Introductory seminar on Computer Vision

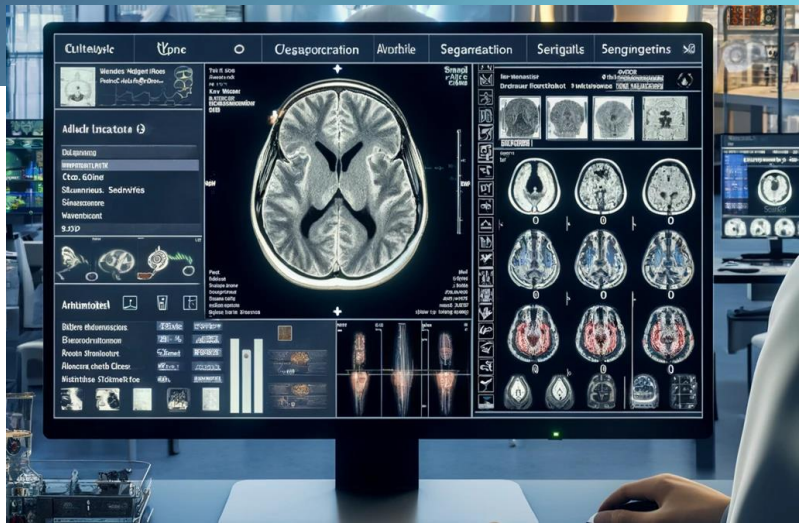
Sara Concas, Cagliari Digital Lab 2024 - Day 2

sara.concas90c@unica.it

Image Classification

Process where an algorithm assigns a label or category to an image based on its visual content.

Autonomous Vehicles - Traffic Sign Recognition

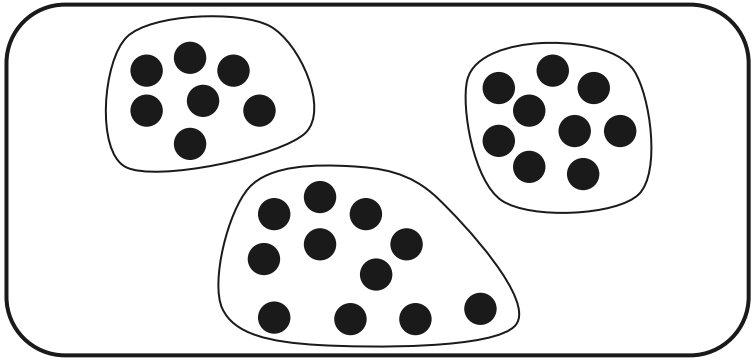


Medical Imaging

Manufacturing - Quality Control

Image Classification

- Unsupervised



- Supervised

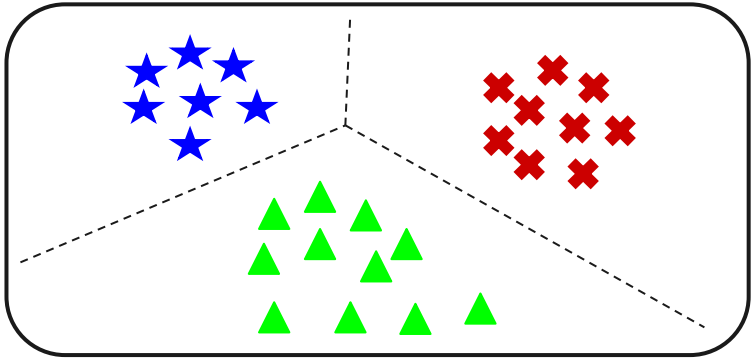


Image Classification - Main steps

1. Problem Definition
2. Data Collection
3. Data Annotation (Supervised techniques)
4. Data Preprocessing
5. Data Splitting
6. Model Selection
7. Model Training and Hyperparameter Tuning
8. Model Evaluation
9. Model Deployment, Monitoring and Maintenance
10. Documentation and Reporting

Image Classification - Data Collection

Gather a large number of images that are relevant to the classification task. Ensure that the dataset is diverse and representative

Purpose: define the objective of your computer vision task

Requirements: Identify the types of images and the specific details needed to achieve your objective

Source Identification: public dataset? Custom data?

Image Classification - Data Annotation

!!Only for Supervised Techniques!!

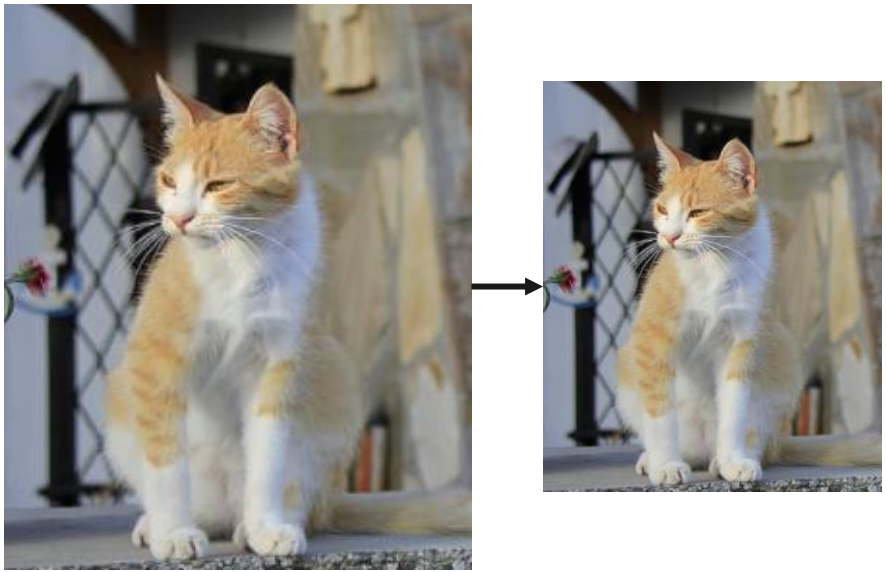
Each image in the dataset is labeled with the correct category or class.

Annotation Guidelines: define the categories or classes and establish rules for consistent annotation

Annotation Tools: manual or automated annotation?

Prepare the Dataset: organize the dataset structure and metadata

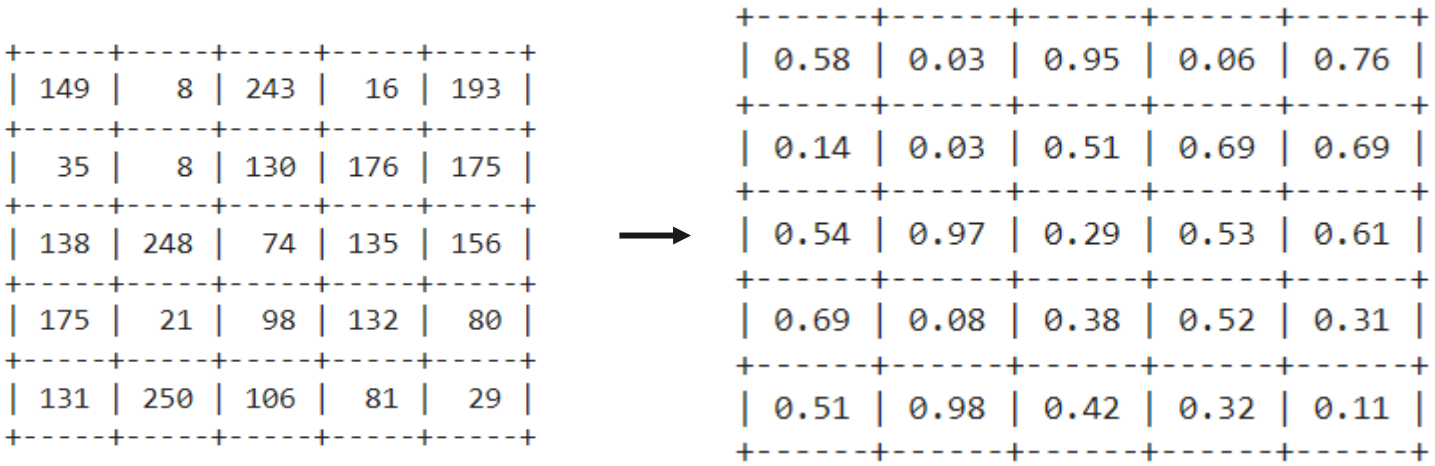
Image Classification - Data Preprocessing



Resizing : resize the images into a consistent size

All images in the dataset must have the same dimensions, which is a requirement for most machine learning models

Image Classification - Data Preprocessing



149	8	243	16	193
35	8	130	176	175
138	248	74	135	156
175	21	98	132	80
131	250	106	81	29

→

0.58	0.03	0.95	0.06	0.76
0.14	0.03	0.51	0.69	0.69
0.54	0.97	0.29	0.53	0.61
0.69	0.08	0.38	0.52	0.31
0.51	0.98	0.42	0.32	0.11

Normalization: normalize pixel values (for example in the 0-1 range)

Standardizing the range of pixel values in the images ensures consistent input for the model. This helps improve the performance and convergence of machine learning models

Image Classification - Data Preprocessing

Augmentation: increase the the number and diversity of images by applying techniques such as rotation, flipping, zooming, and shifting etc.

Artificially expanding the size of the dataset, helps improve the robustness and generalization capability of machine learning models

Image Classification - Data Splitting

The dataset is divided into subsets:

- **Training**: data used for training the model (typically 60-80%)
- **Validation**: data used for tuning hyperparameters and model selection (tipically 10-20%)
- **Testing**: data used for evaluating the final model performance (tipically 10-20%)

Attention!!! The test set should NEVER be used during the training phase!

Image Classification - Model Selection

Choose the most suitable model based on the problem at hand. Consider:

- **Type of Classification**: is it a binary or a multi-label problem? Are the labels available? (Supervised or unsupervised setting)

- **Dataset Size**: large datasets may need more complex models, while smaller datasets may benefit from simpler models

- **Available resources**: do you have access to a GPU? How much memory do you have available?

Image Classification - Model Training

After choosing the most suitable model, it is necessary to train it on the available dataset.

- Feed the preprocessed data into our model
- Choose the best hyperparameters:
  - Grid Search
  - Random Search

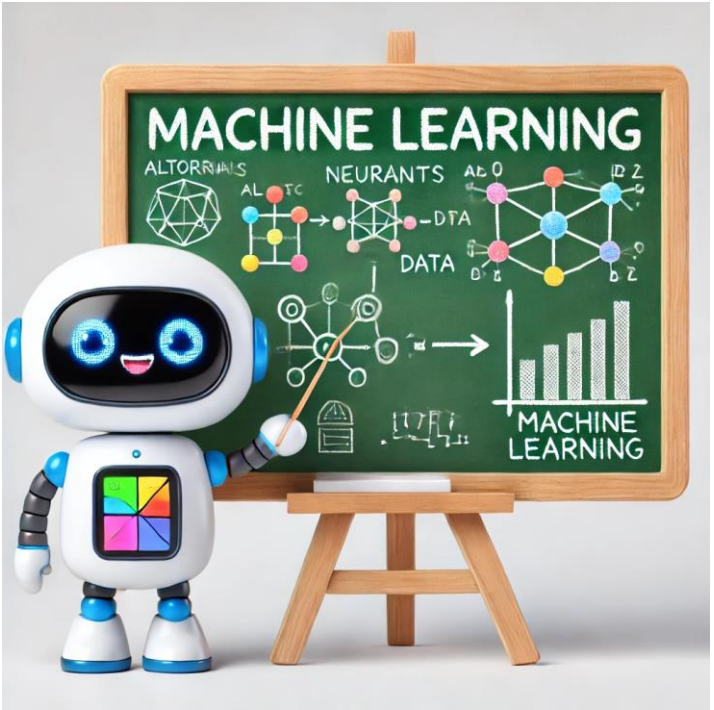


Image Classification - Model Evaluation

Assess the model performance on unseen data.

It is possible to use several metrics:

- **Accuracy**: The proportion of correctly classified instances among the total instances

$$\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

TP (True Positives): The number of instances correctly predicted as positive.
TN (True Negatives): The number of instances correctly predicted as negative.
FP (False Positives): The number of instances incorrectly predicted as positive.
FN (False Negatives): The number of instances incorrectly predicted as negative.

Image Classification - Model Evaluation

Assess the model performance on unseen data.

It is possible to use several metrics:

- **Confusion Matrix**: *nxn* matrix, where *n* is the number of classes. Each cell in the matrix $C_{i,j}$, represents the number of instances of class *i* that were predicted as class *j*.

Actual\Predicted	Class 1	Class 2	⋯	Class n
Class 1	$C_{1,1}$	$C_{1,2}$	⋯	$C_{1,n}$
Class 2	$C_{2,1}$	$C_{2,2}$	⋯	$C_{2,n}$
⋮	⋮	⋮	⋱	⋮
Class n	$C_{n,1}$	$C_{n,2}$	⋯	$C_{n,n}$

Image Classification - Model Evaluation

Assess the model performance on unseen data.

It is possible to use several metrics:

- **Precision**: Precision for a class i is the number of true positives (correctly predicted instances of class i) divided by the total number of instances that were predicted to belong to class i (true positives + false positives): ***Of all instances predicted as class i, how many actually belong to class i?***

$$\text{Precision}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i}$$

  - True Positives_i: number of instances that were correctly predicted as belonging to class i.
  - False Positives_i: number of instances that were incorrectly predicted as belonging to class i (i.e., the instances that do not actually belong to class i but were predicted as such).

Image Classification - Model Evaluation

Assess the model performance on unseen data.

It is possible to use several metrics:

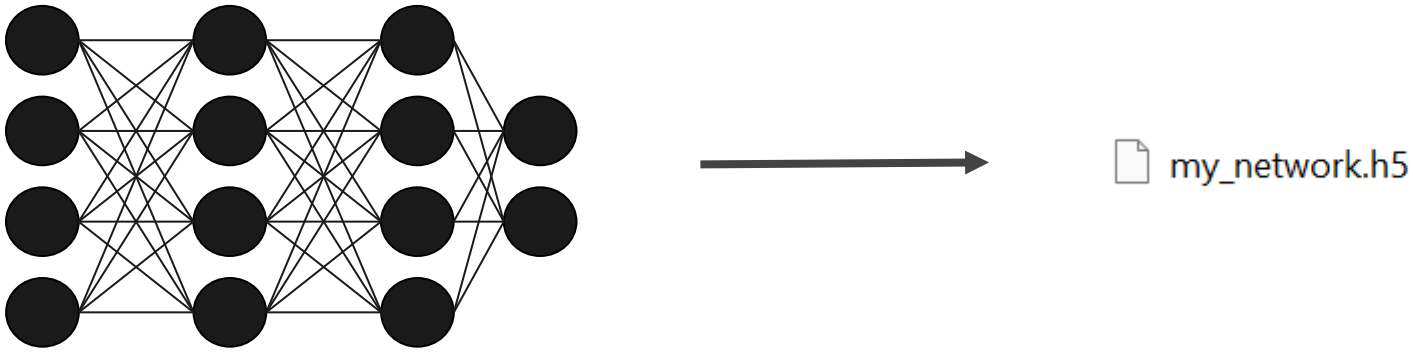
- **Recall**: Recall for a class i is the number of true positives (correctly predicted instances of class i) by the total number of instances that actually belong to class i (true positives + false negatives): **Of all the actual instances of class i, how many were correctly classified as class i?**

$$\text{Recall}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i}$$

- True Positives_i: number of instances that were correctly predicted as belonging to class i.
- False Negatives_i: number of instances that actually belong to class i but were incorrectly predicted as not belonging to class i.

Image Classification - Model Deployment

Save the trained model and make it available to make predictions on new data.



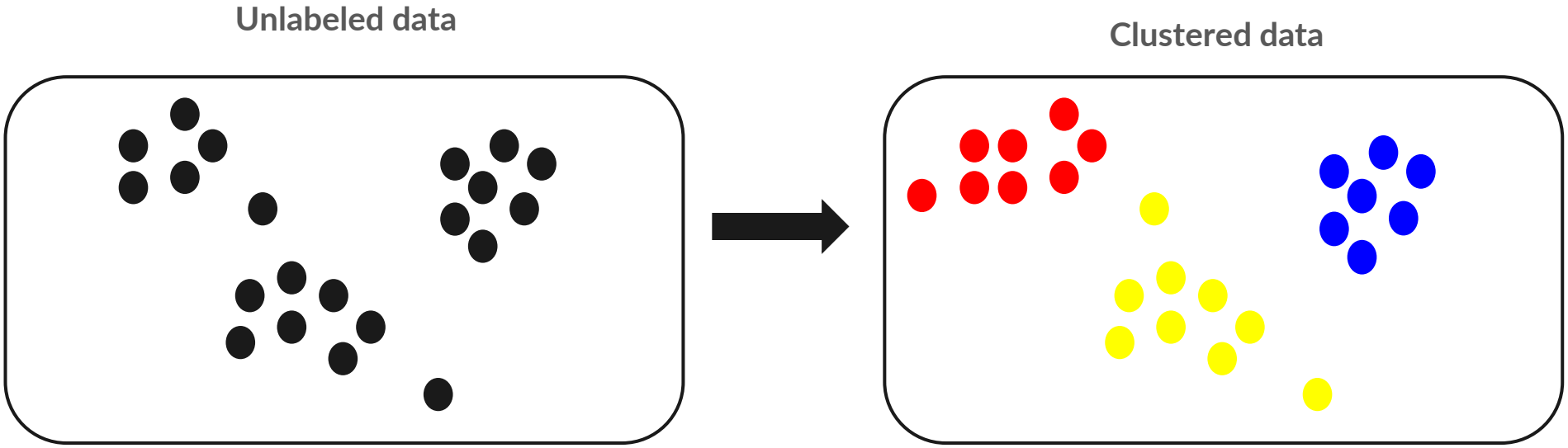
There are various formats for saving, the right format depends on your deployment environment and use case

Unsupervised Image Classification

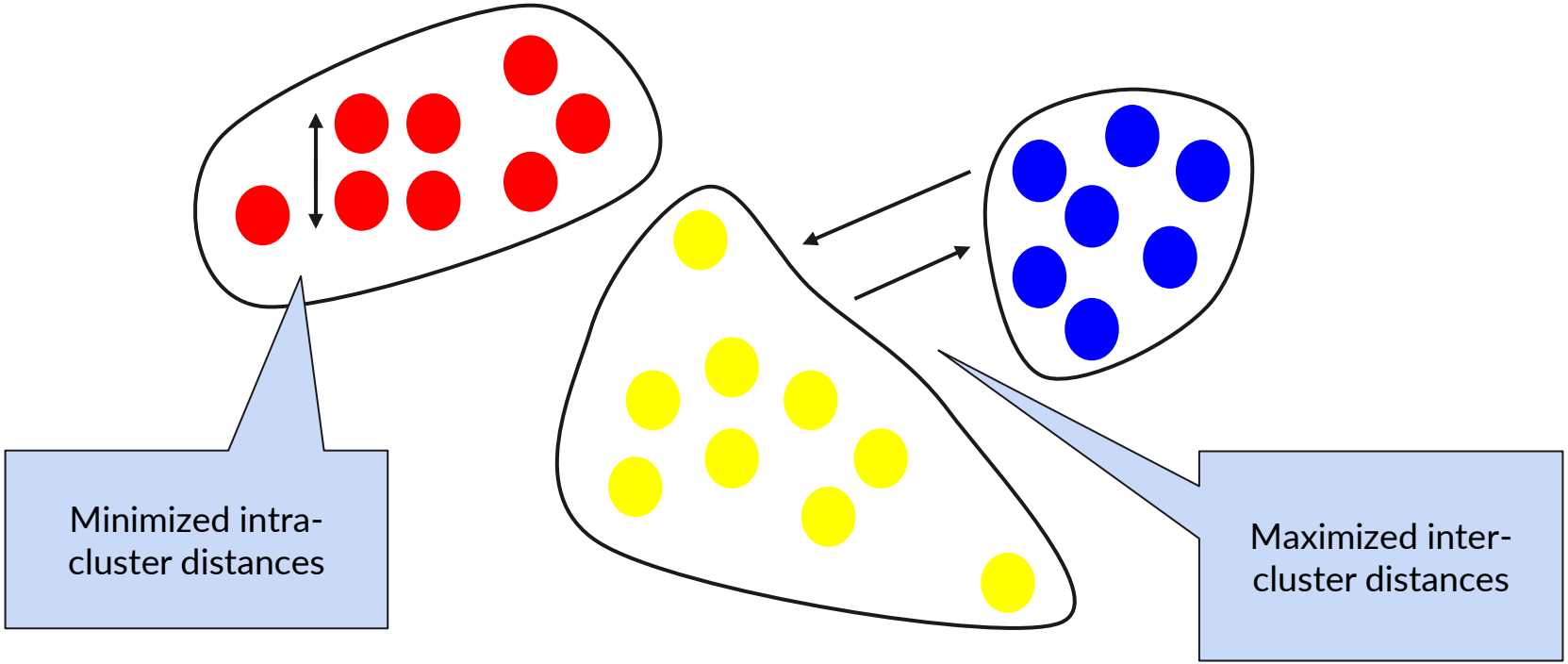
- The training data does not need to be labeled
- Analyze the inherent structures and patterns within the image data to identify clusters or groups

GOAL: explore and understand the nature of given data, identifying groups of objects or clusters, such that the members of each cluster are as similar as possible and dissimilar from those of the other clusters

Unsupervised Image Classification - Clustering



Unsupervised Image Classification - Clustering



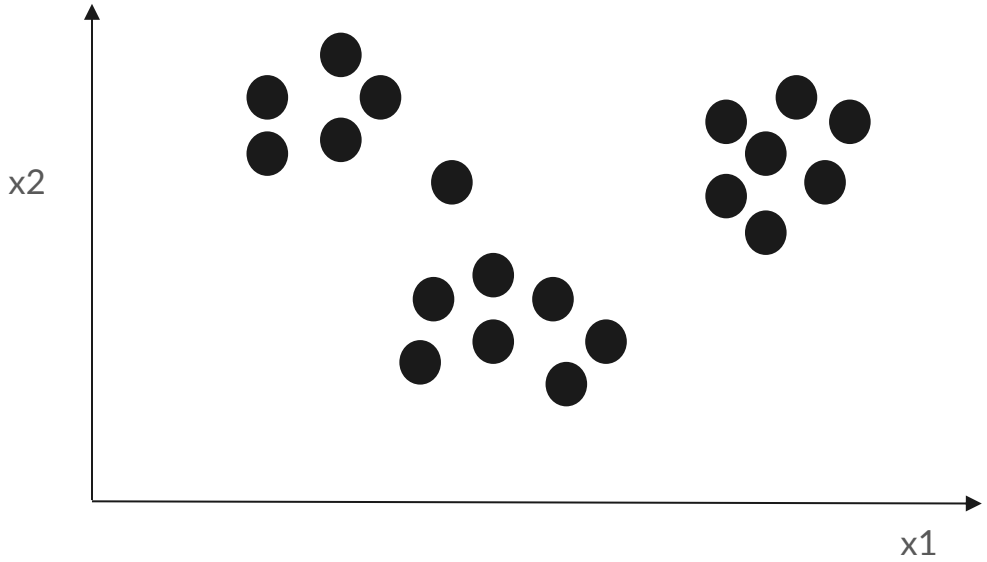
Unsupervised Image Classification – Methods (Some examples)

- Clustering-Based Methods
  - K-Means Clustering
  - Hierarchical Clustering
  - DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- Dimensionality Reduction-Based Methods
  - Principal Component Analysis (PCA)
  - Autoencoders
- Deep Learning-Based Methods
  - Convolutional Neural Networks (CNNs)
- Graph-Based Methods
- Self-Organizing Maps (SOMs)
- Hybrid Methods

Unsupervised Image Classification - K-means

GOAL: subdivide the dataset into a pre-defined number of clusters (group similar data points together and discover underlying patterns or structures within the data)

K = number of clusters we want to divide our data in

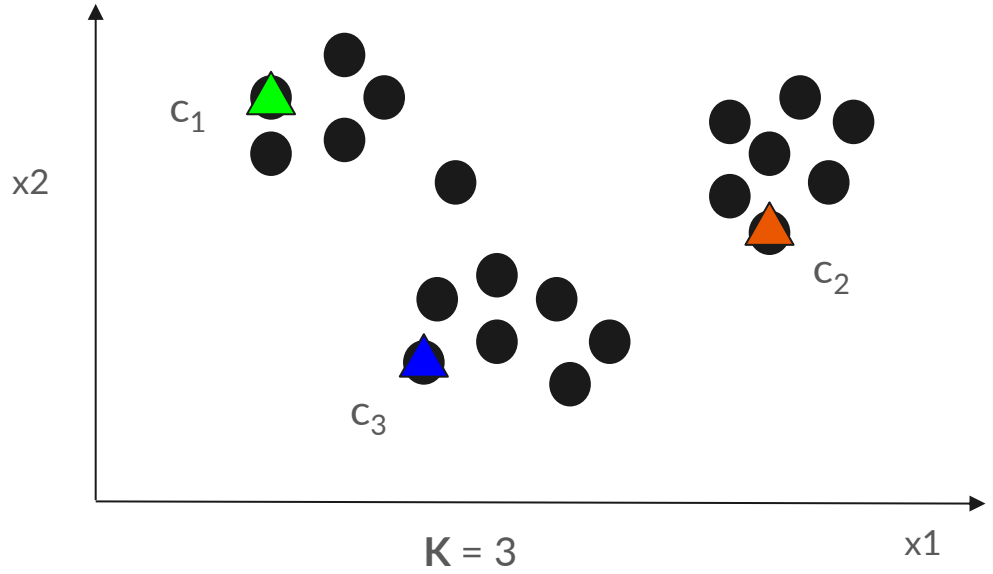


Unsupervised Image Classification - K-means

GOAL: subdivide the dataset into a pre-defined number of clusters (group similar data points together and discover underlying patterns or structures within the data).

1. Specify number of clusters K.

2. Randomly select K data points as centroids c_1, ..., c_k

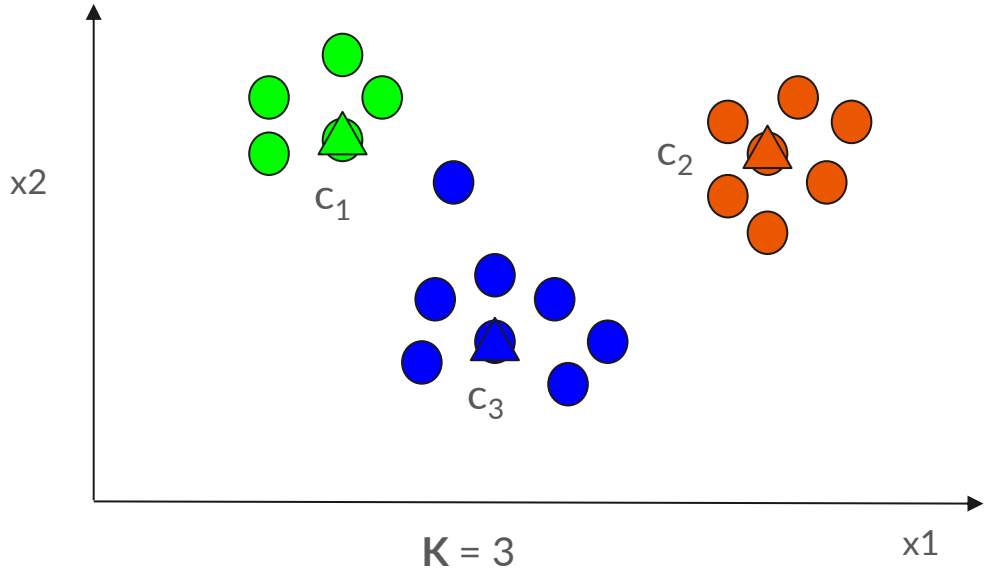


Unsupervised Image Classification - K-means

GOAL: subdivide the dataset into a pre-defined number of clusters (group similar data points together and discover underlying patterns or structures within the data).

3. Assign each point to the cluster related to the nearest centroid.

4. For each cluster compute the new centroid as the average of the points in each cluster.

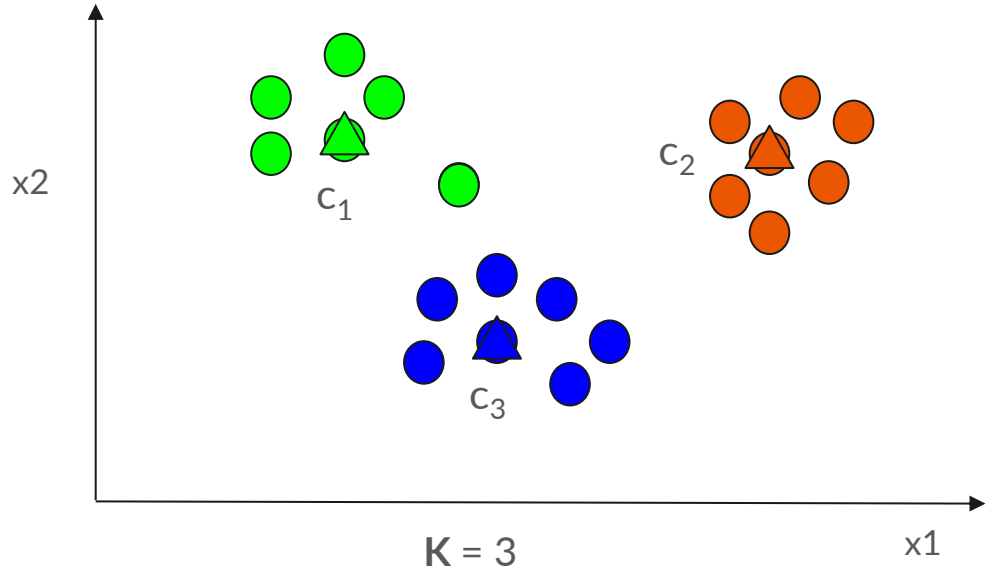


Unsupervised Image Classification - K-means

GOAL: subdivide the dataset into a pre-defined number of clusters (group similar data points together and discover underlying patterns or structures within the data).

5. Assign each data point to their new closest centroid of each cluster

6. If any reassignment occurs, then go to step-4 else go to FINISH.



Autoencoders

Type of artificial neural network used for unsupervised learning that aims to learn efficient codings of input data

GOAL: learn a compressed representation (encoding) of the input data and then reconstruct the input data from this compressed representation as accurately as possible.

It consists of two main parts:

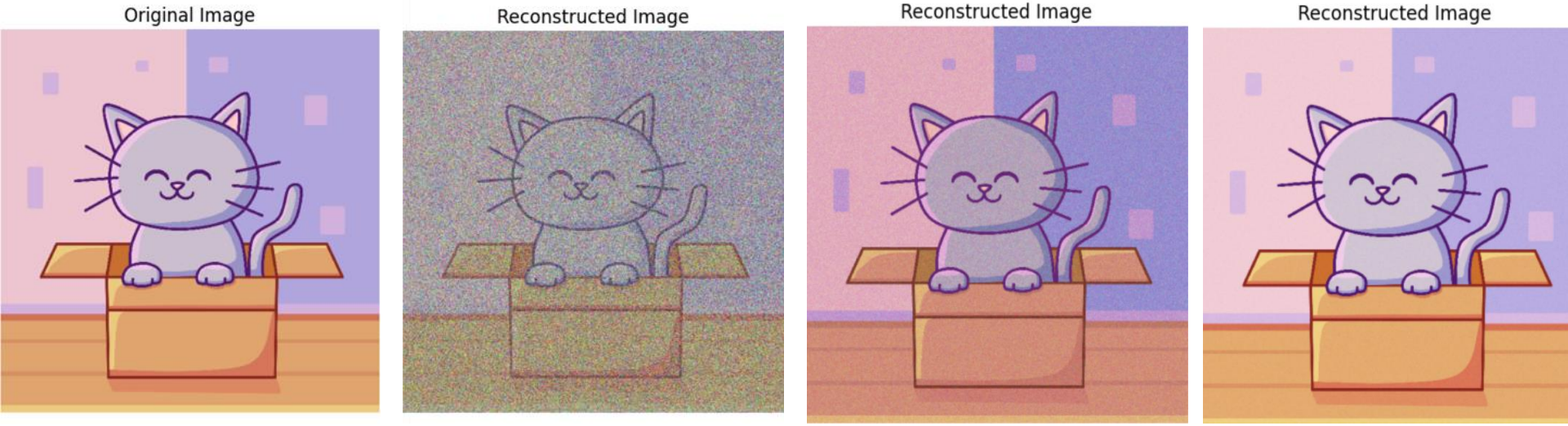
- **Encoder**: compresses the input data into a lower-dimensional representation
- **Decoder**: reconstructs the input data from the lower-dimensional representation.

Output: The reconstructed data is compared to the original data, and the network is trained to minimize the difference between them.

Autoencoder workflow

Autoencoders

Original and reconstructed image, using a different number of epochs to train the autoencoder



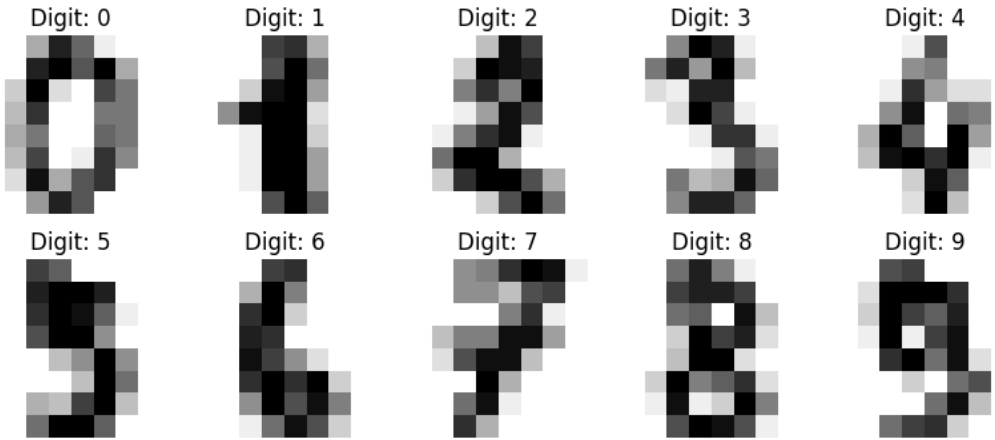
2 epochs

10 epochs

50 epochs

Dimensionality Reduction

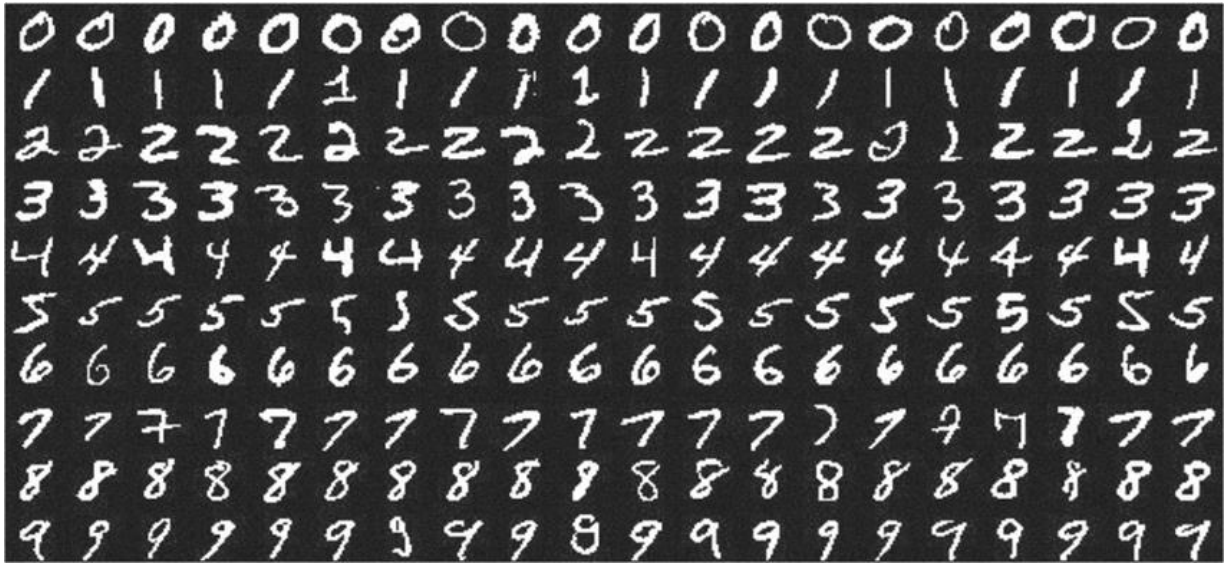
The Digit Dataset



https://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html

Unsupervised Image Classification - Example

MNIST Database of Handwritten Digits



https://bit.ly/4epu5U8

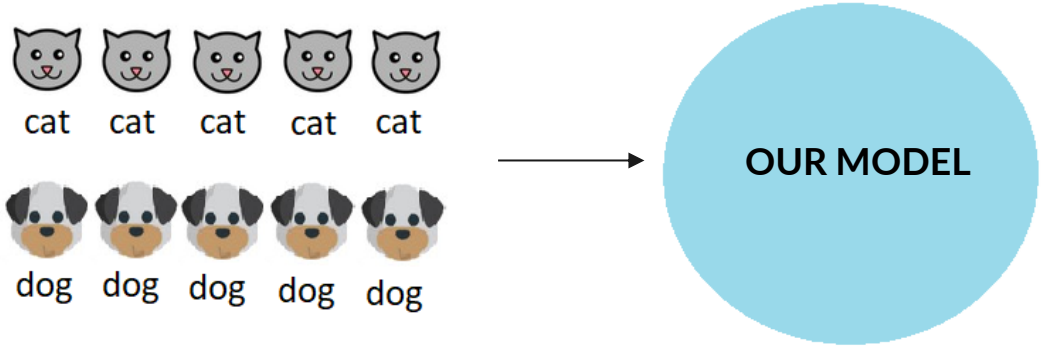
Supervised Image Classification

- The training data needs to be labeled
- the algorithm learns to recognize patterns and features within the images that correspond to specific categories based on the annotations provided in the training dataset

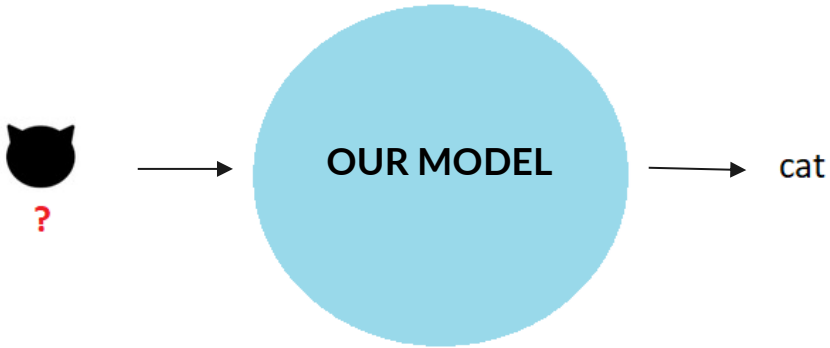
GOAL: create a model that can accurately classify new, unseen images based on the patterns learned during training

Supervised learning

Training



Testing



Supervised Image Classification – Methods (Some examples)

Traditional methods (feature extraction + classification)

- **Support Vector Machines (SVM)**
- K-Nearest Neighbors (KNN)
- Random Forests

Convolutional Neural Networks (CNNs) (classification)

- **Custom networks**
- **Pre-trained networks**

Supervised Image Classification - Example

We want to classify a set of images into cats and dogs.

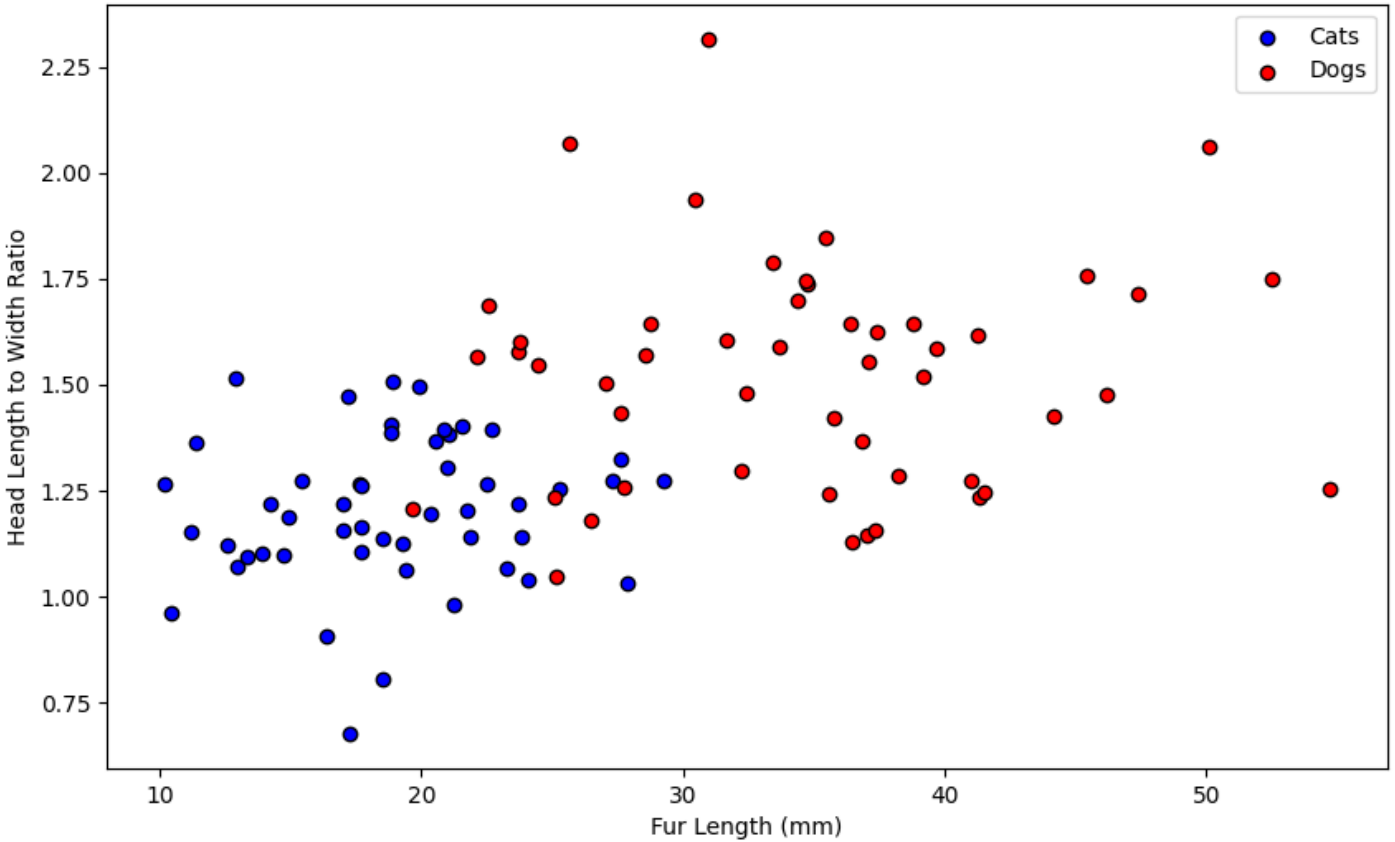
Possible discriminative features:

- **Fur Length**: generally, cats have shorter fur with smaller variability

- **Head Ratio**: cats typically have smaller head length-to-width ratio compared to dogs.

Supervised Image Classification - Example

We can use the two features and plot the samples into a plane.

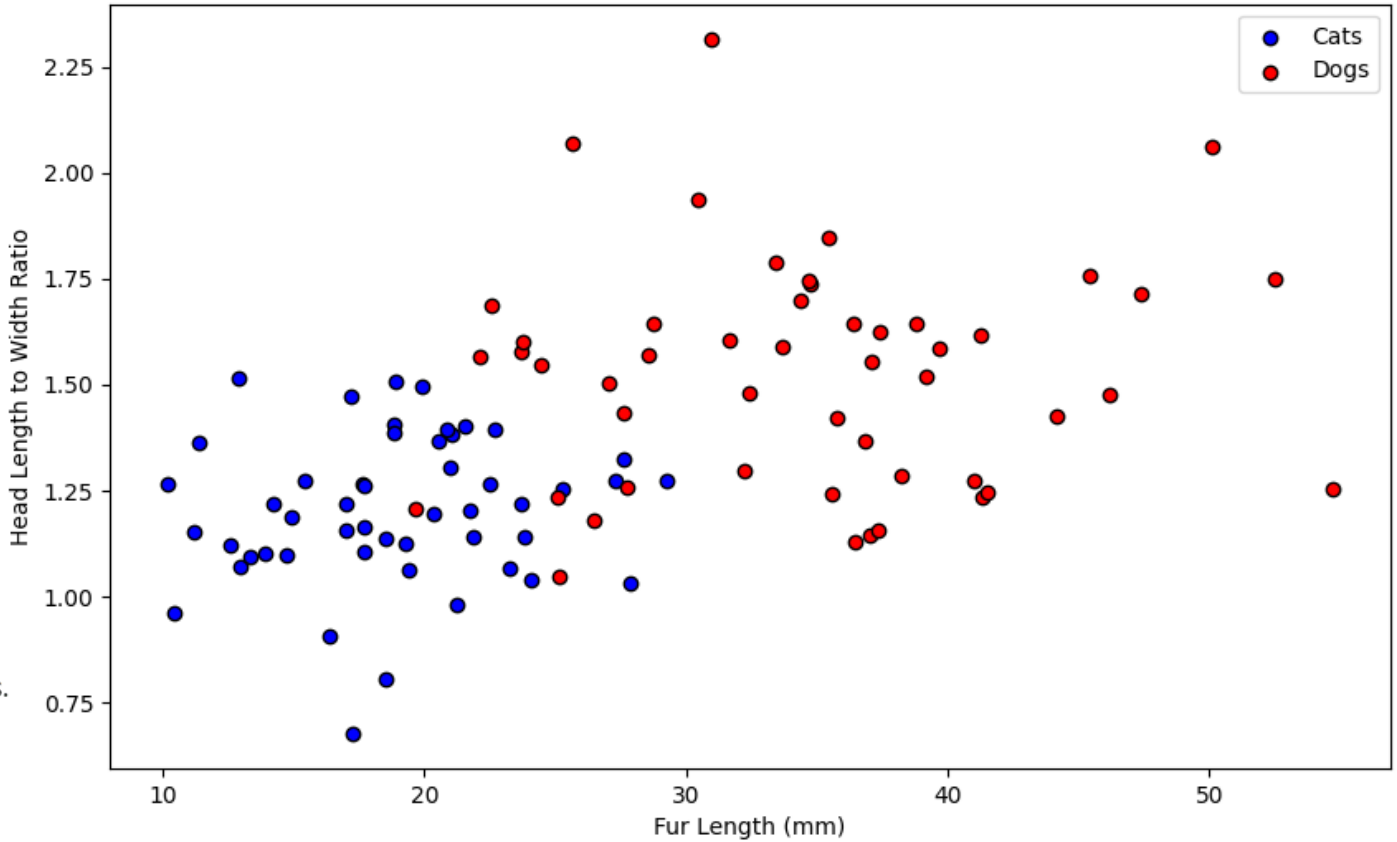


Supervised Image Classification - Example

Generic sample:

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}$$

- x_{i1} is the fur length of sample i, measured in millimeters.
- x_{i2} is the head length-to-width ratio for sample i.



Supervised Image Classification - Example

A sample with a head length to width ratio of 1.2 and a fur length of 25 mm. can be represented as:

$$\mathbf{x}_i = \begin{bmatrix} 25 \\ 1.3 \end{bmatrix}$$

The set of all samples can be represented as a matrix X , where each row corresponds to a different sample:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{pmatrix}$$

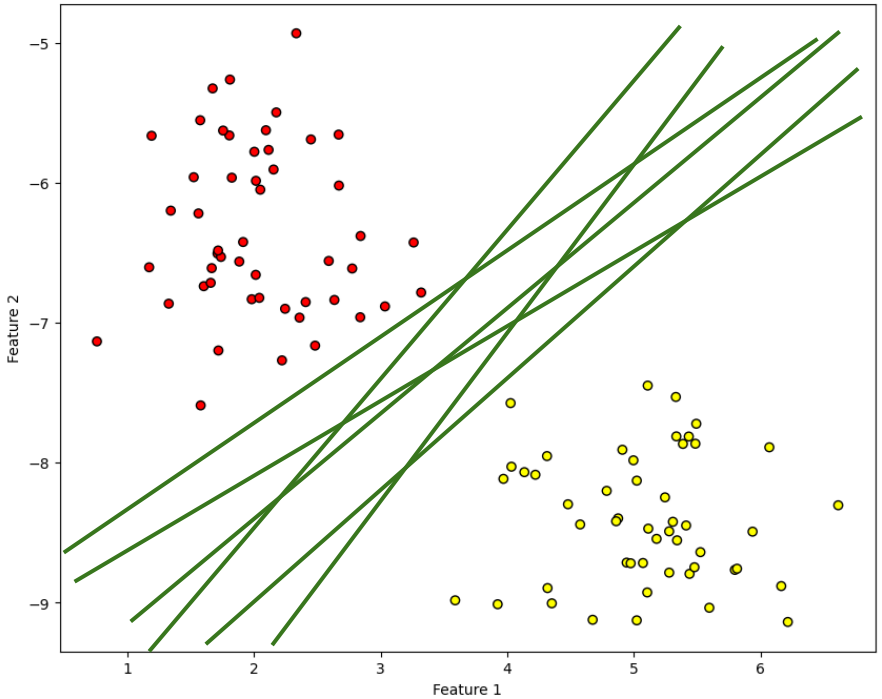
where n is the total number of samples. Each row in the matrix **X** corresponds to the feature vector of a specific sample.

Supervised Image Classification - Support Vector Machines (SVM)

GOAL: given a set of labeled training data, find the optimal hyperplane that correctly categorizes new samples.

There are many possible hyperplanes that can separate the classes of data points.

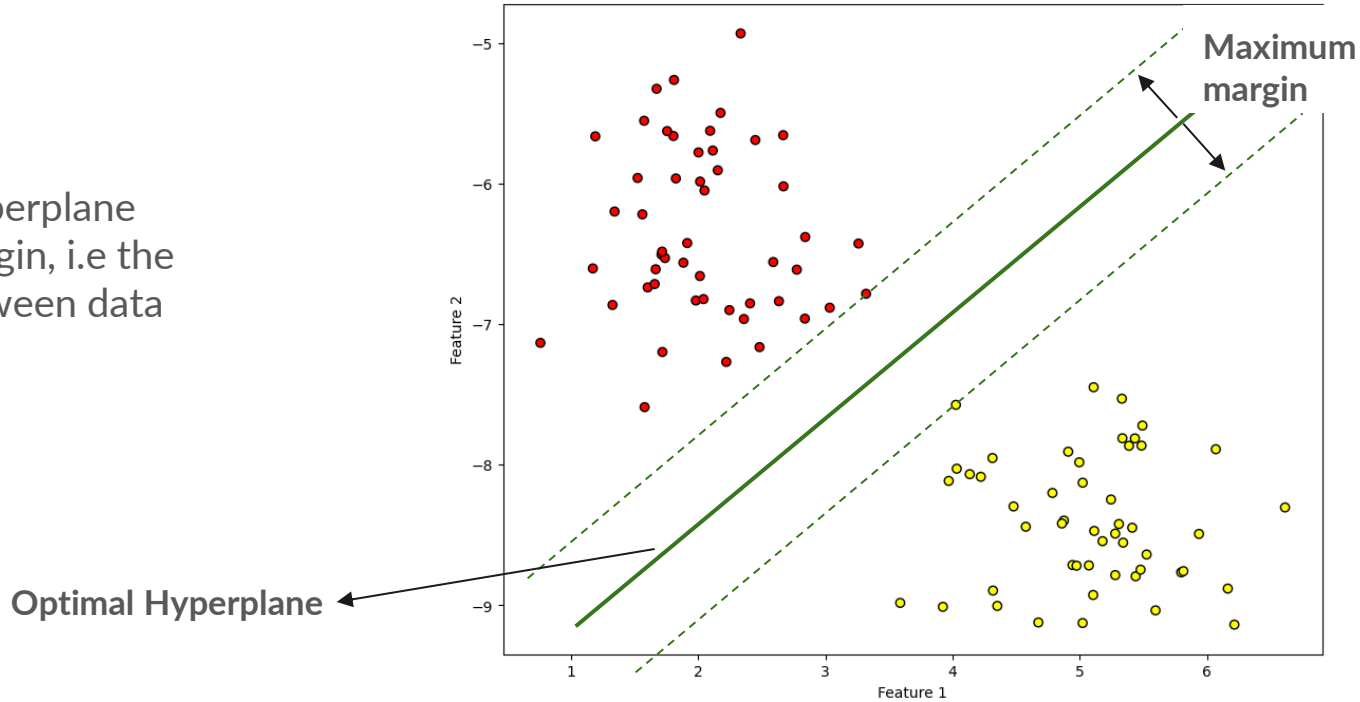
Which one is the optimal??



Supervised Image Classification - Support Vector Machines (SVM)

GOAL: given a set of labeled training data, find the optimal hyperplane that correctly categorizes new samples.

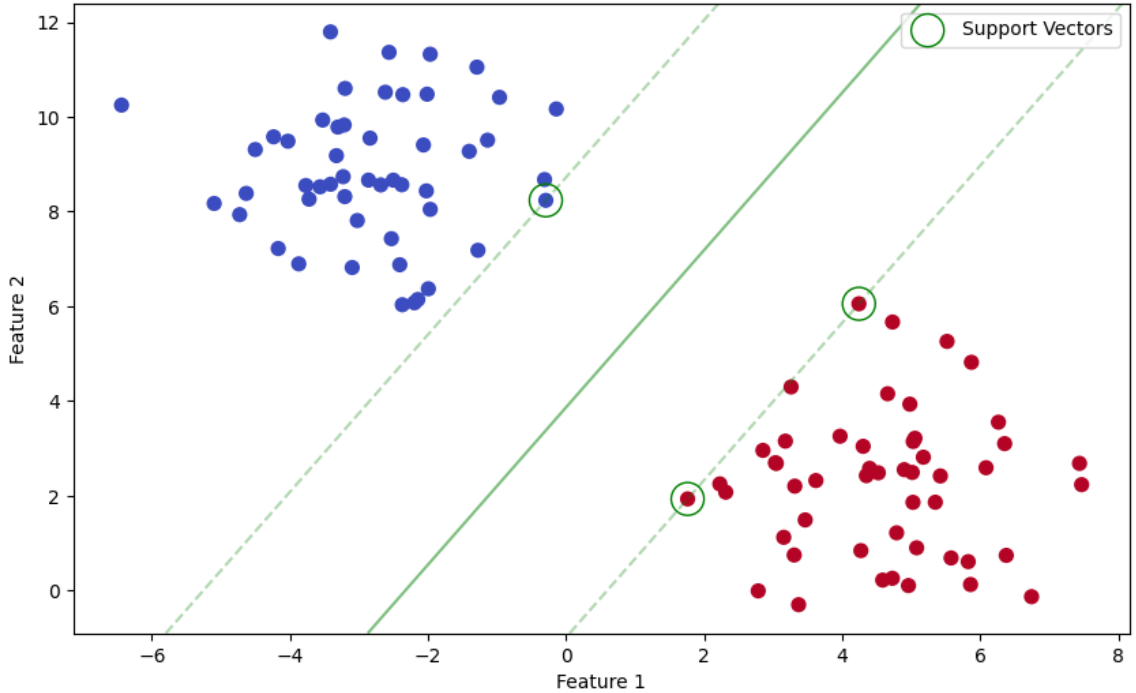
The goal is to find a hyperplane that maximizes the margin, i.e the maximum distance between data points of both classes.



Supervised Image Classification - Support Vector Machines (SVM)

GOAL: given a set of labeled training data, find the optimal hyperplane that correctly categorizes new samples.

The circled points represent the so-called **Support Vectors**. They are the data points that lie closest to the decision boundary and are instrumental in defining the optimal hyperplane that maximizes the margin between different classes.

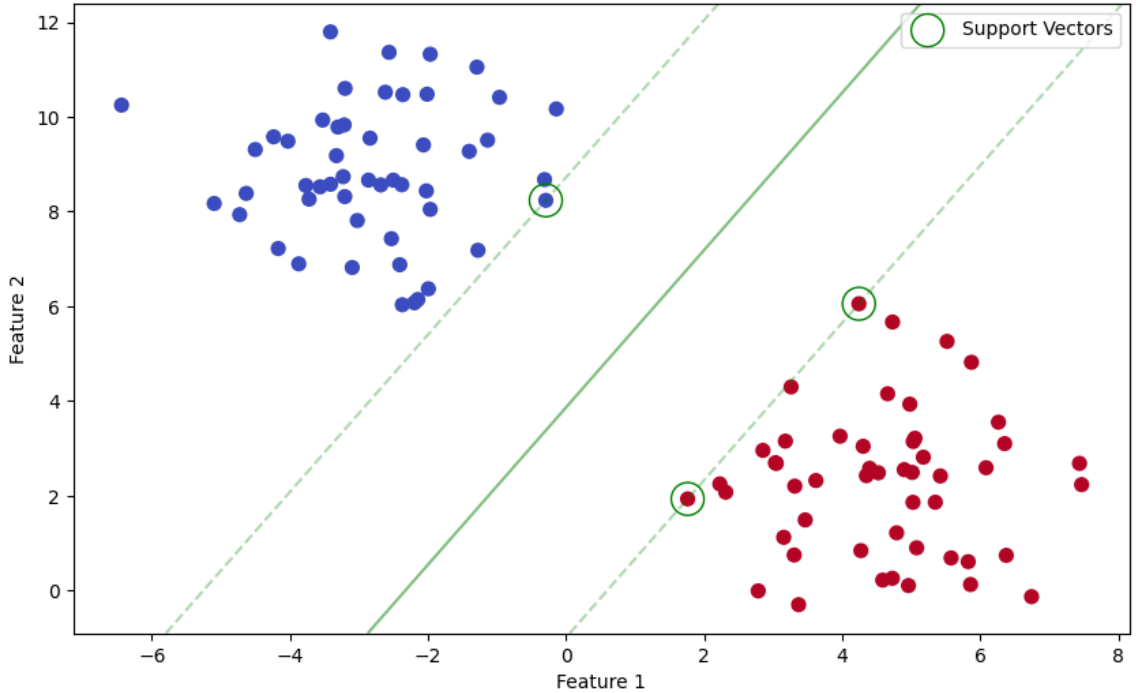


Supervised Image Classification - Support Vector Machines (SVM)

GOAL: given a set of labeled training data, find the optimal hyperplane that correctly categorizes new samples.

The decision hyperplane only depends on the Support Vectors

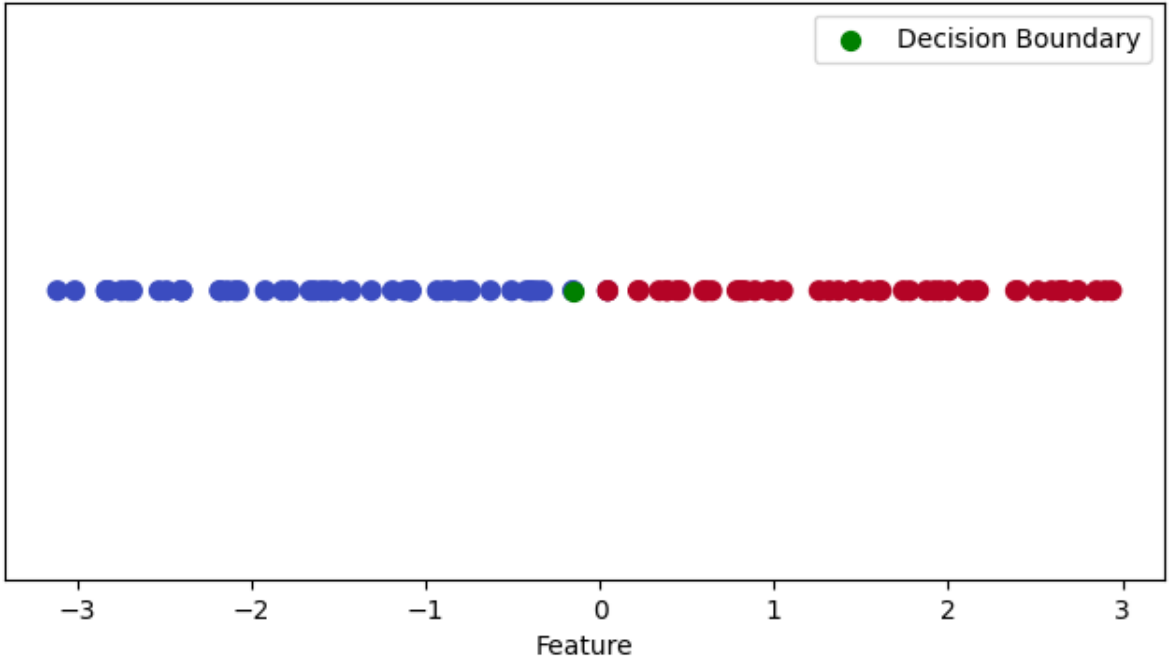
The other points can move without violating the margin



Supervised Image Classification - Support Vector Machines (SVM)

1-dimensional space

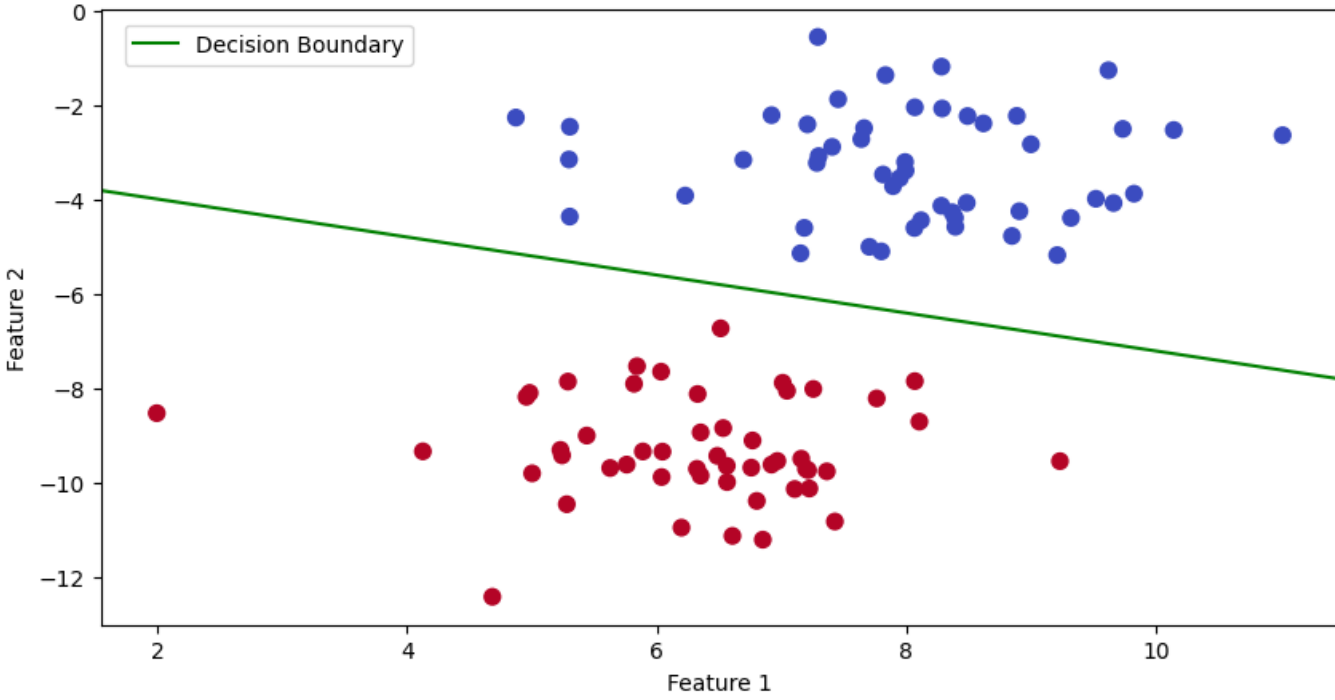
The hyperplane is just a point



Supervised Image Classification - Support Vector Machines (SVM)

2-dimensional space

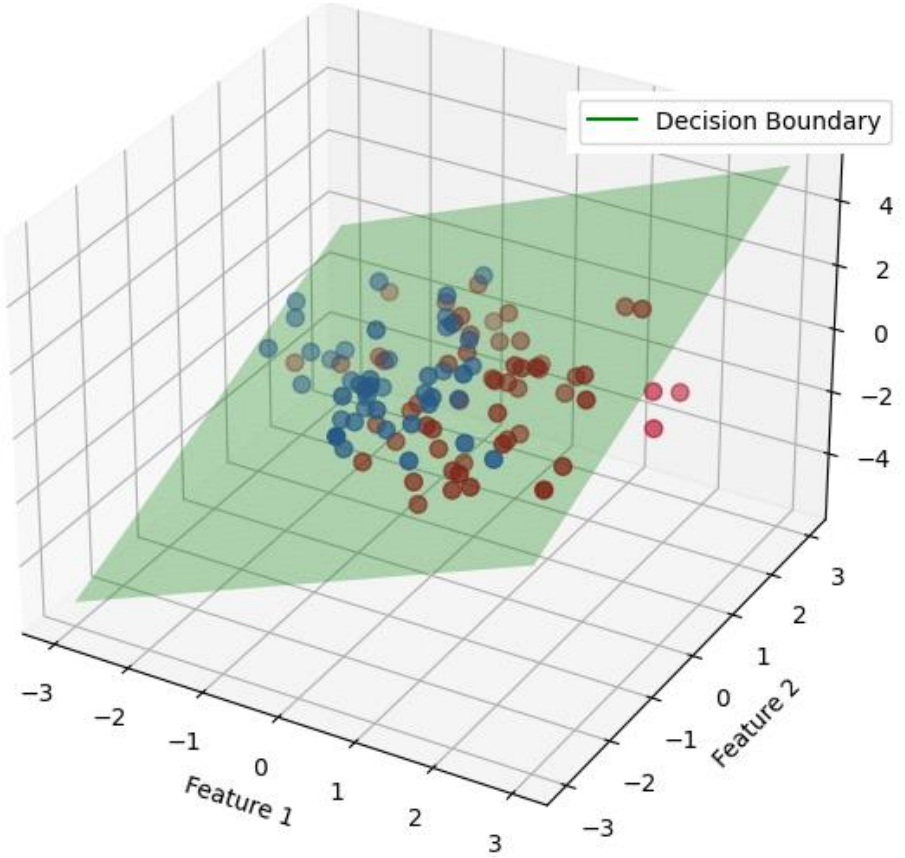
The hyperplane is a line.



Supervised Image Classification - Support Vector Machines (SVM)

3-dimensional space

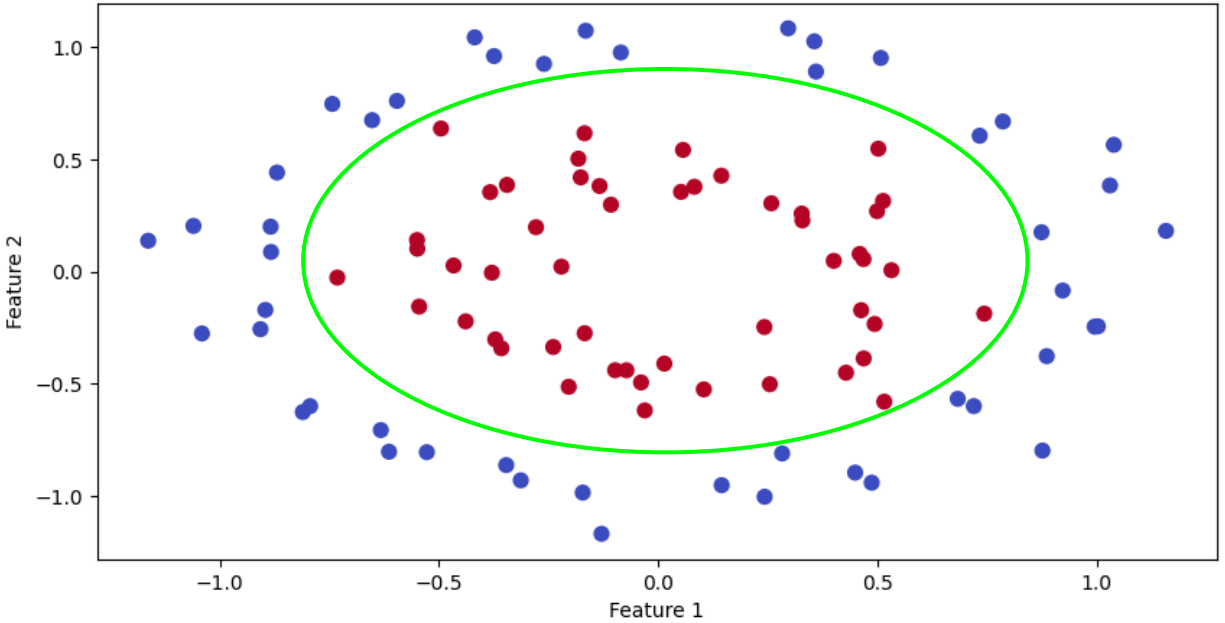
The hyperplane is a surface that splits the space into two parts.



Supervised Image Classification - Support Vector Machines (SVM)

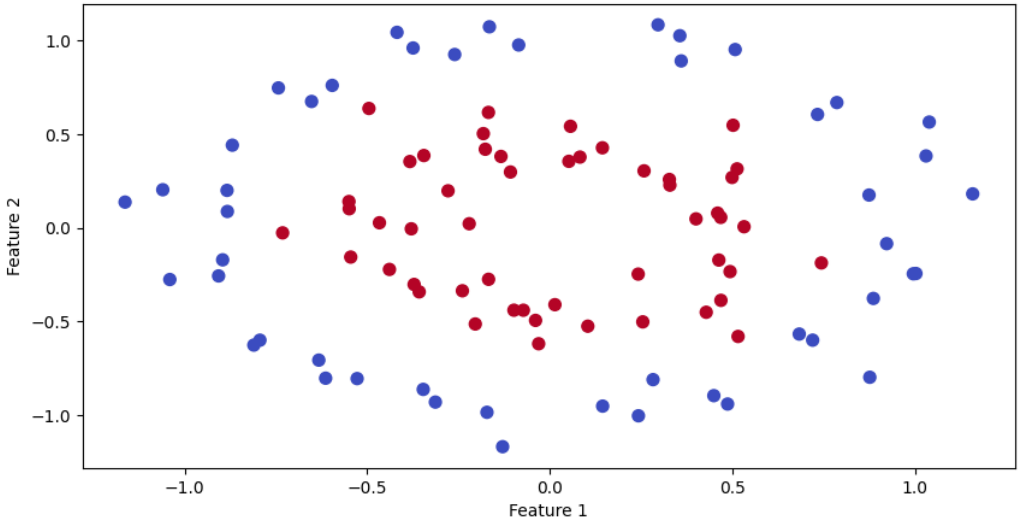
Non-linearly separable data

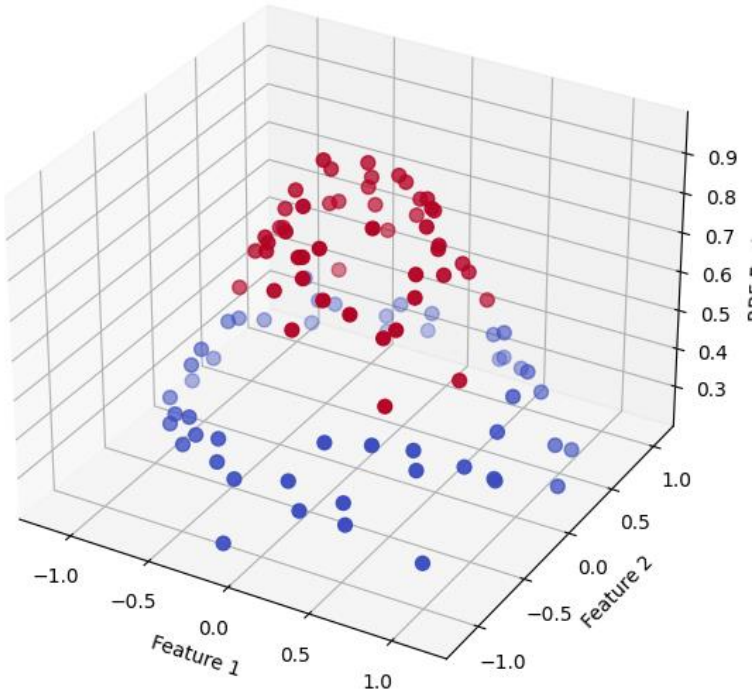
Two classes are not always linearly separable.
It means that it is not possible to separate the data points belonging to different classes using a single straight line (in 2D space), a plane (in 3D space), or a hyperplane (in higher dimensions).



Supervised Image Classification - Support Vector Machines (SVM)

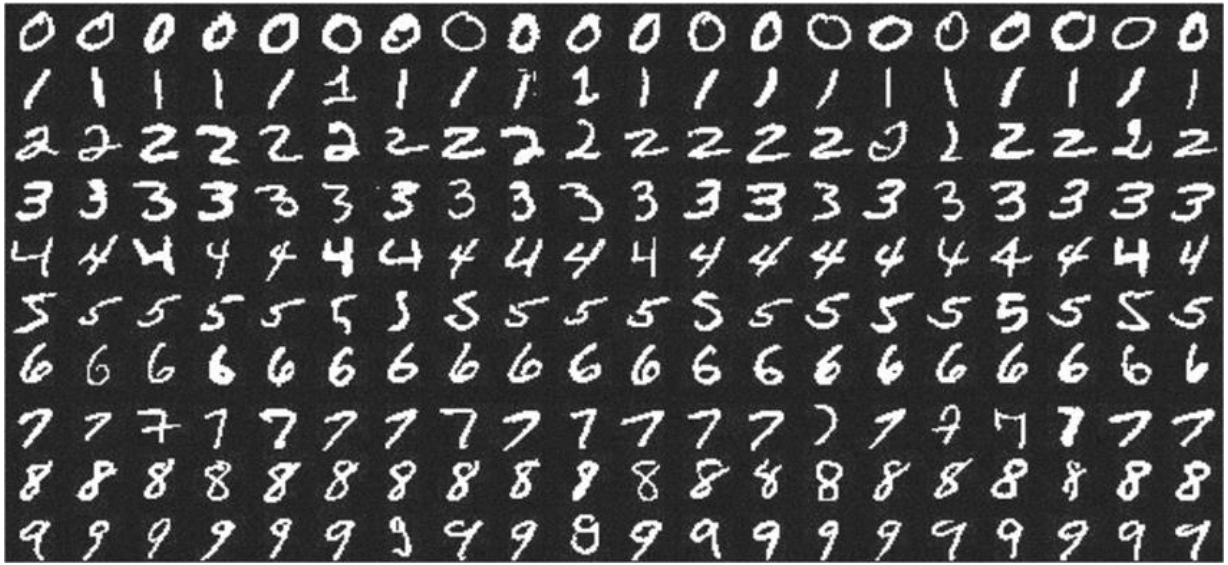
Solution: Project the data onto a higher-dimensional space.





Supervised Image Classification – Example
KNN & SVM

MNIST Database of Handwritten Digits

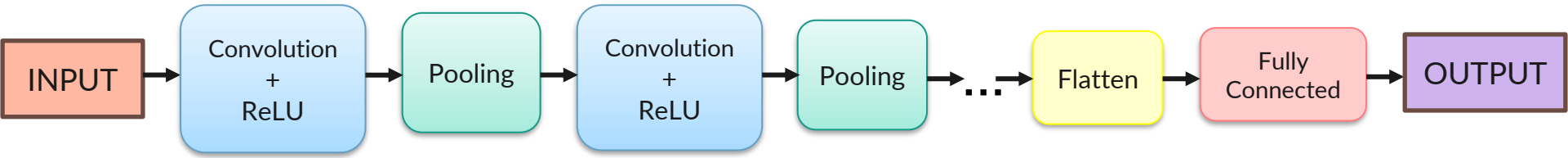


https://bit.ly/3WMrQSC

Convolutional Neural Networks (CNNs)

Specialized type of deep neural network primarily used for analyzing and processing data that has a grid-like structure, such as images.

Particularly powerful for tasks involving visual data, like image recognition, classification, and object detection



Convolutional Neural Networks (CNNs) - Convolution

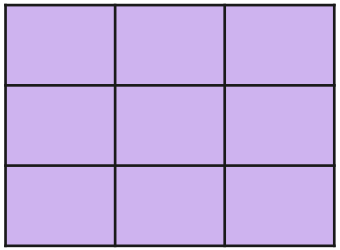
A filter (or kernel) slides over the input data (e.g., an image) to produce feature maps. This operation helps in detecting features such as edges, textures, and patterns within the input.
Multiple filters are applied in a layer to capture different features.

1	0	1
0	1	0
1	0	1

Kernel

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

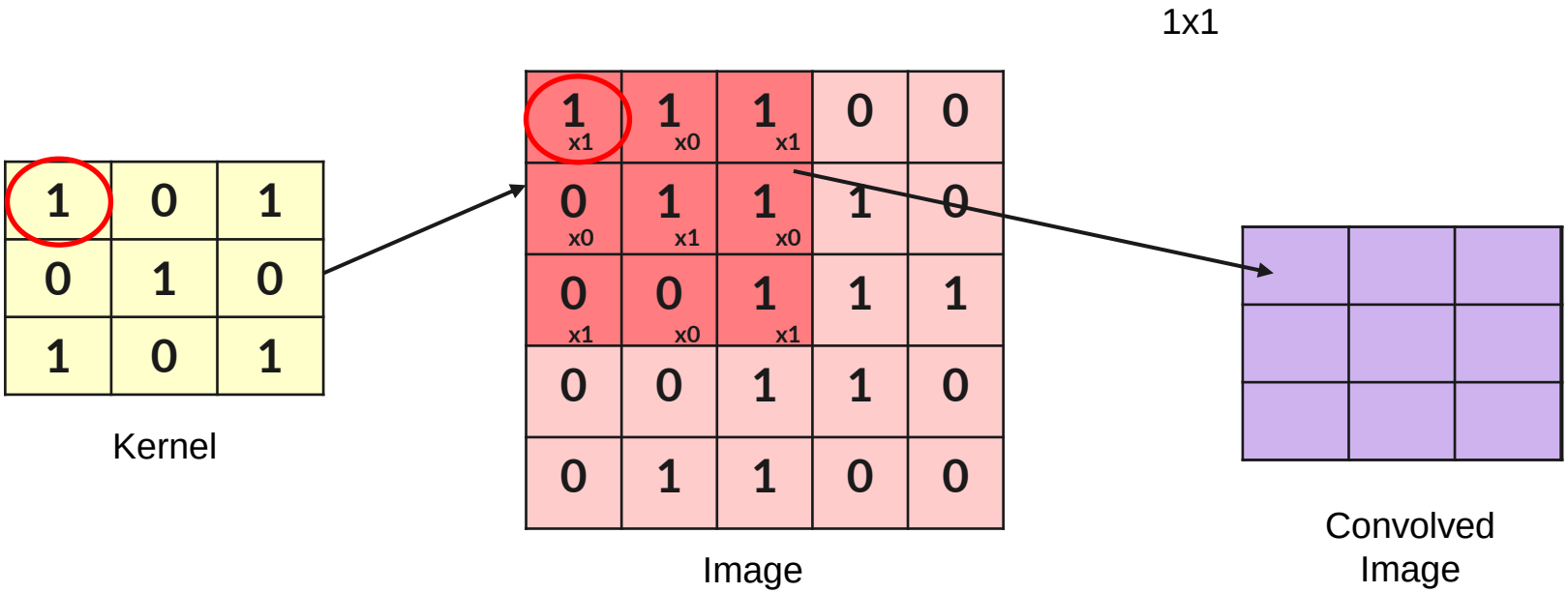
Image



Convolved Image

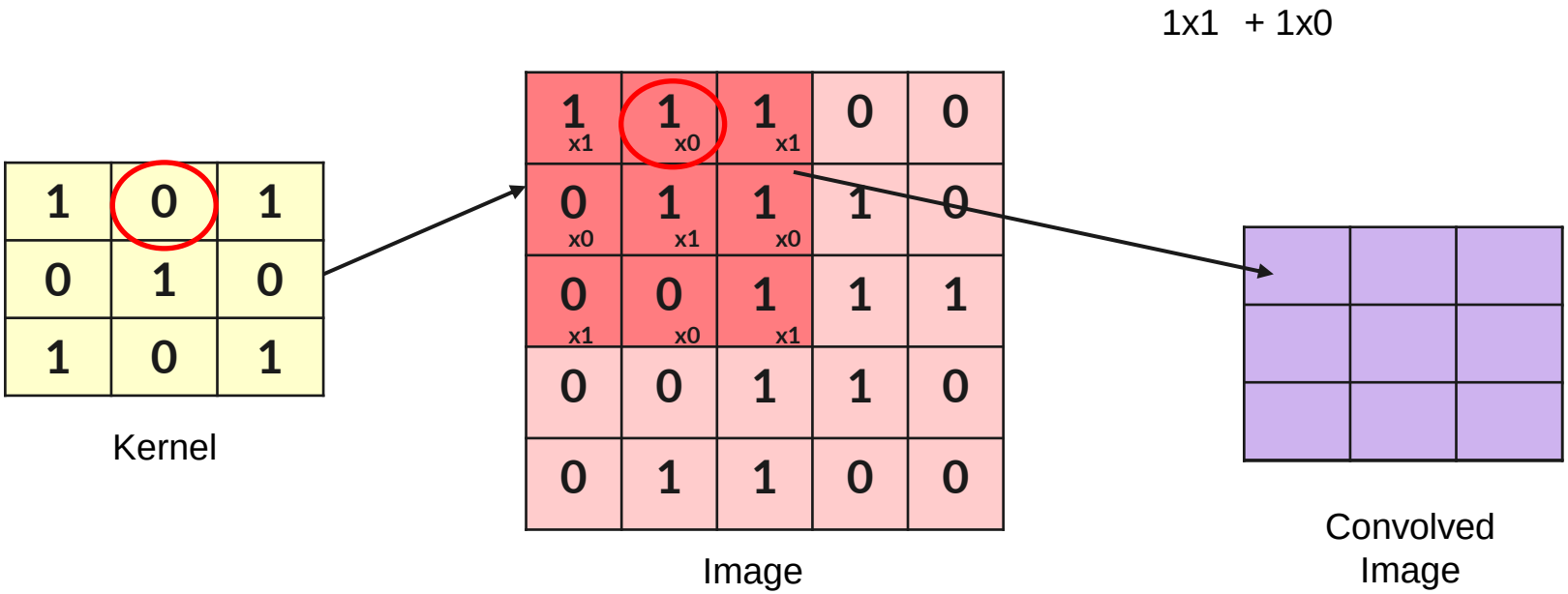
Convolutional Neural Networks (CNNs) - Convolution

A filter (or kernel) slides over the input data (e.g., an image) to produce feature maps. This operation helps in detecting features such as edges, textures, and patterns within the input.
Multiple filters are applied in a layer to capture different features.



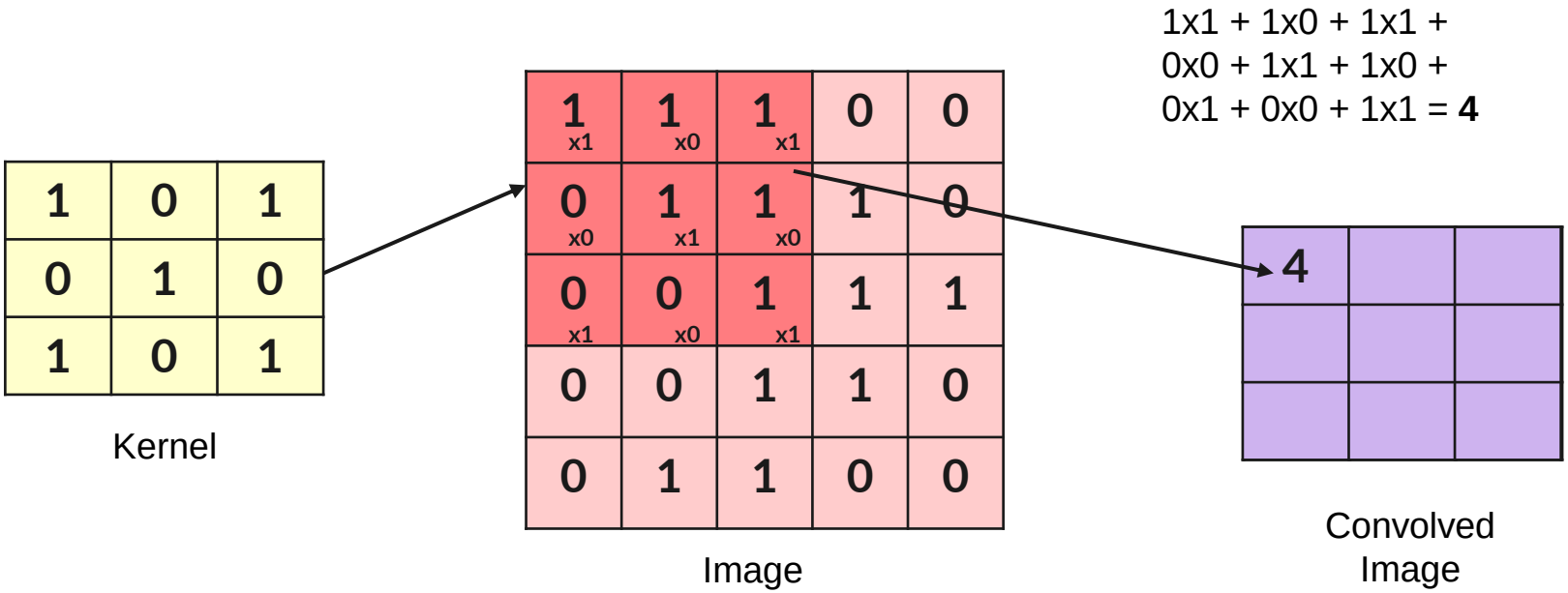
Convolutional Neural Networks (CNNs) - Convolution

A filter (or kernel) slides over the input data (e.g., an image) to produce feature maps. This operation helps in detecting features such as edges, textures, and patterns within the input.
Multiple filters are applied in a layer to capture different features.



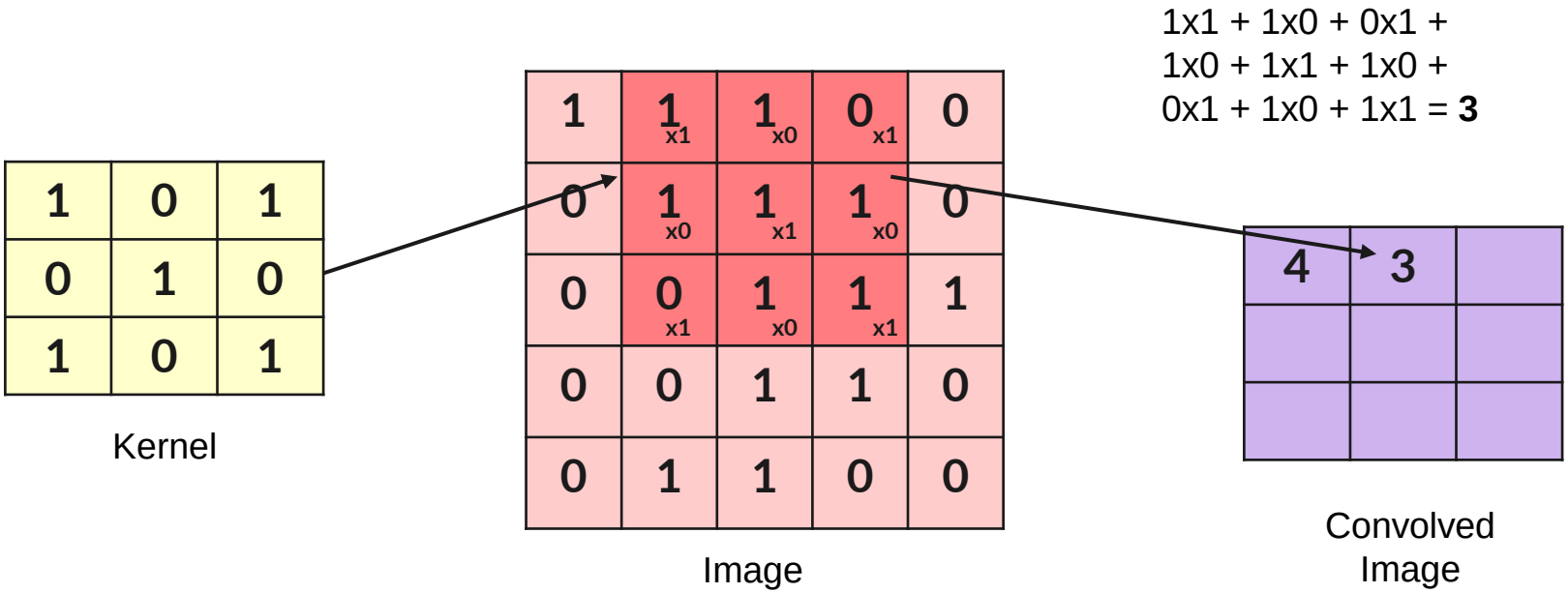
Convolutional Neural Networks (CNNs) - Convolution

A filter (or kernel) slides over the input data (e.g., an image) to produce feature maps. This operation helps in detecting features such as edges, textures, and patterns within the input.
Multiple filters are applied in a layer to capture different features.



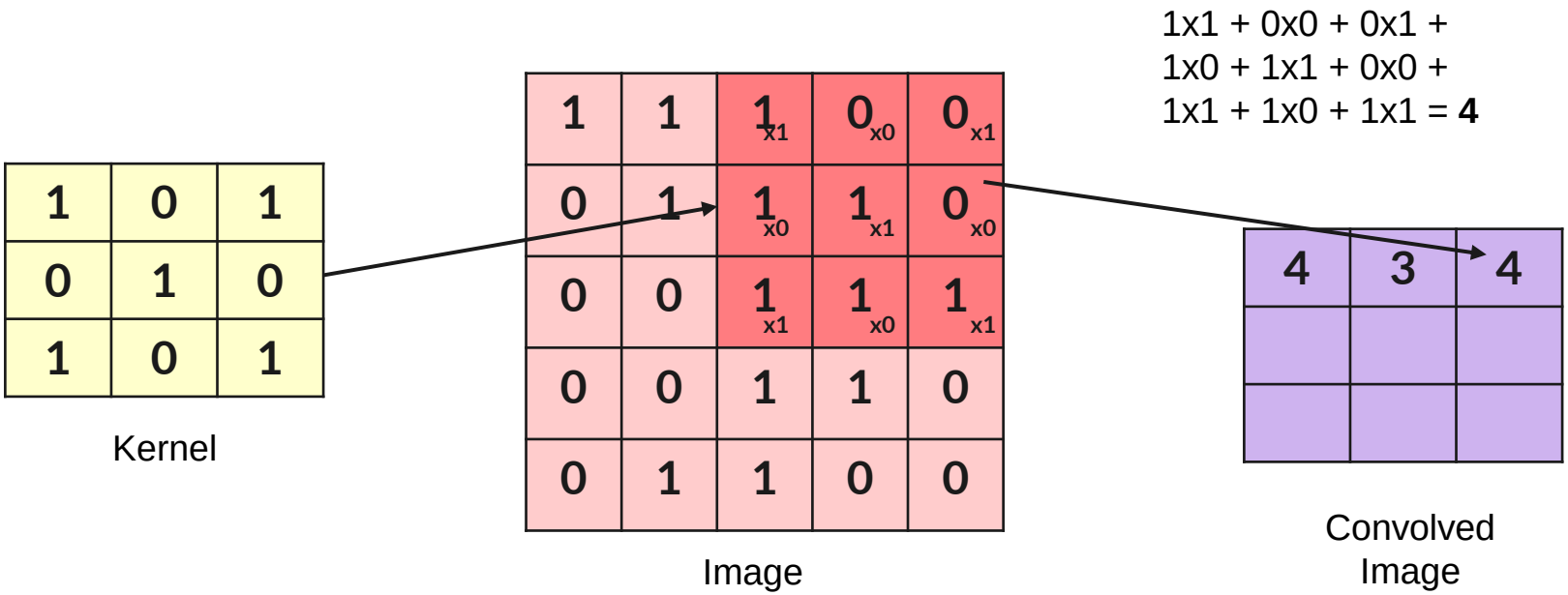
Convolutional Neural Networks (CNNs) - Convolution

A filter (or kernel) slides over the input data (e.g., an image) to produce feature maps. This operation helps in detecting features such as edges, textures, and patterns within the input.
Multiple filters are applied in a layer to capture different features.



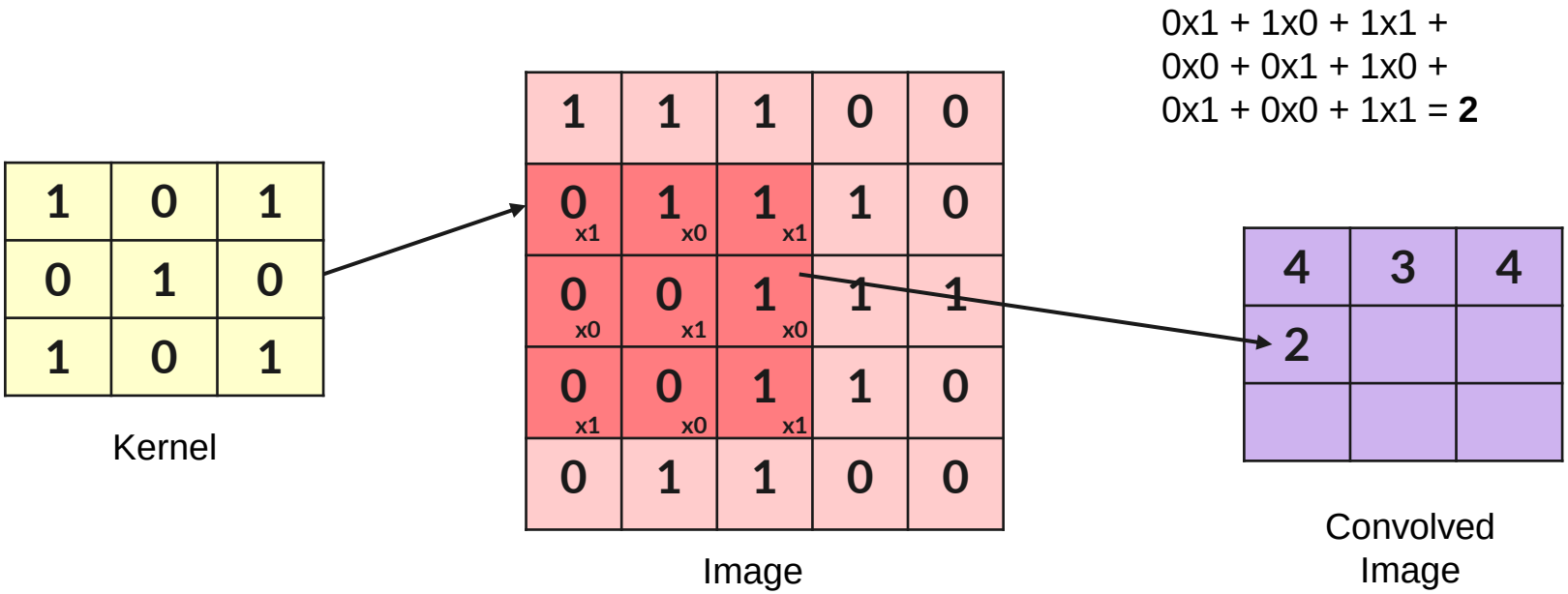
Convolutional Neural Networks (CNNs) - Convolution

A filter (or kernel) slides over the input data (e.g., an image) to produce feature maps. This operation helps in detecting features such as edges, textures, and patterns within the input.
Multiple filters are applied in a layer to capture different features.



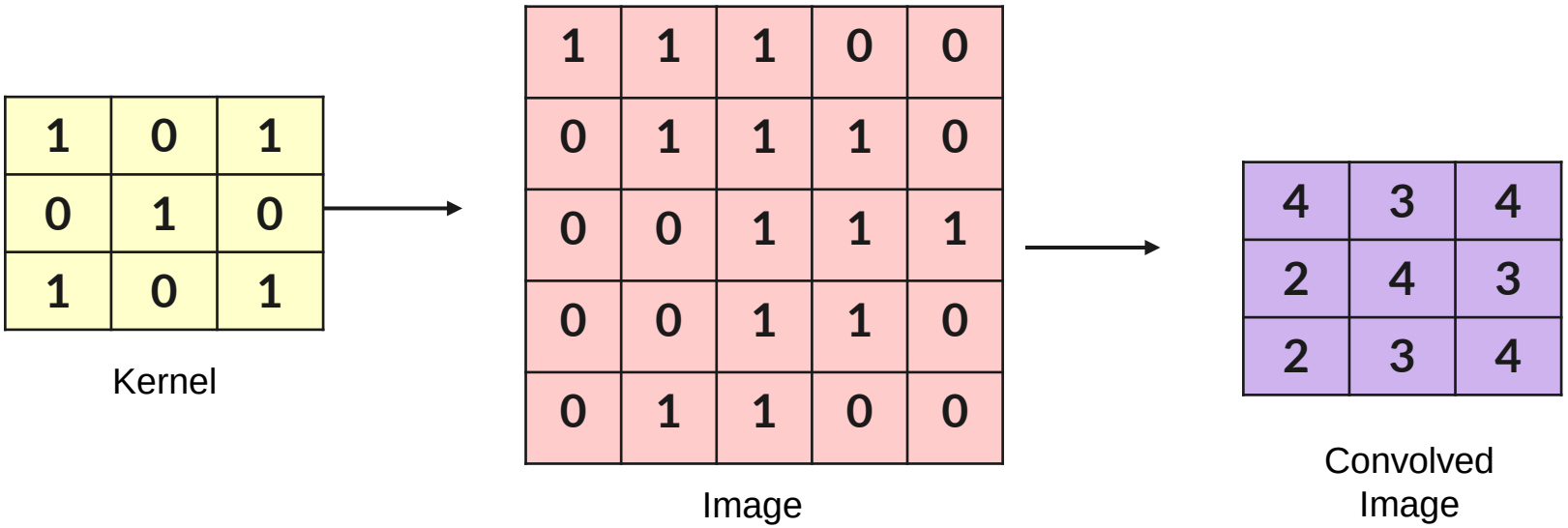
Convolutional Neural Networks (CNNs) - Convolution

A filter (or kernel) slides over the input data (e.g., an image) to produce feature maps. This operation helps in detecting features such as edges, textures, and patterns within the input.
Multiple filters are applied in a layer to capture different features.



Convolutional Neural Networks (CNNs) - Convolution

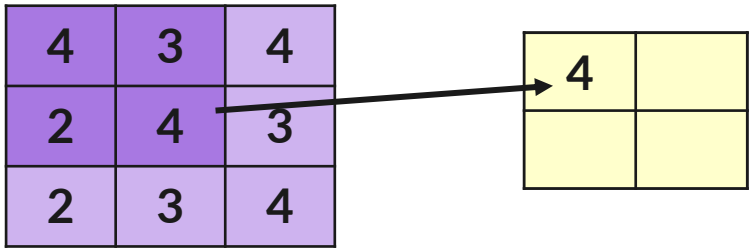
A filter (or kernel) slides over the input data (e.g., an image) to produce feature maps. This operation helps in detecting features such as edges, textures, and patterns within the input.
Multiple filters are applied in a layer to capture different features.



Convolutional Neural Networks (CNNs) - Pooling

Pooling layers are used to reduce the spatial dimensions of the feature maps (width and height), which helps in reducing the computational load and controlling overfitting.
A window slides over the feature map.

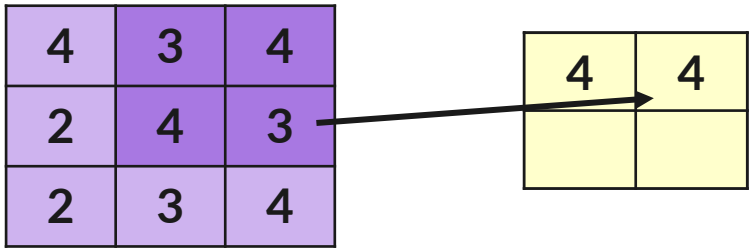
- **Max Pooling**: for each window position, the maximum value within that window is selected.
- **Average Pooling**: takes the average value of all the elements in the window.
- **Min Pooling**: it selects the minimum value in the pooling window.
- **L2-Norm Pooling**: calculates the L2 norm (Euclidean norm) of the elements in the pooling window. The L2 norm is the square root of the sum of the squares of the values in the window.



Convolutional Neural Networks (CNNs) - Pooling

Pooling layers are used to reduce the spatial dimensions of the feature maps (width and height), which helps in reducing the computational load and controlling overfitting.
A window slides over the feature map.

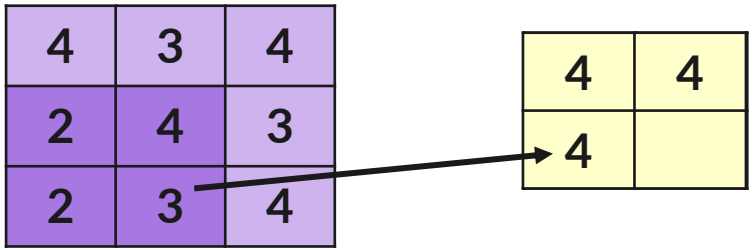
- **Max Pooling**: for each window position, the maximum value within that window is selected.
- **Average Pooling**: takes the average value of all the elements in the window.
- **Min Pooling**: it selects the minimum value in the pooling window.
- **L2-Norm Pooling**: calculates the L2 norm (Euclidean norm) of the elements in the pooling window. The L2 norm is the square root of the sum of the squares of the values in the window.



Convolutional Neural Networks (CNNs) - Pooling

Pooling layers are used to reduce the spatial dimensions of the feature maps (width and height), which helps in reducing the computational load and controlling overfitting.
A window slides over the feature map.

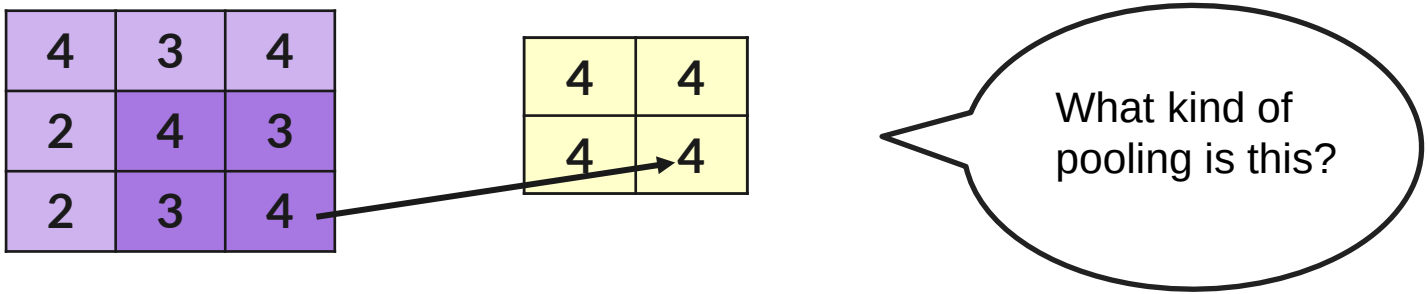
- **Max Pooling**: for each window position, the maximum value within that window is selected.
- **Average Pooling**: takes the average value of all the elements in the window.
- **Min Pooling**: it selects the minimum value in the pooling window.
- **L2-Norm Pooling**: calculates the L2 norm (Euclidean norm) of the elements in the pooling window. The L2 norm is the square root of the sum of the squares of the values in the window.



Convolutional Neural Networks (CNNs) - Pooling

Pooling layers are used to reduce the spatial dimensions of the feature maps (width and height), which helps in reducing the computational load and controlling overfitting.
A window slides over the feature map.

- **Max Pooling**: for each window position, the maximum value within that window is selected.
- **Average Pooling**: takes the average value of all the elements in the window.
- **Min Pooling**: it selects the minimum value in the pooling window.
- **L2-Norm Pooling**: calculates the L2 norm (Euclidean norm) of the elements in the pooling window. The L2 norm is the square root of the sum of the squares of the values in the window.



Convolutional Neural Networks (CNNs) - Fully-Connected layer

Way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer.
The image is first flattened into a column vector and then fed to a feed-forward network . After a certain number of epochs, the network can perform the classification or regression task.

Activation Functions:

- **Softmax**: It converts the raw output scores of the network into a probability distribution, where the sum of all probabilities is 1.
- **Sigmoid**: outputs a value between 0 and 1, making it useful for binary classification problems.
- **Linear Activation (No Activation Function)**: The network's output is directly the weighted sum of the inputs, allowing the model to predict a continuous range of values.

Classification problems

Regression problems

Trained vs pre-trained models

Training your own model

Pros

- Tailored to Your Dataset
- Complete Control
- No Dependency on External Data

Cons

- Time-Consuming
- Requires Large Amounts of Data
- Complexity

When:

- Your dataset or task is significantly different from existing datasets and models.
- You have enough data and computational resources.
- You need a model architecture tailored specifically to your problem.
- You want full control over the model design and training process.

Trained vs pre-trained models

Using a pre-trained model

Pros

- Faster Development:
- Less Data Required
- Leverage Transfer Learning
- Less Computationally Intensive

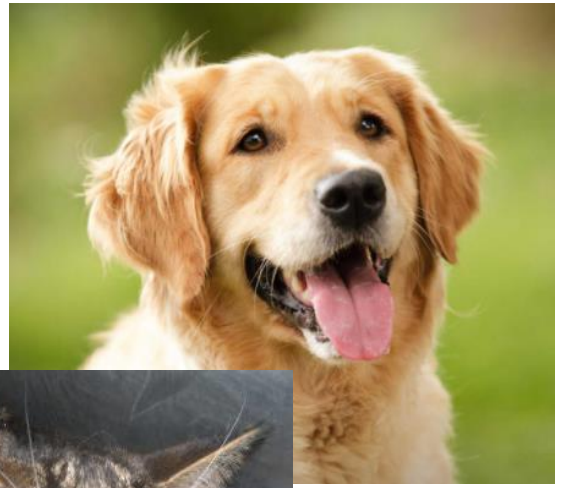
Cons

- Less Flexibility
- Potential for Misalignment
- Transfer of Biases

When:

- You have a limited amount of data.
- You need to achieve good results quickly.
- The task is similar to what the pre-trained model was originally trained for.
- You lack the computational resources to train from scratch.

Trained vs pre-trained models - Example



GOAL: we want to create a model able to classify whether an image contains a dog or a cat.

We will use the Dogs vs. Cats dataset, from kaggle.

We will first build and train our own custom network, and then try to exploit an already trained network through transfer learning and fine tuning.

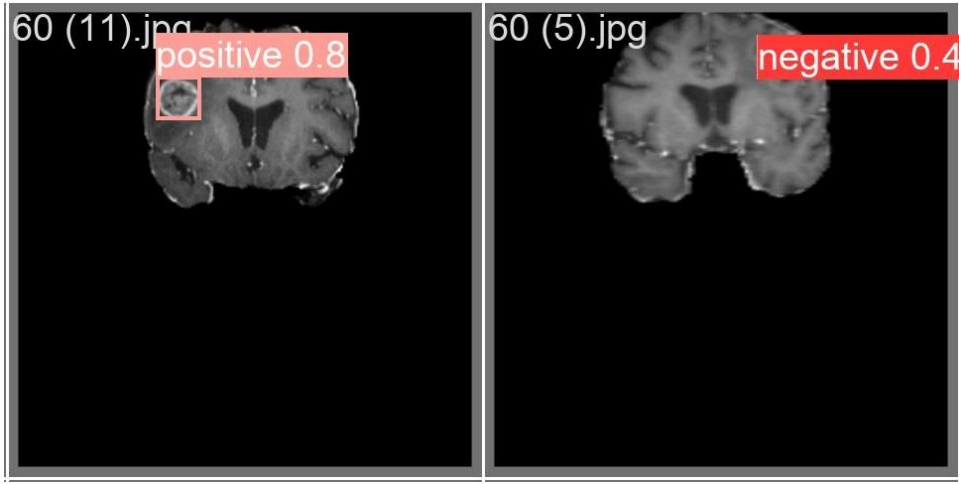
https://bit.ly/4fQkkiF

Binary vs. Multi-label Classification

Binary Classification

Involves classifying images into one of two classes. This is the simplest form of classification where each image is assigned to exactly one of two categories.

Cats VS Dogs



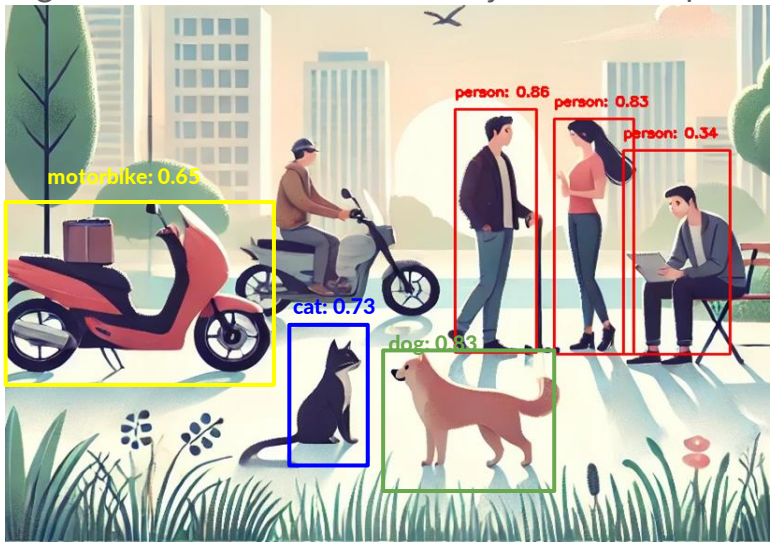
Brain tumor VS No brain tumor

https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/datasets/brain-tumor.yaml

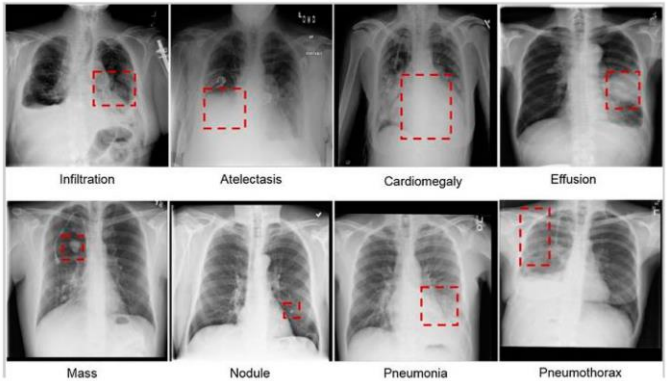
Binary vs. Multi-label Classification

Multi-label Classification

Each image can belong to multiple classes simultaneously. Unlike binary classification, where an image is assigned to only one class, multilabel classification allows the model to predict several classes for a single image. This is common when images contain more than one object or concept.



Object Detection



Medical Diagnosis

Mustafa, Z.; Nsour, H. Using Computer Vision Techniques to Automatically Detect Abnormalities in Chest X-rays. *Diagnostics* **2023**, *13*, 2979. https://doi.org/10.3390/diagnostics13182979

Video Classification

Analyzing and categorizing entire video clips into predefined categories or classes based on their visual, temporeal and/or audio content

Possible applications

- Content Recommendation
- Surveillance
- Sports Analysis
- Healthcare

Video Classification - Challenges

- **Temporal Dimension**: Unlike image classification, where a single image is analyzed, video classification must consider both the spatial (what's in the frame) and temporal (how things change over time) information.
- **Variation in Frames**: Videos contain multiple frames that can vary greatly due to camera movement, object movement, lighting conditions, and scene changes.
- **Long Sequences**: Videos can be long, leading to challenges in efficiently processing them to capture relevant information while reducing computation.
- **Generalization**: Many video classification algorithms are designed to work on a specific dataset or task but may not generalize well to other datasets or tasks.
- **Video annotation**: Labeling large amounts of video data is a time-consuming and labor-intensive task. This can make it difficult to obtain large, high-quality datasets for training and evaluating video classification algorithms
- **Privacy and security**: The use of video data raises a number of privacy and security concerns, particularly when it comes to surveillance and facial recognition.

Video Classification - Techniques

- **Convolutional Neural Networks (CNNs):** Used for extracting spatial features from individual frames of the video. CNNs are effective at recognizing patterns, objects, and structures in the visual content.
- **Recurrent Neural Networks (RNNs) or LSTMs**: These are often used to capture temporal dependencies across the sequence of frames. They help in modeling the progression of actions or events over time.
- **3D CNNs**: These extend the CNN concept to handle both spatial and temporal information simultaneously by applying convolutions across the spatial and temporal dimensions (height, width, and time).
- **Transformer Models**: Vision Transformers or Video Transformers, which excel at capturing long-range dependencies and have been applied to video classification tasks.