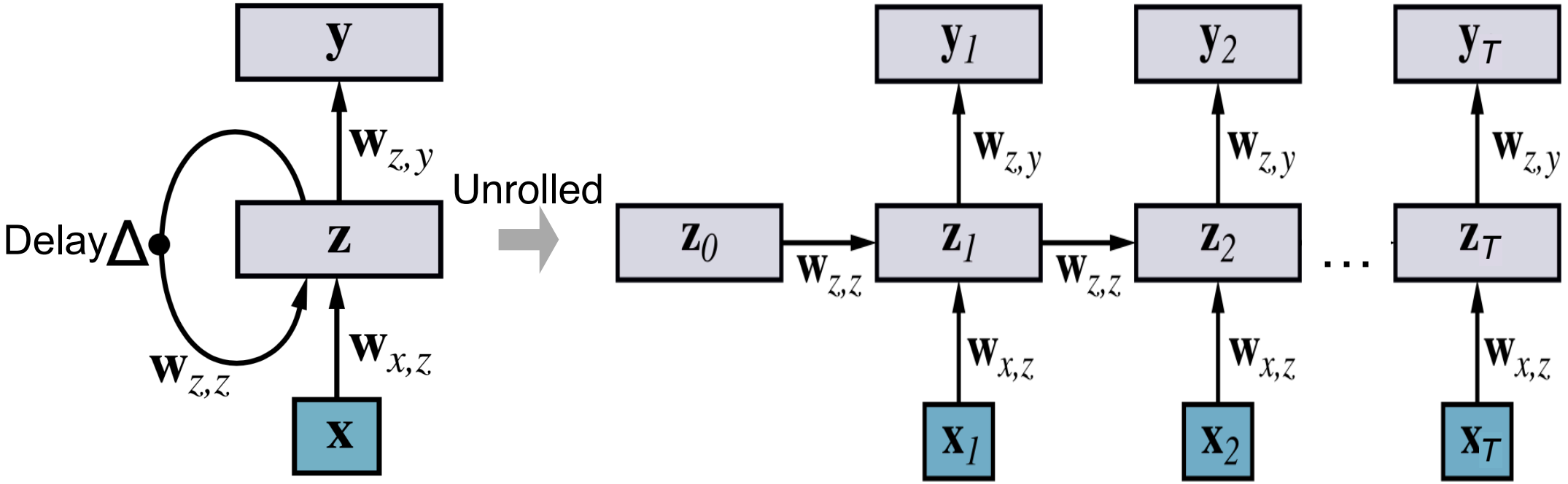
- A recurrent neural network feeds its intermediate or final outputs back into its own inputs



- Each cycle has a delay
- Units may take as input a value computed from their own output at an earlier step in the computation
- RNN has internal state, or memory:
  - Inputs received at earlier time steps affect the RNN's response to the current input

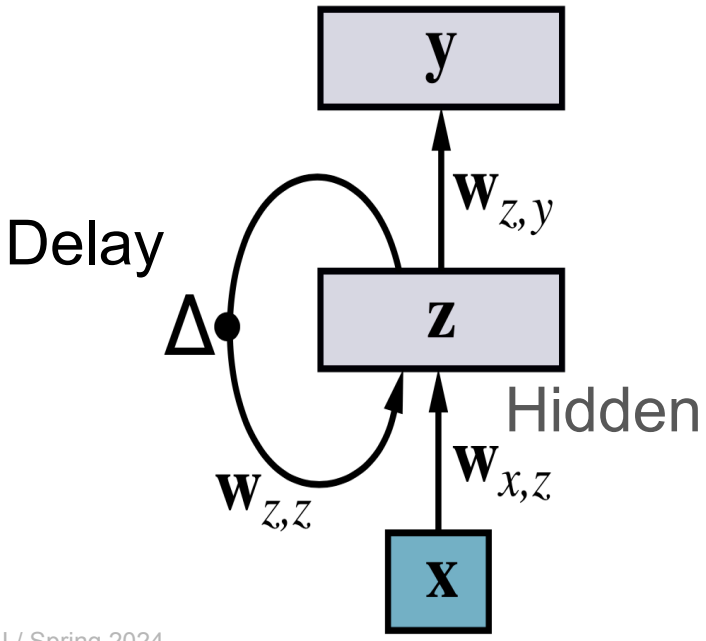
Unrolling Recurrent Neural Networks

- The RNN is unrolled over time steps to create a feedforward network



Recurrent Neural Networks (RNNs)

- Process a sequence of vectors **x** by applying a recurrence formula at every time step
  - The weights (i.e., $\mathbf{w}_{x,z}$, $\mathbf{w}_{z,z}$, $\mathbf{w}_{z,y}$) are shared across all time steps



$$\mathbf{z}_t = f_{\mathbf{w}}(\mathbf{z}_{t-1}, \mathbf{x}_t)$$

$$= \mathbf{g}_z(\mathbf{W}_{z,z}\mathbf{z}_{t-1} + \mathbf{W}_{x,z}\mathbf{x}_t)$$

$$\hat{\mathbf{y}}_t = \mathbf{g}_y(\mathbf{W}_{z,y}\mathbf{z}_t)$$

Activation Function

Long-term dependencies
are impossible to learn.

Information from the sequence can be transported to a later timestep, and jump off, intact, when you need it.

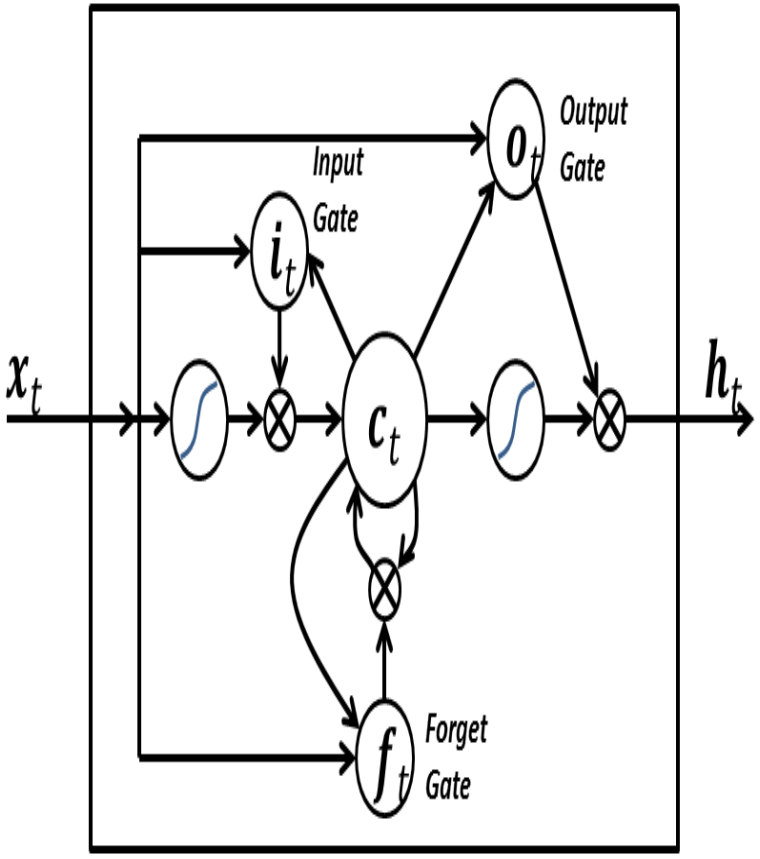
Long Short-Term Memory RNNs (LSTM)

- The long-term memory component of an LSTM, called the memory cell and denoted by *c*, is essentially
  - *c* is copied from time step to time step
- LSTMs include gating units that control the flow of information in the LSTM
  - Forget gate f
  - Input gate i
  - Output gate o

https://upload.wikimedia.org/wikipedia/commons/d/d5/Long_Short_Term_Memory.png

Long Short-Term Memory RNNs (LSTM) (cont.)

- The forget gate
  - Determines if each element of the memory cell is remembered (copied to the next time step) or forgotten (reset to zero)
- The input gate
  - Determines if each element of the memory cell is updated additively by new information from the input vector at the current time step
- The output gate
  - Determines if each element of the memory cell is transferred to the short-term memory, which plays a similar role to the hidden state in basic RNNs



Applications in Natural Langage Processing (NLP)

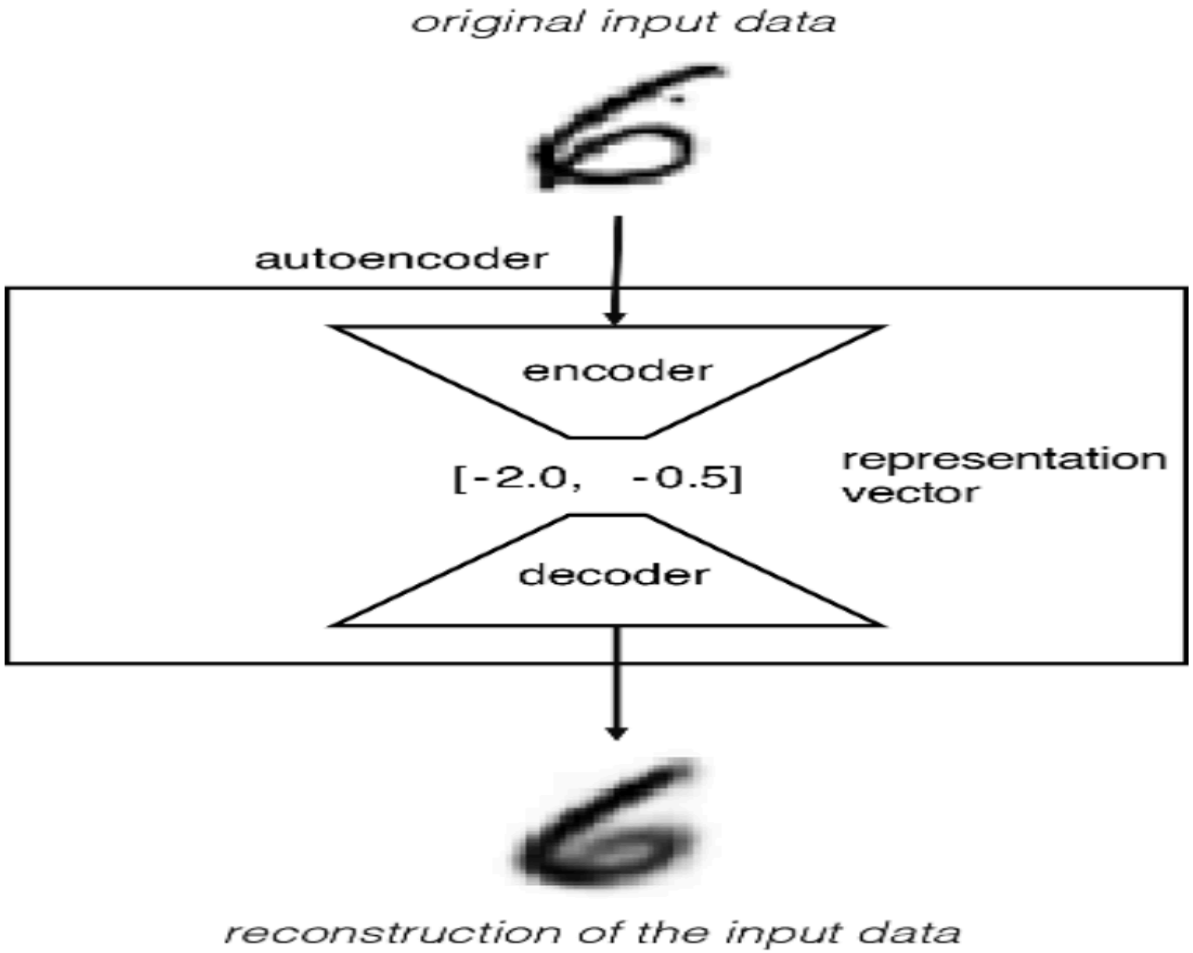
- Speech recognition
- Handwriting recognition
- Named entity recognition
- Automatic summarization
- Text classification
- Machine translation
- …

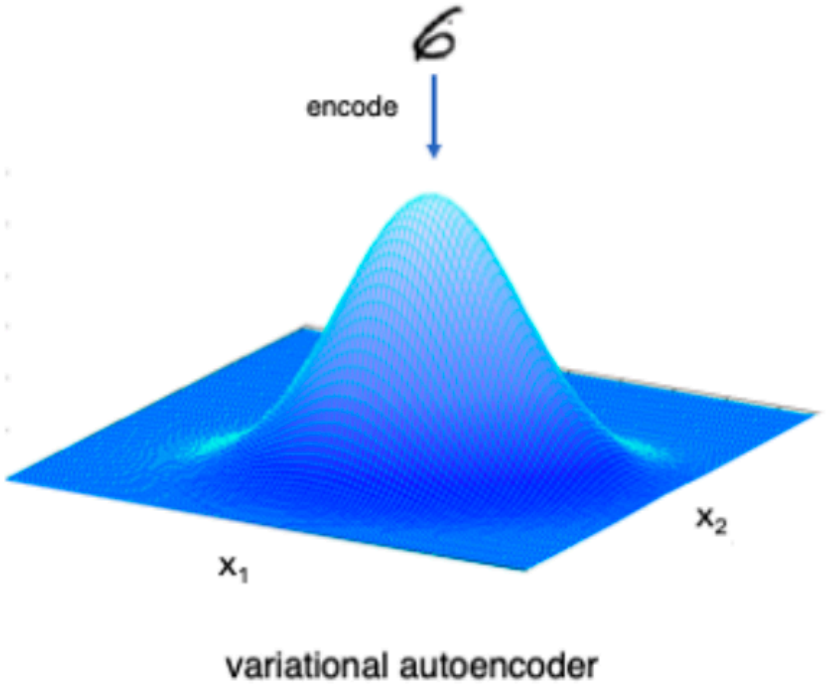
Unsupervised Deep Learning

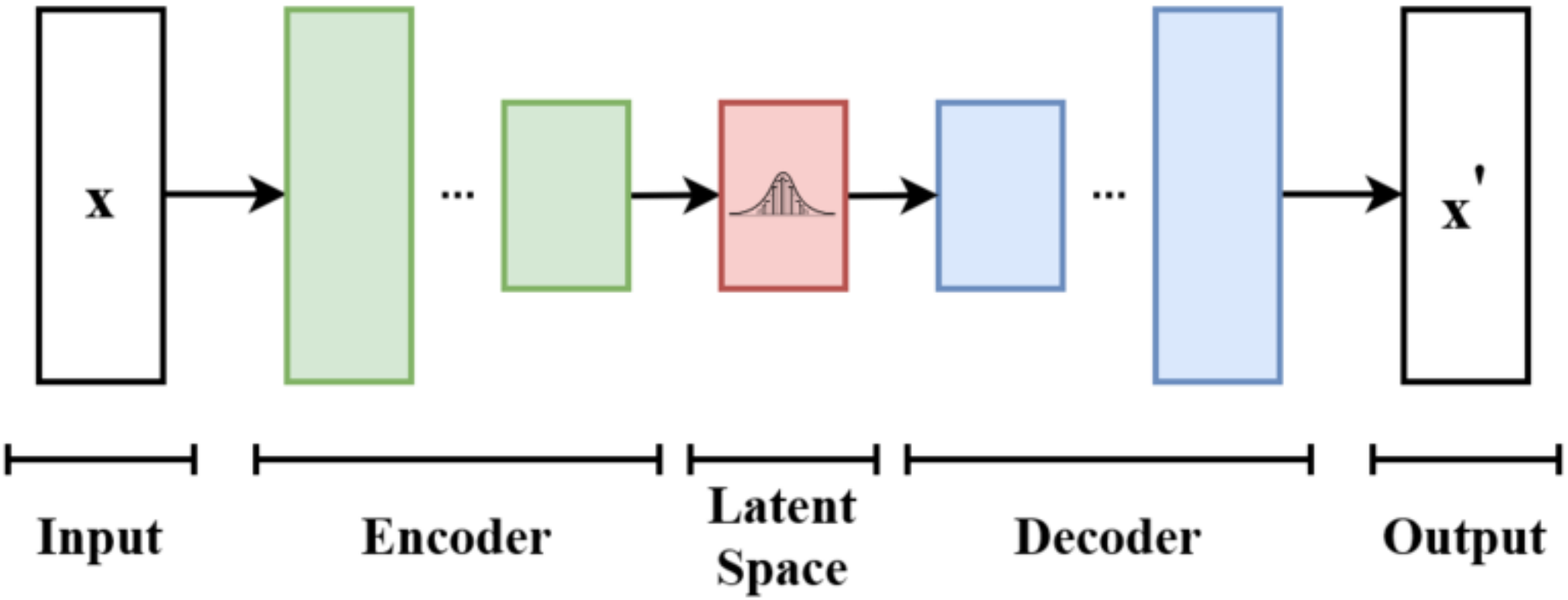
Unsupervised Deep Learning

- Learn solely from unlabeled inputs, which are often more abundantly available than labeled examples
- Unsupervised learning algorithms may try to
  - Learn new representations
    - e.g. new features of images that make it easier to identify the objects in an image
  - Learn a generative model
    - e.g., typically in the form of a probability distribution from which new samples can be generated

Variational Autoencoder (VAE)






x
...
...
x'
Input
Encoder
Latent Space
Decoder
Output

Transfer Learning

- Experience with one learning task helps an agent learn better on another task
- Idea
  - Copy over the weights learned for task A to a network that will be trained for task B
  - The weights are then updated by gradient descent in the usual way using data for task B
  - Often freeze the first few layers of the pre-trained model
    - These layers serve as feature detectors that will be useful for your new model

Multitask learning

- A form of transfer learning in which we simultaneously train a model on multiple objectives
- e.g., training a natural language system simultaneously on part-of-speech tagging, document classification, language detection, word prediction, sentence difficulty modeling, plagiarism detection, sentence entailment, and question answering
  - Solve all eight at once with a common representation layer, the model is more likely to create a common representation that reflects real natural language usage and content