Computer Vision

Lecture 4: Machine Learning

Oliver Bimber

JOHANNES KEPLER UNIVERSITY LINZ

Last Week: Digital Image Processing

Convolution

$$y[m,n] = x * h = \sum_{k,l=-N}^{N} x[m+k, n+l]h[-k,-l]$$

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

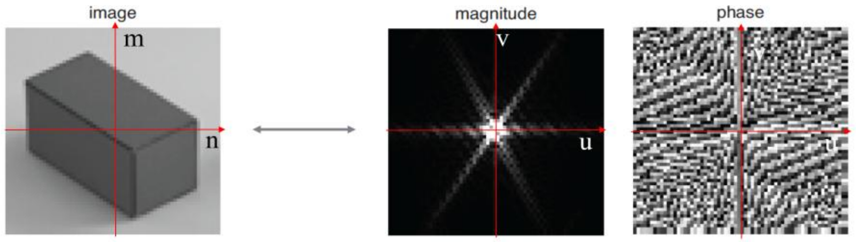
$$\begin{array}{c|ccc} 1 & -1 & -2 & -1 \\ 0 & 0 & 0 & 0 \\ -1 & 1 & 2 & 1 \\ \hline l \; k\rightarrow & -1 & 0 & 1 \end{array}$$

In this case, N = 1.

Convolution is a shift-invariant linear Operation.

Spatial vs. Frequency Domain

$f[n,m]$ $\qquad$ $F[u,v]$



Fourier Transformation /
Inverse Fourier Transformation

JKU JOHANNES KEPLER UNIVERSITY LINZ

Spatial vs. Gradient Domain

Laplacian!

$$\mathrm{div}(\nabla I) = \frac{\partial I_x}{\partial x} + \frac{\partial I_y}{\partial y} = I_{xx} + I_{yy} = \nabla^2 I$$

$$\mathrm{curl}(\nabla I) = \frac{\partial I_y}{\partial x} - \frac{\partial I_x}{\partial y} = I_{yx} - I_{xy}$$

Gradient Domain

Image gradient:

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$$

I_x $\quad$ I_y $\quad$ We can consider this as Coefficients of a Vectorfield.

Derivative / Integral

Processing the Vectorfield is called Gradient Domain Processing.

Spatial Domain

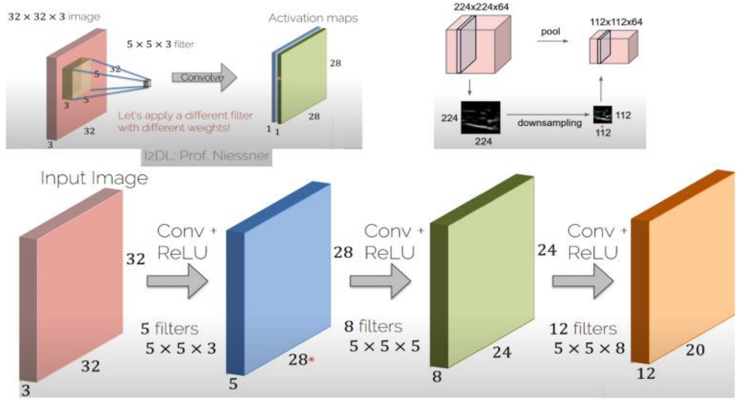
$$I = \int \nabla I$$

Integrating this Vectorfield results in the original Image as long as the Vectorfield has zero Curl. **In Practice: that does not work as the Vectorfields is no longer conservative anymore after processing it (it has non-zero Curl).**

Instead: solve 2D Poisson Equation: $\nabla^2 I = \mathrm{div}(G)$

JKU JOHANNES KEPLER UNIVERSITY LINZ

Convolutional Neural Networks (CNNs)



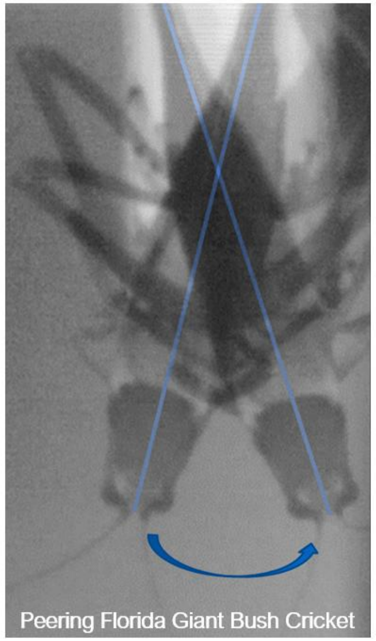
JKU JOHANNES KEPLER UNIVERSITY LINZ

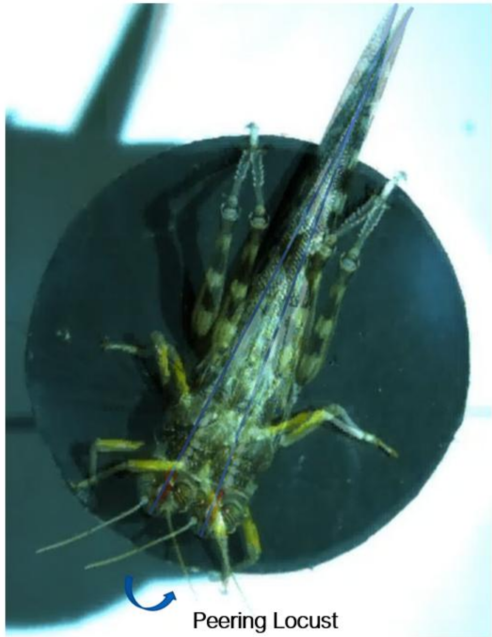
JKU JOHANNES KEPLER UNIVERSITY LINZ

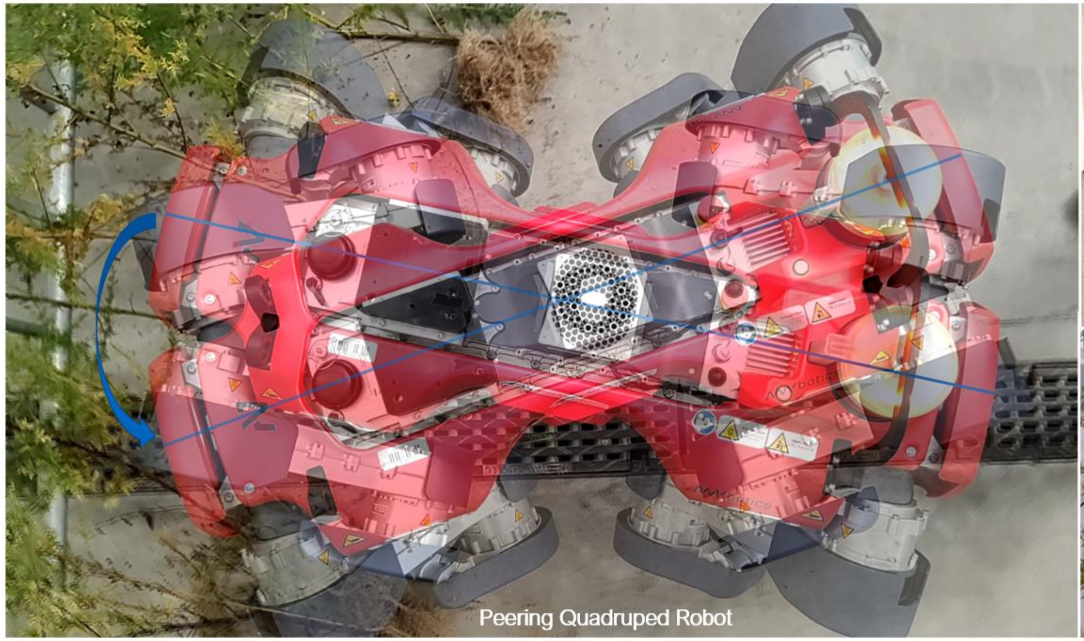
Course Overview

CW	Topic	Date	Place	Lab
41	Introduction and Course Overview	07.10.2025	Zoom	Lab 1
42	Capturing Digital Images	14.10.2025	Zoom	Lab 2
43	Digital Image Processing	21.10.2025	Zoom	Assignment 1
44	Machine Learning	28.10.2025	Zoom	
45	Feature Extraction	04.11.2025	Zoom	Open Lab 1
46	Segmentation	11.11.2025	Zoom	Assignment 2
47	Optical Flow	18.11.2025	Zoom	Open Lab 2
48	Object Detection	25.11.2025	Zoom	Assignment 3
49	Multi-View Geometry	02.12.2025	Zoom	Open Lab 3
50	3D Vision	09.12.2025	Zoom	Assignment 4
3	Trends in Computer Vision	13.01.2026	Zoom	
4	Q&A	20.01.2026	Zoom	Open Lab 4
5	Exam	27.01.2026	HS1 (Linz), S1/S3 (Vienna), S5 (Bregenz)	
9	Retry Exam	24.02.2026	tba	

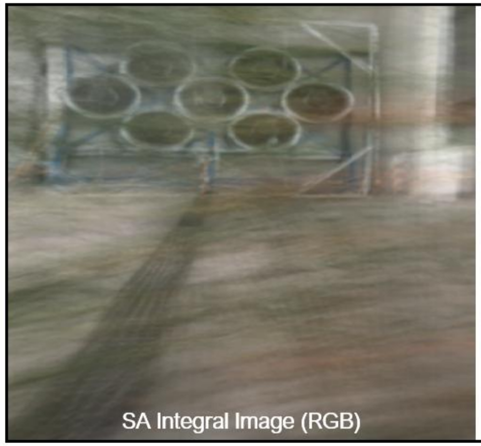
Research Example: How Robot Dogs See the Unseeable





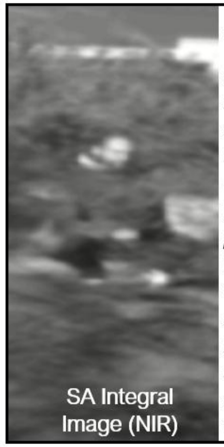


LMM Vision Reasoning:
The image shows some green plants with thin stems and fine leaves growing in front of a metallic fence or railing. Behind the plants, there is an industrial or commercial-looking structure and a concrete floor. The setting appears to be an industrial facility or a technical building exterior.



LMM Vision Reasoning:
The image appears to be quite blurry, but some features can still be made out. It seems to show an industrial or abandoned facility, possibly a warehouse or a factory. In the center, there's a large structure with circular openings or pipes — maybe ventilation ducts, tubes, or barrels stacked in a frame. The surroundings include a metal framework.

LMM Vision Reasoning:
The image appears to show a grayscale or infrared view of a scene with dense vegetation in the foreground — mainly leafy plants or shrubs. In the background, there seems to be a structure or series of buildings, possibly with a road or paved area in between.



LMM Vision Reasoning:
The image appears to be a blurry grayscale or low-resolution photo showing an outdoor scene. It looks like there is some vegetation or bushes in the background, and a person or human-like figure is sitting or crouching on the ground near the center of the image. The surroundings seem natural — possibly a garden, park, or wooded area.

What does ☆ do?

2 ☆ 3 = 36

7 ☆ 1 = 49

5 ☆ 2 = 100

2 ☆ 2 = 16

What does ☆ do?

Training

2 ☆ 3 = 36

7 ☆ 1 = 49

5 ☆ 2 = 100

2 ☆ 2 = 16

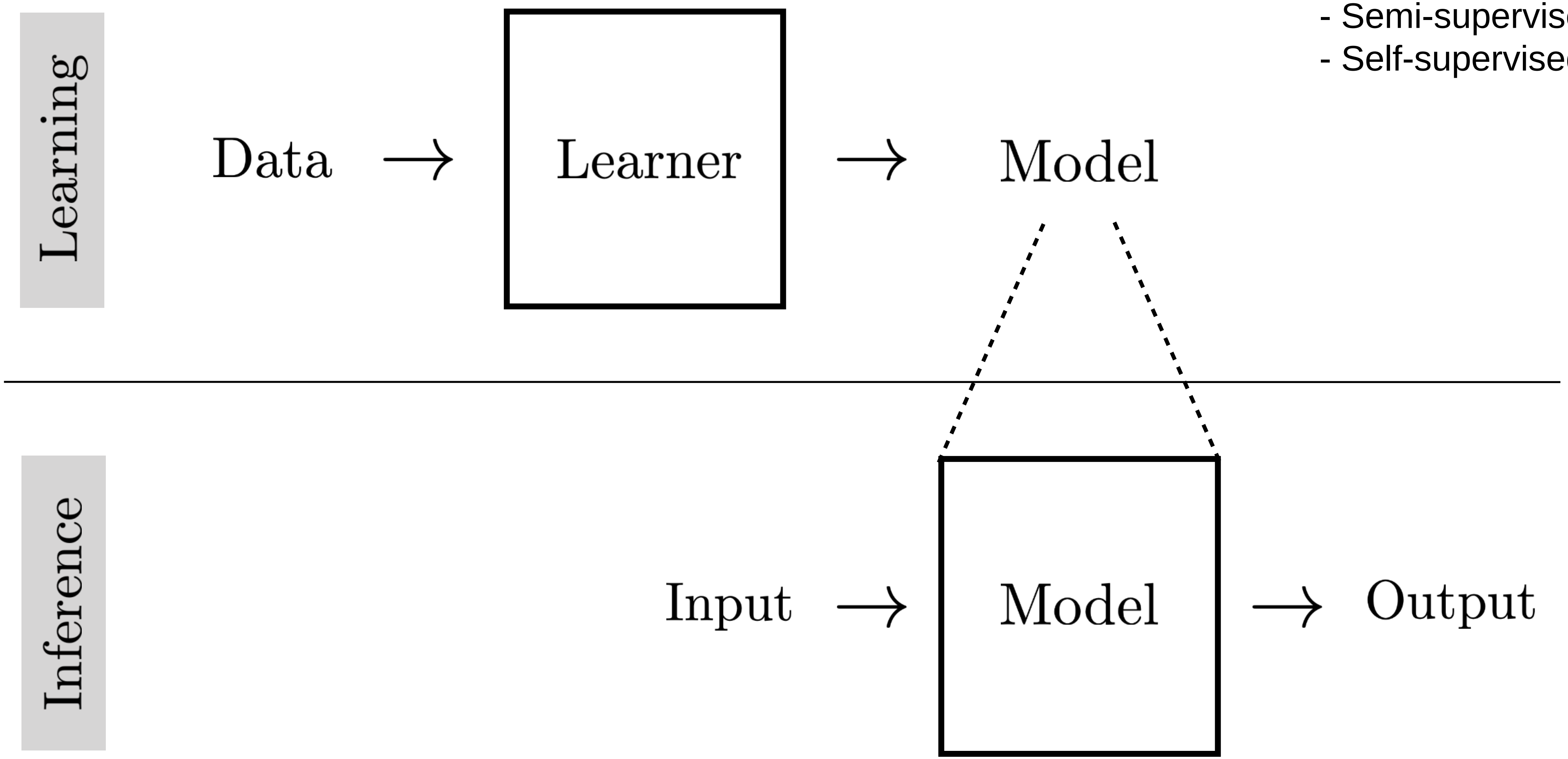
$\rightarrow$ Your brain $\rightarrow$ $\boxed{x \star y \rightarrow (xy)^2}$

Testing

3 ☆ 5 $\rightarrow$ $\boxed{x \star y \rightarrow (xy)^2}$ $\rightarrow$ 225

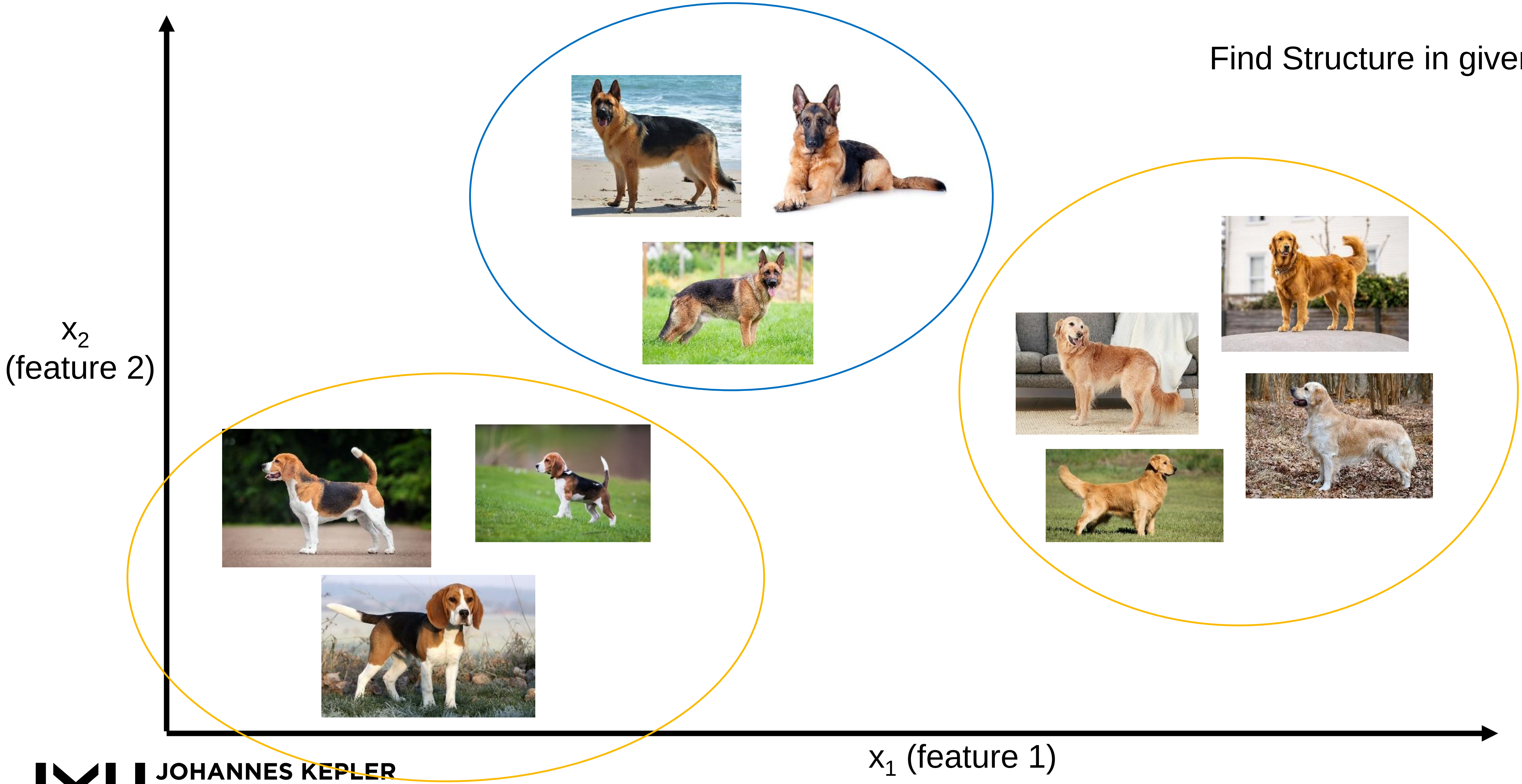
Machine Learning

ML Classes:
- Unsupervised
- Supervised
- Semi-supervised
- Self-supervised

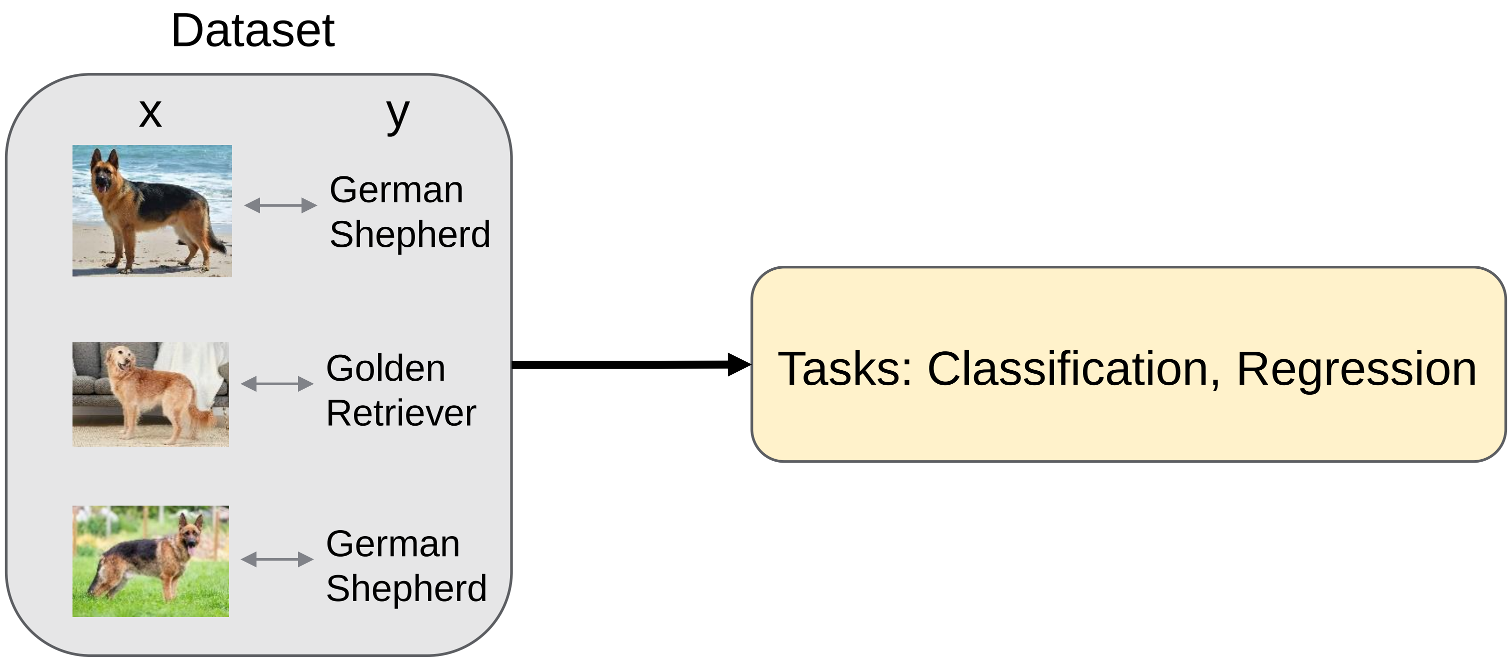


Unsupervised Learning

Find Structure in given Data

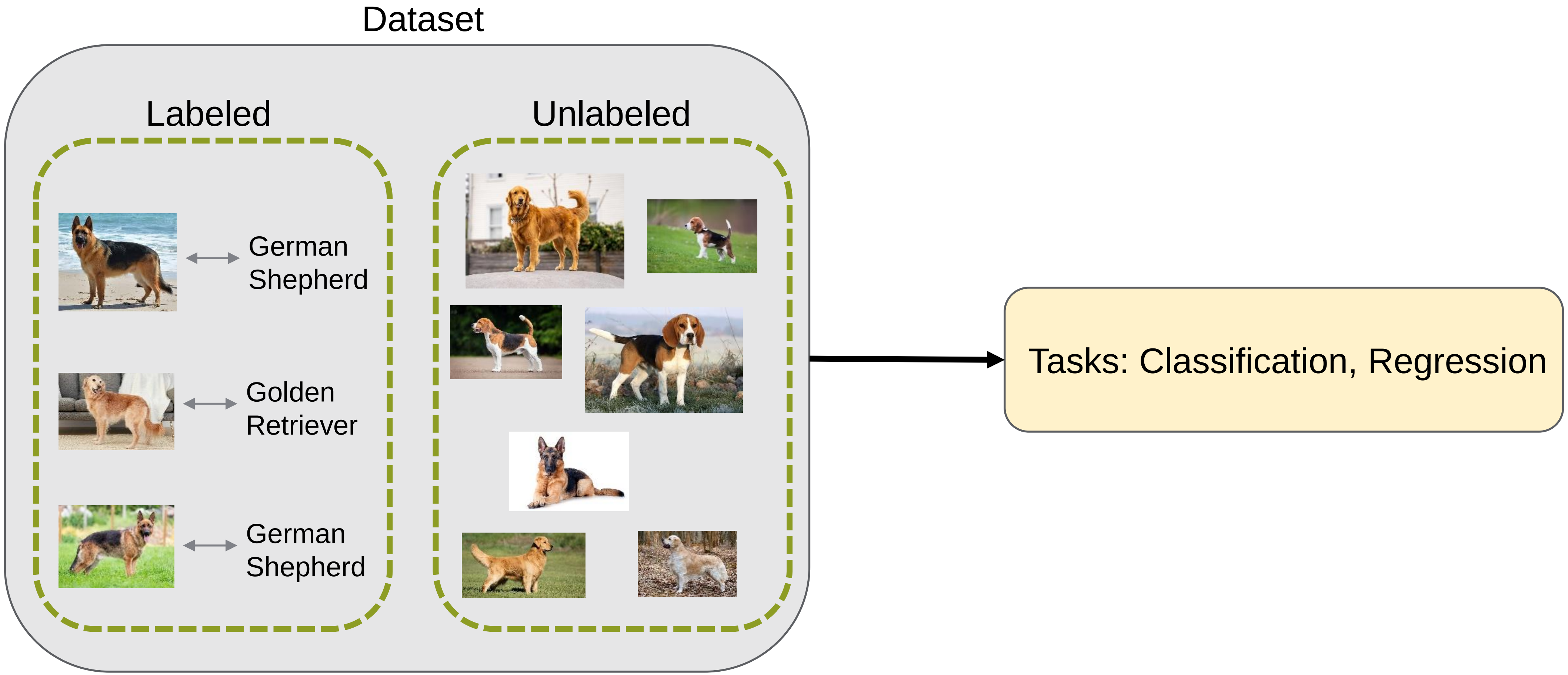


Supervised Learning



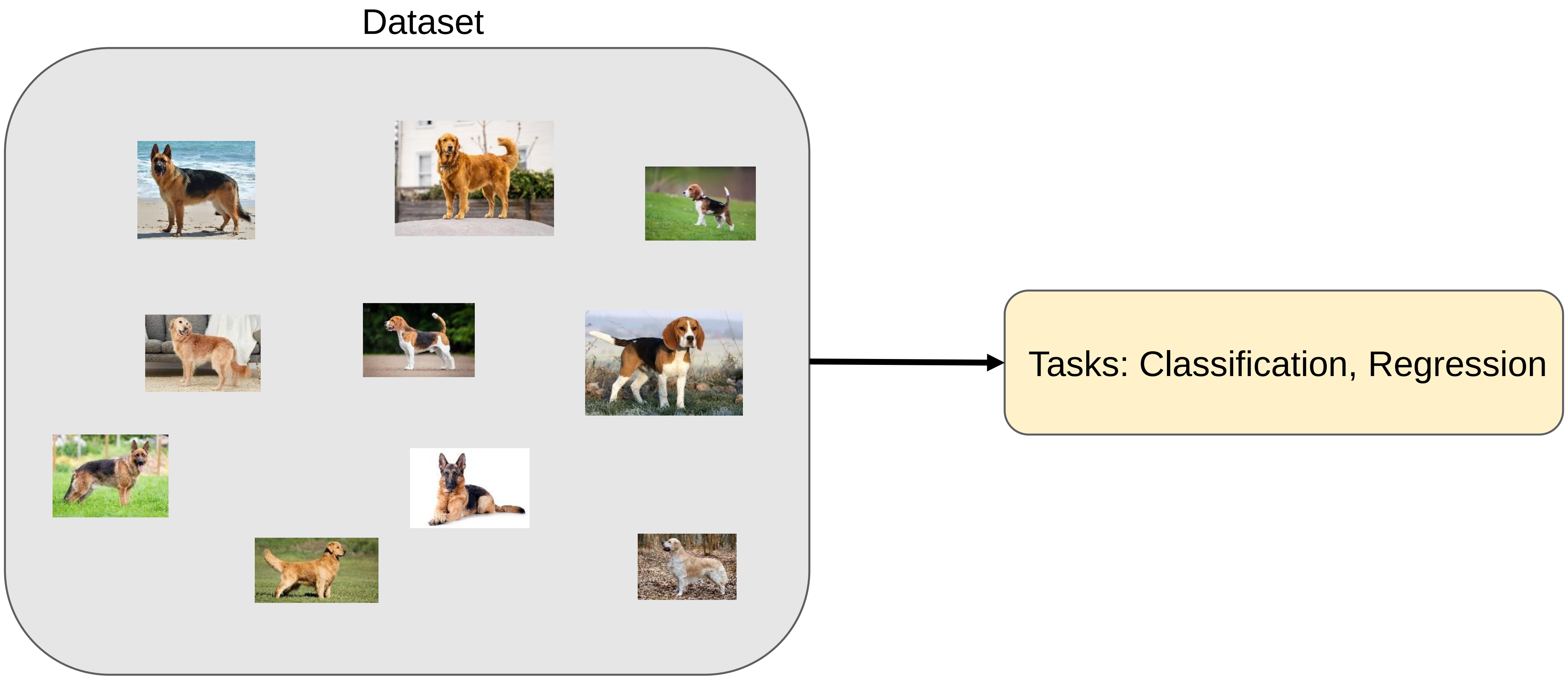
Use labeled Data for a Prediction Task.

Semi-Supervised Learning



Use labeled Data + unlabeled Data for a Prediction Task.

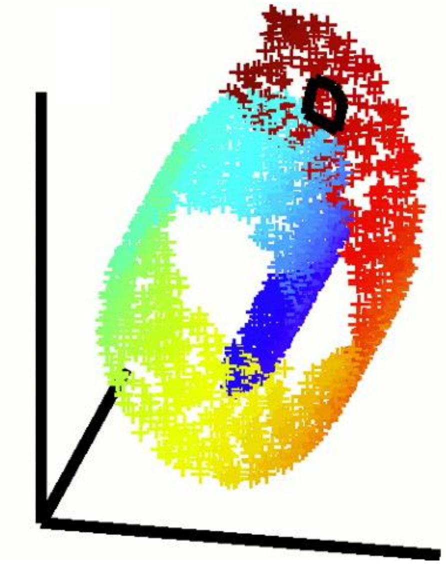
Self-Supervised Learning



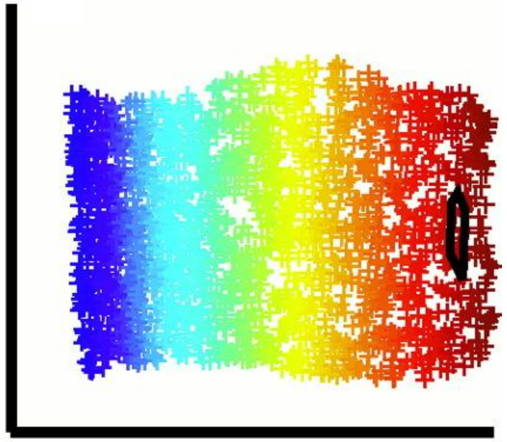
No external Labels. Instead, uses Properties of the Data itself as supervisory Signals. Example (learn how to solve a puzzle): take unlabeled images, rotate them, and train a model to guess the rotation. It needs to understand how the objects look upright → helps to learn how to classify.

Dimensionality Reduction

Given Data Points in *d* Dimensions, convert them to Data Points in *k*<*d* Dimensions with minimal loss of Information.

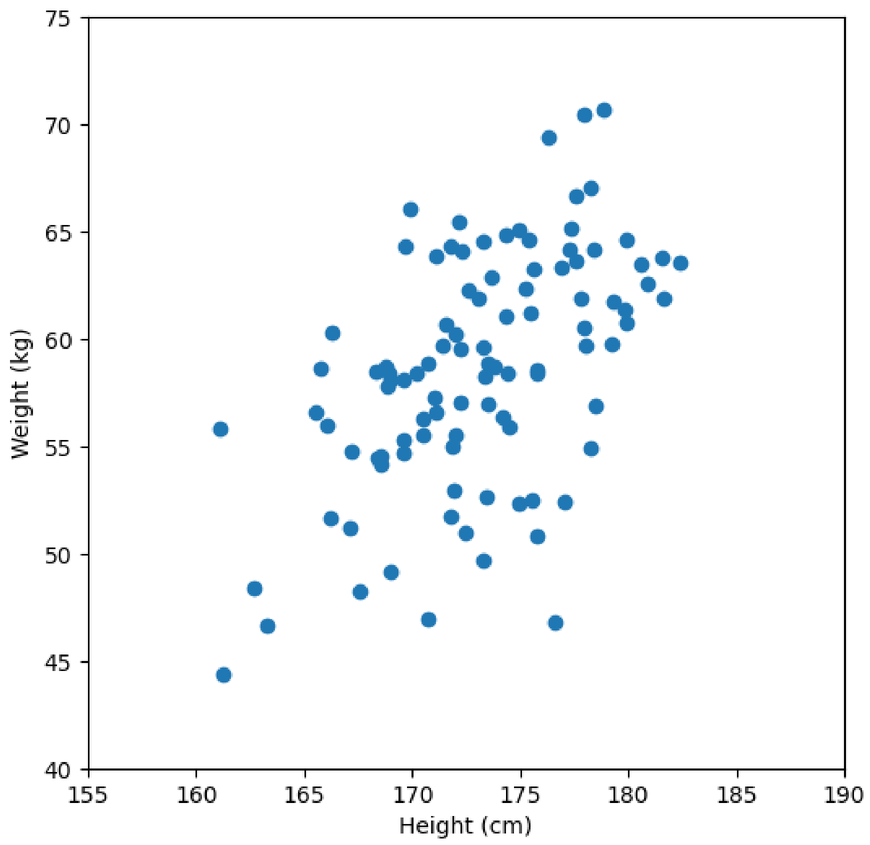


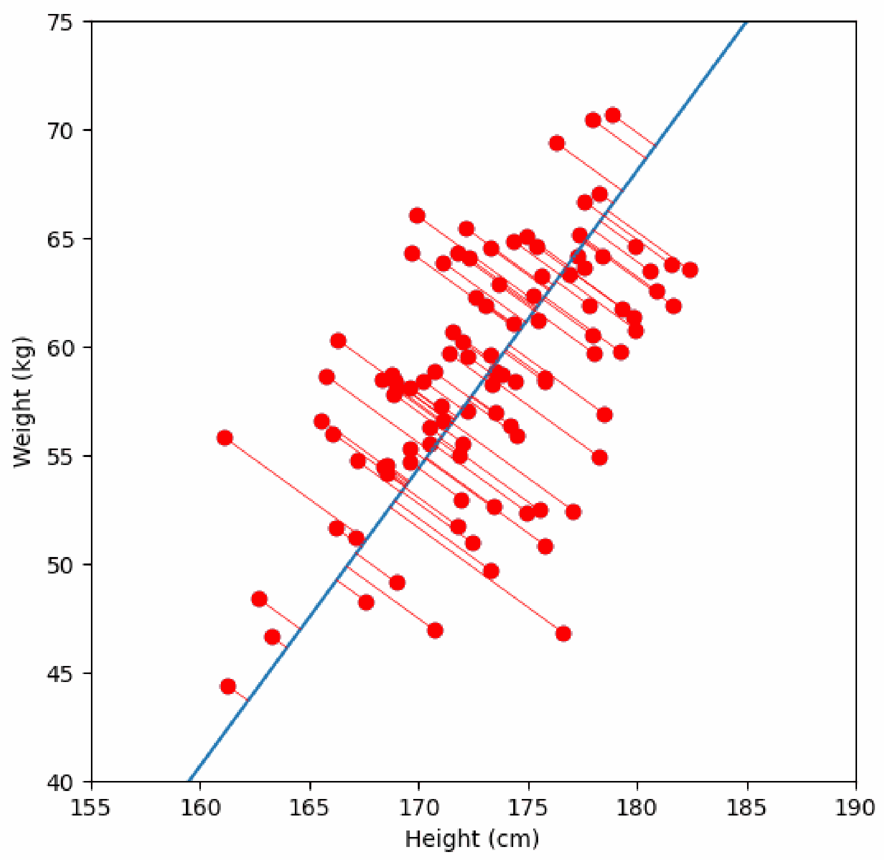
Original
Data in 3D



Same data
in 2D

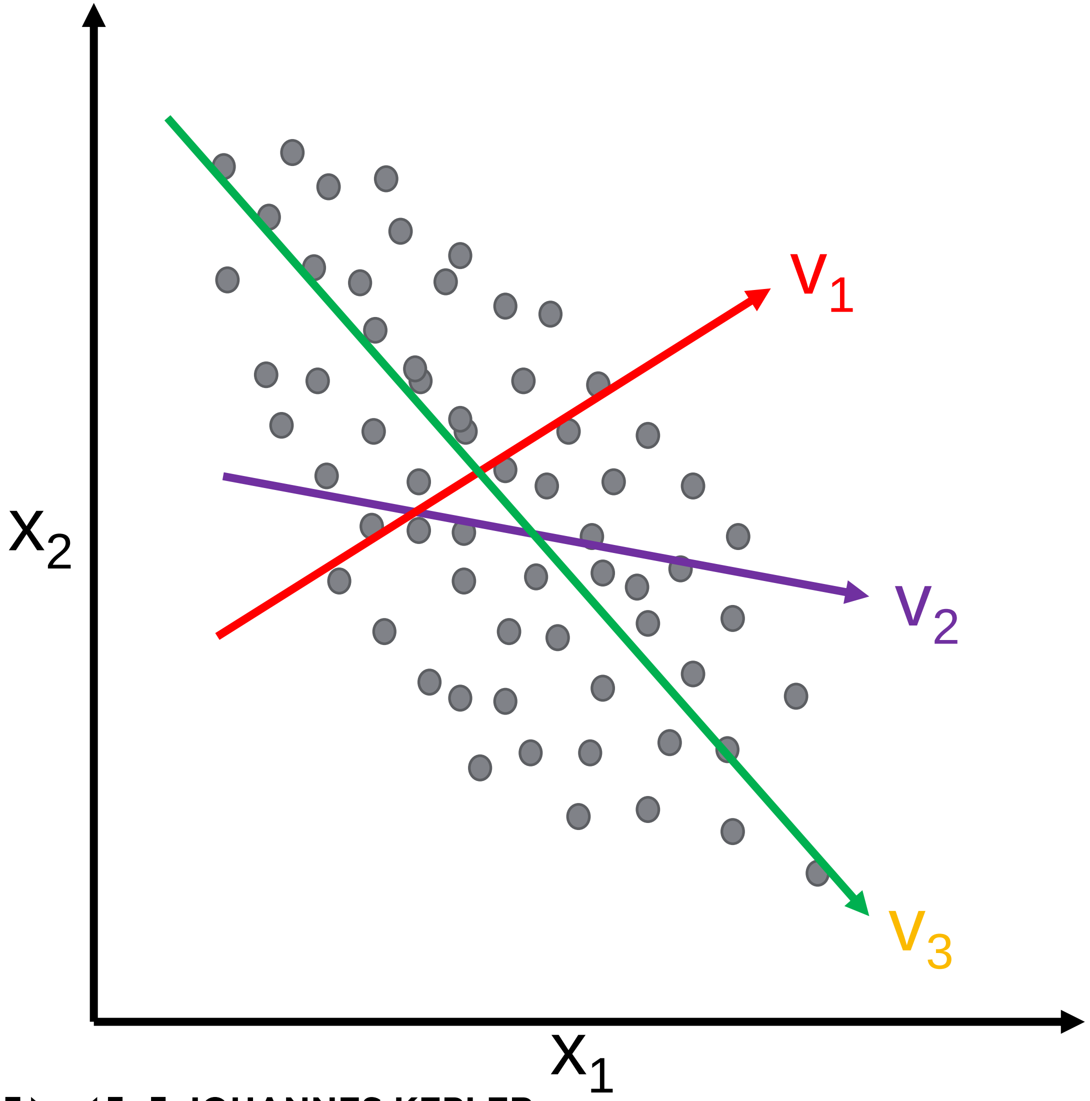
Dimensionality Reduction: Example





Correlation Body Weight vs. Body Size

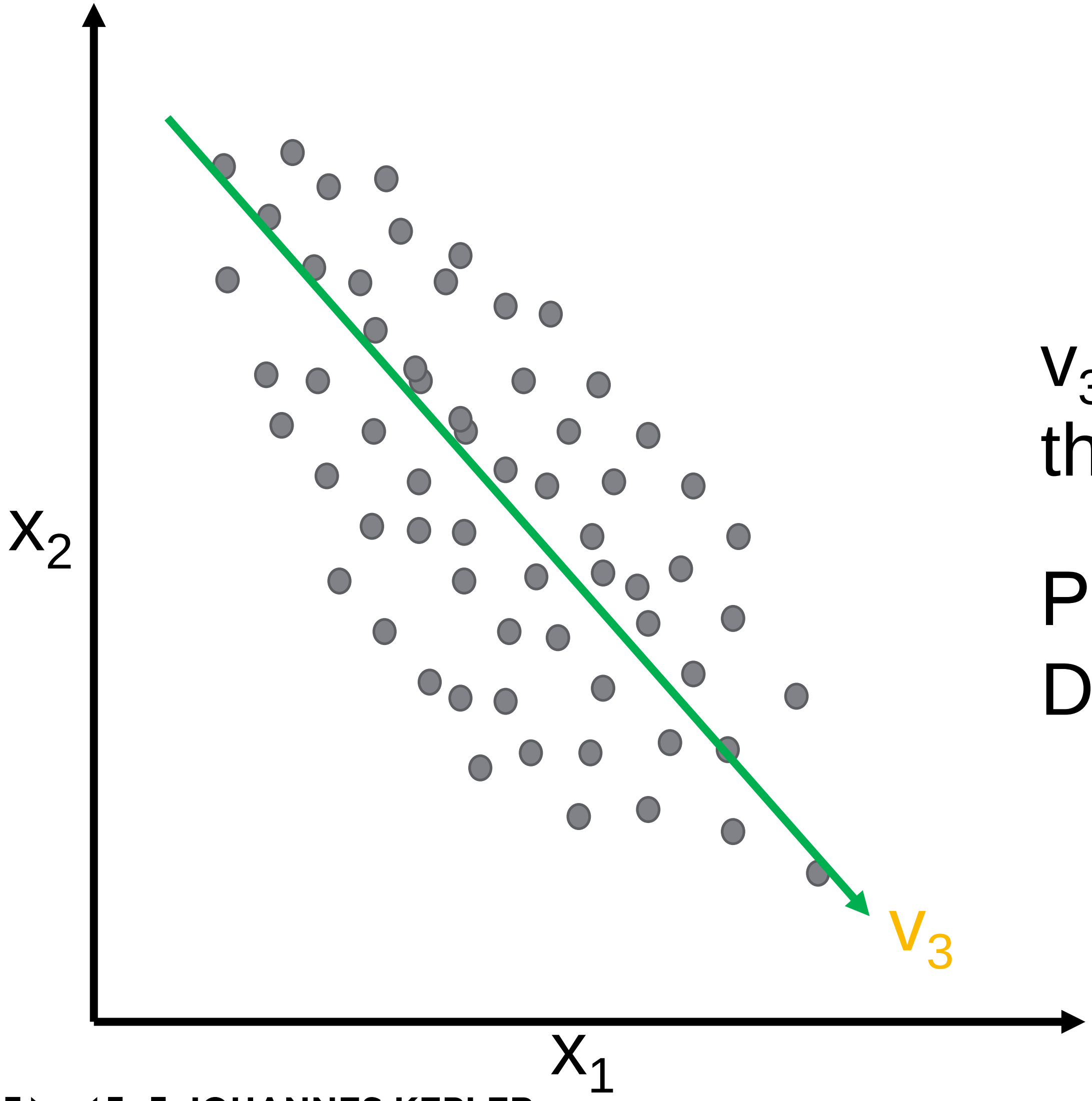
What is the best Direction?



What is the "best" direction to represent this data: v_1, v_2, or v_3 ?

Why?

What is the best Direction?



v_3 explains most of the ***Variance*** in the Data.

Points are most “spread out” in this Direction.

Principle Component Analysis (PCA)

The Variance along ***v*** of all the Point Projections is:

$$var(v) = \sum_x \|(x - \bar{x})^T \cdot v\|^2$$

$$= \sum_x v^T (x - \bar{x})(x - \bar{x})^T v$$

$$= v^T \left[\sum_x (x - \bar{x})(x - \bar{x})^T \right] v$$

$$= v^T C v$$, where $C = X^T X$ is the covariance matrix of X

v = Eigenvectors of C
var(v) = Eigenvalues of v

PCA is used to compute v,var(v)

We are interested in v with the largest var(v)!

This works also in higher dimensions.

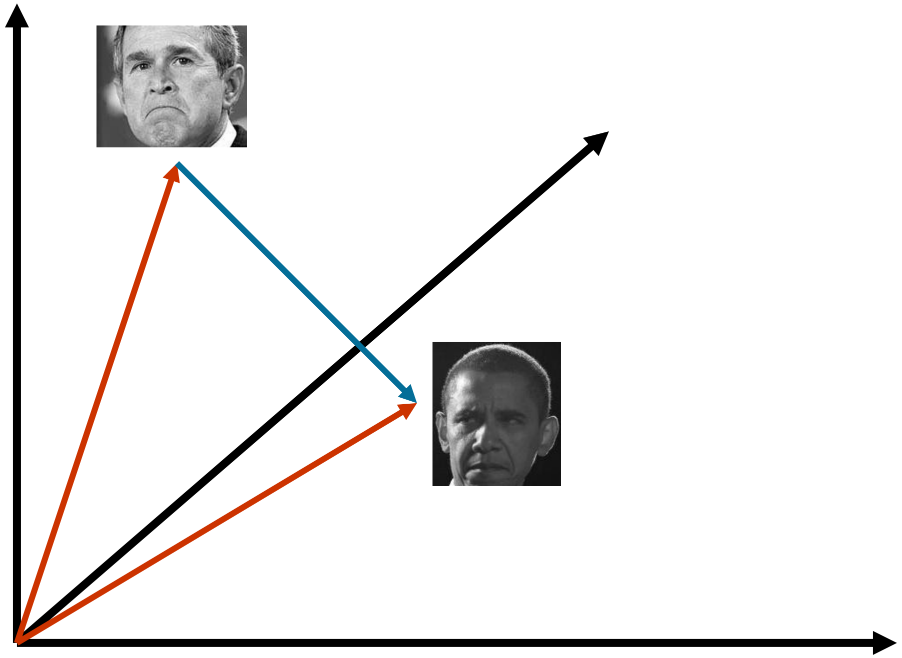
Example: Eigenfaces

Test Data

Does a similar Image
of that Person exist?

Training Data

Example: Eigenfaces



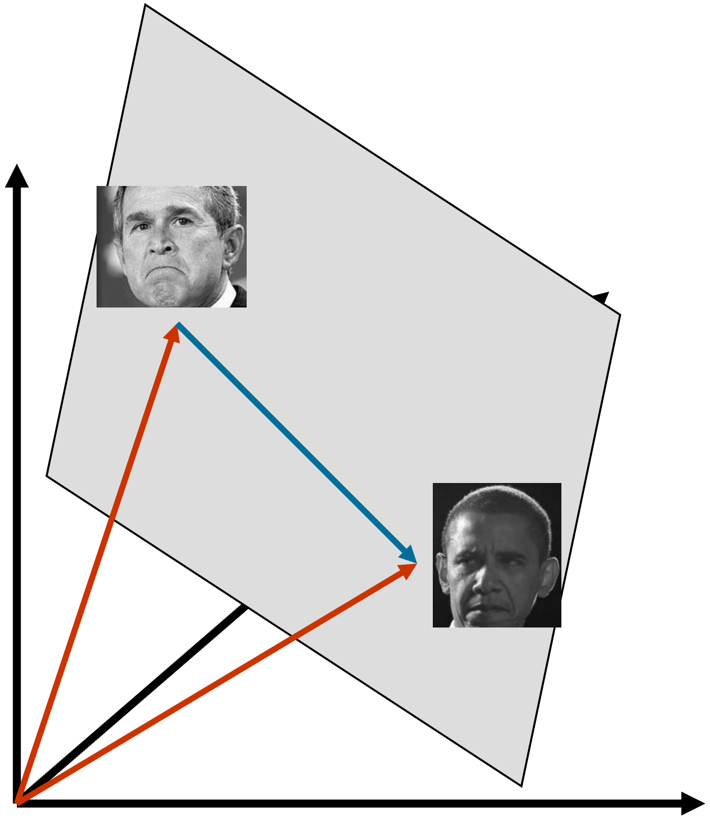
Here, an Image represents a Point in a High Dimensional space

- An N x M (binary) Image is a Point in R^{NM}
- We can define Vectors in this Space

Face Space

- When viewed as Vectors of Pixel Values, Face Images are extremely high-dimensional
  - 100 x 100 Image = 10,000 Dimensional
- But very few 10,000-dimensional Vectors are valid Faces
- Let's find a lower-dimensional Representation using PCA!

Dimensionality Reduction



The set of Faces is a "Subspace" of the Set of Images

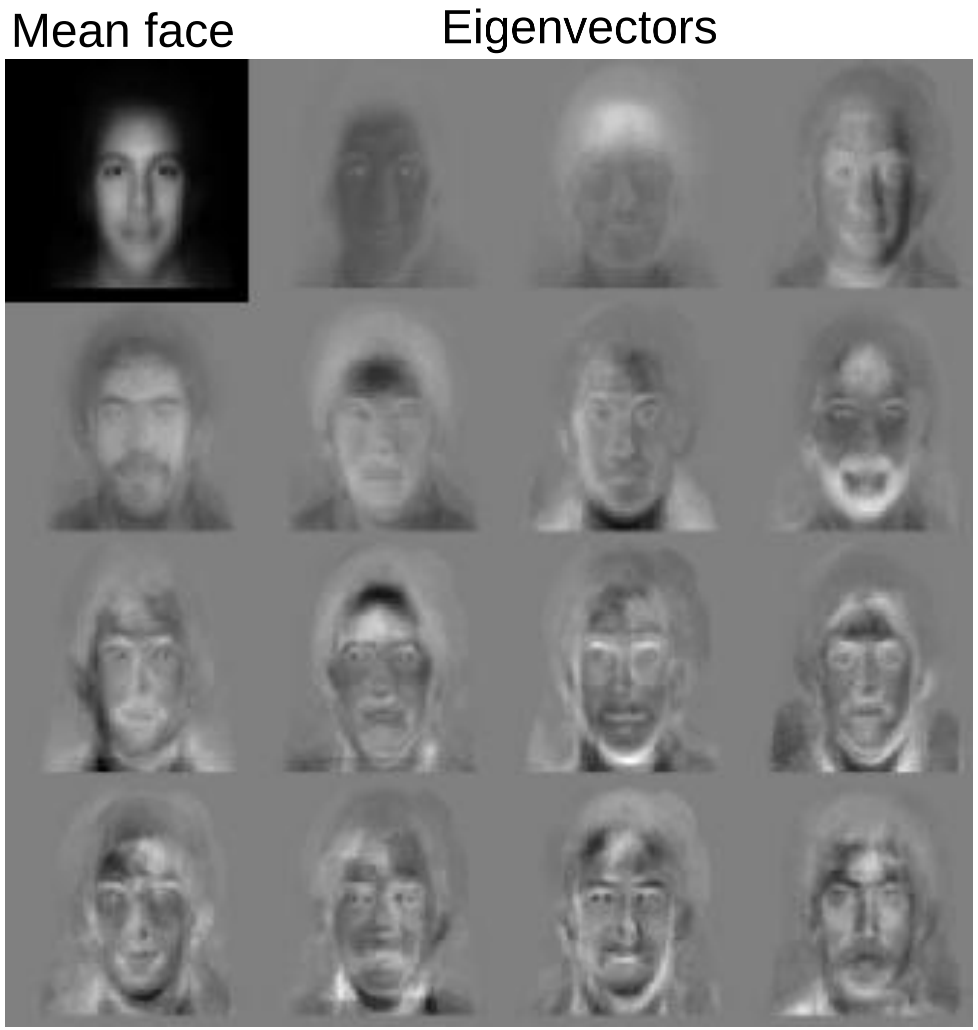
- Suppose it is K-dimensional
- We can find the best Subspace using PCA
- This is like fitting a "Hyperplane" to the Set of Faces spanned by vectors $\mathbf{v_1}, \mathbf{v_2}, \dots, \mathbf{v_K}$

Any Face: $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_k\mathbf{v}_\mathbf{k}$

Eigenfaces

PCA extracts a set of vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots$

Each one of these Vectors is a Direction in Face space. What do these look like?



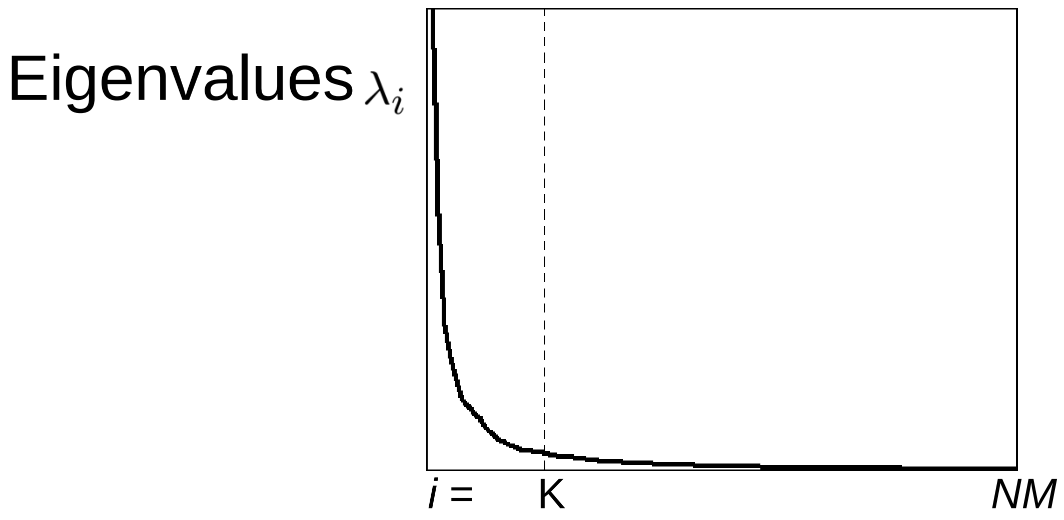
Projecting onto Eigenfaces

The Eigenfaces $\mathbf{v_1}, \ldots, \mathbf{v_K}$ span the Space of Faces. A Face is converted to Eigenface coordinates by

$$\mathbf{x} \rightarrow (\underbrace{(\mathbf{x}-\overline{\mathbf{x}}) \cdot \mathbf{v}_1}_{a_1}, \underbrace{(\mathbf{x}-\overline{\mathbf{x}}) \cdot \mathbf{v}_2}_{a_2}, \ldots, \underbrace{(\mathbf{x}-\overline{\mathbf{x}}) \cdot \mathbf{v}_\mathbf{K}}_{a_K})$$

$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_K\mathbf{v}_\mathbf{K}$$

Choosing the Dimension K



How many Eigenfaces to use? Look at the Decay of the Eigenvalues

- The Eigenvalue tells you the amount of Variance "in the Direction" of that Eigenface
- Ignore Eigenfaces with low Variance

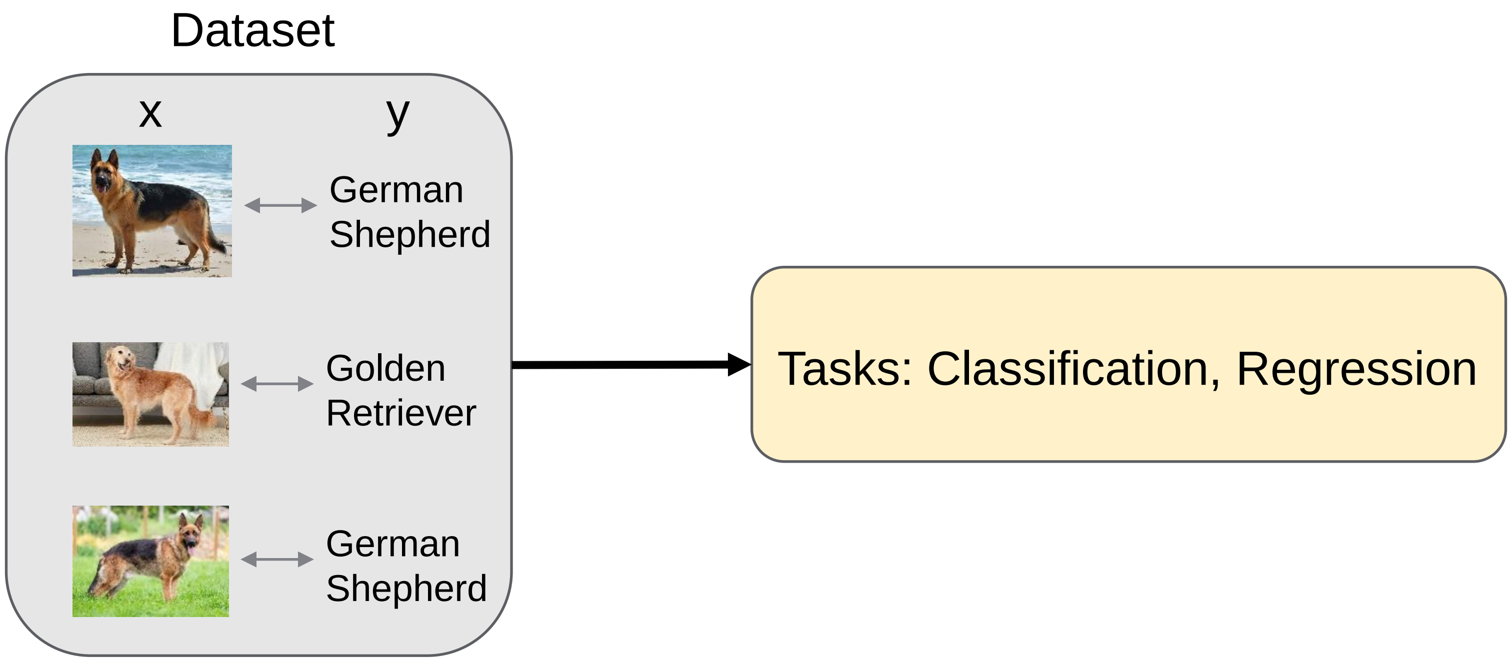
Face Recognition with Eigenfaces

- Project Input Face to Eigenfaces (i.e., represent it by a K-dimensional vector: $\mathbf{a_1, a_2, ..., a_K}$)
- Find the Nearest-Neighbor in Training Image Set (also projected to Eigenfaces)
- Pro: Speed, Size, Fairly Invariant to Resolution
- Cons: Fails quickly if Variation between Training Images and Test Image becomes to large

Dimensionality Reduction Methods

- PCA (Principal Component Analysis):
  - Find Projection that maximize the Variance
- ICA (Independent Component Analysis):
  - Very similar to PCA except that it assumes non-Gaussian features
- Multidimensional Scaling:
  - Find Projection that best preserves Inter-Point Distances
- LDA(Linear Discriminant Analysis):
  - Maximizing the Component Axes for Class-Separation
- …
- …

Supervised Learning – Example: Image Classification



Use labeled Data for a Prediction Task.

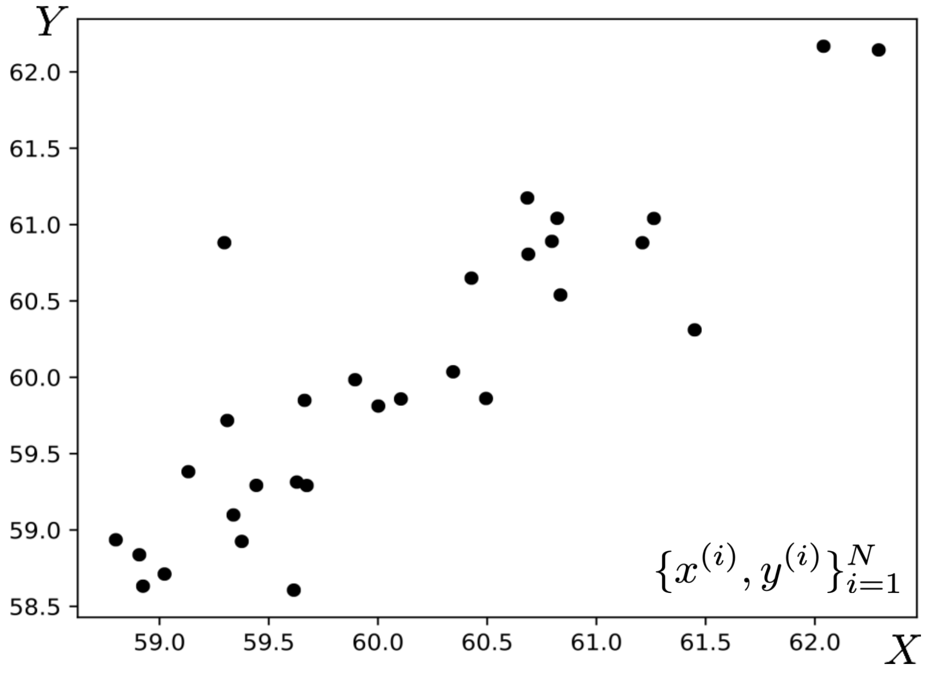
Learning from Labeled Training Data

Training data

$$\begin{matrix} \{x^{(1)}, y^{(1)}\} \\ \{x^{(2)}, y^{(2)}\} \\ \{x^{(3)}, y^{(3)}\} \\ \dots \end{matrix} \quad \rightarrow \quad \boxed{\text{Learner}} \quad \rightarrow \quad f : X \rightarrow Y$$

Linear Regression

Training data



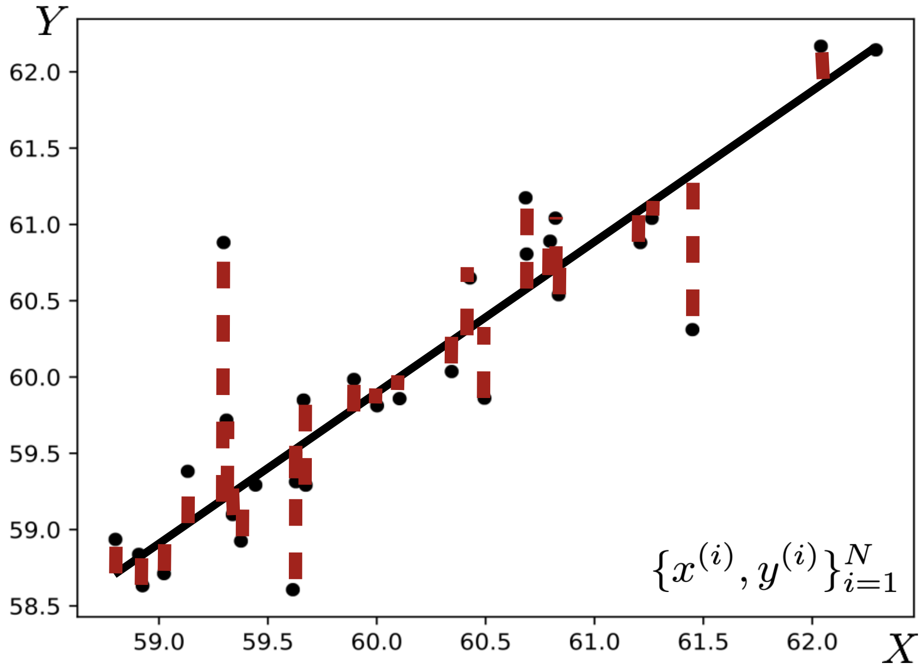
$\rightarrow$ Learner $\rightarrow$ $f_\theta(x) = \theta_1 x + \theta_0$

Hypothesis space

The relationship between X and Y is roughly linear: $y \approx \theta_1 x + \theta_0$

Linear Regression

Training data



Search for the Parameters, $\theta = \{\theta_0, \theta_1\}$, that best fit the Data.

$$f_\theta(x) = \theta_1 x + \theta_0$$

Best Fit in what sense?

The Least-Squares **Objective** (aka **Loss**) says the best Fit is the Function that minimizes the Squared Error between Predictions and Target Values:

$$\mathcal{L}(\hat{y}, y) = \boxed{(\hat{y} - y)}^2 \quad \hat{y} \equiv f_\theta(x)$$

Linear Regression

Data

$$\{x^{(i)}, y^{(i)}\}_{i=1}^{N} \rightarrow$$

Learner

Objective

$$\mathcal{L}(f_\theta(x), y) = (f_\theta(x) - y)^2$$

Hypothesis space

$$f_\theta(x) = \theta_1 x + \theta_0$$

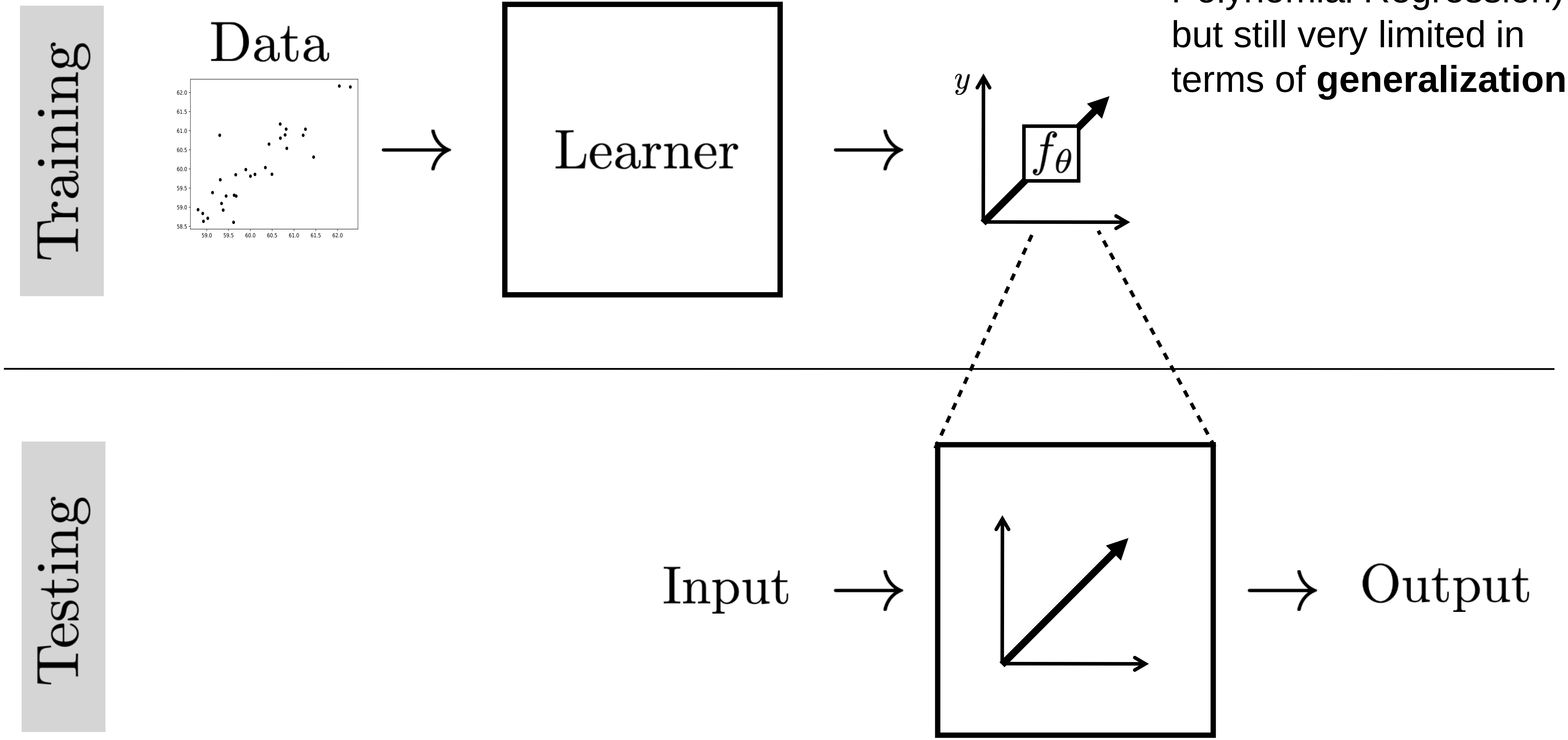
Optimizer

$$\theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

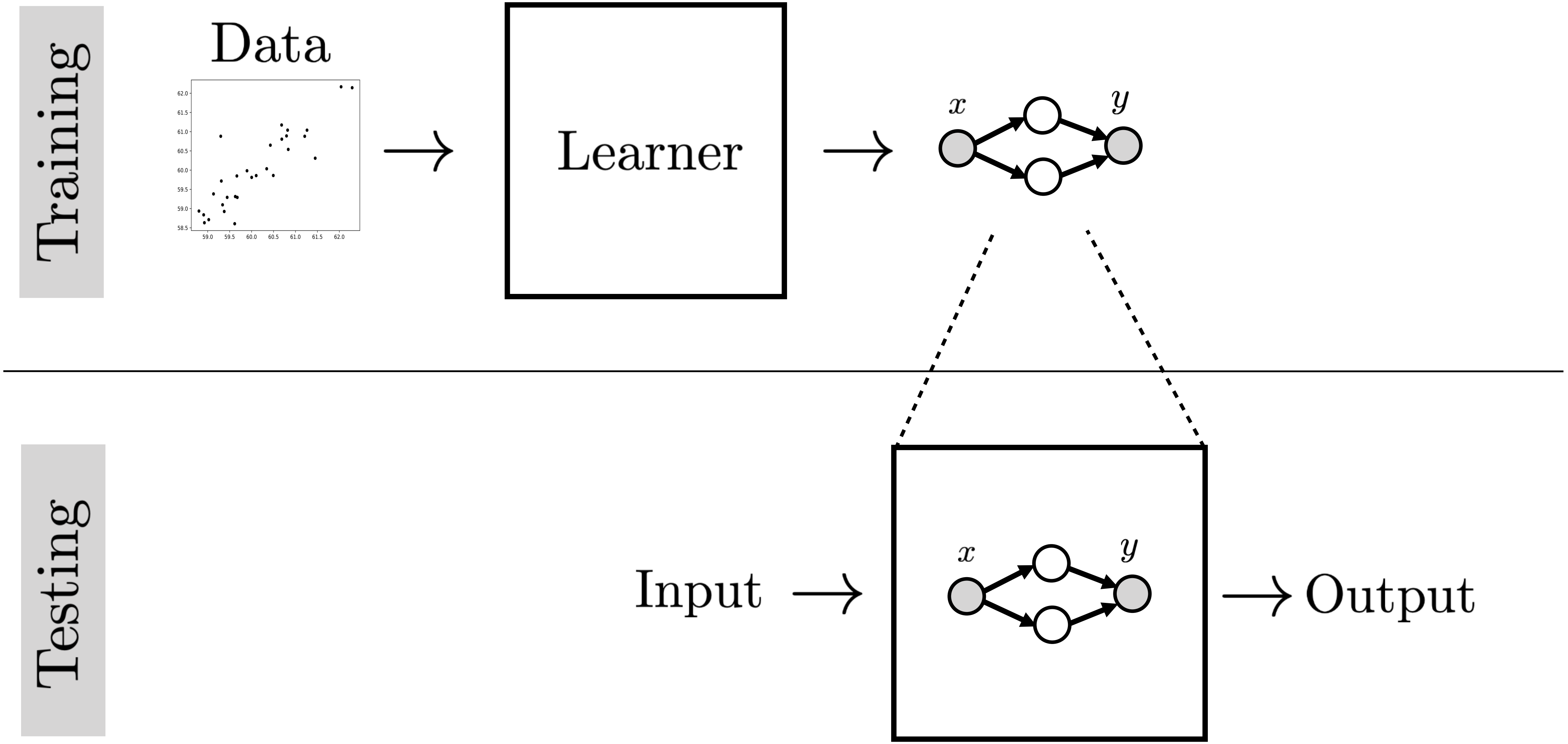
$$\rightarrow f$$

Linear Regression

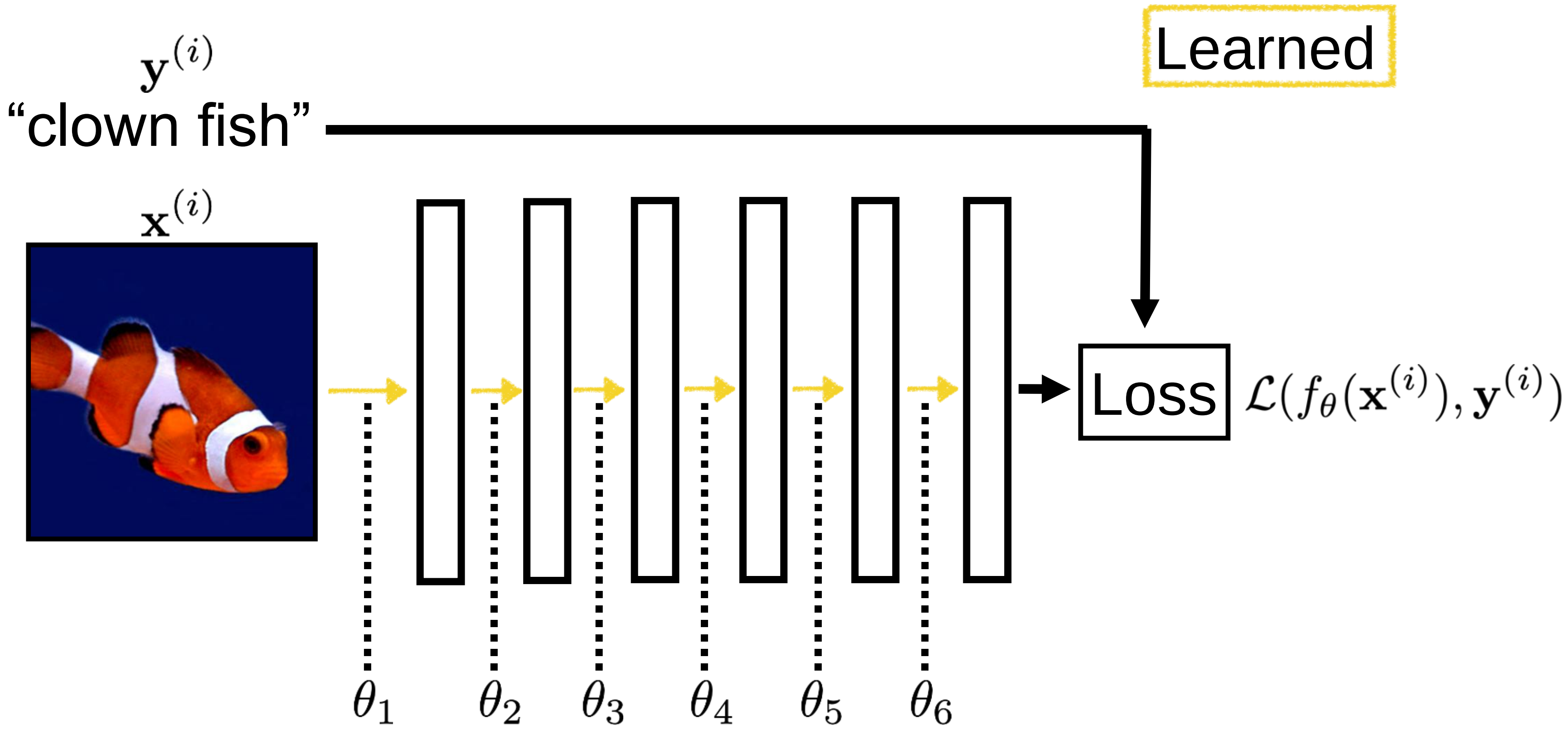
We could assume more complex Functions (like Polynomial Regression) – but still very limited in terms of **generalization**…



Deep Learning

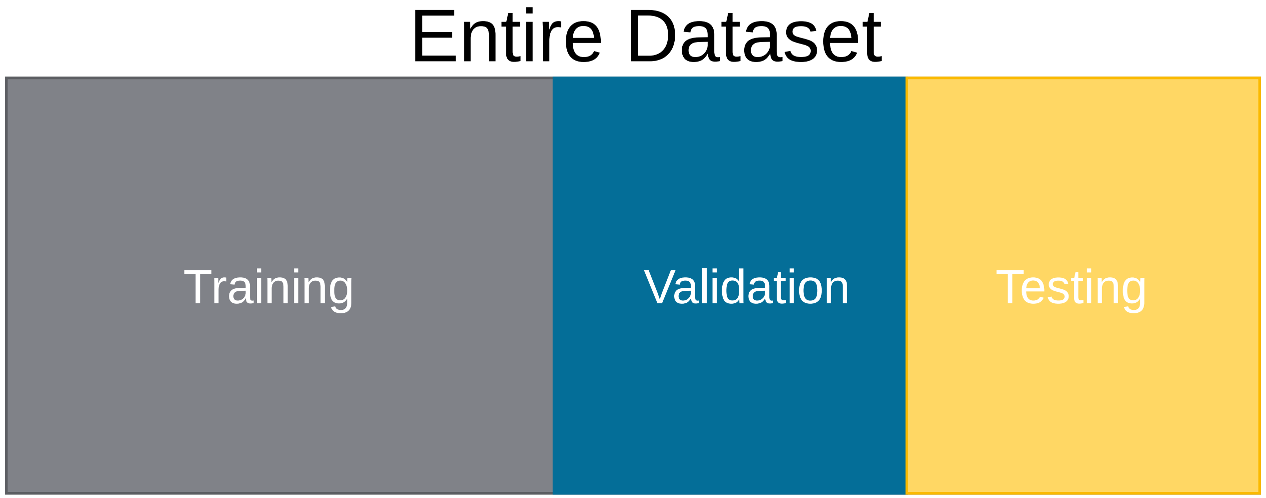


Deep Learning



$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}^{(i)}), \mathbf{y}^{(i)})$$

Deep Learning

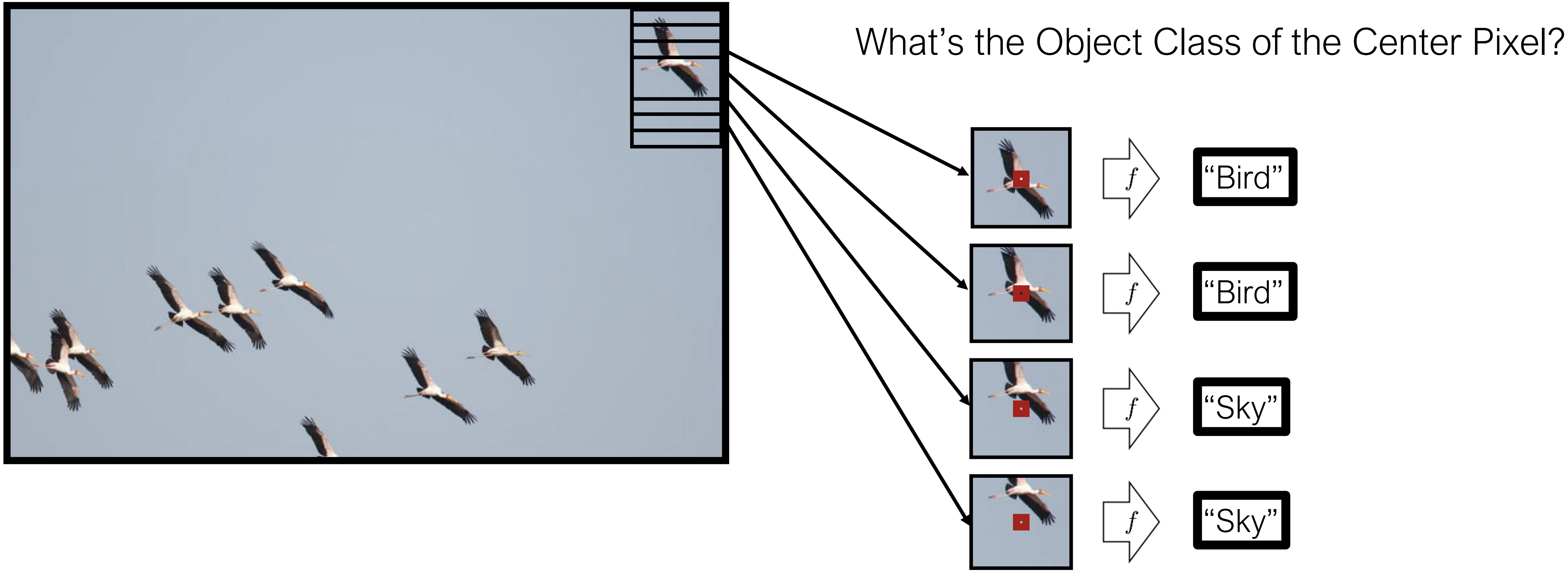


Training: Used to train Model Parameters

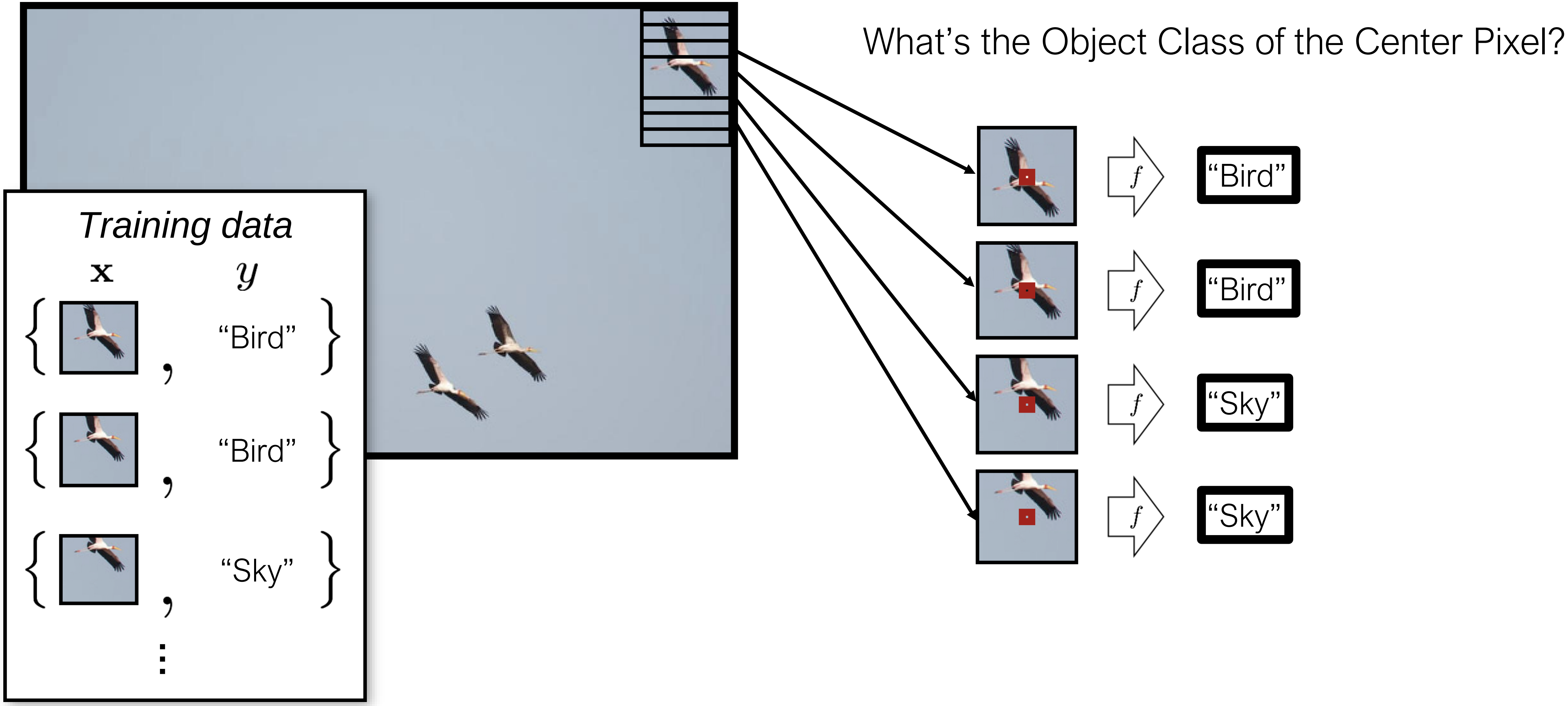
Validation: Used to tune Hyperparameters (e.g., λ in regularized regression)

Testing: Used to report final Accuracy (shouldn’t be touched before!)

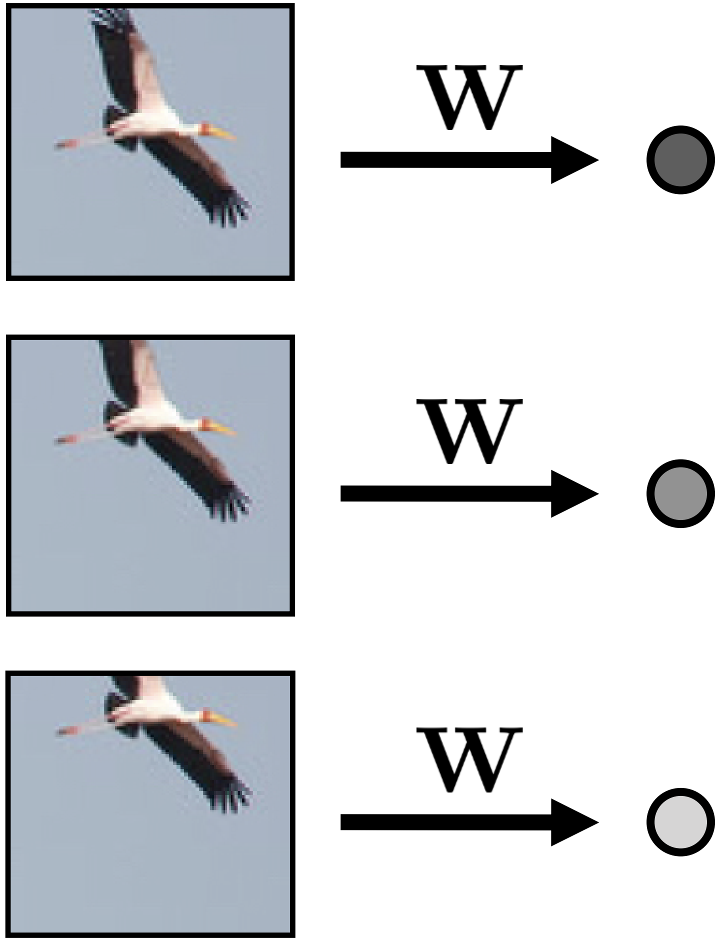
Classification with Convolution



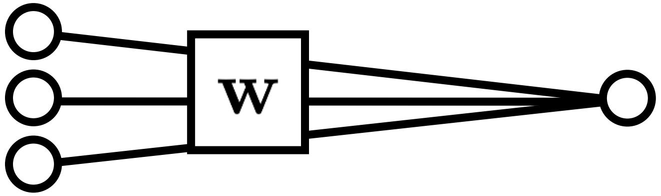
Classification with Convolution



Classification with Convolution



W computes a weighted Sum of all Pixels in the Patch

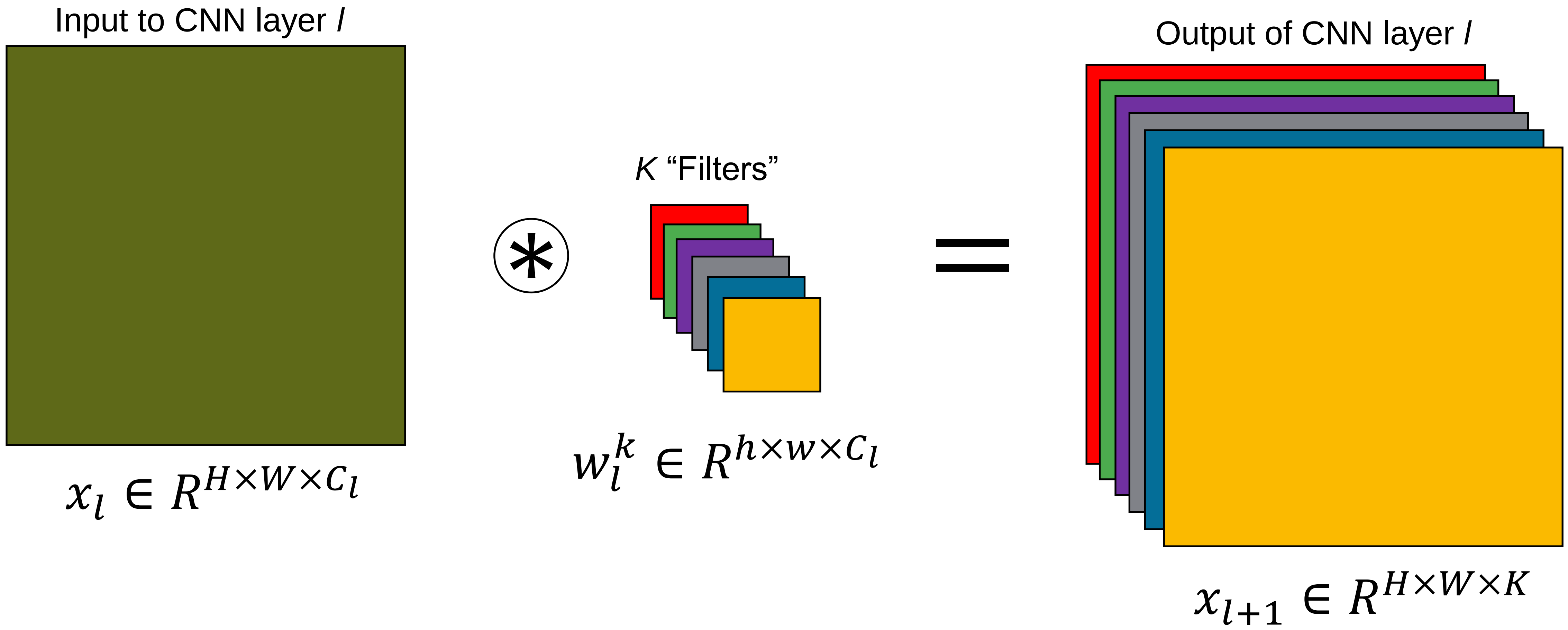


W is a Convolutional Kernel applied to the Full Image!

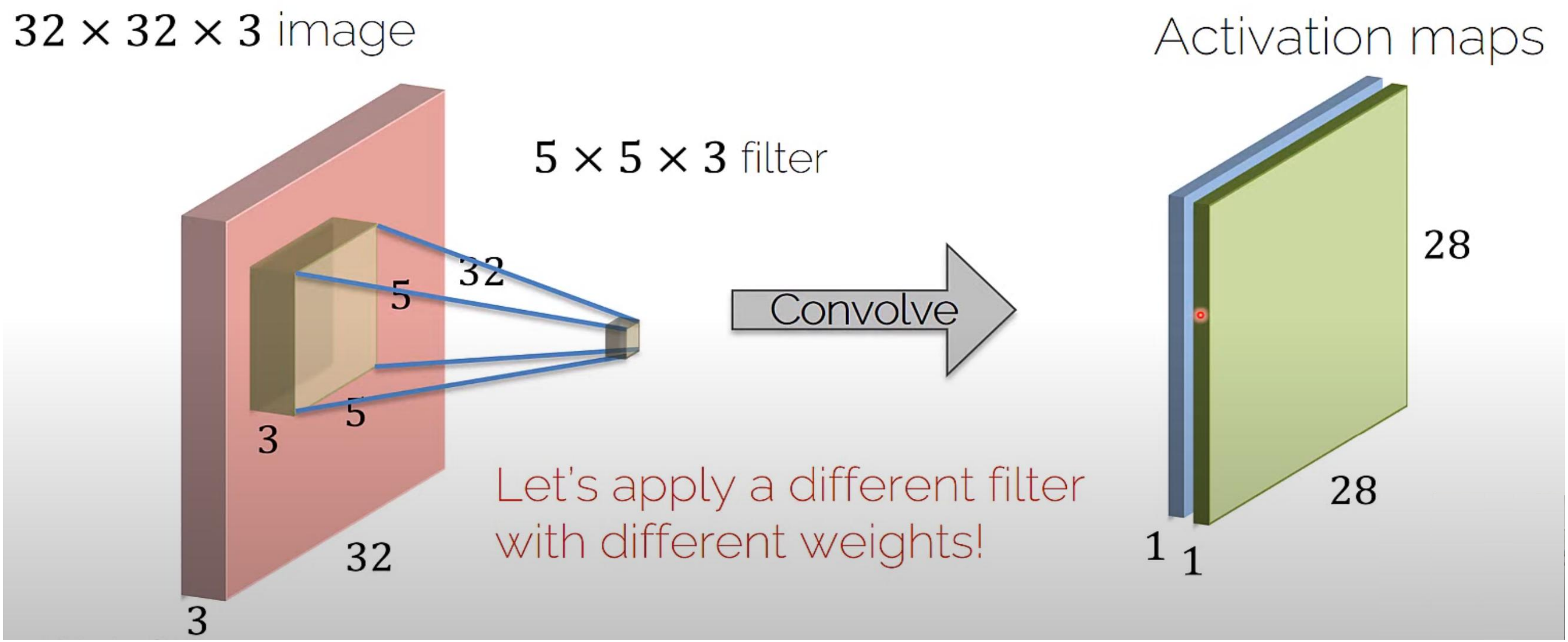
It gives highest Response if the Kernel matches the Image Content

It is Shift Invariant. But how about Scale, Rotation, other Poses, etc.?

Convolutional Neural Networks (CNNs)



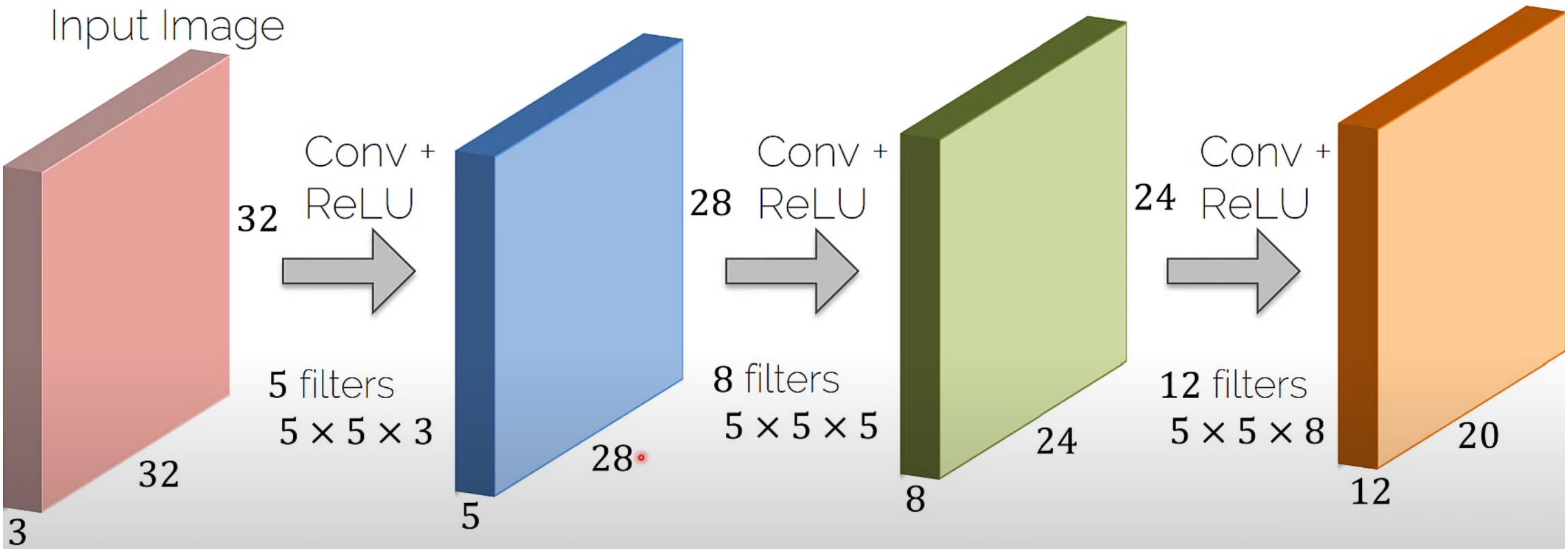
Convolutional Layer



Activation (Feature) Maps

- Each Layer can be thought of as a set of C **Feature Maps** aka **Channels**
- Each Feature Map is an NxM Image
- ReLU (Rectified Linear Unit, *f(x)=max(0,x)*) Activation Function is used to introduce Nonlinearity in a Neural Network, helping mitigate the Vanishing Gradient Problem

Multiple Convolutional Layers

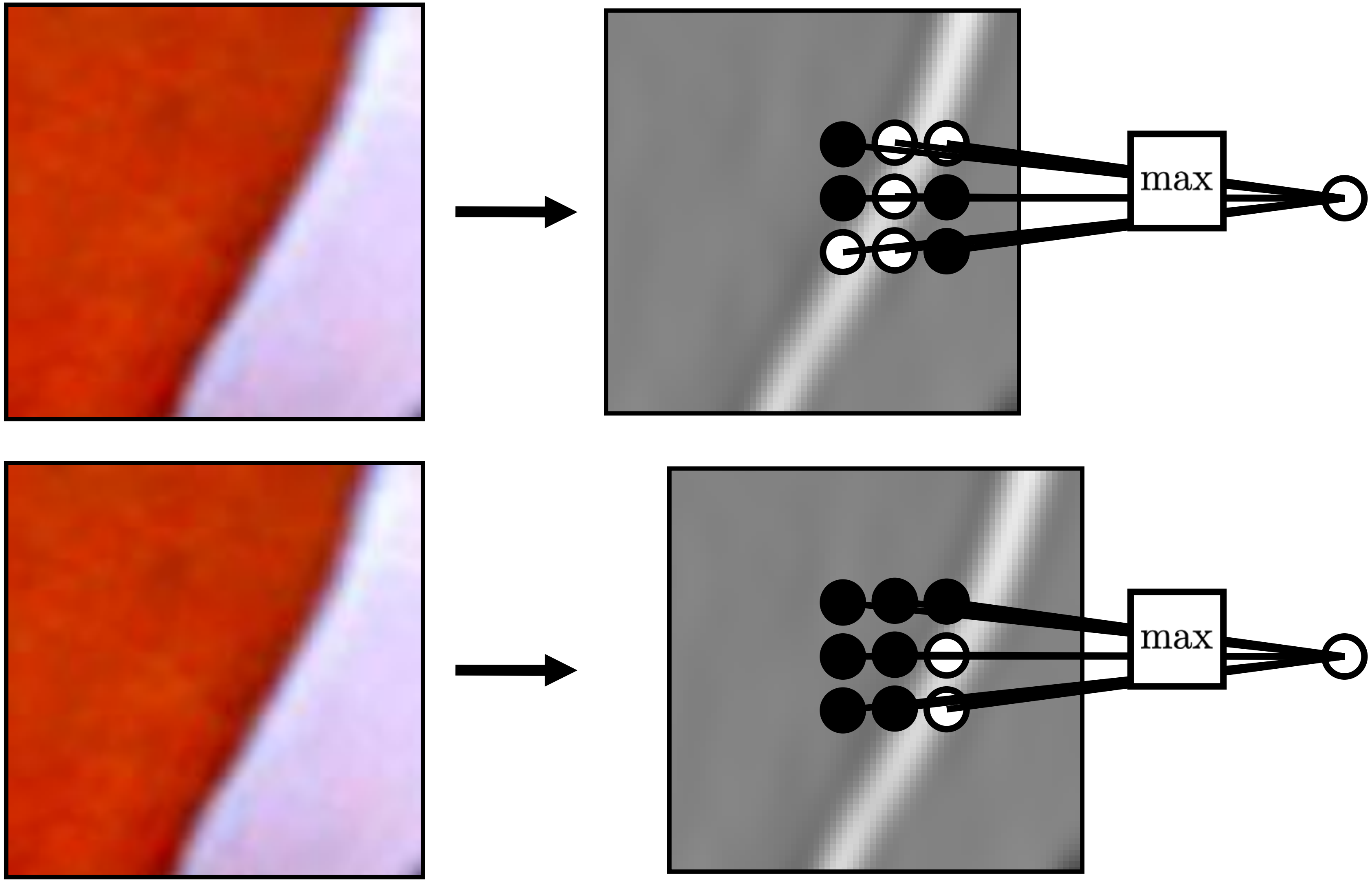


Pooling

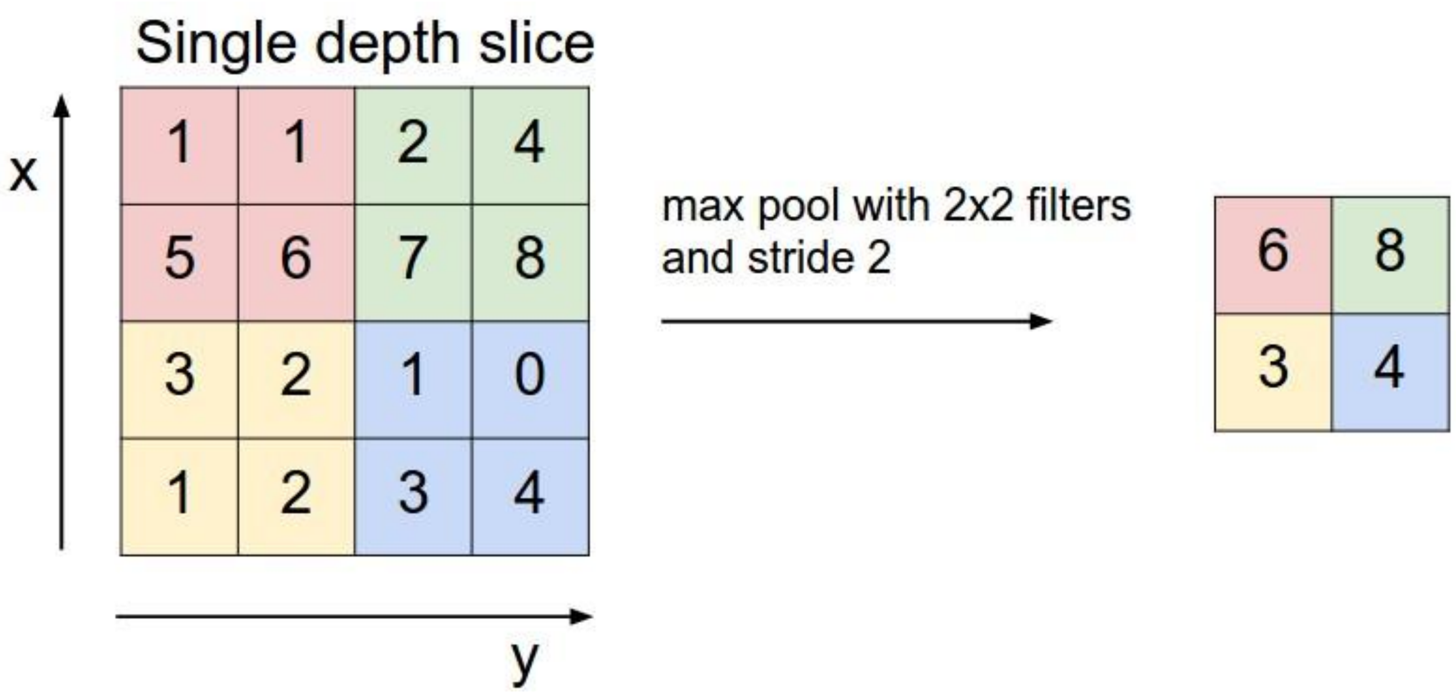
Max pooling
(usually 2x2 or 3x3)

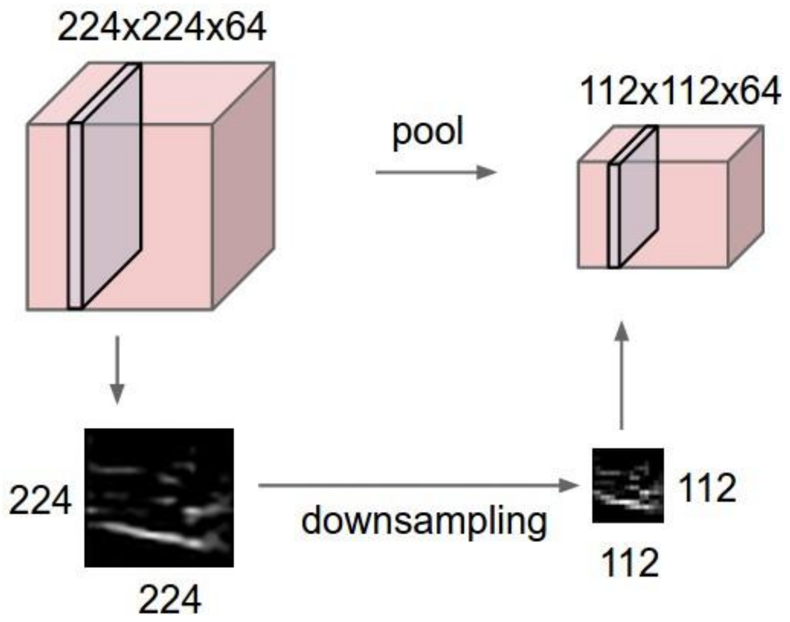
$$y_j = \max_{j \in \mathcal{N}(j)} h_j$$

provides large Response regardless of exact Position of Edge



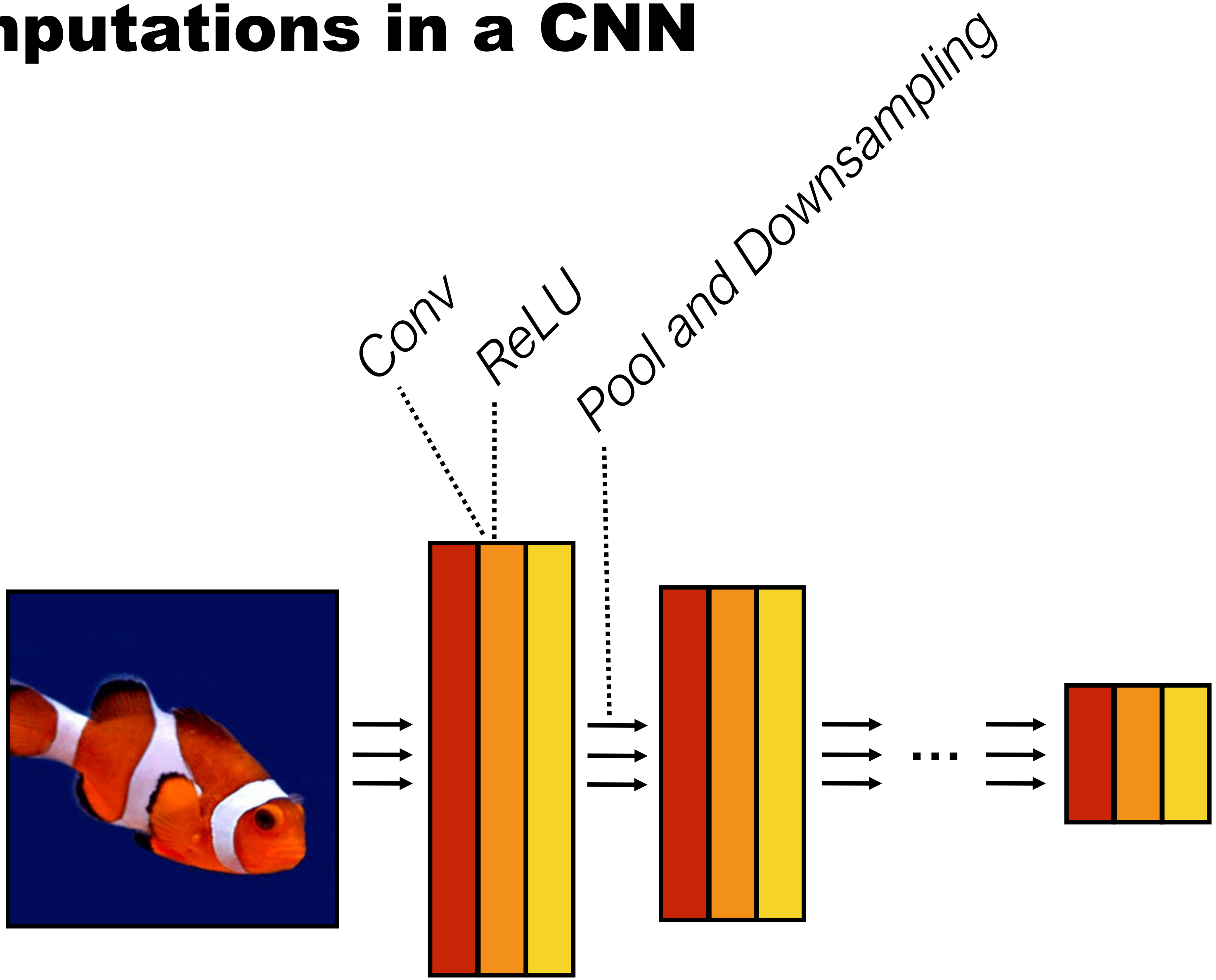
Pooling and Down-Sampling



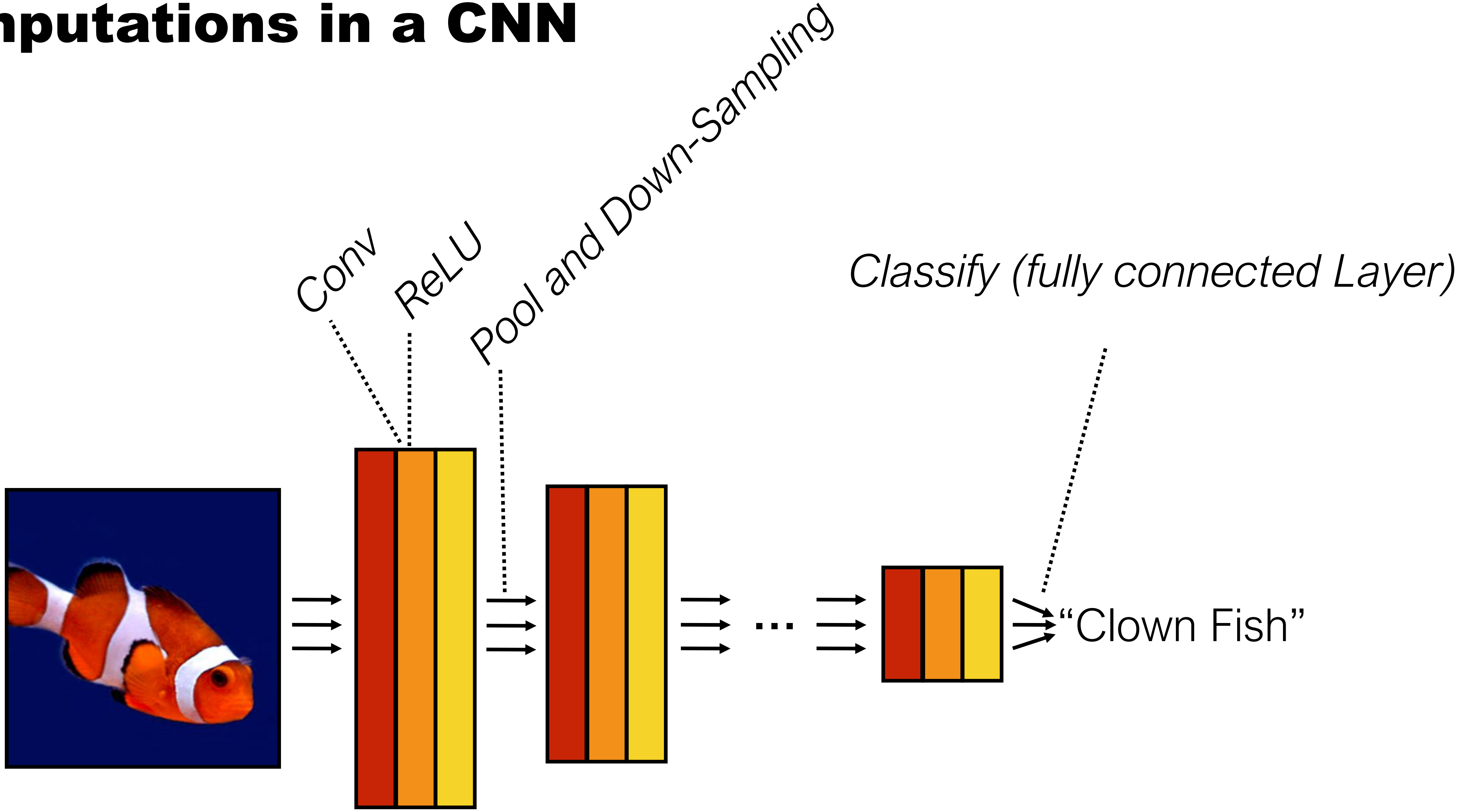


Stride: shift of kernel (for normal Convolution it is 1)

Computations in a CNN

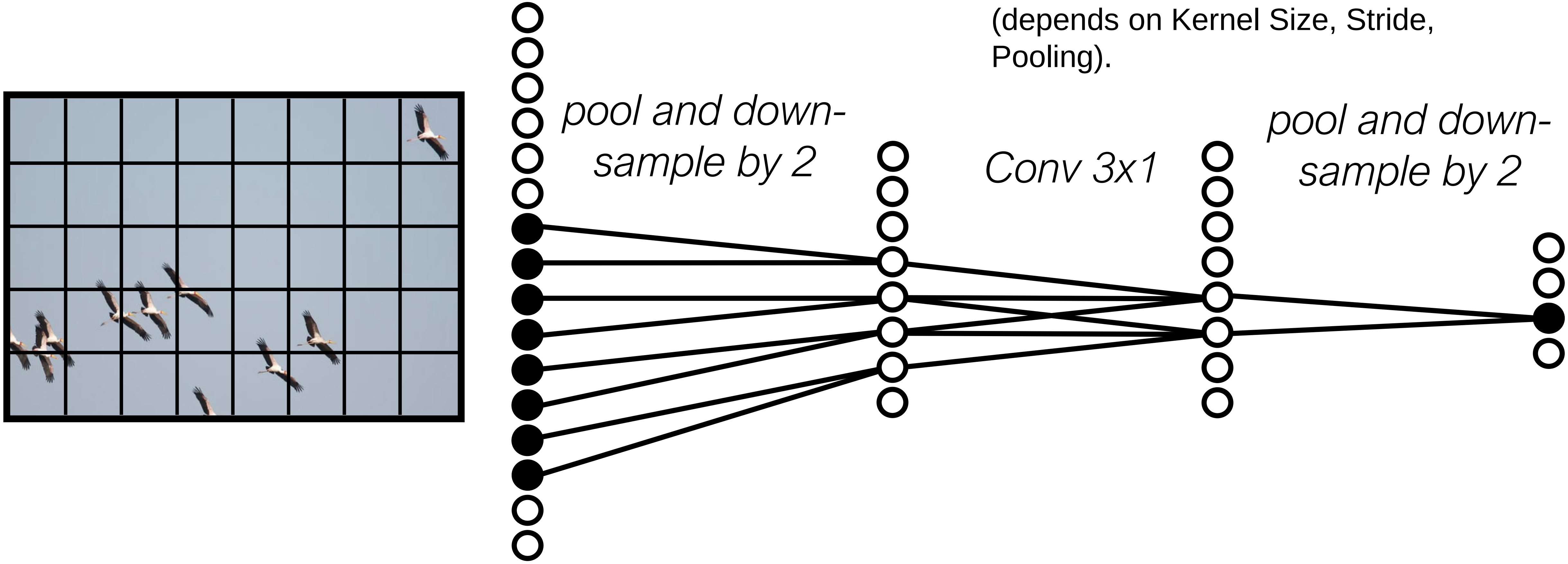


Computations in a CNN



Receptive Field

Receptive Field is the Region of the Image that a particular Neuron in a Convolutional Layer is “looking at” or taking into account when making its Predictions or Feature Extractions (depends on Kernel Size, Stride, Pooling).



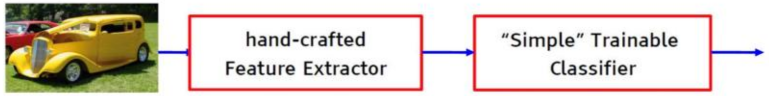
Course Overview

CW	Topic	Date	Place	Lab
41	Introduction and Course Overview	07.10.2025	Zoom	Lab 1
42	Capturing Digital Images	14.10.2025	Zoom	Lab 2
43	Digital Image Processing	21.10.2025	Zoom	Assignment 1
44	Machine Learning	28.10.2025	Zoom	
45	Feature Extraction	04.11.2025	Zoom	Open Lab 1
46	Segmentation	11.11.2025	Zoom	Assignment 2
47	Optical Flow	18.11.2025	Zoom	Open Lab 2
48	Object Detection	25.11.2025	Zoom	Assignment 3
49	Multi-View Geometry	02.12.2025	Zoom	Open Lab 3
50	3D Vision	09.12.2025	Zoom	Assignment 4
3	Trends in Computer Vision	13.01.2026	Zoom	
4	Q&A	20.01.2026	Zoom	Open Lab 4
5	Exam	27.01.2026	HS1 (Linz), S1/S3 (Vienna), S5 (Bregenz)	
9	Retry Exam	24.02.2026	tba	

Next Week: Feature Extraction

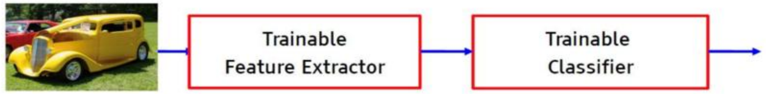
Model-Based vs. Learning-Based Feature Extraction

- Fixed engineered features (or kernels) + trainable classifier



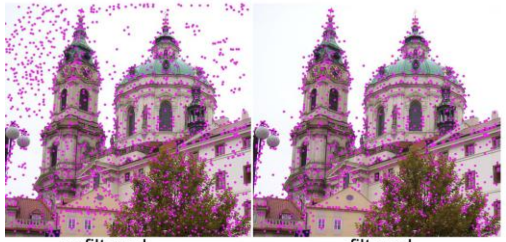
vs.

- End-to-end learning / feature learning / deep learning



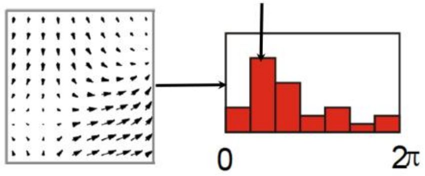
JOHANNES KEPLER UNIVERSITY LINZ

Example: Scale-Invariant Feature Transform (SIFT)



unfiltered filtered

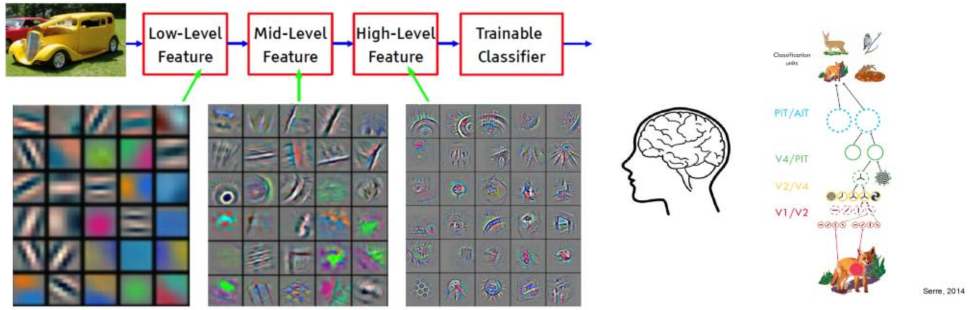
(3) Filter out Features in low-contrast Regions (Noise)



(4) Determine Feature (i.e., Gradient) Orientations and sort them into Histogram (largest Bin = main Orientation)

JOHANNES KEPLER UNIVERSITY LINZ

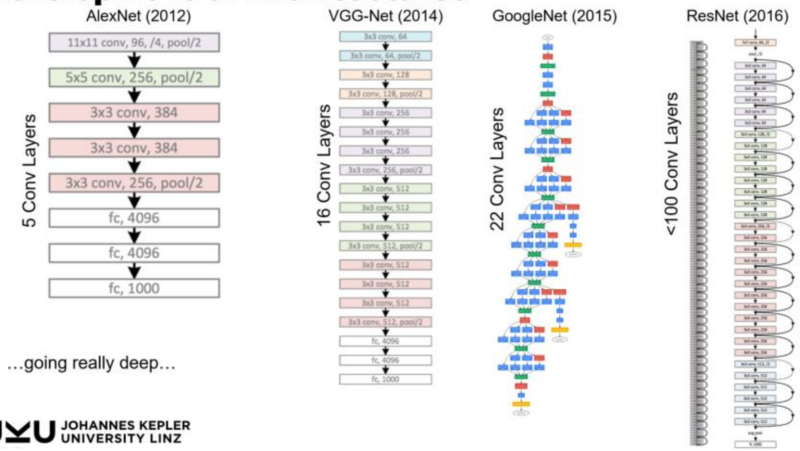
From Low-Level to High-Level Features



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

JOHANNES KEPLER UNIVERSITY LINZ

Development of Architectures



...going really deep...

JOHANNES KEPLER UNIVERSITY LINZ

Thank You