Deep Learning

CS473/CS573: Computer Vision

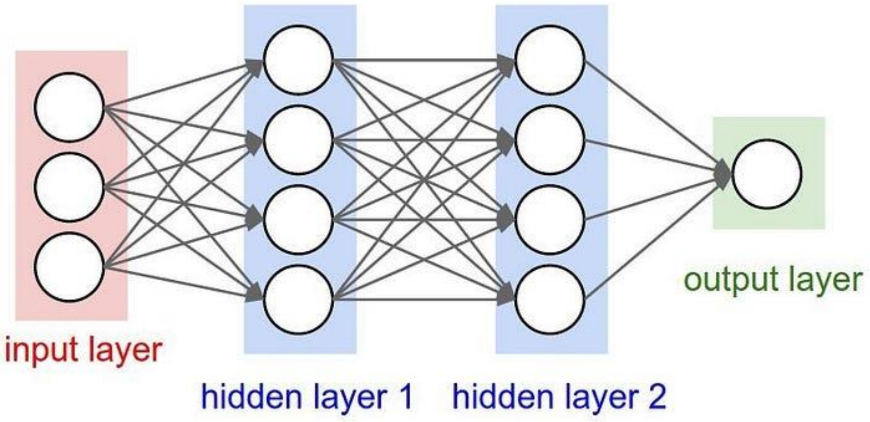
What is Deep Learning?

What is Deep Learning?

The goal of deep learning is to obtain a function that answers a specific question.

What is Deep Learning?

The goal of deep learning is to obtain a function that answers a specific question.

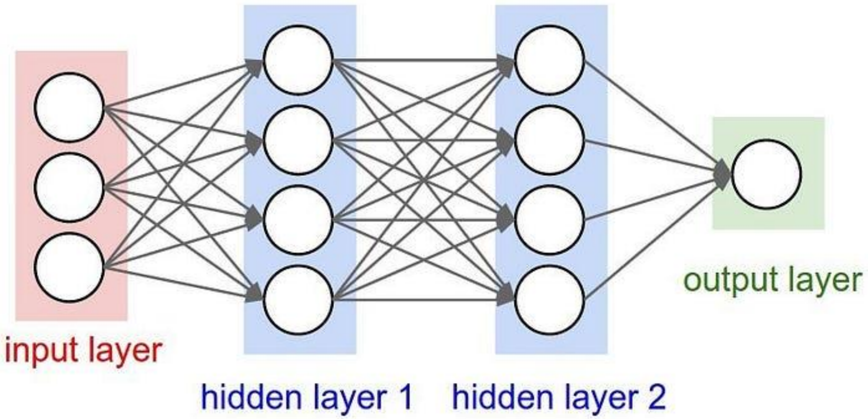


Deep

Function is multi-layered

What is Deep Learning?

The goal of deep learning is to obtain a function that answers a specific question.



dog

cat

elephant

Deep

Function is multi-layered

Learning

Function is estimated from data

Machine Learning

How do machines learn?

Machine Learning

How do machines learn?

Feature Extraction followed by Feature Reasoning

Machine Learning

How do machines learn?

Feature Extraction followed by Feature Reasoning

Example: Let’s answer a specific question:

Machine Learning

How do machines learn?

Input

What animal is in this image?

Machine Learning

How do machines learn?

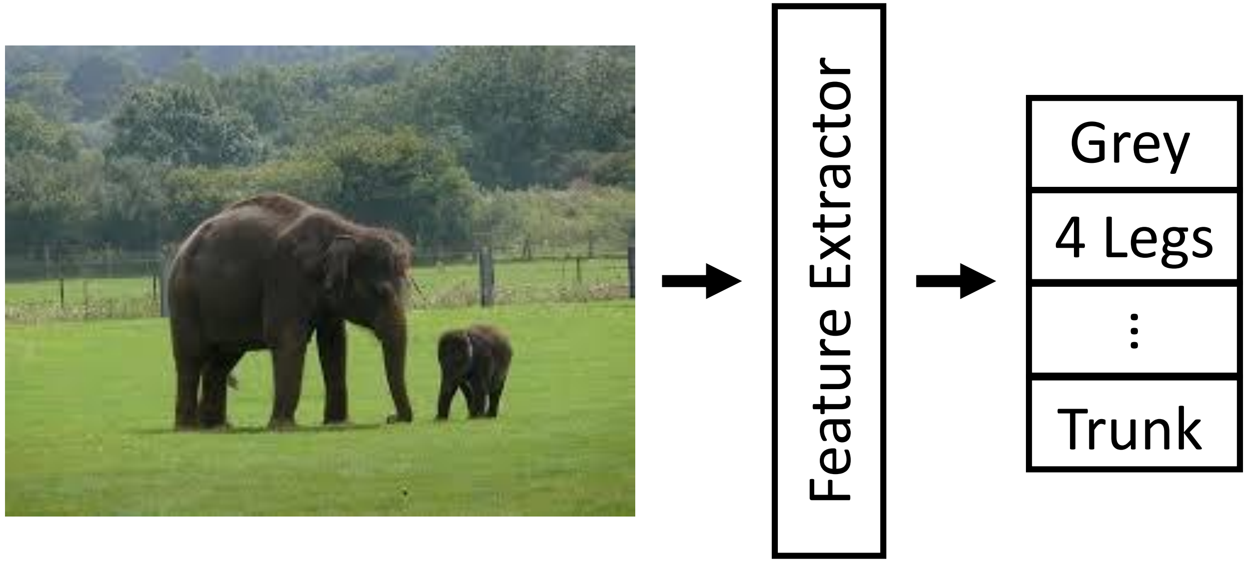
Input

→ Feature Extractor

What animal is in this image?

Machine Learning

How do machines learn?



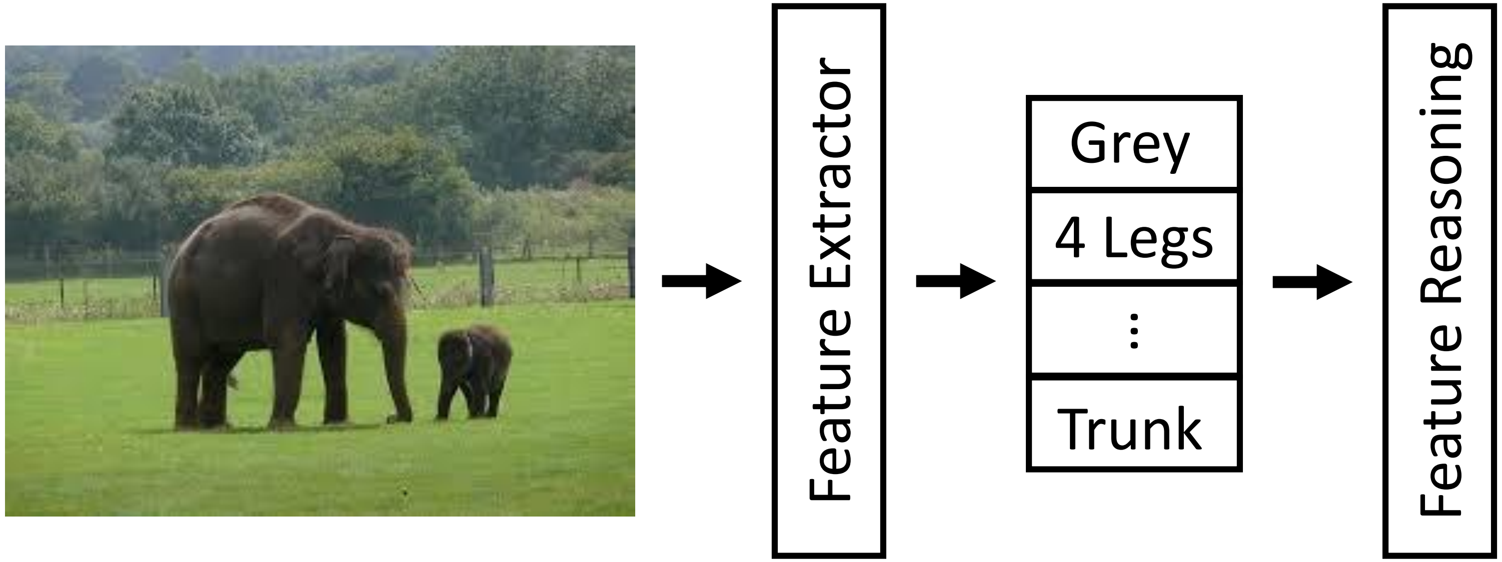
Input

Feature Vector

What animal is in this image?

Machine Learning

How do machines learn?



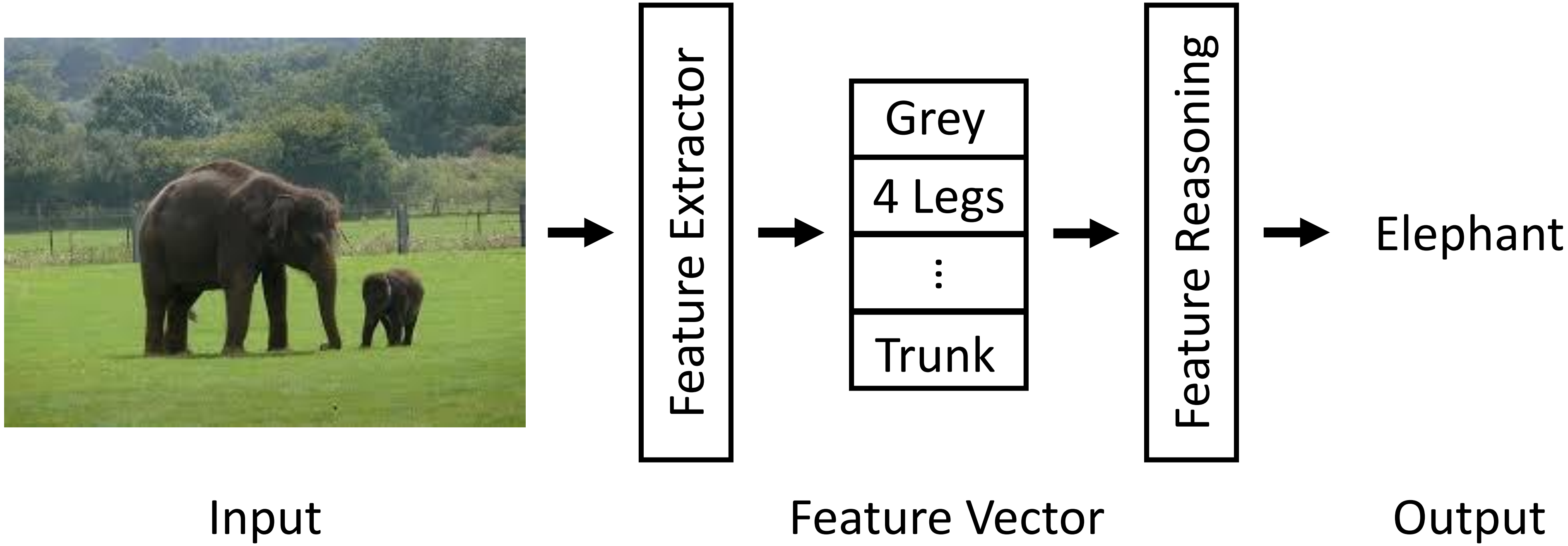
Input

Feature Vector

What animal is in this image?

Machine Learning

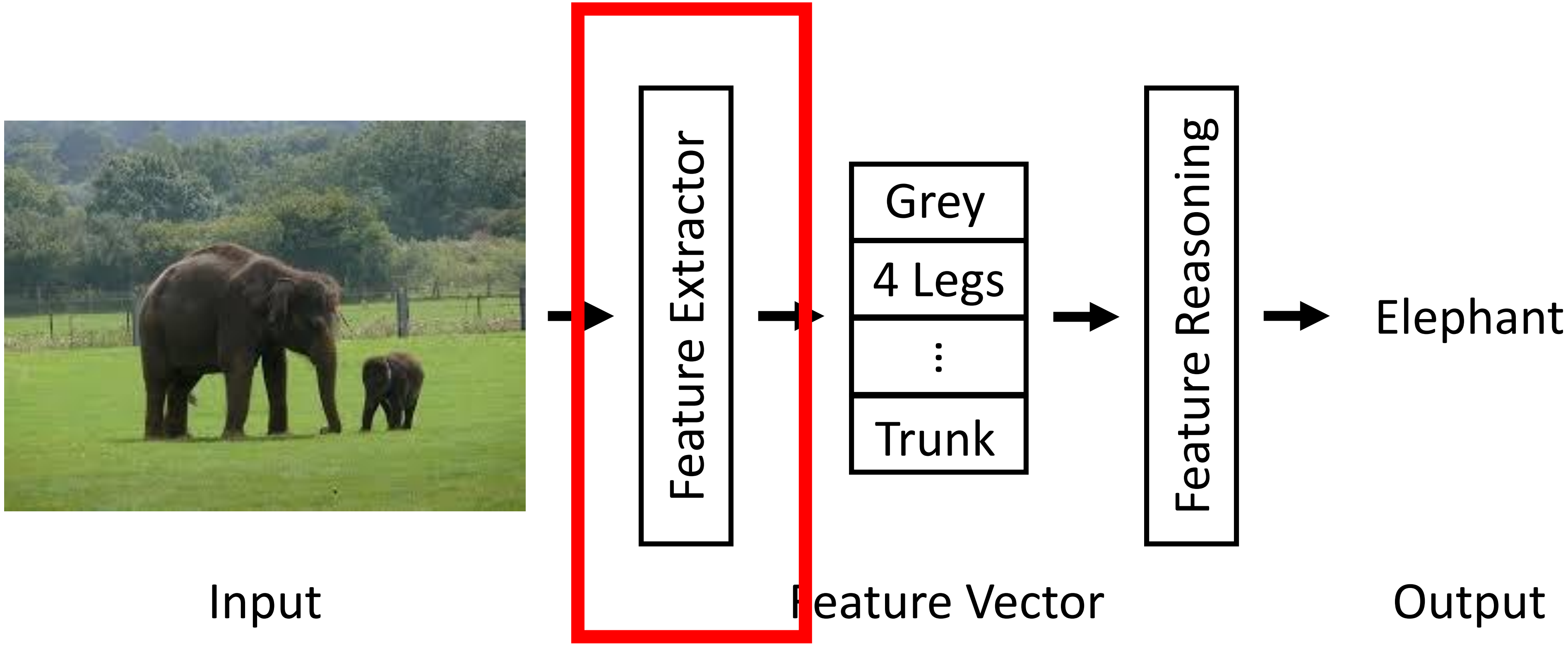
How do machines learn?



What animal is in this image?

Machine Learning

How do machines learn?



What animal is in this image?

Feature Extraction

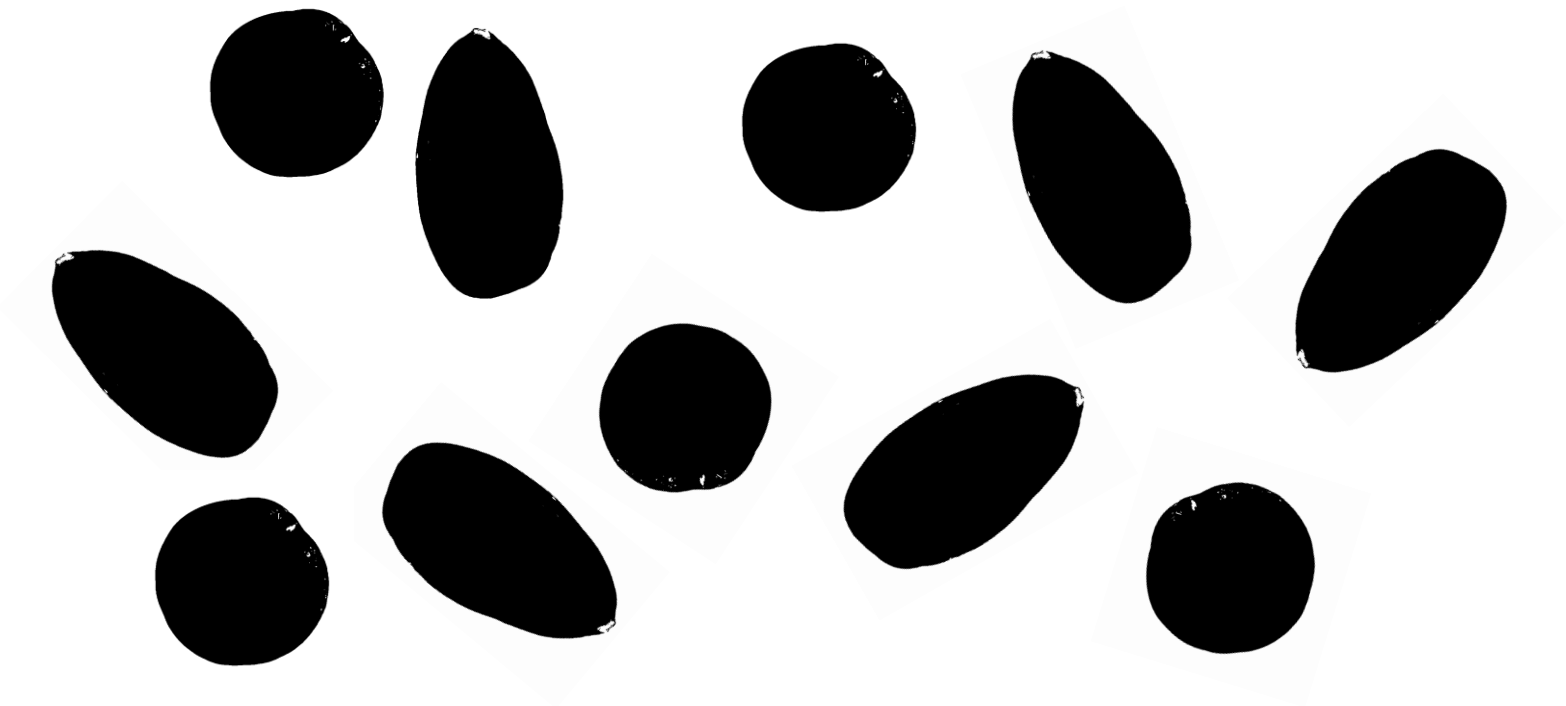
Example: let’s classify hazelnuts and almonds

Feature Extraction

Example: let’s classify hazelnuts and almonds

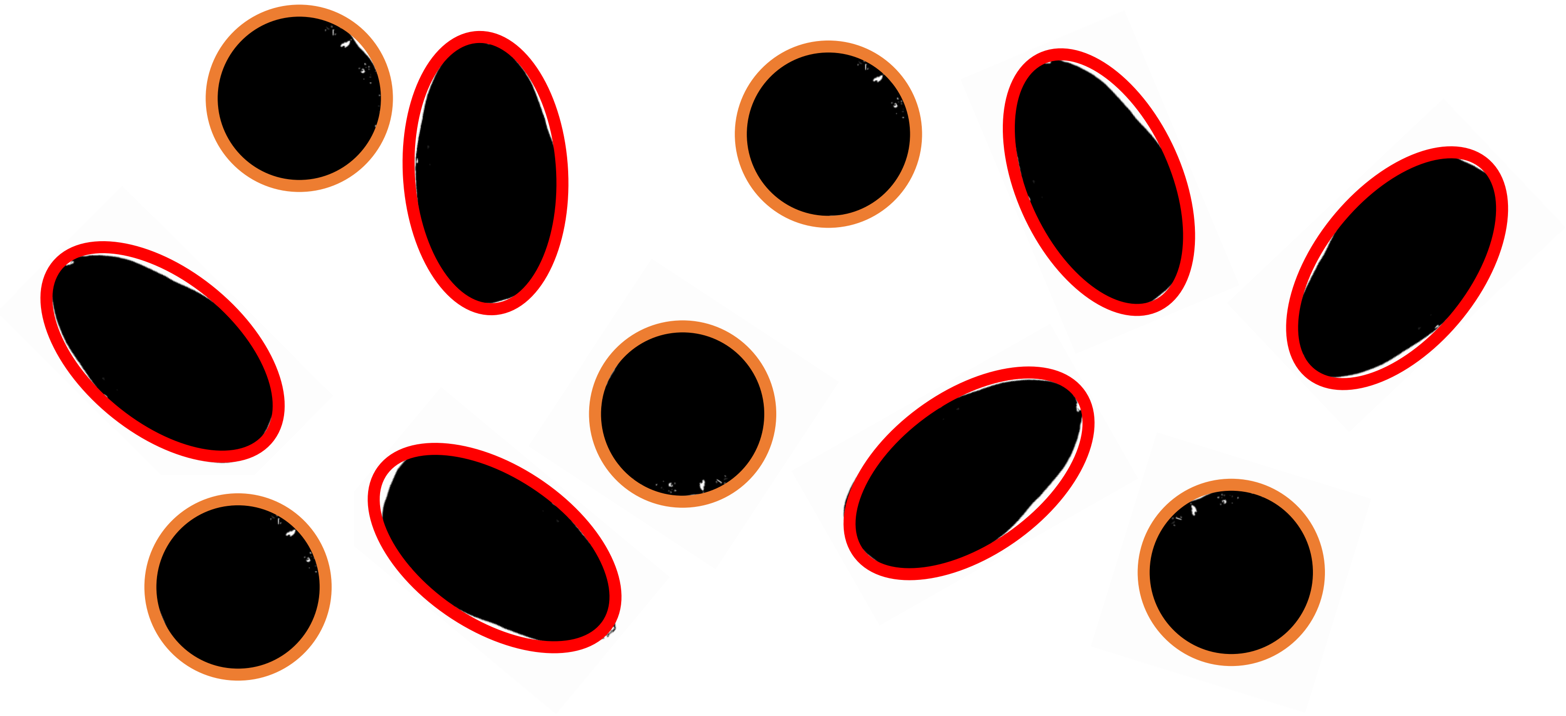
Feature Extraction

Example: let's classify hazelnuts and almonds



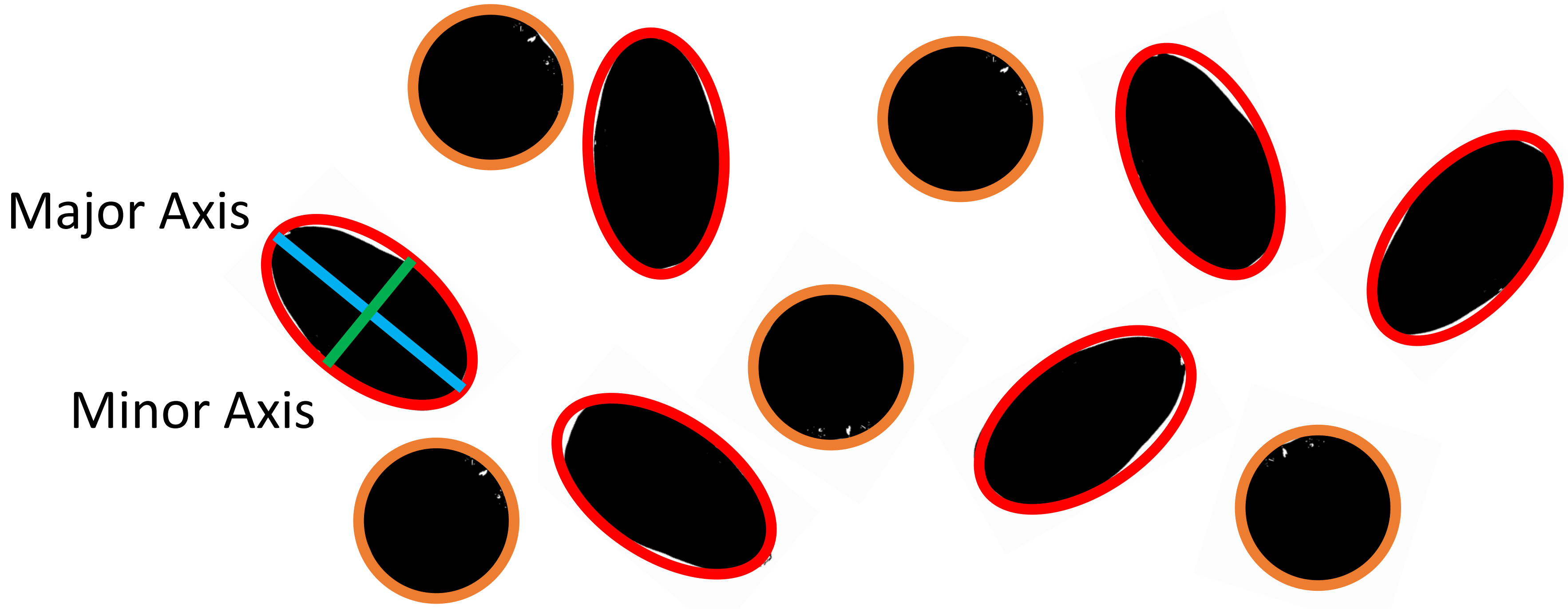
Feature Extraction

Example: let's classify hazelnuts and almonds



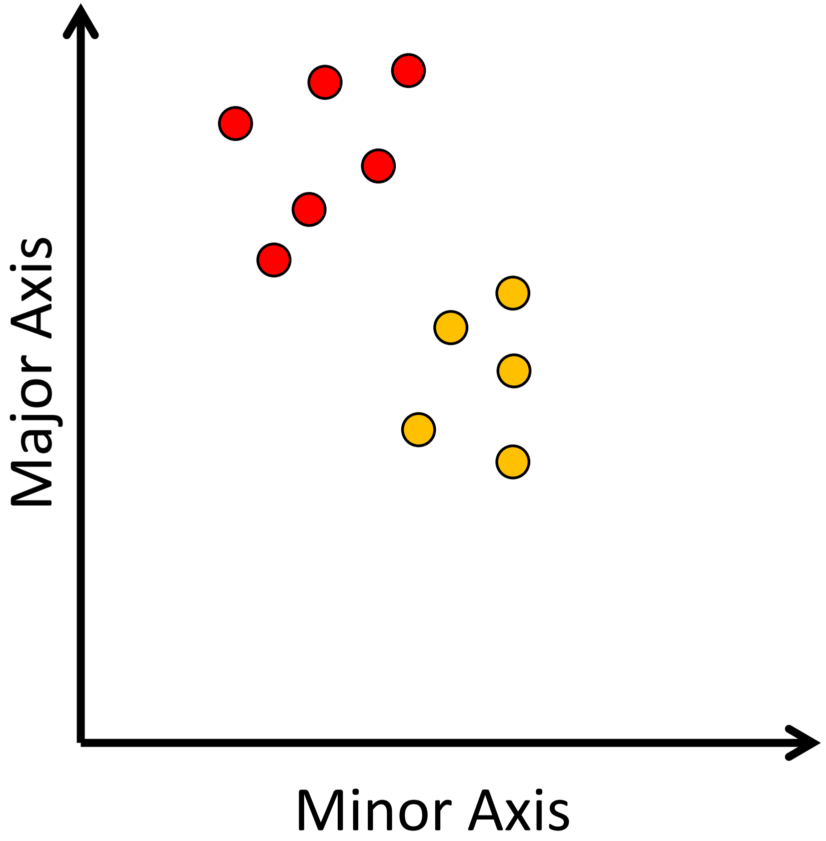
Feature Extraction

Example: let's classify hazelnuts and almonds



Feature Extraction

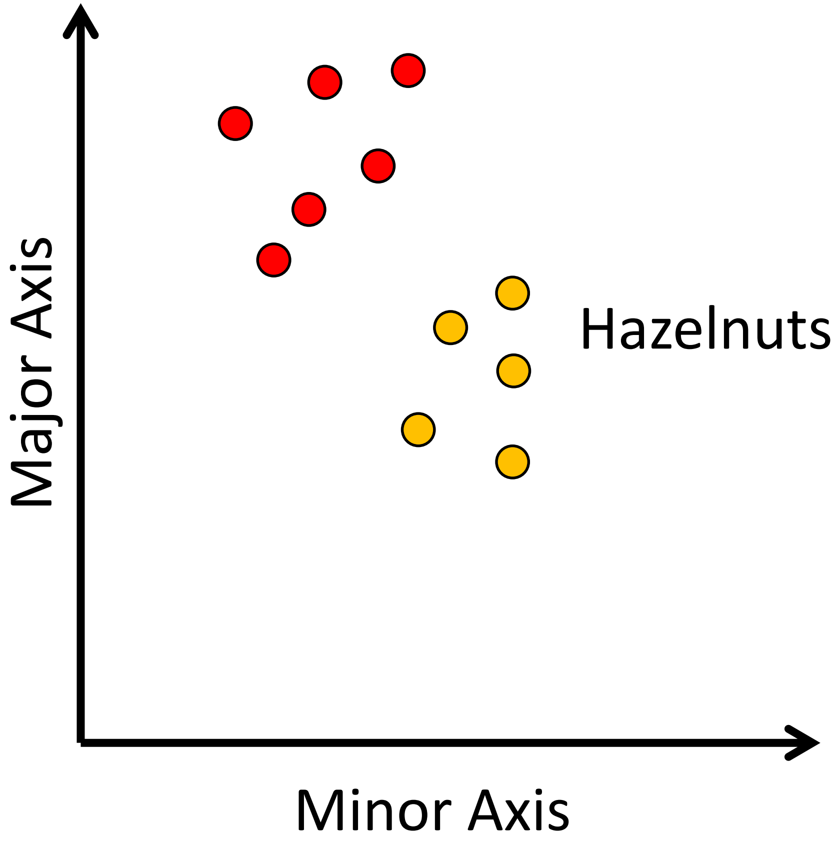
Example: let’s classify hazelnuts and almonds



Hazelnuts vs Almonds

Feature Extraction

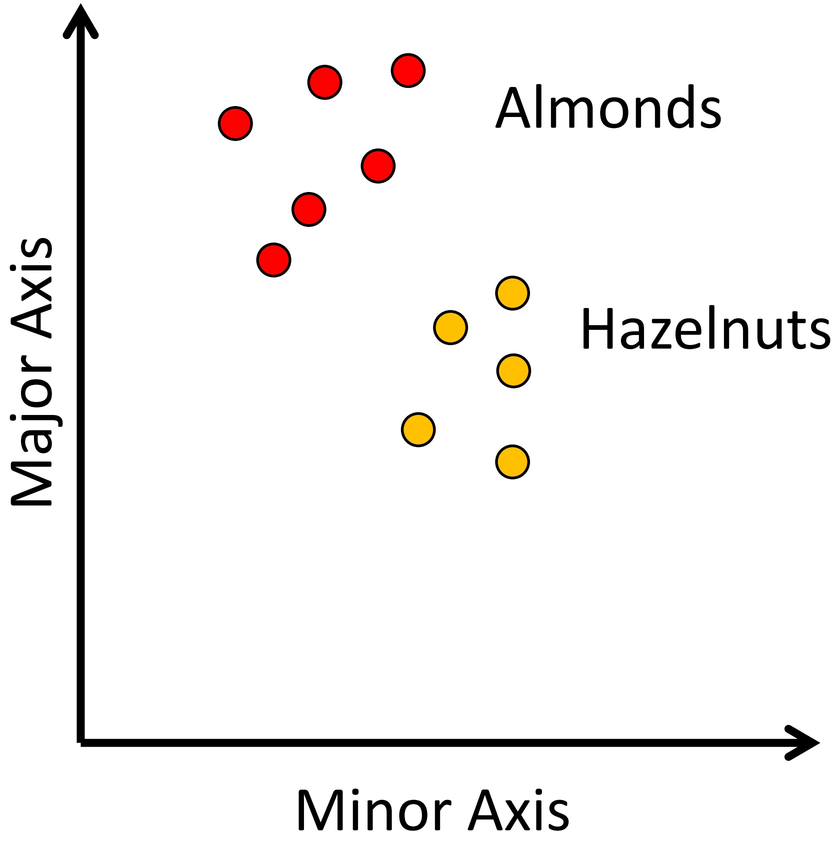
Example: let's classify hazelnuts and almonds



Hazelnuts vs Almonds

Feature Extraction

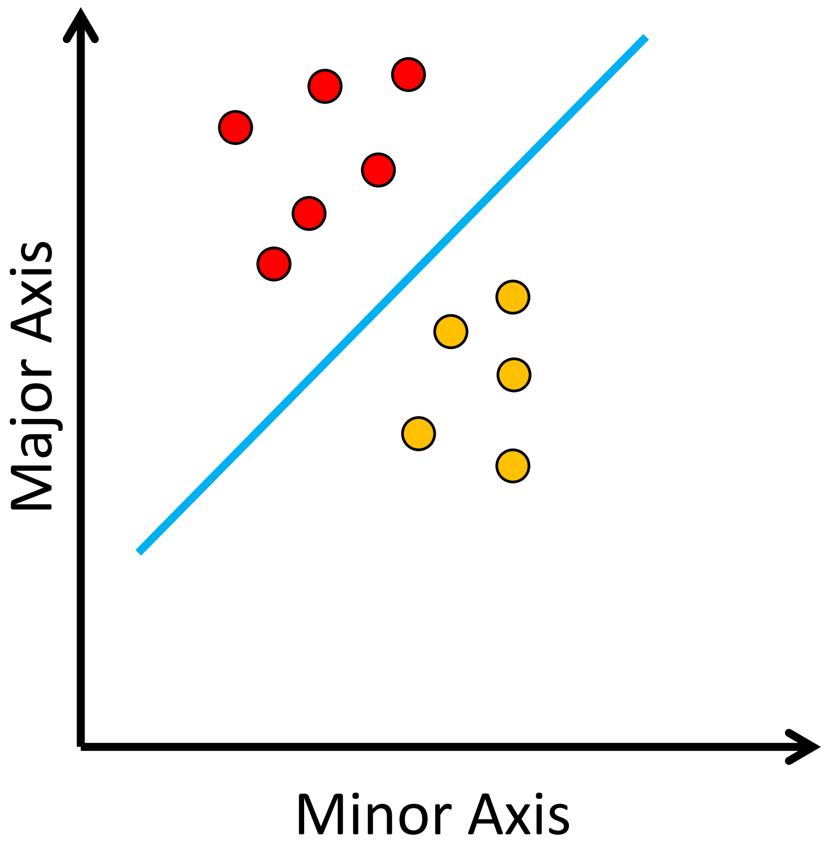
Example: let’s classify hazelnuts and almonds



Hazelnuts vs Almonds

Feature Extraction

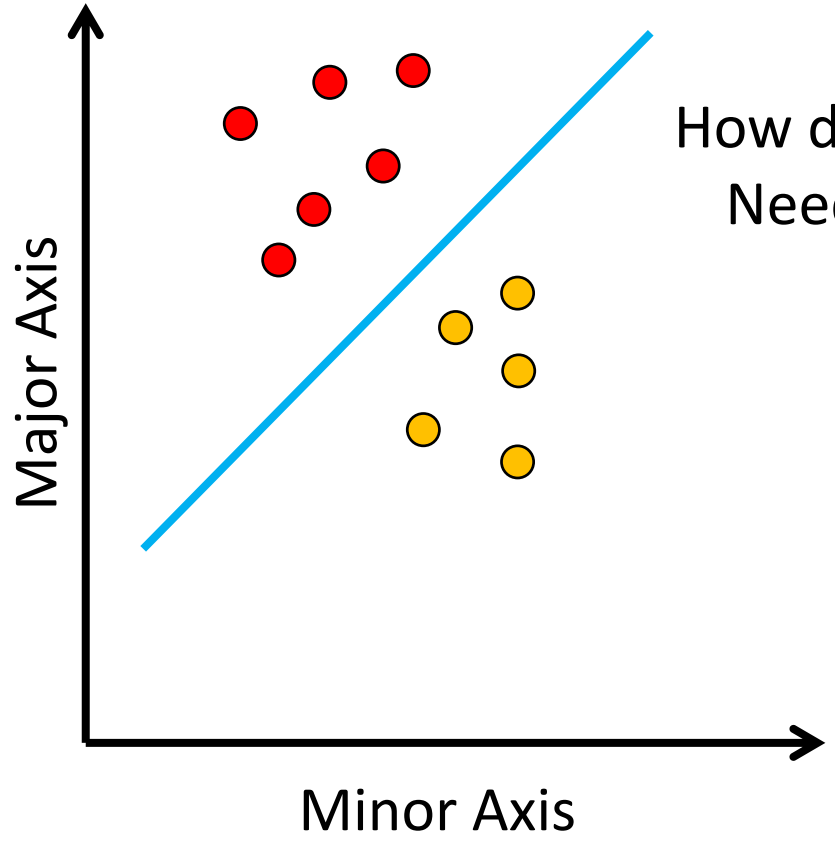
Example: let's classify hazelnuts and almonds



Hazelnuts vs Almonds

Feature Extraction

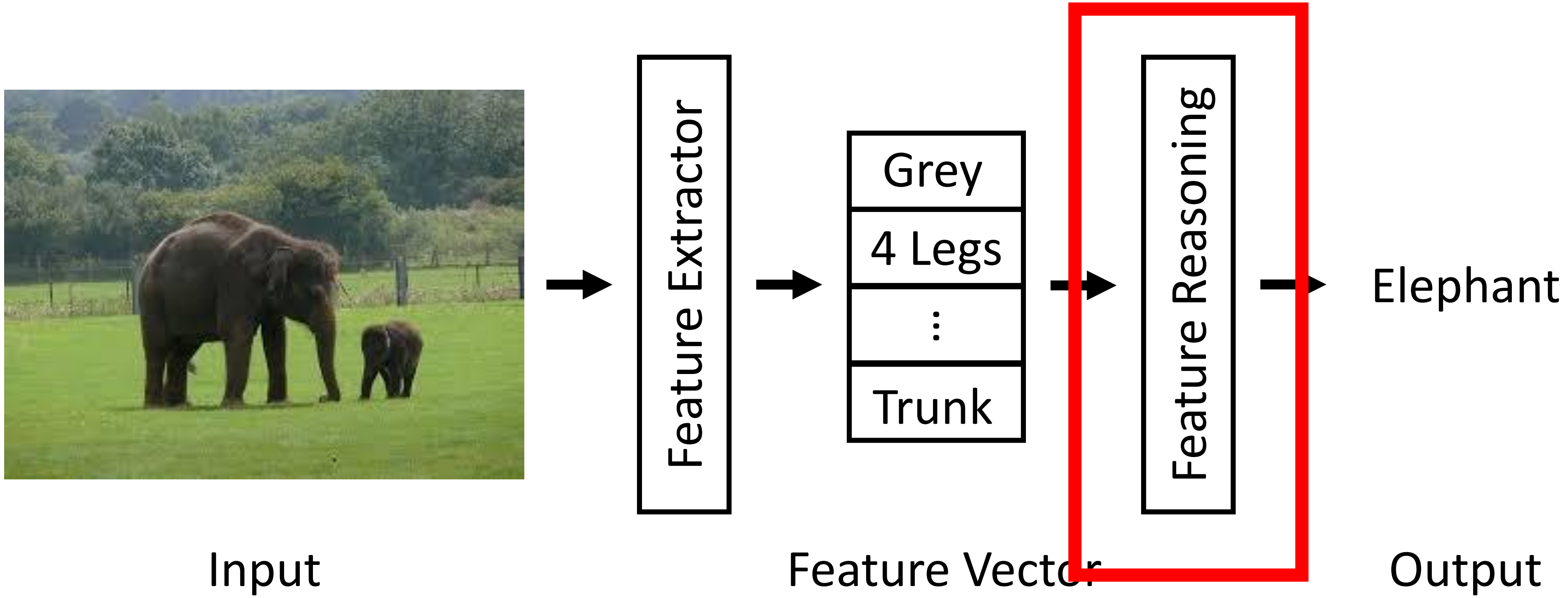
Example: let’s classify hazelnuts and almonds



How do we obtain this line?
Need Machine Learning

Hazelnuts vs Almonds

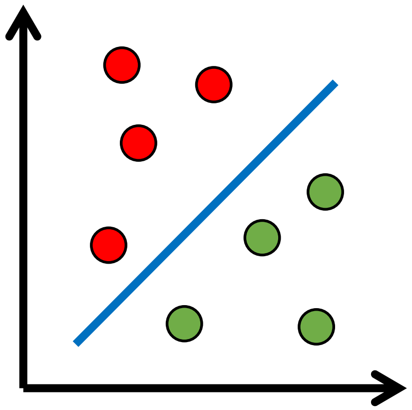
Machine Learning



What animal is in this image?

Machine Learning

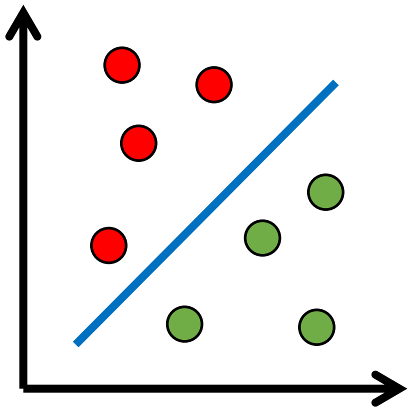
Decision Boundary for Binary Classification



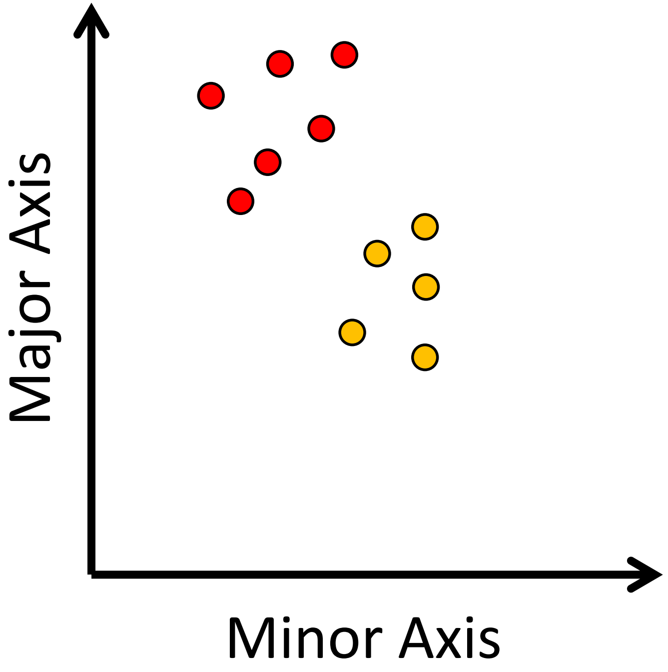
Linear Boundary

Machine Learning

Decision Boundary for Binary Classification



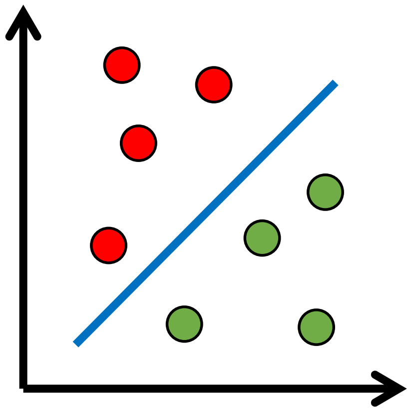
Linear Boundary



Hazelnuts vs Almonds

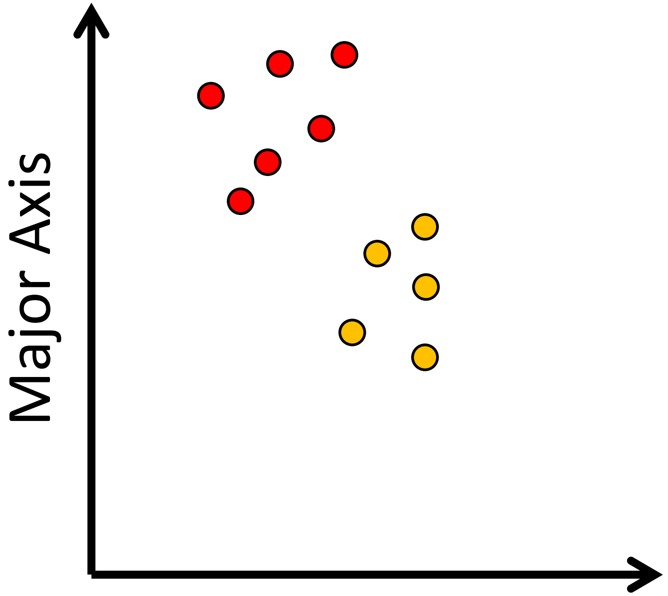
Machine Learning

Decision Boundary for Binary Classification



Linear Boundary

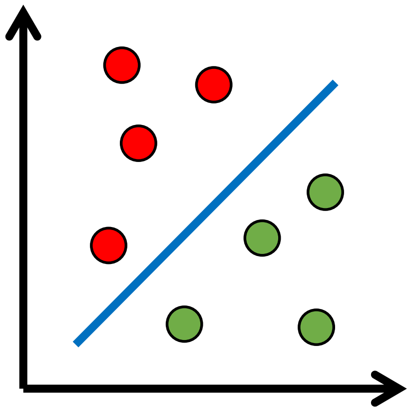
Linear Support Vector Machine



Hazelnuts vs Almonds

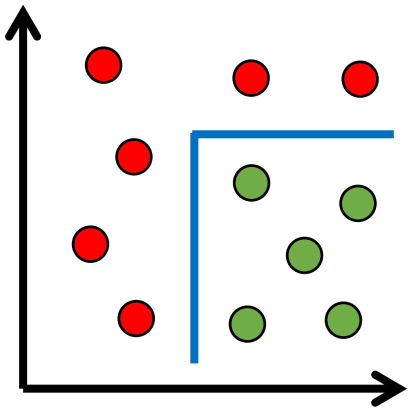
Machine Learning

Decision Boundary for Binary Classification



Linear Boundary

Linear Support Vector Machine

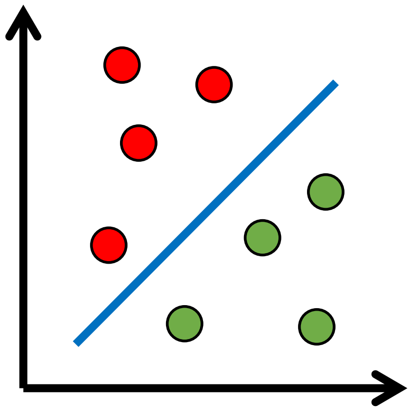


Axis Aligned Non-Linear Boundary

Decision Tree

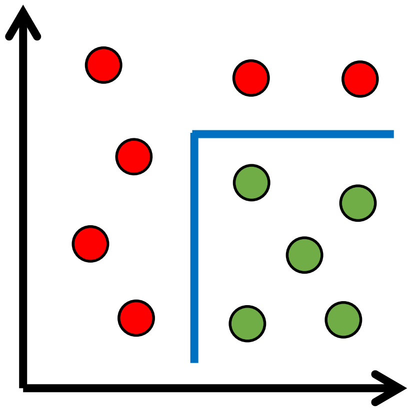
Machine Learning

Decision Boundary for Binary Classification



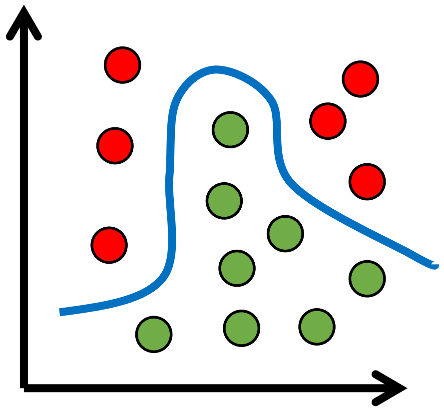
Linear Boundary

Linear Support Vector Machine



Axis Aligned Non-Linear Boundary

Decision Tree



Complex Non-Linear Boundary

Neural Network

When Should You Use Deep Learning?

Classical Machine Learning	Deep Learning
Interpretability and explainability is a priority.	Very high accuracy is a priority.
Smaller amounts of relatively simple data.	Large amounts of precisely labeled data.
Straightforward feature engineering.	Complex feature engineering.

https://blog.dataiku.com/when-and-when-not-to-use-deep-learning

When Should You Use Deep Learning?

Classical Machine Learning	Deep Learning
K-Means Clustering	Fully Connected Network
Linear Regression	Convolutional Neural Network
Decision Trees	
Random Forest	

https://blog.dataiku.com/when-and-when-not-to-use-deep-learning

When Should You Use Deep Learning?

→ Feature Extractor →

Grey
4 Legs
⋮
Trunk

→ Feature Reasoning → Elephant

Input

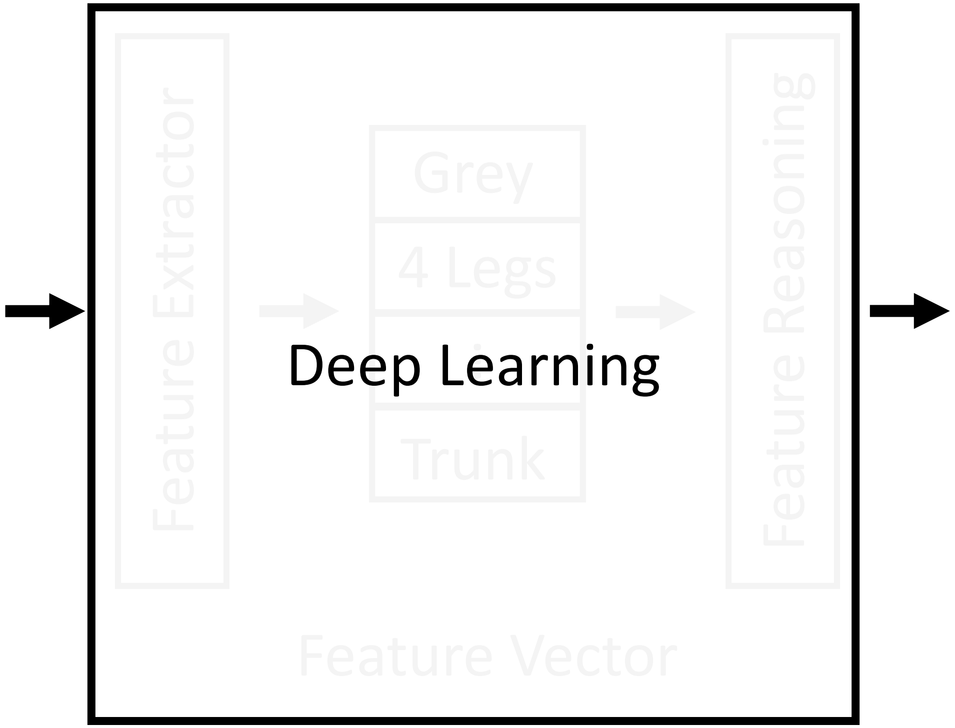
Feature Vector

Output

What animal is in this image?

When Should You Use Deep Learning?

Input



Elephant

Output

What animal is in this image?

Deep Learning for Computer Vision

What specific questions can we answer?

Circa 1966: Marvin Minsky at MIT gives Gerald Jay Sussman, an undergraduate student, a summer project to link a computer to a camera and get the computer to "describe what it saw".

Deep Learning for Computer Vision

What specific questions can we answer?

Circa 1966: Marvin Minsky at MIT gives Gerald Jay Sussman, an undergraduate student, a summer project to link a computer to a camera and get the computer to "describe what it saw".

How would you describe this image?

Deep Learning for Computer Vision

What specific questions can we answer?

Circa 1966: Marvin Minsky at MIT gives Gerald Jay Sussman, an undergraduate student, a summer project to link a computer to a camera and get the computer to "describe what it saw".

How would you describe this image?

First: What does "describe this" even mean?

Deep Learning for Computer Vision

Deep Learning Tasks

- Binary Classification (binary label):
  "Is there an elephant in this image?"

Yes

Deep Learning for Computer Vision

Deep Learning Tasks

- Binary Classification (binary label):
  "Is there an elephant in this image?"
- Classification (integer label):
  "What animal is in this image?"

An Elephant

Deep Learning for Computer Vision

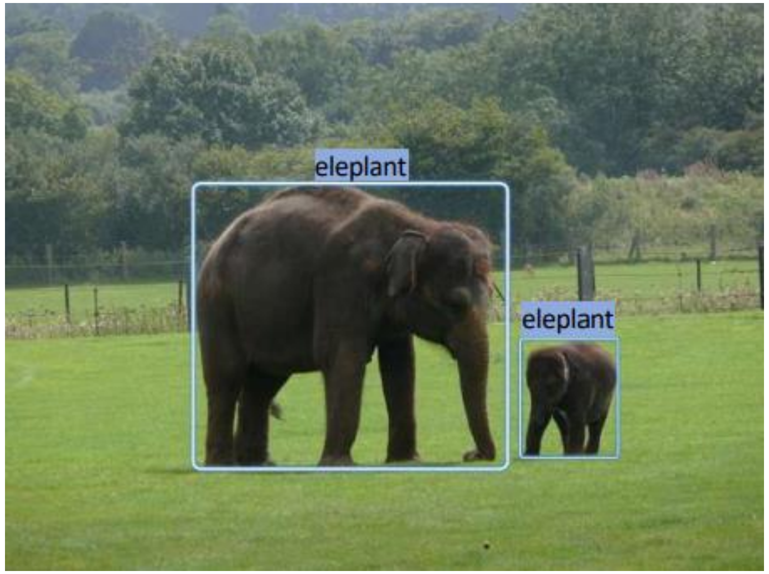
Deep Learning Tasks

- Binary Classification (binary label):
  "Is there an elephant in this image?"
- Classification (integer label):
  "What animal is in this image?"
- Regression (real number):
  "How old is the elephant in this image?" 10.5 Years

Deep Learning for Computer Vision

Deep Learning Tasks

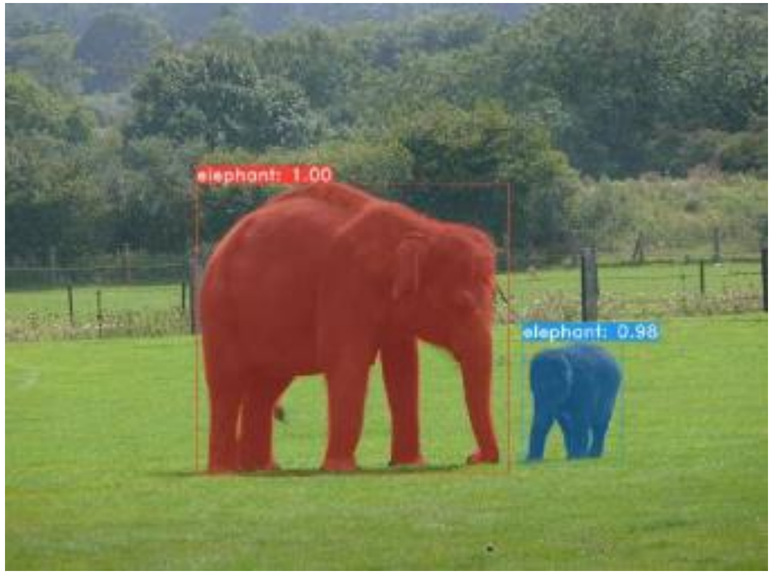
- Binary Classification (binary label):
  “Is there an elephant in this image?”
- Classification (integer label):
  “What animal is in this image?”
- Regression (real number):
  “How old is the elephant in this image?”
- Detection (bounding box):
  “Where is the elephant in this image?”



Deep Learning for Computer Vision

Deep Learning Tasks

- Binary Classification (binary label):
  “Is there an elephant in this image?”
- Classification (integer label):
  “What animal is in this image?”
- Regression (real number):
  “How old is the elephant in this image?”
- Detection (bounding box):
  “Where is the elephant in this image?”
- Segmentation (pixel mask):
  “Which pixels in this image are elephant?”



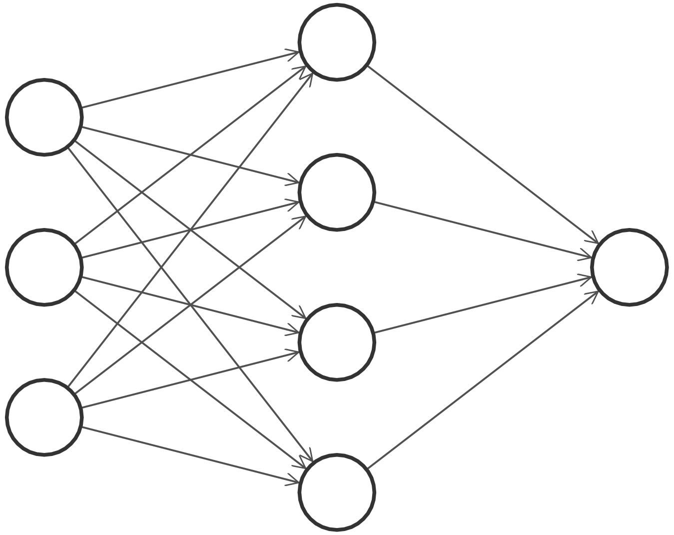
Deep Learning for Computer Vision

We know what specific questions we can ask.

But how do we get a computer to answer them?

Neural Networks

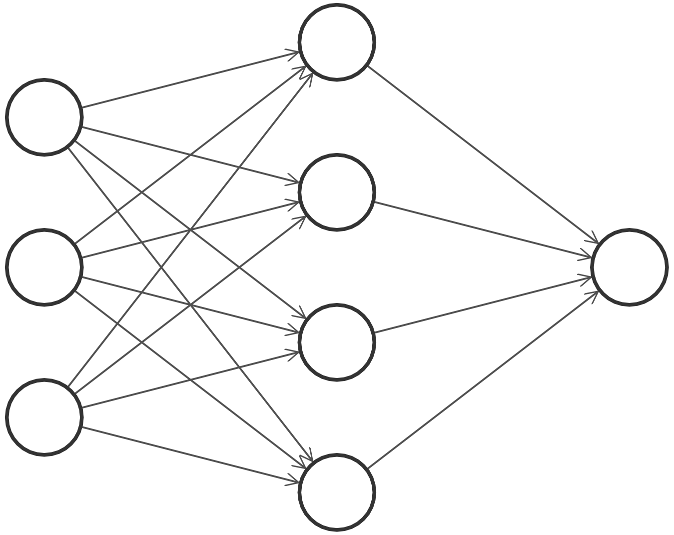
Fully Connected Network (FCN)



Idea: learn a bunch of functions that make incrementally more complex decisions.

Neural Networks

Fully Connected Network (FCN)

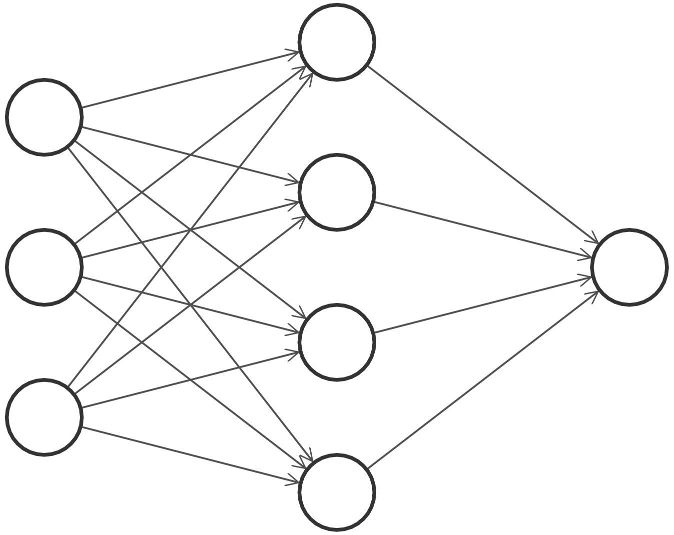


input layer

input your data here

Neural Networks

Fully Connected Network (FCN)

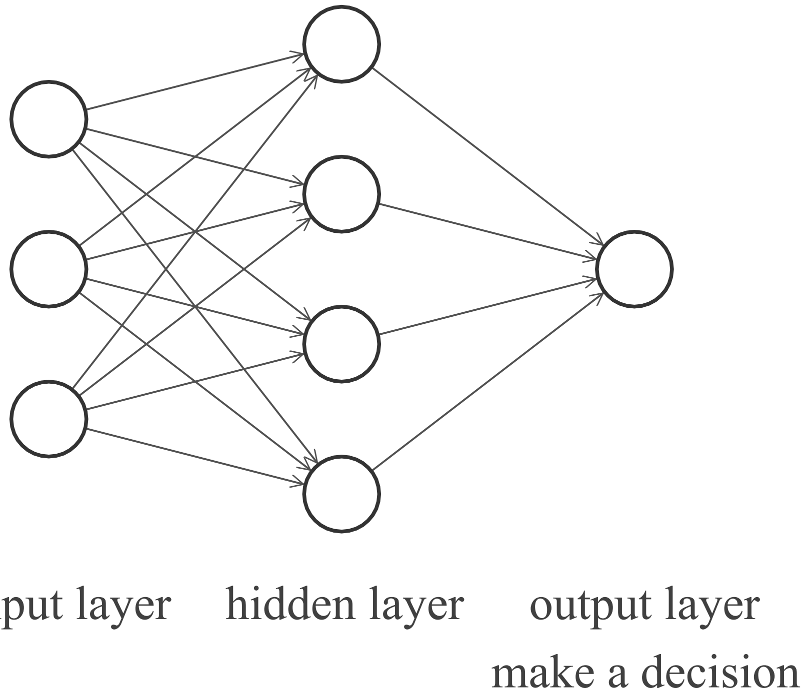


input layer   hidden layer

extract some features and
do some reasoning

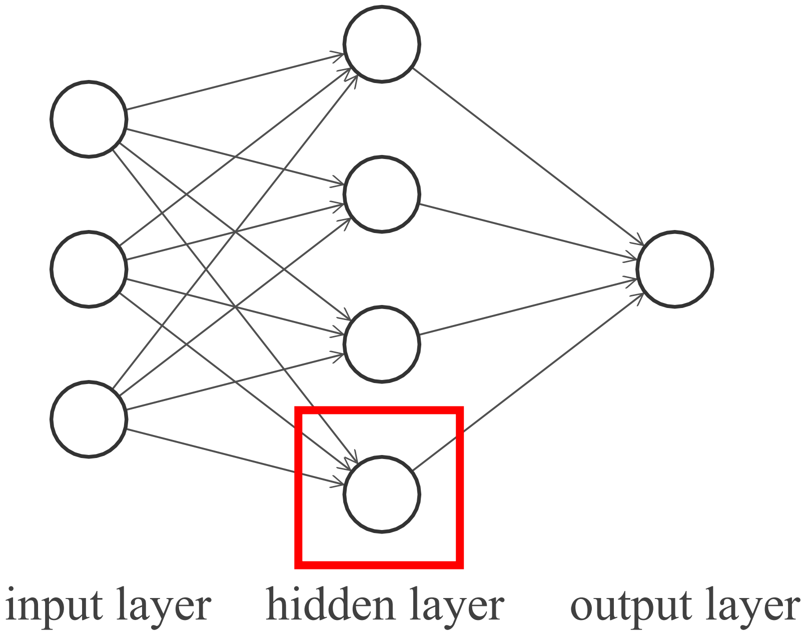
Neural Networks

Fully Connected Network (FCN)



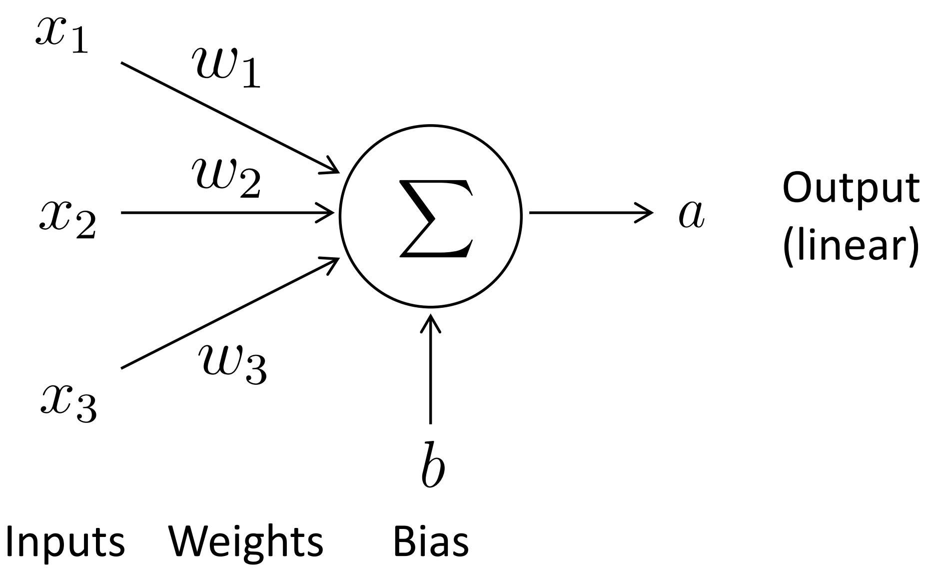
Neural Networks

Fully Connected Network (FCN)



Neural Networks

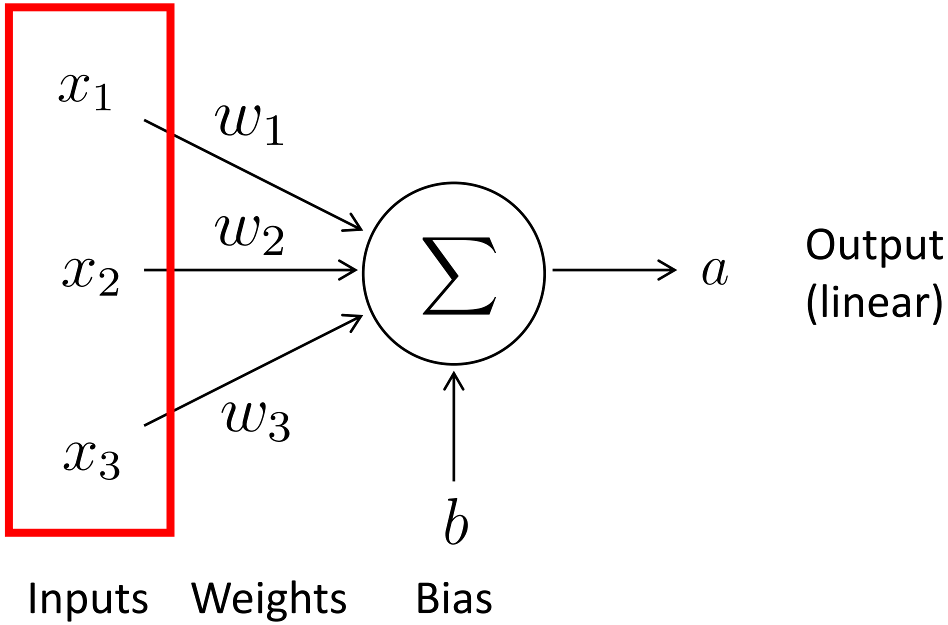
Building Blocks: Perceptron



$$a = \sum_i w_i x_i + b$$

Neural Networks

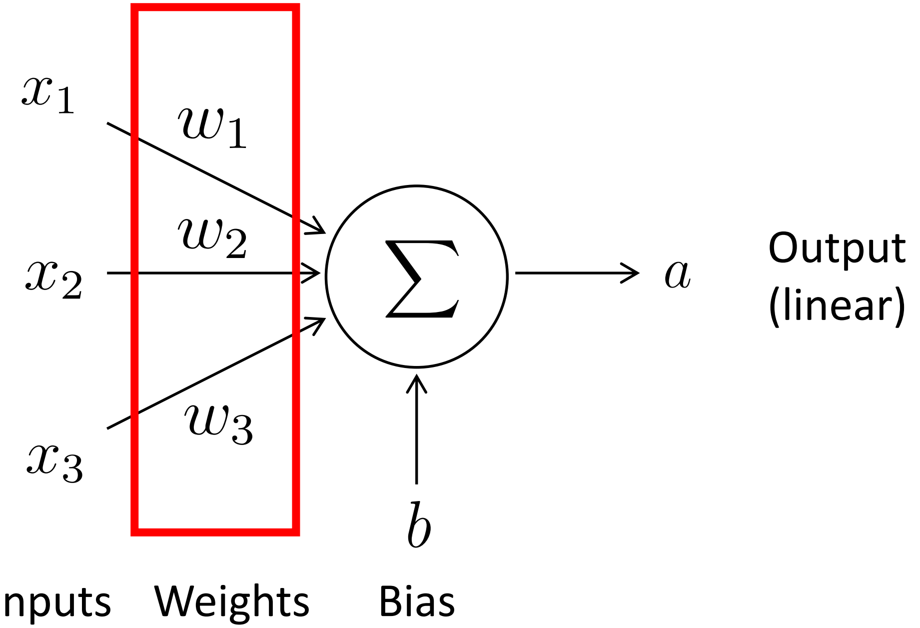
Building Blocks: Perceptron



$$a = \sum_i w_i x_i + b$$

Neural Networks

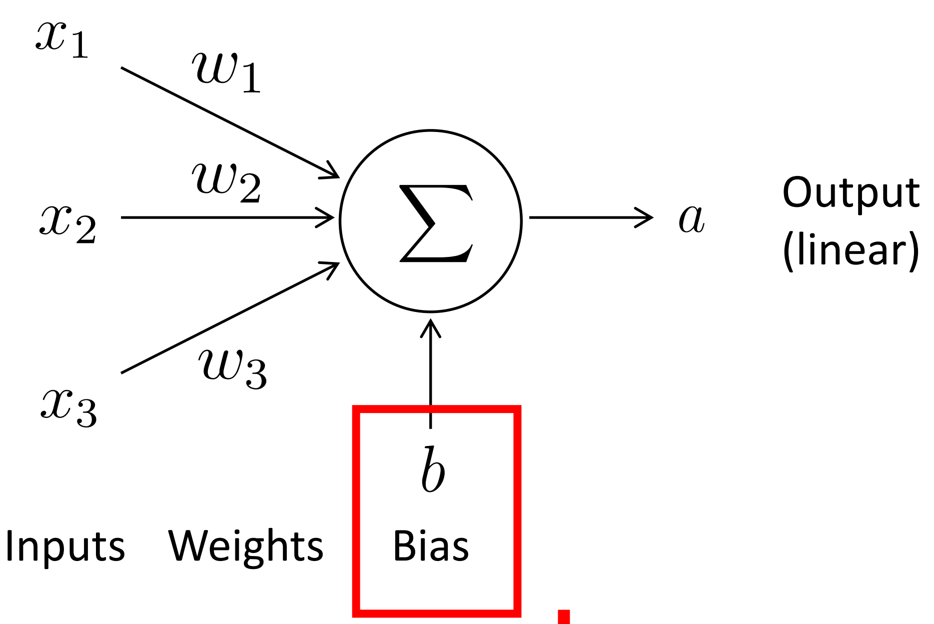
Building Blocks: Perceptron



$$a = \sum_i w_i x_i + b$$

Neural Networks

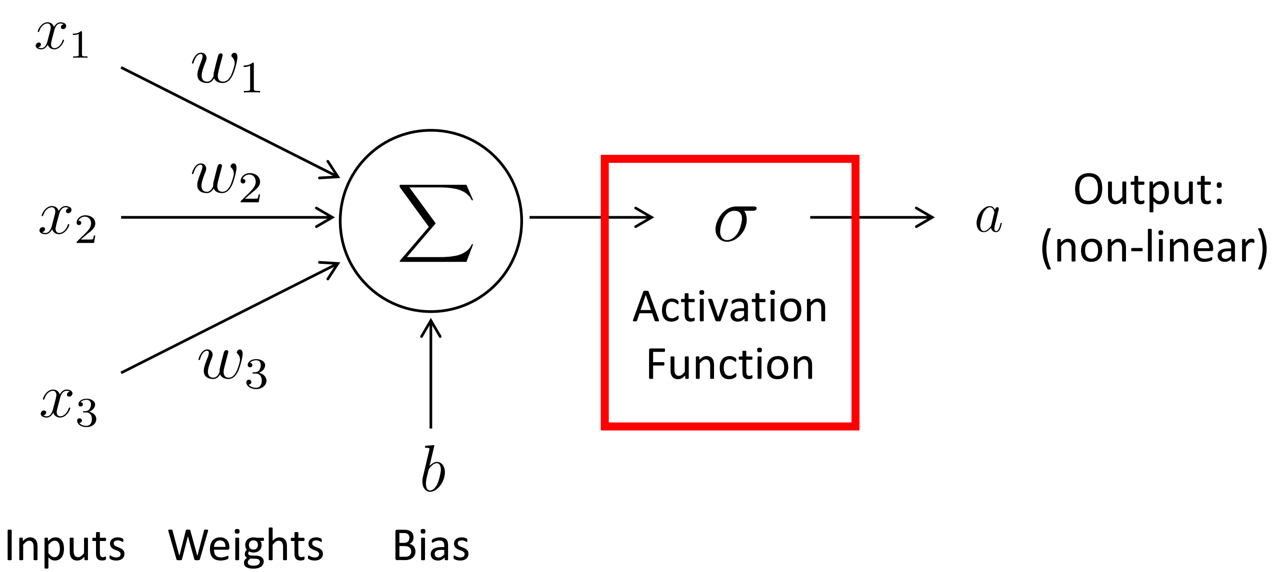
Building Blocks: Perceptron



$$a = \sum_i w_i x_i + b$$

Neural Networks

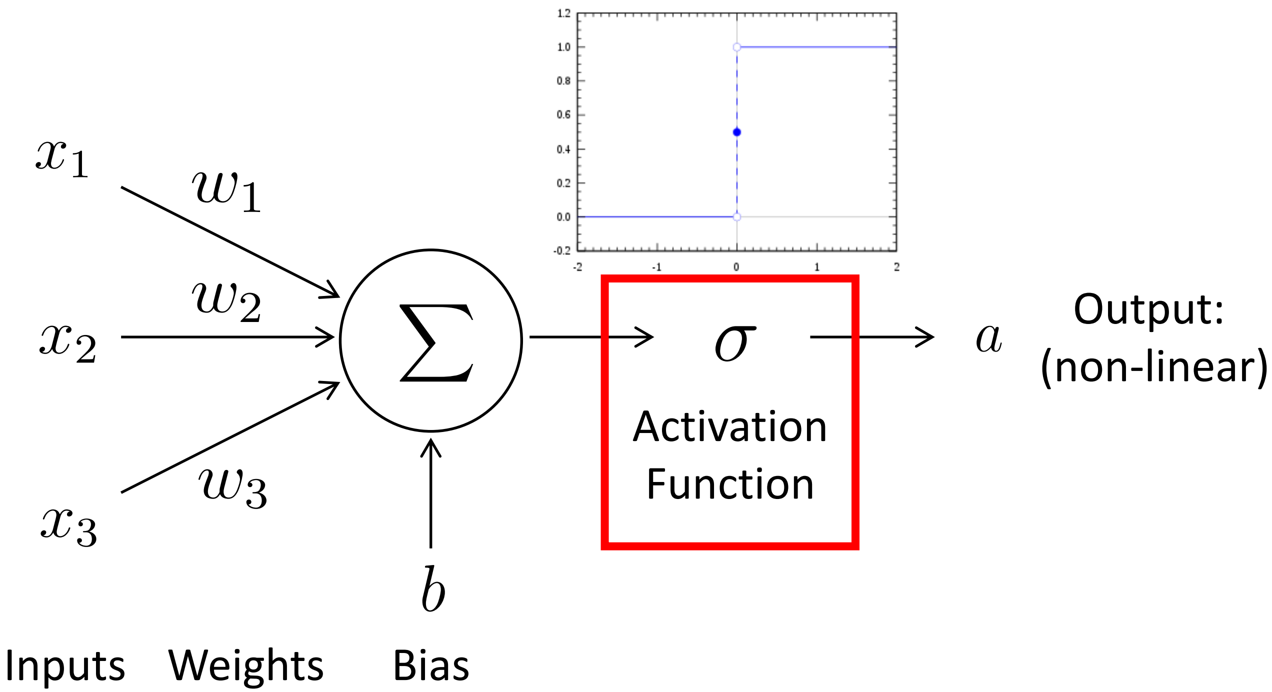
Building Blocks: Perceptron



$$a = \sigma\left(\sum_i w_i x_i + b\right)$$

Neural Networks

Building Blocks: Perceptron

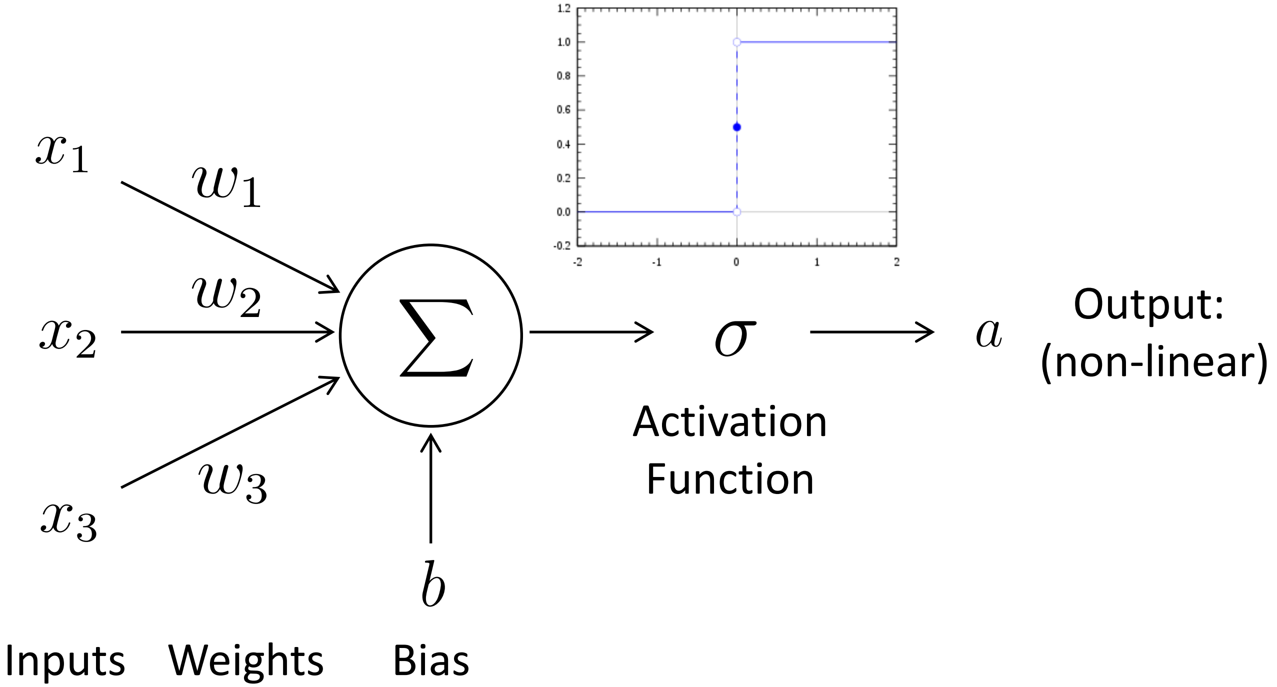


$$a = \sigma(\sum_i w_i x_i + b)$$

Neural Networks

Building Blocks: Perceptron

Neurons (or Nodes) are based on biological neurons that “fire” if the linear output is high. The “firing” is modeled by the activation function.



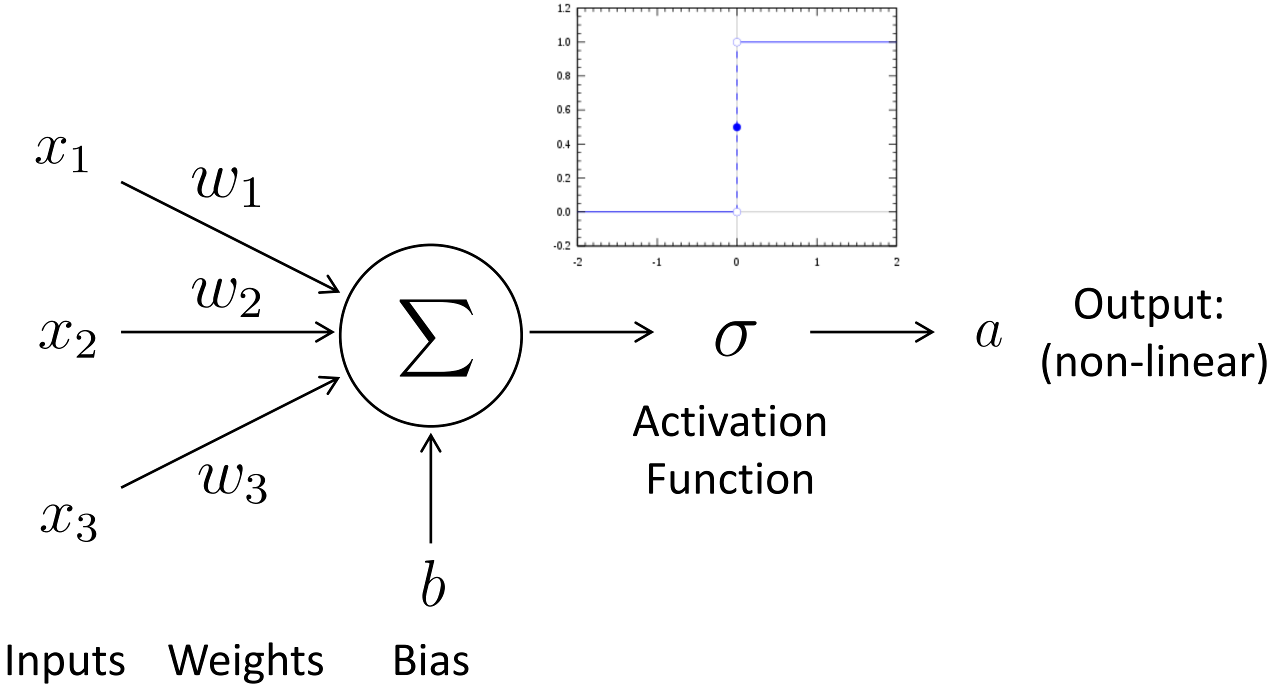
$$a = \sigma(\sum_i w_i x_i + b)$$

Neural Networks

Building Blocks: Perceptron

Neurons (or Nodes) are based on biological neurons that "fire" if the linear output is high. The "firing" is modeled by the activation function.

Neurons correspond to feature descriptors that "fire" when the "see" their corresponding feature.



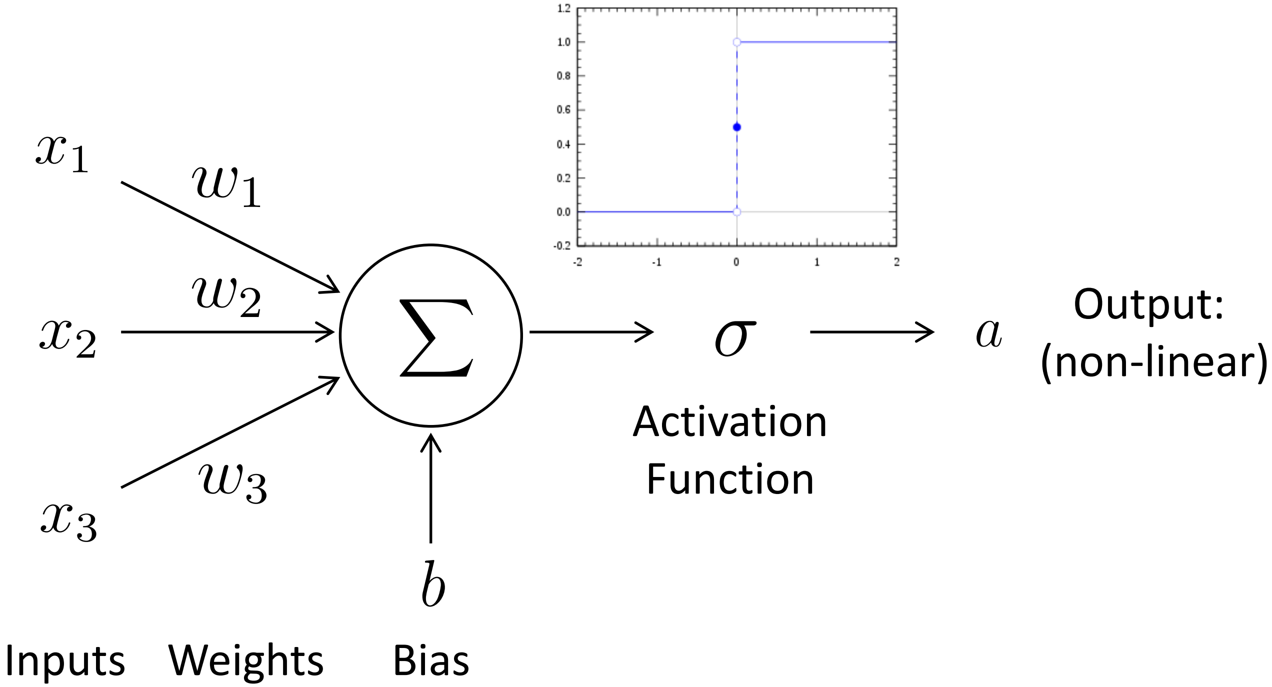
$$a = \sigma(\sum_i w_i x_i + b)$$

Neural Networks

Building Blocks: Perceptron

Neurons (or Nodes) are based on biological neurons that "fire" if the linear output is high. The "firing" is modeled by the activation function.

Neurons correspond to feature descriptors that "fire" when the "see" their corresponding feature.



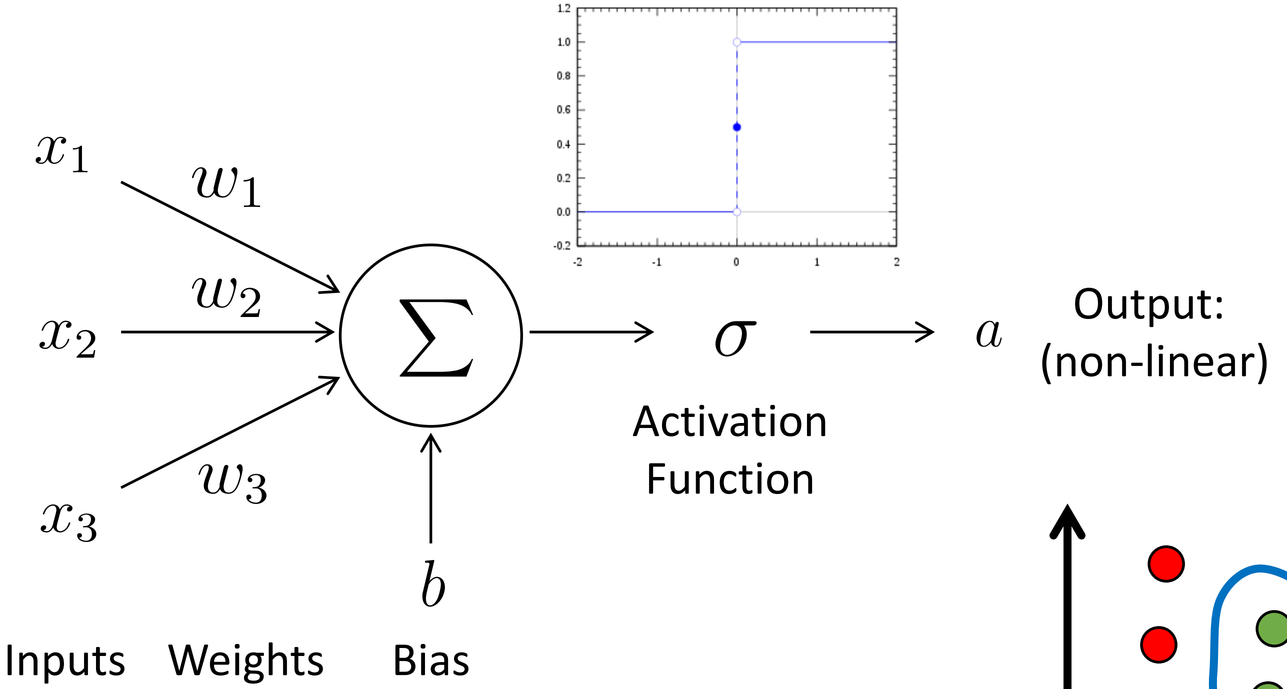
$$a = \sigma(\sum_i w_i x_i + b)$$

Neural Networks

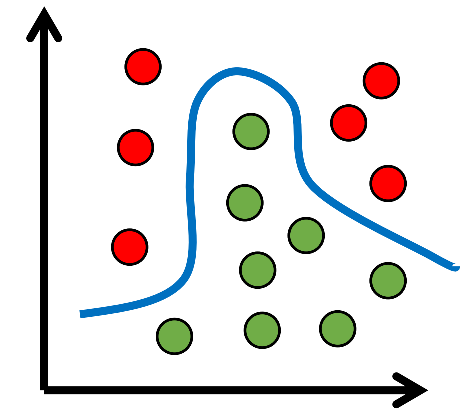
Building Blocks: Perceptron

Neurons (or Nodes) are based on biological neurons that “fire” if the linear output is high. The “firing” is modeled by the activation function.

Neurons correspond to feature descriptors that “fire” when the “see” their corresponding feature.

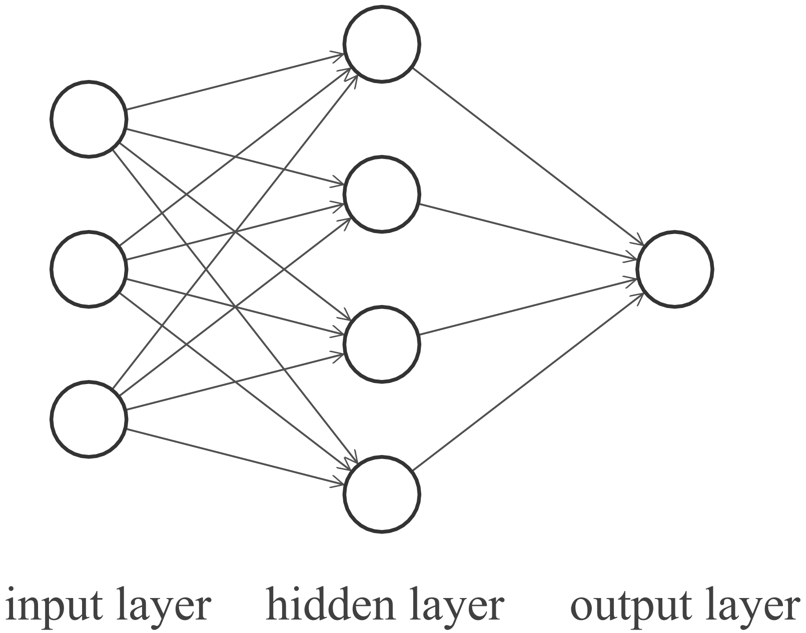


$$a = \sigma(\sum_i w_i x_i + b)$$



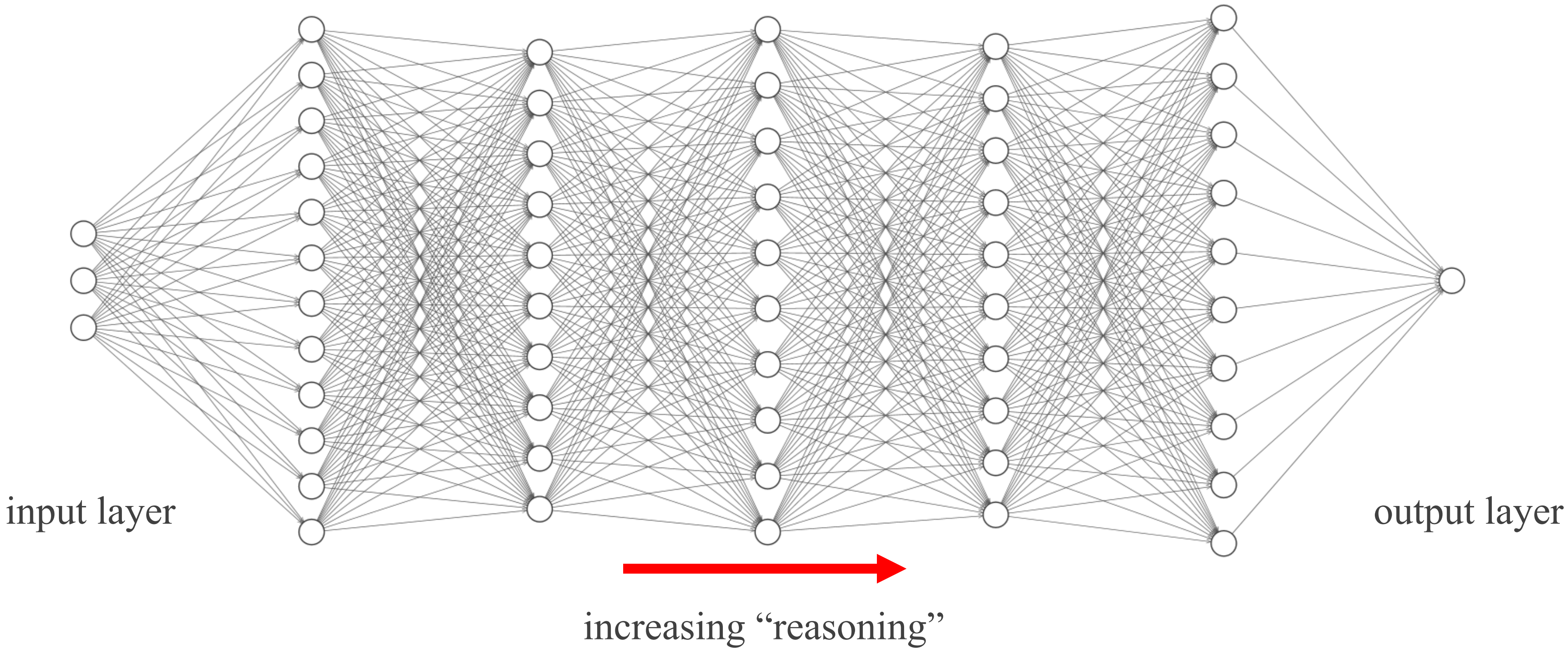
Neural Networks

String many functions together to answer “reason” about the input



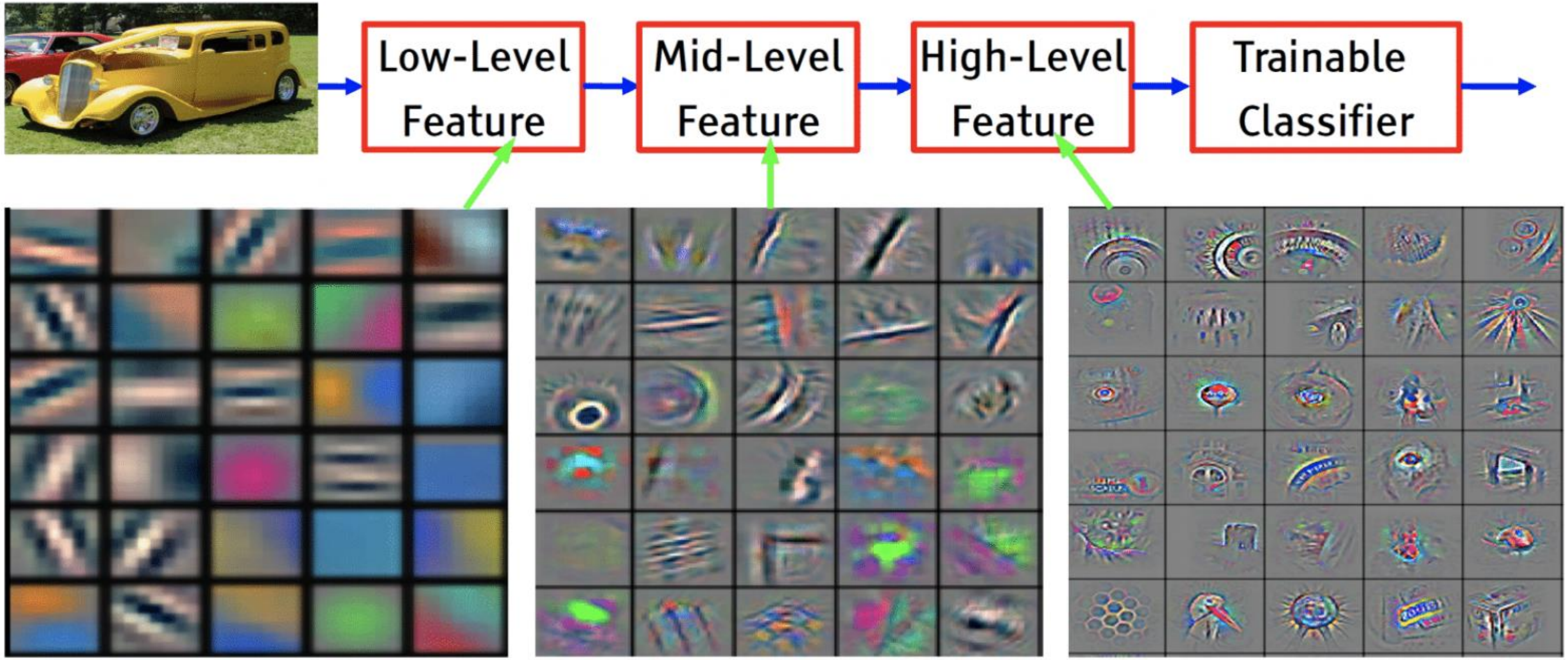
Neural Networks

String many functions together to answer “reason” about the input



Neural Networks

String many functions together to answer "reason" about the input



Neural Networks

Building Blocks: Network Size

How many nodes/layers should I use?

“Width and depth are both important and should be carefully tuned… **depth may determine the abstraction level** but the **width may influence the loss of information in the forwarding pass**.”

https://proceedings.neurips.cc/paper/2017/file/32cbf687880eb1674a07bf717761dd3a-Paper.pdf

Neural Networks

Building Blocks: Network Size

How many nodes/layers should I use?

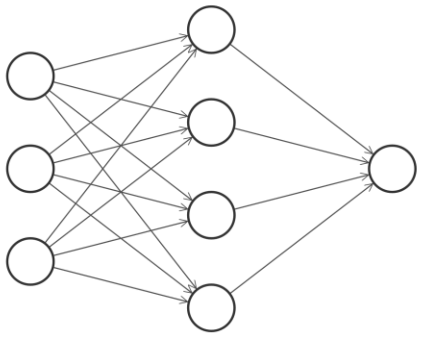
"Width and depth are both important and should be carefully tuned… **depth may determine the abstraction level** but the **width may influence the loss of information in the forwarding pass**."

At a minimum you should have more nodes than the dimensionality of your input.

https://proceedings.neurips.cc/paper/2017/file/32cbf687880eb1674a07bf717761dd3a-Paper.pdf

Convolutional Neural Network

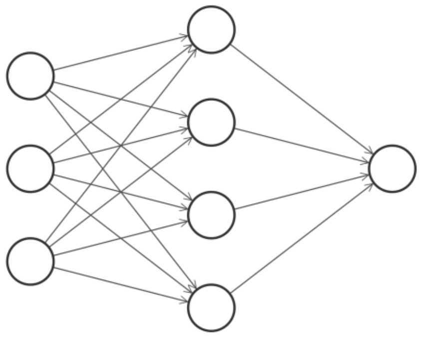
How do we input images to a Neural Network?



Can’t use a FCN

Convolutional Neural Network

How do we input images to a Neural Network?



Can’t use a FCN

Let’s input a 128x128 image

Need 16,384 input nodes

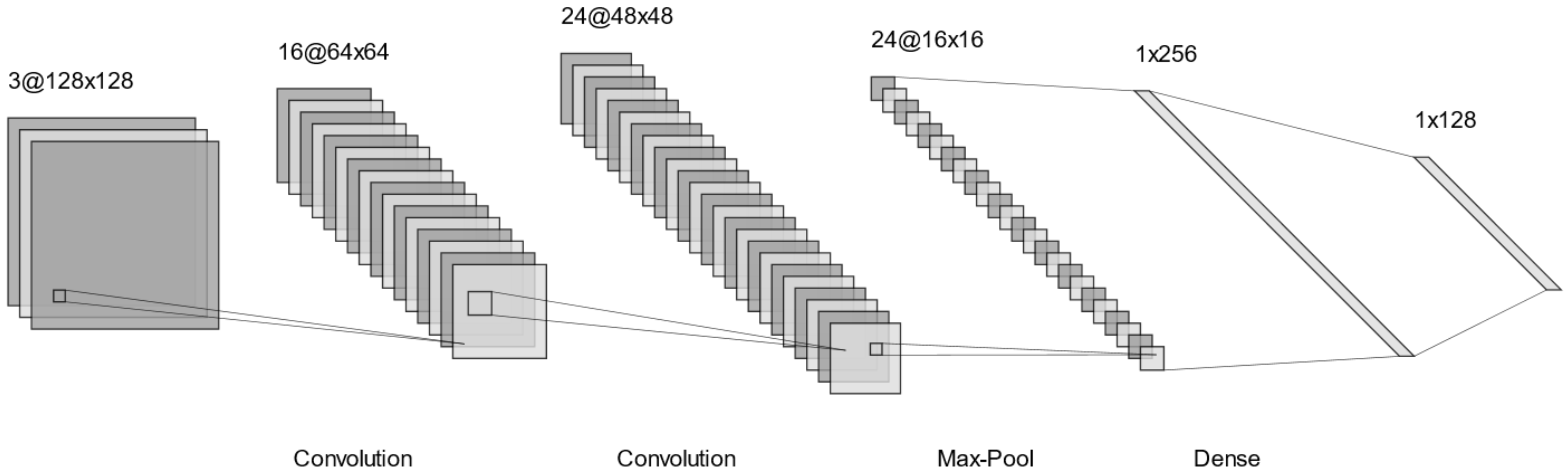
Assume 3 layers with 16,384 nodes each

(16,384x2x16,384)x3 + 2x16,384 = 1.6 billion

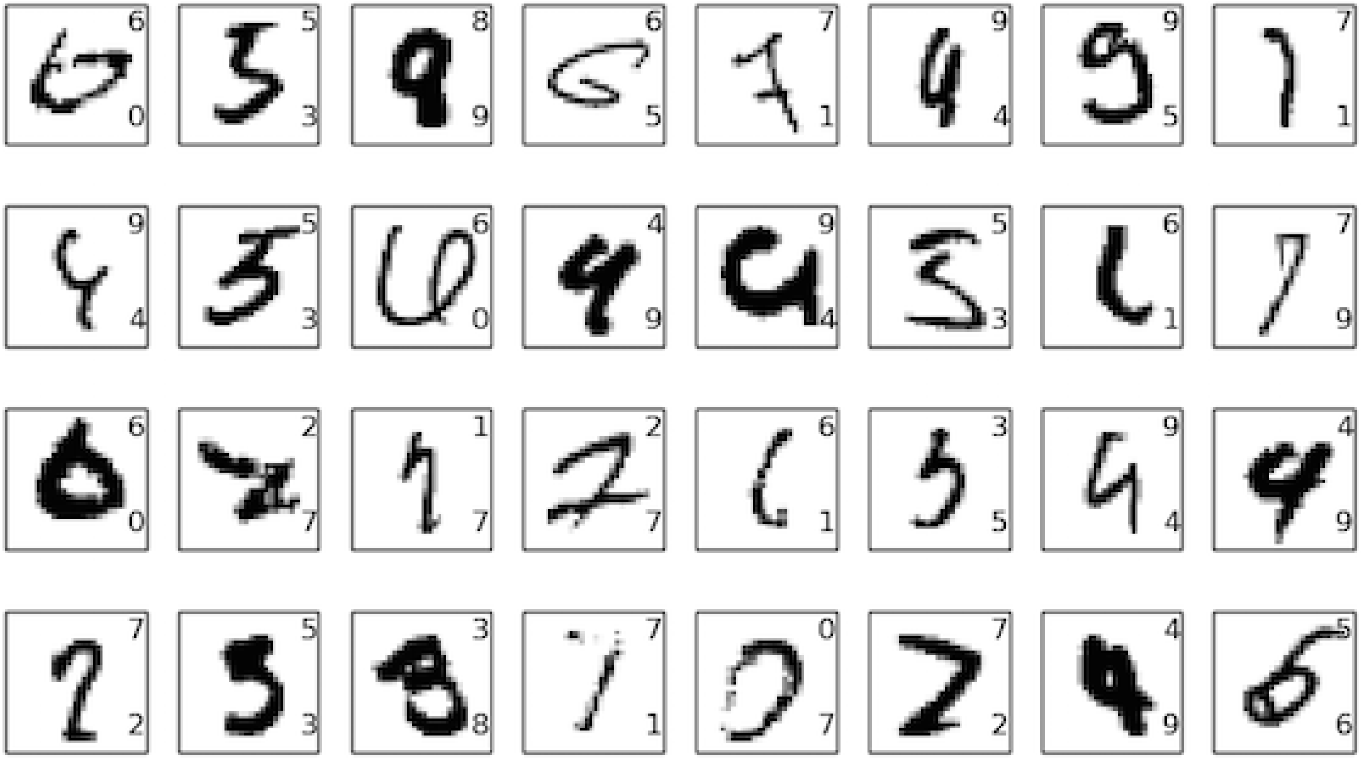
This is not trainable in a reasonable amount of time

Does not model inherent structure of images

Convolutional Neural Network

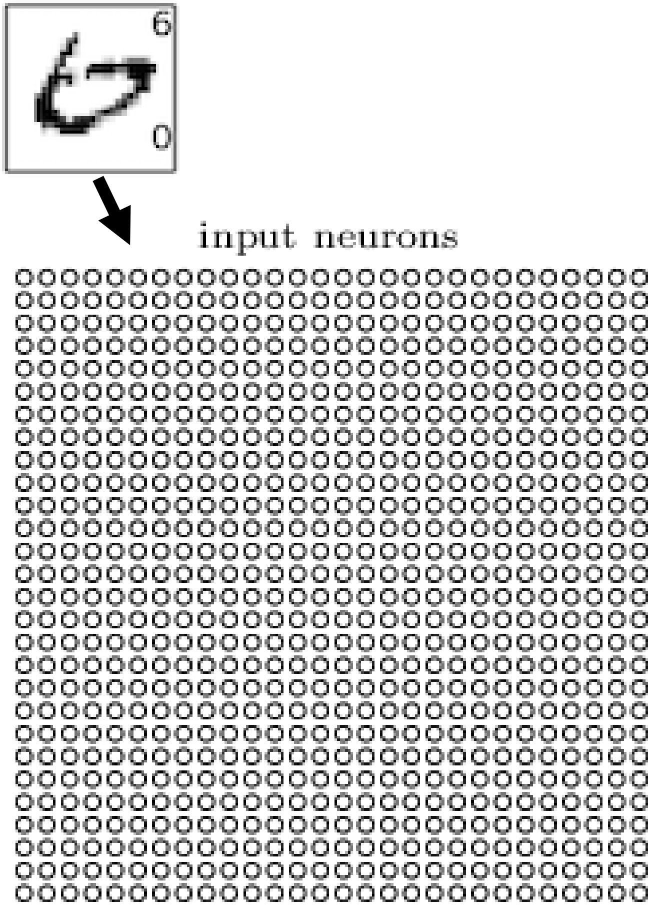


Convolutional Neural Network



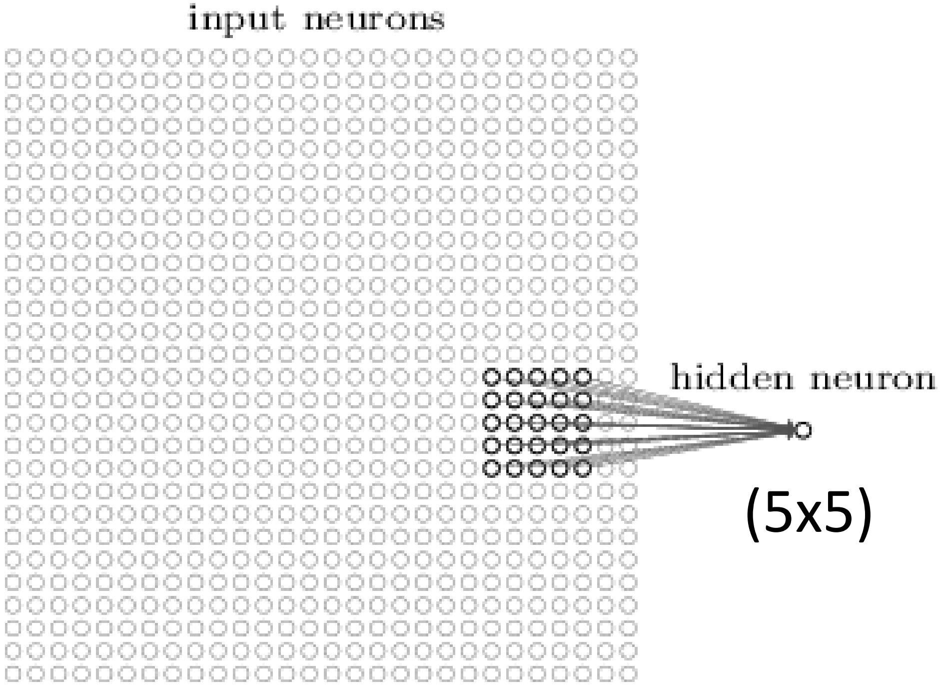
MNIST as an example (28x28px) images

Convolutional Neural Network



(28x28)
Input Values

Convolutional Neural Network



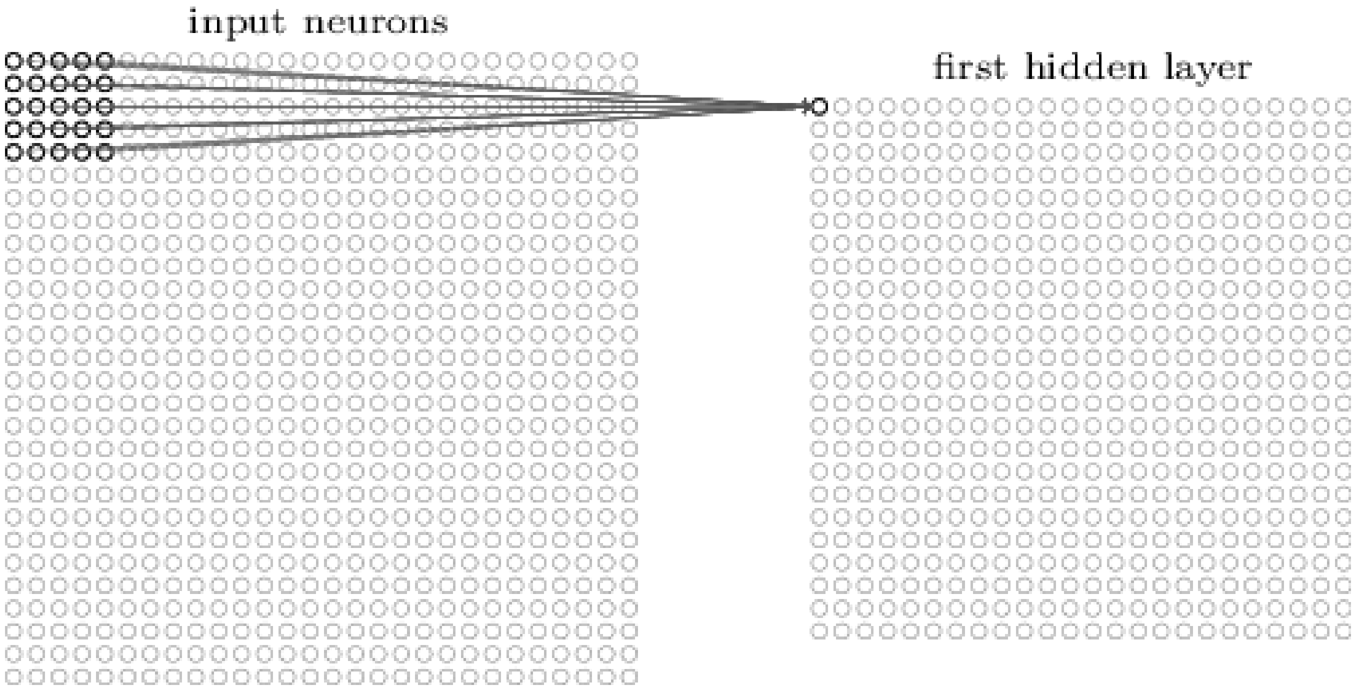
(5x5)

Grid of weights we multiply by the input to get a single output value

(28x28)
Input Values

1. have each hidden neuron process a small window of the input, called the “receptive field”.

Convolutional Neural Network



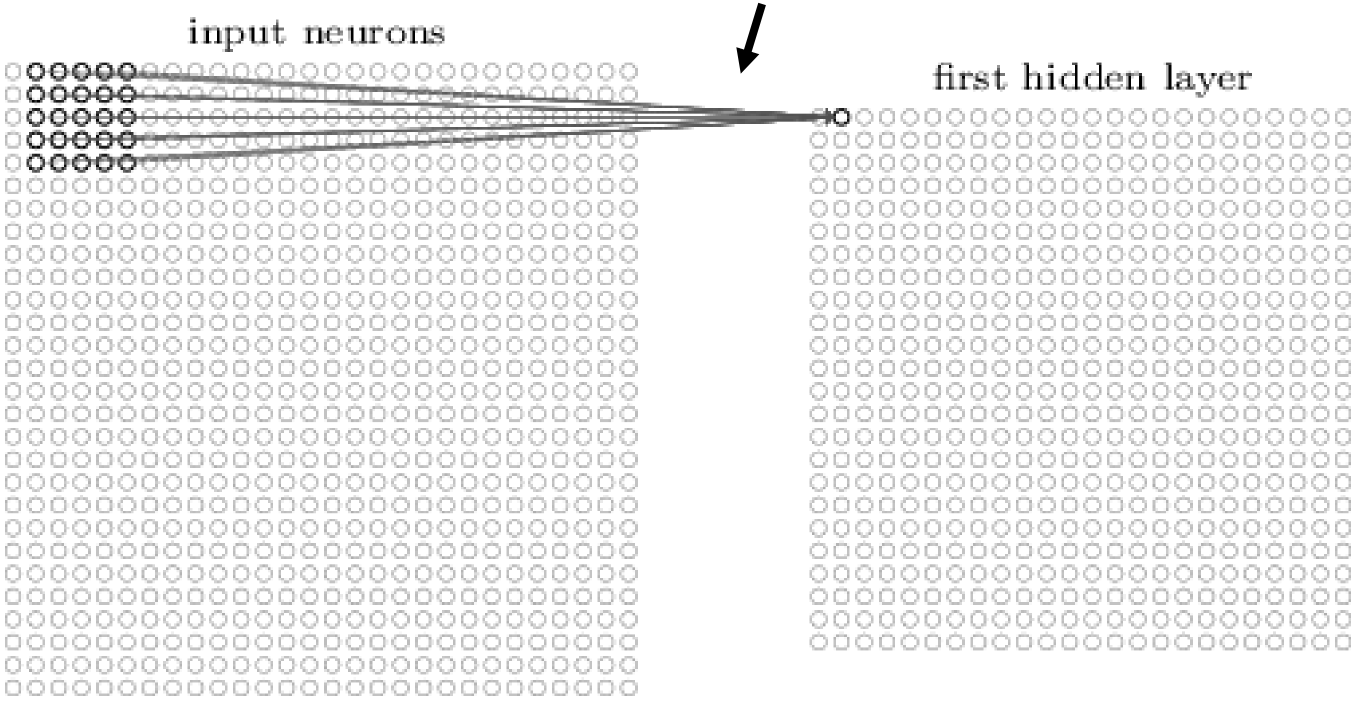
(28x28)
Input Values

(24x24)
Output Values

1. have each hidden neuron process a small window of the input, called the “receptive field”.
2. Slide the receptive field across the image.

Convolutional Neural Network

Same Weights (Kernel)



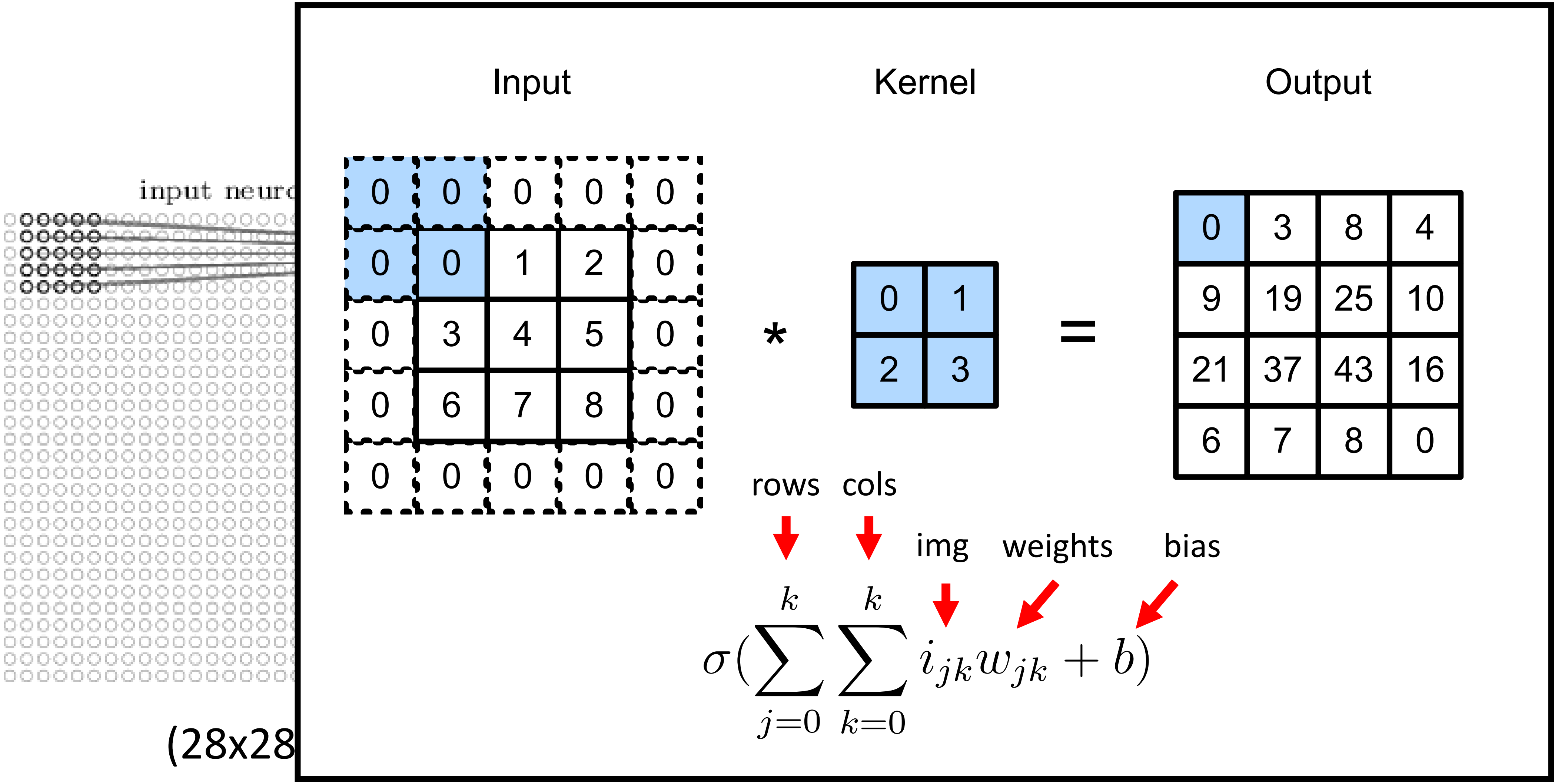
(28x28)
Input Values

(24x24)
Output Values

1. have each hidden neuron process a small window of the input, called the “receptive field”.
2. Slide the receptive field across the image.

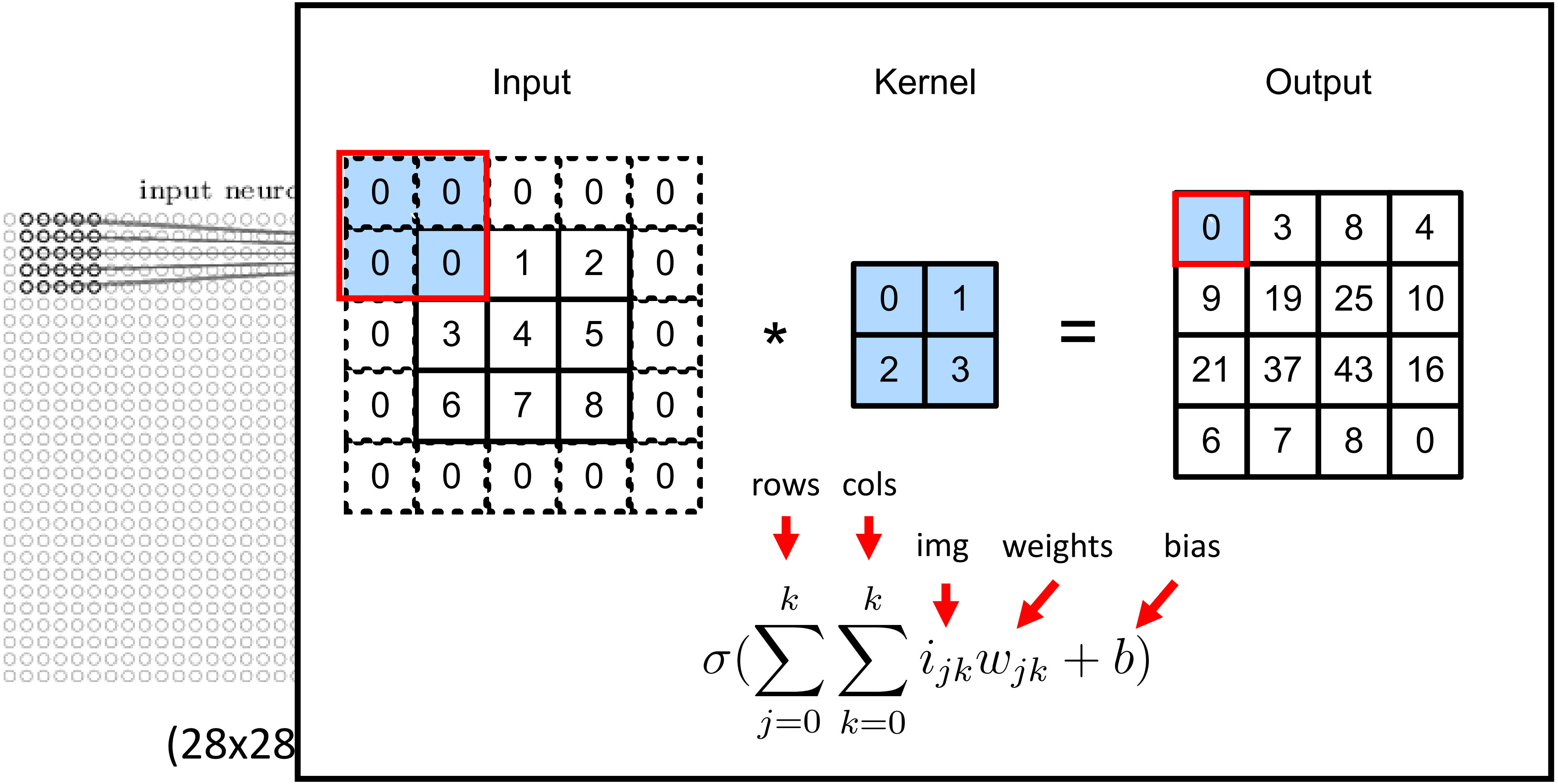
Kernel (or Filter): the weights applied inside the receptive field to obtain the next layer output.

Convolutional Neural Network



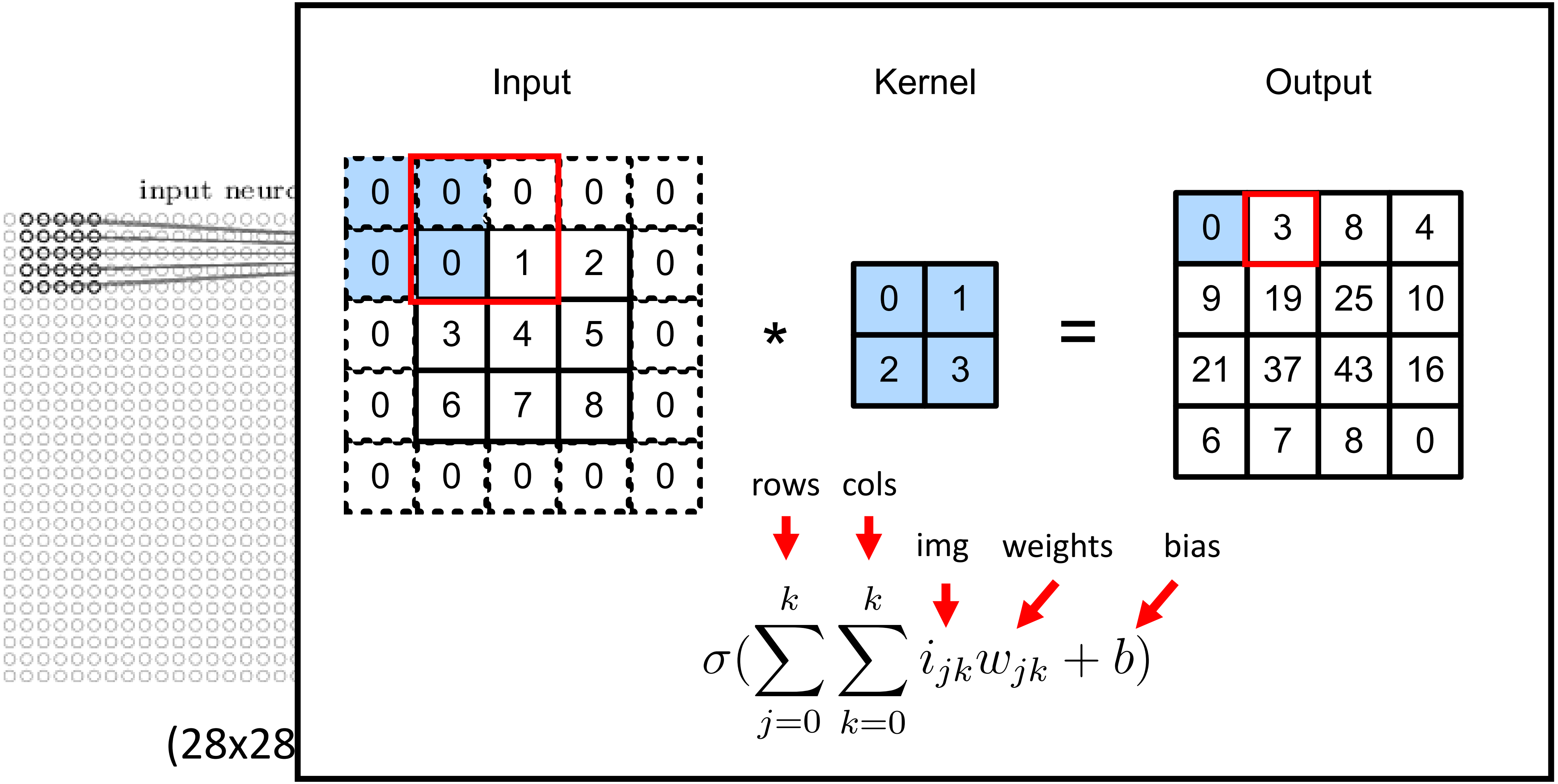
https://d2l.ai/chapter_convolutional-neural-networks/why-conv.html

Convolutional Neural Network



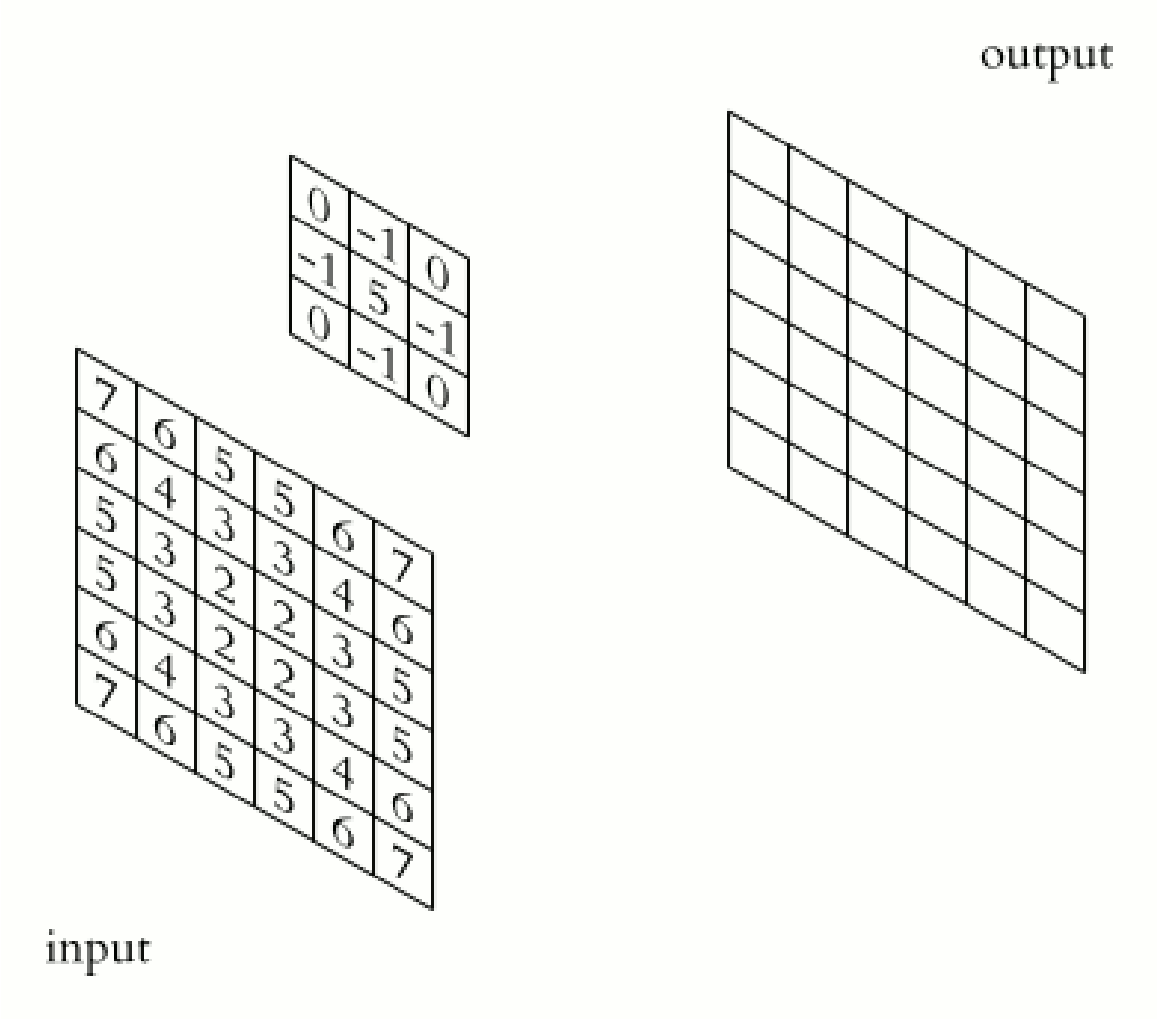
https://d2l.ai/chapter_convolutional-neural-networks/why-conv.html

Convolutional Neural Network

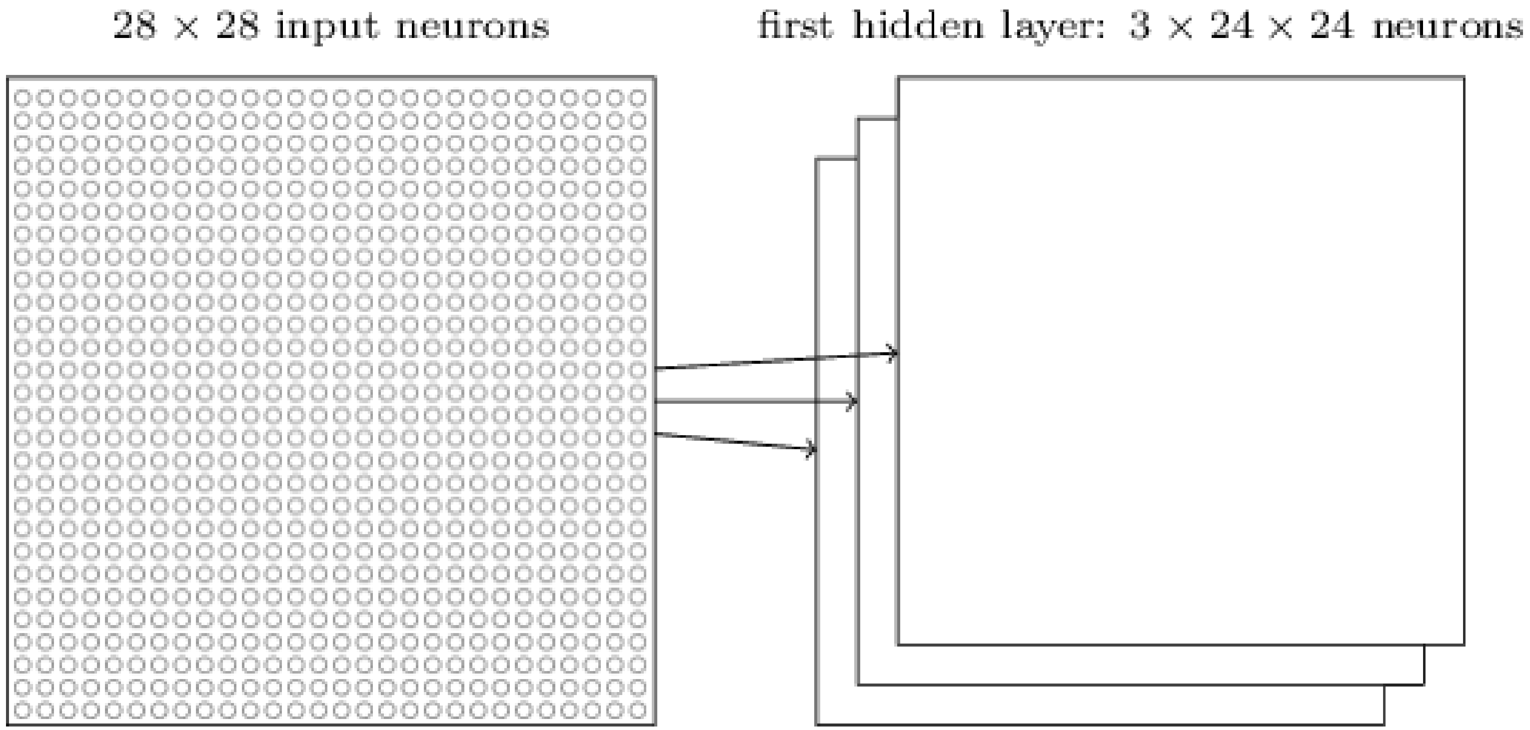


https://d2l.ai/chapter_convolutional-neural-networks/why-conv.html

Convolutional Neural Network

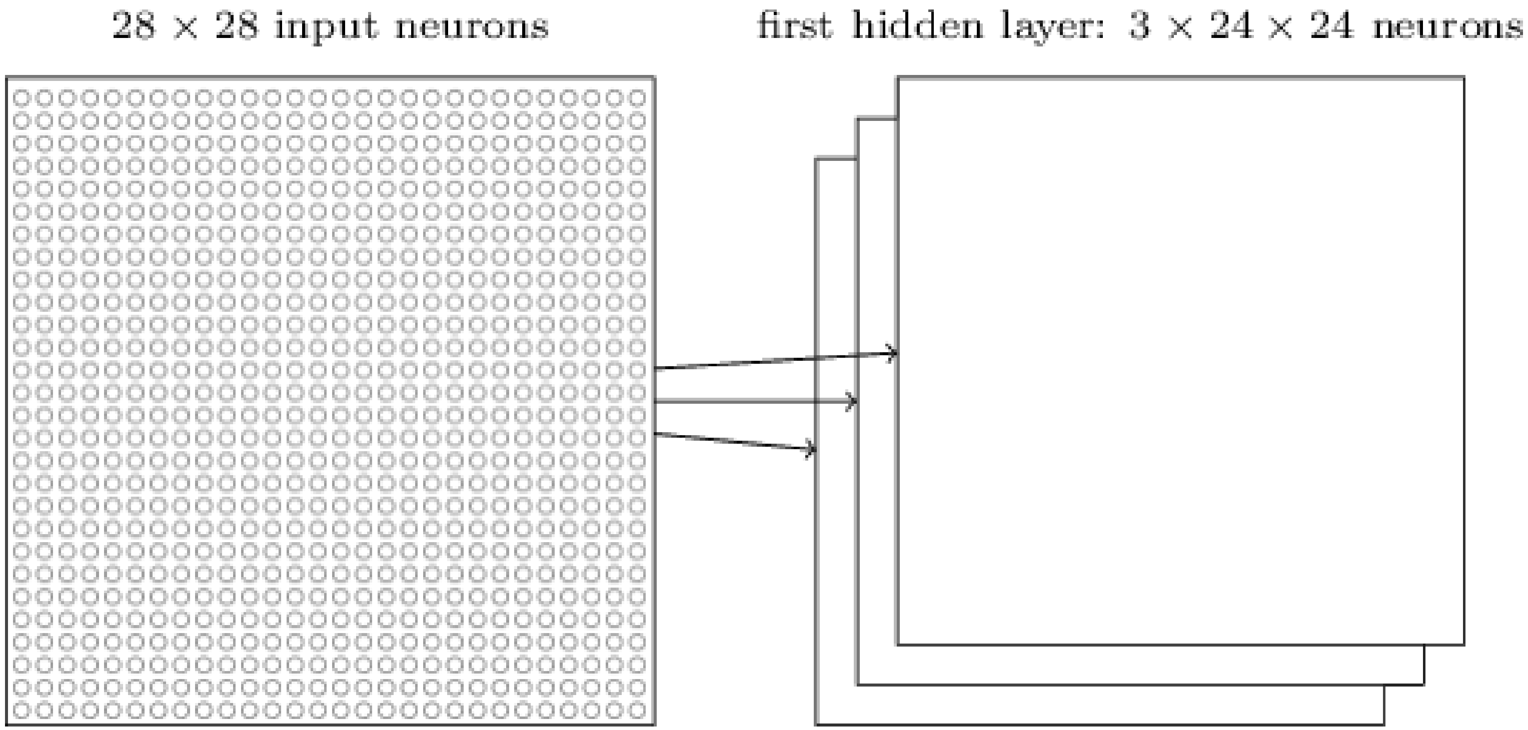


Convolutional Neural Network

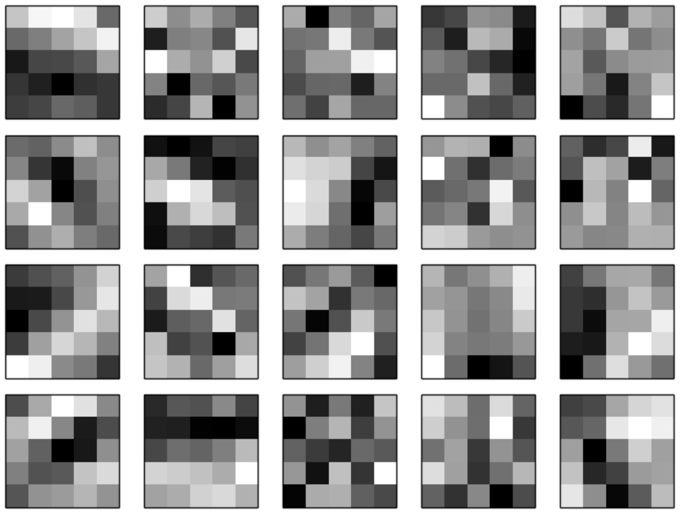


Applying 3 filters

Convolutional Neural Network

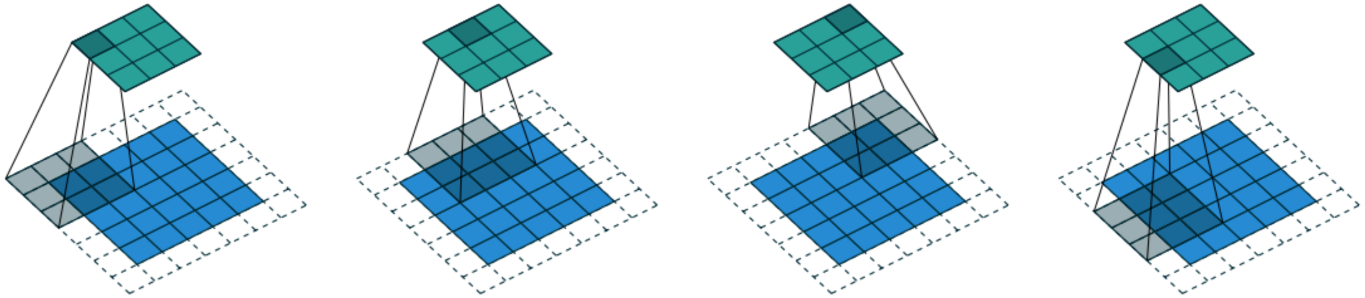


Applying 3 filters



Some Learned Filters

Convolutional Neural Network



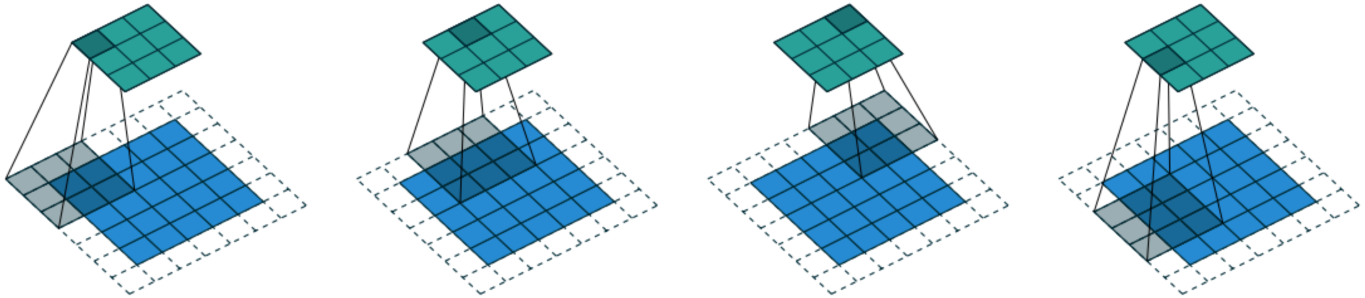
Stride: 2, Padding: 1, no Dilation

Stride: Apply the kernel using a step size.
Padding: Surround the input with a pad value to prevent shrinking of the input.
Dilation: Dilate the kernel to artificially increase the receptive field.

https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network



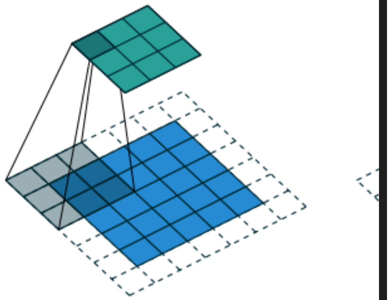
Stride: 2, Padding: 1, no Dilation

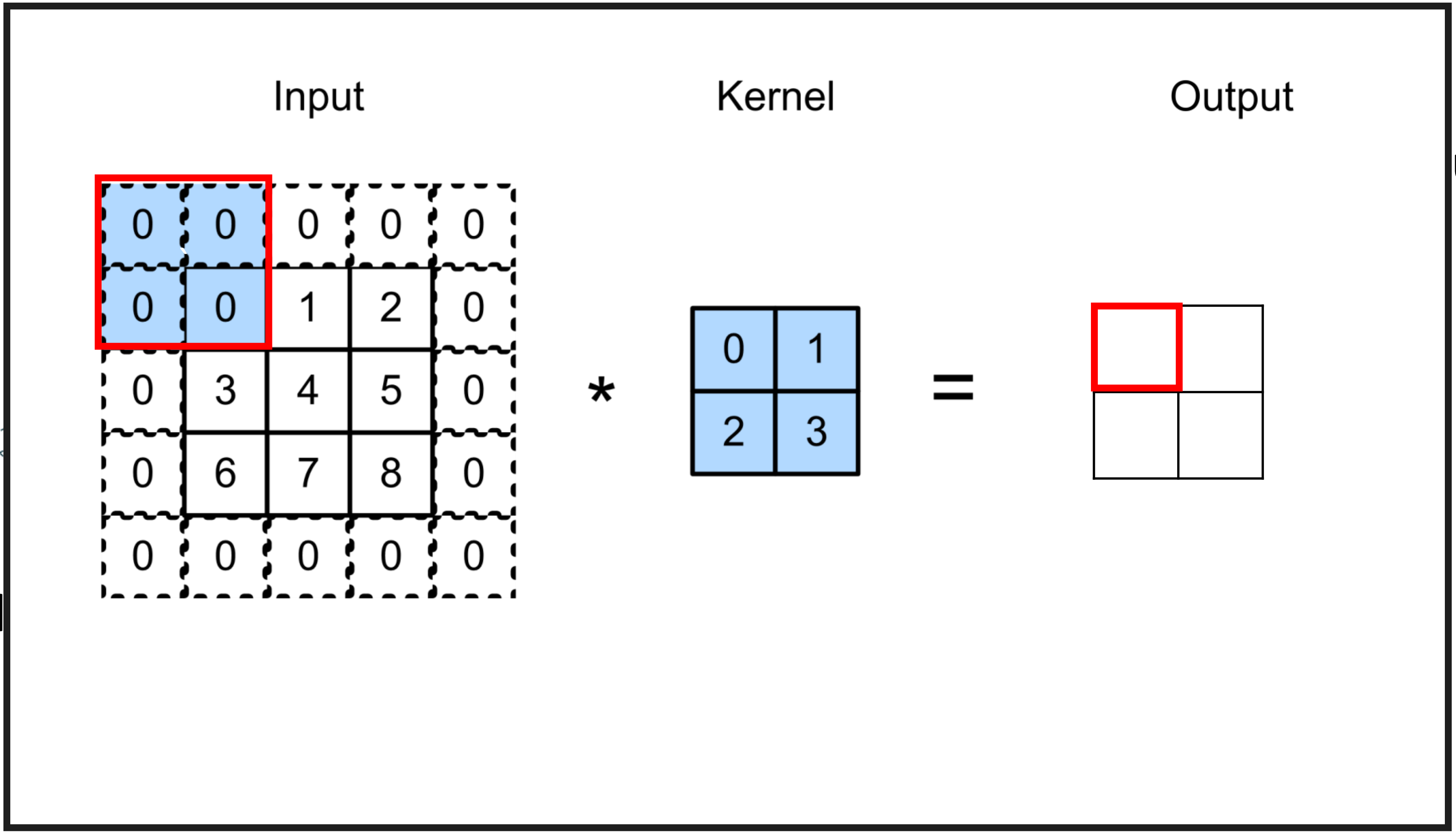
Stride: Apply the kernel using a step size.

https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network





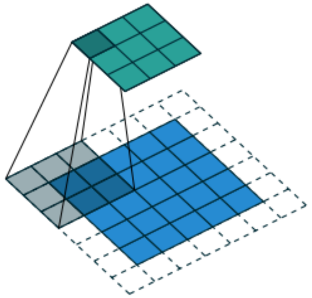
using a

Strid

https://arxiv.org/pdf/1603.07285.pdf

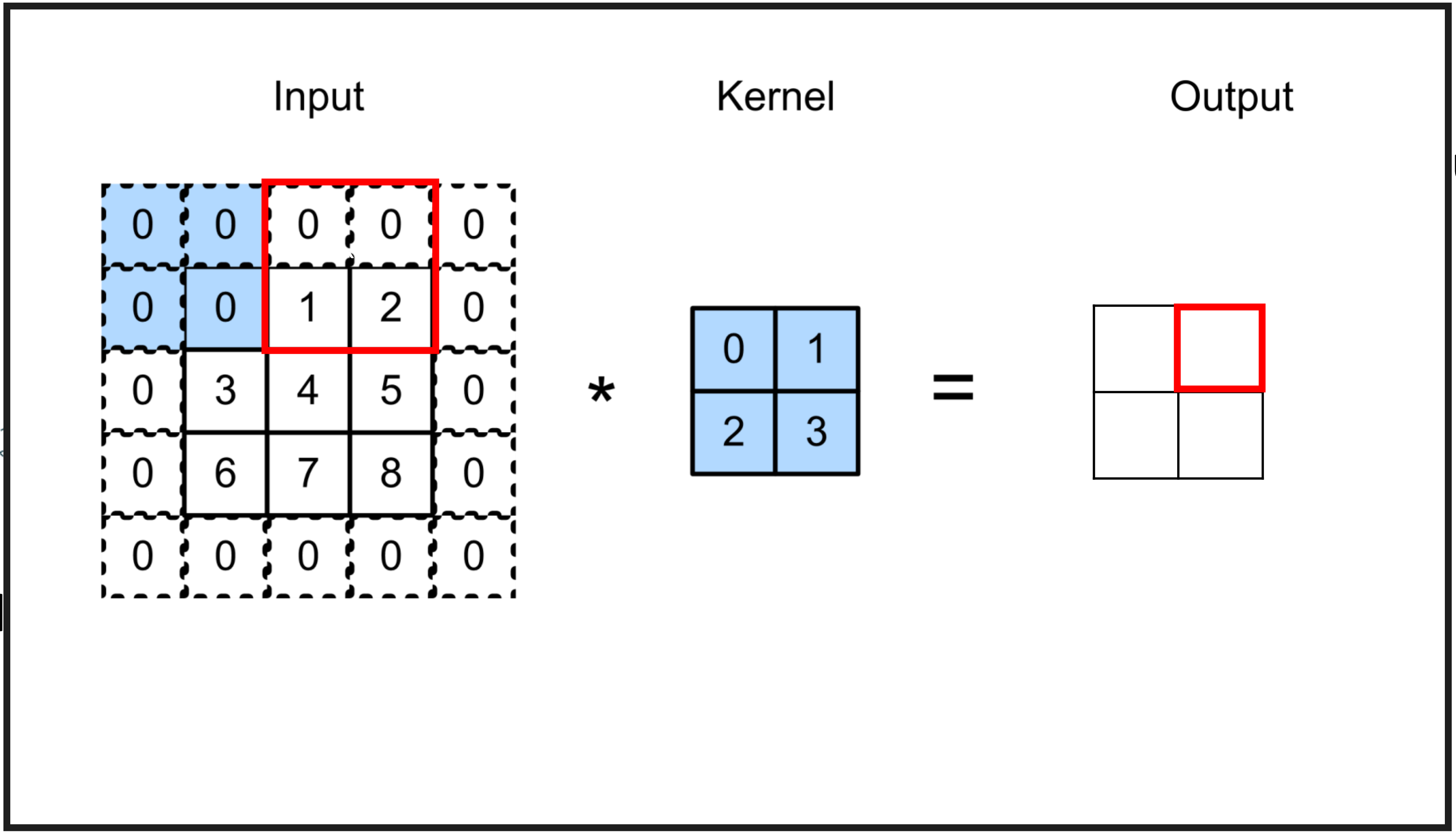
https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network



using a

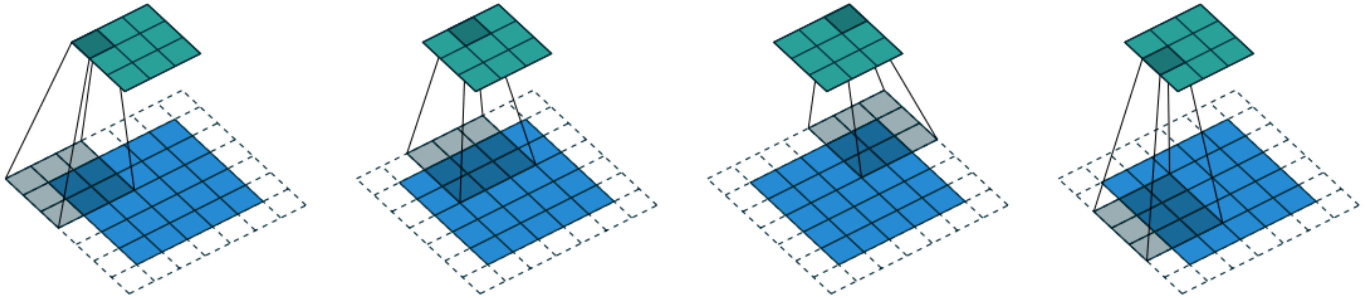
Strid



https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network



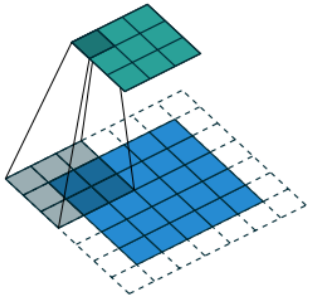
Stride: 2, Padding: 1, no Dilation

Stride: Apply the kernel using a step size.
Padding: Surround the input with a pad value to prevent shrinking of the input.

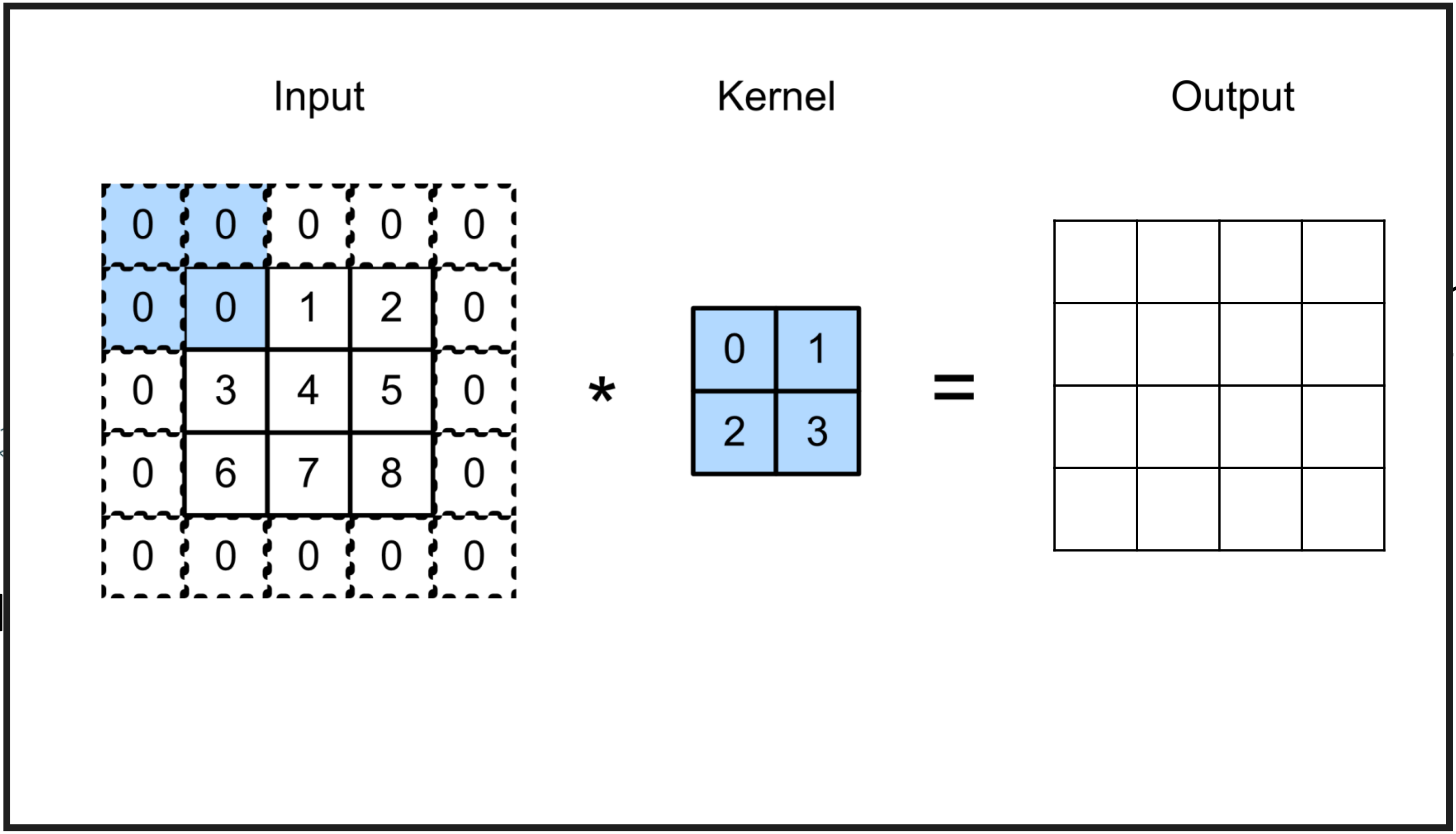
https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network



Strid



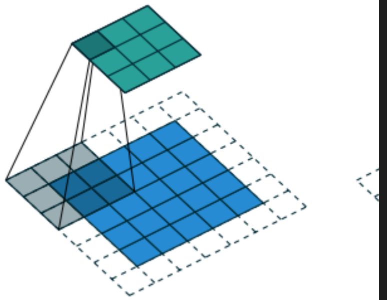
using a

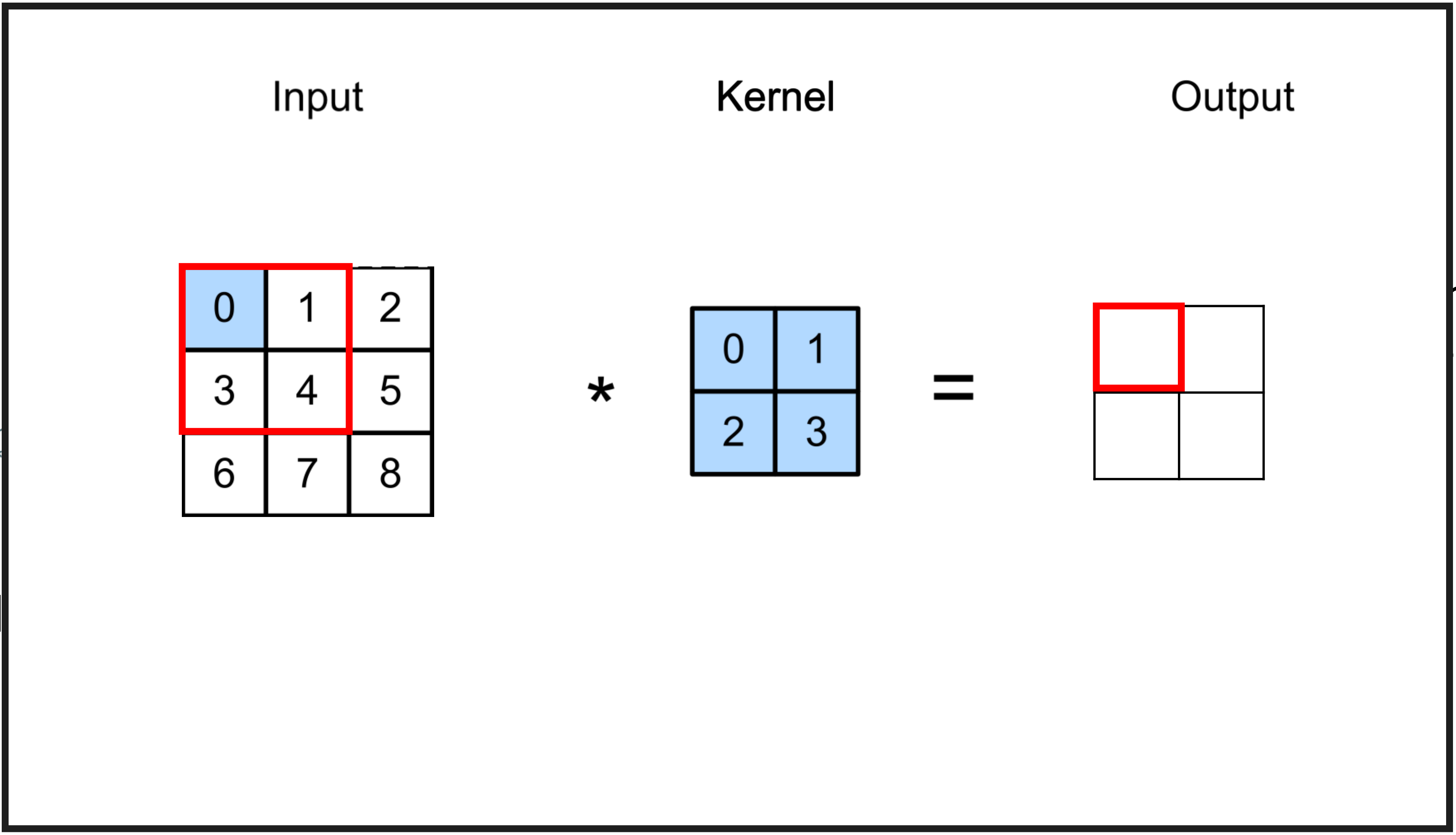
nput with a
inking of

https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network





using a

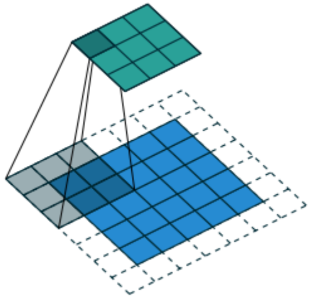
nput with a
inking of

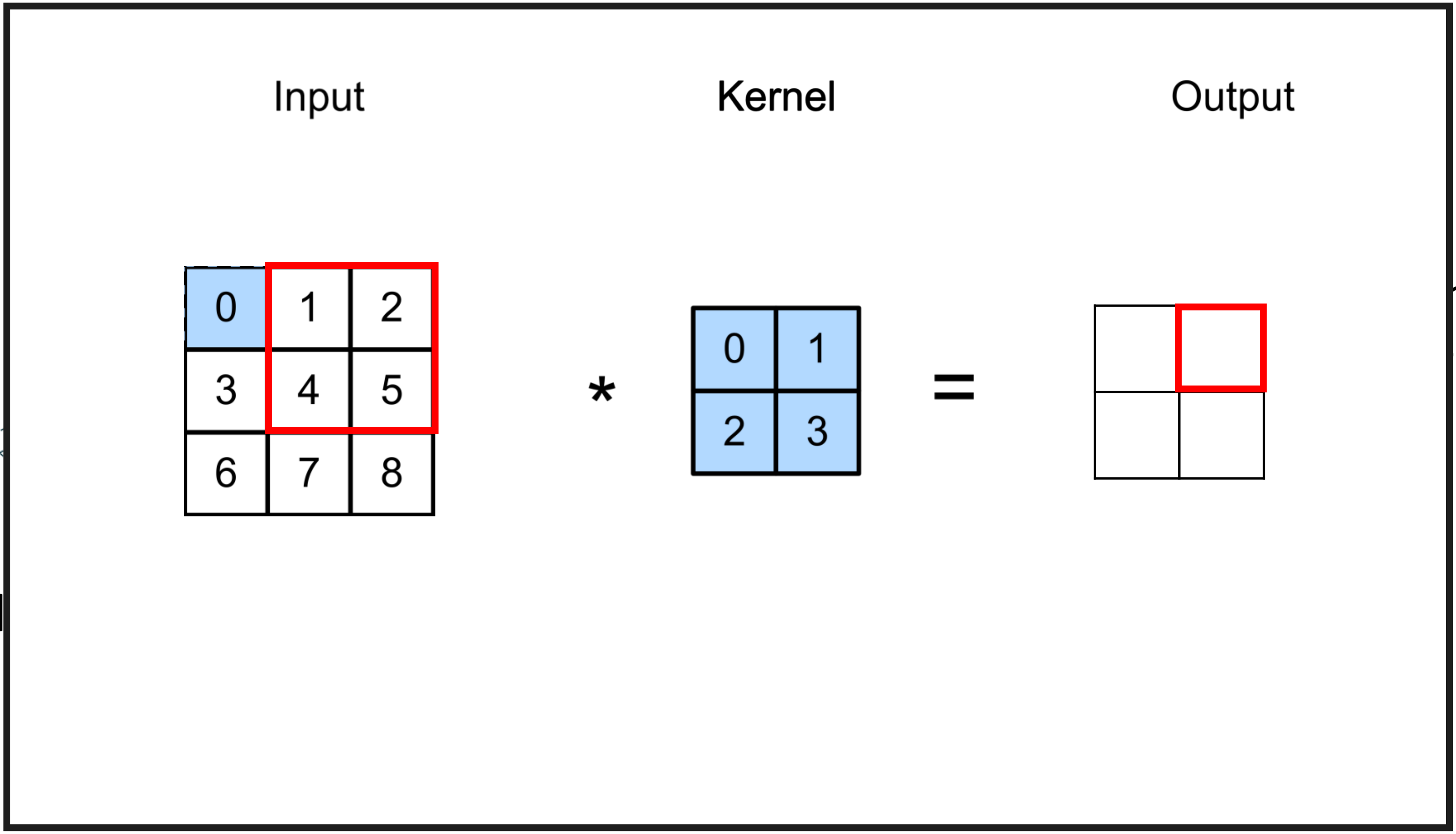
Strid

https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network





using a

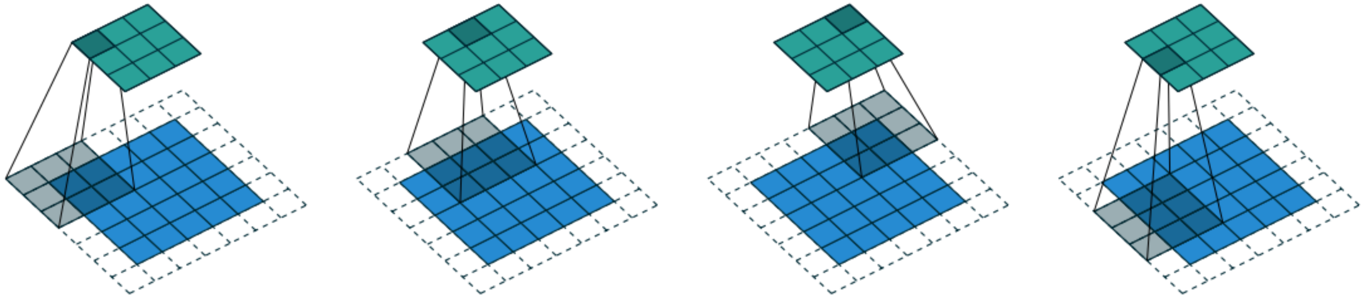
nput with a
inking of

Strid

https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network



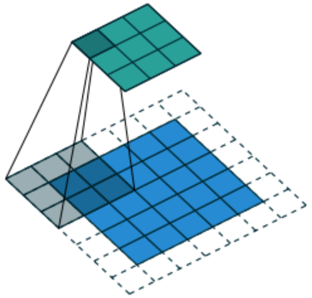
Stride: 2, Padding: 1, no Dilation

Stride: Apply the kernel using a step size.
Padding: Surround the input with a pad value to prevent shrinking of the input.
Dilation: Dilate the kernel to artificially increase the receptive field.

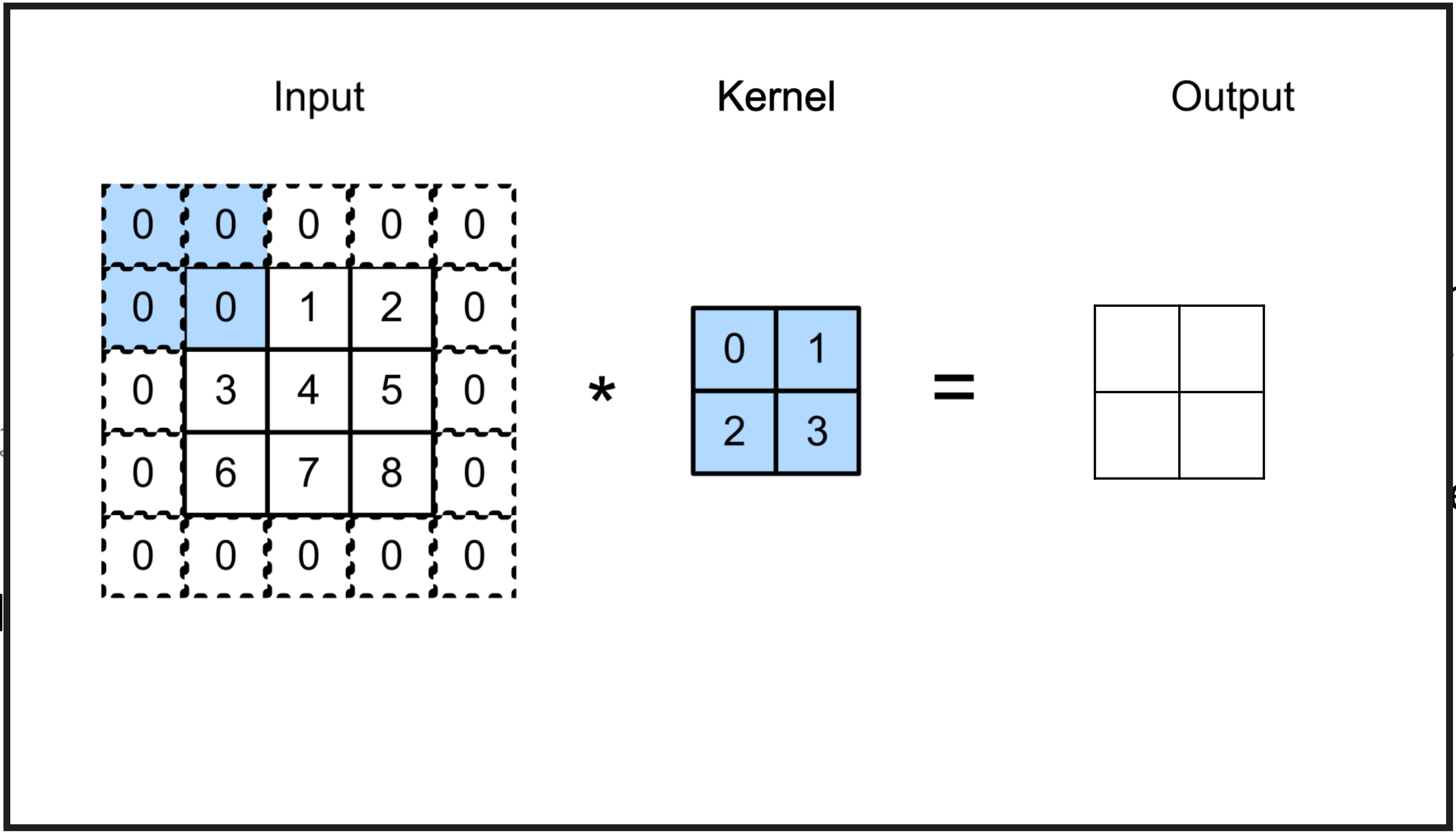
https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network



Strid



using a

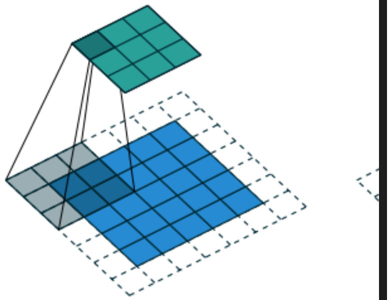
nput with a
inking of

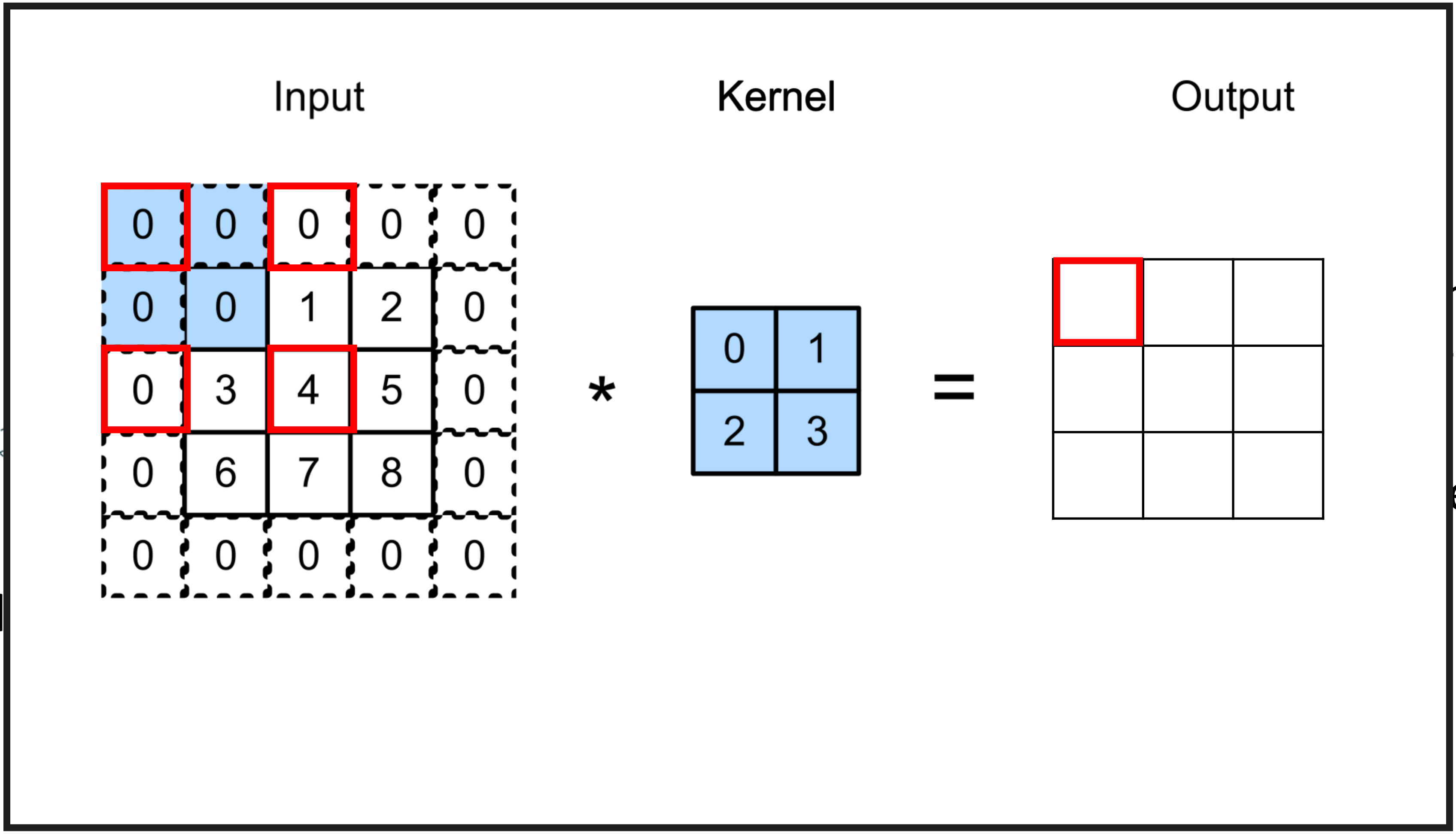
el to

https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network





using a

nput with a
inking of

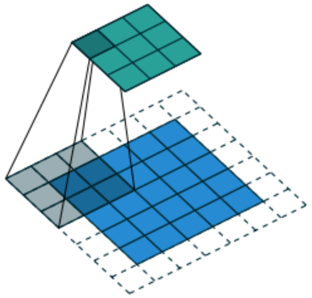
el to

Strid

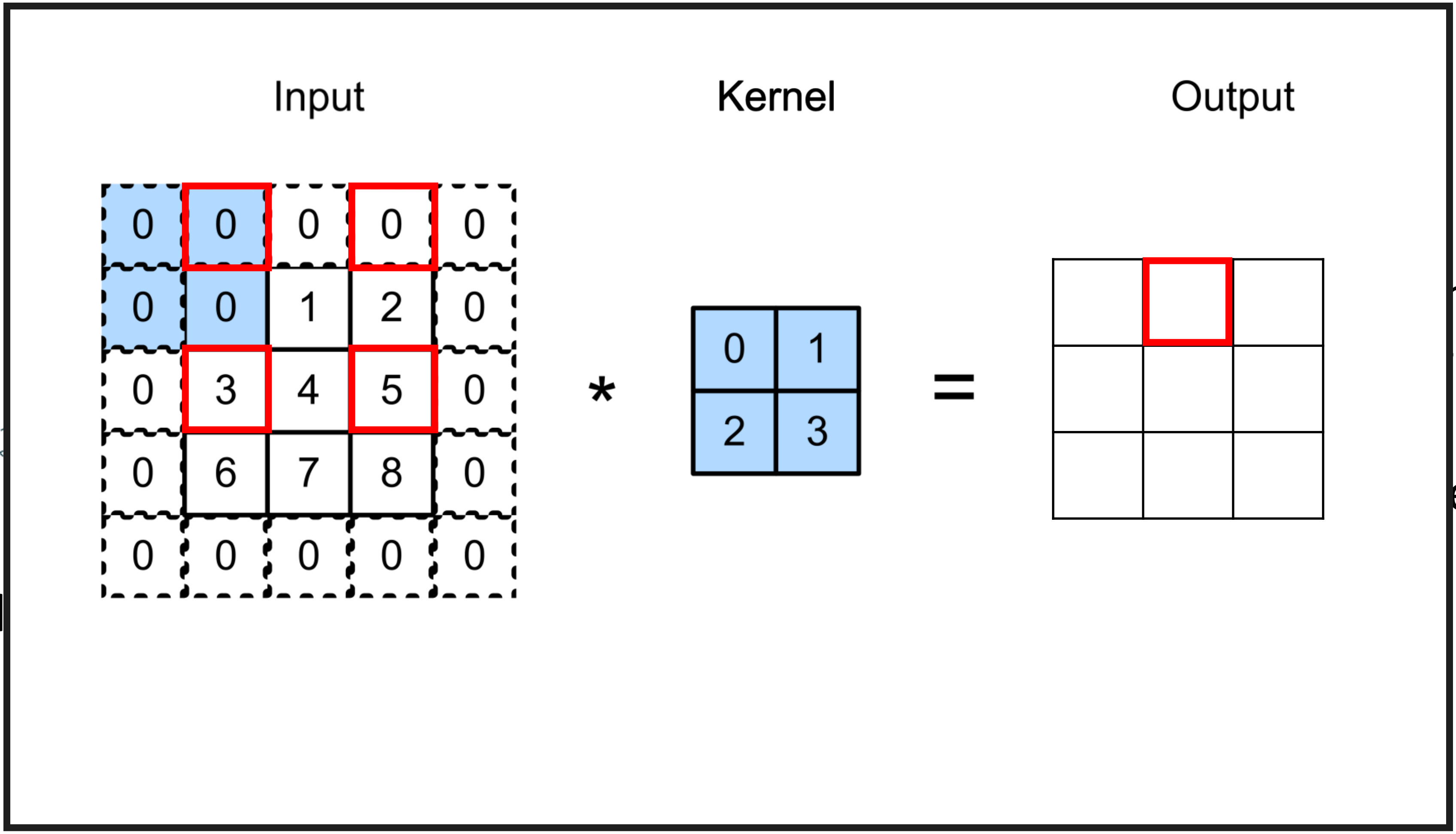
https://arxiv.org/pdf/1603.07285.pdf

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Convolutional Neural Network



Strid

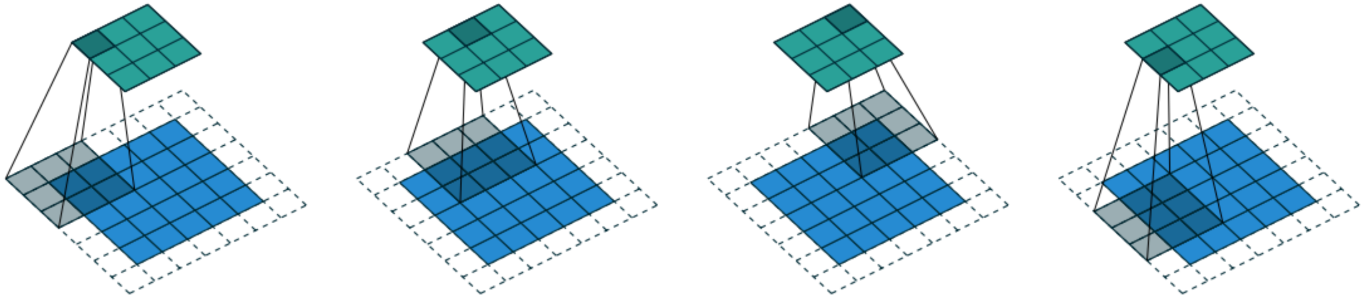


using a

nput with a
inking of

el to

Convolutional Neural Network



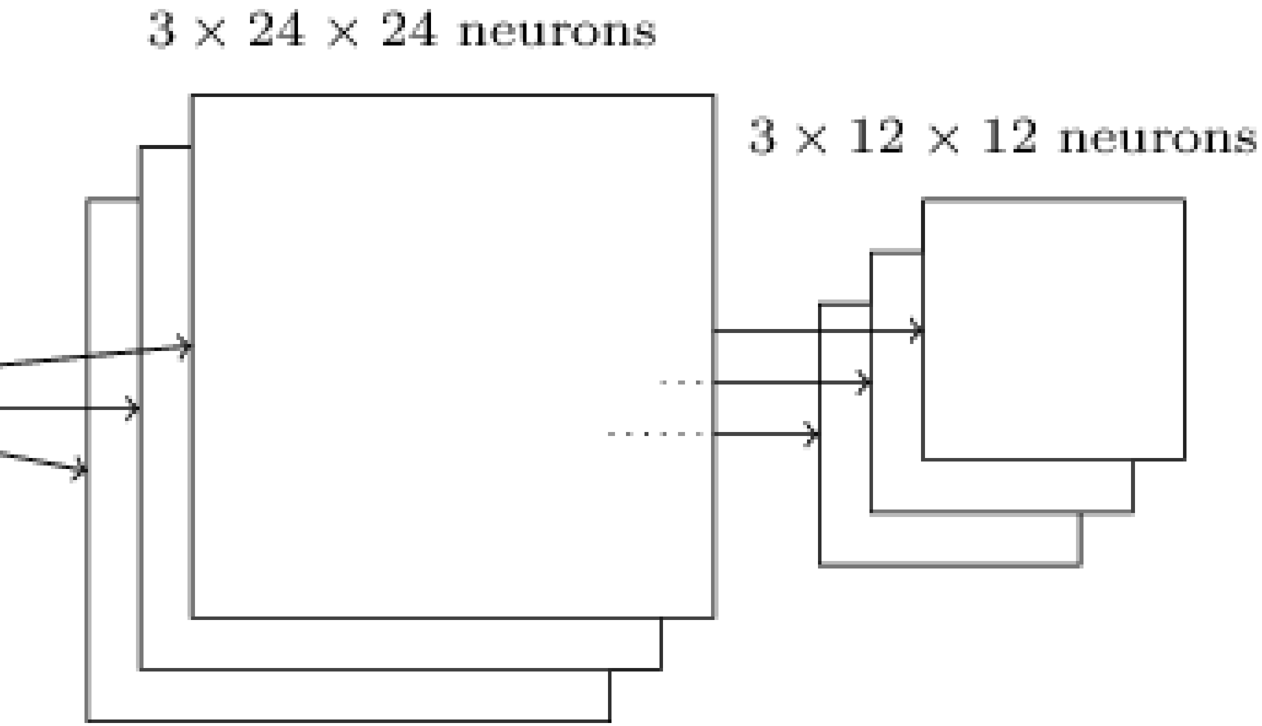
Stride: 2, Padding: 1, no Dilation

Stride: Apply the kernel using a step size.
Padding: Surround the input with a pad value to prevent shrinking of the input.
Dilation: Dilate the kernel to artificially increase the receptive field.

[ConvNet Shape Calculator (madebyollin.github.io)](madebyollin.github.io)

Convolutional Neural Network

Pooling Operation

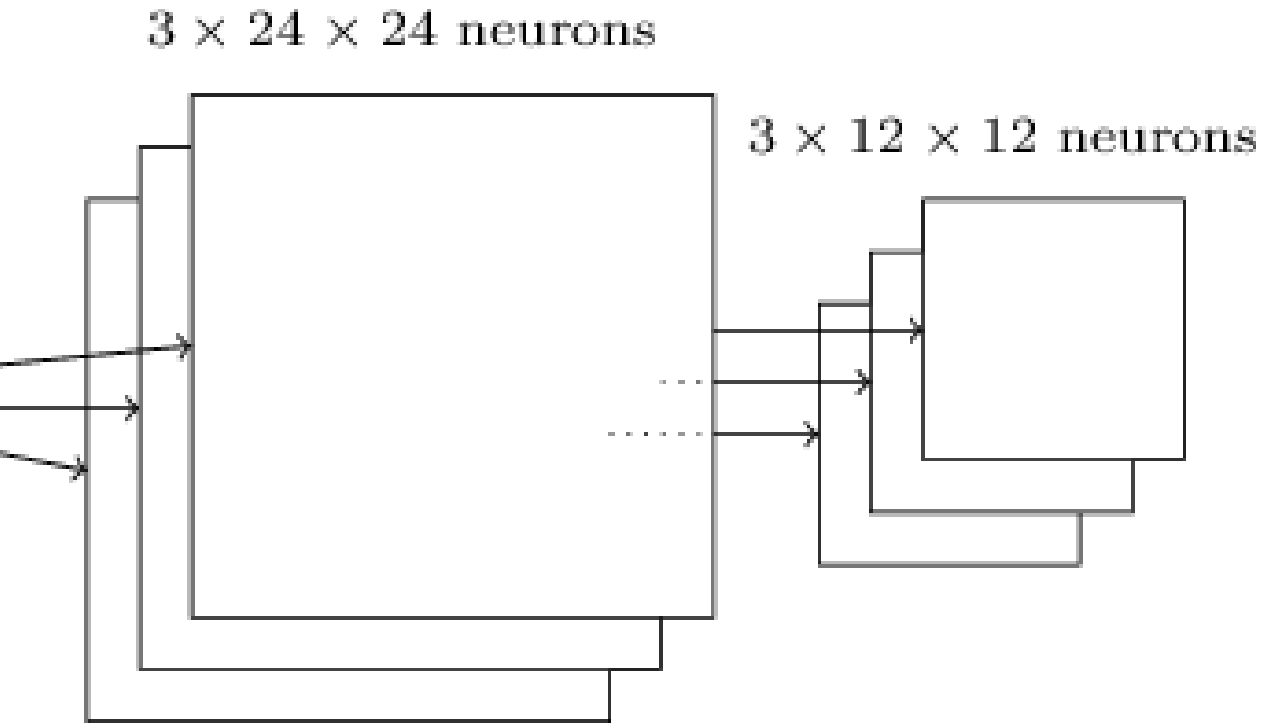


Applying 3 filters   Applying Max Pooling

Max Pooling: Select the most prominent feature in the receptive field.

Convolutional Neural Network

Pooling Operation



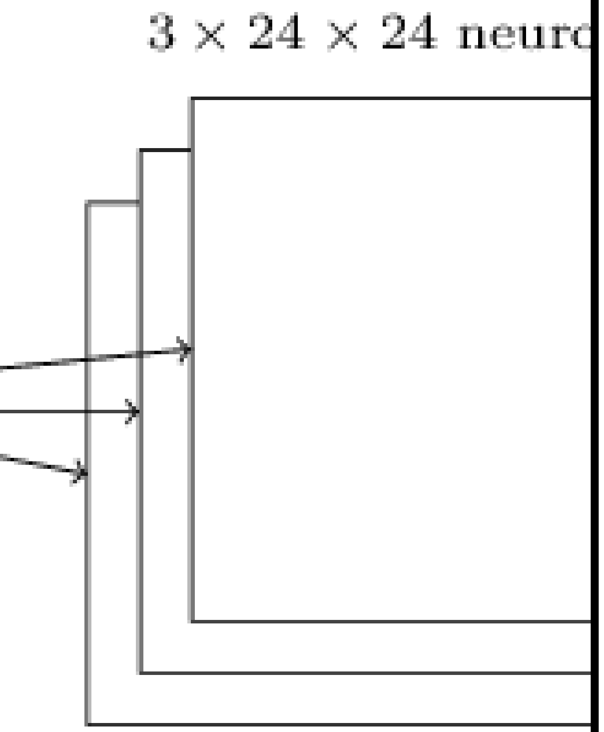
Applying 3 filters   Applying Max Pooling

Max Pooling: Select the most prominent feature in the receptive field.

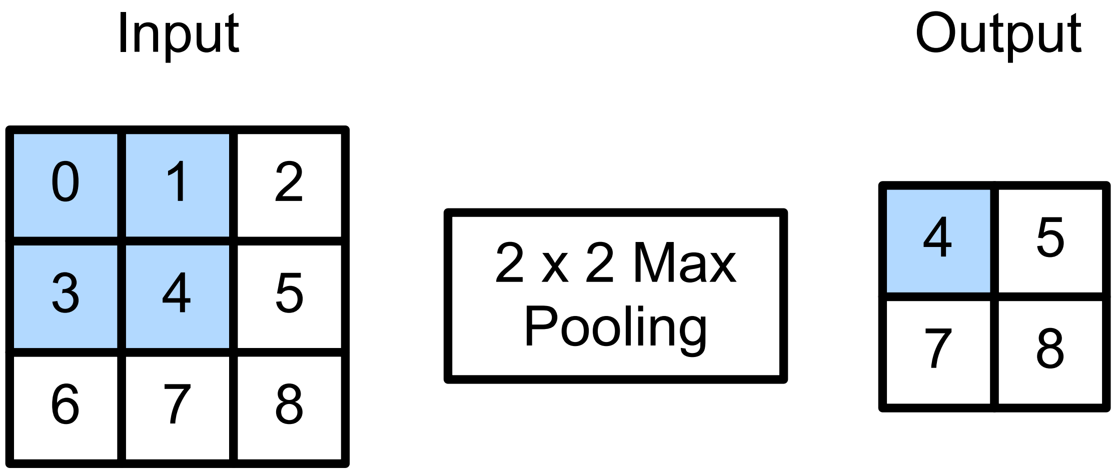
Intuition: Once a feature is found, the precise location isn't as important.

Helps with dimensionality reduction.

Convolutional Neural Network



Applying 3 filters



Max Pooling: Select the most prominent feature in the receptive field.

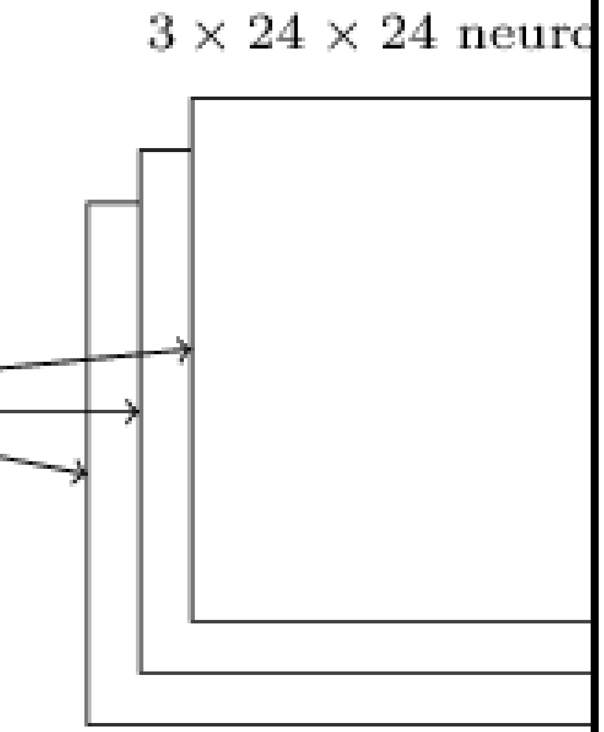
t the most
in the receptive

eature is found,
n isn't as

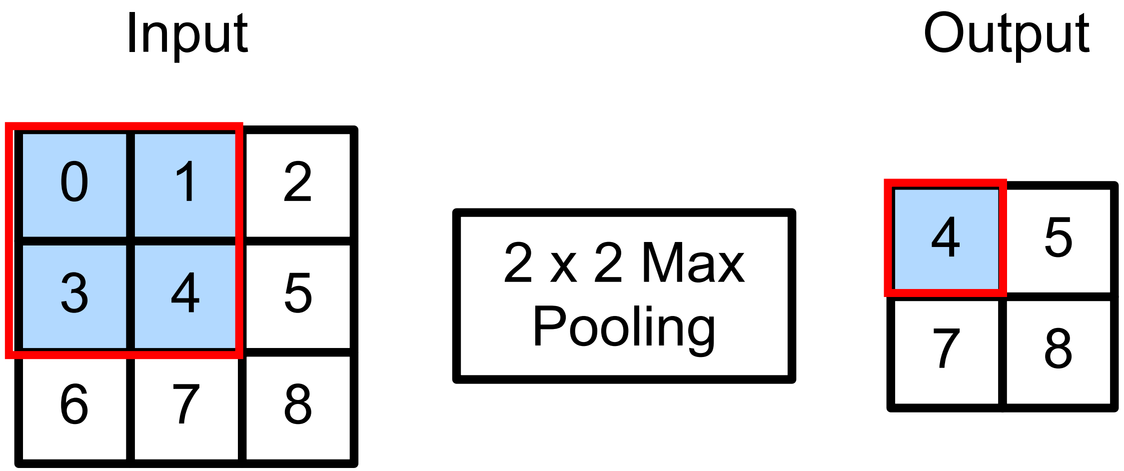
ionality reduction.

https://d2l.ai/chapter_convolutional-neural-networks/pooling.html

Convolutional Neural Network



Applying 3 filters



Max Pooling: Select the most prominent feature in the receptive field.

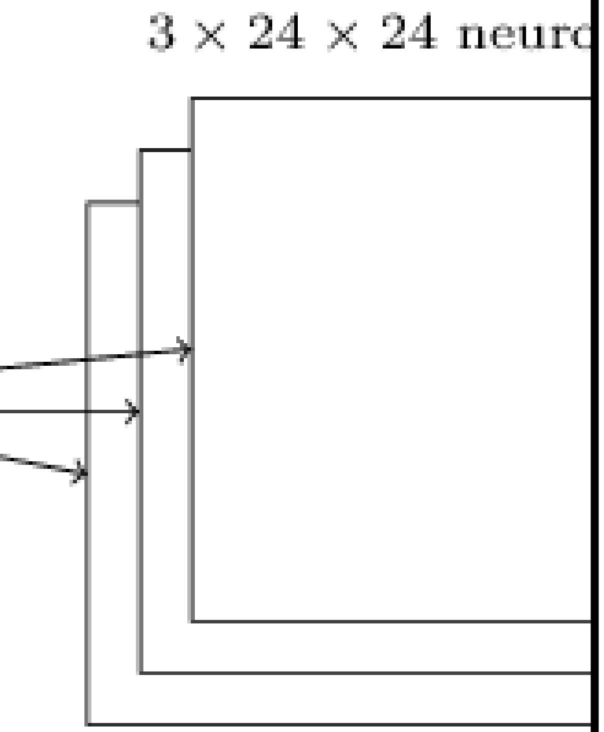
t the most
in the receptive

eature is found,
n isn't as

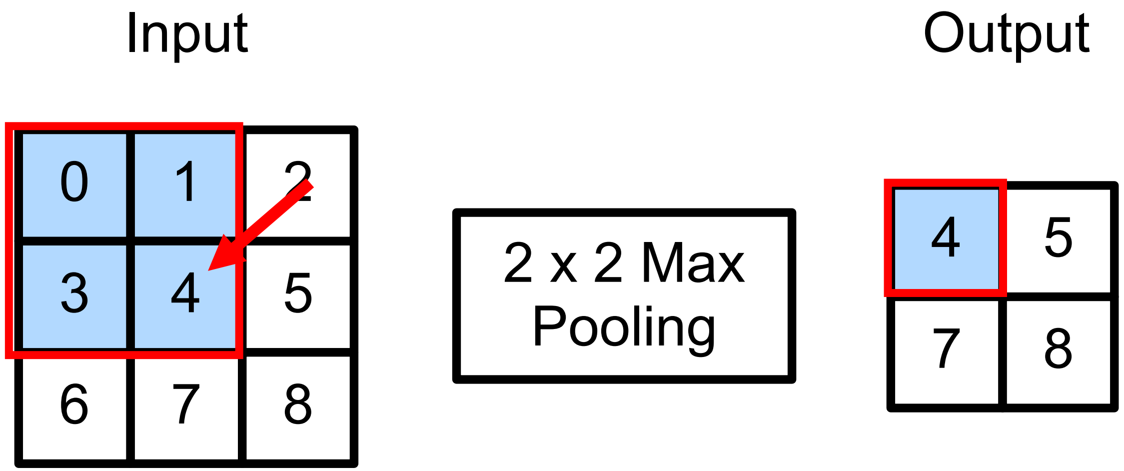
ionality reduction.

https://d2l.ai/chapter_convolutional-neural-networks/pooling.html

Convolutional Neural Network



Applying 3 filters



Max Pooling: Select the most prominent feature in the receptive field.

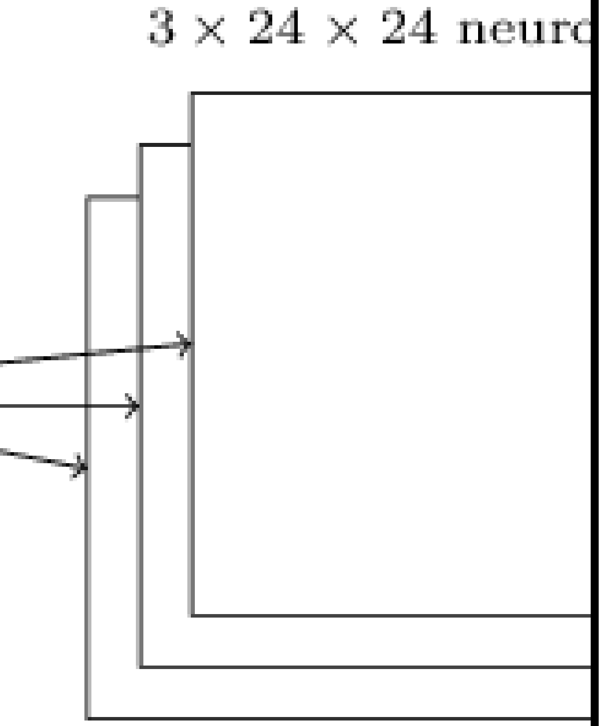
t the most
in the receptive

eature is found,
n isn't as

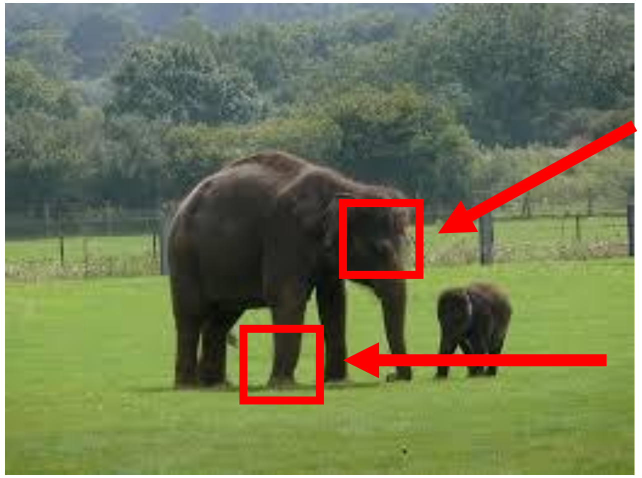
ionality reduction.

https://d2l.ai/chapter_convolutional-neural-networks/pooling.html

Convolutional Neural Network



Applying 3 filters



eye somewhere in here

leg somewhere in here

Max Pooling: Select the most prominent feature in the receptive field.

t the most
in the receptive

eature is found,
h isn't as

ionality reduction.

https://d2l.ai/chapter_convolutional-neural-networks/pooling.html

Convolutional Neural Network

Typically, we use a combination of layers for the task at hand.

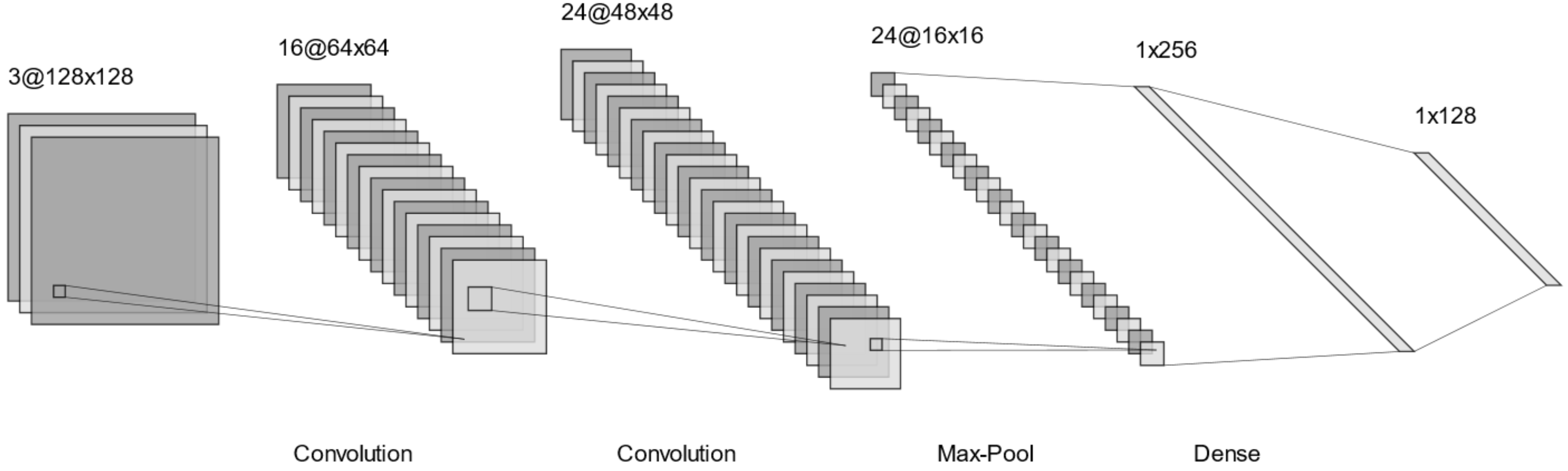


Image classification over 128 classes

Neural Networks

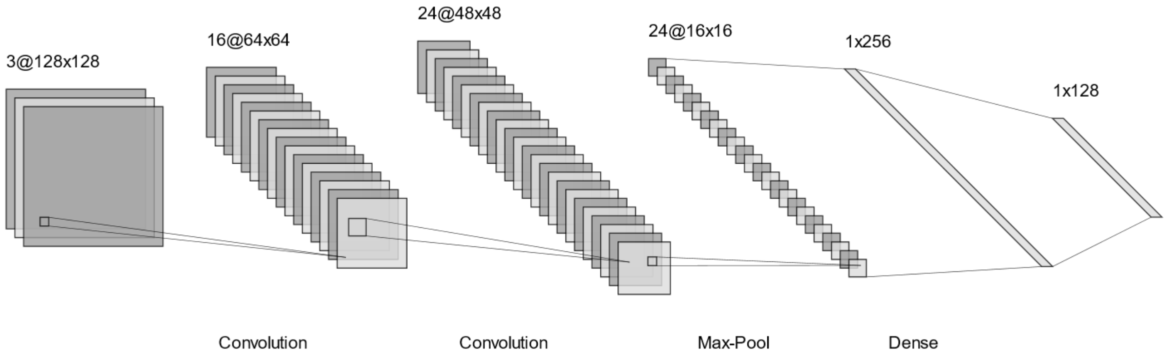
How do we get a network to learn?

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. Estimate the error.
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.



Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

network input	x
ground truth label	y
network	f

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

network input	x
ground truth label	y
network	f
network output	$f(x)$
network parameters (weights and biases)	θ

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

Loss function should be small when the predictions are accurate

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

Loss function should be small when the predictions are accurate

$$\sum_{i}^{n} ||f(x_i) - y_i||_2^2$$

L2 Loss Function

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

input label

$$\sum_{i}^{n} ||f(x_i) - y_i||_2^2$$

L2 Loss Function

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

$$(f(x)_i - y_i)^2$$

Squared L2 Norm

$$\sum_{i}^{n} \boxed{||f(x_i) - y_i||_2^2}$$

L2 Loss Function

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

Sum over training data

$$\sum_{i}^{n} ||f(x_i) - y_i||_2^2$$

L2 Loss Function

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

What are we optimizing wrt?

$$\sum_{i}^{n} ||f(x_i) - y_i||_2^2$$

L2 Loss Function

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

What are we optimizing wrt? θ

$$\sum_i^n ||f(x_i) - y_i||_2^2$$

L2 Loss Function

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

What are we optimizing wrt? θ

$$L(\theta) = \sum_{i}^{n} ||f_\theta(x_i) - y_i||_1$$

L2 Loss Function

https://stats.stackexchange.com/questions/387618/derivative-of-the-loss-function-w-r-t-to-x-for-the-backpropagation

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. **Estimate the error.**
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

What are we optimizing wrt? θ

$$L(\theta) = \sum_{i}^{n} ||f_\theta(x_i) - y_i||_1$$

L2 Loss Function

Obtain θ that minimizes L

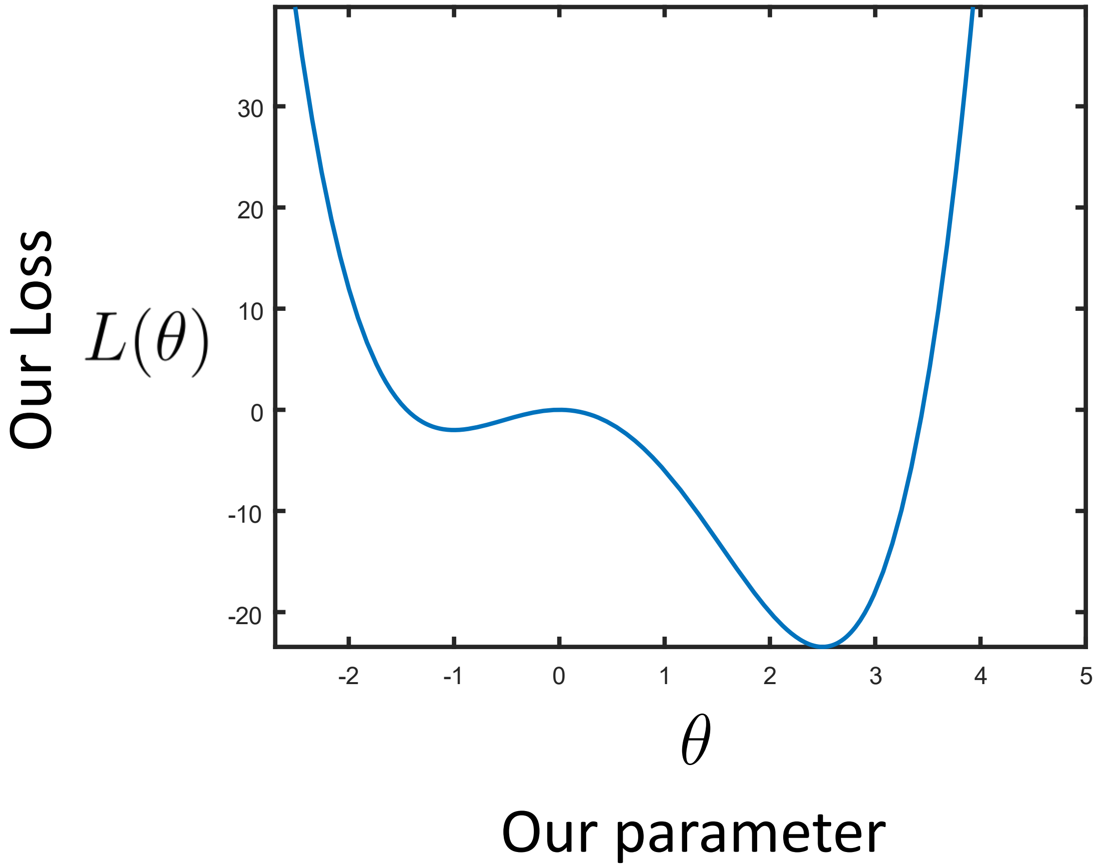
https://stats.stackexchange.com/questions/387618/derivative-of-the-loss-function-w-r-t-to-x-for-the-backpropagation

Neural Networks

How do we get a network to learn?

Gradient Descent.

1. Estimate the error.
2. **Compute the change in weights most likely to reduce the error (gradient).**
3. Update the weights according to the gradient.

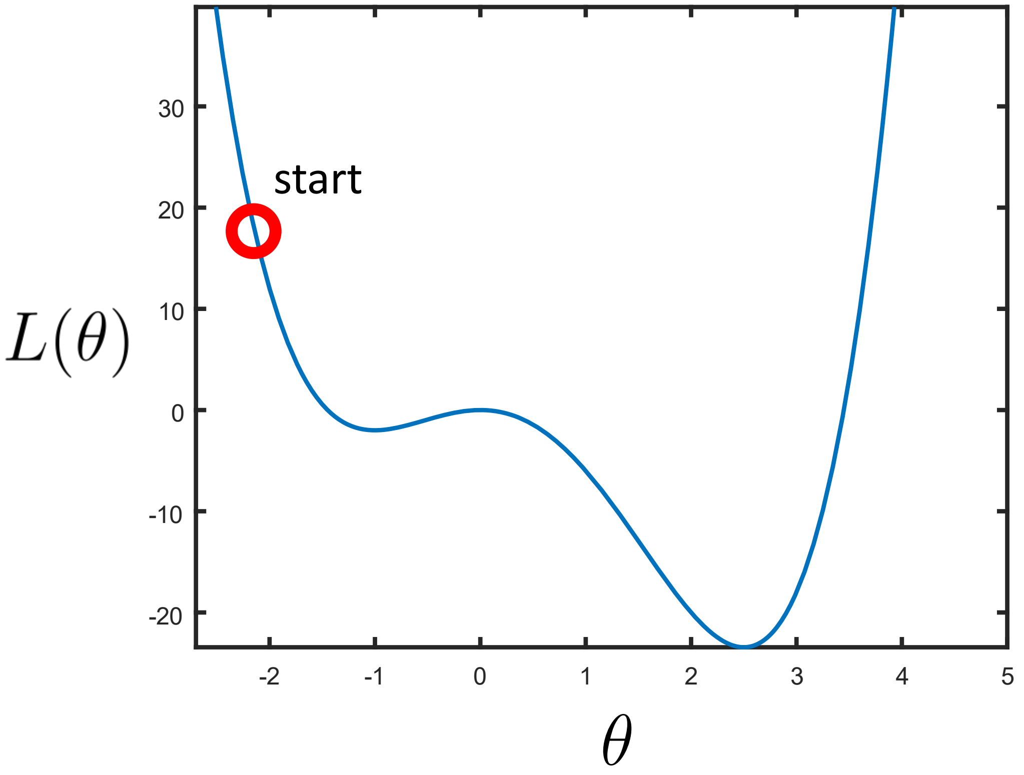


Neural Networks

How do we get a network to learn?

Gradient Descent.

1. Estimate the error.
2. **Compute the change in weights most likely to reduce the error (gradient).**
3. Update the weights according to the gradient.

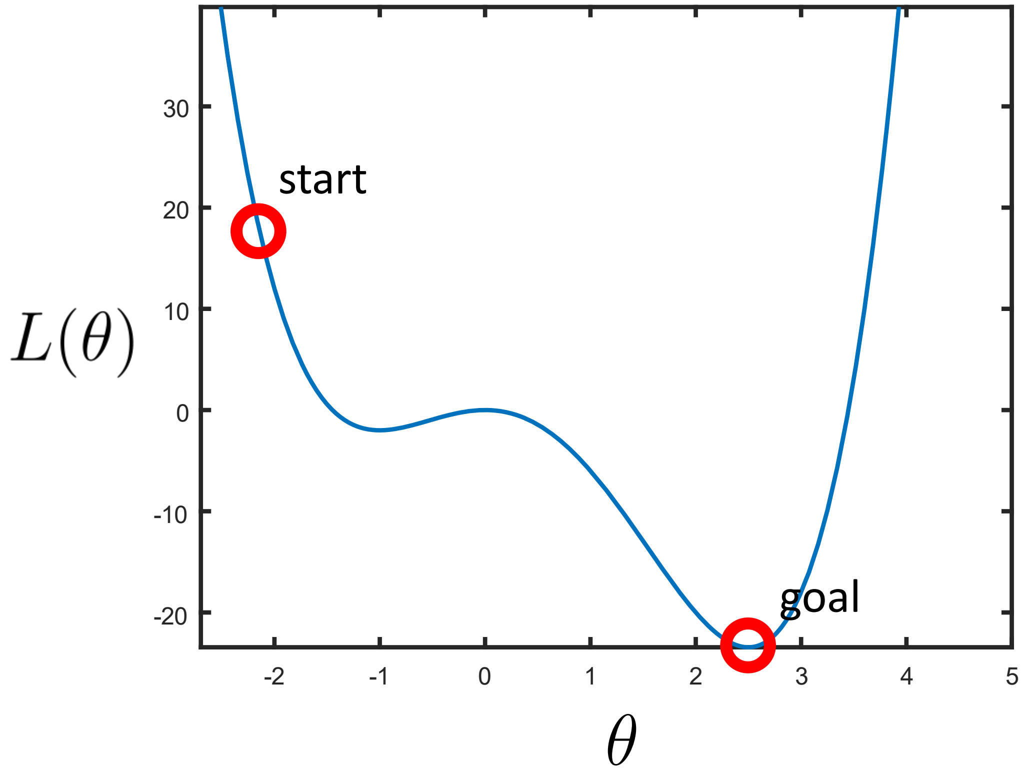


Neural Networks

How do we get a network to learn?

Gradient Descent.

1. Estimate the error.
2. **Compute the change in weights most likely to reduce the error (gradient).**
3. Update the weights according to the gradient.

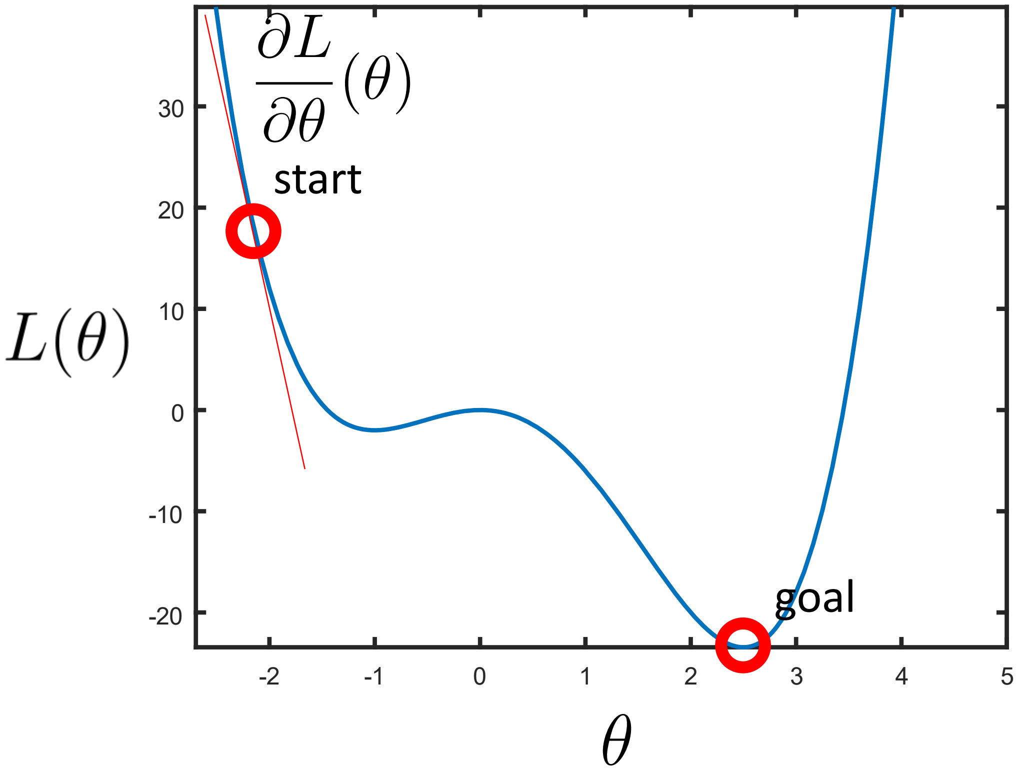


Neural Networks

How do we get a network to learn?

Gradient Descent.

1. Estimate the error.
2. **Compute the change in weights most likely to reduce the error (gradient).**
3. Update the weights according to the gradient.

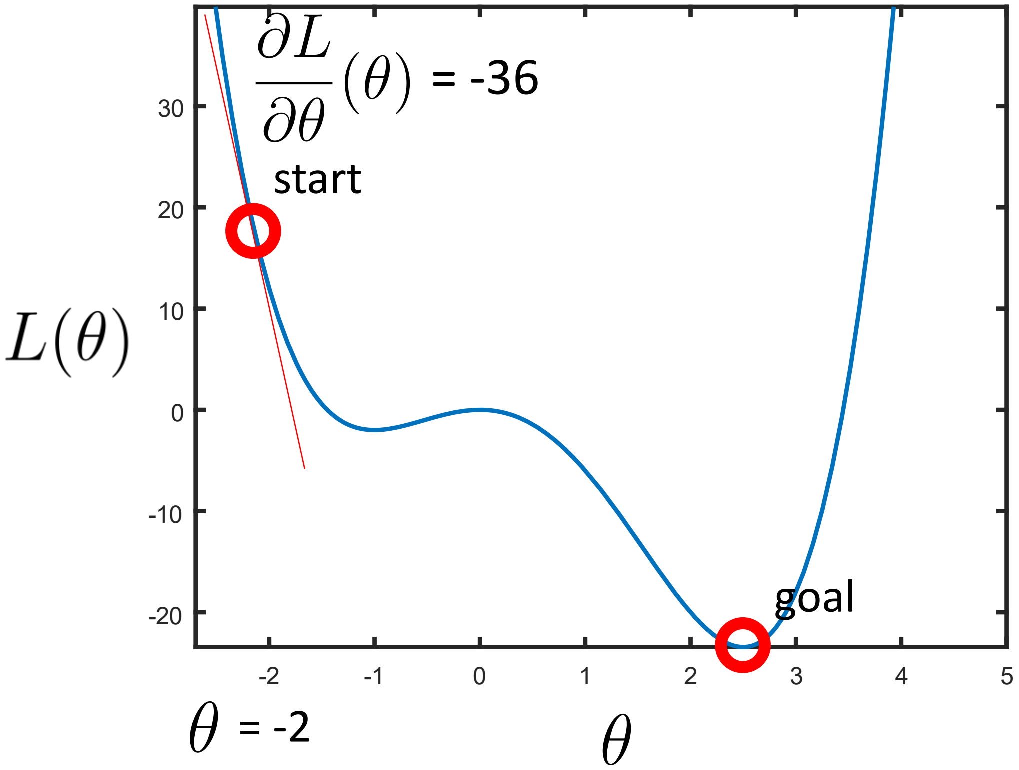


Neural Networks

How do we get a network to learn?

Gradient Descent.

1. Estimate the error.
2. **Compute the change in weights most likely to reduce the error (gradient).**
3. Update the weights according to the gradient.



Neural Networks

How do we get a network to learn?

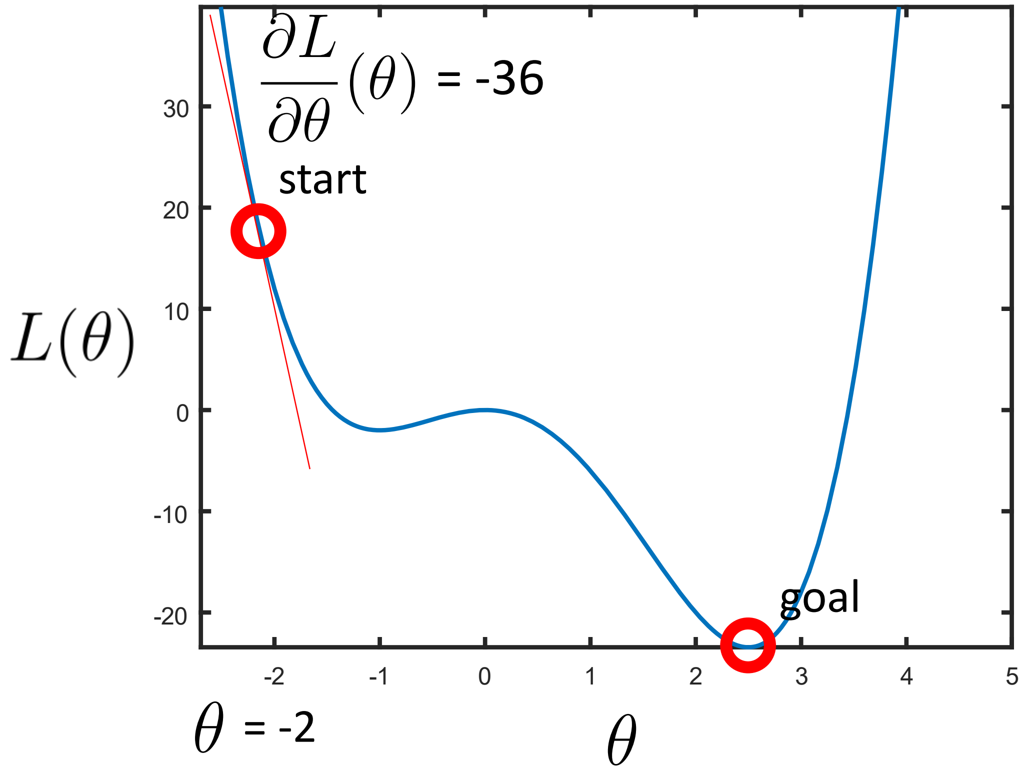
Gradient Descent.

1. Estimate the error.
2. Compute the change in weights most likely to reduce the error (gradient).
3. **Update the weights according to the gradient.**

Learning Rate

$$\theta_{t+1} = \theta - \alpha \frac{\partial L}{\partial \theta}(\theta)$$

Update Rule



Neural Networks

How do we get a network to learn?

Gradient Descent.

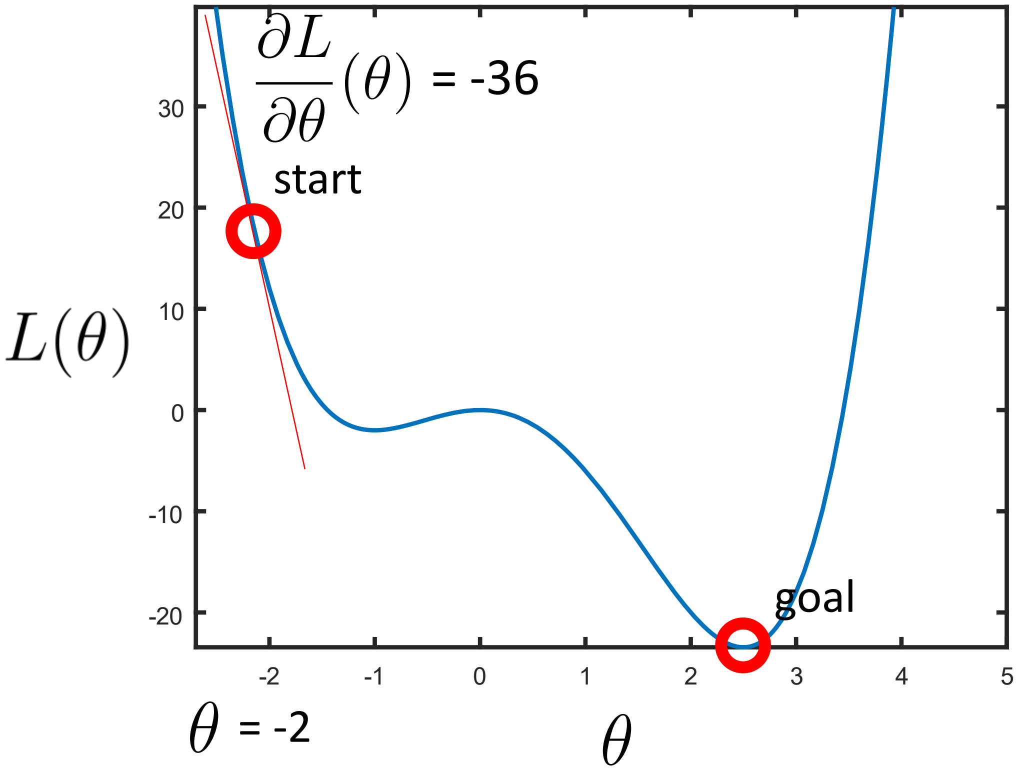
1. Estimate the error.
2. Compute the change in weights most likely to reduce the error (gradient).
3. **Update the weights according to the gradient.**

θ Learning Rate $\frac{\partial L}{\partial \theta}(\theta)$

$$\theta_{t+1} = -2 - (0.01 * -36)$$

$$\theta_{t+1} = -1.64$$

Update Rule



Neural Networks

How do we get a network to learn?

Gradient Descent.

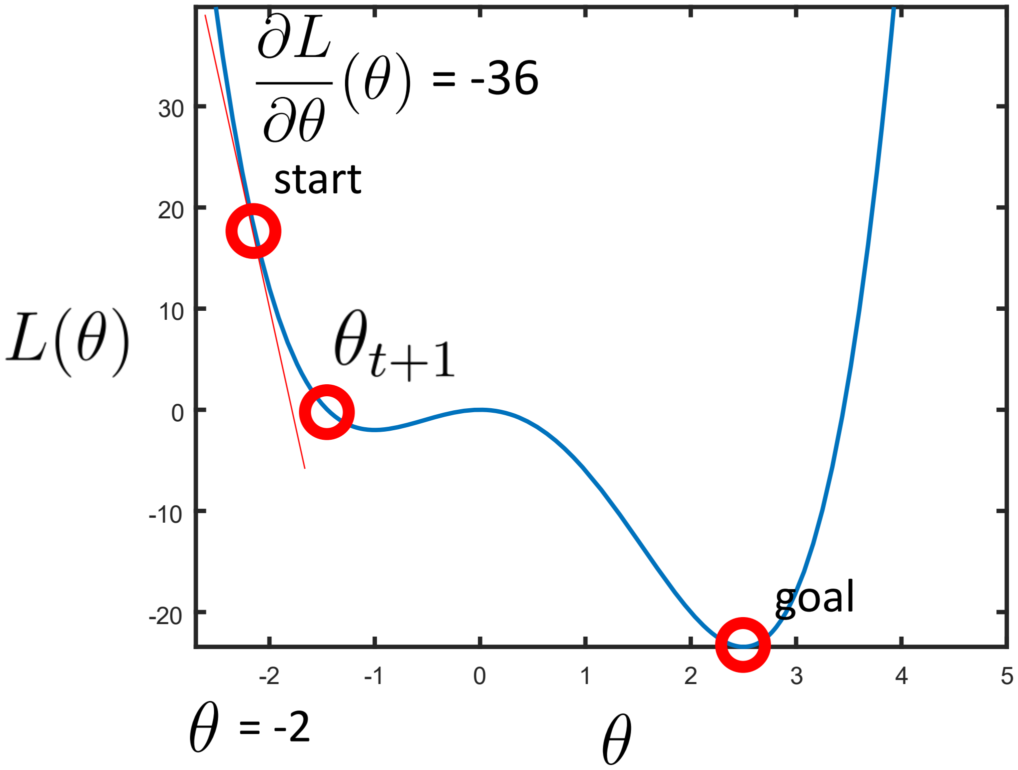
1. Estimate the error.
2. Compute the change in weights most likely to reduce the error (gradient).
3. **Update the weights according to the gradient.**

θ Learning Rate $\frac{\partial L}{\partial \theta}(\theta)$

$$\theta_{t+1} = -2 - (0.01 * -36)$$

$$\theta_{t+1} = -1.64$$

Update Rule

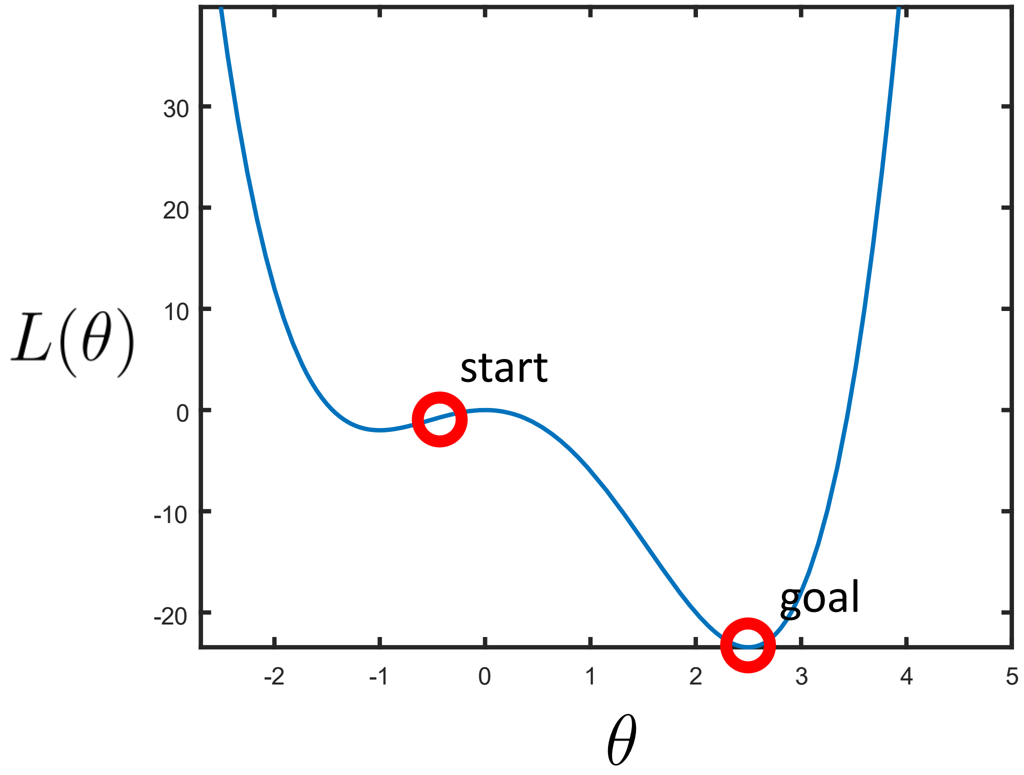


Neural Networks

How do we get a network to learn?

Gradient Descent.

1. Estimate the error.
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

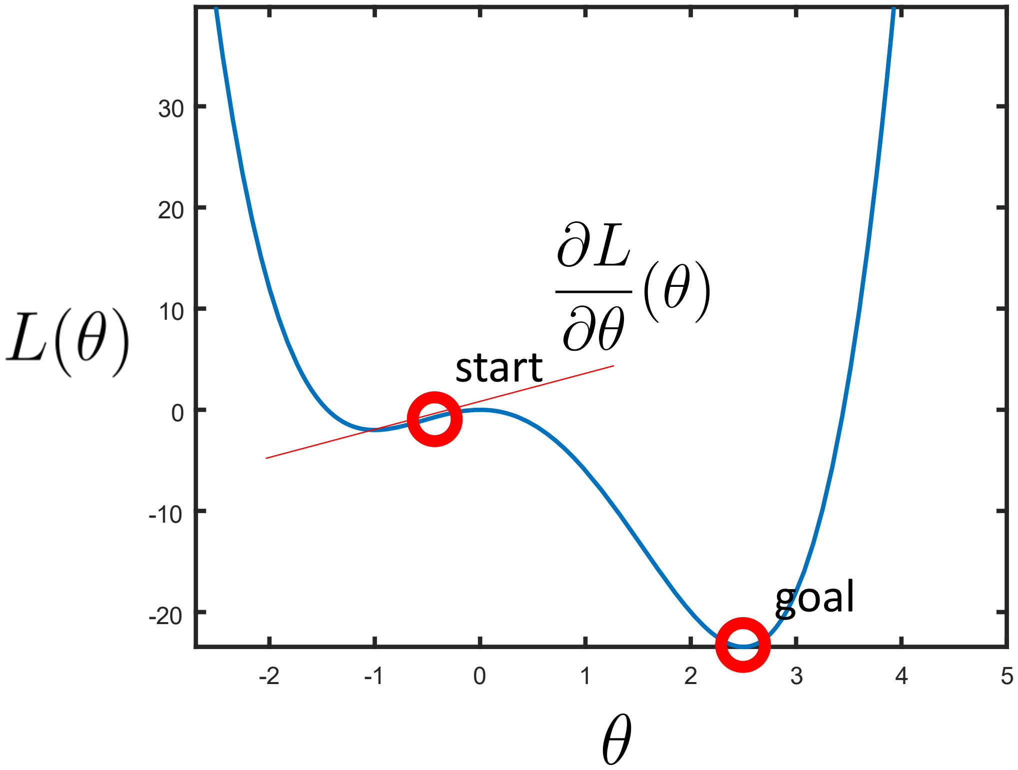


Neural Networks

How do we get a network to learn?

Gradient Descent.

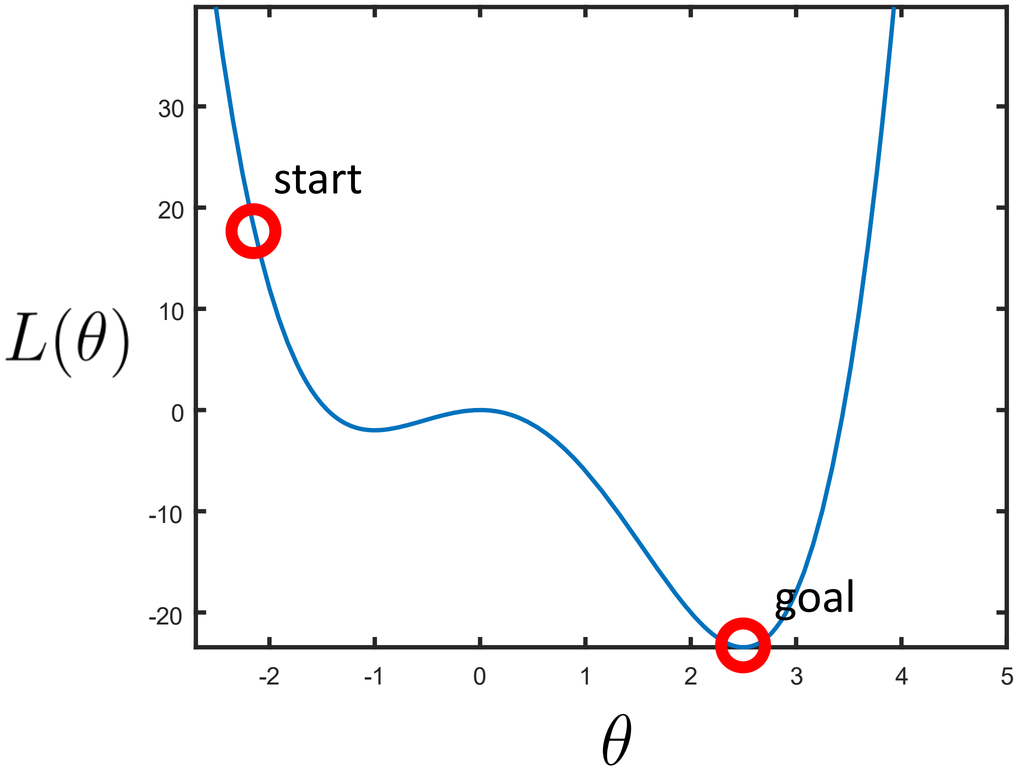
1. Estimate the error.
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.



Neural Networks

How do we get a network to learn?

How do we choose lr (alpha)?

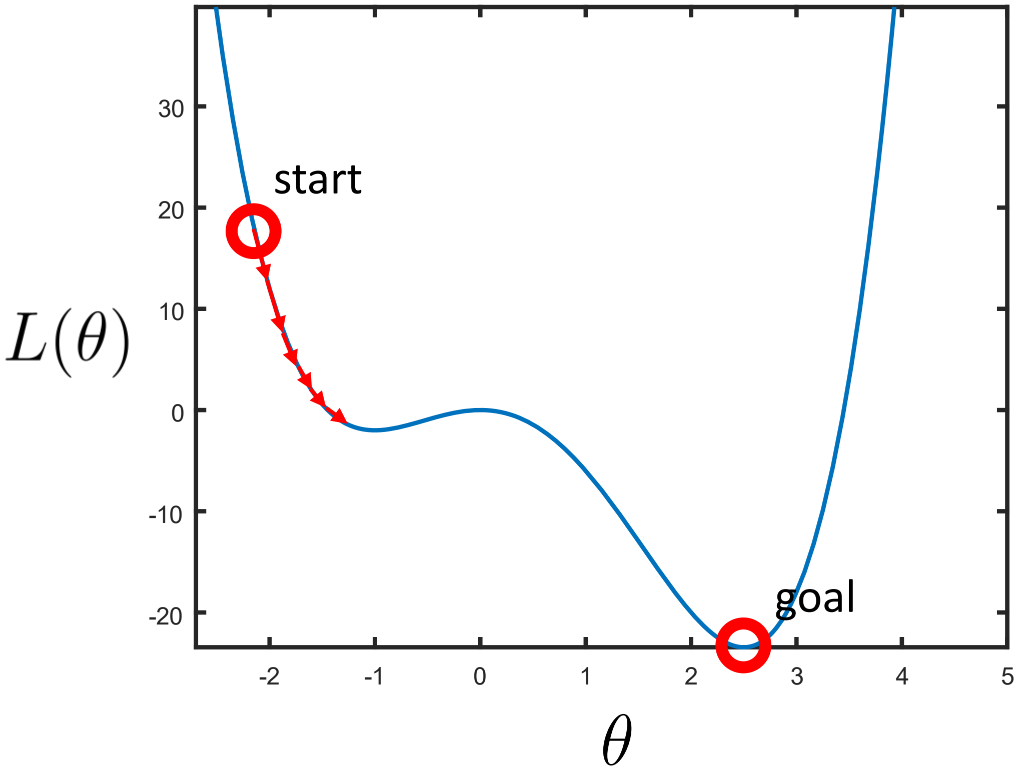


Neural Networks

Building Blocks: Optimizer

How do we choose lr (alpha)?

- Small lr will get caught in local minima.

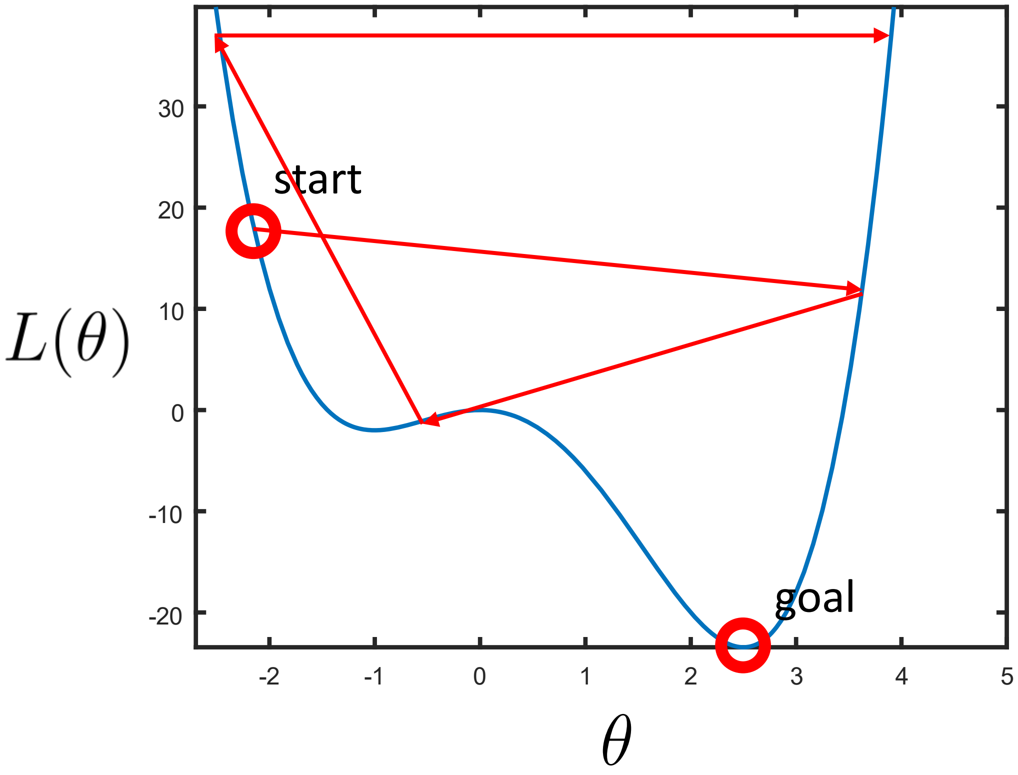


Neural Networks

Building Blocks: Optimizer

How do we choose lr (alpha)?

- Small lr will get caught in local minima.
- Large lr will never converge.



Neural Networks

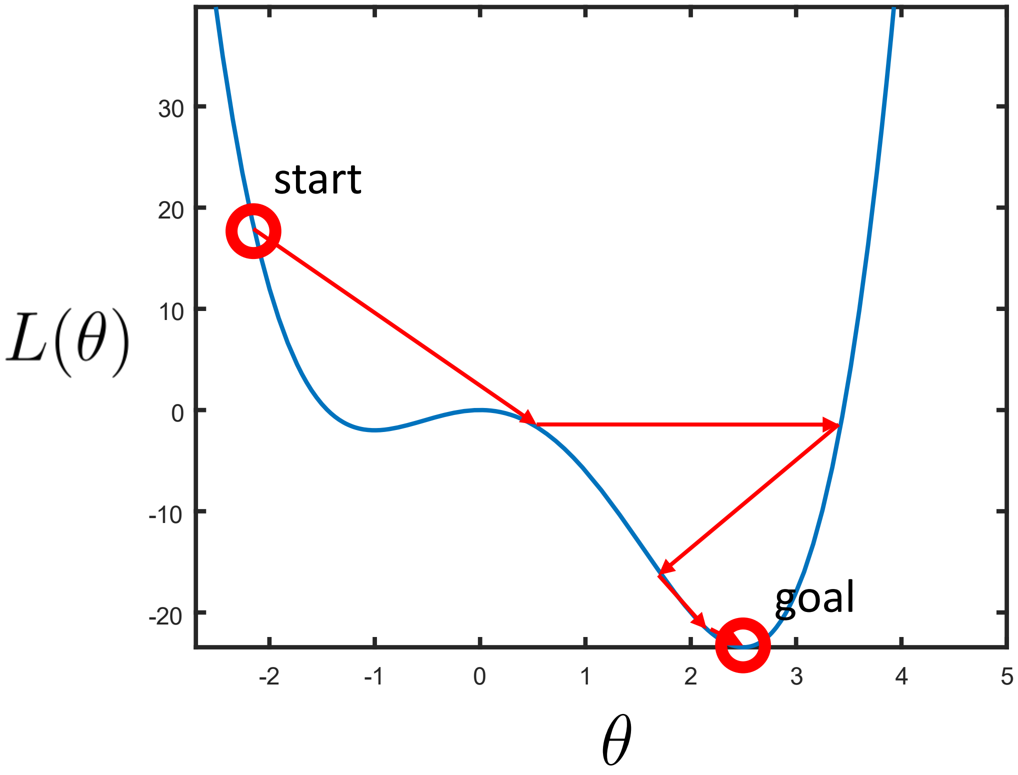
Building Blocks: Optimizer

How do we choose lr (alpha)?

- Small lr will get caught in local minima.
- Large lr will never converge.

Adam Optimizer

- Separate lr for each parameter.
- Update lr based on momentum.
- Decay the lr over time.



Neural Networks

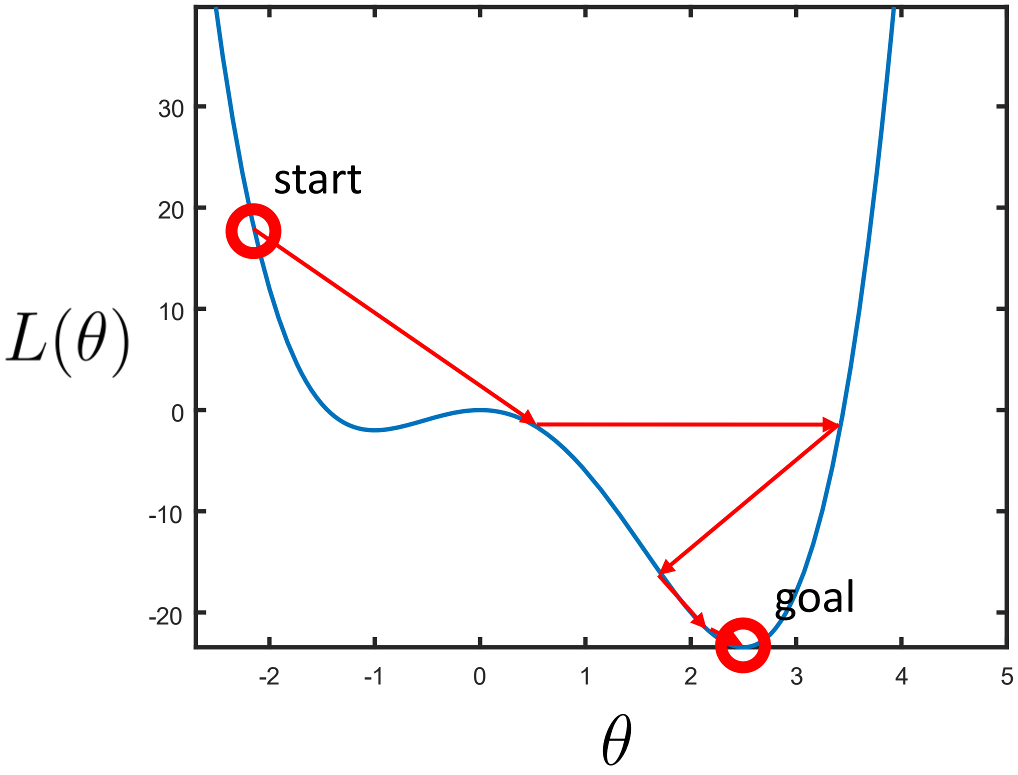
Building Blocks: Optimizer

How do we choose lr (alpha)?

- Small lr will get caught in local minima.
- Large lr will never converge.

Adam Optimizer

- Separate lr for each parameter.
- Update lr based on momentum.
- Decay the lr over time.
- Sensitive to lr initialization.
- Can be used with an overall decay as an upper bound.



Neural Networks

Building Blocks: batch size

Gradient Descent.

1. Estimate the error.
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.

How many samples should we see before we update the weights?

Neural Networks

Building Blocks: batch size

Gradient Descent:

- Use all samples to compute the gradient and then update parameters.

Stochastic Gradient Descent:

- Use one sample to compute the gradient and then update parameters.

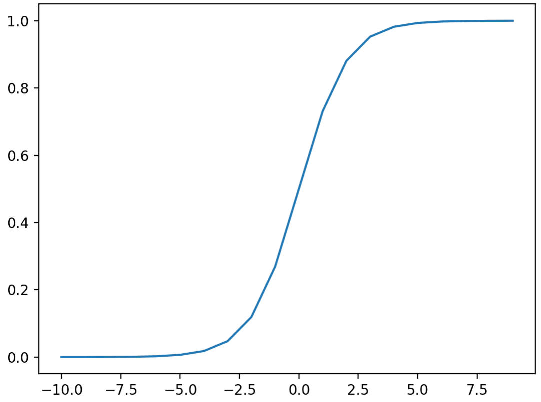
Minibatch Stochastic Gradient Descent:

- Use a subset of samples to compute the gradient and then update parameters.

14,197,122 samples in imagenet

Neural Networks

Building Blocks: Activation Functions



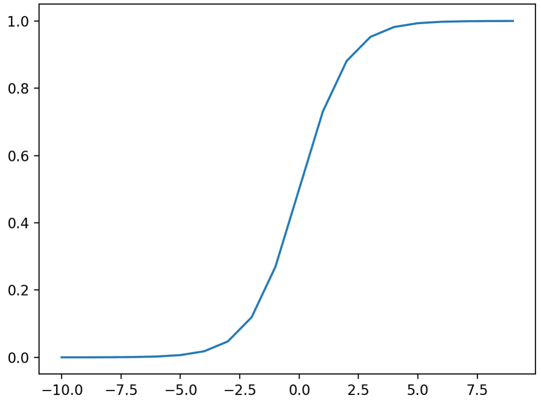
Sigmoid

Used for RNNs and binary classification output

https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c
https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

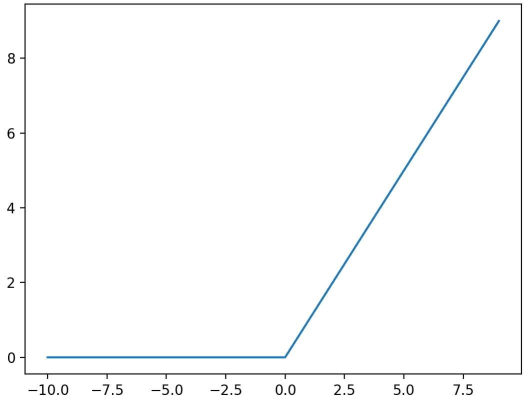
Neural Networks

Building Blocks: Activation Functions



Sigmoid

Used for RNNs and binary classification output



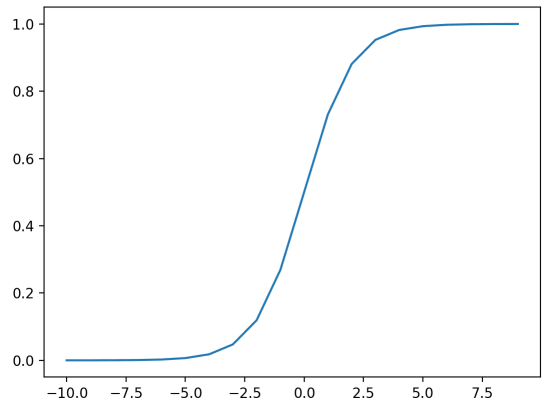
Rectified Linear Unit (ReLU)

Good general purpose fn, overcomes vanishing gradient issues

https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c
https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

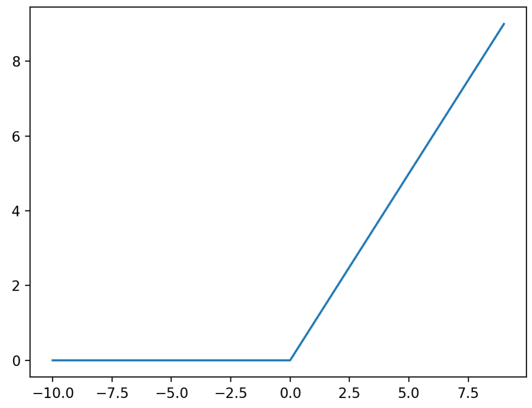
Neural Networks

Building Blocks: Activation Functions



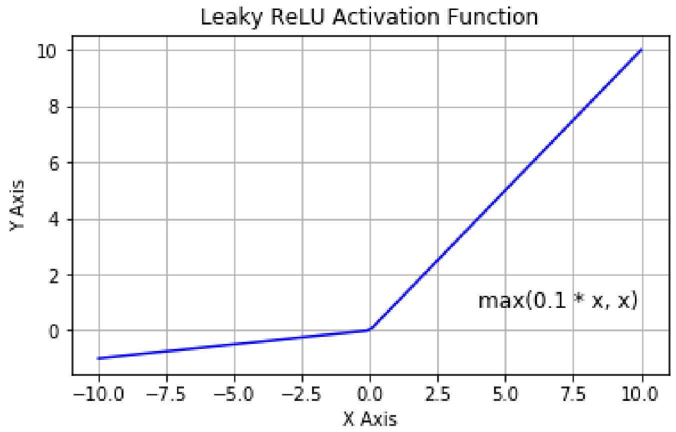
Sigmoid

Used for RNNs and binary classification output



Rectified Linear Unit (ReLU)

Good general purpose fn, overcomes vanishing gradient issues



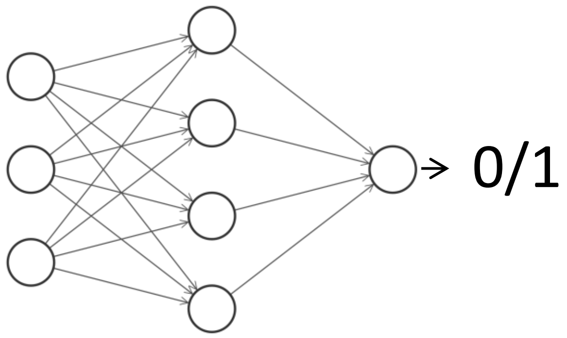
Leaky ReLU

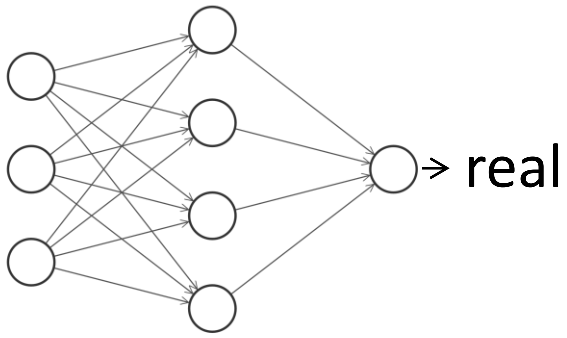
Overcomes "dying relu"

https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c
https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

Neural Networks

Building Blocks: Output Layer





Binary Classification

Single Node
Sigmoid activation

Regression

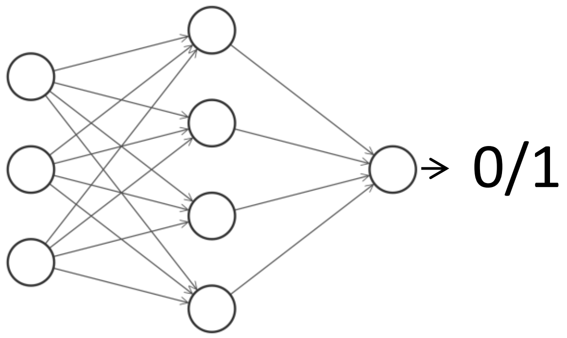
Single Node
Linear activation

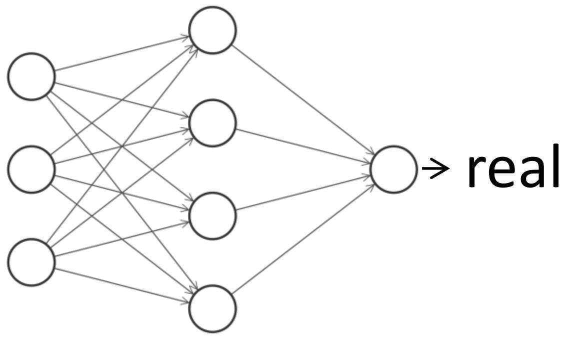
Neural Networks

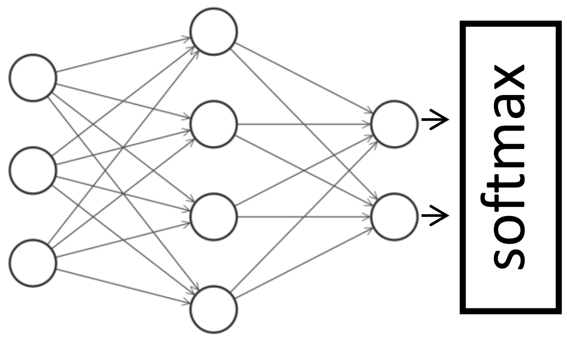
$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

softmax

Building Blocks: Output Layer







Binary Classification

Single Node
Sigmoid activation

Regression

Single Node
Linear activation

Classification

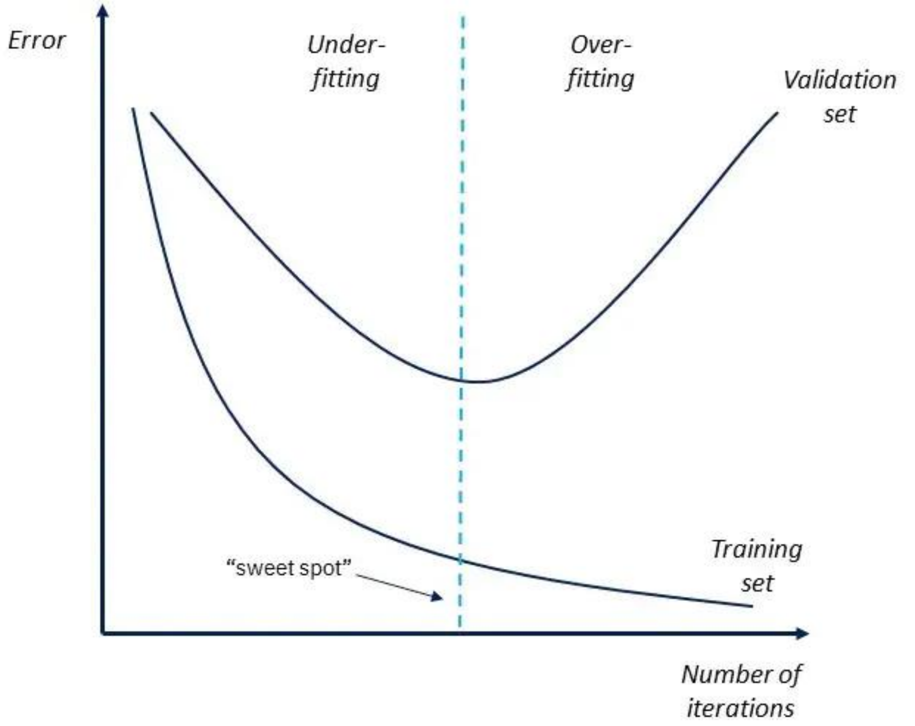
Multiple Nodes
ReLU activation with Softmax

Softmax converts multiple logits to probabilities

Diagnosing Neural Networks

Overfitting

Overfitting occurs when your network memorizes the training data.



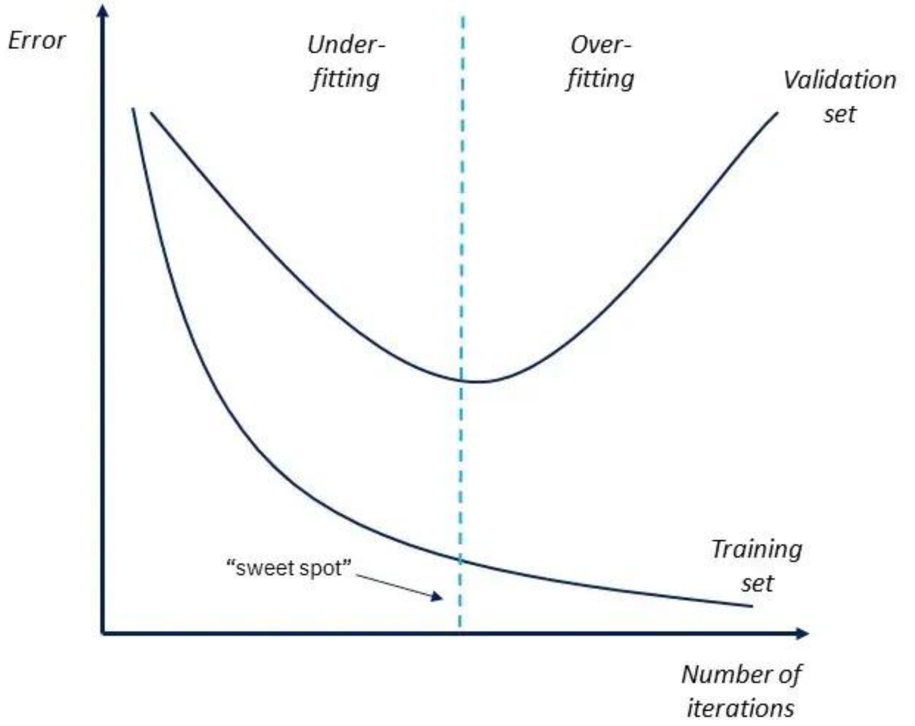
Diagnosing Neural Networks

Overfitting

Overfitting occurs when your network memorizes the training data.

Prevent Overfitting:

- Increase **dropout**. Dropout forces the network to learn multiple networks, increasing generalizability.
- Reduce **model depth**.
- **Early stop** training.



Neural Networks

Summary of Building Blocks

Fully Connected Layer: Every node is connected to every prior node.
Convolutional Layer: Apply a kernel to a sliding receptive field.
Max Pooling Layer: Select the largest value within the receptive field.
Softmax Layer: Rescale outputs for classification so that the outputs sum to 1.
Dropout Layer: Randomly removes nodes from a given layer.

ReLU Activation: Go-to standard activation function.
TanH Activation: Good activation for RNNs.
Sigmoid Activation: Output for binary classification. Good activation for RNNs.
Linear Activation: Output for regression.

Learning Rate: Size of gradient step.
Gradient Descent: Standard lr updates.
Adam Optimizer: Adaptive lr updates.
Batch Size: Number of samples to estimate the gradient

Preparing Your Data

“The literature on machine learning often reverses the meaning of ‘validation’ and ‘test’ sets. This is the most blatant example of the terminological confusion that pervades artificial intelligence research.” -MLM

Preparing Your Data

“The literature on machine learning often reverses the meaning of ‘validation’ and ‘test’ sets. This is the most blatant example of the terminological confusion that pervades artificial intelligence research.” -MLM

Train Set:
A set of examples used for learning, that is to fit the parameters of the classifier.

Validation Set:
A set of examples used to tune the parameters of a classifier.

Test Set:
A set of examples used only to assess the performance of a fully-specified classifier.

Preparing Your Data

"The literature on machine learning often reverses the meaning of 'validation' and 'test' sets. This is the most blatant example of the terminological confusion that pervades artificial intelligence research." -MLM

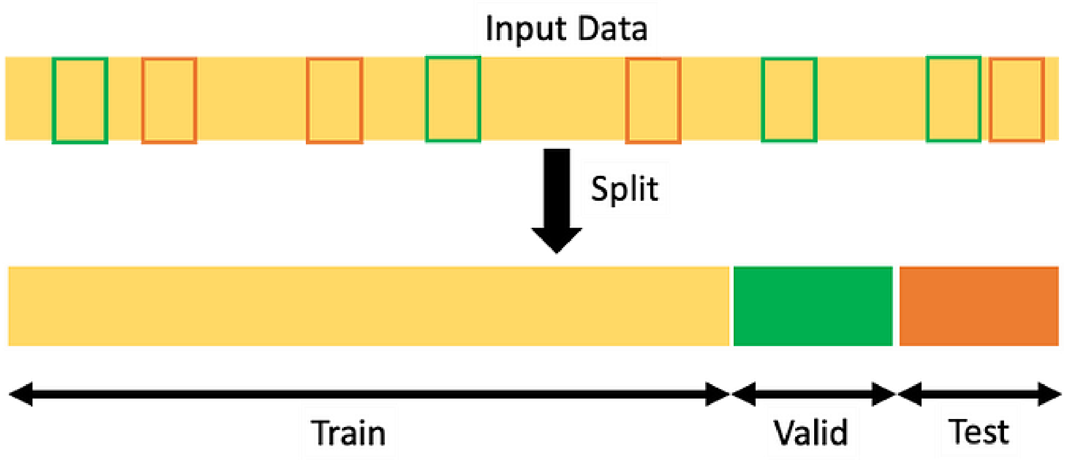
DO NOT MIX THESE

Train Set:
A set of examples used for learning, that is to fit the parameters of the classifier.

Validation Set:
A set of examples used to tune the parameters of a classifier.

Test Set:
A set of examples used only to assess the performance of a fully-specified classifier.

Preparing Your Data



Proper way to split your data.

Train Set:
A set of examples used for learning, that is to fit the parameters of the classifier.

Validation Set:
A set of examples used to tune the parameters of a classifier.

Test Set:
A set of examples used only to assess the performance of a fully-specified classifier.

Diagnosing Neural Networks

Training Neural Networks is Hard :(

Incorrect Learning Rate:
Learning rate can have a strong impact on performance. Try adjusting the learning rate to aid convergence or adding a learning rate decay.

Forgot Data Normalization:
Networks typically perform better with normalized data.

Using large batch sizes:
Evidence suggests that overly large batch sizes inhibit learning.

Wrong Activation Function:
Some applications, e.g. Binary Classification, perform best with specific activation functions.

Network is too Deep:
Practically, deep networks are not always better. Start with a smaller network and grow it as needed.

https://arxiv.org/pdf/1206.5533.pdf

https://www.kaggle.com/general/196487

s://stats.stackexchange.com/questions/164876/what-is-the-trade-off-between-batch-size-and-number-of-iterations-to-train-a-neu

Diagnosing Neural Networks

Training Neural Networks is Hard :(

Incorrect Learning
Learning rate can
performance. Try
rate to aid conver
learning rate deca

Forgot Data Norm
Networks typically
normalized data.

y
vith specific

e not always
etwork and

Number One Cause of Network Failure:

Incorrect or improperly transformed input data.

Before training, be sure to inspect your input data, and visualize network inputs and outputs during training.

Using large batch sizes:
Evidence suggests that overly large batch sizes inhibit learning.

https://arxiv.org/pdf/1206.5533.pdf

https://www.kaggle.com/general/196487

s://stats.stackexchange.com/questions/164876/what-is-the-trade-off-between-batch-size-and-number-of-iterations-to-train-a-neu

Practicals

Operating Systems

Machine learning developers overwhelmingly use Linux.

The open source nature of Linux environments lends itself well to the complex installation and configuration processes required by many machine learning applications. Many machine learning algorithms are built on Linux already, fueling easy adoption.

https://medium.com/mlearning-ai/setup-deep-learning-environment-with-ubuntu-22-04-8ad40cb28f95

Operating Systems

Machine learning developers overwhelmingly use Linux.

The open source nature of Linux environments lends itself well to the complex installation and configuration processes required by many machine learning applications. Many machine learning algorithms are built on Linux already, fueling easy adoption.

WSL (Windows Subsystem for Linux):
Lets developers run a GNU/Linux environment -- including most command-line tools, utilities, and applications -- directly on Windows

https://medium.com/mlearning-ai/setup-deep-learning-environment-with-ubuntu-22-04-8ad40cb28f95

Development Environment

Dependencies

We’ll be using python to code our Neural Network

Effective python coding requires managing your dependencies

ML is usually taught using Colab, which does this (mostly) for you

Development Environment

Dependencies

We'll be using python to code our Neural Network

Effective python coding requires managing your dependencies

ML is usually taught using Colab, which does this (mostly) for you

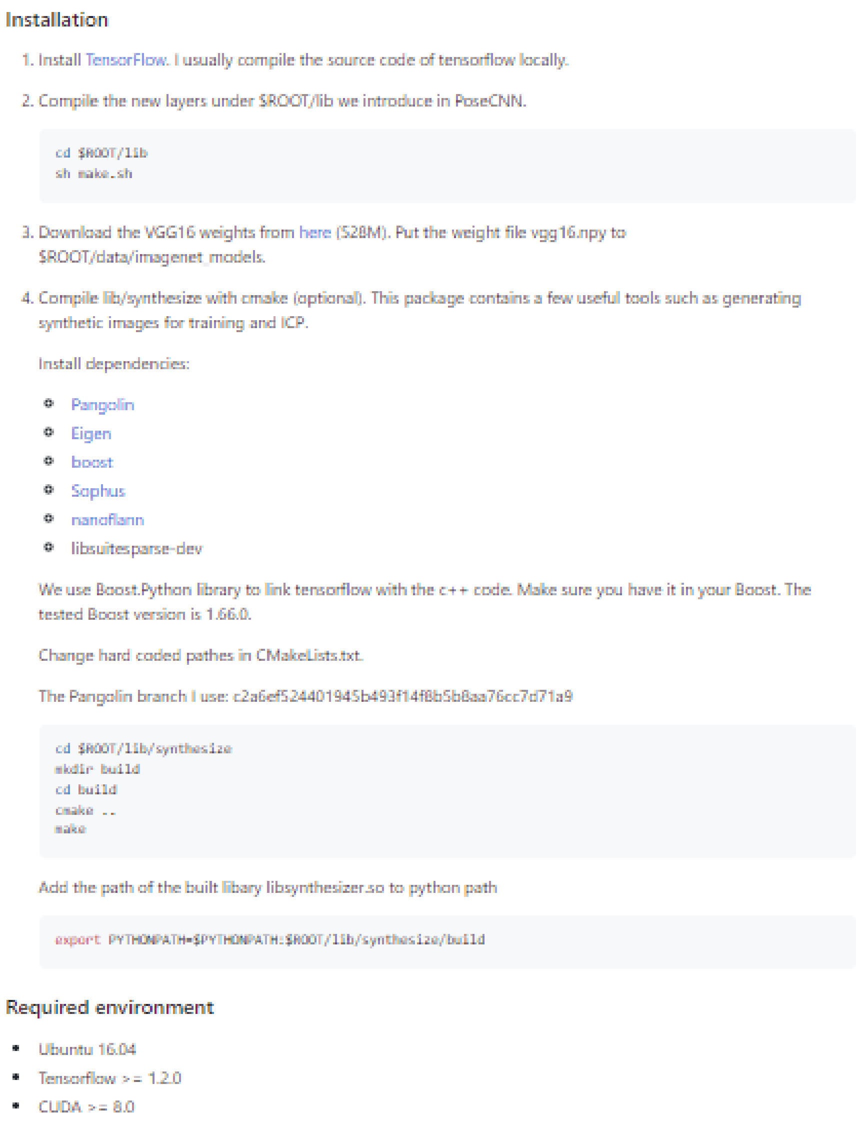
```
from torch.utils.data import Dataset, DataLoader

import pickle
import numpy as np

class MyDataset(Dataset):
  def __init__(self, data, transforms=None, target_transforms=None):
```

Development Environment

We'll be using python to code our Neural Network

Effective python coding requires managing your dependencies

ML is usually taught using Colab, which does this (mostly) for you



Installation

1. Install TensorFlow. I usually compile the source code of tensorflow locally.
2. Compile the new layers under $ROOT/lib we introduce in PoseCNN.

```
cd $ROOT/lib
sh make.sh
```

3. Download the VGG16 weights from here (528M). Put the weight file vgg16.npy to $ROOT/data/imagenet_models.
4. Compile lib/synthesize with cmake (optional). This package contains a few useful tools such as generating synthetic images for training and ICP.

   Install dependencies:

   - Pangolin
   - Eigen
   - boost
   - Sophus
   - nanoflann
   - libsuitesparse-dev

   We use Boost.Python library to link tensorflow with the c++ code. Make sure you have it in your Boost. The tested Boost version is 1.66.0.

   Change hard coded pathes in CMakeLists.txt.

   The Pangolin branch I use: c2a6ef524401945b493f14f8b5b8aa76cc7d71a9

```
cd $ROOT/lib/synthesize
mkdir build
cd build
cmake ..
make
```

   Add the path of the built libary libsynthesizer.so to python path

```
export PYTHONPATH=$PYTHONPATH:$ROOT/lib/synthesize/build
```

Required environment

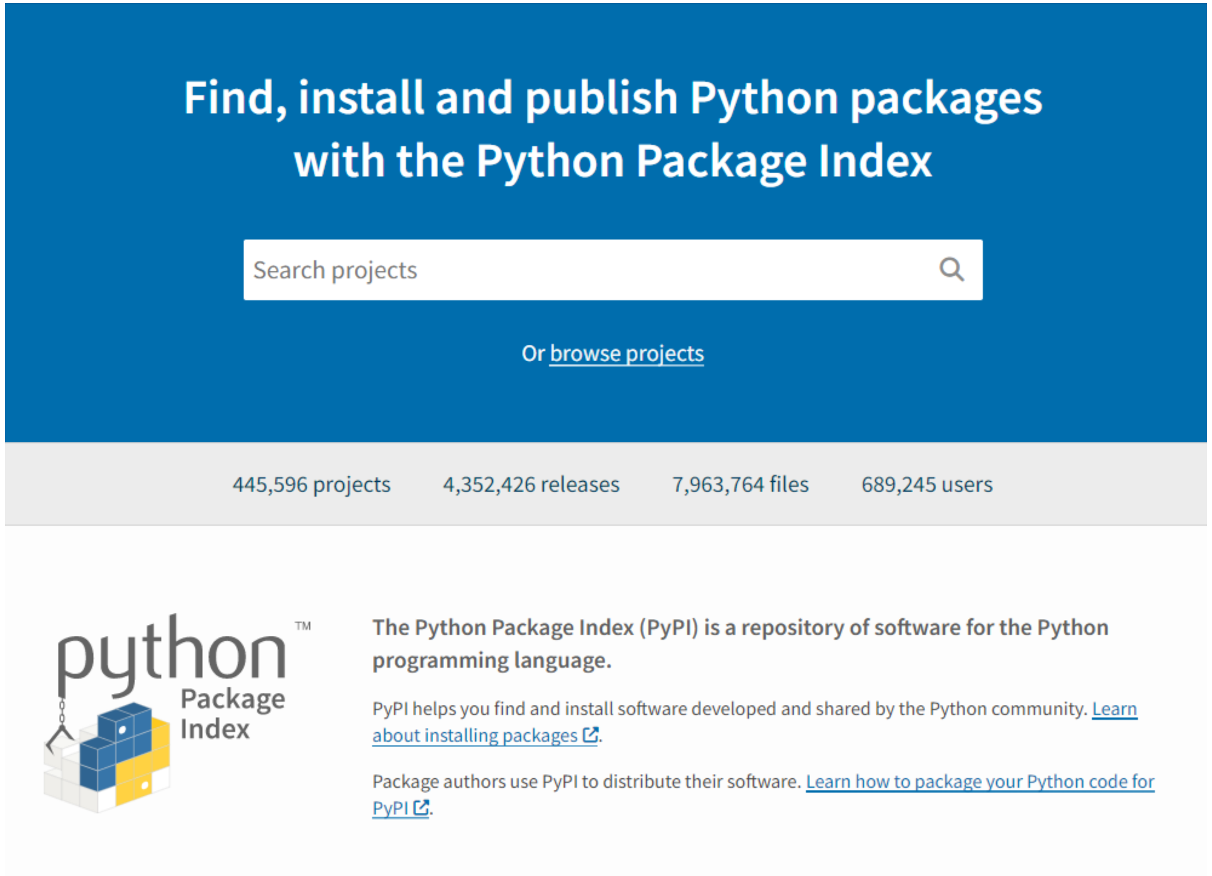
- Ubuntu 16.04
- Tensorflow >= 1.2.0
- CUDA >= 8.0

Development Environment

Dependencies

Most dependencies are installed with PIP (pip installs packages)

Interfaces with the Python Package Index



Environments (containerization)

For real applications, you'll need to install your own dependencies.

We do this inside a virtual environment.

python environment

numpy
torch
pandas

Environments (containerization)

For real applications, you'll need to install your own dependencies.

We do this inside a virtual environment.

Why is this good?

1) You can instruct people how to install your dependences.
2) You can work on multiple projects at the same time.
3) When you inevitably screw something up, makes it easy to start over.

python environment

numpy
torch
pandas

Environments (containerization)

For real applications, you'll need to install your own dependencies.

We do this inside a virtual environment.

Why is this good?

1) You can instruct people how to install your dependences.
2) You can work on multiple projects at the same time.
3) When you inevitably screw something up, makes it easy to start over.

virtualenv:
Python package manager bundled with python. Sufficient for most projects.

anaconda:
System package manager with a gui. Useful for installing complex dependencies that extend beyond python.

docker:
Machine-level virtualization. Useful for installing and distributing projects with extremely complex dependencies.

Tools

VSCode:
Lightweight IDE replacement that is extremely popular for python programing with an extensive extensions library.

Screen:
Enables training “in the background”.

Virtualenv:
Python environment containerization.

Tools

VSCode:
Lightweight IDE replacement that is extremely popular for python programing with an extensive extensions library.
Screen:
Enables training “in the background”.
Virtualenv:
Python environment containerization.

Remote Explorer (VSCode Extension):
Connect to remote servers through VSCode.
Pylance (VSCode Extension):
IntelliSense for python.

Tools

VSCode:
Lightweight IDE replacement that is extremely popular for python programing with an extensive extensions library.

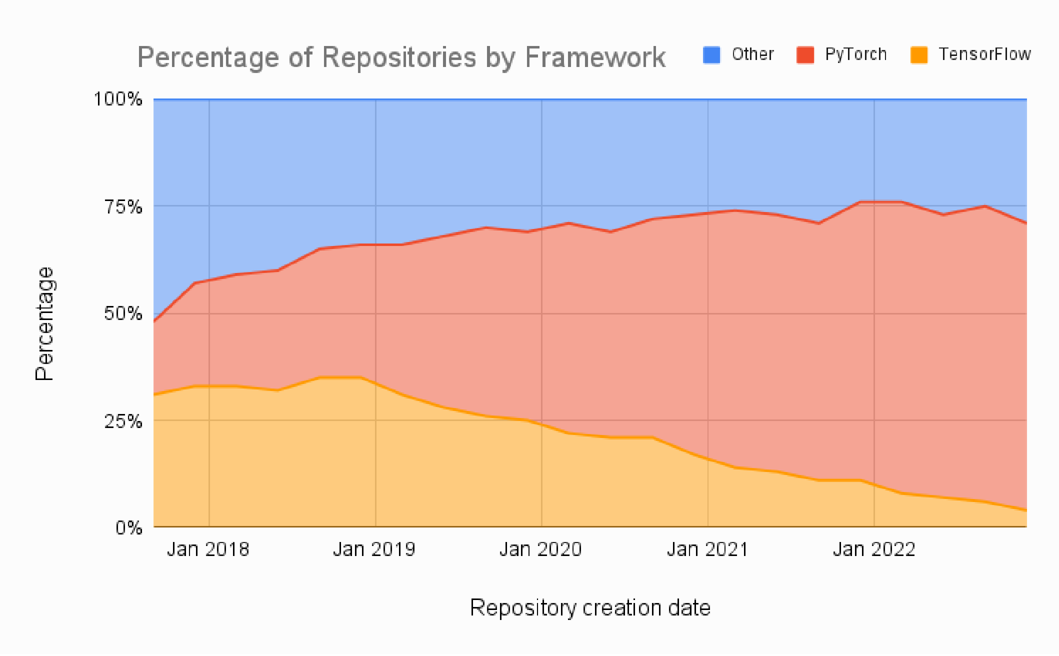
Screen:
Enables training “in the background”.

Virtualenv:
Python environment containerization.

PyTorch:
Machine learning framework based on the open-source torch library.

Remote Explorer (VSCode Extension):
Connect to remote servers through VSCode.

Pylance (VSCode Extension):
IntelliSense for python.



Using Virtual Environments

virtualenv

```
lambne@tarsserver2:~/dev/classes/cs473$ python3 -m venv env
lambne@tarsserver2:~/dev/classes/cs473$ source env/bin/activate
```

Using Virtual Environments

virtualenv

```
lambne@tarsserver2:~/dev/classes/cs473$ python3 -m venv env
lambne@tarsserver2:~/dev/classes/cs473$ source env/bin/activate
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip install pyjokes
Collecting pyjokes
  Downloading pyjokes-0.6.0-py2.py3-none-any.whl (26 kB)
Installing collected packages: pyjokes
Successfully installed pyjokes-0.6.0
(env) lambne@tarsserver2:~/dev/classes/cs473$ pyjoke
An SEO expert walks into a bar, bars, pub, public house, Irish pub, tavern, bartender, beer, liquor, wine, alcohol, spirits...
```

Using Virtual Environments

virtualenv

```
lambne@tarsserver2:~/dev/classes/cs473$ python3 -m venv env
lambne@tarsserver2:~/dev/classes/cs473$ source env/bin/activate
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip install pyjokes
Collecting pyjokes
  Downloading pyjokes-0.6.0-py2.py3-none-any.whl (26 kB)
Installing collected packages: pyjokes
Successfully installed pyjokes-0.6.0
(env) lambne@tarsserver2:~/dev/classes/cs473$ pyjoke
An SEO expert walks into a bar, bars, pub, public house, Irish pub, tavern, bartender, beer, liquor, wine, alcohol, spirits...
(env) lambne@tarsserver2:~/dev/classes/cs473$ deactivate
```

Using Virtual Environments

virtualenv

```
lambne@tarsserver2:~/dev/classes/cs473$ python3 -m venv env
lambne@tarsserver2:~/dev/classes/cs473$ source env/bin/activate
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip install pyjokes
Collecting pyjokes
  Downloading pyjokes-0.6.0-py2.py3-none-any.whl (26 kB)
Installing collected packages: pyjokes
Successfully installed pyjokes-0.6.0
(env) lambne@tarsserver2:~/dev/classes/cs473$ pyjoke
An SEO expert walks into a bar, bars, pub, public house, Irish pub, tavern, bartender, beer, liquor, wine, alcohol, spirits...
(env) lambne@tarsserver2:~/dev/classes/cs473$ deactivate
lambne@tarsserver2:~/dev/classes/cs473$ rm -rf env/
lambne@tarsserver2:~/dev/classes/cs473$ 
```

Using Virtual Environments

virtualenv

Every project should come with a requirements.txt file

requirements.txt

home > lambne > requirements.txt

```
PyOpenGL==3.1.0
pyparsing==3.0.9
pyrender==0.1.45
python-dateutil==2.8.2
PyWavelets==1.3.0
scikit-image==0.19.3
scikit-learn==1.1.1
scipy==1.8.1
six==1.16.0
threadpoolctl==3.1.0
tifffile==2022.5.4
torch==1.12.0
tqdm==4.64.0
trimesh==3.12.8
Twisted==22.4.0
txaio==22.2.1
typing_extensions==4.3.0
vedo==2022.2.3
vtk==9.0.3
wrapt==1.14.1
wslink==1.6.6
yarl==1.7.2
```

Using Virtual Environments

virtualenv

Every project should come with a requirements.txt file

To create a requirements.txt file:

```
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip list
Package       Version
------------- -------
pip           20.0.2
pkg-resources 0.0.0
pyjokes       0.6.0
setuptools    44.0.0
wheel         0.34.2
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip freeze > requirements.txt
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip install -r requirements.txt
Requirement already satisfied: pyjokes==0.6.0 in ./env/lib/python3.8/site-packages (from -r requirements.txt (line 1)) (0.6.0)
(env) lambne@tarsserver2:~/dev/classes/cs473$
```

Using Virtual Environments

virtualenv

Every project should come with a requirements.txt file

To create a requirements.txt file:

```
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip list
Package       Version
------------- -------
pip           20.0.2
pkg-resources 0.0.0
pyjokes       0.6.0
setuptools    44.0.0
wheel         0.34.2
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip freeze > requirements.txt
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip install -r requirements.txt
Requirement already satisfied: pyjokes==0.6.0 in ./env/lib/python3.8/site-packages (from -r requirements.txt (line 1)) (0.6.0)
(env) lambne@tarsserver2:~/dev/classes/cs473$
```

Coding

Using Virtual Environments

virtualenv

Every project should come with a requirements.txt file

To create a requirements.txt file:

```
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip list
Package       Version
------------- -------
pip           20.0.2
pkg-resources 0.0.0
pyjokes       0.6.0
setuptools    44.0.0
wheel         0.34.2
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip freeze > requirements.txt
(env) lambne@tarsserver2:~/dev/classes/cs473$ pip install -r requirements.txt
Requirement already satisfied: pyjokes==0.6.0 in ./env/lib/python3.8/site-packages (from -r requirements.txt (line 1)) (0.6.0)
(env) lambne@tarsserver2:~/dev/classes/cs473$
```

1) Data
2) Model
3) Training Loop

Using PyTorch

*1) **Data***
2) Model
3) Training Loop

Using PyTorch

```python
from torch.utils.data import Dataset
import numpy as np
from PIL import Image


class CustomDataset(Dataset):
    def __init__(self, split, img_dir, transform=None, target_transform=None):
        self.split = split
        self.img_dir = img_dir
        self.data = []
        self.labels = []
        img_dir = os.path.join(img_dir, split)
        for f in os.listdir(img_dir):
            img = Image.open(os.path.join(img_dir, f))
            label = f.split(".")[0]
            label = int(label)
            self.data.append(img)
            self.labels.append(label)
        print(f"Loaded {len(self.data)} images")
        self.transform = transform
        self.target_transform = target_transform

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        image = self.data[idx]
        label = self.labels[idx]
        if self.transform:
            image = self.transform(image)
        if self.target_transform:
            label = self.target_transform(label)
        return image, label
```

The dataset handles loading and transforming the input data.

Must implement the __len__ and __getitem__ methods.

1) Data
2) Model
3) Training Loop

Using PyTorch

Model defines the network you're going to use.

Must implement the forward method.

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.fc1 = nn.Linear(9216, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        output = F.log_softmax(x, dim=1)
        return output
```

1) Data
2) Model
3) ***Training Loop***

Using PyTorch

The training loop does the following:

- Set model mode to train
- Iterate over the data and:
  - Reset the gradients
  - Pass the data through the network
  - Compute the loss
  - Perform backpropogation
  - Take a gradient step

```
def train(model, device, train_loader, optimizer):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
```

https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html

Resources

Machine learning mastery list of all tutorials:
https://machinelearningmastery.com/start-here/#statistical_methods

Stanford introduction to Deep Learning:
http://ufldl.stanford.edu/tutorial/

Neural Networks and Deep Learning:
http://neuralnetworksanddeeplearning.com/chap1.html

Deep Learning Specialization:
https://www.coursera.org/specializations/deep-learning

Convolutional Calculator:
https://madebyollin.github.io/convnet-calculator/

Figure Creation Resources

Neural Network Diagram Maker:
https://alexlenail.me/NN-SVG/index.html

Common Latex Formulas:
https://blmoistawinde.github.io/ml_equations_latex/#softmax

Latex Compiler:
https://latex.codecogs.com/eqneditor/editor.php