Understanding the Foundations of TensorFlow

INTRODUCING TENSORFLOW

Overview

Introduce TensorFlow(TF), a language for numerical computations

Understand the basics of machine learning, deep learning and neural networks

Learn why TF is slowly becoming the default library for ML

Install and set up TensorFlow on your local machine

What You Need in Your Toolkit

Prerequisites

Familiarity with the command line on a Mac, Linux or Windows machine

Comfortable with writing programs in Python

Install and Setup

The latest version of TensorFlow 1.2rc0

A compatible version of Python, version 2.7 and 3.x

- Only works with versions >3.5 on Windows

A Mac, Linux or Windows machine on which TensorFlow can be installed

Course Overview

Introduction to TensorFlow, install and set up

Basics of TensorFlow, computation graphs, tensors, sessions and TensorBoard

Fundamentals of TensorFlow, placeholders, variables, the feed dictionary

Working with images, representing RGB and grayscale images, image operations

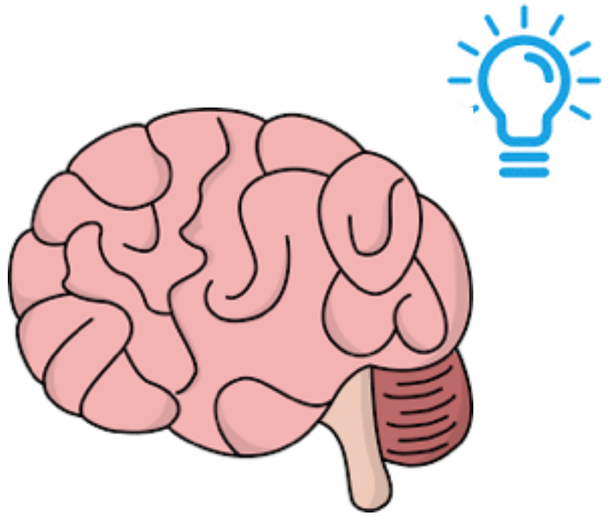
Machine Learning with TensorFlow, identifying handwritten digits in the MNIST dataset using the nearest neighbors algorithm

Understanding Machine Learning

Machine Learning

Work with a huge maze of data

Find patterns



Make intelligent decisions

A machine learning algorithm is an algorithm that is able to learn from data

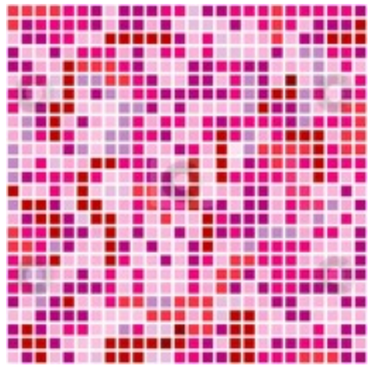
Machine Learning

Emails on a server

Spam or Ham?

Trash or Inbox

Machine Learning



Images represented as pixels

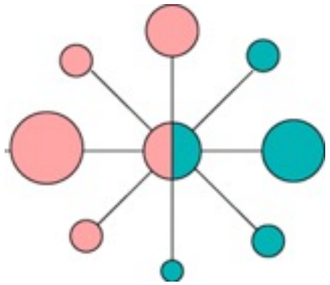
Identify edges, colors, shapes

A photo of a little bird

Types of Machine Learning Problems

Classification

Regression



Clustering

Rule-extraction

Types of Machine Learning Problems

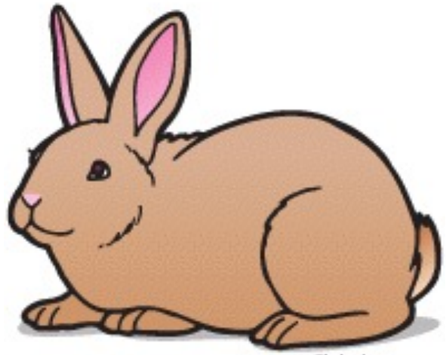
Classification

Regression

Clustering

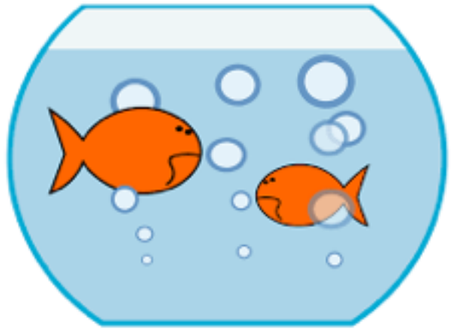
Rule-extraction

Whales: Fish or Mammals?



Mammals

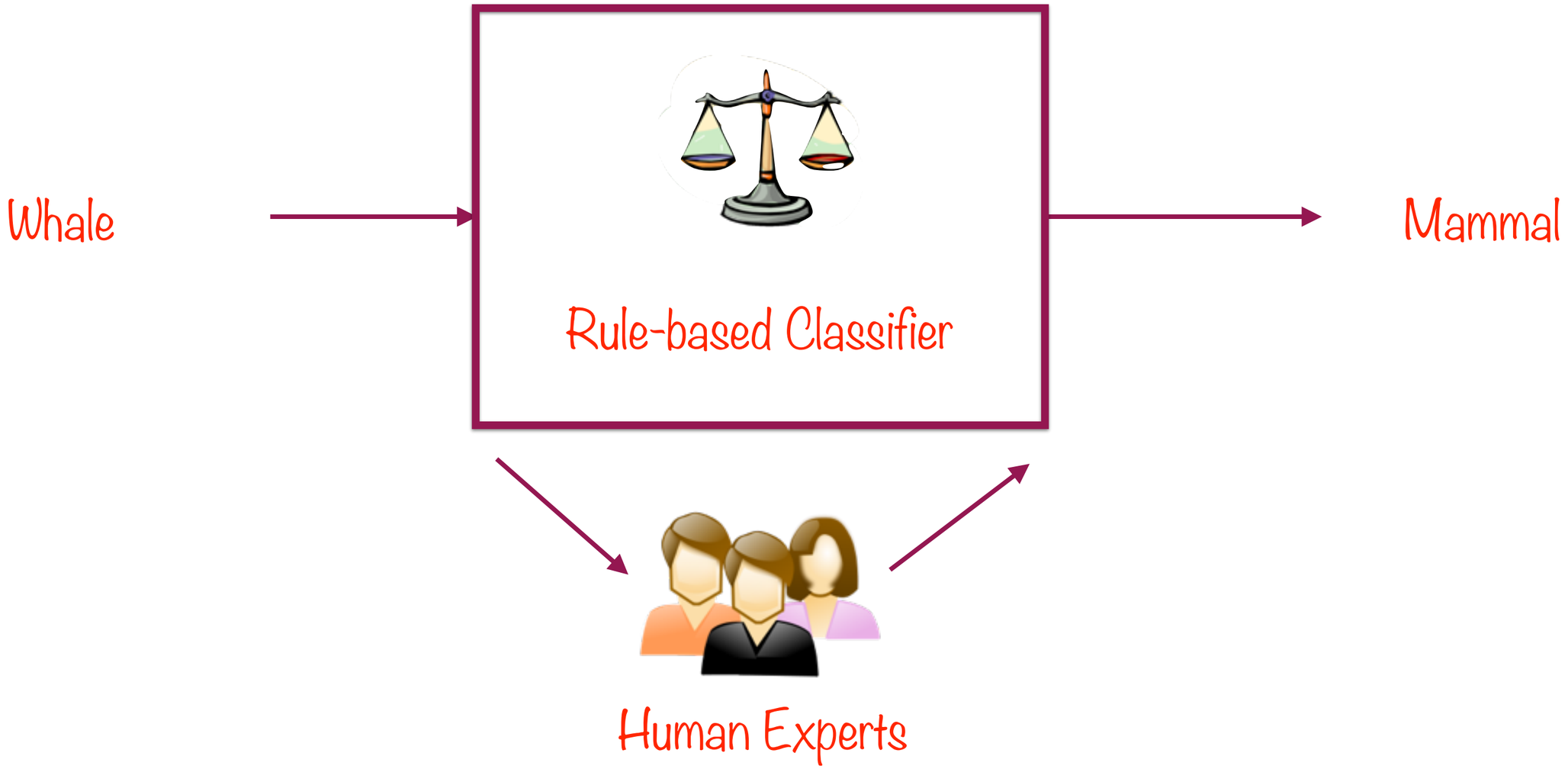
Members of the infraorder Cetacea



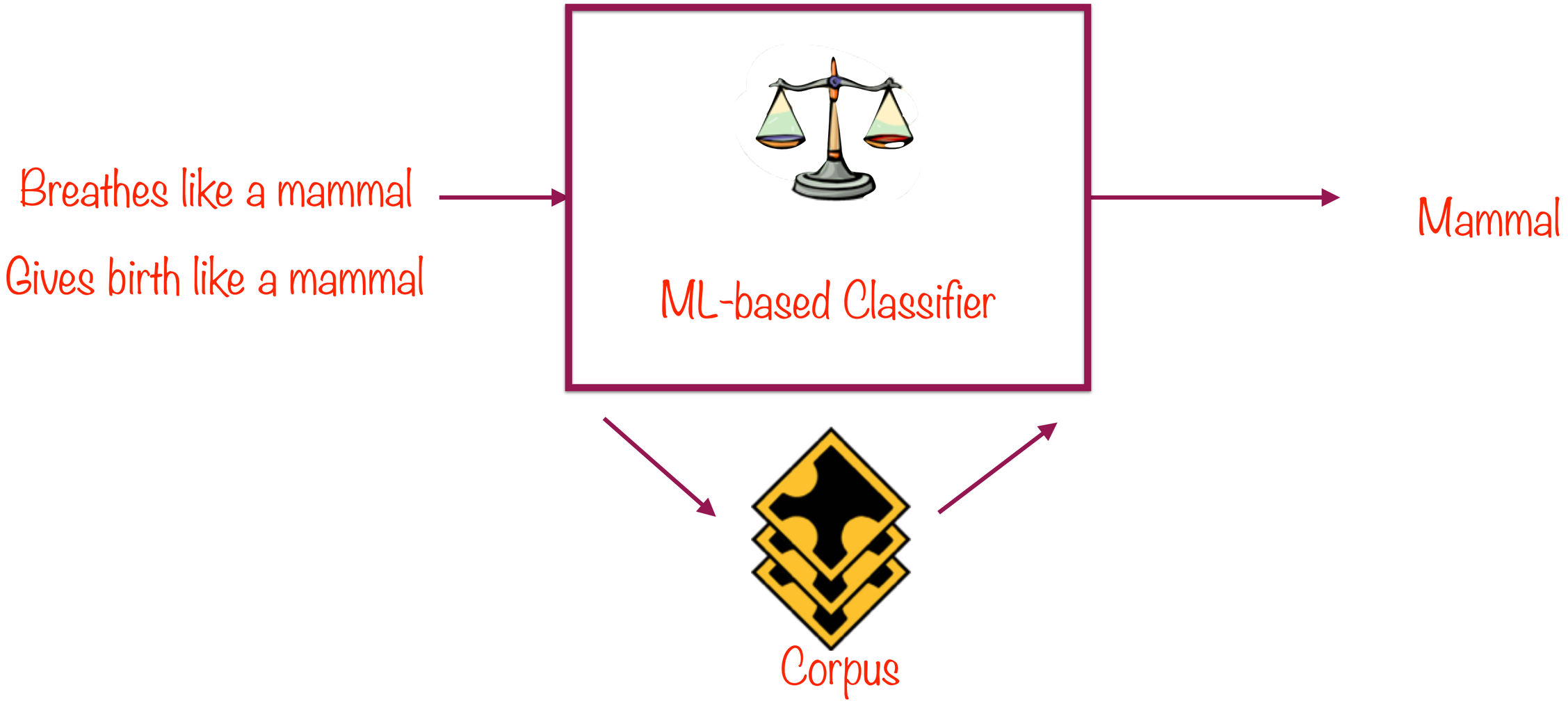
Fish

Look like fish, swim like fish, move with fish

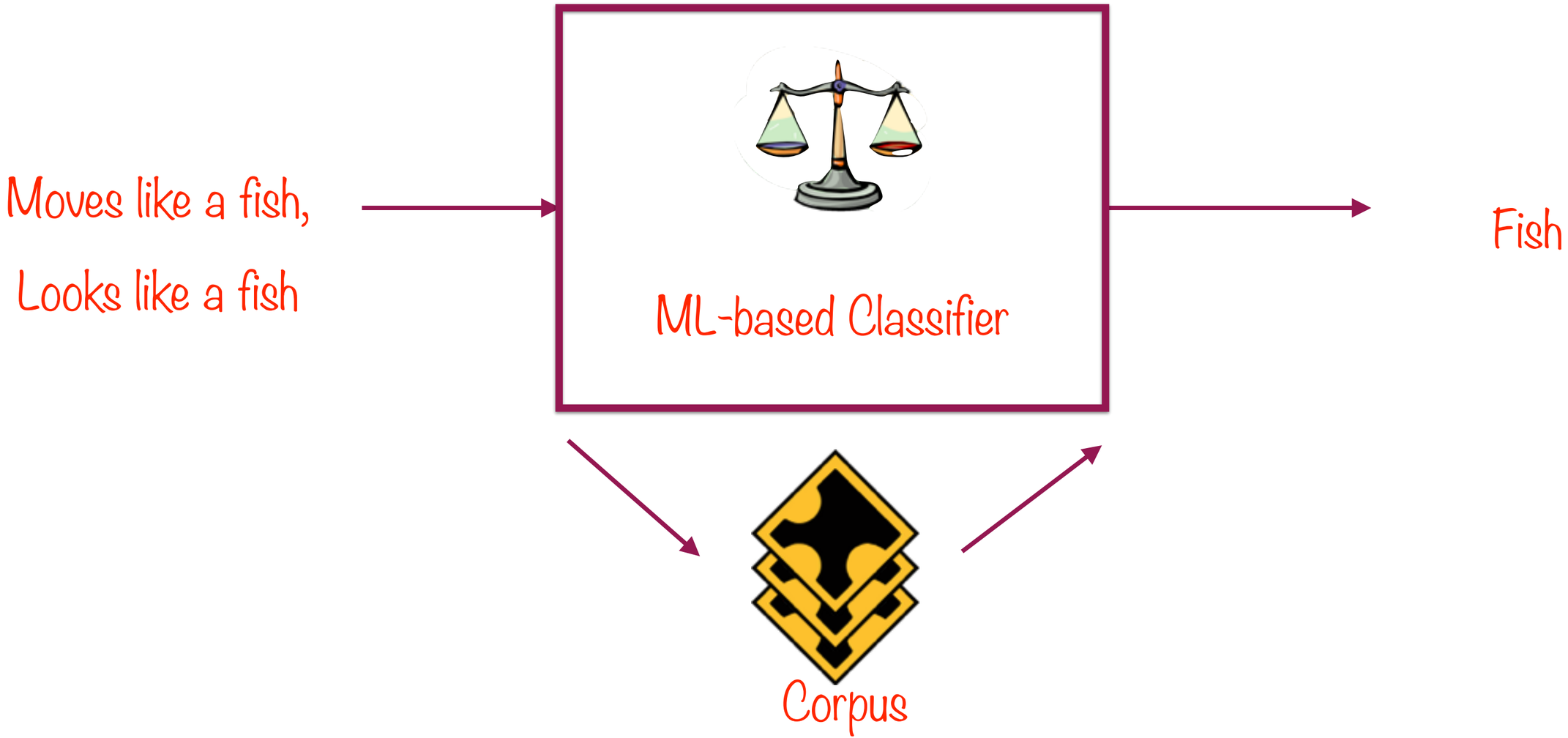
Rule-based Binary Classifier



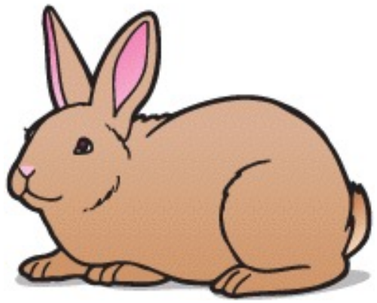
ML-based Binary Classifier

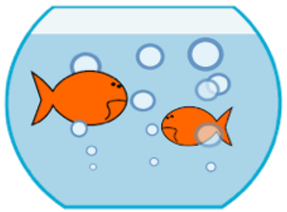


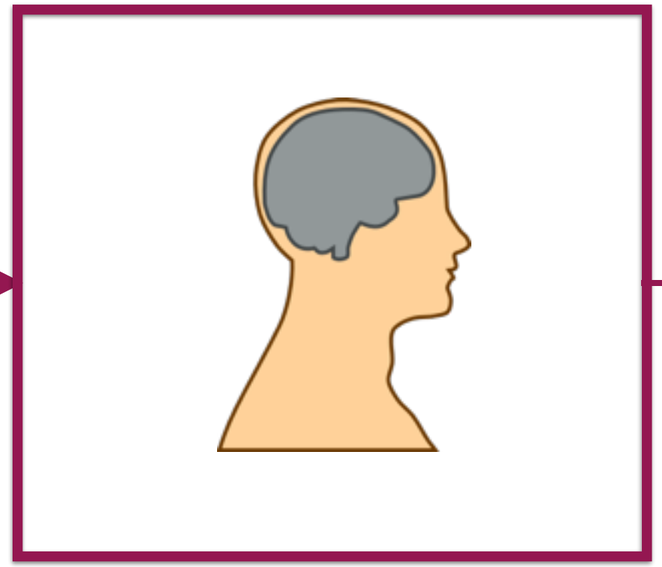
"Traditional" ML-based Binary Classifier



ML-based Binary Classifier







Corpus

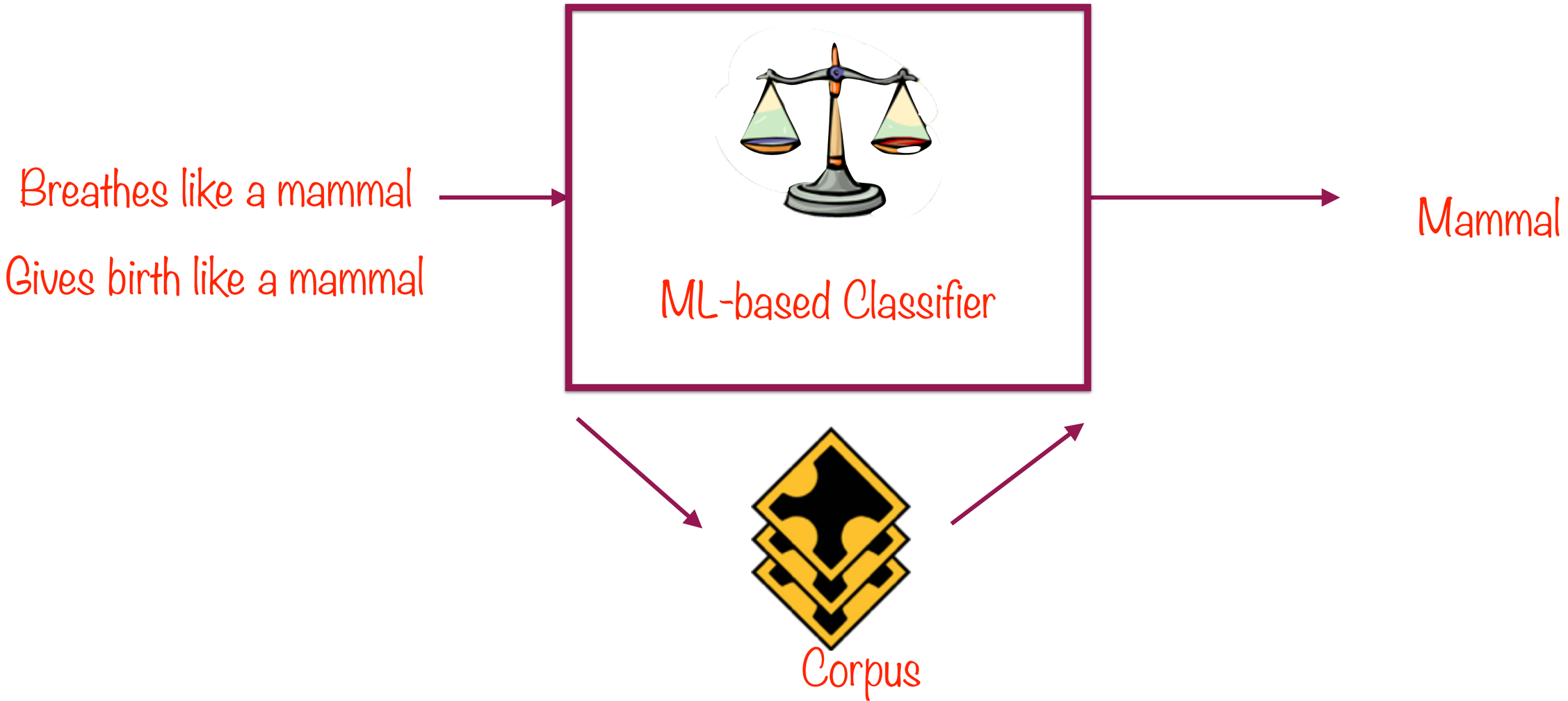
Classification Algorithm

ML-based Classifier

ML-based Binary Classifier

ML-based	Rule-based
Dynamic	Static
Experts optional	Experts required
Corpus required	Corpus optional
Training step	No training step

"Traditional" ML-based Binary Classifier



"Traditional" ML-based Binary Classifier

Breathes like a mammal

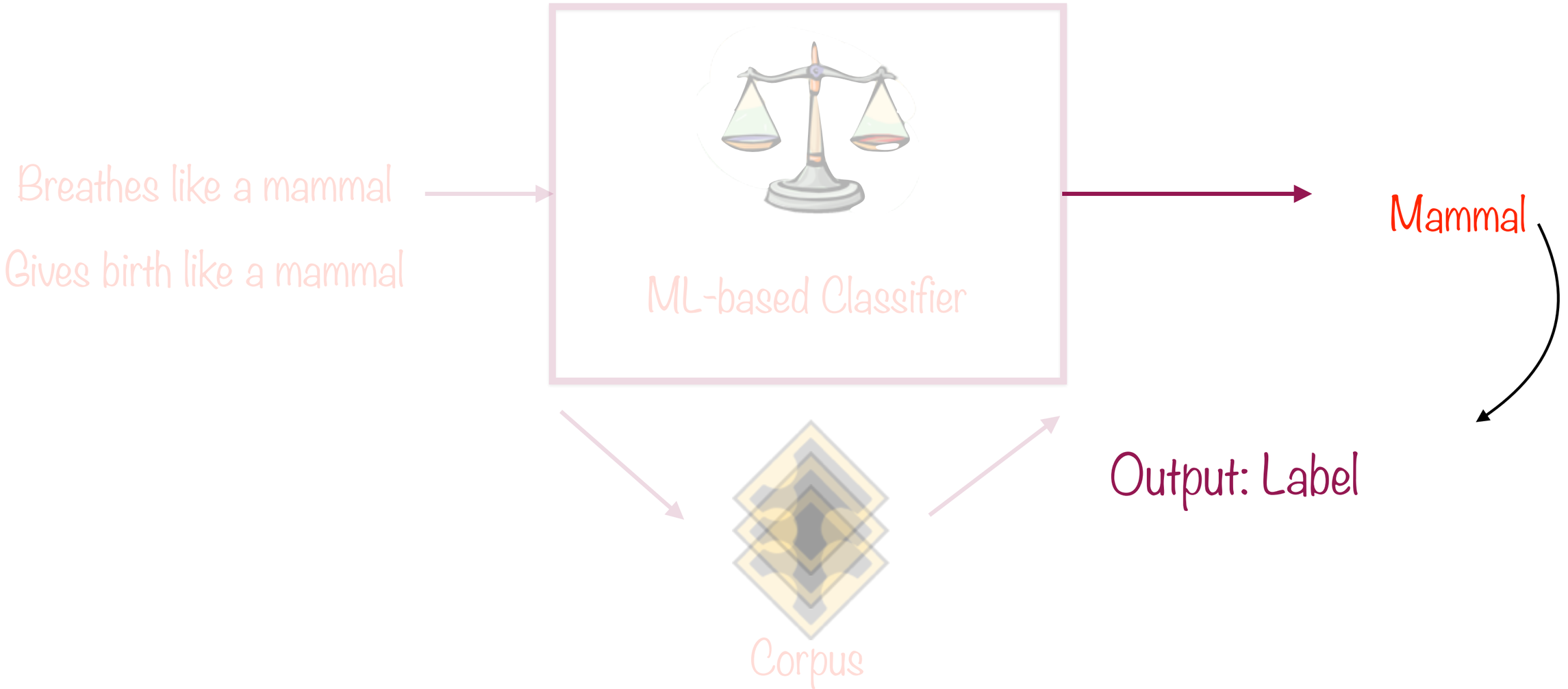
Gives birth like a mammal

Mammal

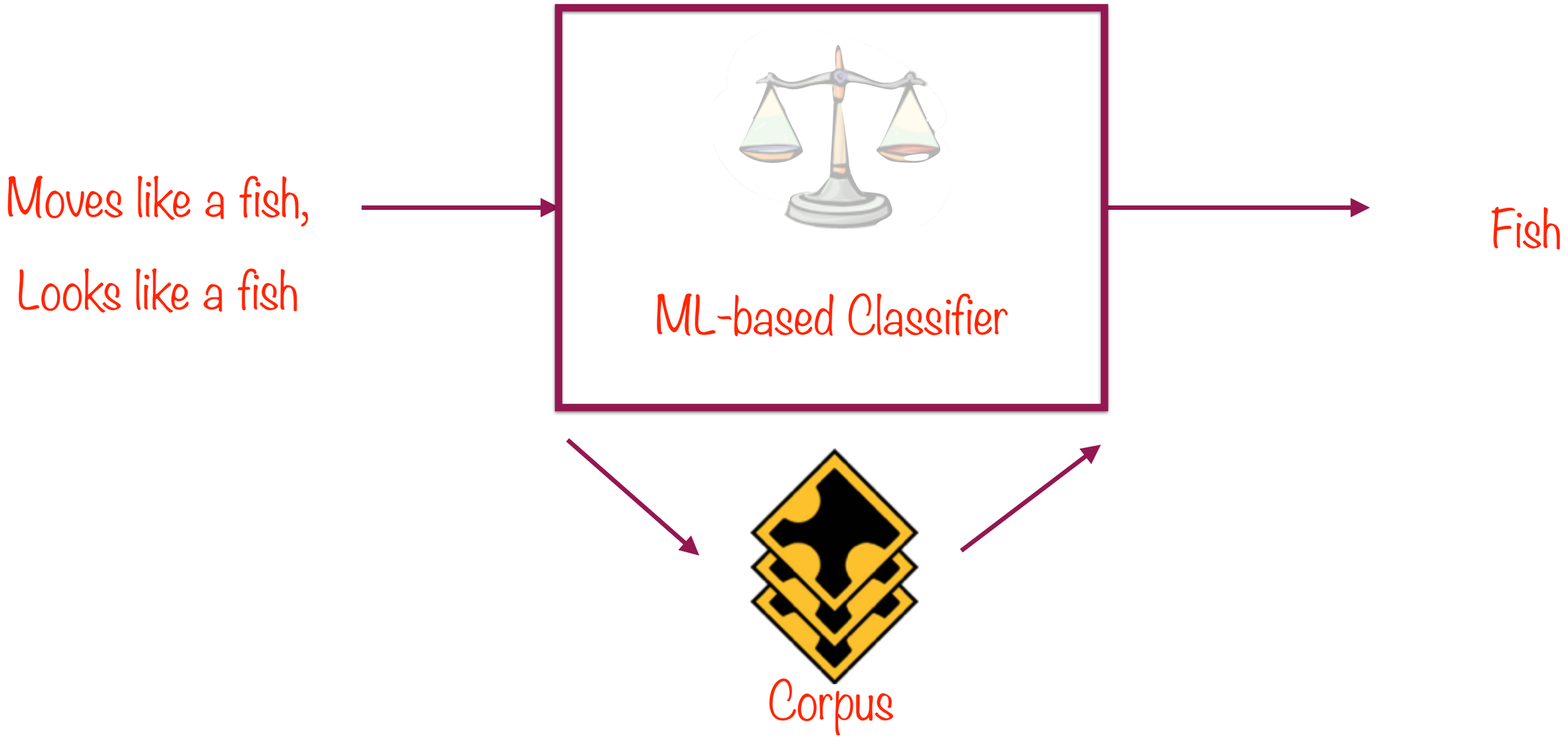
Input: Feature Vector

Corpus

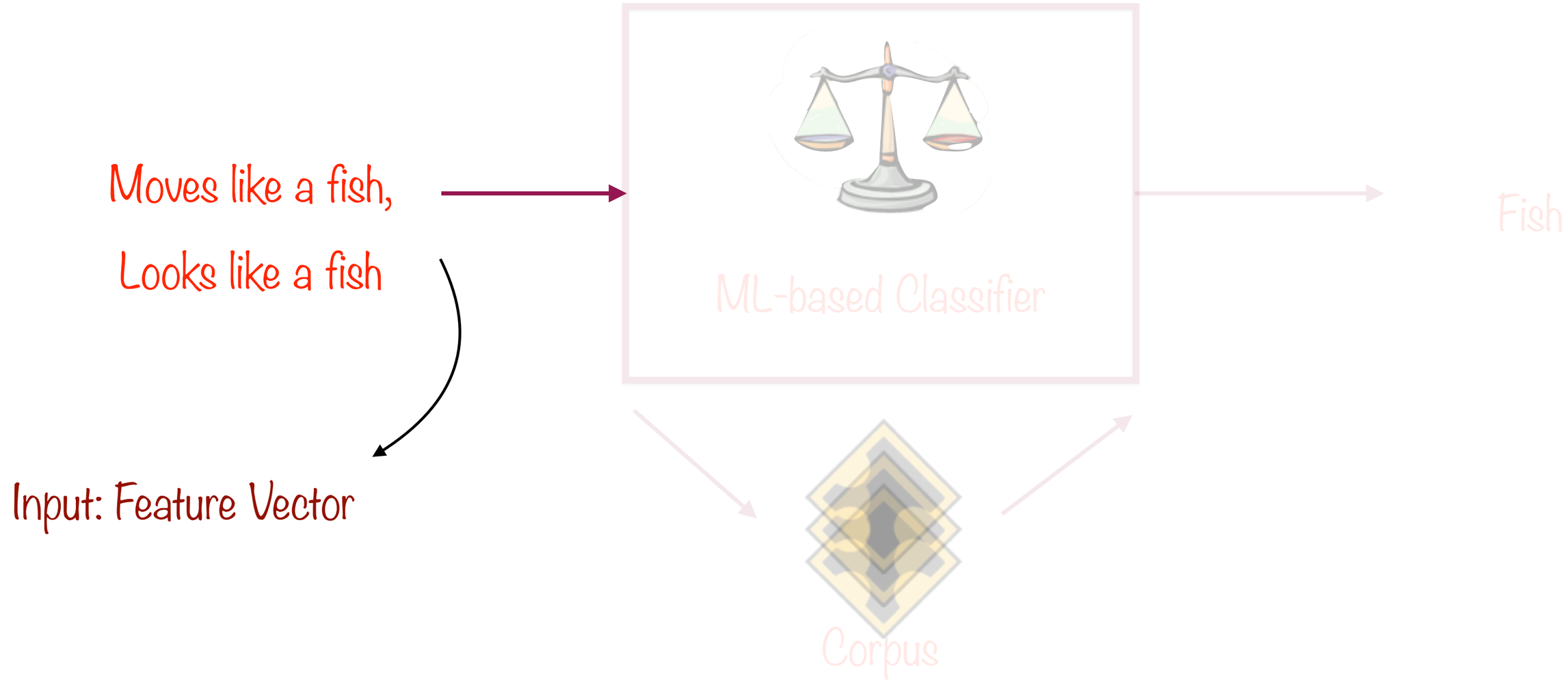
"Traditional" ML-based Binary Classifier



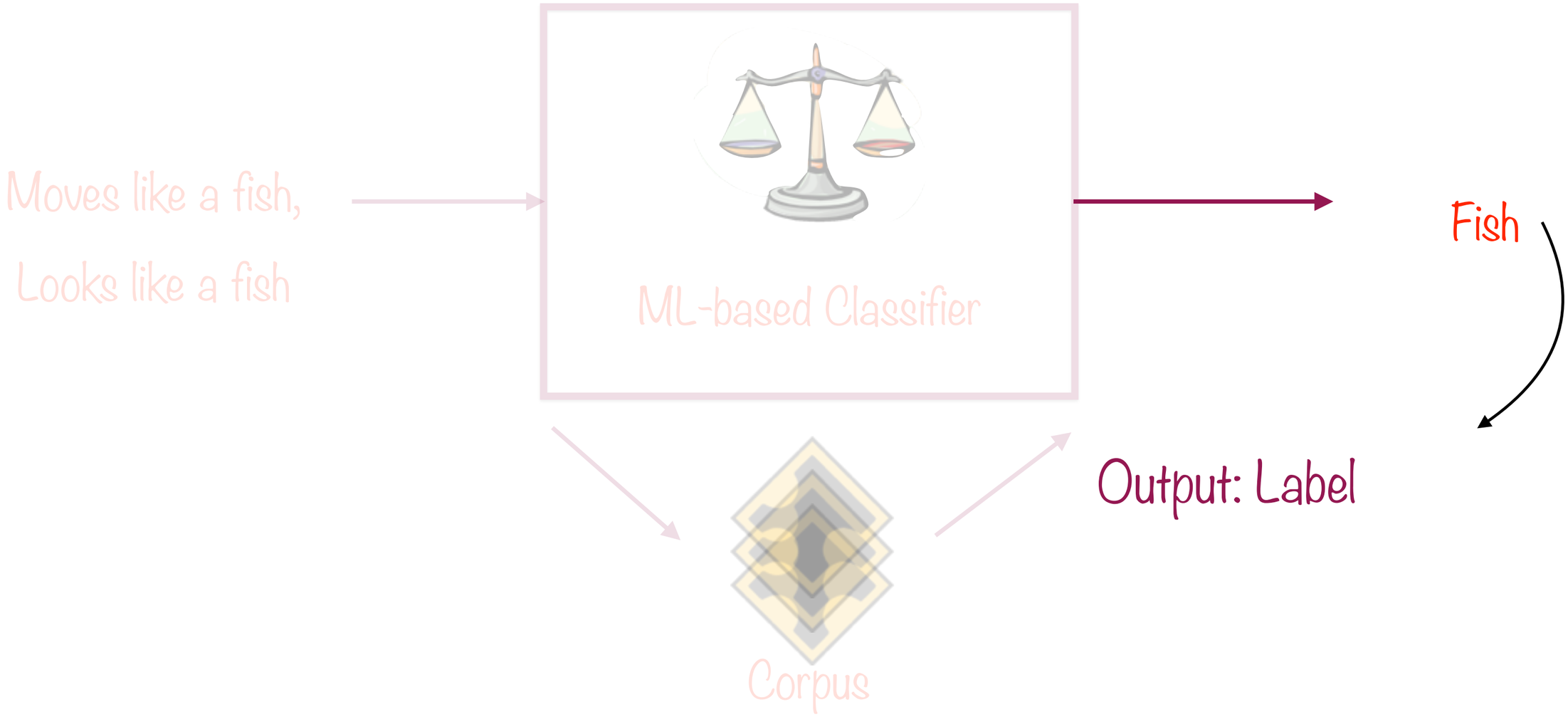
"Traditional" ML-based Binary Classifier



"Traditional" ML-based Binary Classifier



"Traditional" ML-based Binary Classifier



Feature Vectors

The attributes that the ML algorithm focuses on are called features

Each data point is a list - or vector - of such features

Thus, the input into an ML algorithm is a feature vector

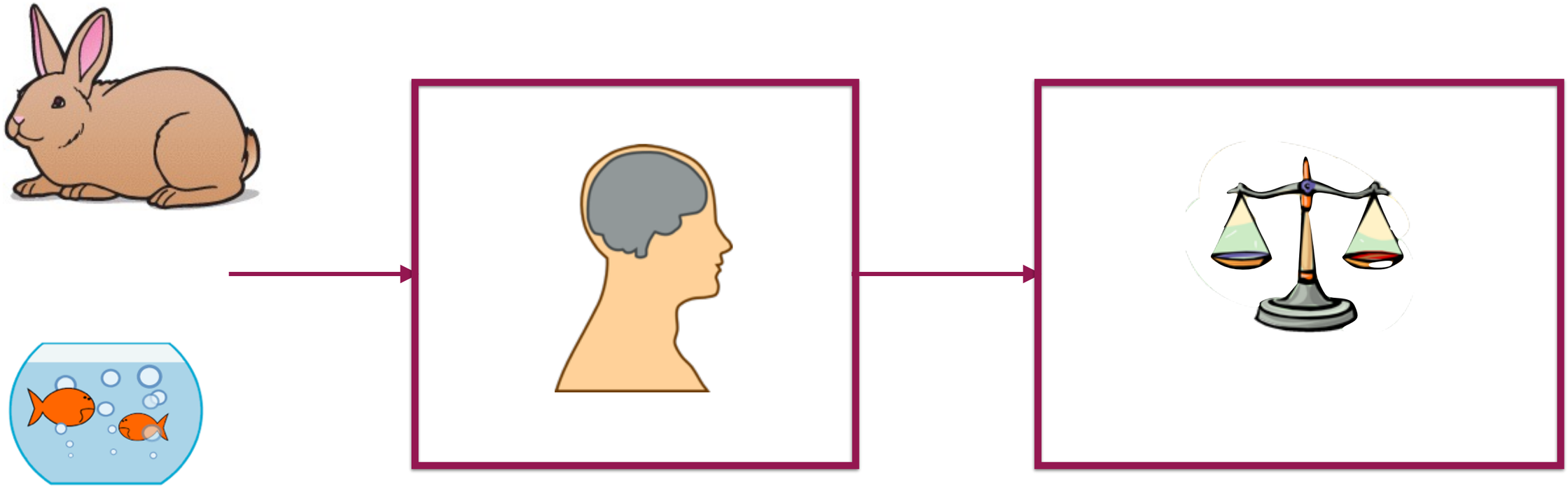
"Traditional" ML-based systems still rely on experts to decide what features to pay attention to

"Representation" ML-based systems figure out by themselves what features to pay attention to

Understanding Deep Learning

"Representation" ML-based systems figure out by themselves what features to pay attention to

"Traditional" ML-based Binary Classifier

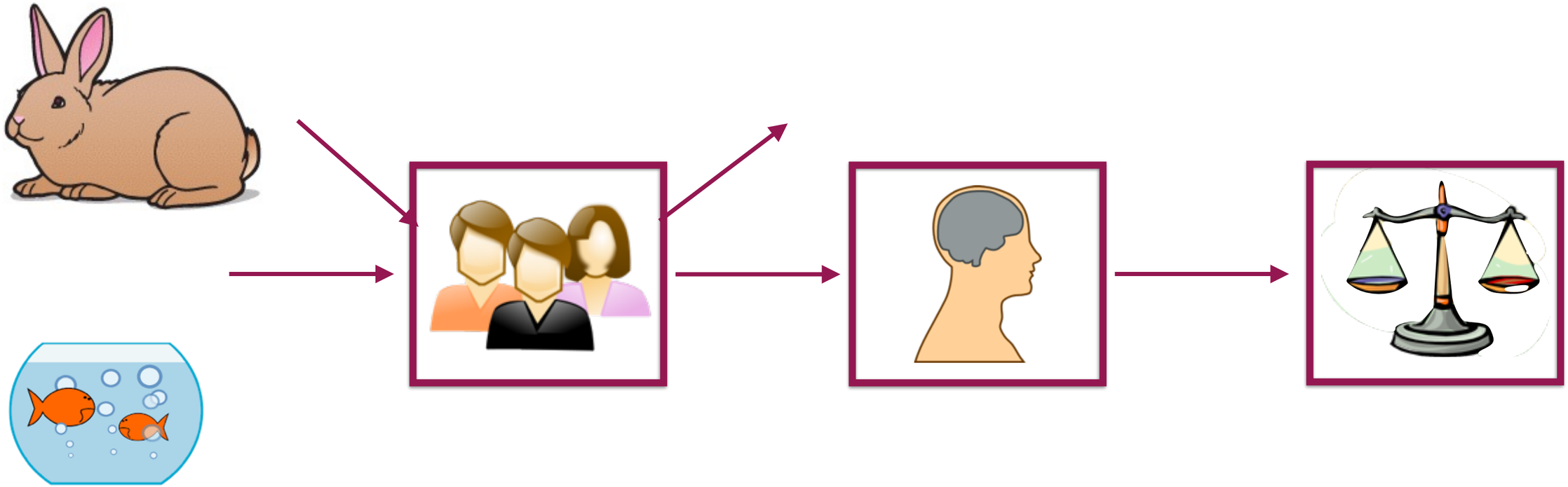


Corpus

Classification Algorithm

ML-based Classifier

"Traditional" ML-based Binary Classifier



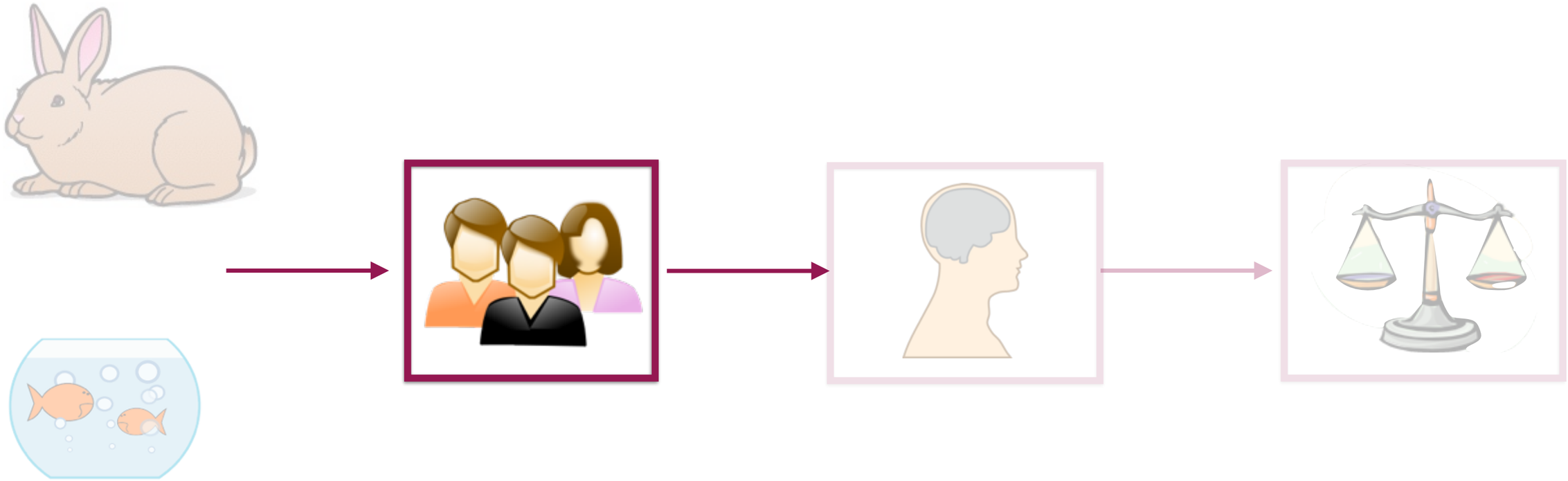
Corpus

Feature Selection by Experts

Classification Algorithm

ML-based Classifier

"Traditional" ML-based Binary Classifier



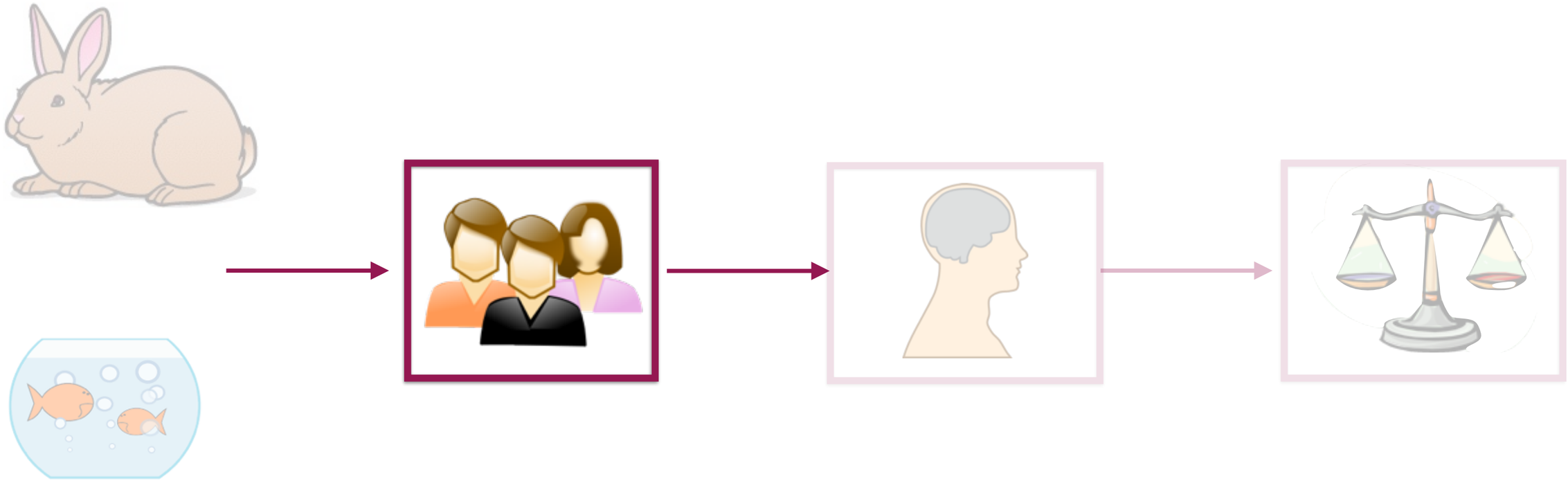
Corpus

Feature Selection by Experts

Classification Algorithm

ML-based Classifier

"Traditional" ML-based Binary Classifier



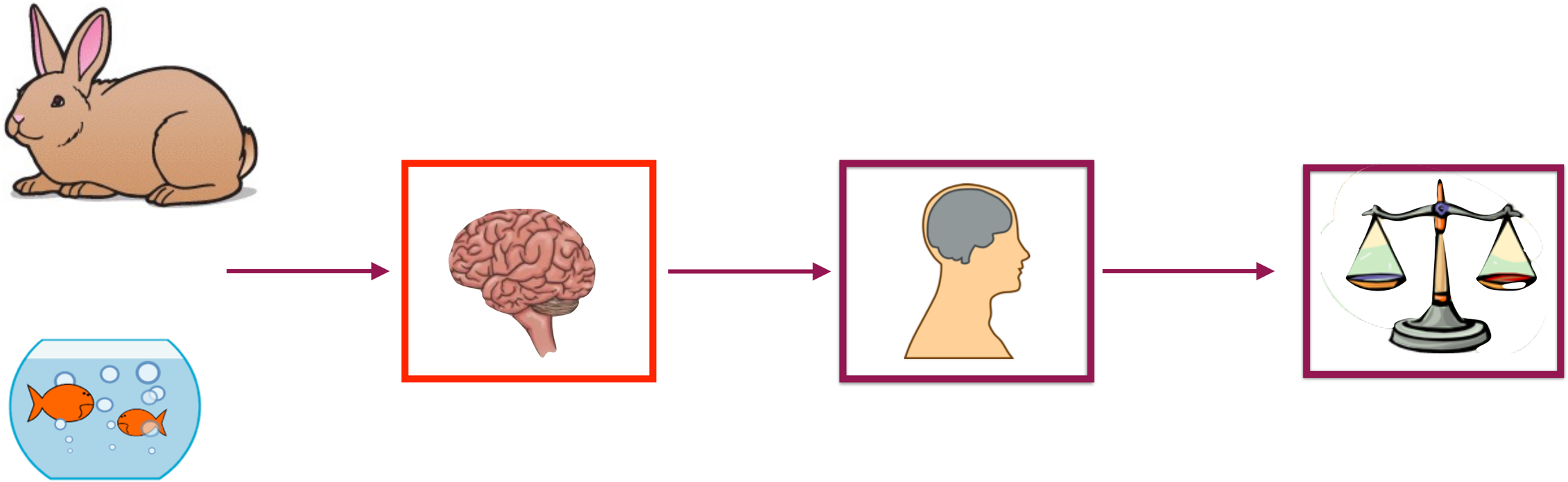
Corpus

Feature Selection by Experts

Classification Algorithm

ML-based Classifier

"Representation" ML-based Binary Classifier



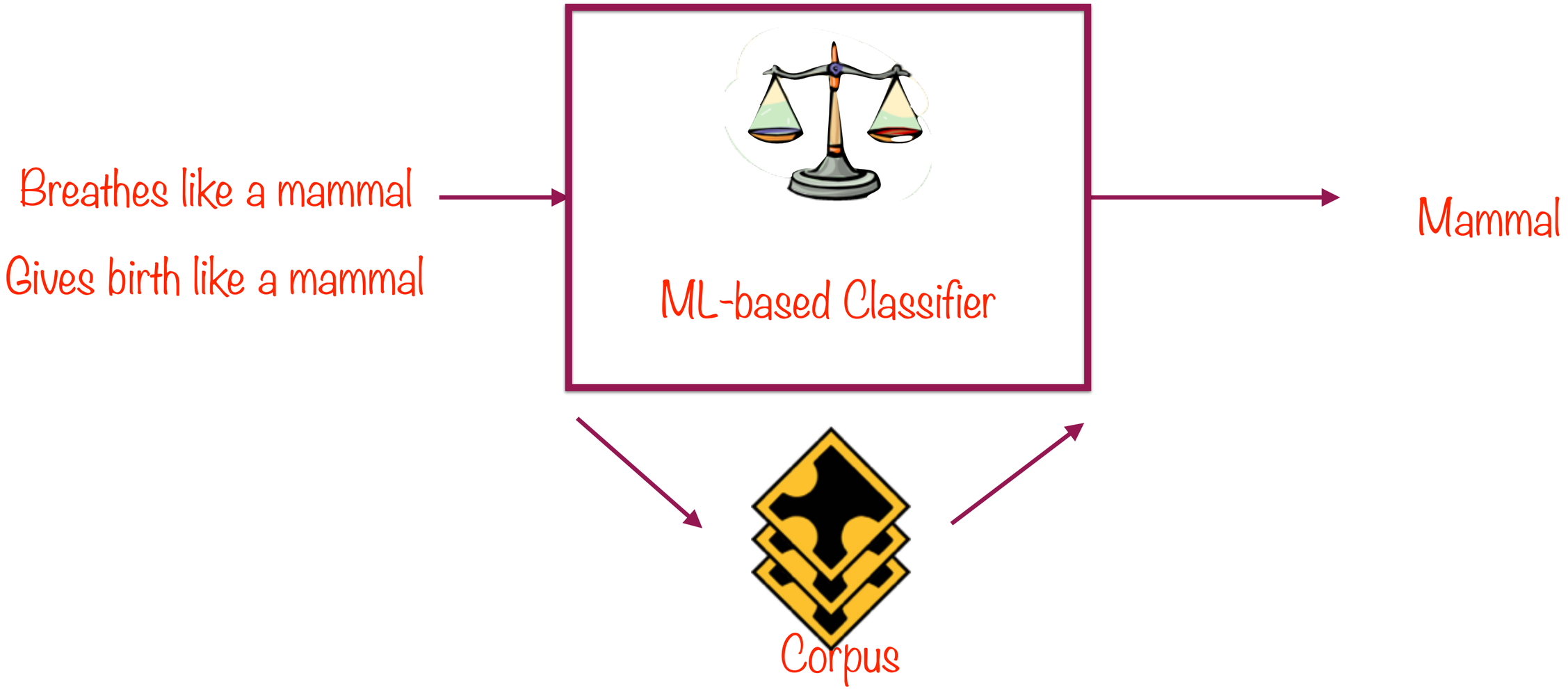
Corpus

Feature Selection Algorithm

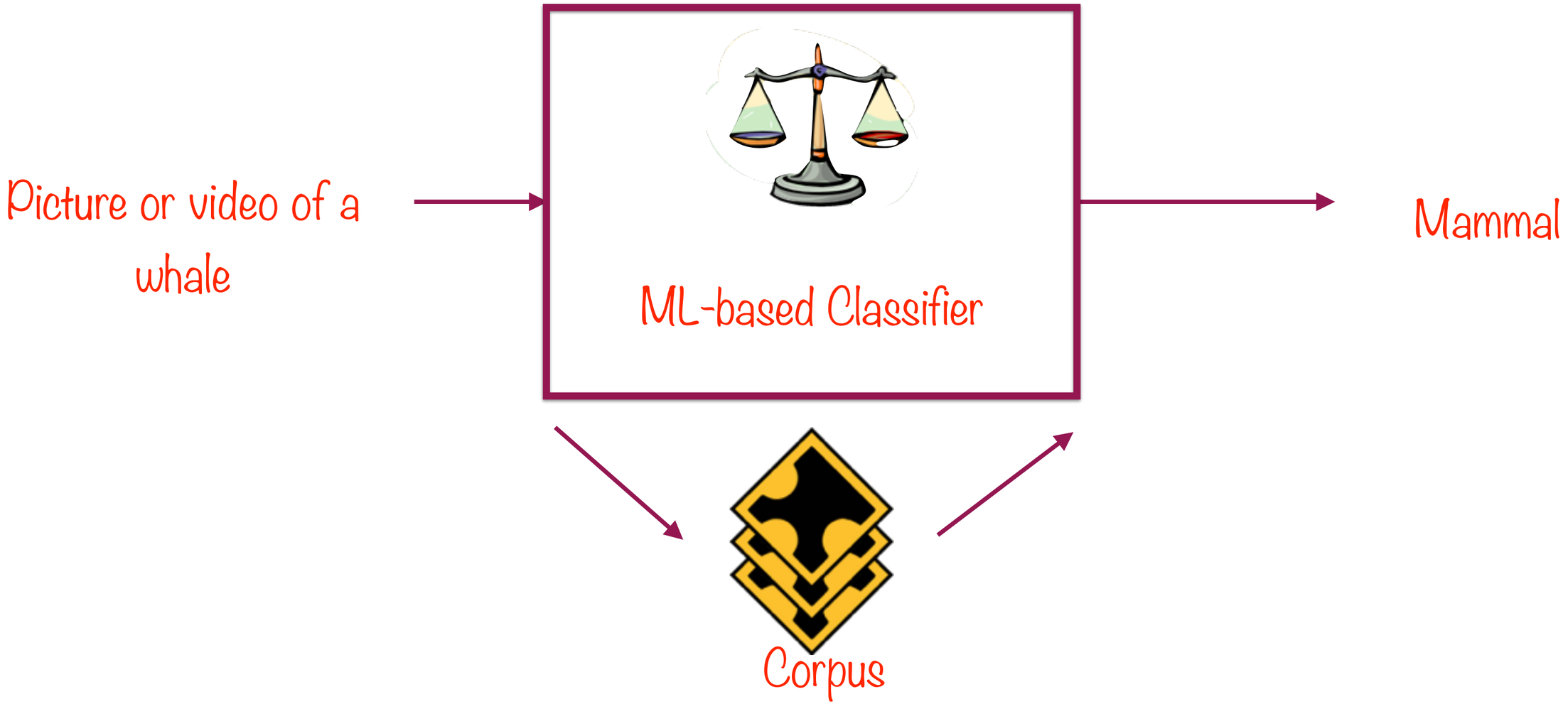
Classification Algorithm

ML-based Classifier

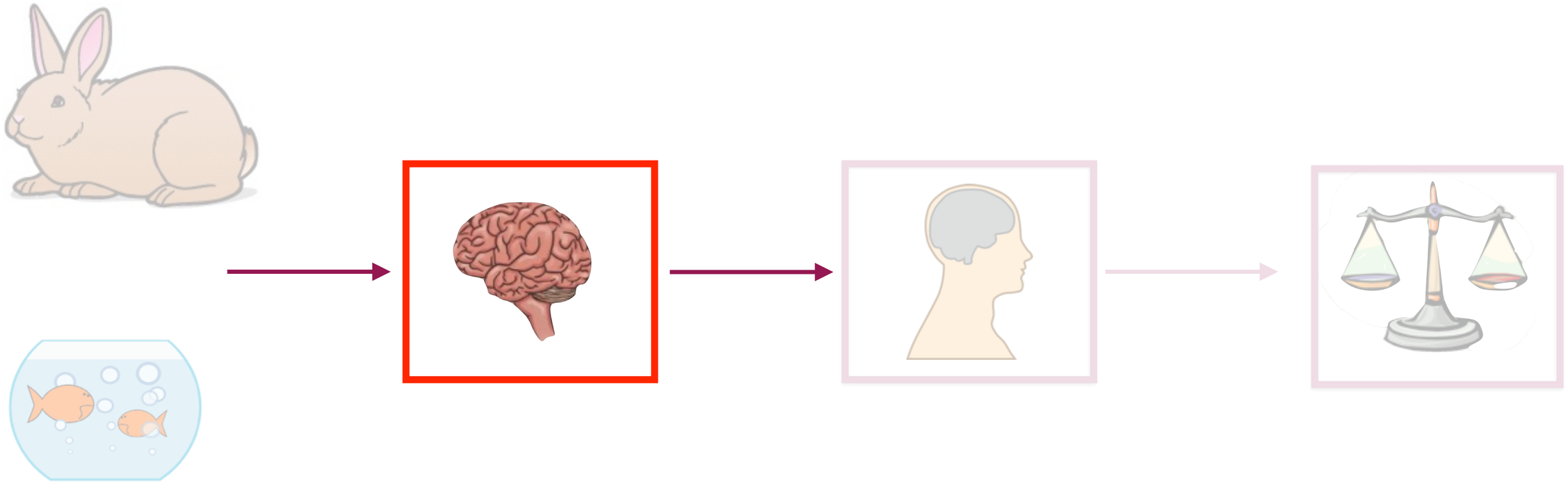
"Traditional" ML-based Binary Classifier



"Representation" ML-based Binary Classifier



"Representation" ML-based Binary Classifier



Corpus

Feature Selection Algorithm

Classification Algorithm

ML-based Classifier

"Deep Learning" systems are one type of representation systems

Deep Learning and Neural Networks

Deep Learning and Neural Networks

Deep Learning

Algorithms that learn what features matter

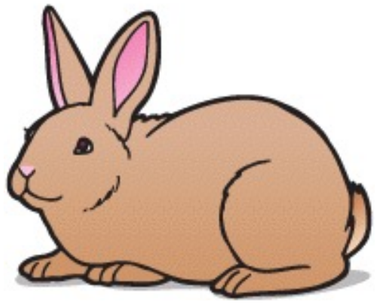
Neural Networks

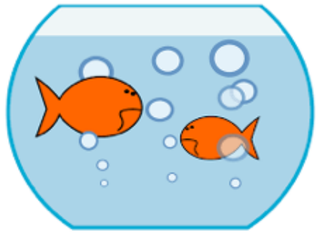
The most common class of deep learning algorithms

Neurons

Simple building blocks that actually "learn"

"Deep Learning"-based Binary Classifier





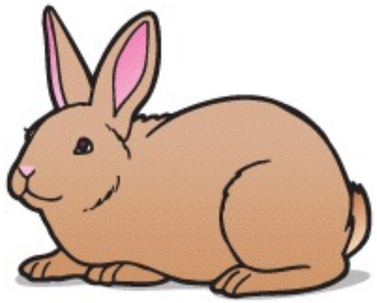
Pixels → Edges → Corners → Object Parts

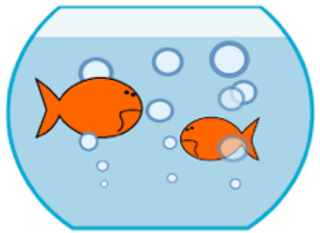
Corpus of Images

Feature Selection & Classification Algorithm

ML-based Classifier

"Deep Learning"-based Binary Classifier





Corpus of Images

Pixels

Edges

Corners

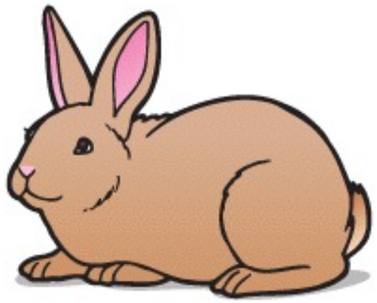
Object Parts

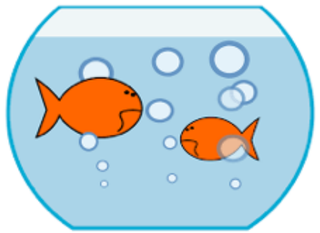
"Visible layer"

"Visible layer"

ML-based Classifier

"Deep Learning"-based Binary Classifier





Pixels

Edges

Corners

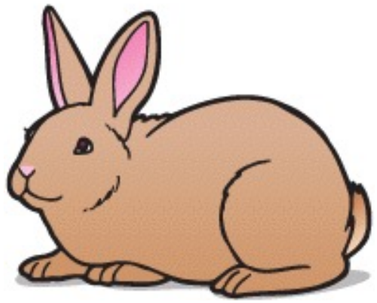
Object Parts

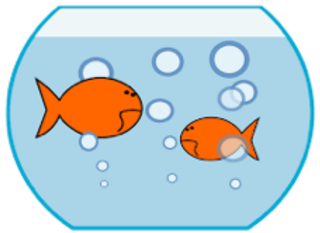
Corpus of Images

"Hidden Layers"

ML-based Classifier

Neural Networks Introduced





Layer 1

Layer 2

...

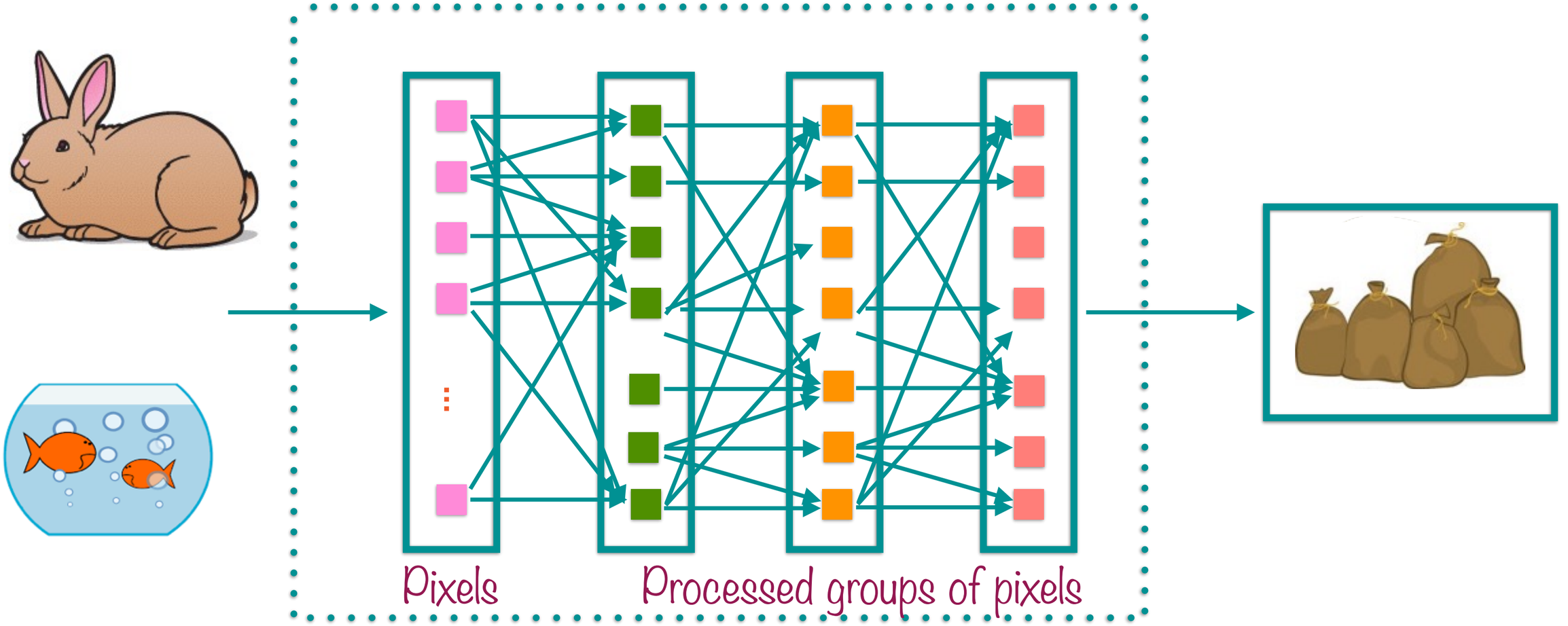
Layer N

Corpus of Images

Layers in a neural network

ML-based Classifier

Neural Networks Introduced



Corpus of Images

Each layer consists of individual interconnected neurons

ML-based Classifier

Neural networks help find unknown patterns in massive data sets

TensorFlow for Machine Learning

TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs.

https://www.tensorflow.org/

TensorFlow

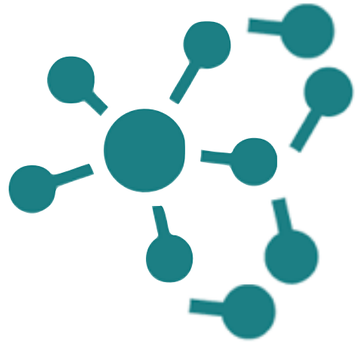
TensorFlow™ is an open source software library for numerical computation using data flow graphs.

https://www.tensorflow.org/

TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs.

https://www.tensorflow.org/

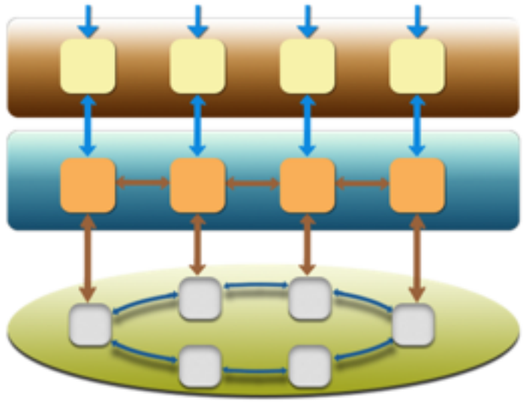


TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs.

https://www.tensorflow.org/

Advantages of TensorFlow



Distributed

Runs on a cluster or machines or multiple CPUs/GPUs on the same machine

Suite of software

TensorFlow, TensorBoard, TensorFlow Serving

TensorFlow

Uses

Strengths

Challenges

Uses

Research and development of new ML algorithms

Taking models from training to production

Large scale distributed models

Models for mobile and embedded systems

Strengths

Easy to use, stable Python API

Runs on large as well small systems

Efficient and performant

Great support from Google

Additional tools like TensorBoard and TensorFlow serving

Challenges

Distributed support still has a ways to go

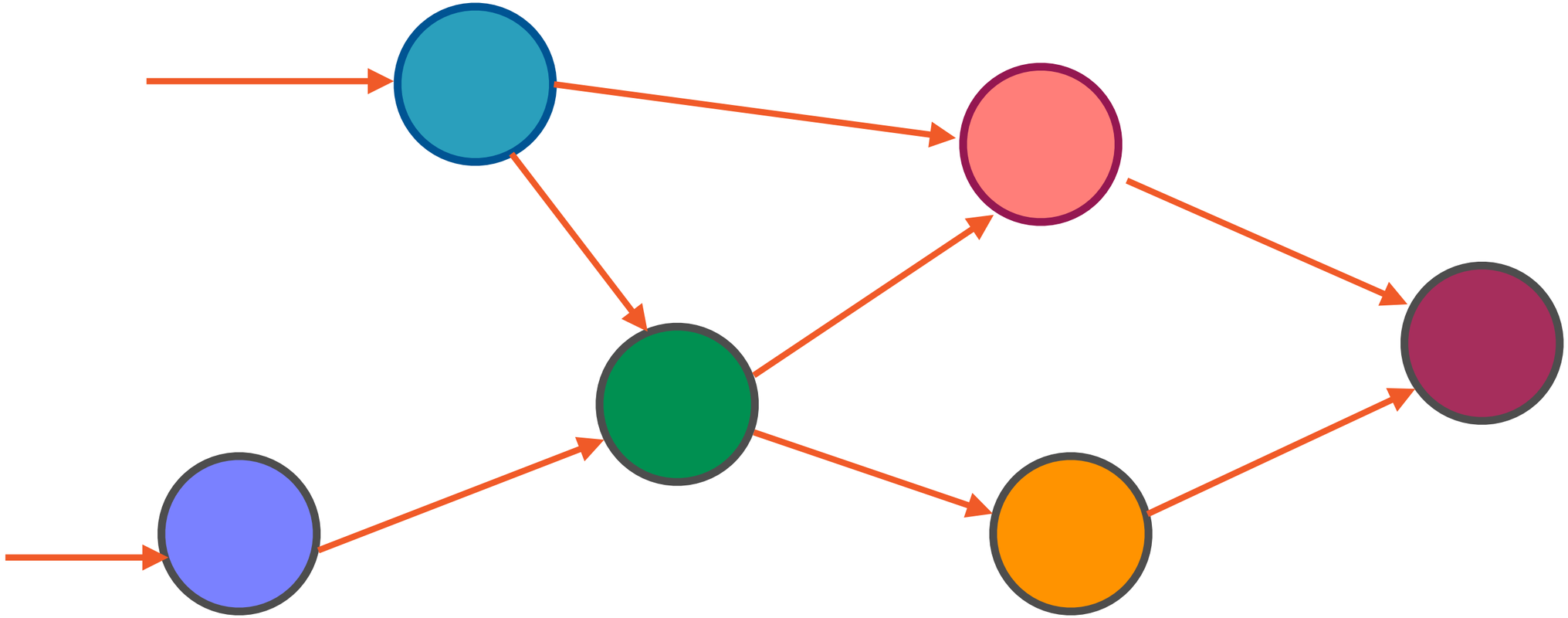
Libraries still being developed

Writing custom code is not straightforward

TensorFlow is on its way to becoming the default library for machine learning

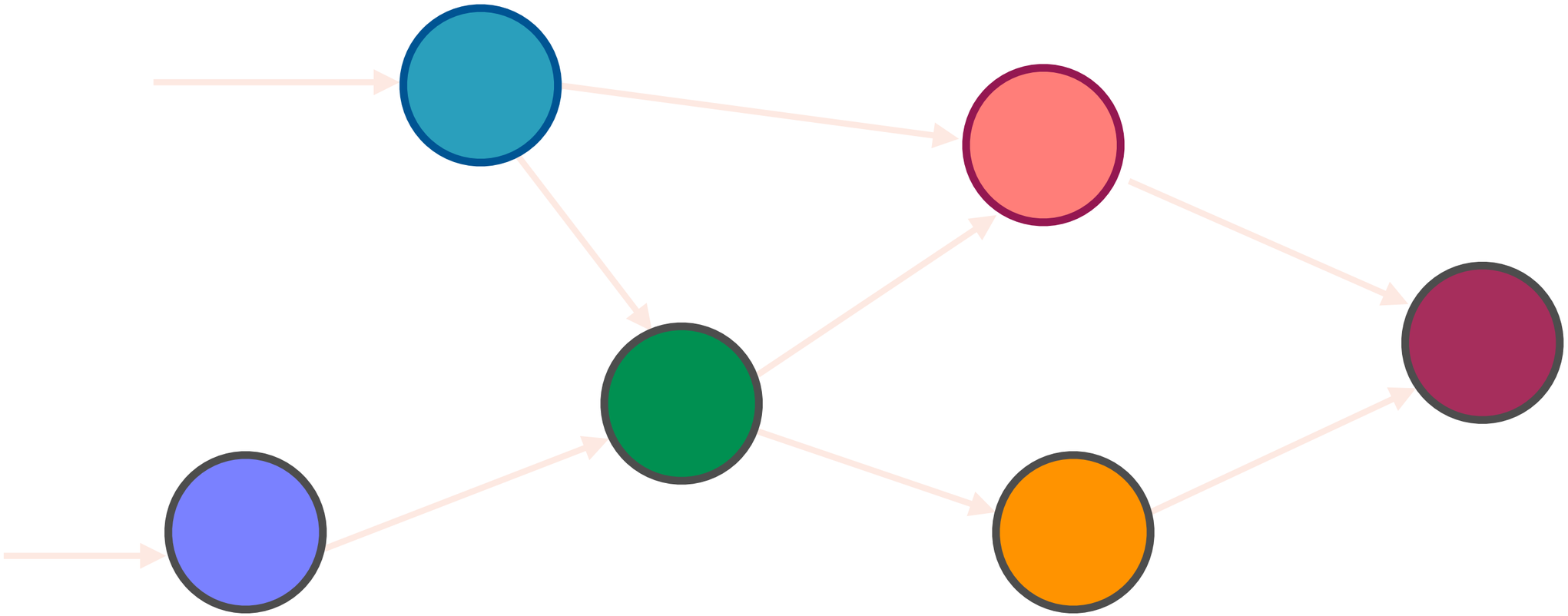
The TensorFlow World

Everything Is a Graph



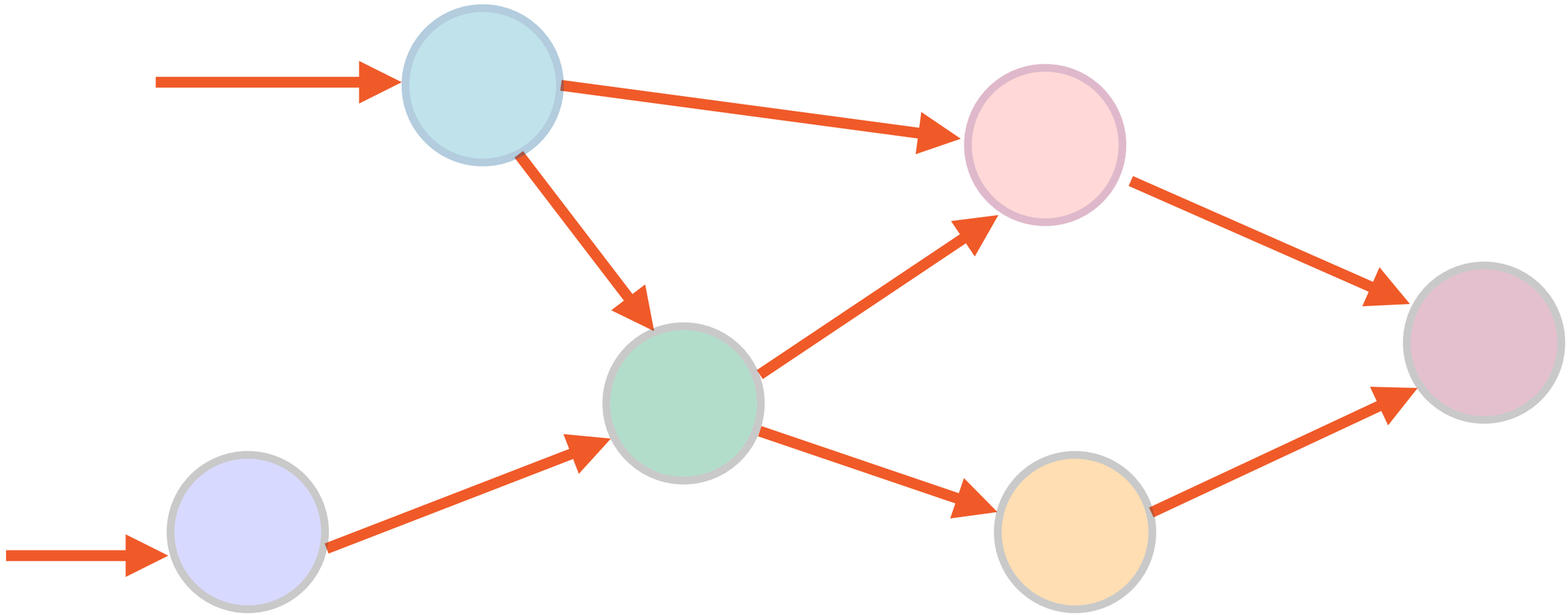
A network

Everything Is a Graph



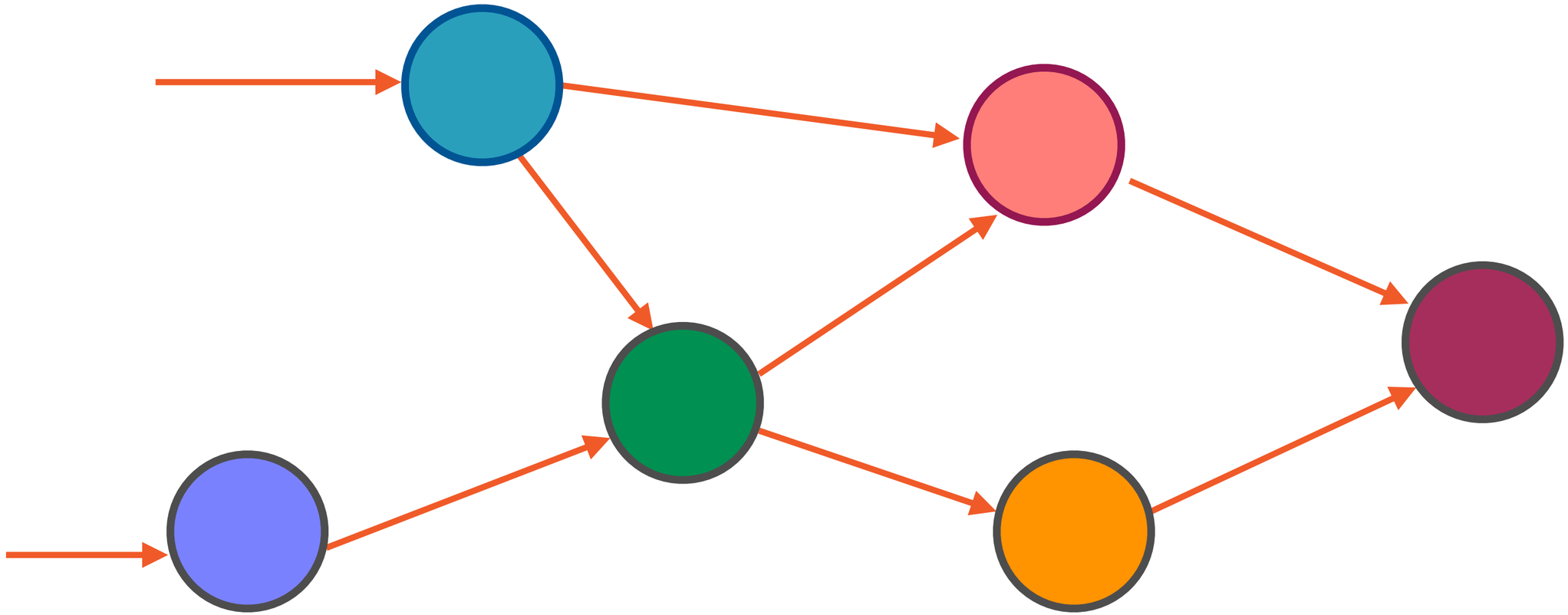
Computations

Everything Is a Graph



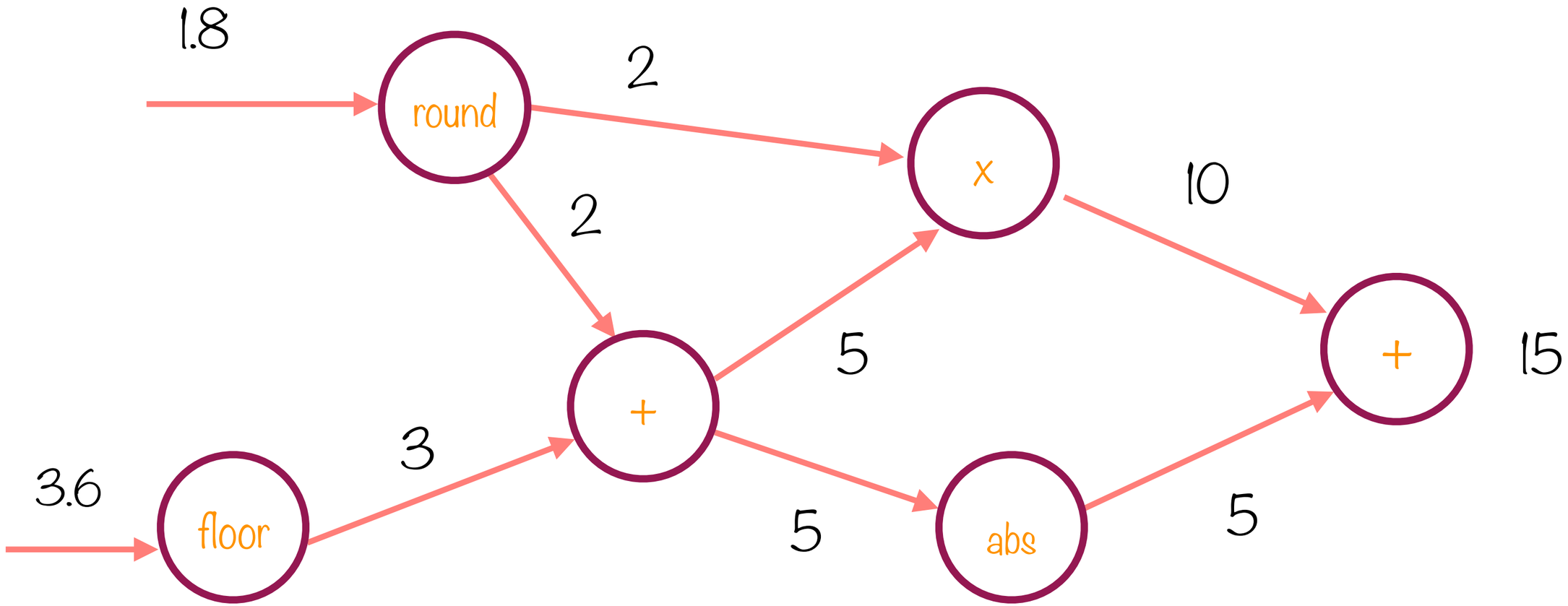
Tensors Data

Tensors Flow Through the Graph



...and get transformed along the way

Tensors Flow Through the Graph



TensorFlow

Demo

Download and install TensorFlow on your local machine

Validate that the TensorFlow libraries work and can be referenced

Summary

Learnt the basics of machine learning, deep learning and neural networks

Understood the strengths and challenges of using TensorFlow for ML

Understood the modeling of problem as a computational graph

Got TensorFlow up and running on your local machine

Introducing Computation Graphs

Overview

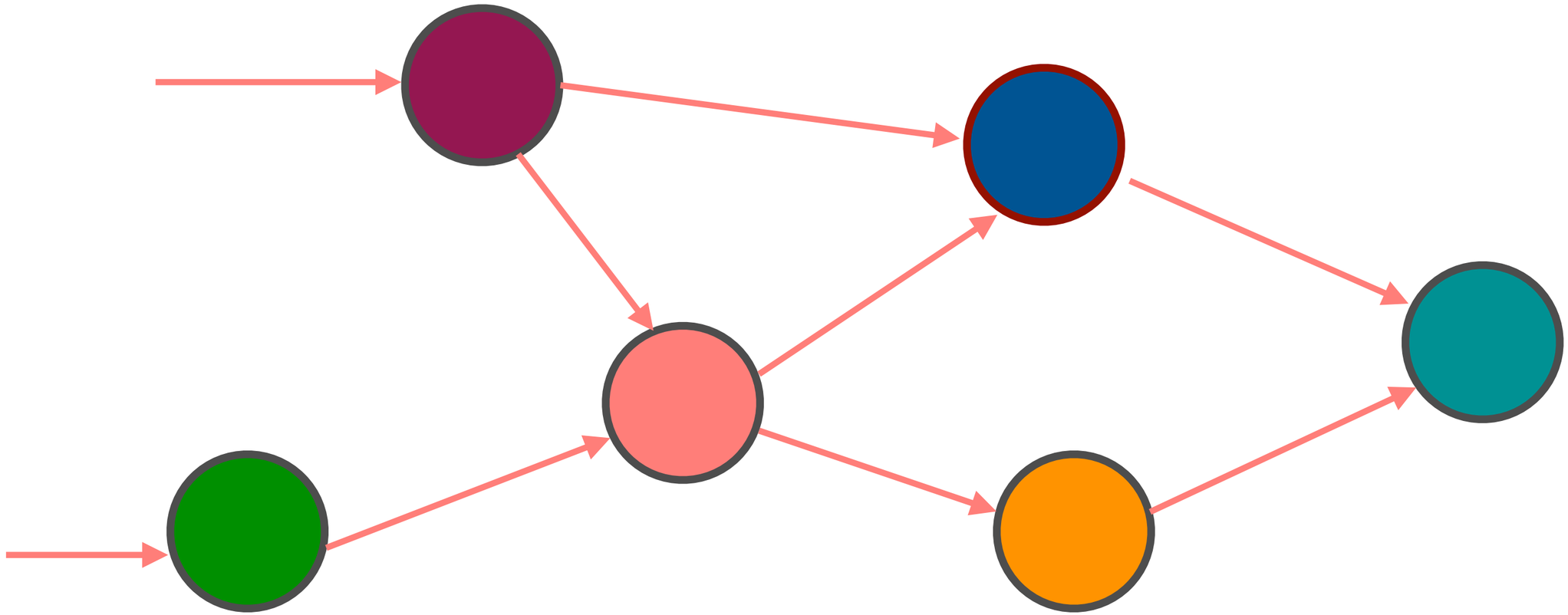
Model nodes, edges and dependencies in a computation graph

Understand the basic parts of a program in TensorFlow

Run TensorFlow programs and visualize results using TensorBoard

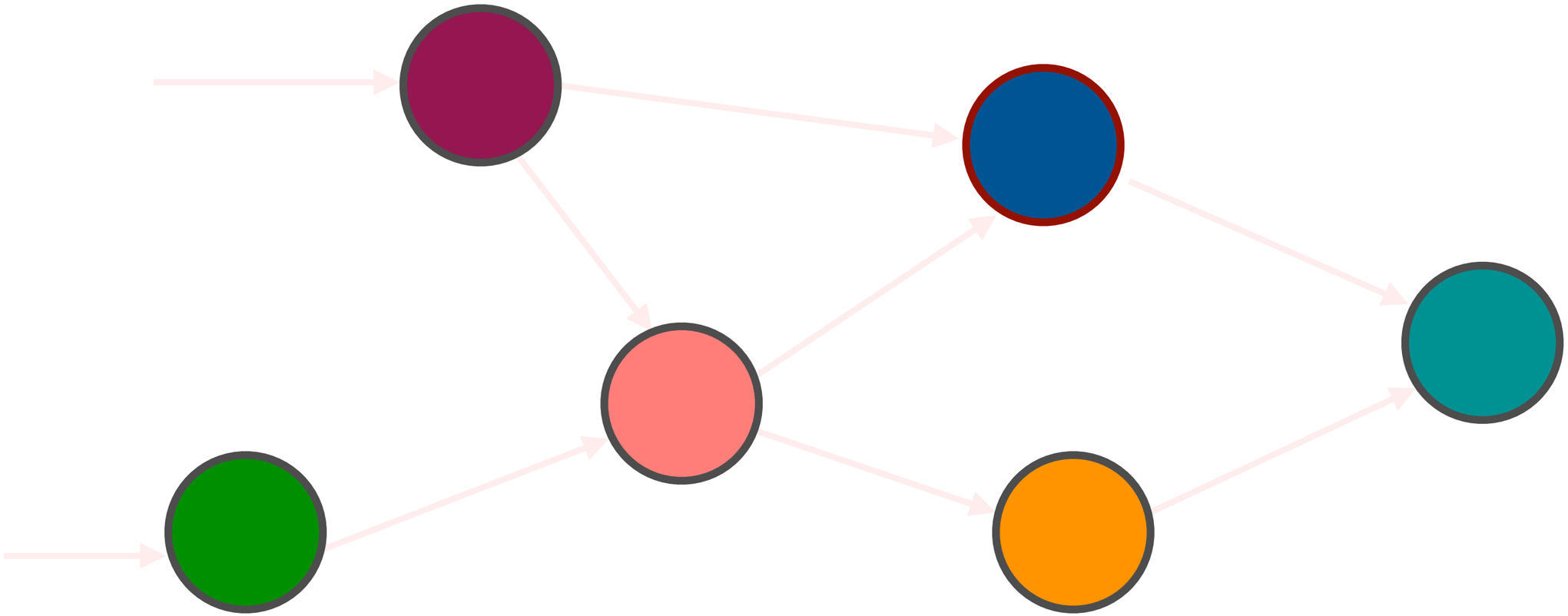
The TensorFlow World

Everything Is a Graph



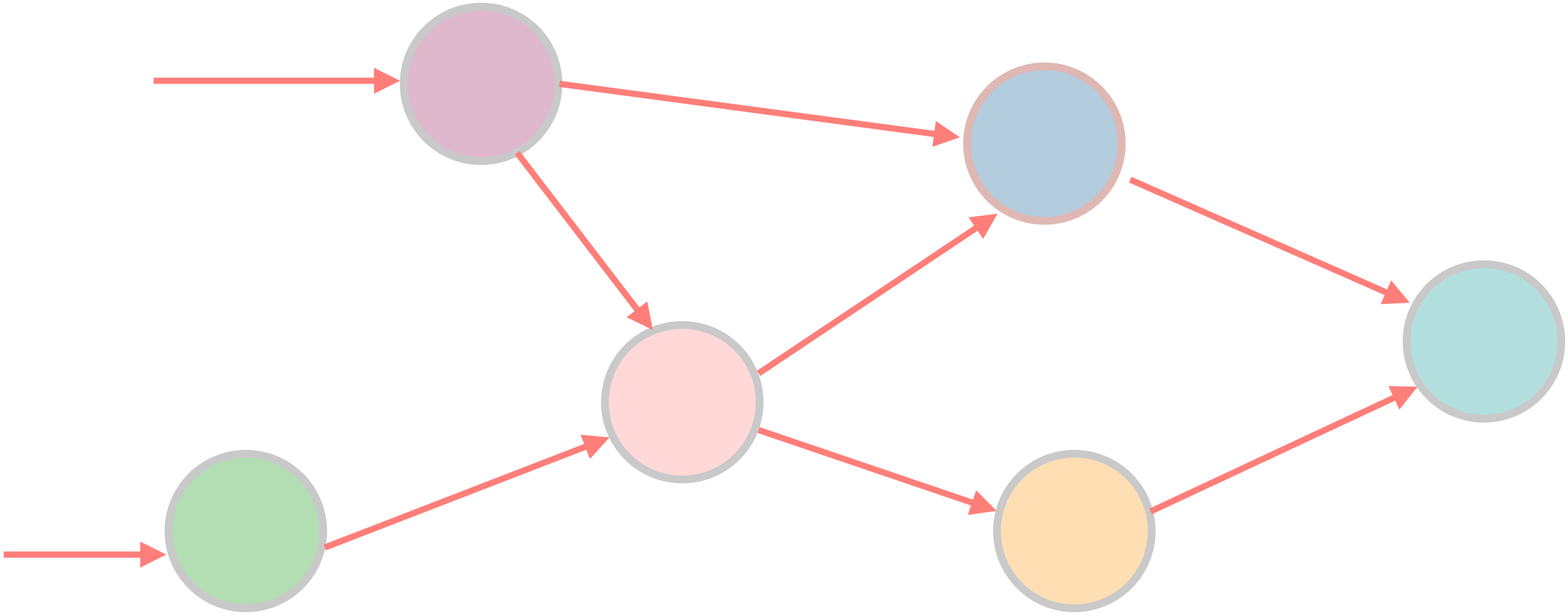
A network

Everything Is a Graph



Computations Operations

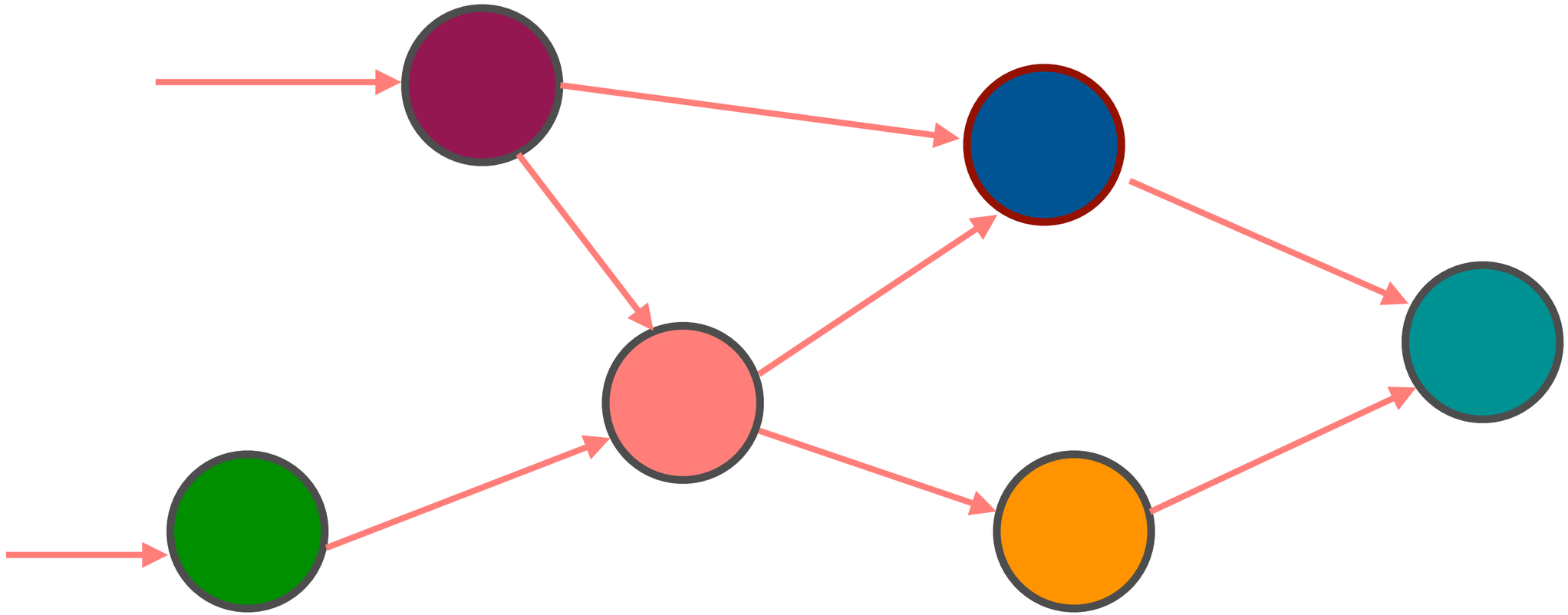
Everything Is a Graph



Tensors

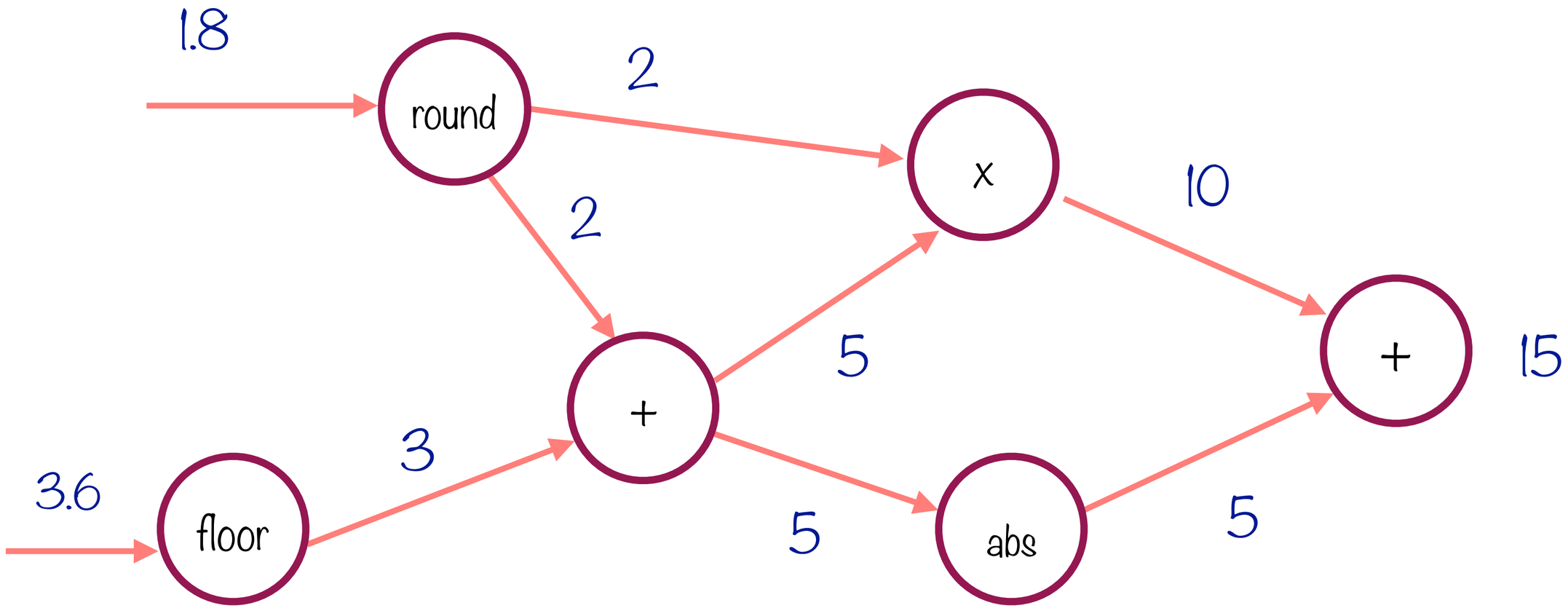
Data

Tensors Flow Through the Graph



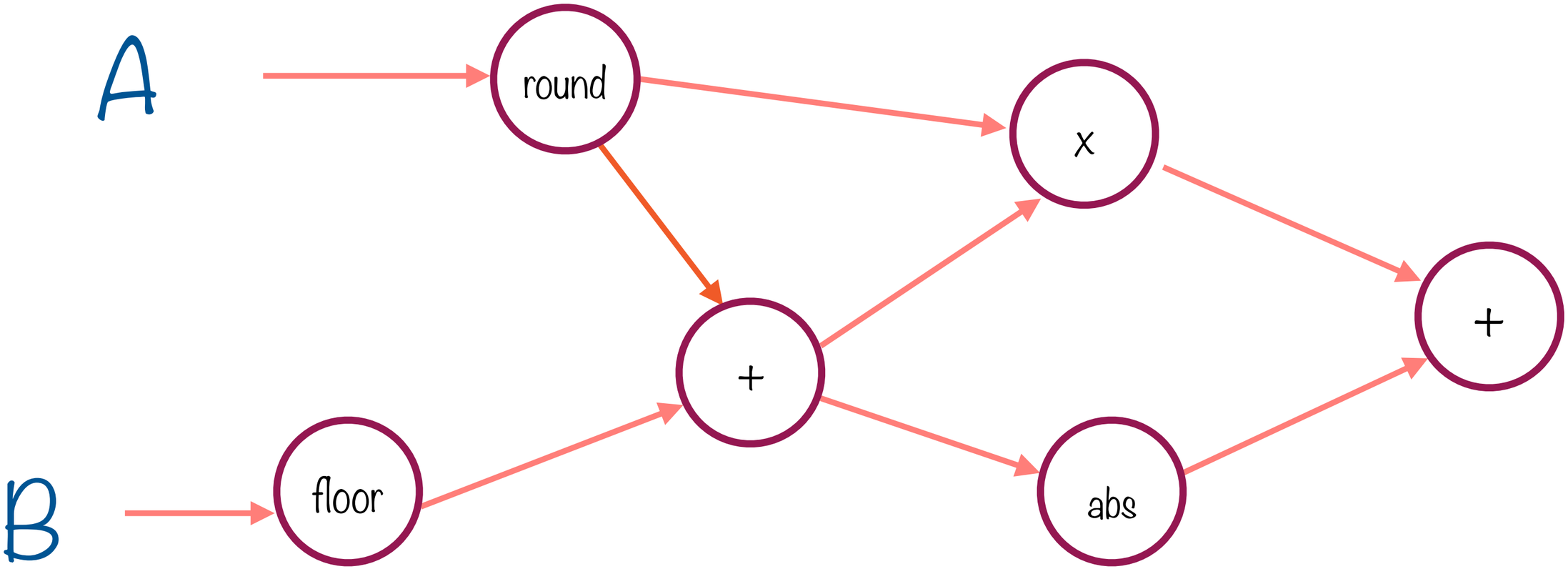
...and gets transformed along the way

Tensors Flow Through the Graph



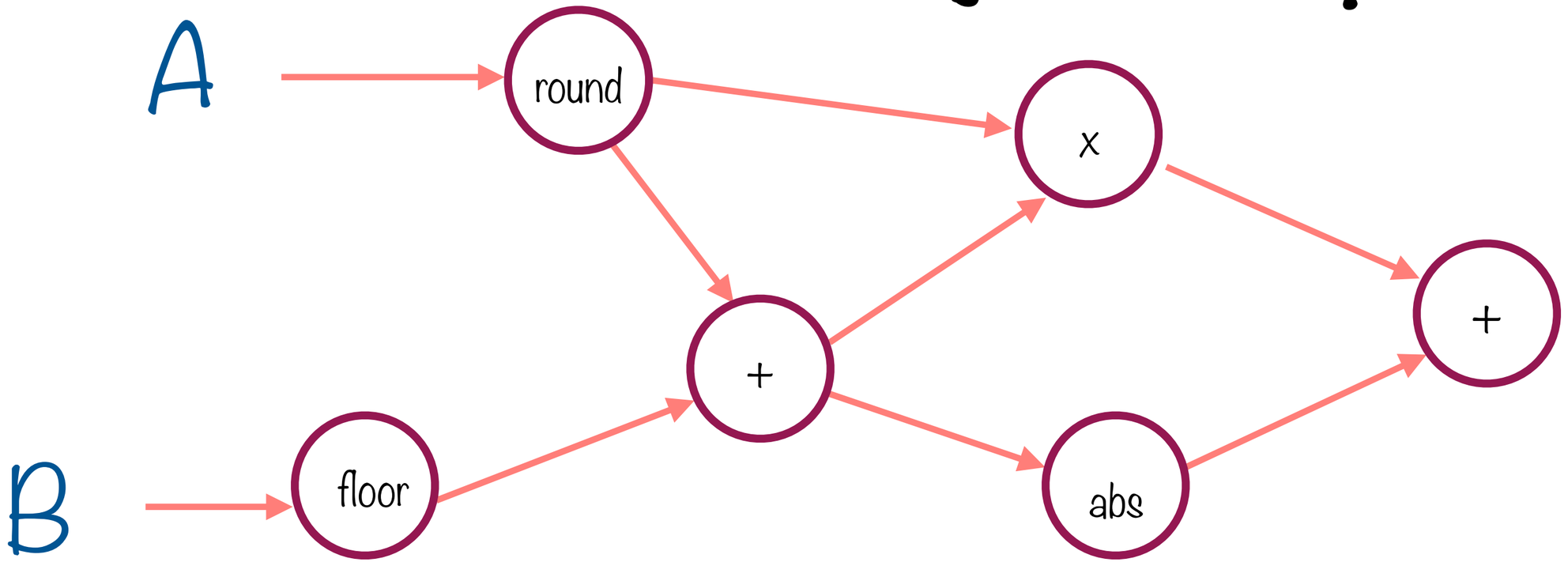
TensorFlow

Tensors Flow Through the Graph



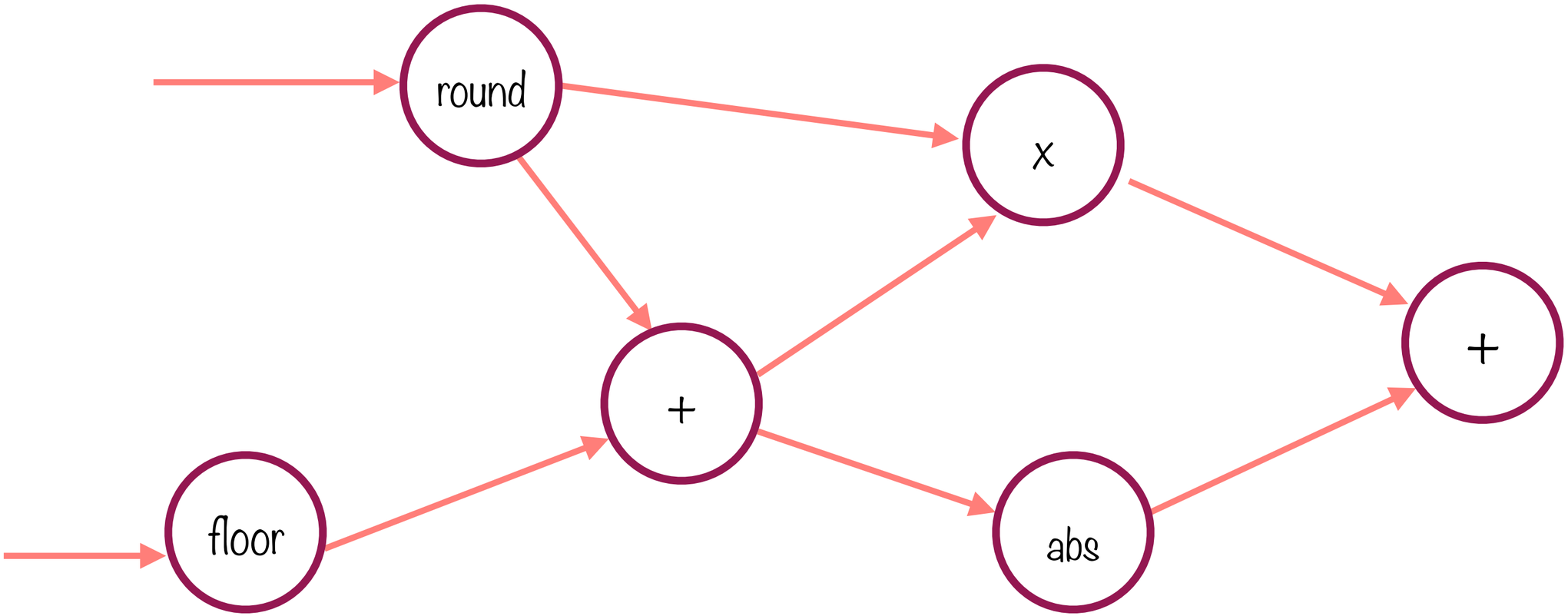
TensorFlow

Tensors Flow Through the Graph

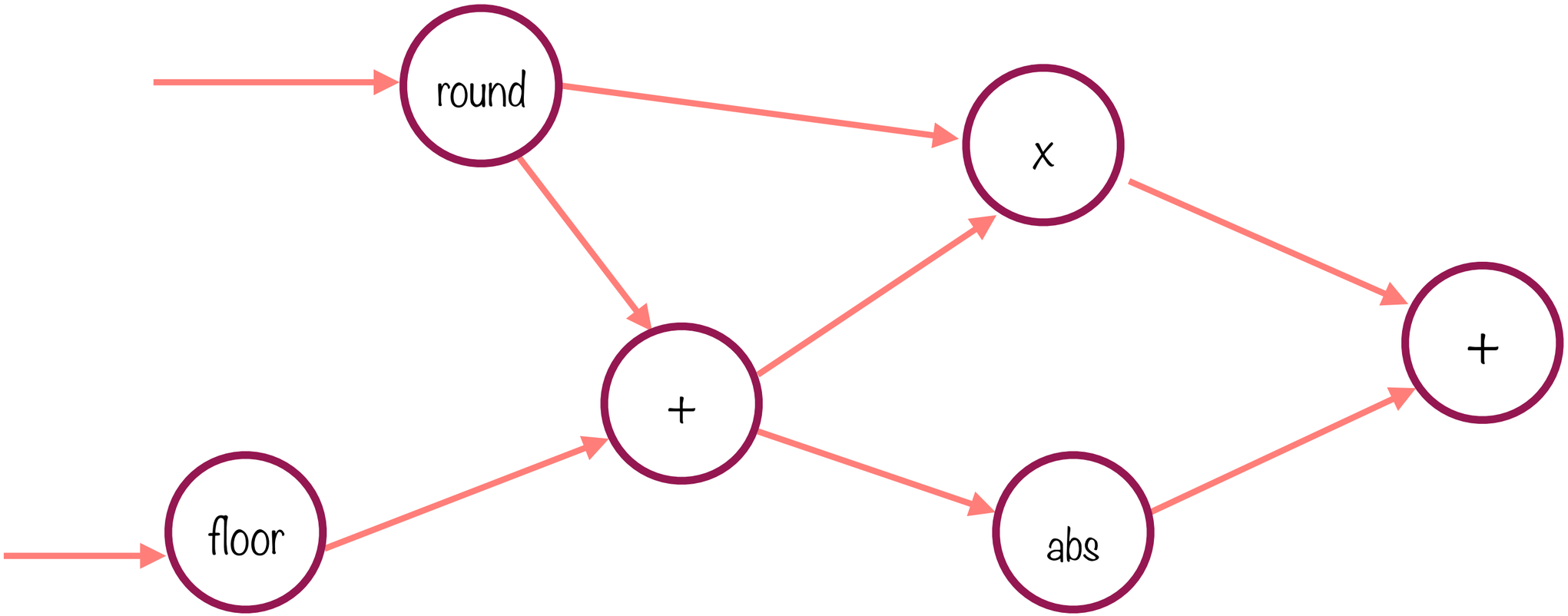


$$Y = (\text{round}(A) + \text{floor}(B)) * \text{round}(A) + \text{abs}(\text{round}(A) + \text{floor}(B))$$

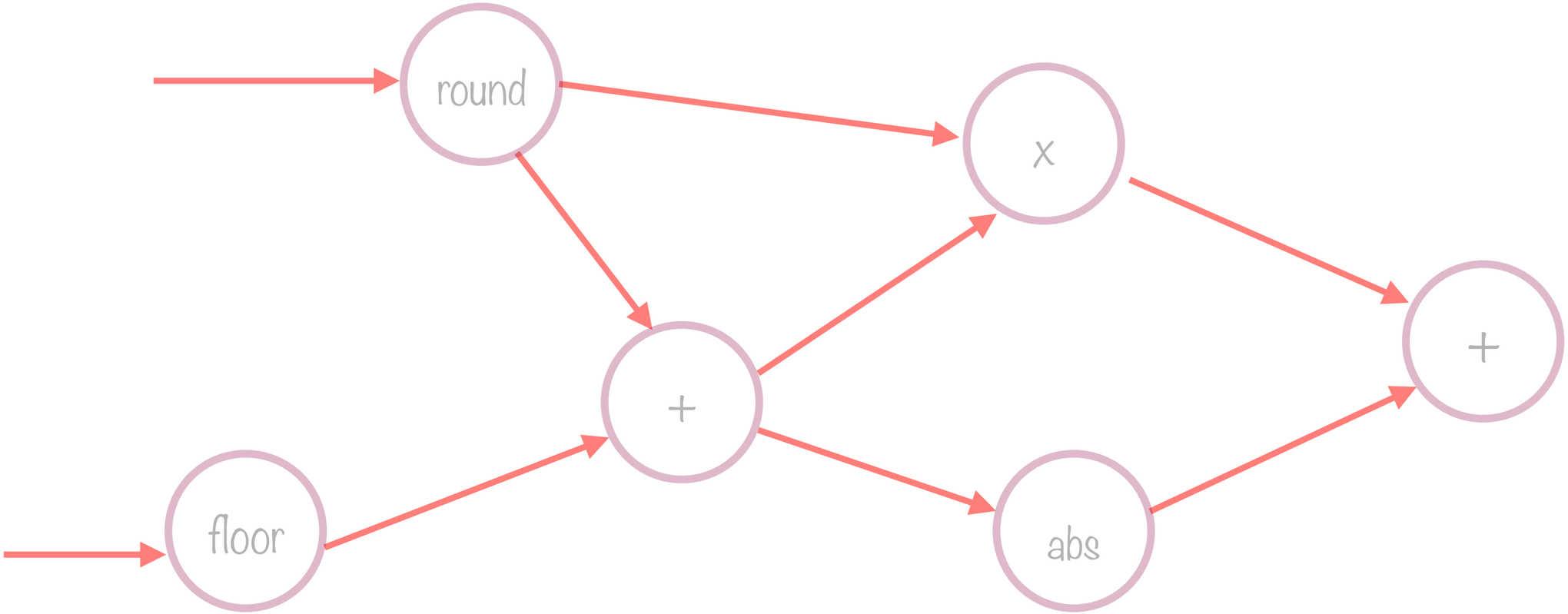
Directed-Acyclic Graph



Directed-Acyclic Graph

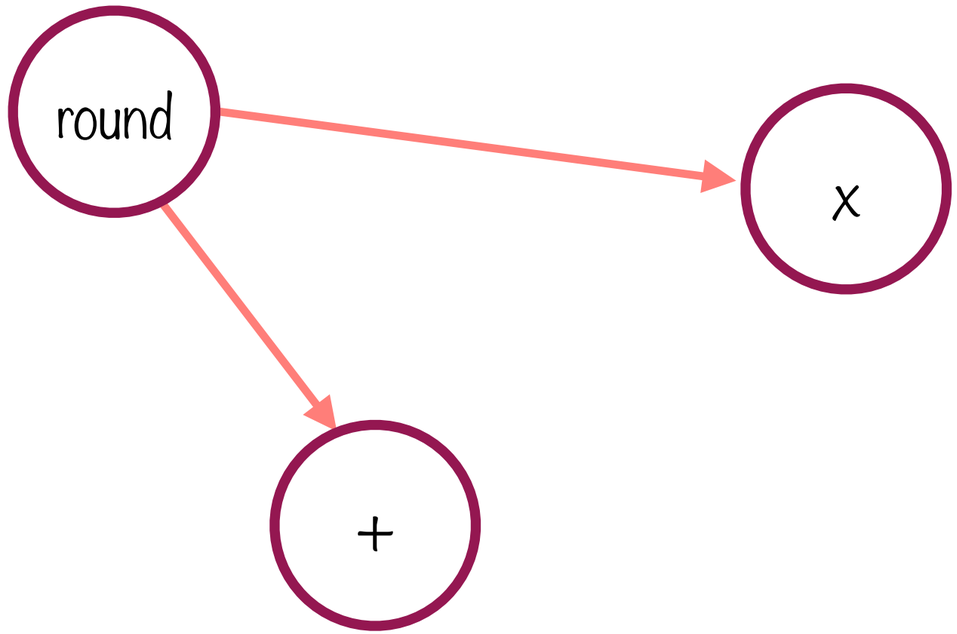


Directed-Acyclic Graph



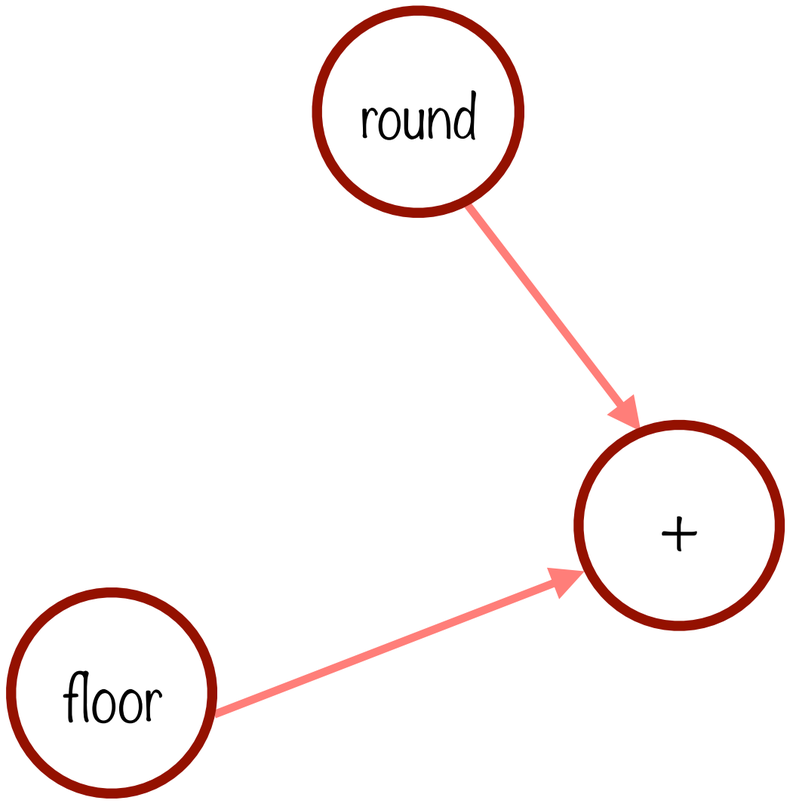
Edges point forward towards a result i.e. directed

Dependencies



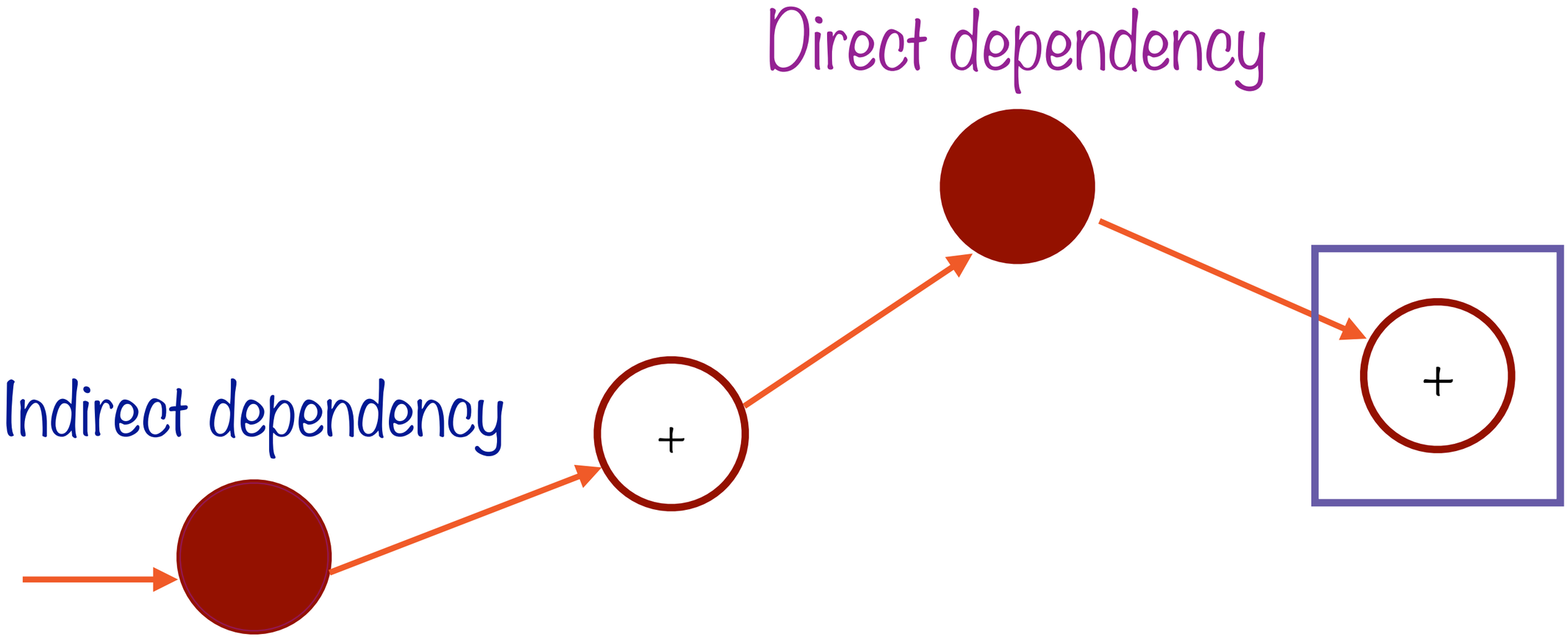
One node can send its output to multiple nodes

Dependencies

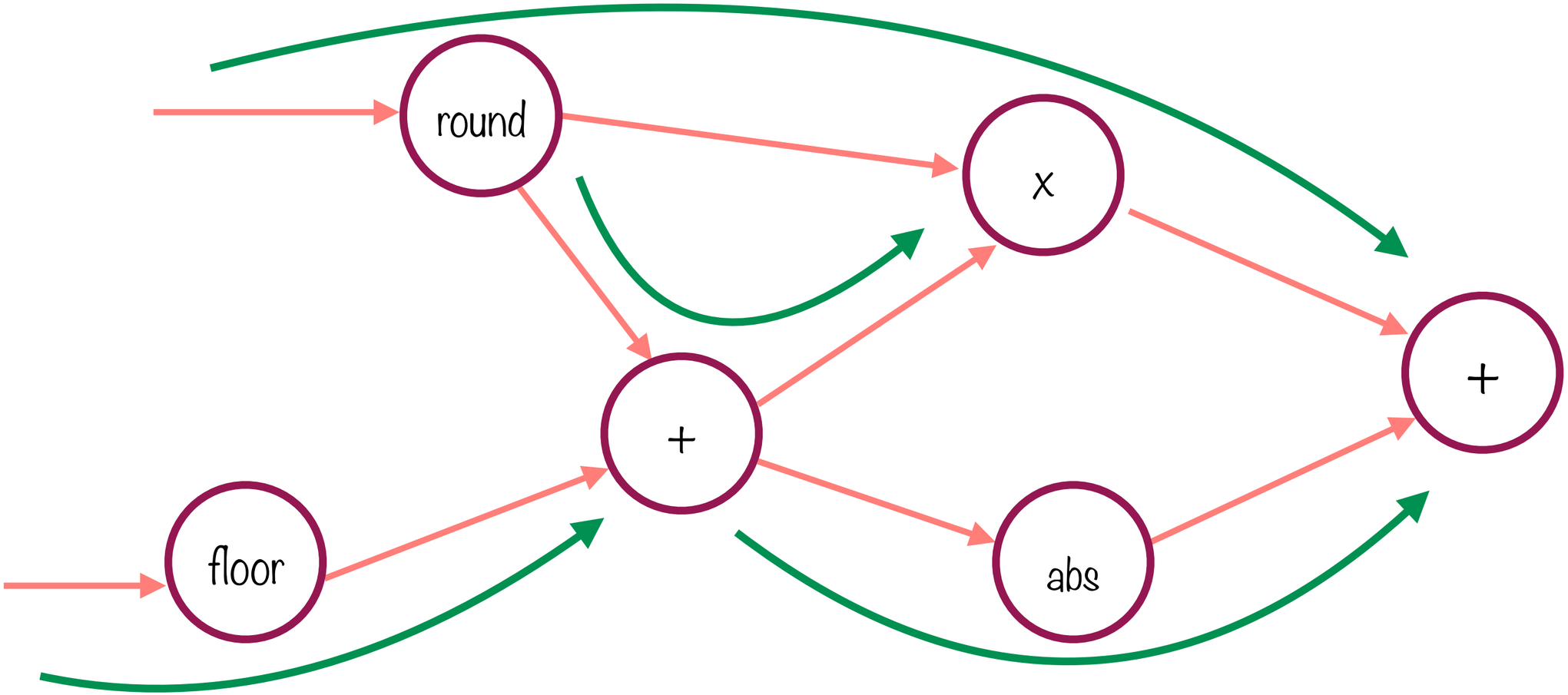


Or receive inputs from multiple nodes

Dependencies

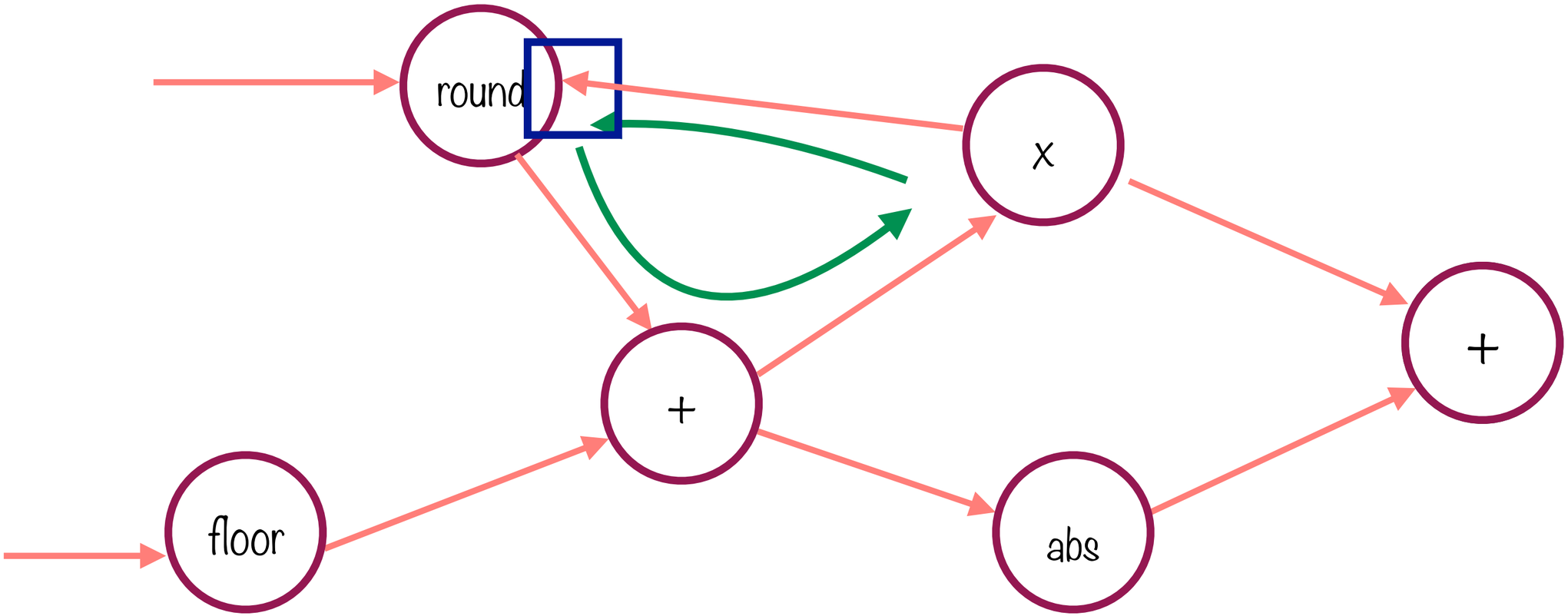


Directed-Acyclic Graph



There are no cycles in the graph i.e. acyclic

Directed-Acyclic Graph

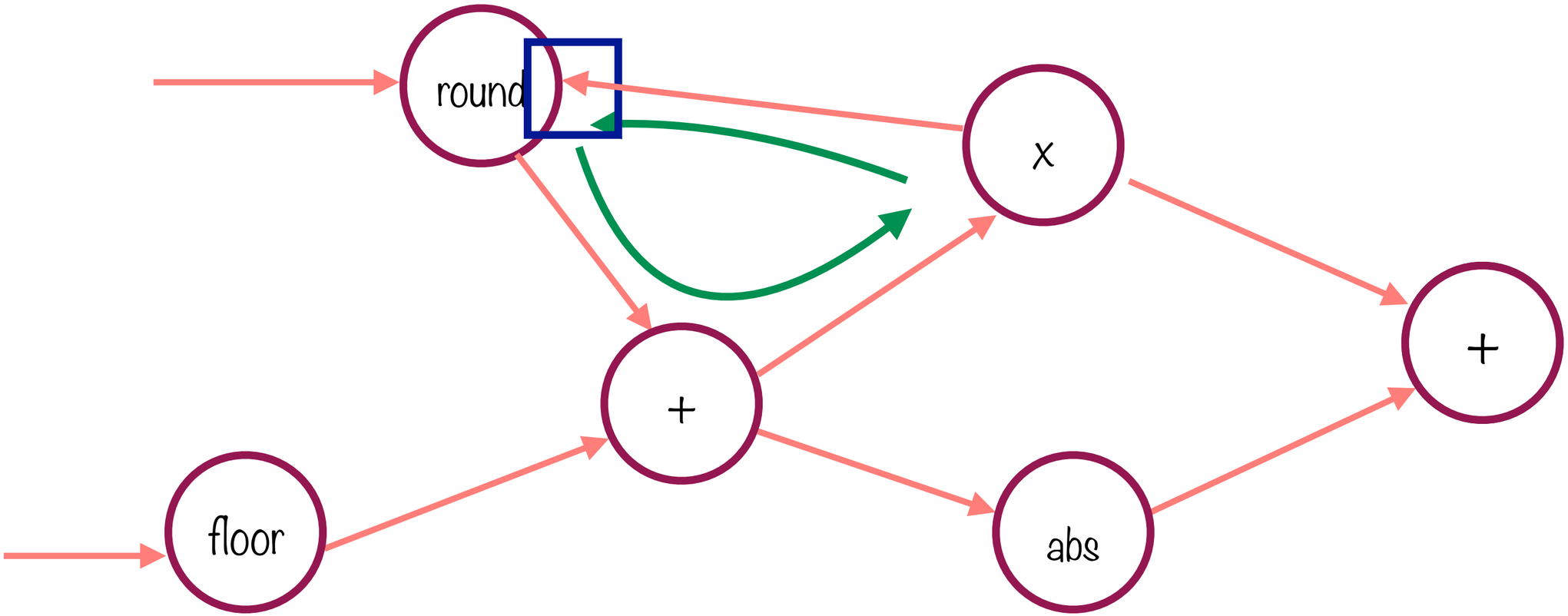


A graph with cycles will never finish computation

Problems in TensorFlow are represented as a directed-acyclic graph

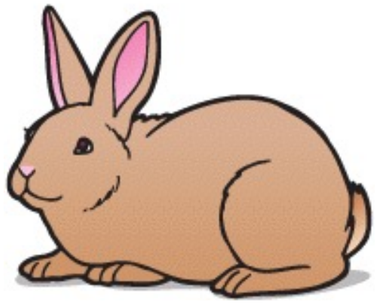
Cyclical Dependencies in Machine Learning

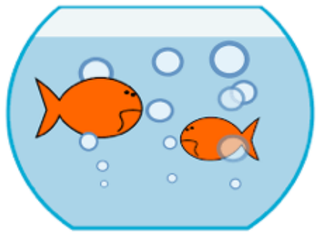
Directed-Acyclic Graph



A graph with cycles will never finish computation

The Process of Machine Learning



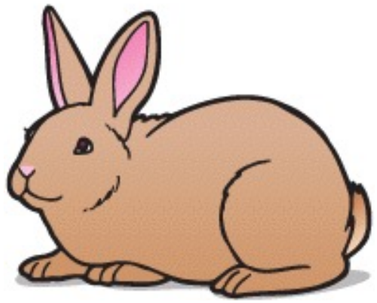


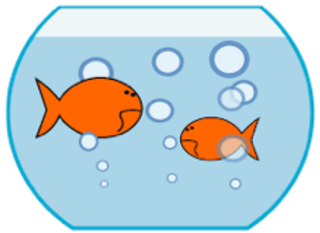
Corpus of data

Feature Selection & ML

Result

The Process of Machine Learning

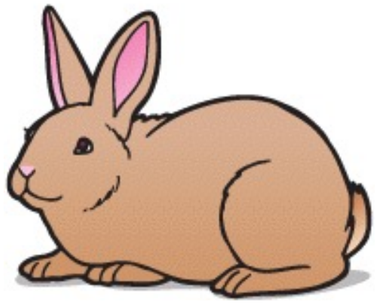


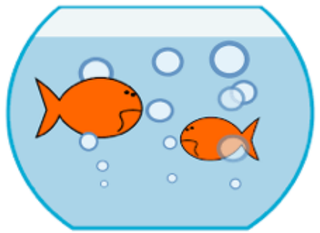


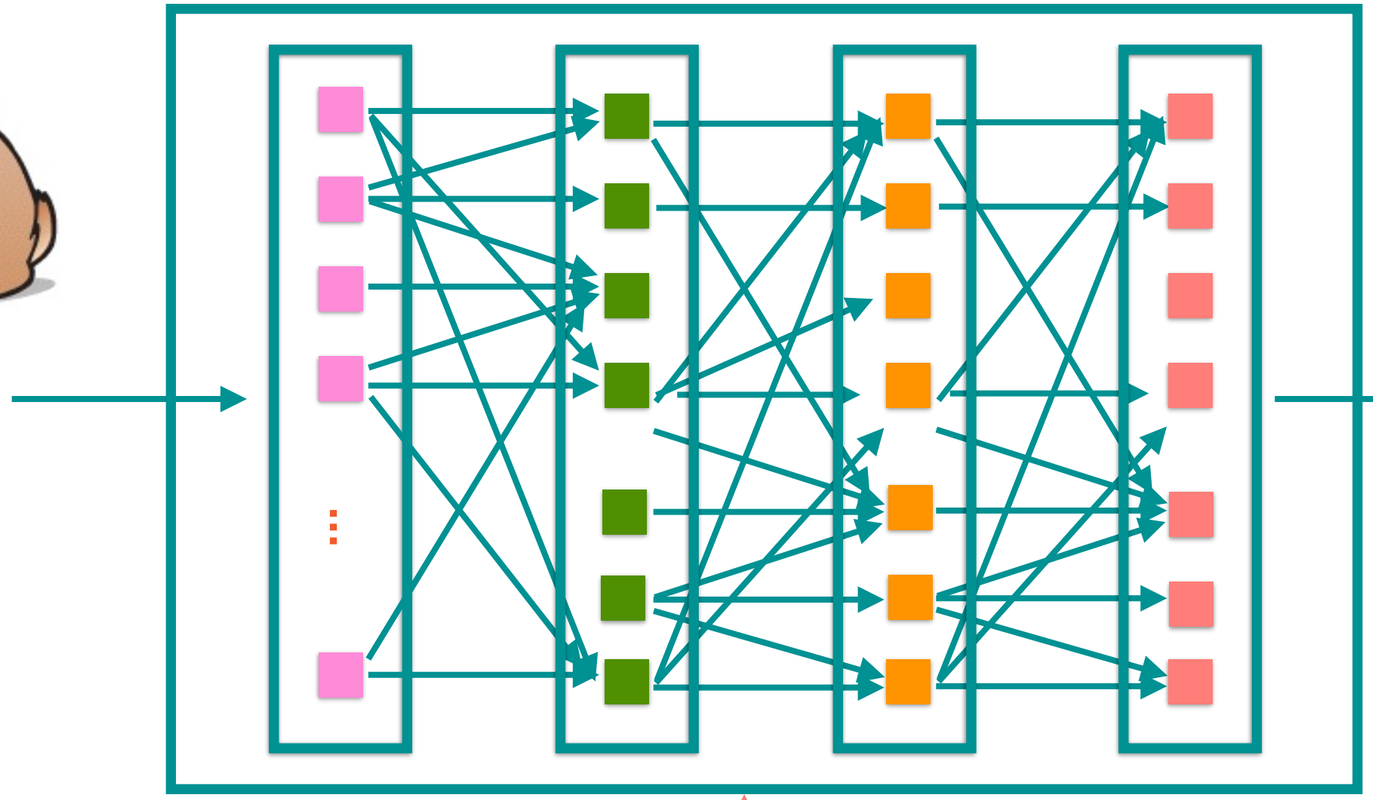
Calculate loss

Feedback

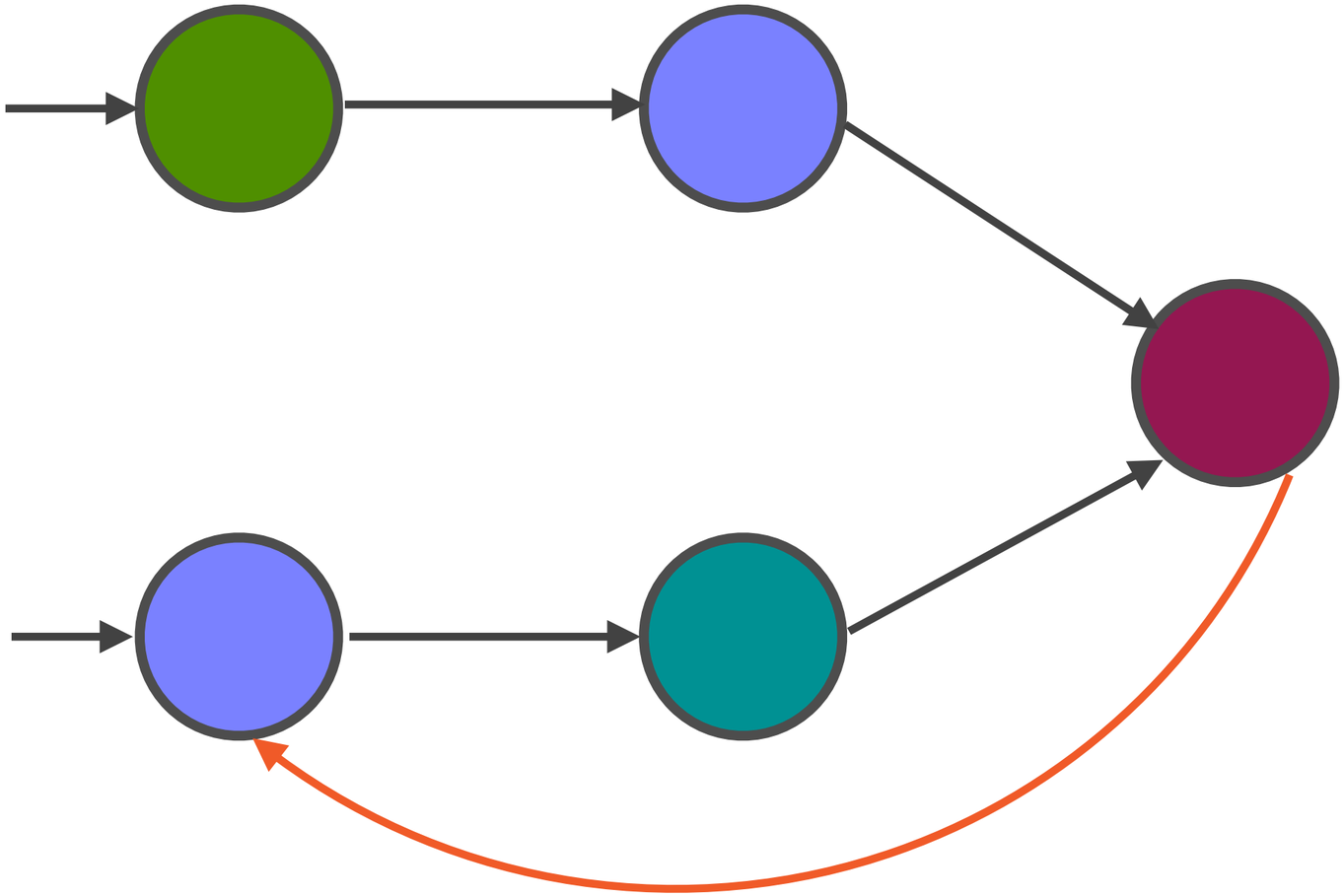
The Process of Machine Learning



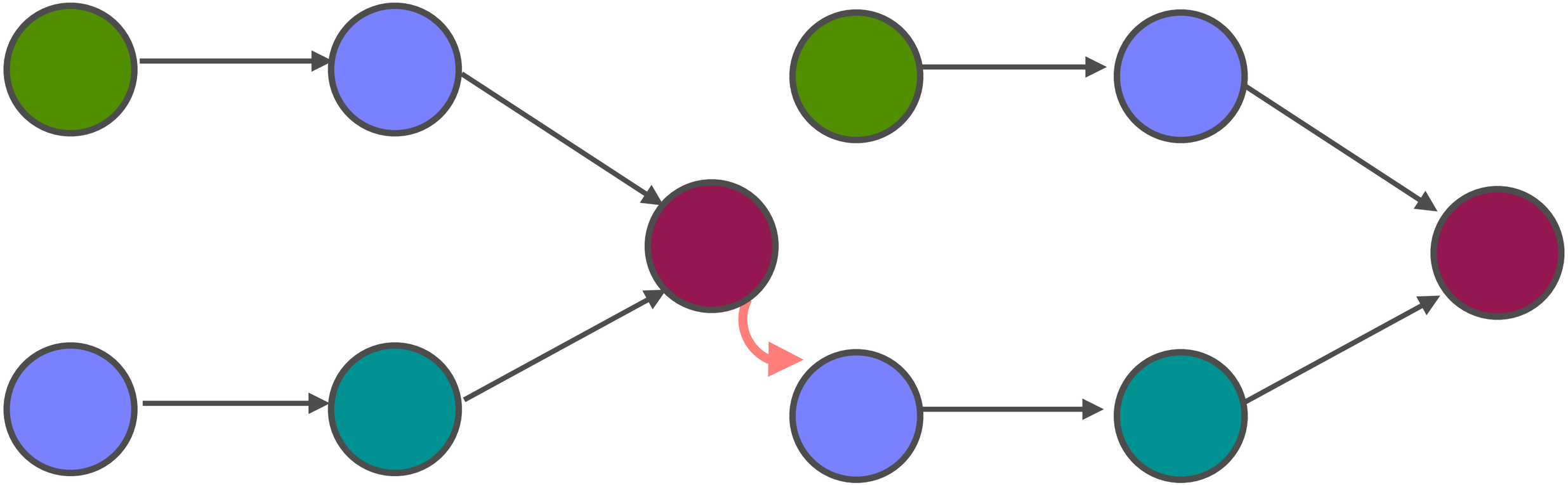




Modelling Cyclical Dependencies

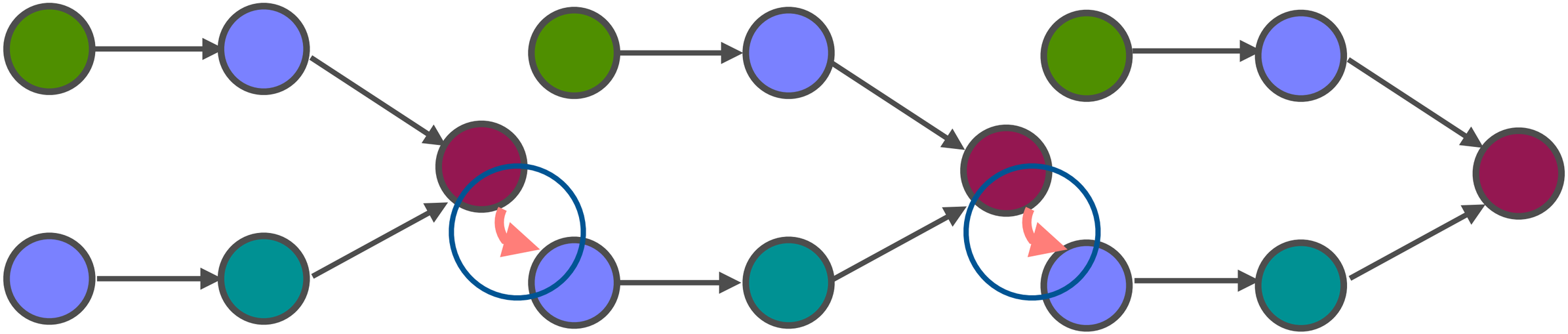


Modelling Cyclical Dependencies



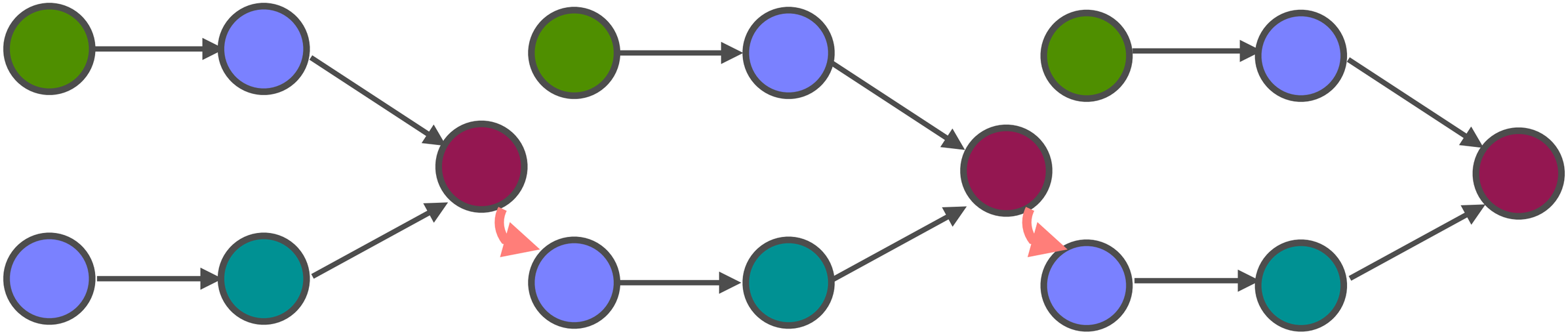
Unrolling the graph

Modelling Cyclical Dependencies



Unrolling the graph

Modelling Cyclical Dependencies



How much you unroll depends on the number of iterations you want to run

Unroll graphs to model cyclic dependencies

Building and Running Graphs

2 Steps in a TensorFlow Program

Building a Graph

Specify the operations and the data

Running a Graph

Execute the graph to get the final result

Demo

Building and running a graph in TensorFlow

Exploring the graph using TensorBoard

2 Steps in a TensorFlow Program

Building a Graph

Specify the operations and the data

Running a Graph

Execute the graph to get the final result

2 Steps in a TensorFlow Program

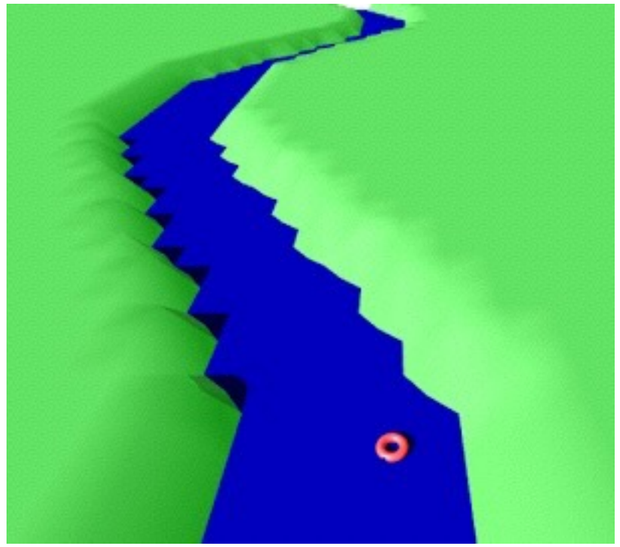
Building a Graph

Specify the operations and the data

Running a Graph

Execute the graph to get the final result

Visualizing a Graph

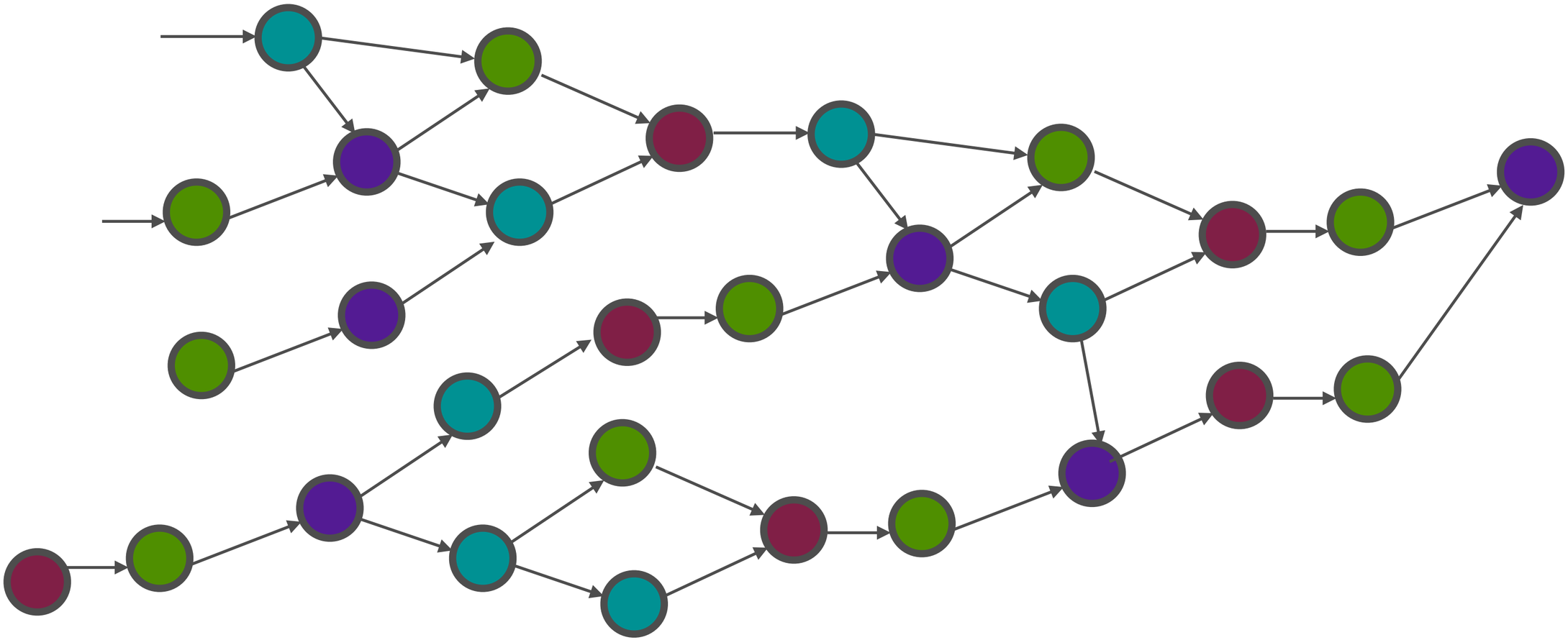


TensorBoard

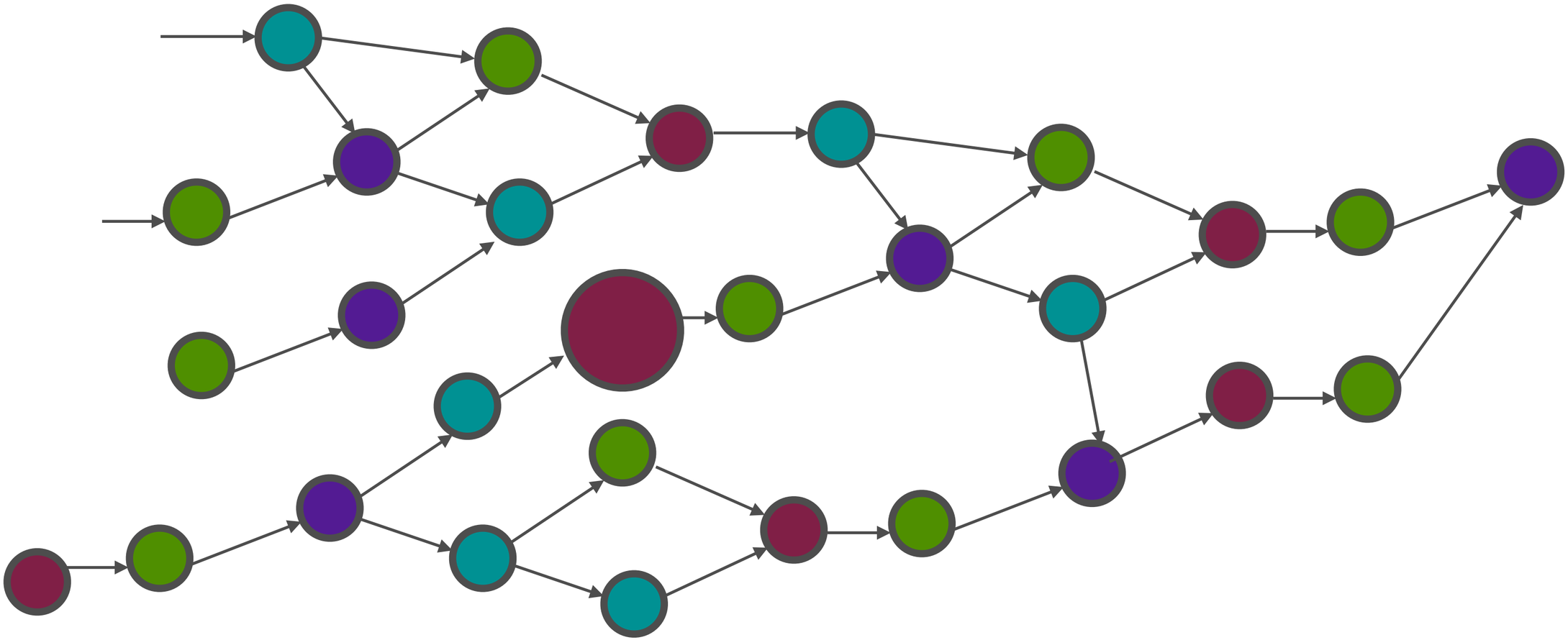
Visualize how data flows and what computations operate on it

Modeling Computations as Graphs

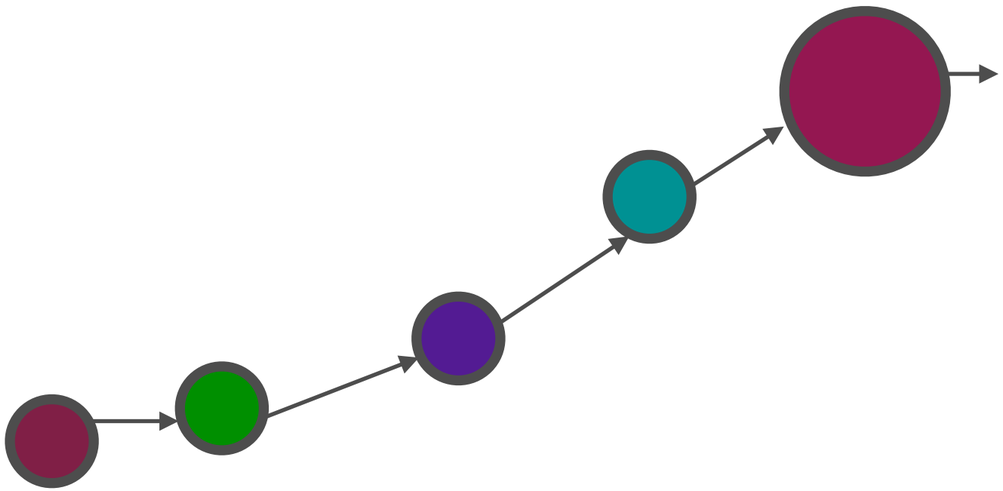
Computation Graphs



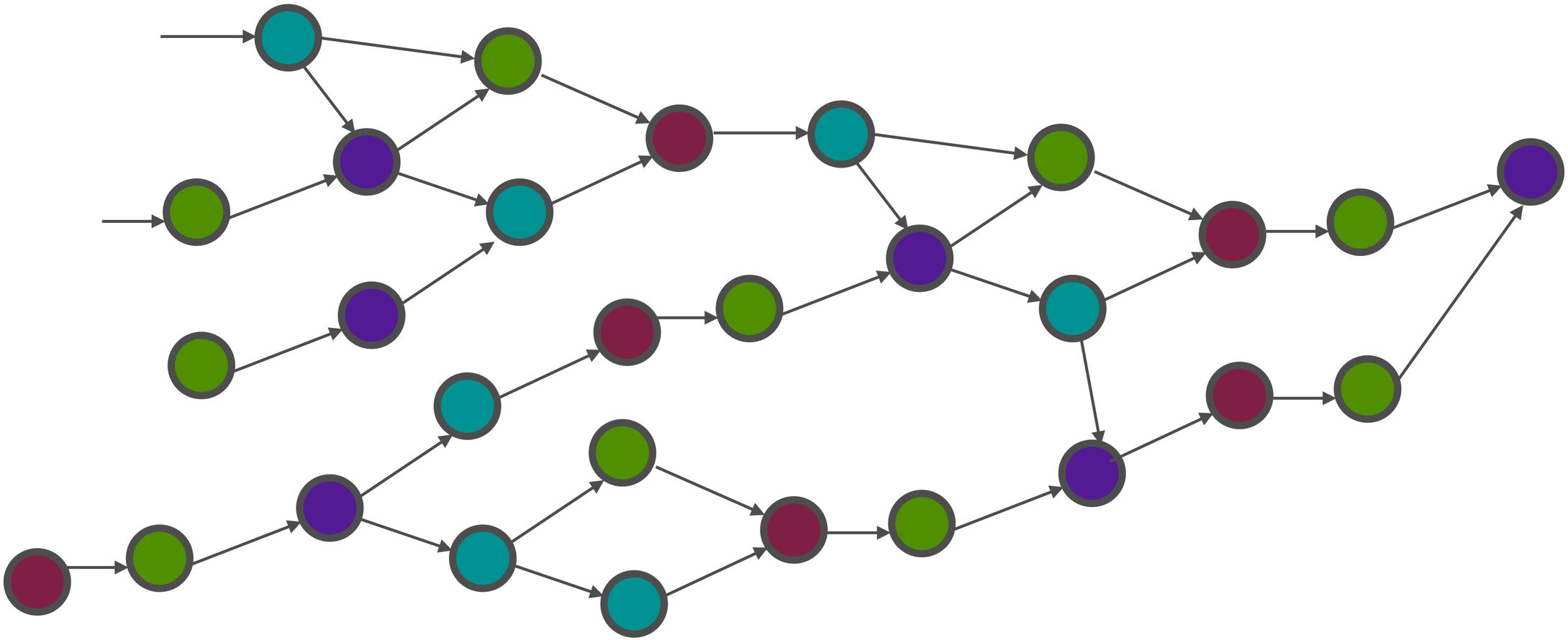
Computation Graphs



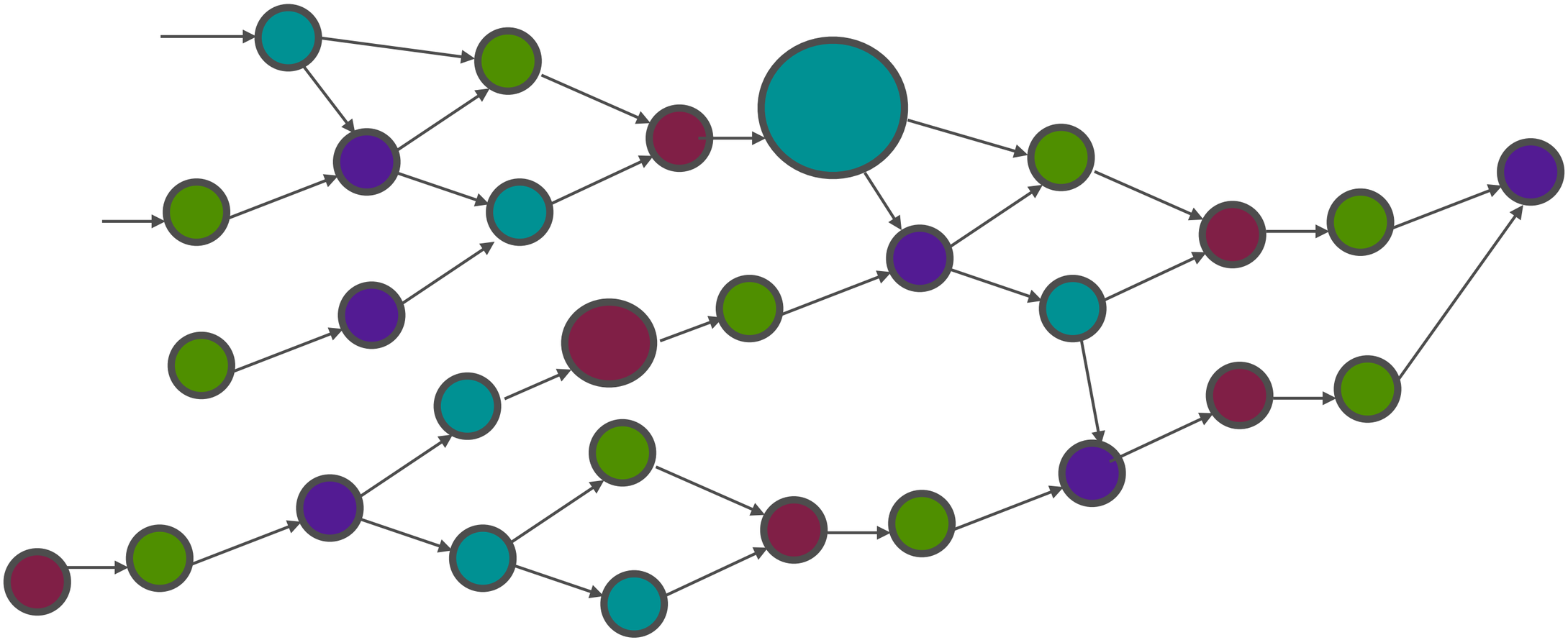
Computation Graphs



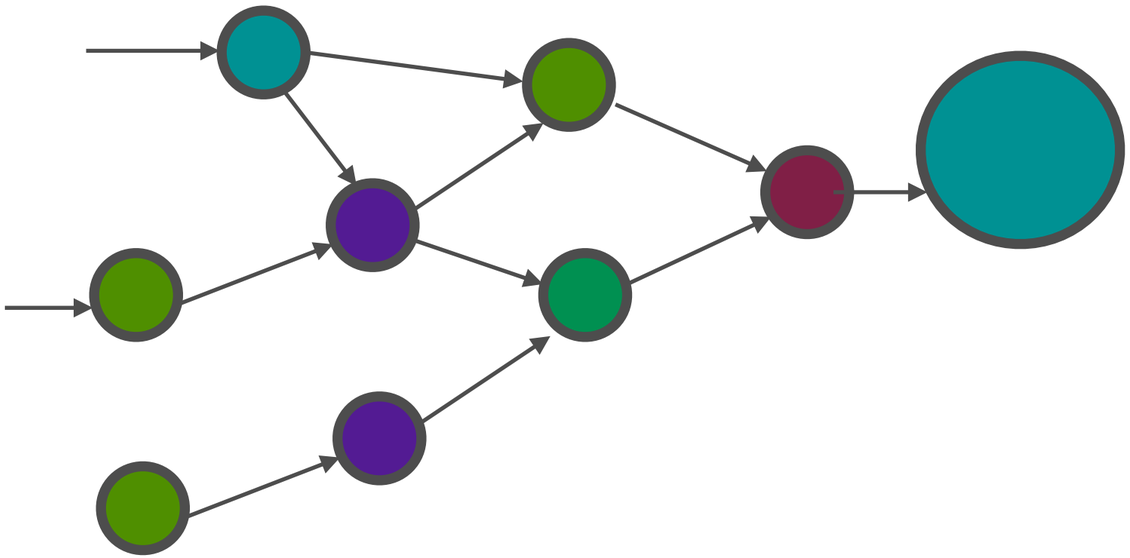
Computation Graphs



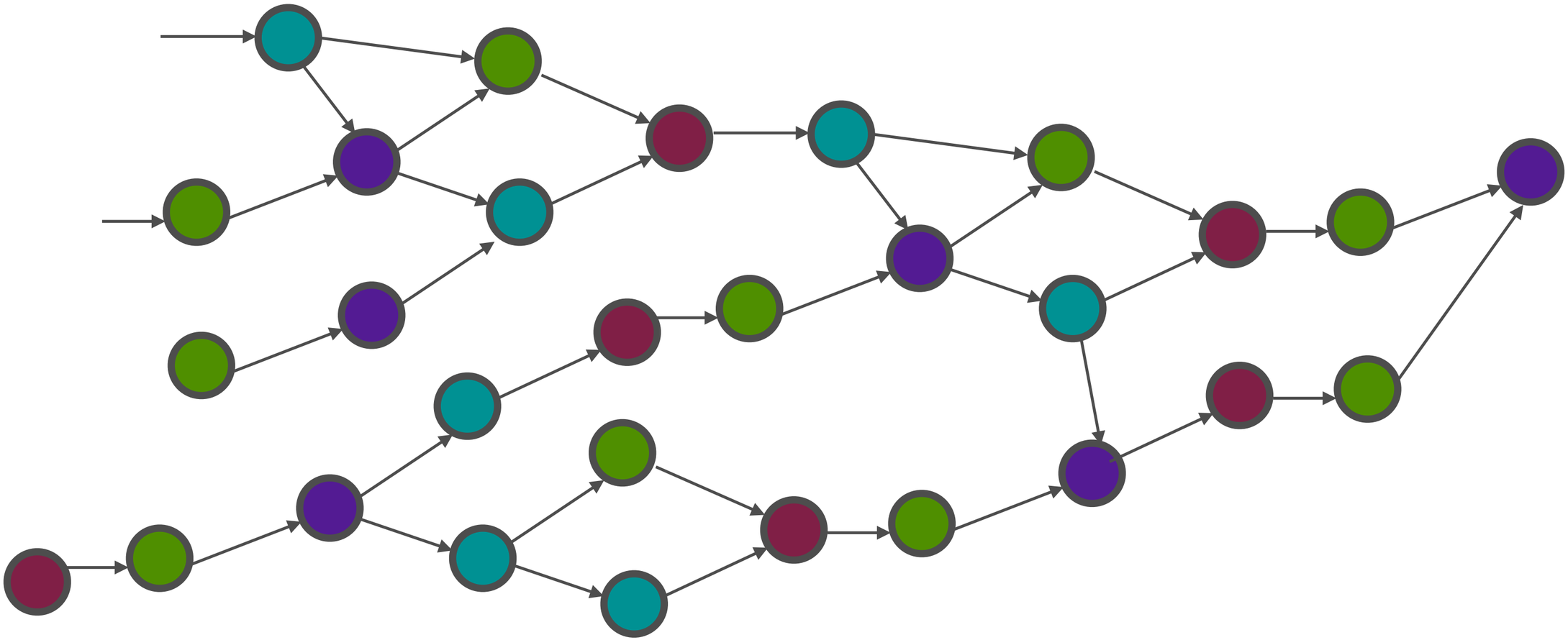
Computation Graphs



Computation Graphs

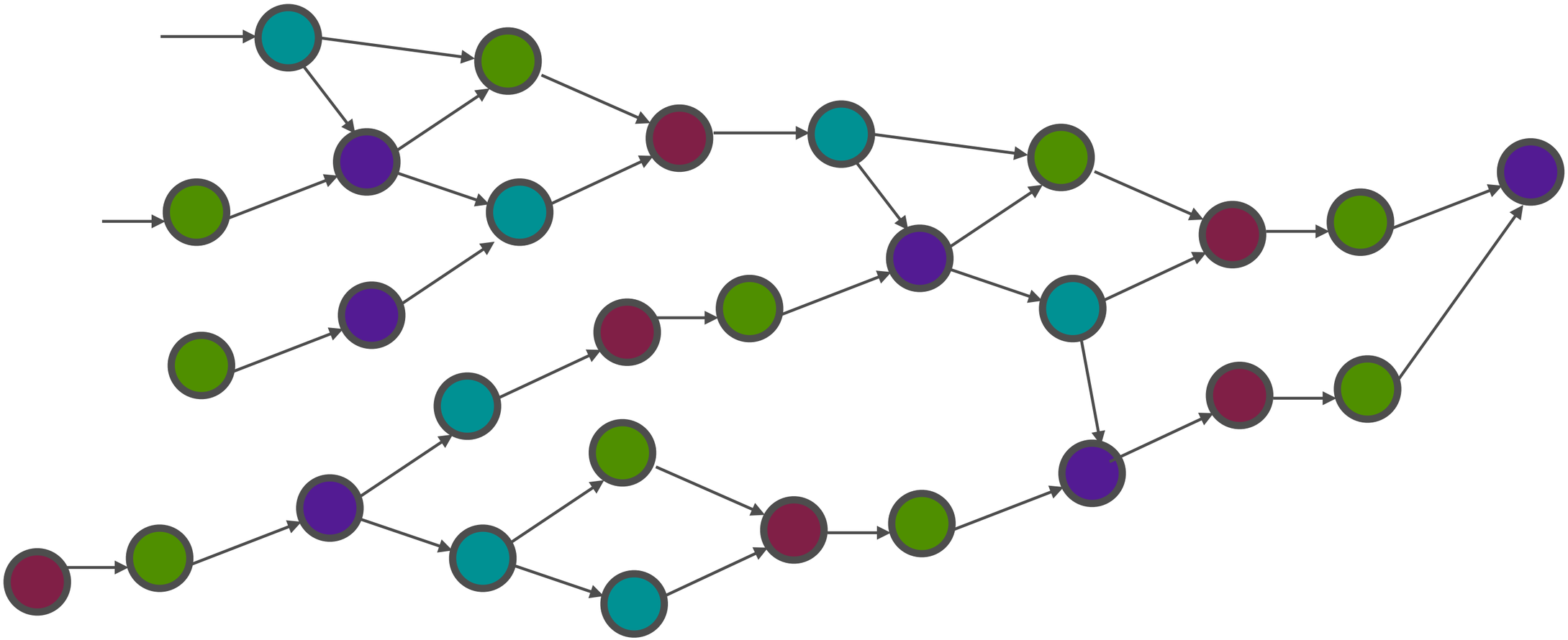


Computation Graphs

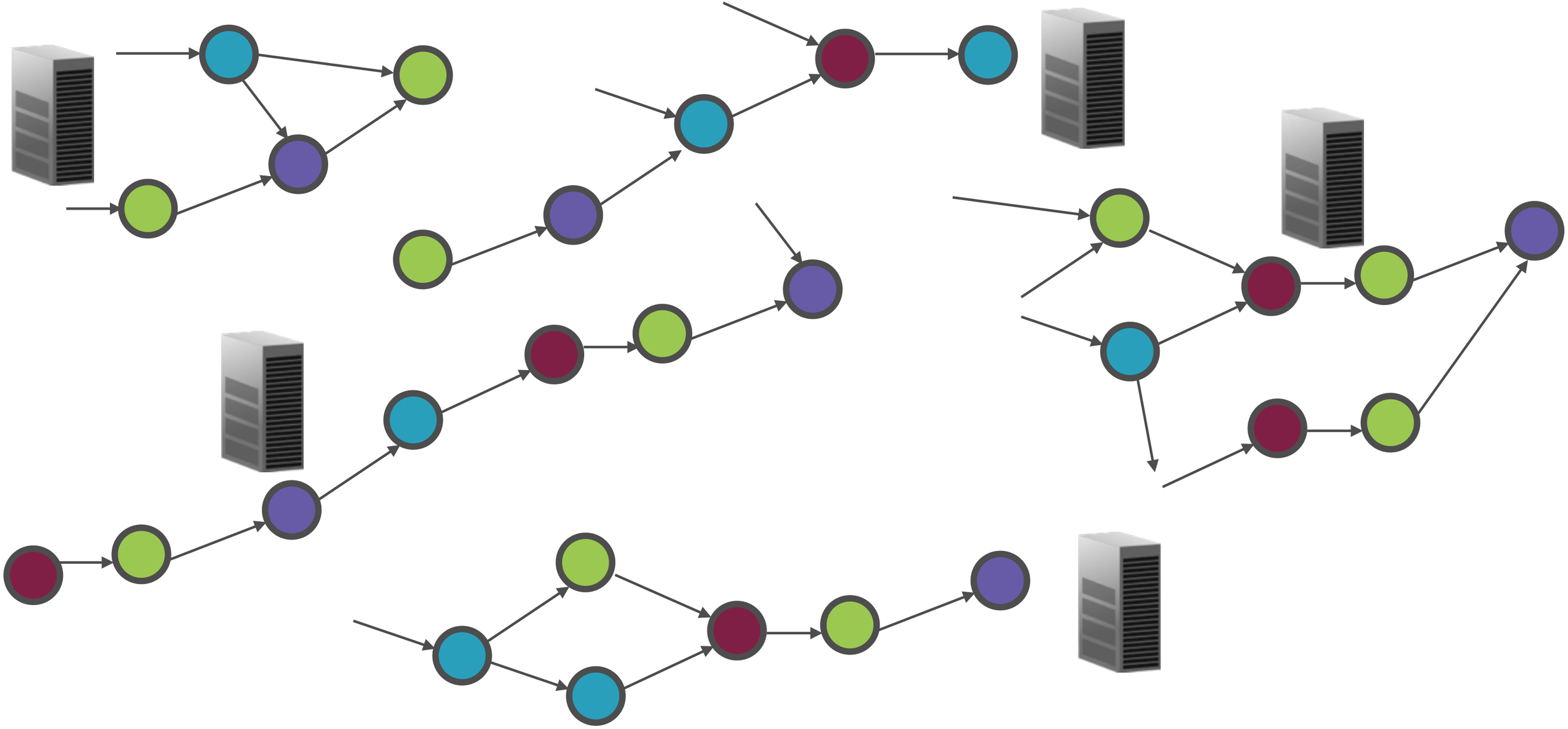


TensorFlow calculates only that portion of the graph which is required

Computation Graphs



Running Graphs on a Distributed System



Multiple portions of the graph can be run in parallel across machines in the cluster

Demo

Executing simple math commands in TensorFlow

Tensor

Tensor

The central unit of data in TensorFlow. A tensor consists of a set of primitive values shaped into an array of any number of dimensions.

https://www.tensorflow.org/

Tensor

The central unit of data in TensorFlow. A tensor consists of a set of primitive values shaped into an array of any number of dimensions.

https://www.tensorflow.org/

Tensor

The central unit of data in TensorFlow. A tensor consists of a set of primitive values shaped into an array of any number of dimensions.

https://www.tensorflow.org/

Data Is Represented as Tensors

Scalars are 0-D tensors

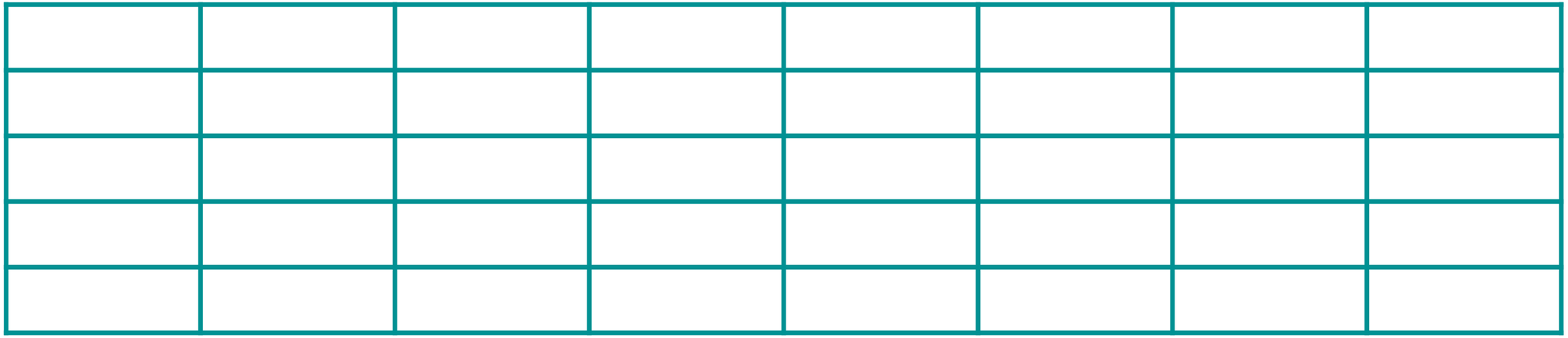
3, 6.7, “a”

Data Is Represented as Tensors

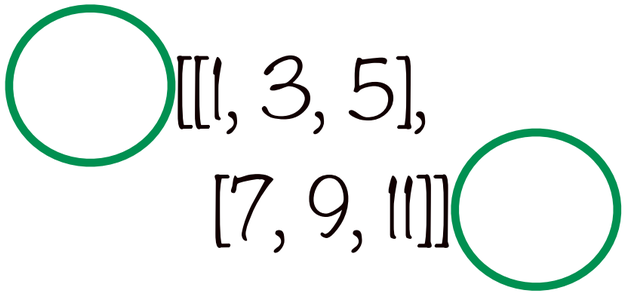
Vectors are 1-D tensors

[1, 3, 5, 7, 9]

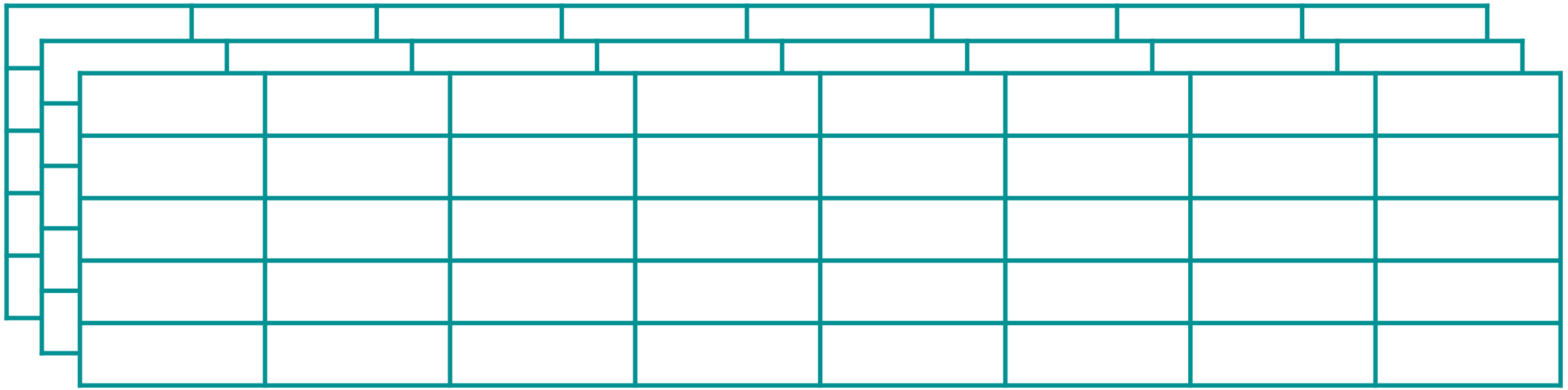
Data Is Represented as Tensors



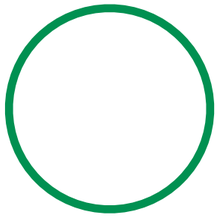
Matrices are 2-D tensors



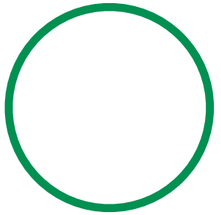
Data Is Represented as Tensors



N-Dimensional matrices are N-D tensors



[[[1, 2], [3, 4], [5, 6]],
[[7, 8], [9, 10], [11, 12]]]



Characterization of Tensors

Rank

The number of dimensions in a tensor

Shape

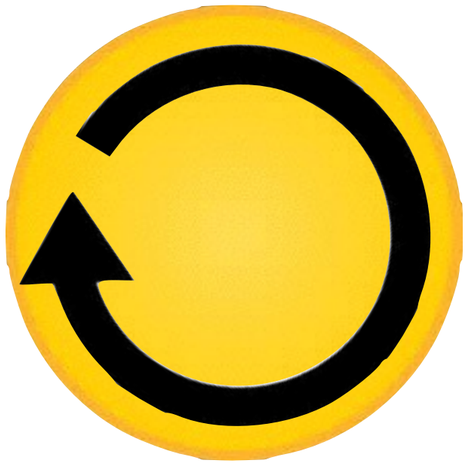
The number of elements in each dimension

Data Type

The data type of each element in the tensor

Rank

Tensor	Rank
4	0
[1, 2, 3]	1
[[1, 2], [3, 4]]	2
[[[1], [2]], [[3], [4]]]	3



Shape

Tensor	Shape
4	[]
[1, 2, 3]	[3]
[[1, 2, 3], [4, 5, 6]]	[3, 2]
[[[1], [2]], [[3], [4]]]	[2, 2, 1]

Data Type

int

float

string

boolean

Rank, shape and data types are 3 important characteristics which define a Tensor

Summary

Worked with the directed-acyclic graph to model problems in TensorFlow

Understood constants, operators and sessions

Understood Tensor characteristics such as rank, shape and data type

Run TensorFlow programs and visualized results using TensorBoard

Digging Deeper into Fundamentals

Overview

Work through the logic of a specific machine learning problem

Run programs on different inputs using placeholders and feed dictionaries

Use variables to hold values which the program updates

Make TensorBoards more useful using named scopes

Understanding Linear Regression

X Causes Y

Cause

Independent variable

Effect

Dependent variable

Wealth Increases Life Expectancy

Cause

Wealth of individuals

Effect

How long they expect to live

Lower Home Prices Away from the City

Cause

Distance in miles from the city center

Effect

Price per square foot of homes

X Causes Y

Cause

Explanatory variable

Effect

Dependent variable

Linear Regression

y

Price per square foot

Regression Line: y = A + Bx

X

Distance from the city center

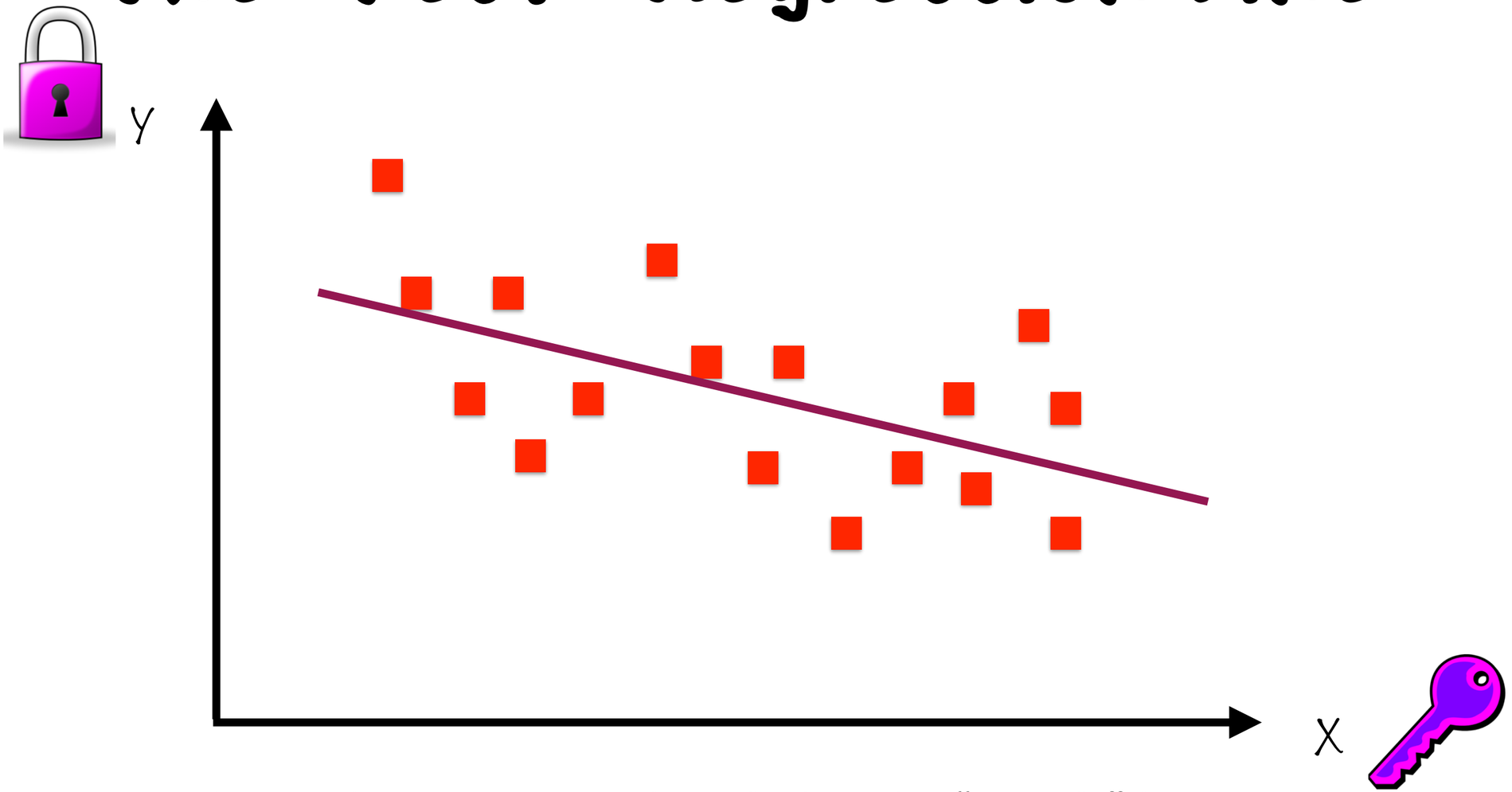
Simple Regression

Regression Equation:

$$y = A + Bx$$

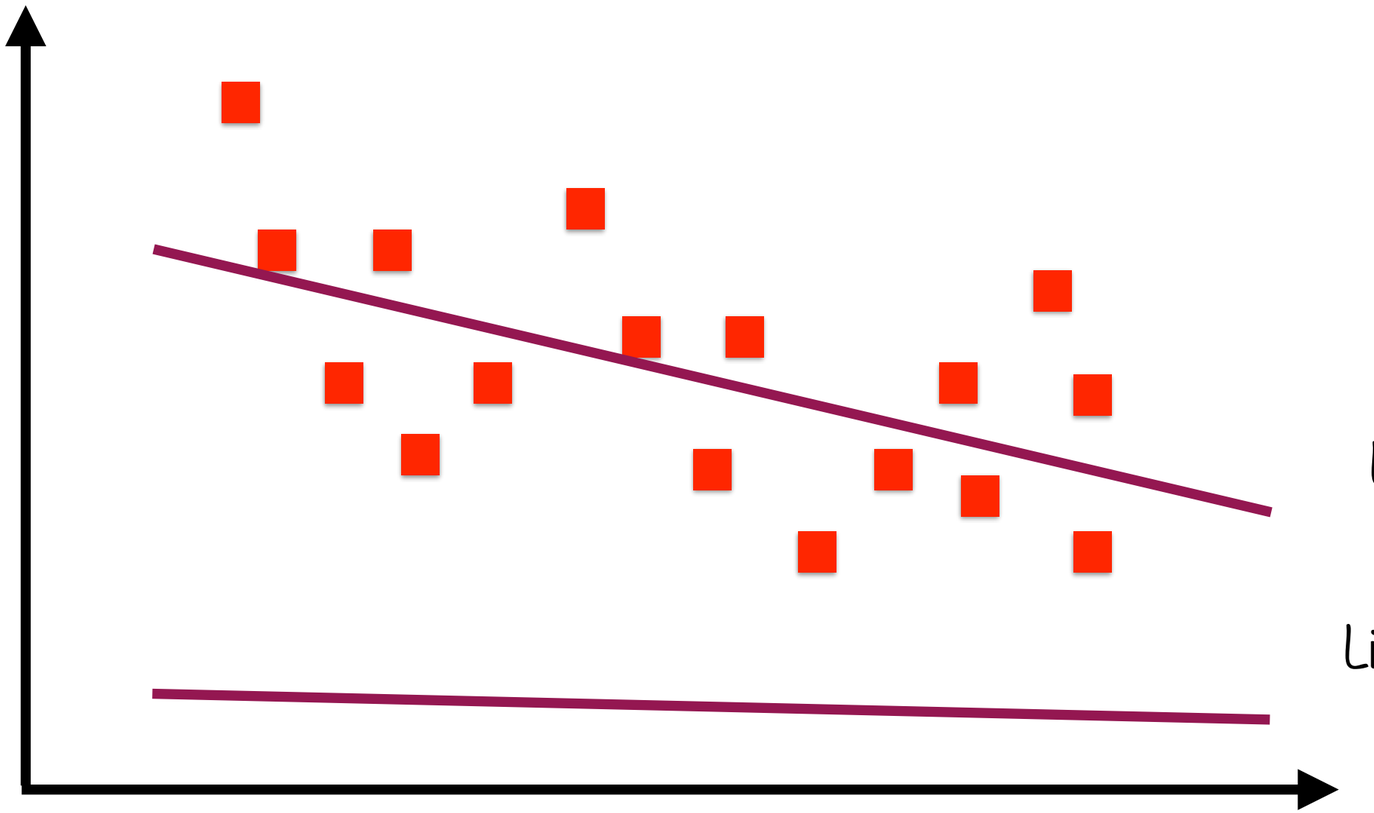
- $y_1 = A + Bx_1$
- $y_2 = A + Bx_2$
- $y_3 = A + Bx_3$
- ...
- $y_n = A + Bx_n$

The "Best" Regression Line



Linear Regression involves finding the "best fit" line

The "Best" Regression Line

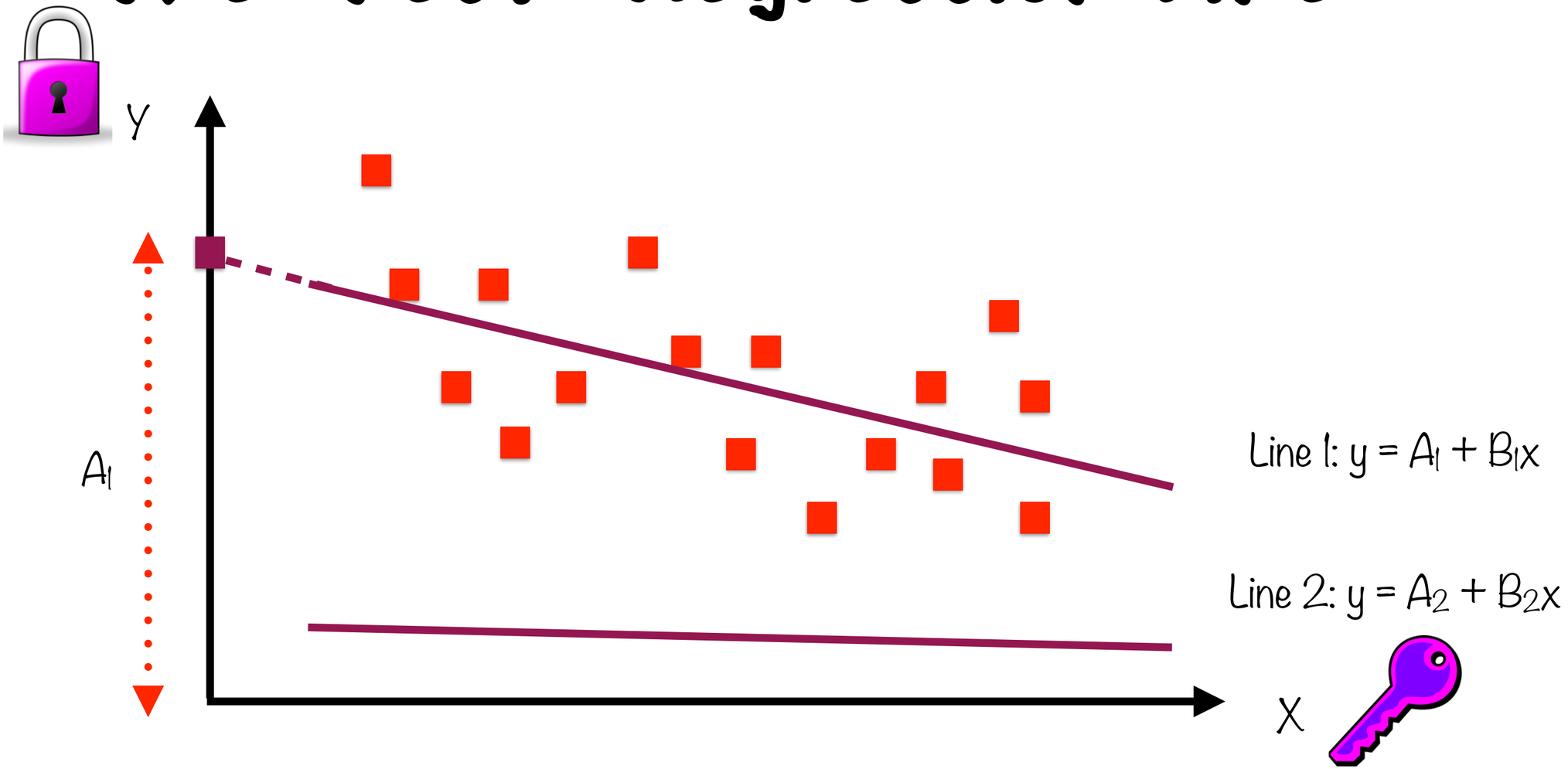


Line 1: $y = A_1 + B_1x$

Line 2: $y = A_2 + B_2x$

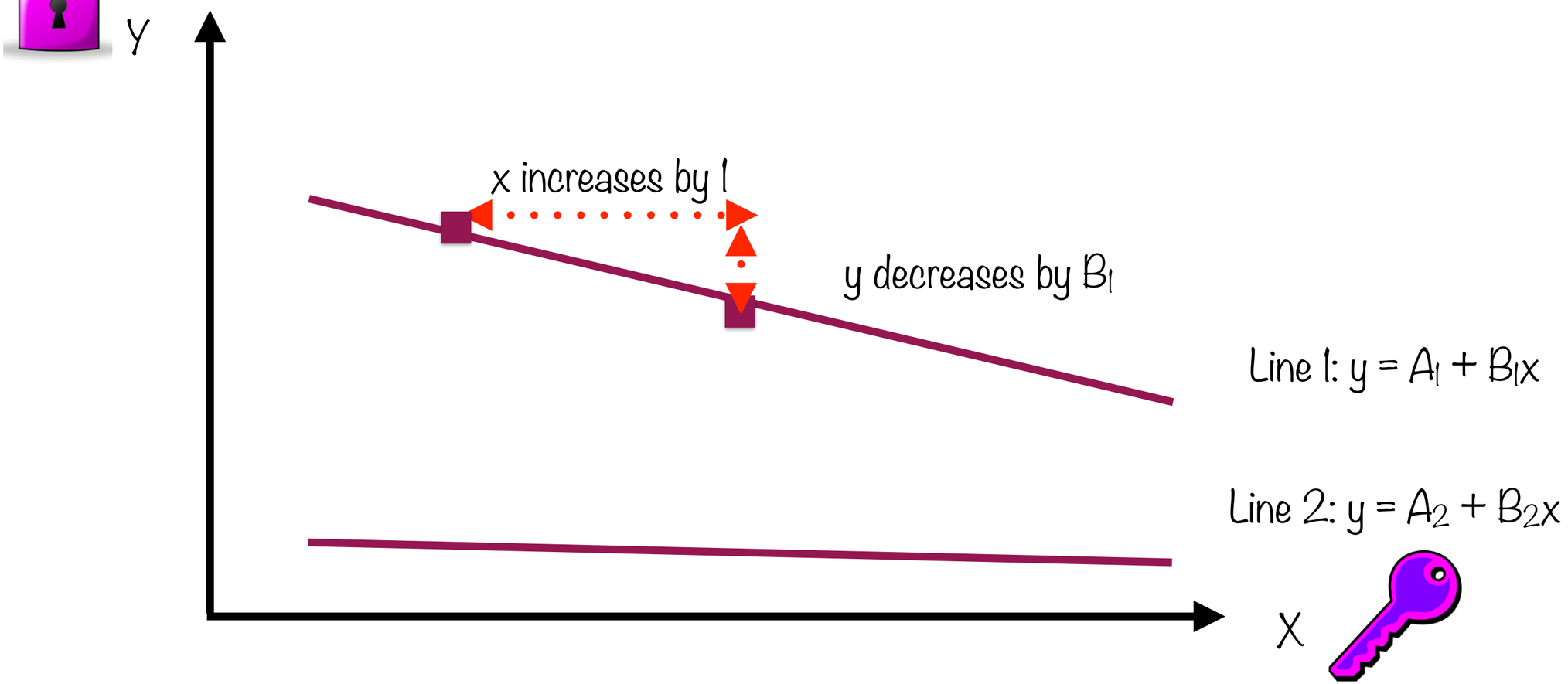
Let's compare two lines, Line 1 and Line 2

The "Best" Regression Line



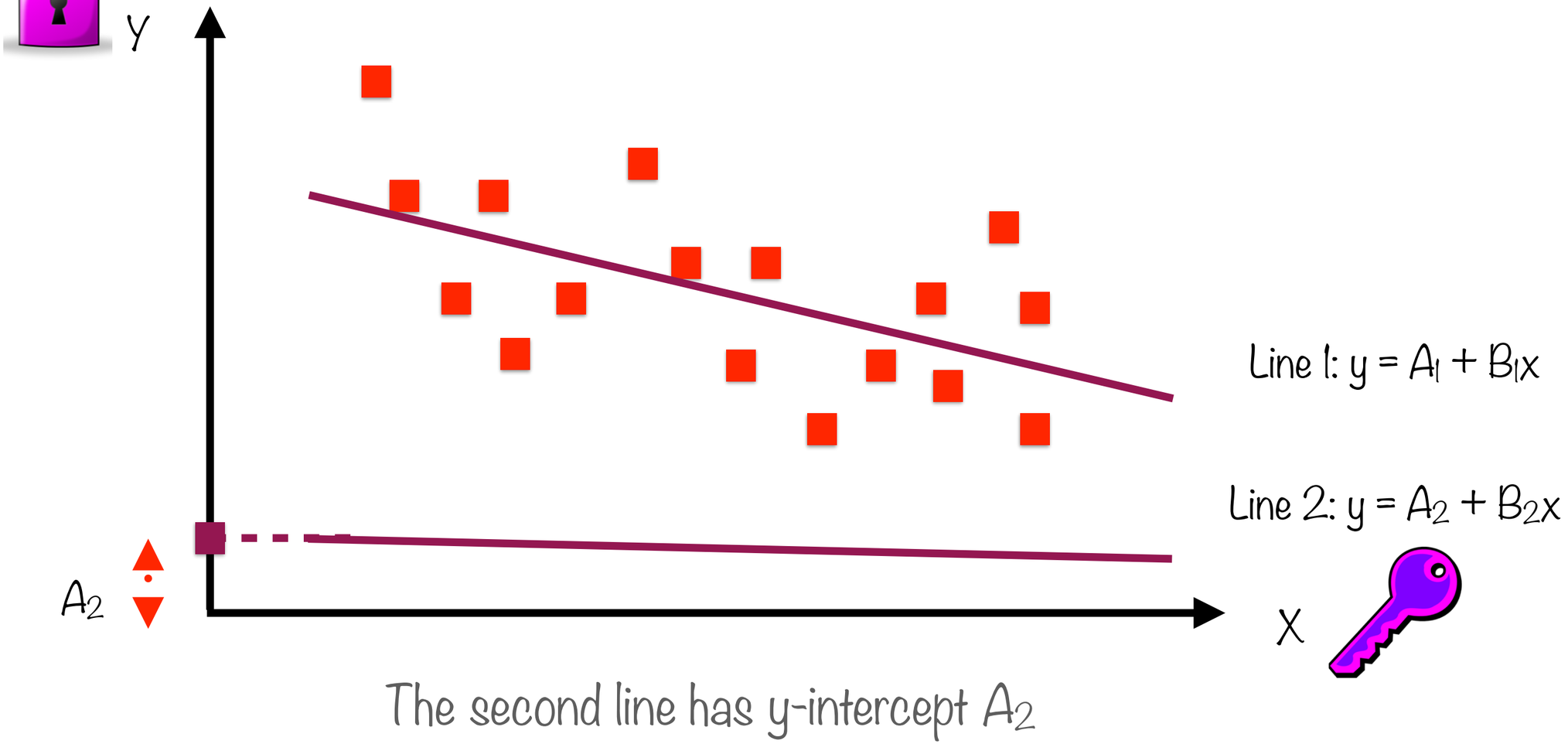
The first line has y-intercept A_1

The "Best" Regression Line



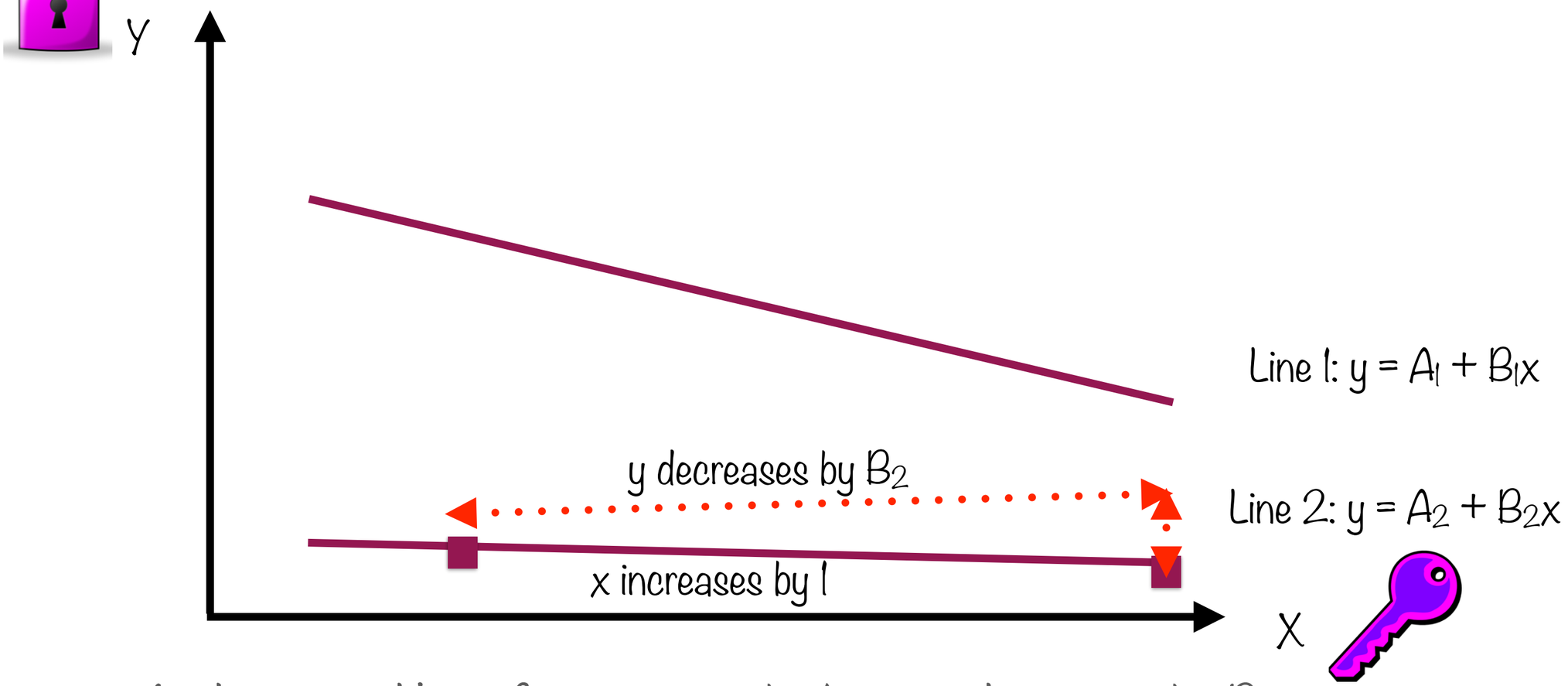
In the first line, if x increases by 1 unit, y decreases by B_1 units

The "Best" Regression Line



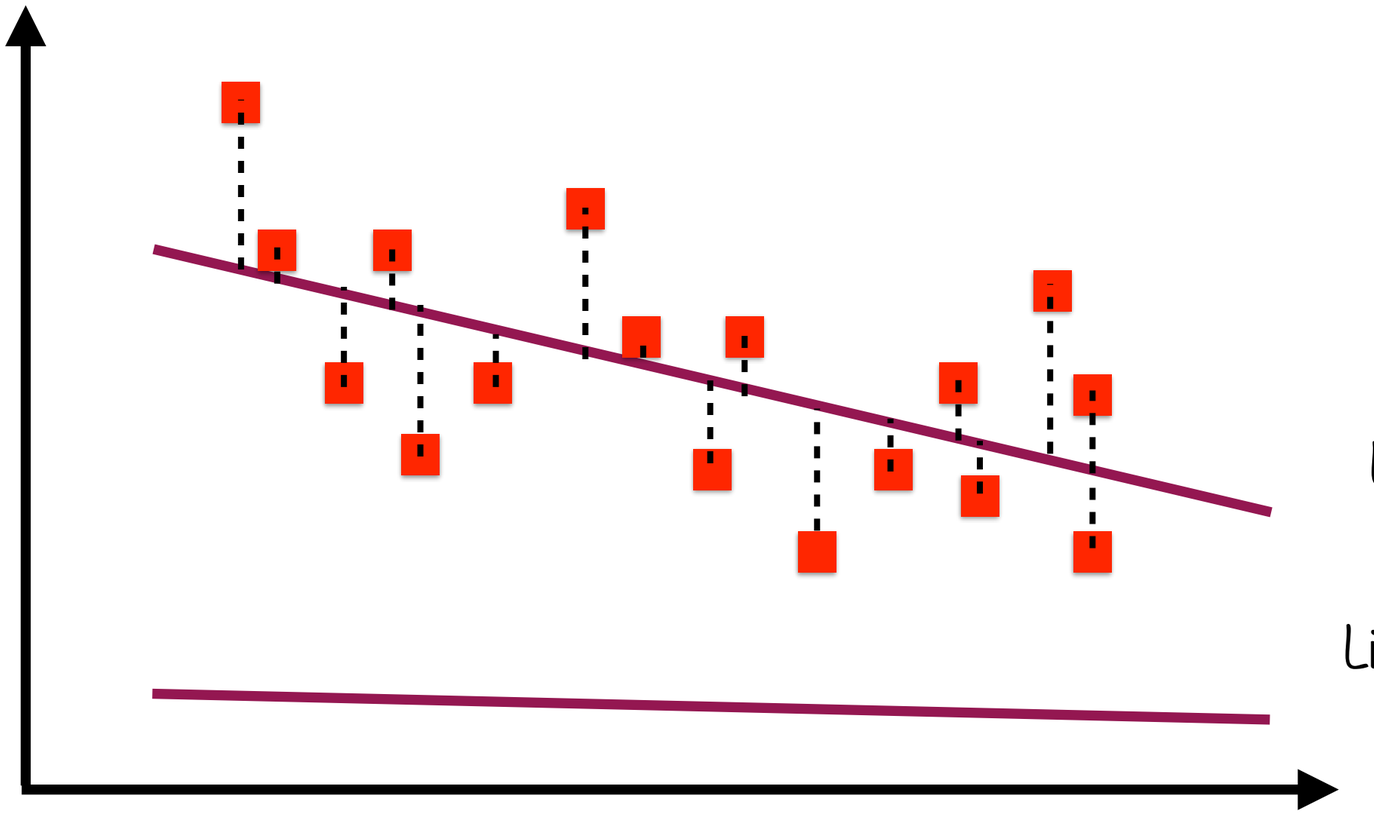
The second line has y-intercept A_2

The "Best" Regression Line



In the second line, if x increases by 1 unit, y decreases by B_2 units

Minimising Least Square Error

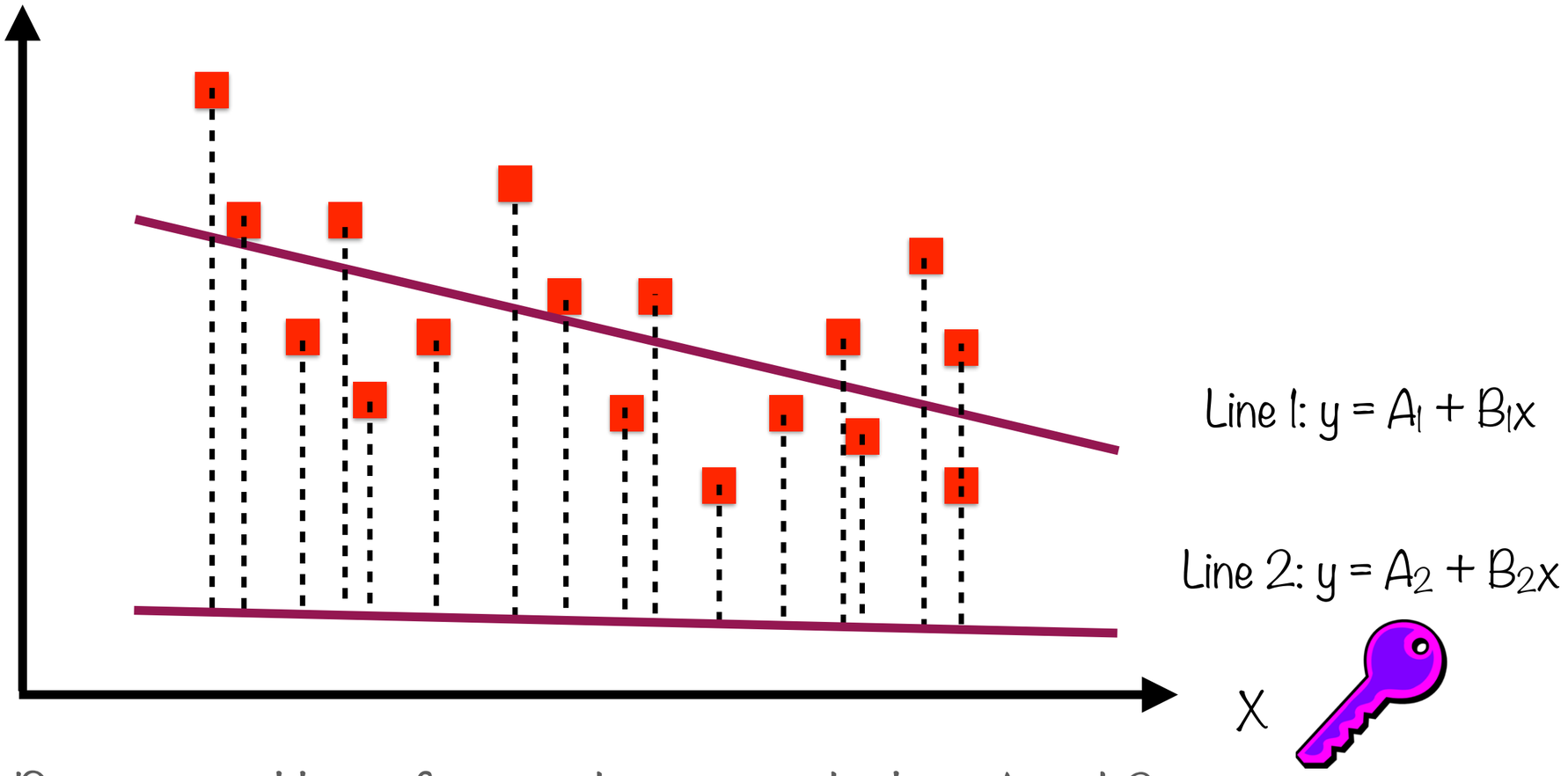


Line 1: $y = A_1 + B_1x$

Line 2: $y = A_2 + B_2x$

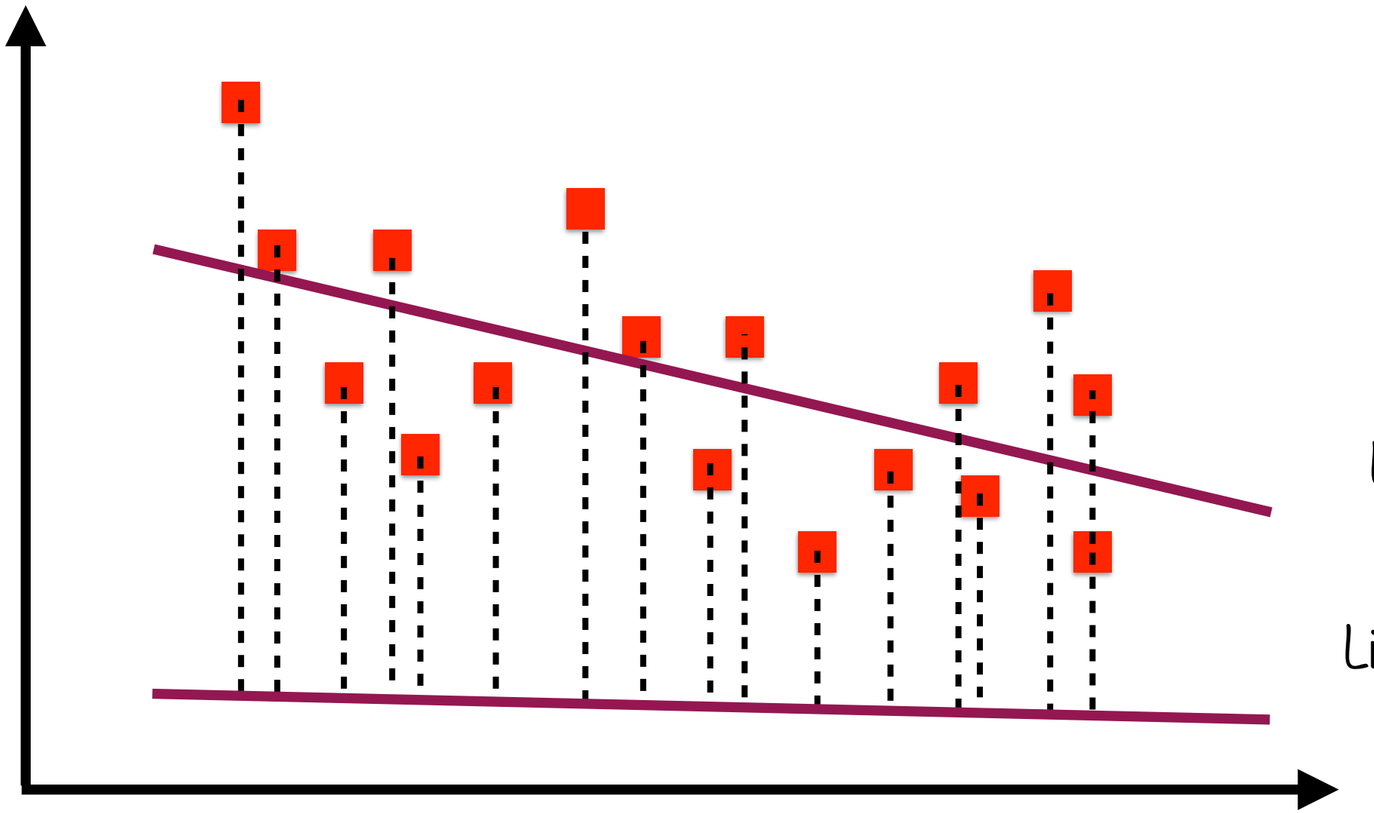
Drop vertical lines from each point to the lines 1 and 2

Minimising Least Square Error



Drop vertical lines from each point to the lines 1 and 2

Minimising Least Square Error

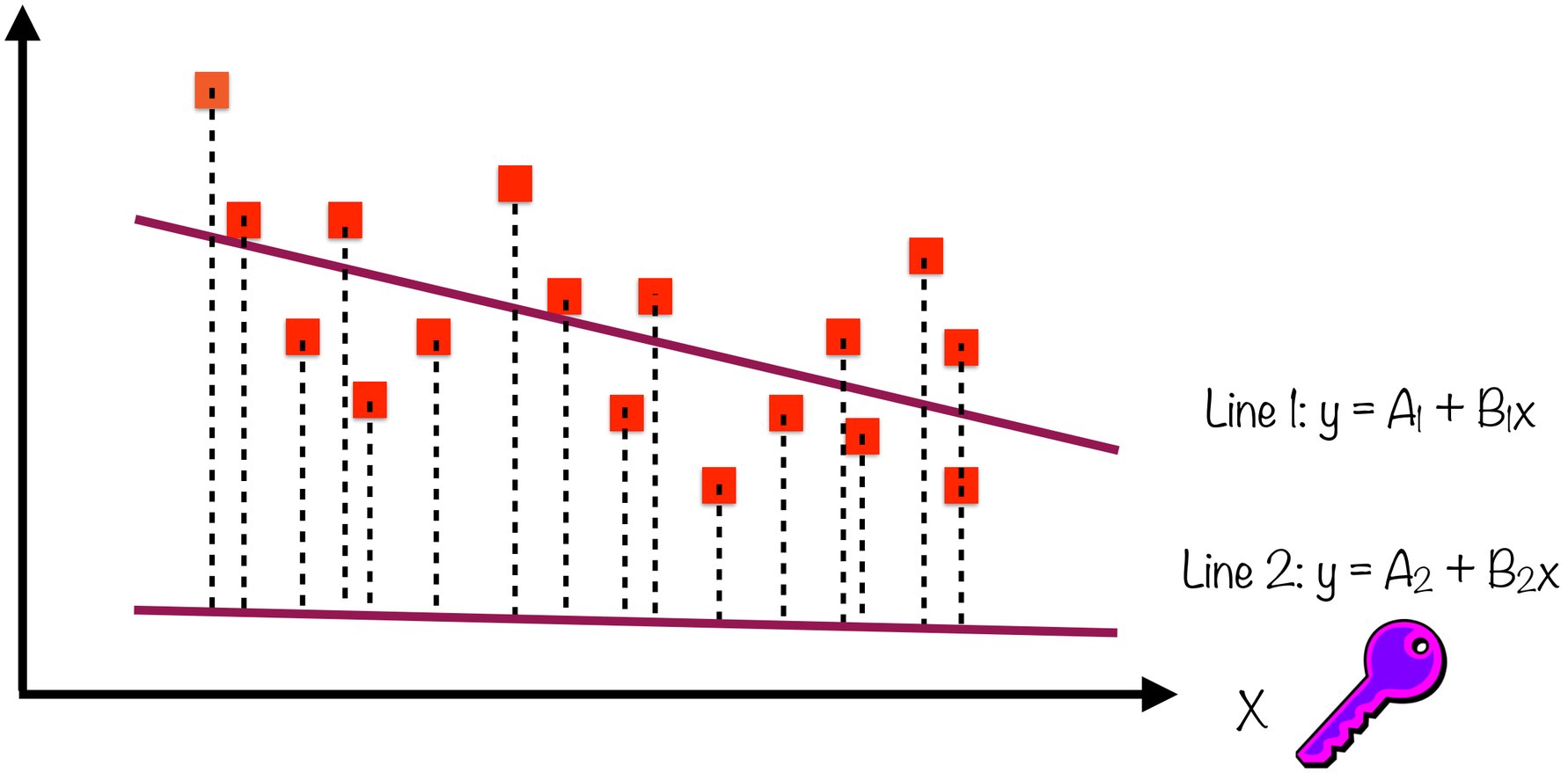


Line 1: $y = A_1 + B_1x$

Line 2: $y = A_2 + B_2x$

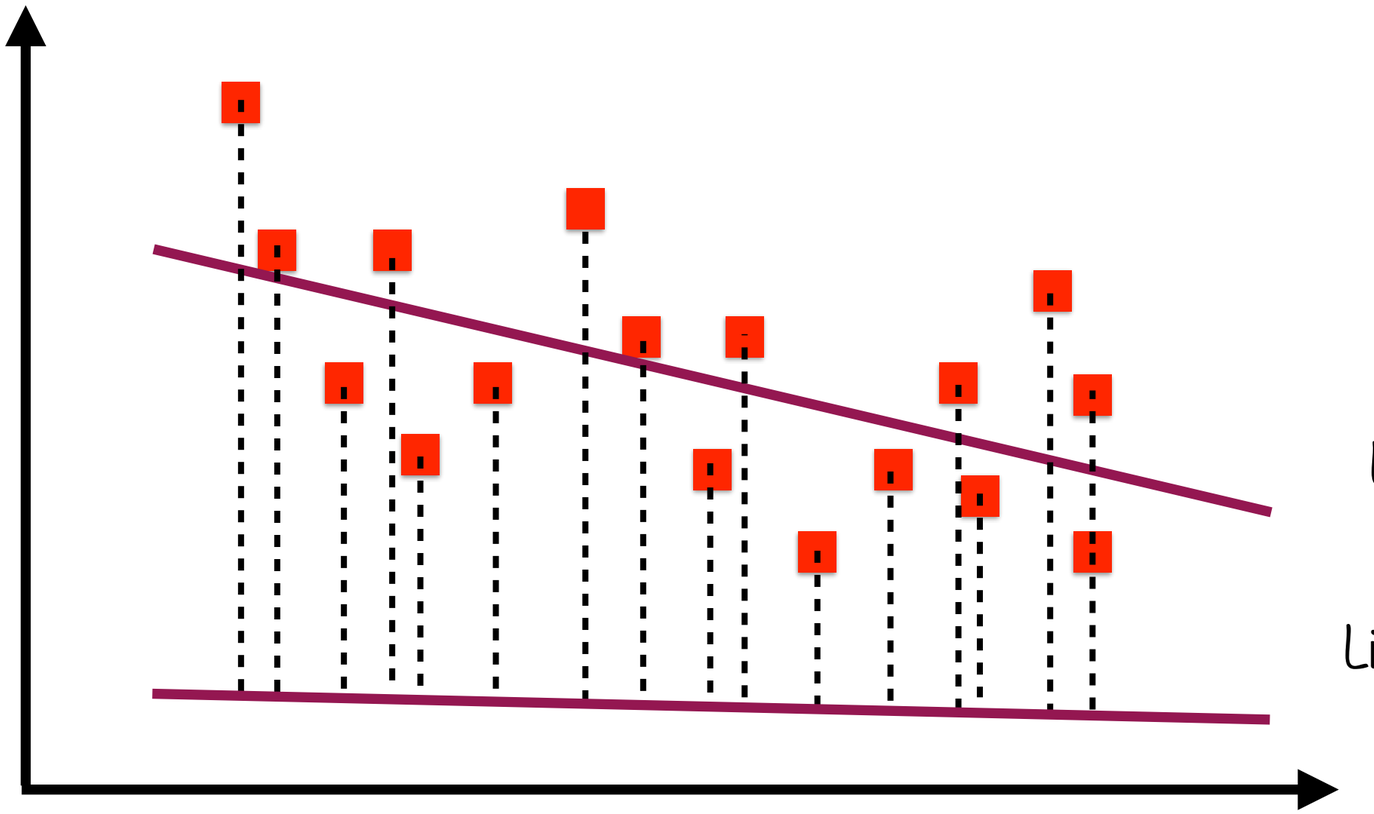
The "best fit" line is the one where the sum of the squares of the lengths of these dotted lines is minimum

Minimising Least Square Error



The "best fit" line is the one where the sum of the squares of the lengths of these dotted lines is minimum

Minimising Least Square Error

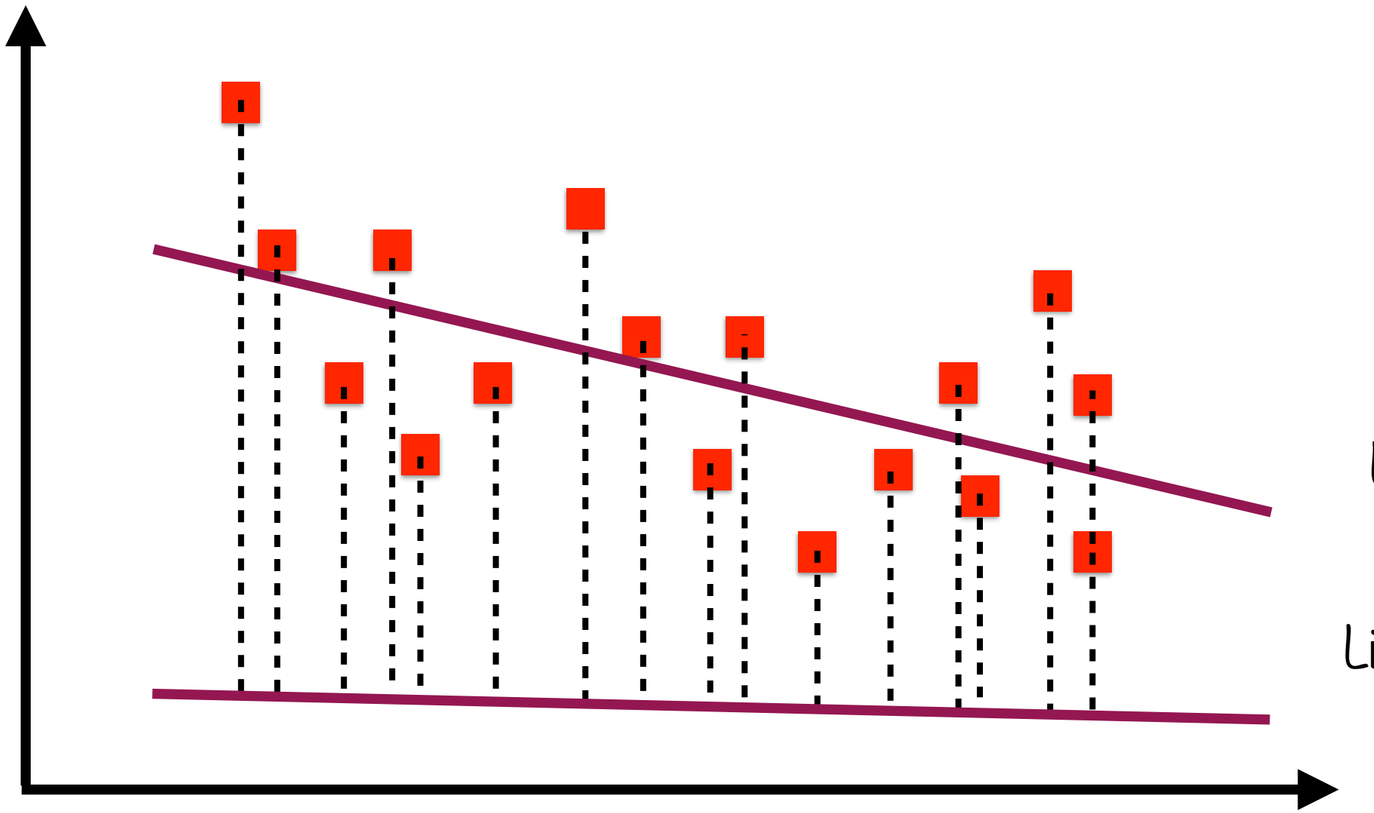


Line 1: $y = A_1 + B_1x$

Line 2: $y = A_2 + B_2x$

The "best fit" line is the one where the sum of the squares of the lengths of the errors is minimum

Minimising Least Square Error

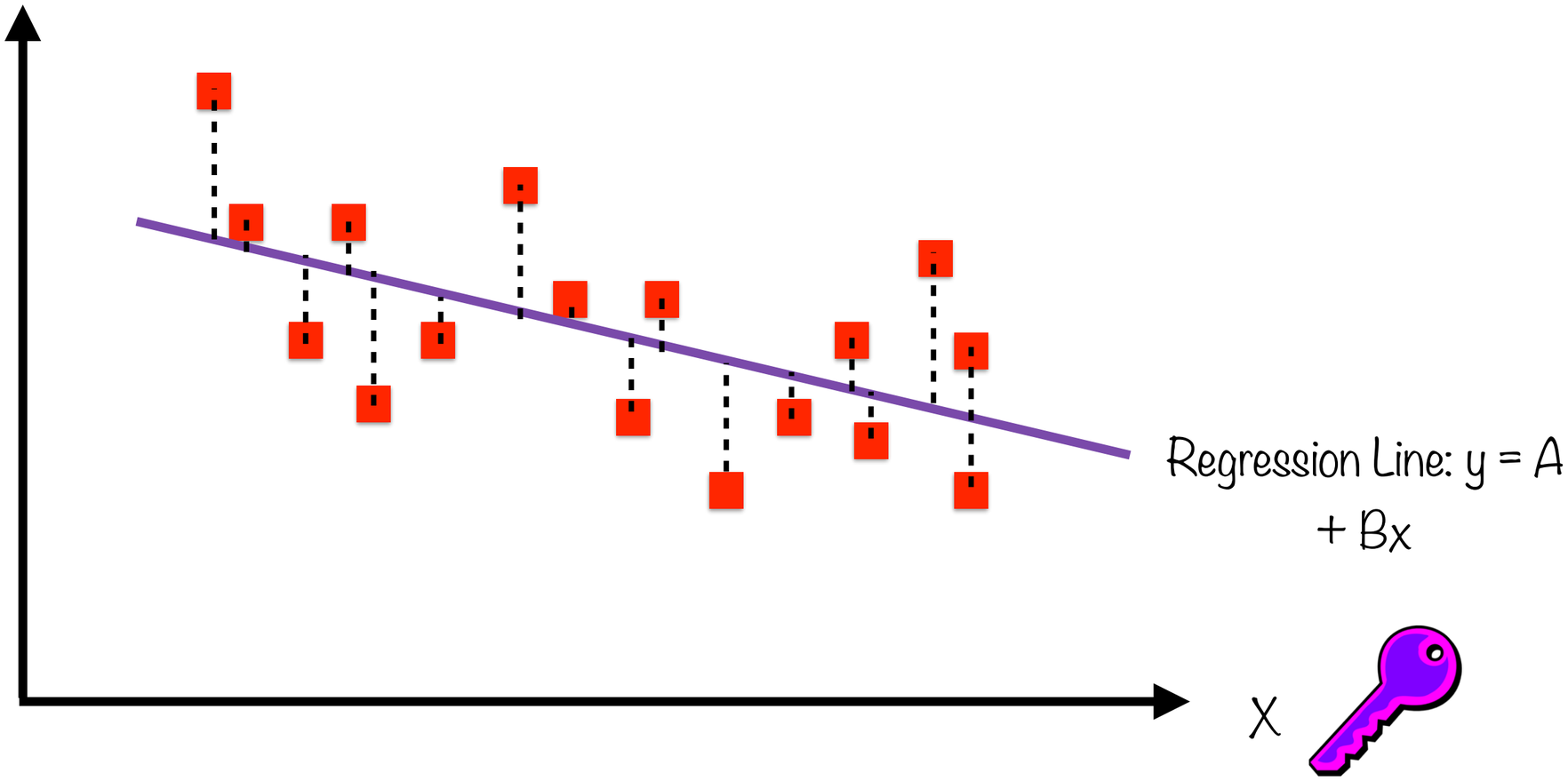


Line 1: $y = A_1 + B_1x$

Line 2: $y = A_2 + B_2x$

The "best fit" line is the one where the sum of the squares of the lengths of the errors is minimum

Minimising Least Square Error



Linear Regression

y

Price per square foot

Regression Line: y = A + Bx

X

Distance from the city center

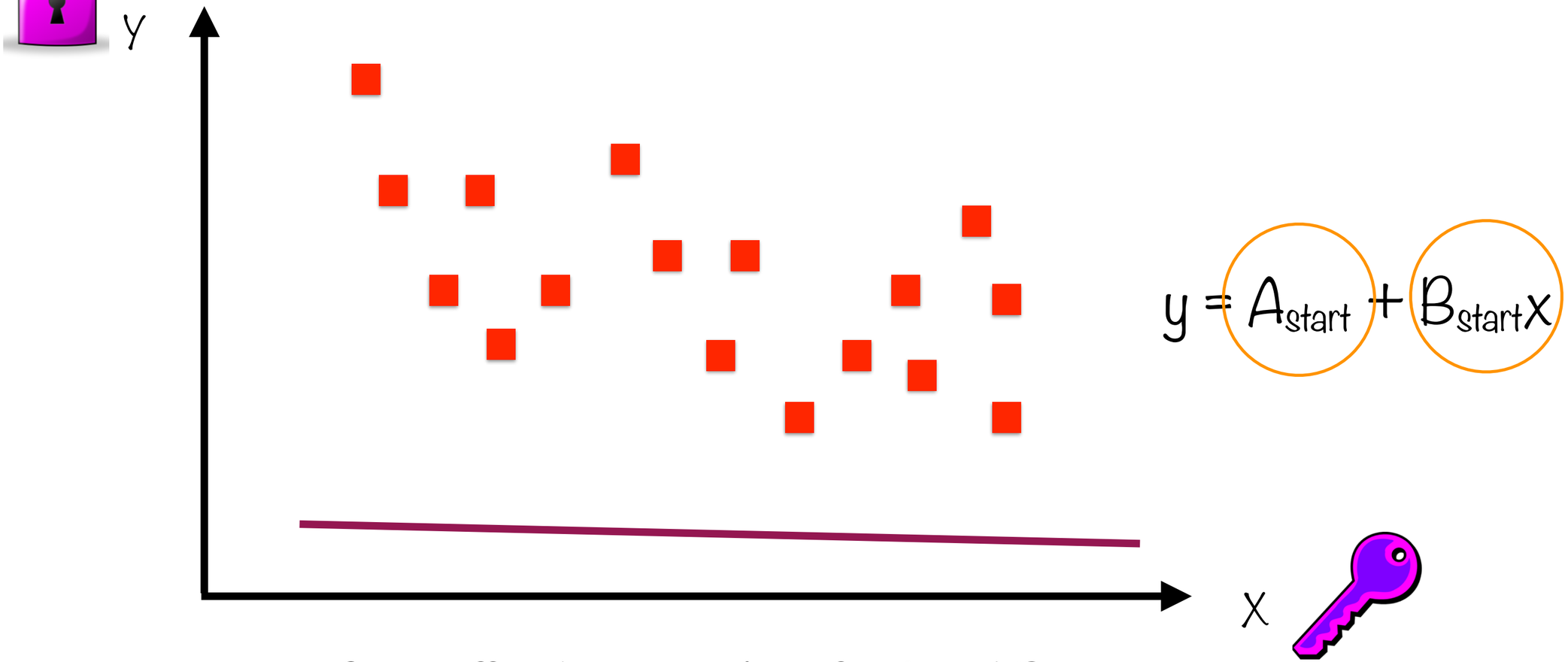
Linear Regression Algorithms in Practice

Estimate initial values for A and B

Find the errors for the regression line with those values

Feed errors back and get new values for A and B

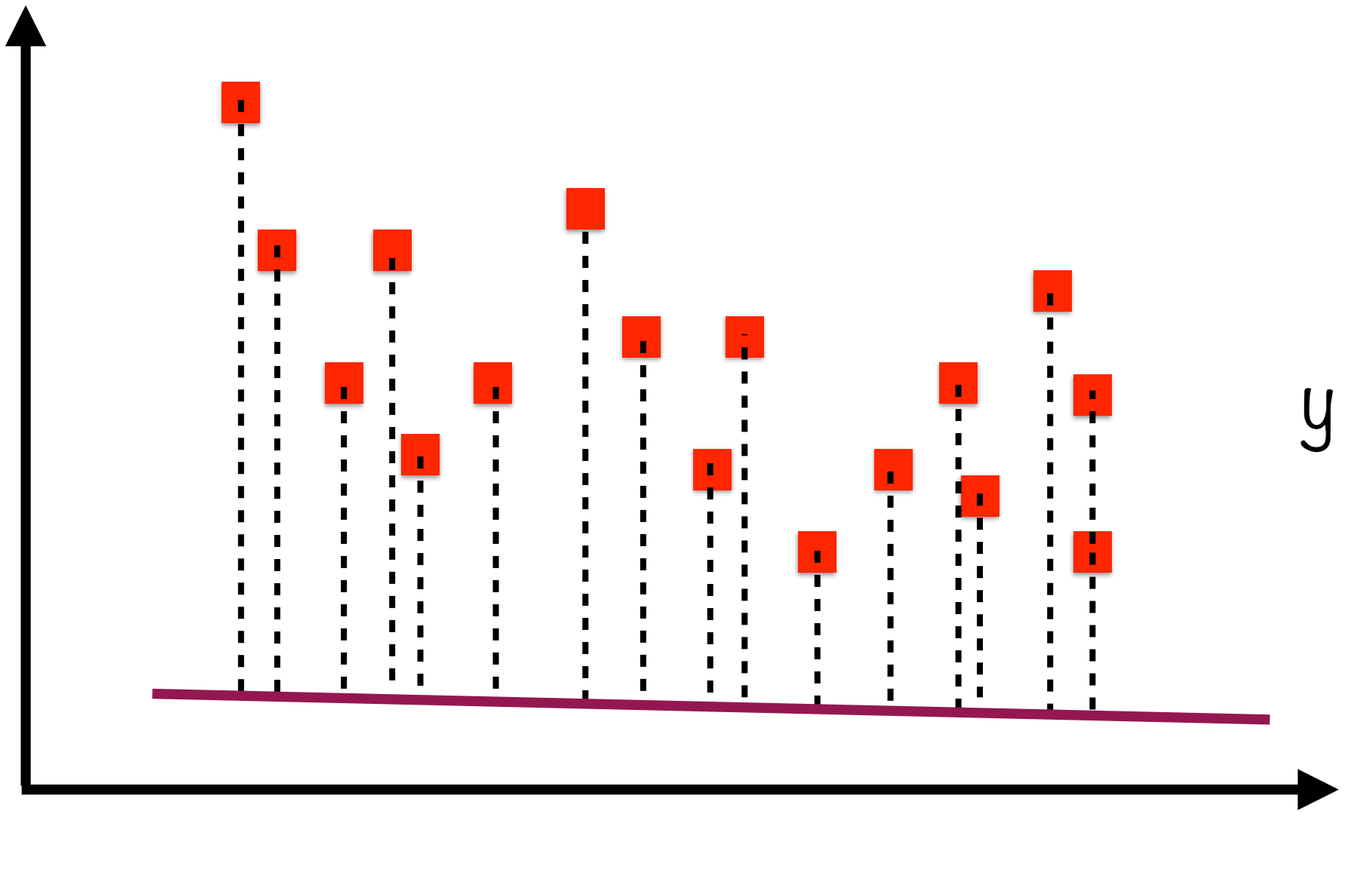
The "Best" Regression Line



$$y = A_{start} + B_{start}x$$

Start off with some values for A and B

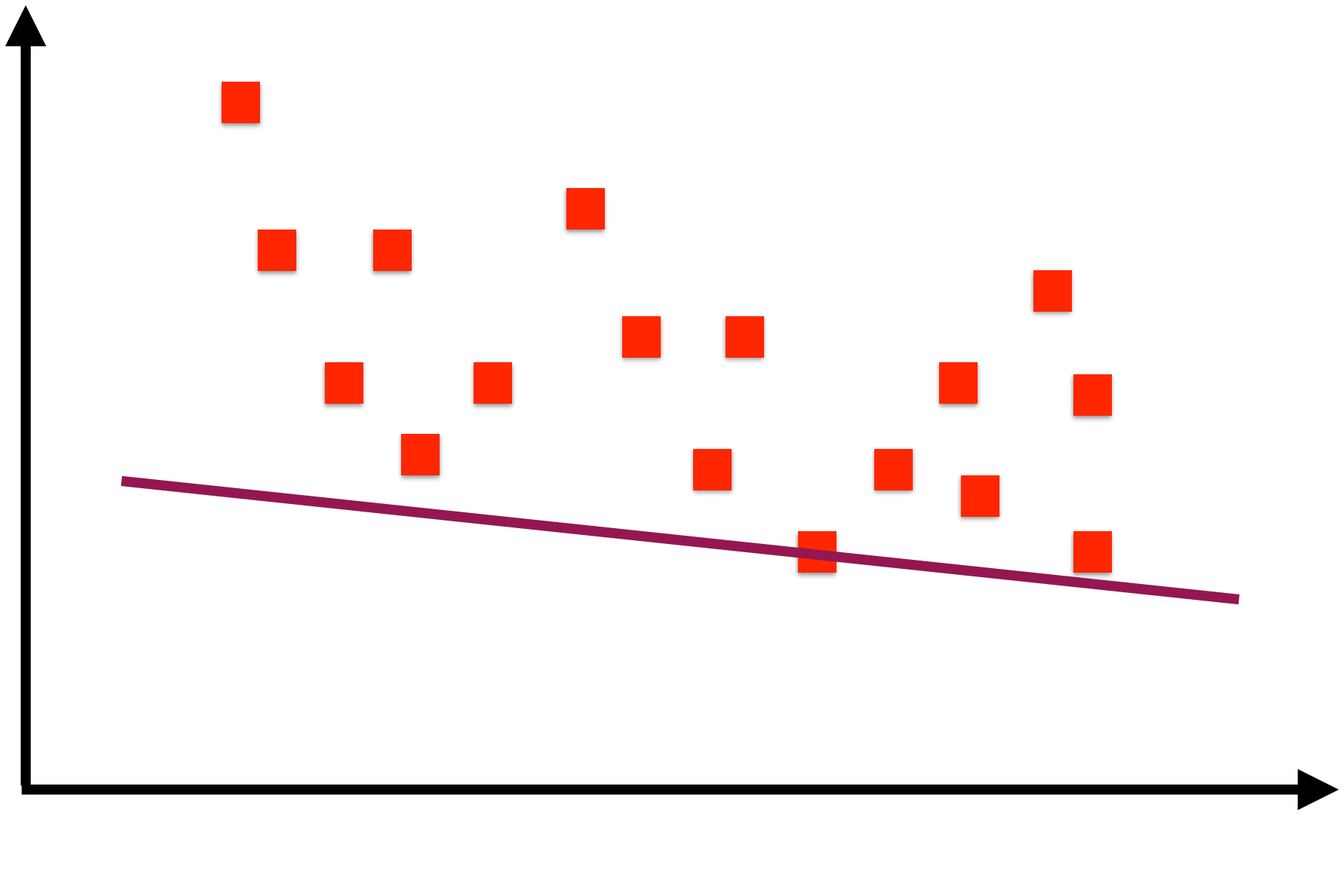
The "Best" Regression Line



$$y = A_{start} + B_{start}x$$

Calculate the least square error and feed that back

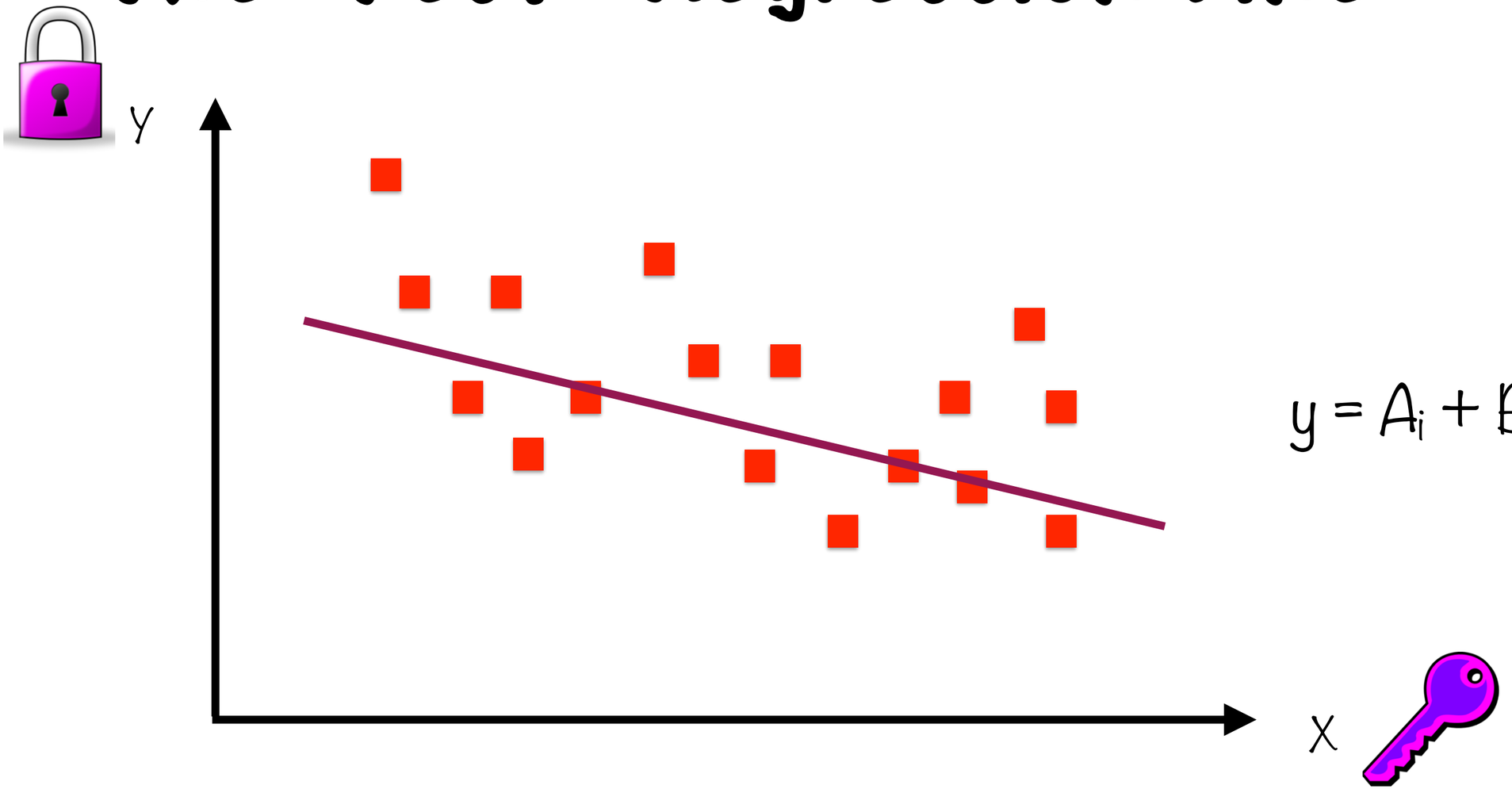
The "Best" Regression Line



$$y = A_i + B_i x$$

This will give us new values for A and B

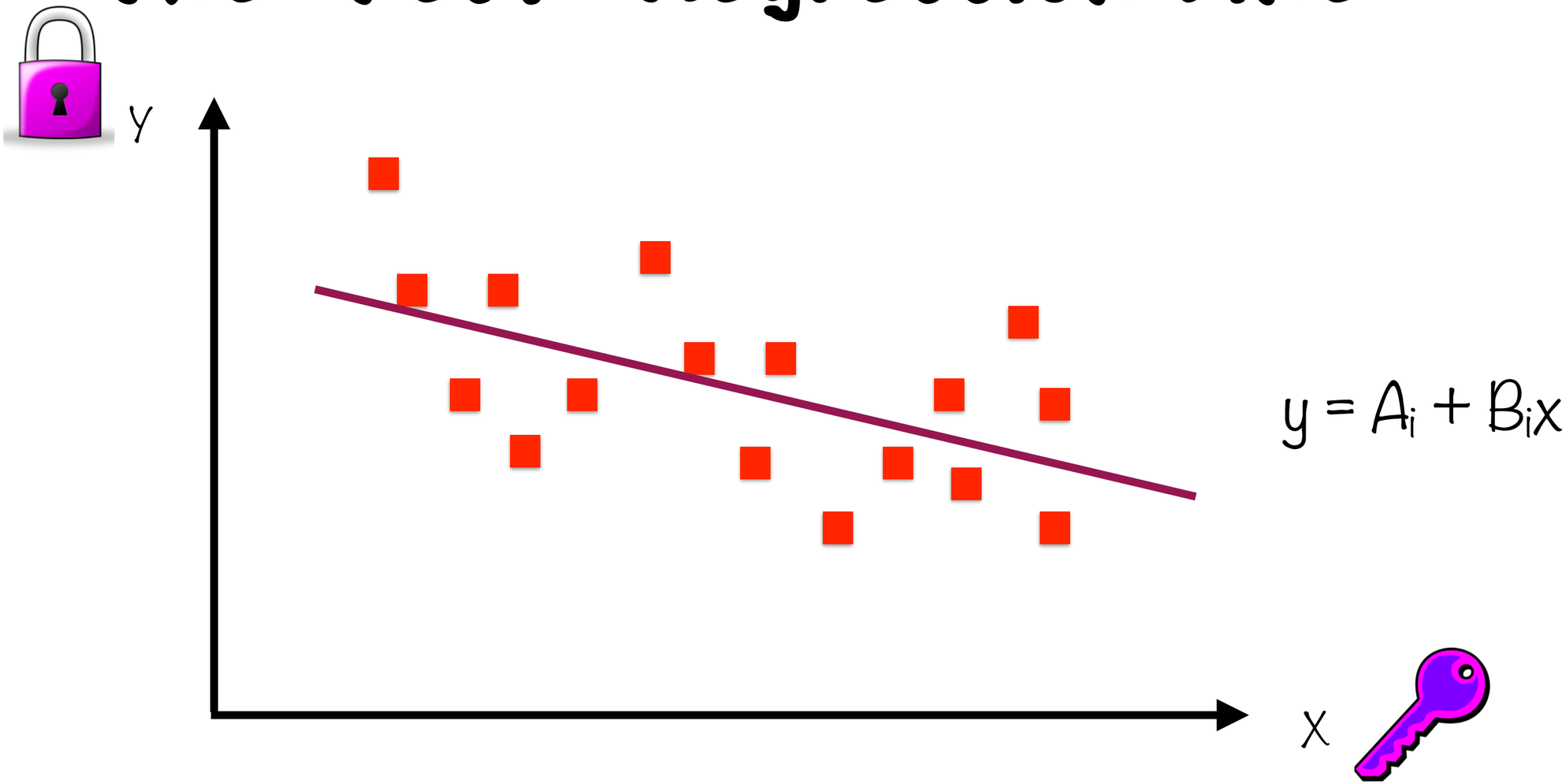
The "Best" Regression Line



$y = A_i + B_i x$

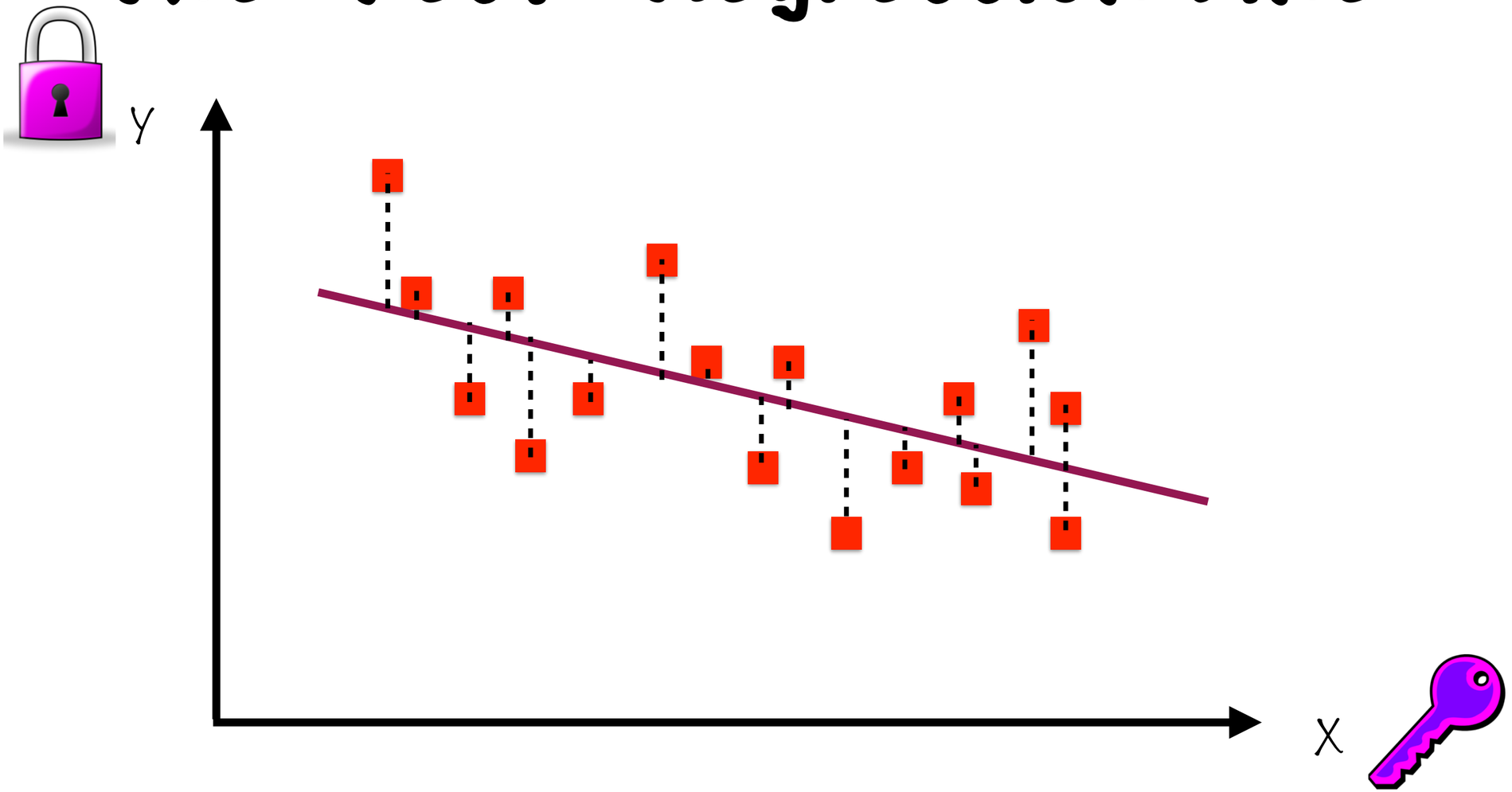
Adjust values of A and B by feeding back the error values

The "Best" Regression Line



Adjust values of A and B by feeding back the error values

The "Best" Regression Line



The "best fit" line is called the regression line

Regression is an example of a supervised learning algorithm

Placeholders

Linear Regression

y

Price per square foot

Regression Line: y = A + Bx

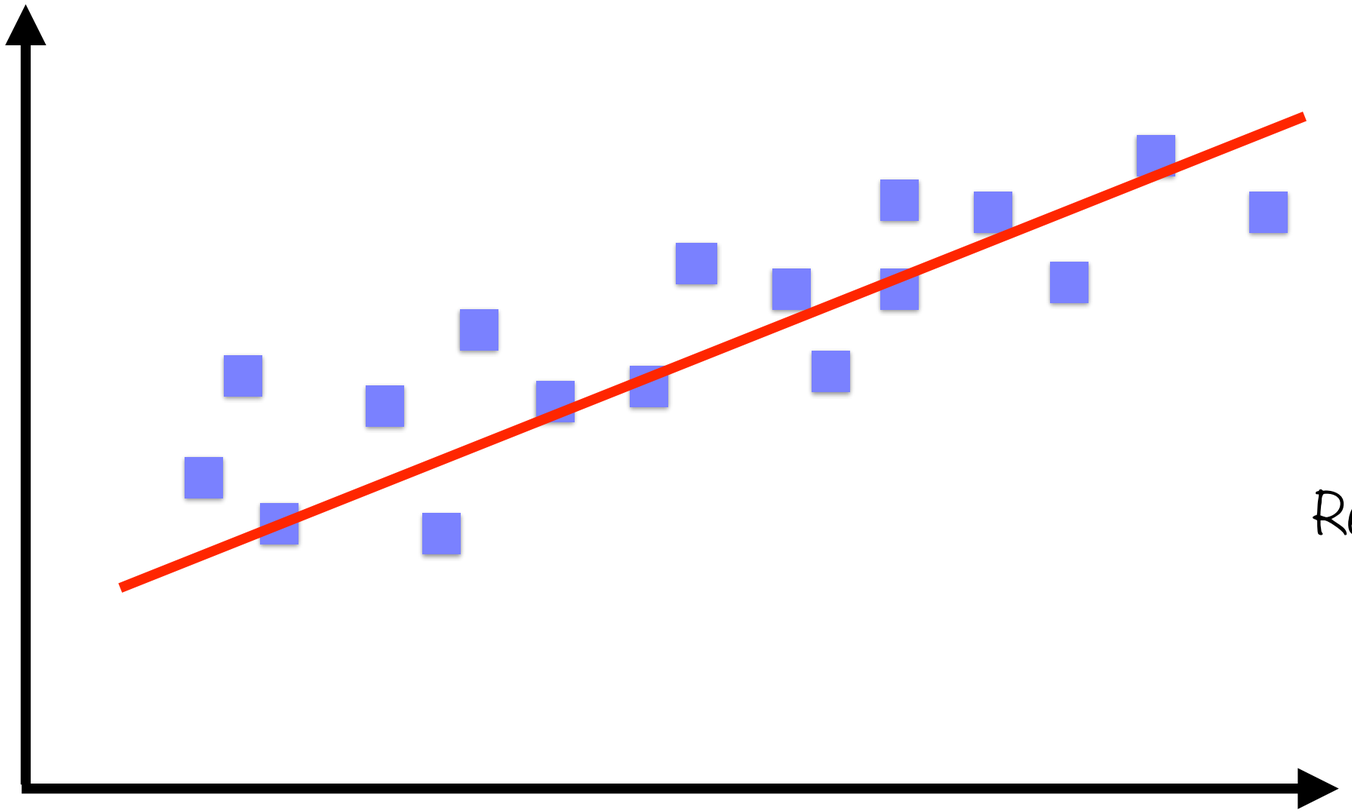
X

Distance from the city center

Linear Regression

y

Price of GOOG



Regression Line: y = A + Bx

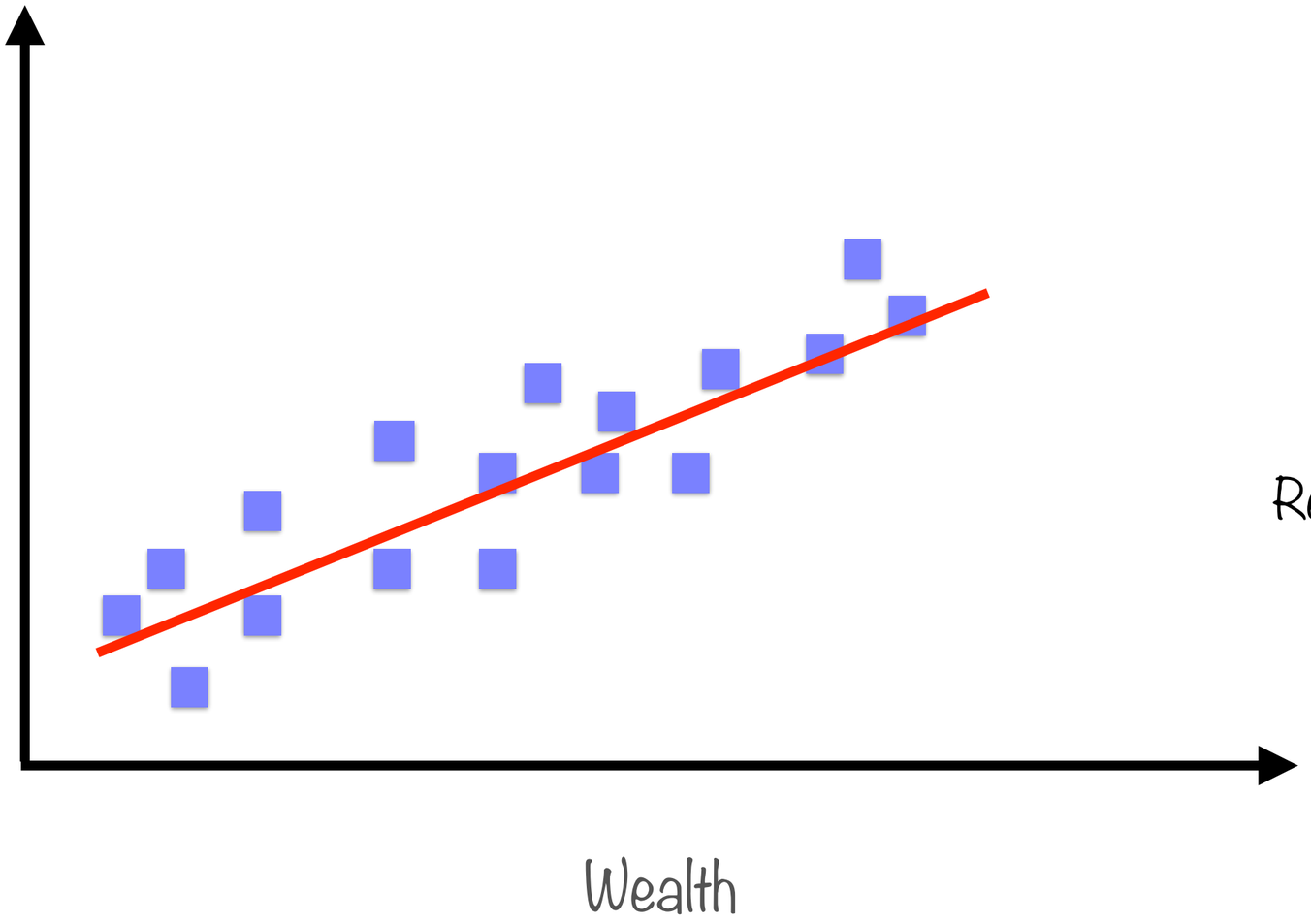
X

Dow Jones index

Linear Regression

y

Life expectancy



Regression Line: y = A + Bx

X

Wealth

Machine learning algorithms can be applied to a variety of problems

The model should have the ability to accept the different X and Y values

Placeholder

Hold the place for a Tensor that will be fed at runtime, in effect becoming an “input” node

Abrahams, Sam; Hafner, Danijar; Erwitt, Erik; Scarpinelli, Ariel (2016-07-23). TensorFlow For Machine Intelligence: A hands-on introduction to learning algorithms

Demo

Specify placeholders in our simple math operations

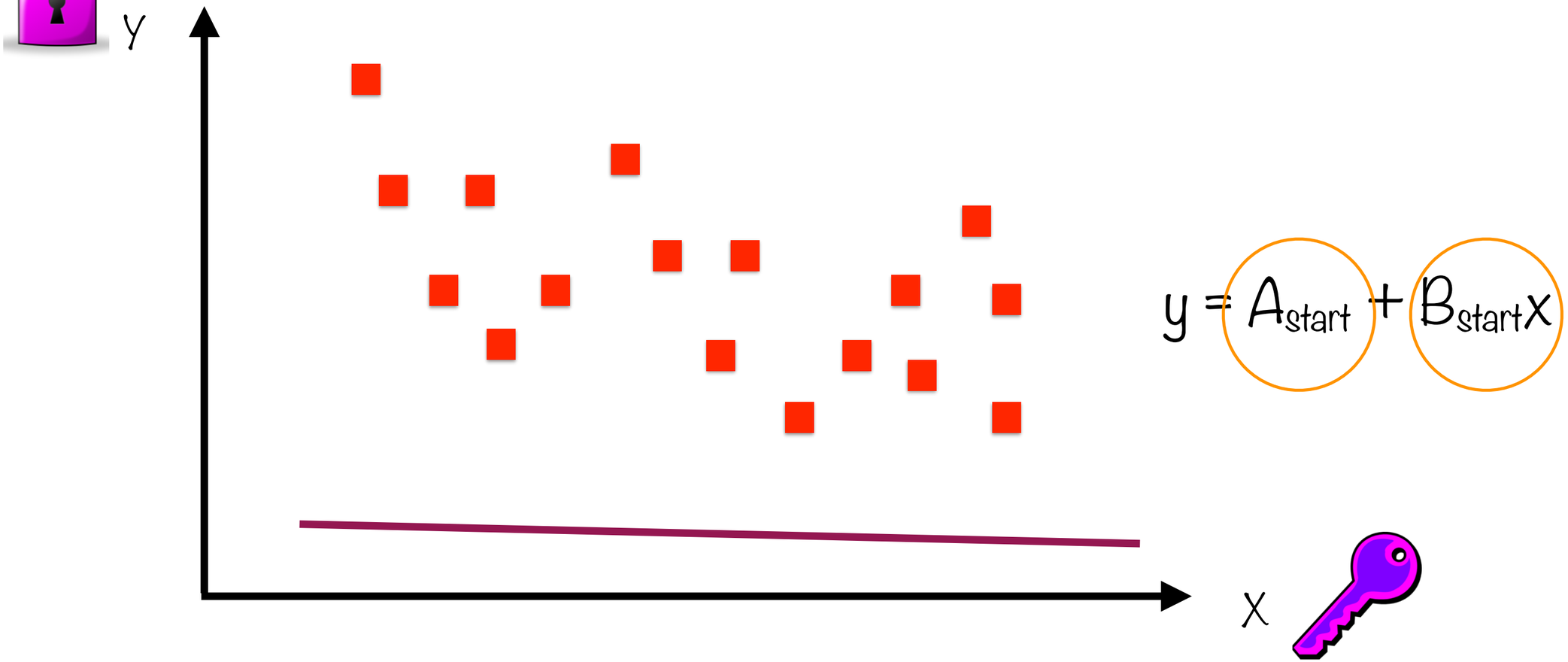
Use a feed dictionary to feed these into TensorFlow operations

Demo

Work with fetches and the feed_dict passed to Session.run() operations

Variables

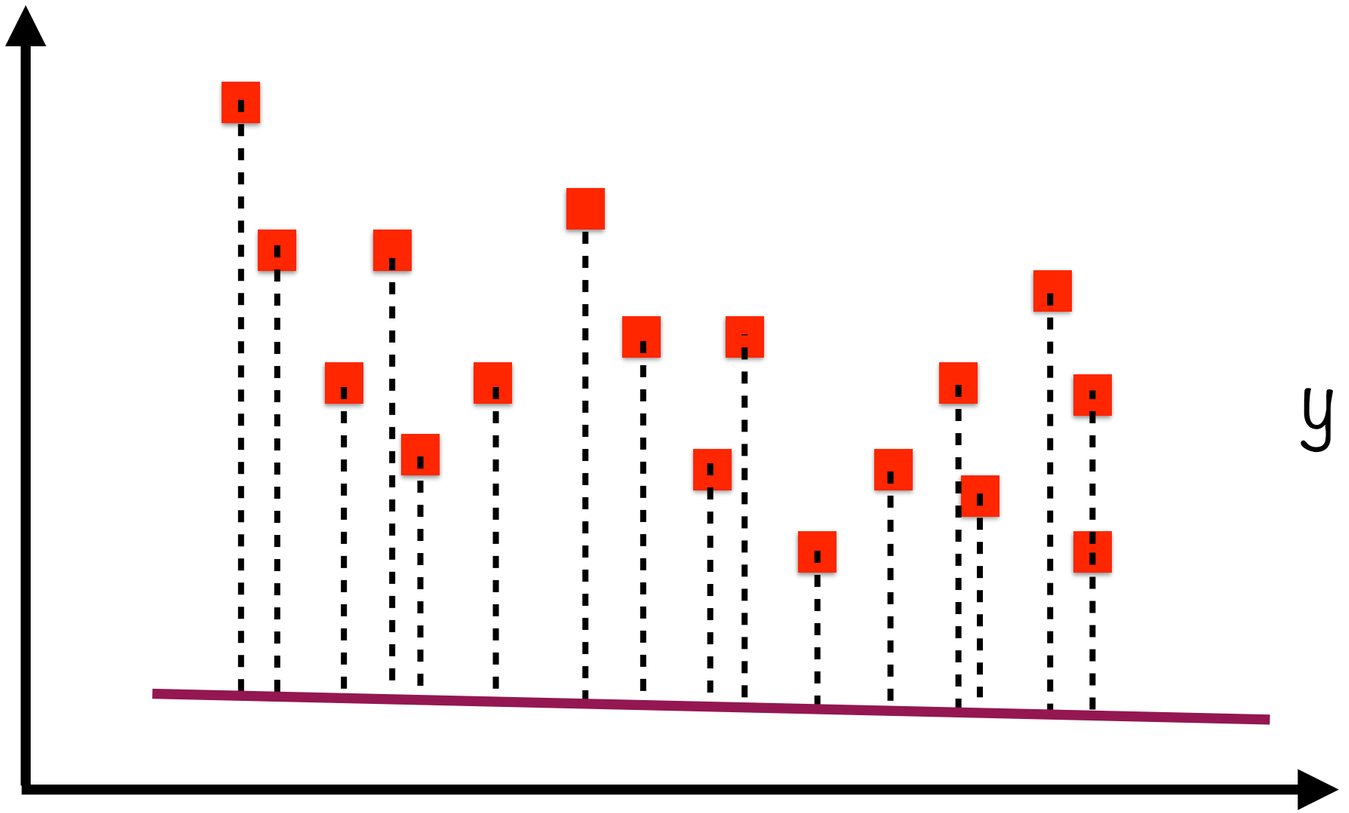
The "Best" Regression Line



$$y = A_{start} + B_{start}x$$

Start off with some values for A and B

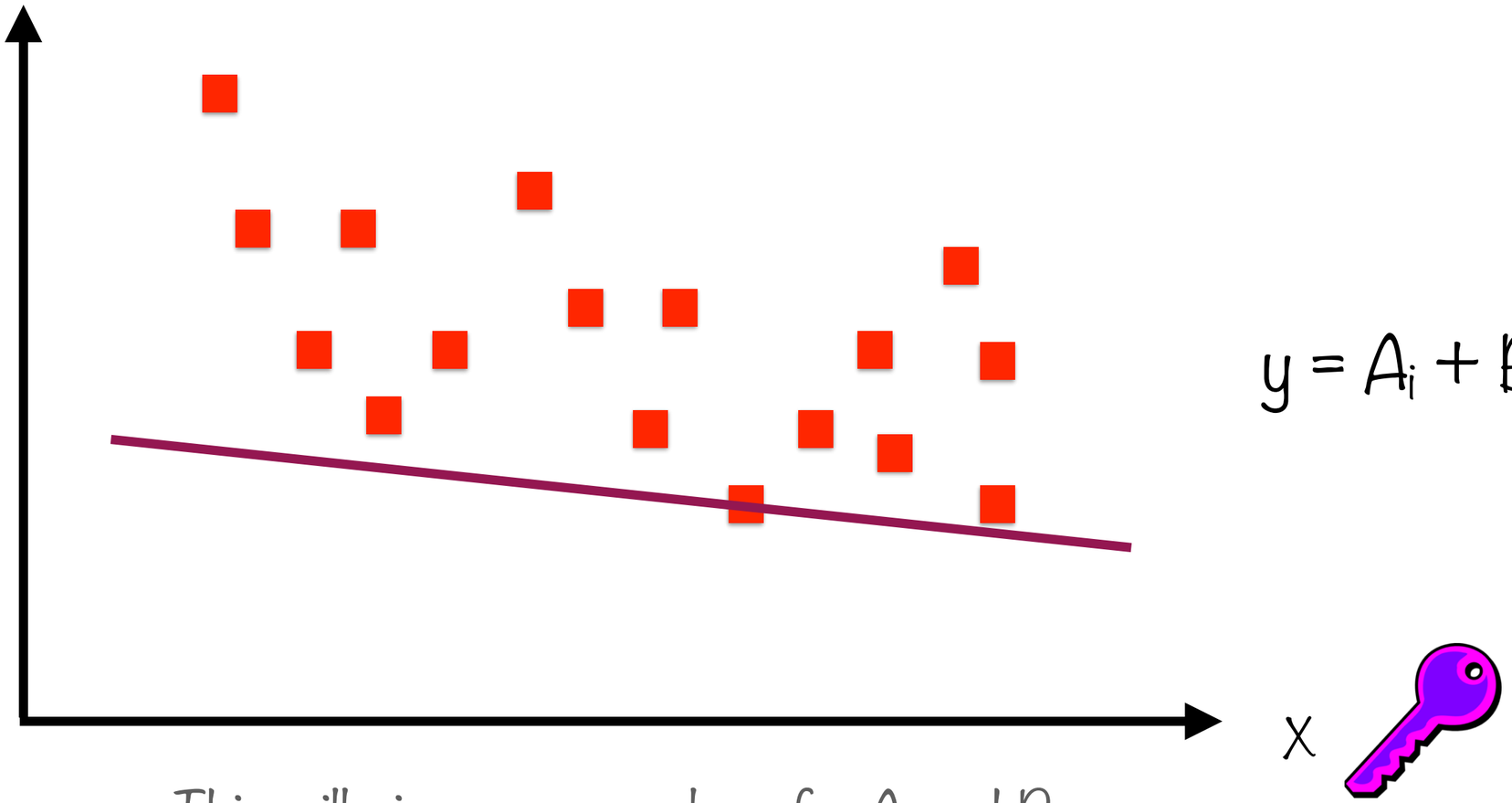
The "Best" Regression Line



$$y = A_{start} + B_{start}x$$

Calculate the least square error and feed that back

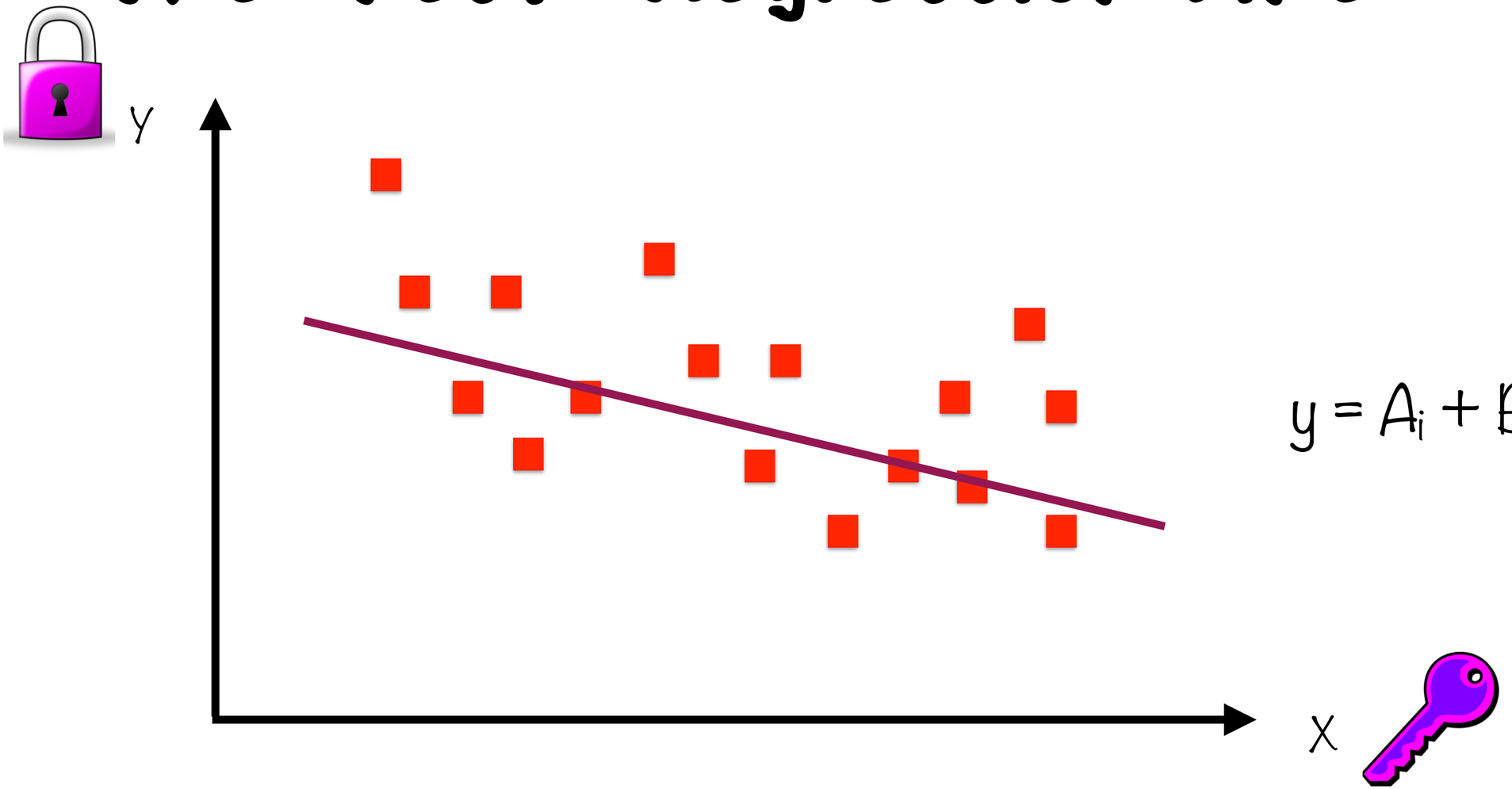
The "Best" Regression Line



$$y = A_i + B_i x$$

This will give us new values for A and B

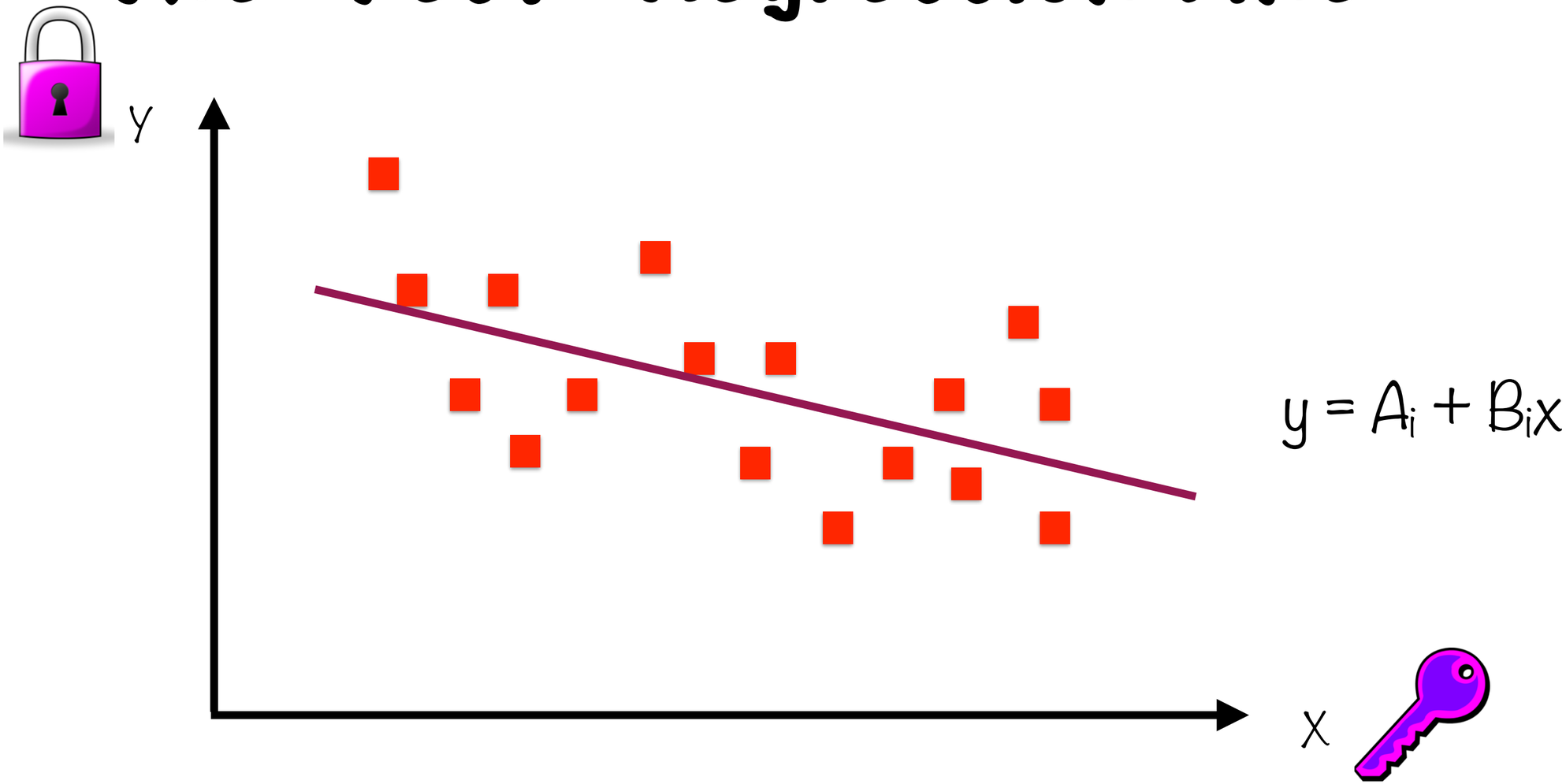
The "Best" Regression Line



$y = A_i + B_i x$

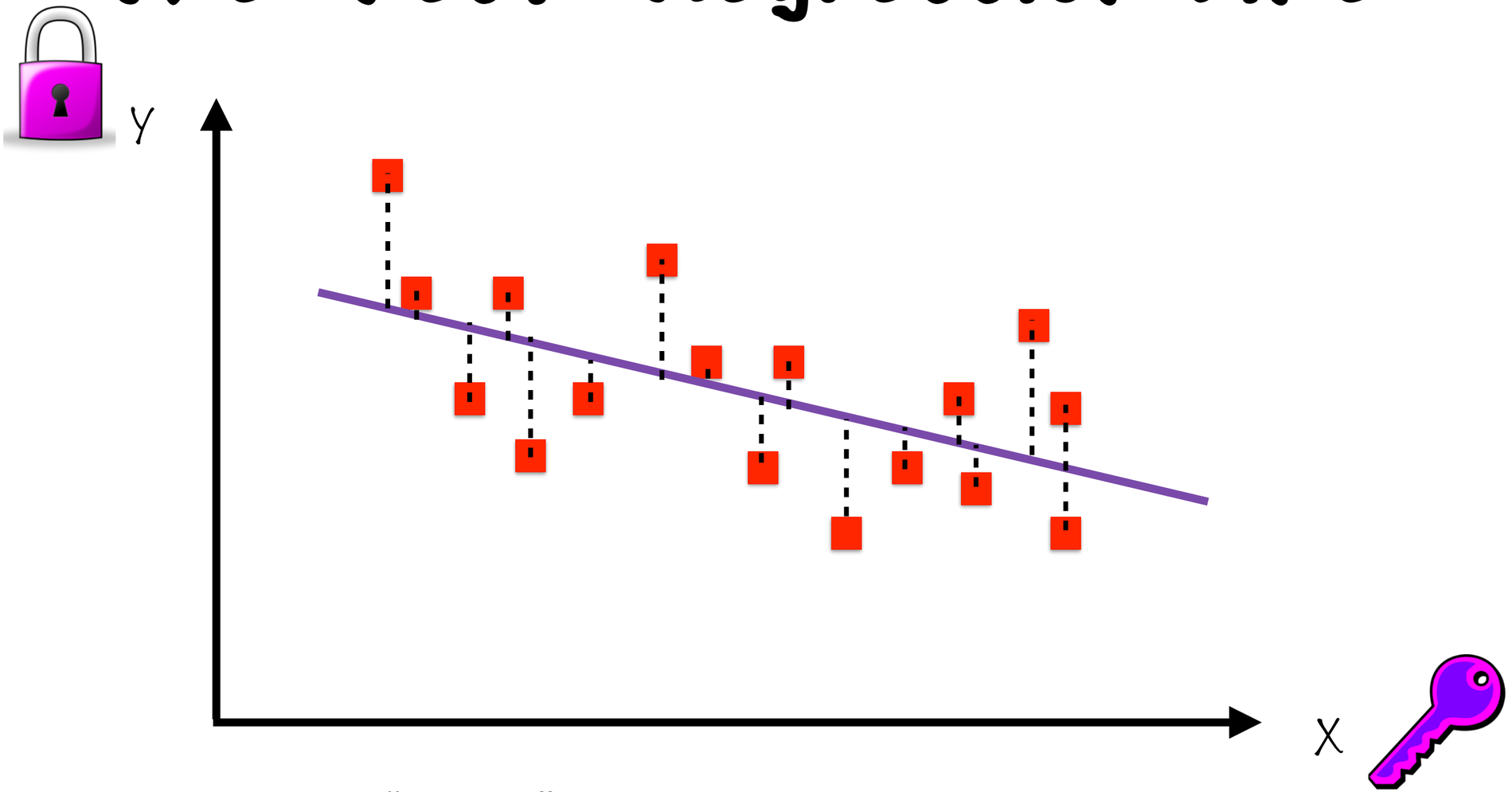
Adjust values of A and B by feeding back the error values

The "Best" Regression Line



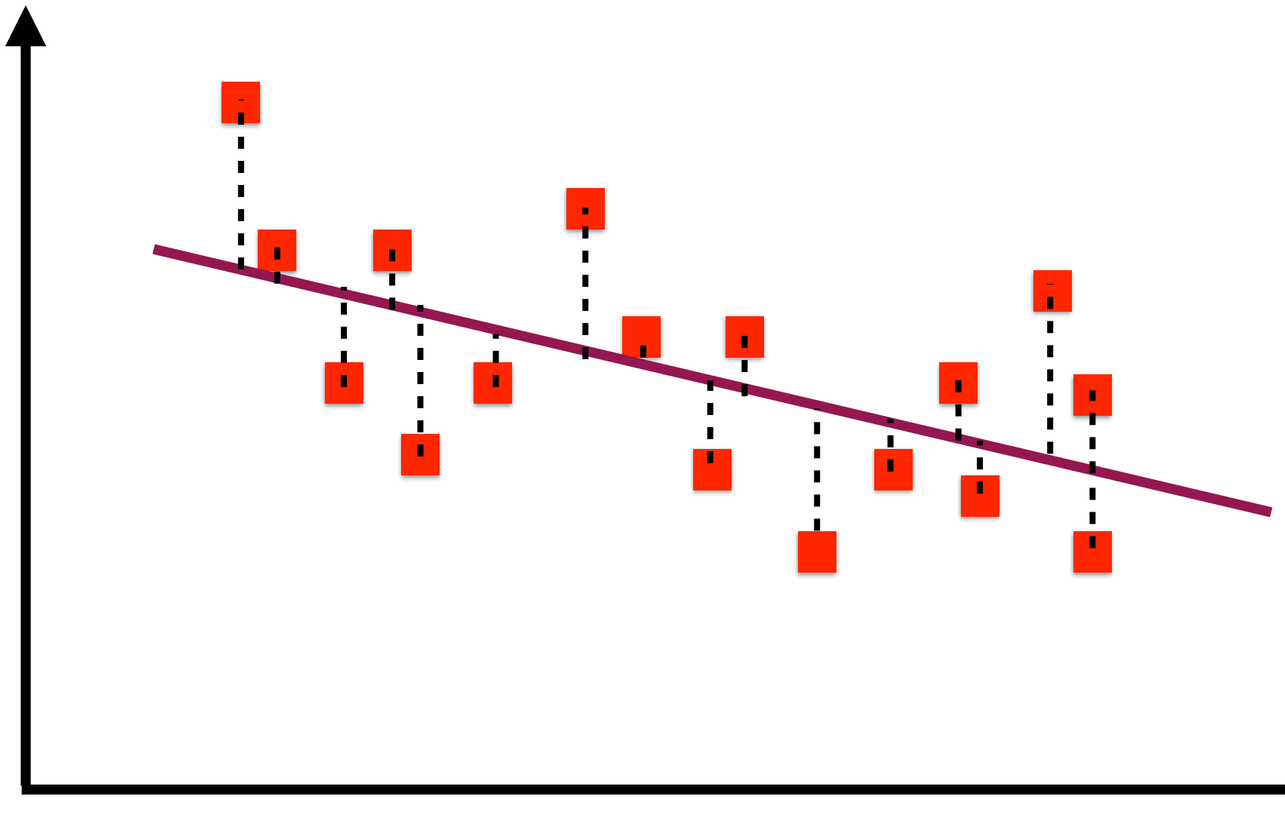
Adjust values of A and B by feeding back the error values

The "Best" Regression Line



The "best fit" line is called the regression line

The "Best" Regression Line



Regression Line: y = A + Bx

The values of A and B are updated till we get the best fit line

Machine learning algorithms iterate to get closer to the solution

The model should have the ability to hold constantly changing values

Constants, Placeholders and Variables

Constants

Immutable values which do not change

Placeholders

Assigned once and do not change after

Variables

Are constantly recomputed

Variables

Mutable Tensor values that persist across multiple calls to Session.run()

Abrahams, Sam; Hafner, Danijar; Erwitt, Erik; Scarpinelli, Ariel (2016-07-23). TensorFlow For Machine Intelligence: A hands-on introduction to learning algorithms

Summary

Implement placeholders and variables in TensorFlow

Make TensorBoards more useful using named scopes

Working with Images

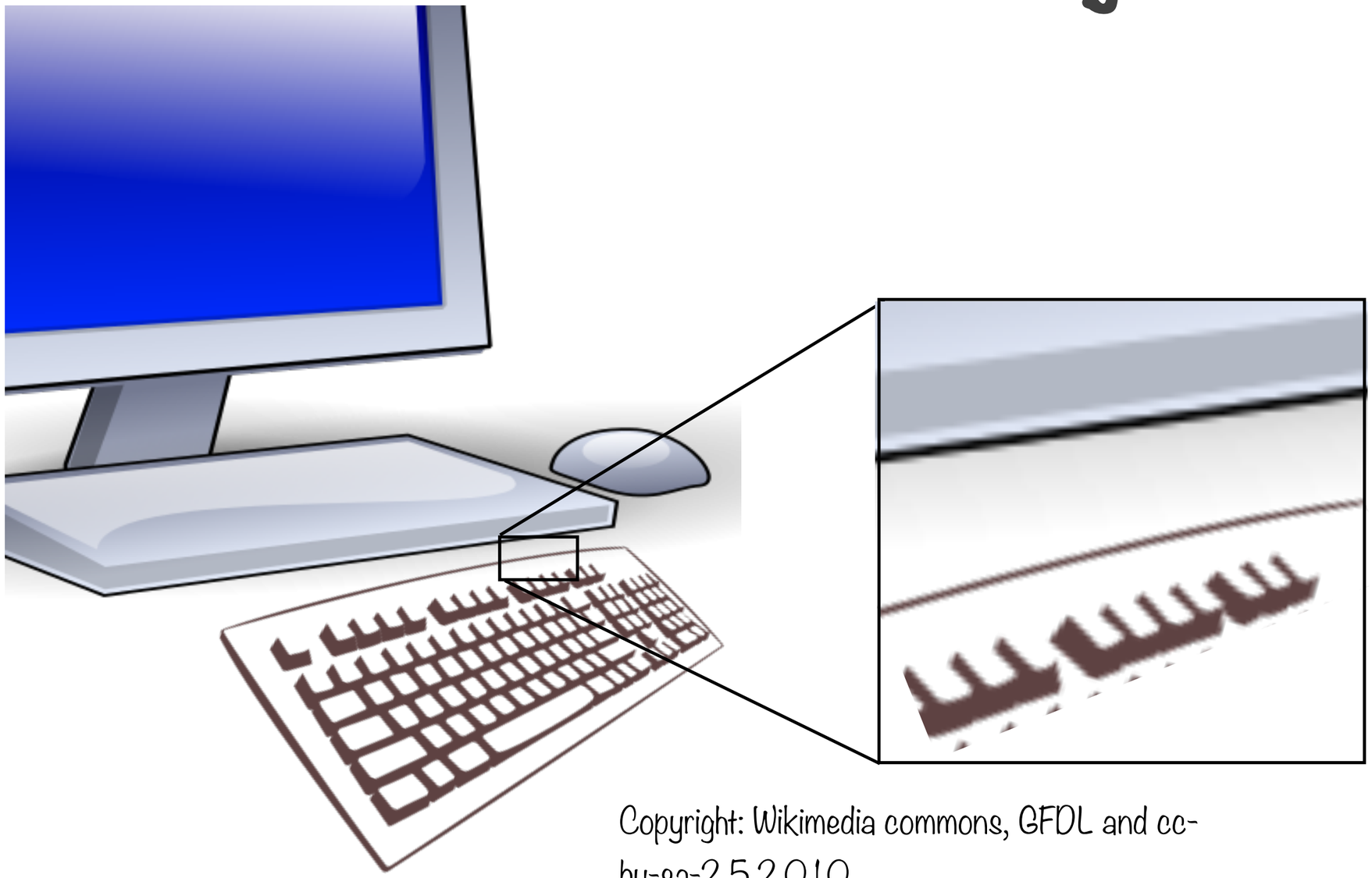
Overview

Representing color and grayscale images as Tensors

Implementing image operations such as transpose, resize, cropping

Image Recognition

Pixels in Images



Pixels in Images

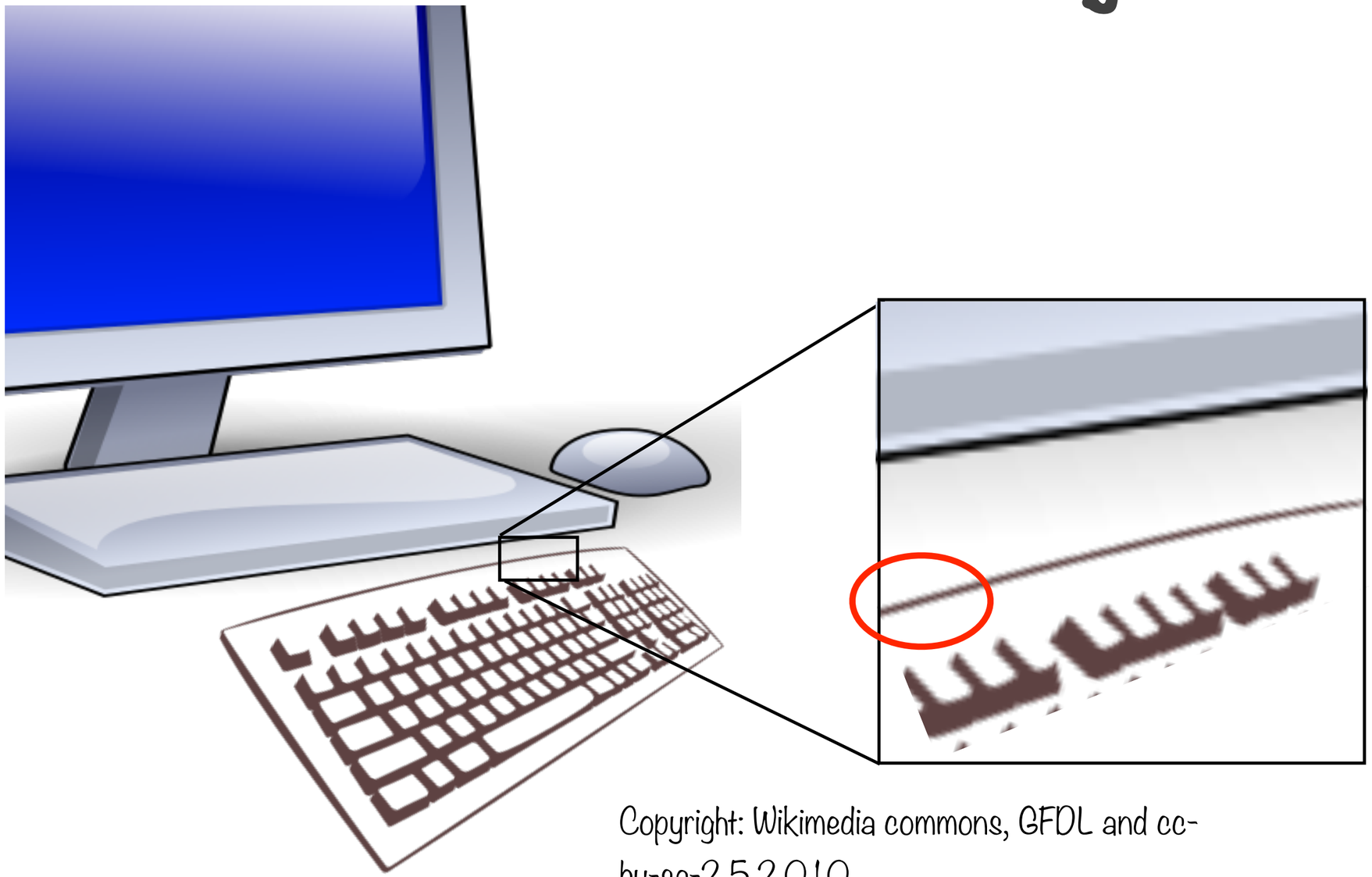


Image Recognition

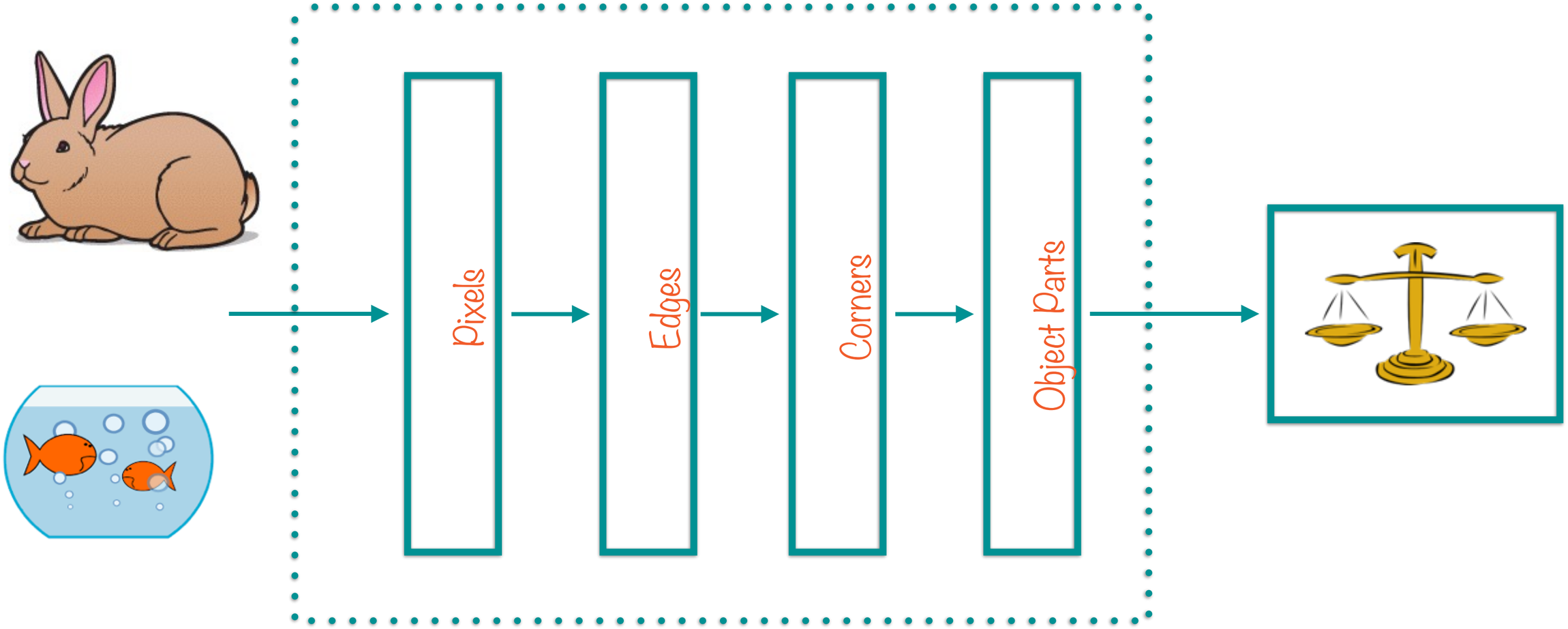
Images represented as pixels

Identify edges, colors, shapes

A photo of a horse

Neural networks, specifically convolutional neural networks (CNNs) work well for hard image recognition tasks

Image Recognition Using Neural Networks

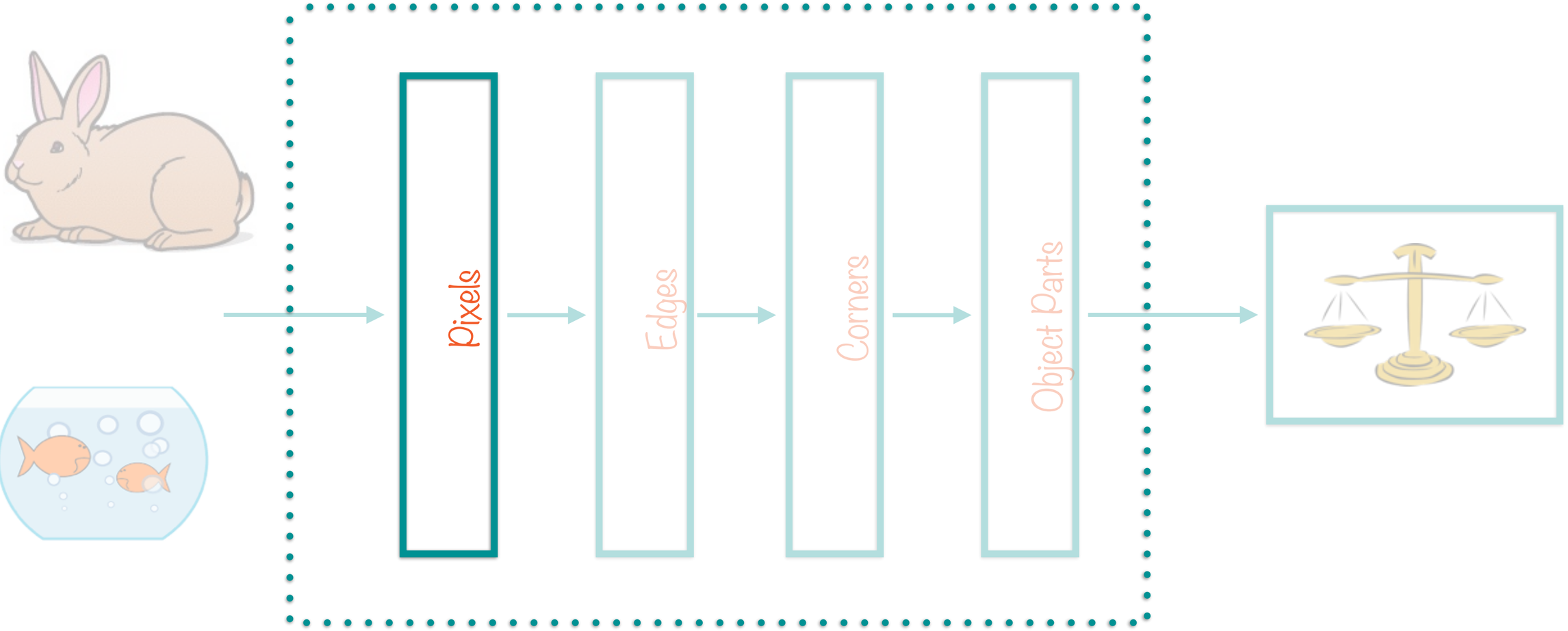


Corpus of Images

Feature Selection & Classification Algorithm

ML-based Classifier

Image Recognition Using Neural Networks

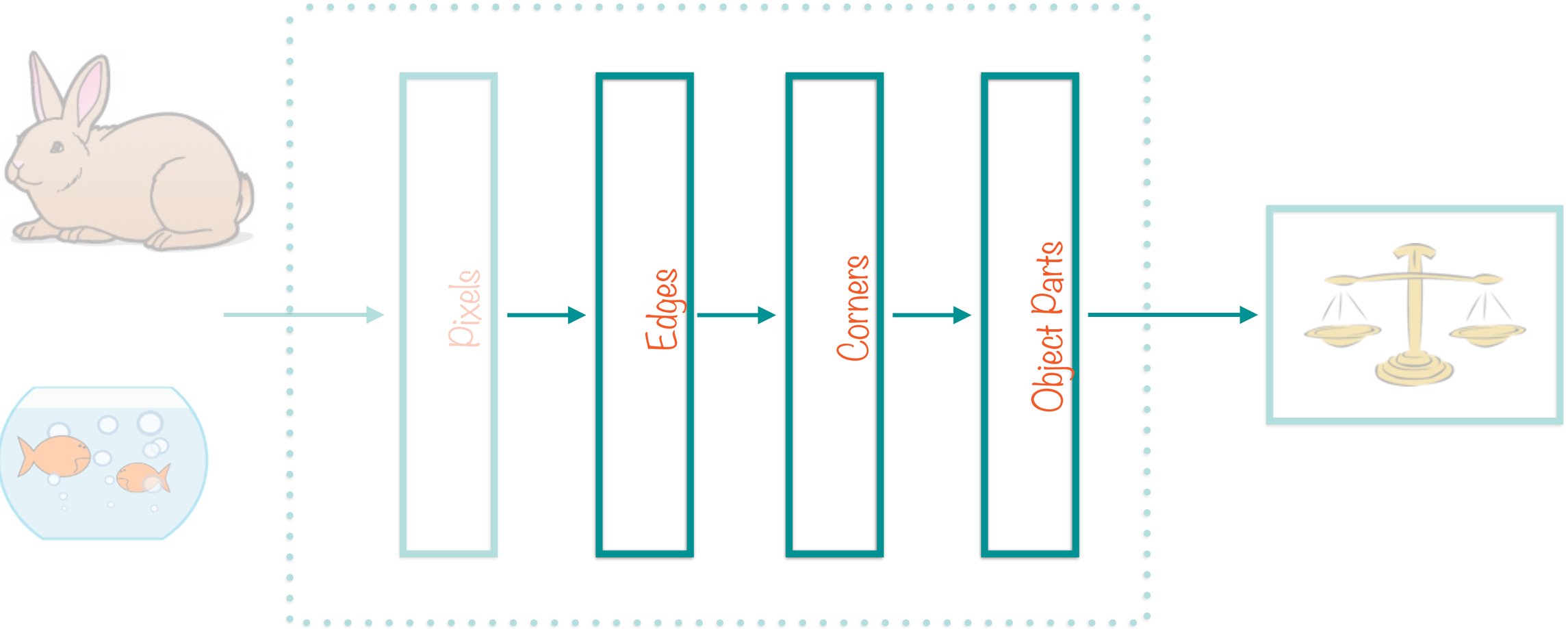


Corpus of Images

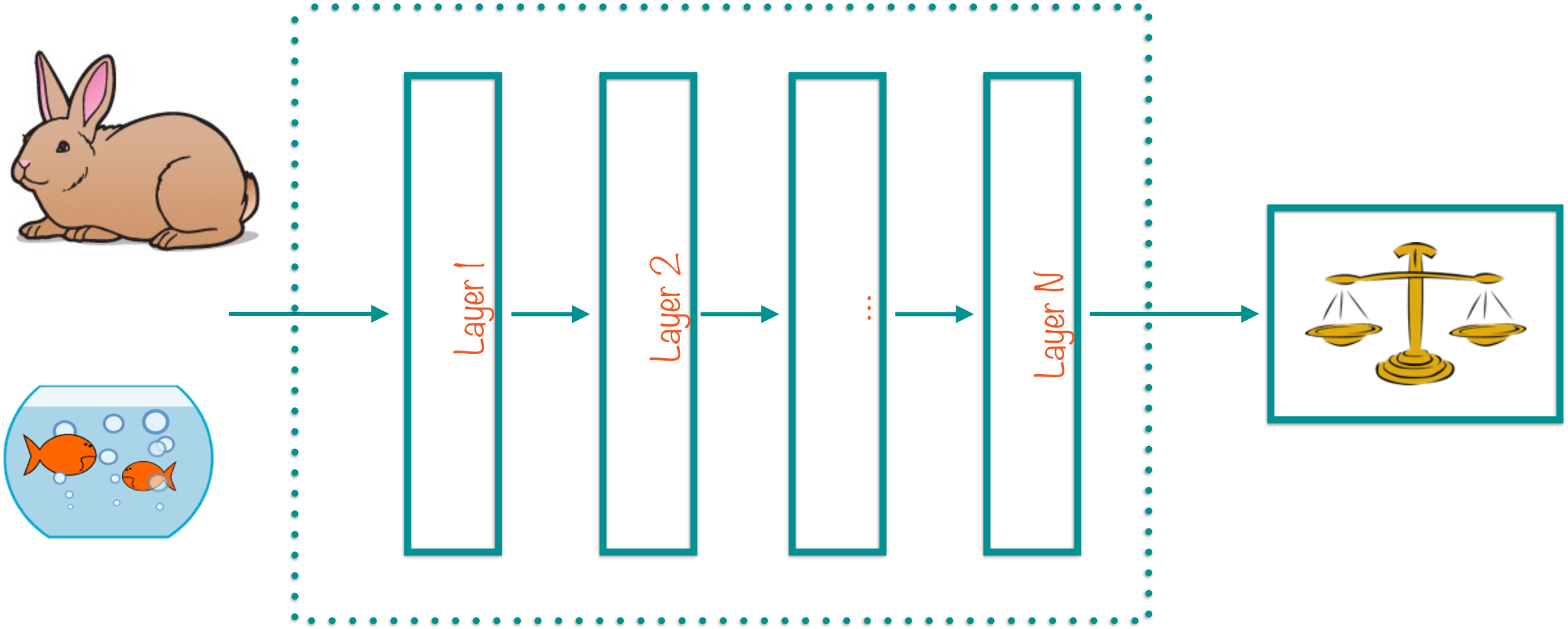
"Visible layer"

ML-based Classifier

Image Recognition Using Neural Networks



Neural Networks Introduced

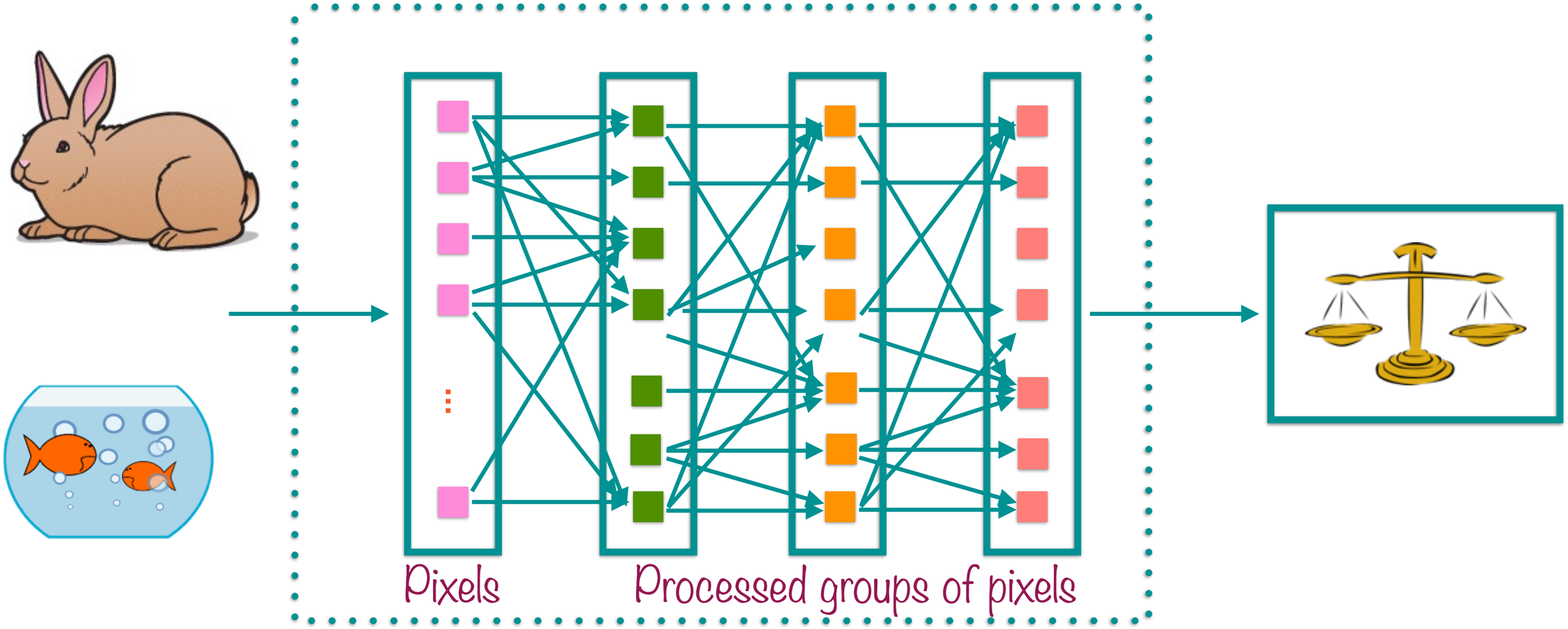


Corpus of Images

Layers in a neural network

ML-based Classifier

Neural Networks Introduced



Corpus of Images

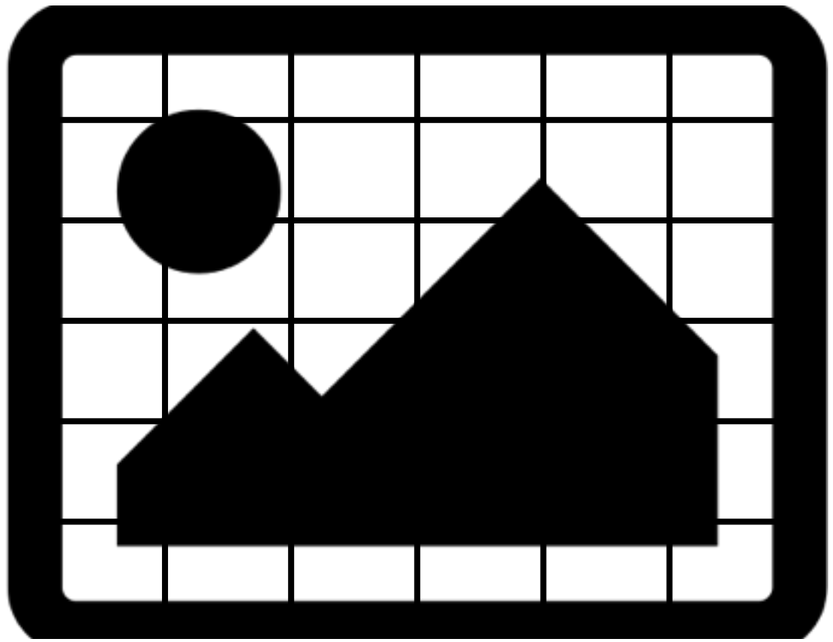
Each layer consists of individual interconnected neurons

ML-based Classifier

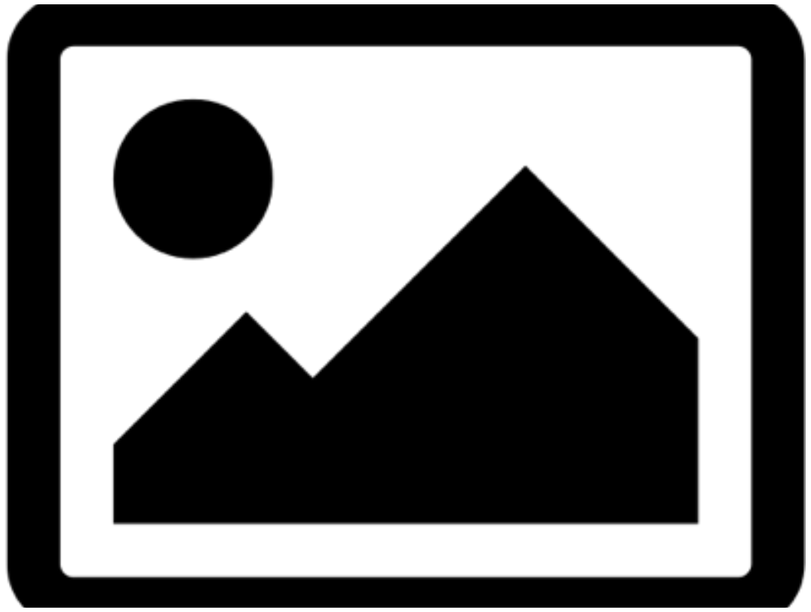
TensorFlow is optimized at building neural network solutions for image recognition

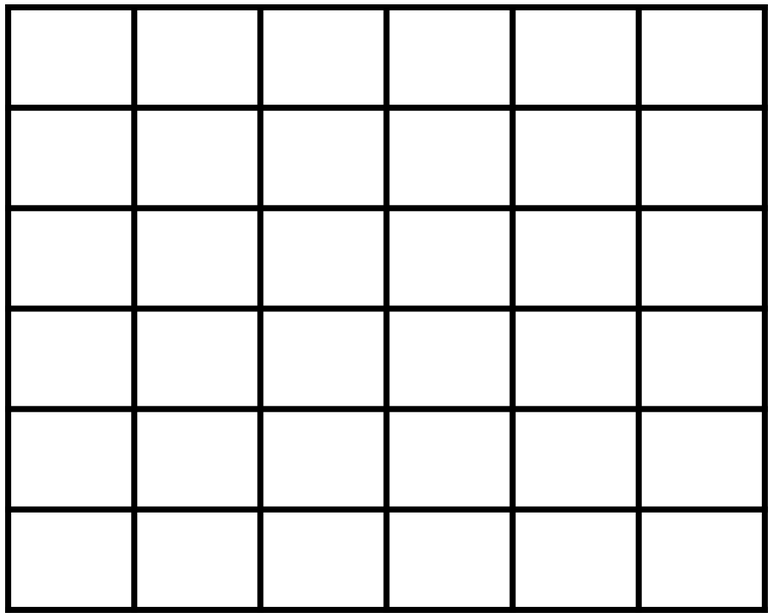
Representing Images as 3-D Tensors

Images as Tensors



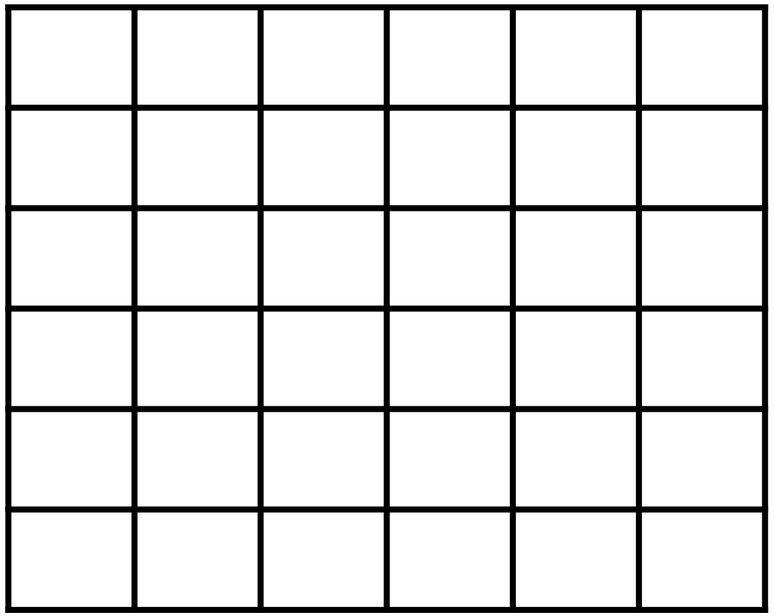
Images as Tensors





Each pixel holds a value based on the type of image

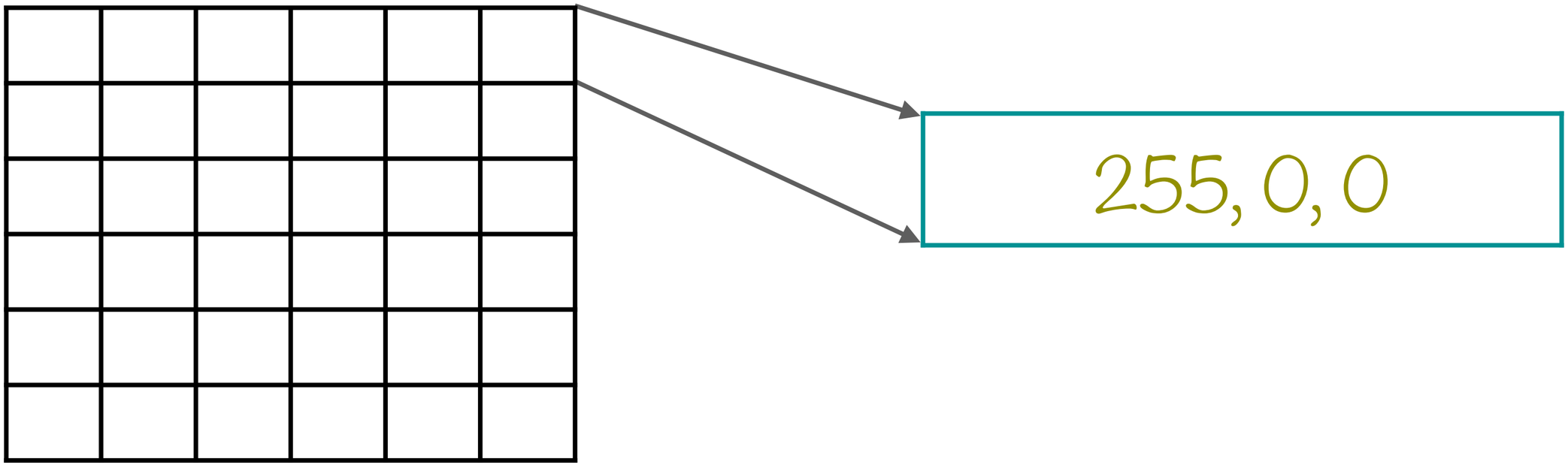
RGB Images



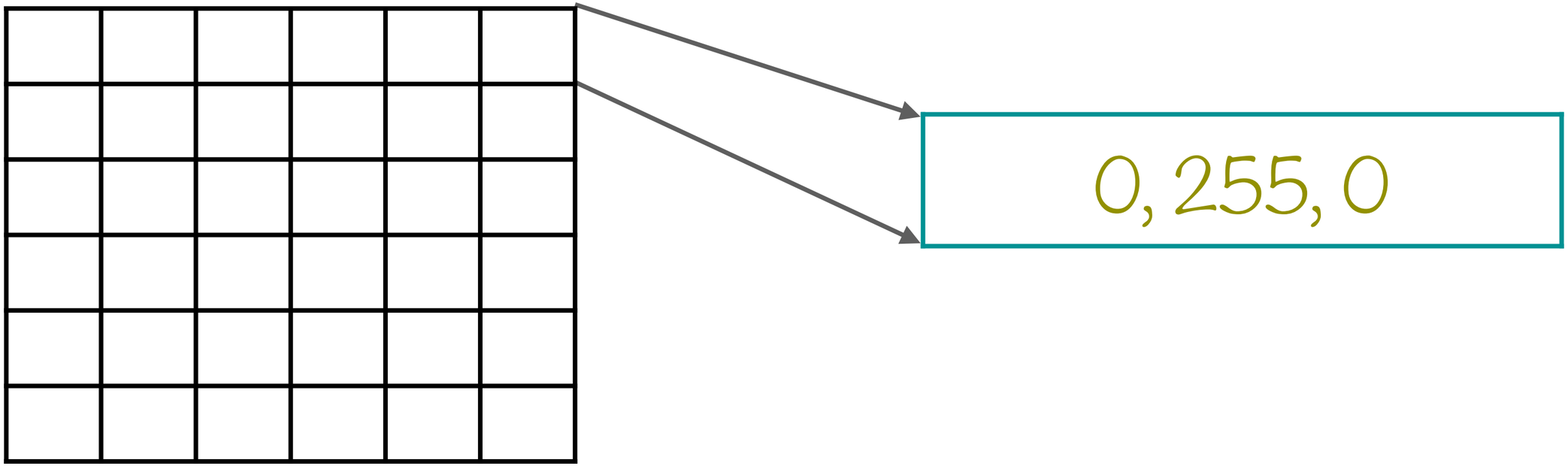
RGB values are for color images

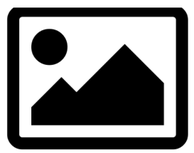
R, G, B: 0-255

RGB Images

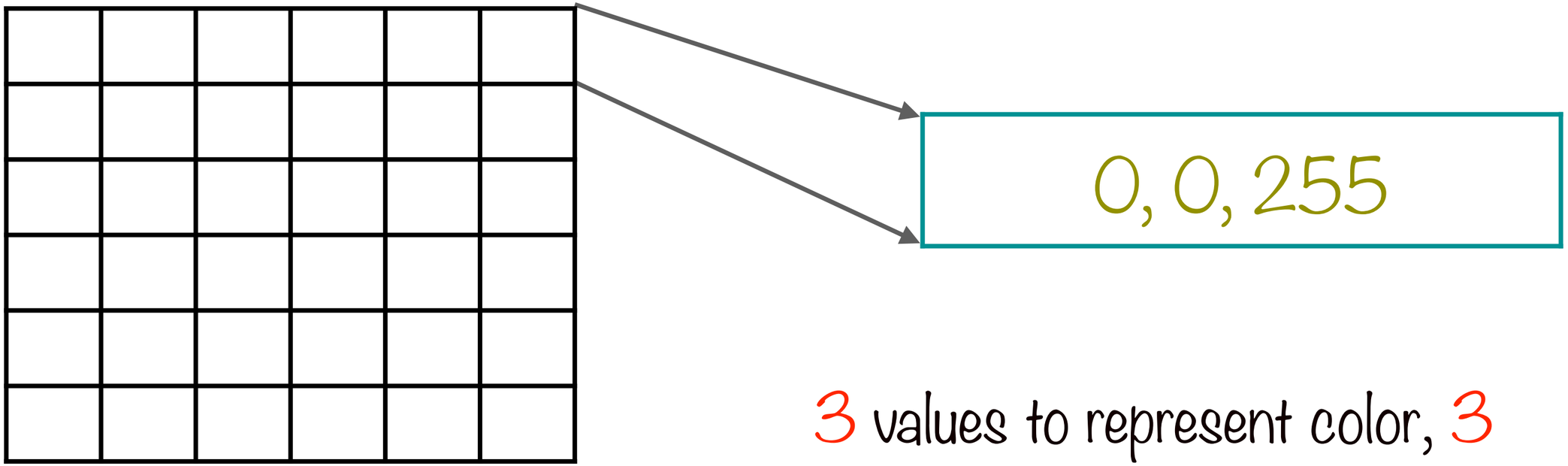


RGB Images



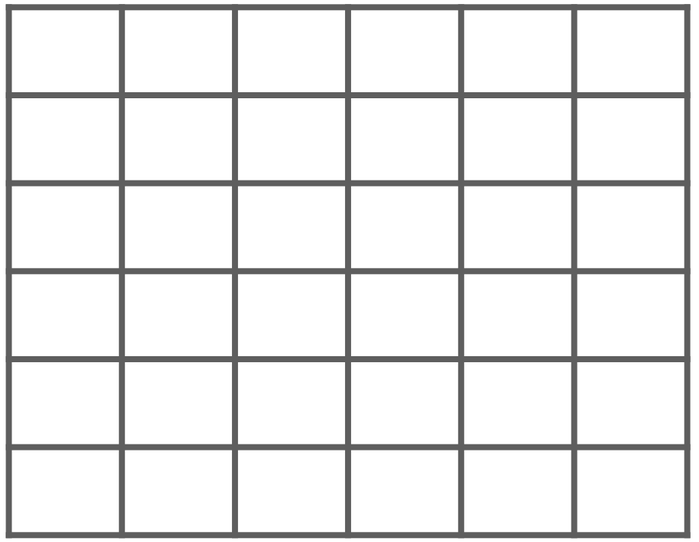


RGB Images

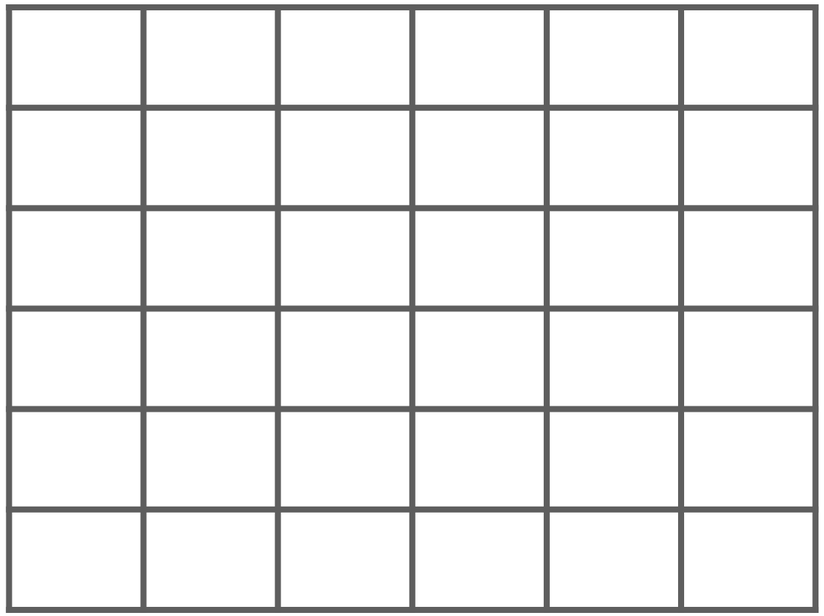


3 values to represent color, 3 channels

Grayscale Images



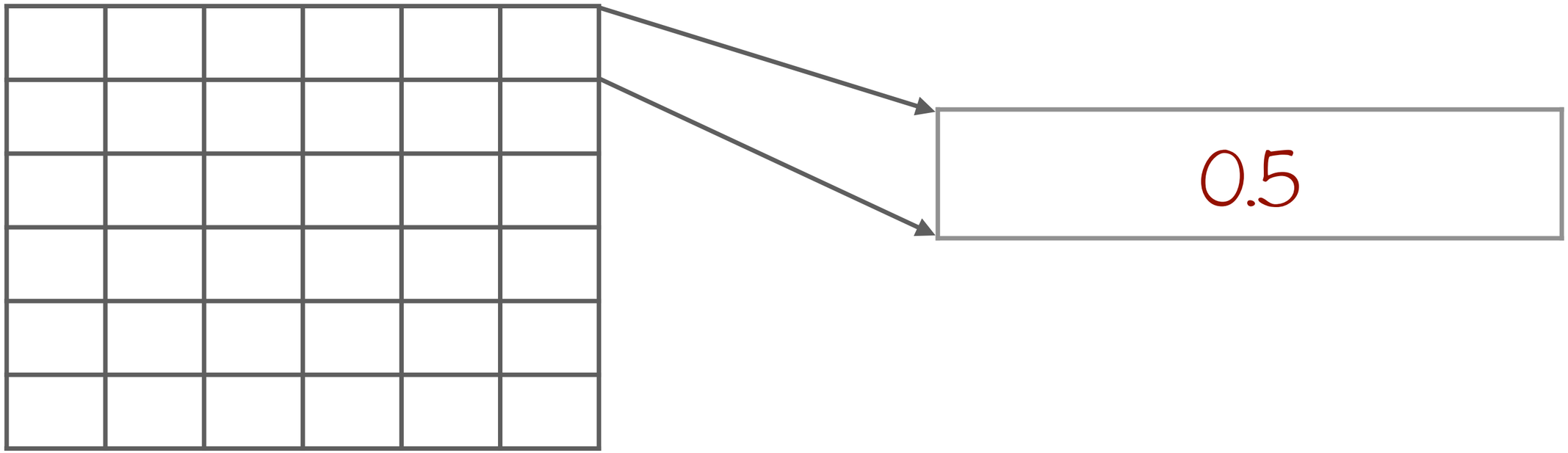
Grayscale Images



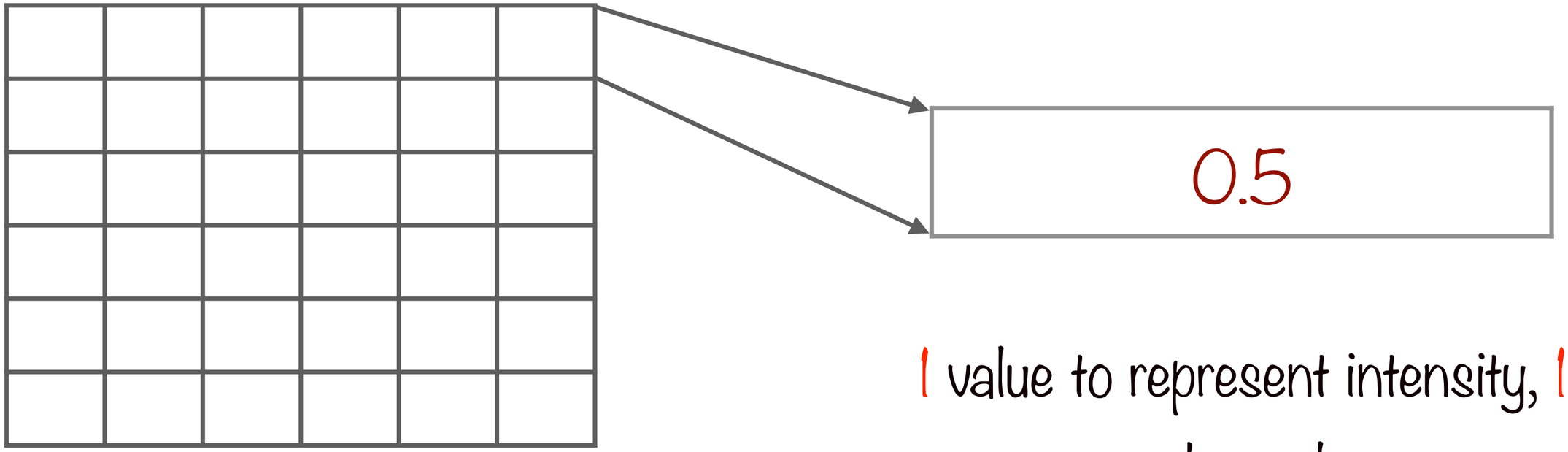
Each pixel represents only intensity information

0.0 - 1.0

Grayscale Images

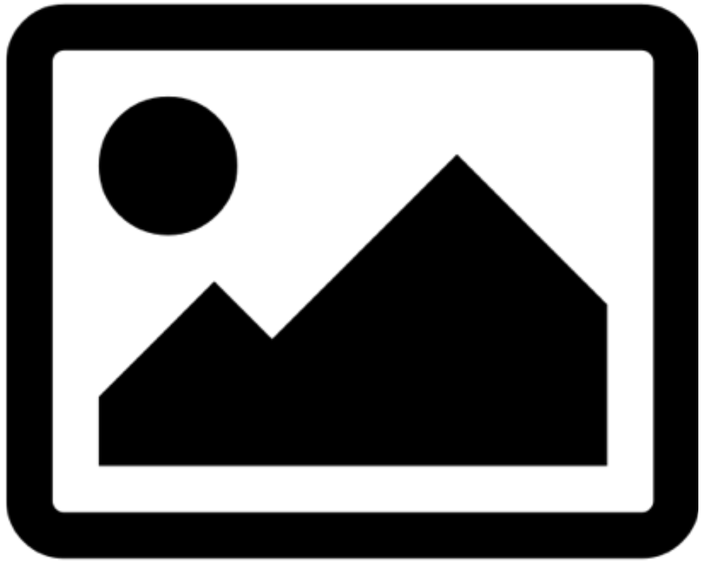


Grayscale Images



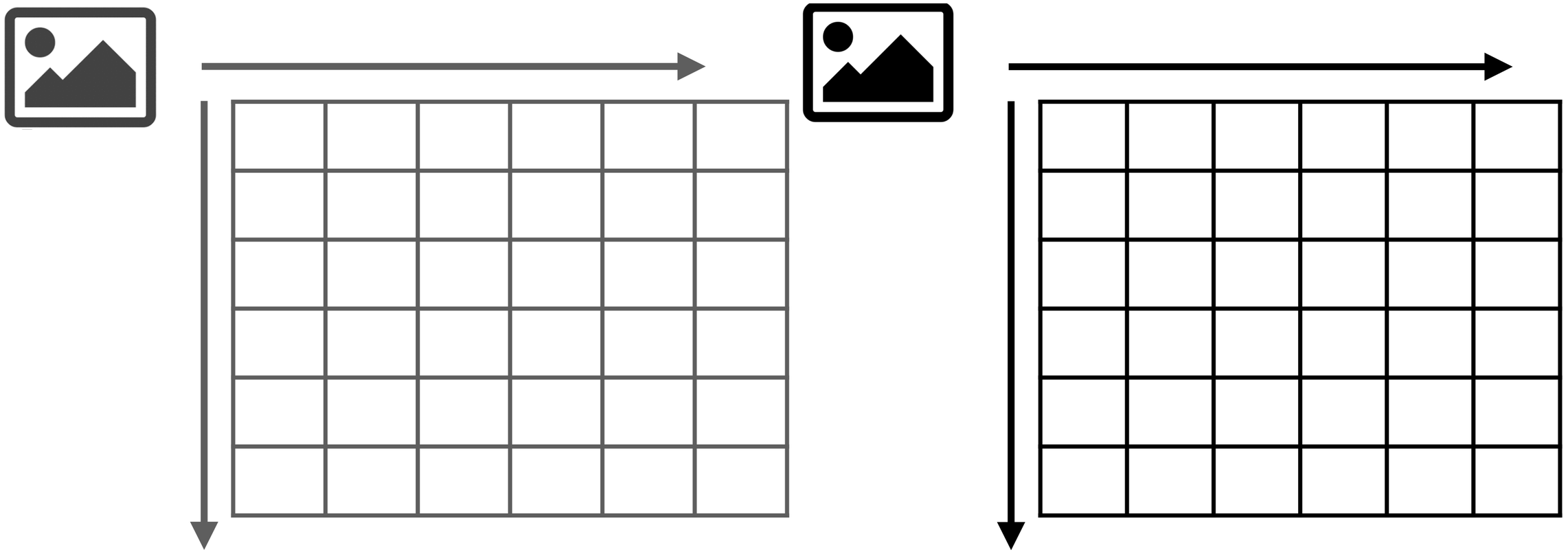
1 value to represent intensity, 1 channel

Images as Tensors



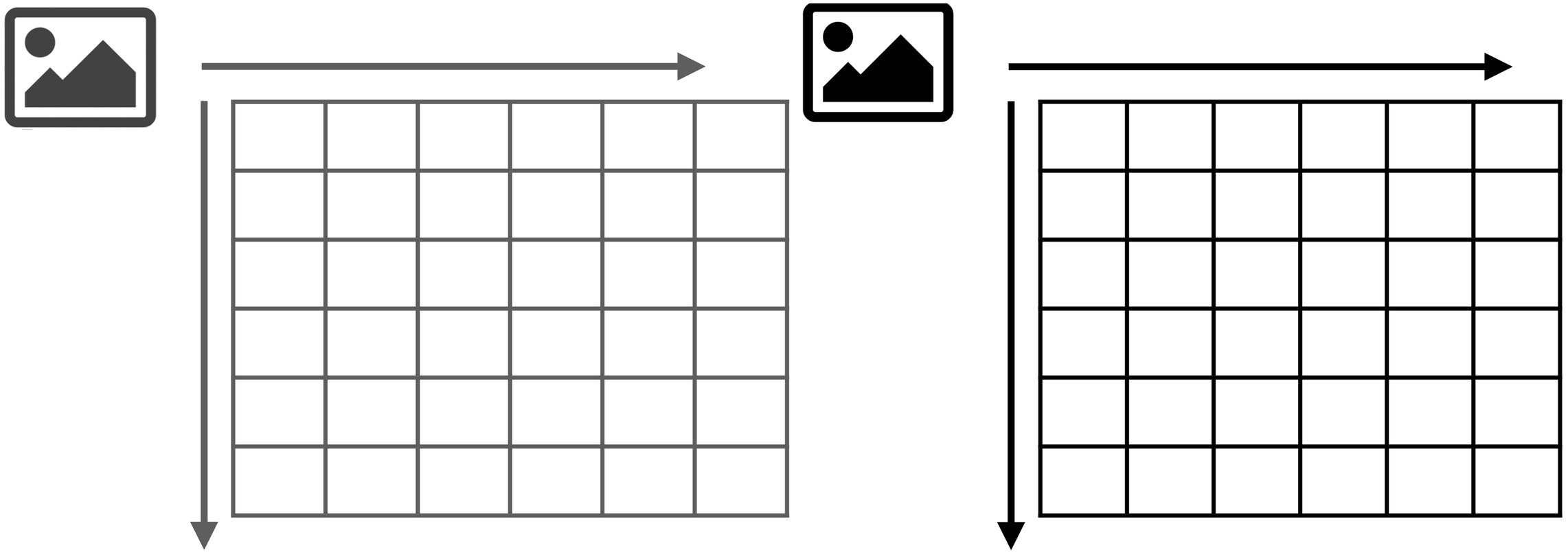
Single channel and multi-channel images

Images as Tensors



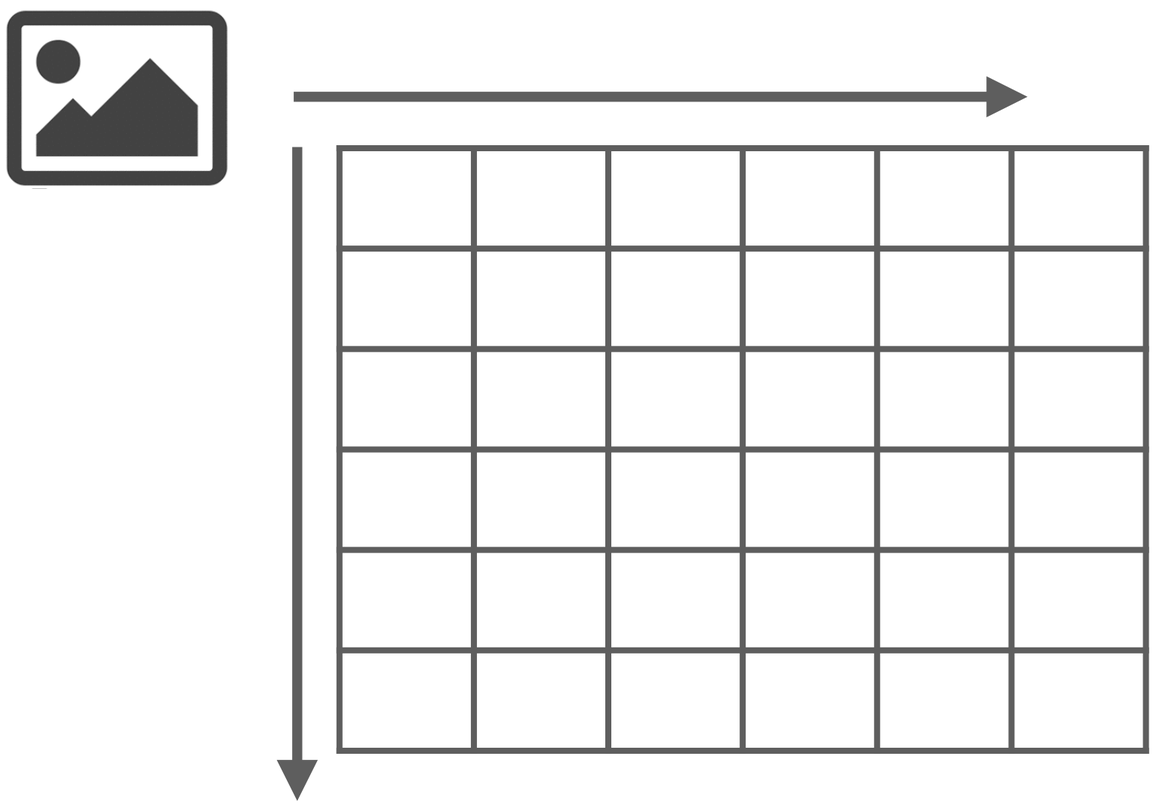
Images can be represented by a 3-D matrix

Images as Tensors

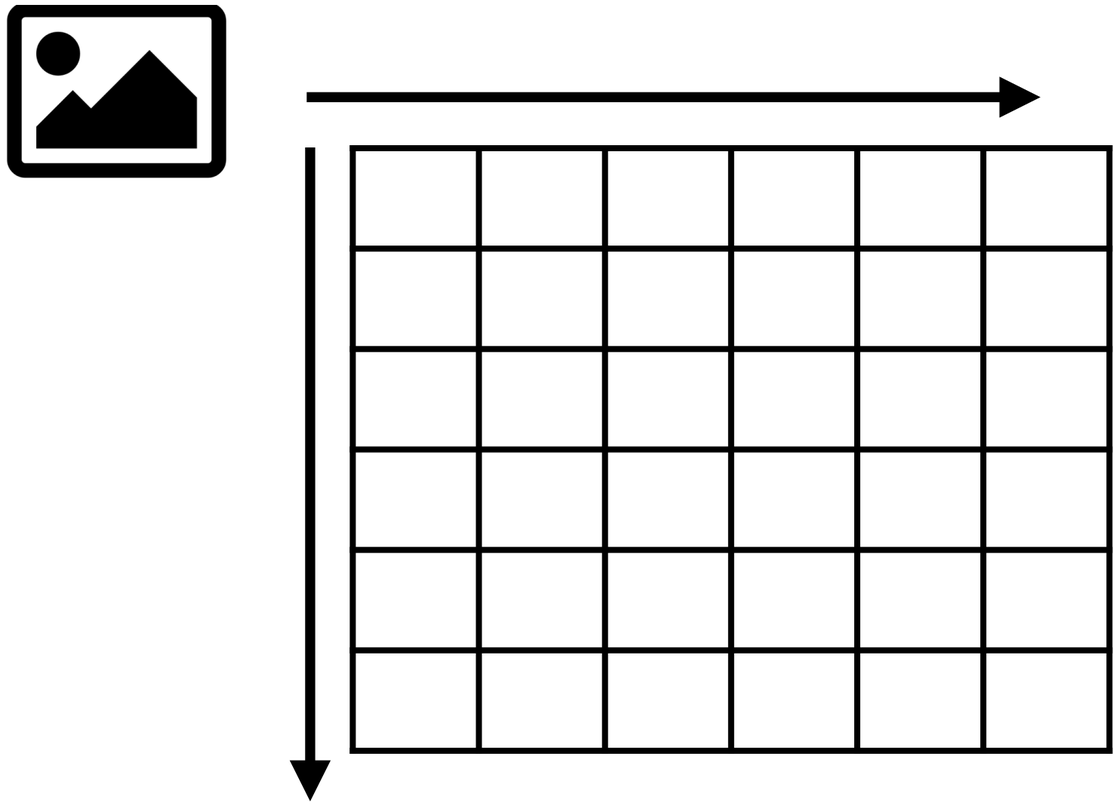


The number of channels specifies the number of elements in the 3rd dimension

Images as Tensors



(6, 6, 1)



(6, 6, 3)

Demo

Read in an image using matplotlib and then transpose it using TensorFlow

Demo

Read in a list of images in TensorFlow using a queue and coordinators

Resize images to be of the same dimensions

Show image summaries in TensorBoard

List of Images as 4-D Tensors

List of Images

TensorFlow usually deals with a list of images in one 4-D Tensor

List of Images

The images should all be the same size

List of Images

(10, 6, 6, 3)

The number of channels

List of Images

(10, 6, 6, 3)

The height and width of each image in the list

List of Images

(10, 6, 6, 3)

The number of images

Summary

Understood image representation of color and grayscale images as Tensors

Learnt image transformations such as resize, flip and crop

Worked with multiple images in TensorFlow

Using K-nearest-neighbors for Digit Recognition

Overview

Introduce the MNIST handwritten digit dataset

Understand the K-nearest-neighbors machine learning algorithm

Implement K-nearest-neighbors in TensorFlow to identify handwritten digits from 0 to 9

The MNIST Handwritten Digits Dataset

MNIST Dataset

Handwritten digits database

Large quantity of handwritten digits commonly used for training image processing systems

MNIST Dataset

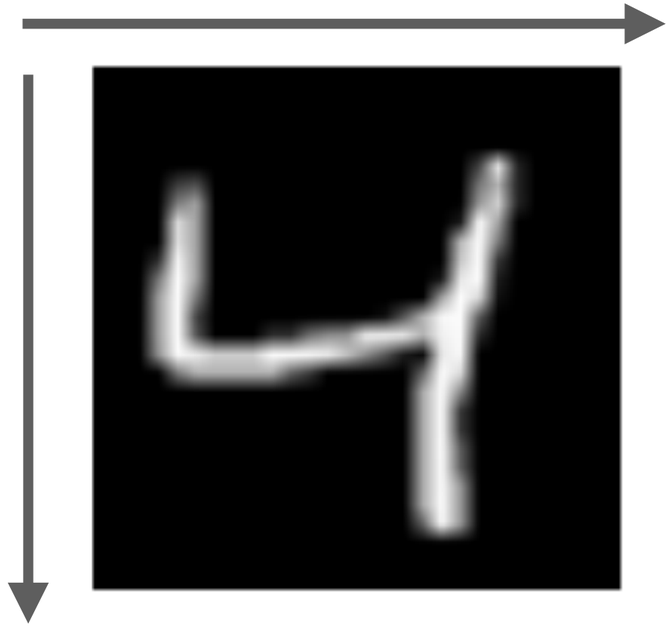
Handwritten digits database

Modified National Institute of Standards and Technology

MNIST Dataset

Each digit is in grayscale

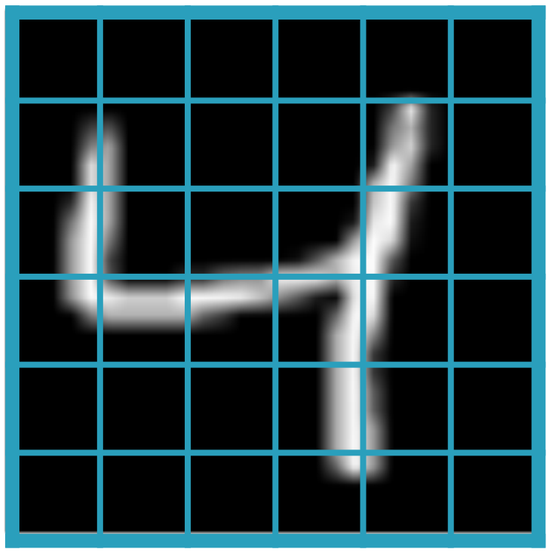
MNIST Dataset



Every image is standardized
to be of size 28x28

= 784 pixels

MNIST Dataset



Every pixel holds a single value for intensity

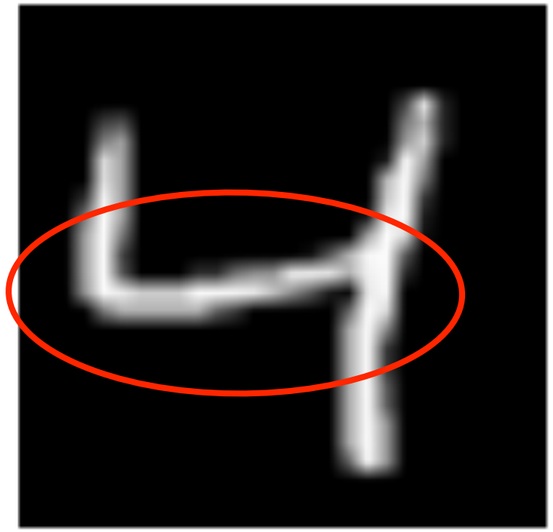
MNIST Dataset

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

MNIST Dataset

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

MNIST Dataset



0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

MNIST Dataset

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

MNIST Dataset

Every image has an associated label

MNIST Dataset

5 0 4 1

MNIST for machine learning is the equivalent of the "Hello World" for programming

The K-nearest-neighbors Algorithm

Types of ML Algorithms

Supervised

Labels associated with the training data is used to correct the algorithm

Unsupervised

The model has to be set up right to learn structure in the data

Supervised Learning

Input variable x and output variable y

Learn the mapping function y = f(x)

Approximate the mapping function so for new values of x we can predict y

Use existing dataset to correct our mapping function approximation

Unsupervised Learning

Only have input data x no output data

Model the underlying structure to learn more about data

Algorithms self discover the patterns and structure in the data

Training Data

Breathes like a mammal

Gives birth like a mammal

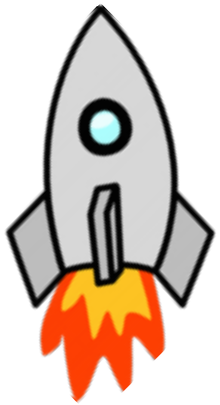
ML-based Classifier

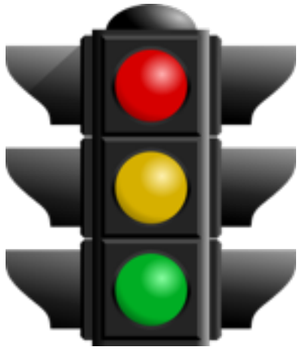
Mammal

Corpus

KNN is an supervised learning algorithm which uses training data to find what is most similar to the current sample

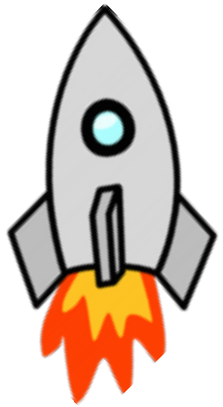
K-nearest-neighbors





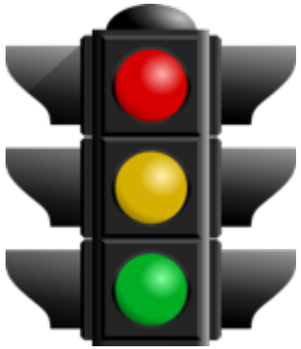
Uses the entire training dataset as a model

K-nearest-neighbors



Rocket

Buildings



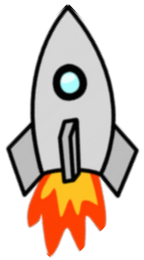
Signal

Pig

Shop

Each element in training data has a label

K-nearest-neighbors



Rocket

Buildings

Signal

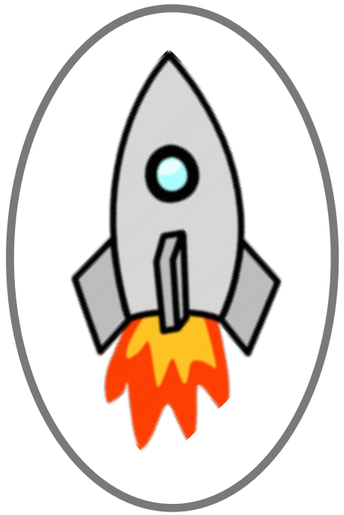
Pig

Shop

Predictions for a new sample involves figuring out which element in the training data it is similar to

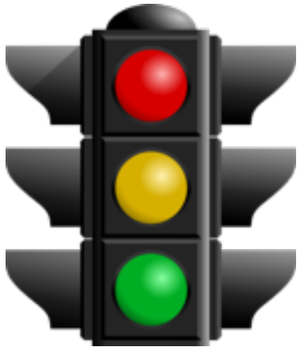
The nearest neighbor

K-nearest-neighbors



Rocket

Buildings

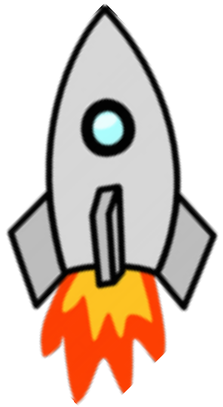


Signal

Pig

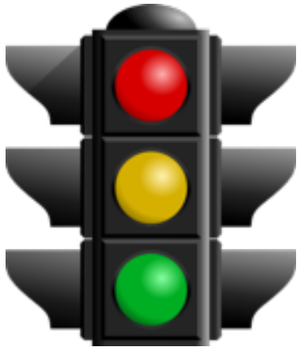
Shop

K-nearest-neighbors



Rocket

Buildings

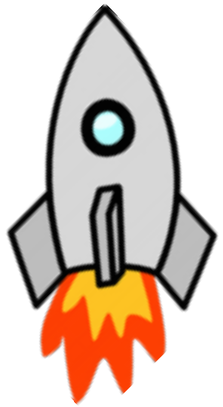


Signal

Pig

Shop

K-nearest-neighbors



Rocket

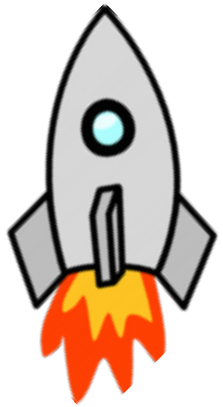
Buildings

Signal

Pig

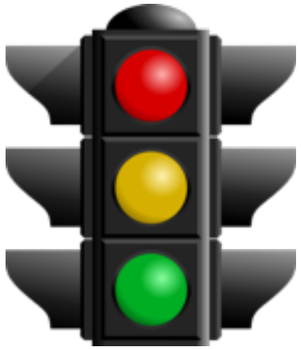
Shop

K-nearest-neighbors



Rocket

Buildings

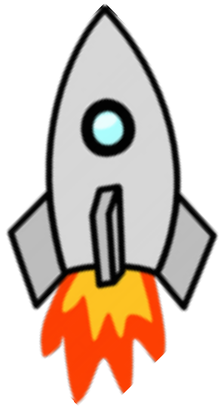


Signal

Pig

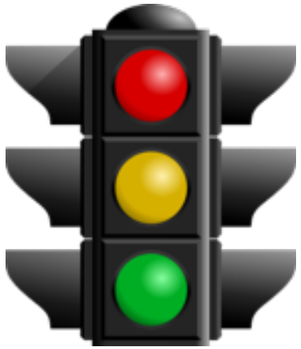
Shop

K-nearest-neighbors



Rocket

Buildings



Signal

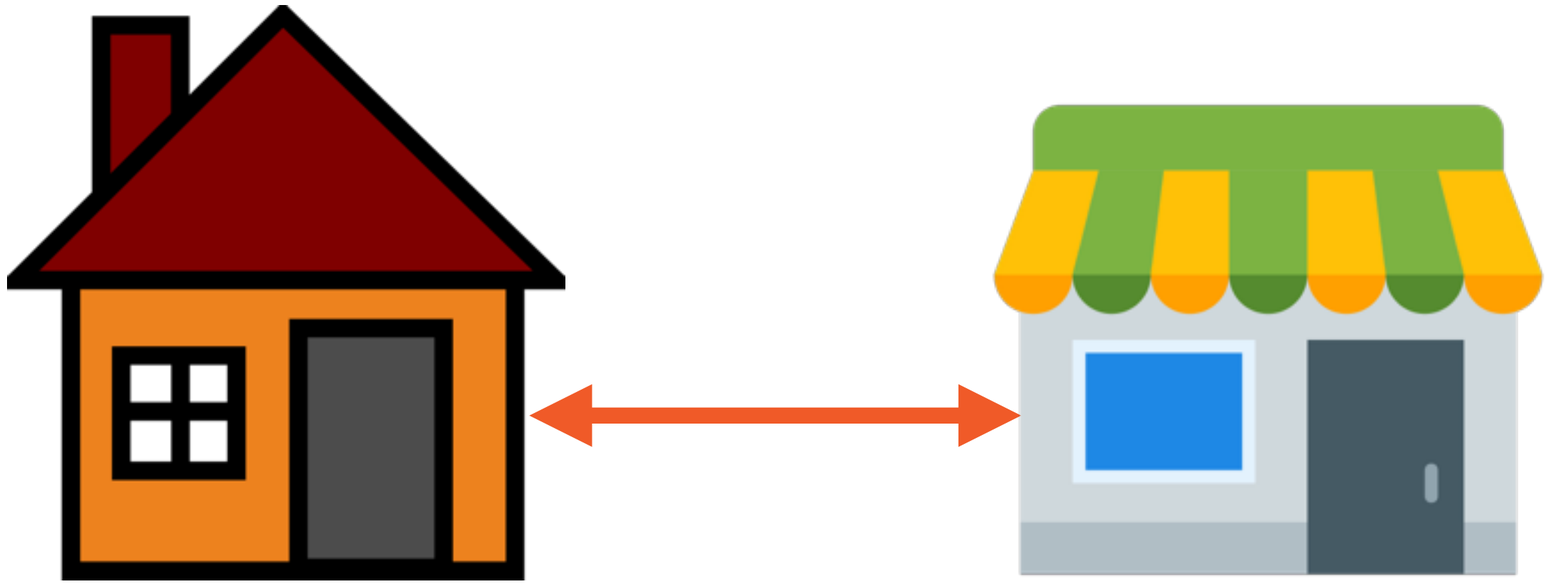
Pig

Shop

K-nearest-neighbors

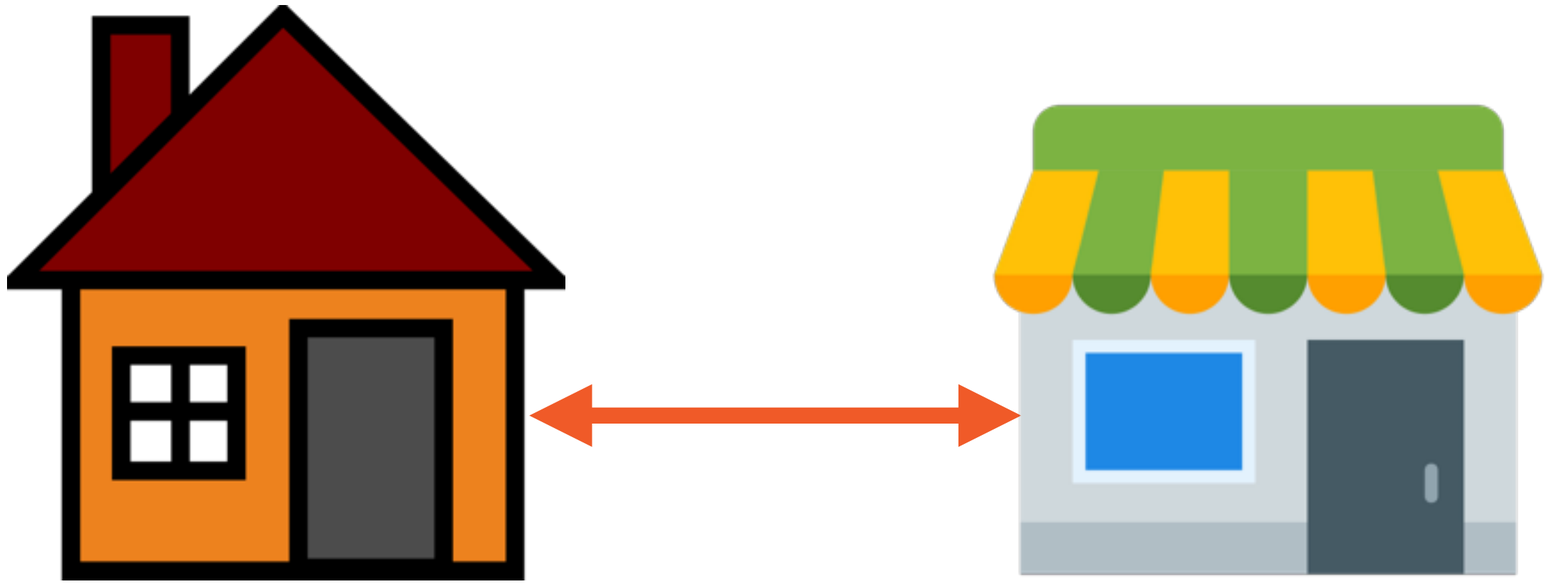
How do we calculate neighbors of a sample?

K-nearest-neighbors



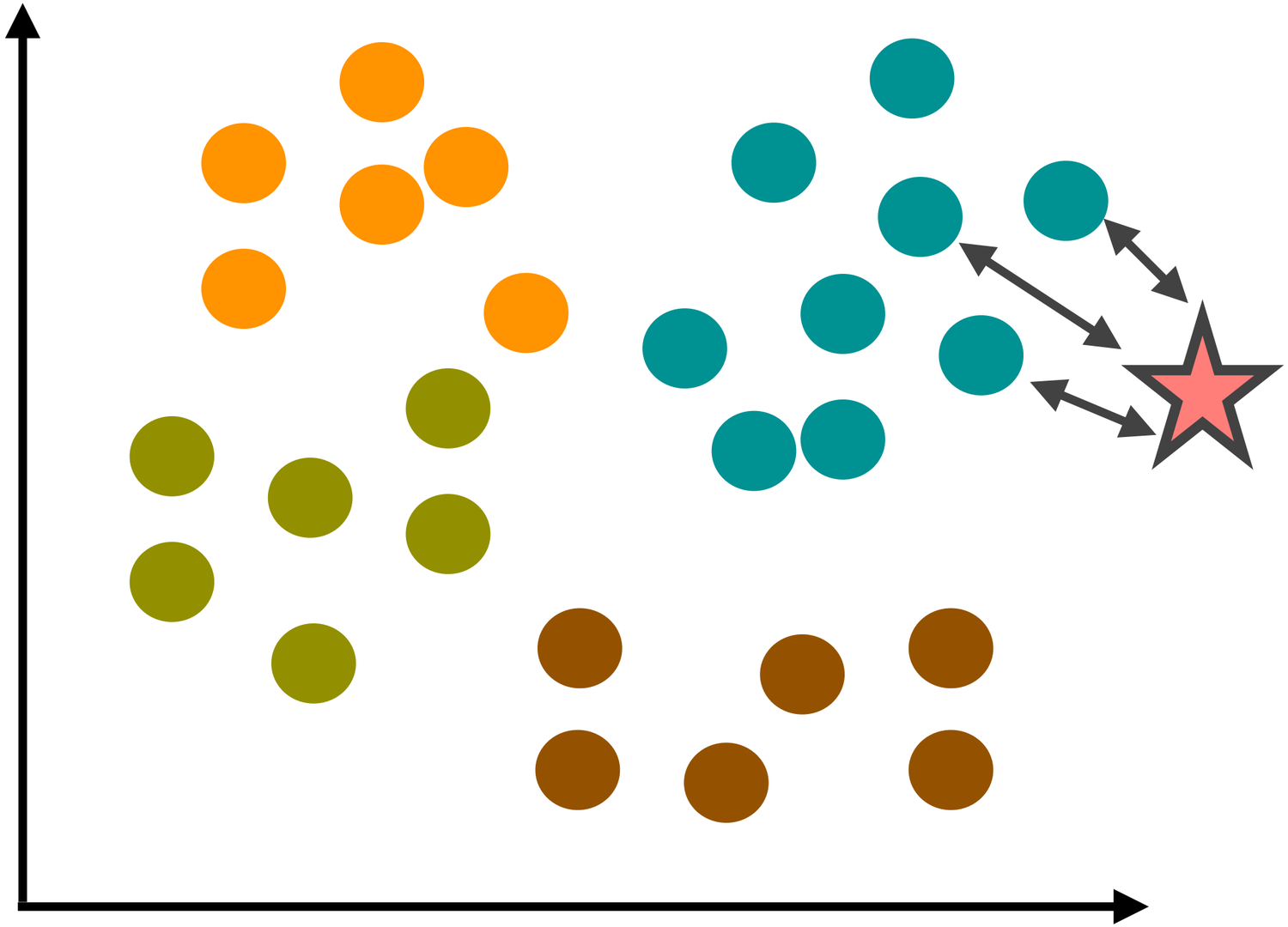
Distance measures

K-nearest-neighbors

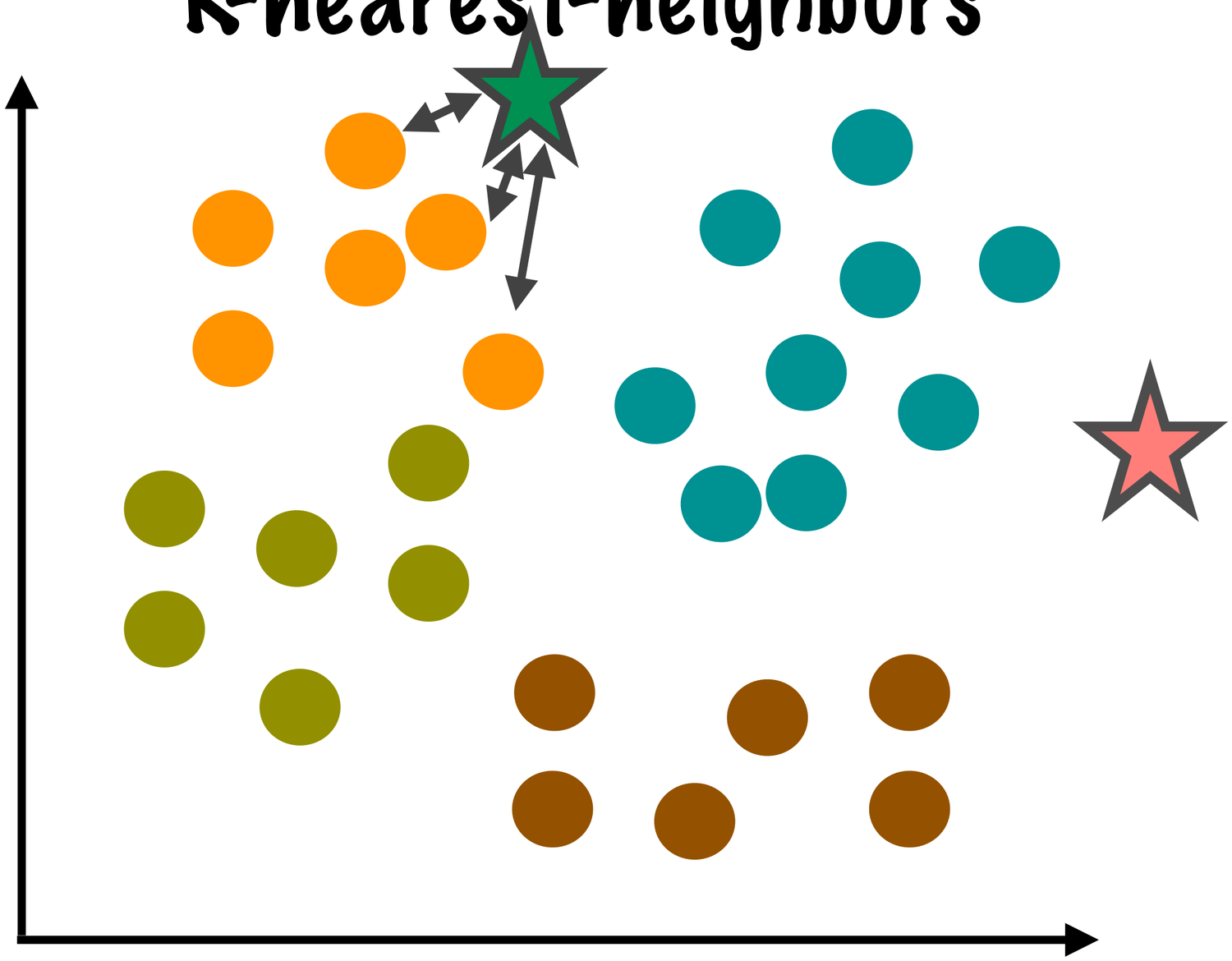


Euclidean distance, Hamming distance, Manhattan distance

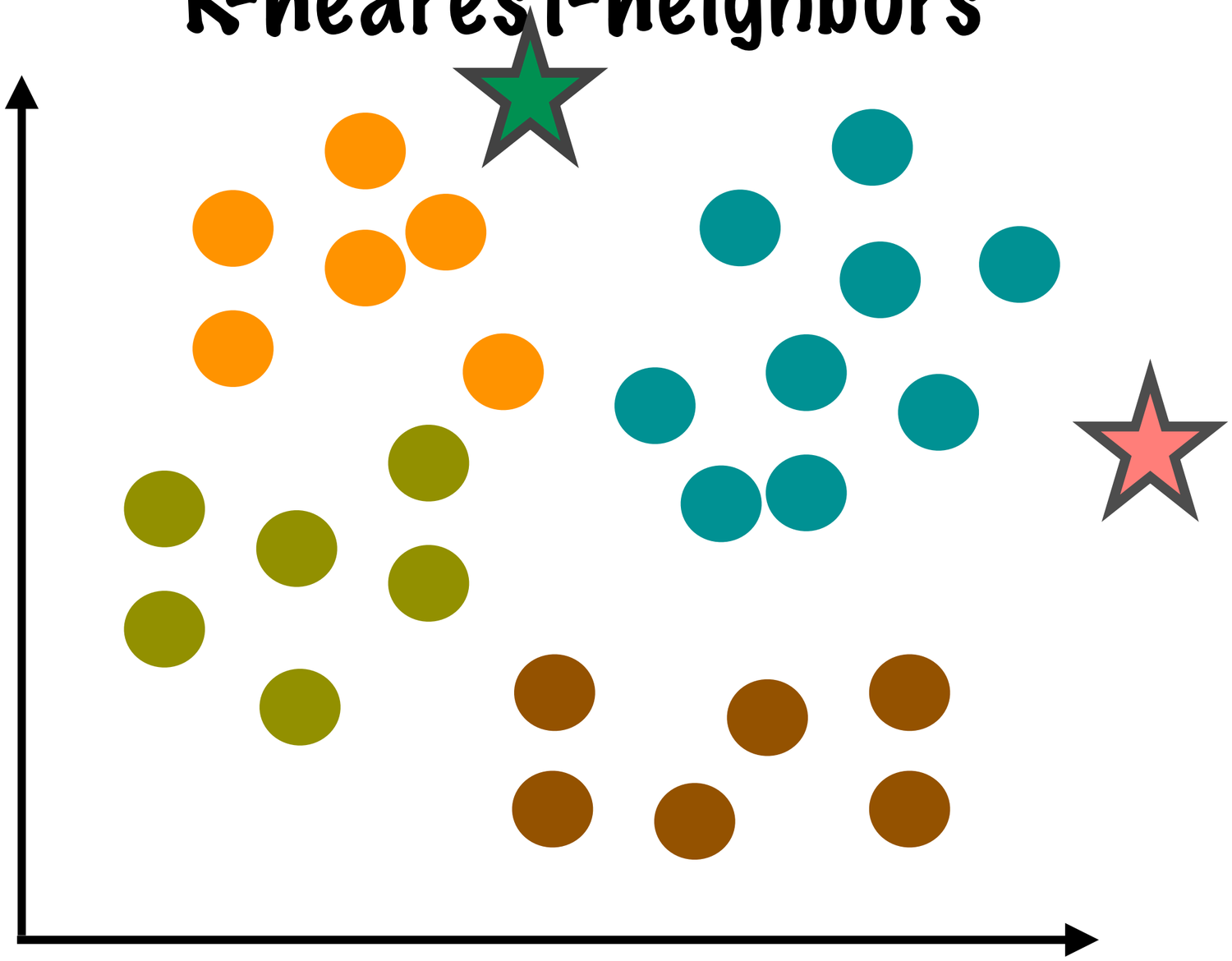
K-nearest-neighbors



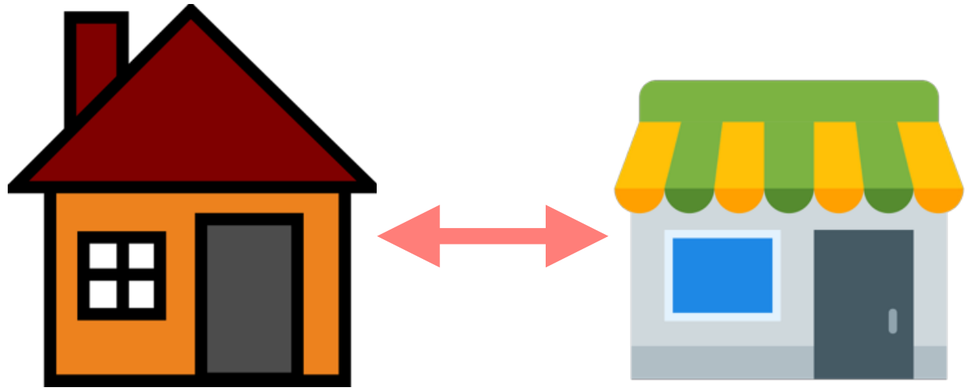
K-nearest-neighbors



K-nearest-neighbors

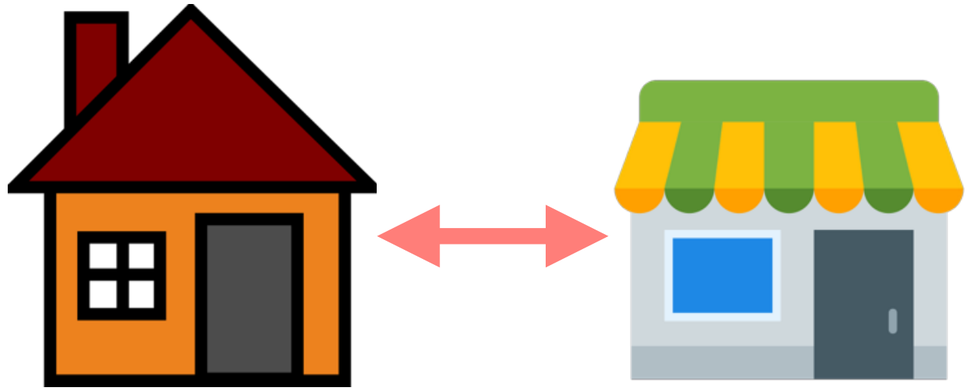


Distance Measures



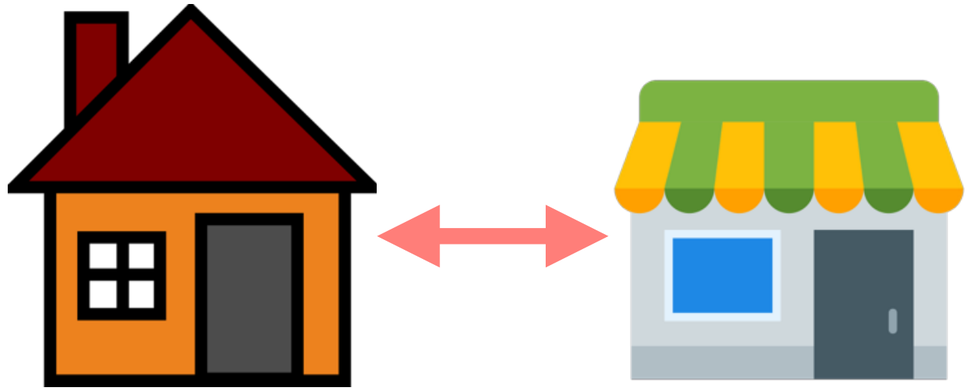
EuclideanDistance(x, xi) = sqrt(sum((xj - xij)^2))

Distance Measures



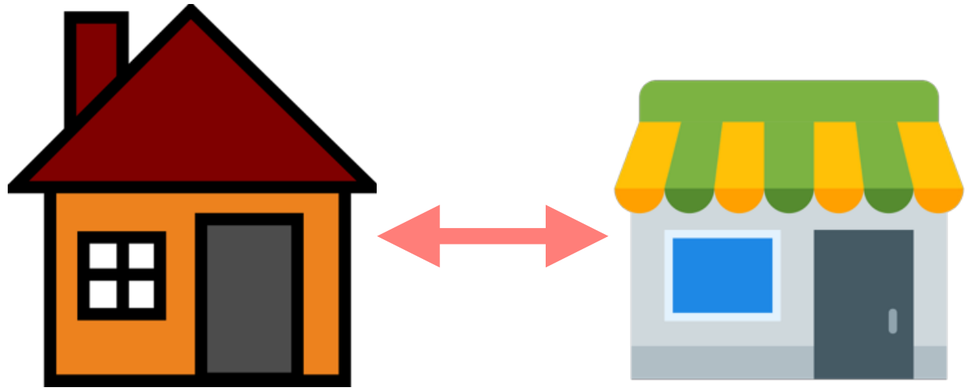
EuclideanDistance(x, xi) = sqrt(sum((xj - xij)^2))

Distance Measures



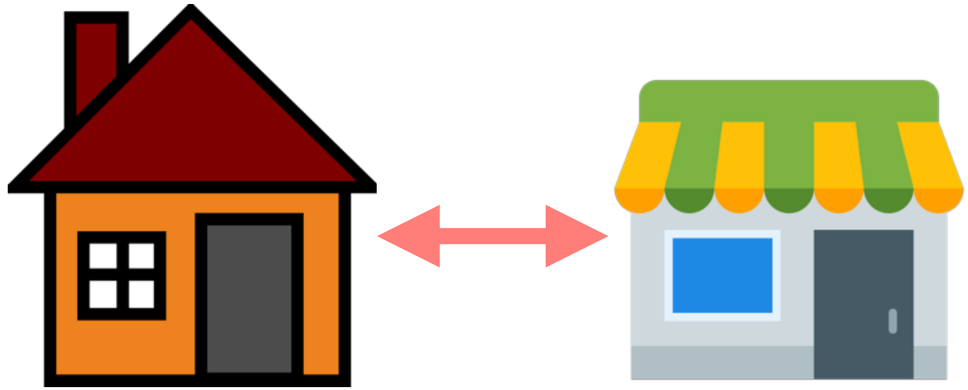
EuclideanDistance(x, xi) = sqrt(sum((xj - xij)^2))

Distance Measures



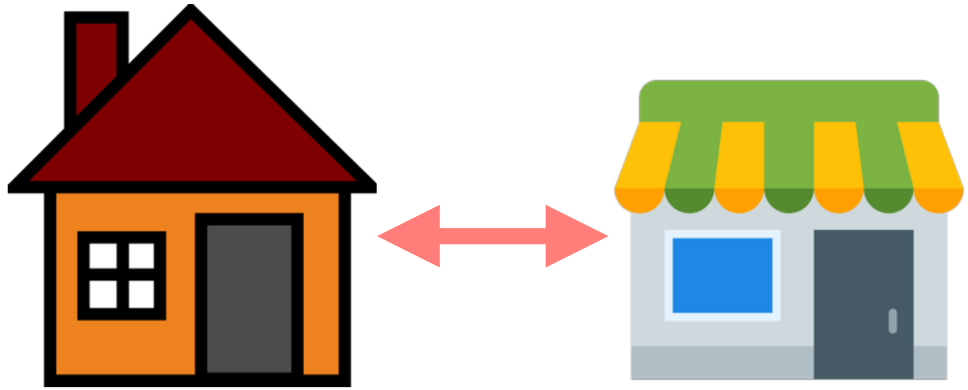
EuclideanDistance(x, xi) = sqrt(**sum**((xj - xij)^2))

Distance Measures



EuclideanDistance(x, xi) = sqrt(sum((xj - xij)^2))

Distance Measures



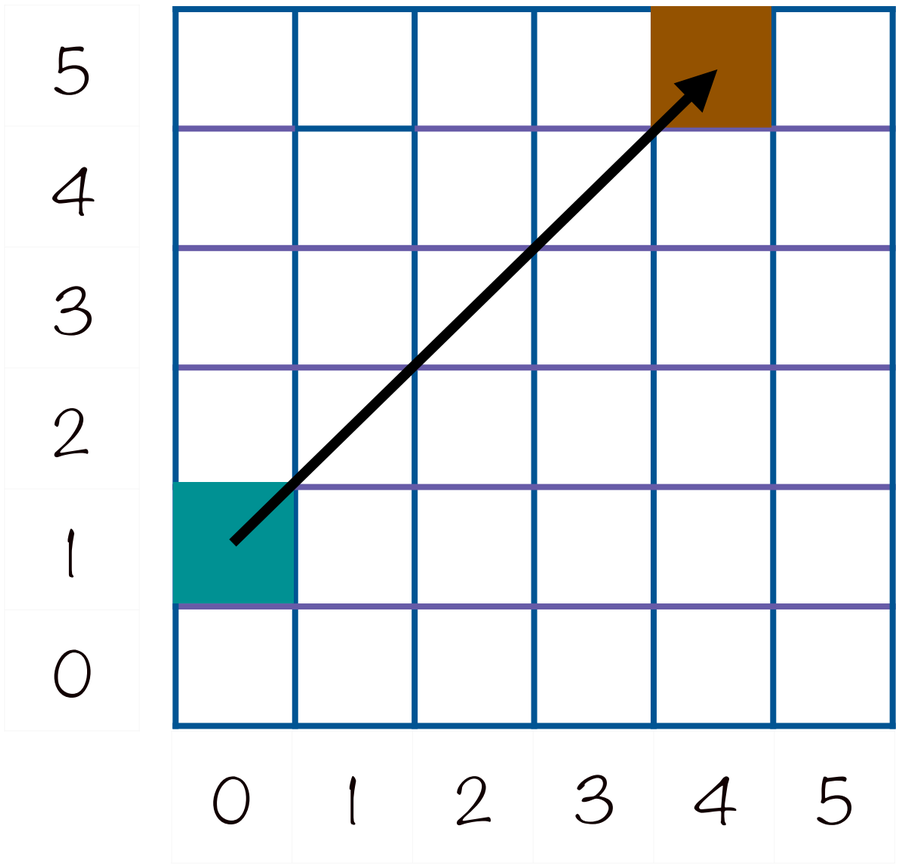
EuclideanDistance(x, xi) = sqrt(sum((xj - xij)^2))

Distance Measures

Distance Measure

Euclidean Distance

As the crow flies



L1 Distance

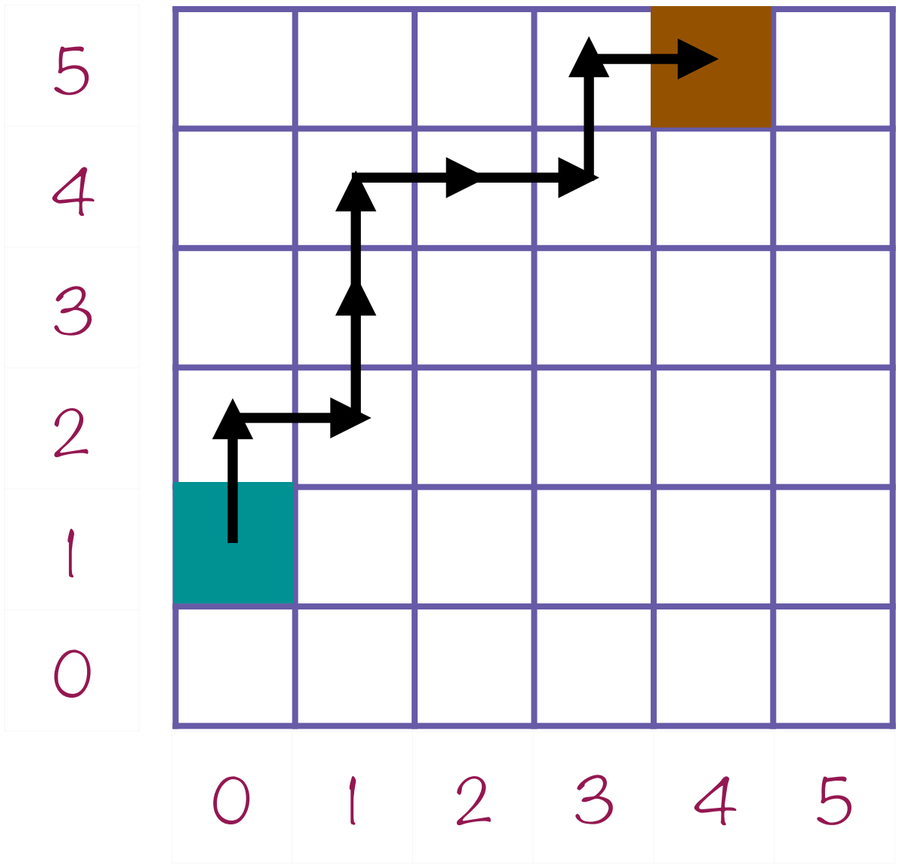
Distance Measure

L1 distance

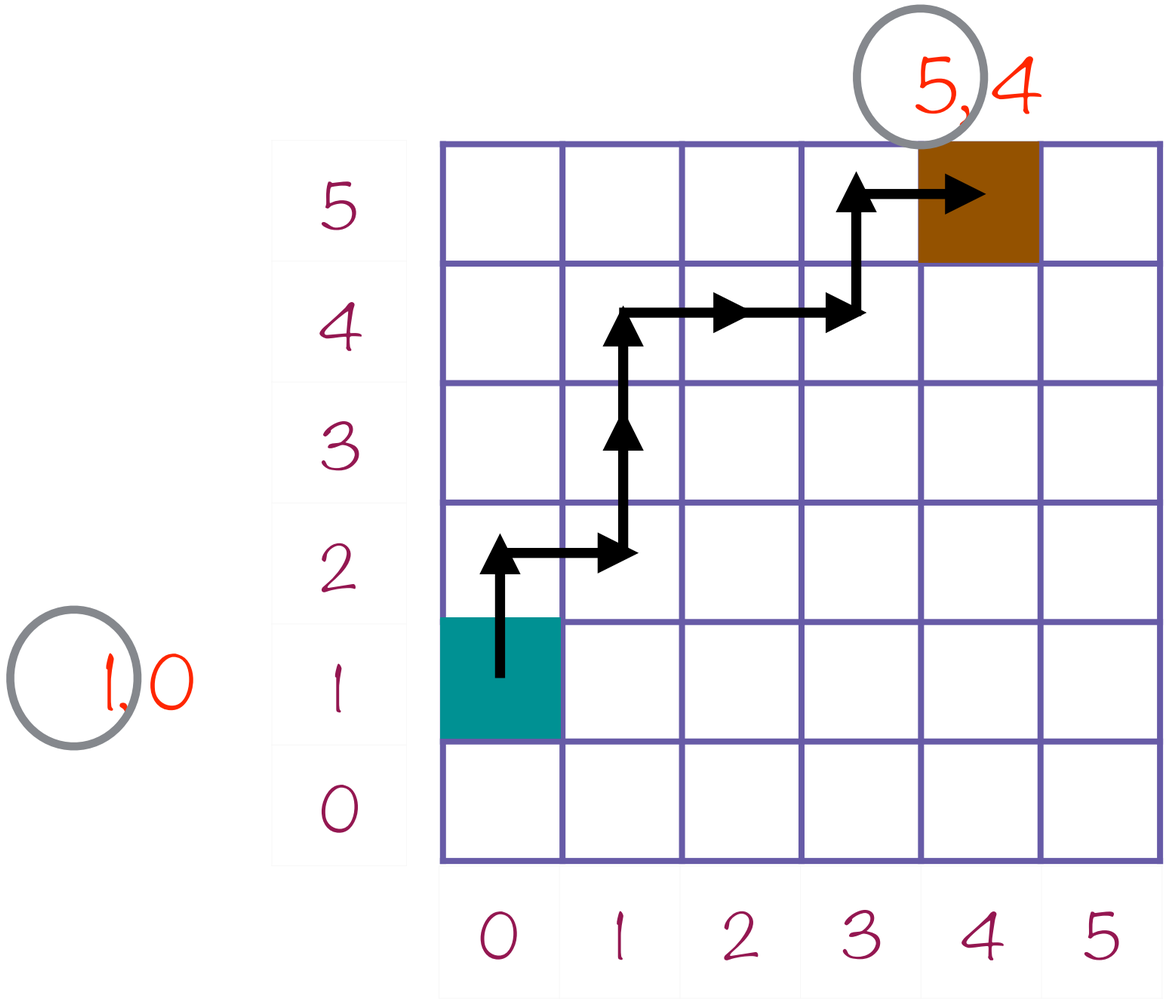
Snake distance

City block distance

Manhattan distance

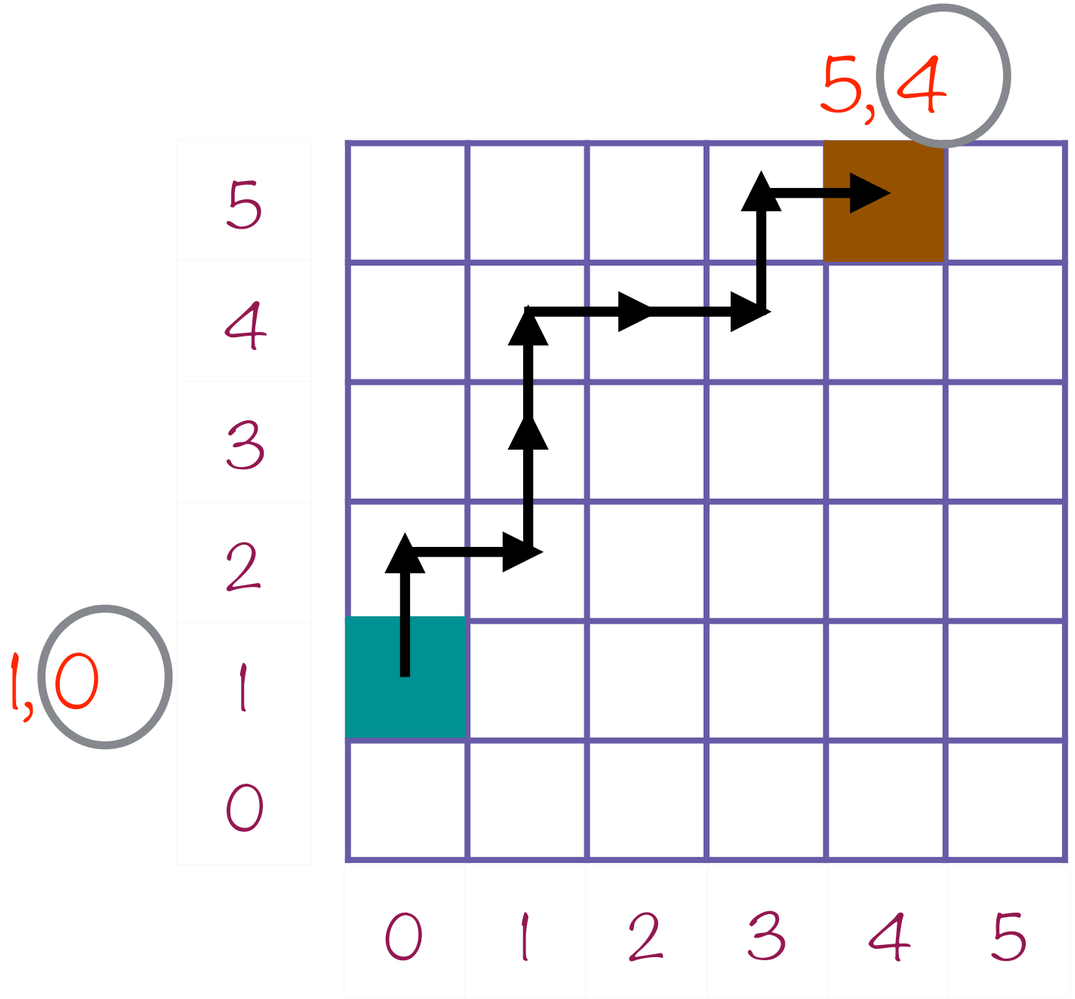


L1 Distance



5-1 = 4

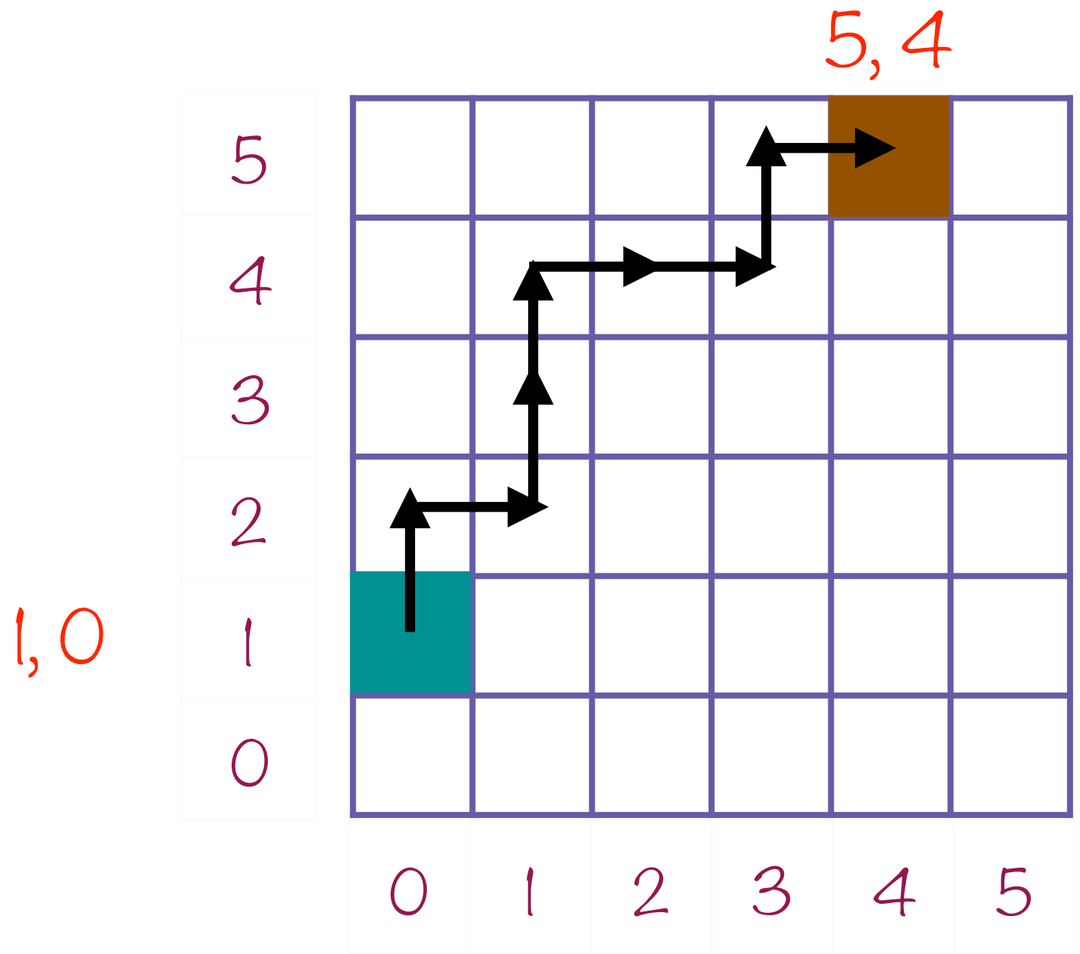
L1 Distance



5-1 = 4

4-0 = 4

L1 Distance



5-1 = 4

4-0 = 4

= 8

Demo

Handwritten image recognition using the k-nearest-neighbors ML algorithm

- Use the L1 distance measure to find the nearest neighbor
- Measure the accuracy of the algorithm on the test data

KNN Implemented in TensorFlow

Getting MNIST images

Access the MNIST training and test images in batches using the TensorFlow libraries

Running the algorithm

Predict labels for all the test data and measure accuracy

Calculating L1 distance

Find the distance between the test digit and all training digits

KNN Implemented in TensorFlow

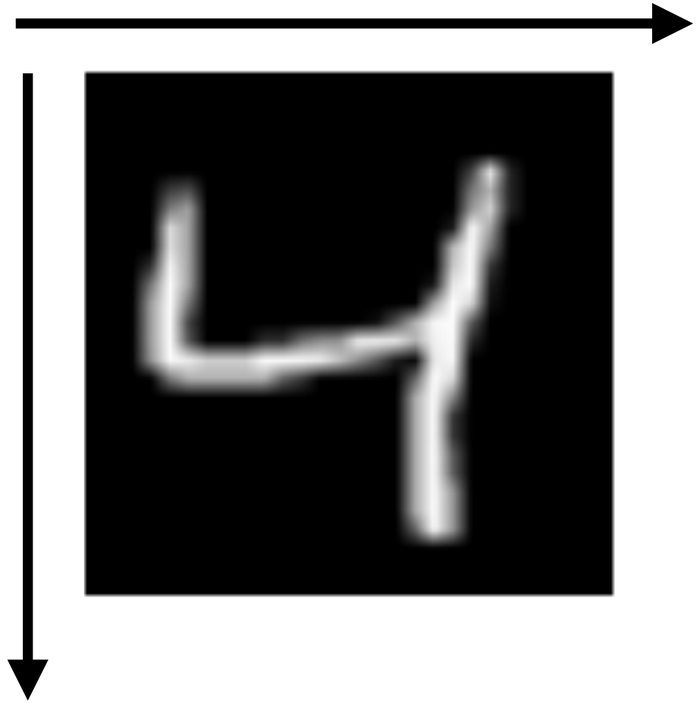
Getting MNIST images

Access the MNIST training and test images in batches using the TensorFlow libraries

MNIST Dataset

5 0 4 1

MNIST Dataset

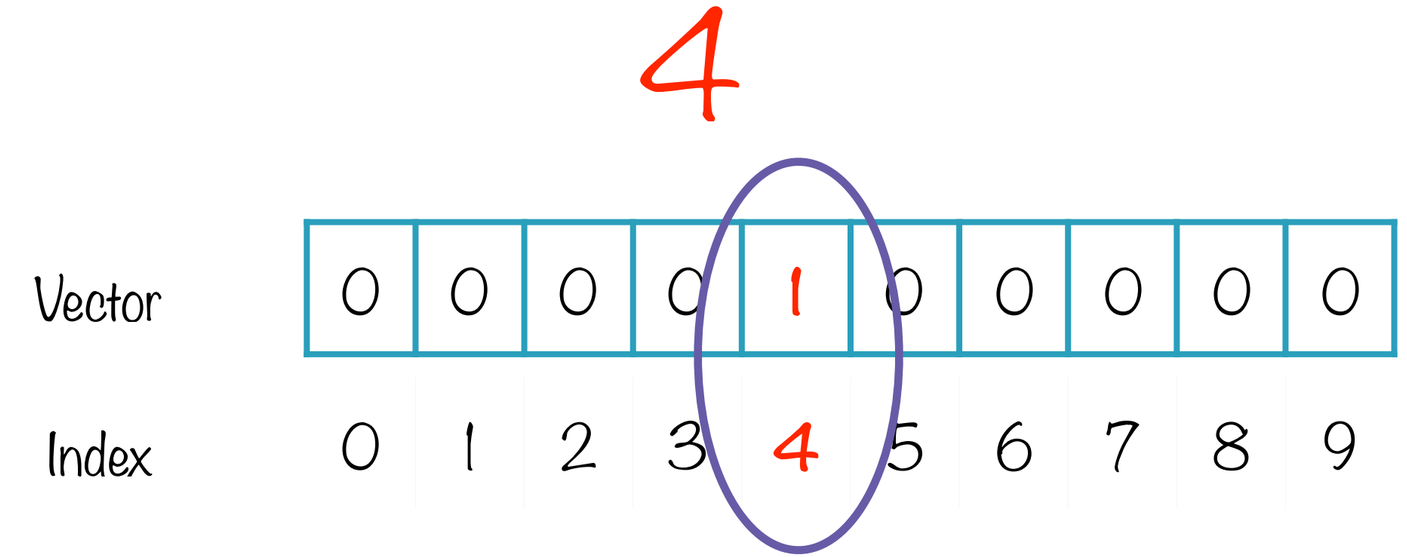


4

Every image is standardized
to be of size 28x28

= 784 pixels

Representing Labels

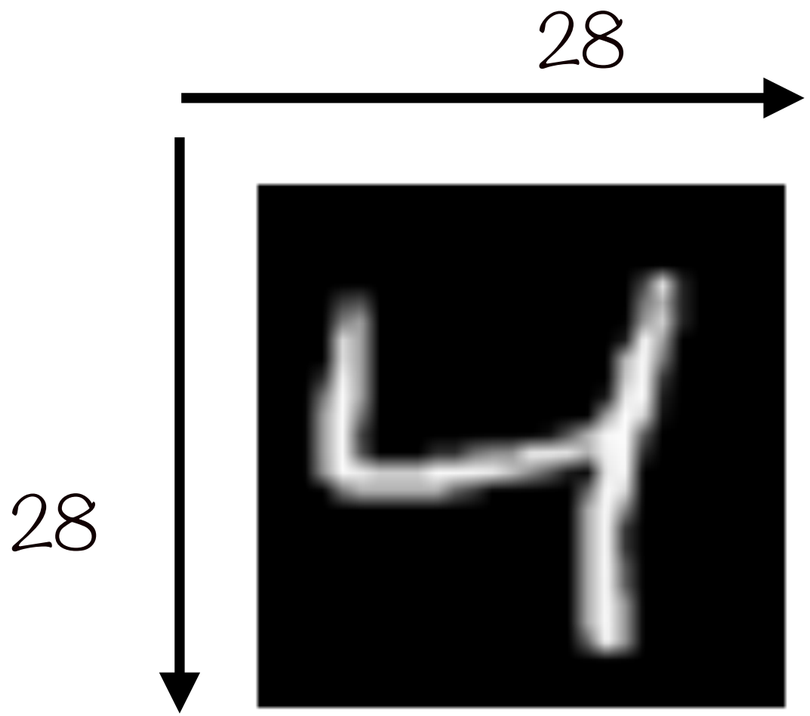


Representing Labels

5

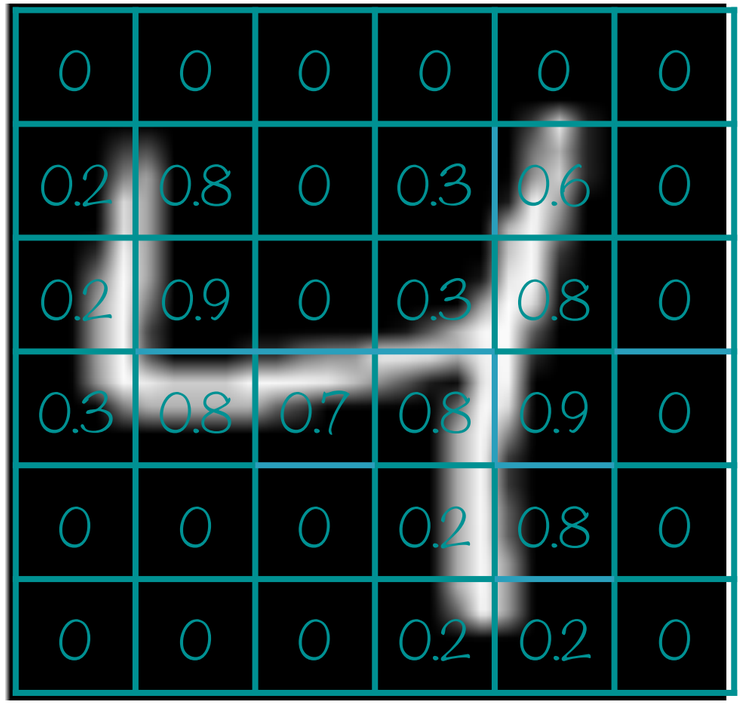
Vector	0	0	0	0	0	1	0	0	0	0
Index	0	1	2	3	4	5	6	7	8	9

Representing Images



= 784 pixels

Representing Images



= 784 pixels

Representing Images

0	0	0	0	0	1
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Representing Images

0	0	0	0	0	0

0	0	0	0	0	0
~~0.2~~	~~0.8~~	~~0~~	~~0.3~~	~~0.6~~	~~0~~
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Representing Images

0	0	0	0	0	0	0.2	0.8	0	0.3	0.6	0

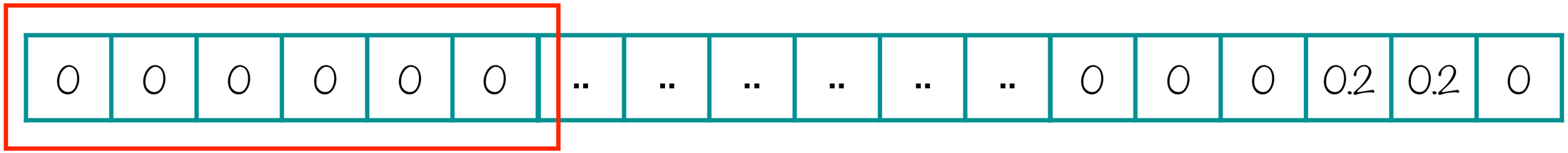
0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
~~0.2~~	~~0.9~~	~~0~~	~~0.3~~	~~0.8~~	~~0~~
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Representing Images

0	0	0	0	0	0	0.2	0.9	0	0.3	0.8	0	0.2	0.8	0	0.3	0.6	0

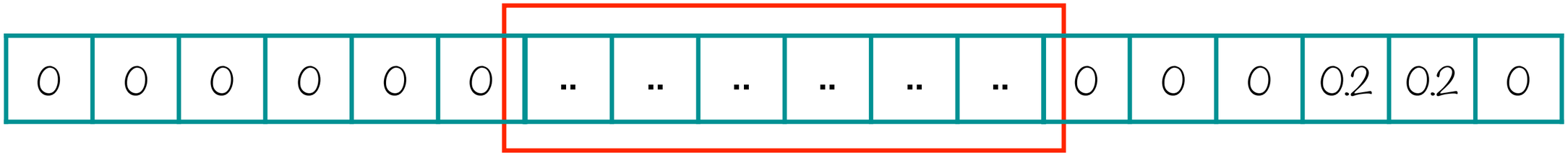
0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Representing Images



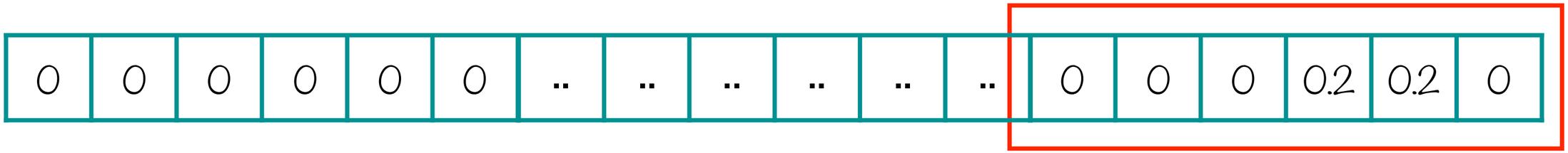
0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Representing Images



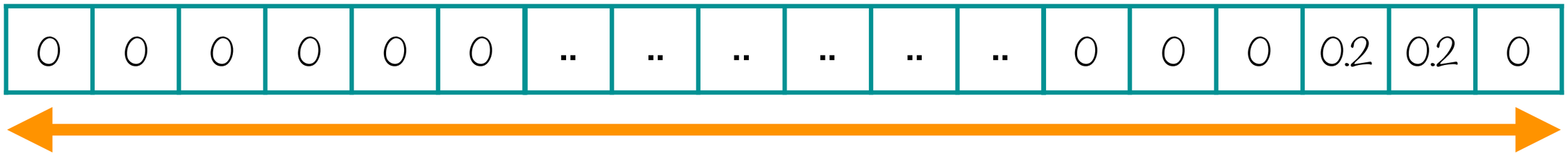
0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Representing Images



0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Representing Images



= 784 pixels

KNN Implemented in TensorFlow

Getting MNIST images

Access the MNIST training and test images in batches using the TensorFlow libraries

Calculating L1 distance

Find the distance between the test digit and all training digits

L1 Distance

Training

0.2	0.8	0	0.3	0.6	0

Test

0	0.6	0.2	0.3	0.3	0.1

0.2	0.8	0	0.3	0.6	0

tf.negative()

0	-0.6	-0.2	-0.3	-0.3	-0.1

tf.add()

0.2	0.2	-0.2	0	0.3	-0.1

L1 Distance

tf.add()

0.2	0.2	-0.2	0	0.3	-0.1

tf.abs()

0.2	0.2	0.2	0	0.3	0.1

tf.reduce_sum()

1.0

tf.add()

0	0.2	0.3	0.6	-0.3	-0.1

tf.abs()

0	0.2	0.3	0.6	0.3	0.1

tf.reduce_sum()

1.5

tf.add()

-0.2	0.4	-0.2	0	-0.3	-0.1

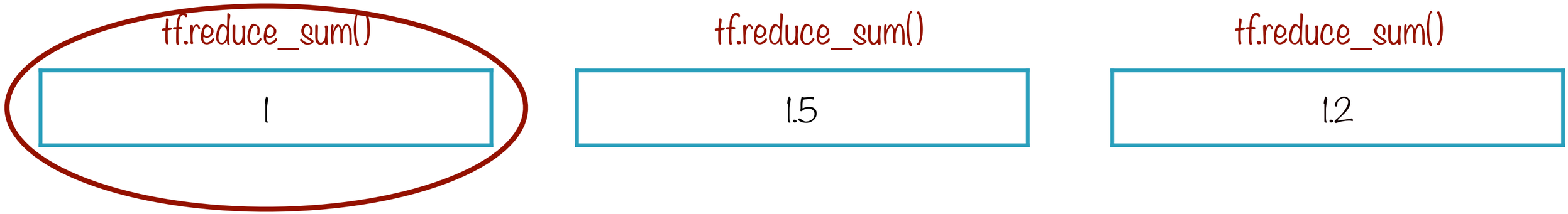
tf.abs()

0.2	0.4	0.2	0	0.3	0.1

tf.reduce_sum()

1.2

L1 Distance



index = 0

KNN Implemented in TensorFlow

Getting MNIST images

Access the MNIST training and test images in batches

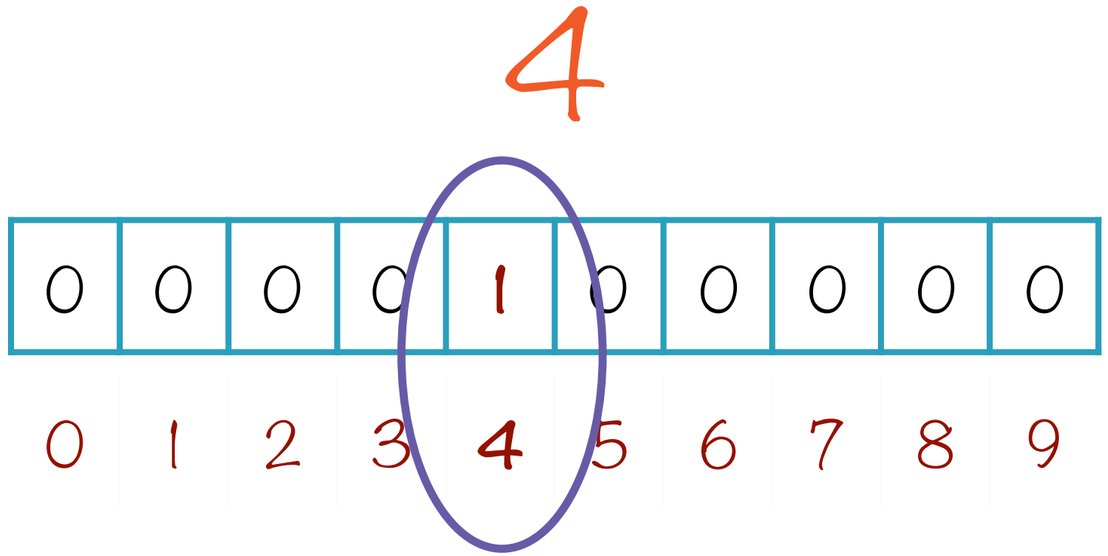
Calculating L1 distance

Find the distance between the test digit and all training digits

Running the algorithm

Predict labels for all the test data and measure accuracy

Representing Labels



np.argmax()

Summary

Familiar with the MNIST handwritten digit dataset

Understood the logic behind the K-nearest-neighbors algorithm

Implemented K-nearest-neighbors using L1 distance to identify handwritten digits from 0 to 9