Chris Parrish
chris@twenty3.me
@twenty3

EXPLORING NEURAL NETWORKS

FOR SHAPE DETECTION

OVERVIEW

- Motivation
- Introduction to Neural Networks
- Building a Prototype
- What’s next?

MOTIVATION

ARTIFICIAL NEURAL NETWORKS

EVOLUTION IN USER INTERFACE

- Macintosh popularized Point and Click
- iPhone brought tap, swipe and pinch
- What’s next?

NEXT GENERATION INTERFACE

- Location and Positioning - GPS, iBeacon/Bluetooth LE, Magnetic Positioning
- Voice Assistants - Siri, Alexa, Cortona, Google Now
- Haptics, Handwriting, Drawing, Advanced Gestures

ARTIFICIAL INTELLIGENCE / MACHINE LEARNING

- Require some component of interpreting intent or meaning from the input signals
- Natural Language Understanding
- Computer Vision
- Prediction

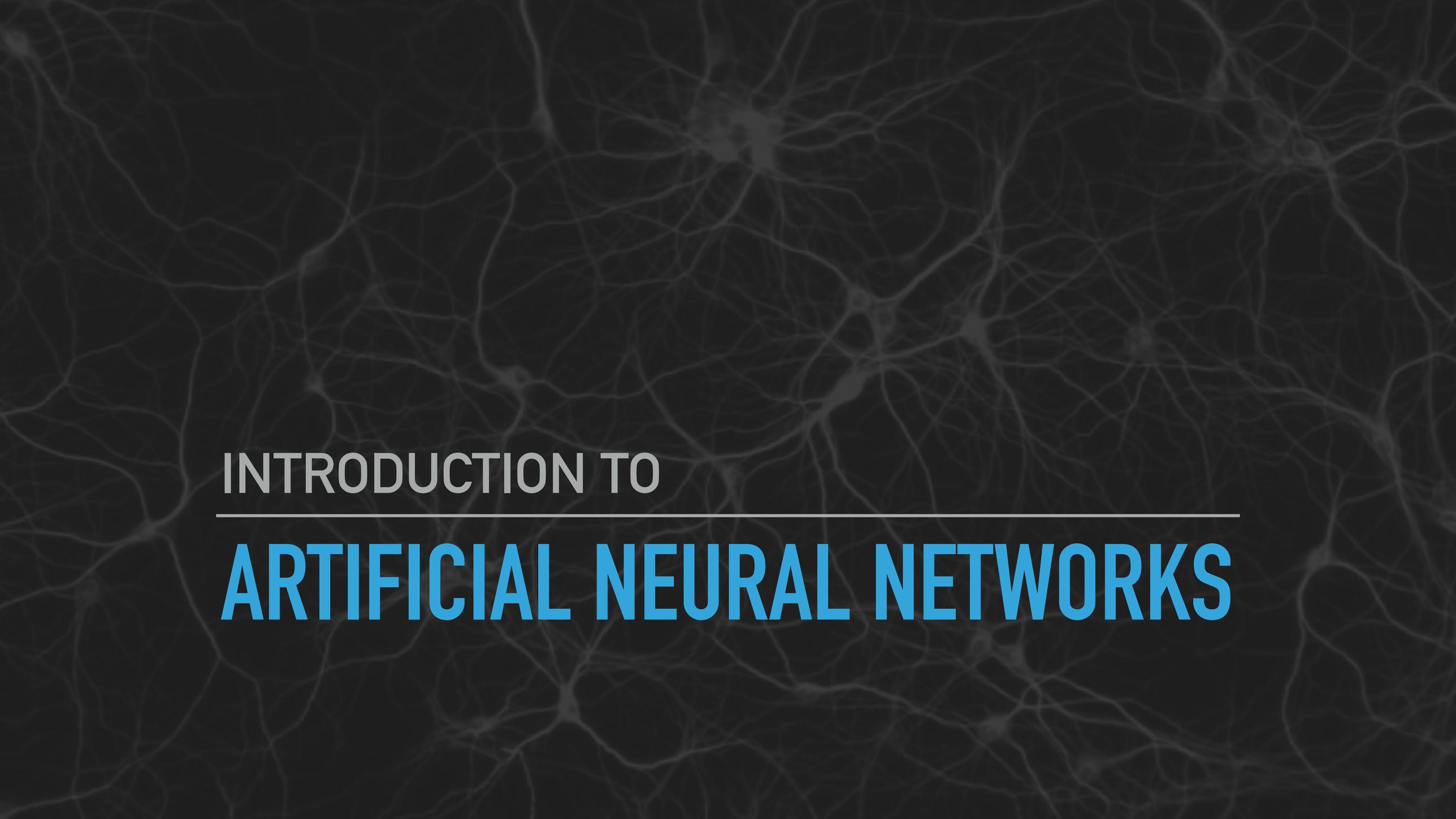
CONSUMER HARDWARE ADVANCES

- Processing Power
  - Mobile “workstations”
  - Realistic to apply machine learning techniques in consumer applications
- New devices are ramping up the sensors and input methods
- iPad Pro
  - High resolution, low latency touch screen
  - Pencil

GESTURES

- Haven’t advanced much beyond tap, swipe and pinch
- ‘Hidden’ and conflicting
- Drawing and handwriting are ‘natural’ skills
- Pencil gives us a great opportunity to explore a natural interface on iOS

NAPKIN DEMO


INTRODUCTION TO
ARTIFICIAL NEURAL NETWORKS

ARTIFICIAL NEURAL NETWORK

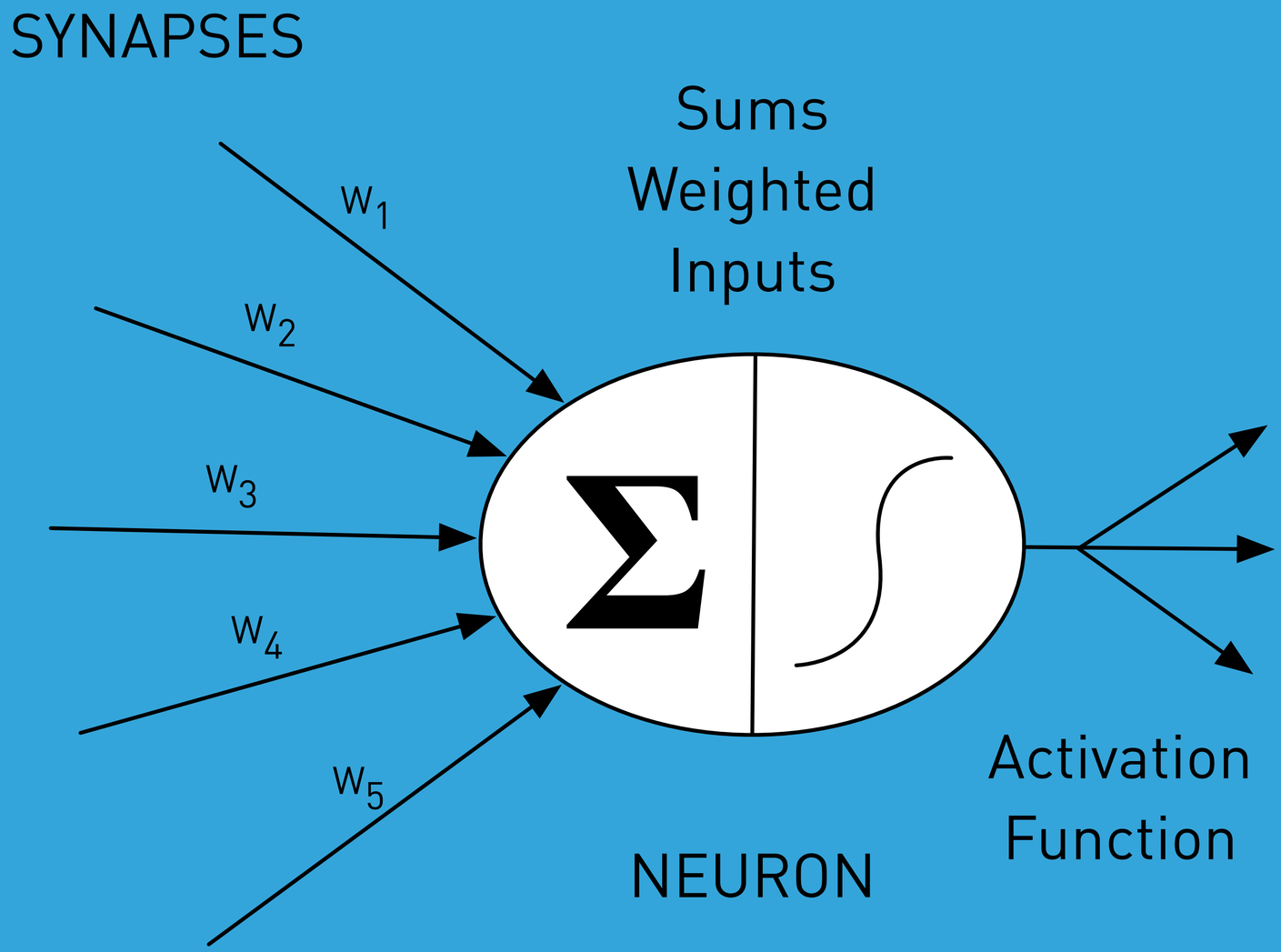
- Machine Learning technique
- Simple model inspired by biological nervous systems / brain
- Low-level compared to high-level, rule-driven Expert System
- Excellent at pattern recognition
- Adaptive / capable of learning

PROBLEMS WELL SUITED TO NEURAL NETWORKS

- Problems with a large number of inputs
- Problems that are hard to solve with rules-based systems
  - Image recognition
  - Handwriting recognition
  - Speech and language understanding
  - Autonomous driving / piloting

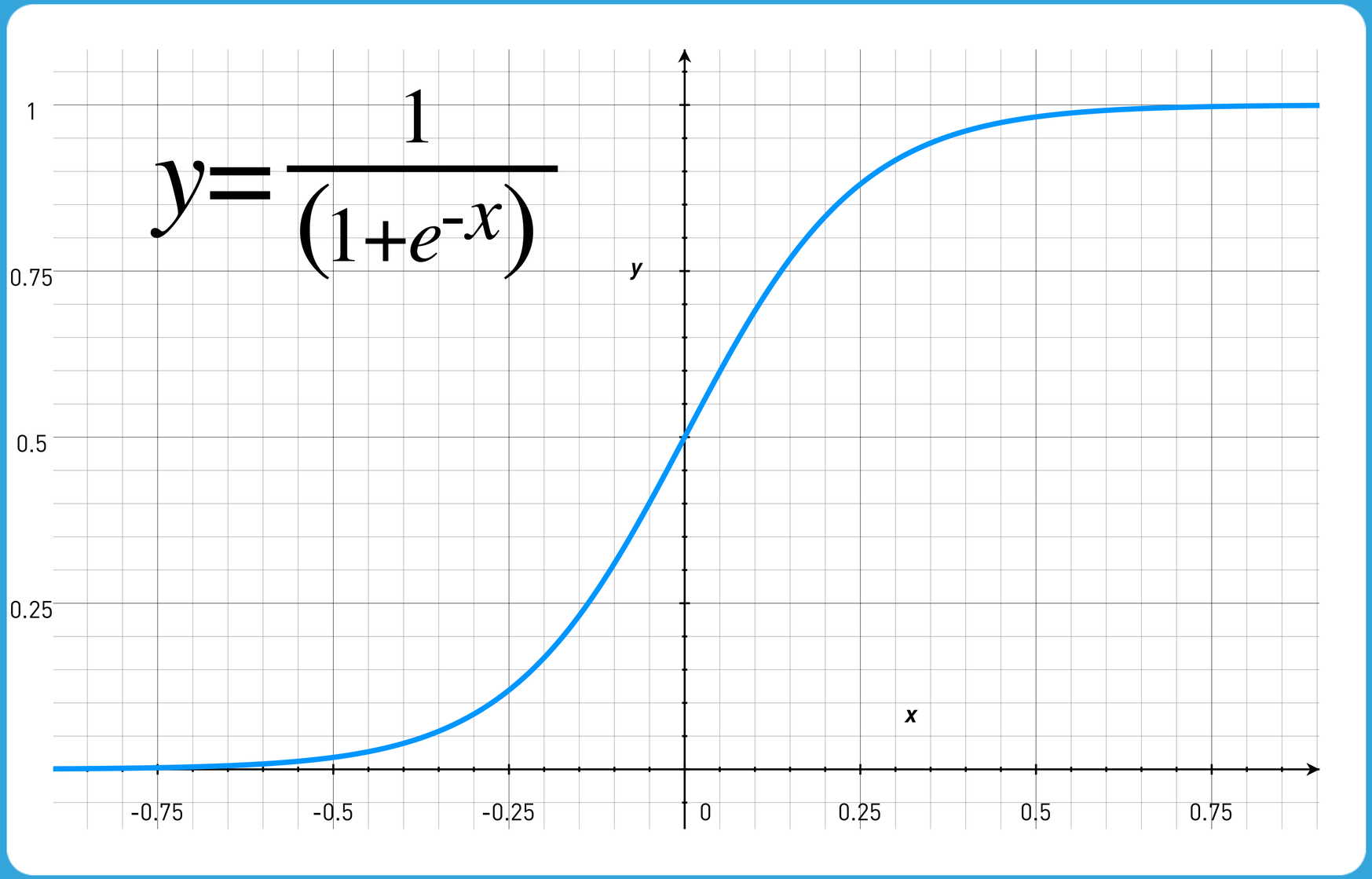
MODEL

- NEURON
  - A node that sums incoming ‘signals’ and outputs a signal
  - Activation function controls output based on the inputs
- SYNAPSE
  - Connections between neurons
  - Synapse scales or ‘weights’ the input traveling along it


SYNAPSES
Sums
Weighted
Inputs
w_1
w_2
w_3
w_4
w_5
Σ
∫
Activation
Function
NEURON

ACTIVATION FUNCTIONS

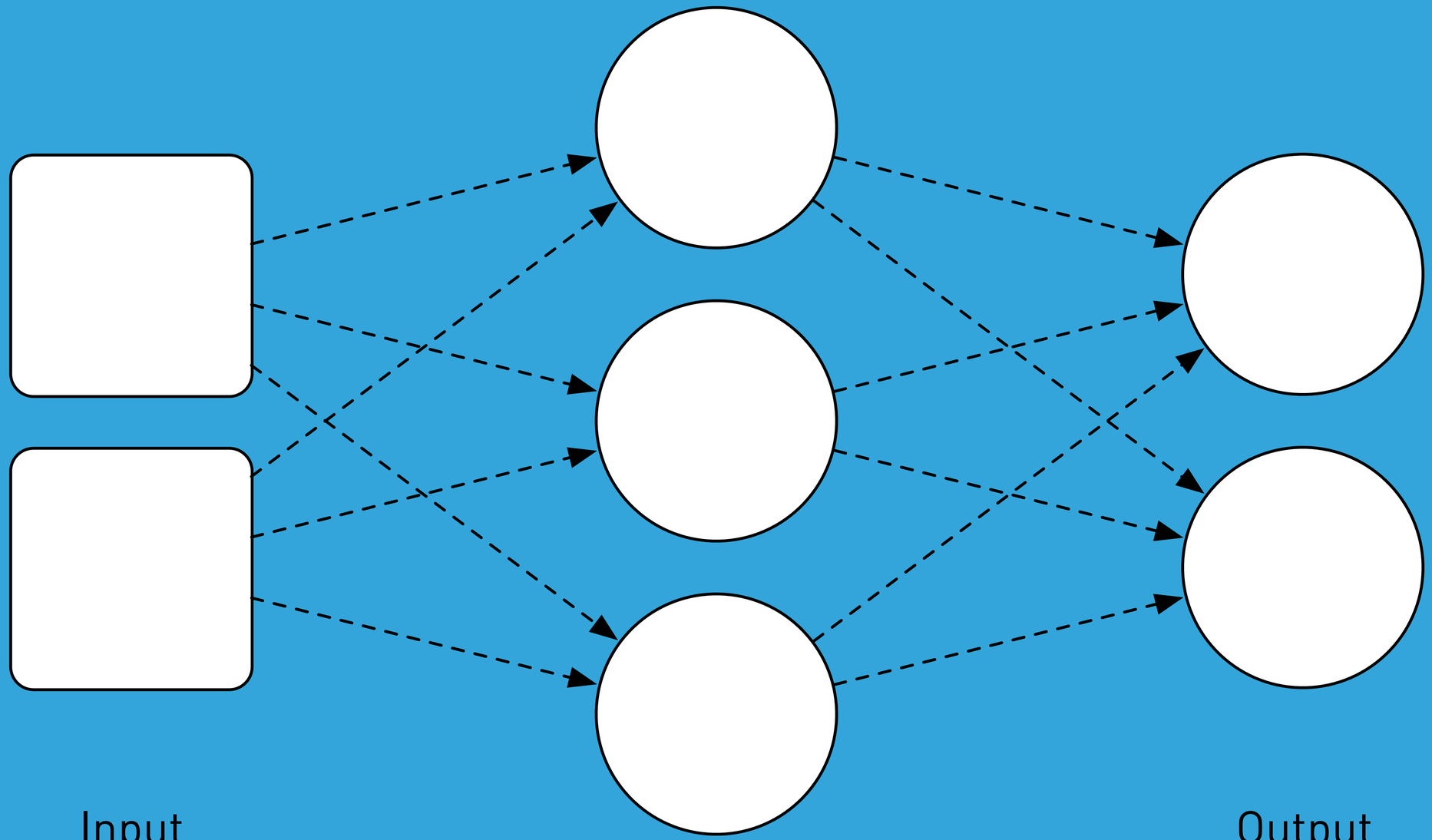
- A neuron's activation function controls how the inputs are converted into an output
- LINEAR
  - just scales the input
- THRESHOLD
  - If the sum of the inputs is greater than some X, activate
- SIGMOID
  - 'Squashes' the inputs to min and max (0, 1) or (-1, 1)

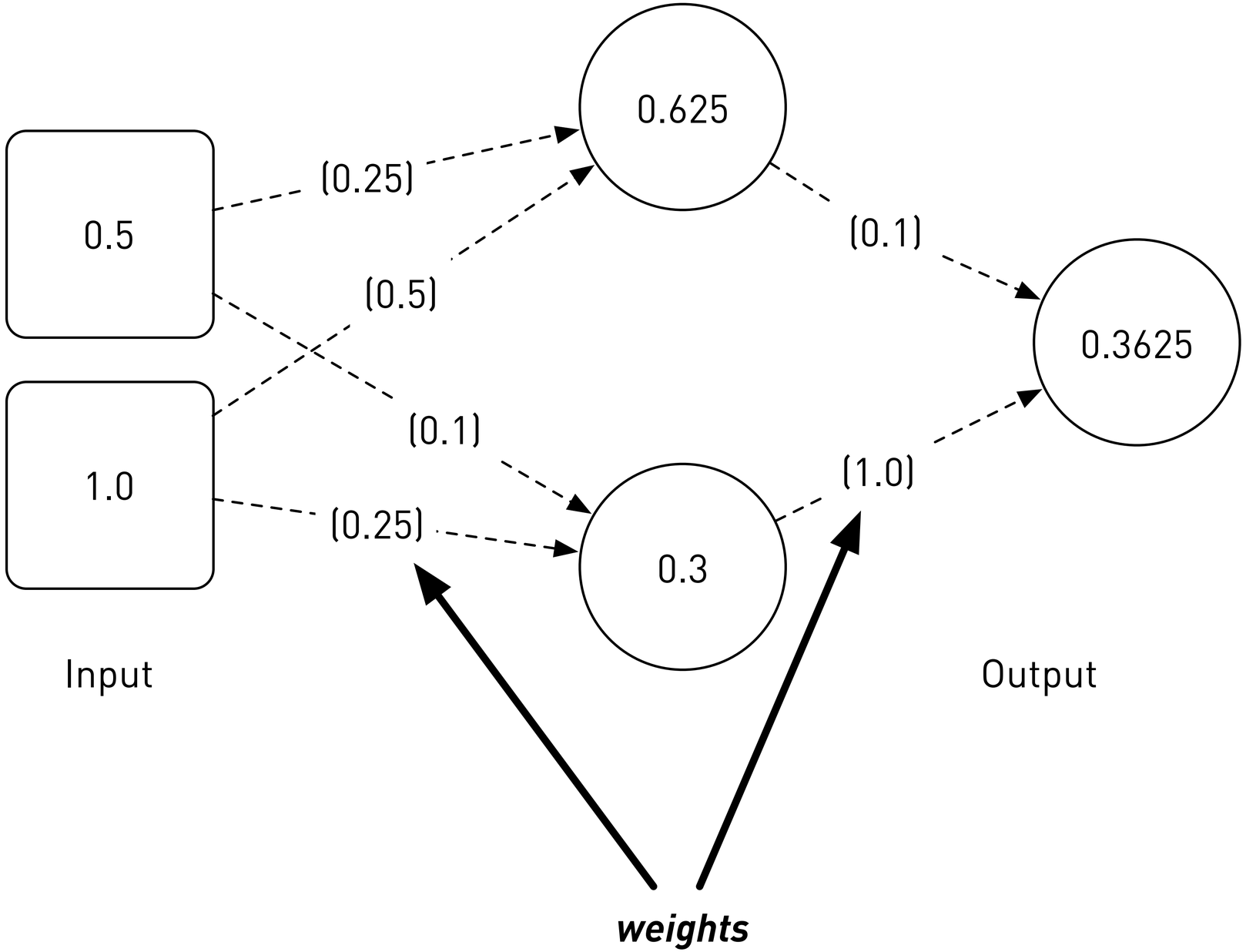


Sigmoid Activation Function

NETWORK

- Neurons connected in a network
  - Feed Forward network - signals only travel one direction (no feedback)
- Layers between the input and output are called ‘Hidden Layers’
  - Simple model: inputs, 1 hidden layer, outputs
  - Additional hidden layers allow approximating non-linear functions. Intuitively they classify higher levels of abstraction


Input
Hidden Layer
Output


0.625
(0.25)
0.5
(0.1)
0.3625
(0.5)
(0.1)
1.0
(1.0)
(0.25)
0.3
Input
Output
weights

OUTPUTS

- For classification problems (shapes, handwriting, etc) we have one output node per class (Circle, Square, Triangle)
- If an output node activates, we read that as the inputs being in that node's class
- Often, the output node's activation is between 0.0 and 1.0 and all nodes will have some value
- We can treat the node with the maximum activation as the answer to which class (confidence)

LEARNING

- The learning or adaptive nature of the network comes from adjusting the weights of the synapses between neurons.
- Network structure is unchanged
- In supervised learning we provide an input with a known answer and adjust the weights of each synapse to reduce the error
- Repeat with many examples from a training set to arrive at weights that can give correct answers for any input

ERROR (COST FUNCTION)

- How do we measure how wrong a wrong answer is?
  - Square Error of some form generally

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

ADJUSTING WEIGHTS

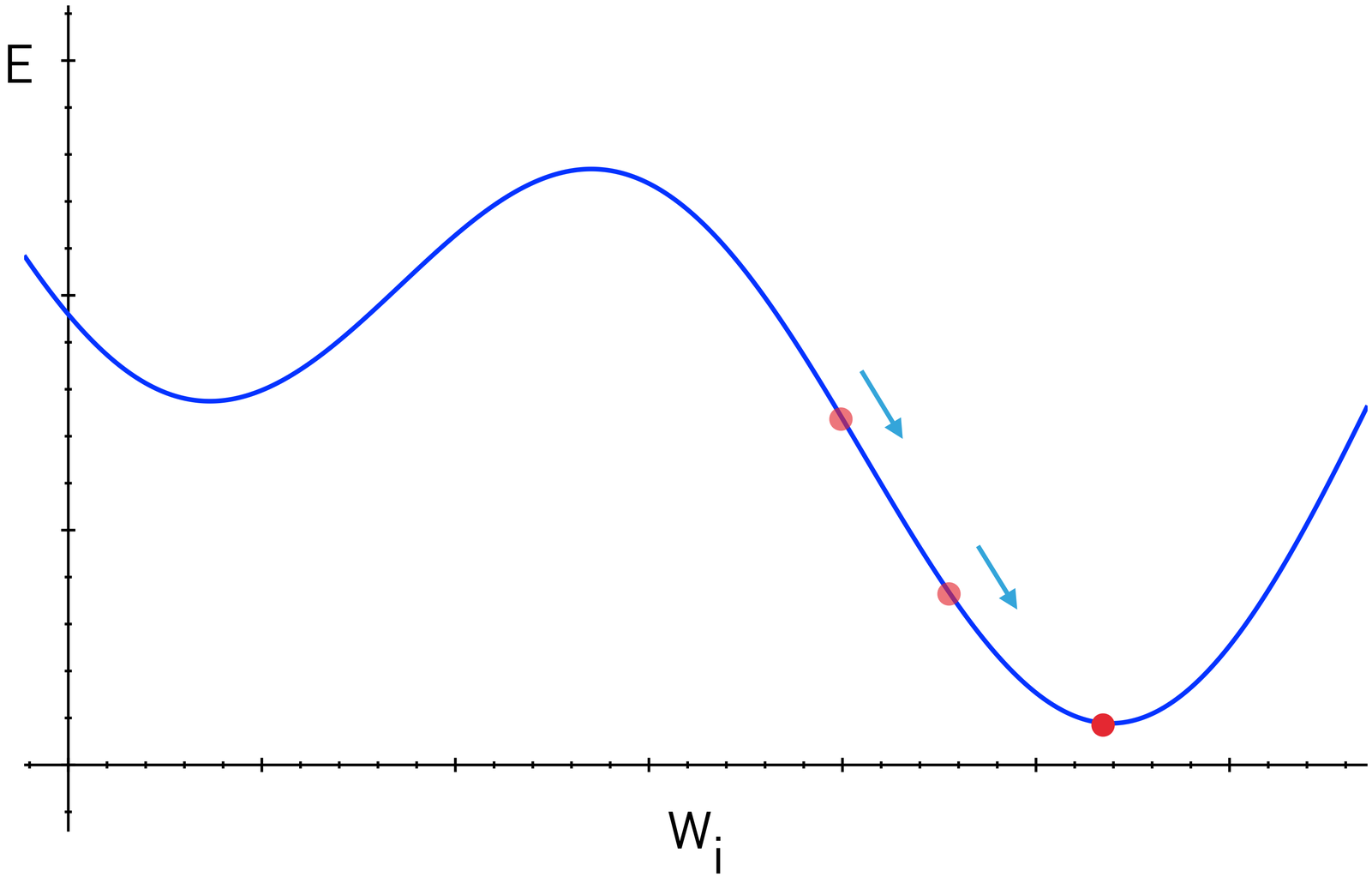
- How do we adjust the weights?
  - How about just trying every possible weight and picking the ones that give the least error?
  - For even the most simple networks, the computations required would require days, years, centuries, etc.

BACK PROPAGATION AND GRADIENT DESCENT

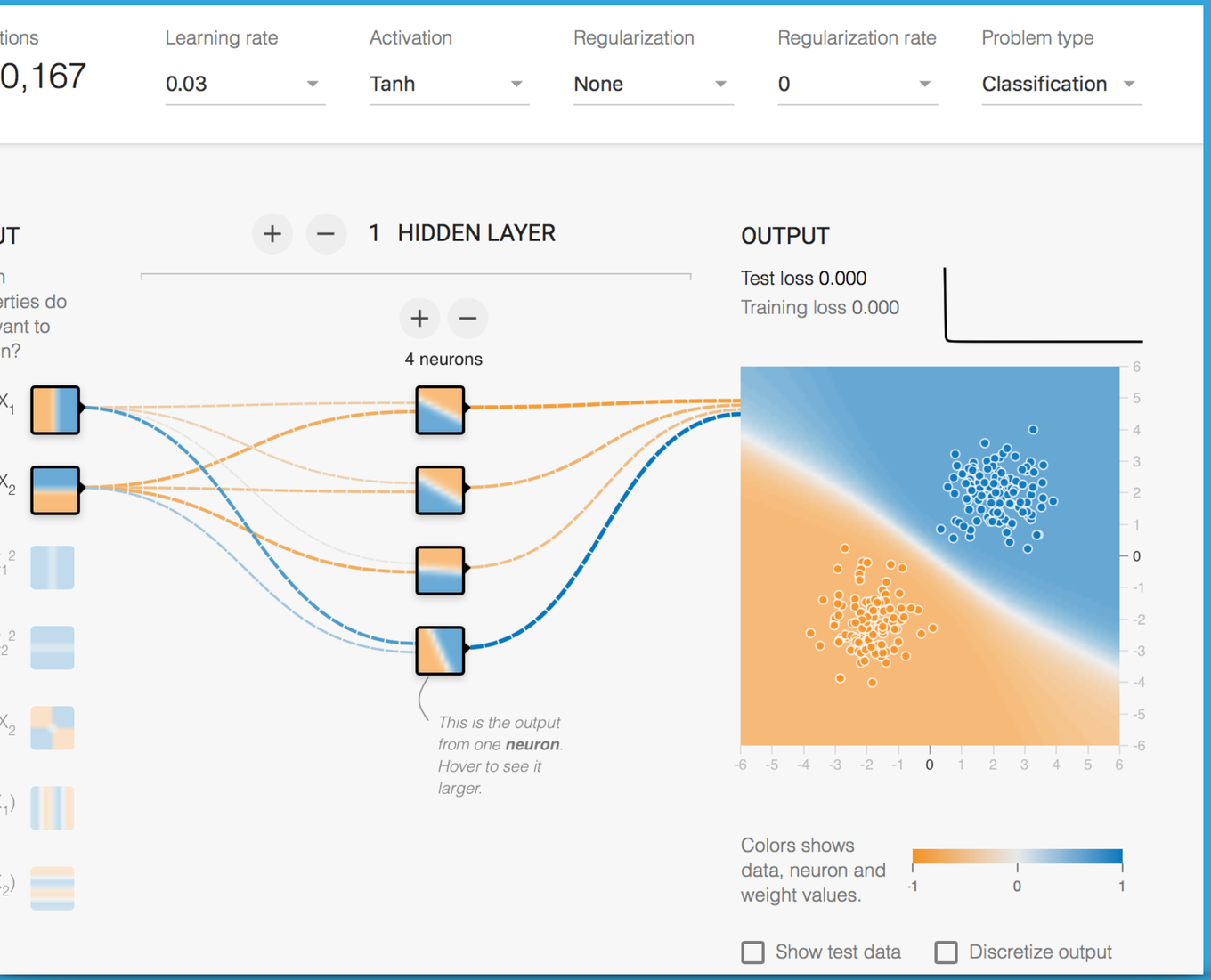
- Adjust the weights in small steps
- We want to go in the right ‘direction’ to make the error smaller. For each weight do we get larger or smaller?
- Construct the mathematical function of the error of the network in relation the inputs and weights
- Take the partial derivative (slope) of the error function with respect to each weight to determine which direction to adjust the weight

TUTORIALS

- Gradient Descent
  - https://www.youtube.com/watch?v=5u0jaA3qAGk
- Back Propagation
  - https://www.youtube.com/watch?v=aVId8KMsdUU



Error function for a weight



DEMO

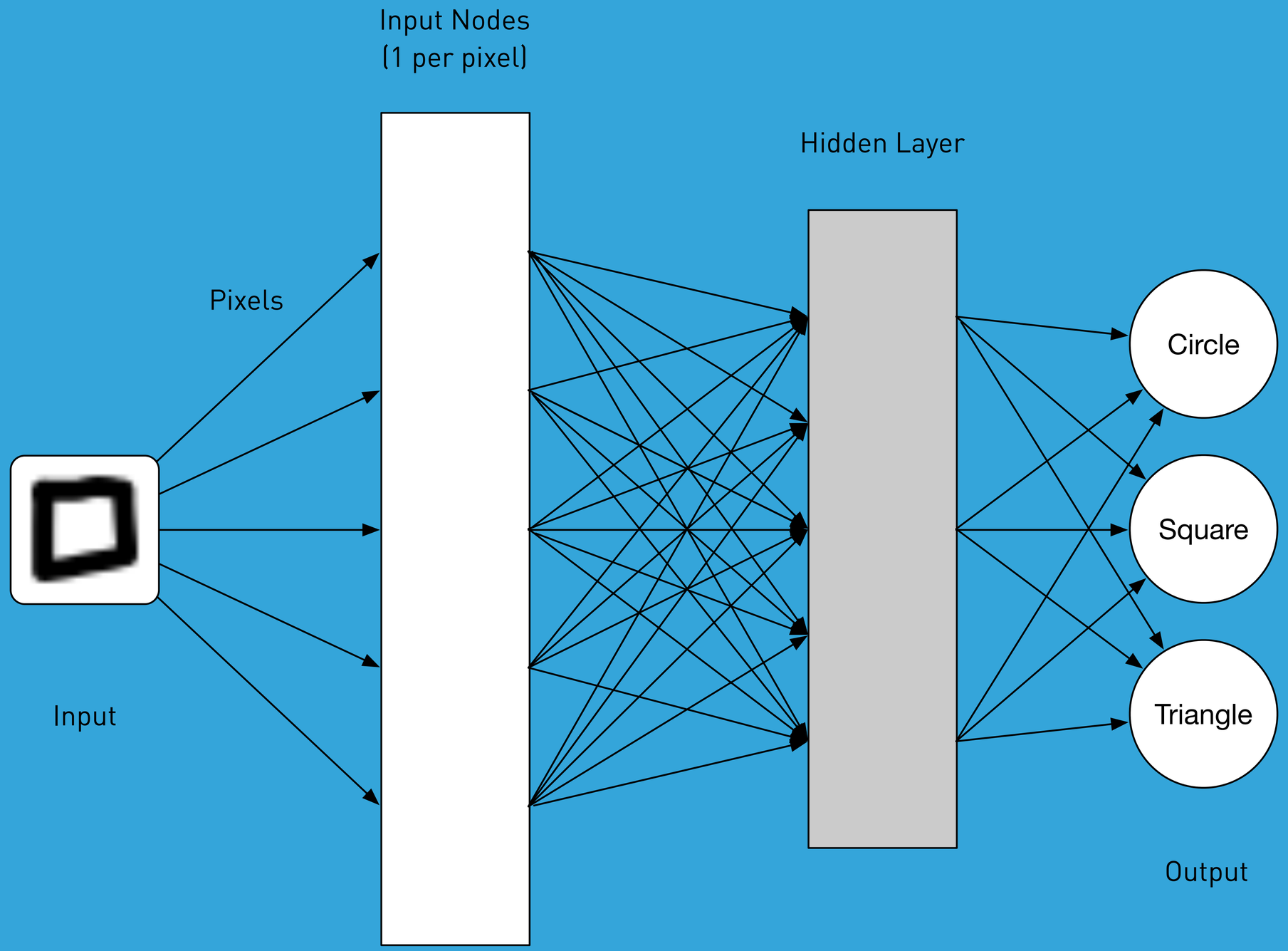
http://playground.tensorflow.org

PROTOTYPE

DETECTING DRAWN SHAPES

PROBLEM AND MODEL

- iOS iPad application
- Detect simple drawn shapes – Circle, Square, Triangle
- Feed shape ‘data’ into a Feed Forward Neural Network
- Output is confidence input matches a particular shape
- Supervised training on sample drawings


Input Nodes
(1 per pixel)
Hidden Layer
Pixels
Circle
Square
Triangle
Input
Output

INPUT – MANY POSSIBLE SHAPE “FEATURES”

- Stroke count and direction
- Stroke velocity
- Number of angles
- Convexity
- **Image Bitmap**

OUTPUT

- 1 Output Node for each shape
  - O_0: Circle, O_1: Square, O_2: Triangle
- Resulting value 0.0 – 1.0 interpreted as confidence inputs match that output node's shape
- If output == [0.0, 1.0, 0.0], 100% confidence input is a square

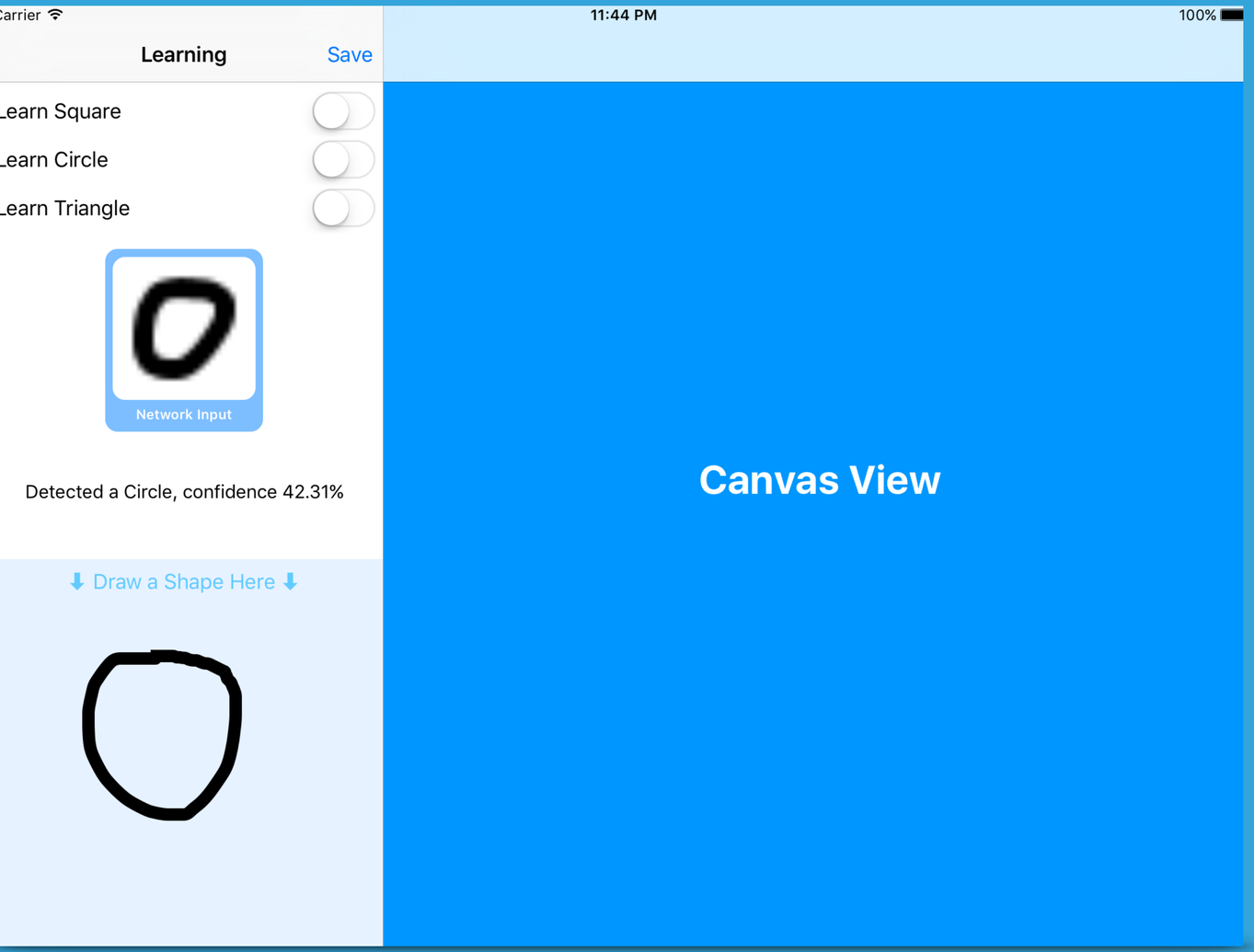
NETWORK STRUCTURE

- 3 Layer, full-connected network
- 1 input for each image pixel
- 1 output for each type of shape to classify

PARAMETERS

- 784 input nodes
  - 28 x 28 pixel bitmap of drawn shape
- 280 hidden nodes
- 3 Output nodes
- Sigmoid Activation Function
- Average Error Function

TRIAL, ERROR, GUESSING — SCIENCE!


Carrier
11:44 PM
100%
Learning
Save
Learn Square
Learn Circle
Learn Triangle
Network Input
Detected a Circle, confidence 42.31%
⬇ Draw a Shape Here ⬇
Canvas View

DEMO

WE NEED SOME NEURONS AND SYNAPSES!

- The model for synapses and neurons isn't too complex,

  We got this, couple of hours, easy!

- What about that activation function - Was his name Sigmund or something?

  Hmmm, we can probably find an example

WE NEED SOME NEURONS AND SYNAPSES!

- What about computing the error or cost?

  Lets just use that squared error, not to bad right?

- We also need Gradient Descent to adjust the weights– partial derivatives, matrix math, dot products, summation, local minima, non-convex functions

OKAY... THAT'S A LOT OF MATH.
MAYBE MY ESTIMATE WAS OFF A LITTLE BIT.

SWIFT AI

- Fortunately there are a lot of open source libraries that provide basic Neural Network implementations
- Swift-AI to the rescue
  - https://github.com/collinhundley/Swift-AI
  - Compact and simple interface

OVERVIEW OF SWIFT AI INTERFACE

```
class FFNN {
 public init(inputs: Int, hidden: Int, outputs: Int,
             learningRate: Float = 0.7, momentum: Float = 0.4,
             weights: [Float]? = nil,
             activationFunction: ActivationFunction = .Default,
             errorFunction: ErrorFunction = .Default(average: false))
 }
```

OVERVIEW OF SWIFT AI INTERFACE

```
    /// Propagates the given inputs through the neural network,
returning the network's output.
    /// - Parameter inputs: An array of `Float`s, each element
corresponding to one input node.
    /// - Returns: The network's output after applying the given
inputs, as an array of `Float`s.
    public func update(inputs inputs: [Float]) throws -> [Float]
```

OVERVIEW OF SWIFT AI INTERFACE

```
    /// Trains the network by comparing its most recent output to the
given 'answers', adjusting the network's weights as needed.
    /// - Parameter answer: The 'correct' desired output for the most
recent update to the network, as an array of `Float`s.
    /// - Returns: The total calculated error from the most recent
update.

public func backpropagate(answer answer: [Float]) throws -> Float
```

OVERVIEW OF SWIFT AI INTERFACE

```
/// Reads a FFNN from file.
/// - Parameter filename: The name of the file, located in the
default Documents directory.
public static func fromFile(filename: String) -> FFNN?

/// Writes the FFNN to file.
/// - Parameter filename: The name of the file to write to. This file
will be written to the default Documents directory.
public func writeToFile(filename: String)
```

CAPTURING THE DRAWING

- UIPanGestureRecognizer
- Connect the touch points in a quadratic UIBezierPath
- Stroke the path
- Update and redraw path as new touches occur
- If no new touches after 1 second, consider the shape complete

WHY A QUADRATIC BEZIER PATH?

- We could just draw a ‘brush tip‘ along the touch points instead of a bezier path
- Path gives us an easy method to smooth out touches to get a more natural result
- Path gives us a way to normalize the stroke width when pre-processing for the network (more later)
- Path doesn’t lend itself to variable width strokes and produces odd effects on direction reversal

SMOOTHING THE PATH

- Use the mid-point between current touch and previous touch as the next path point
- Use the current and previous touch point as the bezier path control points
- See WWDC 2012 session – Building Advanced Gestures Recognizers

PREPARING THE BITMAP FOR PROCESSING

- Capture drawing as an image
  - Crop to just the path bounds
  - Normalizes the aspect (*rectangles become squares*)
- Scale the image down
  - just enough pixels to capture the essential features
  - Re-stroke the path so stroke-width is invariant

PREPARING THE BITMAP FOR PROCESSING

- Convert to grayscale
  - RGB would mean we'd need 3 input nodes for each pixel
  - Color is not important for our problem
  - Grayscale vs. Monotone
    - Grayscale values influence the networks behavior
  - Network expect inputs between (0.0, 1.0)

XCODE TIME

HOW DO WE TRAIN THIS THING?

- Depending on the problem space and the network configuration, you might require hundreds or thousands of samples
- Feed samples with answers to network and let it go
- Any online corpuses for hand-drawn shapes? 
- Drawing is fun!

CONCLUSION

WHERE DO WE GO NOW?

NEXT STEPS

- Rejection
- Skew, rotation
- Capture additional features
- Adaptive learning / user correction
- Performance

SOME RESOURCES THAT MIGHT BE USEFUL

- BrainCore https://github.com/aleph7/BrainCore
- SwiftSimpleNueralNetowrk https://github.com/davecom/SwiftSimpleNeuralNetwork
- AIToolBox https://github.com/KevinCoble/AIToolbox
- BirdBrain https://github.com/jordenhill/Birdbrain
- MLKit. https://github.com/Somnibyte/MLKit

The End