# THE PATH TO DEEP LEARNING

Ștefan Máthé

BOSCH

# Outline

- Math Refresher: Probability Theory
- The Challenge
- Machine Learning
- The Path to Deep Learning

# MATH REFRESHER: PROBABILITY THEORY

# Quick Math Refresher
## Uncertainty

- Sources of uncertainty
  - Stochastic universe
  - Incomplete observations
  - Incomplete modelling

- Approaches to measure uncertainty:
  - **Frequentist**: repeatable event outcomes
  - **Bayesian**: degree of believe

- **Random Variables**

# Quick Math Refresher
## Uncertainty

- Sources of uncertainty
  - Stochastic universe
  - Incomplete observations
  - Incomplete modelling

- Approaches to measure uncertainty:
  - **Frequentist**: repeatable event outcomes
  - **Bayesian**: degree of believe

***Governed by the same set of axioms***

- **Random Variables**

# Quick Math Refresher
## Probabilities

- Probability Distributions
  - Probability Mass Function (PMF) ⇔ discrete variables
  - Probability Distribution Function (PDF) ⇔ continuous variables

- Joint Probability Distribution: $p(x, y)$

- Prior Probability Distribution: $p(x)$

- Conditional Probability Distribution: $p(y|x) = \frac{p(x,y)}{p(x)}$

# Quick Math Refresher
## Common Probability Distributions

- Binary Variables:
  - Bernoulli
  - Multinoulli
  - Multinomial
- Continuous Variables:
  - Gaussian
  - Dirac
  - Exponential
  - Laplace

$$\mathrm{P}(X = 1) = \theta$$
$$\mathrm{P}(X = 0) = 1 - \theta$$

*Bernoulli*

$$\mathrm{P}(X = i) = \theta\ _i,\ \forall i, 1 \leq i \leq k - 1$$
$$\mathrm{P}(X = 0) = 1 - \sum_{i=1}^{k-1} \theta\ _i$$

*Multinoulli*

$$p(X = x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

*Gaussian (1-dimensional)*

$$p(X = \boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} e^{\frac{(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma (\boldsymbol{x}-\boldsymbol{\mu})}{2\sigma^2}}$$

*Gaussian (n-dimensional)*

$$p(X = x) = \delta(x - \mu) \qquad \int_a^b \delta(x) = \begin{cases} 1 & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

*Dirac (1-dimensional)*

# Quick Math Refresher

## Expectation and KL Divergence

- Expectation: “average” value of a function $f(x)$ when $x$ is drawn from $p(x)$
  - For PDFs: $\mathbb{E}_{x\sim p(x)}[f(x)] = \int_x \, f(x)p(x)\mathrm{d}x$
  - For PMFs: $\mathbb{E}_{x\sim P(x)}[f(x)] = \sum_x \, f(x)P(x)\mathrm{d}x$

- KL Divergence: “distance” between two probability distributions

$$D_{KL}[P||Q] = \mathbb{E}_{x\sim P(x)}\left[\log\frac{P(x)}{Q(x)}\right]$$

  - Not a true distance (non-negative, but asymmetric and does not obey the triangle inequality)

# THE CHALLENGE

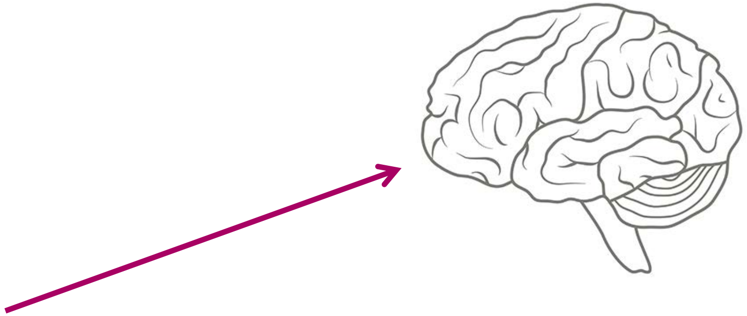
# The Challenge

## Object Class Recognition: Easy or Hard?

Is this an image of a cat?

# The Challenge

## Object Class Recognition: Easy or Hard?



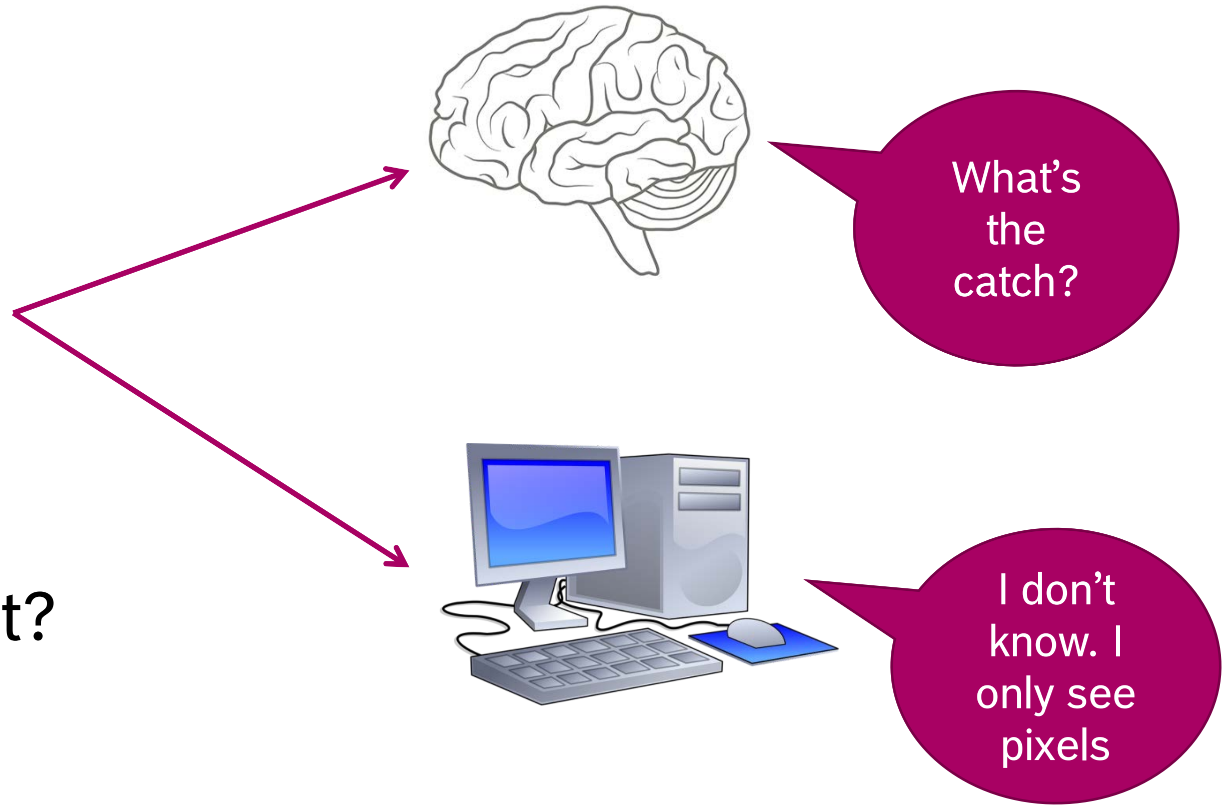
What’s the catch?

Is this an image of a cat?

# The Challenge

## Object Class Recognition: Easy or Hard?

Is this an image of a cat?



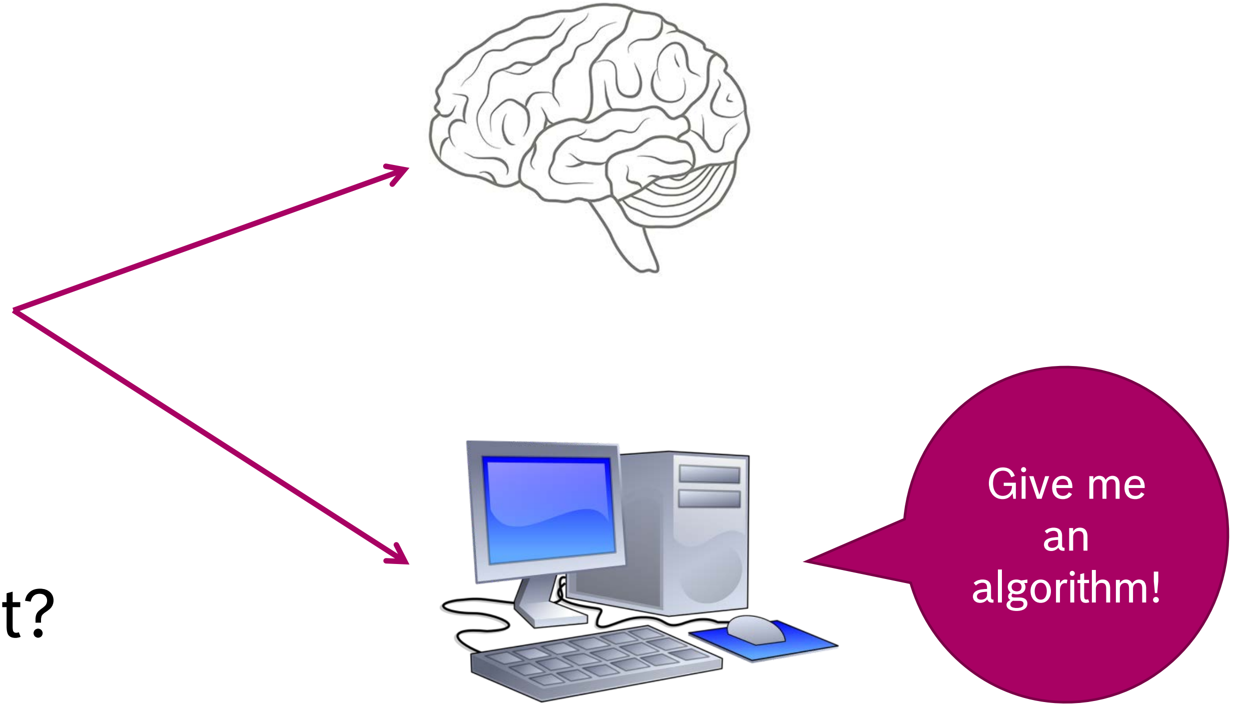
What's the catch?

I don't know. I only see pixels

# The Challenge

## Object Class Recognition: Easy or Hard?

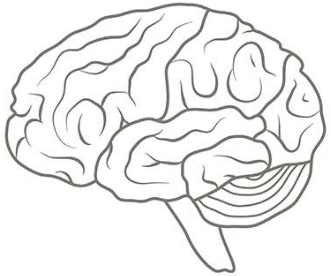
Is this an image of a cat?



Give me an algorithm!

# The Challenge
## Object Class Recognition: Easy or Hard?

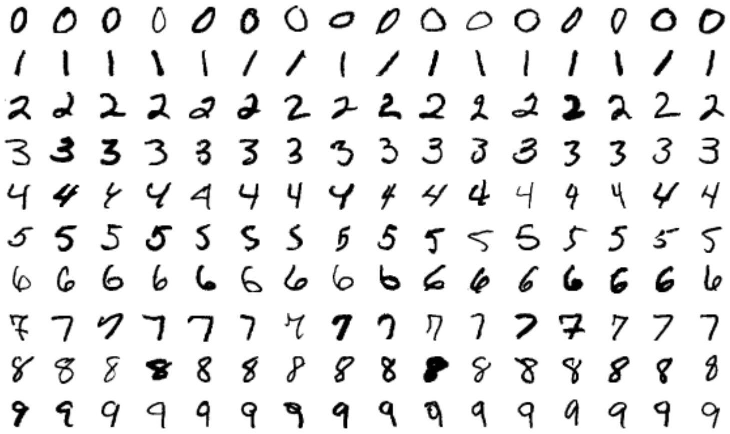
Is this an image of a cat?



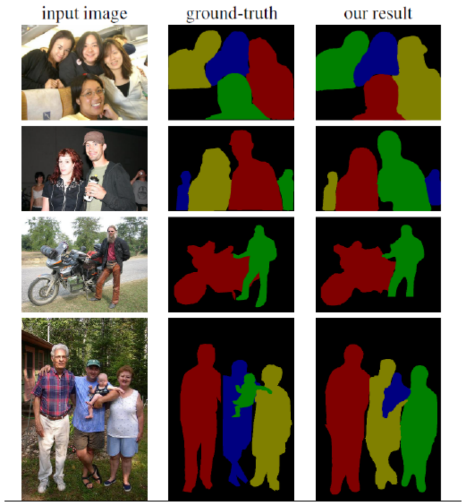
I don’t have one.
I look for a furry
mammal with
long pointy ears,
etc.

# The Challenge

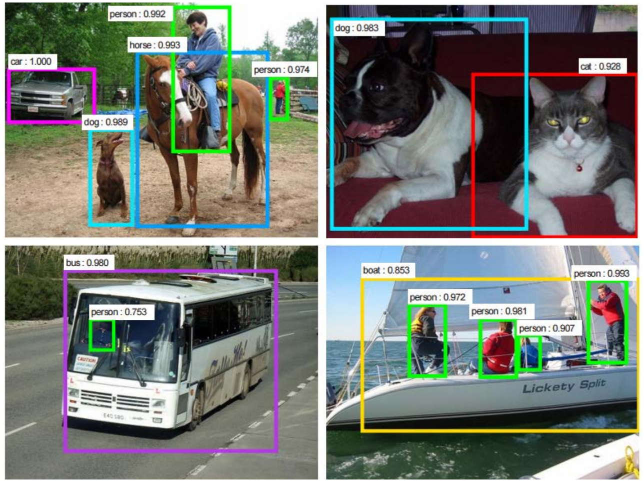
## Problems with Inconspicuous Solutions



handwritten digit recognition



instance level segmentation



object detection

# The Challenge
## Humans vs. Thinking Machines

| | | World of Thinking Machines | |
|---|---|---|---|
| | | easy | hard |
| World of Humans | easy | **(not interesting)** | • **speech recognition**<br>• **face recognition**<br>• **car driving** |
| | hard | • chess<br>• go<br>• question-answering (Quora) | • plant identification<br>• cancer diagnosis |

# The Challenge
## Humans vs. Thinking Machines

| | | World of Thinking Machines | |
|---|---|---|---|
| | | easy | hard |
| World of Humans | easy | **(not interesting)** | **• speech recognition**<br>**• face recognition**<br>**• car driving** |
| | hard | • chess<br>• go<br>• question-answering (Quora) | • plant identification<br>• cancer diagnosis |

⇩

- formal problem environments
- rule-based inference

# The Challenge
## Humans vs. Thinking Machines

|  |  | World of Thinking Machines |  |
|---|---|---|---|
|  |  | easy | hard |
| World of Humans | easy | (not interesting) | • speech recognition<br>• face recognition<br>• car driving |
|  | hard | • chess<br>• go<br>• question-answering (Quora) | • plant identification<br>• cancer diagnosis |

⇩

- formal problem environments
- rule-based inference

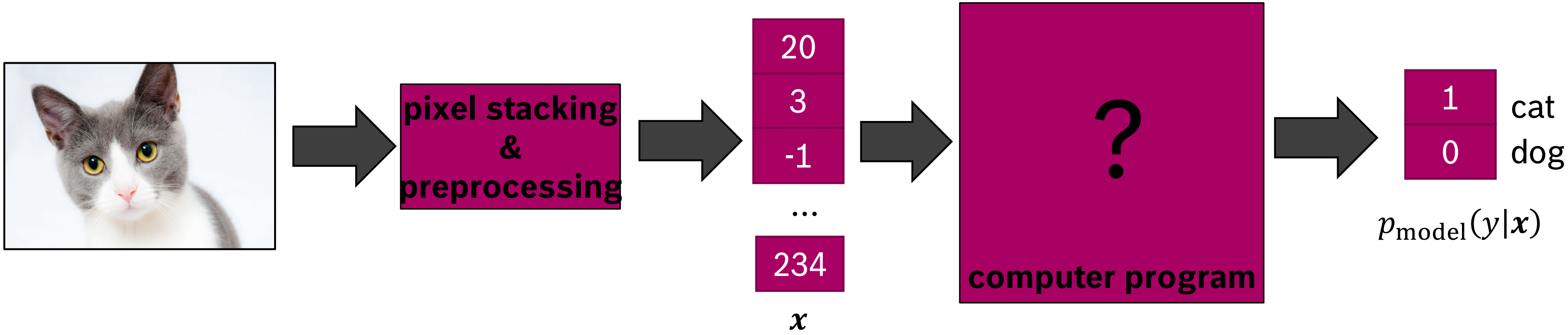
⇩

- real-world environments
- require informal knowledge

# MACHINE LEARNING

# Machine Learning
## What do we want?

- A computer program
  - Input: a feature vector $\boldsymbol{x}$ obtained from the input image
  - Output: the probability $p_{\text{model}}(y|\boldsymbol{x})$ for each object class $y$



pixel stacking & preprocessing

| $\boldsymbol{x}$ |
|---|
| 20 |
| 3 |
| -1 |
| … |
| 234 |

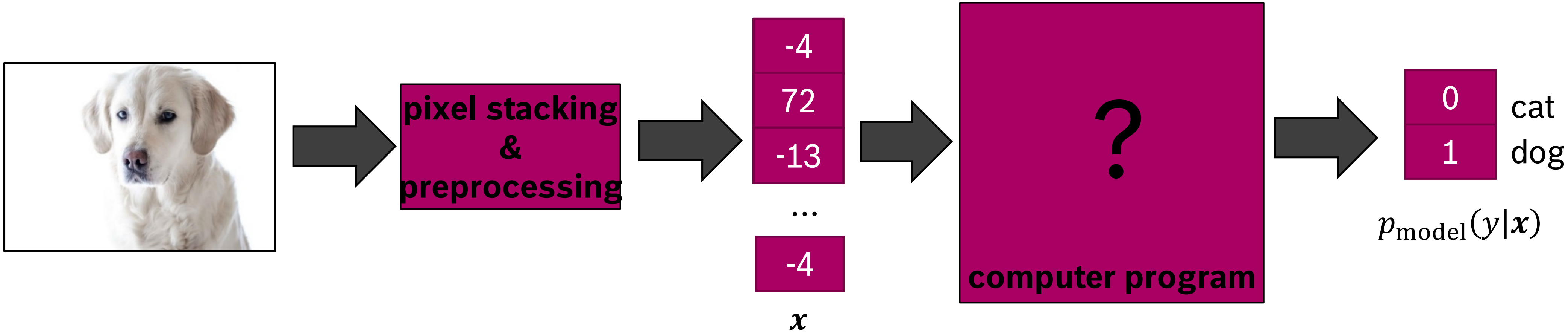
?

computer program

| $p_{\text{model}}(y\|\boldsymbol{x})$ | |
|---|---|
| 1 | cat |
| 0 | dog |

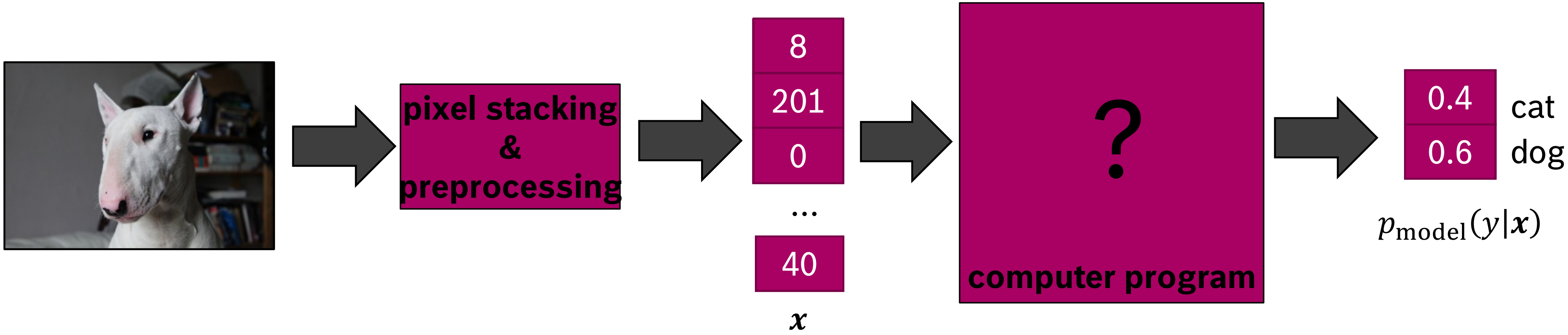
# Machine Learning
## What do we want?

- A computer program
  - Input: a feature vector $\boldsymbol{x}$ obtained from the input image
  - Output: the probability $p_{\mathrm{model}}(y|\boldsymbol{x})$ for each object class $y$



# Machine Learning
## What do we want?

- A computer program
  - Input: a feature vector $\boldsymbol{x}$ obtained from the input image
  - Output: the probability $p_{\text{model}}(y|\boldsymbol{x})$ for each object class $y$



pixel stacking & preprocessing

8
201
0
…
40
$\boldsymbol{x}$

?
computer program

| | |
|---|---|
| 0.4 | cat |
| 0.6 | dog |

$p_{\text{model}}(y|\boldsymbol{x})$

# Machine Learning
## What is a good solution?

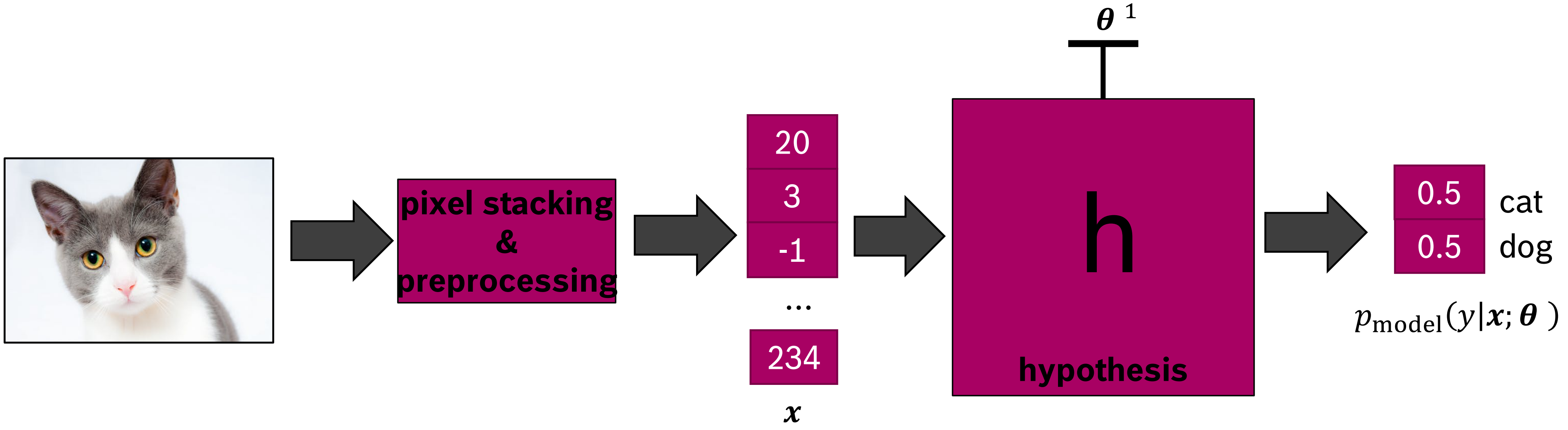
- Let $p_{\text{data}}(x, y)$ be the **true distribution** over features and labels
- Program output has a high **expected likelihood** on the true distribution:

$$\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}}[\log p_{\text{model}}(y|\boldsymbol{x})]$$

$p_{\text{data}}(x, y)$

# Machine Learning
## Parametric Hypothesis Space

- Consider a whole set H of possible programs H, called **hypothesis space**
- Assume each hypothesis is uniquely characterized by a **parameter vector $\boldsymbol{\theta}$**
- The parameter vector changes the behavior of our program



**pixel stacking & preprocessing**

20

3

-1

…

234

$\boldsymbol{x}$

$\boldsymbol{\theta}^{1}$

h

**hypothesis**

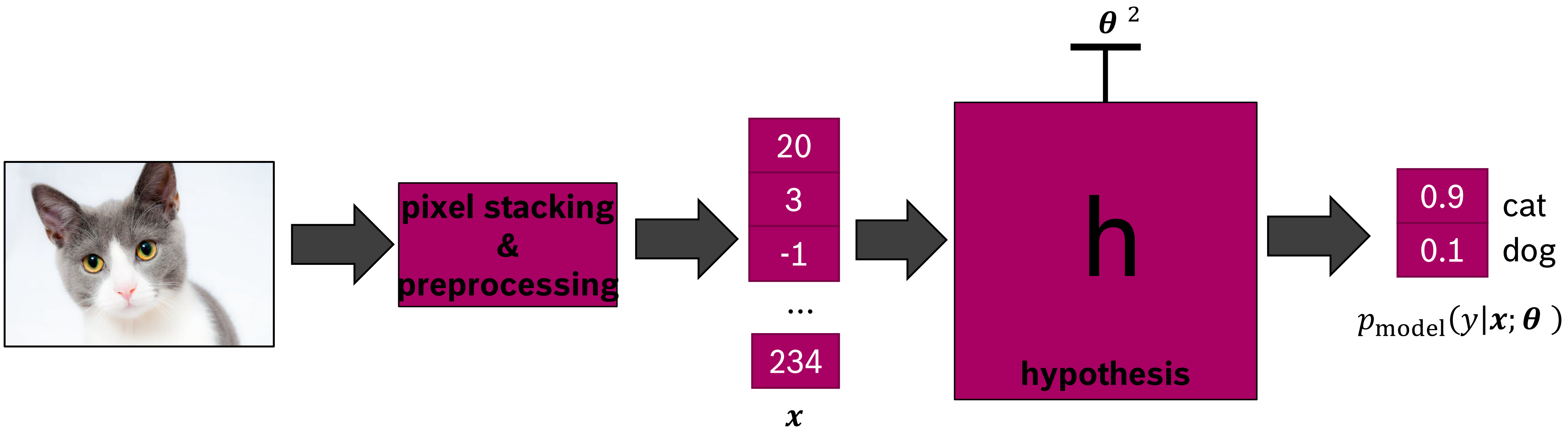
0.5 cat

0.5 dog

$p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})$

# Machine Learning
## Parametric Hypothesis Space

- Consider a whole set H of possible programs H, called **hypothesis space**
- Assume each hypothesis is uniquely characterized by a **parameter vector $\boldsymbol{\theta}$**
- The parameter vector changes the behavior of our program


pixel stacking & preprocessing
20
3
-1
…
234
$\boldsymbol{x}$
$\boldsymbol{\theta}^{2}$
h
hypothesis
0.9
0.1
cat
dog
$p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})$

# Machine Learning
## The Ideal Setting

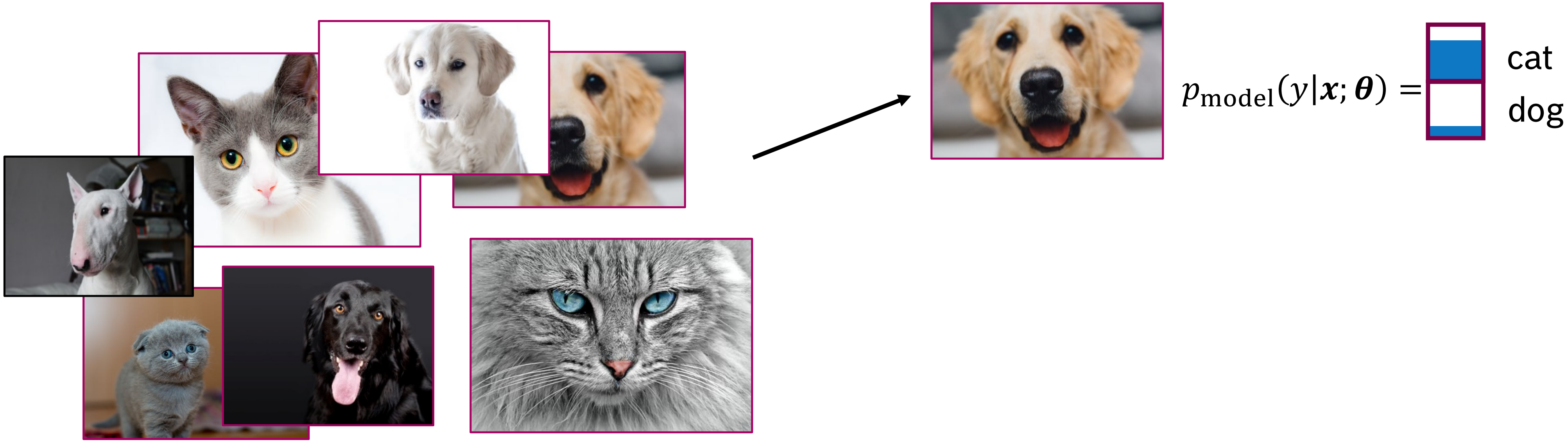
- Choose the hypothesis $\boldsymbol{\theta}^*$ that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\boldsymbol{\theta}^{\,*} = \operatorname{argmax}_{\boldsymbol{\theta}}\, \mathbb{E}_{x,y\,\sim p_{\text{data}}}[\log p_{\text{model}}\,(y|\boldsymbol{x};\boldsymbol{\theta})]$$

# Machine Learning
## The Ideal Setting

- Choose the hypothesis $\boldsymbol{\theta}^*$ that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\boldsymbol{\theta}^{\,*} = \mathrm{argmax}_{\boldsymbol{\theta}}\, \mathbb{E}_{x,y\,\sim p_{\mathrm{data}}}[\log p_{\mathrm{model}}\,(y|\boldsymbol{x};\boldsymbol{\theta})]$$

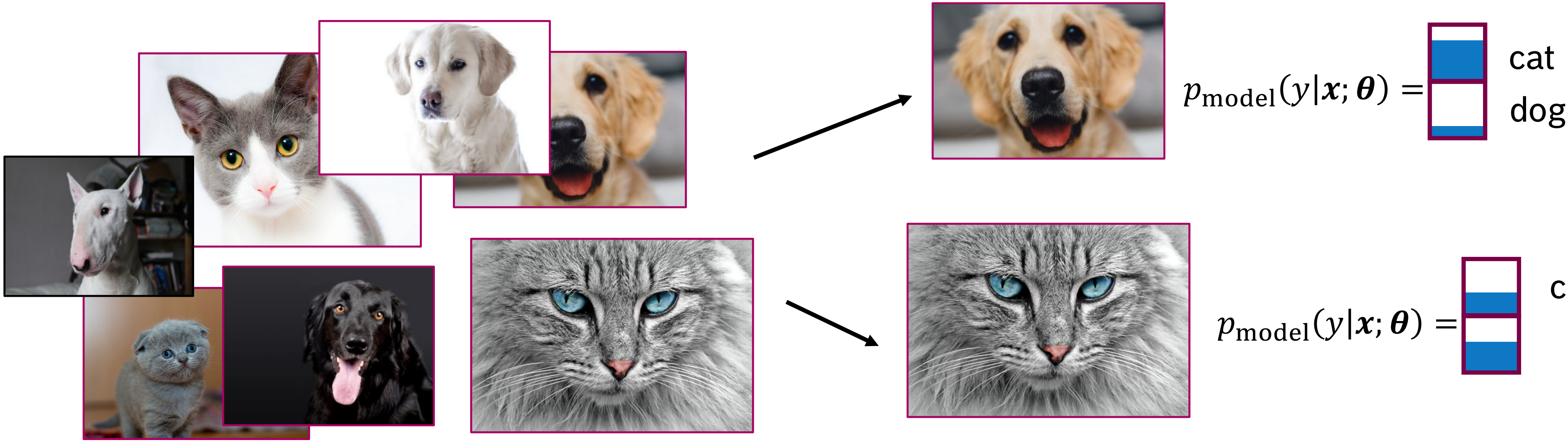


$p_{\mathrm{model}}(y|\boldsymbol{x};\boldsymbol{\theta}) =$

# Machine Learning
## The Ideal Setting

- Choose the hypothesis $\boldsymbol{\theta}^*$ that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y \sim p_{\text{data}}}[\log p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})]$$



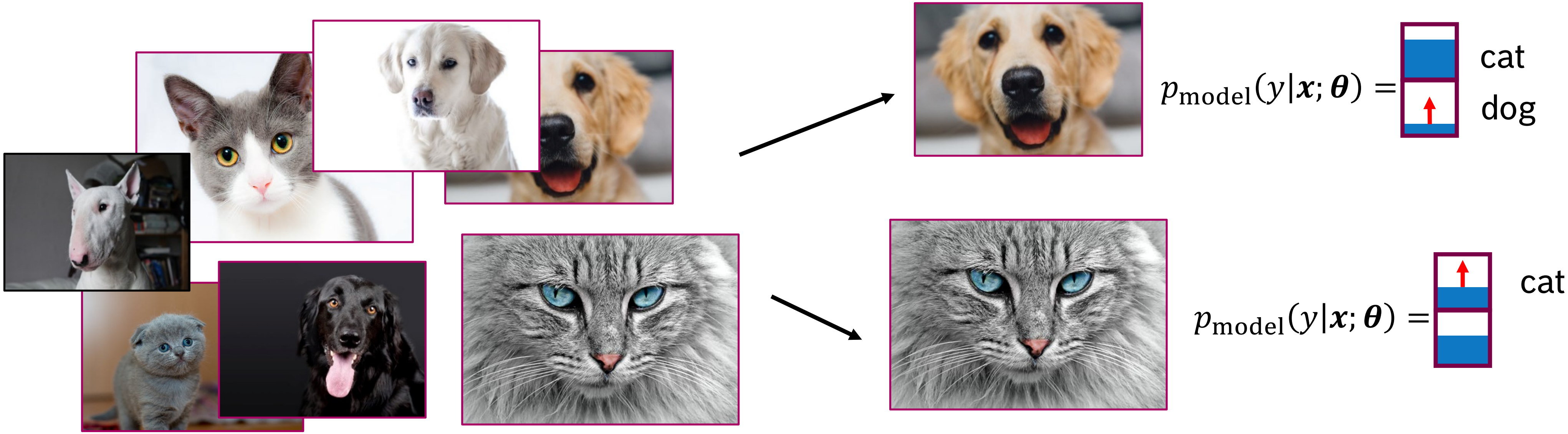
$p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta}) =$

$p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta}) =$

# Machine Learning
## The Ideal Setting

- Choose the hypothesis $\boldsymbol{\theta}^*$ that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\boldsymbol{\theta}^{\,*} = \text{argmax}_{\boldsymbol{\theta}}\, \mathbb{E}_{x,y\,\sim p_{\text{data}}}[\log p_{\text{model}}\,(y|\boldsymbol{x};\boldsymbol{\theta})]$$



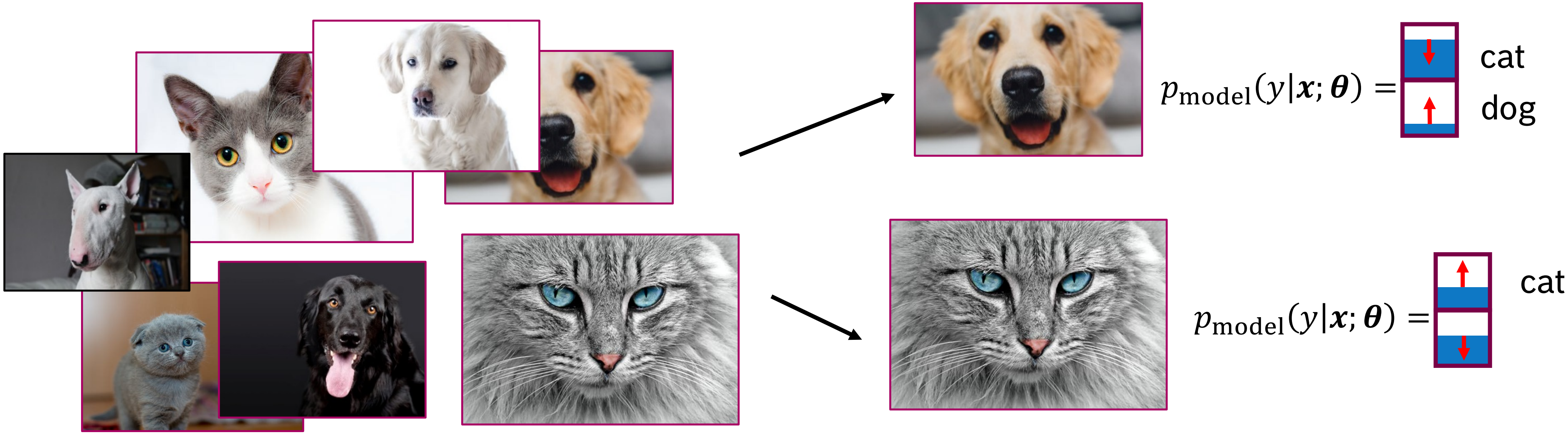
$p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta}) =$

$p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta}) =$

# Machine Learning
## The Ideal Setting

- Choose the hypothesis $\boldsymbol{\theta}^*$ that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}} (y|\boldsymbol{x};\boldsymbol{\theta})]$$



$p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta}) =$

$p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta}) =$

# Machine Learning
## Why Maximize the Log Likelihood?

- Numerical reasons: will come back to these later!
- Information theoretic reasons: same as minimizing the KL divergence between the predicted and ground truth posteriors

$$\boldsymbol{\theta}^{*} = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y \sim p_{\text{data}}(x,y)}[\log p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})]$$

$$\Updownarrow$$

$$\boldsymbol{\theta}^{*} = \operatorname{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p_{\text{data}}(x)}\left[D_{\text{KL}}\left(p_{\text{data}}(y|\boldsymbol{x};\boldsymbol{\theta}) \,||\, p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})\right)\right]$$

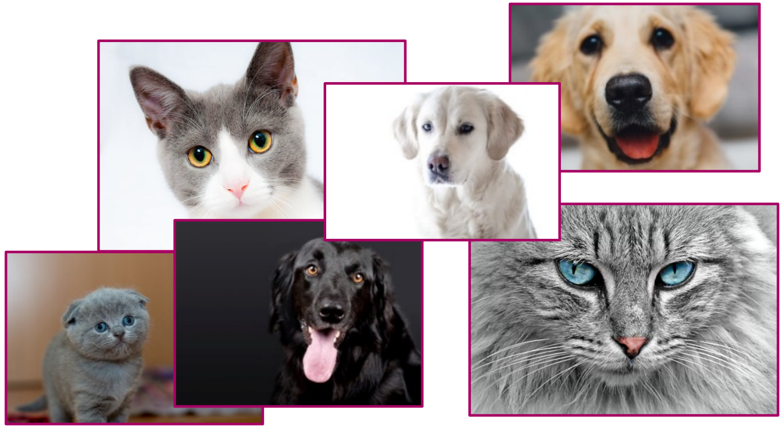
**Prove the equivalence as homework!**

# Machine Learning
## Expectation Meets Reality

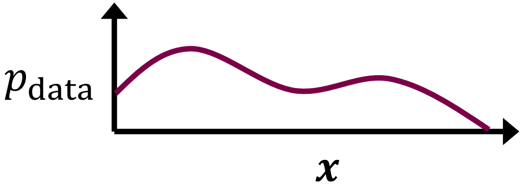
- But we do not have the true distribution!
- We only have a small **training set** drawn from the true distribution

$$\boldsymbol{x}_i, y_i \sim p_{\text{data}}(\boldsymbol{x}, y),\ i = \overline{1, n}$$

- The training set can be used to define the **empirical data distribution**:



$p_{\text{data}}(x, y)$



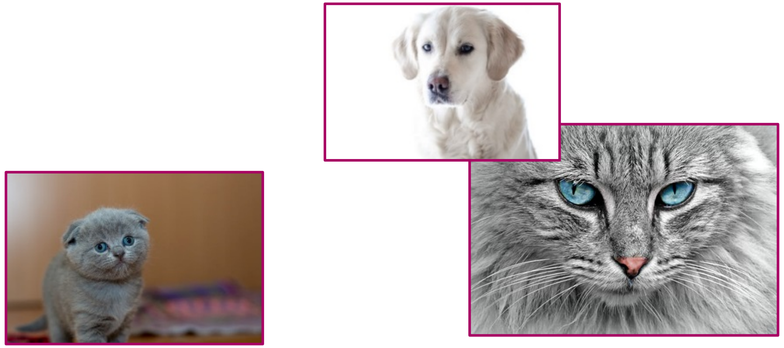
# Machine Learning
## Expectation Meets Reality

- But we do not have the true distribution!
- We only have a small **training set** drawn from the true distribution

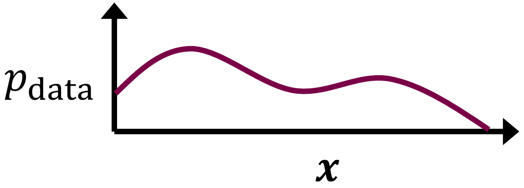
$$\boldsymbol{x}_i, y_i \sim p_{\text{data}}(\boldsymbol{x}, y),\ i = \overline{1, n}$$

- The training set can be used to define the **empirical data distribution**:

$$\hat{p}_{\text{data}}(\boldsymbol{x}, y) = \sum_{i=1}^{n} [\![ \boldsymbol{x} = \boldsymbol{x}_i \ \wedge y = y_i ]\!]$$



$\hat{p}_{\text{data}}(x, y)$



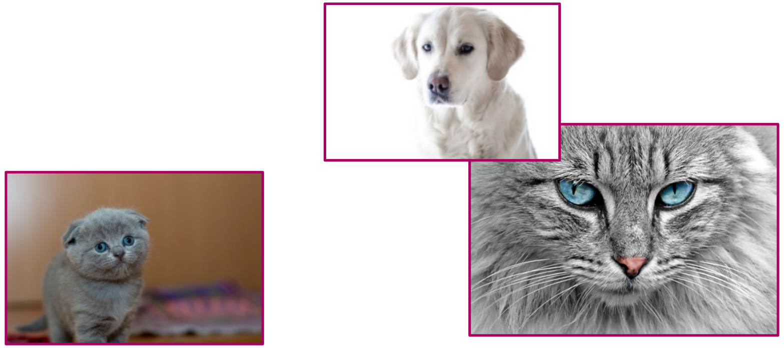
# Machine Learning
## Expectation Meets Reality

- But we do not have the true distribution!
- We only have a small **training set** drawn from the true distribution

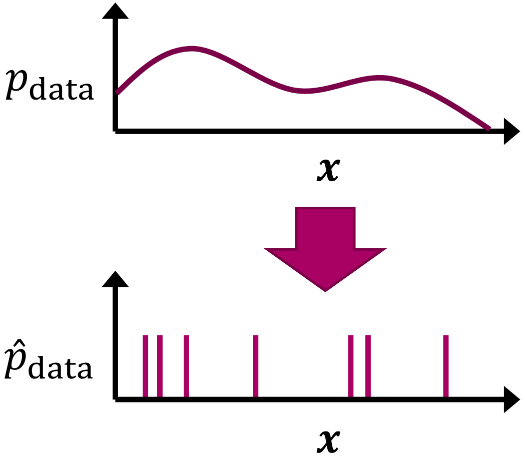
$$\boldsymbol{x}_i, y_i \sim p_{\text{data}}(\boldsymbol{x}, y),\ i = \overline{1, n}$$

- The training set can be used to define the **empirical data distribution**:

$$\hat{p}_{\text{data}}(\boldsymbol{x}, y) = \sum_{i=1}^{n} [\![\boldsymbol{x} = \boldsymbol{x}_i \ \wedge y = y_i]\!]$$



$\hat{p}_{\text{data}}(x, y)$



# Machine Learning
## The Compromise

- Solution: maximize the likelihood of the empirical data distribution
- Why would this work?

$p_{\text{data}}(x, y)$

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y\sim p_{\text{data}}}[\log p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})]$$

# Machine Learning
## The Compromise

- Solution: maximize the likelihood of the empirical data distribution
- Why would this work?

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y \sim p_{\text{data}}}[\log p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})]$$

$\hat{p}_{\text{data}}(x,y)$

# Machine Learning
## The Compromise

- Solution: maximize the likelihood of the empirical data distribution
- Why would this work?

$\hat{p}_{\text{data}}(x, y)$

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y\sim p_{\text{data}}}[\log p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})]$$

$$\boldsymbol{\theta}^{\text{ML}} = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{1}{n}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \log p_{\text{model}}(y_i|\boldsymbol{x}_i;\boldsymbol{\theta})$$

# Machine Learning
## Justification: The IID Assumptions

- Look at the way we built our empirical distribution:

$$x_i, y_i \sim p_{\text{data}}(x, y),\ i = \overline{1, n}$$

- Assumption 1:
  - Data samples are drawn **independently** of each other
  - Why? Hypothesis program always processes inputs from scratch.
- Assumption 2:
  - Training samples were **identically distributed**, i.e. drawn from the true distribution
  - Why? No learning unless training set is representative of the true world
- Are these assumptions enough?

# Machine Learning
## Justification: Maximum Likelihood Consistency

- An estimator is **consistent** if it converges given enough training data ($n \to \infty$)

- ML is **consistent** if:
  1. The hypothesis set contains the true model. For classifiers:

$$\exists \boldsymbol{\theta}, \forall \boldsymbol{x}, \forall y, p_{\text{data}}(y|\boldsymbol{x}) = \log p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})$$

  2. Each hypothesis is uniquely identified by $\boldsymbol{\theta}$

- In practice we never have enough data!

# Machine Learning

## The Loss Function

- Can also see ML as minimizing the **negative log likelihood** (NLL):

$$\boldsymbol{\theta}^{\mathrm{ML}} = \operatorname{argmin}_{\boldsymbol{\theta}} - \tfrac{1}{m} \textstyle\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \log p_{\mathrm{model}} \left(y_i | \boldsymbol{x}_i; \boldsymbol{\theta}\right)$$

- In general, we seek to minimize a **loss function** over the training set:

$$\boldsymbol{\theta}^{*} = \operatorname{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

- NLL is a special case of loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{m} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \log p_{\mathrm{model}} \left(y_i | \boldsymbol{x}_i; \boldsymbol{\theta}\right)$$

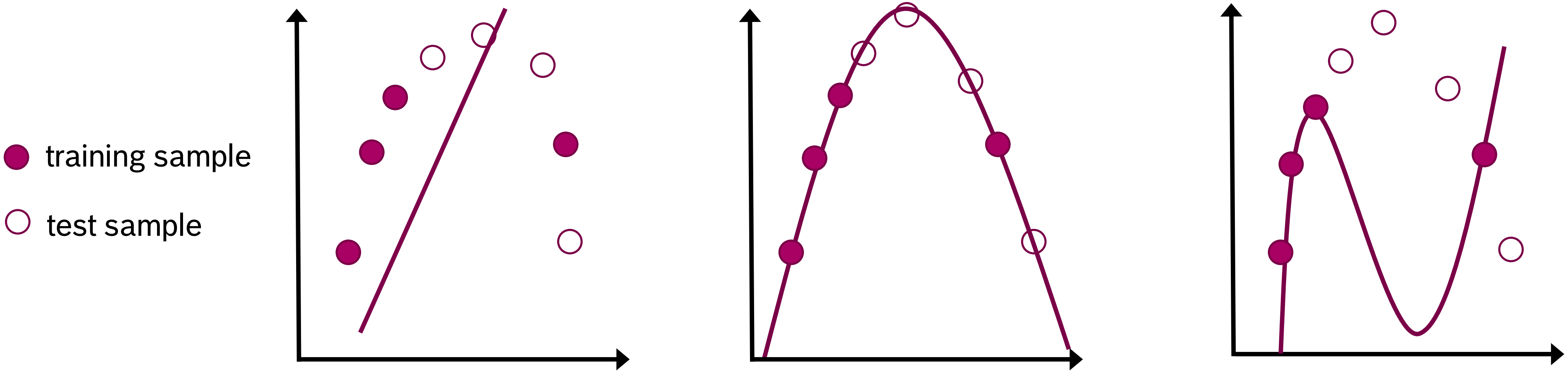
- The training loss is also called the **empirical risk**

# Machine Learning
## The Dangers of Failed Assumptions: Underfitting and Overfitting

- **Underfitting**: cannot find a model that fits the training data

  ⇔high loss on training data
- **Overfitting**: cannot find a model that generalizes on test (unseen) data

  ⇔ low training loss, high test loss



underfitting

appropriate capacity

overfitting

# Machine Learning
## Model Capacity

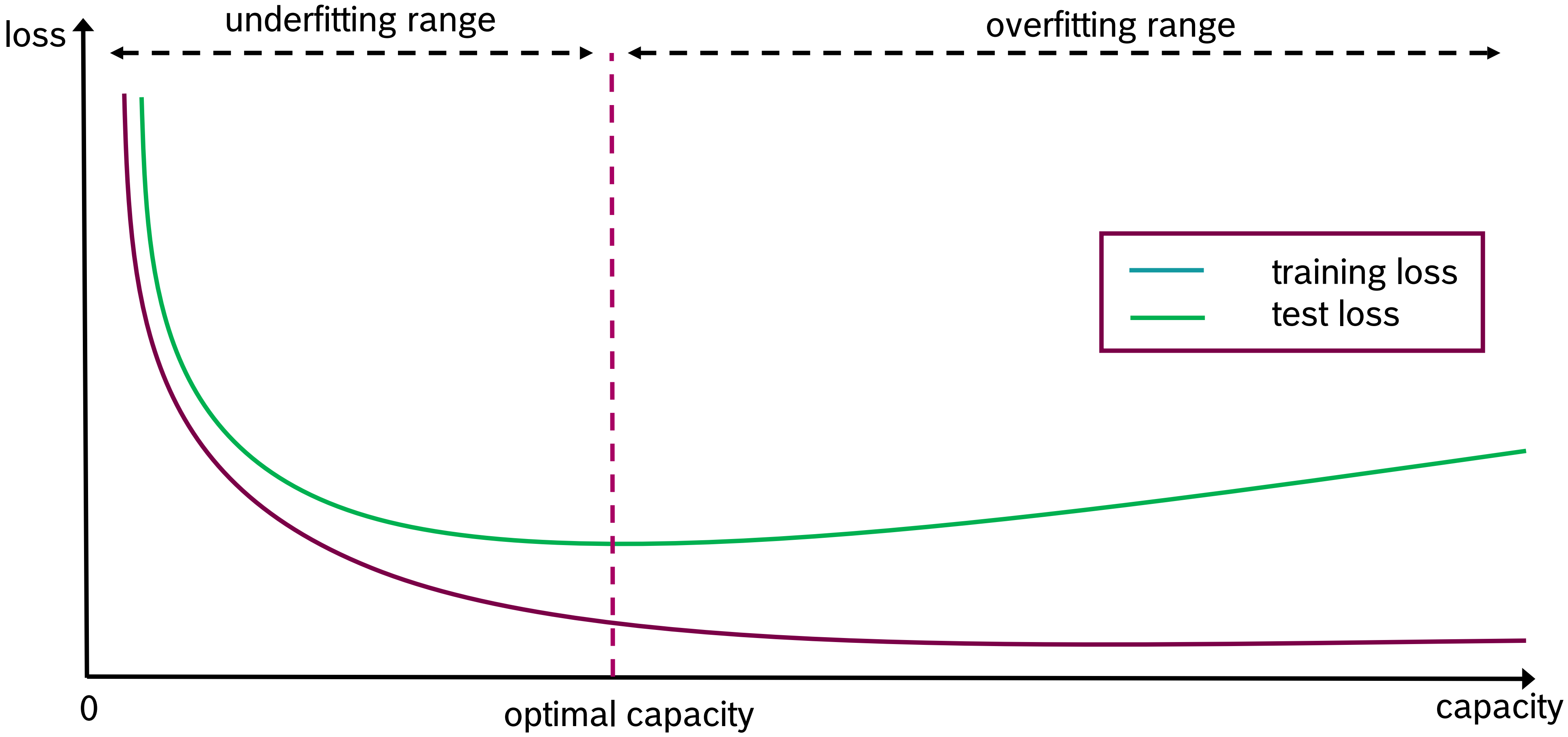
- **(Effective) capacity** = ability of the model to fit a wide variety of functions
- **Representational capacity** = ability of the hypothesis set to fit a wide variety of functions
- Due to imperfect optimizers:

$$\boxed{\text{effective capacity} \leq \text{representational capacity}}$$

- Underfitting ⇔effective capacity too low
- Overfitting ⇔effective capacity too high
- Formal definitions of capacity exist: Vapnik-Chervonenkis Dimension, Rademacher Complexity

# Machine Learning

## Capacity-Loss Relationship Curve



# Machine Learning
## Regularization (Capacity Control)

- Need to control model capacity:
  - via the optimization process (do not look for the optimal solution)
  - optimize a different objective that also looks at the parameters
  - restrict hypothesis set
- All of these methods introduce an inductive bias:
  - Favor some hypotheses over others
  - Ockham’s Razor principle: prefer simple explanations of the data
- There is no universally applicable bias:

**The “no free lunch” theorem:** ***Averaged over all data distributions, any two machine learning algorithms have the same accuracy (Wolpert, 1996)***

# Machine Learning
## Definition

- Machine learning can do much more than classification!

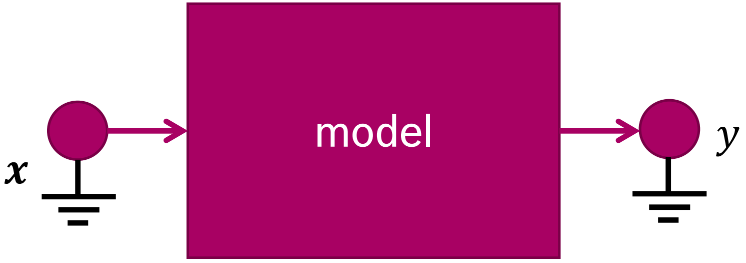
> ***“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E.” (Mitchell, 1997)***

# Machine Learning
## The Task (T)

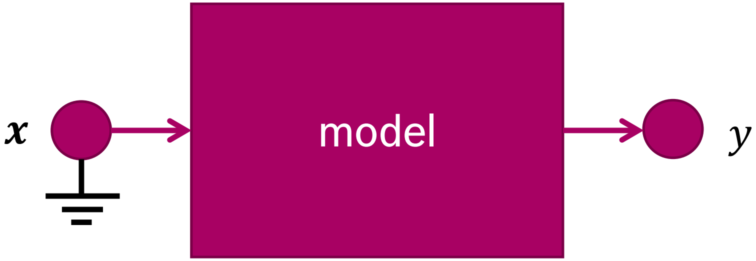
| Task | Input | Output | Example Problem |
|---|---|---|---|
| classification | feature vector | discrete category | object recognition |
| regression | feature vector | real-valued quantity | image quality assessment |
| transcription | feature vector | sequence of symbols | transcribe text from image |
| translation | sequence of symbols | sequence of symbols | translate from English to German |
| synthesis | sequence of symbols<br>feature vector | real-valued signal | speech synthesis<br>style transfer |
| denoising | real-valued signal | real-valued signal | image restoration |

- The list above is far from exhaustive!

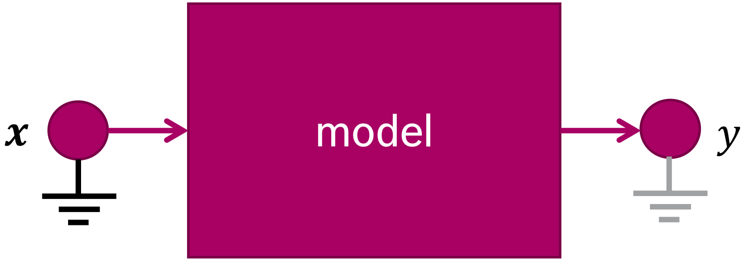
# Machine Learning
## The Experience (E)



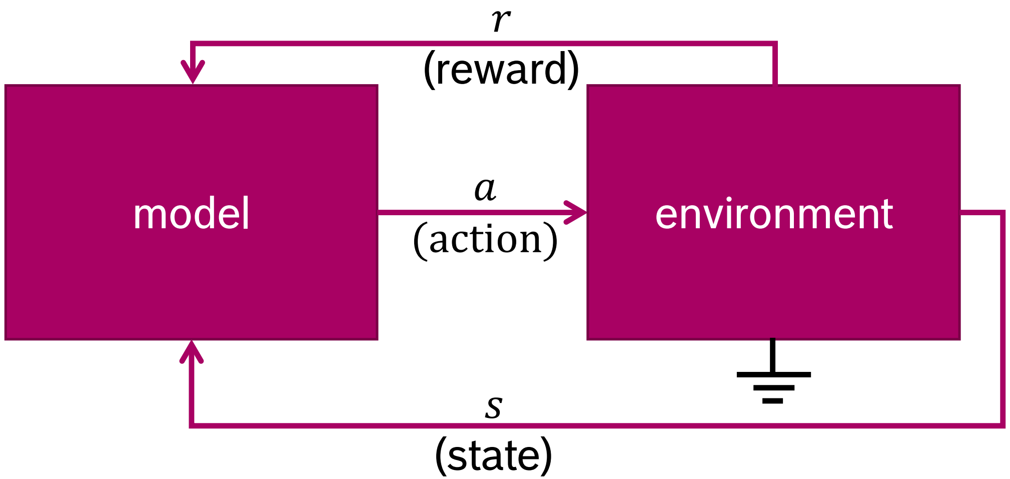
**supervised learning**



**unsupervised learning**



**semi supervised learning**
(some labels may be missing)



**reinforcement learning**

# Machine Learning
## The Performance Measure (P)

- Very much task dependent

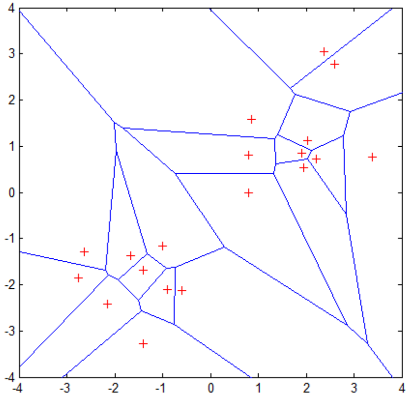
- For non-interactive methods: loss measured over the test set
  - Negative log likelihood (cross entropy)
  - 0-1 loss
  - Euclidean Loss
  - Hinge Loss
  - etc.

- For reinforcement learning: expected reward

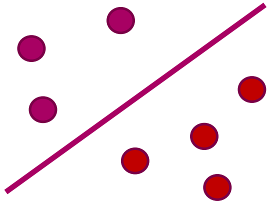
- Many losses correspond to probabilistic interpretations of model outputs (e.g. Euclidean Loss ⇔ Gaussian Output Distributions)

# Machine Learning
## Model Examples

- Shallow models
  - Non-parametric:
    - k-Nearest Neighbor (kNN) (Fixed & Hodges, 1951)
    - Kernel Density Estimation (Parzen, 1956; Rosenblatt,1962)
  - Parametric:
    - Linear Regression (Legendre, 1805)
    - Logistic Regression (David Cox, 1958)
    - ADALINE (Widrow & Hoff, 1960)
    - Support Vector Machines (SVM) (Vapnik & Chervonenkis, 1963)
    - Decision Trees (Morgan & Sonquist, 1963)
- Deep Models
  - Multilayer Perceptron (MLP) (Ivaknenko & Lapa, 1965)
  - Neocognitron (Fukushima, 1980)
  - Lenet-5 (Lecun et al., 1998)
  - AlexNet (Krizhevsky et al., 2012)
  - VGG (Simonyan and Zisserman, 2014)
  - ResNet (He et al., 2015)
  - DETR (Carion et al. 2020)



kNN decision surface



SVM decision surface

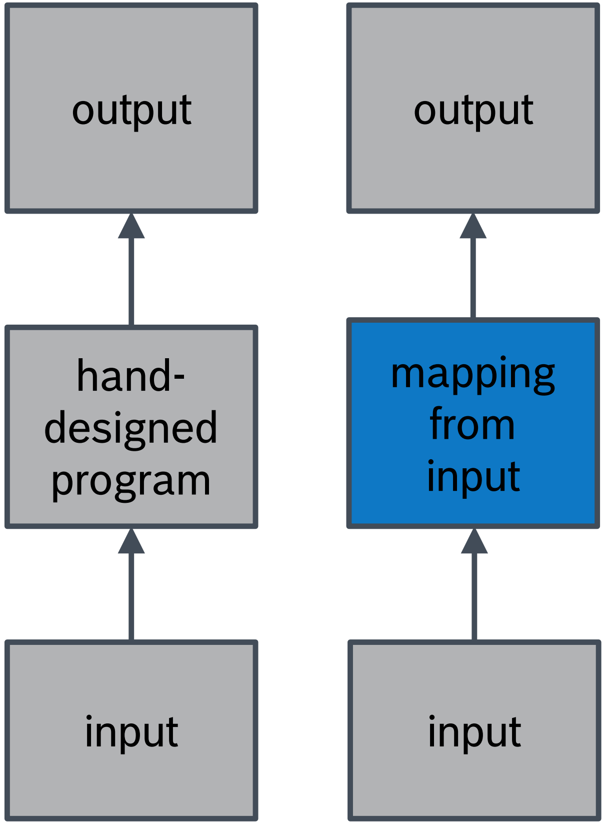
# Machine Learning
## Recap

- Reformulated the inference problem as a machine learning problem
- Select the best hypothesis such that:
  - Fits the training data (seen)
  - Generalizes to the real world data (unseen)
- These are conflicting goals!
- The true challenges:
  - Controlling model capacity (underfitting, overfitting)
  - Introducing the right inductive bias
  - Finding the hypothesis that fits the data (optimization problem)

# THE PATH TO DEEP LEARNING

# The Path to Deep Learning
## Approaches to AI

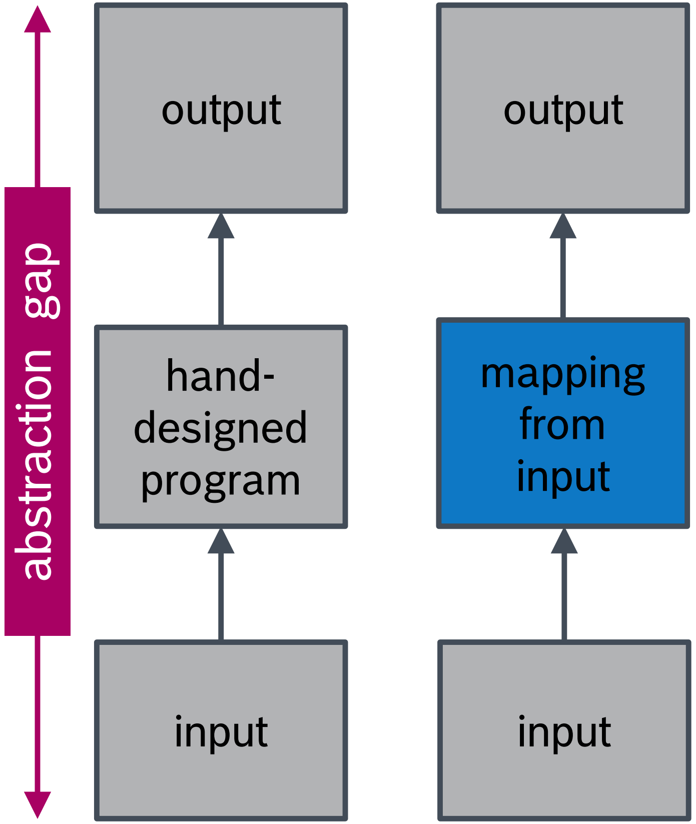
- Rule Based Approaches
  - Hard-code informal knowledge in a formal language
  - Not very successful ☹
- Machine Learning Approaches
  - Acquire knowledge automatically from real-world data
  - Most promising to date ☺
  - But wait! What kind of knowledge? Can we learn a mapping from input to output?
  - We need to cover a large abstraction gap!



BOSCH

# The Path to Deep Learning
## Approaches to AI

- Rule Based Approaches
  - Hard-code informal knowledge in a formal language
  - Not very successful ☹
- Machine Learning Approaches
  - Acquire knowledge automatically from real-world data
  - Most promising to date ☺
  - But wait! What kind of knowledge? Can we learn a mapping from input to output?
  - We need to cover a large abstraction gap!



# The Path To Deep Learning

## Closing the Abstraction Gap is Difficult!

- Images are the complex result of multiple interacting **factors of variation**
- Need to separate what is relevant from what is not!

illumination

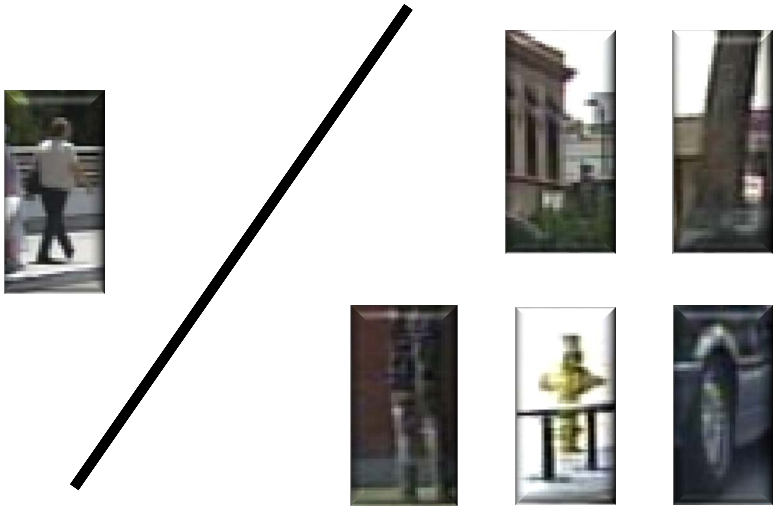
deformation

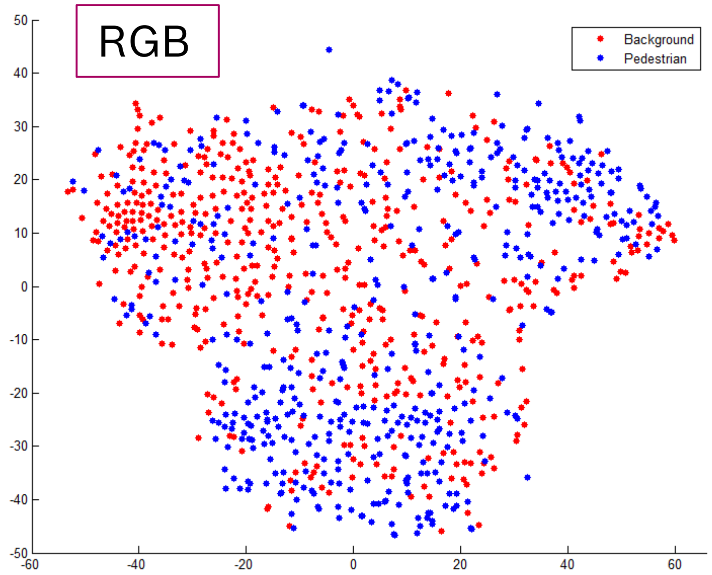
occlusion

*Images taken from Fei-Fei, Karpathy & Johnson, Lecture Notes, 2016*

# The Path to Deep Learning
## The Representation Problem

- We need **features**: relevant information extracted from the input
- How do we come up with good features?

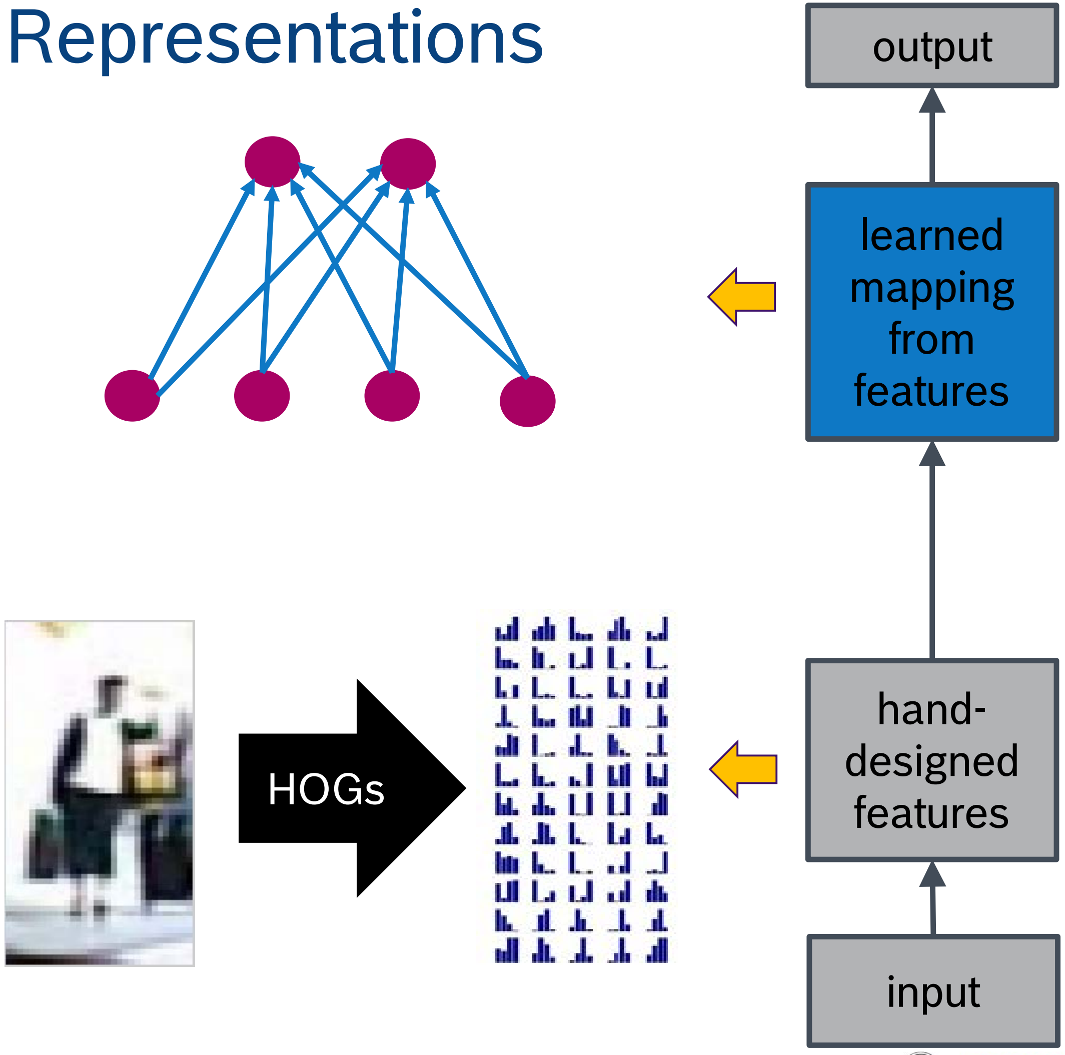




# The Path to Deep Learning

## First Generation: Hand-Designed Representations

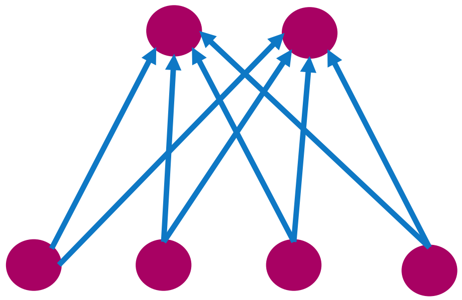
- Write a program to extract features!
- Examples:
  - **Edges (Canny)**
  - **Histograms of Gradients (HoGs)**
  - **Haar-like features**
  - **Aggregated Channel Features (ACF)**
- Pros:
  - better than rule-based systems
  - can inject human bias
- Cons:
  - still worse than human performance
  - large feature-output abstraction gap
  - hard to design good features
  - does not scale

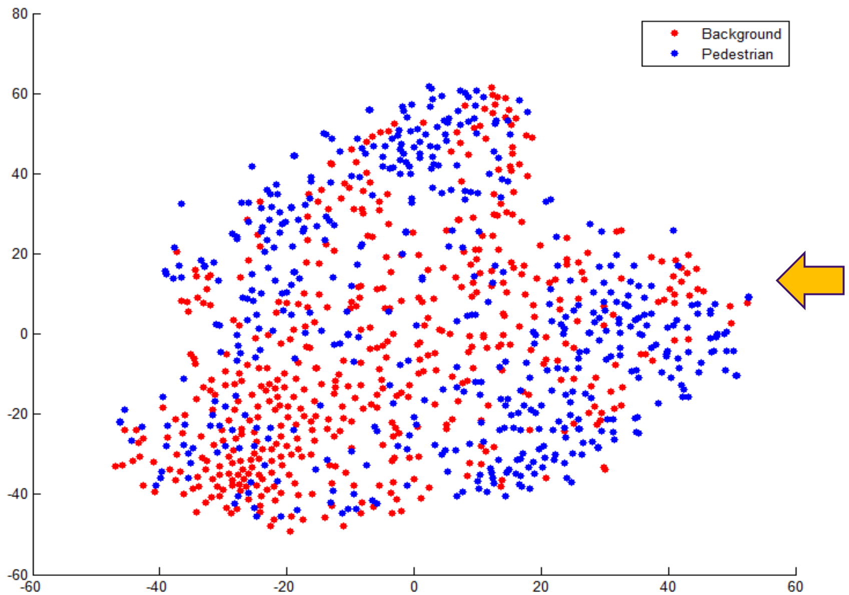


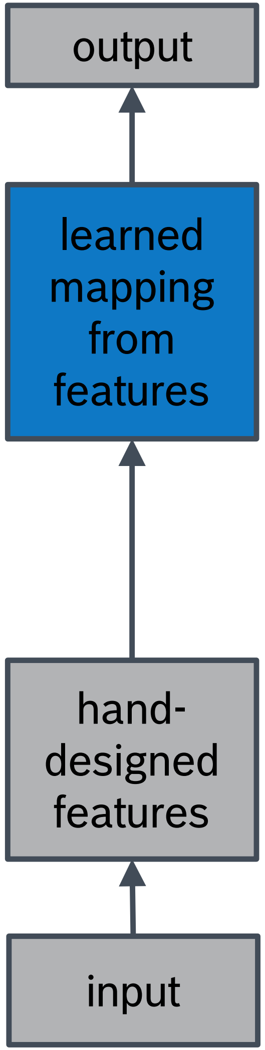
# The Path to Deep Learning

## First Generation: Hand-Designed Representations

- Write a program to extract features!
- Examples:
  - **Edges (Canny)**
  - **Histograms of Gradients (HoGs)**
  - **Haar-like features**
  - **Aggregated Channel Features (ACF)**
- Pros:
  - better than rule-based systems
  - can inject human bias
- Cons:
  - still worse than human performance
  - large feature-output abstraction gap
  - hard to design good features
  - does not scale



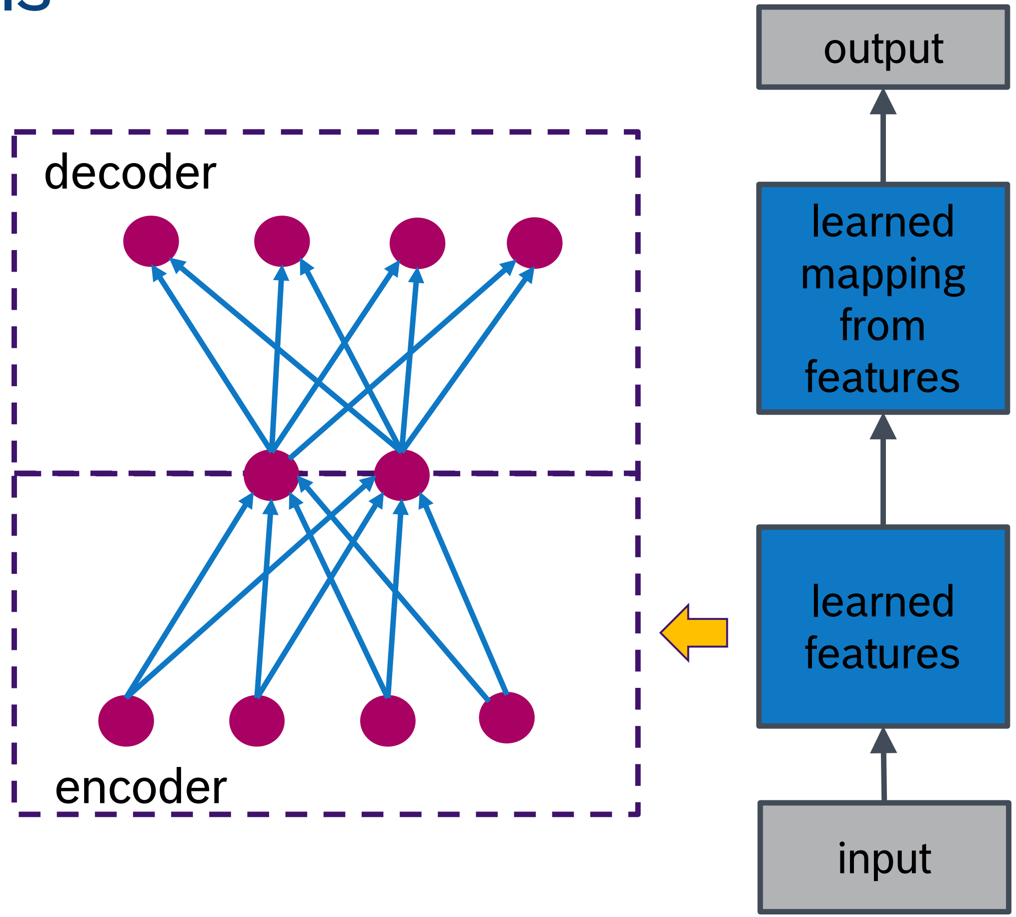




# The Path to Deep Learning

## Second Generation: Shallow Models

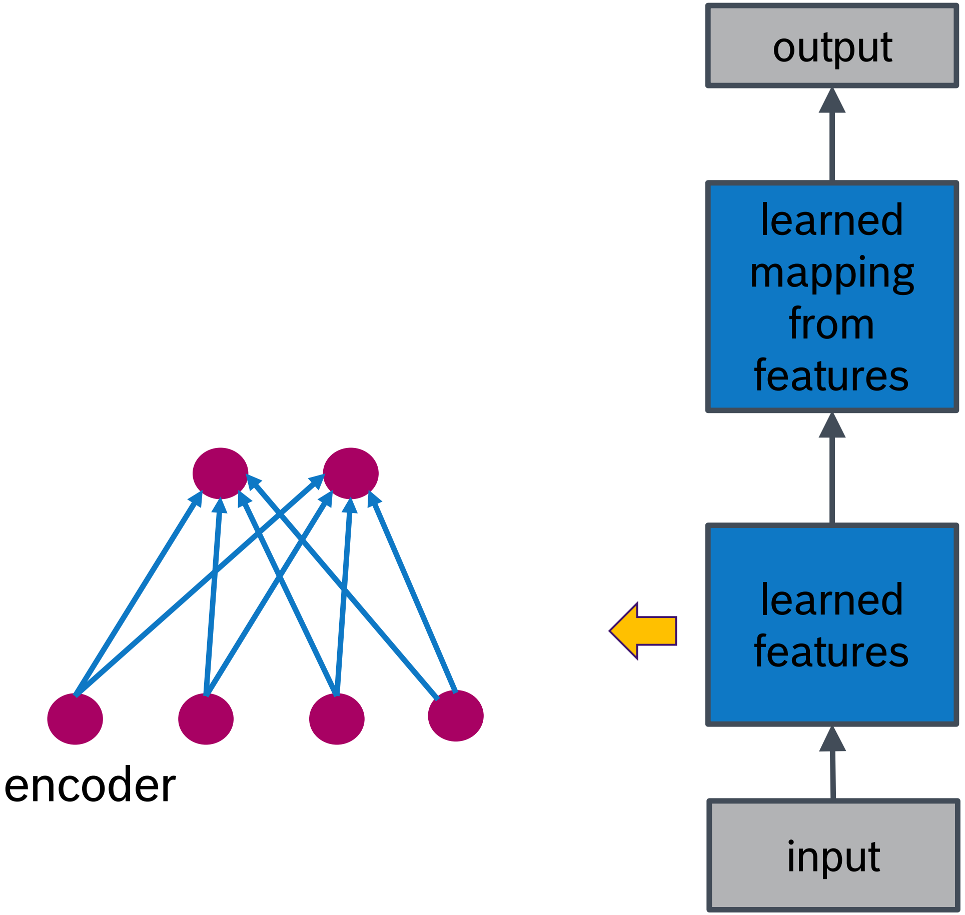
- Learn the representation too!
    - Separately: *e.g.* **auto-encoders**
    - Jointly with the mapping: **end-to-end learning**
- Better, but we still need to cover a large **abstraction gap**



# The Path to Deep Learning

## Second Generation: Shallow Models

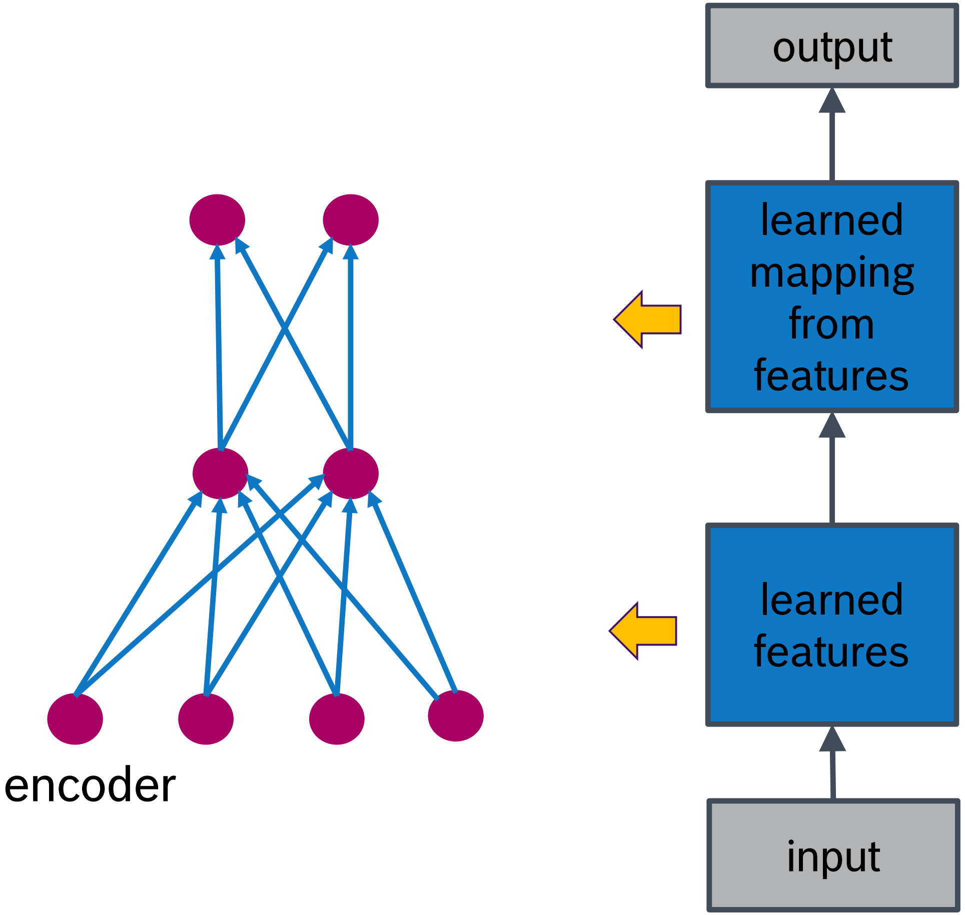
- Learn the representation too!
  - Separately: *e.g.* **auto-encoders**
  - Jointly with the mapping: **end-to-end learning**
- Better, but we still need to cover a large **abstraction gap**



# The Path to Deep Learning

## Second Generation: Shallow Models

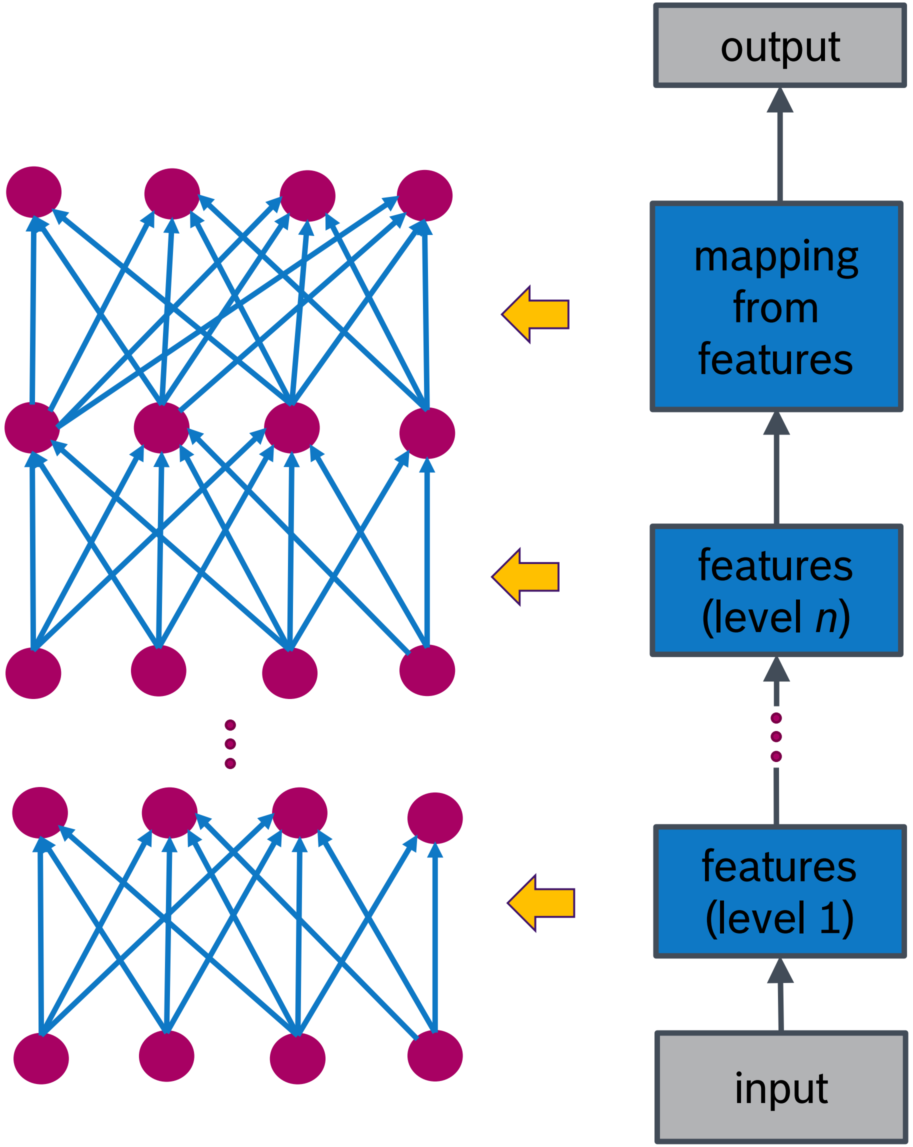
- Learn the representation too!
  - Separately: *e.g.* **auto-encoders**
  - Jointly with the mapping: **end-to-end learning**
- Better, but we still need to cover a large **abstraction gap**



# The Path to Deep Learning

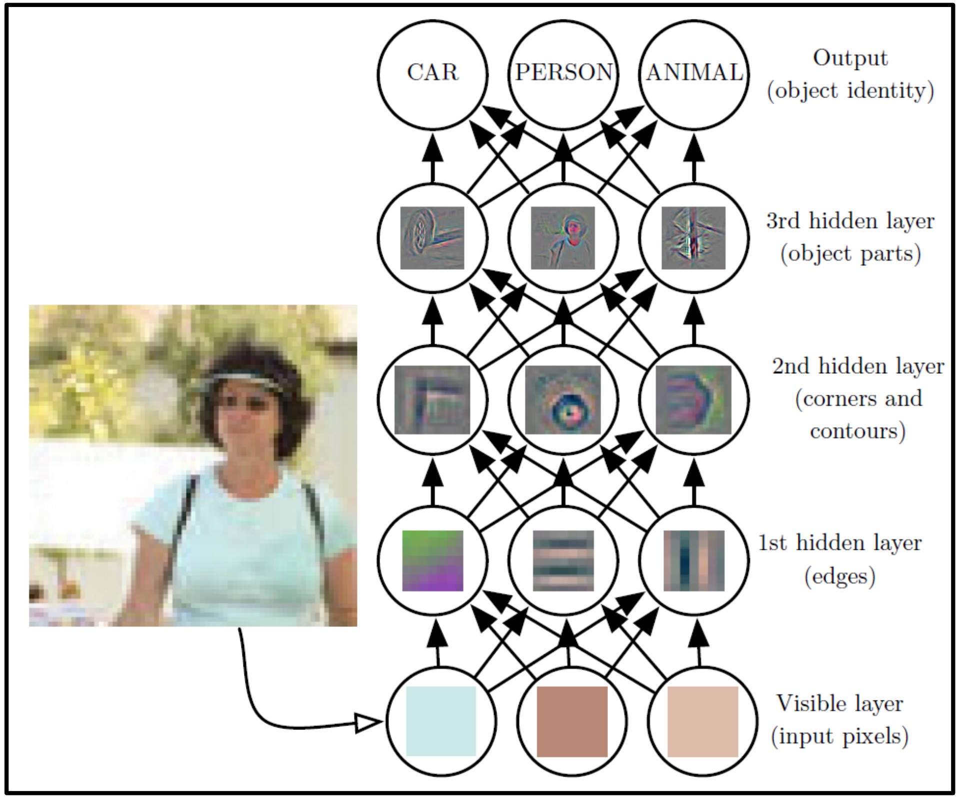
## Third Generation: Deep Learning

- Why not learn a hierarchy of features?
- This is **deep learning**
- Examples:
  - **deep auto-encoders**
  - **multi-layer-perceptron (MLP)**
  - **convolutional neural networks (CNNs)**



# The Path to Deep Learning
## Third Generation: Deep Learning

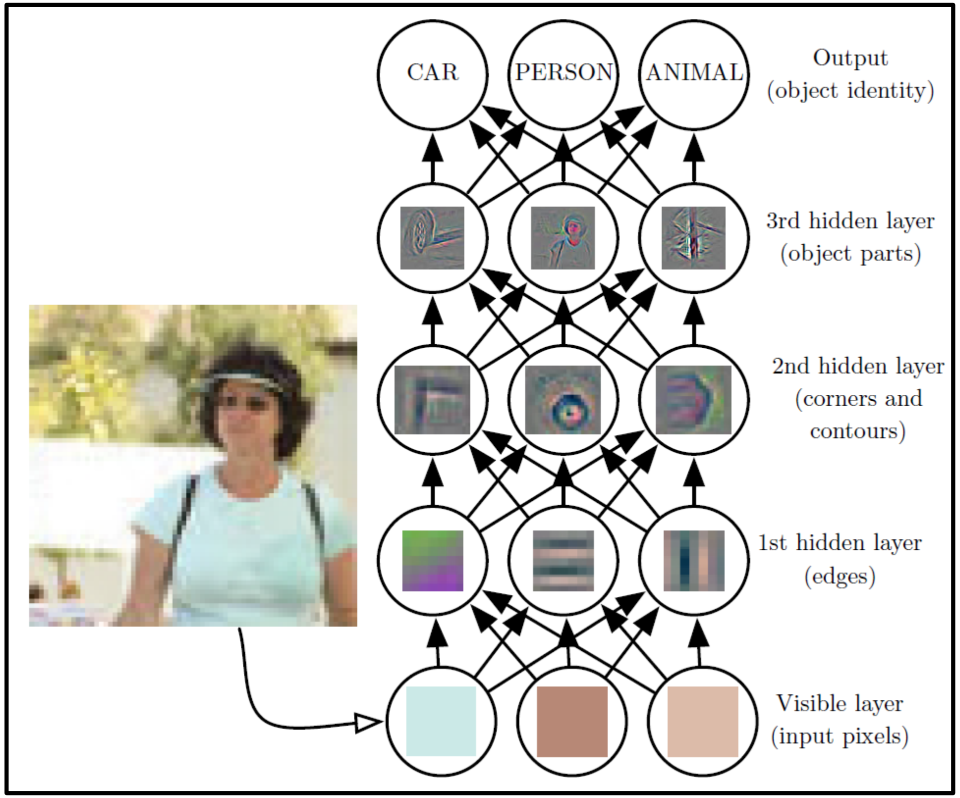
- Pros:
  - Near human performance on many hard problems
  - Fully scalable to any problem domain
- Cons:
  - Need large amounts of data
  - Expensive to train and run


CAR
PERSON
ANIMAL
Output (object identity)
3rd hidden layer (object parts)
2nd hidden layer (corners and contours)
1st hidden layer (edges)
Visible layer (input pixels)

# The Path To Deep Learning

## A Dual View of Deep Learning

- **Representational:** function composition
  - layer: a simple function (multi-dimensional)
  - layer output: a new representation of the input
- **Computational:** computer program
  - layer: set of parallel instructions
  - layer output: state of computer memory (may not be entirely input-related!)

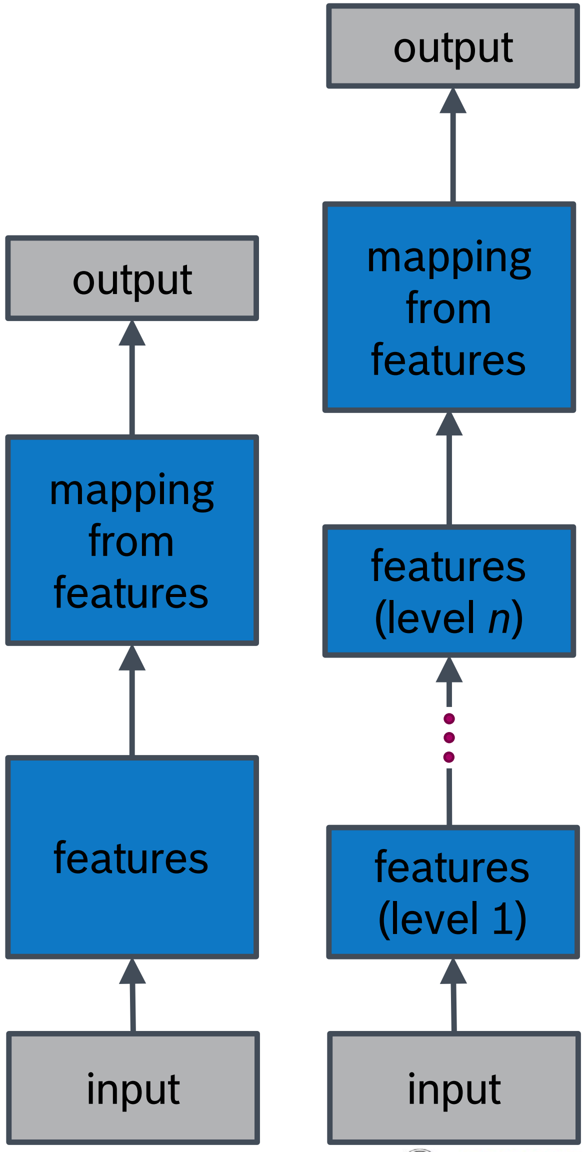


# The Path To Deep Learning

## Deep or not deep?

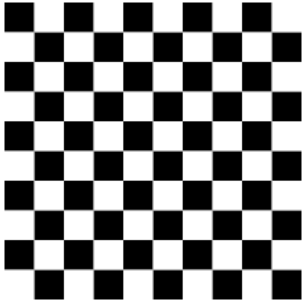
- No single definition of depth:
  - Maximum number of instructions to evaluate the entire output?
  - Maximum number of concepts to arrive at the output representation?
- No clear definition of deep:
  - Having at least 4 layers? (some authors)
  - In any case:

    *A **deep model** is a model with more learned instructions/concepts than a traditional machine learning model.*

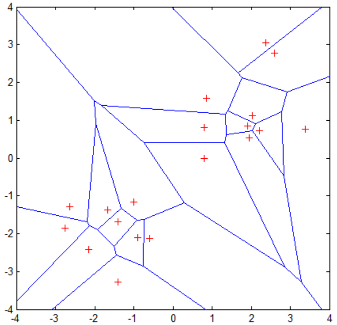


# The Path to Deep Learning
## Distributed Representations

- We need to **dis-entangle** factors of variation
- Example: recognize cars/trucks/birds of three colors (red, green, blue)
- Encodings:
  - Option 1: all combinations => 9 features
  - Option 2: **distributed representation** => 6 features (3 for object type, 3 for color)
- Deep models can learn distributed representations!



Entangled features



kNN in image space needs one example in each cell!

# The Path To Deep Learning
## Recap

- We need to address the **abstraction gap**
- Extremely Difficult:
  - Representations are not conspicuous!
  - Hard to dis-entangle factors of variation
- Hand designed features
  - Time consuming to design
  - Do not generalize across problems
  - Brittle to variations (illuminations, pose changes)
- Learned feature hierarchies (**end-to-end learning**)
  - Trade training data for design time
  - Generalize across problems
  - Increased variance
  - Distributed representations of the world
- **Deep learning** is just end-to-end learning with a lot of feature levels

# REFERENCES

# Deep Learning for Computer Vision
## References

- [1] Goodfellow, I. and Bengio, Y. and Courville, A., "*Deep Learning*", MIT Press, 2016, http://www.deeplearningbook.org/
- [2] Bishop, C. M. "Pattern Recognition and Machine Learning", Springer, 2006
- [3] Murphy, K.P. "*Machine Learning: a Probabilistic Perspective*", MIT Press, 2012
- [4] Fei-Fei, Karpathy and Johnson, *Lecture Notes, 2016,* http://cs231n.stanford.edu/slides/2017/
- [5] Zhang, A. and Lipton, Z.C. and Li, M. and Smola, J.A., "*Dive into Deep Learning*", *2017,* https://d2l.ai/