Rajalakshmi eduverse

Deep Learning

Deep Learning

- Subset of machine learning
- Neural networks with three or more layers
- Imitates how humans process data and make decisions
- Exponential growth in the last few years
- Powered by advances in large-scale data processing and inference

Applications of Deep Learning

- Natural language processing (NLP)
- Speech recognition and synthesis
- Image recognition
- Self-driving cars
- Customer experience, healthcare, and robotics

Scope and Prerequisites

Deep Learning: Background

- Vast domain with a variety of tools and technologies
- Multiple courses exist for concepts and implementations
- Math requirements and coverage depth
- Tools—understanding and depth

Scope of the Course

- Educate students about the basic concepts of deep learning
- Focus on what happens behind-the-scenes
- Use Keras for implementation—mask complexity
- Minimal math coverage, deeper topics omitted
- Simple examples to get started with deep learning

Course Prerequisites

- Machine learning concepts and technologies
- Python programming, notebooks
- Keras and TensorFlow
- scikit-learn, text processing with NLTK

Deep Learning

- Subset of machine learning
- Neural networks with three or more layers
- Imitates how humans process data and make decisions
- Exponential growth in the last few years
- Powered by advances in large-scale data processing and inference

Applications of Deep Learning

- Natural language processing (NLP)
- Speech recognition and synthesis
- Image recognition
- Self-driving cars
- Customer experience, healthcare, and robotics

Linear Regression

Linear Regression

- A linear model
- Relationship between two or more variables
- Predict dependent variable based on independent variable
- Slope and intercept models the relationship

y	Dependent Variable
x	Independent Variable
a	Slope
b	Intercept

$$y = ax + b$$

$$y = a_1x_1 + a_2x_2 \ldots + a_nx_n + b$$

Building a Linear Regression Model

- Find values for slope and intercept
- Use known values of x and y (multiple samples)
- Multiple independent variables make it complex

$5 = 2a + b,\ 9 = 4a + b$

$b = 5 - 2a$

$9 = 4a + 5 - 2a$

$a = (9-5)/2 = 2$

$b = 5 - 2*2 = 1$

Logistic Regression

- A binary model
- Relationship between two or more variables
- Output is 0 or 1

y	Dependent Variable
x	Independent Variable
a	Slope
b	Intercept
f	Activation Function

$$y = f(ax + b)d$$

$$y = f(a_1x_1 + a_2x_2 \ldots + a_nx_n + b)$$

Deep Learning Process

- Deep learning is a complex and iterative process
- Starts with random initialization and works towards the right values by trial and error

10 = 3a + b

Trial	a	b	3a + b	Error
1	1	1	4	6
2	4	3	15	-5
3	2	2	8	2
4	3	2	11	-1
5	2	3	9	1
6	2	4	10	0

What Is a Perceptron?

- An algorithm for supervised learning for binary classification
- Resembles a cell in the human brain
- A single cell neural network
- Based on logistic regression

Perceptron Formula

$$y = f(a_1x_1 + a_2x_2 \ldots + a_nx_n + b)$$

$$y = f (w_1x_1 + w_2x_2 \ldots + w_nx_n + b)$$

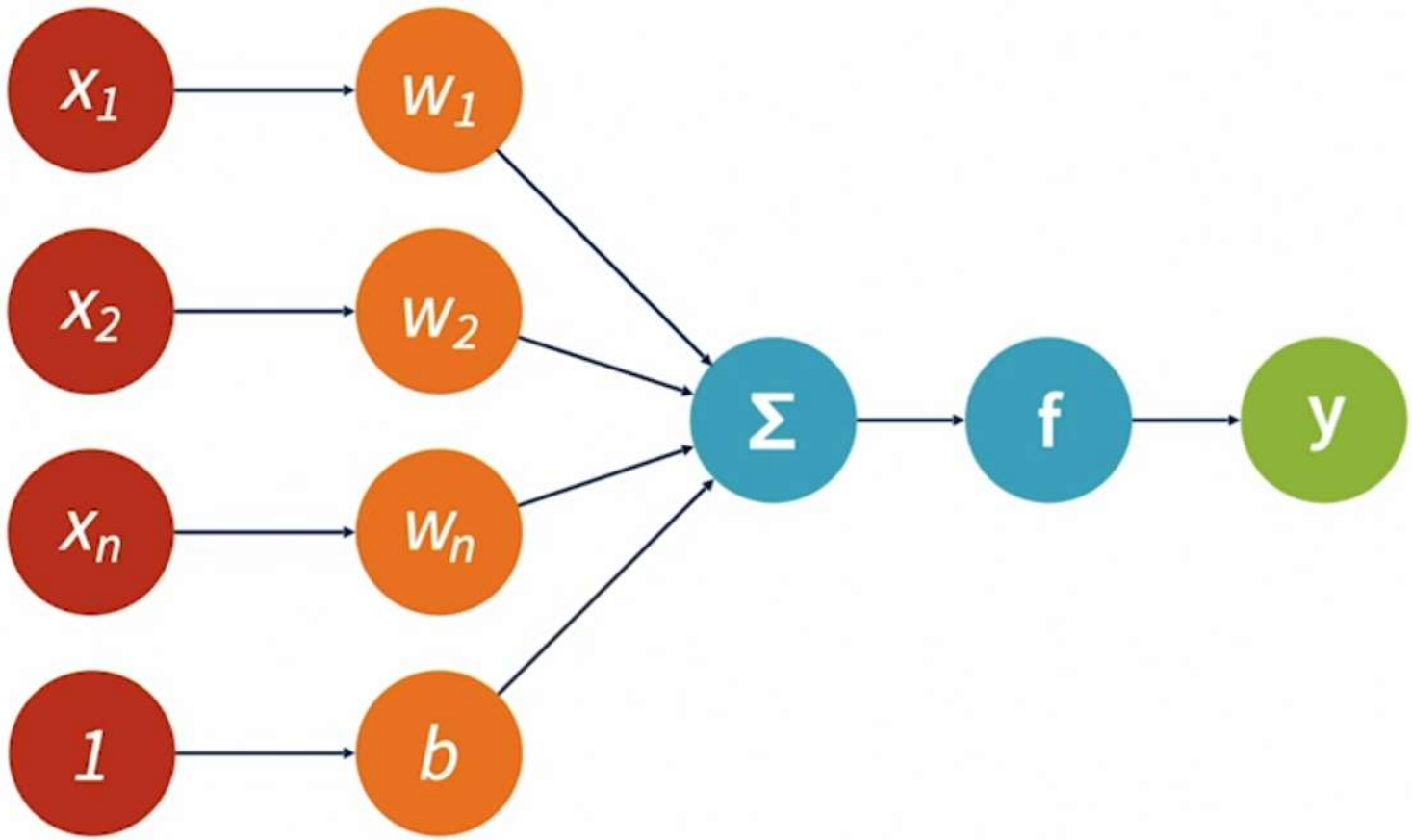
w	Weight
b	Bias
f	Activation Function

Activation Function Example

1 if value > 0

0 otherwise

Perceptron



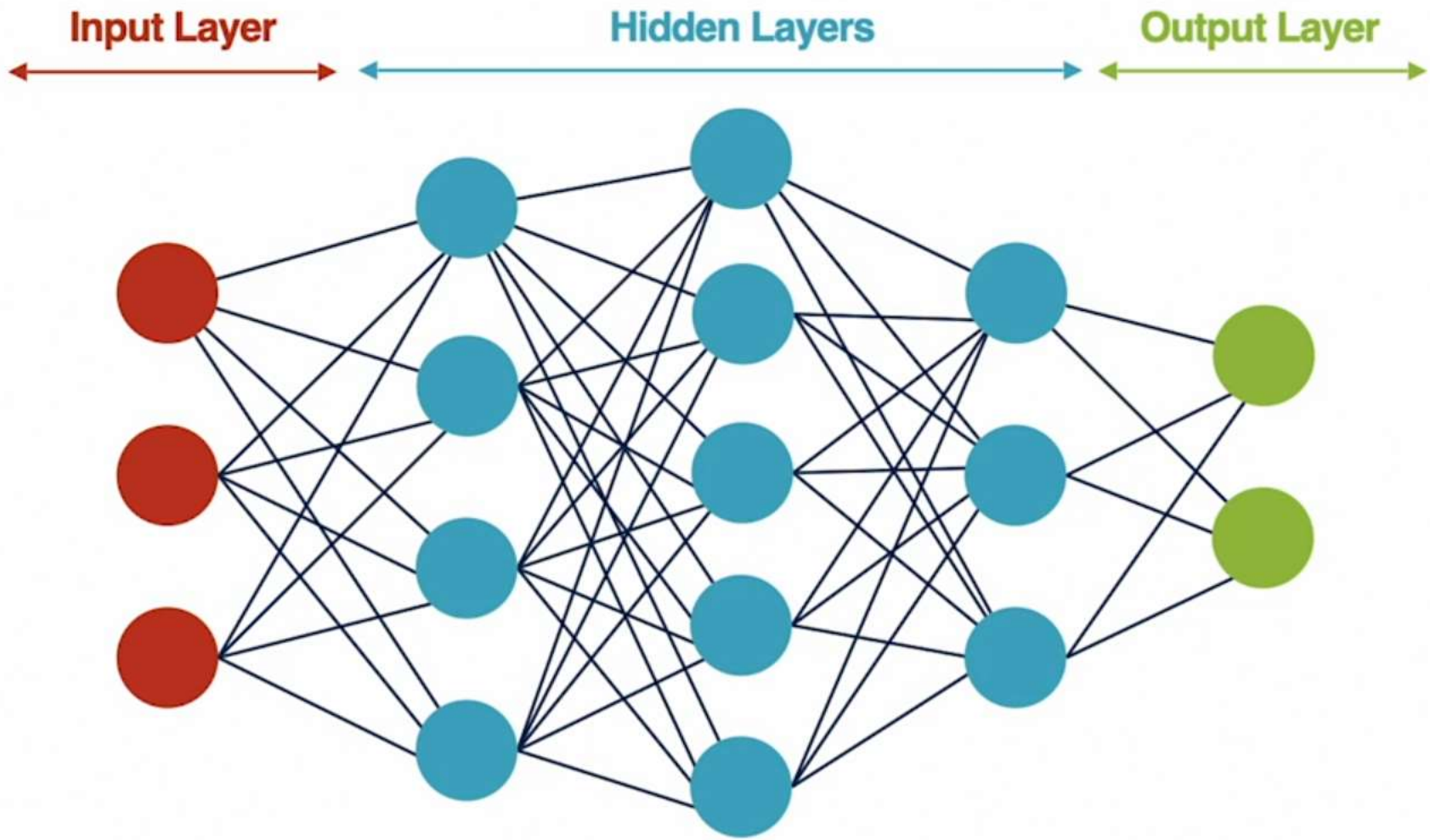
Artificial Neural Network (ANN)

A network of perceptrons, modeled after the human brain

What Is an Artificial Neural Network?

- Perceptrons are called *nodes* in the neural network
- Nodes organized as layers
- Each node has weights, biases, and activation functions
- Each node is connected to all nodes in the next layer

Artificial Neural Network



How ANN Works

- Inputs (independent variables) are sent from the input layer
- Inputs passed on to the nodes in the next hidden layer
- Each node computes its output based on its weights, biases, and activation functions
- Node output is then passed on as inputs to the next layer

Training an ANN

What Does Training Mean?

A model is represented by parameters and hyperparameters.

- Weights, biases, nodes, layers, layers, etc.

Training a model means determining the best possible values for the parameters and hyperparameters that maximize accuracy.

Inputs, weights, and biases might be n-dimensional arrays.

Recall the Linear Regression Analogy

Weight | Bias

Trial	a	b	3a + b	Error
1	1	1	4	6
2	4	3	15	-5
3	2	2	8	2
4	3	2	11	-1
5	2	3	9	1
6	2	4	10	0

Training Process

- Use training data (known values of inputs and outputs)
- Create network architecture with intuition
- Start with random values for weights and biases
- Minimize error in predicting known outputs from inputs

Training Process

- Adjust weights and biases until error is minimized
- Improve model by adjusting layers, node counts, and other hyper parameters

The Input Layer

Vectors

The input to deep learning is a vector of numeric values

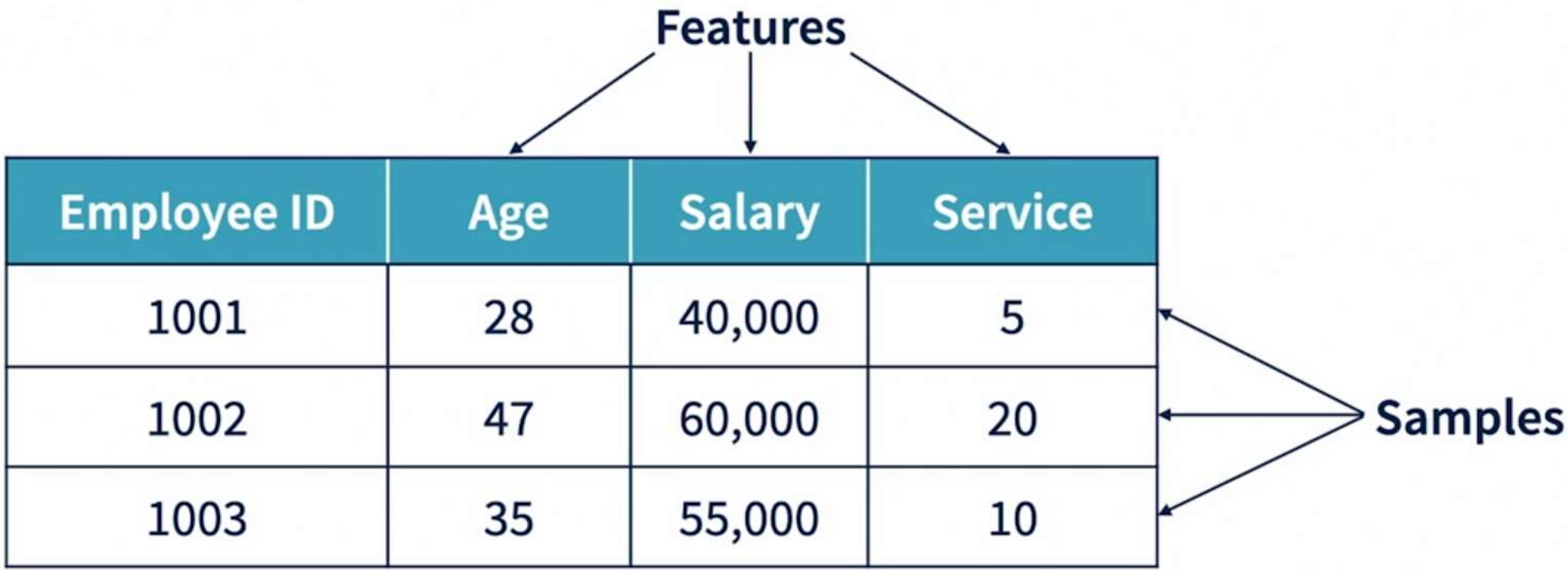
A vector is a tuple of one or more values

- E.g., (1.0, 2.1, 3.5)

Defined usually as a NumPy array

Represents the feature variables for prediction

Samples and Features



Input Preprocessing

Features need to be converted to numeric representations.

Input Type	Preprocessing Needed
Numeric	Centering and scaling
Categorical	Integer encoding, one-hot encoding
Text	TF-IDF, embeddings
Image	Pixels – RGB representation
Speech	Time series of numbers

Samples to Input Vectors

Raw Data

Age	Salary	Service
28	40,000	5
47	60,000	20
35	55,000	10

Centered and Scaled

x1	x2	x3
-1.10	-1.37	-1.07
1.32	0.98	1.34
-0.21	0.39	-0.27

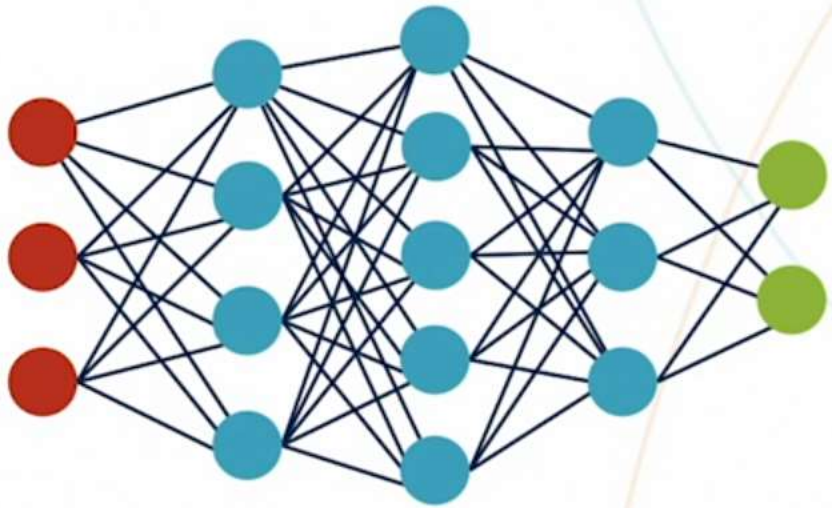
Transposed

x1	-1.1	1.32	-0.21
x2	-1.37	0.98	0.39
x3	-1.07	1.34	-0.27

Hidden Layers

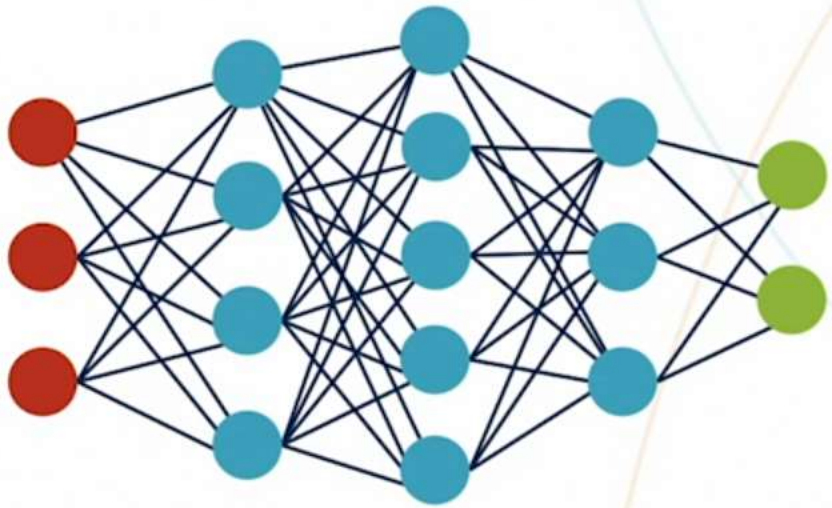
Hidden Layers

- An ANN can have one or more hidden layers
- Each hidden layer can have one or more nodes (or perceptron) (count in 2n)
- A neural network architecture is defined by the number of layers and nodes



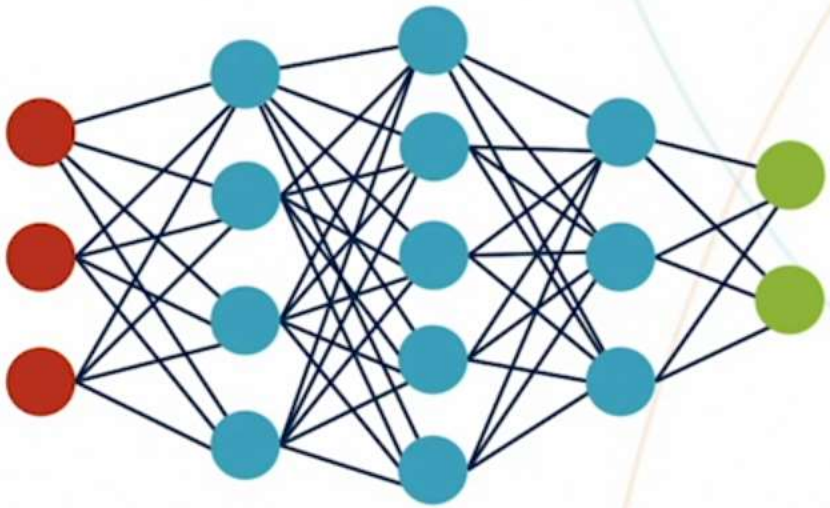
Inputs and Outputs

- The output of each node in the previous layer becomes the input for each node in the current layer (fully connected)
- Each node produces one output that is forwarded to the next layer



Determining Hidden Layer Architecture

Each node *learns* something about the feature-target relationship



More nodes and layers mean

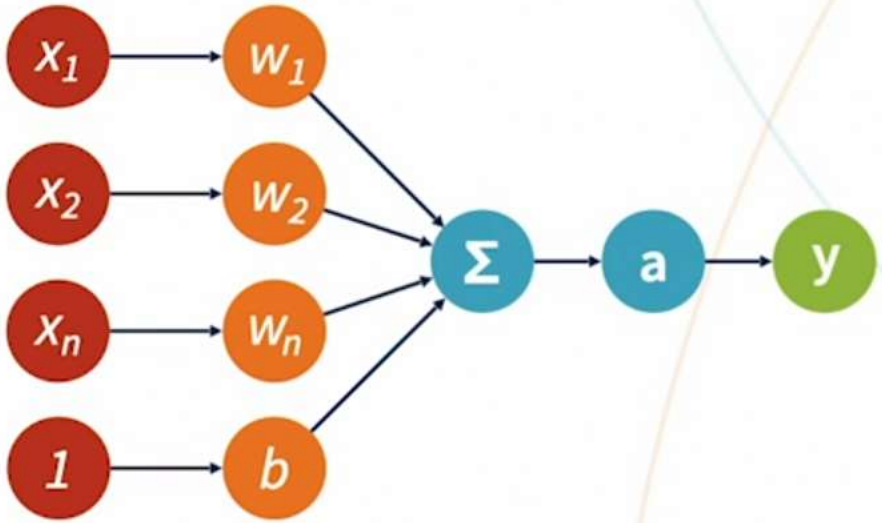
- Web applications on AWS
- Analytics on GCP

Architecture decided by experimentation

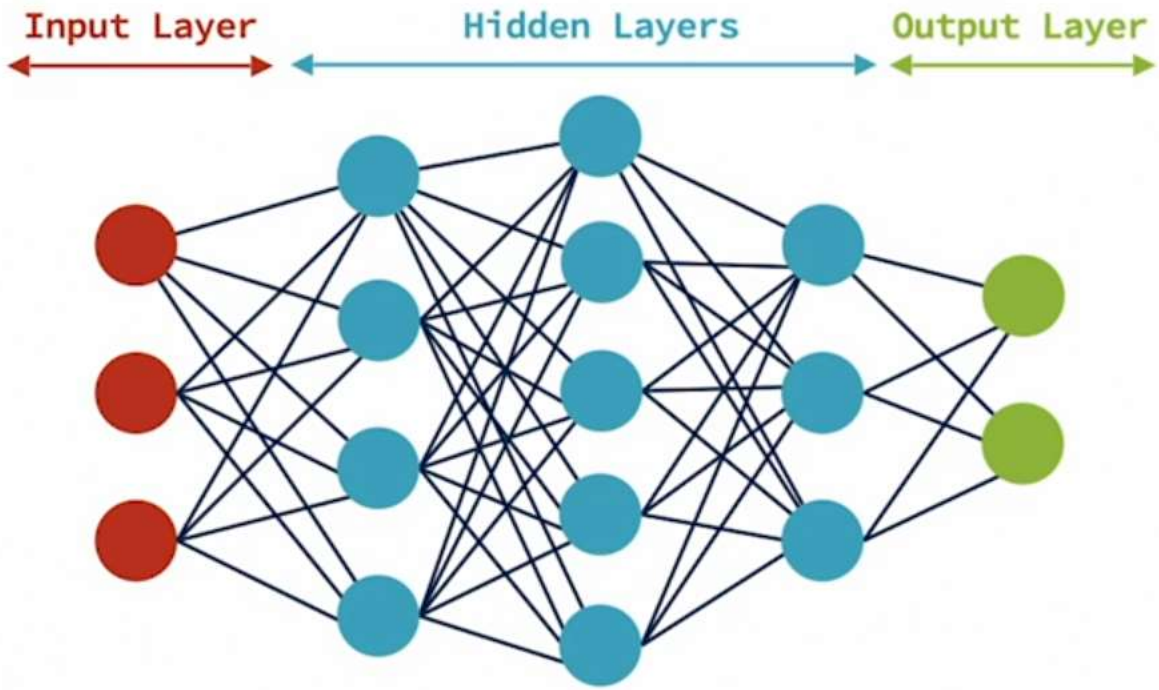
Weights and Biases

Weights and Biases Overview

- Weights and biases represent the trainable parameters in an ANN
- Numeric values
- Each input for each node has a weight associated with it
- Each node has a bias associated with it

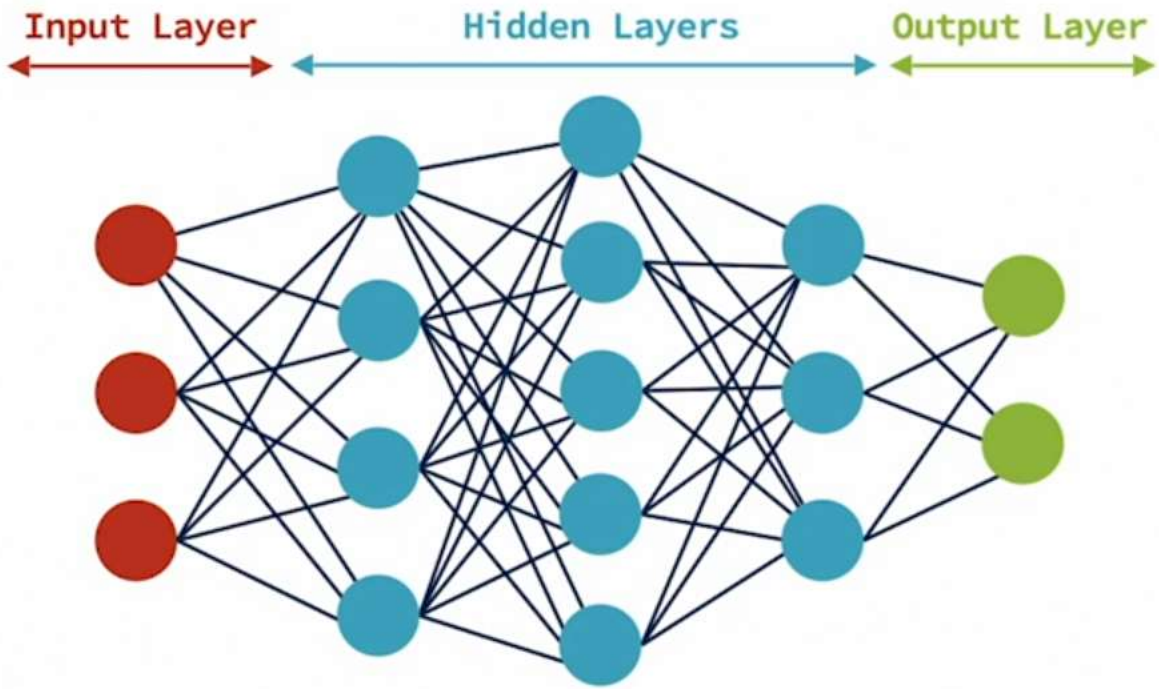


Weights and Biases for a Layer



Layer	Inputs	Nodes	Weights	Biases
HL 1	3	4	12	4
HL 2	4	5	20	5
HL 3	5	3	15	3
Output	3	2	6	2
Total			**53**	**14**

Computing Outputs: Hidden Layer 2



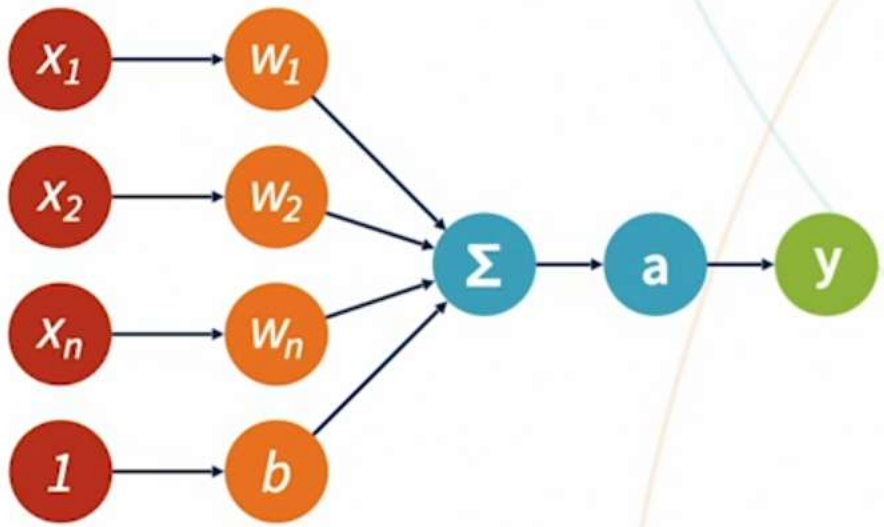
$$\begin{bmatrix} w11 & w21 & w31 & w41 \\ w12 & w22 & w32 & w42 \\ w13 & w23 & w34 & w43 \\ w14 & w24 & w34 & w44 \\ w15 & w25 & w35 & w45 \end{bmatrix} * \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} + \begin{bmatrix} b1 \\ b2 \\ b3 \\ b4 \\ b5 \end{bmatrix} = \begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \\ y5 \end{bmatrix}$$

$$W^T X + B = Y$$

Activation Functions

Activation Functions

- Determines which nodes propagate information to the next layer
- Filters and normalizes
- Converts output to nonlinear
- Critical in learning patterns



Popular Activation Functions

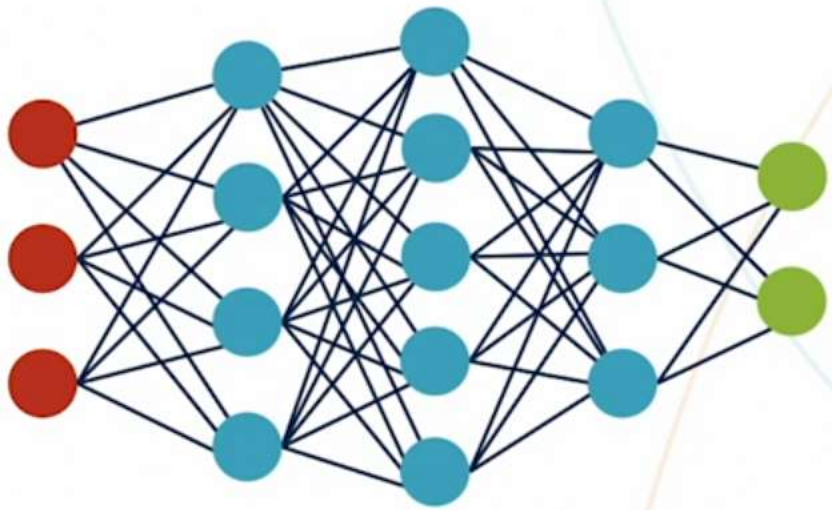
Activation Function	Output
Sigmoid	0 to 1
Tanh	-1 to +1
Rectified Linear Unit (ReLU)	0 if $x < 0$; x otherwise
Softmax	Vector of probabilities, with sum=1

Choice depends on problem and experimentation.

The Output Layer

Output Layer

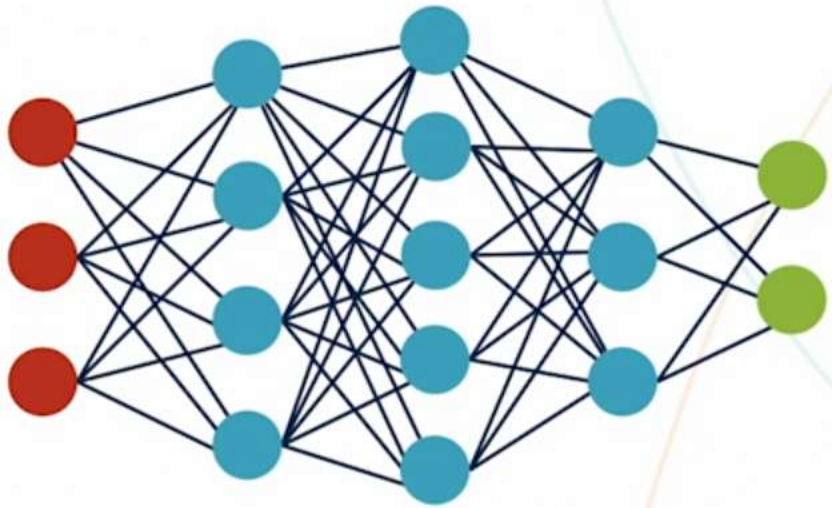
- One layer of output, produces desired Y
- Has its own weights and biases
- Softmax activation used for classification problems
- May need postprocessing to convert to business values



Output Layer Size

Size depends on the problem

- 1 for binary classification
- n for a n-class classification
- 1 for regression problems
- Vary based on other problem domains



Setup and Initialization

Input Preprocessing

	Sample 1	Sample 2	Sample 3	Sample 4
Feature 1	x11	x21	x31	x41
Feature 2	x12	x22	x32	x42
Feature 3	x13	x23	x34	x43
Feature 4	x14	x24	x34	x44
Feature 5	x15	x25	x35	x45
Target	y1	y2	y3	y4

Splitting Input

- Training set: Used to fit the parameters
- Validation set: Used for model selection/tuning
- Test set: Used to measure the final model performance
- Usual split: 80:10:10

Model Architecture and Hyper Parameters

Select values for the model

- Layers and nodes in the layer, activation functions
- Hyper parameters

Selection criteria

- Initial selection based on intuition/reference
- Adjustment based on results

Weights Initialization

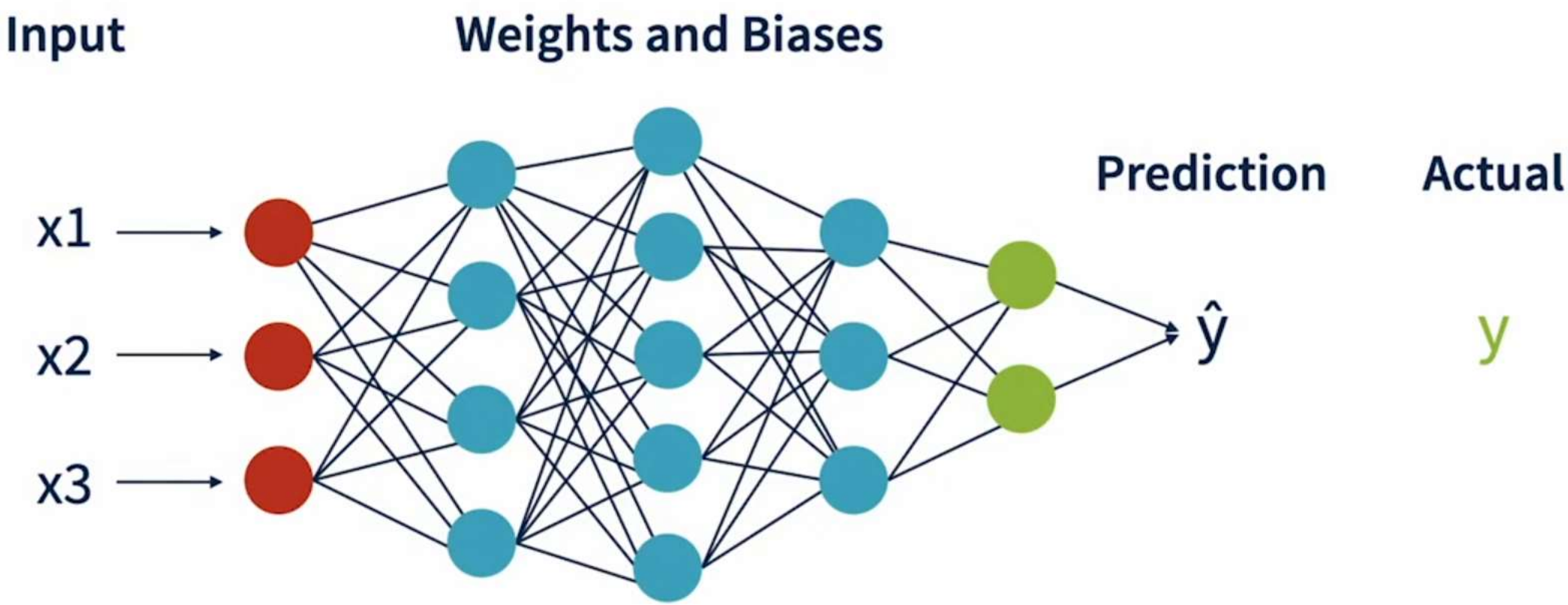
- All weights and bias parameters need to be initialized to some value before we start training
- Zero initialization: Initialize to zeros, not recommended
- Random initialization: Initialize to random values from a standard normal distribution (mean = 0, SD = 1)

Forward Propagation

Inputs, Targets, and Predictions

	Sample 1	Sample 2	Sample 3	Sample 4
Feature 1	x11	x21	x31	x41
Feature 2	x12	x22	x32	x42
Feature 3	x13	x23	x34	x43
Target	y1	y2	y3	y4
Prediction	$\hat{y}1$	$\hat{y}2$	$\hat{y}3$	$\hat{y}4$

Forward Propagation: 1 Sample

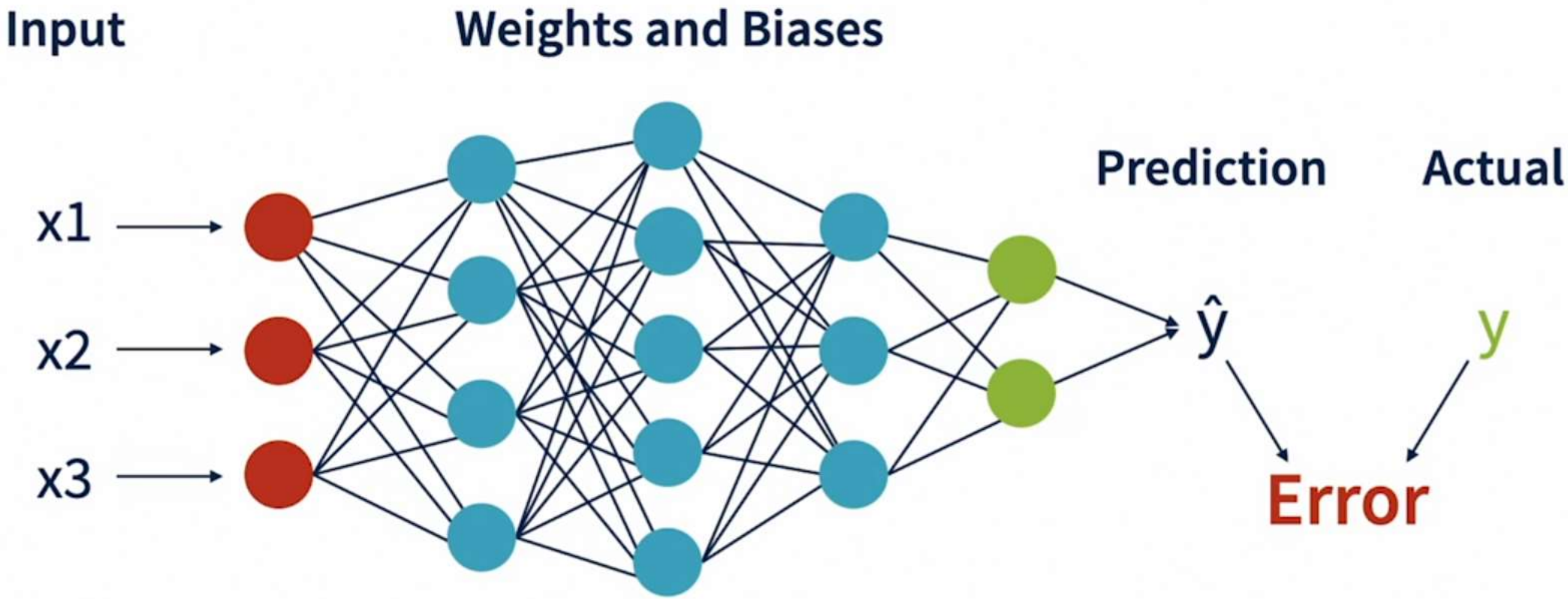


Forward Propagation: All Samples

- Send each sample through the neutral network and obtain the value of $\hat{y}$
- Repeat for all samples and collect a set of $\hat{y}$
- Compare the values of $\hat{y}$ to y to obtain error rates

Measuring Accuracy and Error

Error in Prediction



Loss and Cost Function

- A loss function measures the prediction error for a single sample
- A cost function measures the error across a set of samples

Popular Cost Functions

Cost Functions	Applications
Mean Square Error (MSE)	Regression
Root Mean Square Error (RMSE)	Regression
Binary Cross Entropy	Binary classification
Categorical Cross Entropy	Multi-class classification

Measuring Accuracy

- Send a set of samples through the ANN and predict outcome
- Estimate the prediction error between the predicted outcome and expected outcome using a cost function
- Use back propagation to adjust weights based on the error value

Back Propagation

Why Back Propagate?

- Each node in a neutral network contributes to the overall error in prediction (differing contributions)
- A node's contribution is driven by its weights and bias
- Weights and biases need to be adjusted to lower the error contribution by each node

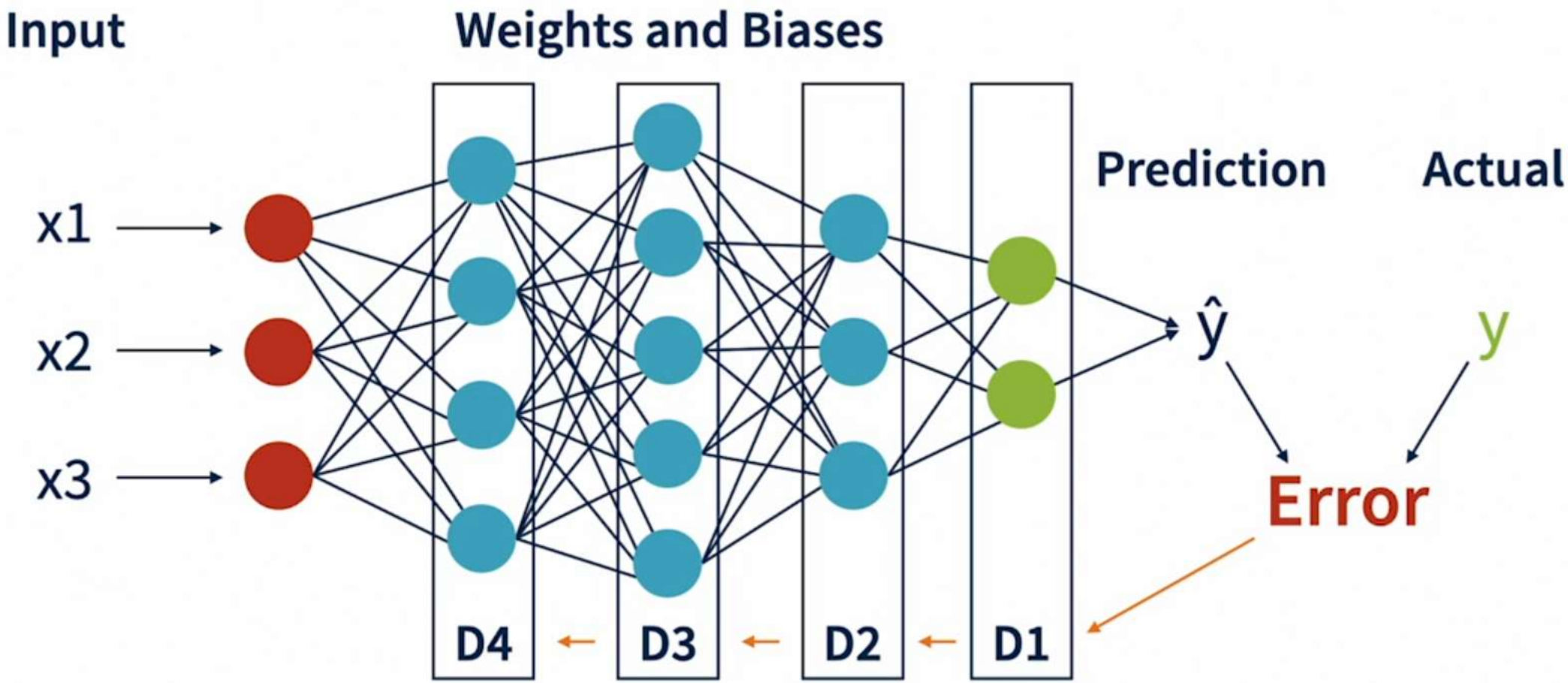
How Back Propagation Works

- Back propagation works in reverse of the forward propagation
- Start from the output layer
- Compute a delta value based on the error found
- Apply the delta to adjust the weights and biases in the layer

How Back Propagation Works

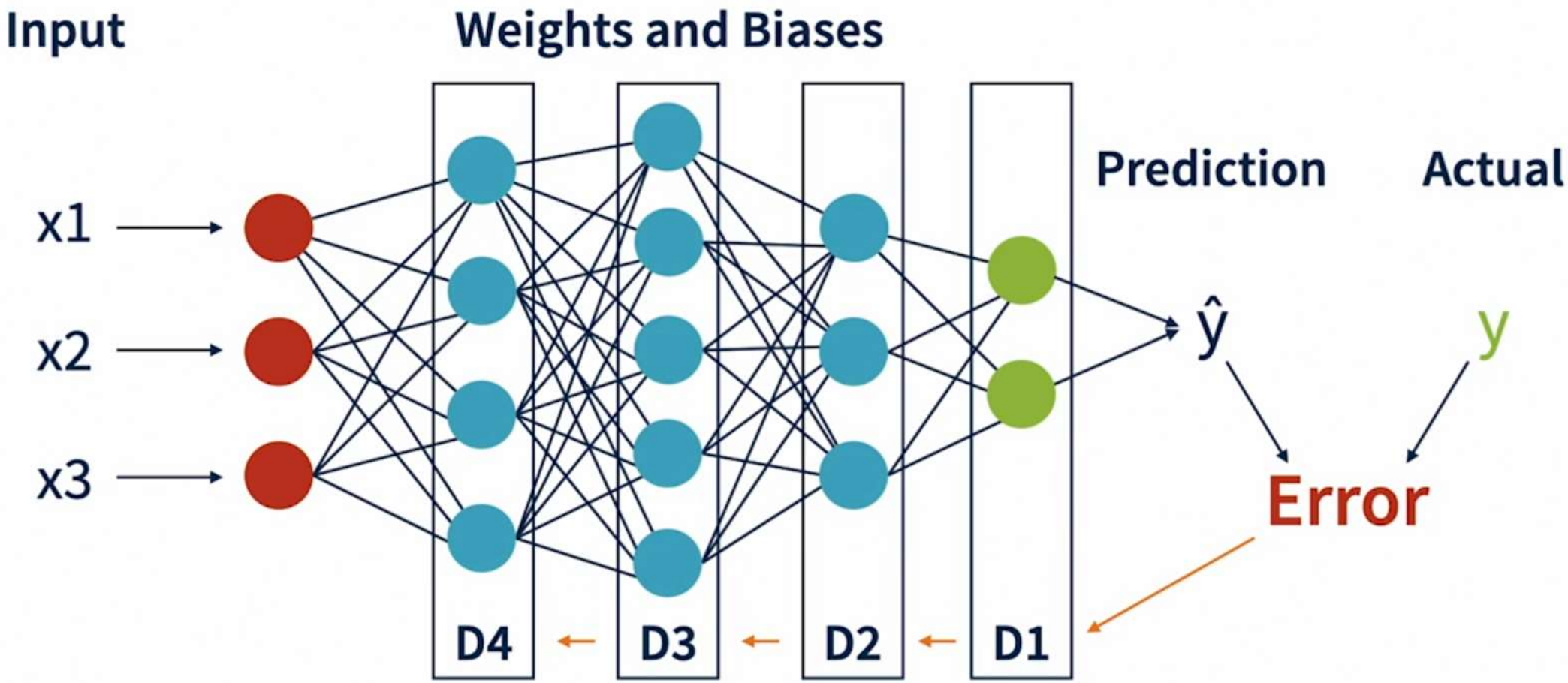
- Derive a new error value
- Back propagate the new error to the previous layer and repeat

Back Propagation



Gradient Descent

Back Propagation



Gradient Descent Process

Repeat the learning process.

- Forward propagation
- Estimate error
- Backward propagate
- Adjust weights and biases

Batches and Epoch

Batch

- A set of samples sent through ANN in a single pass
- The training data set can be divided into one or more batches
- Training data is sent to the ANN one batch at a time
- Cost estimated and parameters updated one batch at a time

Batch

- Batch gradient descent: batch size = training set size
- Mini-batch gradient descent: batch size < training set size
- Typical batch sizes are 32, 64, 128, etc.

Epoch

- The number of times the entire training set is sent through the ANN
- An epoch has one or more batches
- The training process completes when all epoch is complete
- Epoch sizes can be higher to achieve better accuracy

Epoch and Batch Example

- Training set size = 1000, batch size = 128, epoch = 50
- Batches per epoch = ceil (1000 / 128) = 8
- Total iterations (passes) through ANN = 8 * 50 = 400
- Batch size and epoch are hyperparameters that can be tuned to improve model accuracy

Validation and Testing

Learning Process: Validation

- During learning, the predictions are obtained for the same data that is used to train the parameters (weights and biases)
- After each epoch and corresponding parameter updates, the model can be used to predict for the validation data set
- Accuracy and/or loss can be measured and investigated
- Model can be fine-tuned and learning process repeated based on results

Evaluation

- After all fine-tuning is completed and final model obtained, the test data set can be used to evaluate the model
- Results obtained with test data set is used to measure the performance of the model

What is an ANN model and what does it contain?

ANN Model

Parameters

- Weights
- Biases

Hyperparameters

- Number of layers, nodes in each layer, activation function
- Cost functions, learning rate, optimizers
- Batch size, epoch

Prediction Process

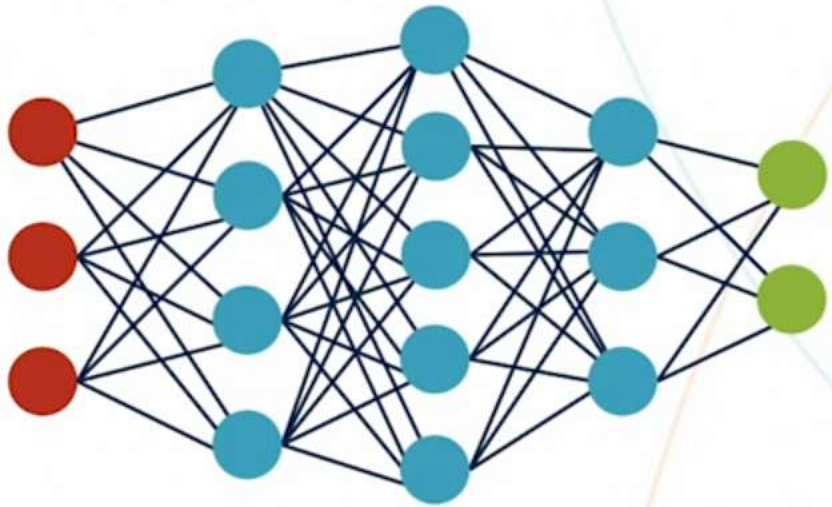
Preprocess and prepare inputs

Pass inputs to the first layer

- Compute Y using weights, biases, activation
- Pass to the next layer

Repeat process until output layer

Postprocess output for predictions



The Iris Classification Problem

Feature Variables

- Sepal length
- Sepal width
- Petal length
- Petal width

Build an ANN model to predict species.

Image Recognition

Keras

- Keras is a high-level library for building neural networks in Python with only a few lines of code

A Simple Neural Network

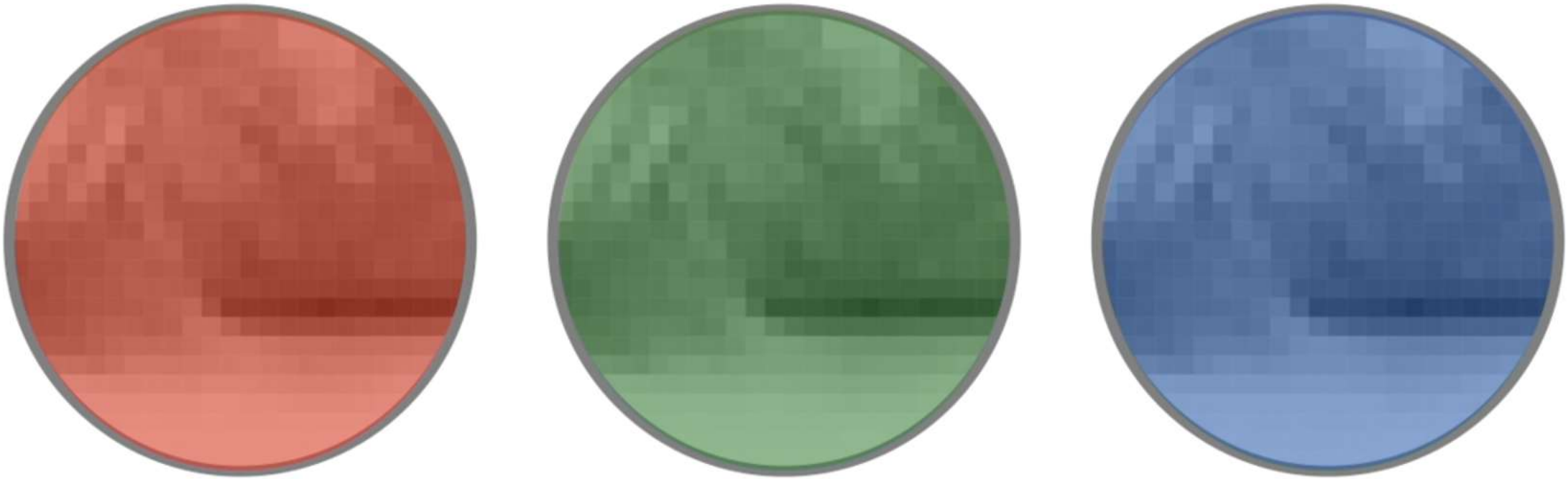
```
model = keras.models.Sequential()
model.add(Dense(3, input_dim=2, activation='relu'))
model.add(Dense(3, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

Using Images as Input to a Neural Network

Digital Images

Digital Images

Digital Images



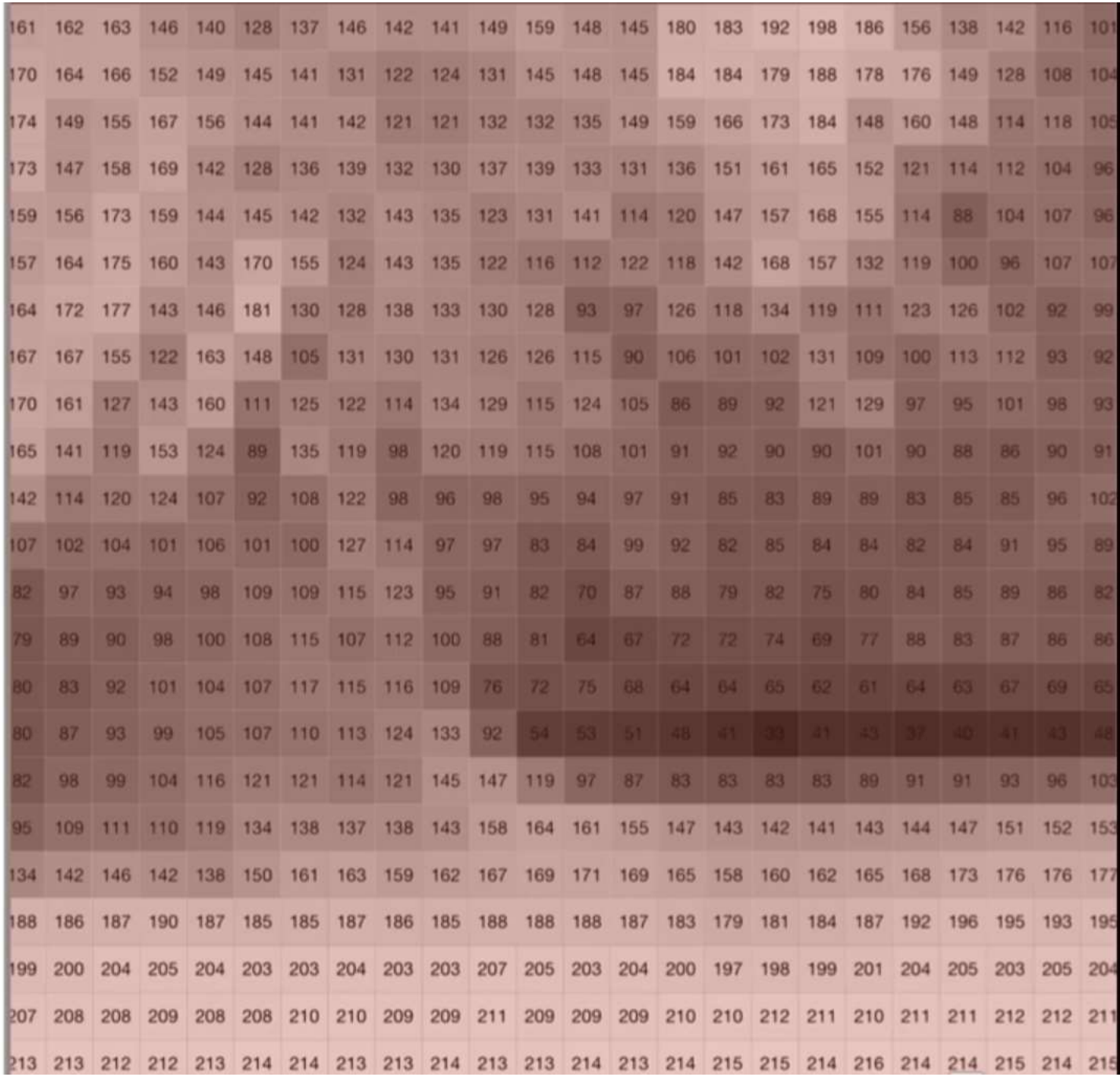
Red channel

Green channel

Blue channel

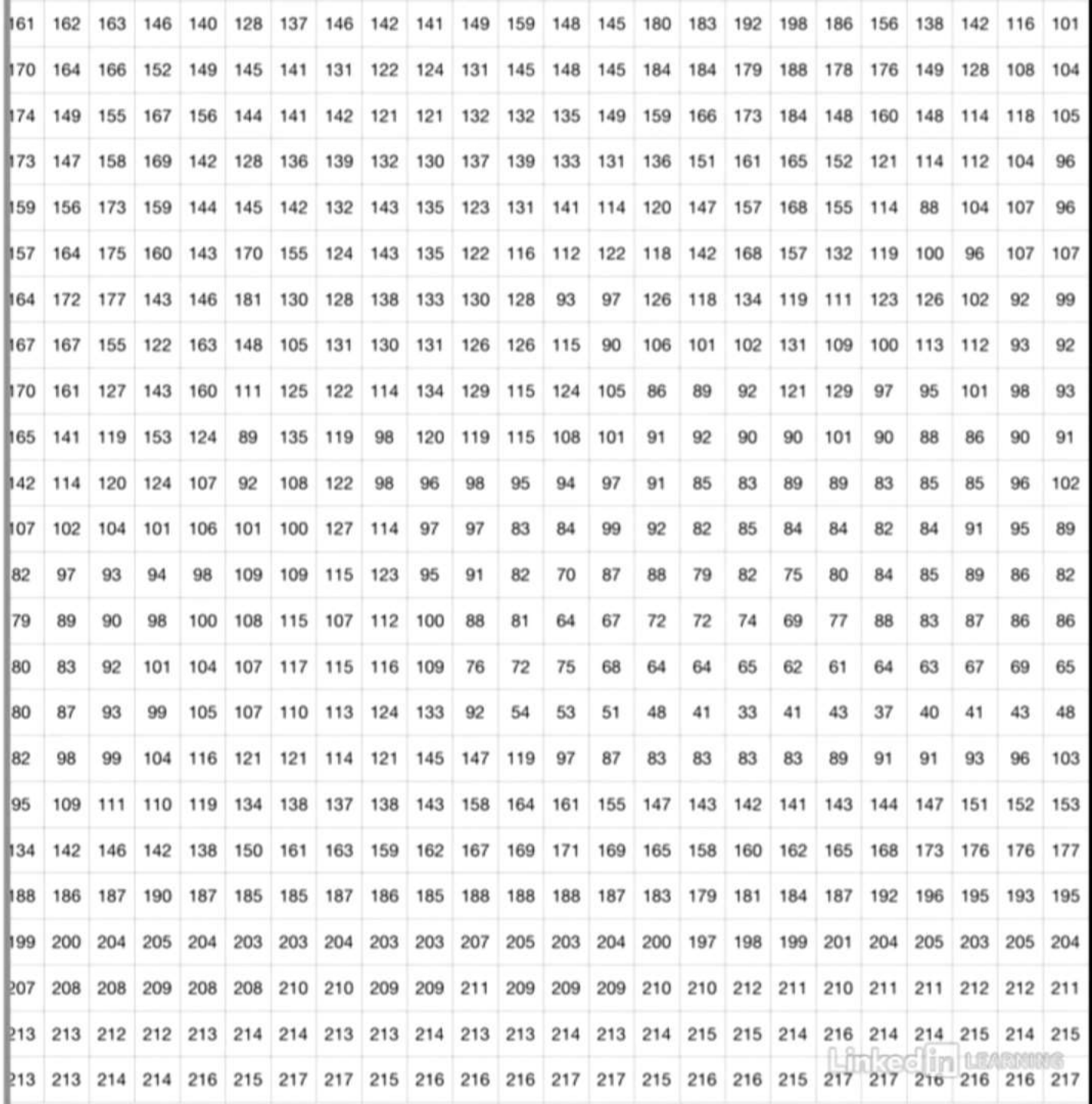
Red Channel

- Red channel =



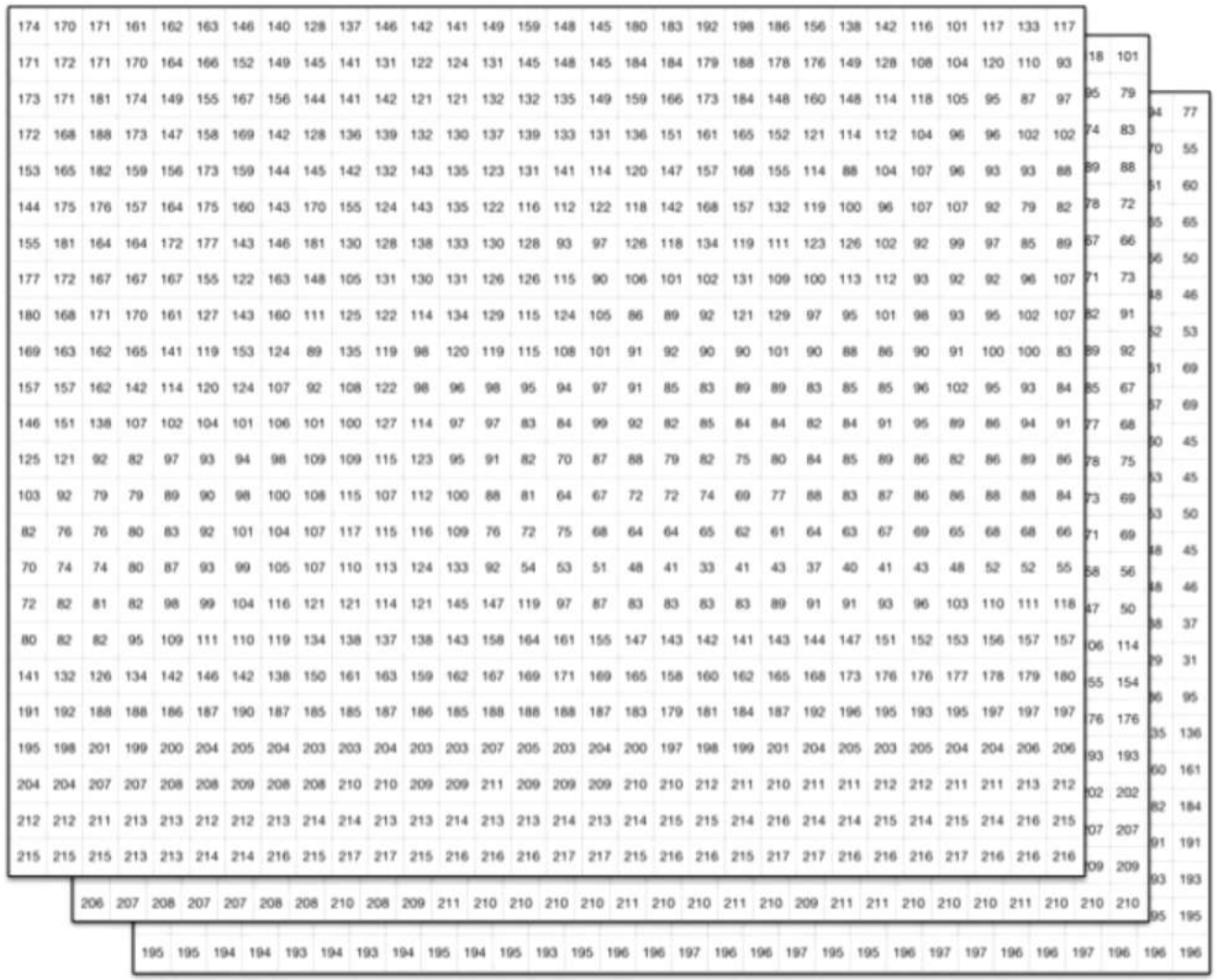
Red Channel

- Red channel =



Full Image

- Image (Three color channels) =



Number of Neural Net Input Nodes

- For a small 256x256 pixel image
- 256 x 256 x 3 = 196,608 input nodes required
- And that's just in the input layer!

Working with Image Data

- Images are computationally intensive
- Image recognition systems tend to use small image sizes
- Images are scaled down to that smaller size before being fed into a neural network

Recognizing Image Contents with a Neural Network

Training Data Set

"8"

61.png 84.png 110.png 128.png

179.png 181.png 184.png 226.png

257.png 260.png 266.png 268.png

338.png 344.png 355.png 373.png

Not "8"

34.png 51.png 54.png 75.png

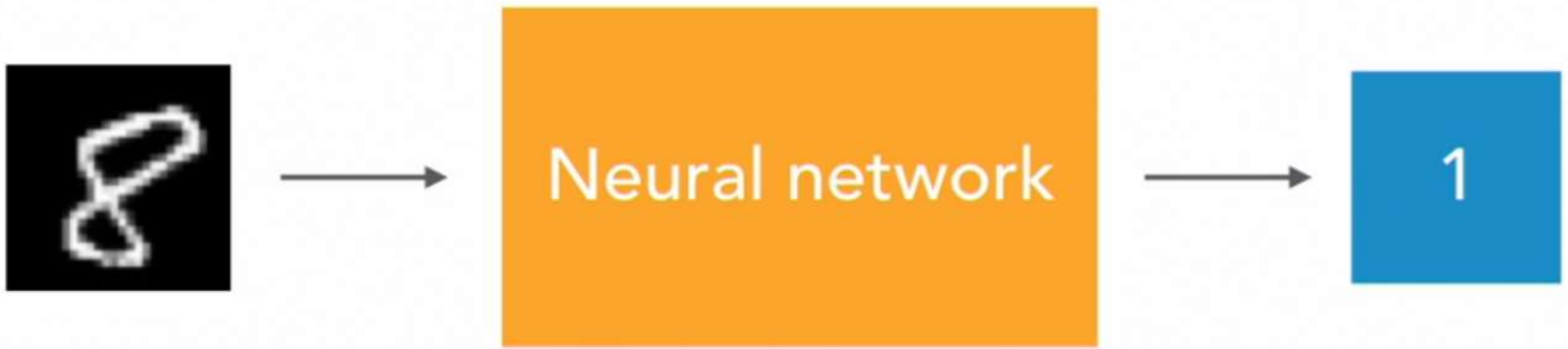
100.png 104.png 111.png 115.png

118.png 135.png 154.png 156.png

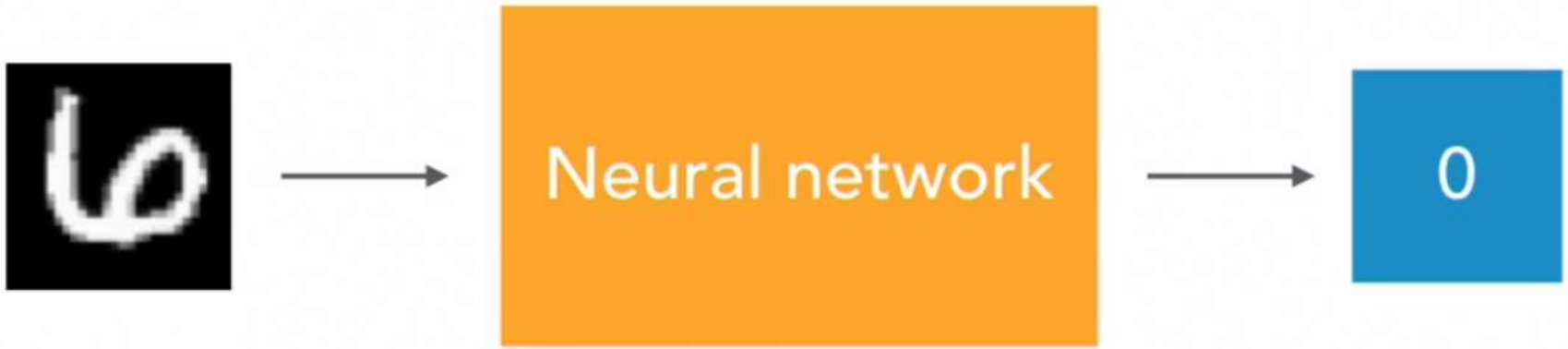
165.png 170.png 187.png 199.png

Training Phase

Training Phase



Training Phase

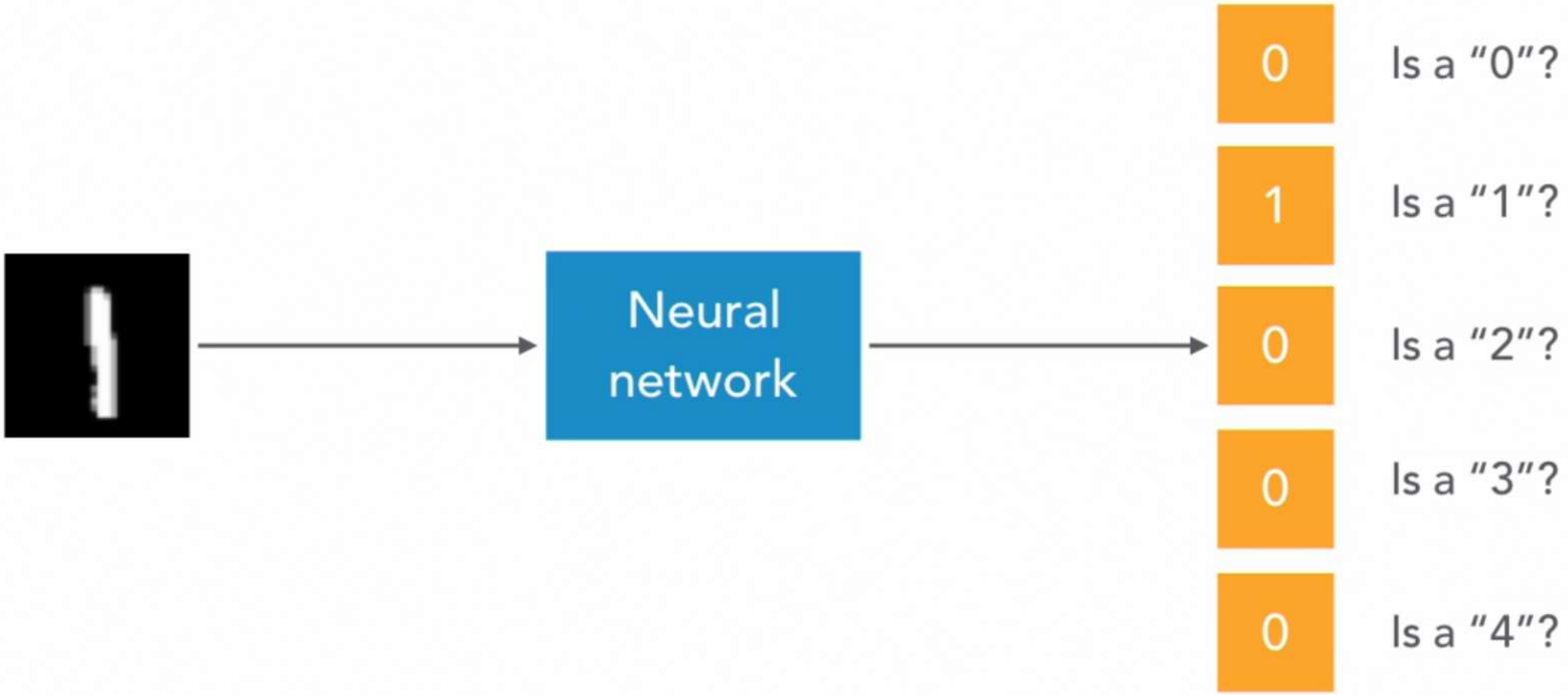


Inference Phase

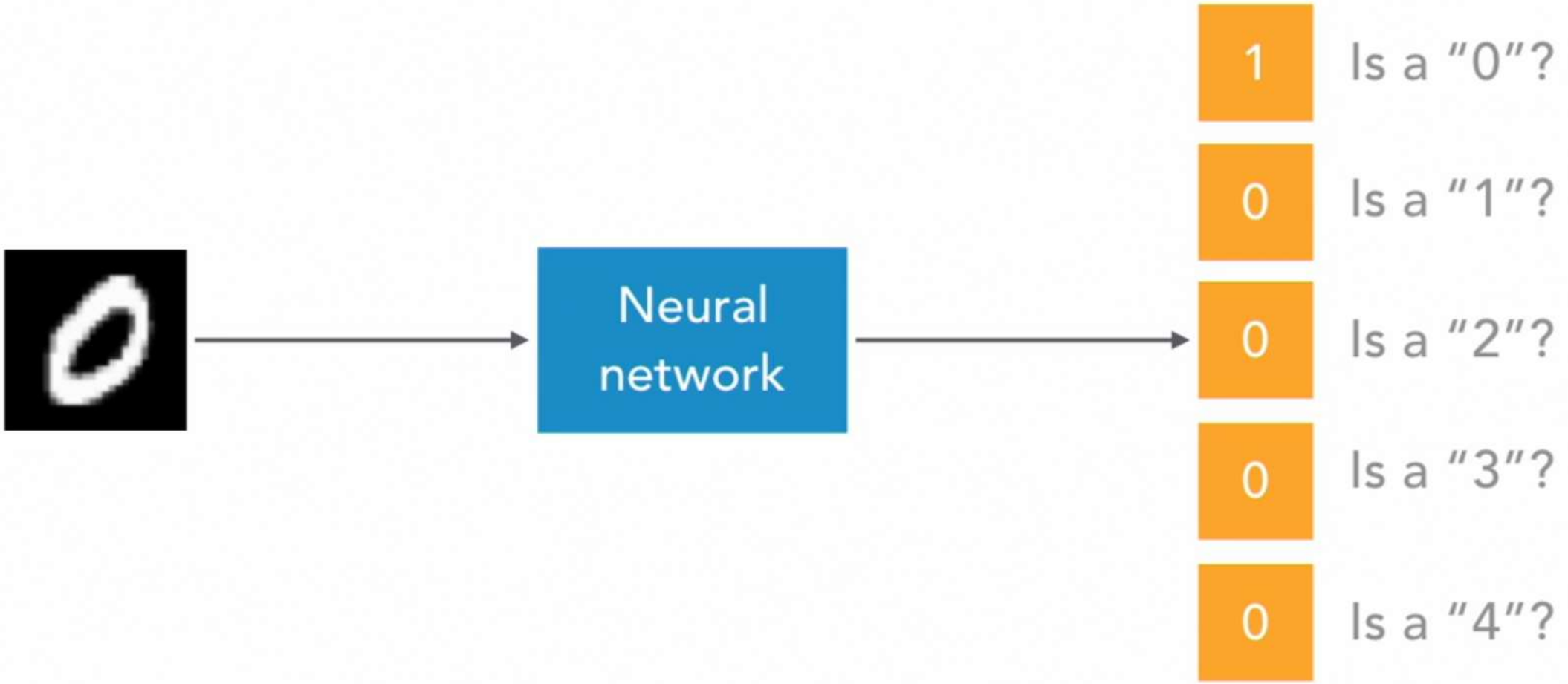
Inference Phase

Inference Phase

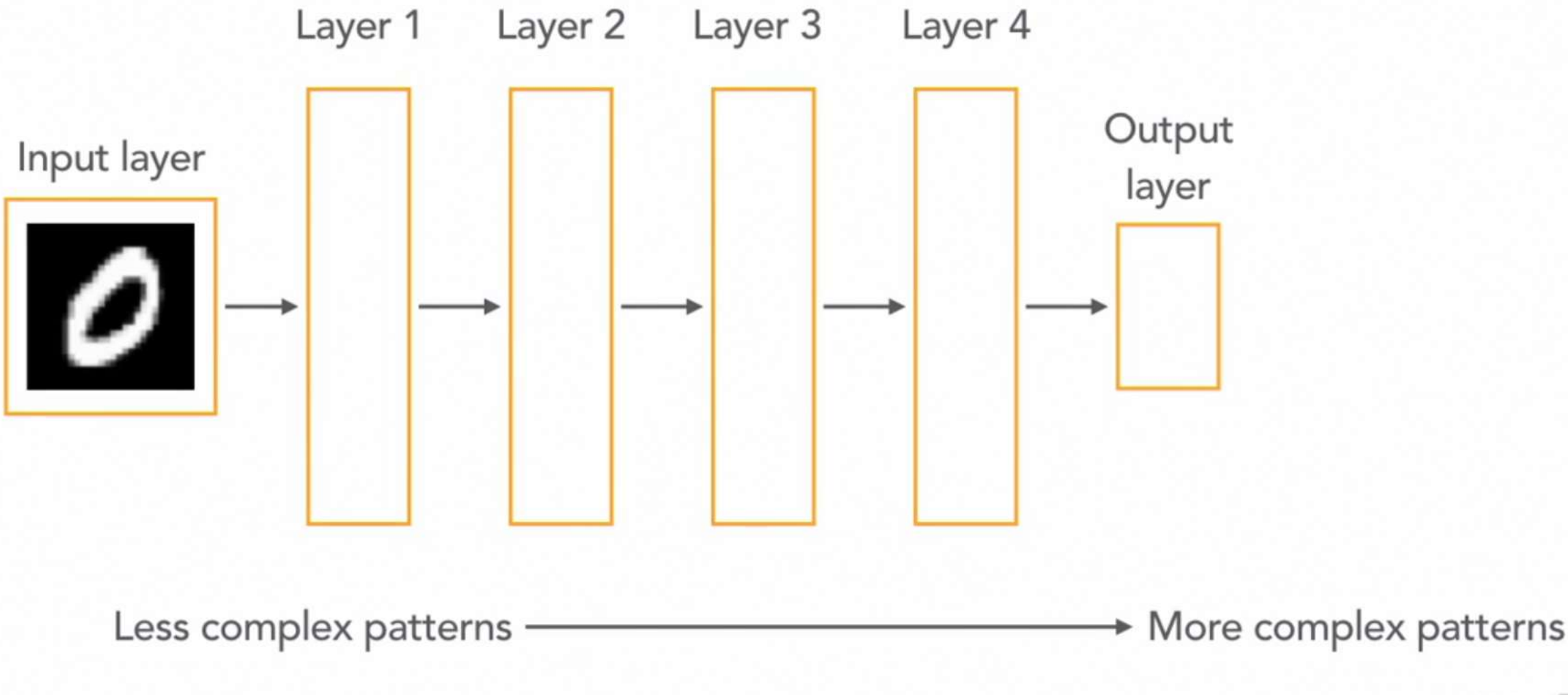
Training Phase



Training Phase



Deep Neural Networks



Adding Convolution for Translational Invariance

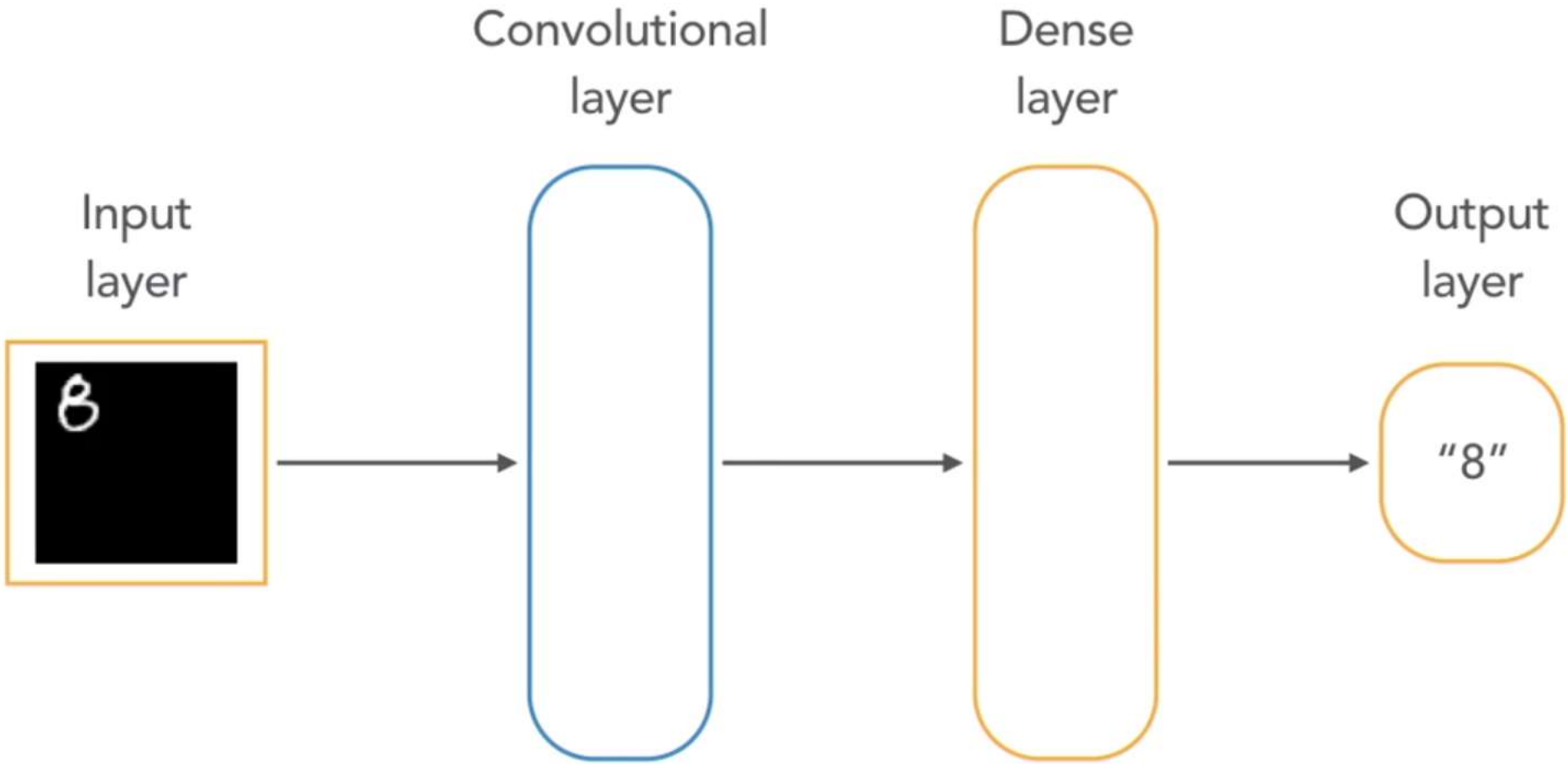
Traditional Image Classification

Traditional Image Classification

Translation invariance

When a machine learning model can recognize an object no matter whether it is moved (or *translated*) in the image

Convolutional Neural Network



Convolution: Step 1

Input image

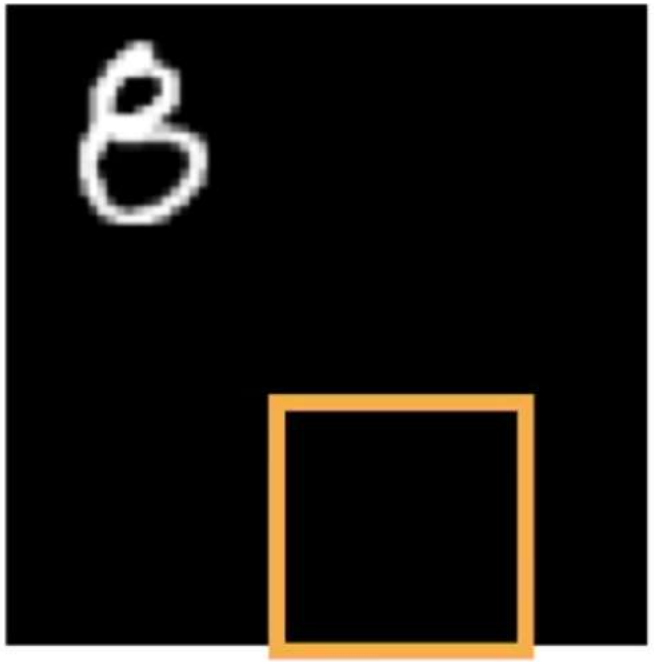
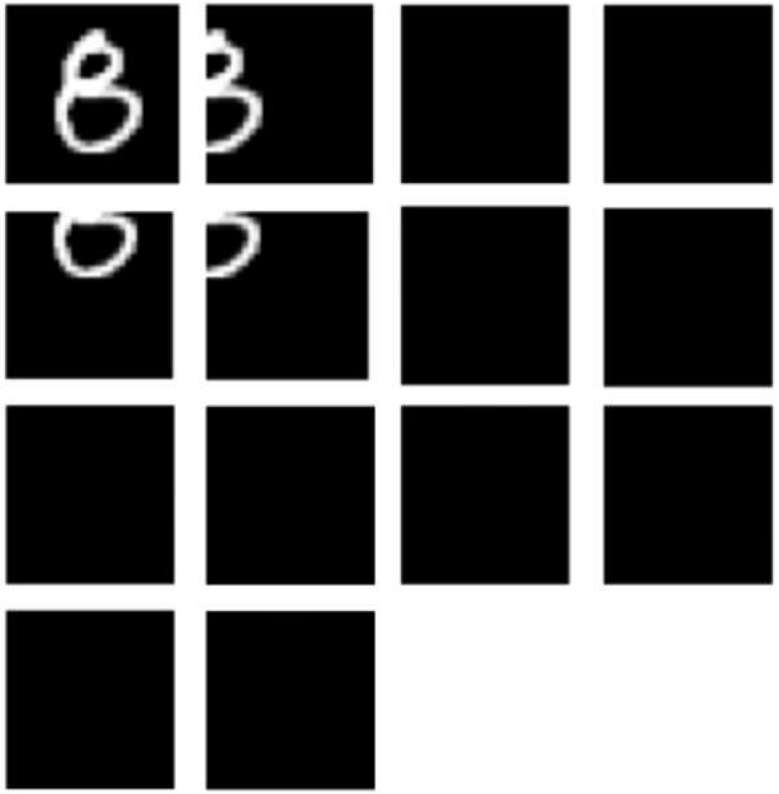


Image tiles



Convolution: Step 1

Input image

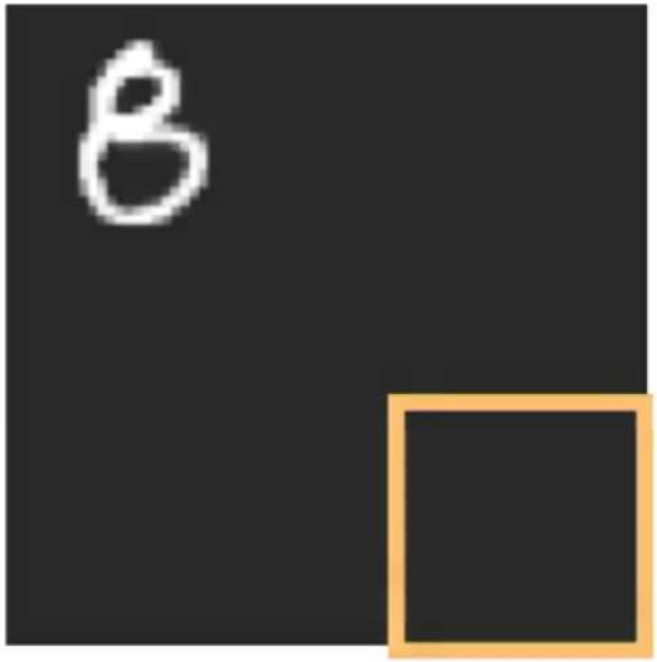
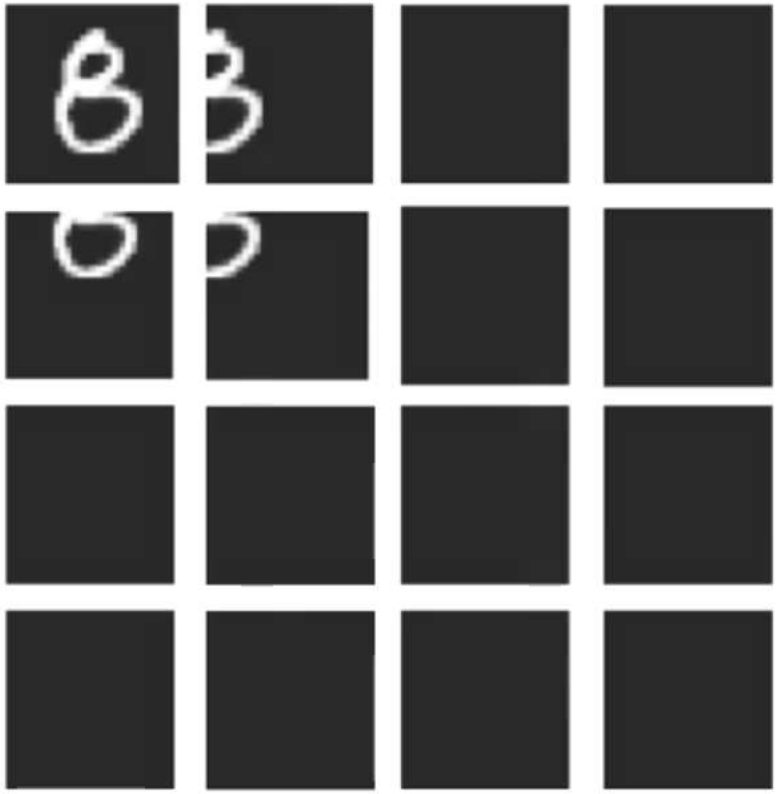
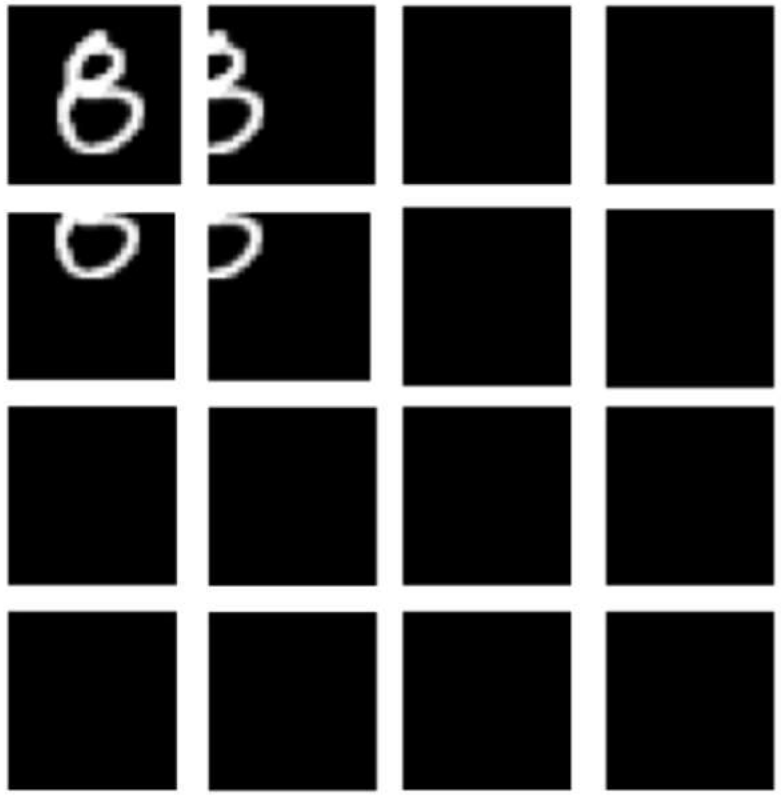


Image tiles



Convolution: Step 2

Image tiles

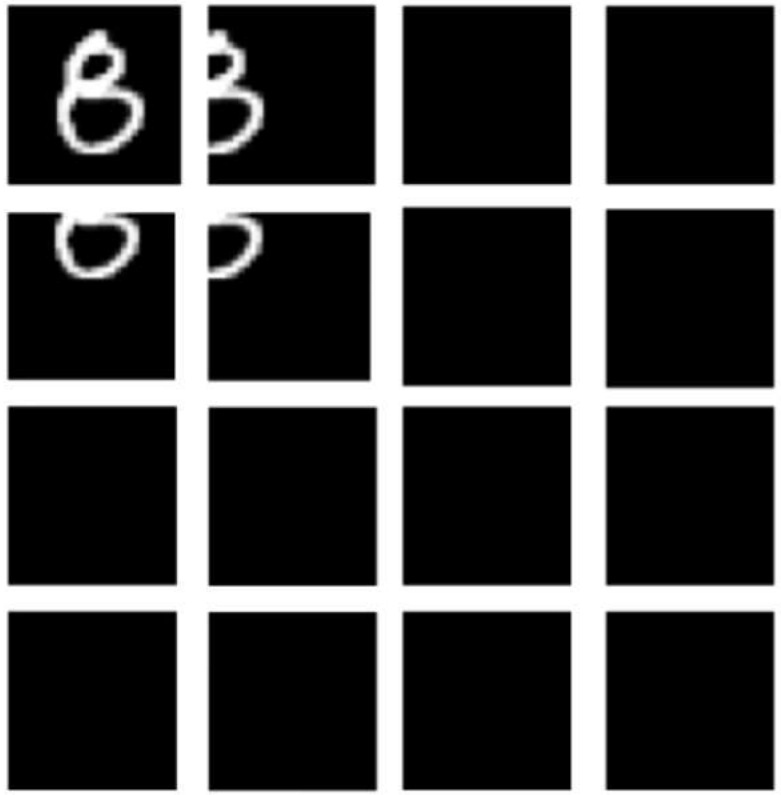


Neural network layer

0.12	0.85	0.0	0.0
0.76	0.321	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

Convolution: Step 3

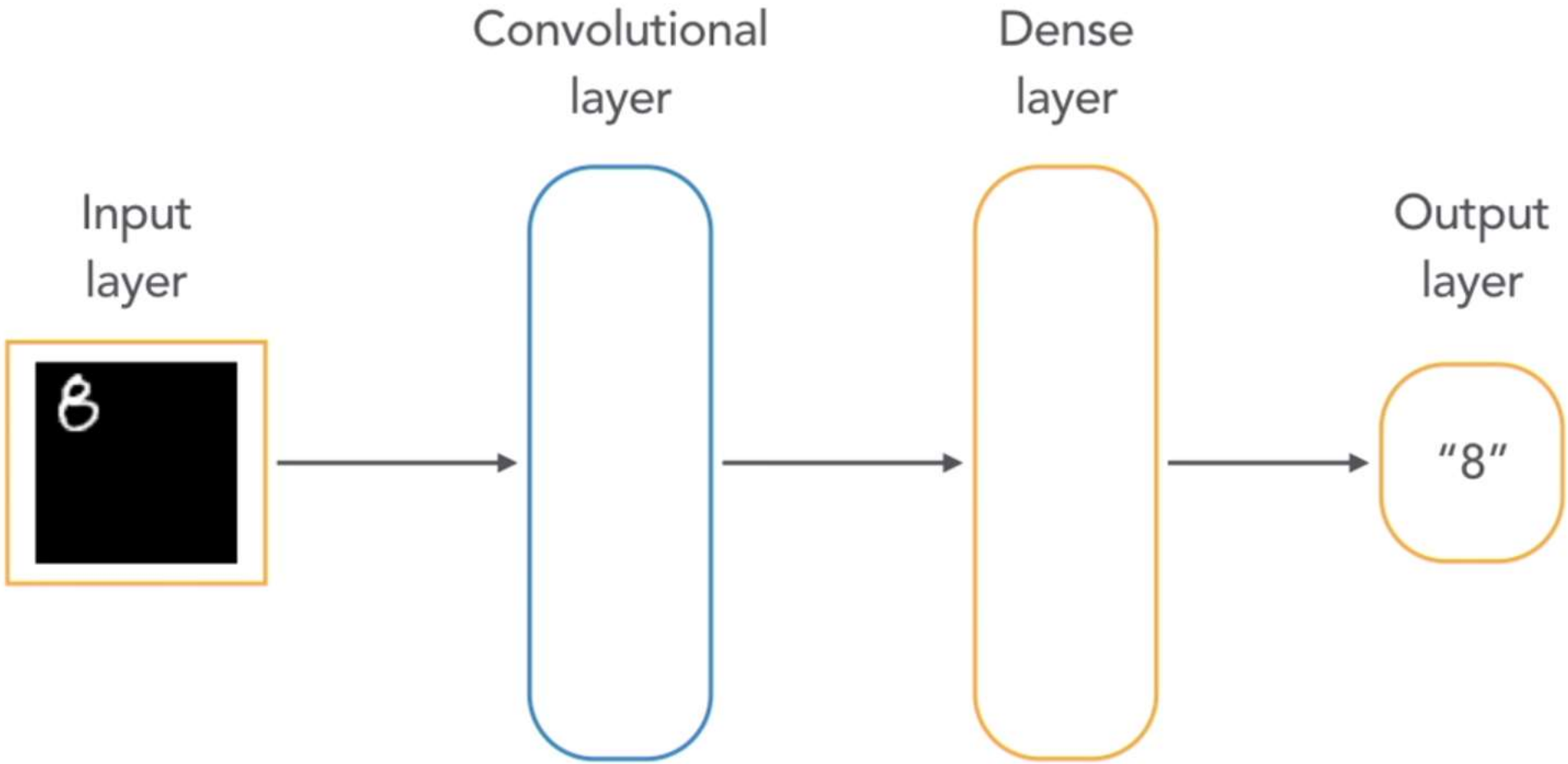
Image tiles



Neural network layer

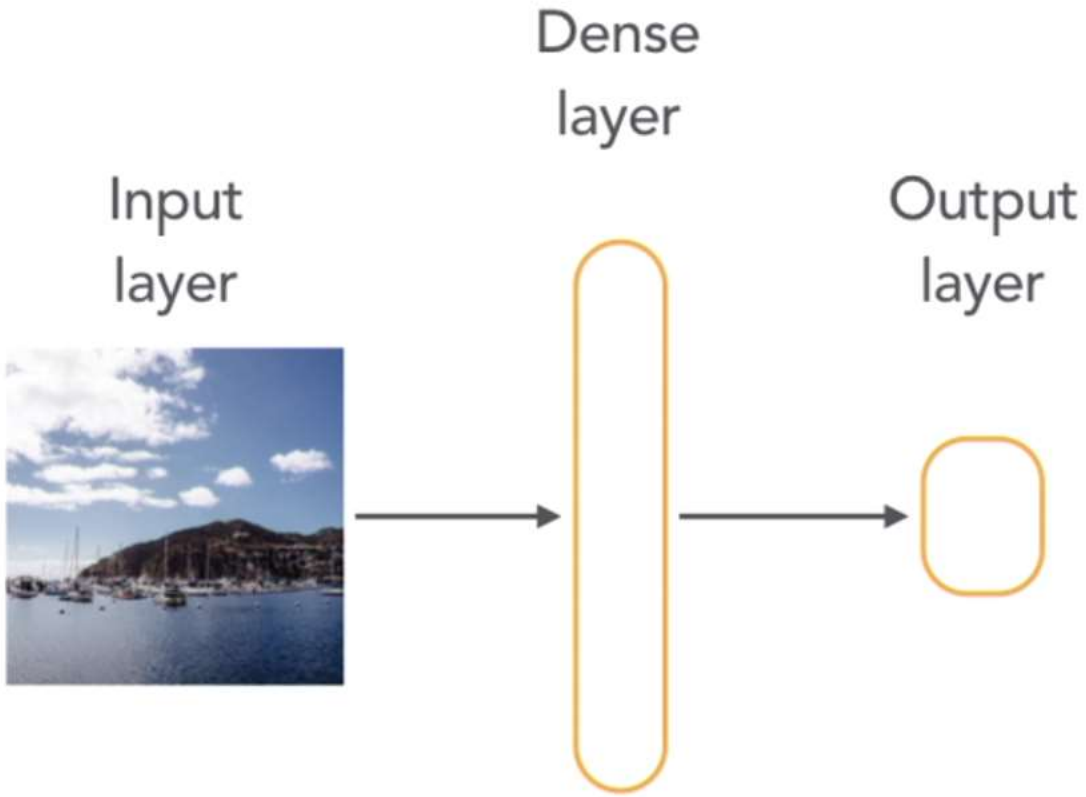
0.45	0.54	0.11	0.11
0.12	0.43	0.11	0.11
0.11	0.11	0.11	0.11
0.11	0.11	0.11	0.11

Convolutional Neural Network

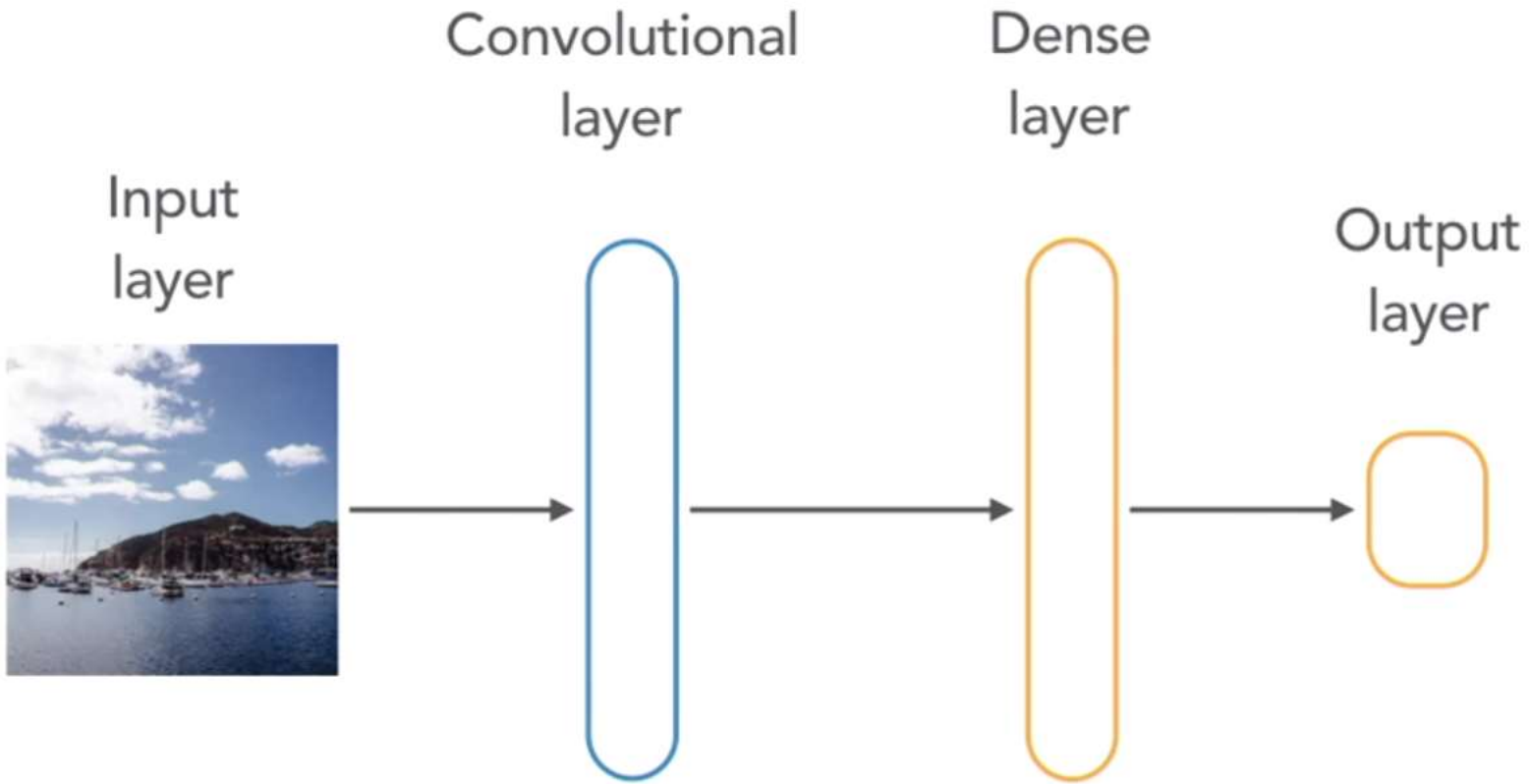


Designing a Neural Network Architecture for Image Recognition

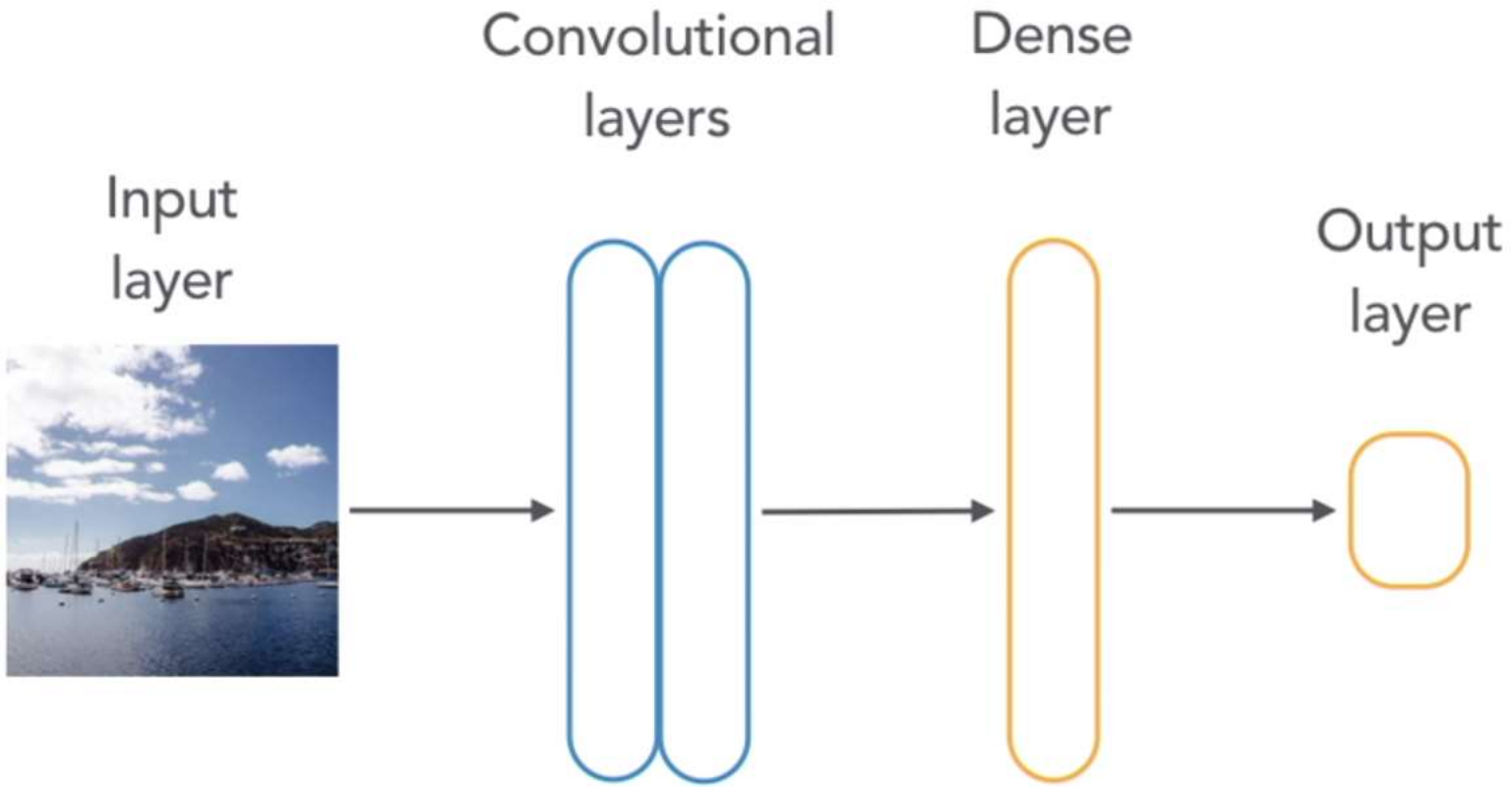
Basic Neural Network



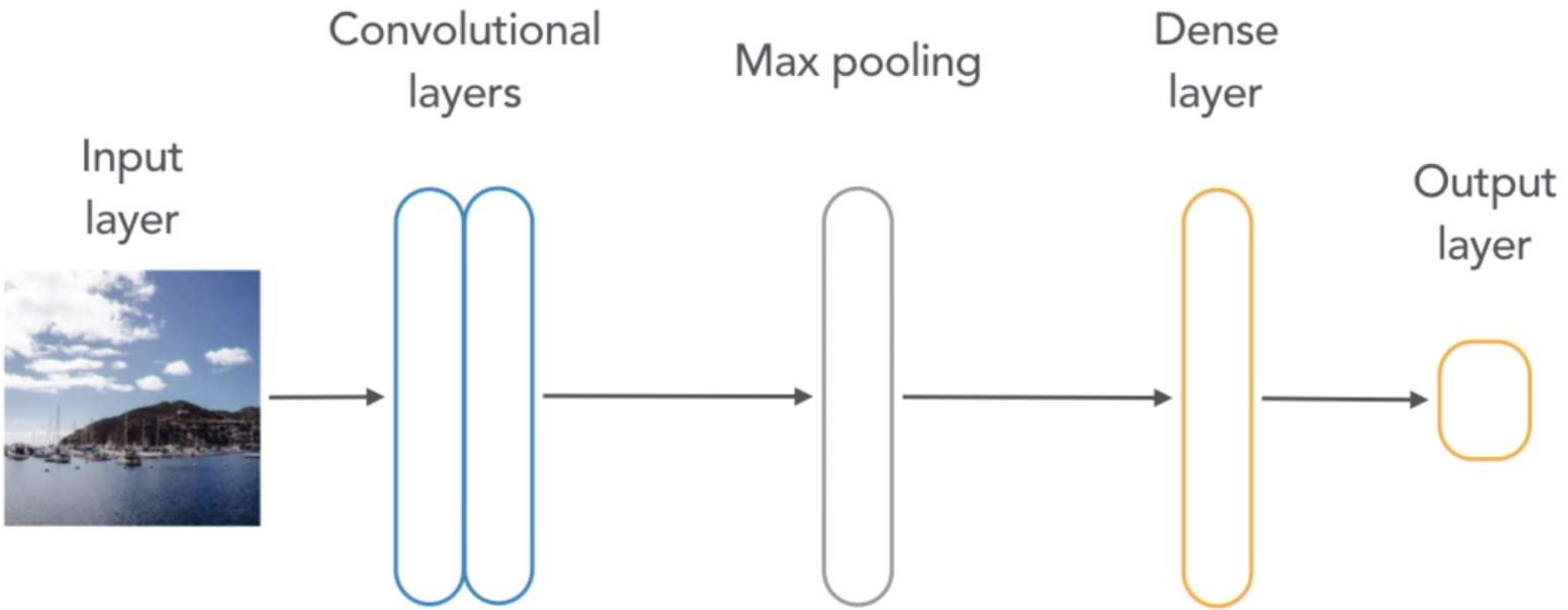
Convolutional Neural Network



Convolutional Neural Network



Convolutional Neural Network



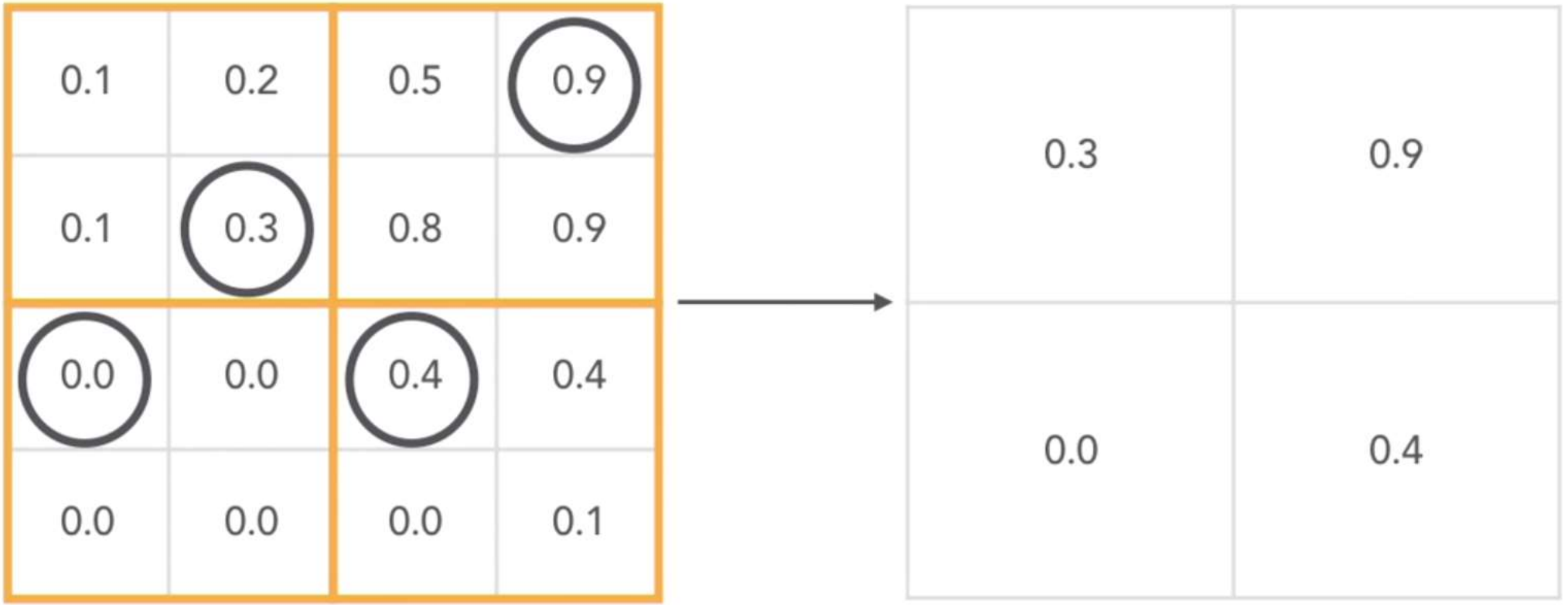
Max Pooling

0.1	0.2	0.5	0.9
0.1	0.3	0.8	0.9
0.0	0.0	0.4	0.4
0.0	0.0	0.0	0.1

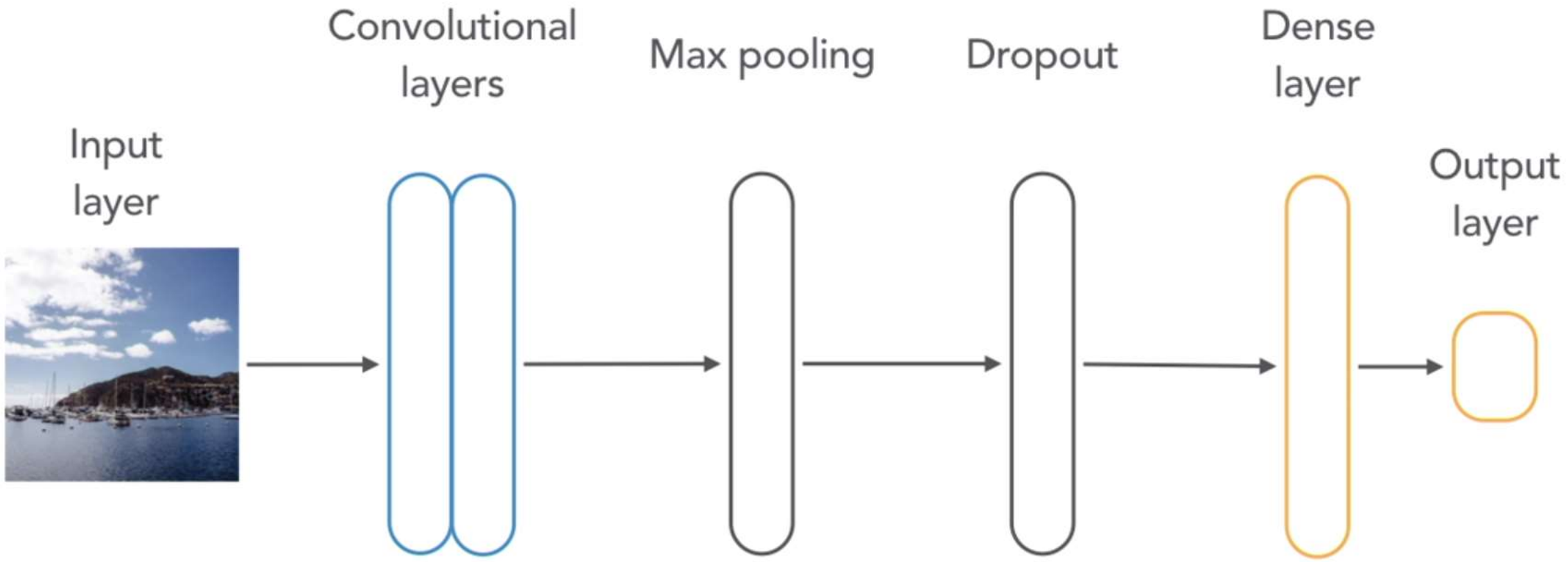
Max Pooling

0.1	0.2	0.5	0.9
0.1	0.3	0.8	0.9
0.0	0.0	0.4	0.4
0.0	0.0	0.0	0.1

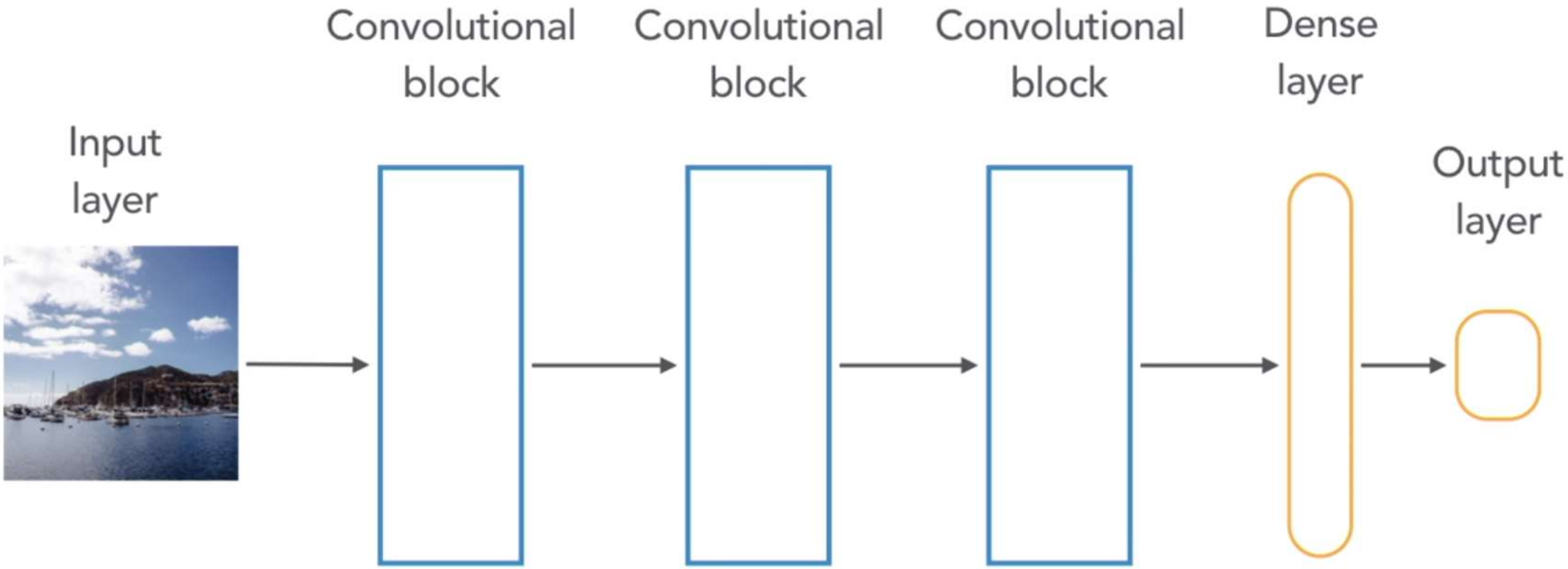
Max Pooling



Convolutional Neural Network



Convolutional Neural Network



Chrome File Edit View History Bookmarks People Window Help

CIFAR-10 and CIFAR-100 data

Secure | https://www.cs.toronto.edu/~kriz/cifar.html

< Back to Alex Krizhevsky's home page

The CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:

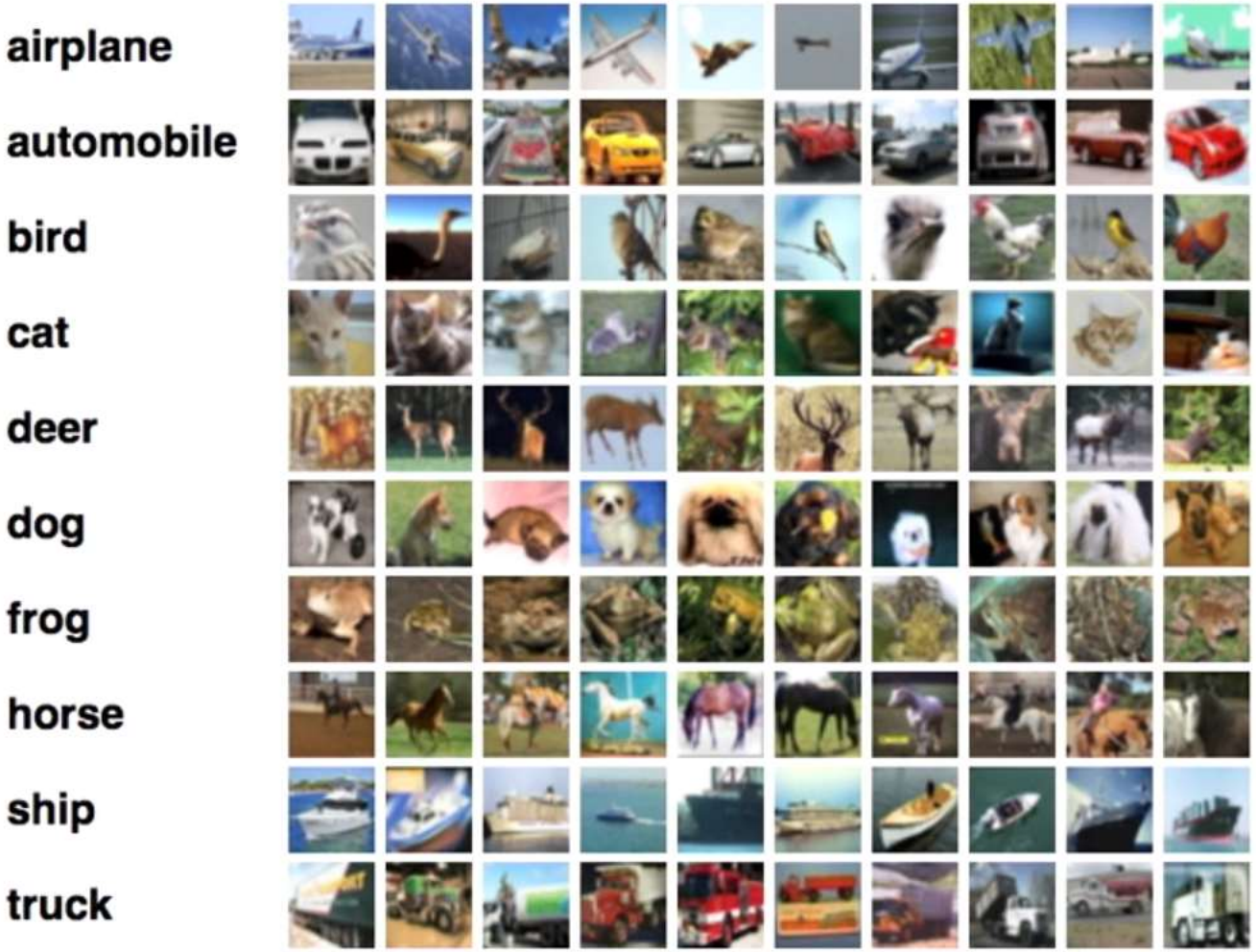
airplane

automobile

bird

cat

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

Exploring Your Data Set

- Look through the data by hand
- Check for obvious errors
- Verify that the data makes sense

CIFAR-10 Specifications

- Images are 32x32 pixels, with 2 color channels
- 10 different types, or classes, of objects
- 60,000 total images
- 5,000 training images and 1,000 test images per class