Deep Learning

Intelligence and Learning: From Nature to Machine

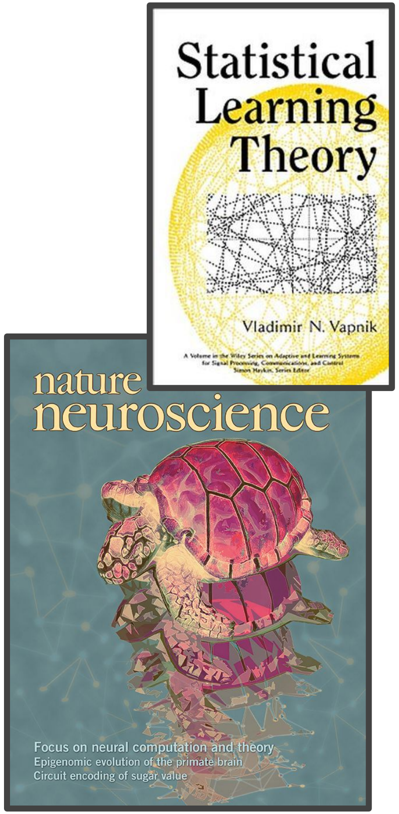
Dr Chris Willcocks

Department of Computer Science

Durham University

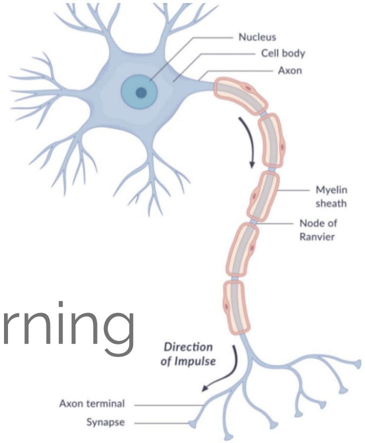
Course Aims

- To be able to approach complex ill-defined problems that require deep layers of learning.
- To understand learning in nature, and its statistics, and differential geometry.
- To ask the right scientific questions given a new task, and use modern deep learning libraries to effectively design, train & test.


Statistical Learning Theory
Vladimir N. Vapnik
A Volume in the Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications, and Control
Simon Haykin, Series Editor
nature neuroscience
Focus on neural computation and theory
Epigenomic evolution of the primate brain
Circuit encoding of sugar value

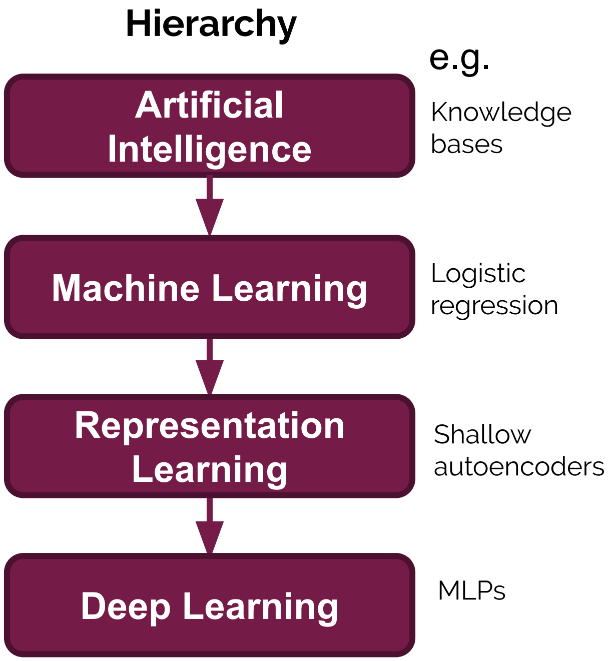
Today

- What is intelligence, machine learning and deep learning?
- How learning works in nature (neuropsychology)
- How the brain works, how neurons work
- Neuroplasticity and Hebbian theory
- Basics of artificial neurons
- Fundamental (high-level) principles of machine learning
  - Introduction to “tensors”
  - Introduction to loss functions



Artificial Intelligence

- **Intelligence** is generally ill-defined:
  - “The ability to **acquire** and apply **knowledge** and **skills**”
  - “The **capacity** for **logic**, **understanding**, **self-awareness**, **learning**, **emotional knowledge**, **reasoning**, **planning**, **creativity**, and **problem solving**.”
  - “The **ability to learn** or **understand** or to deal with **new** or trying **situations**”
- **Artificial** Intelligence (or **Machine** Intelligence)
  - Intelligence demonstrated by machines (in contrast to **natural intelligence**)



Progressive training of GANs

Are these tasks intelligent?

The ability to succeed at goals?

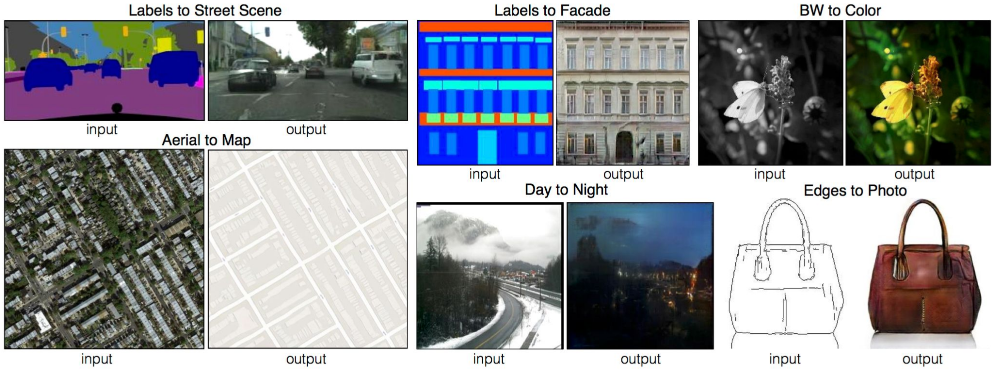
The ability to generalise?

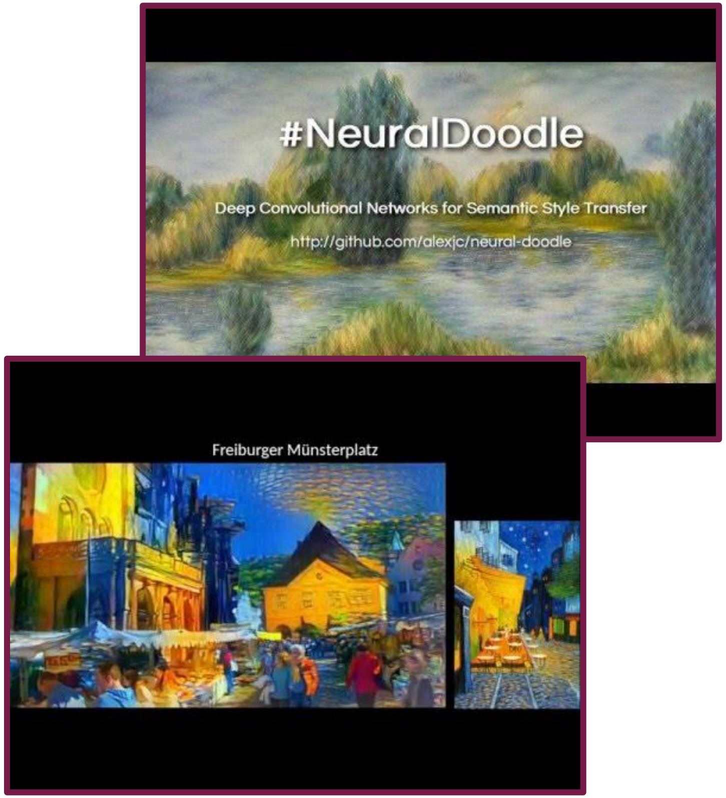
The ability to create?

The ability to infer to new tasks?

The ability to comprehend?

What is creativity?




#NeuralDoodle
Deep Convolutional Networks for Semantic Style Transfer
http://github.com/alexjc/neural-doodle
Freiburger Münsterplatz

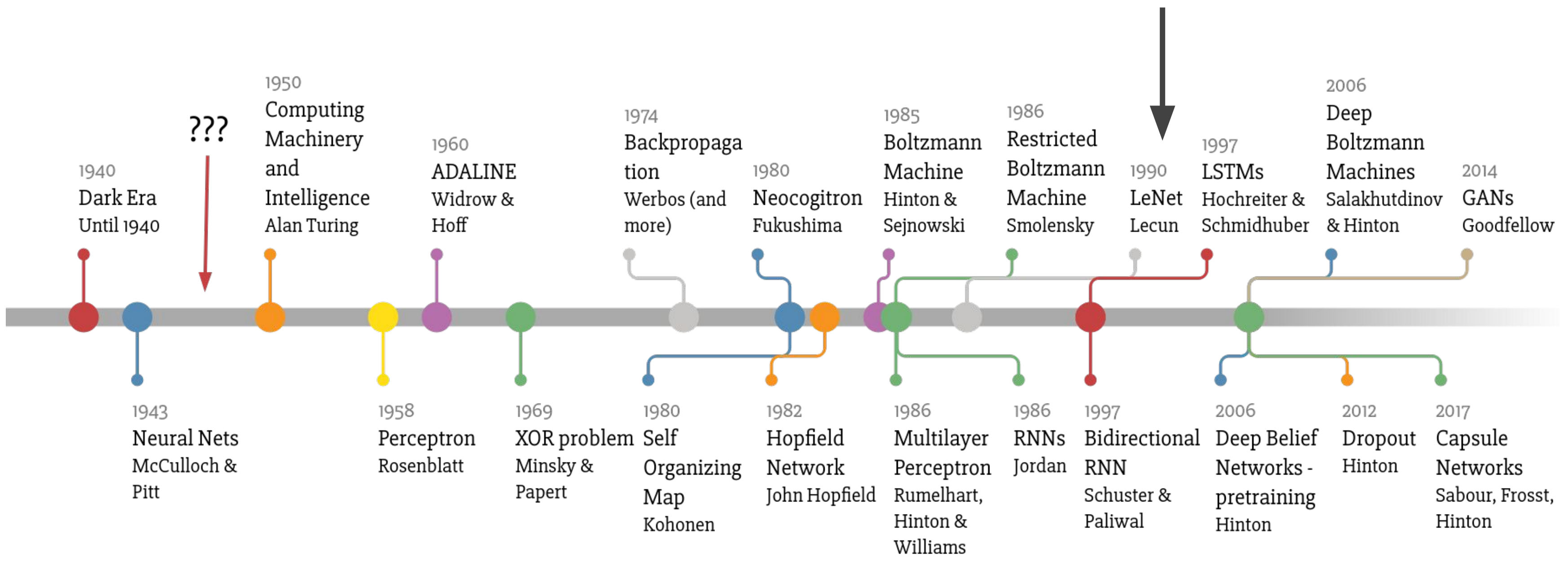
Style transfer in videos, neural doodle

The Waves of Artificial Intelligence

Durham University

The "recent" interest = GPU pricing, tooling, deep learning

*1998



???

1940
Dark Era
Until 1940

1943
Neural Nets
McCulloch &
Pitt

1950
Computing
Machinery
and
Intelligence
Alan Turing

1958
Perceptron
Rosenblatt

1960
ADALINE
Widrow &
Hoff

1969
XOR problem
Minsky &
Papert

1974
Backpropaga
tion
Werbos (and
more)

1980
Self
Organizing
Map
Kohonen

1980
Neocogitron
Fukushima

1982
Hopfield
Network
John Hopfield

1985
Boltzmann
Machine
Hinton &
Sejnowski

1986
Multilayer
Perceptron
Rumelhart,
Hinton &
Williams

1986
Restricted
Boltzmann
Machine
Smolensky

1986
RNNs
Jordan

1990
LeNet
Lecun

1997
Bidirectional
RNN
Schuster &
Paliwal

1997
LSTMs
Hochreiter &
Schmidhuber

2006
Deep Belief
Networks -
pretraining
Hinton

2006
Deep
Boltzmann
Machines
Salakhutdinov
& Hinton

2012
Dropout
Hinton

2014
GANs
Goodfellow

2017
Capsule
Networks
Sabour, Frosst,
Hinton

Made by Favio Vázquez

What is Learning?

Environment

Experiences	Actions	Changed Actions
Delicious	Put in mouth	Eat it again!
Cold	Get closer	Don’t touch.
Warm	Touch	
Burning		

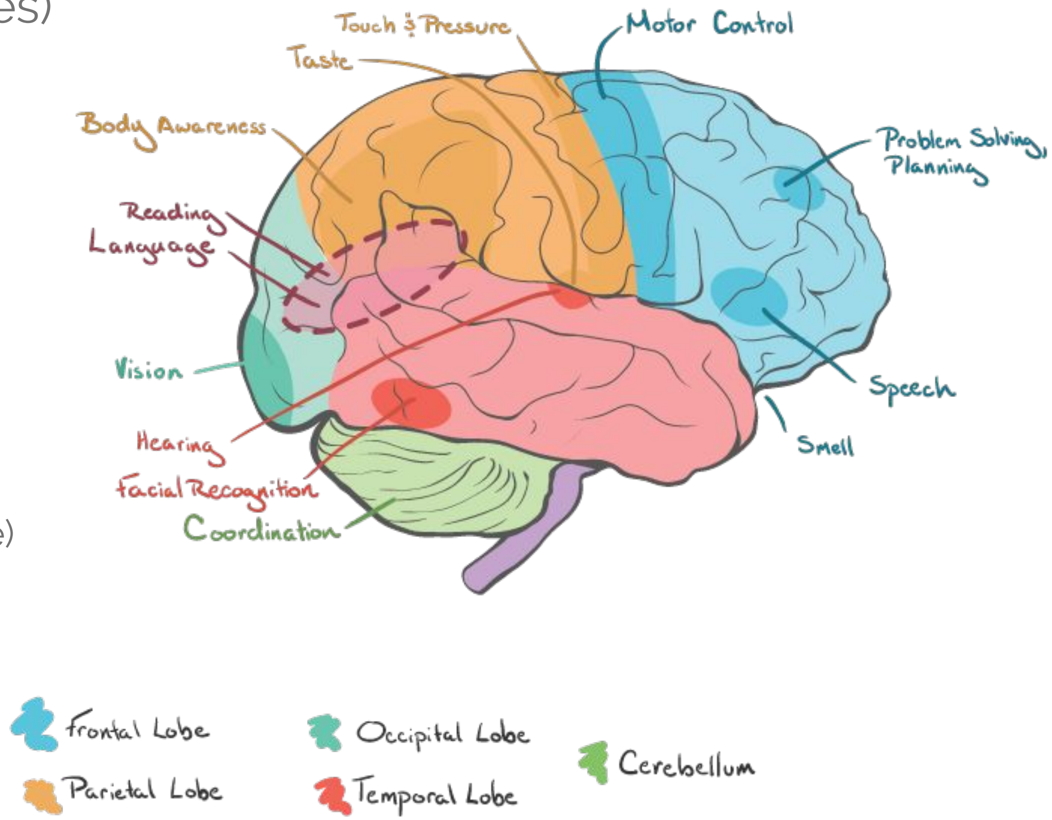
What is Learning?

“We define learning as the **transformative process** of taking in information that—when internalized and mixed with what we have **experience**d—changes **what we know** and builds on **what we do.** It’s based on input, process, and reflection. It is what changes us.”

–From The New Social Learning by Tony Bingham and Marcia Conner.

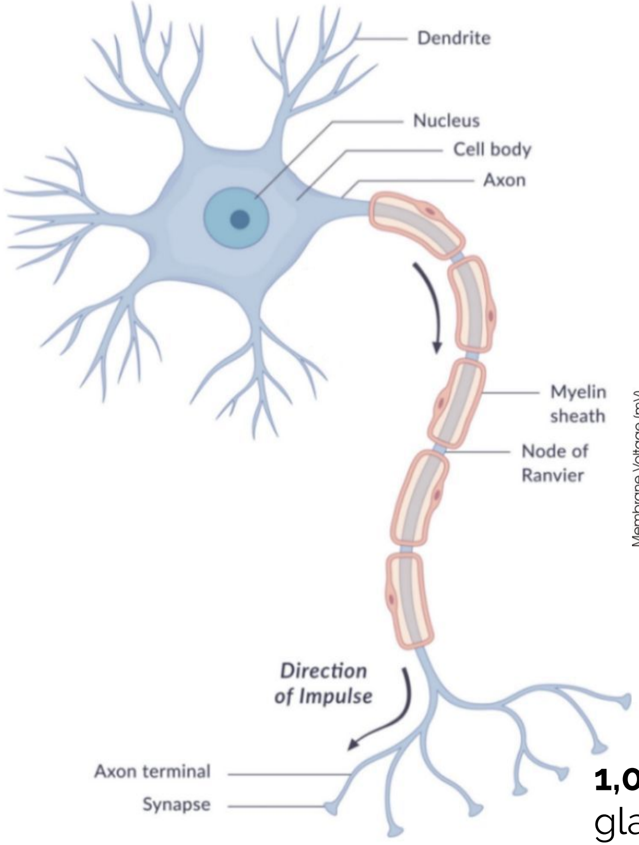
How does the Brain Work?

- Right side/left side (hemispheres)
  - Lobes
    - Frontal lobe
      - Executive functions
      - Memory
      - Learning
      - Planning
    - Parietal lobe
      - Sensation
      - Spatial awareness
    - Temporal lobe (banana shape)
      - Hearing
      - Language functions
    - Occipital lobe at back
      - Vision from front along optic nerves

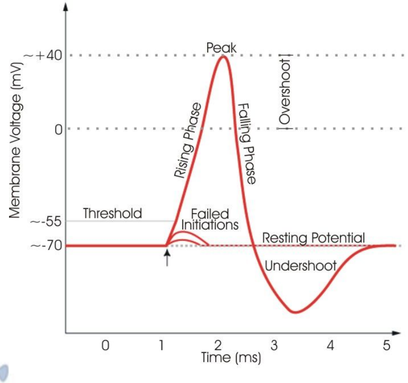


How does the Brain Work?

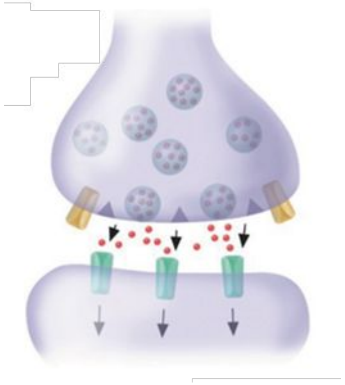
1,000's of inputs (other neurons, sensory neurons e.g. taste buds from a salt or sugar molecule...)



- Neurons send out branches called **dendrites**, and a large output called an **axon**
- The axon is coated in myelin that helps it conduct electrical impulses.
- The places where the nerve cells make their connections with each other is called a **synapse**.



Synapse:



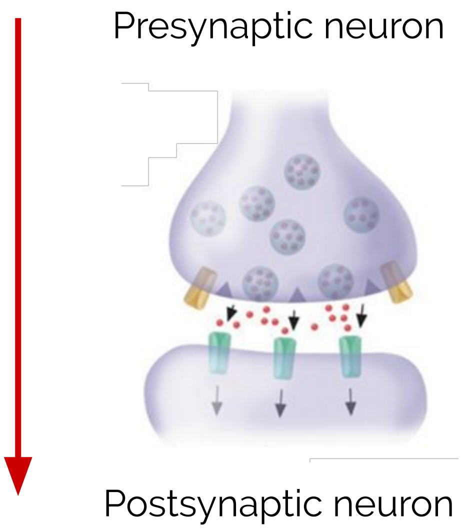
"Synapse" (from the greek meaning "to clasp together") **Signals** get **summed up**, and travel to the hillock (Axon neck)

- If large enough, triggers an action potential travels down axon

1,000's of targets (e.g. other neurons, muscle cells, gland cells, blood vessels to release hormones...)

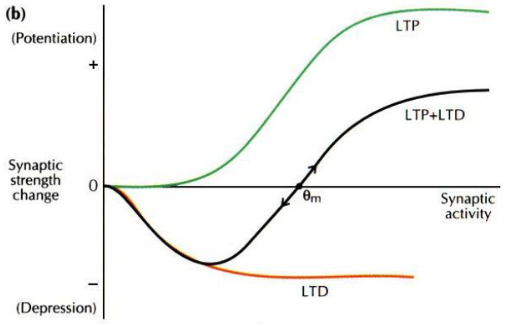
Figure by wetcake (left) and Andrej Kral (right)

Synaptic Plasticity



Presynaptic neuron

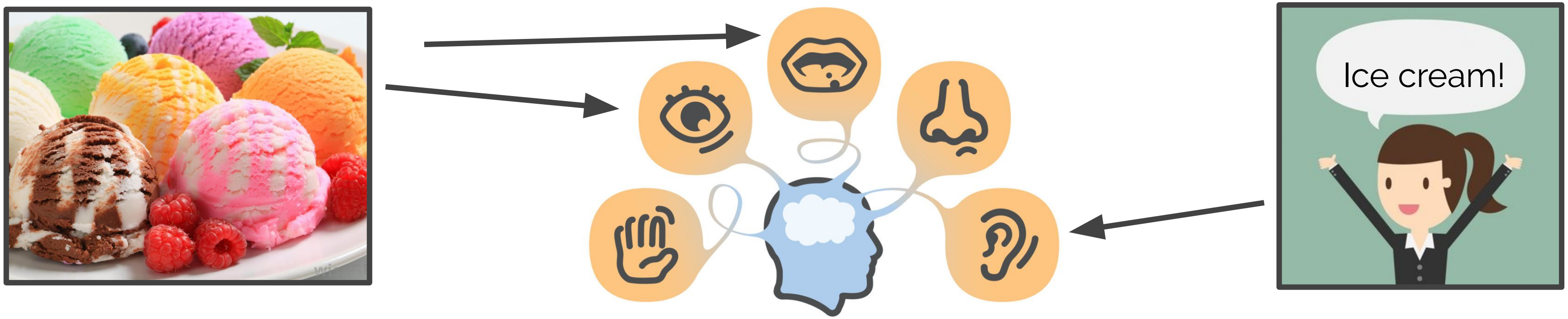
Postsynaptic neuron



- What's very cool is that with frequent repeated stimulation, the same level of presynaptic stimulation converts into **greater** postsynaptic potential.
    - In other words, as a neuron gets a lot of practice sending signals to a specific target neuron, it gets better at sending those signals (the synapse strength increases).
        - Increased strength that lasts for a long time (from minutes to many months) is called **Long Term Potentiation** (weakening is **Long Term Depression**).
    - As synapses are strengthened and retain strength, we're able to more easily recall previous experiences.

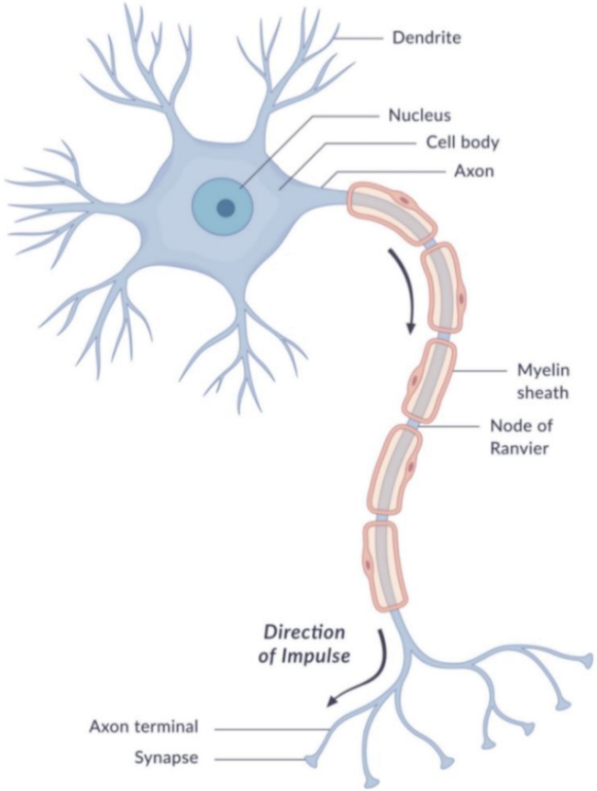
Figure from: "Synaptic Plasticity: A molecular mechanism for metaplasticity", Journal of Current Biology.

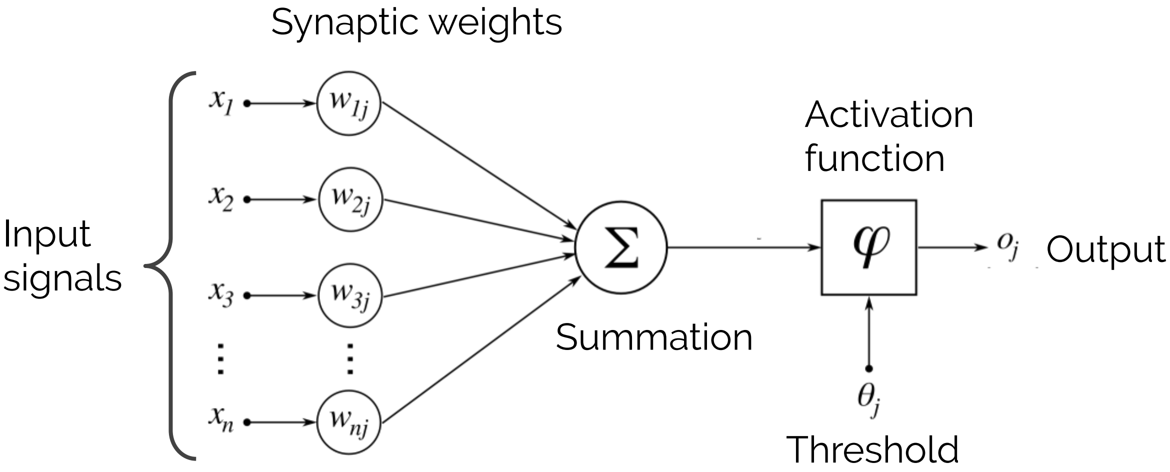
Hebbian Theory



- **Hebbian theory**
  - If two neurons fire at the same time, the connections between them are strengthened, and thus are more likely to fire again together in the future.
  - If two neurons fire in an uncoordinated manner, their connections are weakened and their more likely to act independently in the future.
- **Updated hebbian hypothesis based on recent findings**
  - If the presynaptic neuron fires within a window of 20ms before the postsynaptic neuron, the synapse will be strengthened.
  - However if the presynaptic neuron fires within a window of 20ms after the postsynaptic neuron, the synapse will be weakened.

Artificial Neurons




Synaptic weights

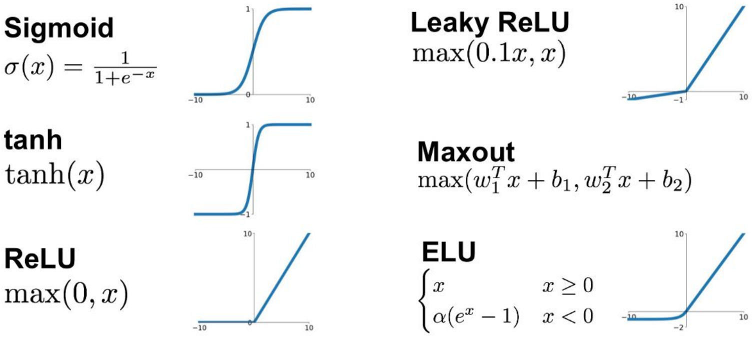
Input signals

Activation function

Summation

Output

Threshold


Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$

tanh
$\tanh(x)$

ReLU
$\max(0, x)$

Leaky ReLU
$\max(0.1x, x)$

Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

What is Machine Learning?

Durham University

...learning, but with a machine! (ok *throw in some stats, calculus, geometry, ...*)

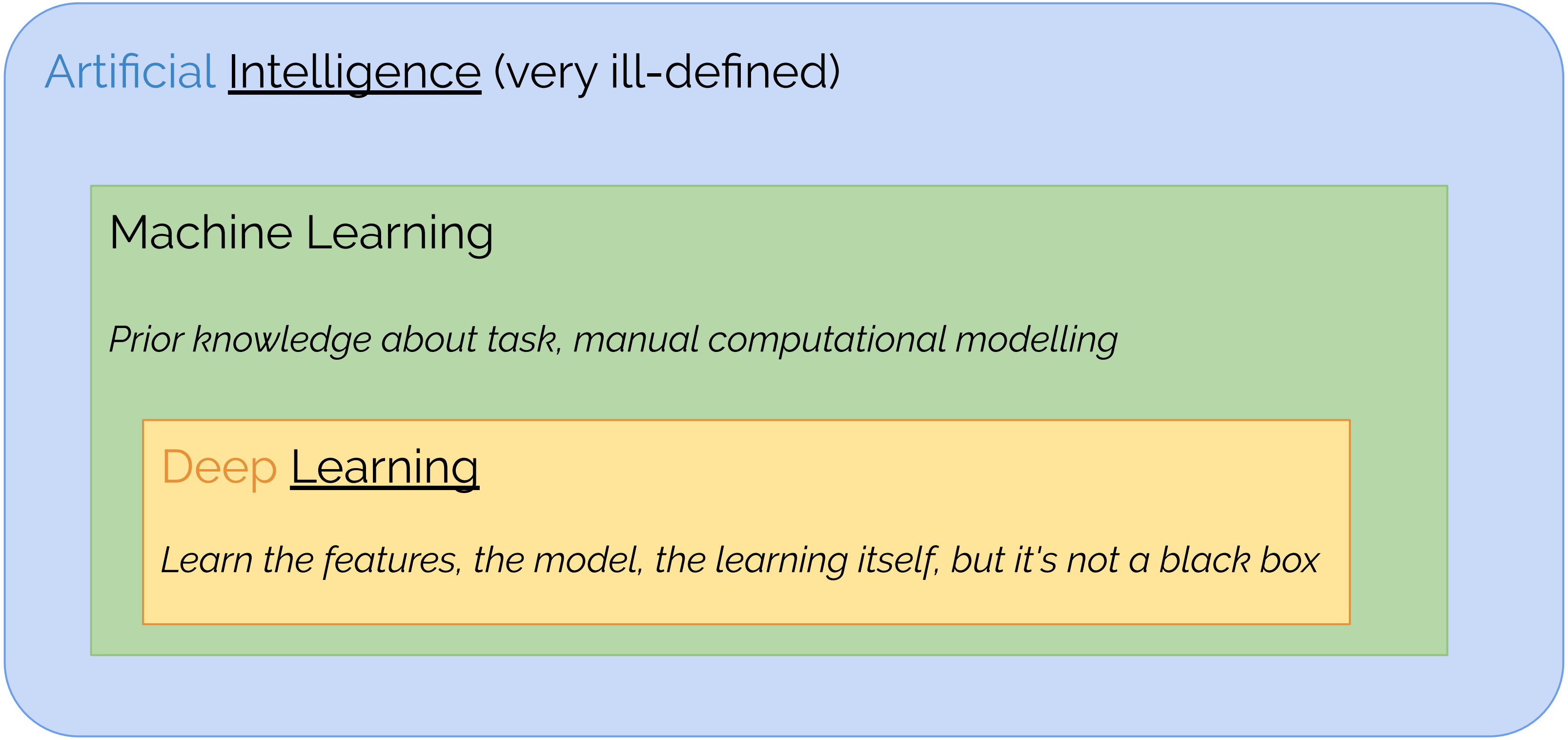
"A computer program is said to learn from experience $\boldsymbol{E}$ with respect to some class of tasks $\boldsymbol{T}$ and performance measure $\boldsymbol{P}$, if its performance at tasks in $\boldsymbol{T}$ as measured by $\boldsymbol{P}$, improves with experience $\boldsymbol{E}$"

Mitchell, 1997

So we have 3 components:

1. The Task $\boldsymbol{T}$
2. The Experience $\boldsymbol{E}$
3. The Performance Measure $\boldsymbol{P}$

What is Deep Learning?



Artificial Intelligence (very ill-defined)

Machine Learning

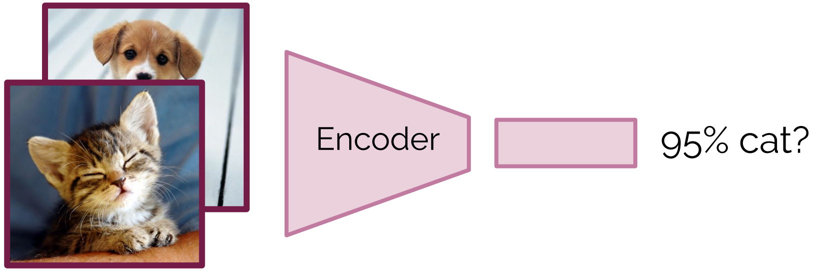
Prior knowledge about task, manual computational modelling

Deep Learning

Learn the features, the model, the learning itself, but it's not a black box

Tasks *T*

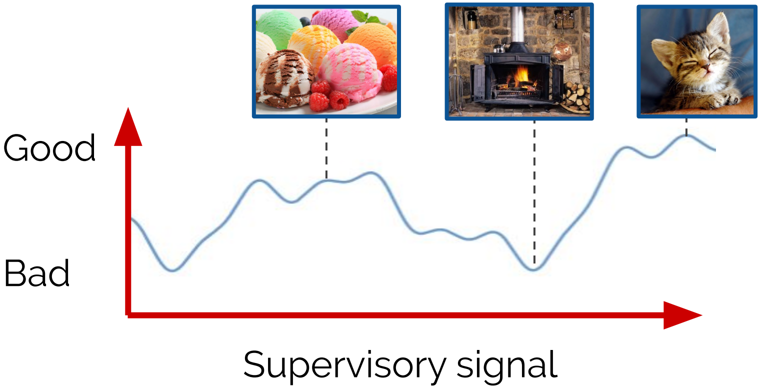
There is a large set of tasks, here's some listed by Goodfellow:



1. Classification
2. Regression
3. Transcription (image to characters, audio to characters)
4. Translation (english characters to french characters)
5. Anomaly detection (flag stuff that is unusual)
6. Synthesis & Inpainting (generating new examples similar to the training data)
7. Denoising, Superresolution, Colorisation, Relighting,...
8. Density estimation (estimating how likely a data example is observed)
9. ...

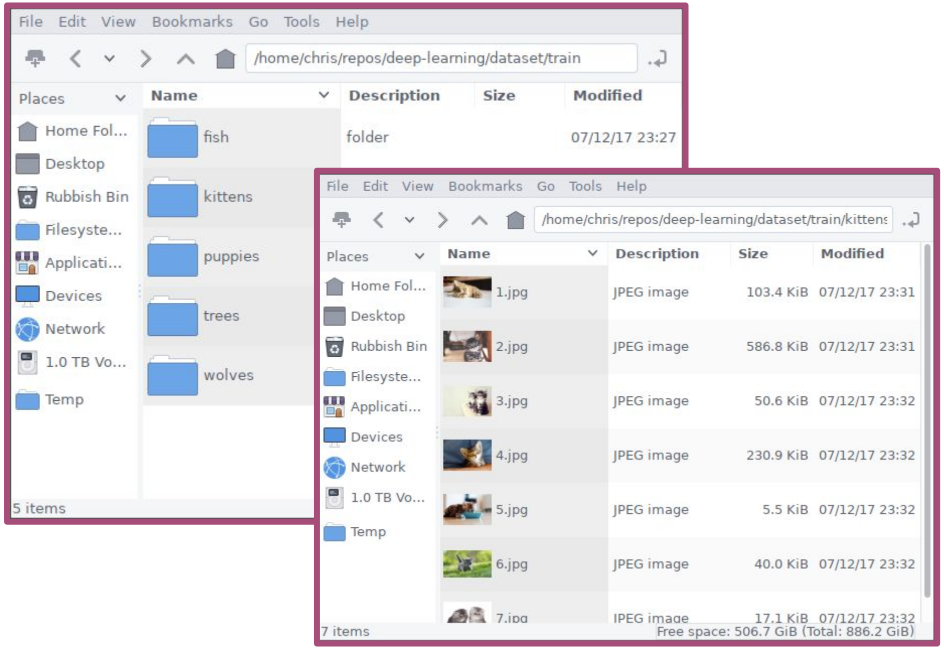
The Experience E

- Supervised
  - Supervisory signal/labels
    Guide the learning
- Unsupervised
  - Clustering
  - Dimensionality reduction
  - Generative models
- Semi-supervised
- Reinforcement learning
- One-shot learning
- Batch learning vs Online (incremental) learning



The Dataset

- Preparing a **good dataset is hard**
  - Garbage in, garbage out
- Learning the data distribution
- Train, test, and validate
- Overfitting and underfitting
- Bias
- Balance
- Augmentation



/home/chris/repos/deep-learning/dataset/train

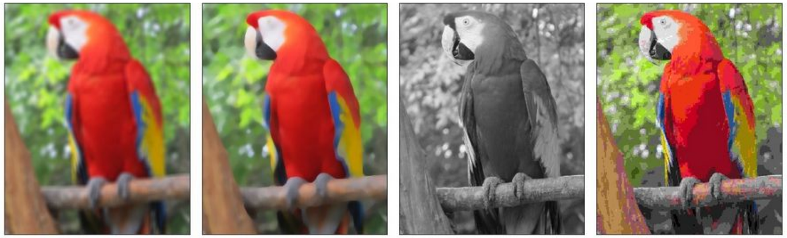
Name	Description	Size	Modified
fish	folder		07/12/17 23:27
kittens			
puppies			
trees			
wolves			

5 items

/home/chris/repos/deep-learning/dataset/train/kittens

Name	Description	Size	Modified
1.jpg	JPEG image	103.4 KiB	07/12/17 23:31
2.jpg	JPEG image	586.8 KiB	07/12/17 23:31
3.jpg	JPEG image	50.6 KiB	07/12/17 23:32
4.jpg	JPEG image	230.9 KiB	07/12/17 23:32
5.jpg	JPEG image	5.5 KiB	07/12/17 23:32
6.jpg	JPEG image	40.0 KiB	07/12/17 23:32
7.jpg	JPEG image	17.1 KiB	07/12/17 23:32

7 items Free space: 506.7 GiB (Total: 886.2 GiB)

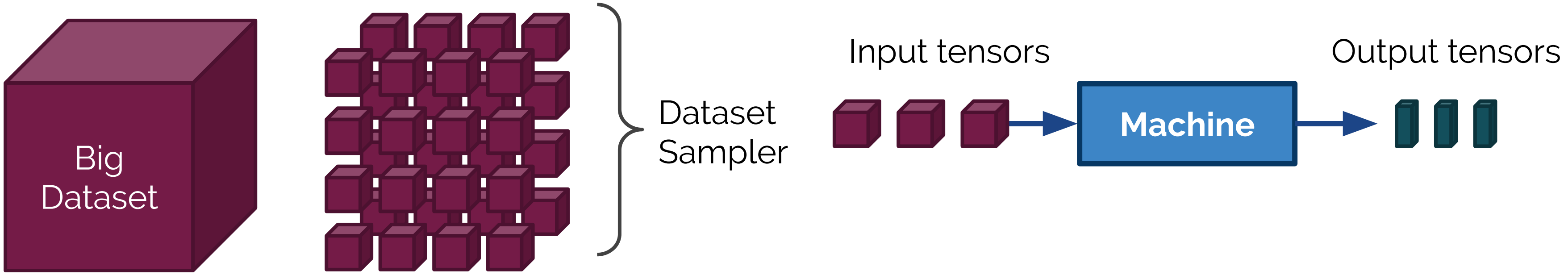


Source: https://github.com/albu/albumentations

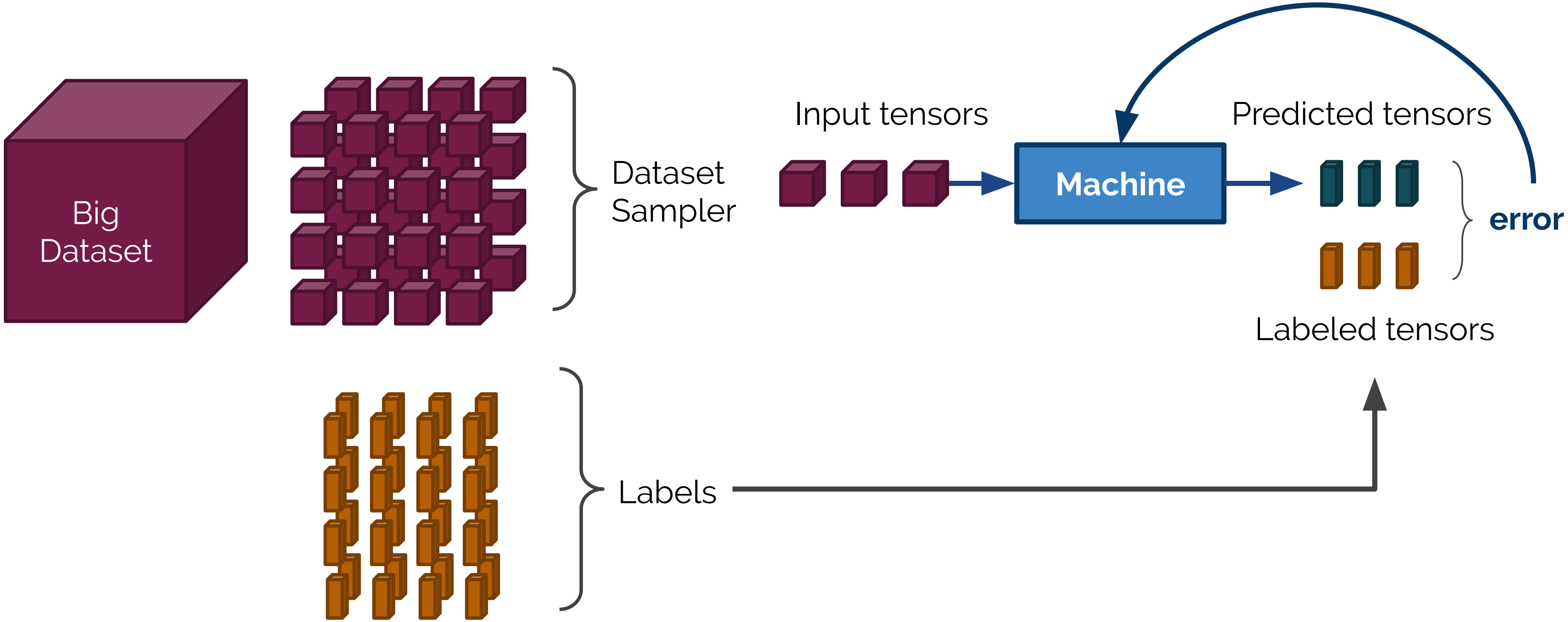
Experiencing a Dataset

In **machine** learning in practice, we have constraints:

- In most practical ML applications we don't have access to a continuous influx of **data over time**
- We have a static dataset that we wish to learn from, which can be quite large
- The **machine** we want to use has a "tiny" amount of available memory
  - Therefore we split our dataset down into **tensors** represented as multidimensional arrays

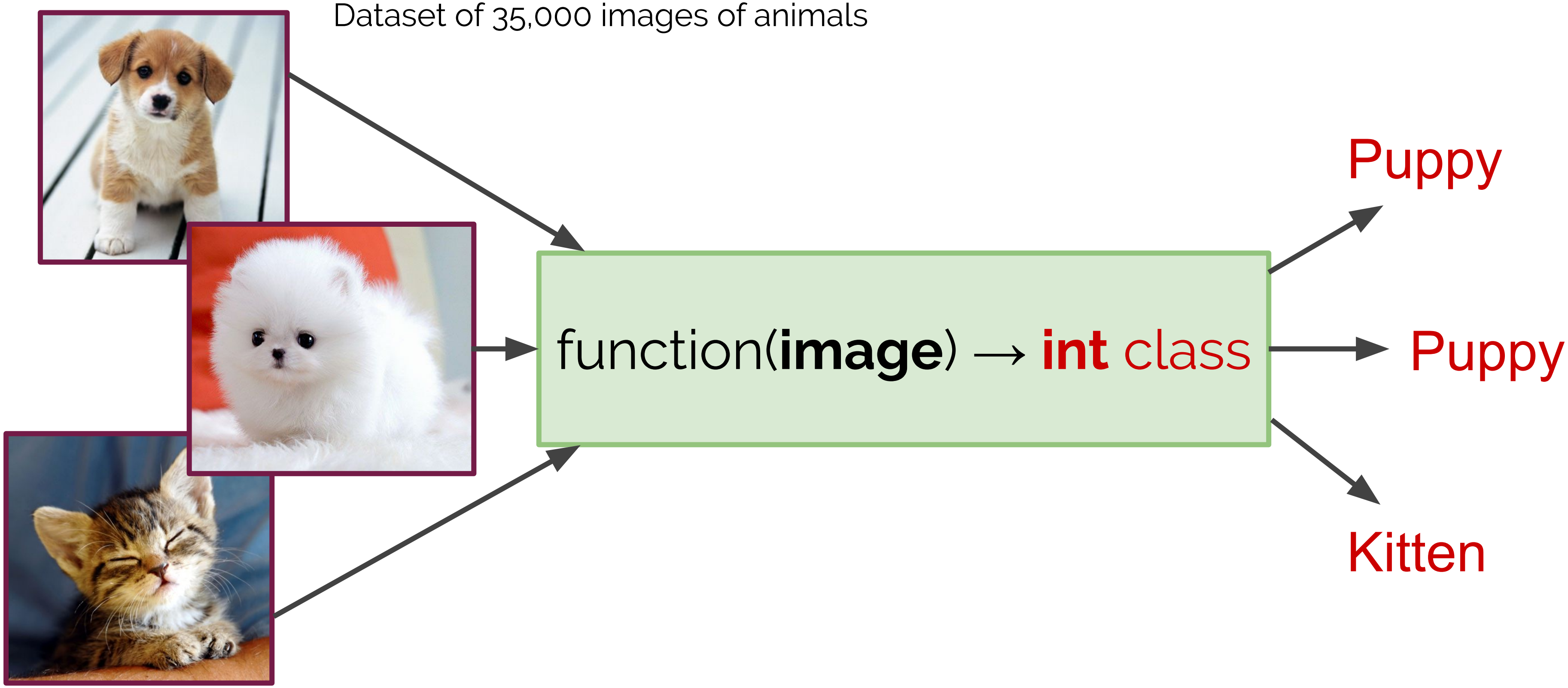


Supervised Learning



Classification Example

Dataset of 35,000 images of animals



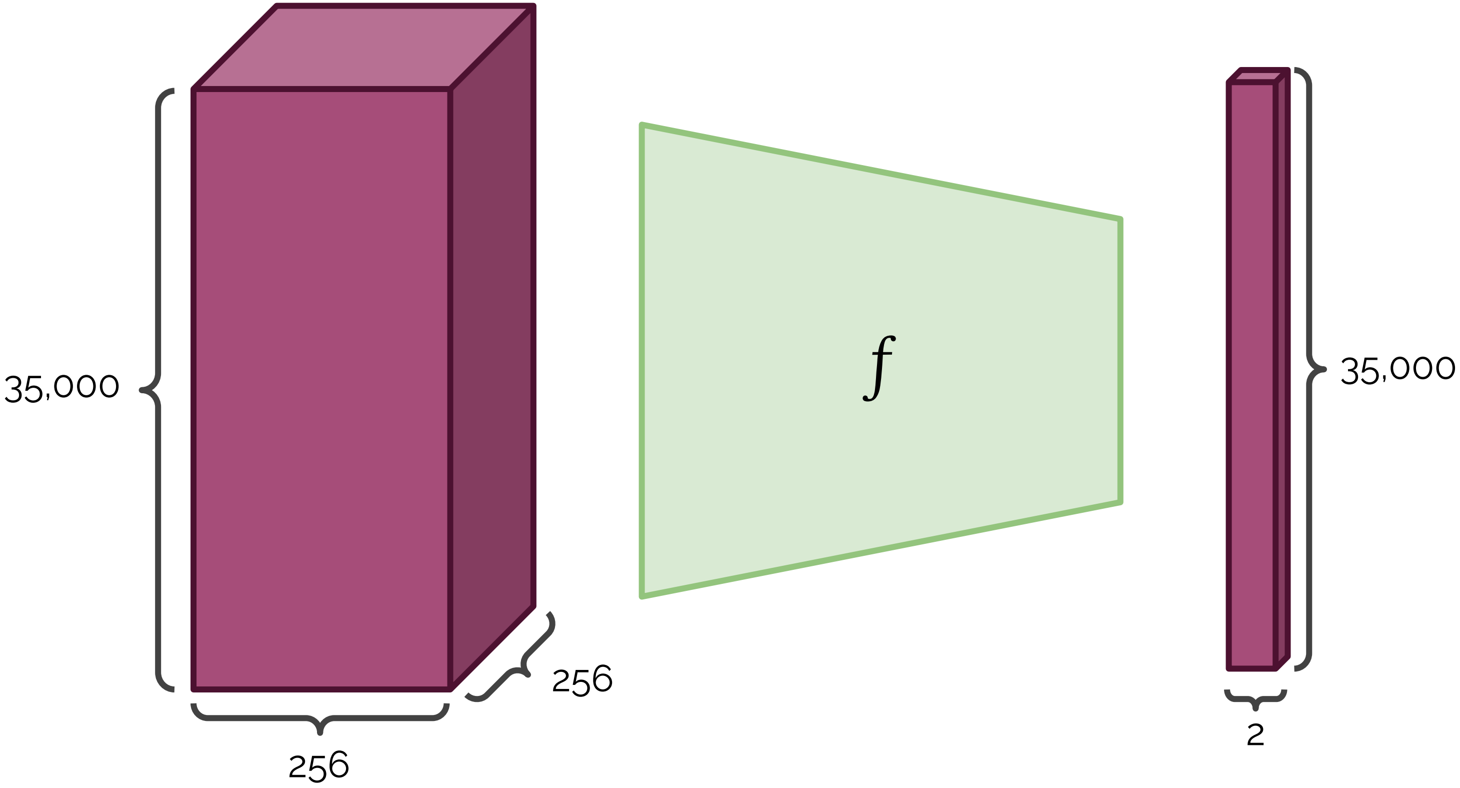
function(**image**) → **int** class

Puppy

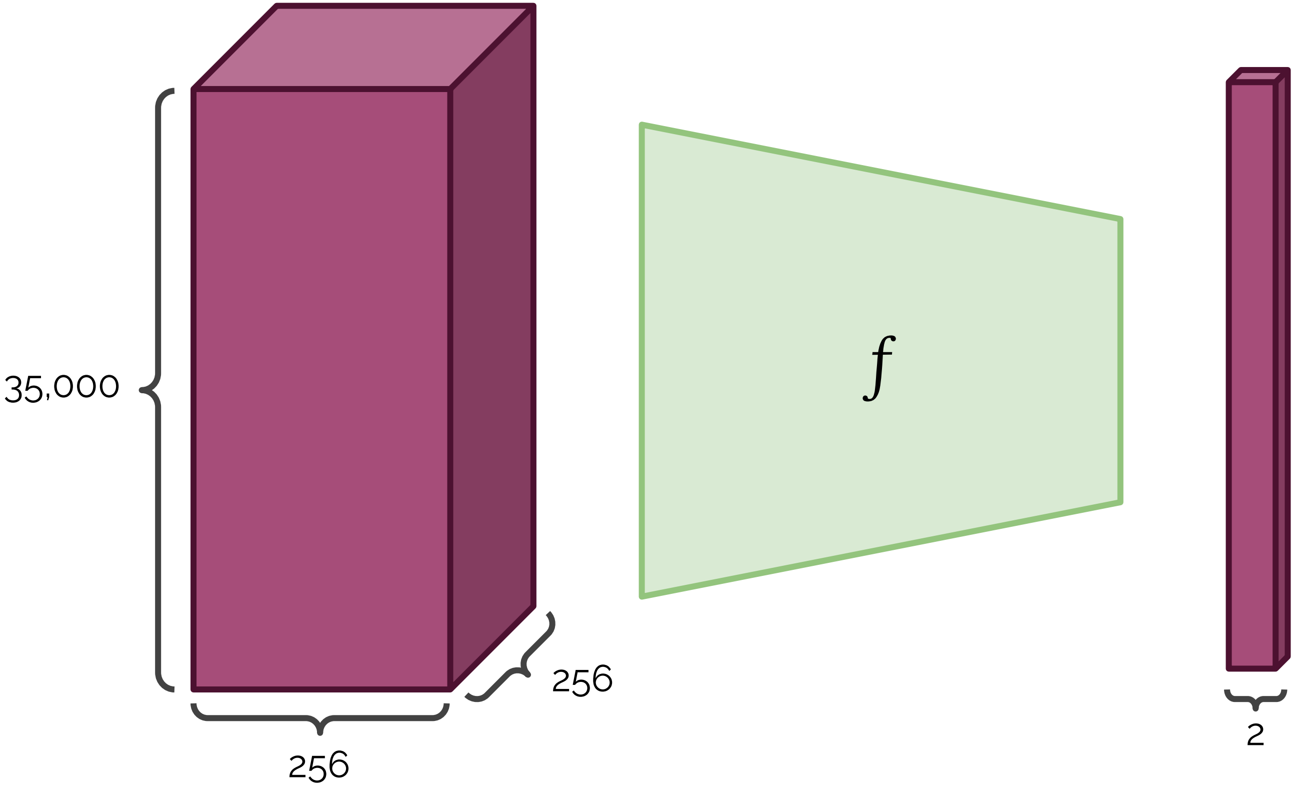
Puppy

Kitten

Tensors: Images → Classes

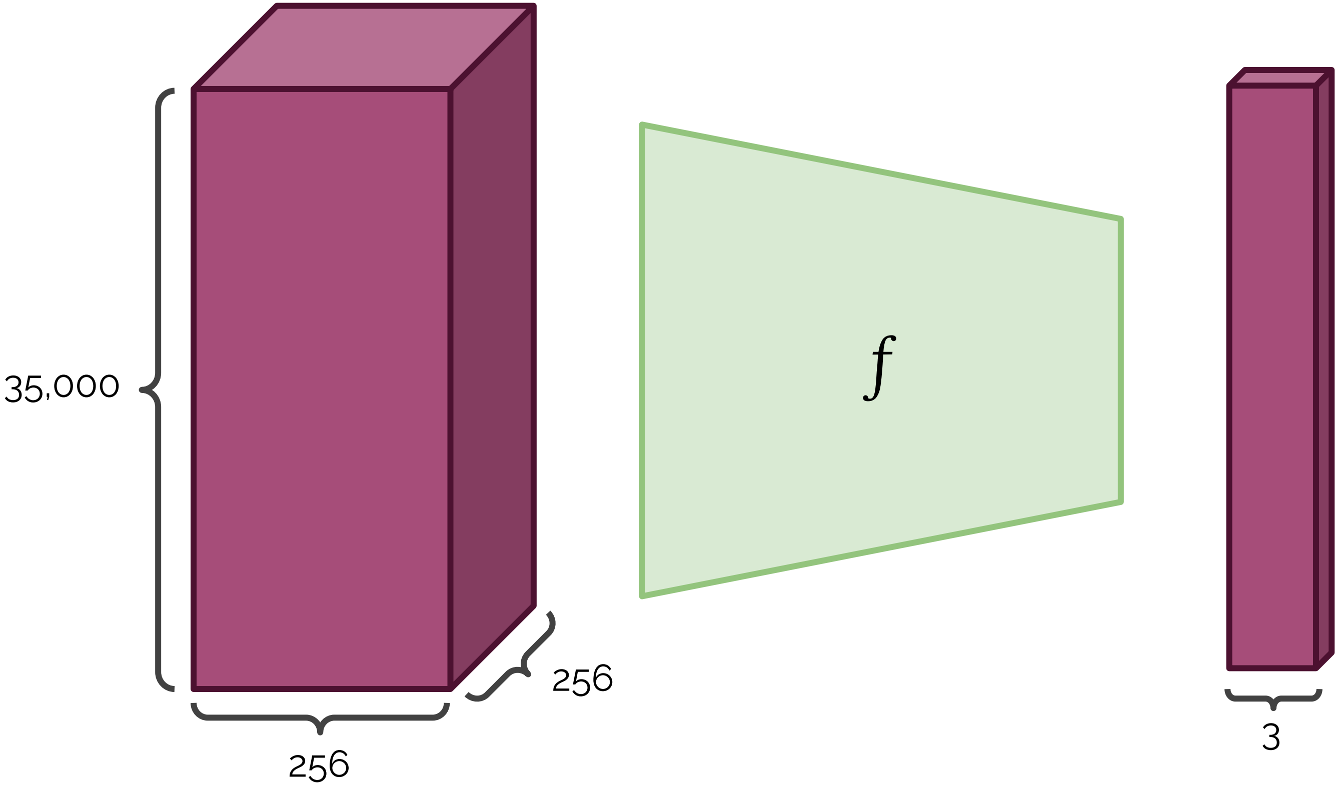


Tensors: Images → Classes



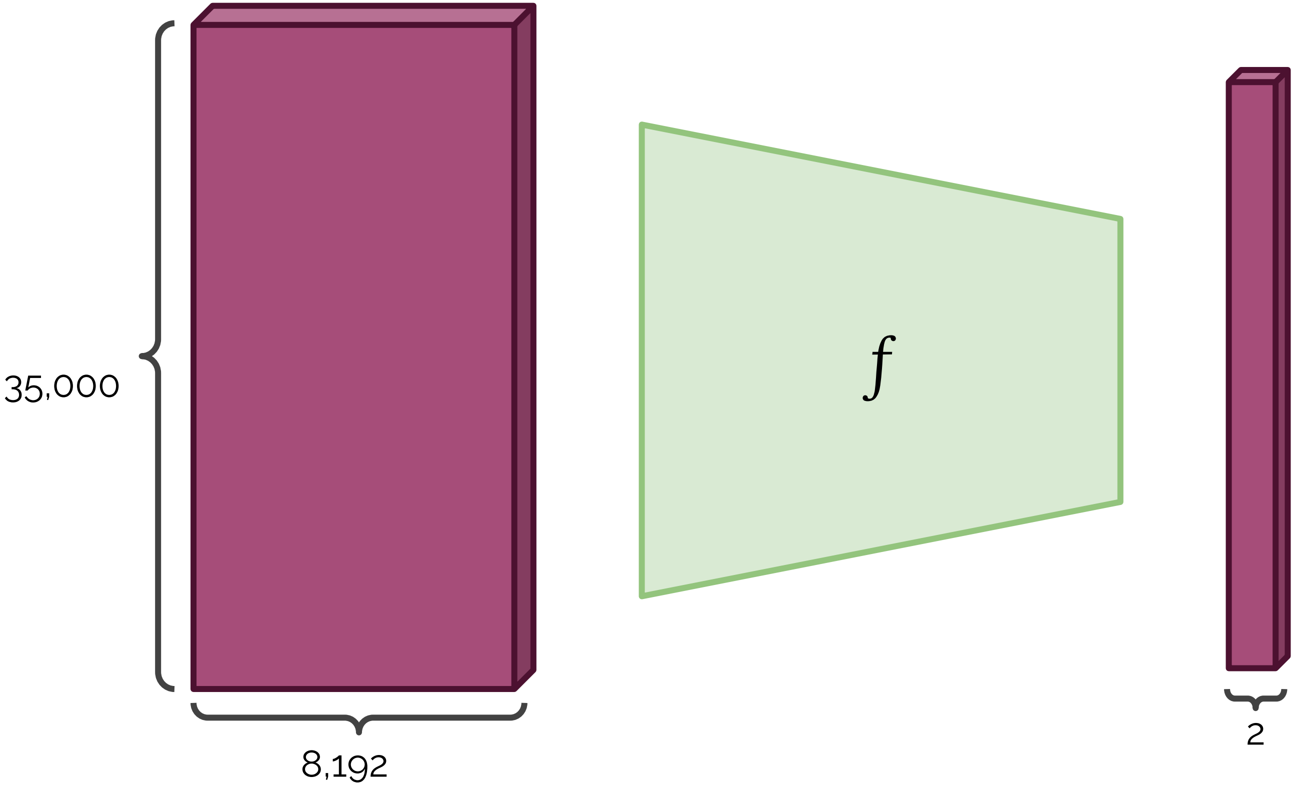
Dog	Cat
1	0
0	1
0	1
1	0
0	1
0	1
1	0
...	...

Tensors: Images → Classes



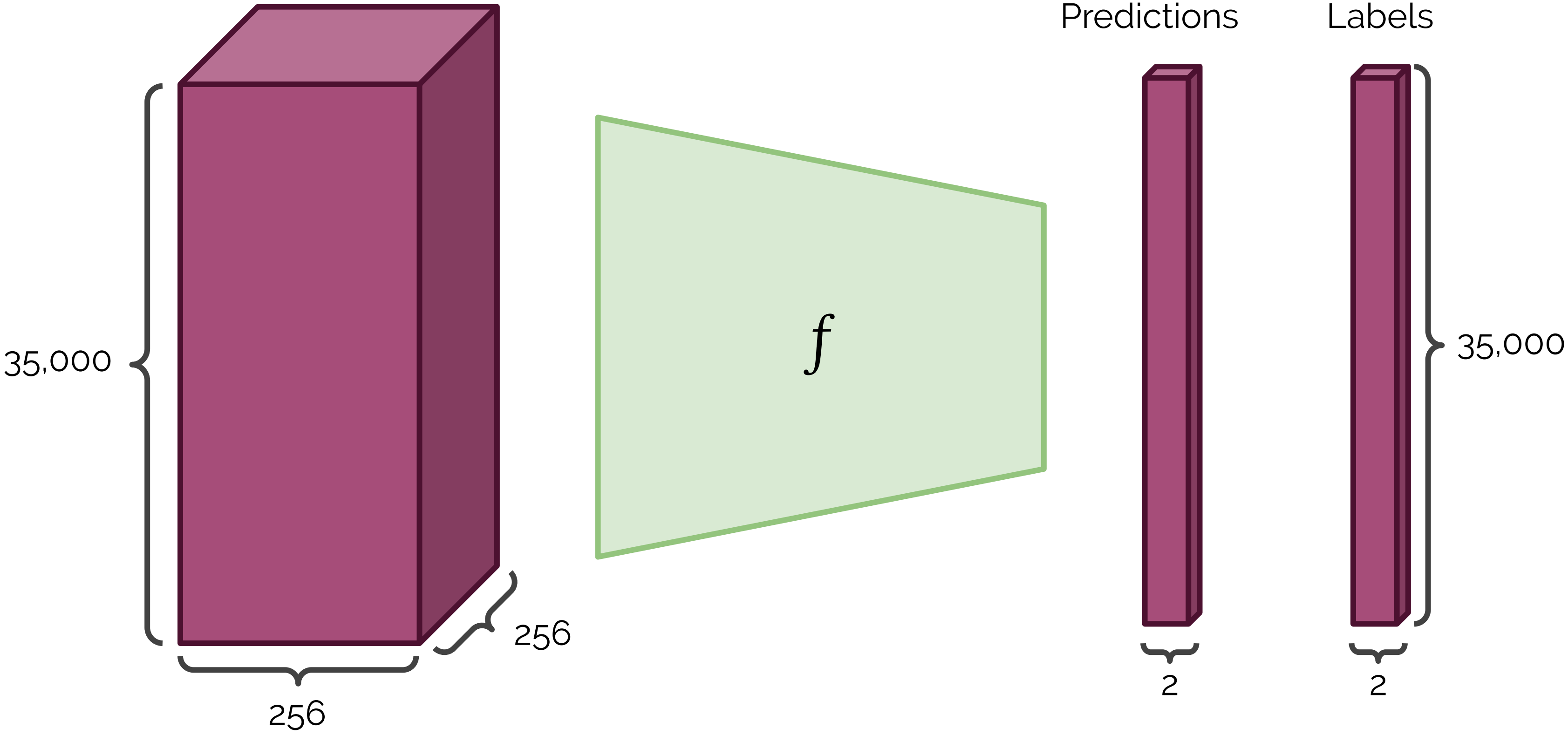
Dog	Cat	Fish
0	0	1
1	0	0
0	1	0
0	0	1
0	1	0
0	0	1
0	1	0
...	...	

Tensors: Audio → Classes

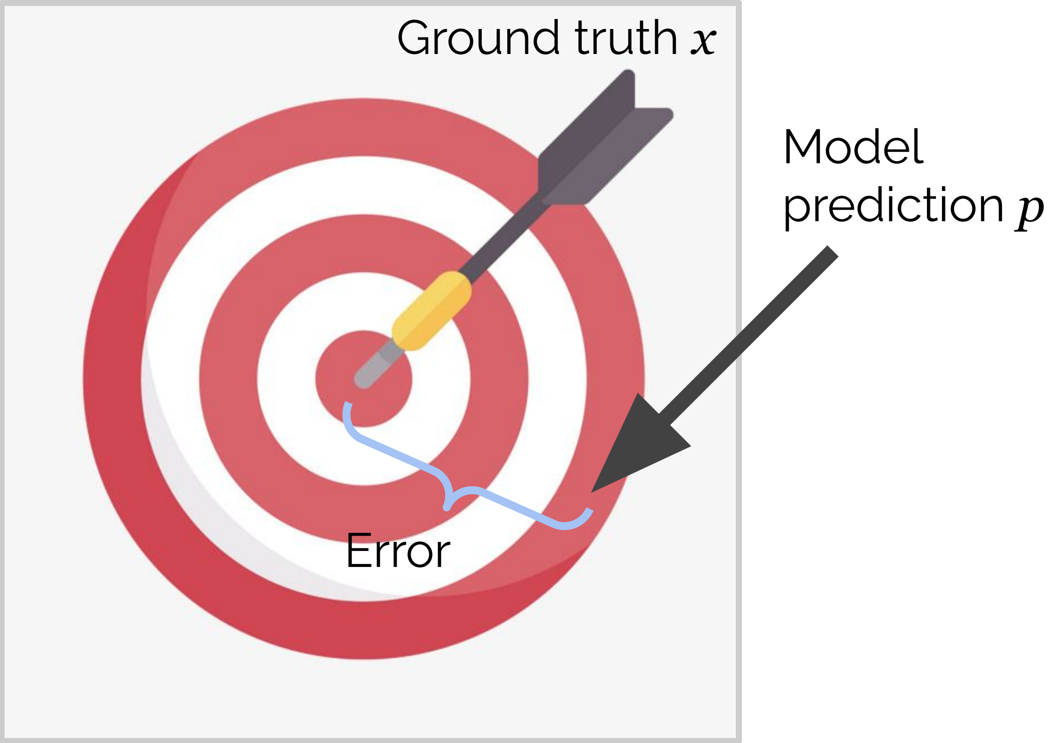


English	French
1	0
0	1
0	1
1	0
0	1
0	1
1	0
...	...

Predictions and Labels

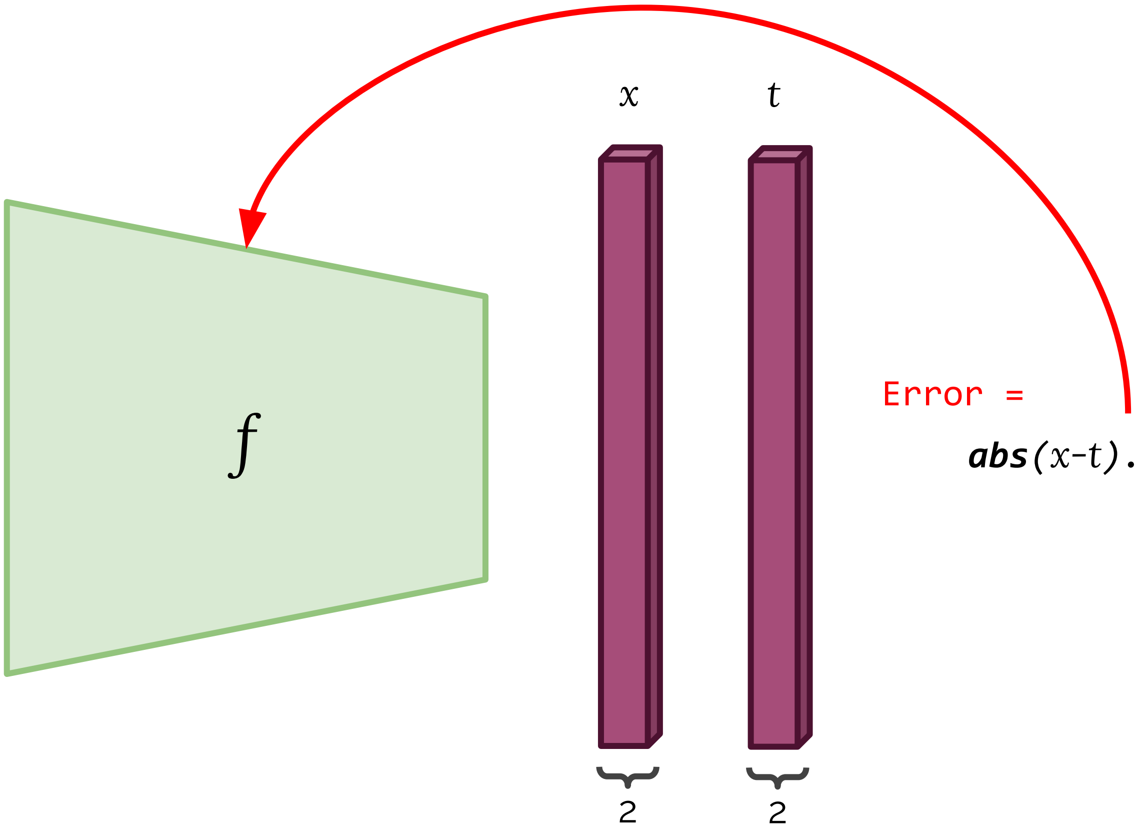


The Performance Measure *P*



- We need to define a quantitative measure of performance
- Typically we want it to give a **continuous**-valued score for each example
- Can be difficult to define for certain tasks

Loss (cost) function



Error =

abs(x-t).*sum*()

= 0.3914

L_1, L_2, loss functions

Durham University

$$\mathcal{L}_1(\mathbf{x}, \mathbf{t}) = \sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{t}_i|$$

```
loss = torch.abs(x-t).sum()
```

$$\mathcal{L}_2(\mathbf{x}, \mathbf{t}) = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{t}_i)^2}$$

similarly mean squared error, which is L_2, but ignore the $\sqrt{\ }$ and take the mean

```
loss = ((x-t)**2).mean()
```

L_p norm:

$$\|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

$$\|x\|_1 = \sum_i |x_i|$$

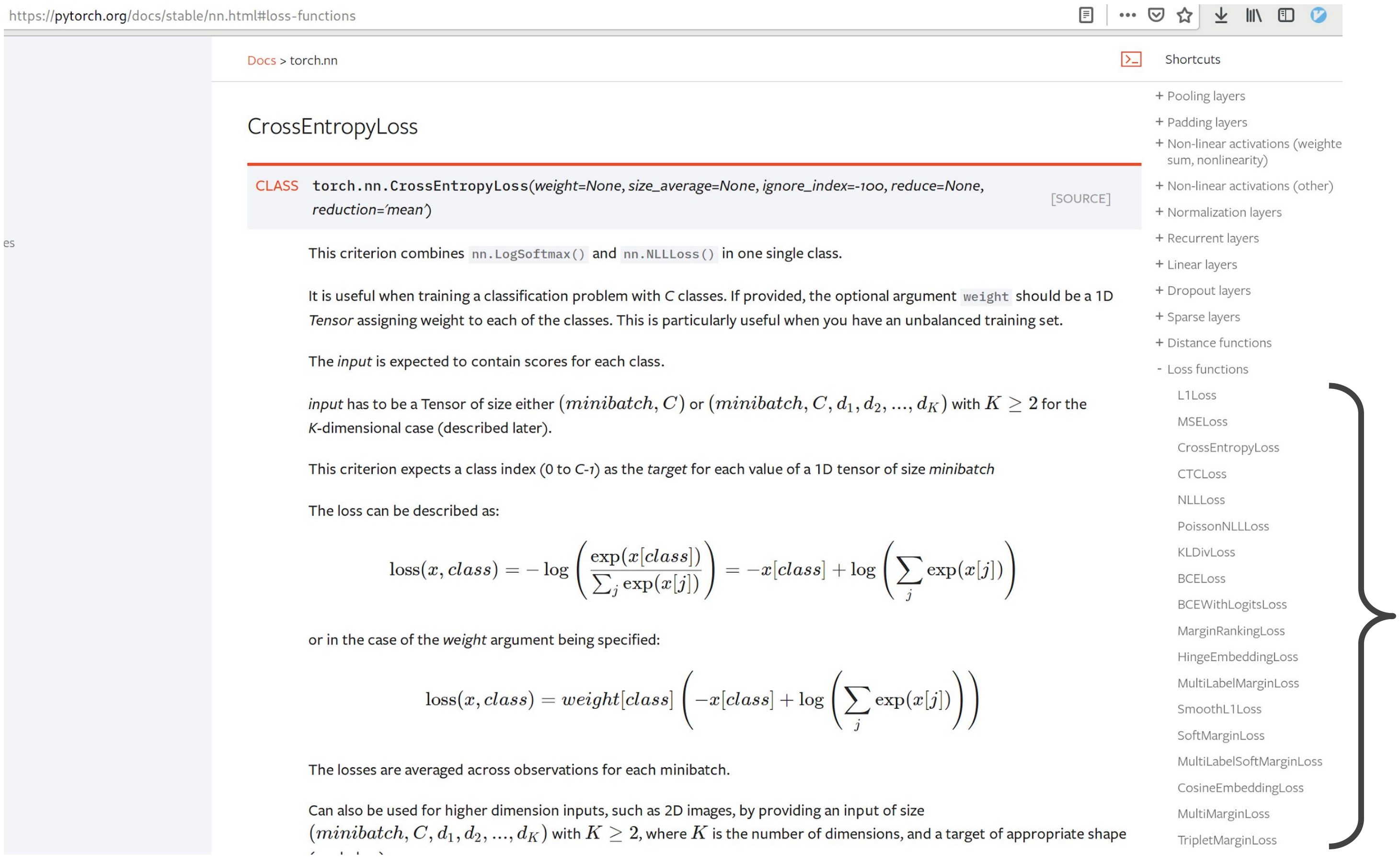
$$\|x\| = \|x\|_2 = \sqrt{\sum_i x_i^2}$$

Other error functions:

- Negative Log Likelihood
- Cross Entropy Loss

…

More loss functions



https://pytorch.org/docs/stable/nn.html#loss-functions

Docs > torch.nn

CrossEntropyLoss

CLASS `torch.nn.CrossEntropyLoss`(*weight=None*, *size_average=None*, *ignore_index=-100*, *reduce=None*, *reduction='mean'*) [SOURCE]

This criterion combines `nn.LogSoftmax()` and `nn.NLLLoss()` in one single class.

It is useful when training a classification problem with *C* classes. If provided, the optional argument `weight` should be a 1D *Tensor* assigning weight to each of the classes. This is particularly useful when you have an unbalanced training set.

The *input* is expected to contain scores for each class.

input has to be a Tensor of size either $(minibatch, C)$ or $(minibatch, C, d_1, d_2, ..., d_K)$ with $K \geq 2$ for the *K*-dimensional case (described later).

This criterion expects a class index (0 to *C-1*) as the *target* for each value of a 1D tensor of size *minibatch*

The loss can be described as:

$$\text{loss}(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right)$$

or in the case of the *weight* argument being specified:

$$\text{loss}(x, class) = weight[class]\left(-x[class] + \log\left(\sum_j \exp(x[j])\right)\right)$$

The losses are averaged across observations for each minibatch.

Can also be used for higher dimension inputs, such as 2D images, by providing an input of size $(minibatch, C, d_1, d_2, ..., d_K)$ with $K \geq 2$, where K is the number of dimensions, and a target of appropriate shape

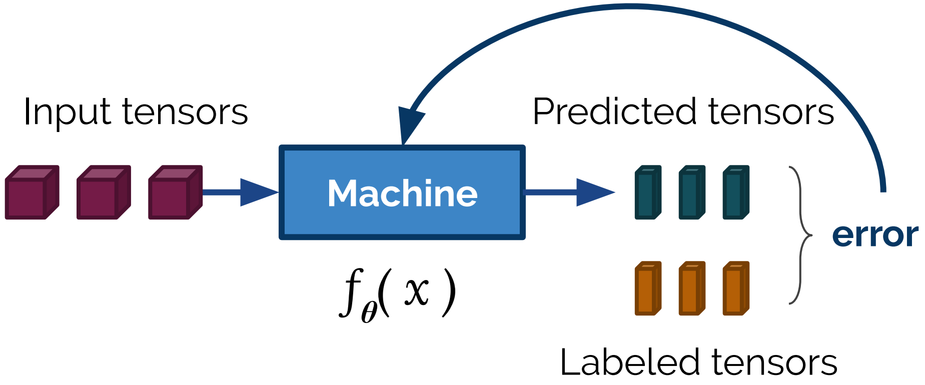
es

Shortcuts

- + Pooling layers
- + Padding layers
- + Non-linear activations (weighte sum, nonlinearity)
- + Non-linear activations (other)
- + Normalization layers
- + Recurrent layers
- + Linear layers
- + Dropout layers
- + Sparse layers
- + Distance functions
- - Loss functions
  - L1Loss
  - MSELoss
  - CrossEntropyLoss
  - CTCLoss
  - NLLLoss
  - PoissonNLLLoss
  - KLDivLoss
  - BCELoss
  - BCEWithLogitsLoss
  - MarginRankingLoss
  - HingeEmbeddingLoss
  - MultiLabelMarginLoss
  - SmoothL1Loss
  - SoftMarginLoss
  - MultiLabelSoftMarginLoss
  - CosineEmbeddingLoss
  - MultiMarginLoss
  - TripletMarginLoss

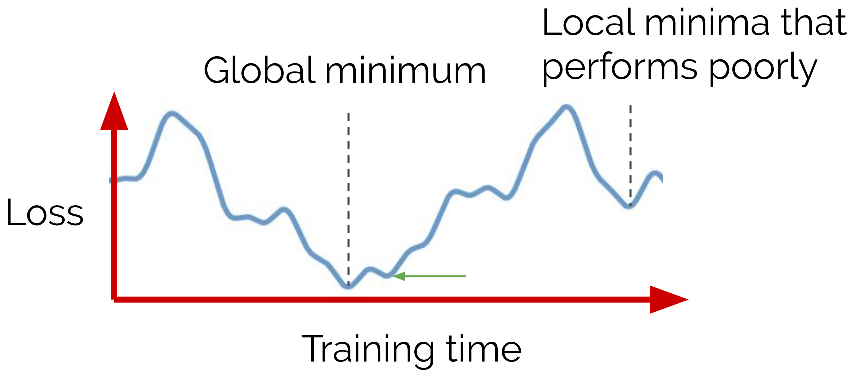
Further reading

Optimization

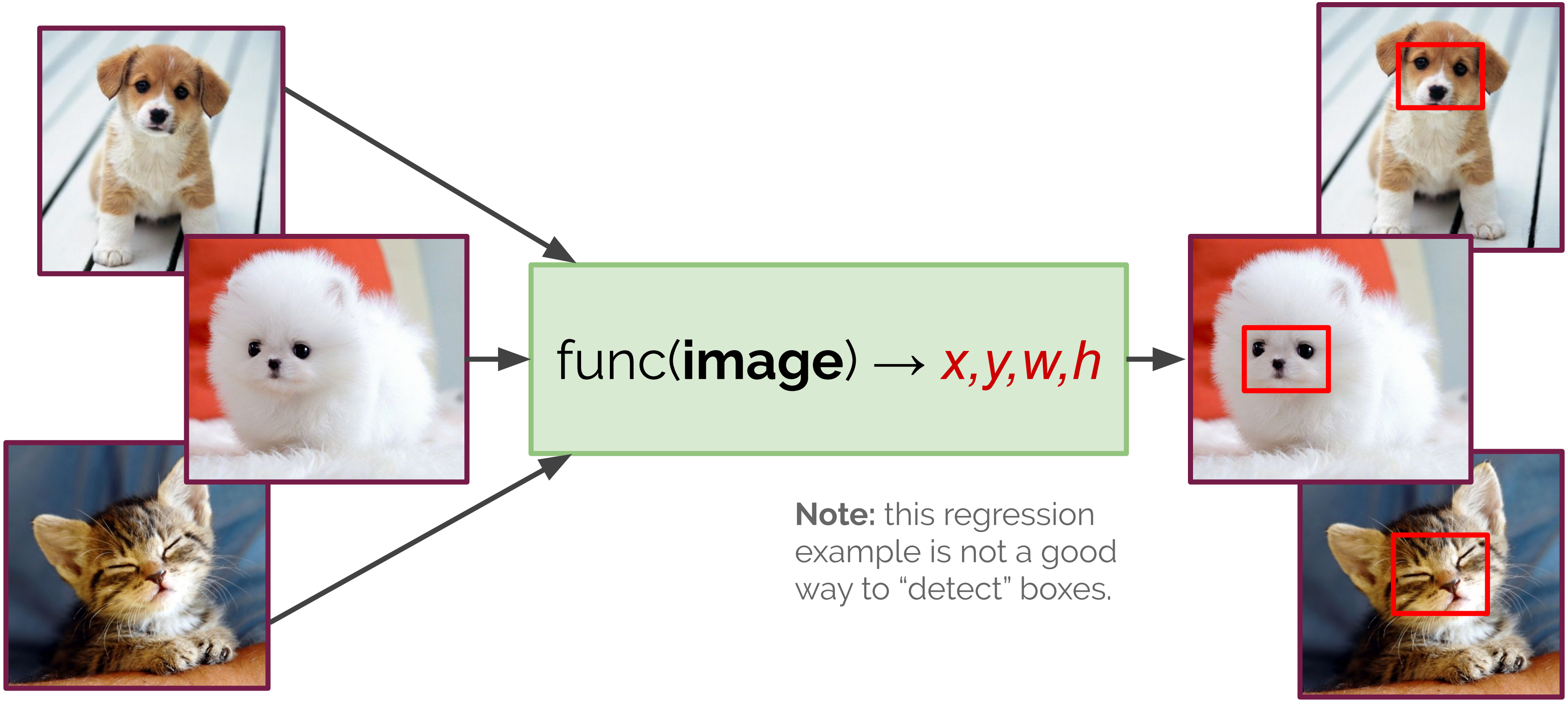


- We update the machine parameters $\boldsymbol{\theta}$ such as to minimise the objective function.
- We can choose an optimisation strategy based on the shape of the output space.

- We don't always want to find the global minimum - often this means the machine has simply memorised the input dataset
- Instead a good local minima such as the green arrow, may be sufficient.



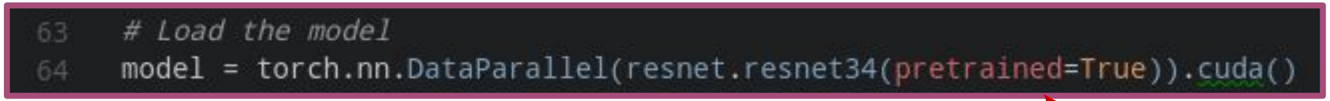
Regression Example



func(**image**) → *x,y,w,h*

Note: this regression example is not a good way to “detect” boxes.

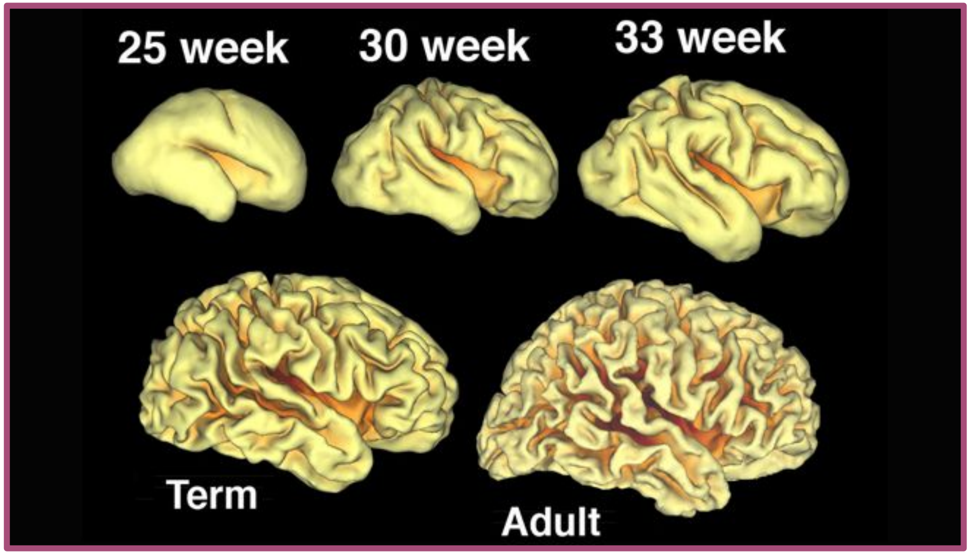
Transfer Learning



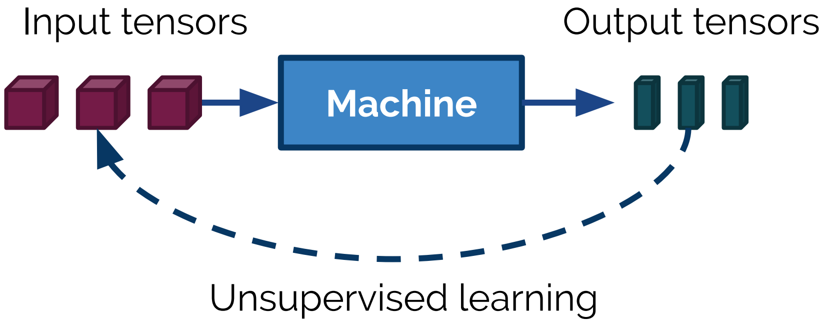
```
# Load the model
model = torch.nn.DataParallel(resnet.resnet34(pretrained=True)).cuda()
```

Nice!

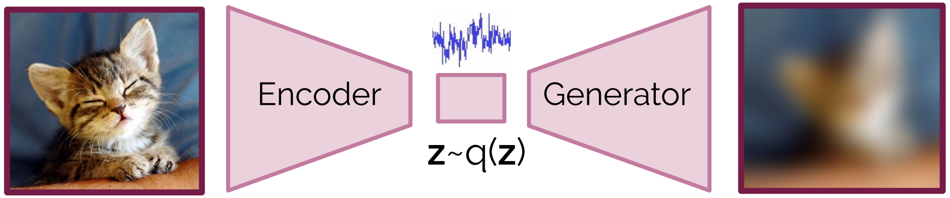
- Is it easier to train a **baby** to detect cancer or an **adult** to detect cancer?
- Train model on complex tasks with **<u>lots</u>** of public data (even until they outperform humans)
- Then change the data to our tasks, and continue training



Unsupervised Learning



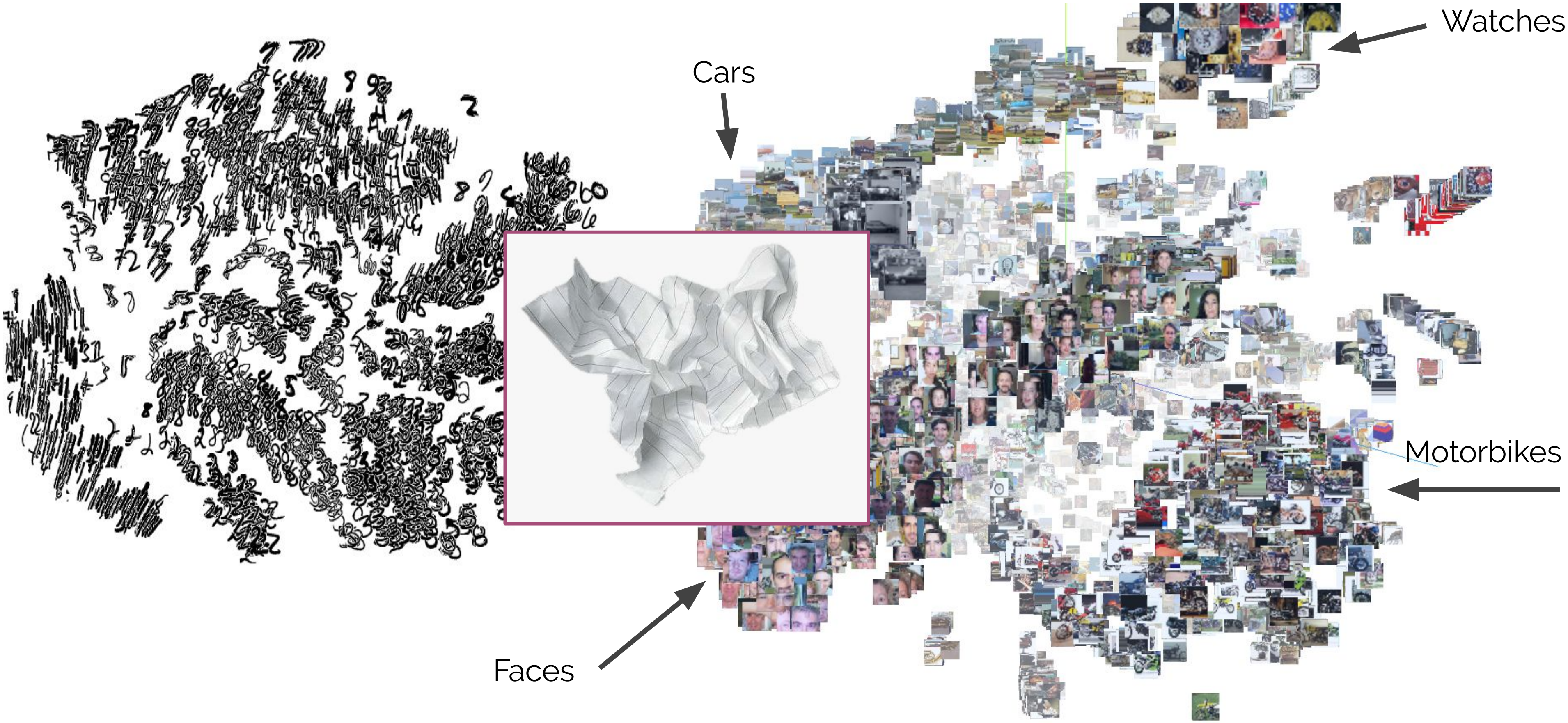
Example: reconstructive autoencoder



Example tasks:

- Generating new examples similar to the training data
- Inpainting (with some data removed, can predict value of missing entries)
- Denoising
- Density estimation (estimating how likely a data example is observed)

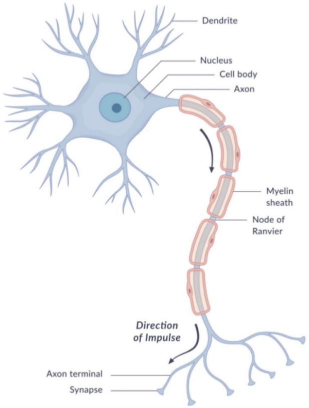
Manifolds, Density Estimation, and Dimensionality Reduction



Take away points

Durham University

- Deep learning has lots of crossover with learning in nature
- Most research is very different and uses overly simplistic models
- Mainly advances in compute hardware and tooling have given rise of the past waves
  - GPUs
  - Automatic differentiation
- The field is currently mostly data driven
  - Especially for supervised learning
  - It's very easy to think a model is doing well, when it is actually just overfitting
- Next week, PyTorch!



Good books:

1. Deep Learning Book, Goodfellow
2. Pattern Recognition and Machine Learning, Bishop