# Foundations of PyTorch

## GETTING STARTED WITH PYTORCH FOR MACHINE LEARNING

**Janani Ravi**
CO-FOUNDER, LOONYCORN

www.loonycorn.com

# Overview

Deep learning using neural networks

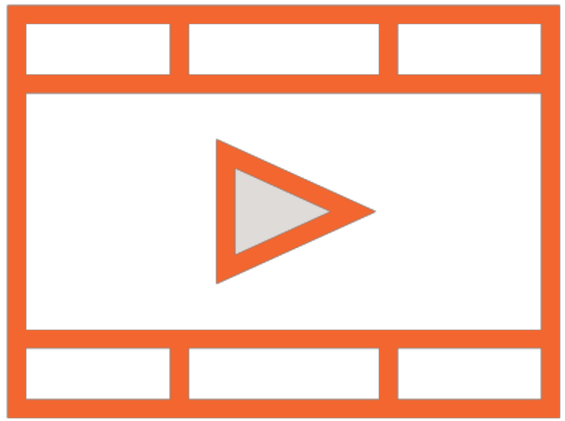
Neurons and activation functions

Introducing PyTorch to build neural networks

Understanding the differences between PyTorch and TensorFlow

# Prerequisites and Course Outline

# Prerequisites

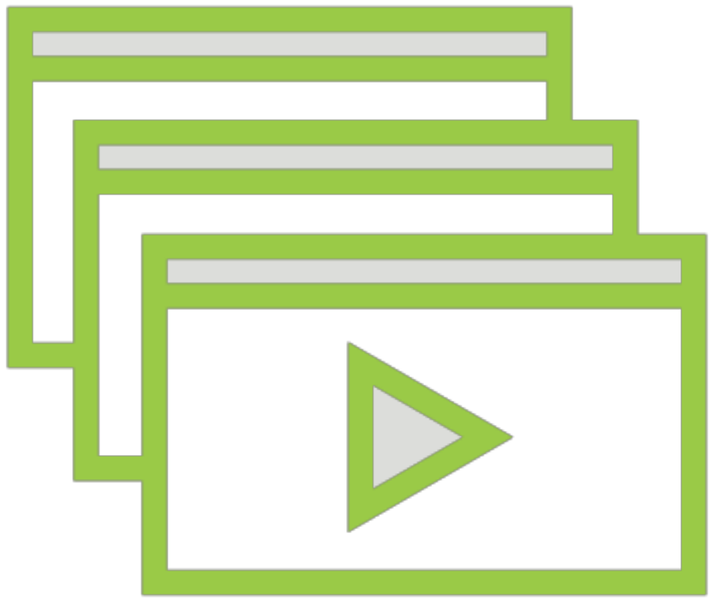


**Working with Python and Python libraries**

**Basic understanding of machine learning algorithms**

**No prior experience with neural networks necessary**

# Prerequisites



**Understanding Machine Learning by David Chappell**

**Understanding Machine Learning with Python by Jerry Kurata**

**Building Machine Learning Models in Python with scikit-learn by Janani Ravi**

# Course Outline

**Getting started with PyTorch**

- Introducing neural networks and PyTorch
- Tensor operations and CUDA support

**Gradients and the Autograd library**

- Gradient descent to train NNs
- Working with gradients in PyTorch

**Dynamic computation graphs**

- Pros and cons of working with each
- Static graph in TF vs. dynamic graph in PyTorch

# Introducing Neural Networks

# Reviews: Positive or Negative?

# ML-based Classifier

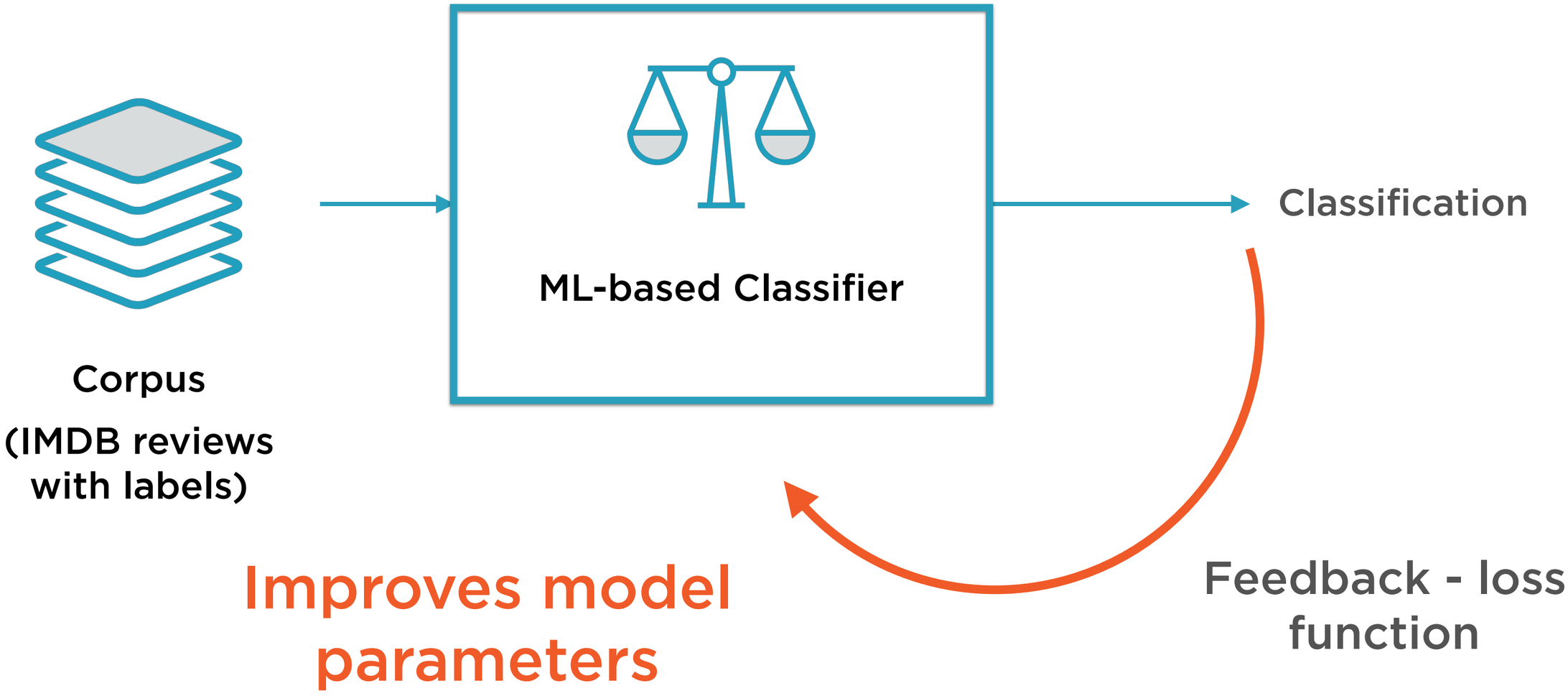
## Training

Feed in a large corpus of data classified correctly

## Prediction

Use it to classify new instances which it has not seen before

# Training the ML-based Classifier



**“Traditional”** ML-based systems rely on experts to decide what features to pay attention to

**“Representation”** ML-based systems figure out by themselves what features to pay attention to

Neural networks are examples of such systems

# What is a Neural Network?

## Deep Learning

Algorithms that learn what features matter

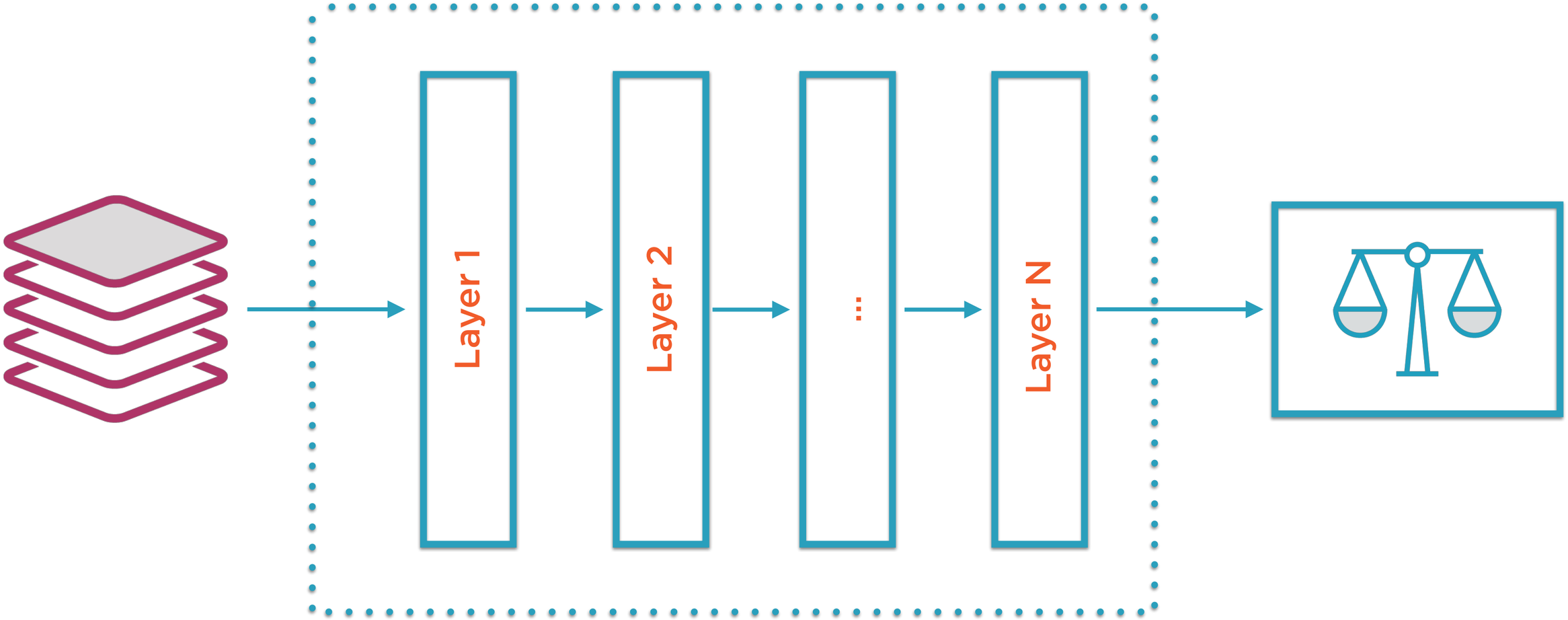
## Neural Networks

The most common class of deep learning algorithms

## Neurons

Simple building blocks that actually “learn”

# Neural Networks

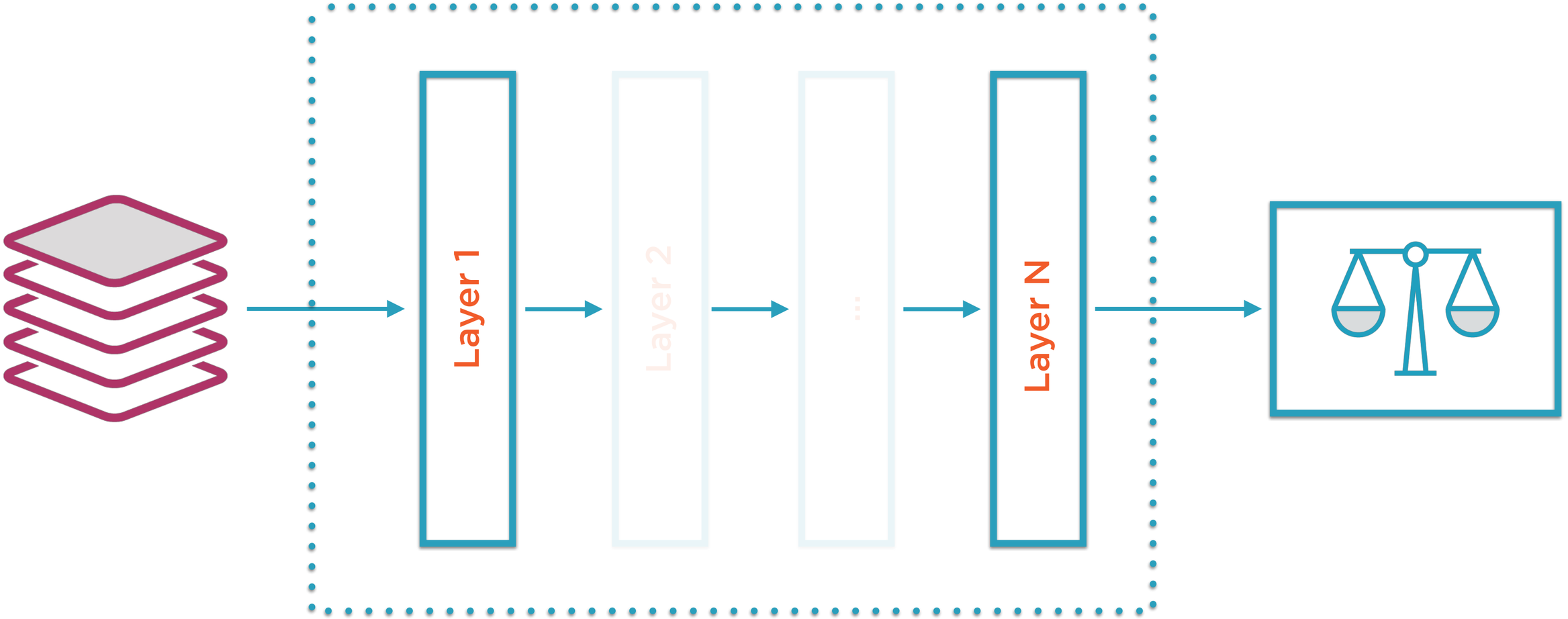



Corpus

Layers in a neural network

ML-based Classifier

# Neural Networks

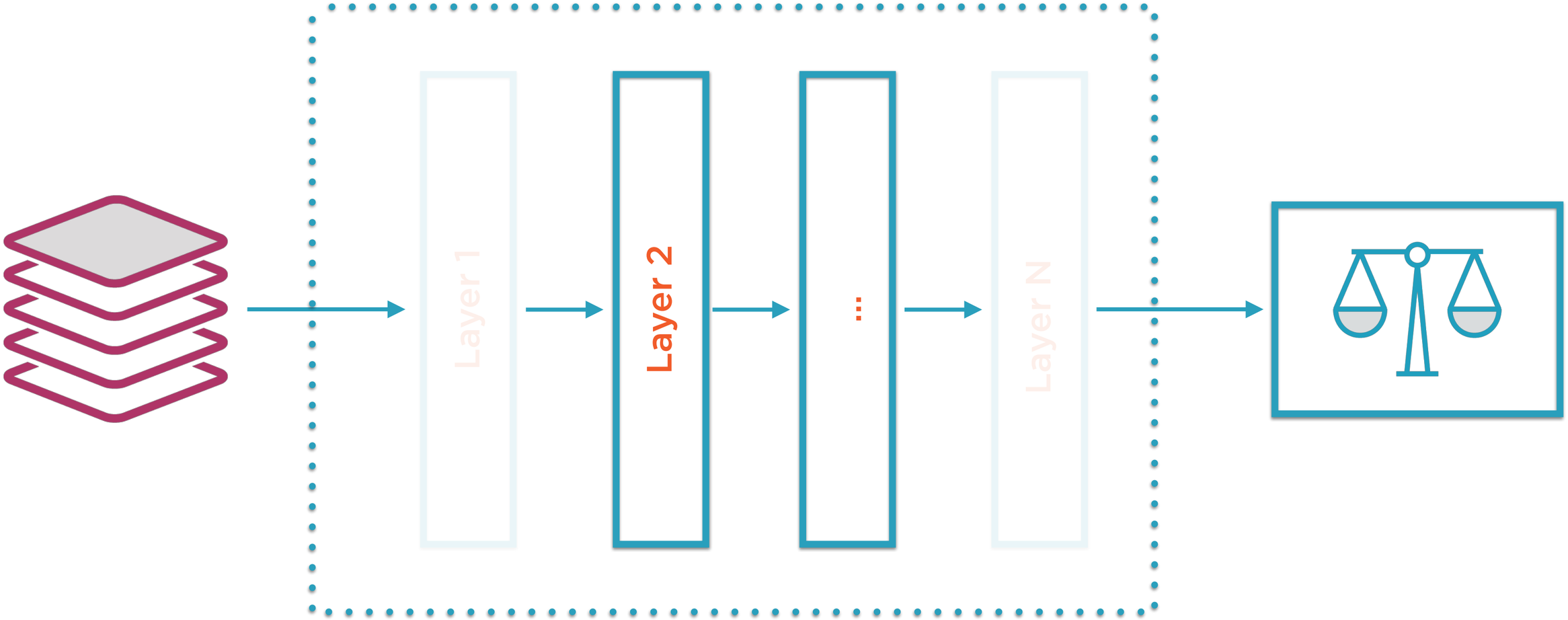



Corpus

ML-based Classifier

Visible layers

# Neural Networks

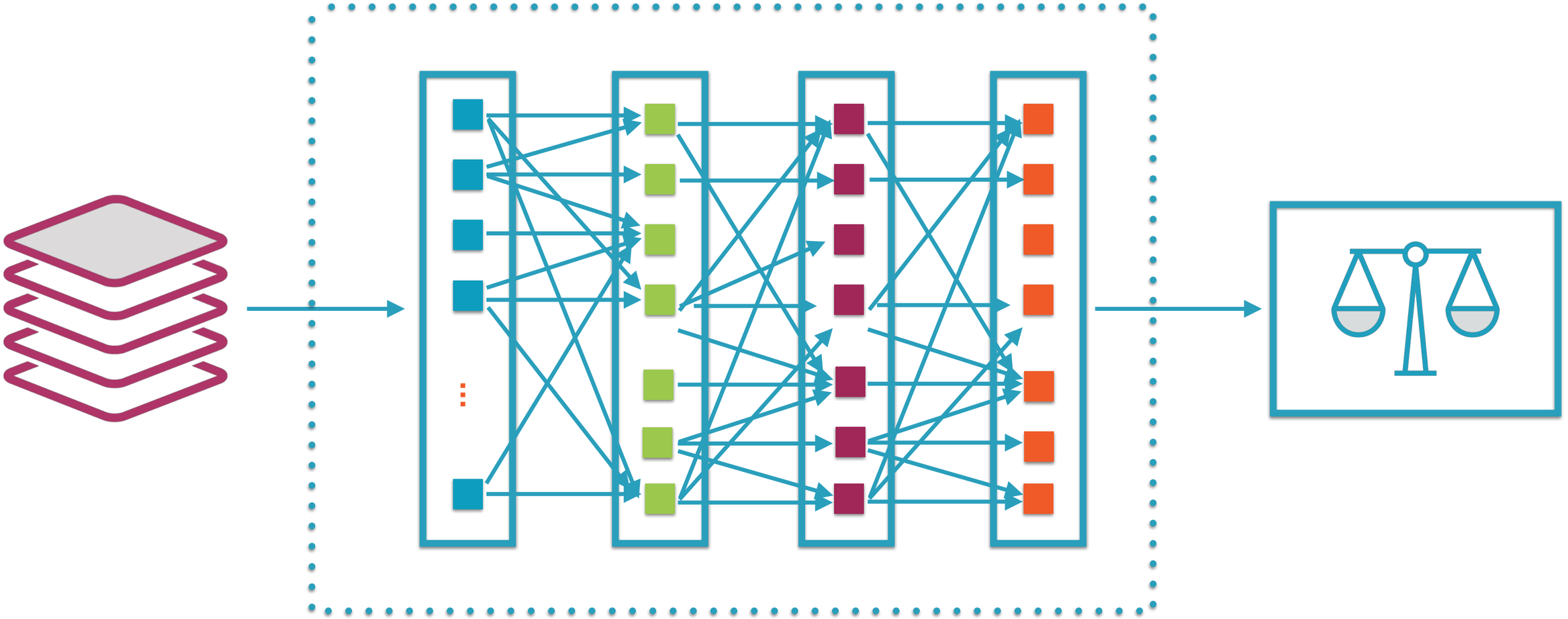



Corpus

ML-based Classifier

Hidden layers

# Neural Networks



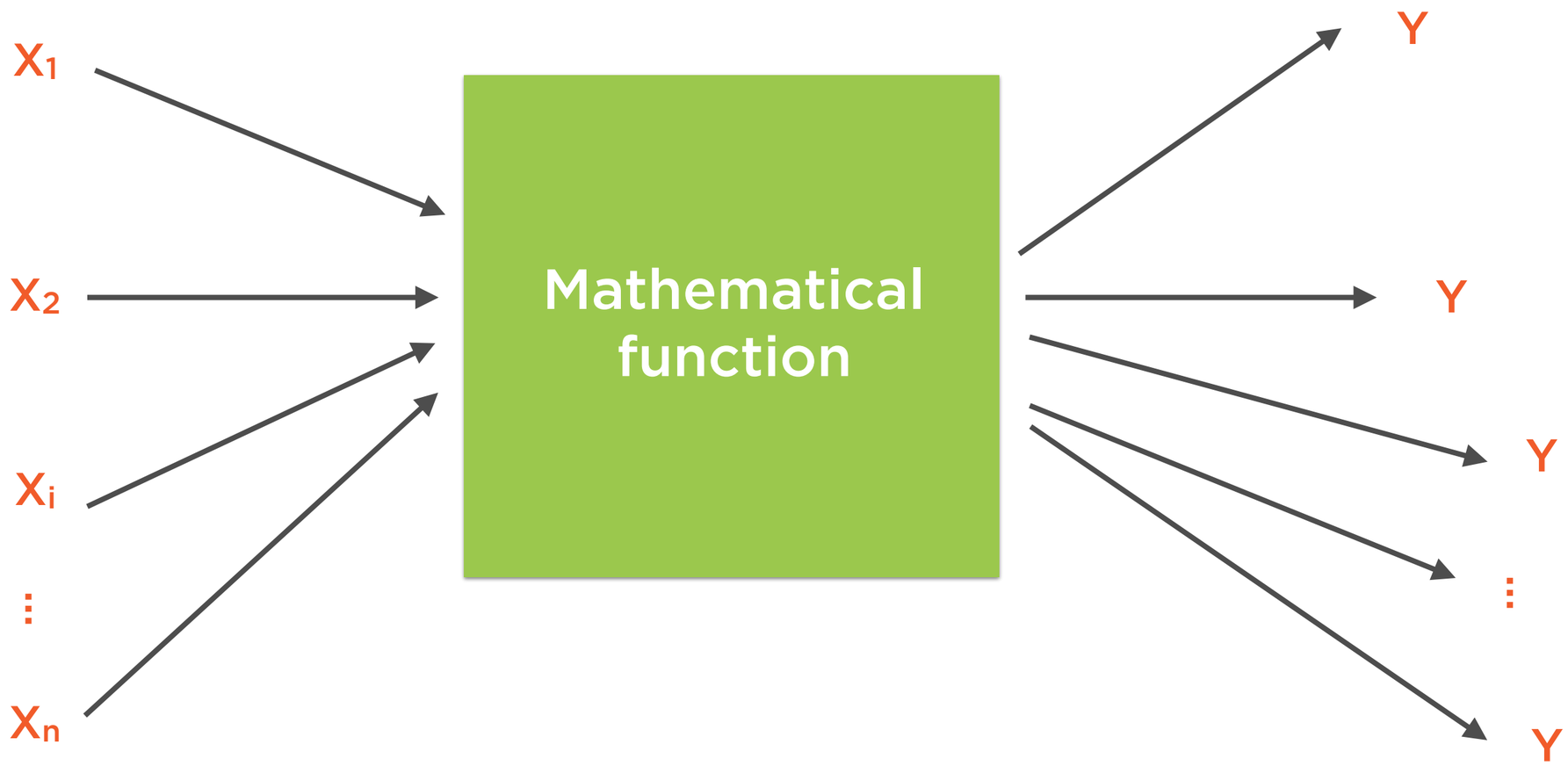
Corpus

Each layer consists of individual interconnected neurons

ML-based Classifier

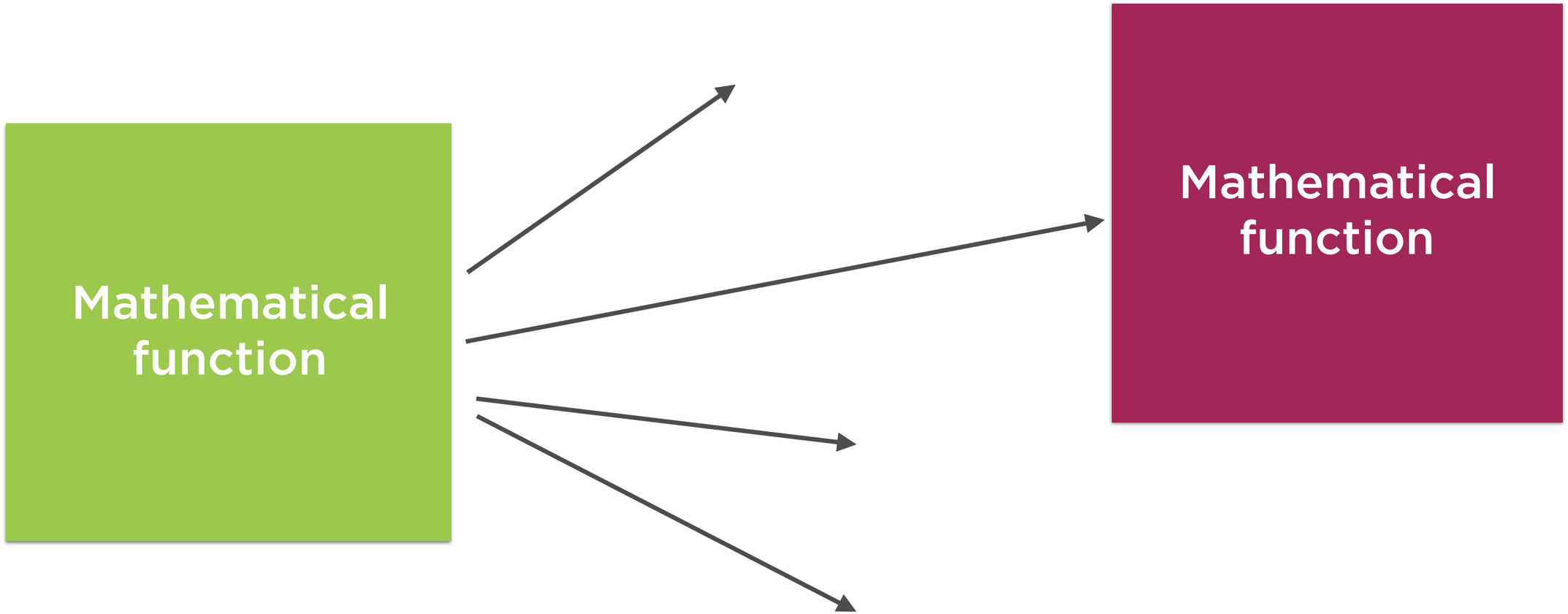
# Neurons and Activation Functions

# Operation of a Single Neuron



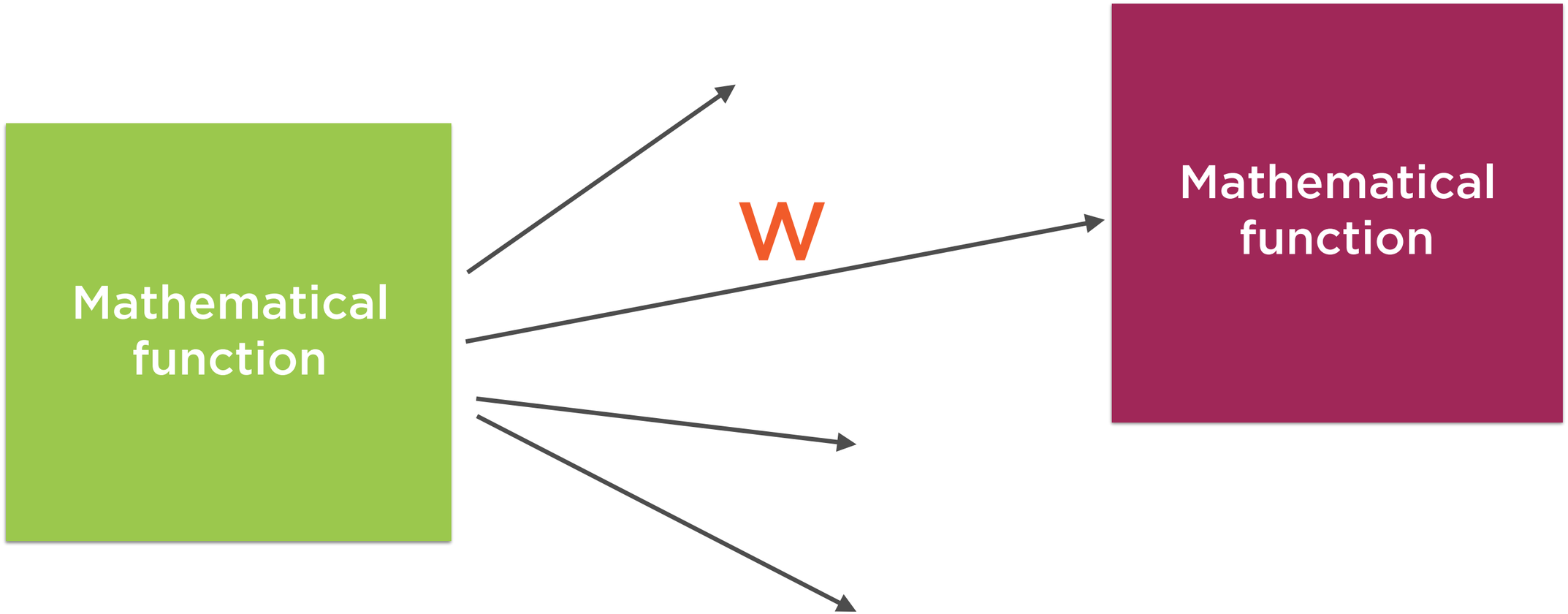

**For an active neuron a change in inputs should trigger a corresponding change in the outputs**

# Operation of a Single Neuron



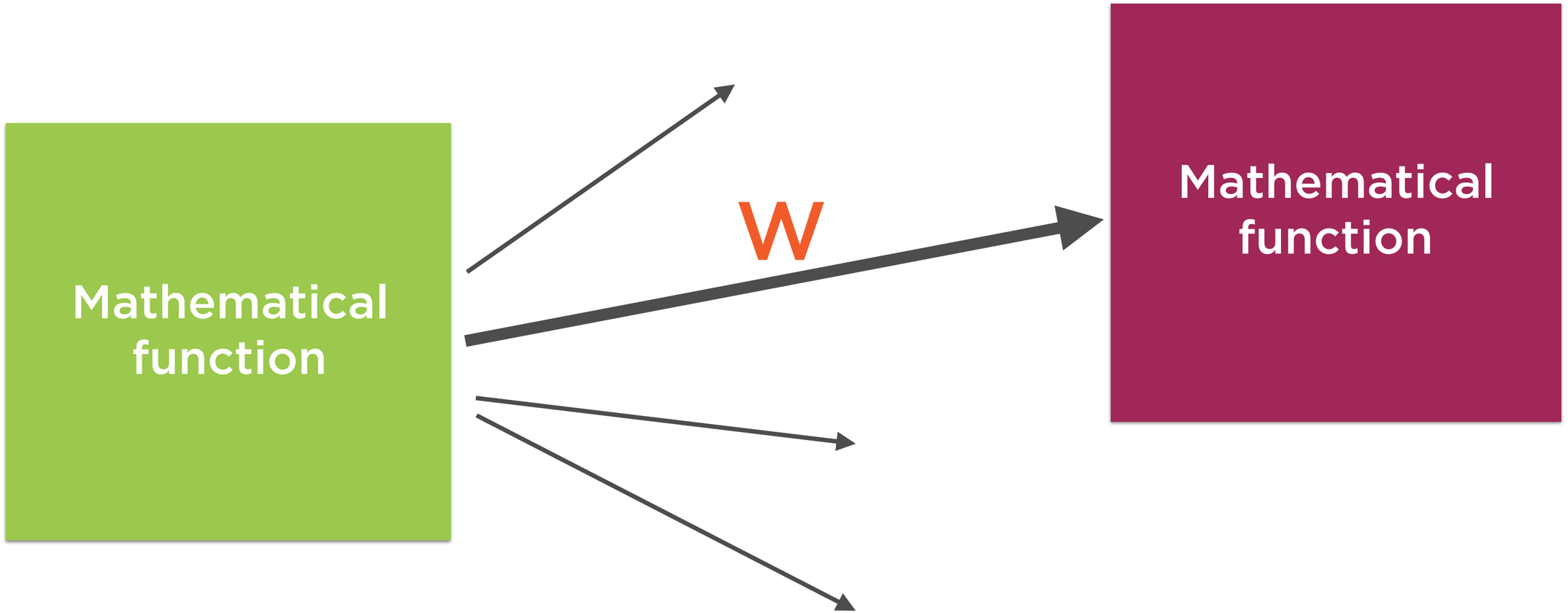

**The outputs of neurons feed into the neurons from the next layer**

# Operation of a Single Neuron




**Each connection is associated with a weight**

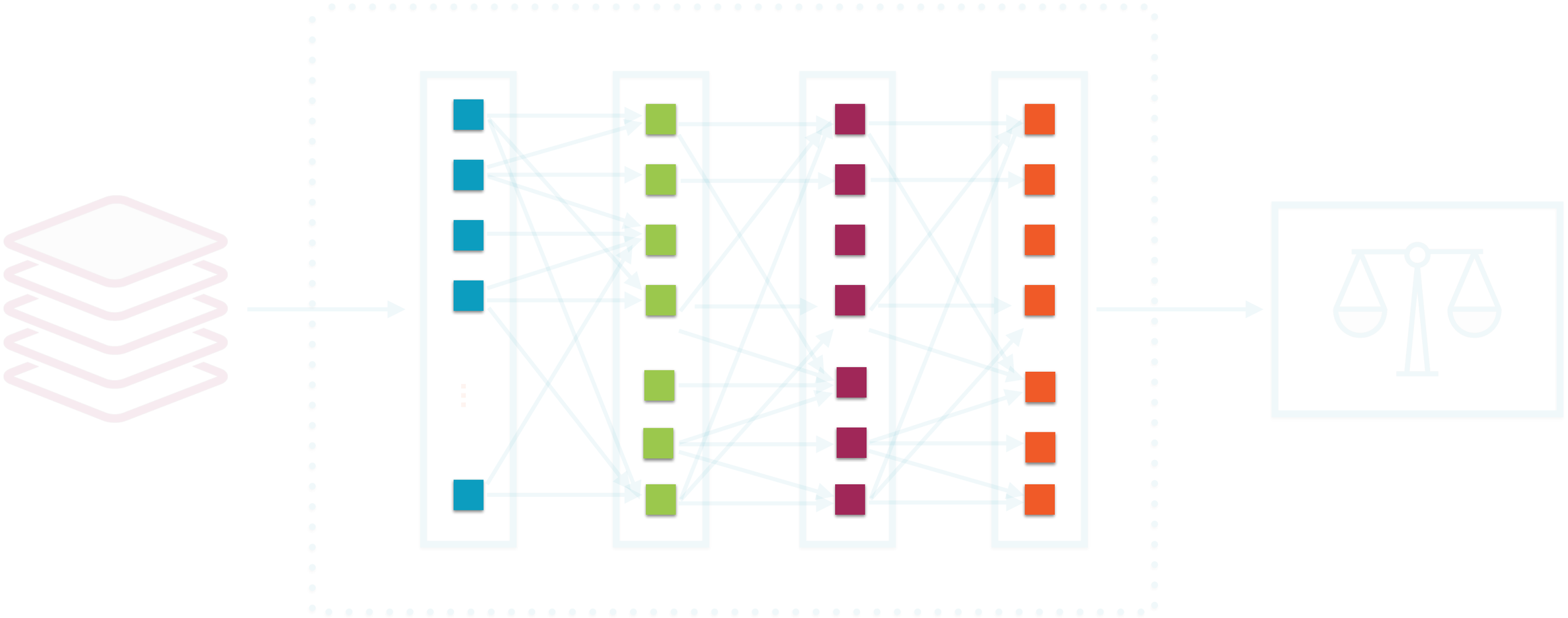
# Operation of a Single Neuron




If the second neuron is sensitive to the output of the first neuron, the connection between them gets stronger

W increases

# The Computational Graph

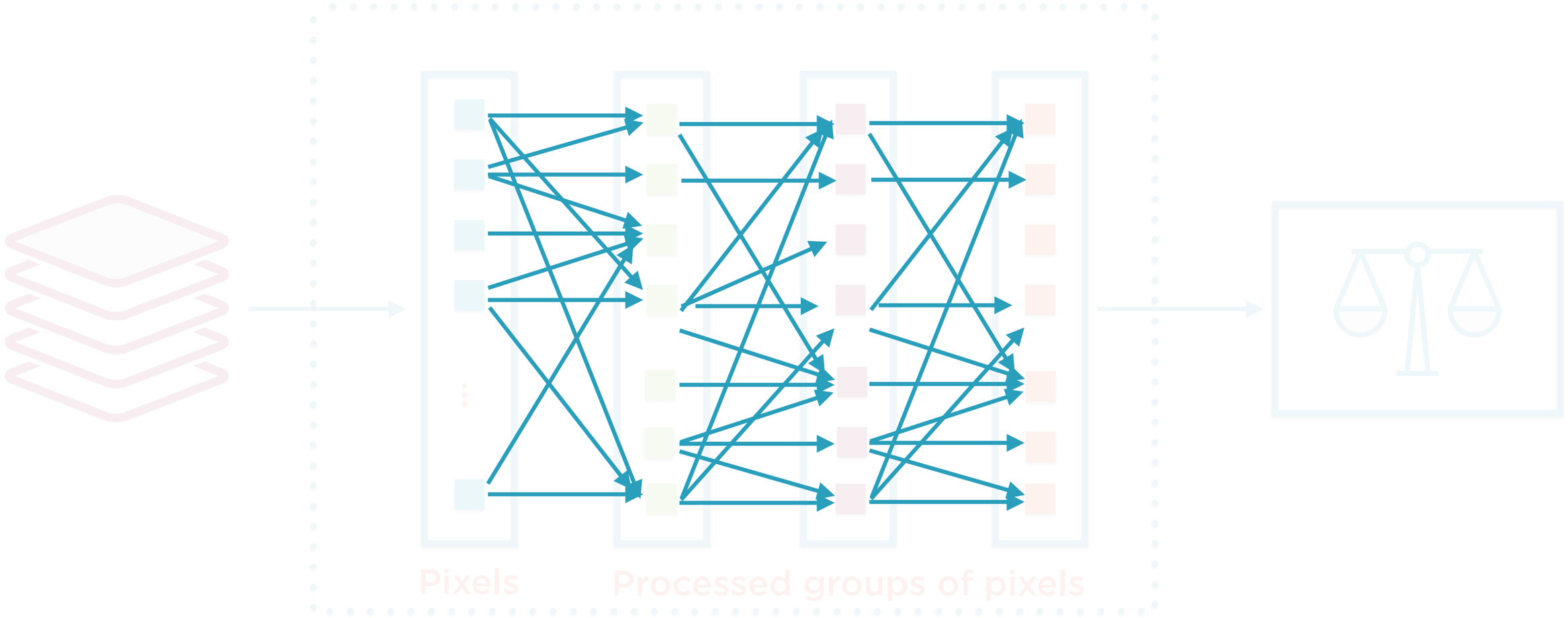


Corpus

ML-based Classifier

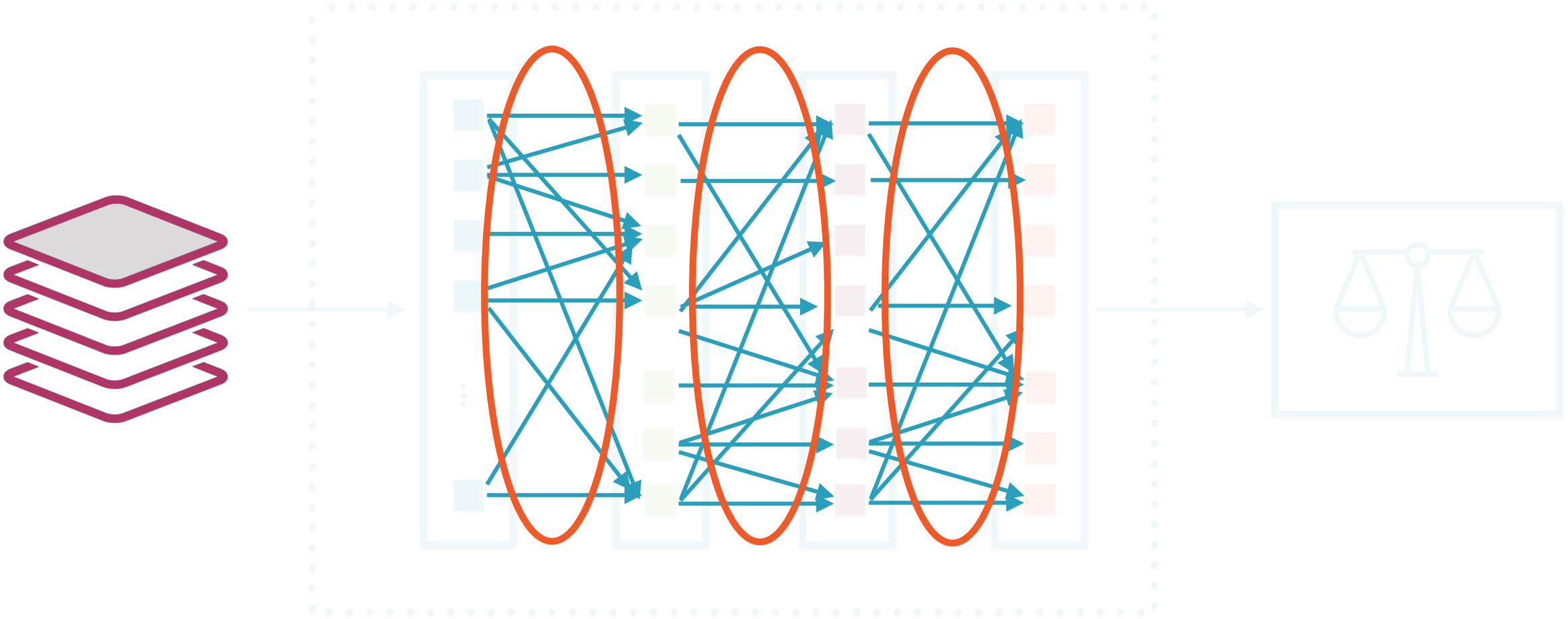
**The nodes in the computation graph are neurons (simple building blocks)**

# The Computational Graph




**The edges in the computation graph are data called tensors**

# A Neural Network



Corpus

**Once a neural network is trained, all edges have weights which help it make predictions**

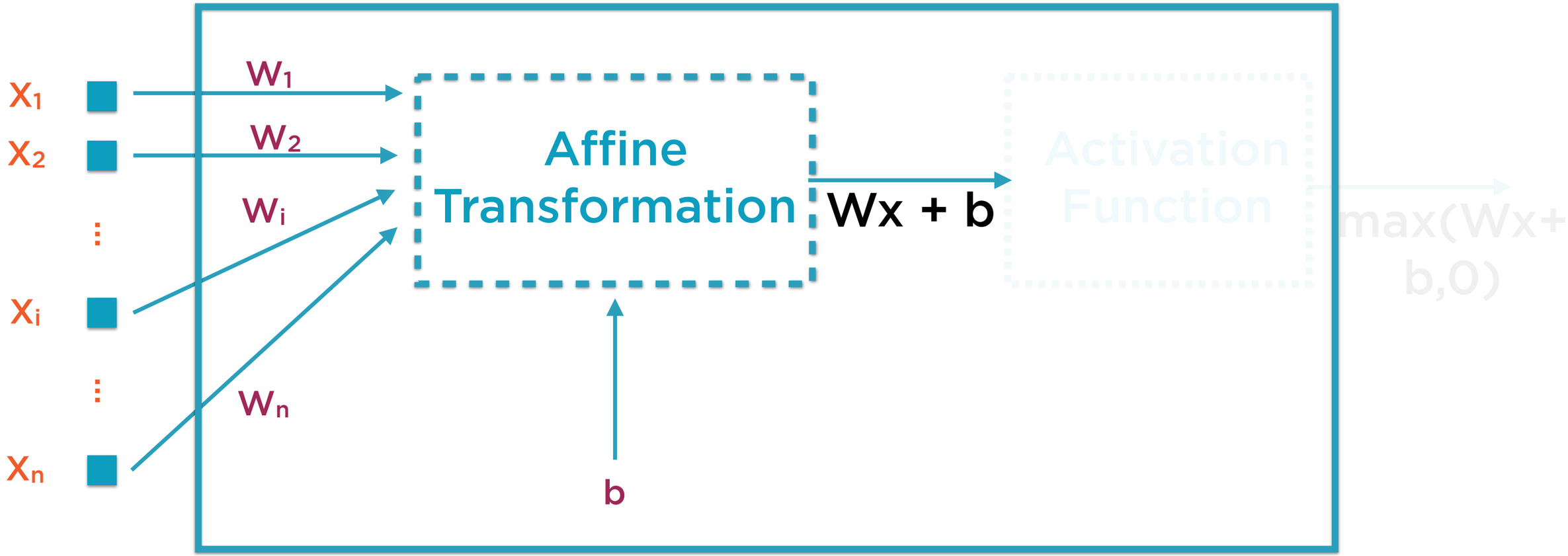
# Operation of a Single Neuron


**Each neuron only applies two simple functions to its inputs**

# Affine Transformation

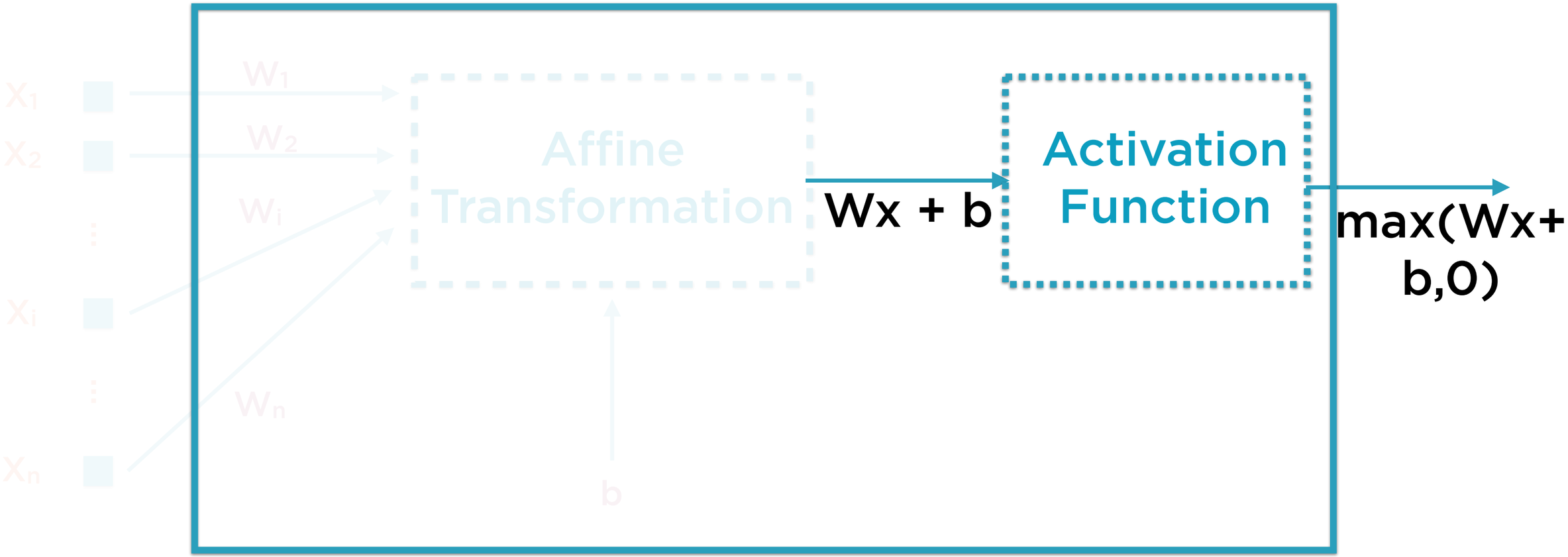

**The affine transformation alone can only learn linear relationships between the inputs and the output**

# Affine Transformation



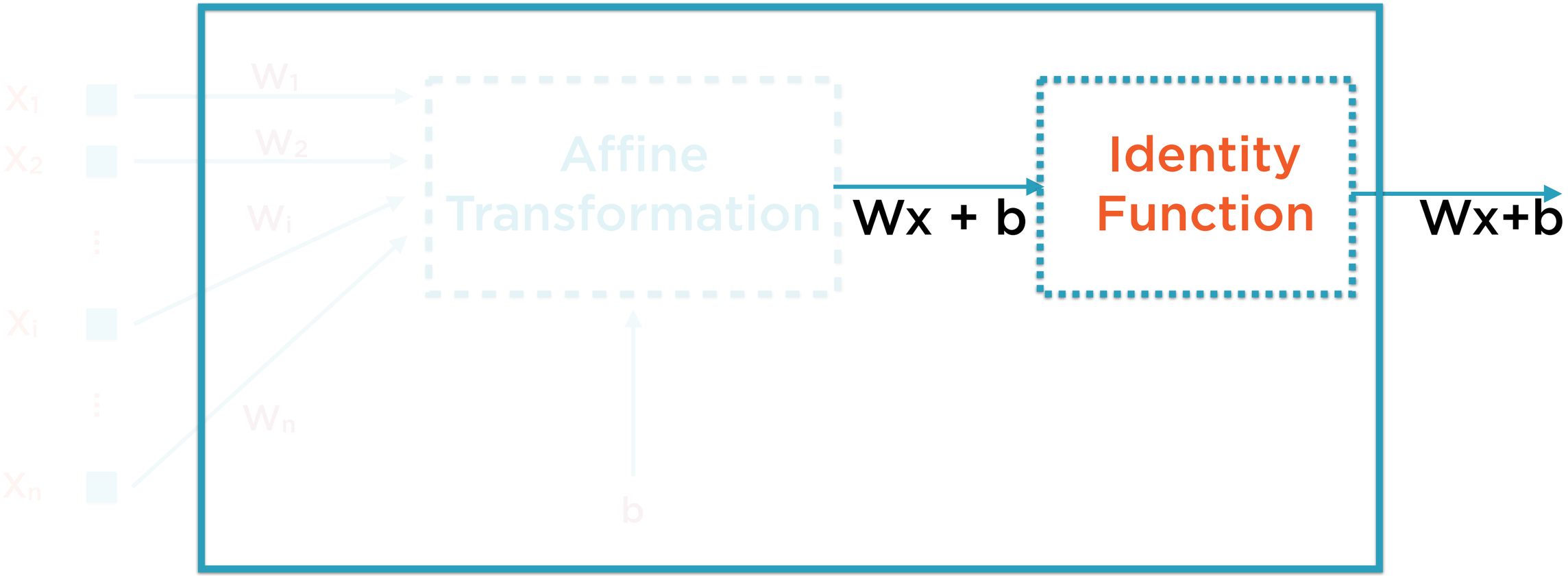

**The affine transformation is just a weighted sum with a bias added: $W_1x_1 + W_2x_2 + \ldots + W_nx_n + b$**

# Activation Function



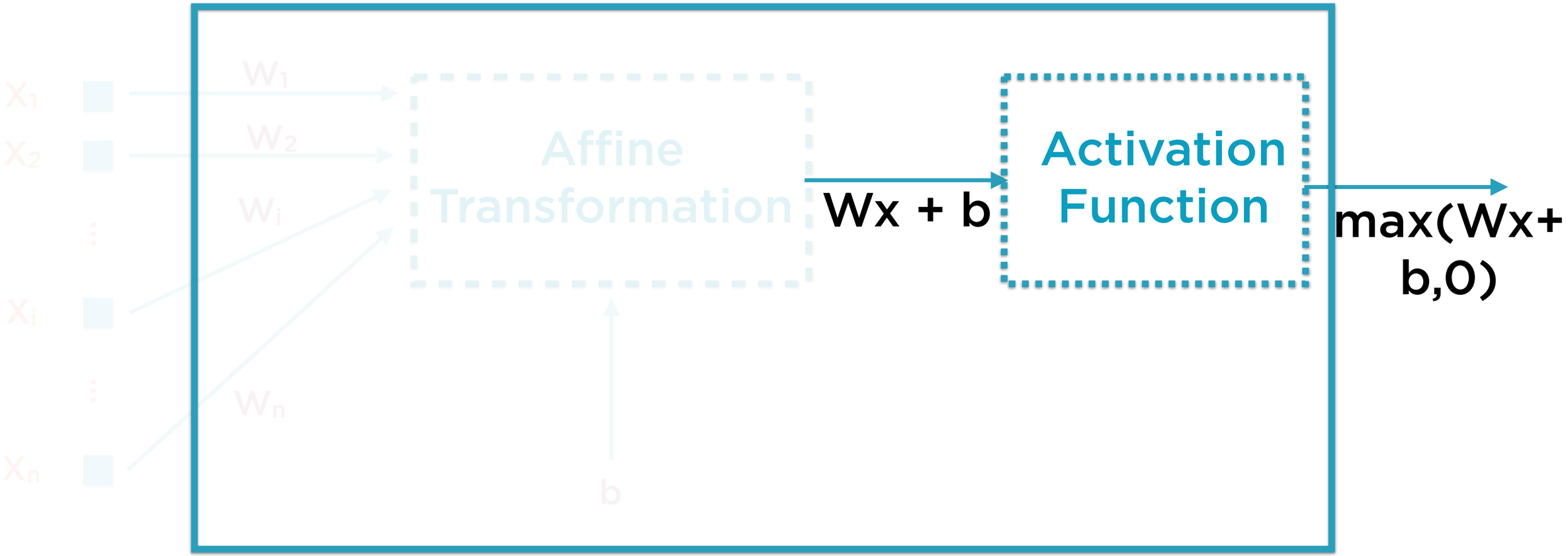

**A function which helps discover non-linear relationships**

# Linear Neuron




**When the activation function is the identity function, the neuron is often referred to as a linear neuron**

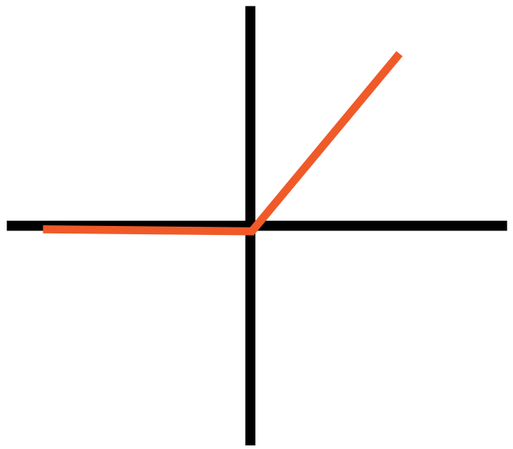
# Activation Function



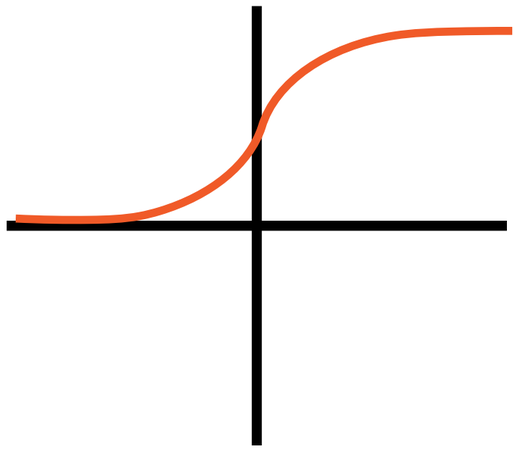

The combination of the affine transformation and the activation function can learn any arbitrary relationship

# Common Activation Functions

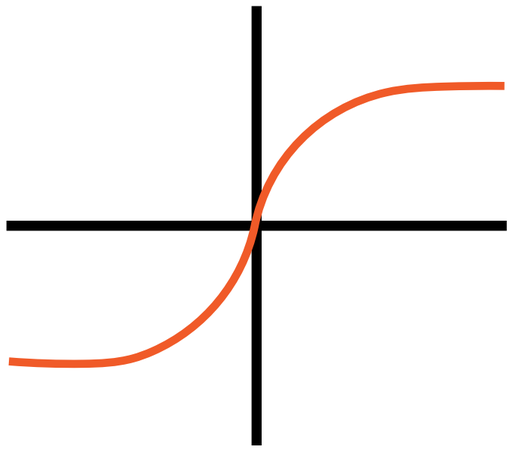
**ReLU**



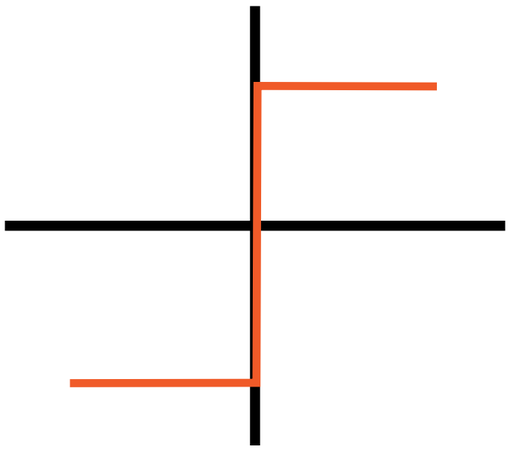
**logit**



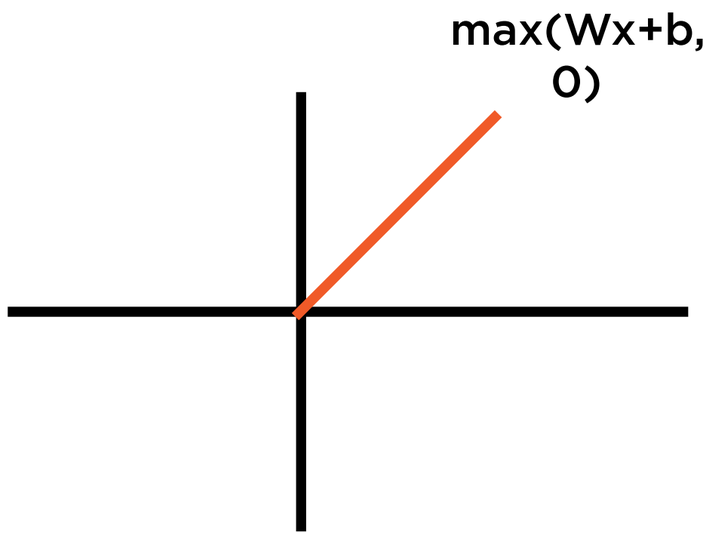
**tanh**



**step**



# ReLU Activation

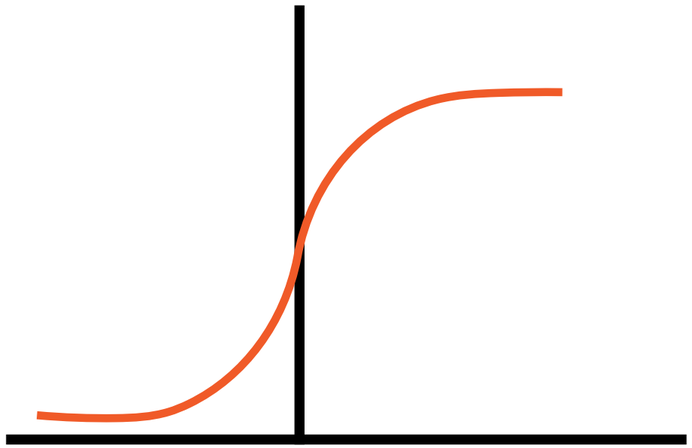



**The most common form of the activation function is the ReLU**

**ReLU : Rectified Linear Unit**

**ReLU(x) = max(0,x)**

# SoftMax Activation



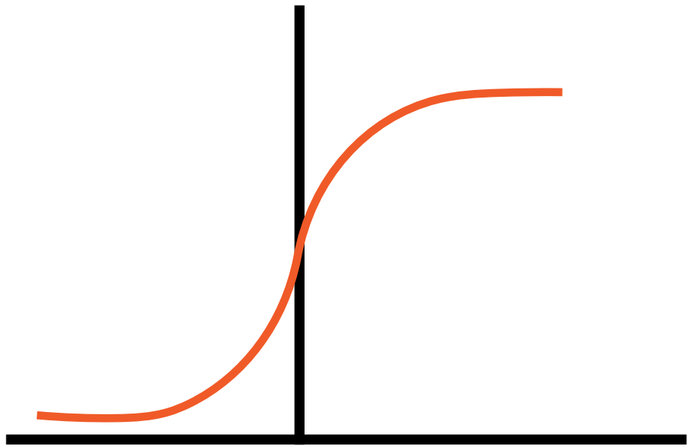
Another very common form of the activation function is the SoftMax

SoftMax(x) outputs a number between 0 and 1

This output can be interpreted as a probability

This curve is also called a logit curve

# Importance of Activation

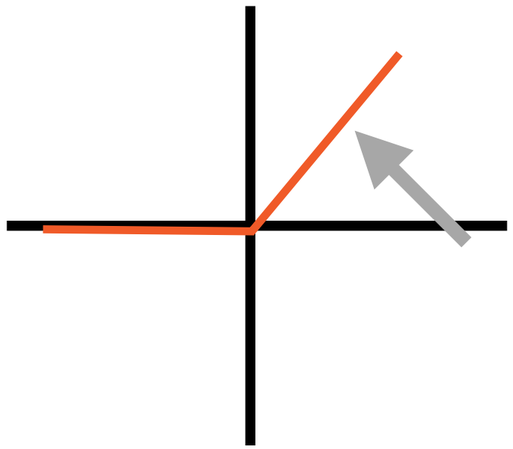


**The choice of activation function is crucial in determining performance**

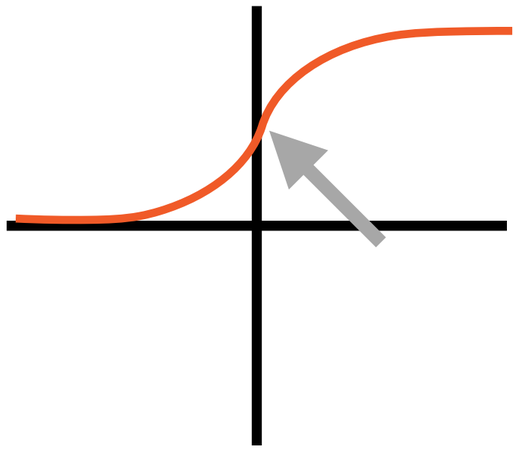
**To see why, we must understand the training process of a Neural Network**

# Active Region

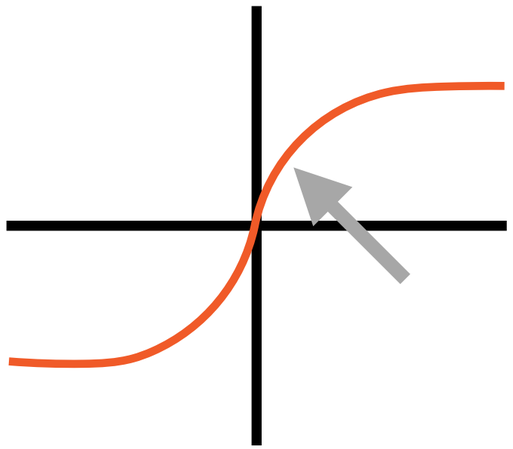
**ReLU**



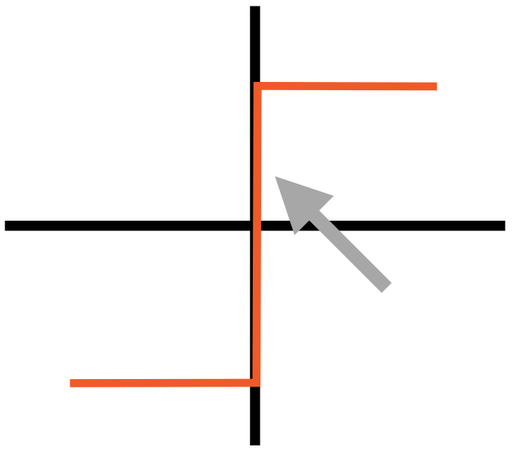
**logit**



**tanh**



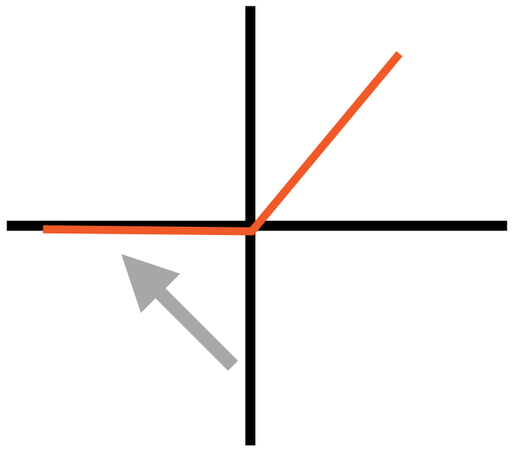
**step**



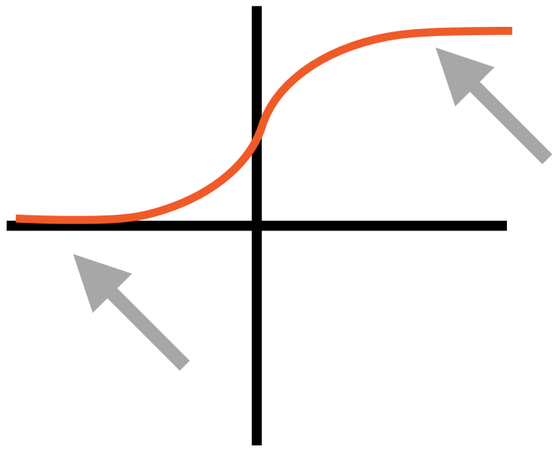
**Notice how activation functions have a gradient, this gradient allows them to be sensitive to input changes**

# Saturation

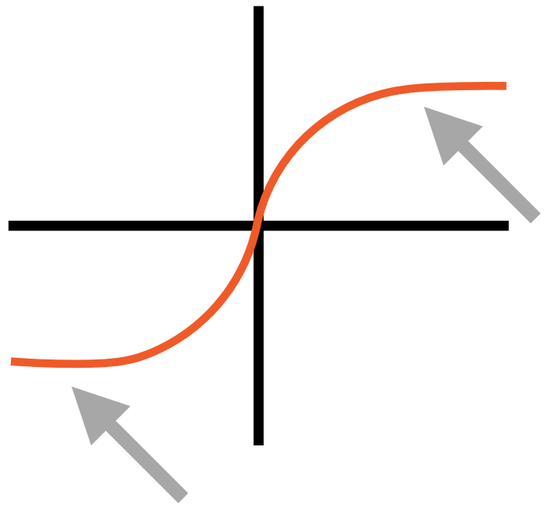
**ReLU**



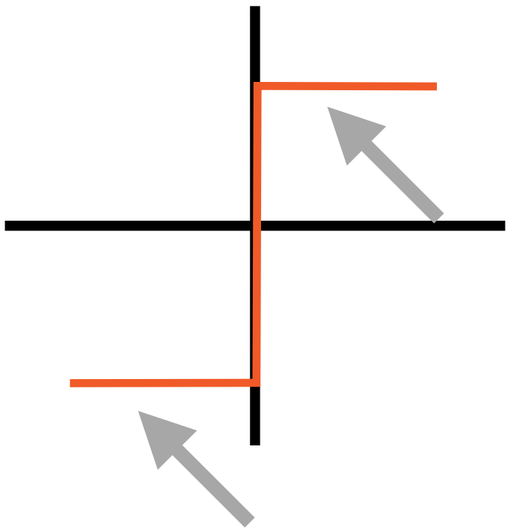
**logit**



**tanh**

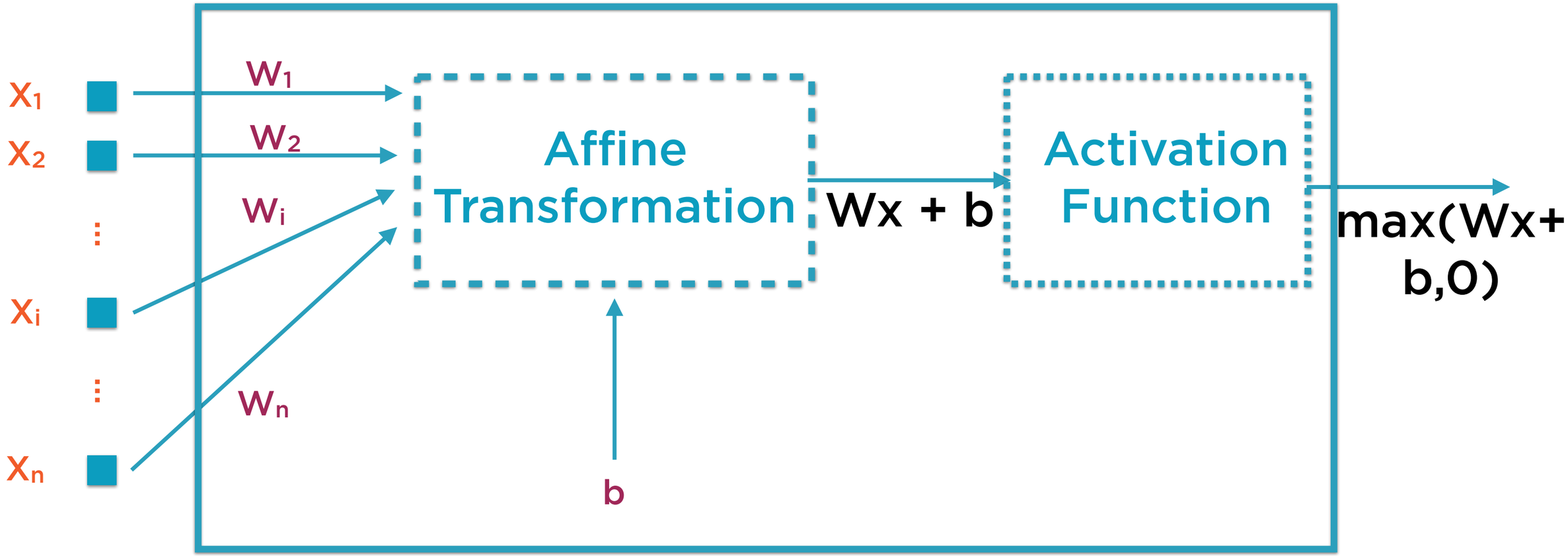


**step**



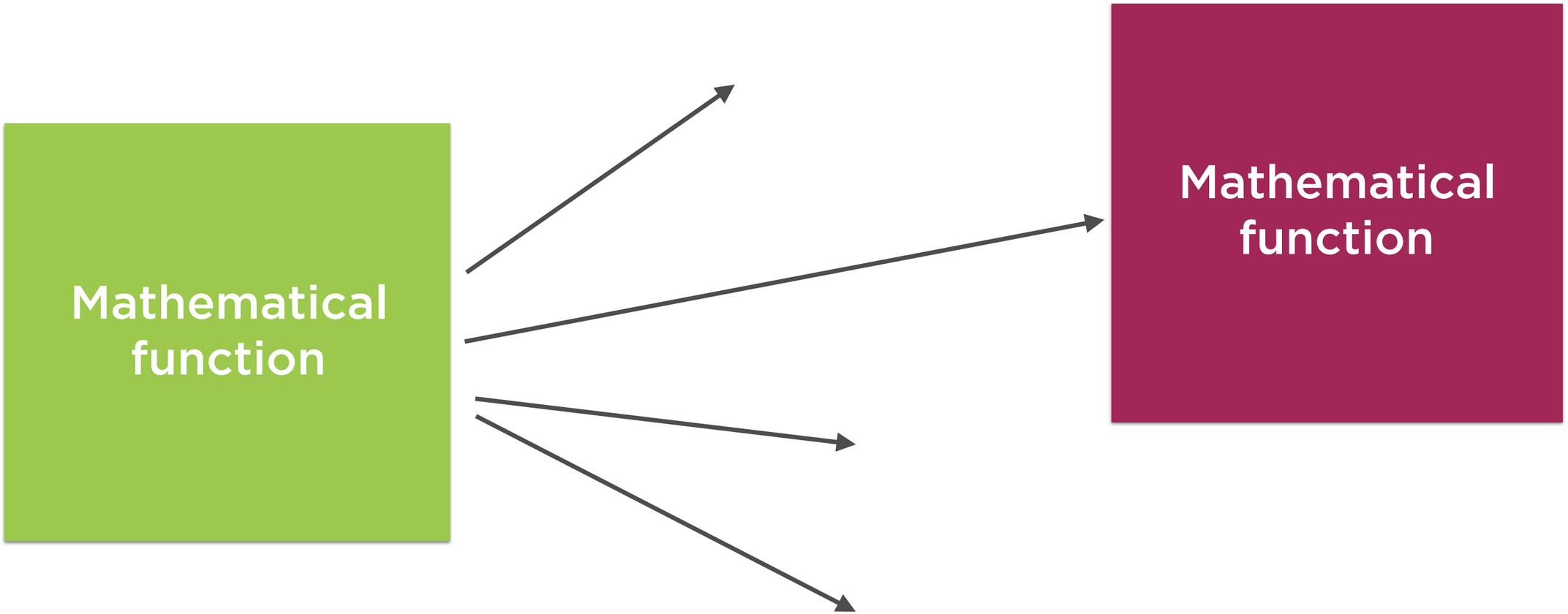
**In order to train and adjust the weights of the neural network the activation functions should operate in their active region**

# Neuron as a Learning Unit

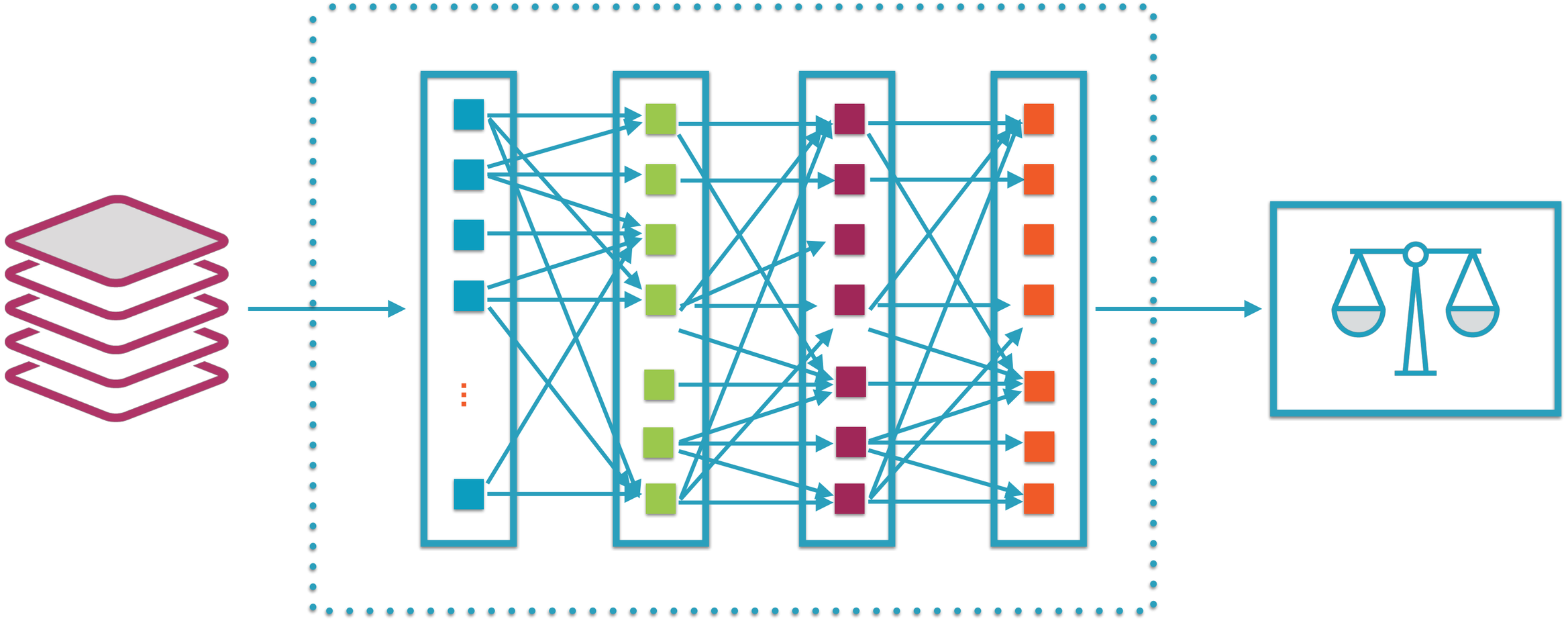



**Many of these simple neurons arranged in layers can do magical stuff**

# Interconnected Neurons in Layers



# Neural Networks



Corpus

ML-based Classifier

The **weights** and **biases** of individual neurons are determined during the **training** process

# Introducing PyTorch

# PyTorch

A deep learning framework for fast, flexible experimentation.

*https://pytorch.org/*

# PyTorch Background

**Relatively new**

- Initial release October 2016
- Stable release April 2018

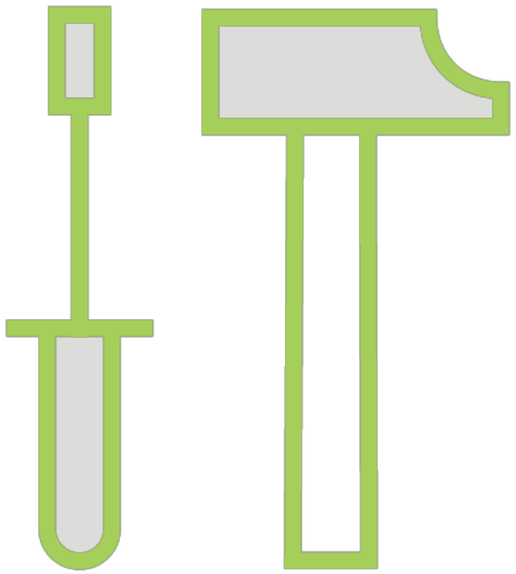
**Based on Torch**

- Open-source ML library (since 2002!)

**Facebook connection**

- Developed by Facebook AI researchers

# GPU-ready Tensor Library

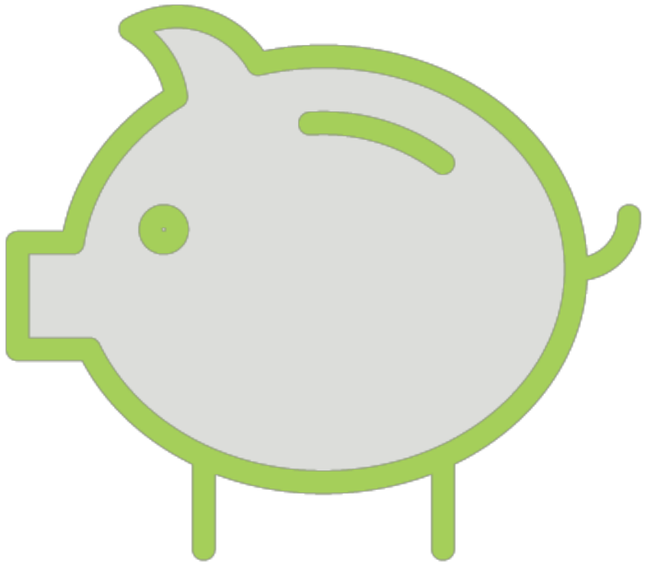


**Tensors for either CPU or GPU**

**Powerful, fast NumPy-like functionality**

- slicing
- indexing
- reductions
- linear algebra

# Tight Python Integration



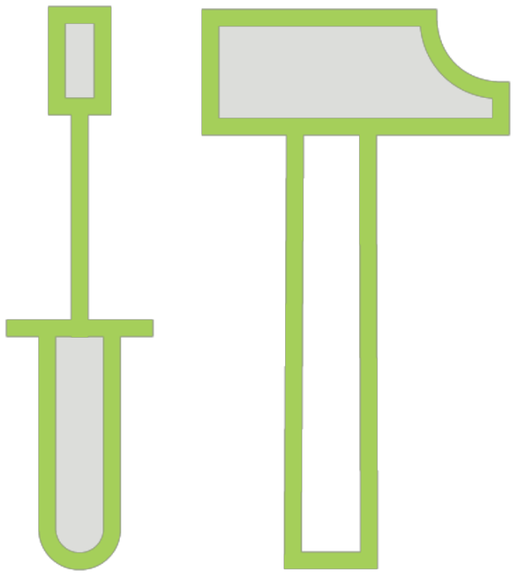
**Deeply tied to Python**

**Approach similar to NumPy/scikit-learn**

**Create neural networks in Python**

**Use existing Python libraries and debuggers**

# Deep Learning Framework

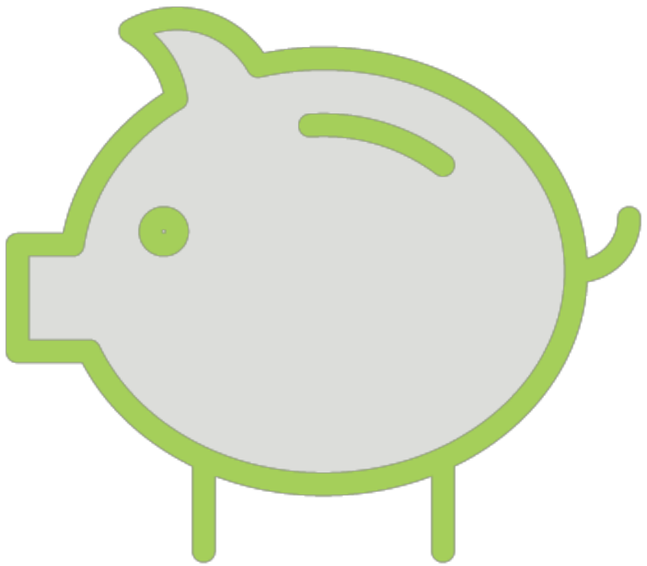


**“Tape-based Autograd”**

**Neural network can be redefined dynamically**

**Different from TensorFlow, CNTK**

# Imperative Execution

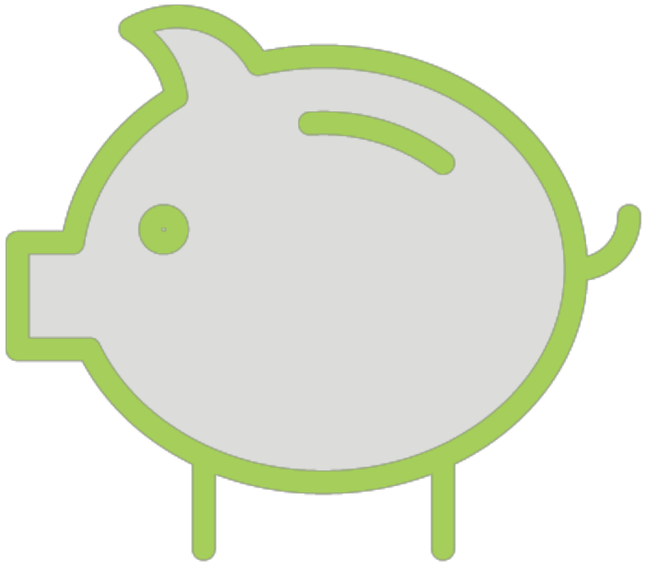


**Write code, run immediately**

**No separate build and run phases**

**Debugging easier**

# Ease of Extensibility



**Easily write new neural network layers**

**Different alternatives**

- Torch API
- Scipy
- C/C++ extension API

# TensorFlow and PyTorch

# TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. A comprehensive, flexible ecosystem of tools, libraries and community resources to easily build and deploy ML powered applications.

*https://tensorflow.org/*

# TensorFlow vs. PyTorch

| TensorFlow | PyTorch |
| --- | --- |
| Originally developed at Google by the Google Brain team | Originally developed by AI researchers at Facebook |
| First released in November 2015 | First released in October 2016 |
| Tensors as fundamental data structures for computation | Tensors as fundamental data structures for computation |
| CUDA support for GPUs | CUDA support for GPUs |

# TensorFlow vs. PyTorch

## TensorFlow

Computation graph is static

Must be defined before being run

tf.Session for separation from Python

## PyTorch

Computation graph is dynamic

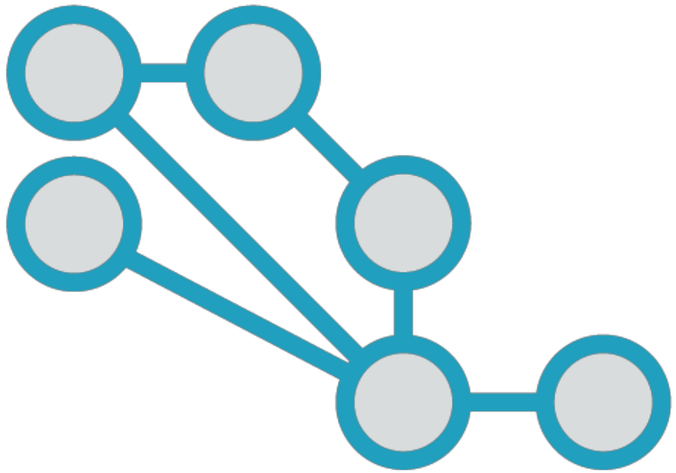
Can be defined and run as you go

Tightly integrated with Python

# TensorFlow vs. PyTorch

| TensorFlow | PyTorch |
| --- | --- |
| Debugging via tfdbg | Debugging with PyCharm, pdb |
| Visualization using built-in TensorBoard | Visualization using matplotlib, seaborn |
| Deployment using TF Serving | Need to set up REST API e.g. Flask |
| tf.device and tf.DeviceSpec to use GPUs (relatively hard) | torch.nn.DataParallel to use GPUs (relatively easy) |

# Learning From PyTorch



**TensorFlow now has eager execution mode for dynamic graph execution**

**Higher level abstraction to build neural network layers using the Keras API**

# Demo

**Installation and setup of PyTorch**

# Summary

Deep learning using neural networks

Neurons and activation functions

Introducing PyTorch to build neural networks

Install and set up PyTorch on your local machine