

Deep Learning

CS473/CS573: Computer Vision

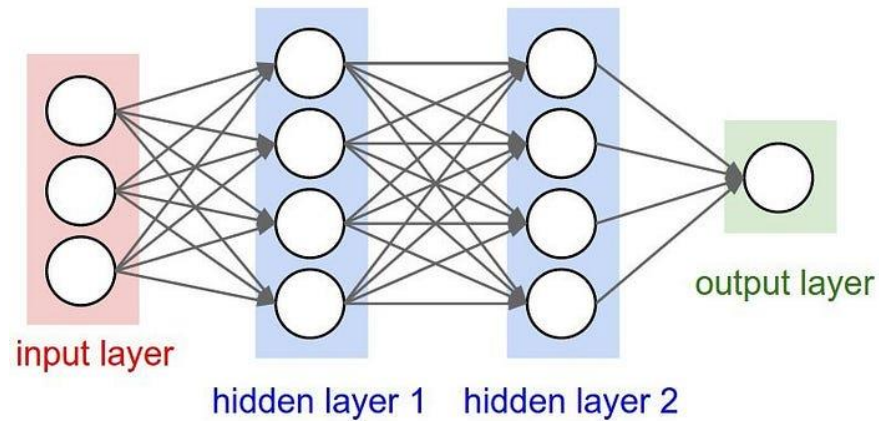
What is Deep Learning?

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.

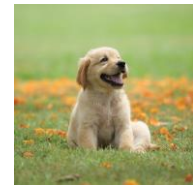
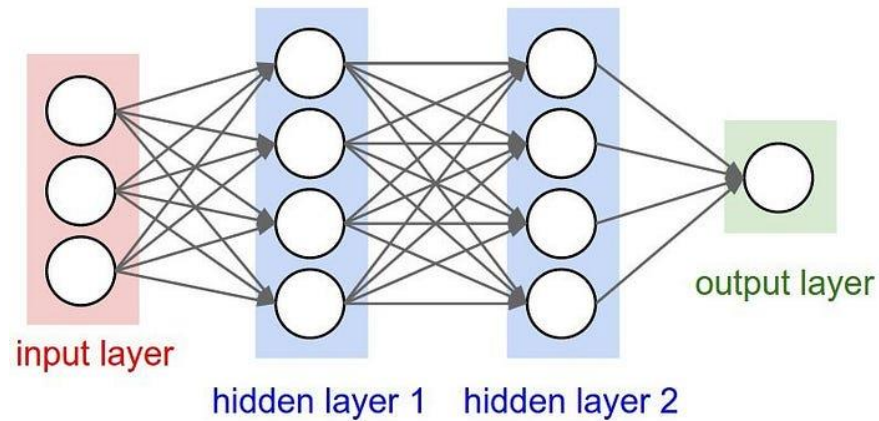


Deep

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?



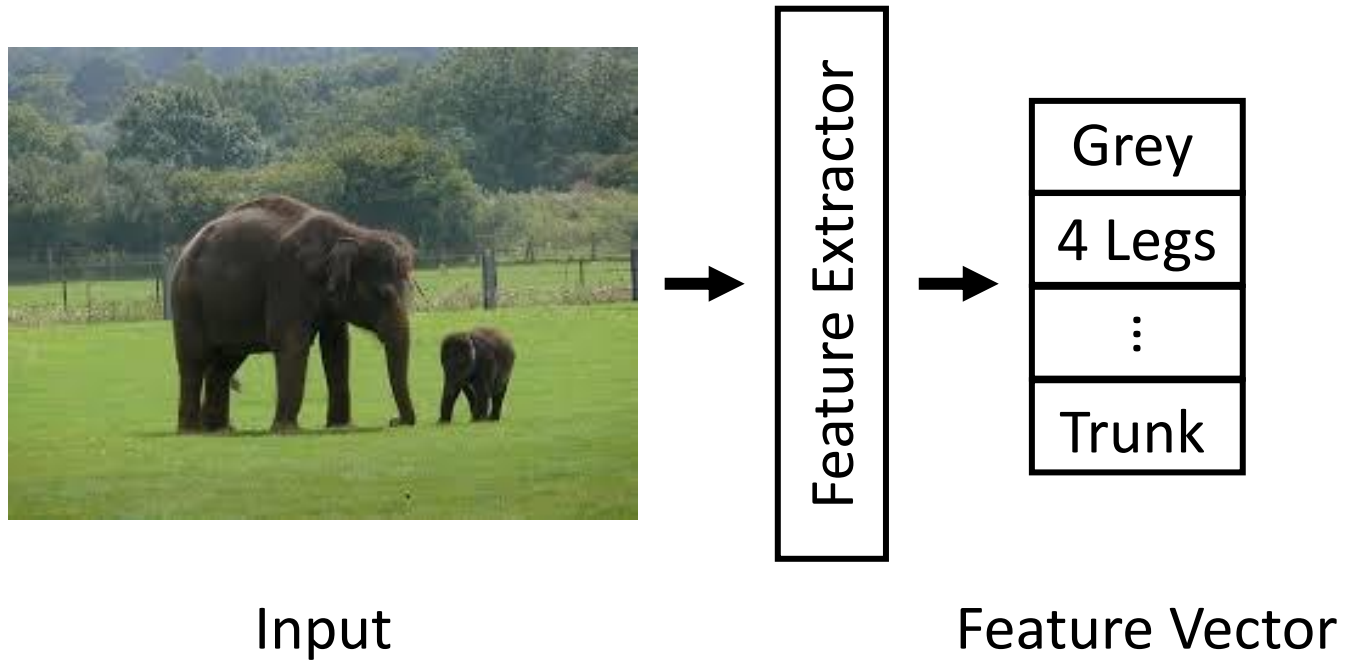
Feature Extractor

Input

What animal is in this image?

Machine Learning

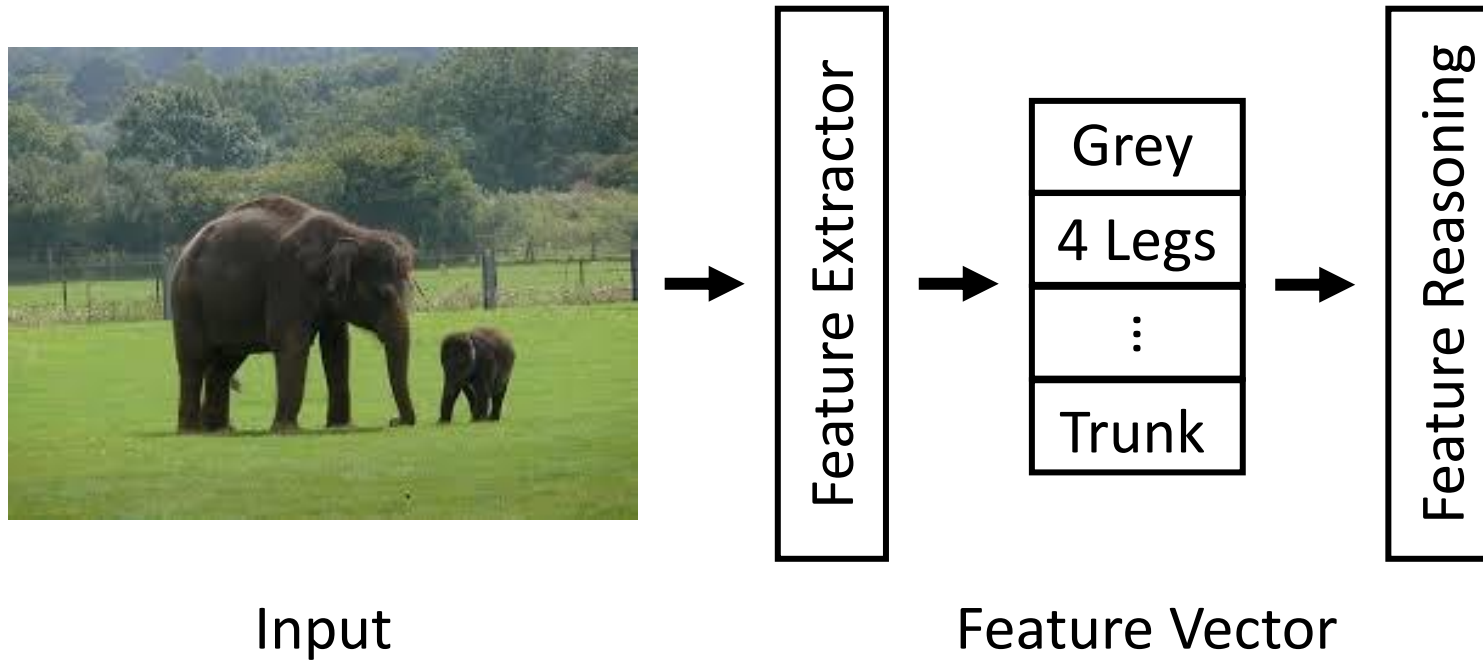
How do machines learn?



What animal is in this image?

Machine Learning

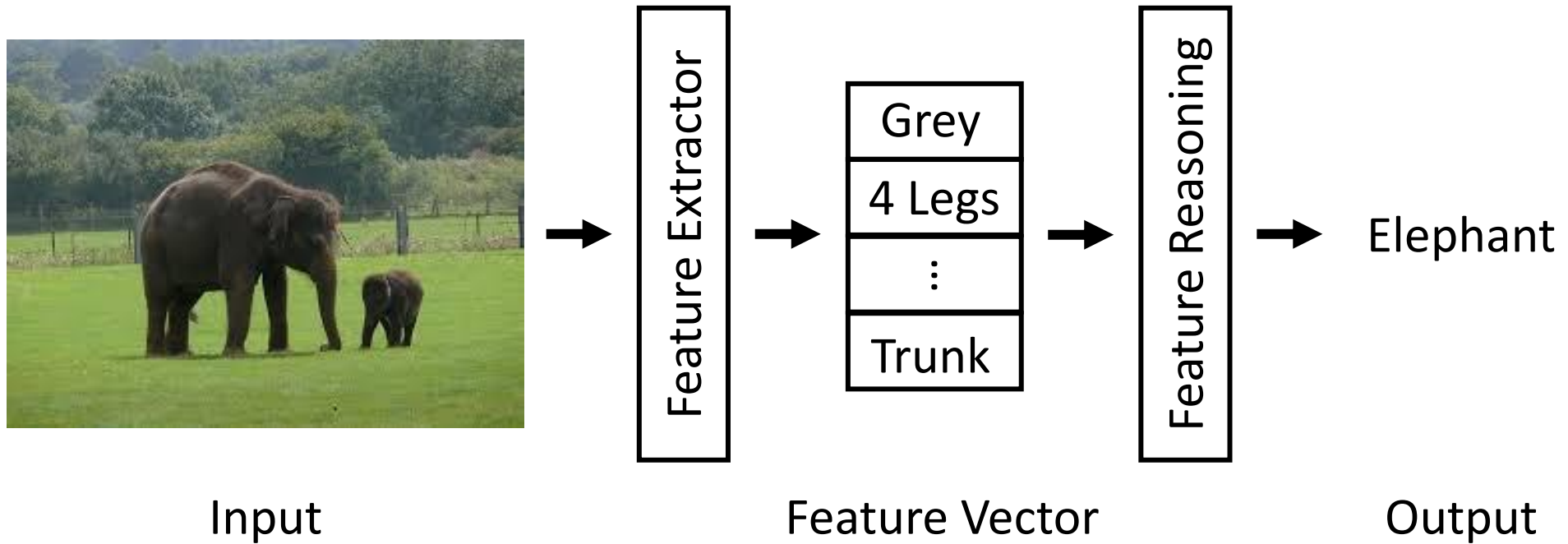
How do machines learn?



What animal is in this image?

Machine Learning

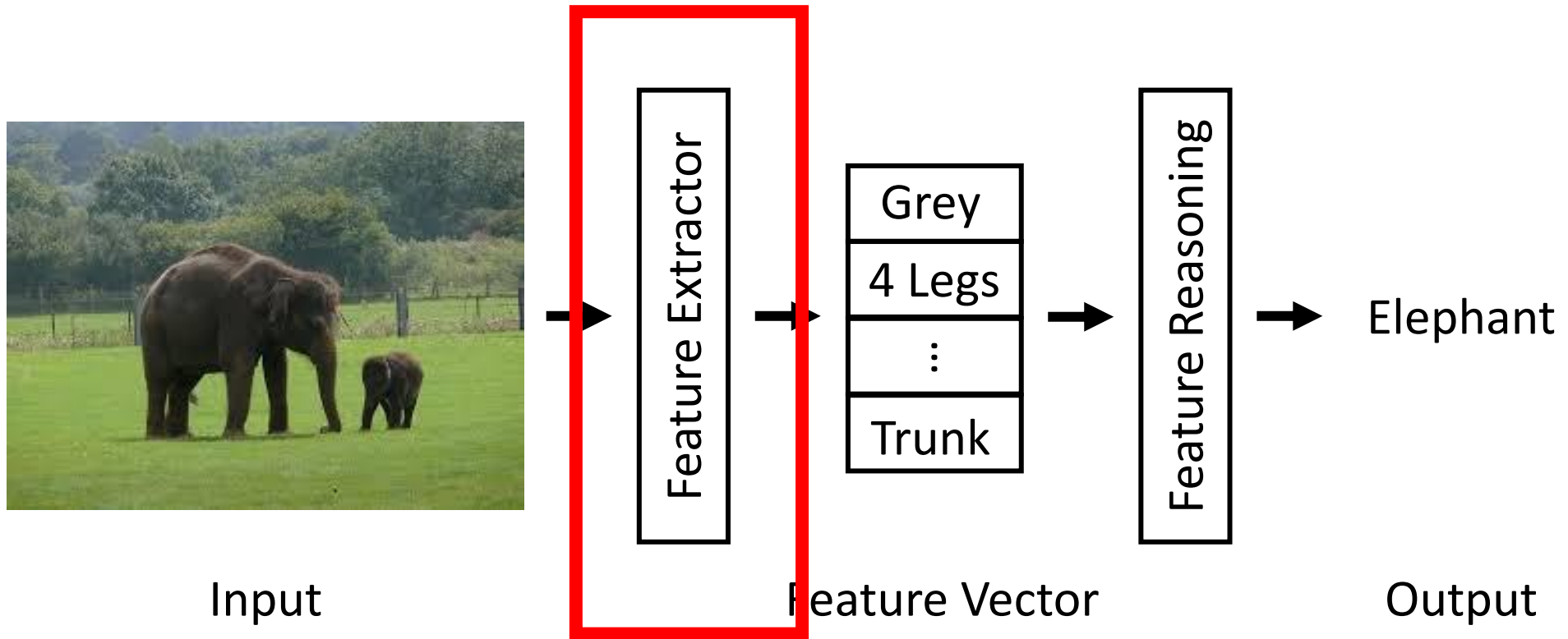
How do machines learn?



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



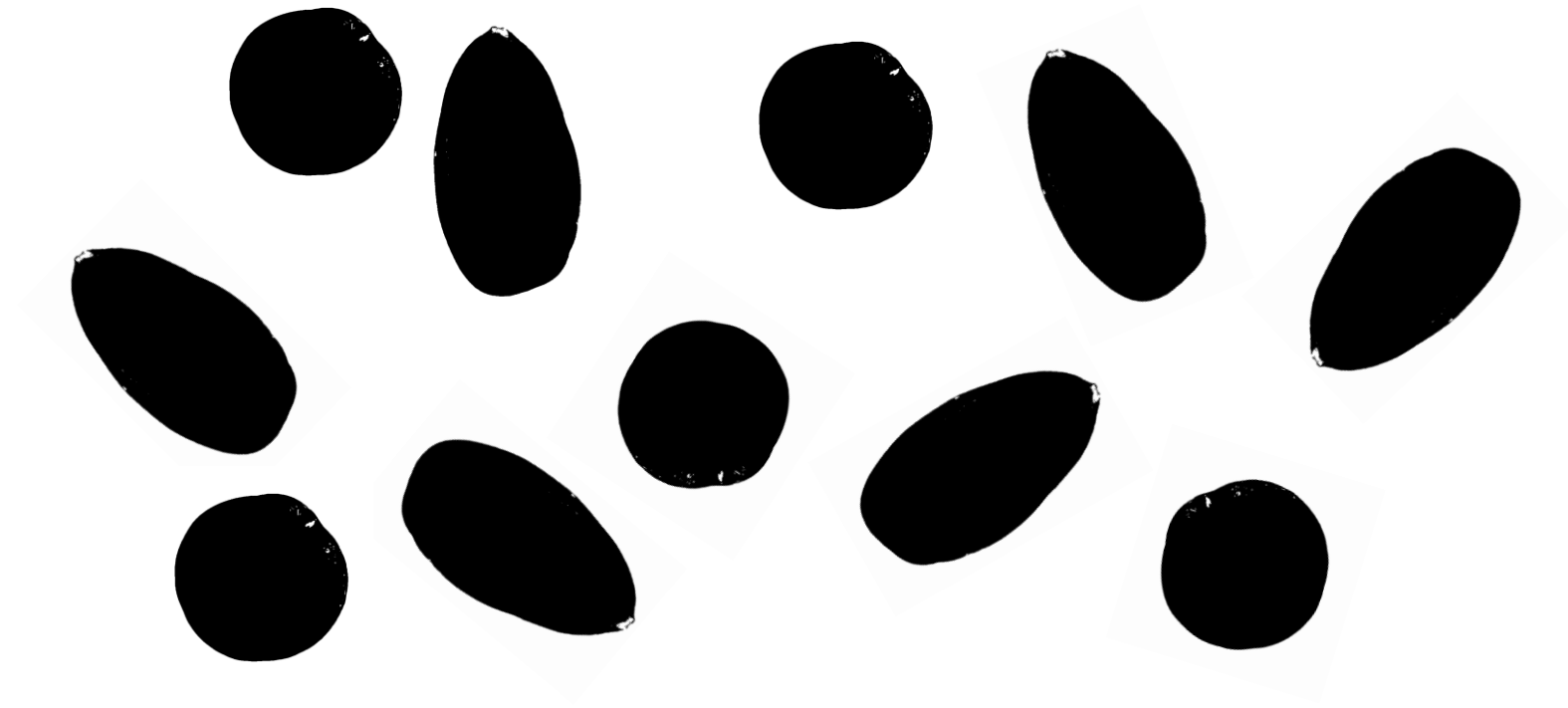
Feature Extraction

Example: let's classify hazelnuts and almonds



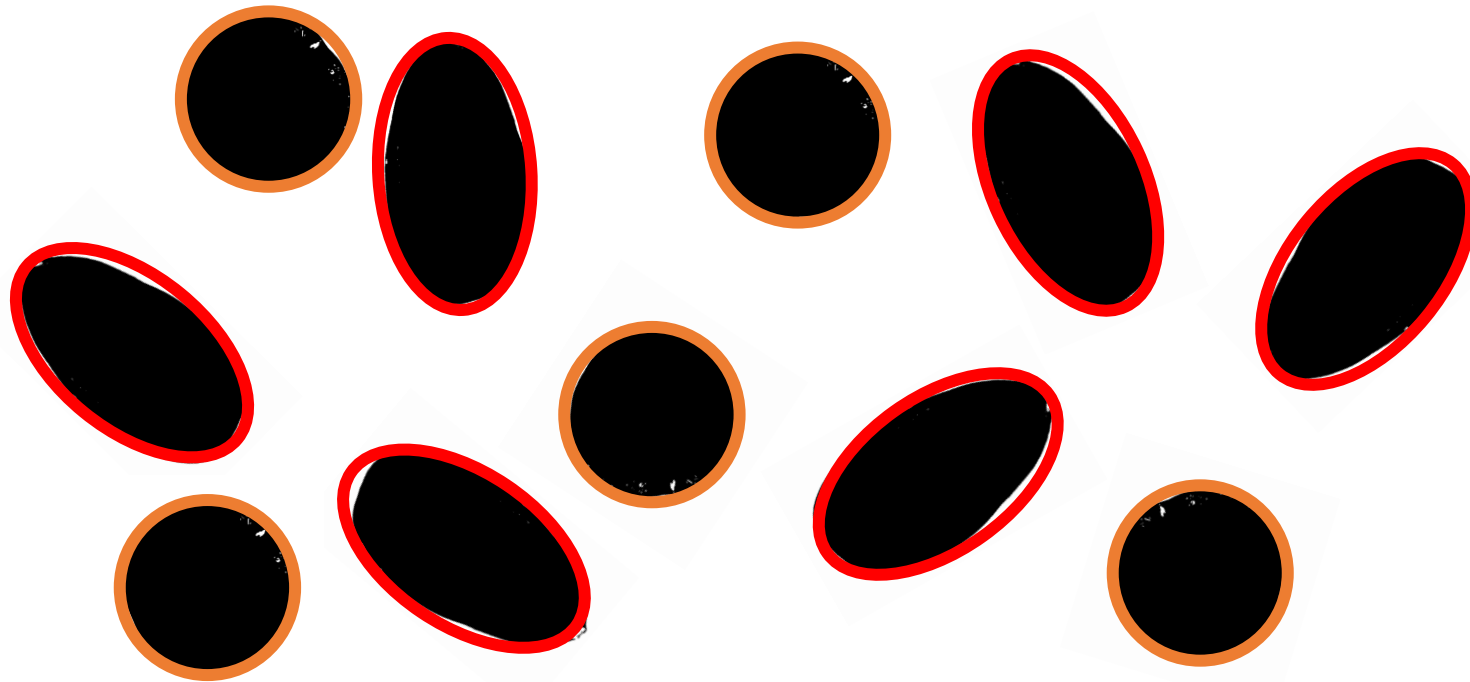
Feature Extraction

Example: let's classify hazelnuts and almonds



Feature Extraction

Example: let's classify hazelnuts and almonds



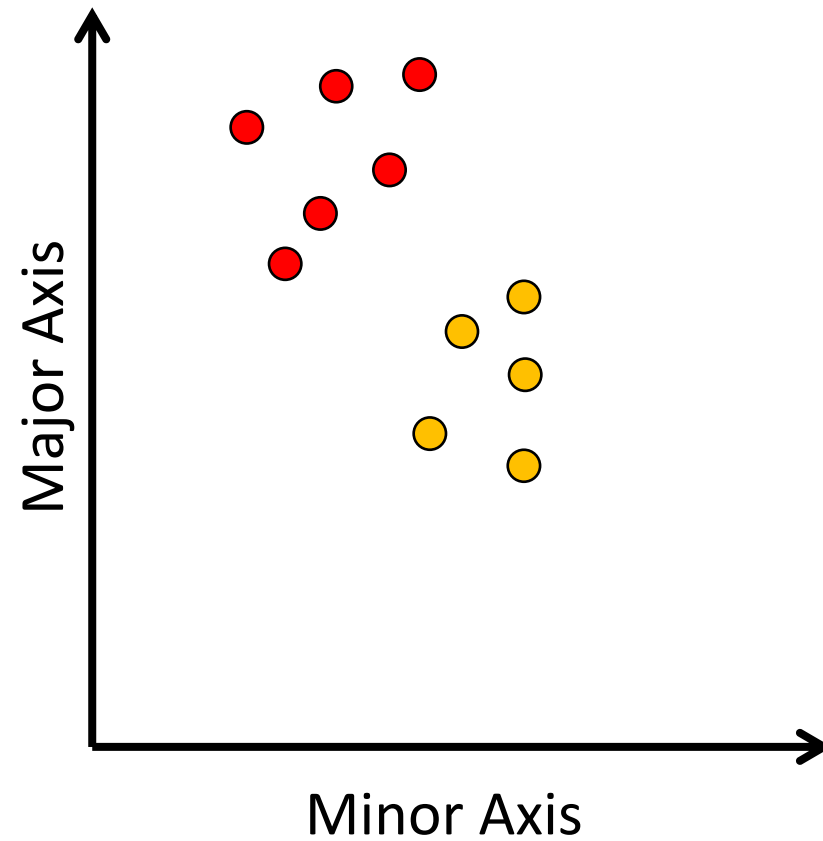
Feature Extraction

Example: let's classify hazelnuts and almonds



Feature Extraction

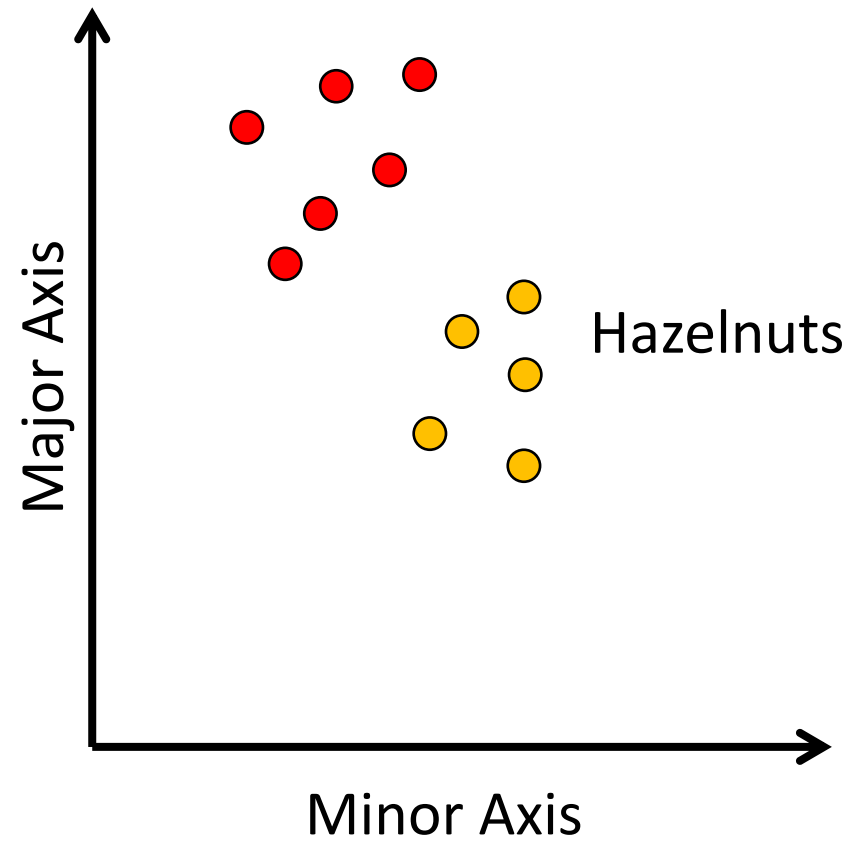
Example: let's classify hazelnuts and almonds



Hazelnuts vs Almonds

Feature Extraction

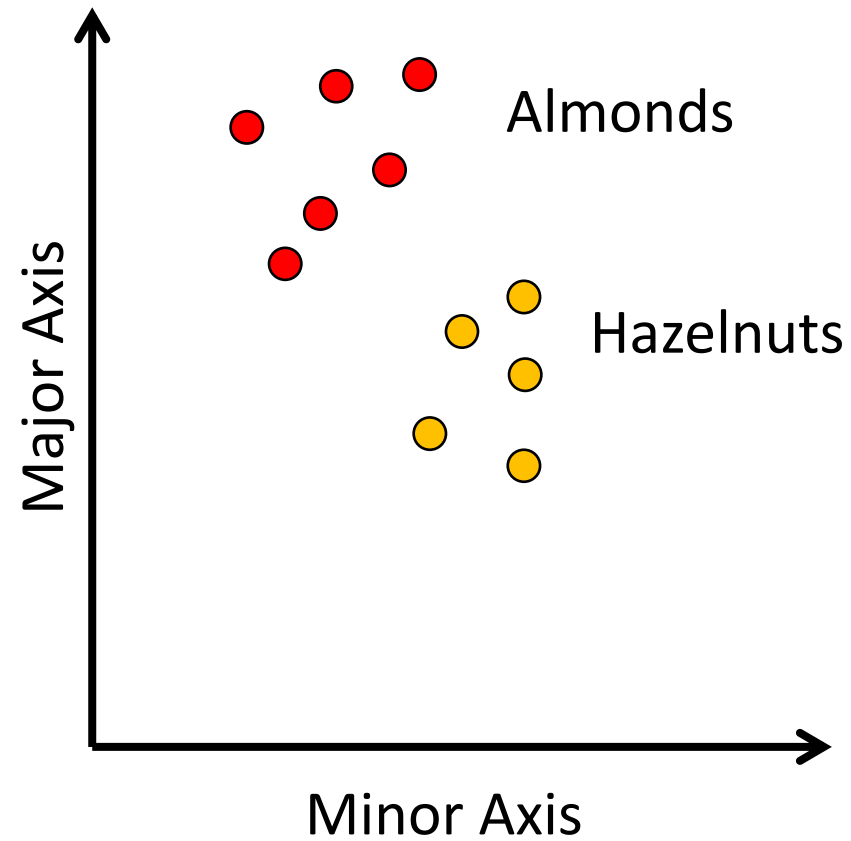
Example: let's classify hazelnuts and almonds



Hazelnuts vs Almonds

Feature Extraction

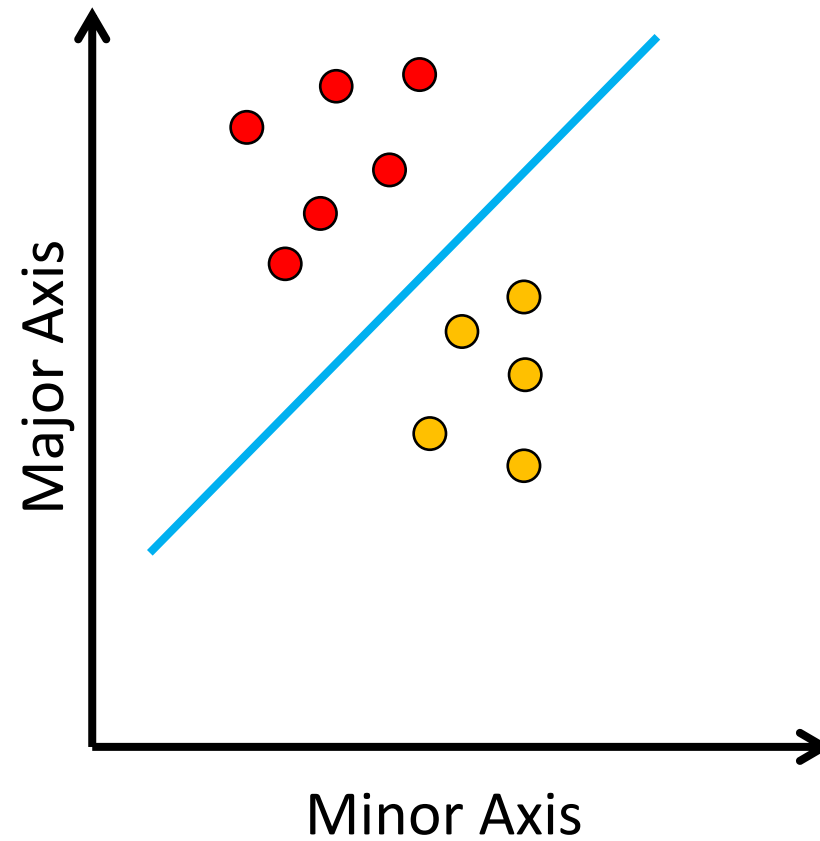
Example: let's classify hazelnuts and almonds



Hazelnuts vs Almonds

Feature Extraction

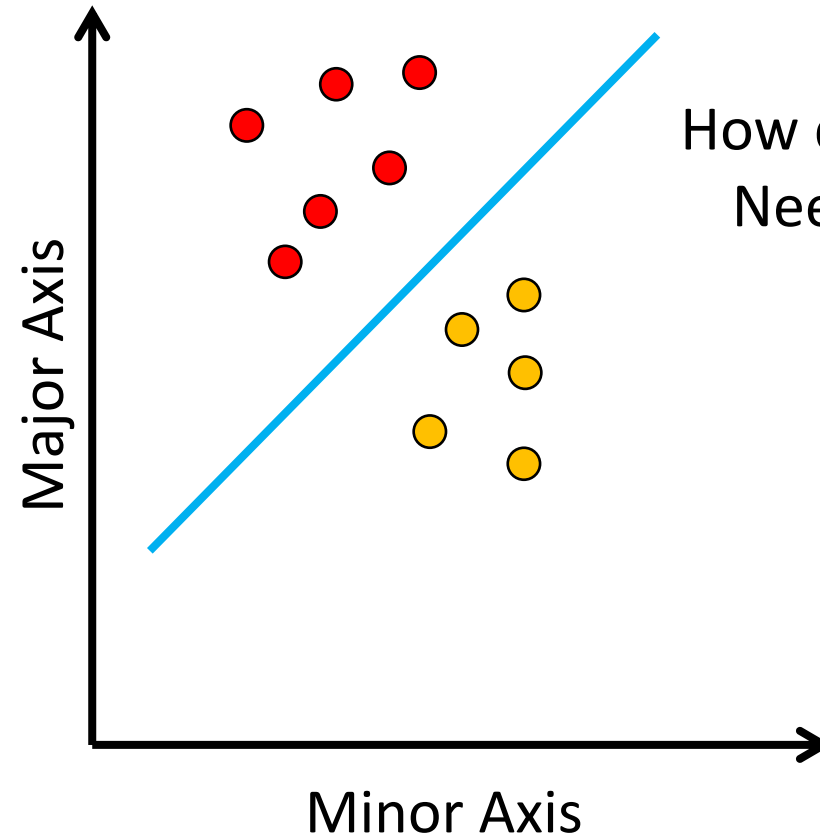
Example: let's classify hazelnuts and almonds



Hazelnuts vs Almonds

Feature Extraction

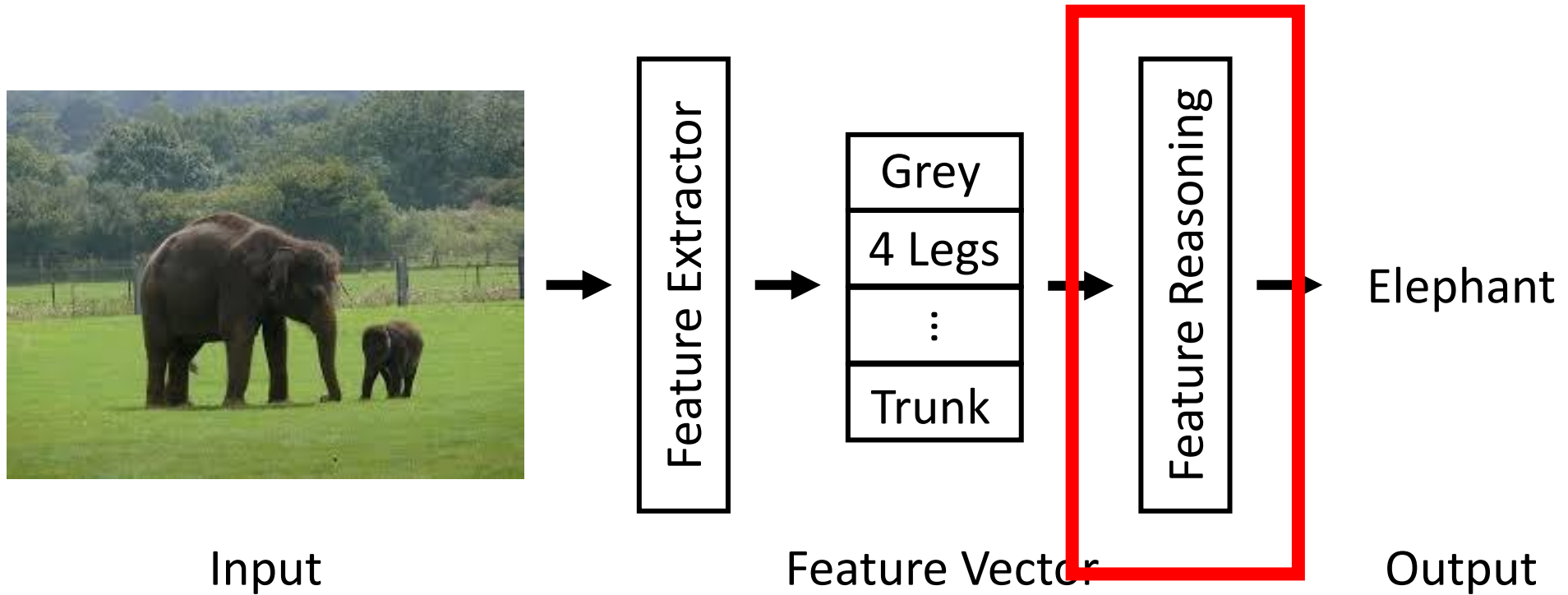
Example: let's classify hazelnuts and almonds



How do we obtain this line?
Need Machine Learning

Hazelnuts vs Almonds

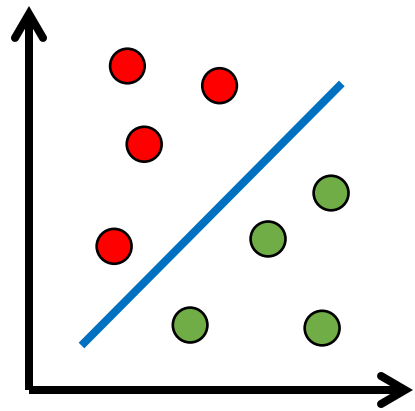
Machine Learning



What animal is in this image?

Machine Learning

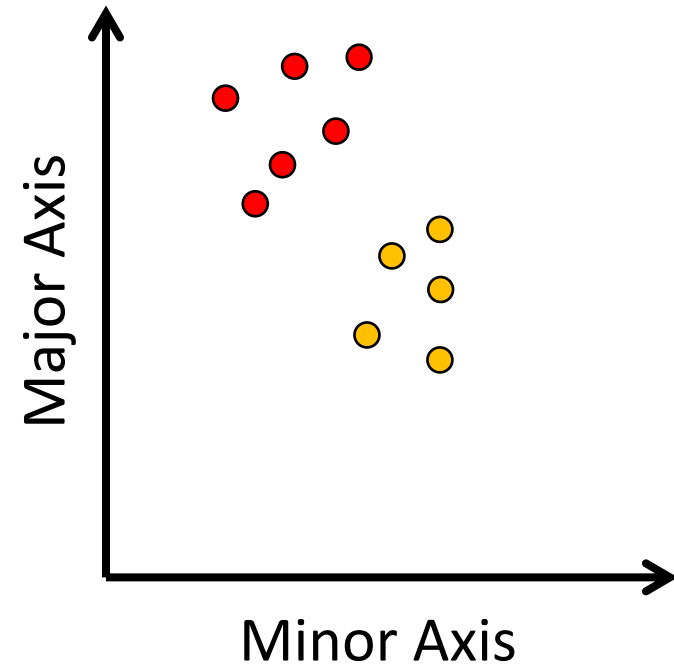
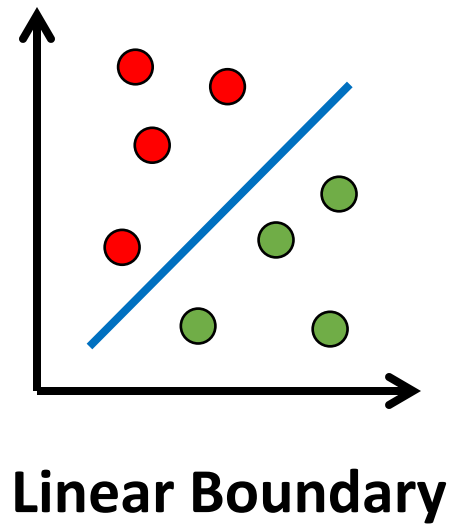
Decision Boundary for Binary Classification



Linear Boundary

Machine Learning

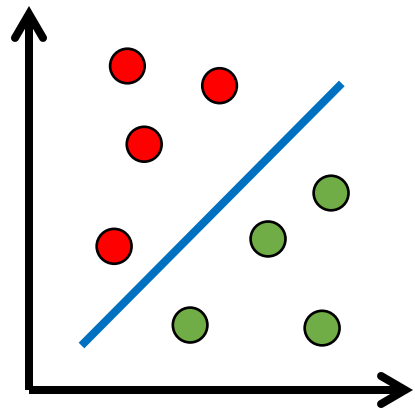
Decision Boundary for Binary Classification



Hazelnuts vs Almonds

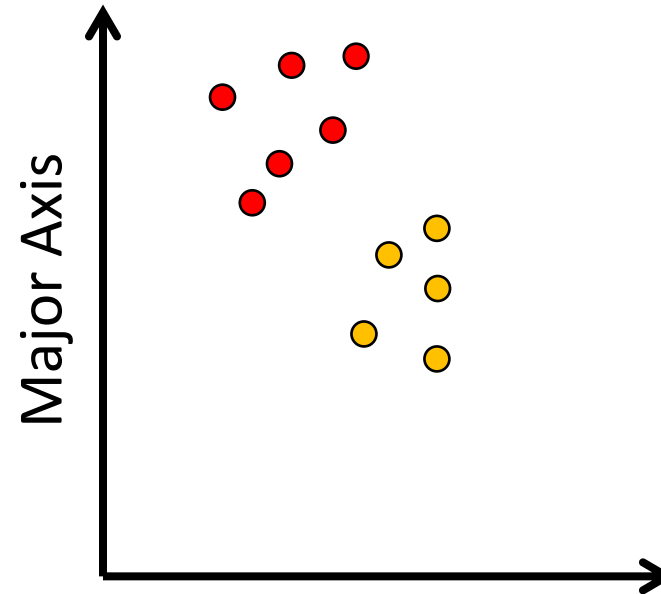
Machine Learning

Decision Boundary for Binary Classification



Linear Boundary

Linear Support Vector
Machine

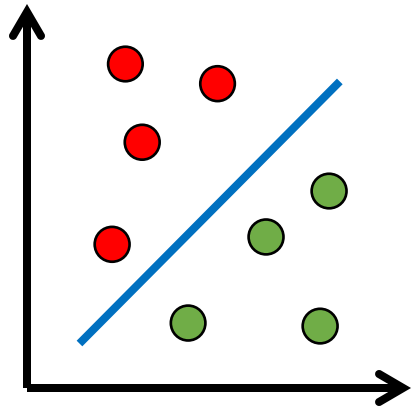


Minor Axis

Hazelnuts vs Almonds

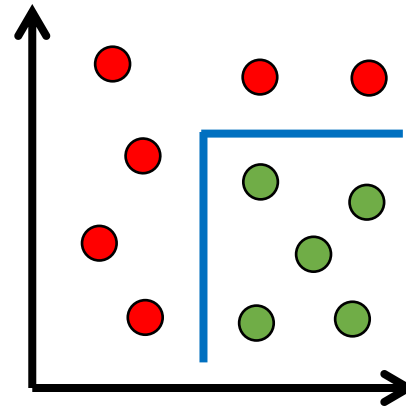
Machine Learning

Decision Boundary for Binary Classification



Linear Boundary

Linear Support Vector
Machine

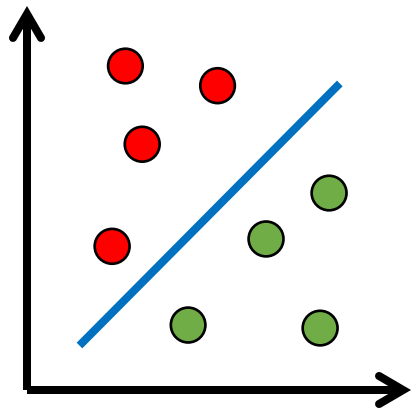


**Axis Aligned Non-Linear
Boundary**

Decision Tree

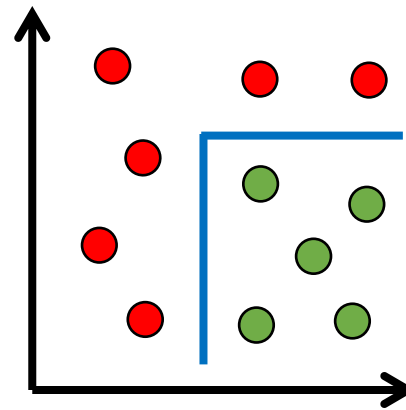
Machine Learning

Decision Boundary for Binary Classification



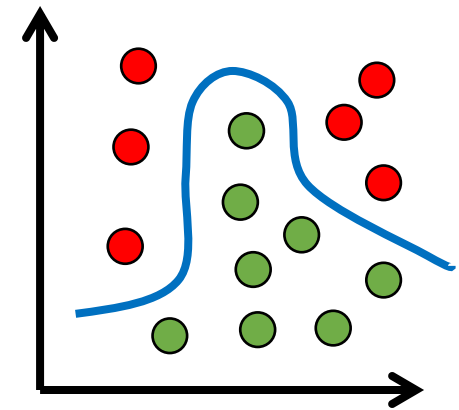
Linear Boundary

Linear Support Vector
Machine



**Axis Aligned Non-Linear
Boundary**

Decision Tree



**Complex Non-Linear
Boundary**

Neural Network

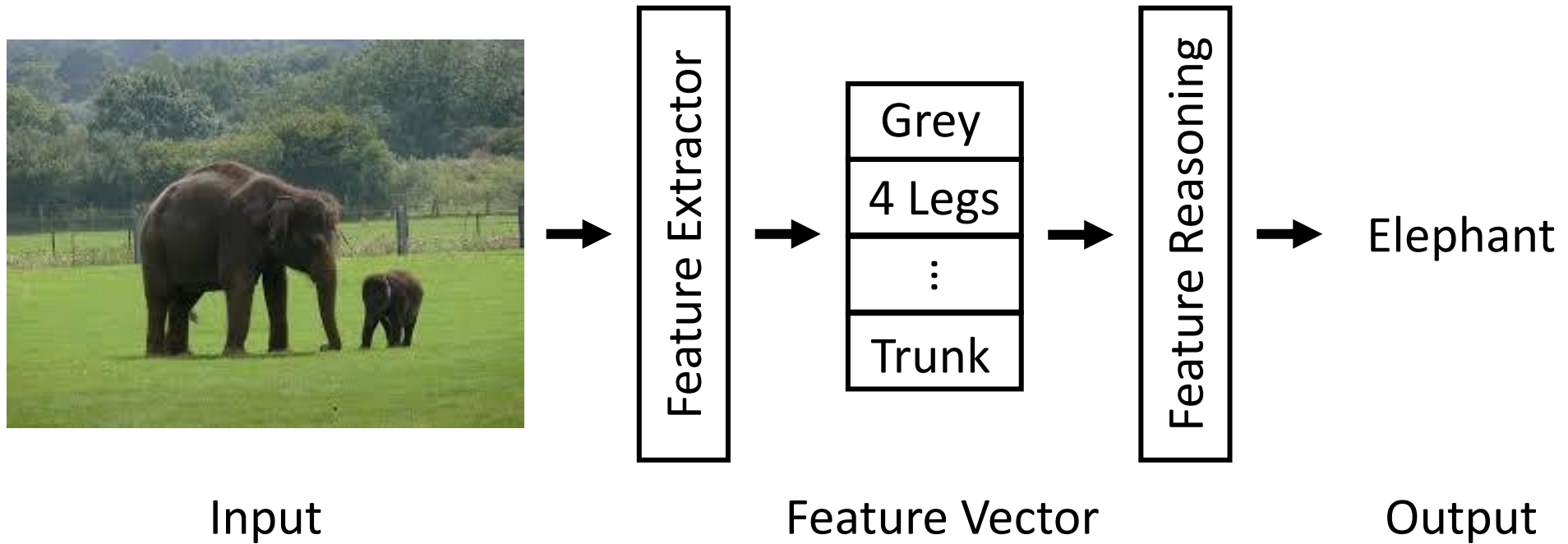
When Should You Use Deep Learning?

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.

When Should You Use Deep Learning?

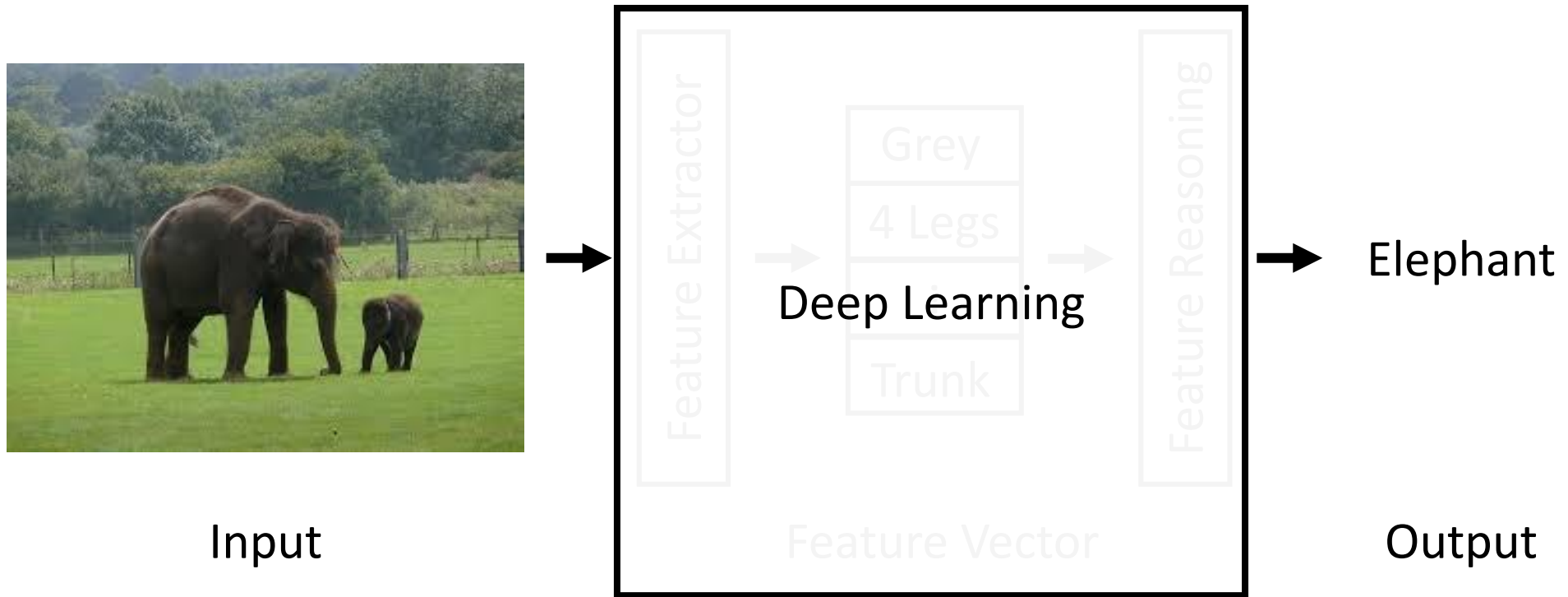
Classical Machine Learning	Deep Learning
K-Means Clustering Linear Regression Decision Trees Random Forest	Fully Connected Network Convolutional Neural Network

When Should You Use Deep Learning?



What animal is in this image?

When Should You Use Deep Learning?



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

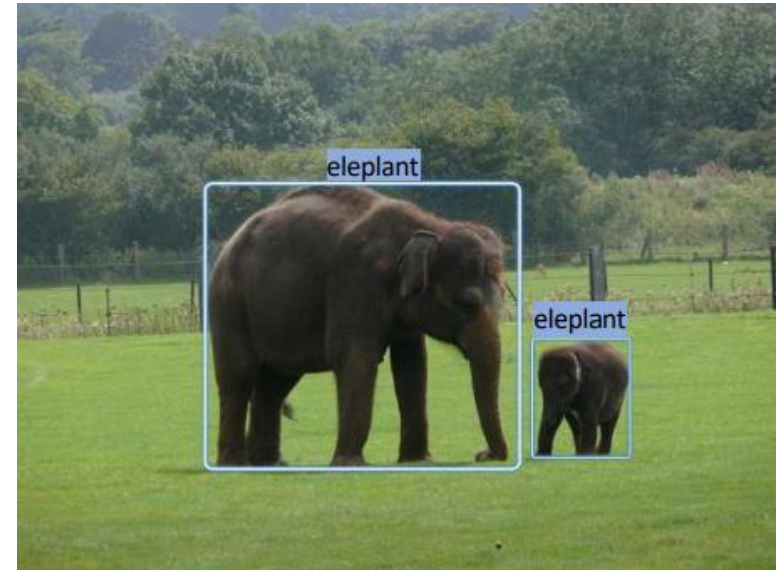
- Binary Classification (binary label):
“Is there an elephant in this image?”
- Classification (integer label):
“What animal is in this image?”
- Regression (real number):
“How old is the elephant in this image?” 10.5 Years



Deep Learning for Computer Vision

Deep Learning Tasks

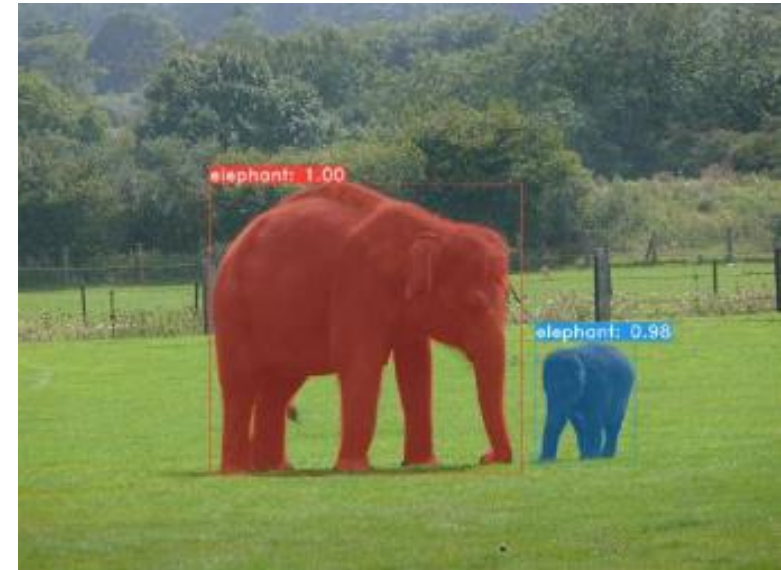
- Binary Classification (binary label):
“Is there an elephant in this image?”
- Classification (integer label):
“What animal is in this image?”
- Regression (real number):
“How old is the elephant in this image?”
- Detection (bounding box):
“Where is the elephant in this image?”



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?”



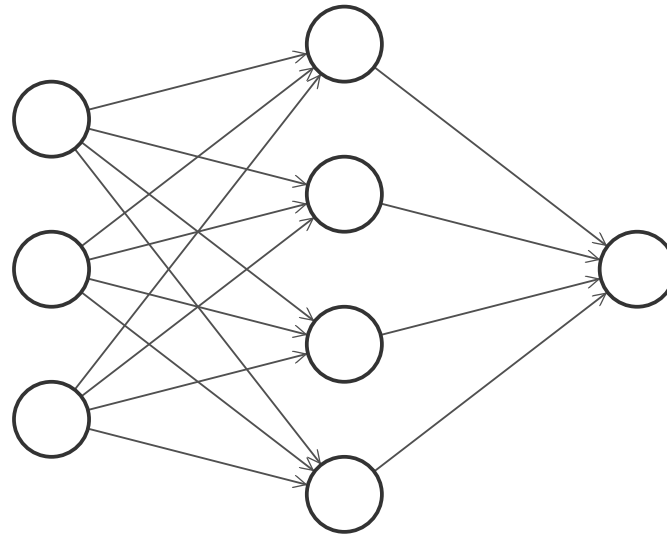
Deep Learning for Computer Vision

We know what specific questions we can ask.

But how do we get a computer to answer them?

Neural Networks

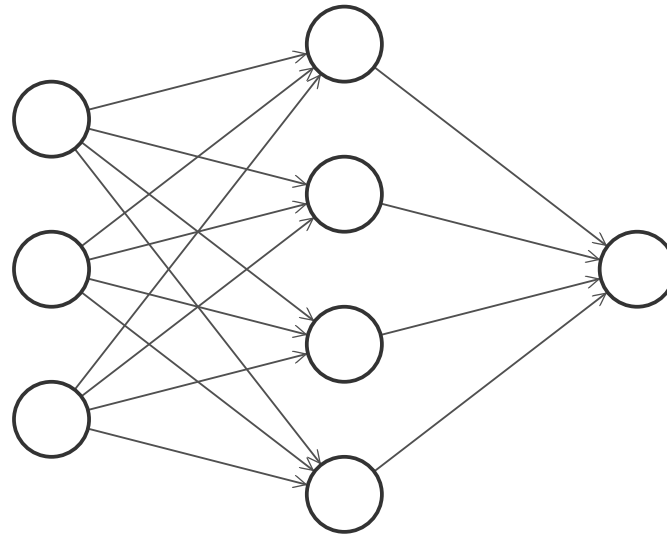
Fully Connected Network (FCN)



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

Neural Networks

Fully Connected Network (FCN)

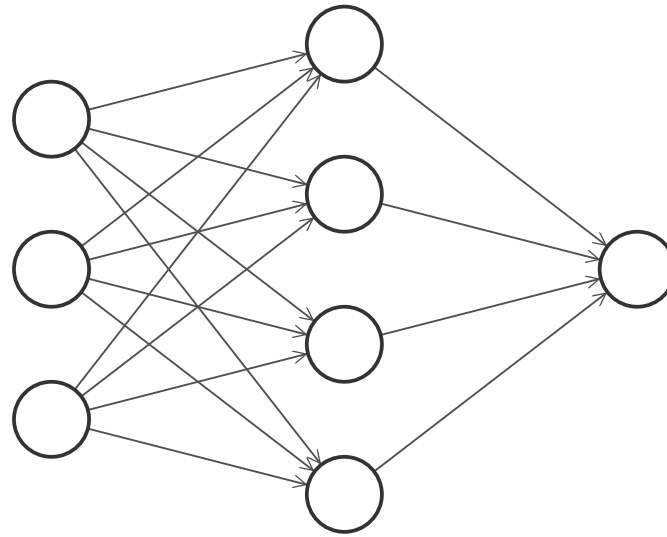


input layer

input your data here

Neural Networks

Fully Connected Network (FCN)

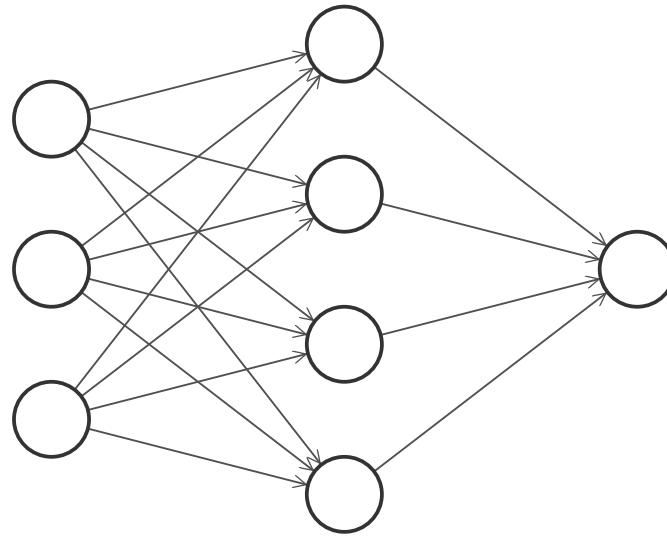


input layer hidden layer

extract some features and
do some reasoning

Neural Networks

Fully Connected Network (FCN)



input layer

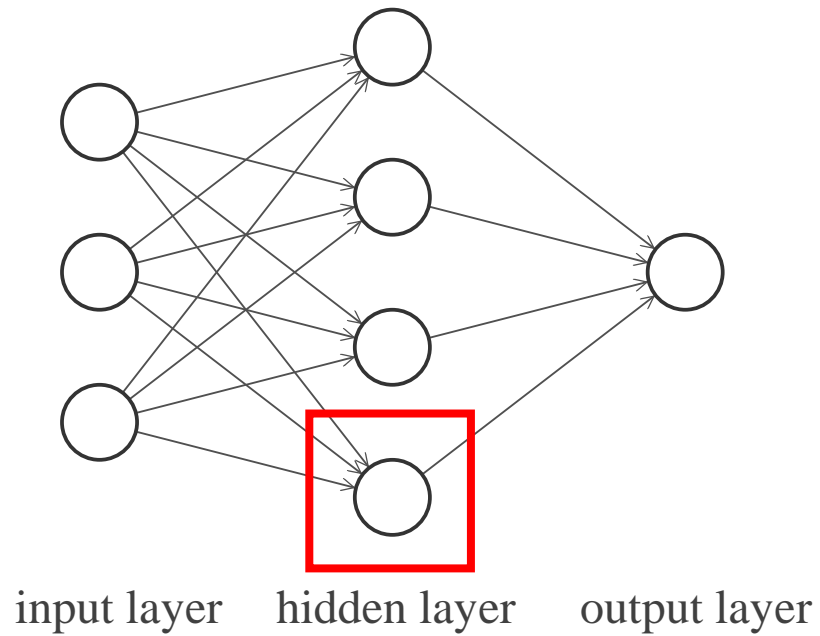
hidden layer

output layer

make a decision

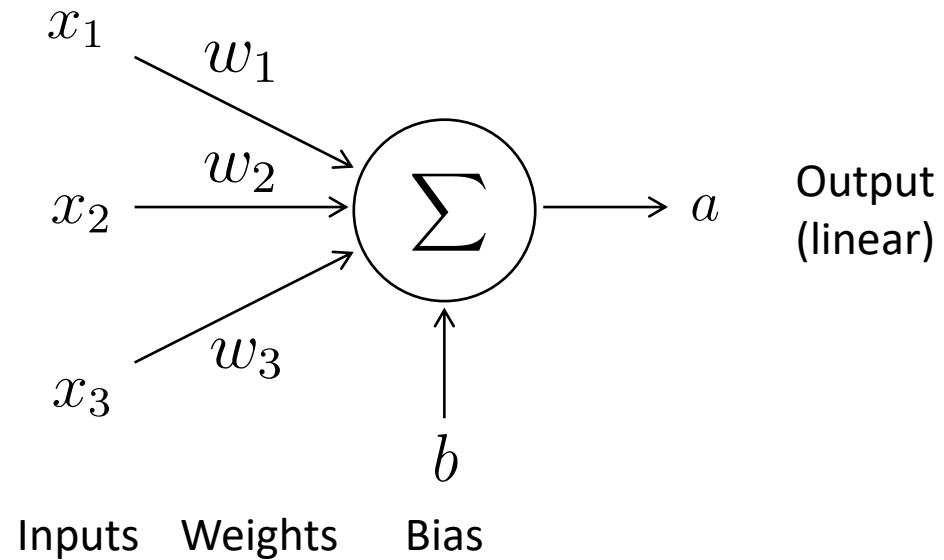
Neural Networks

Fully Connected Network (FCN)



Neural Networks

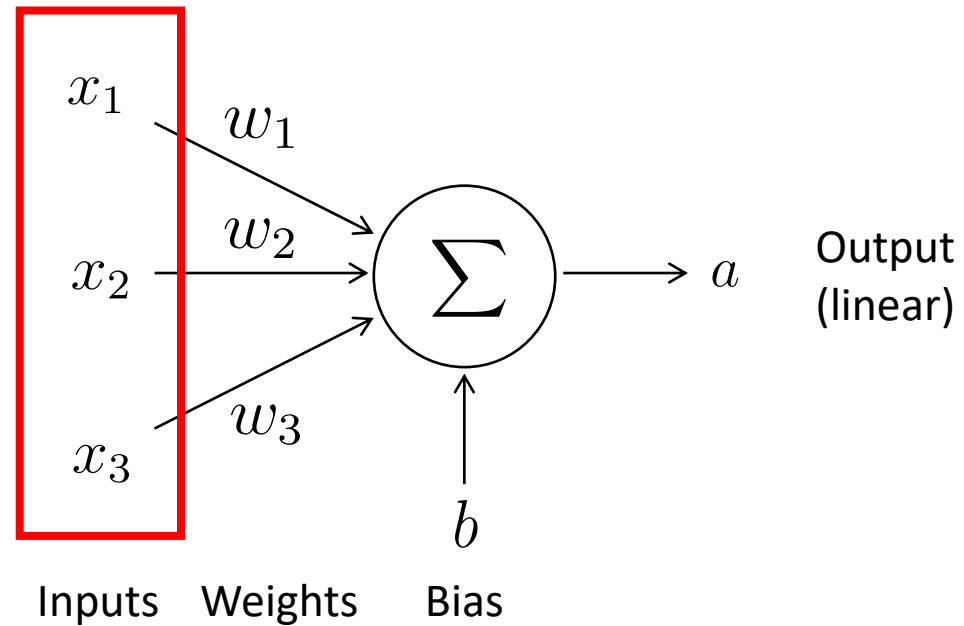
Building Blocks: Perceptron



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

Neural Networks

Building Blocks: Perceptron

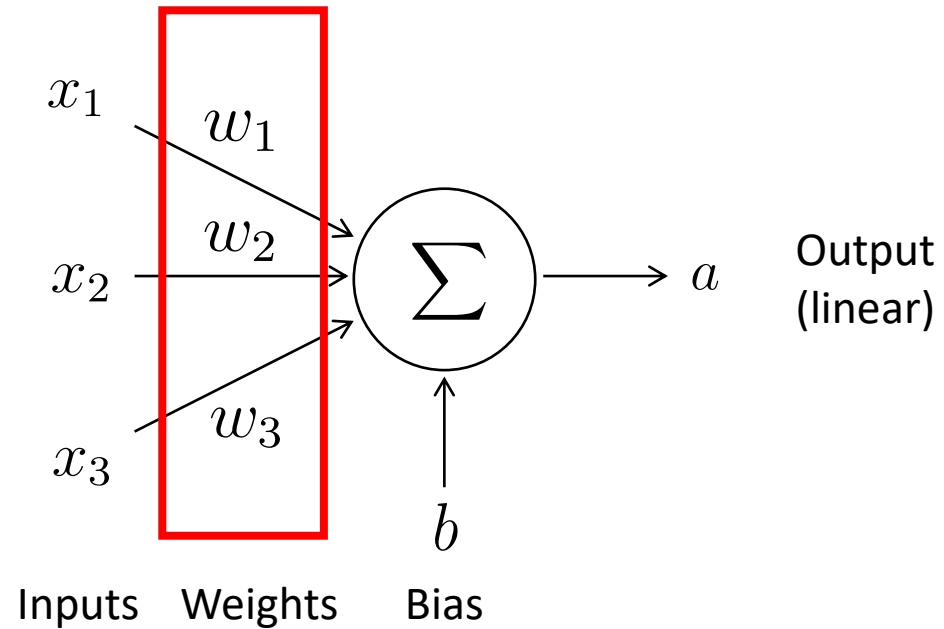


$$a = \sum_i w_i x_i + b$$

A red arrow points from the summation symbol \sum in the equation above to the $w_i x_i$ term.

Neural Networks

Building Blocks: Perceptron

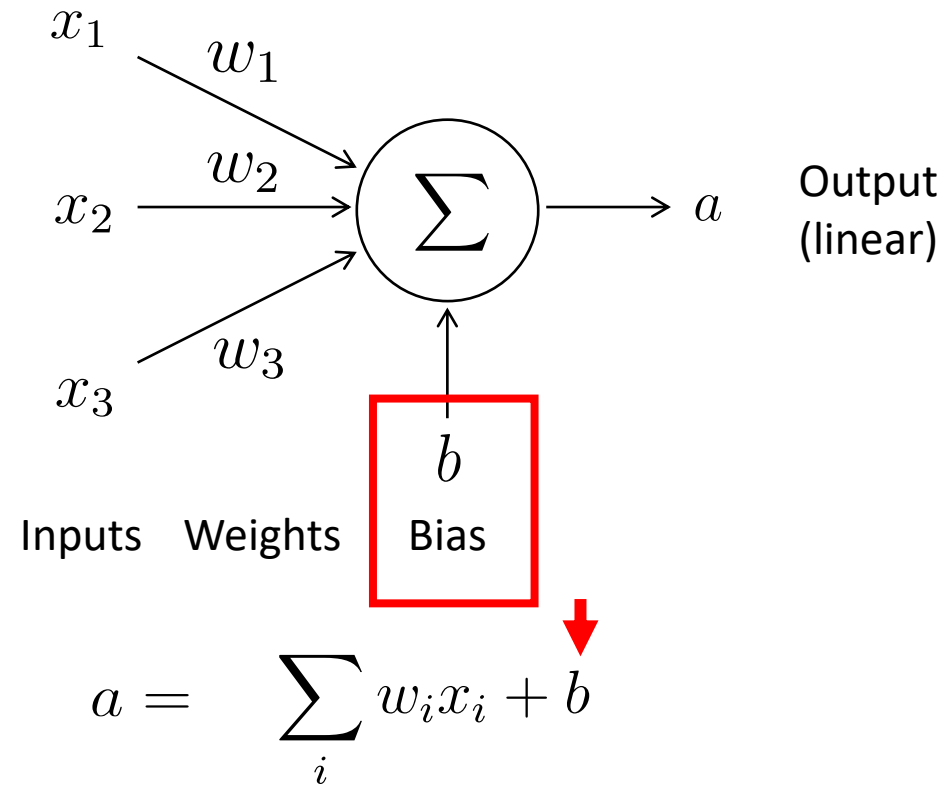


$$a = \sum_i w_i x_i + b$$

A red arrow points down to the summation symbol \sum_i in the equation above.

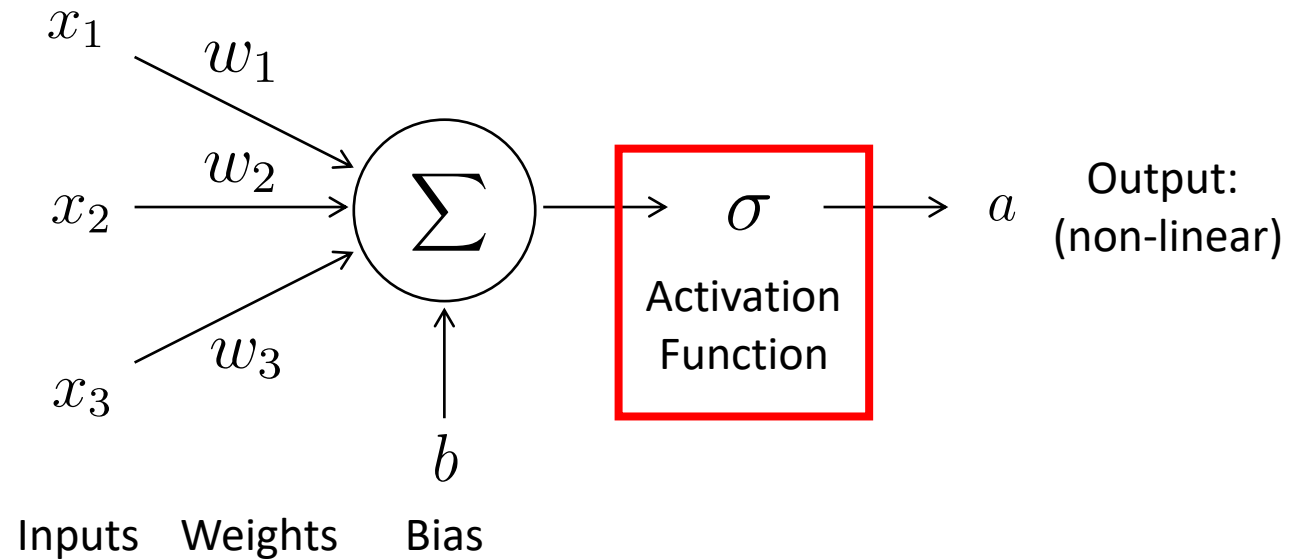
Neural Networks

Building Blocks: Perceptron



Neural Networks

Building Blocks: Perceptron

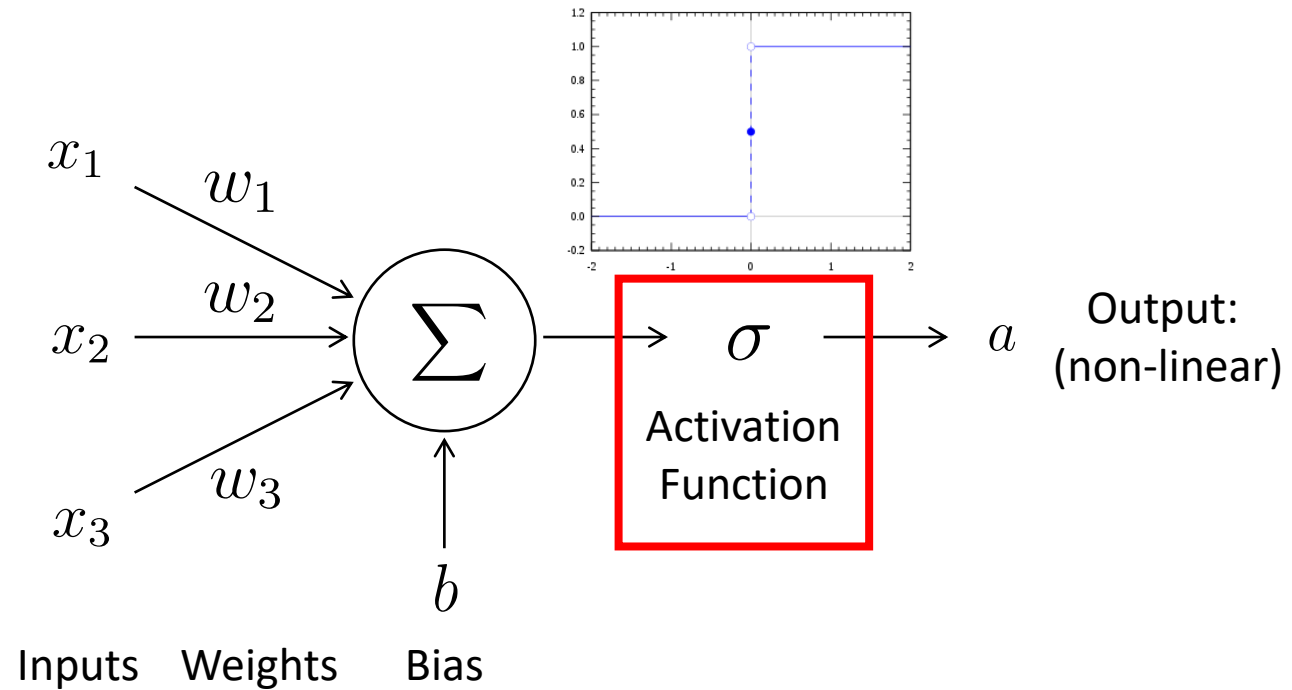


↓

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

Neural Networks

Building Blocks: Perceptron



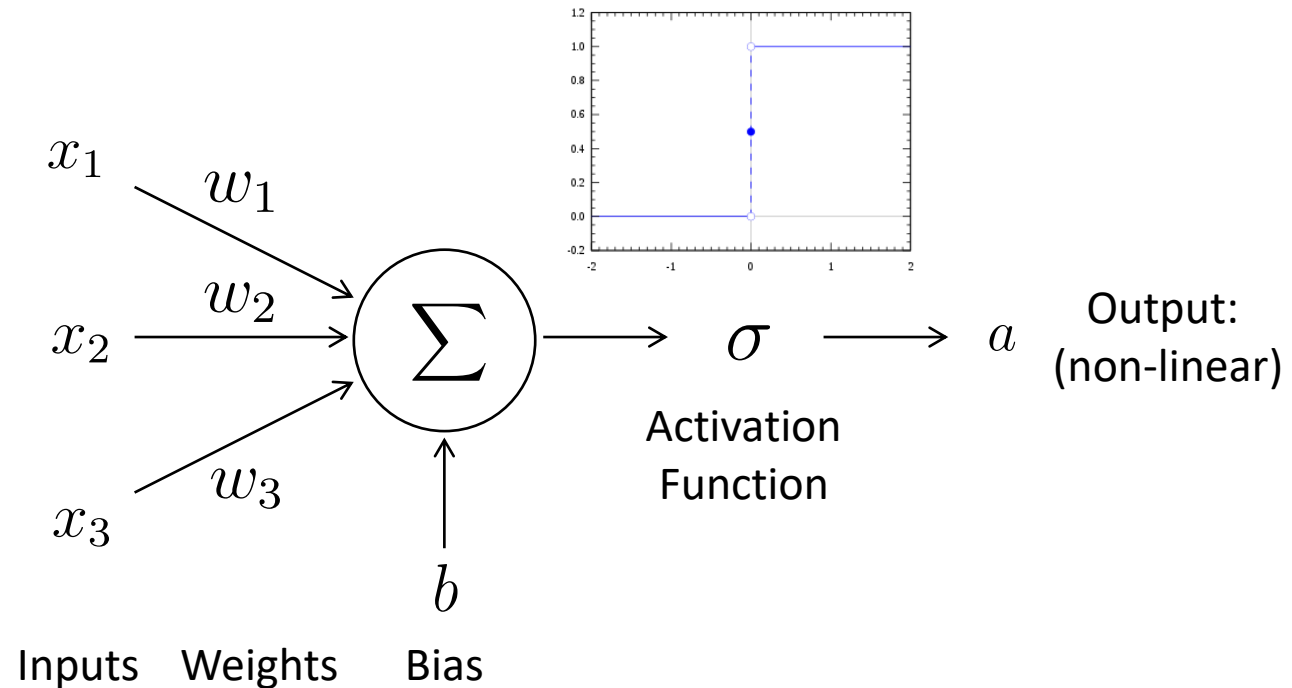
↓

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

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.



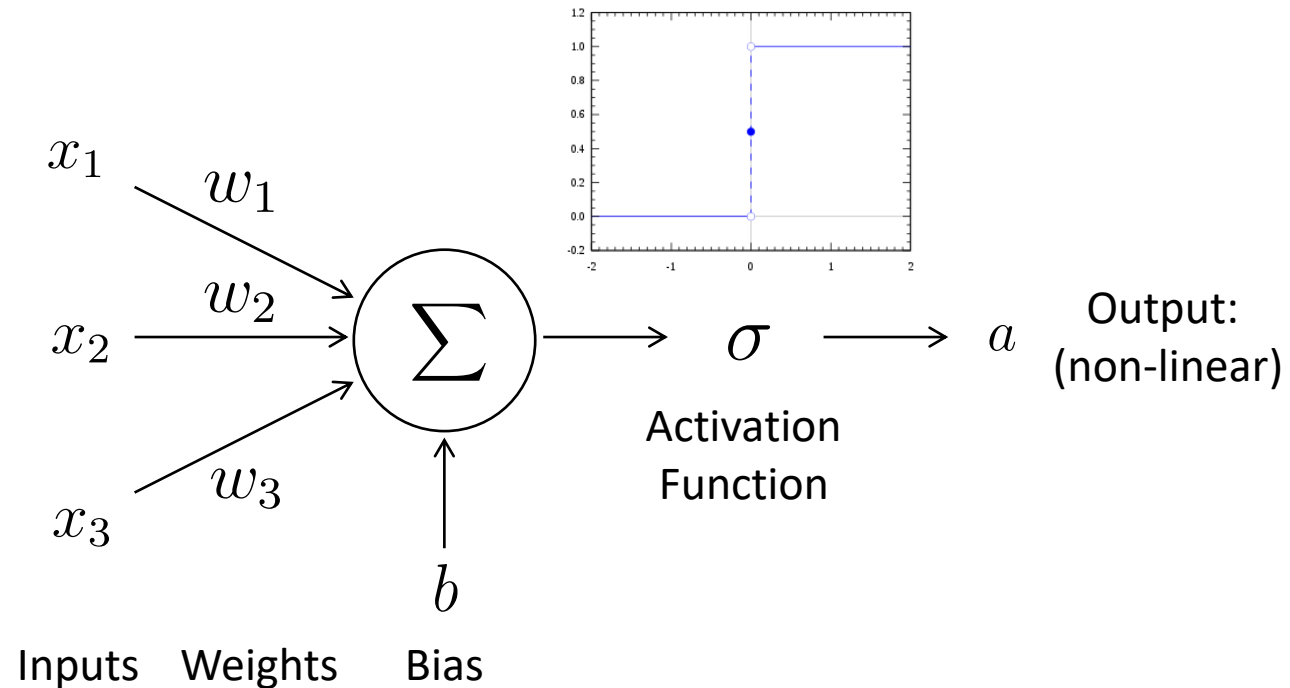
$$a = \sigma\left(\sum_i w_i x_i + b\right)$$

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.



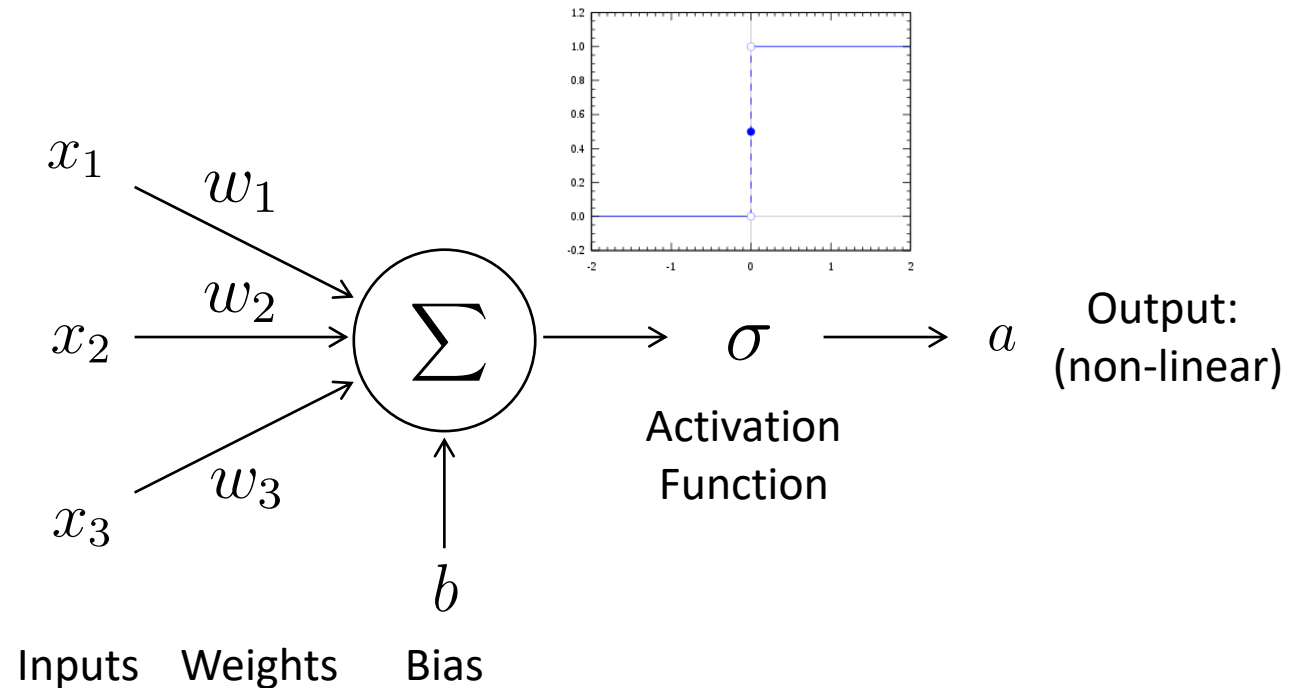
$$a = \sigma\left(\sum_i w_i x_i + b\right)$$

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.



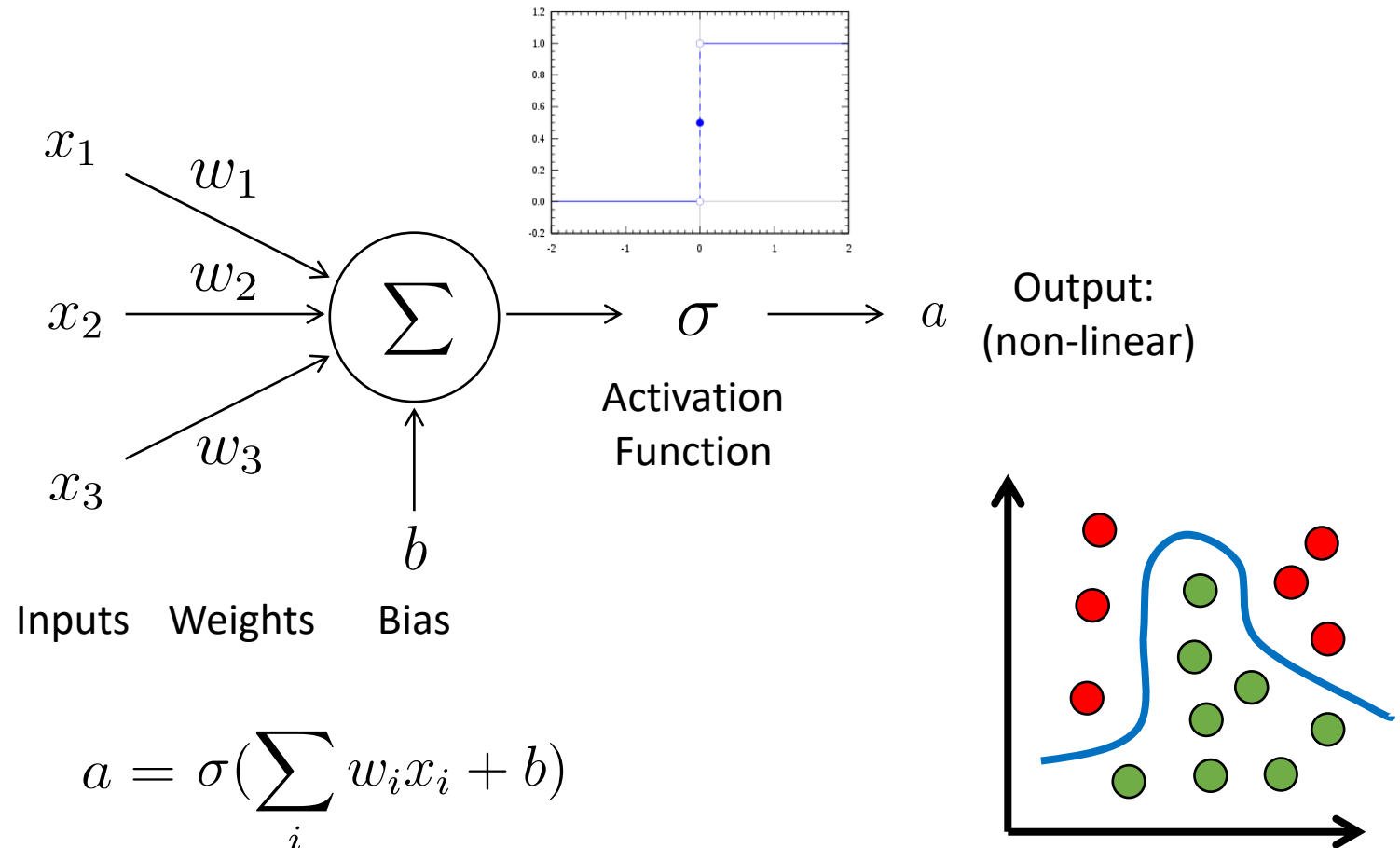
$$a = \sigma\left(\sum_i w_i x_i + b\right)$$

Neural Networks

Building Blocks: Perceptron

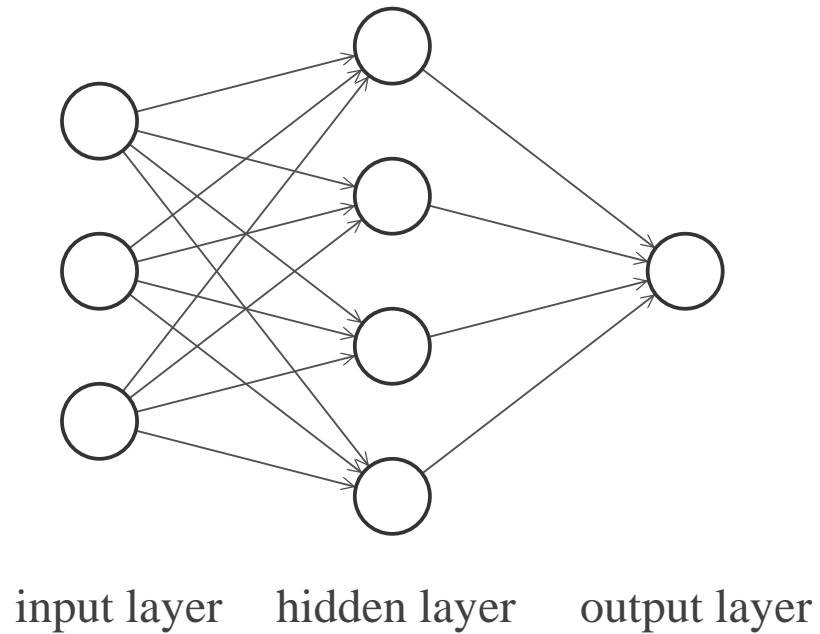
Neurons (or Nodes) are based on biological neurons that “fire” if the linear output is high. The “firing” is modeled by the activation function.

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



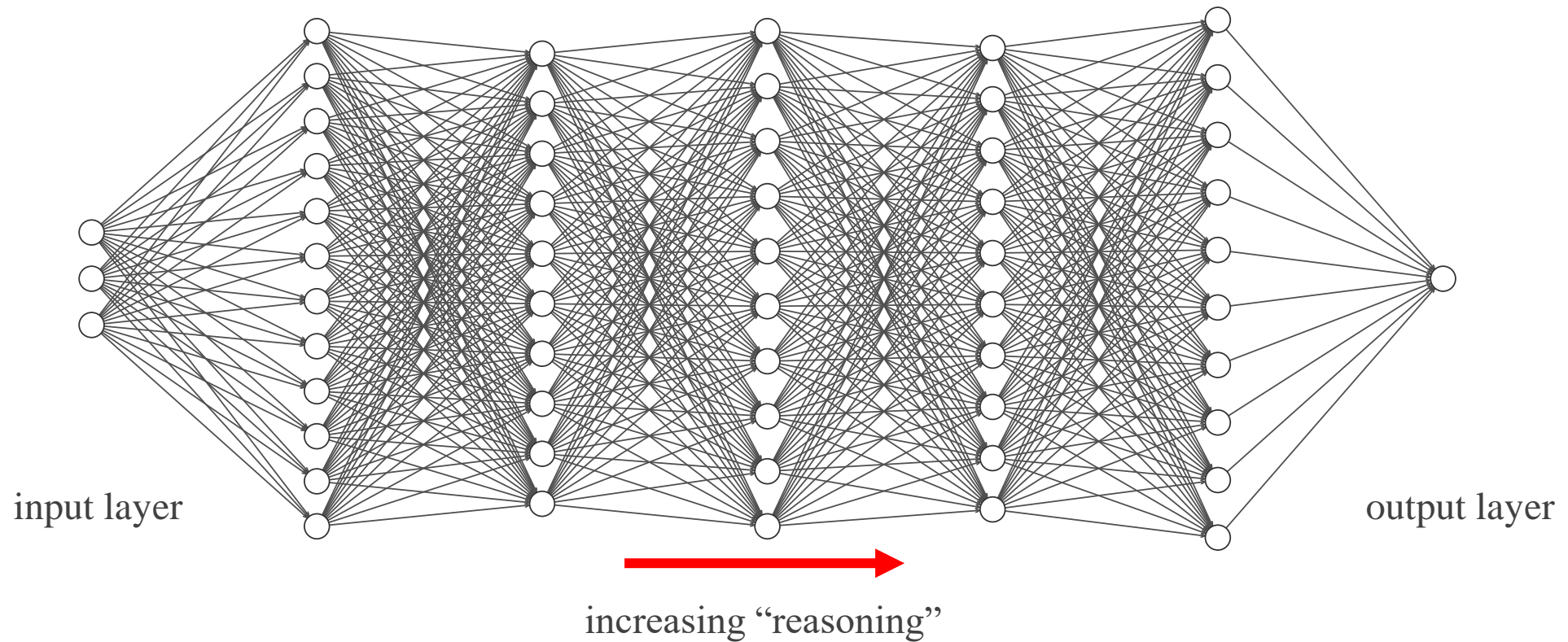
Neural Networks

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



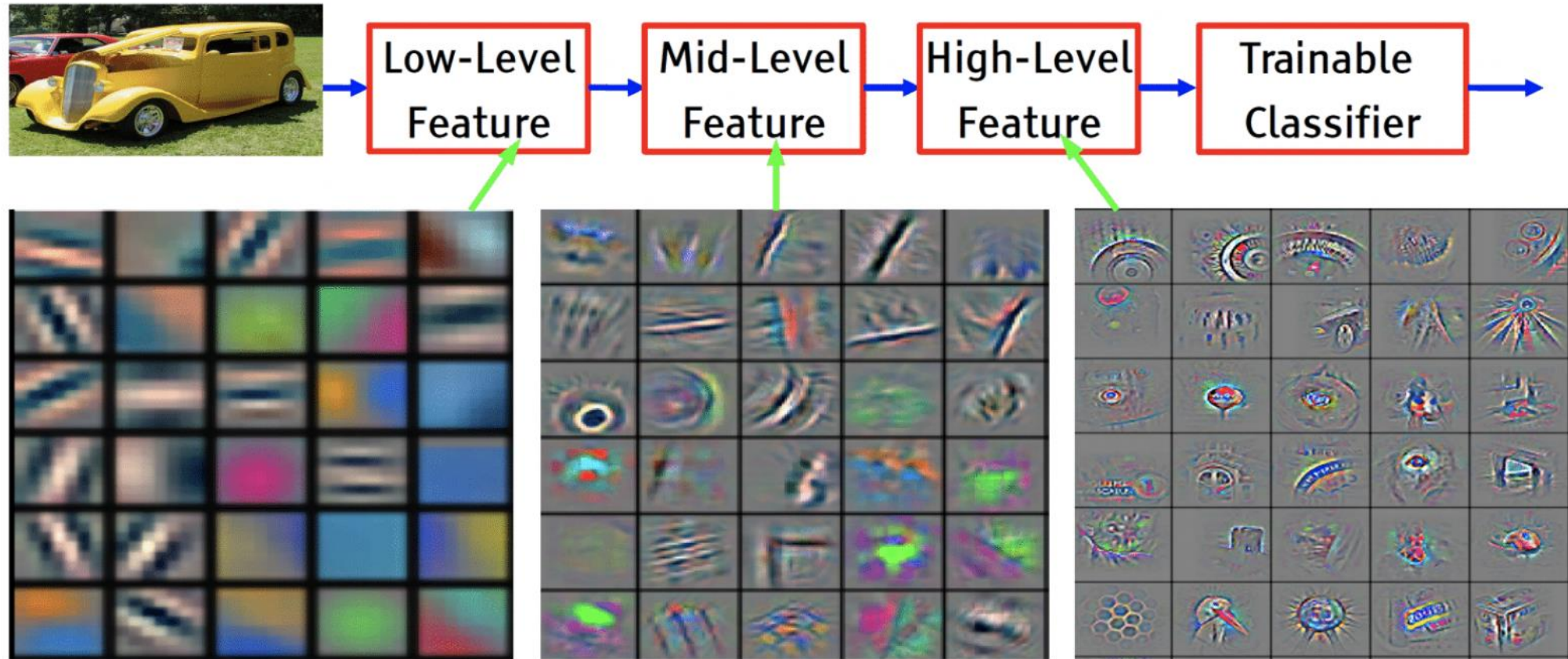
Neural Networks

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



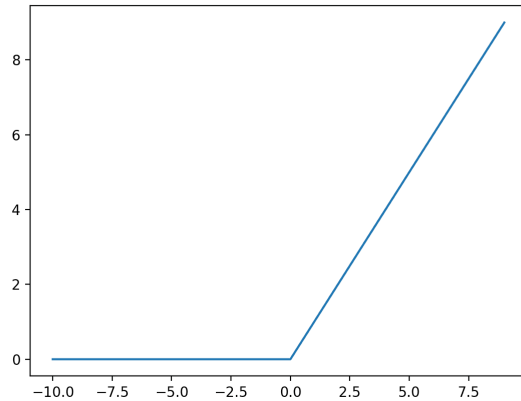
Neural Networks

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



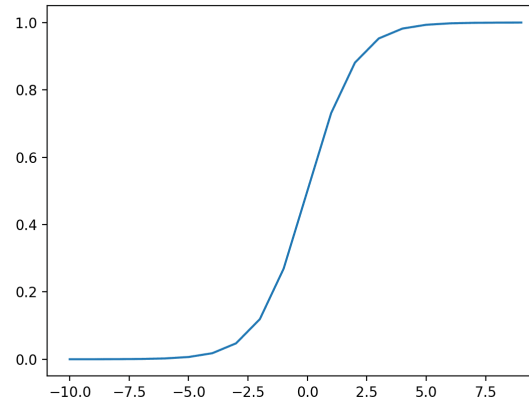
Neural Networks

Building Blocks: Activation Functions



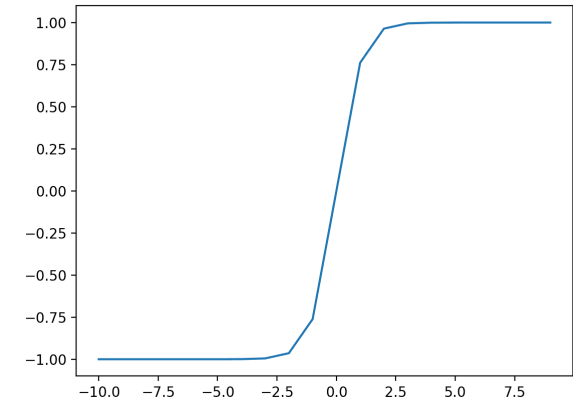
Rectified Linear Unit (ReLU)

Good general purpose fn,
overcomes vanishing
gradient issues



Sigmoid

Used for RNNs and
binary classification
output



Hyperbolic Tangent

Used for RNNs

<https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c>

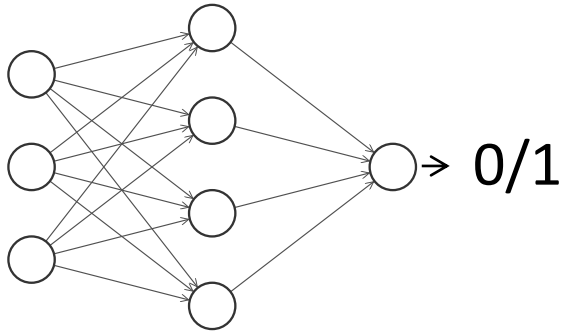
<https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/>

Neural Networks

Building Blocks: Output Layer

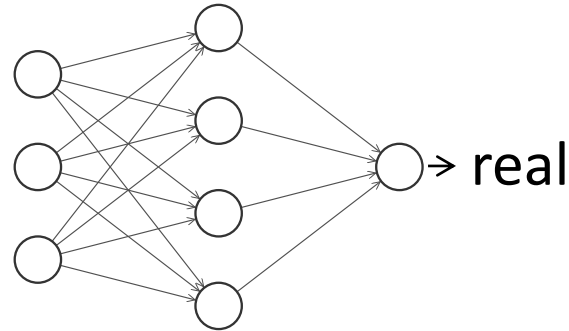
$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

softmax



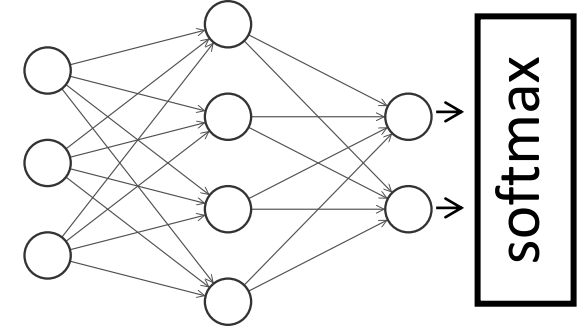
Binary Classification

Single Node
Sigmoid activation



Regression

Single Node
Linear activation



Classification

Multiple Nodes
ReLU activation with Softmax

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.**”

Neural Networks

Building Blocks: Network Size

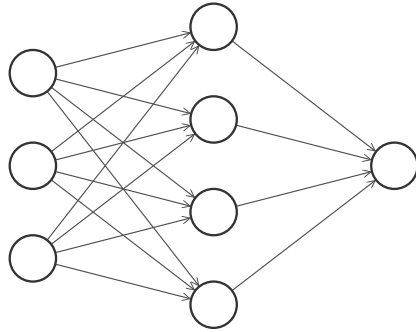
How many nodes/layers should I use?

“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.

Convolutional Neural Network

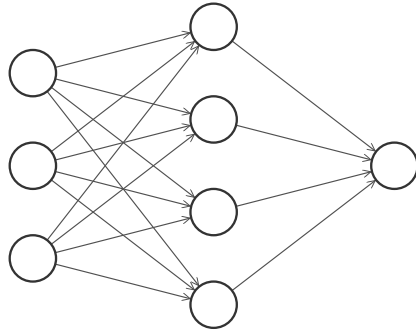
How do we input images to a Neural Network?



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

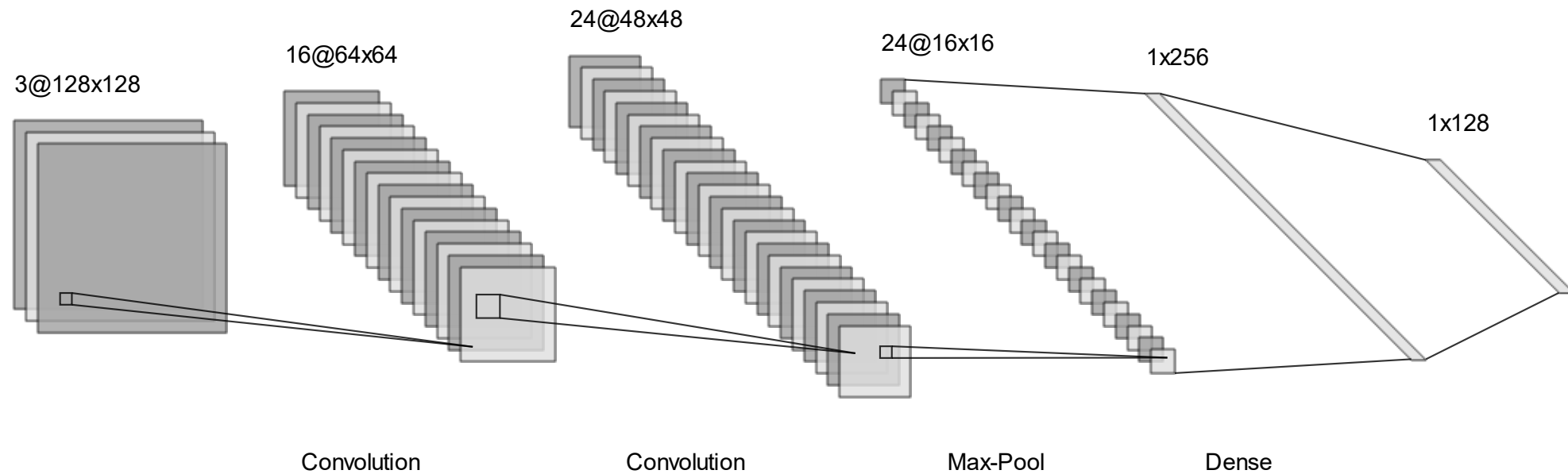
Assume 3 layers with 16,384 nodes each

$(16,384 \times 2 \times 16,384) \times 3 + 2 \times 16,384 = 1.6$ billion

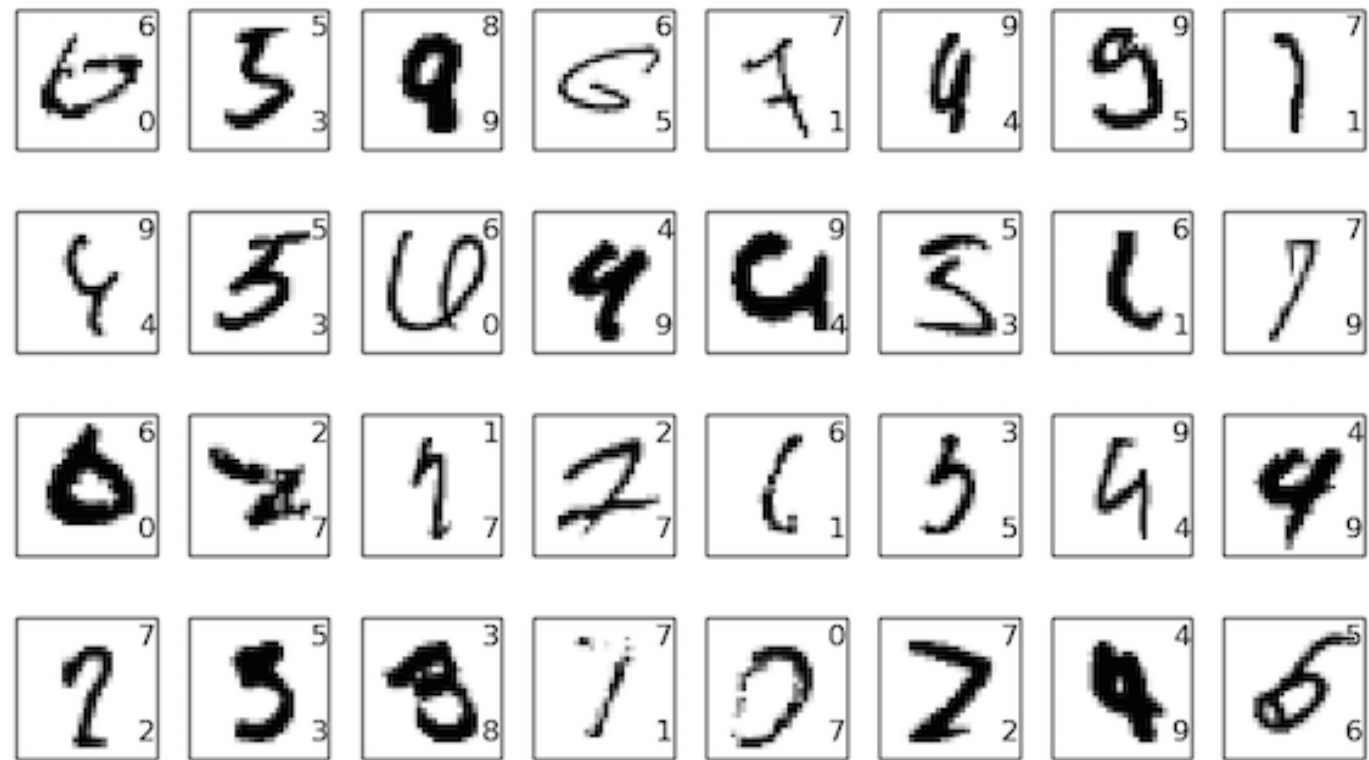
This is not trainable in a reasonable amount of time

Does not model inherent structure of images

Convolutional Neural Network

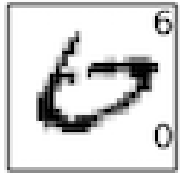


Convolutional Neural Network

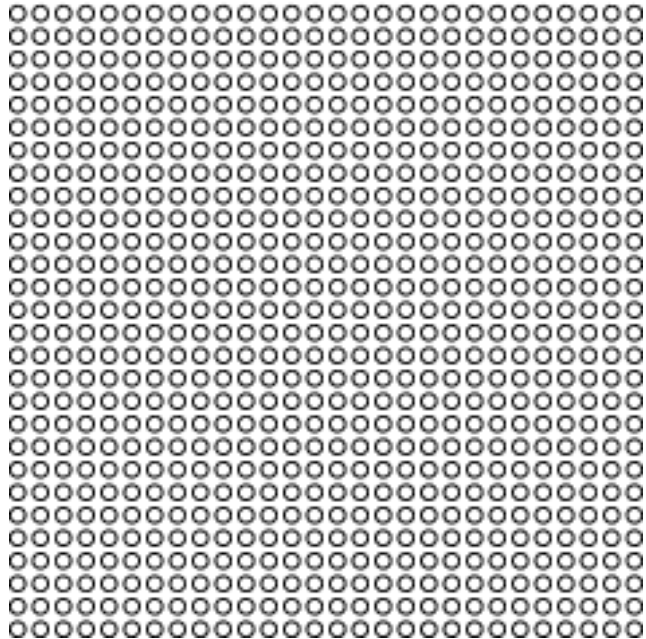


MNIST as an example (28x28px) images

Convolutional Neural Network



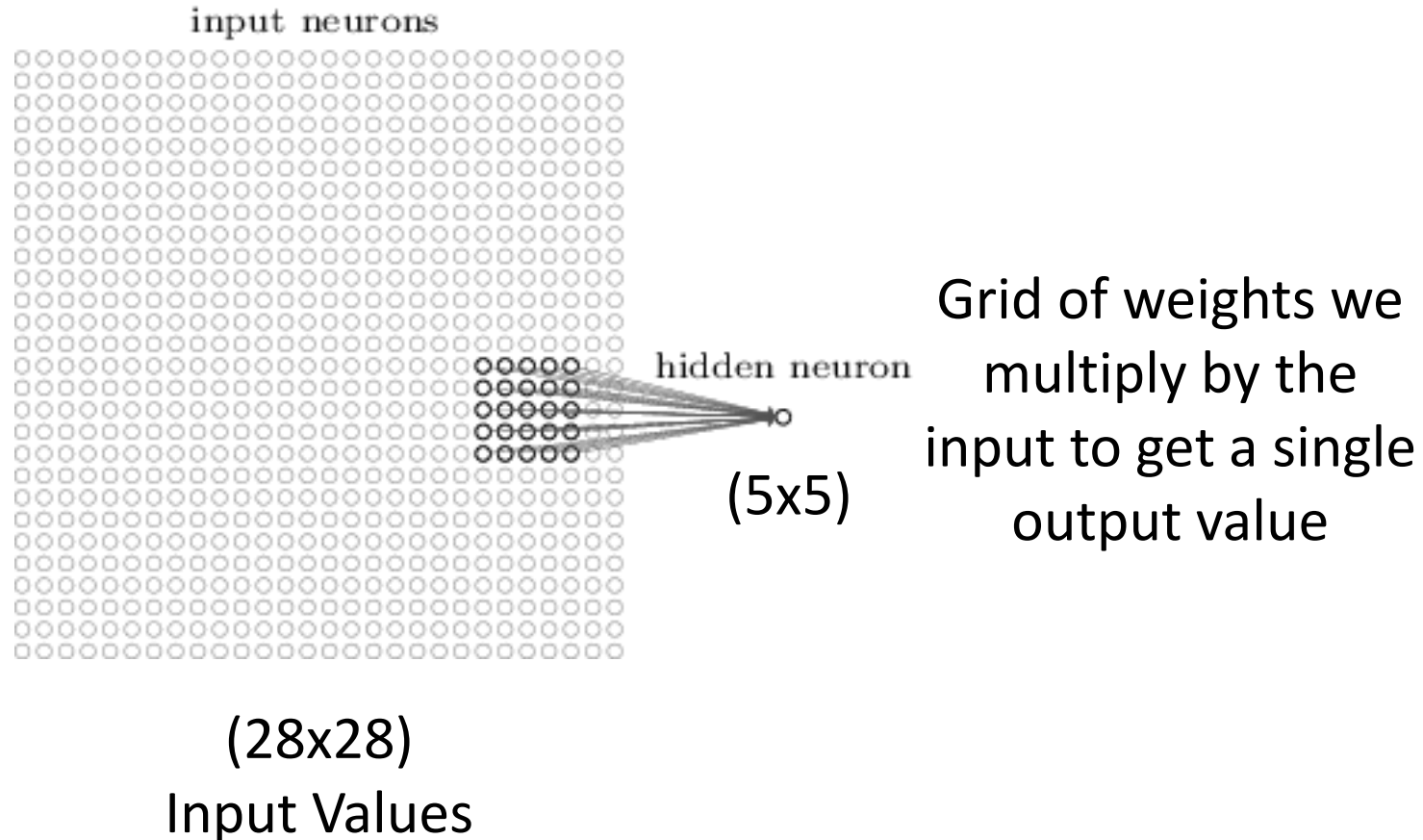
input neurons



(28x28)

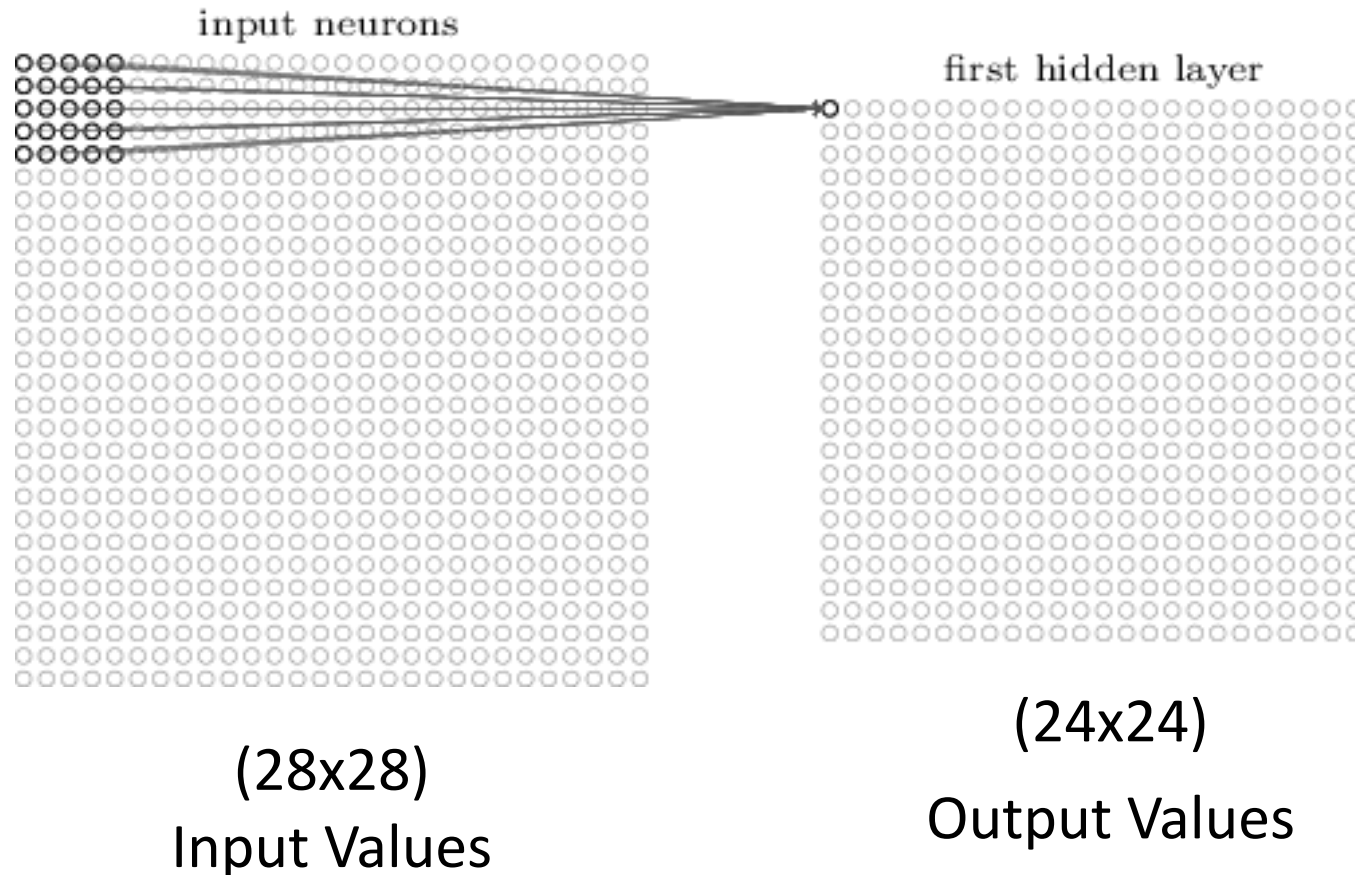
Input Values

Convolutional Neural Network



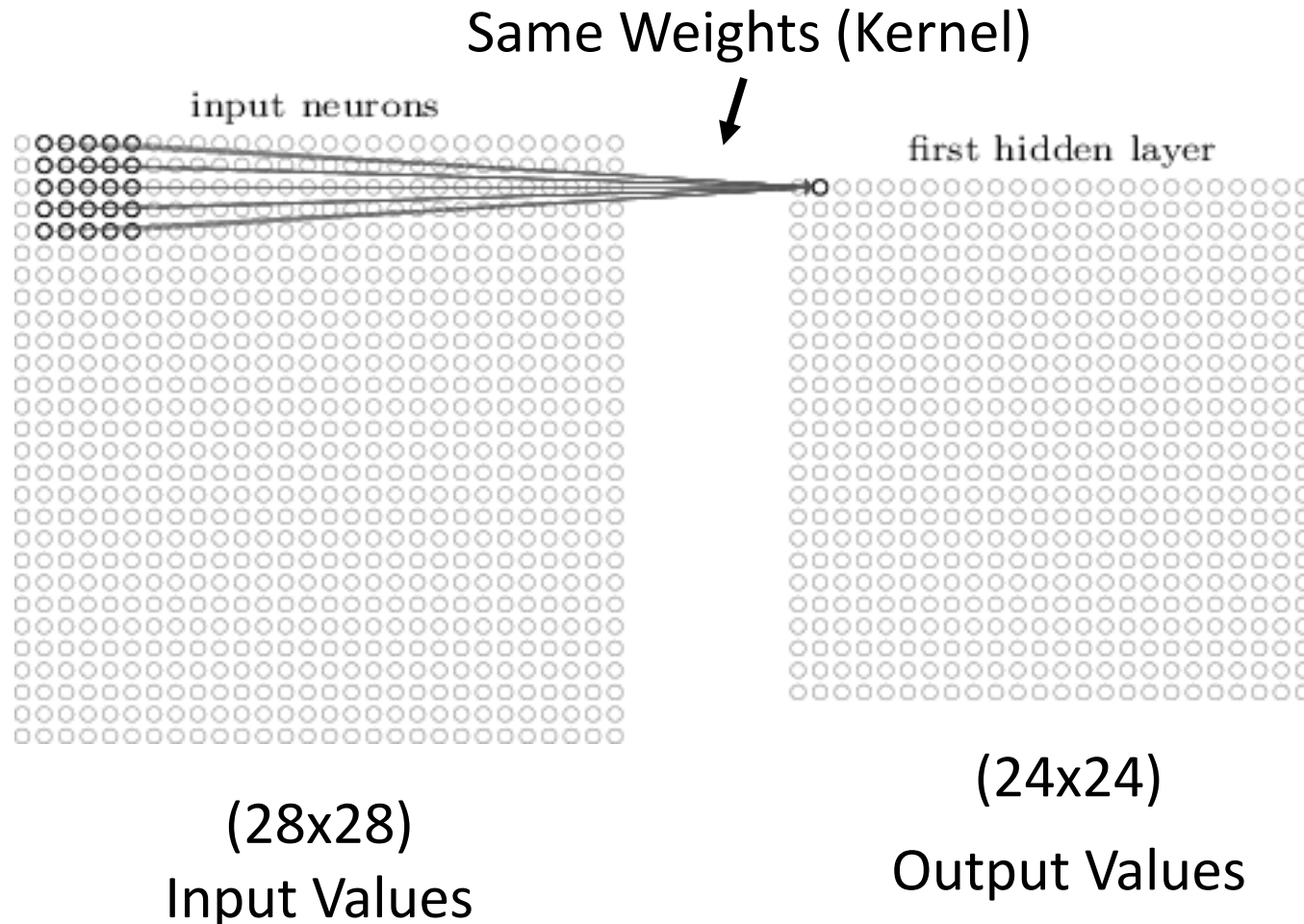
1. have each hidden neuron process a small window of the input, called the “receptive field”.

Convolutional Neural Network



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

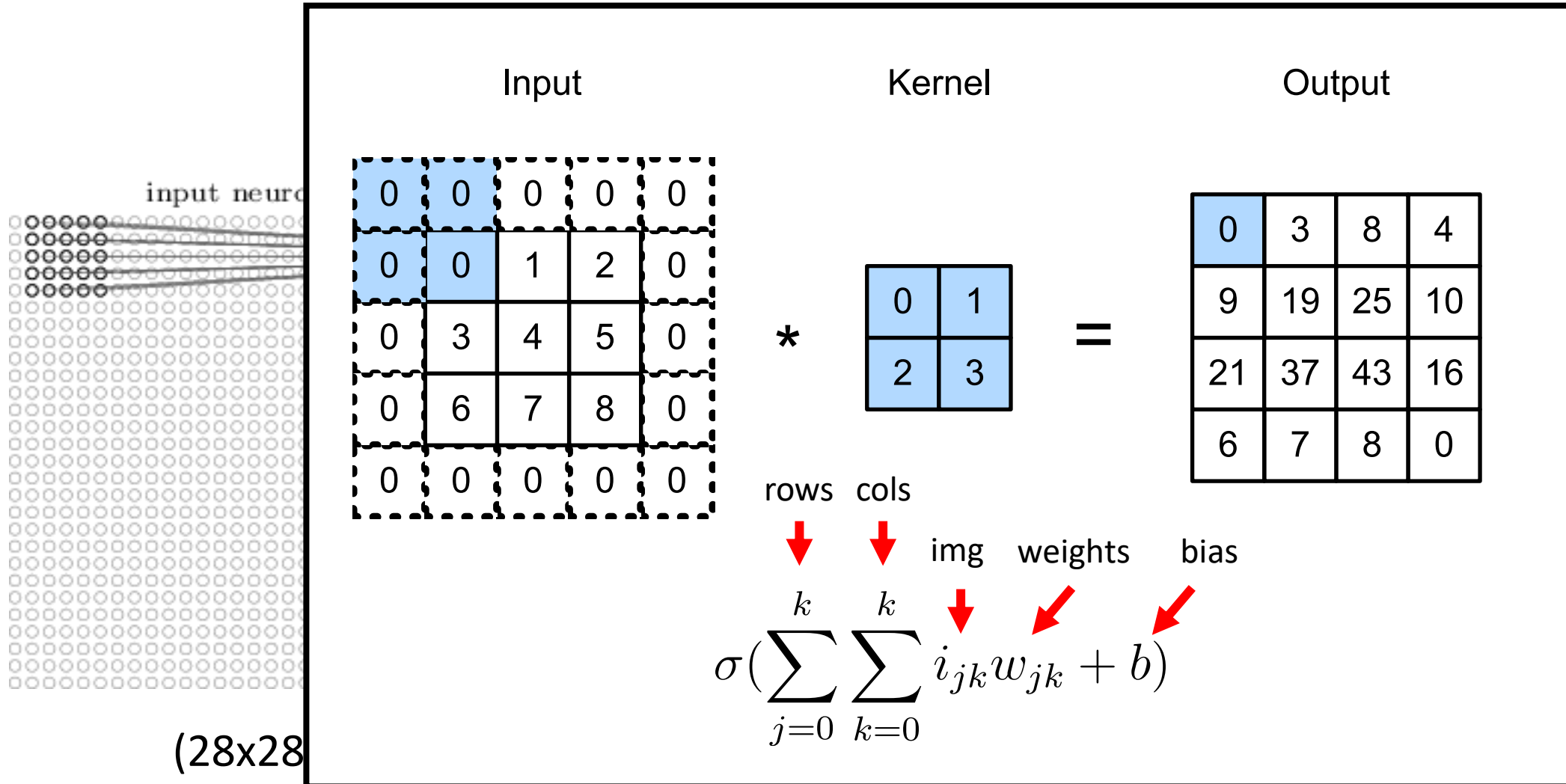
Convolutional Neural Network



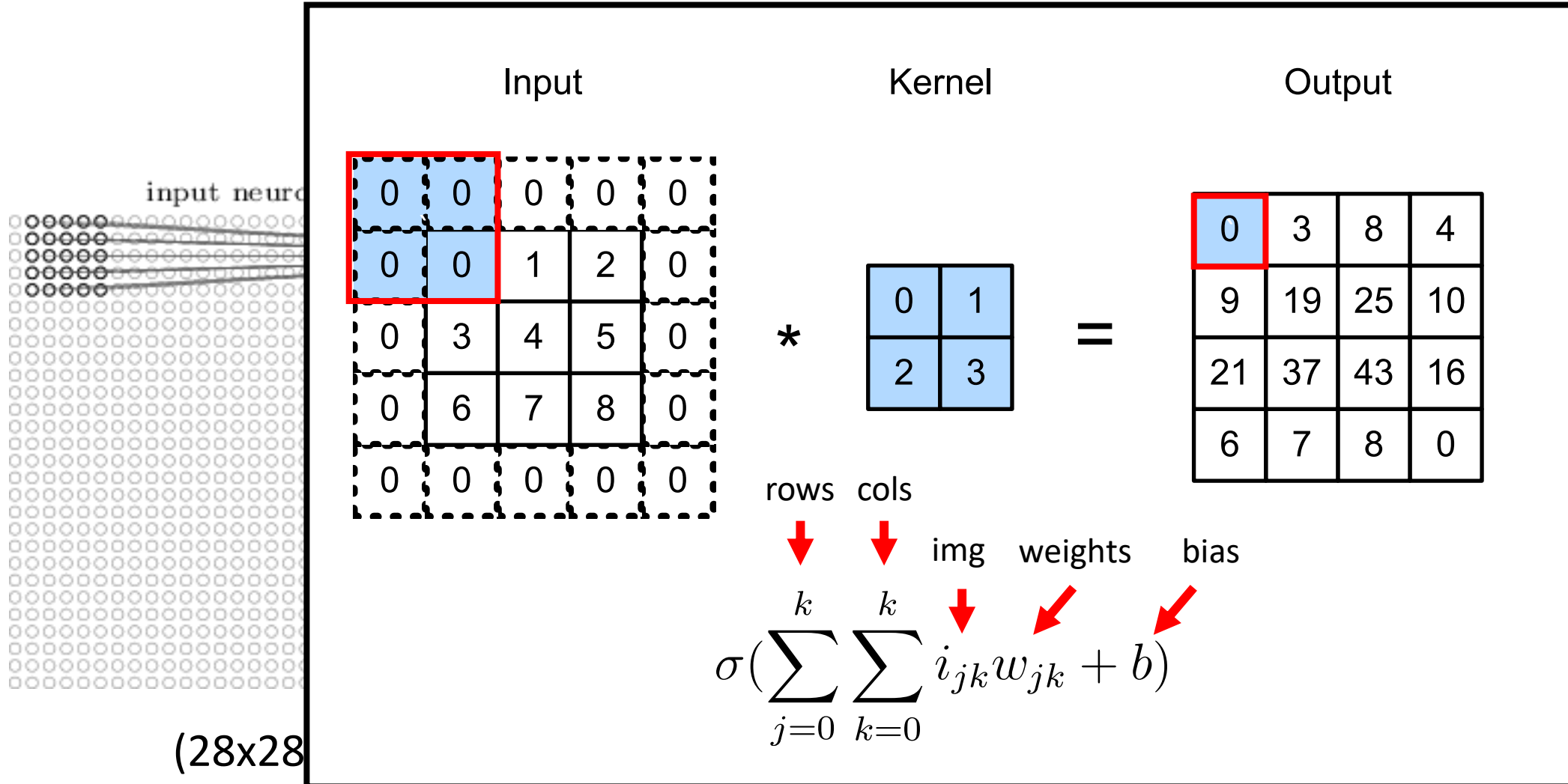
1. have each hidden neuron process a small window of the input, called the “receptive field”.
2. Slide the receptive field across the image.

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

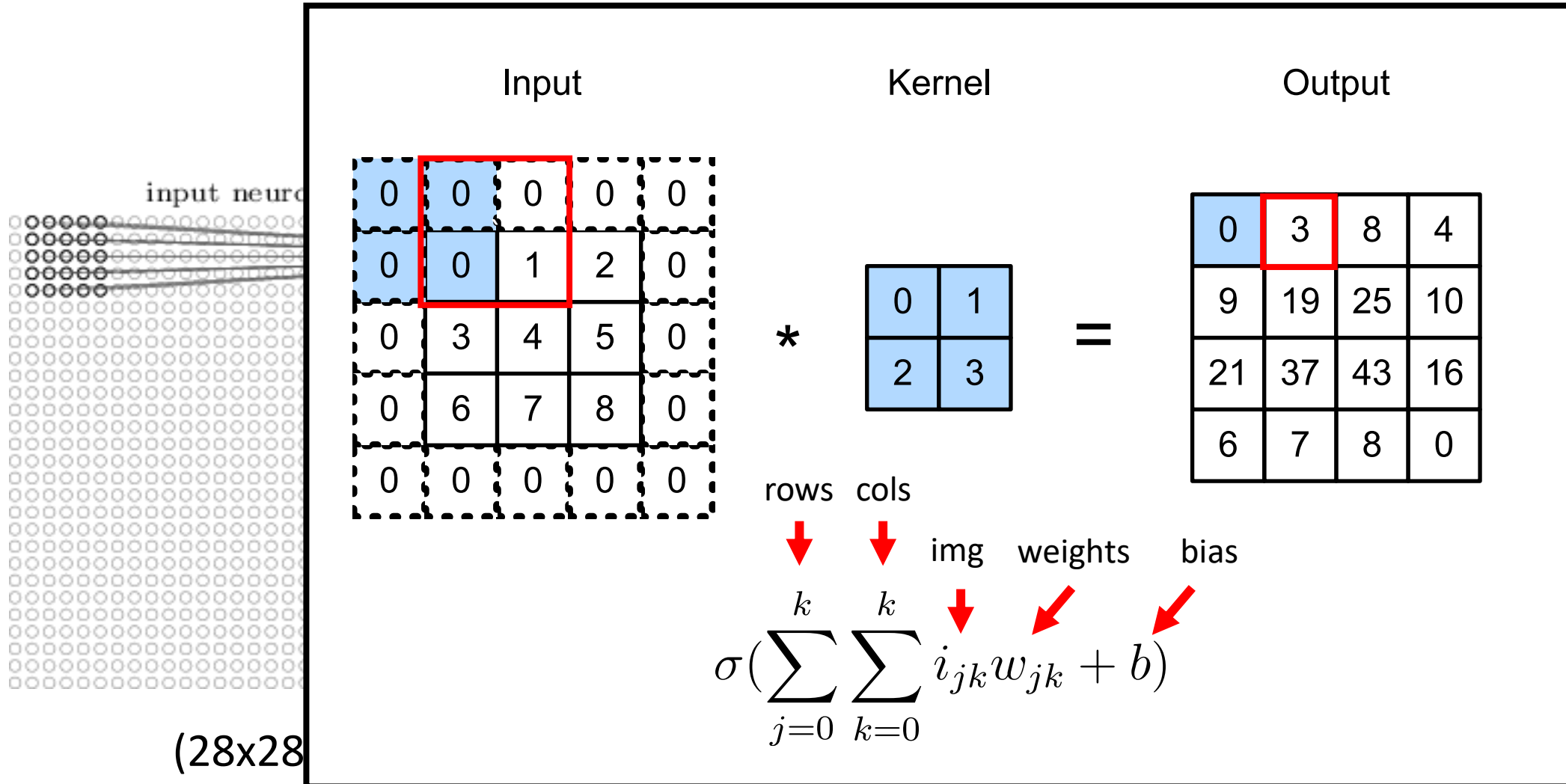
Convolutional Neural Network



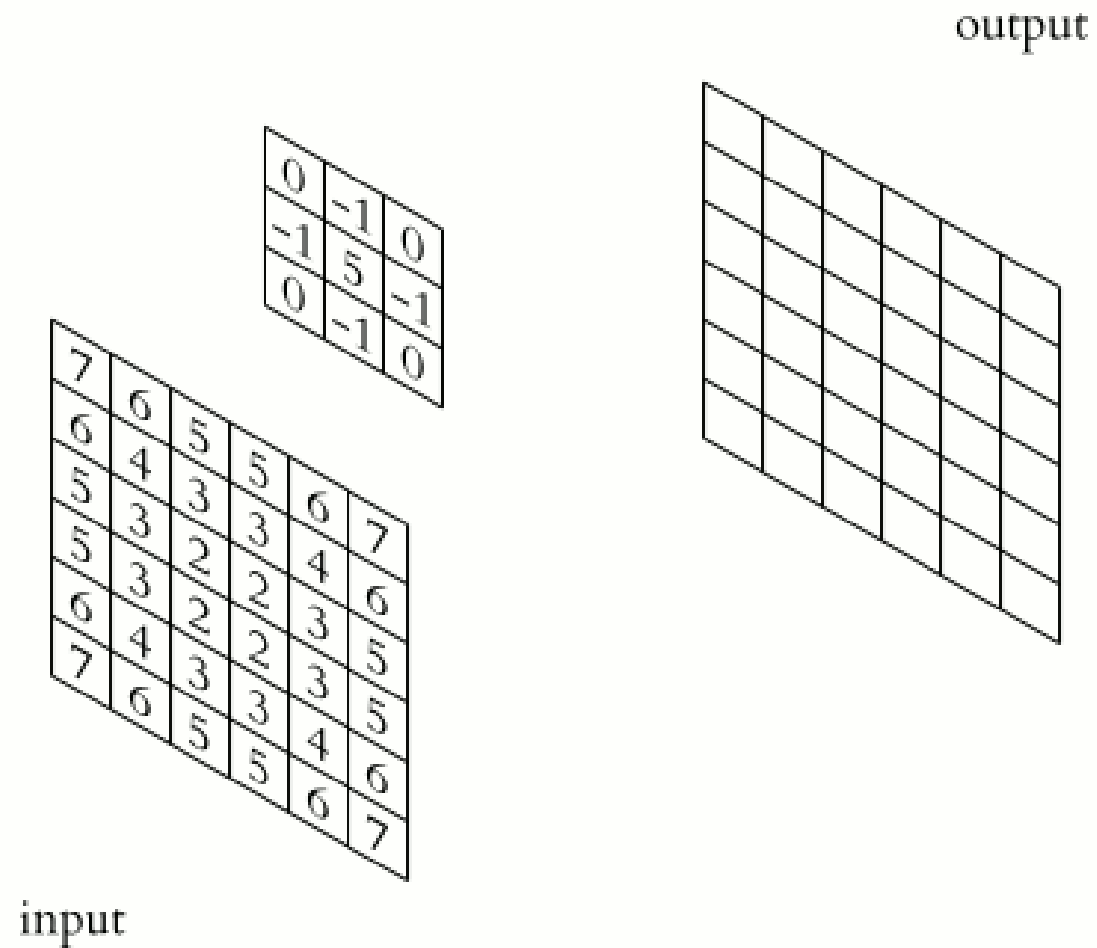
Convolutional Neural Network



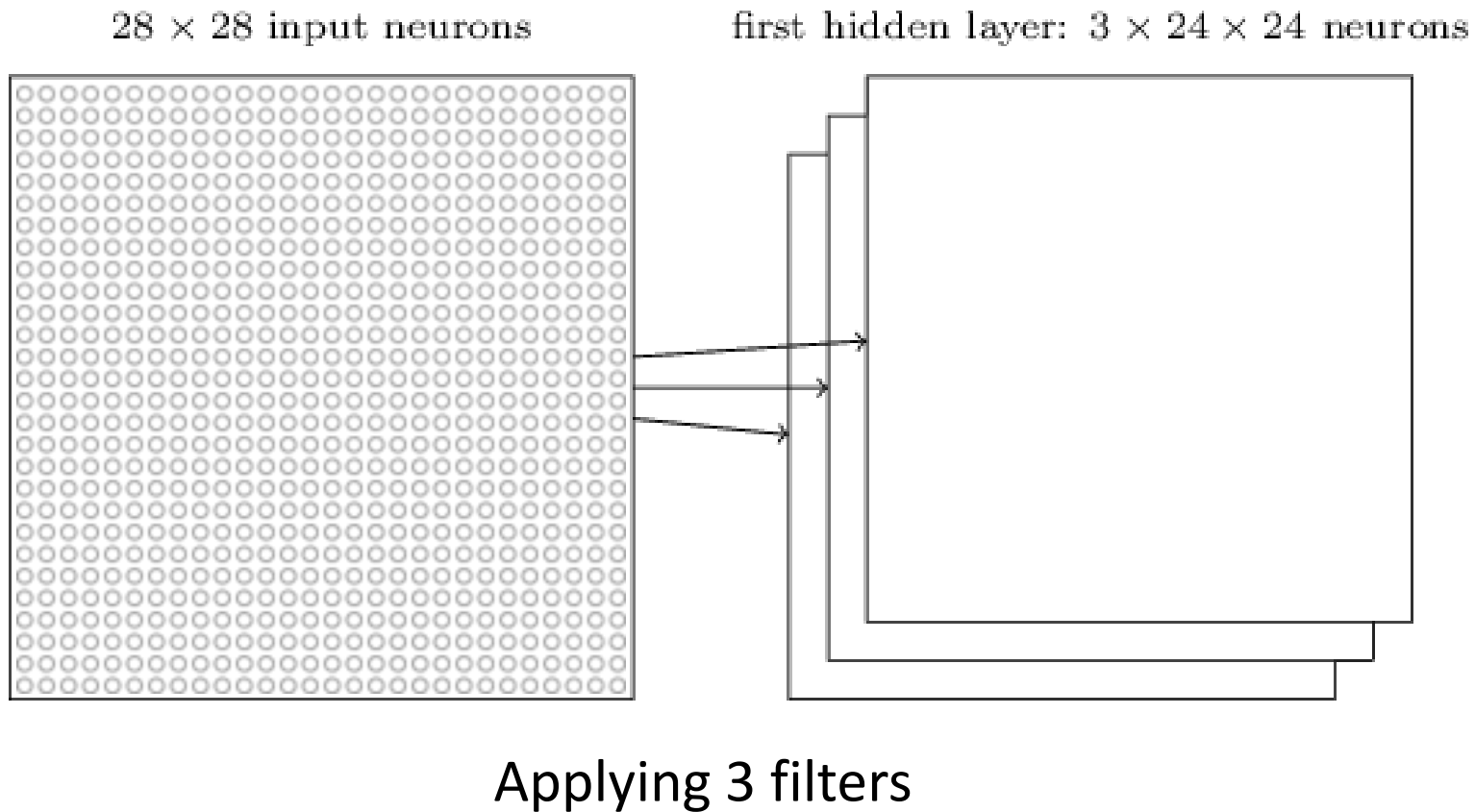
Convolutional Neural Network



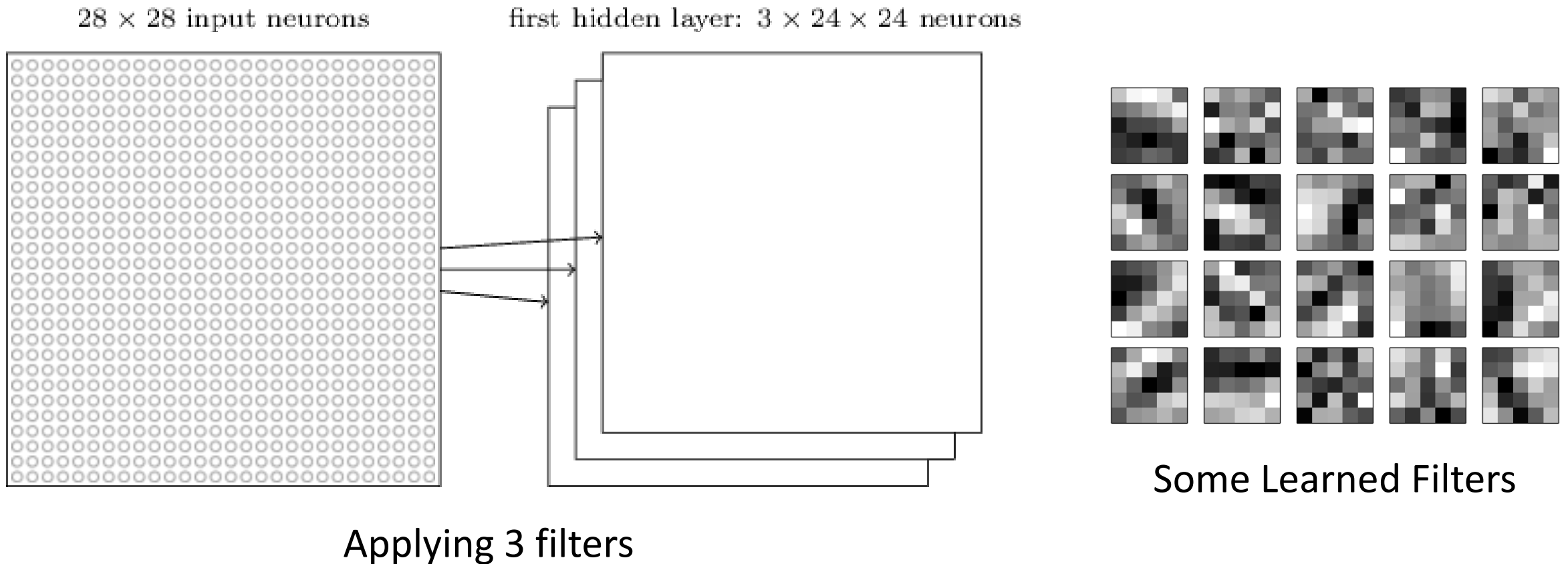
Convolutional Neural Network



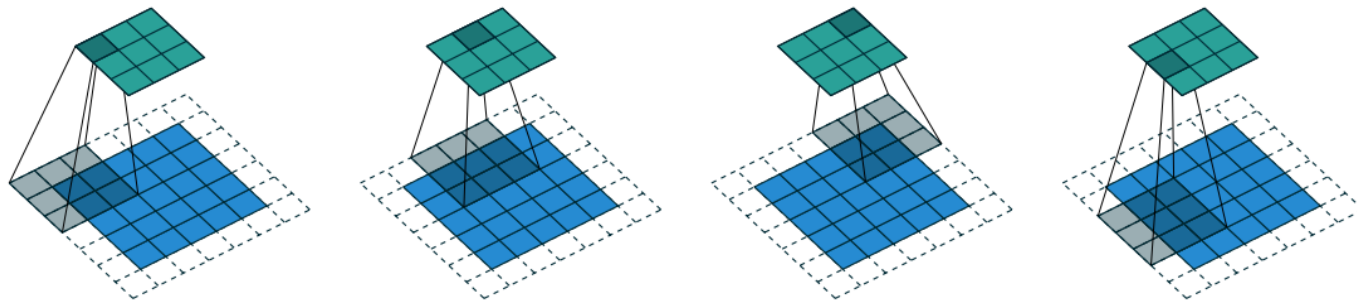
Convolutional Neural Network



Convolutional Neural Network



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.

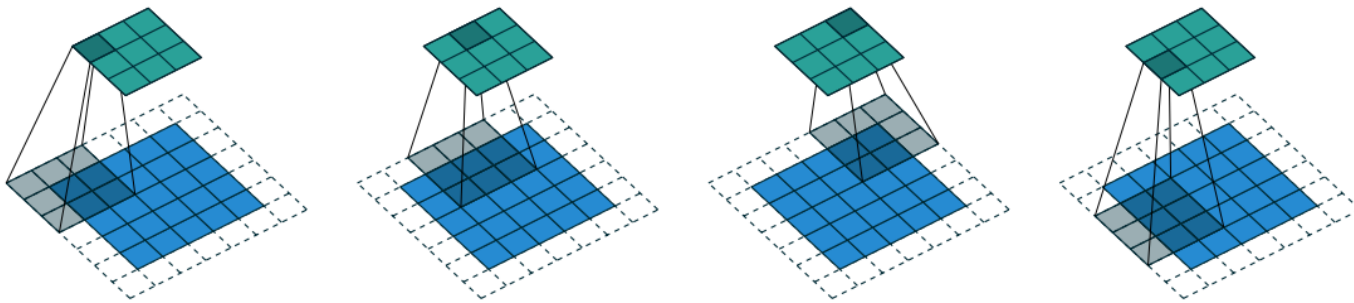
Dilation: Dilate the kernel to artificially increase the receptive field.

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

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

Convolutional Neural Network

Stride: Apply the kernel using a step size.

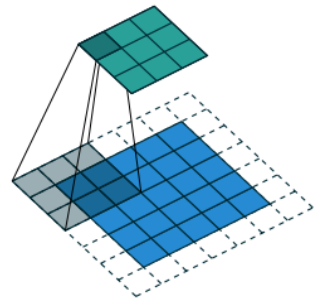


Stride: 2, Padding: 1, no Dilation

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

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

Convolutional Neural Network



Stride

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

*

0	1
2	3

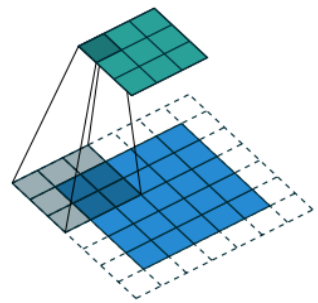
=

using a

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

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

Convolutional Neural Network



Stride

Input

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

*

Kernel

0	1
2	3

=

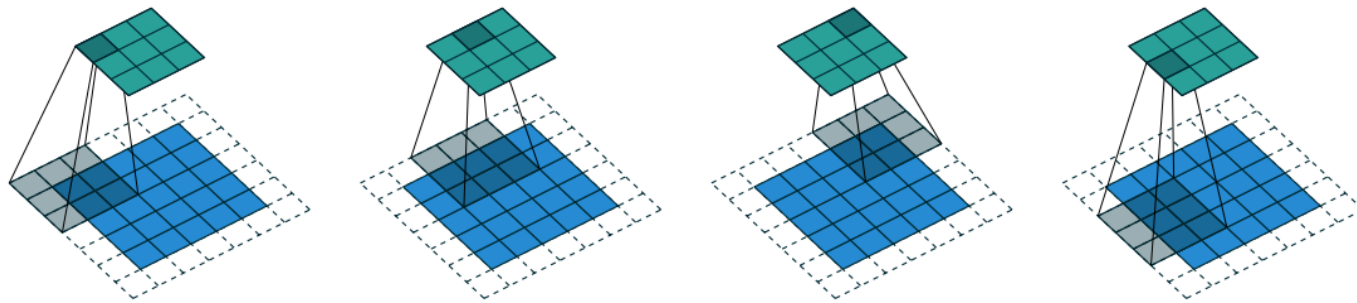
Output

using a

<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

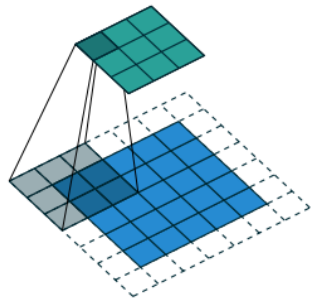
Stride: Apply the kernel using a step size.

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



Stride

Input

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

*

Kernel

0	1
2	3

=

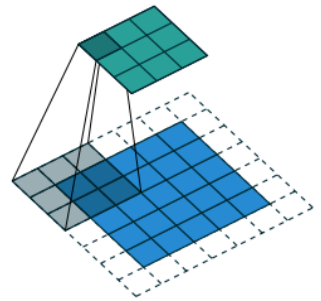
Output

using a
input with a
inking of

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

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

Convolutional Neural Network



Stride

Input

0	1	2
3	4	5
6	7	8

Kernel

0	1
2	3

*

=

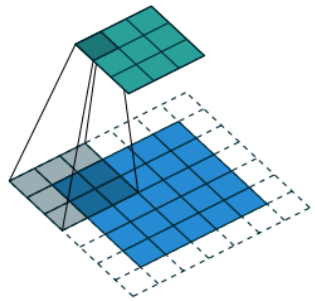
Output

using a
input with a
inking of

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

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

Convolutional Neural Network



Input

0	1	2
3	4	5
6	7	8

Kernel

0	1
2	3

*

=

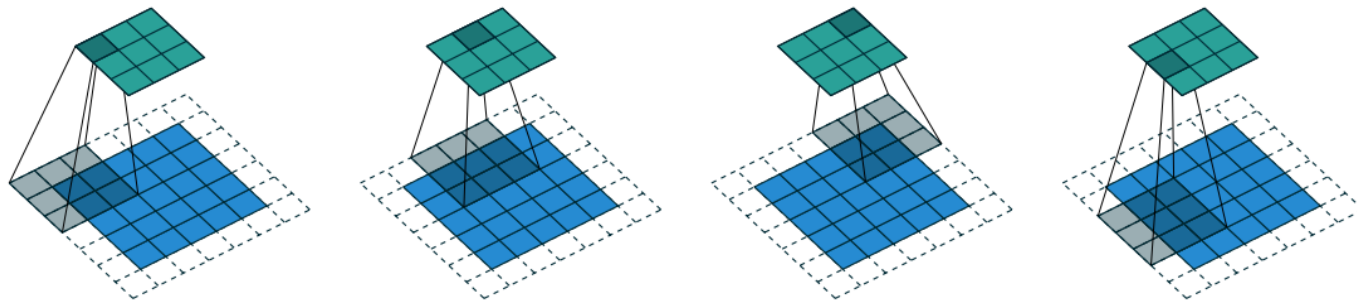
Output

using a
input with a
linking of

<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.

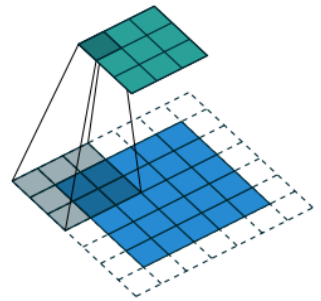
Padding: Surround the input with a pad value to prevent shrinking of the input.

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



Stride

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

*

0	1
2	3

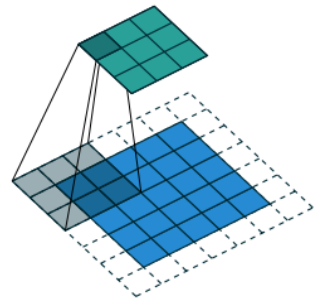
=

using a
input with a
linking of
el to

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

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

Convolutional Neural Network



Stride

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

*

0	1
2	3

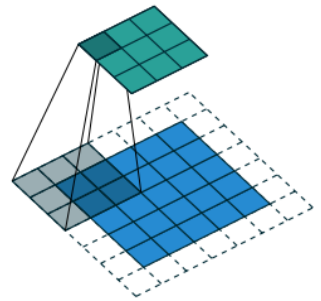
=

using a
input with a
linking of
el to

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

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

Convolutional Neural Network



Stride

Input				
0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

*

Kernel	
0	1
2	3

=

Output		

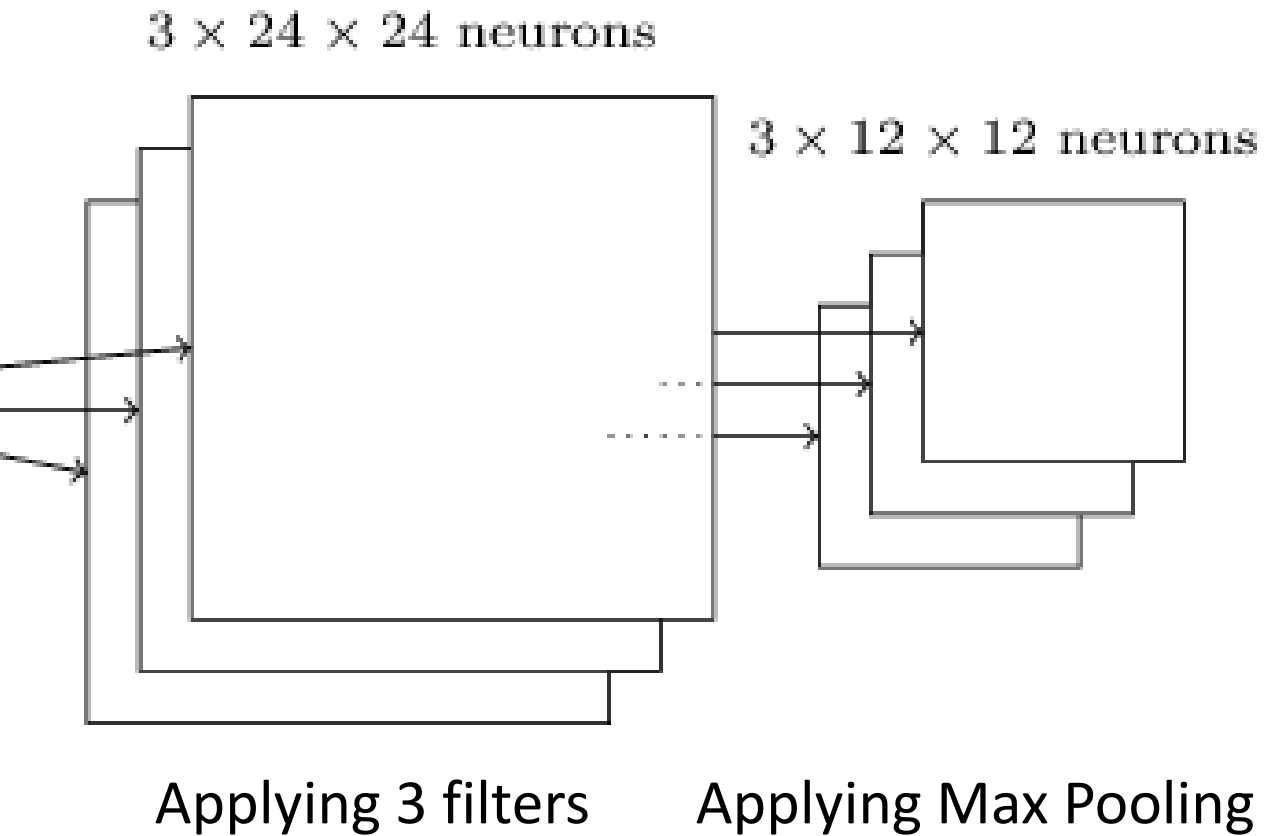
using a
input with a
linking of
el to

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

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

Convolutional Neural Network

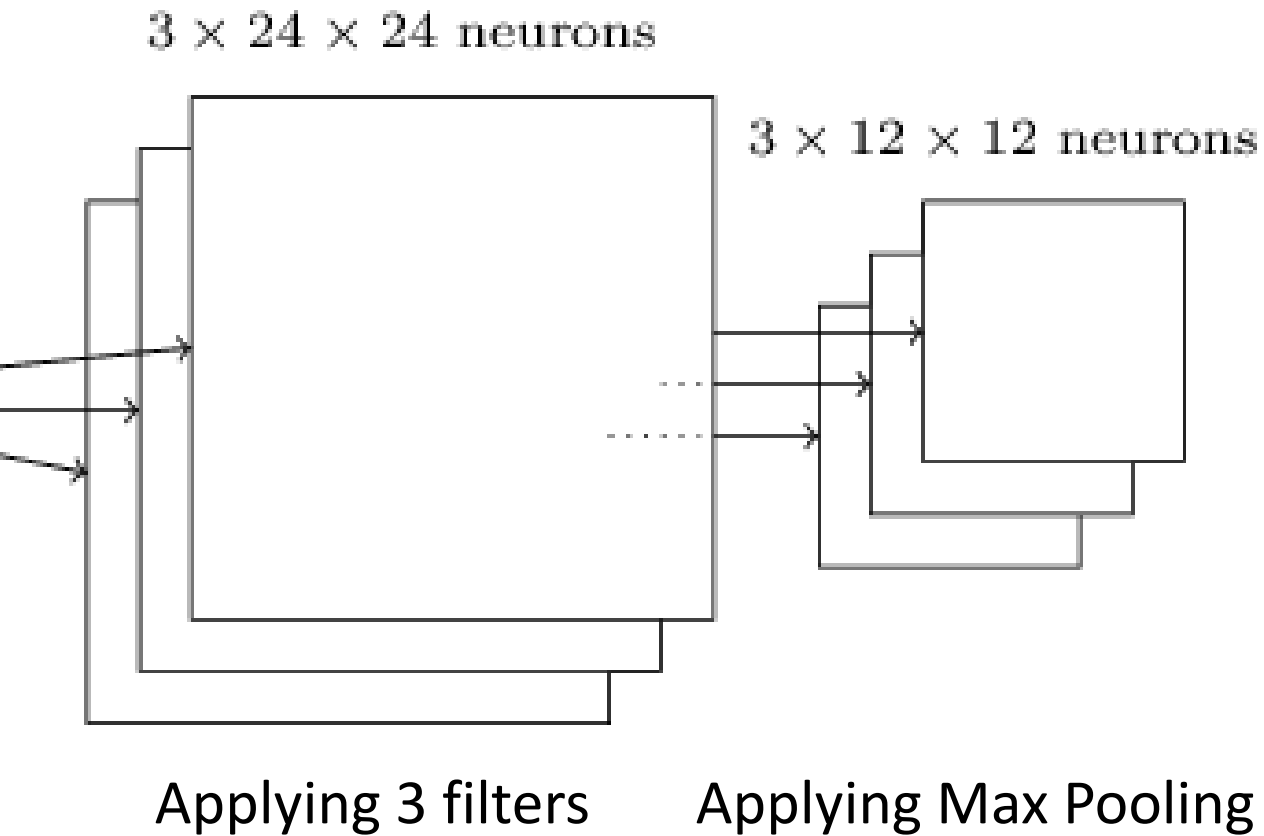
Pooling Operation



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

Convolutional Neural Network

Pooling Operation

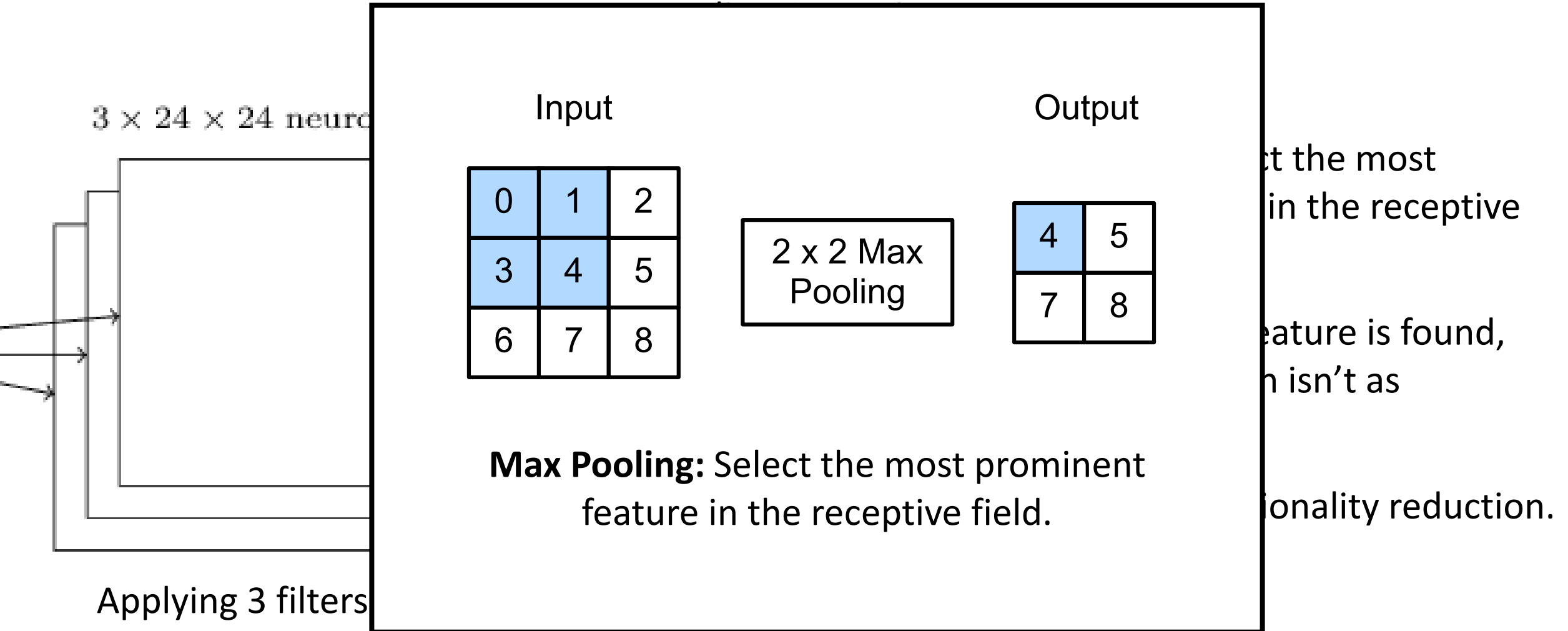


Max Pooling: Select the most prominent feature in the receptive field.

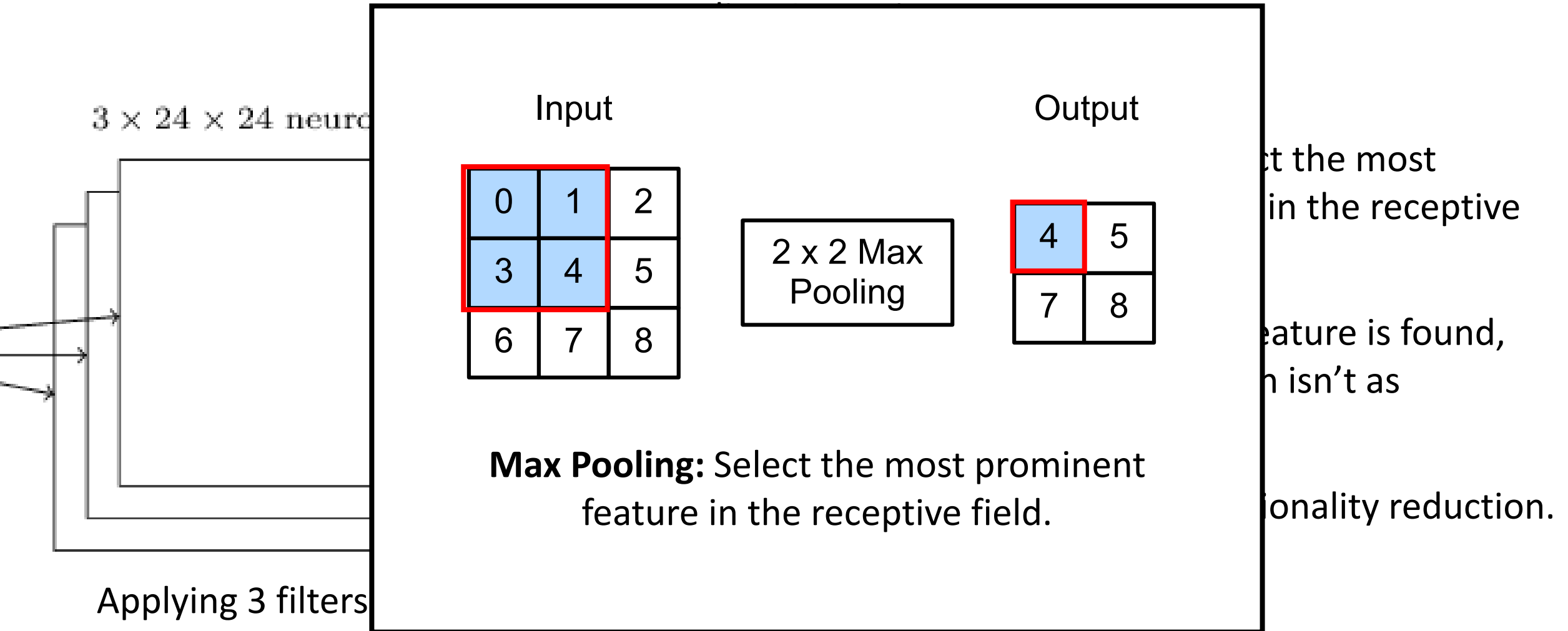
Intuition: Once a feature is found, the precise location isn't as important.

Helps with dimensionality reduction.

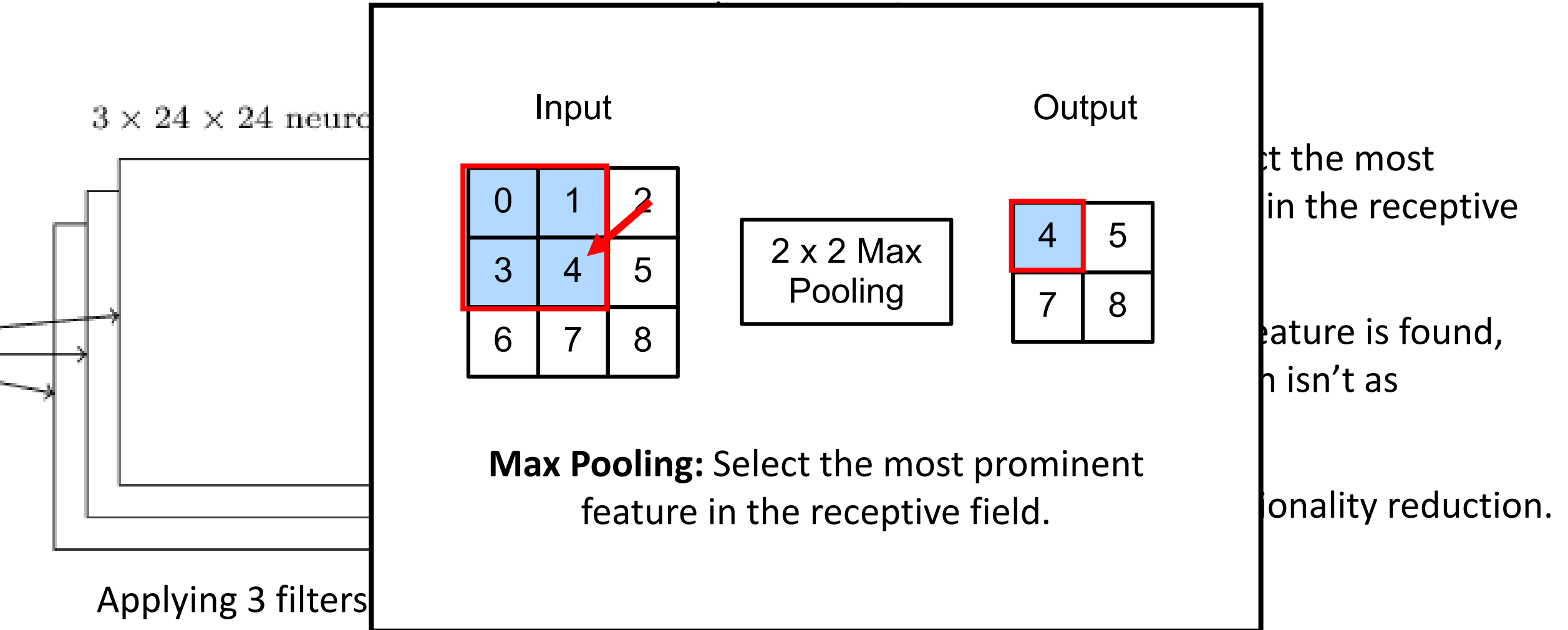
Convolutional Neural Network



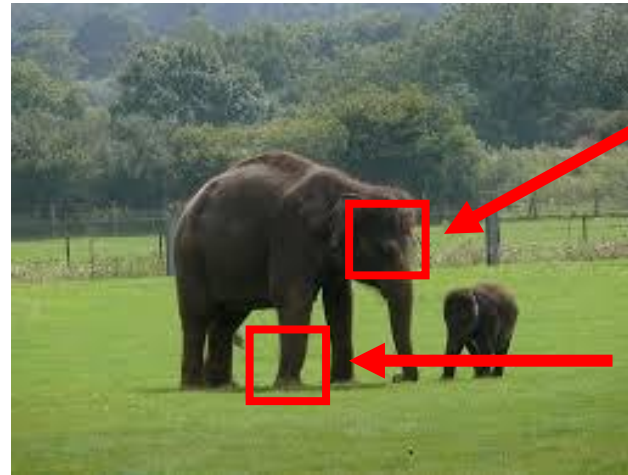
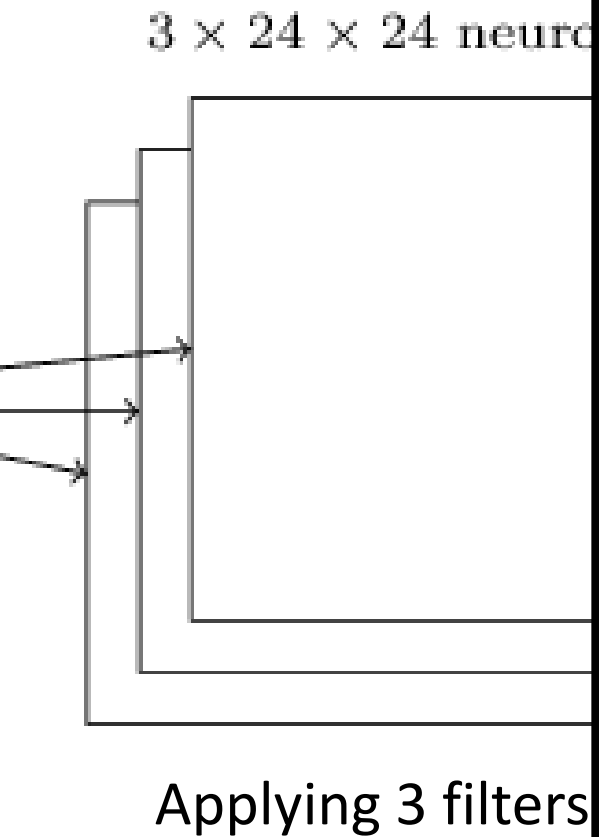
Convolutional Neural Network



Convolutional Neural Network



Convolutional Neural Network



eye somewhere in here

leg somewhere in here

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

at the most
in the receptive

feature is found,
n isn't as

onality reduction.

Convolutional Neural Network

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

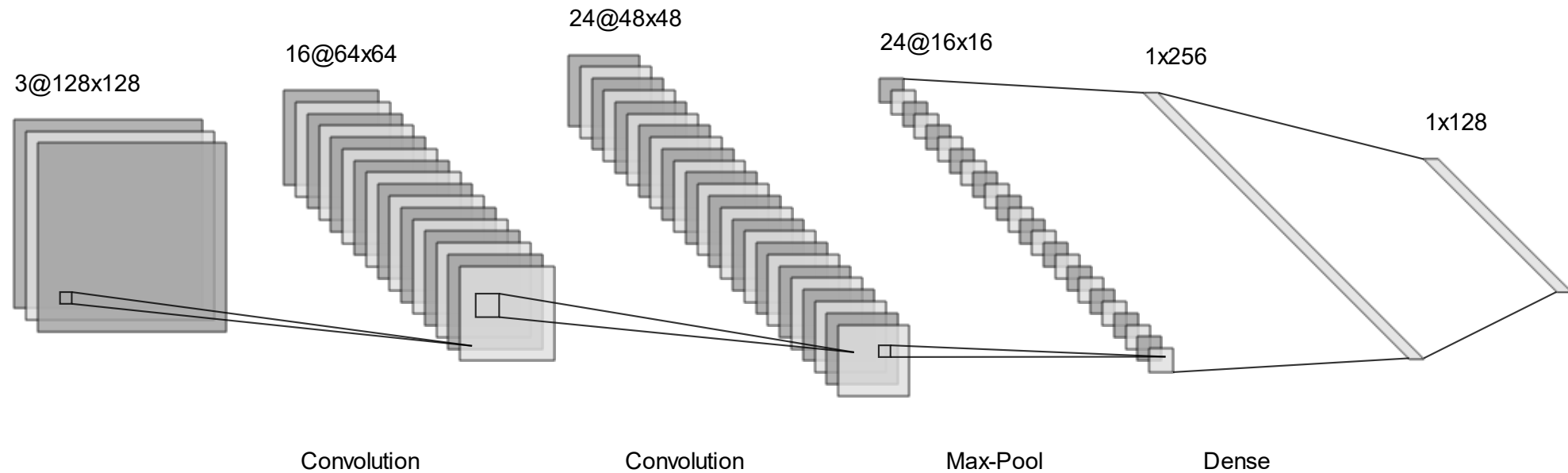


Image classification over 128 classes

Neural Networks

Building Blocks: Normalization and Stability

Batch Normalization:

Tries to overcome “internal covariate shift.”

Gradient updates assume the prior layer has a static distribution. Batch normalization normalizes the output of the prior layer so the distribution is standardized.

Dropout:

Tool to prevent overfitting.

Randomly removes nodes from a given layer, simulating training many architectures in parallel.

Neural Networks

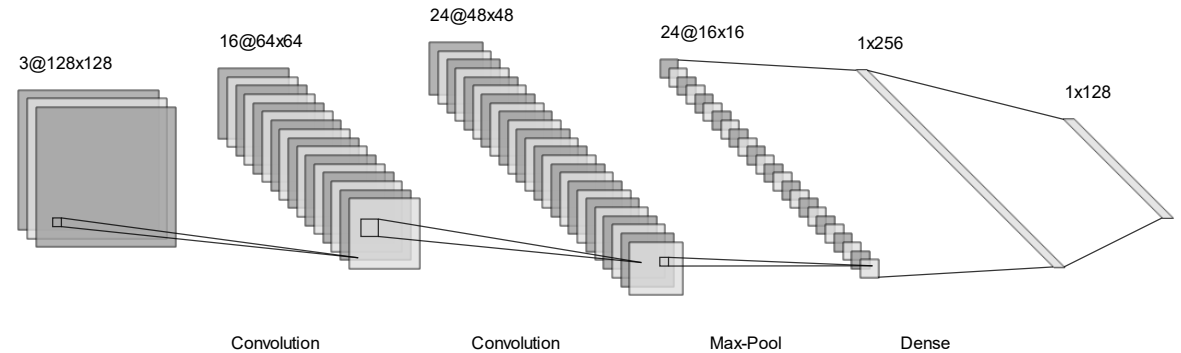
How do we get a network to learn?

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 $f(x)$

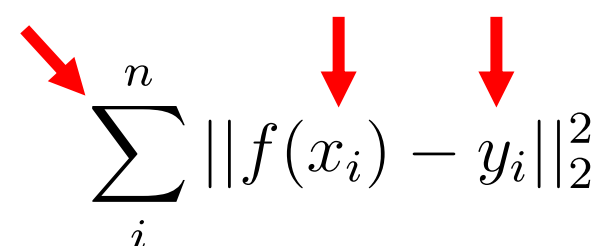
Neural Networks

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 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.

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

L2 Loss Function

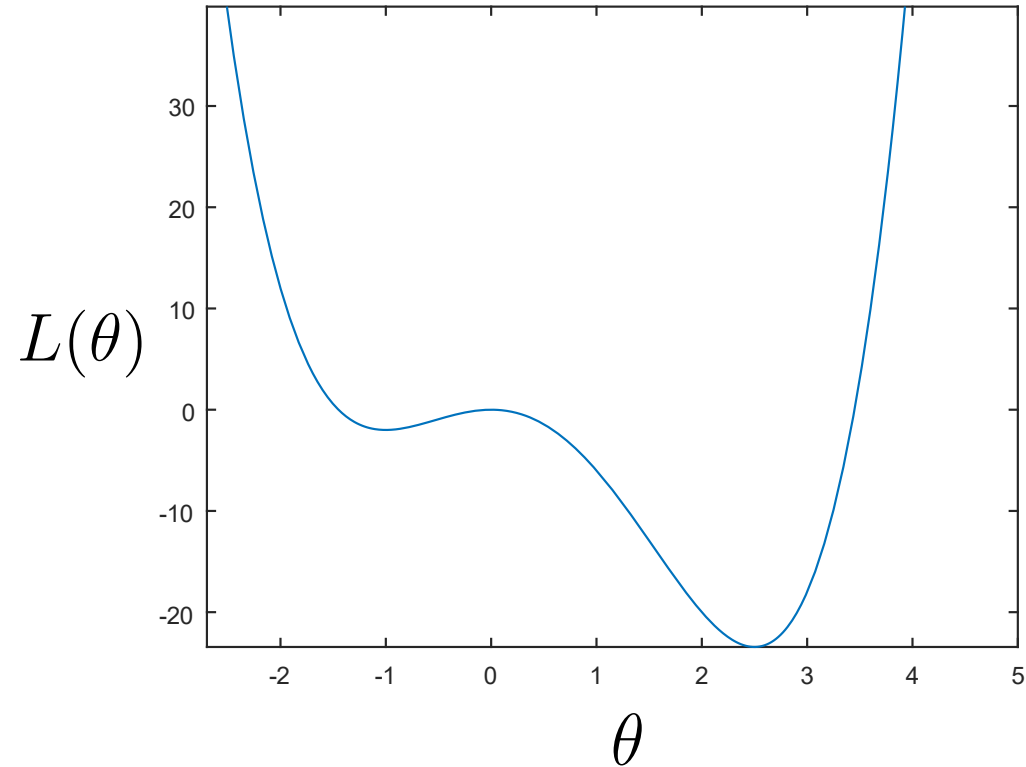
Obtain θ that minimizes L

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.

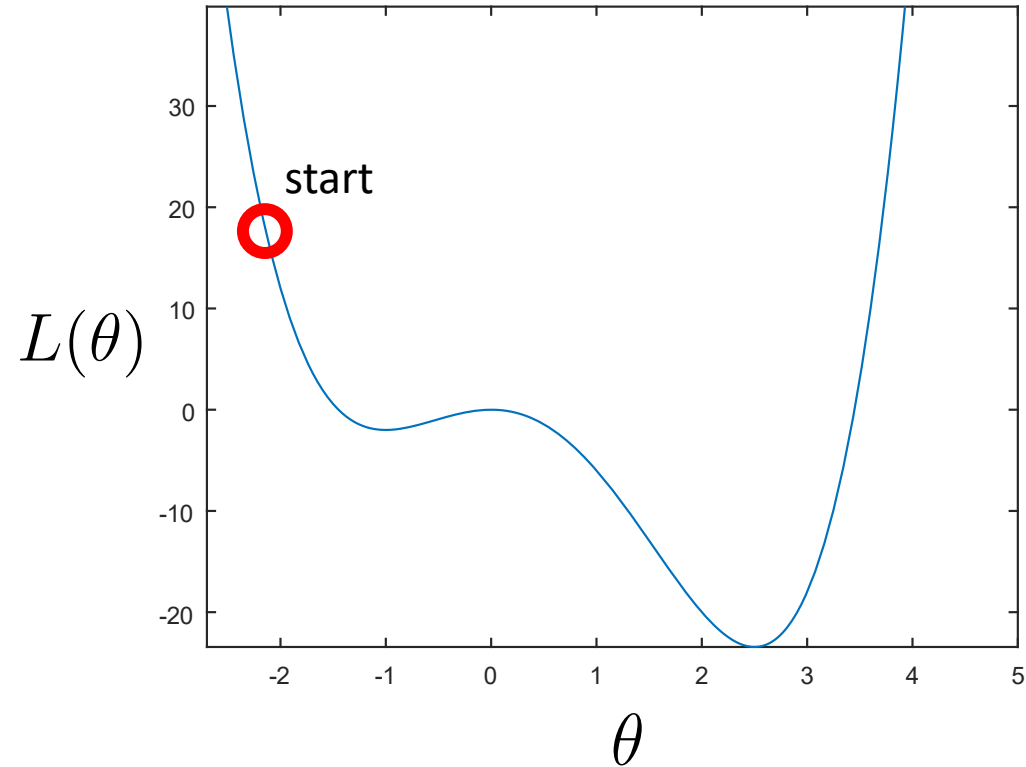


Neural Networks

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.

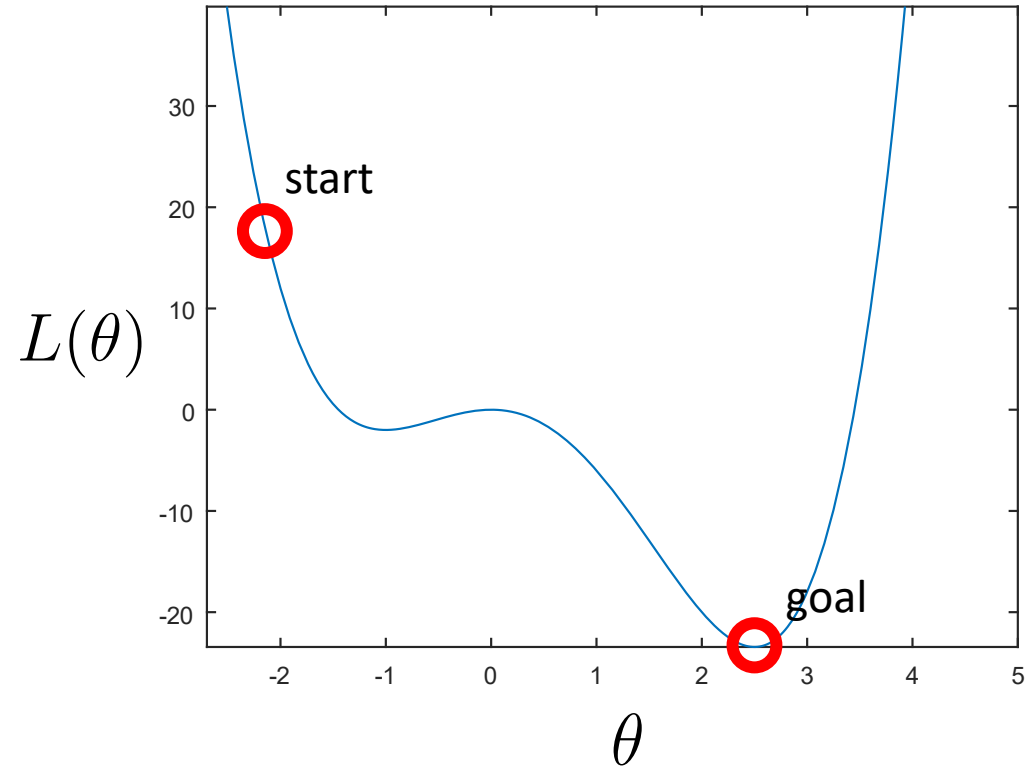


Neural Networks

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.

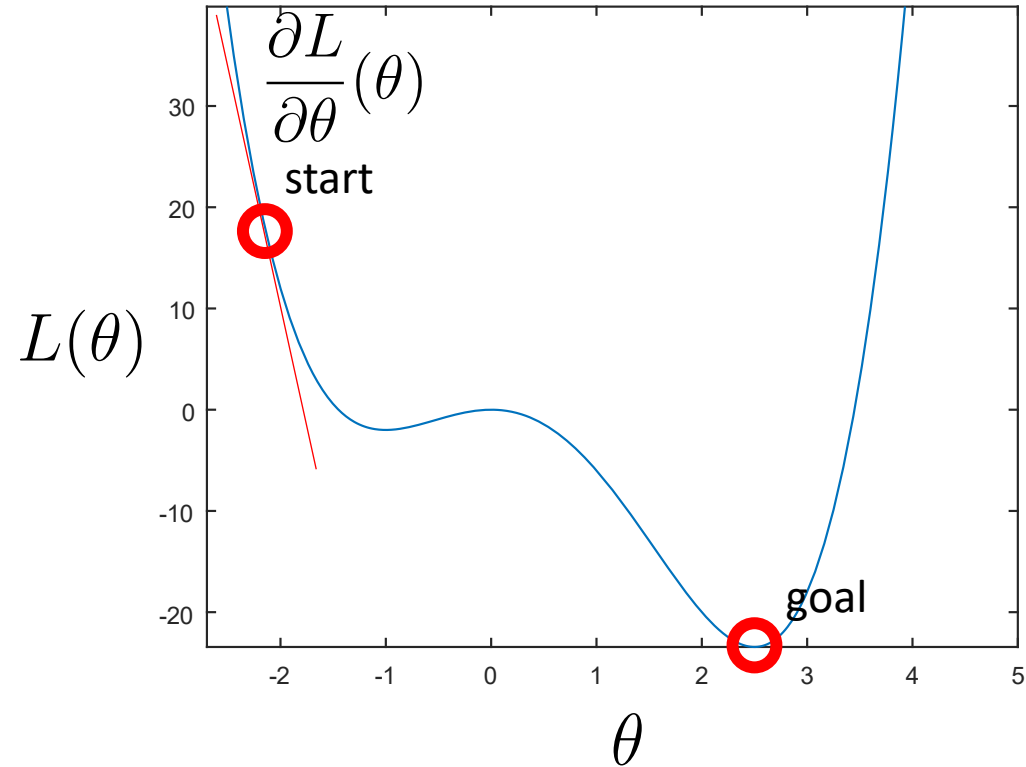


Neural Networks

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.

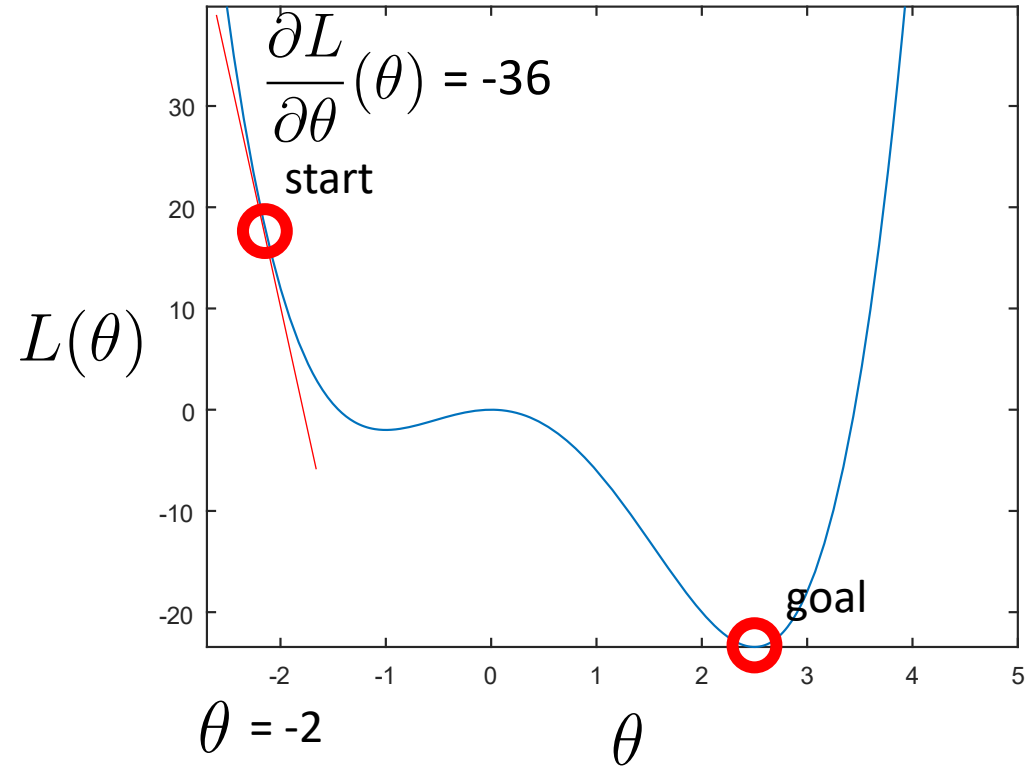


Neural Networks

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?

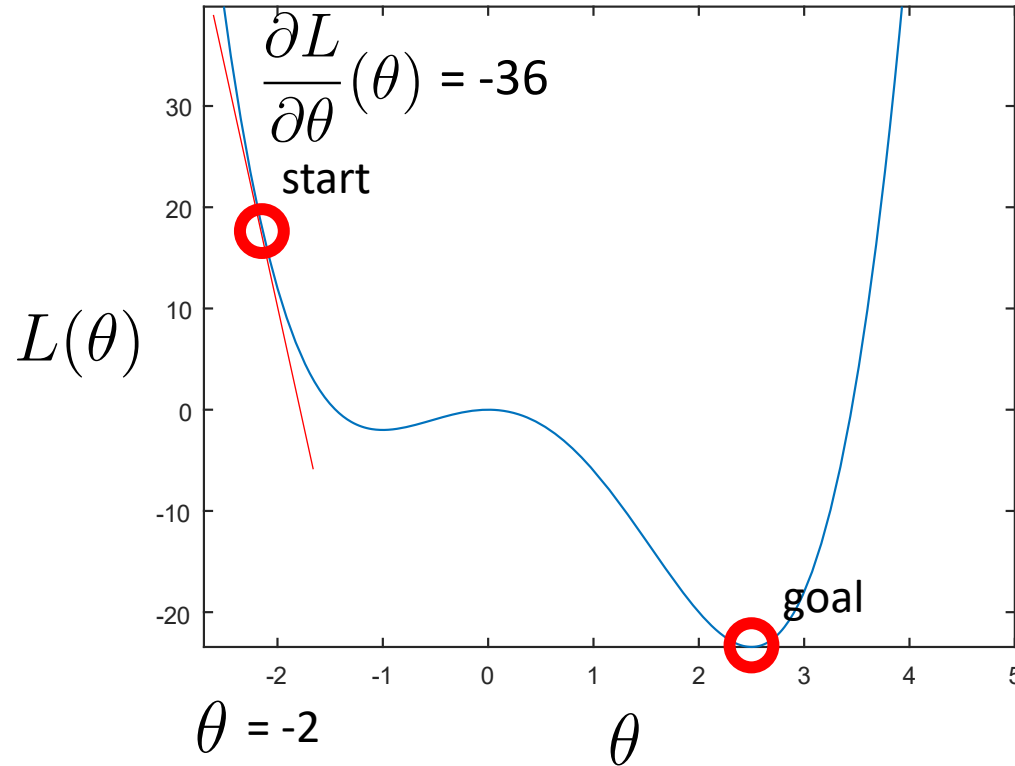
Gradient Descent.

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?

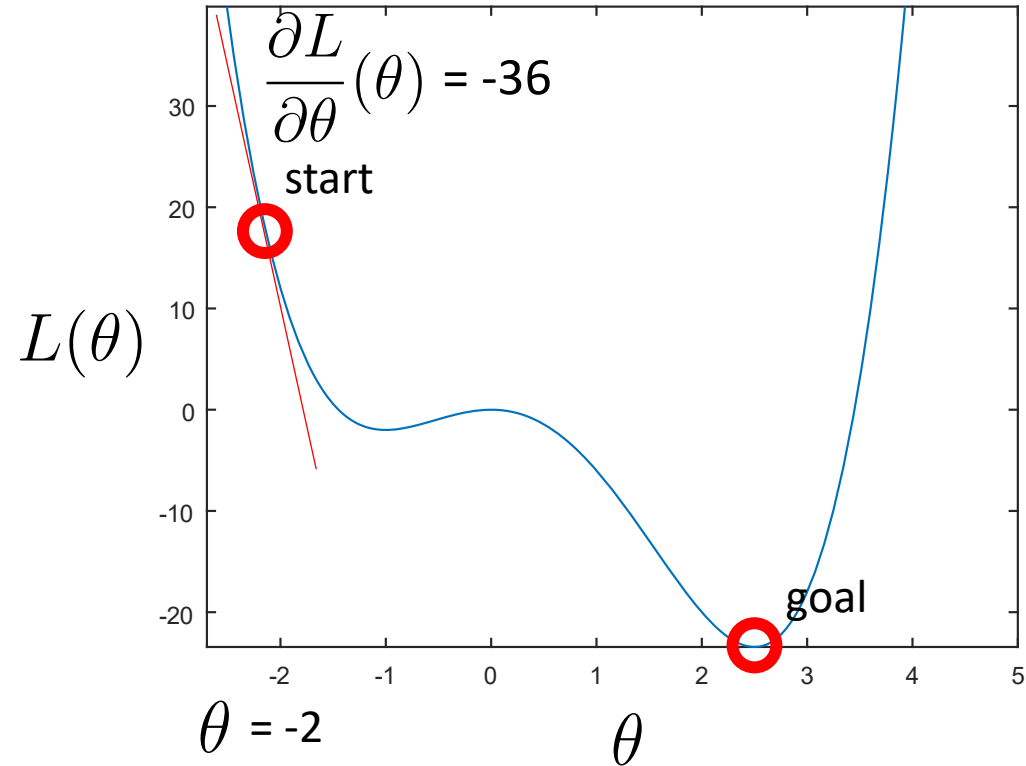
Gradient Descent.

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.**

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

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

Update Rule

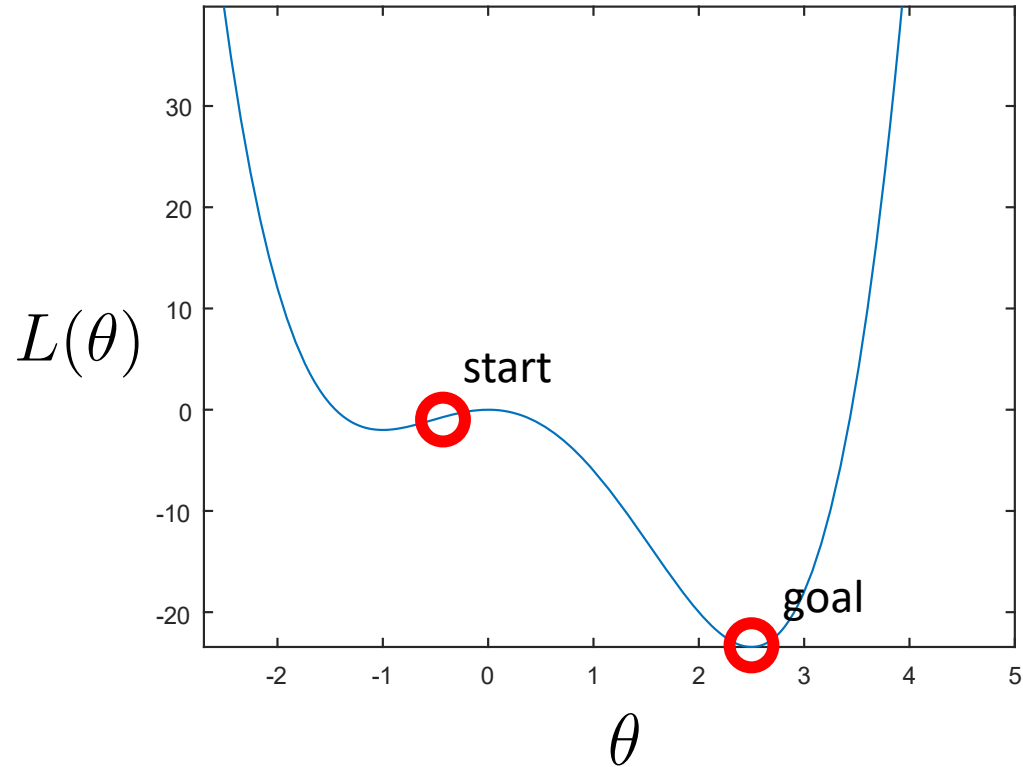


Neural Networks

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.

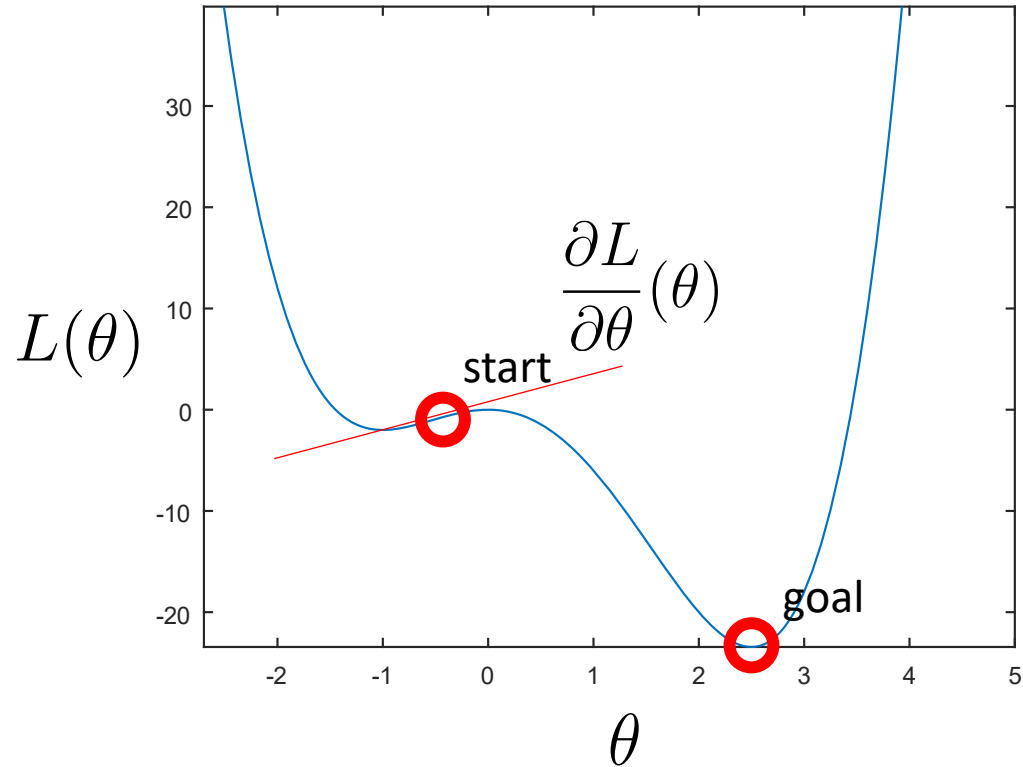


Neural Networks

How do we get a network to learn?

Gradient Descent.

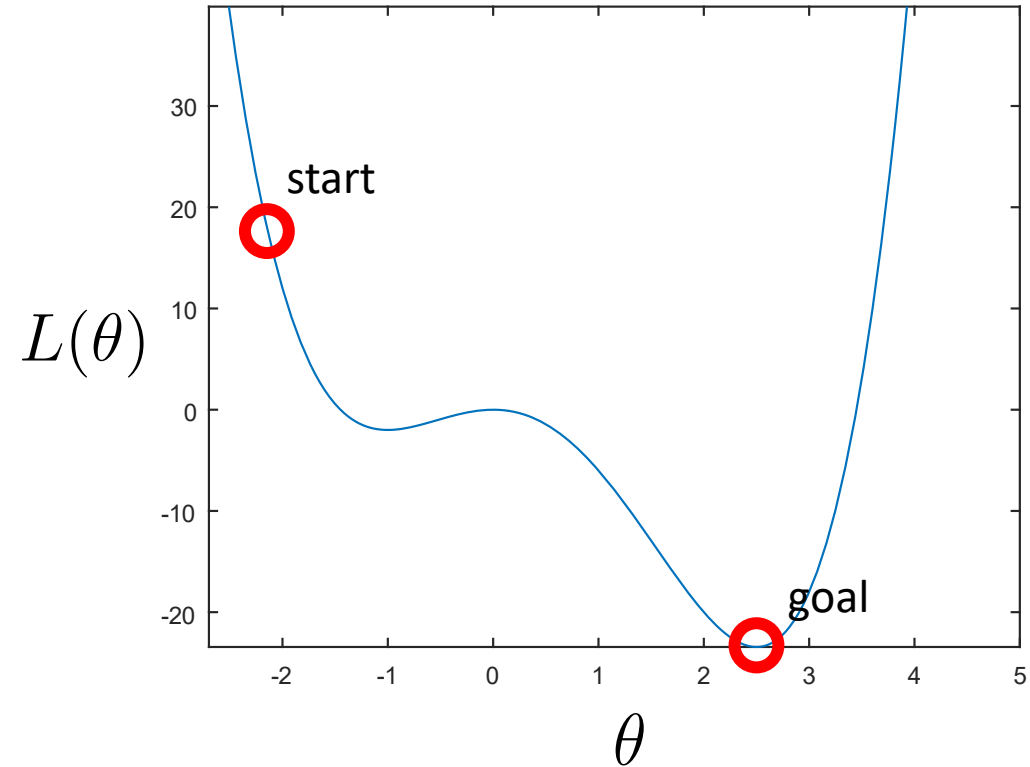
1. Estimate the error.
2. Compute the change in weights most likely to reduce the error (gradient).
3. Update the weights according to the gradient.



Neural Networks

How do we get a network to learn?

How do we choose lr (alpha)?

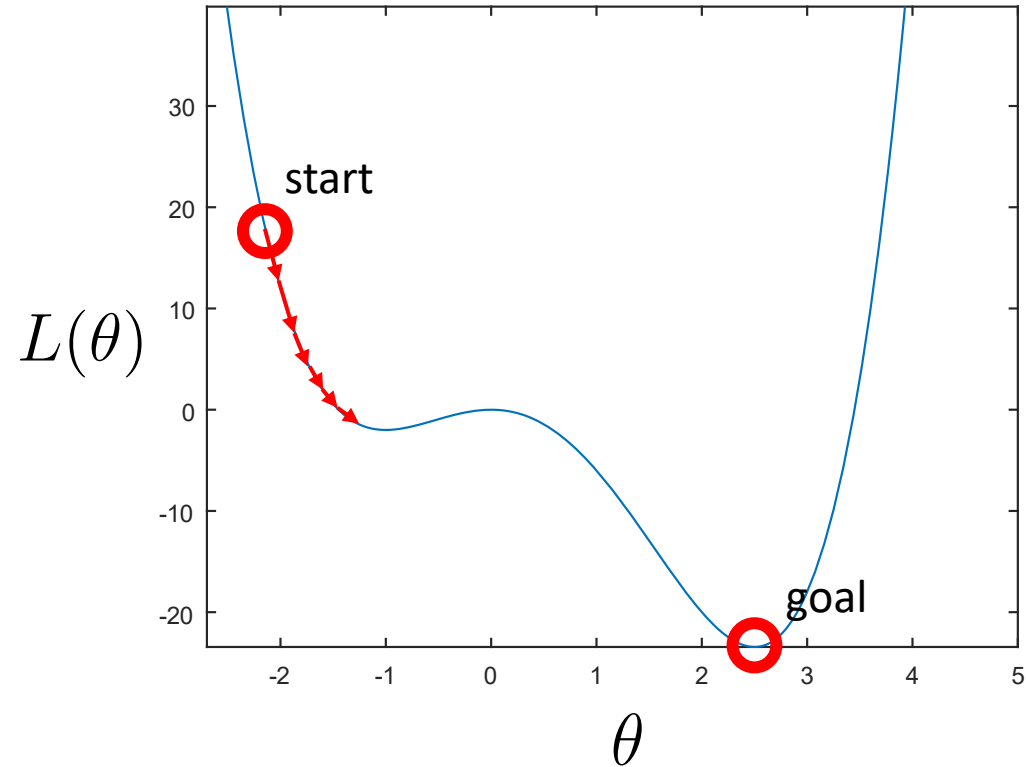


Neural Networks

Building Blocks: Optimizer

How do we choose lr (alpha)?

- Small lr will get caught in local minima.

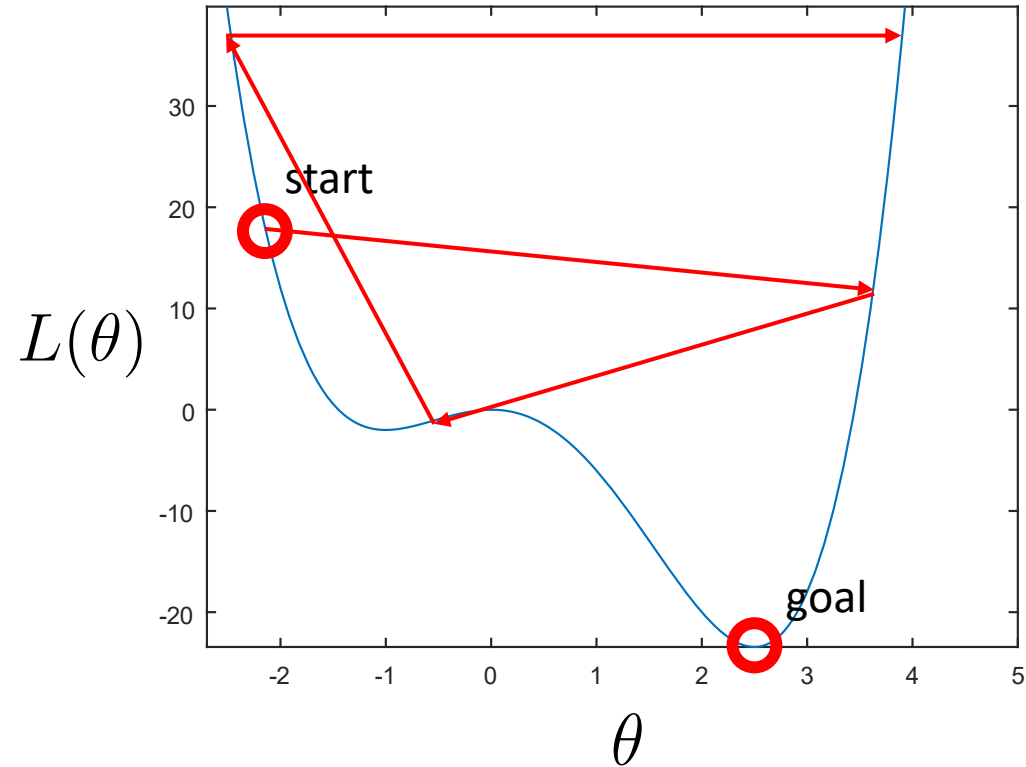


Neural Networks

Building Blocks: Optimizer

How do we choose lr (alpha)?

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



Neural Networks

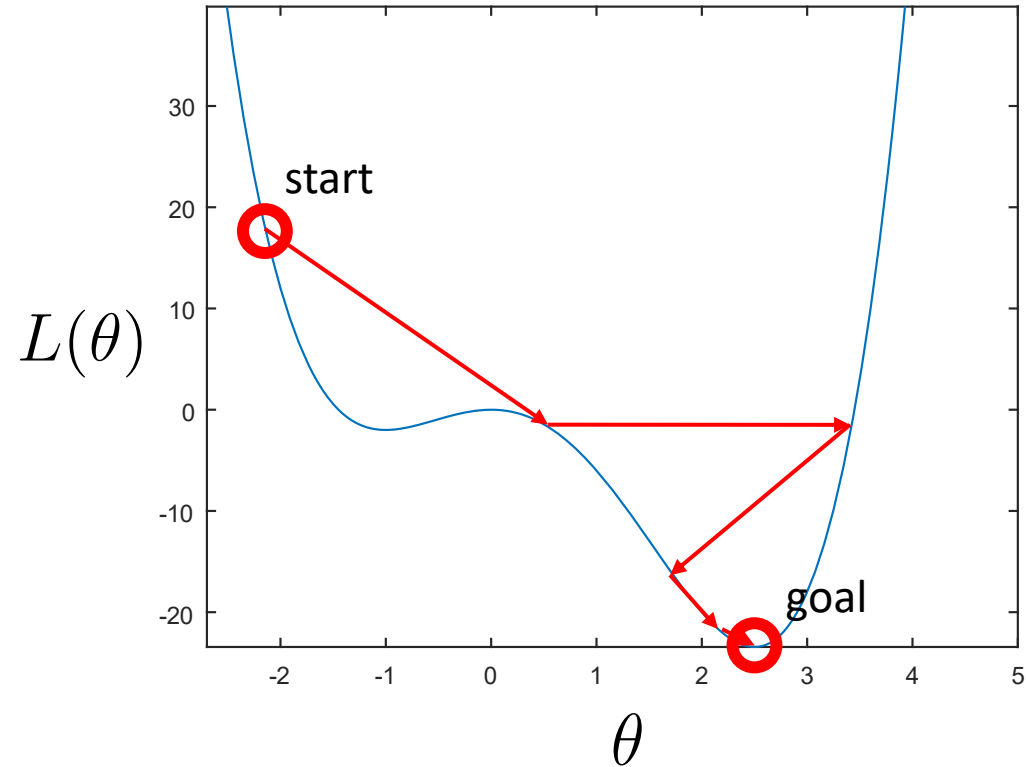
Building Blocks: Optimizer

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

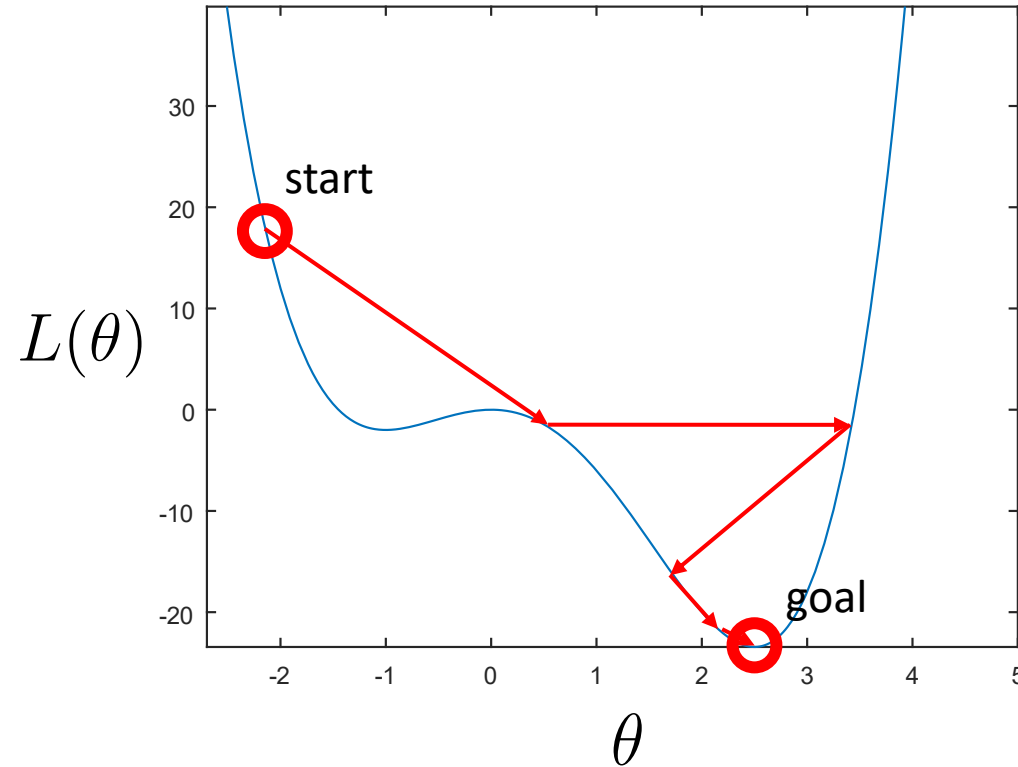
Building Blocks: Optimizer

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: 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

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.

Batch Normalization Layer: Improves training stability.

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.

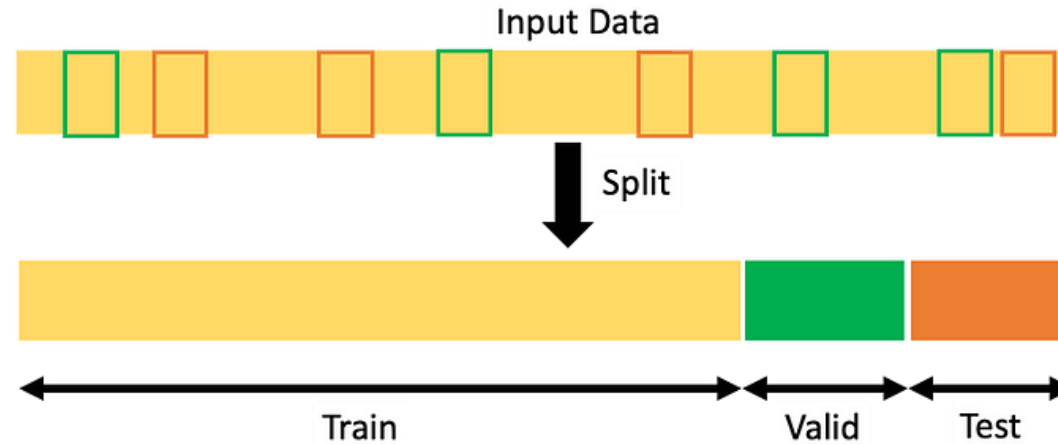
Validation Set:

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>

<https://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 hurt performance. Try adjusting rate to aid convergence. learning rate decay

Forgot Data Normalization

Neural Networks typically require normalized data.

Using large batch sizes:

Evidence suggests that overly large batch sizes inhibit learning.

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.

by with specific

are not always network and

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

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

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

Tools

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.

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

Development Environment

Environments (containerization)

Virtual environments allow multiple dependency configurations to be installed on a system at once.

Environments (containerization)

Virtual environments allow multiple dependency configurations to be installed on a system at once.

Why is this good?

- 1) You can instruct people how to install your dependences.
- 2) When you inevitably screw something up, makes it easy to start over.

Environments (containerization)

Virtual environments allow multiple dependency configurations to be installed on a system at once.

Why is this good?

- 1) You can instruct people how to install your dependences.
- 2) 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.

Using Virtual Environments

virtualenv

- `lambne@tarserver2:~/dev/classes/cs473$ python3 -m venv env`
- `lambne@tarserver2:~/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

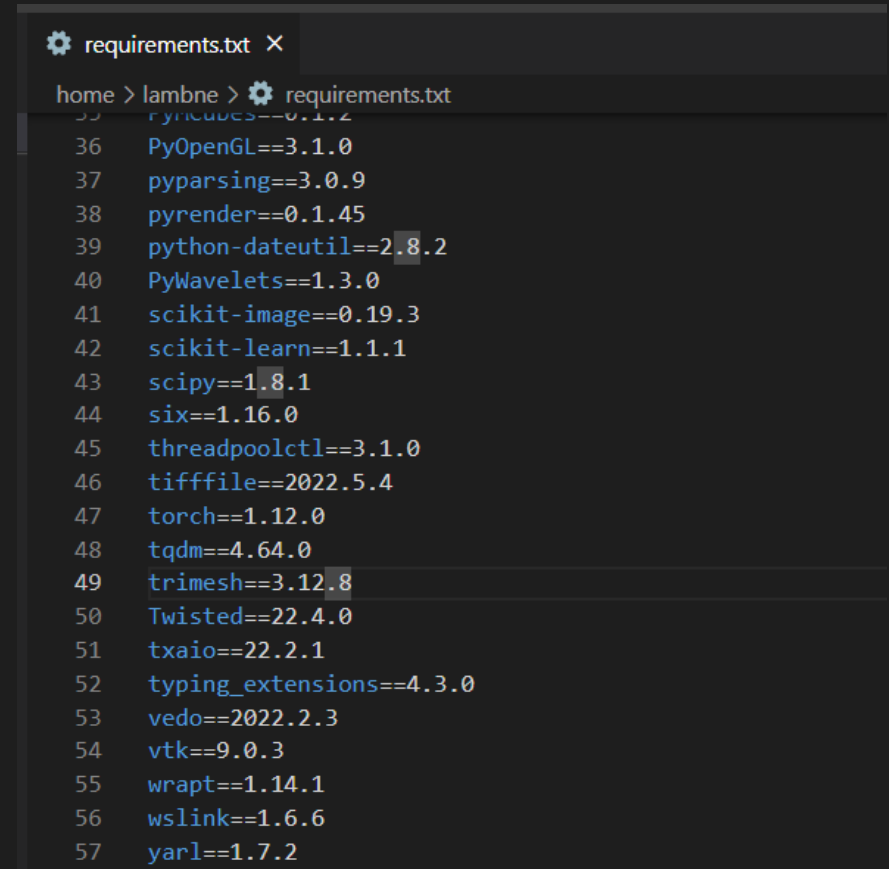
virtualenv

```
● 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 X
home > lambdne > requirements.txt
35 PyOpenGL==3.1.0
36 PyOpenGL==3.1.0
37 pyparsing==3.0.9
38 pyrender==0.1.45
39 python-dateutil==2.8.2
40 PyWavelets==1.3.0
41 scikit-image==0.19.3
42 scikit-learn==1.1.1
43 scipy==1.8.1
44 six==1.16.0
45 threadpoolctl==3.1.0
46 tifffile==2022.5.4
47 torch==1.12.0
48 tqdm==4.64.0
49 trimesh==3.12.8
50 Twisted==22.4.0
51 txaio==22.2.1
52 typing_extensions==4.3.0
53 vedo==2022.2.3
54 vtk==9.0.3
55 wrapt==1.14.1
56 wslink==1.6.6
57 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**\$

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

The dataset handles loading and transforming the input data.

Must implement the `__len__` and `__getitem__` methods.

```
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
```

- 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 backpropagation
 - 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()
```

Tools

VSCode:

Lightweight IDE replacement that is extremely popular for python programming with an extensive extensions library.

Screen:

Enables training “in the background”.

Virtualenv:

Python environment containerization.

PyTorch:

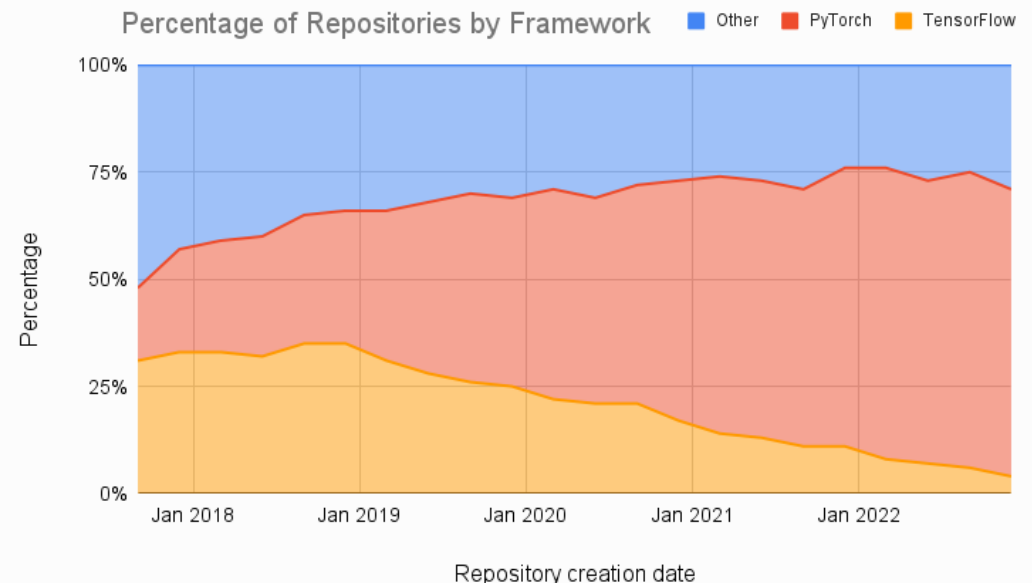
Machine learning framework based on the open-source torch library.

Remote Explorer (VSCode Extension):

Connect to remote servers through VSCode.

Pylance (VSCode Extension):

IntelliSense for python.



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>