Exploring Feature Extraction Techniques

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Representing images as numeric data

Extracting features from images

Representing text as numeric data

Extracting features from text

Scope of Feature Engineering

Feature selection

Feature learning

Feature extraction

Feature combination

Dimensionality reduction

Scope of Feature Engineering

Feature selection

Feature learning

Feature extraction

Feature combination

Dimensionality reduction

Feature Extraction

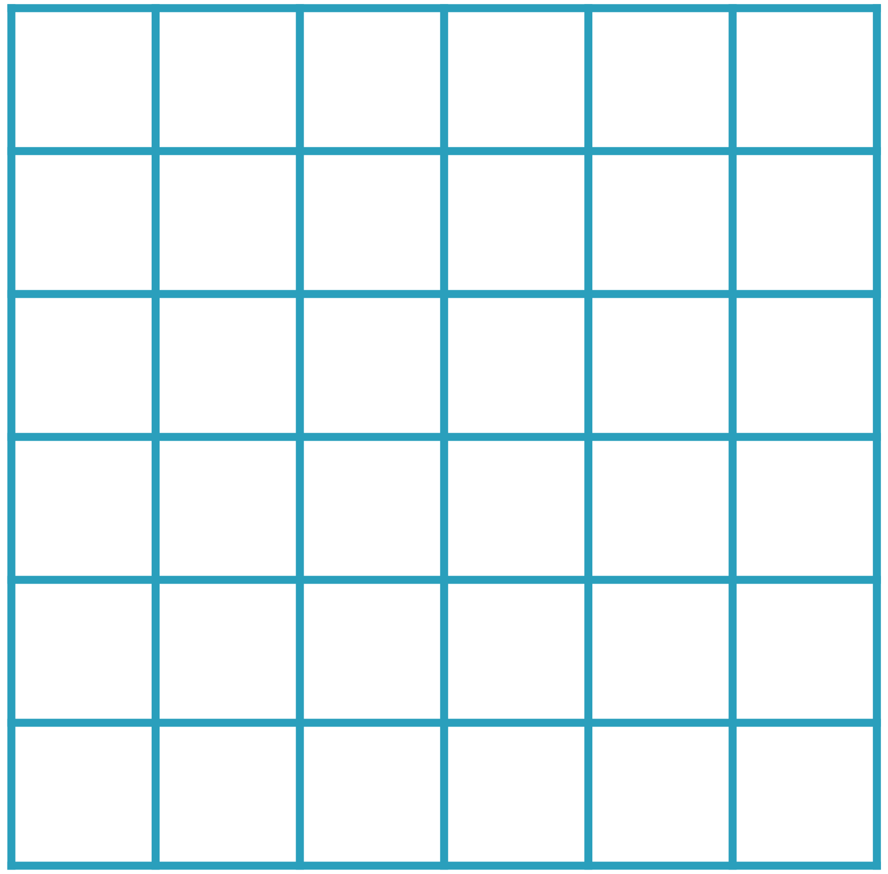
Differs from feature selection in that input features are fundamentally transformed into derived features, which are often unrecognizable and hard to interpret.

Representing Images in Numeric Form

When working with images and text, feature extraction is often the bottleneck to scaling

Images as Matrices

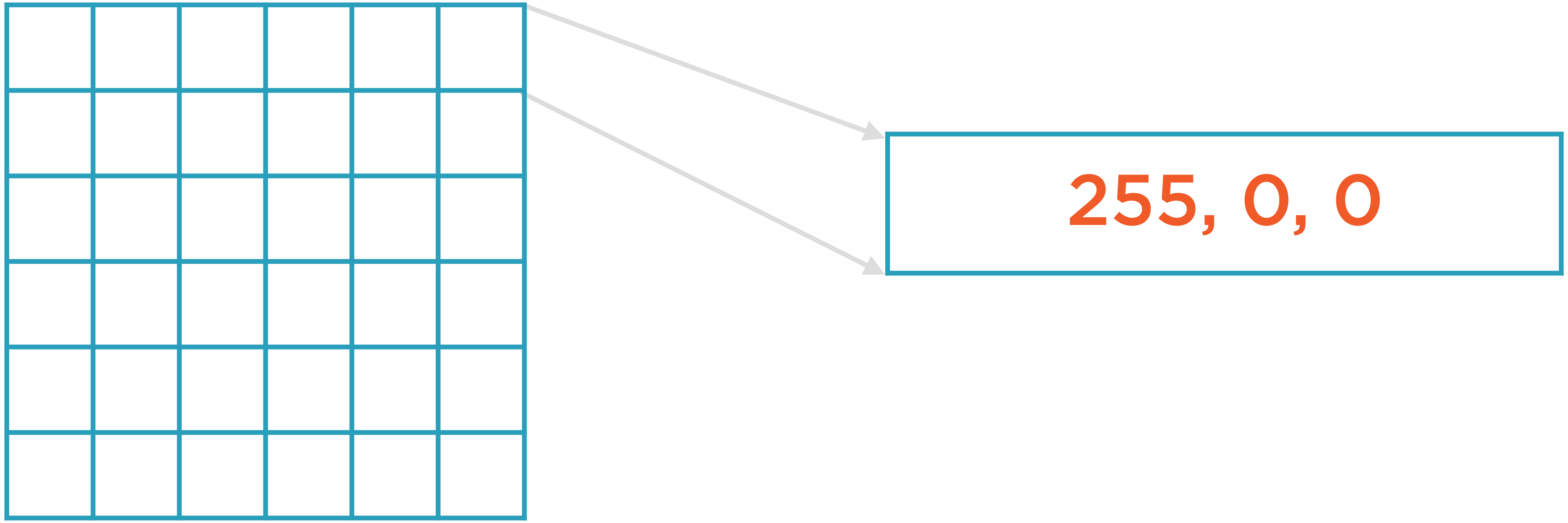
RGB Images



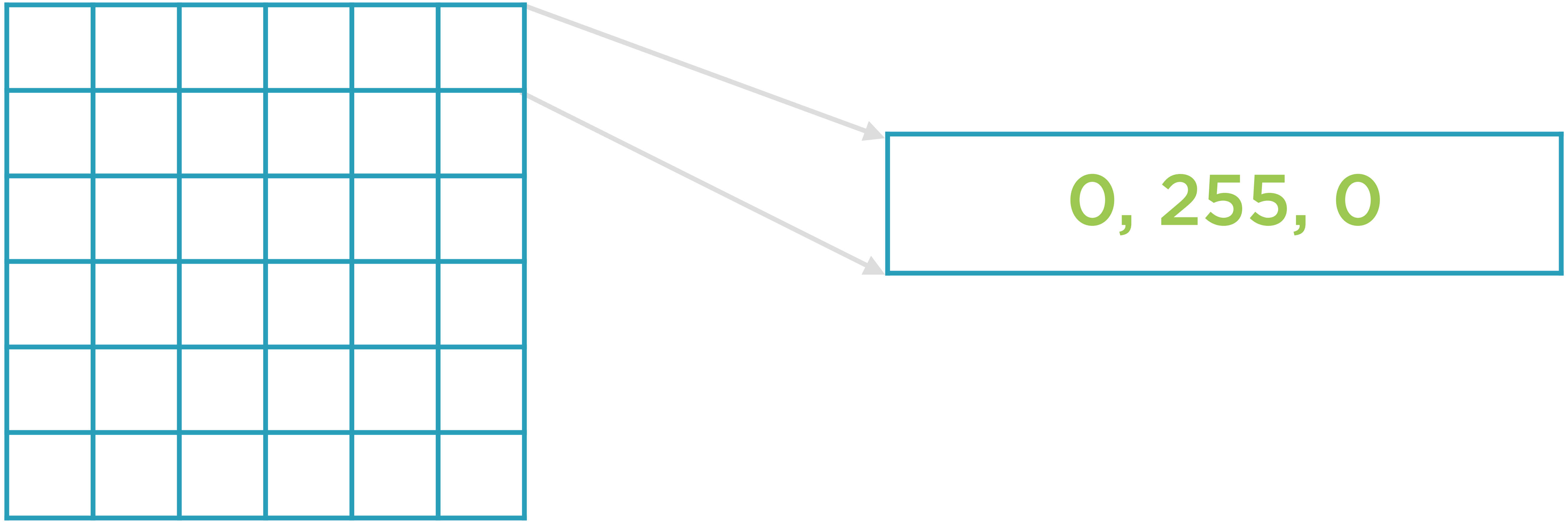
RGB values are for color images

R, G, B: 0-255

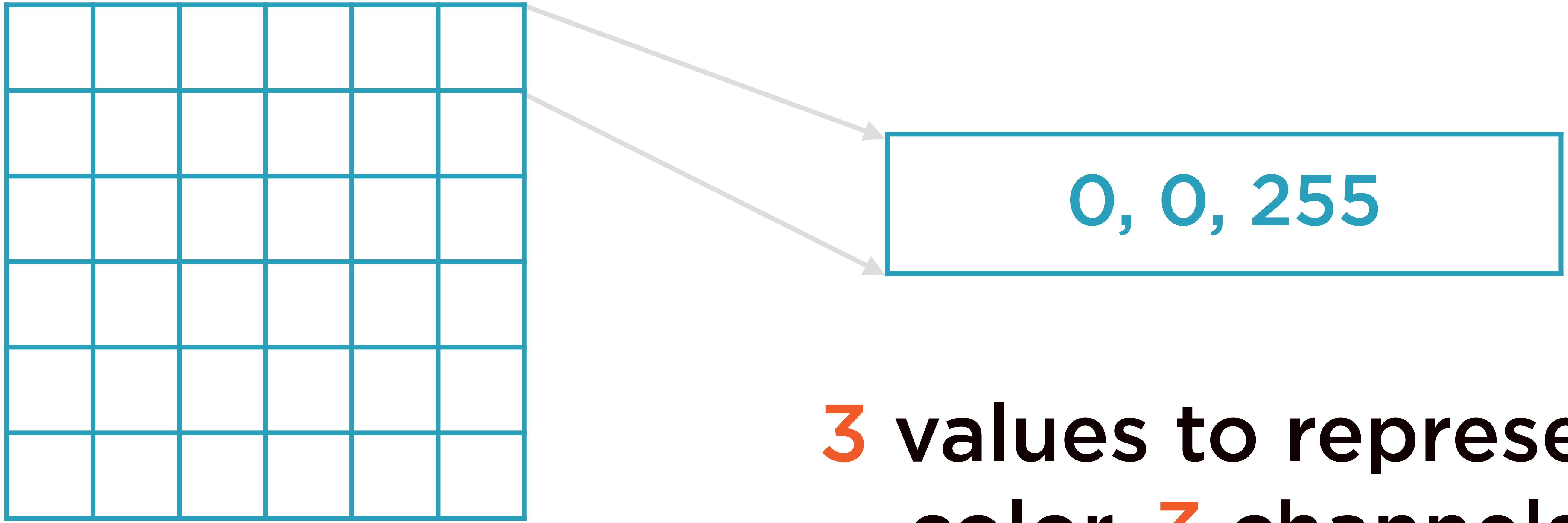
RGB Images



RGB Images

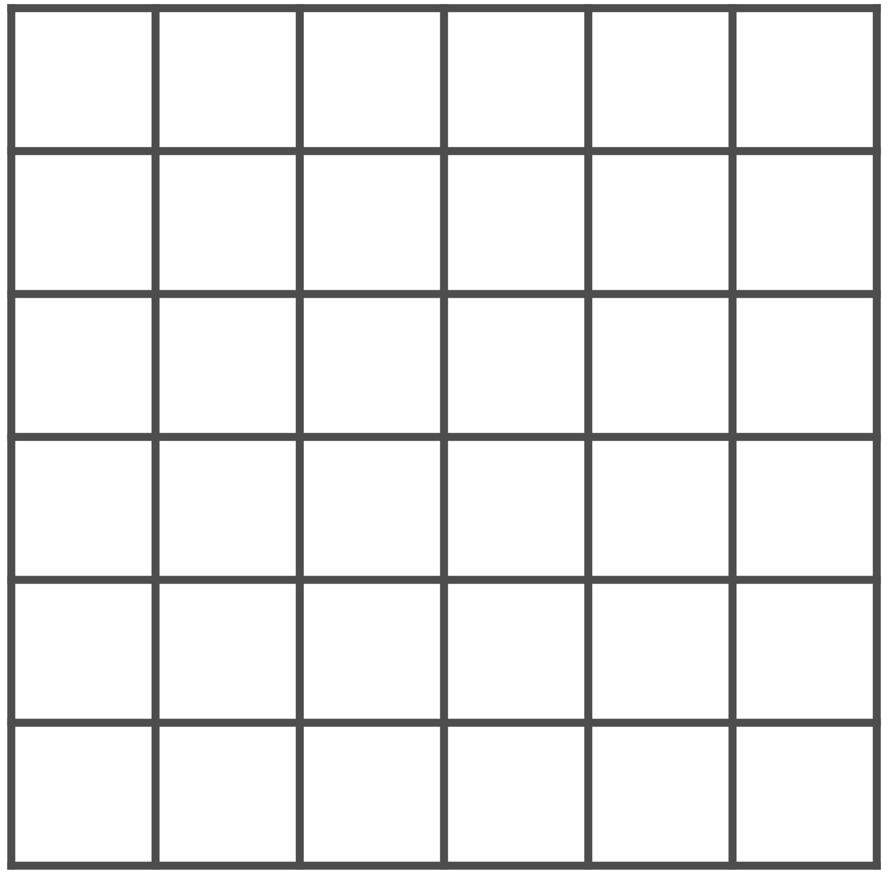


RGB Images

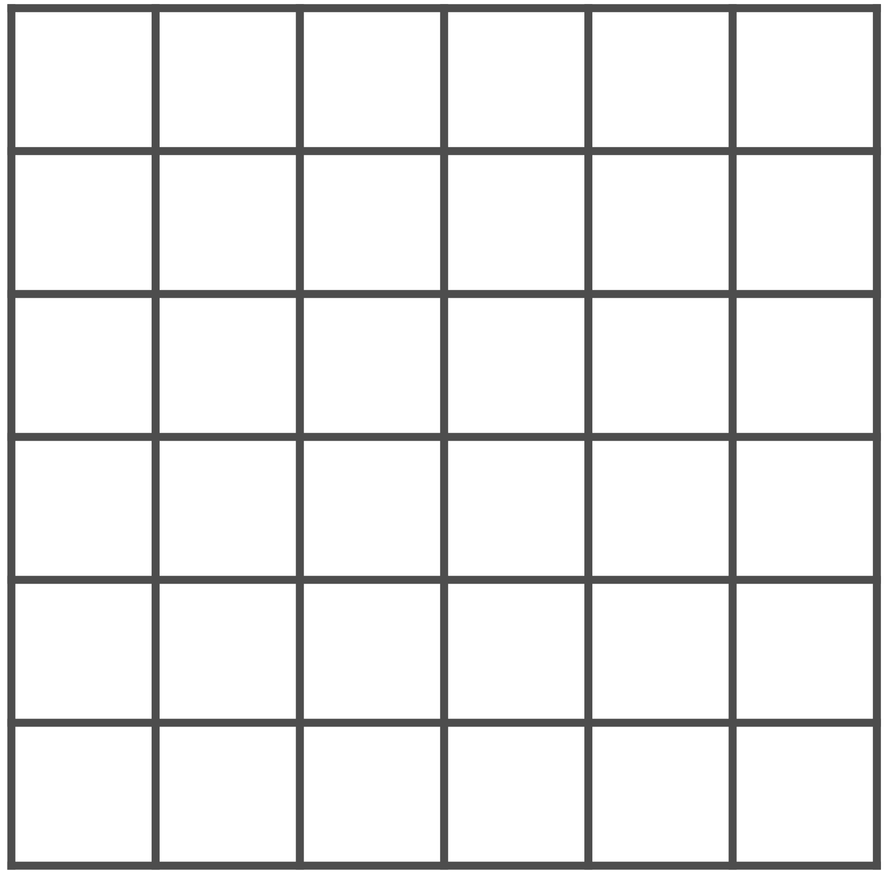


3 values to represent color, 3 channels

Grayscale Images



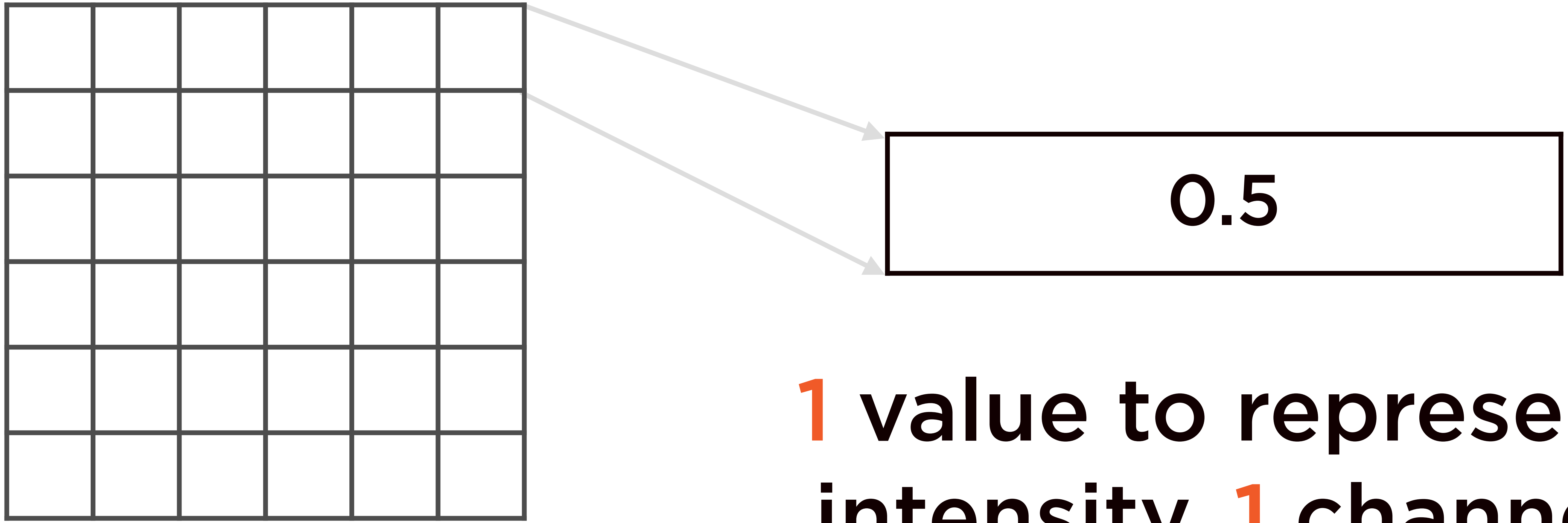
Grayscale Images



Each pixel represents only intensity information

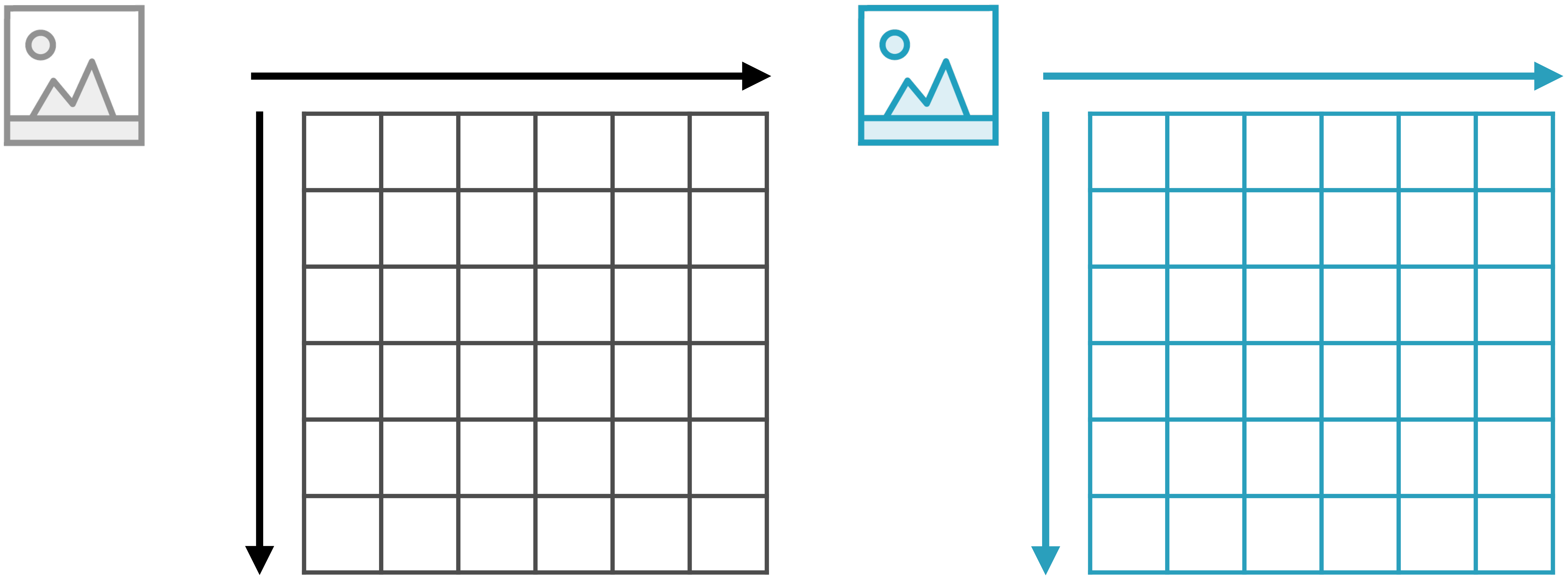
0.0 - 1.0

Grayscale Images



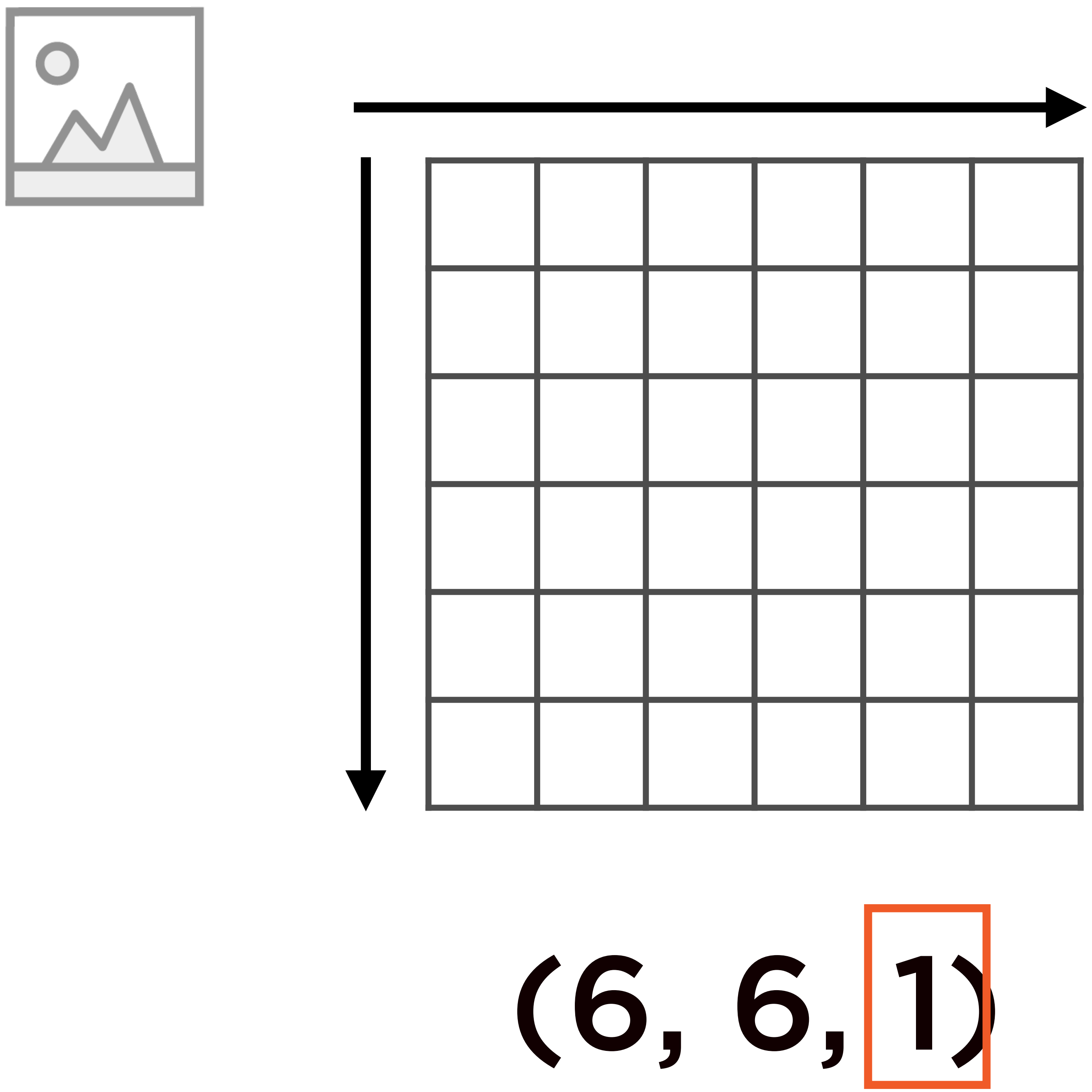
1 value to represent intensity, 1 channel

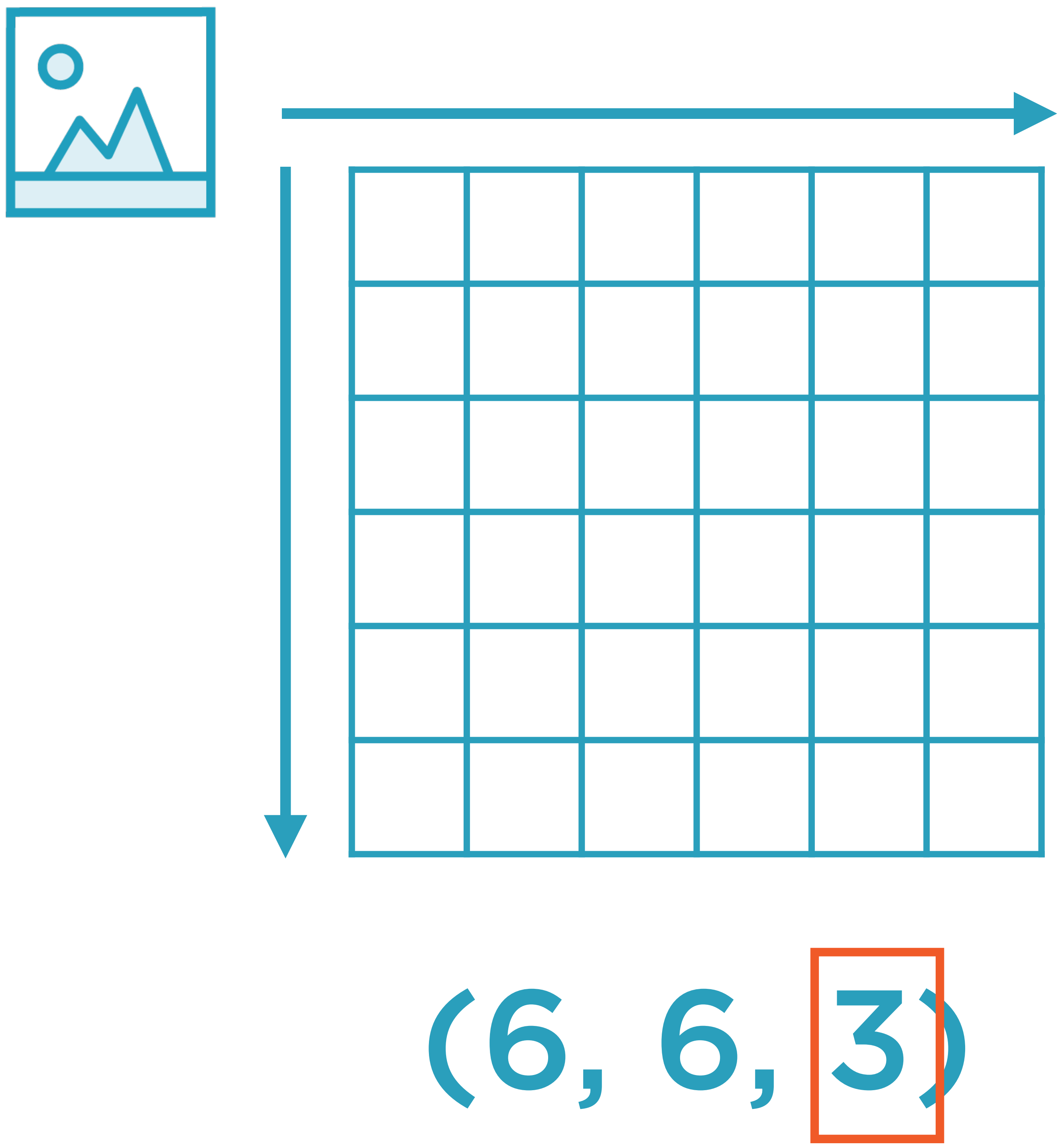
Images as Matrices



Images can be represented by a 3-D matrix

Images as Tensors





List of Images

ML frameworks (e.g. TensorFlow) usually deal with a list of images in one 4-D Tensor

List of Images

The images should all be the same size

List of Images

$(10, 6, 6, \boxed{3})$

The number of channels

List of Images

$(10, \boxed{6, 6}, 3)$

The height and width of each image in the list

List of Images

(10, 6, 6, 3)

The number of images

Image Pre-processing Methods

Uniform Aspect Ratio	Uniform Image Size	Mean and Perturbed Images
Normalized Image Inputs	Dimensionality Reduction	Data Augmentation

Common techniques to improve machine learning model performance

Feature Detection and Extraction in Image Processing

Feature Detection and Extraction

In the context of image processing, algorithms that detect and extract the appropriate, most interesting, features from images.

Feature Detection

Identify “right” feature representations of images

Starting point of many computer vision algorithms

Repeatability an important factor

Whether the same feature will be detected in two or more images

Feature Detection

Abstractions such as style, texture

Content such as edges, corners

Feature Detection

Points of interest

- Corner points in edge detection
- Should be stable and repeatably identifiable

Regions of interest

- Blob detection in object tracking

Feature Detection

Edge detection

- Boundary between two image regions
- Can be of arbitrary shape

Ridge detection

- One dimensional curve which represents an axis of symmetry

Feature Detection

Which points are most interesting?

Which regions are most interesting?

What is interesting about them?

Feature Detection

Detection of interest points

Which regions are most interesting?

What is interesting about them?

Feature Detection

Detection of interest points

Detection of blobs (areas) of interest

What is interesting about them?

Feature Detection

Detection of interest points

Detection of blobs (areas) of interest

Computation of image descriptors

Key Points (a.k.a Points of Interest)

Points in the image that define what is interesting and must be captured in the feature representation of the image.

Properties of Key Points

Should be well-defined

Should not be affected by operations such as

- Rotation
- Translation
- Expansion
- Warping

Properties of Key Points

Interest point = Point with well-defined position that can be clearly defined

Types of interest points

- Corners: Intersection of two edges
- Intensity maxima/minima
- Line endings

Feature Detection

Detection of interest points

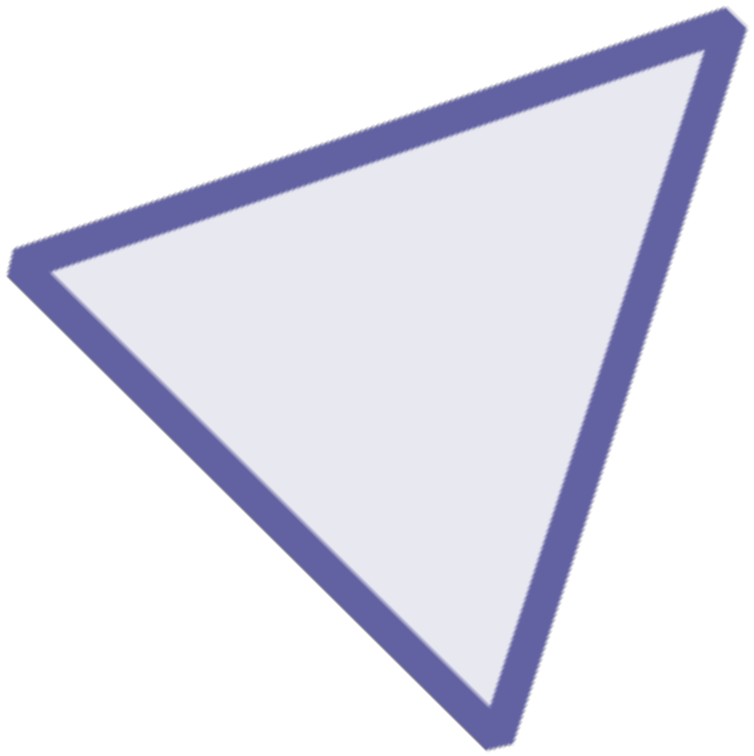
Detection of blobs (areas) of interest

Computation of image descriptors

Blobs (a.k.a Regions of Interest)

Interesting regions within an image within which points are similar and share properties that are different from surrounding points.

Blob Detection Applications



Complementary to points of interest

Capture additional information

- Object recognition
- Object motion tracking
- Texture analysis and detection
- Image segmentation

Blob detection

文A

This article may be too technical for most readers to understand. Please help improve it to make it understandable to non-experts, without removing the technical details.

Learn more

In computer vision, **blob detection** methods are aimed at detecting regions in a digital image that differ in properties, such as brightness or color, compared to surrounding regions. Informally, a blob is a region of an image in which some properties are constant or approximately constant; all the points in a blob can be considered in some sense to be similar to each other. The most common method for blob detection is convolution.

Given some property of interest expressed as a function of position on the image, there are two main classes of blob detectors: (i) *differential methods*, which are based on derivatives of the function with respect to position, and (ii) *methods based on local extrema*, which are based on finding the local maxima and minima of the function. With the more recent terminology used in the field, these detectors can also be referred to as *interest point operators*, or alternatively interest region operators (see also interest point detection and corner detection).

There are several motivations for studying and developing blob detectors. One main reason is to provide complementary information about regions, which is not obtained from edge detectors or corner detectors. In early work in the area, blob detection was used to obtain regions of interest for further processing. These regions could signal the presence of objects or parts of objects in the image domain with application to object recognition and/or object tracking. In other domains, such as histogram analysis, blob descriptors can also be used for peak detection with application to segmentation. Another common use of blob descriptors is as main primitives for texture analysis and texture recognition. In more recent work, blob descriptors have found increasingly popular use as interest points for wide baseline stereo matching and to signal the presence of informative image features for appearance-based object recognition based on local image statistics. There is also the related notion of ridge detection to signal the presence of elongated objects.

Feature Detection

Detection of interest points

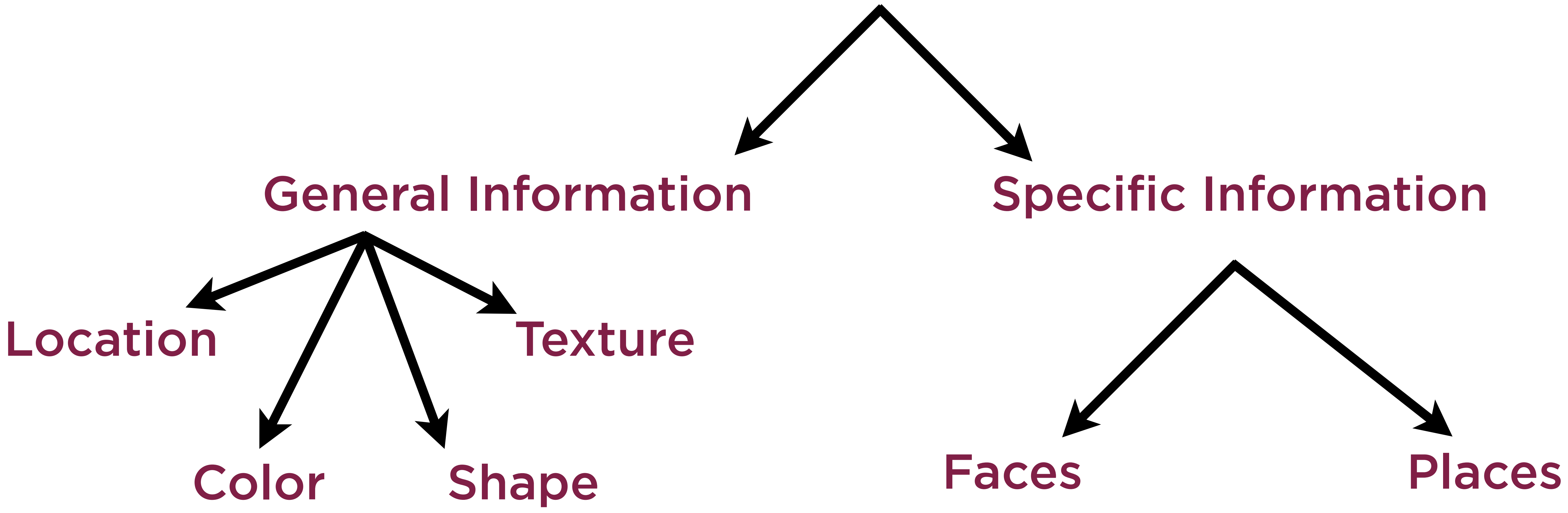
Detection of blobs (areas) of interest

Computation of image descriptors

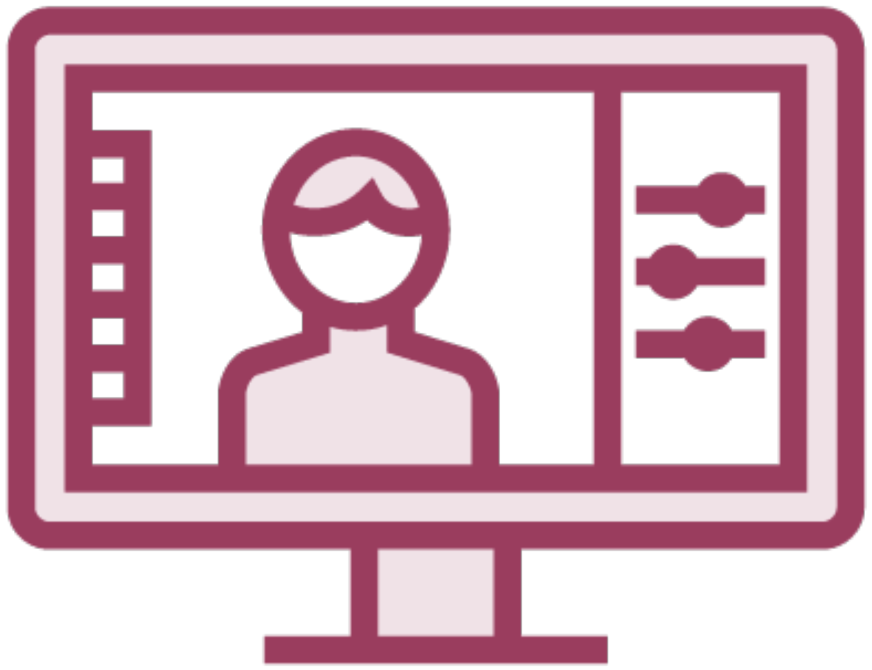
Image Descriptors

Descriptions of key features of images, such as shape, color, texture (and motion, in the case of videos)

Image Descriptors



Good Image Descriptors



Should be independent of position of associated key points

Should be robust to transformations

Should be scale independent

Image Descriptors

So, **here come descriptors**: they are the way to compare the keypoints. They summarize, *in vector format* (of constant length) some characteristics about the keypoints. For example, it could be their intensity in the direction of their most pronounced orientation. **It's assigning a numerical description to the area of the image the keypoint refers to.**

Some important things for descriptors are:

- they should be **independent of keypoint position**

  If the same keypoint is extracted at different positions (e.g. because of translation) the descriptor should be the same.

- they should be **robust against image transformations**

  Some examples are changes of contrast (e.g. image of the same place during a sunny and cloudy day) and changes of perspective (image of a building from center-right and center-left, we would still like to recognize it as a same building).

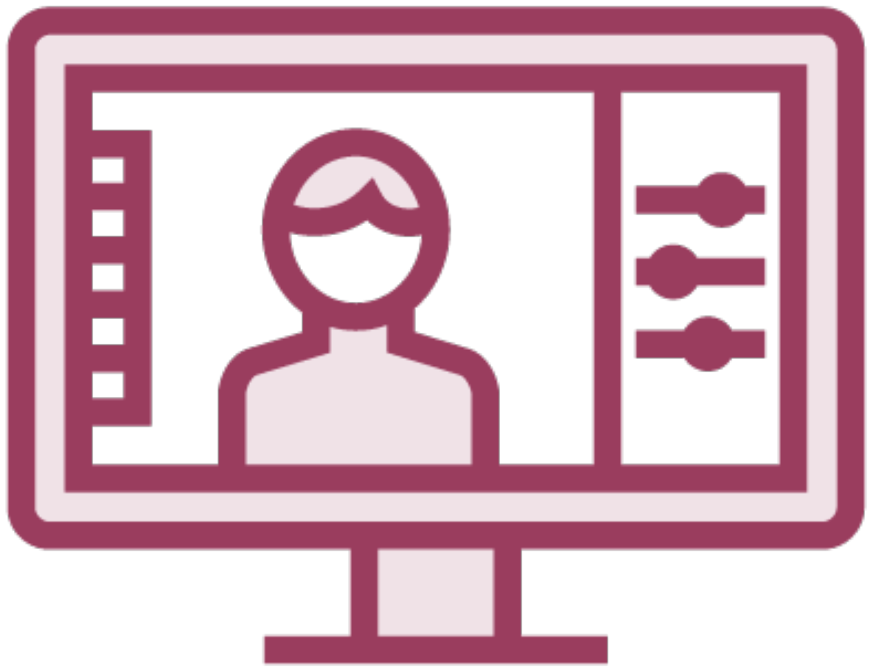
  Of course, no descriptor is completely robust against all transformations (nor against any single one if it is strong, e.g. big change in perspective).

  Different descriptors are designed to be robust against different transformations which is sometimes opposed to the speed it takes to calculate them.

- they should be **scale independent**

  The descriptors should take scale in to account. If the "prominent" part of the one keypoint is a vertical line of 10px (inside a circular area with radius of 8px), and the prominent part of another a vertical line of 5px (inside a circular area with radius of 4px) -- these keypoints should be assigned similar descriptors.

Image Descriptors for Feature Matching



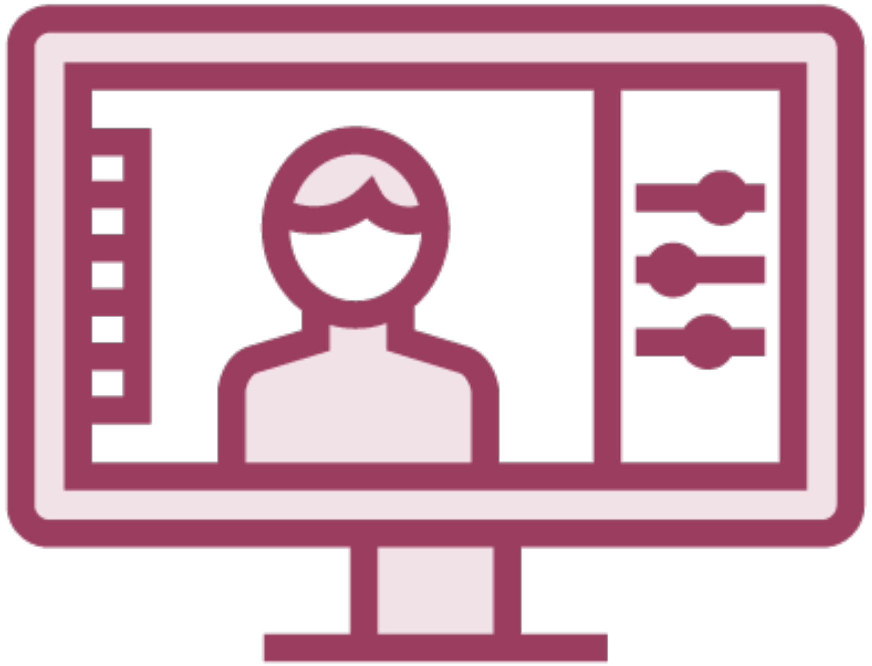
Descriptors are vectors of numbers

Help compare key points across images

Can use distance measures to compare

Key points whose descriptors have the smallest distances are matches

Image Descriptors



*S*cale *I*nvariant *F*eature *T*ransform (SIFT)

DAISY descriptors

Feature Extraction From Text

```
d = “This is not the worst restaurant in the metropolis,
not by a long way”
```

Document as Word Sequence

Model a document as an ordered sequence of words

```
d = “This is not the worst restaurant in the metropolis,
not by a long way”
```

```
(“This”, “is”,”not”,”the”,”worst”,”restaurant”,”in”,”the”,
“metropolis”, “not”,”by”,”a”,”long”,”way”)
```

Document as Word Sequence

Tokenize document into individual words

x_0 ⟶ Some numeric encoding of word

w_0 ⟶ Word (text)

(“This”, “is”,”not”,”the”,”worst”,”restaurant”,”in”,”the”, “metropolis”, “not”,”by”,”a”,”long”,”way”)

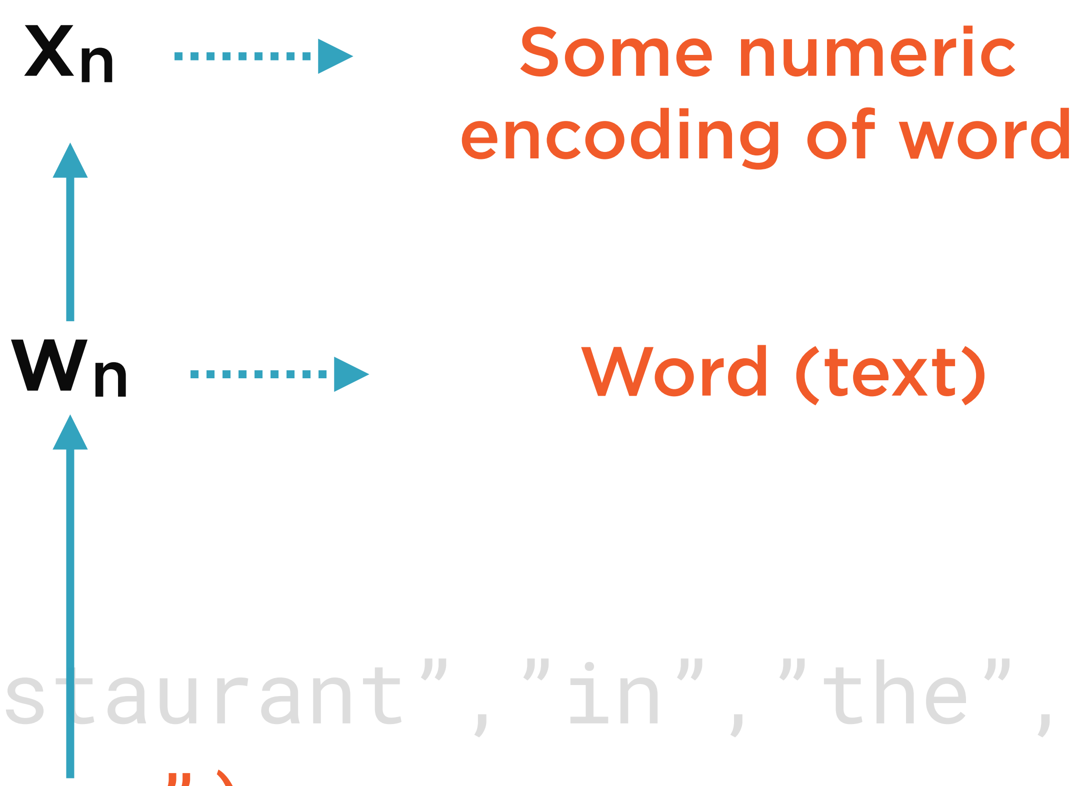
Represent Each Word as a Number

x_1 Some numeric encoding of word

w_1 Word (text)

```
("This", "is","not","the","worst","restaurant","in","the",
"metropolis", "not","by","a","long","way")
```

Represent Each Word as a Number



(“This”, “is”,”not”,”the”,”worst”,”restaurant”,”in”,”the”,
“metropolis”, “not”,”by”,”a”,”long”,”way”)

Represent Each Word as a Number

$$d = [x_0,\ x_1,\ \ldots\ x_n]$$

Document as Tensor

Represent each word as numeric data, aggregate into tensor

$x_i = [?]$

The Big Question

How best can words be represented as numeric data?

```
d = [[?], [?], … [?]]
```

The Big Question

How best can words be represented as numeric data?

Word Embeddings

One-hot

Frequency-based

Prediction-based

Word Embeddings

One-hot

Frequency-based

Prediction-based

One-hot encoding does NOT capture any semantic information or relationship between words

Word Embeddings

One-hot

Frequency-based

Prediction-based

Frequency-based Embeddings

Count

TF-IDF

Co-occurrence

Frequency-based Embeddings

Count

TF-IDF

Co-occurrence

Capture how often a word occurs in a document i.e. the **counts** or the **frequency**

Frequency-based Embeddings

Count

TF-IDF

Co-occurrence

Captures how often a word occurs in a **document** as well as the **entire corpus**

Tf-Idf

Frequently in a single document

Might be important

Frequently in the corpus

Probably a common word like "a", "an", "the"

Frequency-based Embeddings

Count

TF-IDF

Co-occurrence

Similar words will occur together and will have similar context

Word Embeddings

One-hot

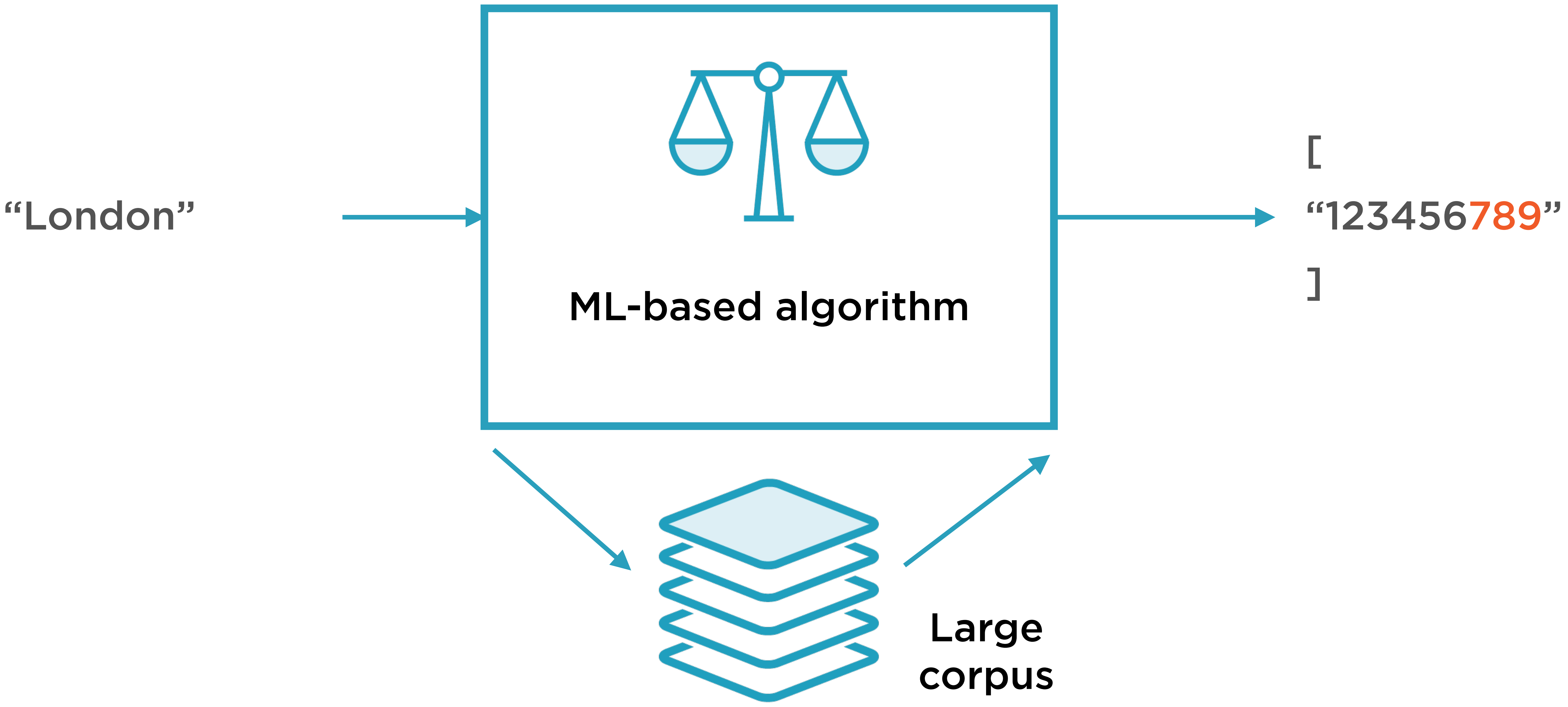
Frequency-based

Prediction-based

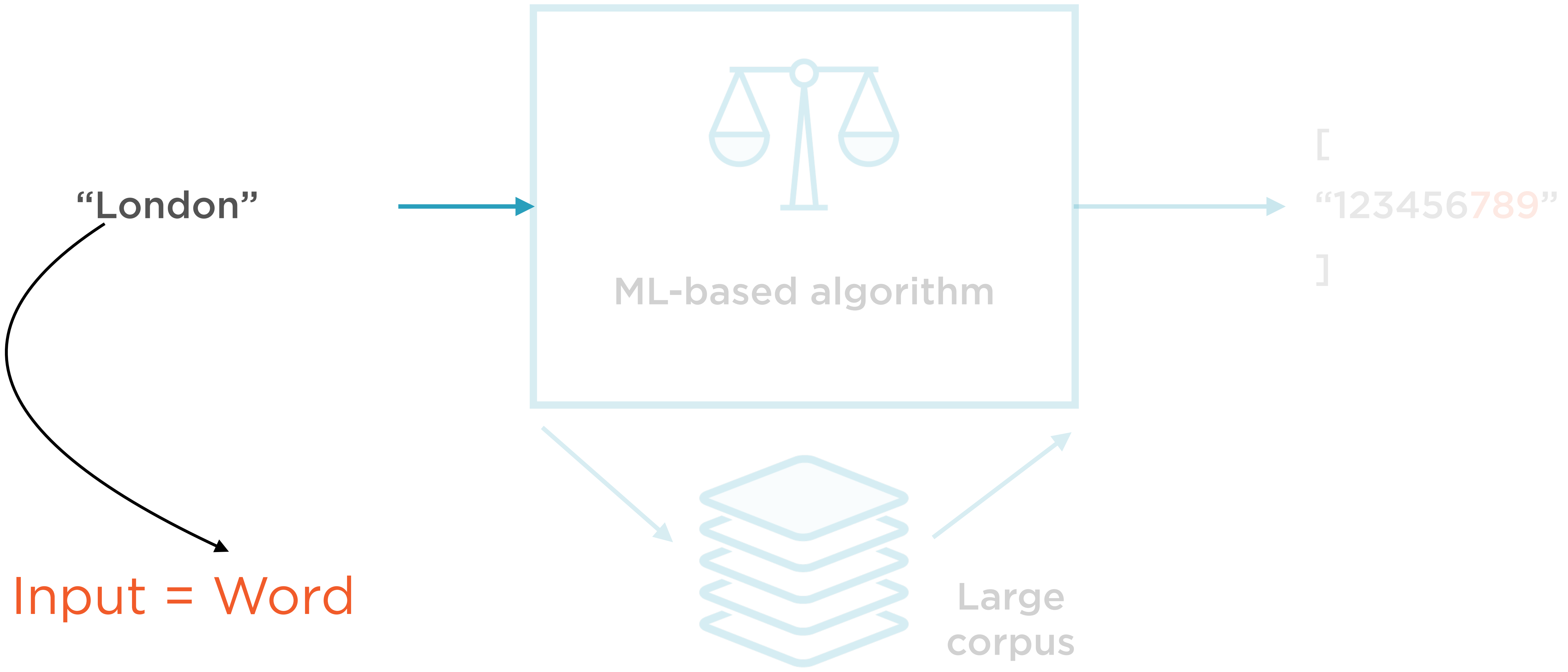
Word Embeddings

Numerical representations of text which capture meanings and semantic relationships

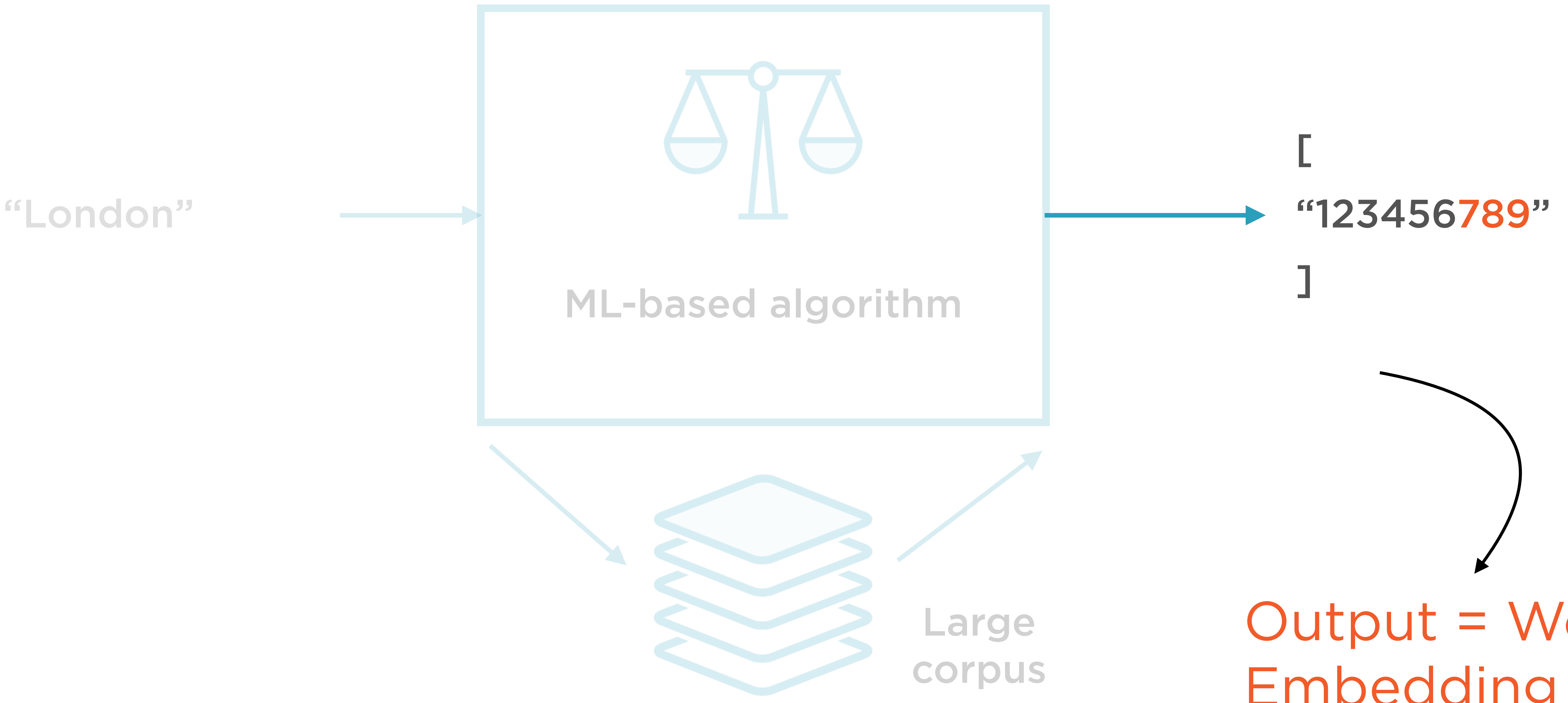
Prediction-based Word Embeddings



Prediction-based Word Embeddings

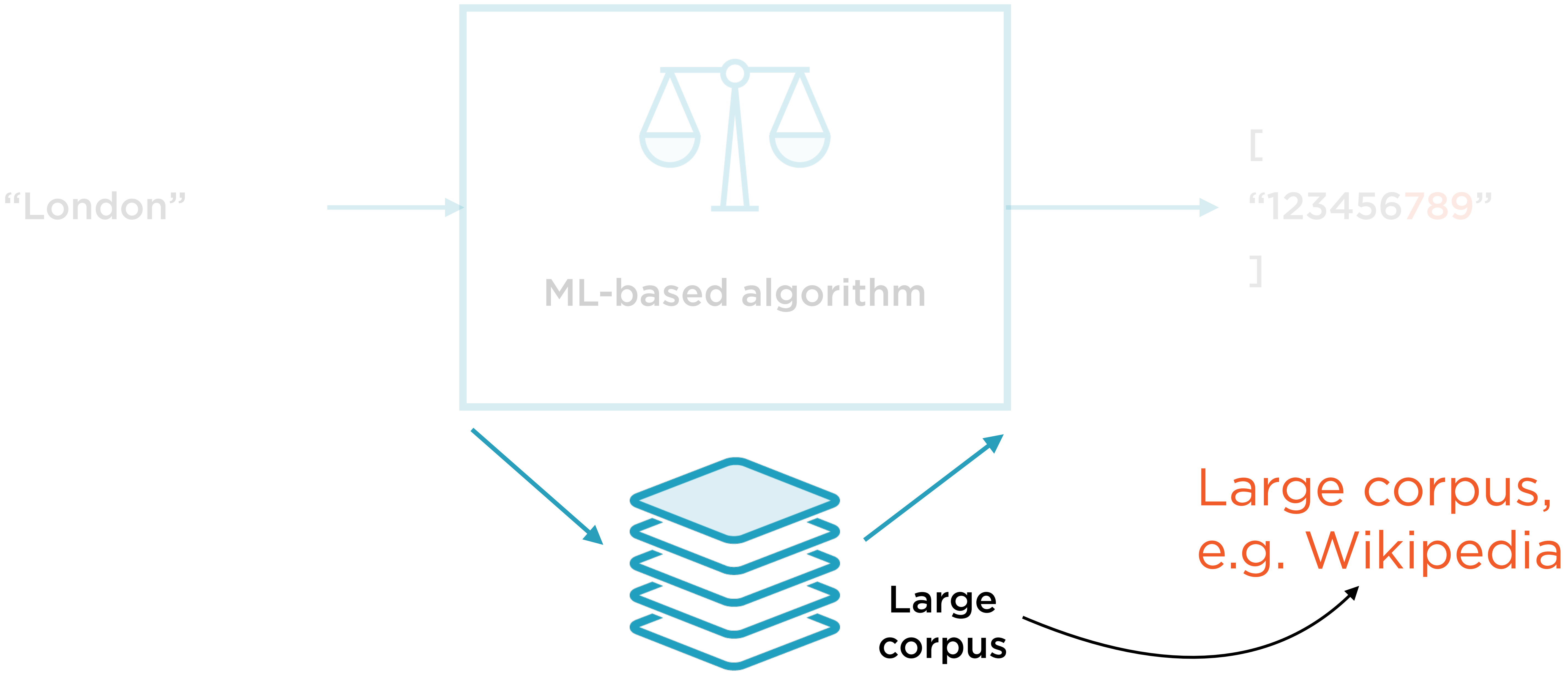


Prediction-based Word Embeddings



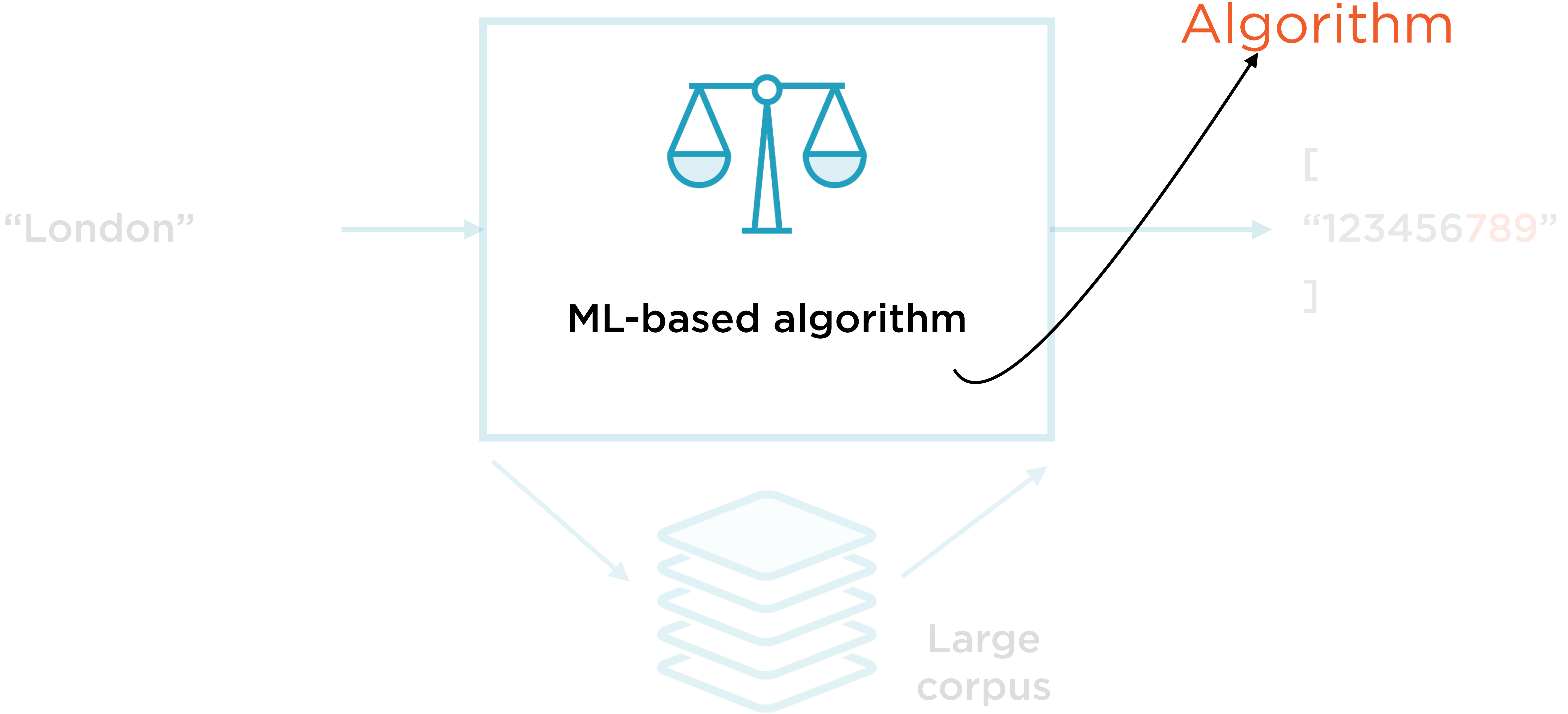
Output = Word Embedding

Prediction-based Word Embeddings



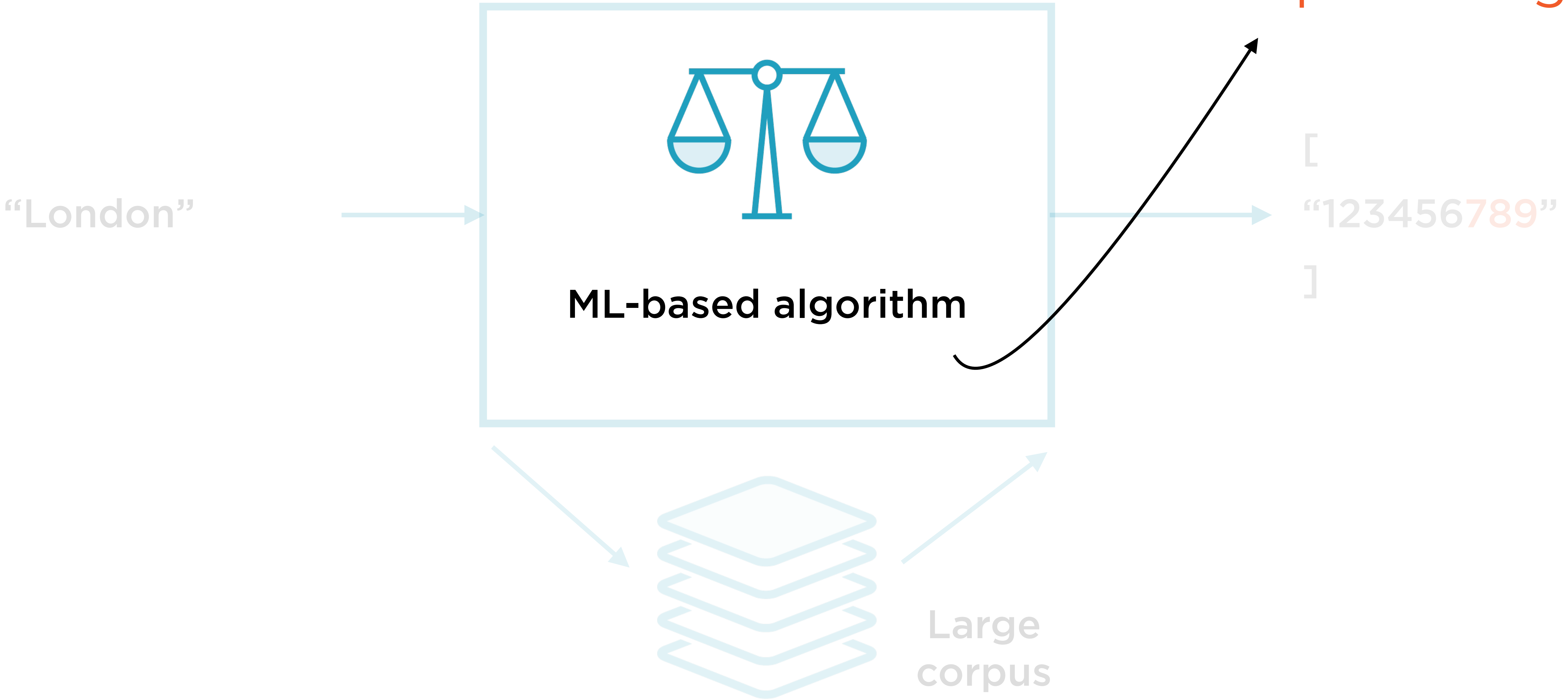
Prediction-based Word Embeddings

Unsupervised
Learning
Algorithm



Prediction-based Word Embeddings

Can use NN,
deep learning



Summary

Representing images as numeric data

Extracting features from images

Representing text as numeric data

Extracting features from text

Feature Extraction for all Data Types

Geospatial data

Time series

Text in images

Date and time