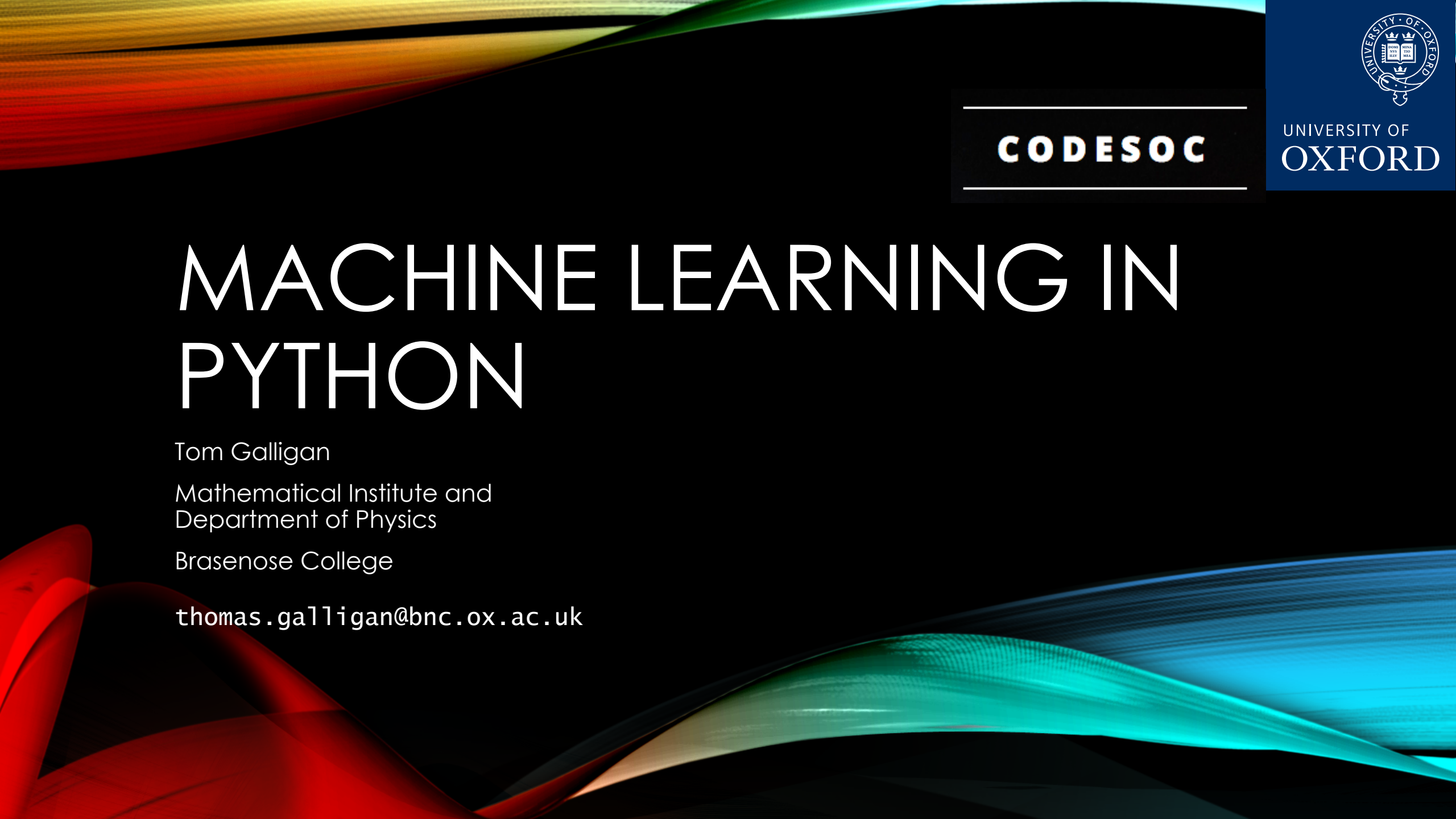


MACHINE LEARNING IN PYTHON

Tom Galligan

Mathematical Institute and
Department of Physics

Brasenose College

thomas.galligan@bnc.ox.ac.uk

PREREQUISITES

To get the most out of this course, you should:

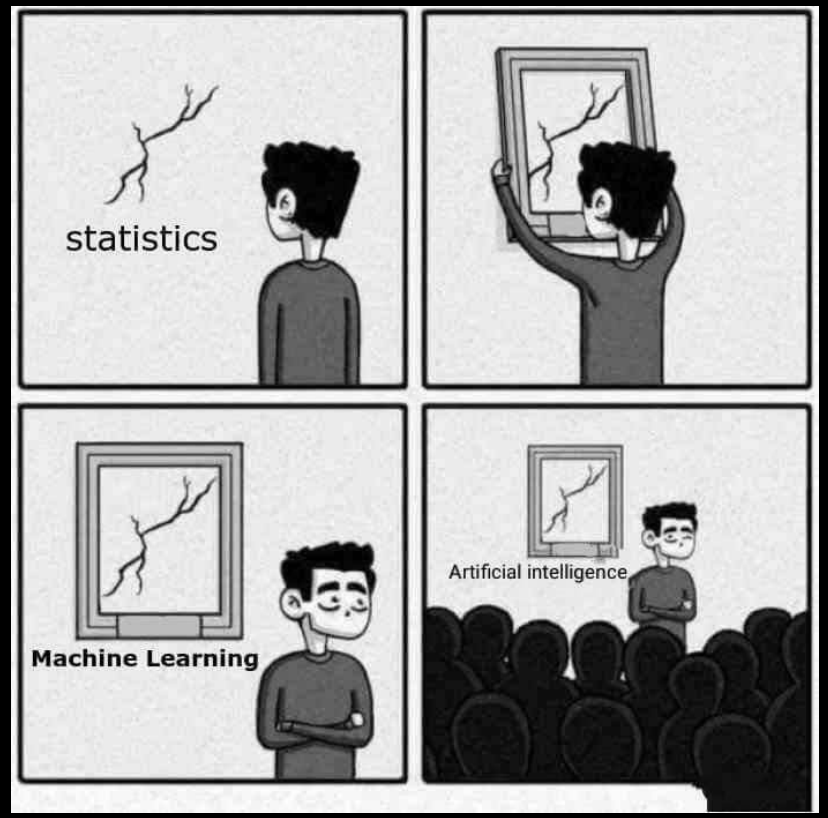
- Have a basic understanding of programming (this course will use Python)
- Have preinstalled the required software onto your laptop, which you can bring to each session
- Be comfortable with basic maths and statistics

INTRODUCTION: WHAT IS MACHINE LEARNING?

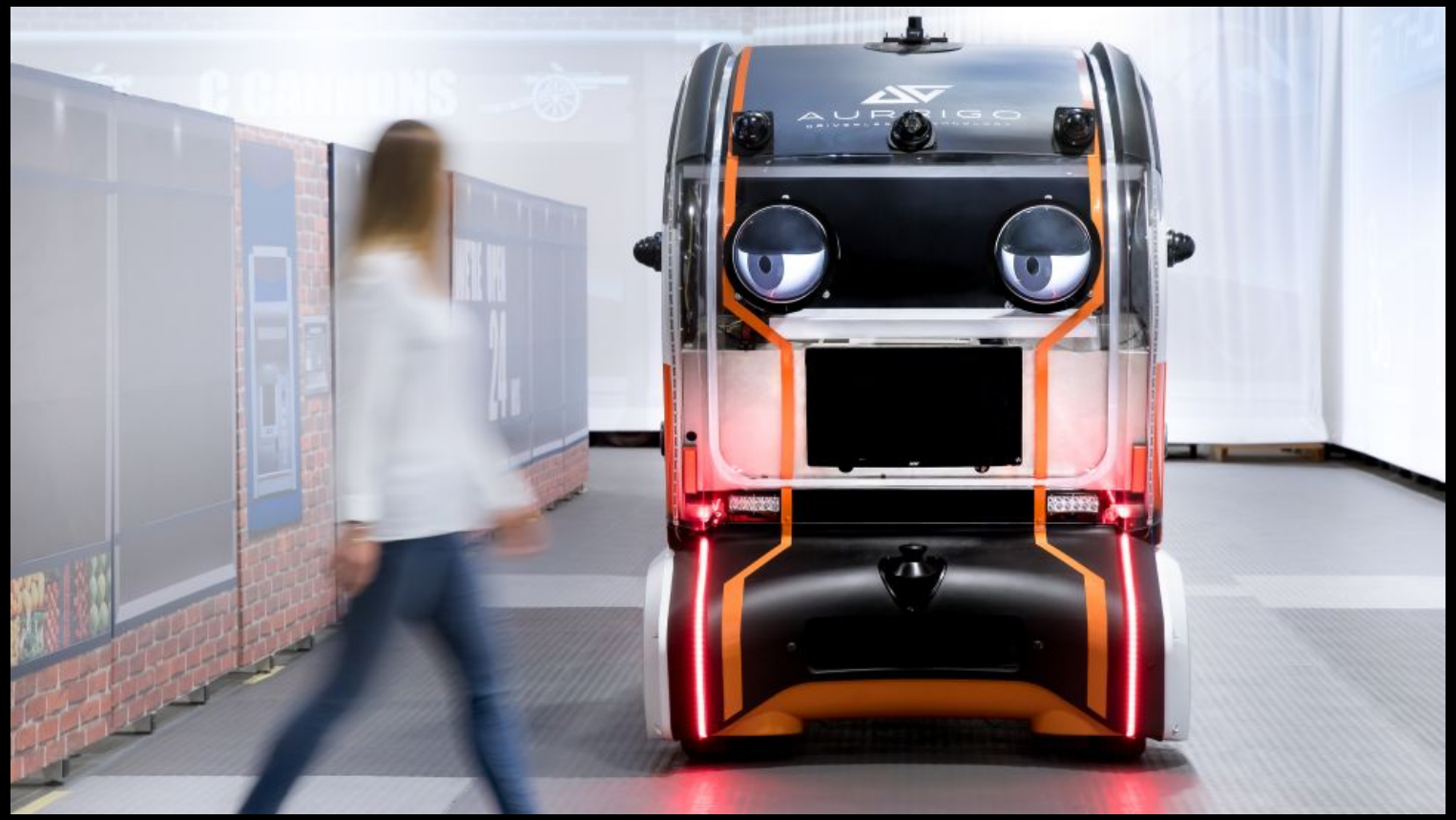
- Teaching computers to perform a particular task
- As proficiently as possible
- **Without explicit programming**

WHAT IS MACHINE LEARNING **NOT**?

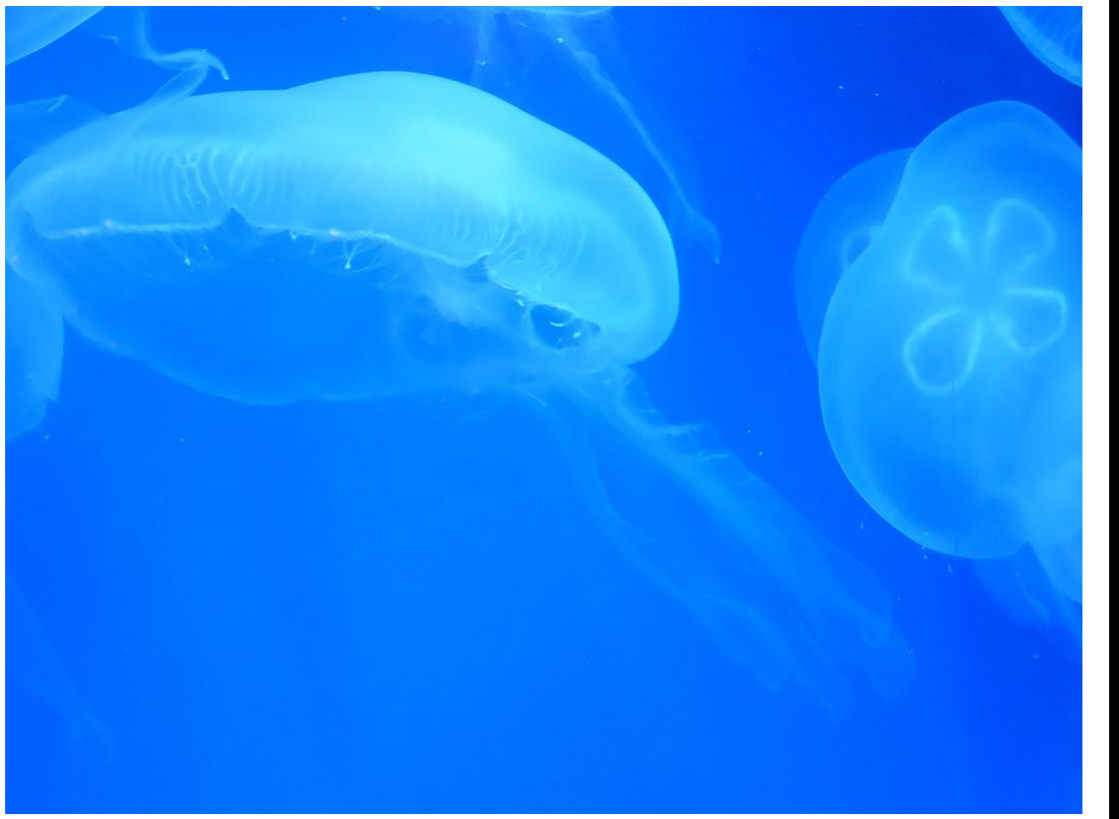
- Teaching computers to "think"
- "Just glorified statistics"
- Capable of inferring causation
- Magic

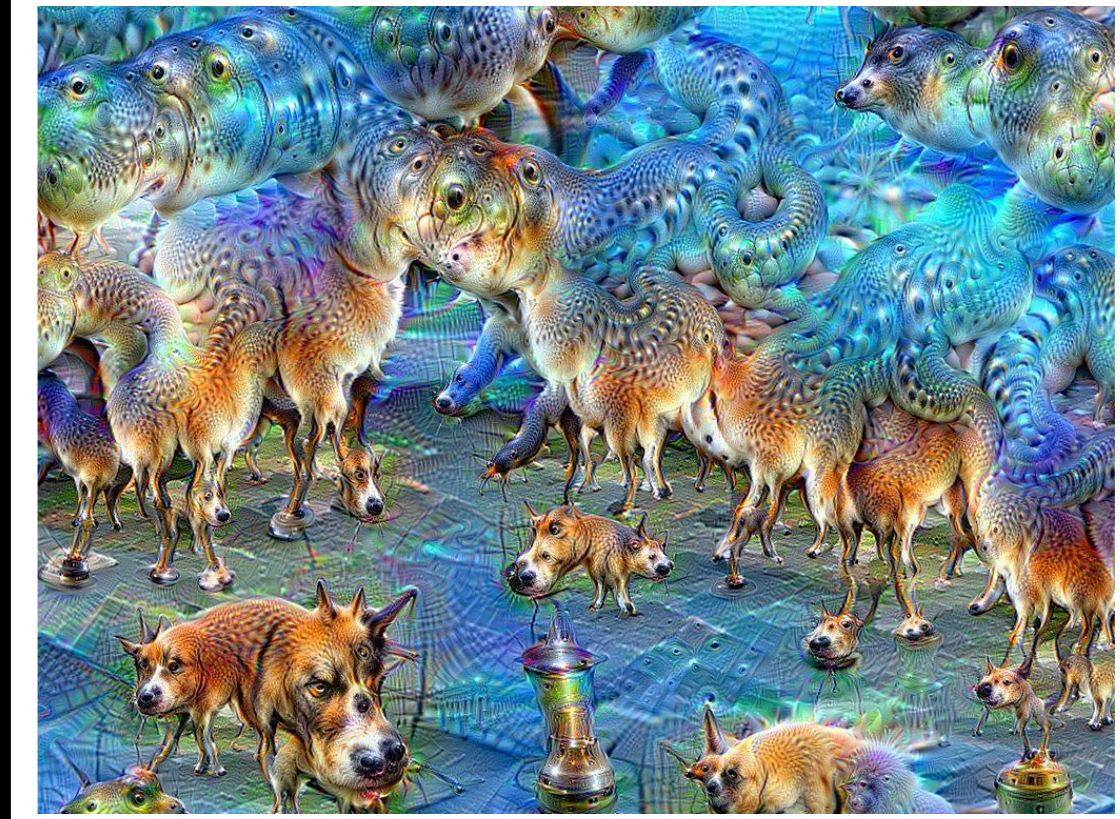


MACHINE LEARNING IN PRACTICE



Jaguar/Land Rover





https://deepdreamgenerator.com/

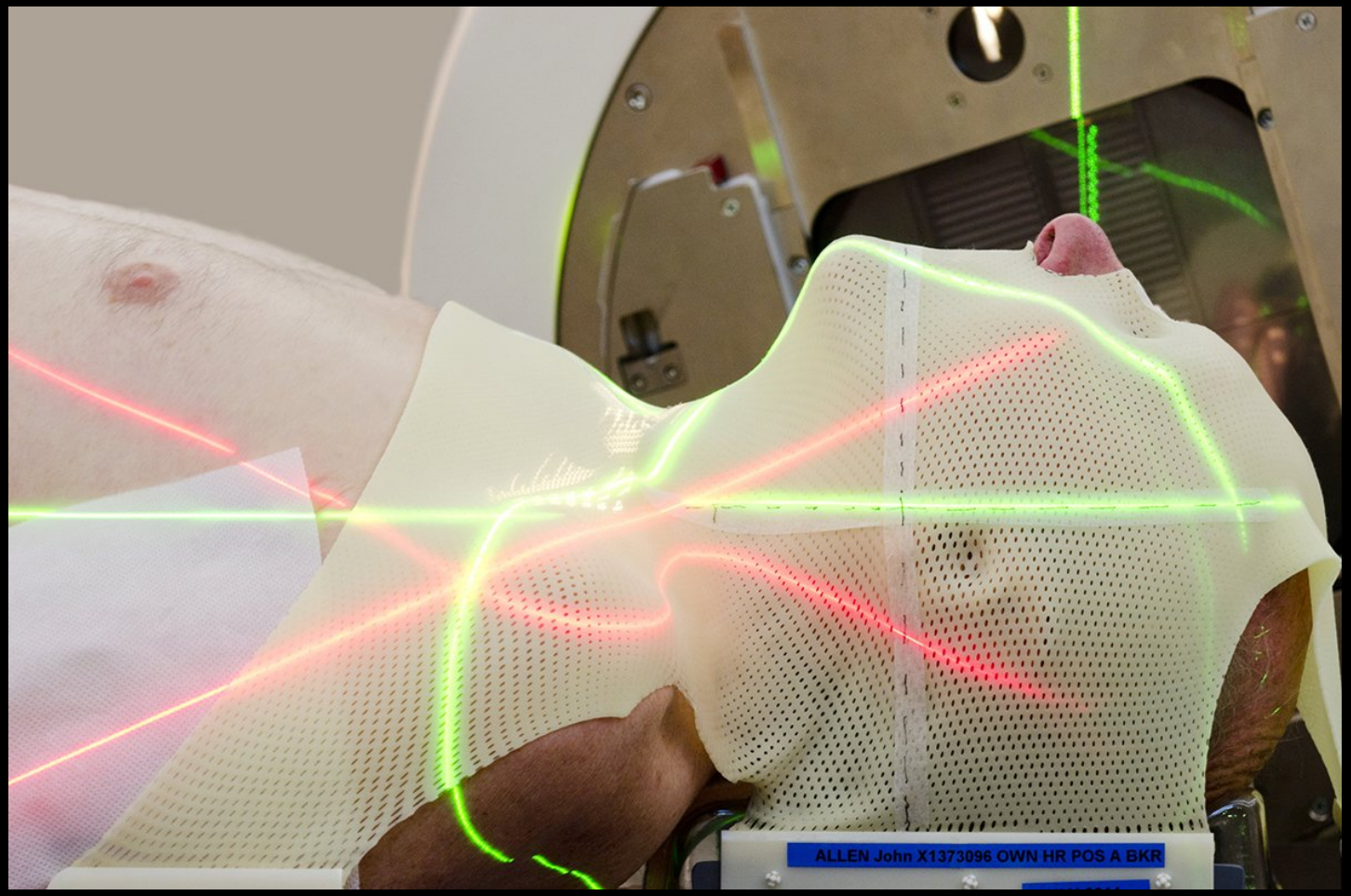


Image credit: Google DeepMind Health – radiotherapy planning

PHYSICS

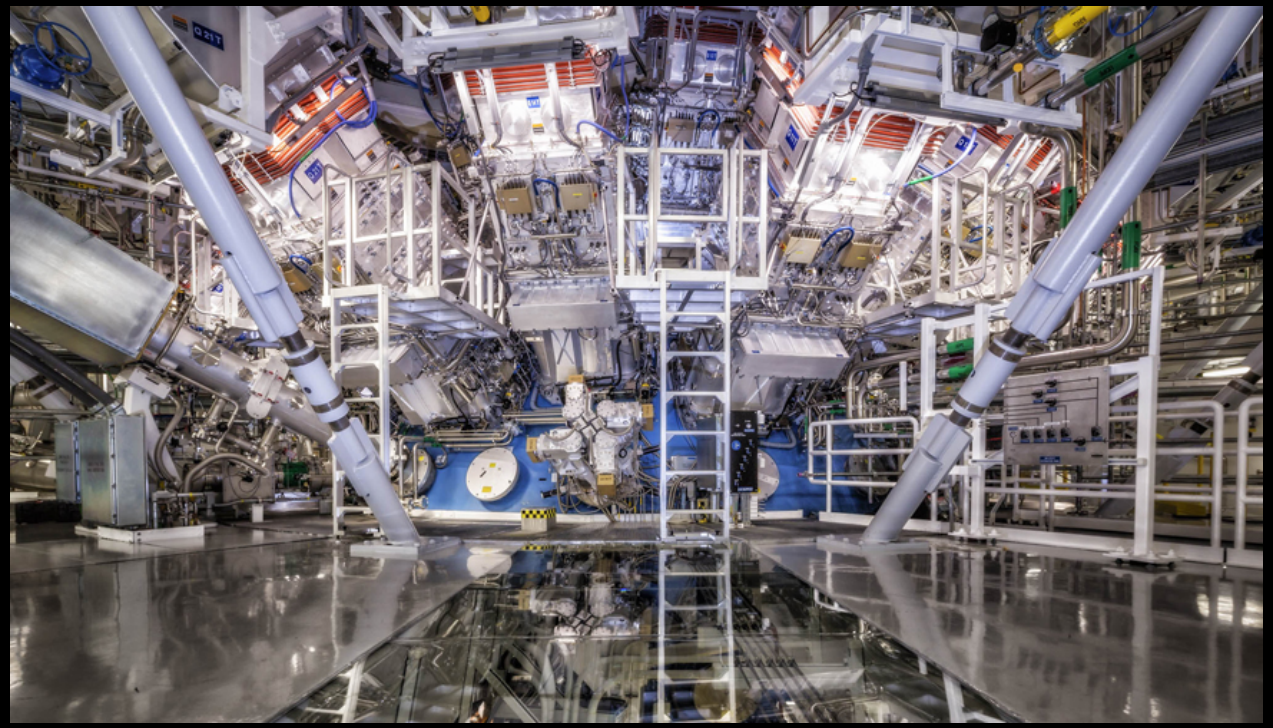
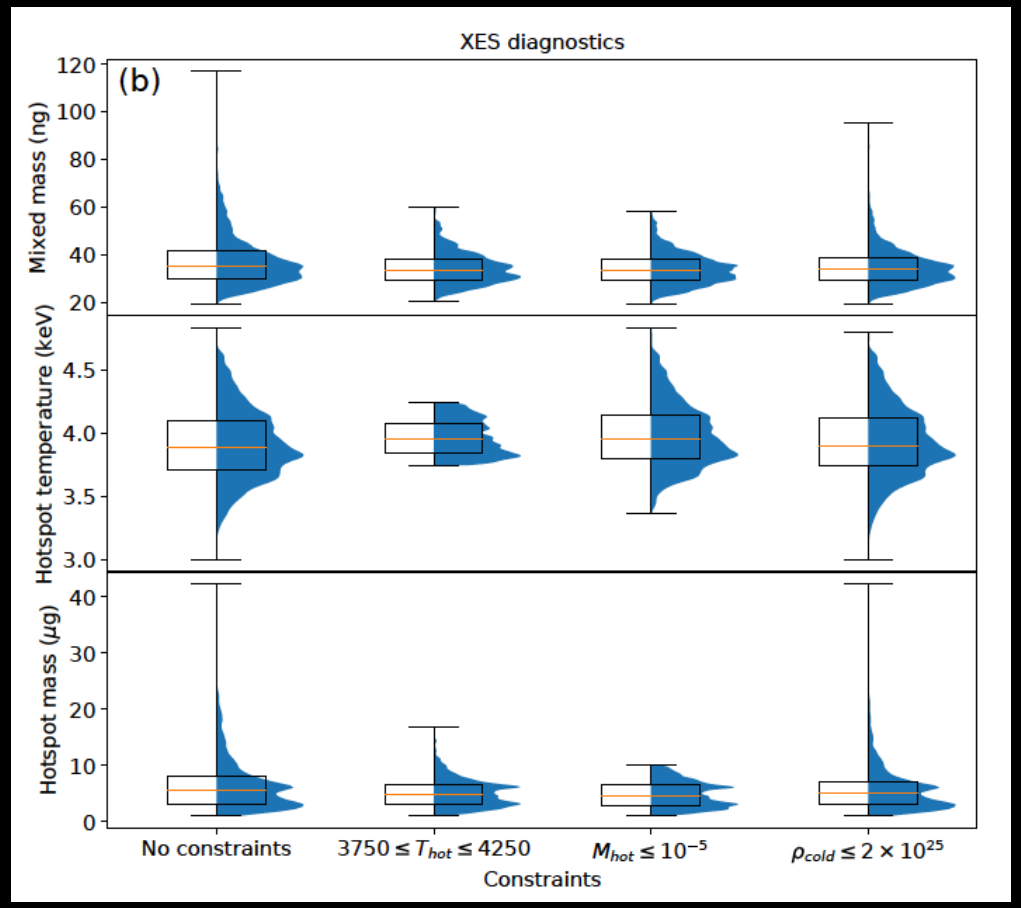
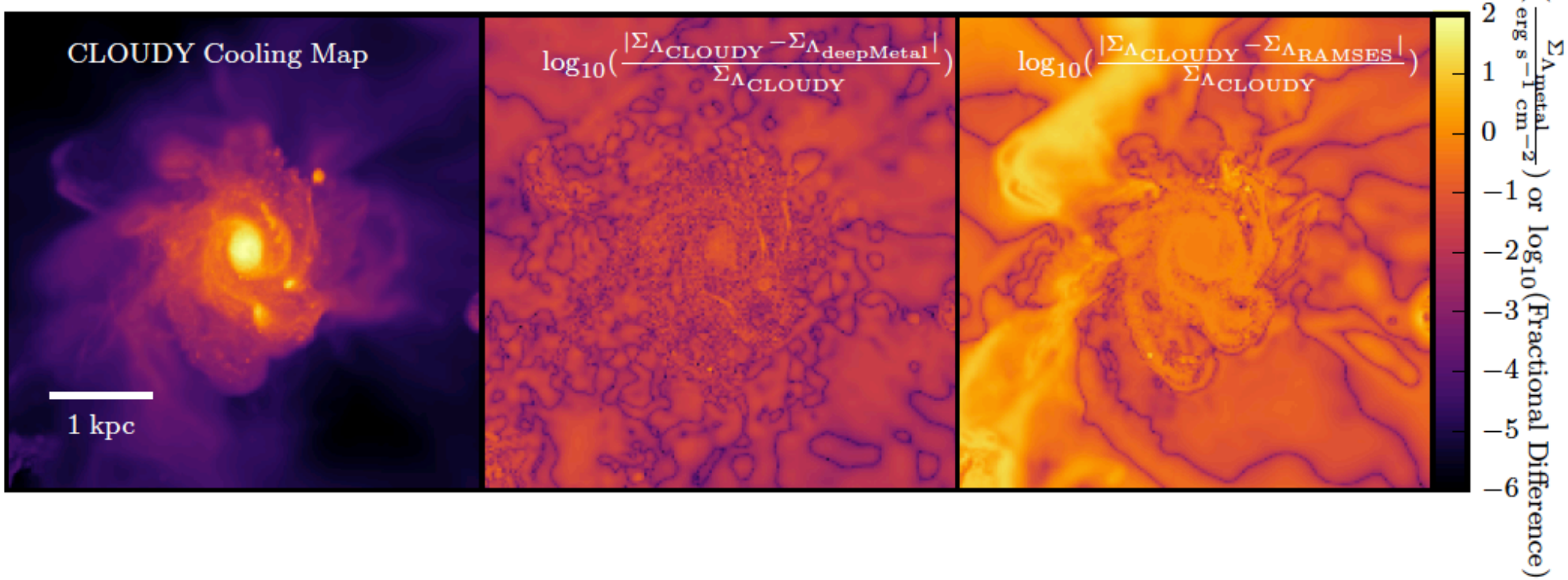


Image Credit: Damien Jameson/LLNL



Kasim, Galligan et al., *(submitted)* (2018)

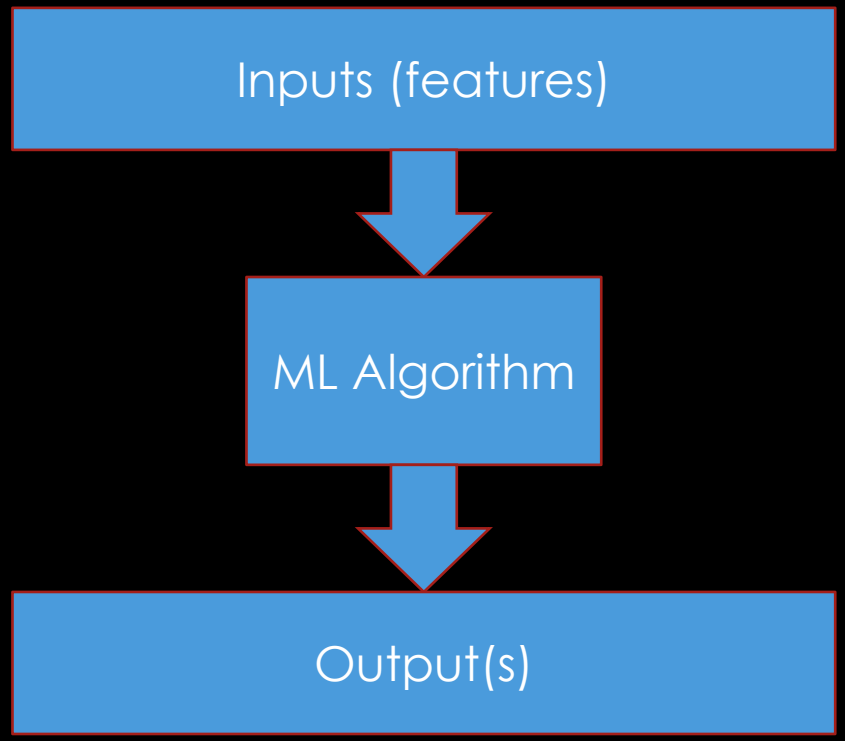
PHYSICS



Galligan & Katz, *MNRAS* (2018) (in prep)

THE GOAL OF A MACHINE LEARNING ALGORITHM

- We have an object that belongs to one of several categories.
- The category it's in depends on certain parameters (features)
- We don't know the relationship between the features and the form of the object
- BUT we have lots of examples of this object, its features and what category each example belongs to!
- Can we train an algorithm on these examples and get it to accurately predict the categories of new object.

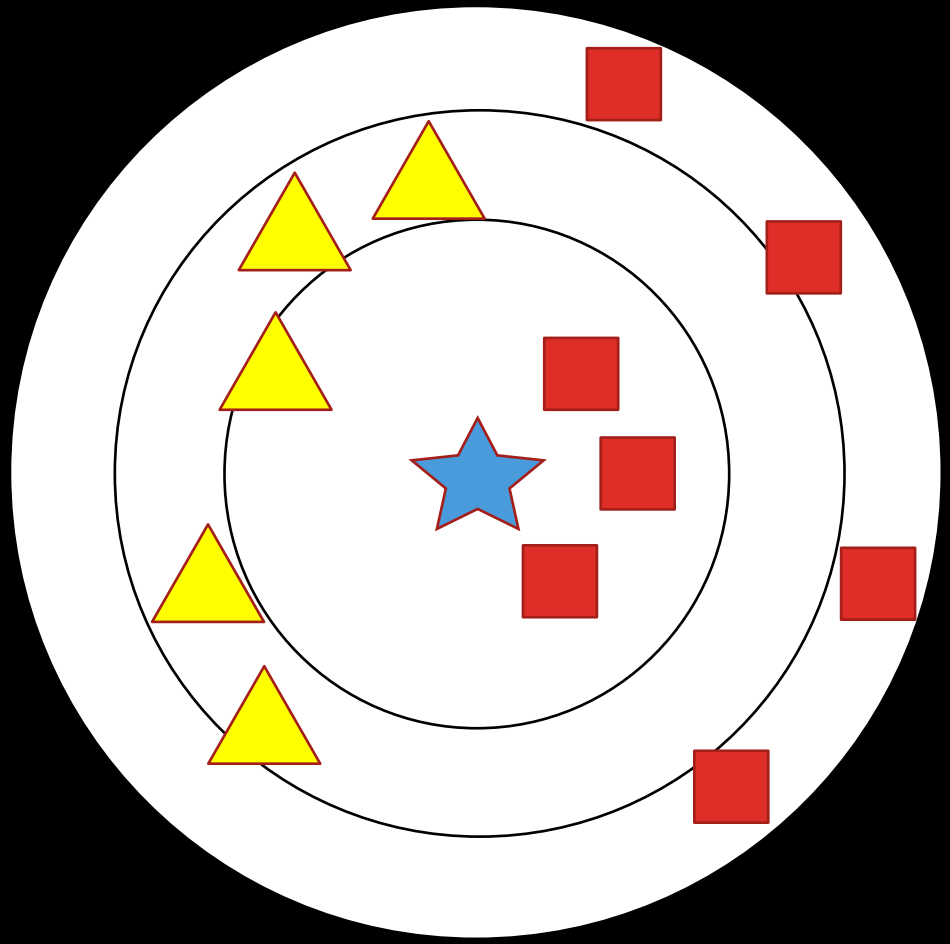


SOME TERMINOLOGY

- Feature
- Feature space
- Prediction
- Hyperparameter
- Hyperparameter space
- Training set
- Test set
- Cost function

CLASSIFICATION VS REGRESSION

K-NEAREST NEIGHBORS

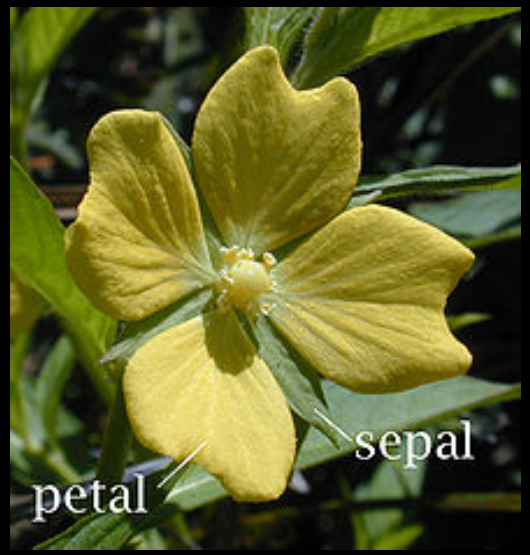


k	Prediction for ★
1	■
3	■
5	■
7	▲
9	▲
11	■

IRIS DATA SET

- Database of 150 flowers
- 3 species of flower:
  - setosa (0)
  - versicolor (1)
  - virginica (2)

- 4 features:
  - Sepal length
  - Sepal width
  - Petal length
  - Petal width

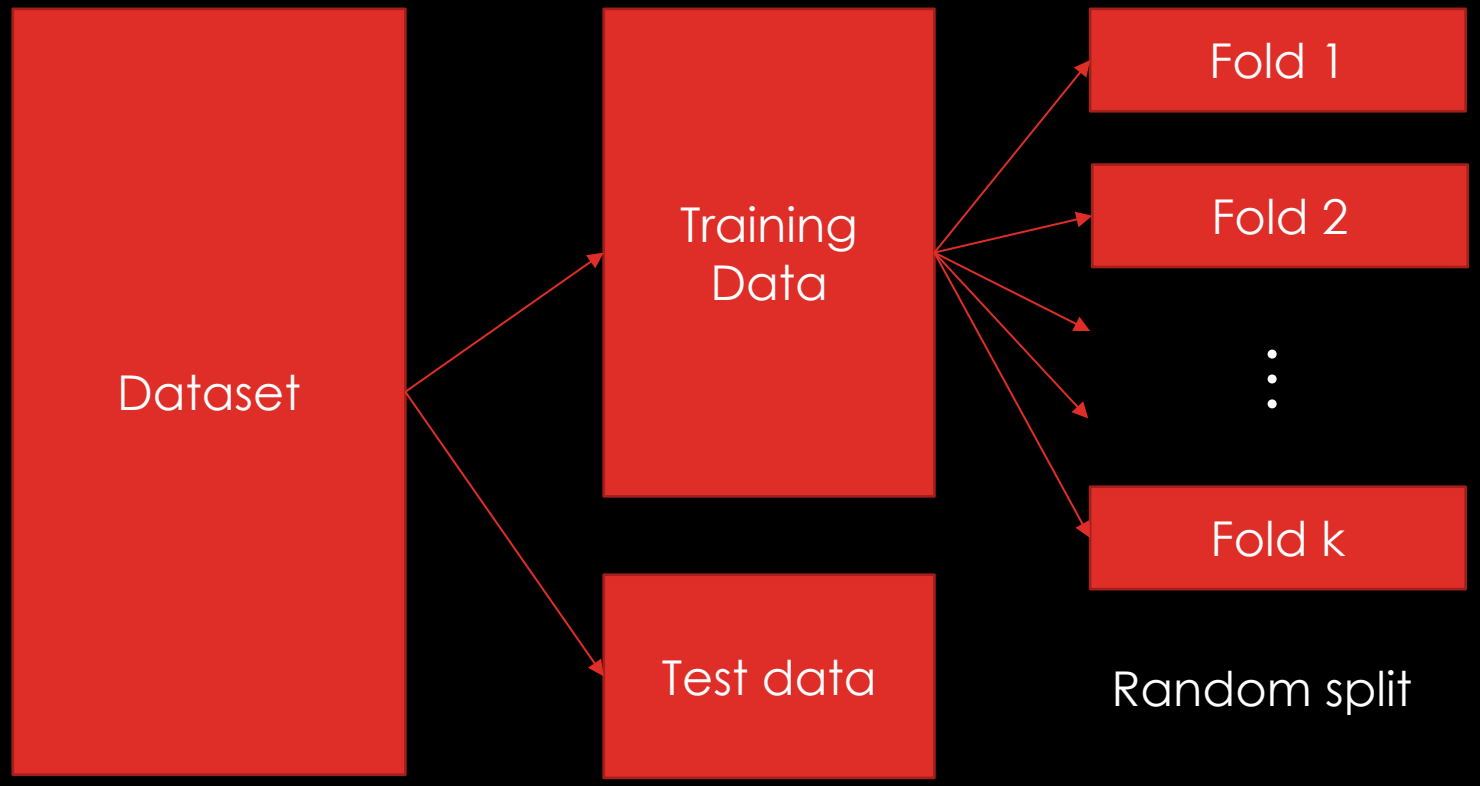


THE IMPORTANCE OF SCALING

- Many ML algorithms are sensitive to the scale of the data.
- KNN is particularly vulnerable to this – sepal widths are much smaller than petal widths, so they will count more than they should!
- To get round this, scale the data in each dimension to have mean 0 and variance 1.

K-FOLD CROSS VALIDATION

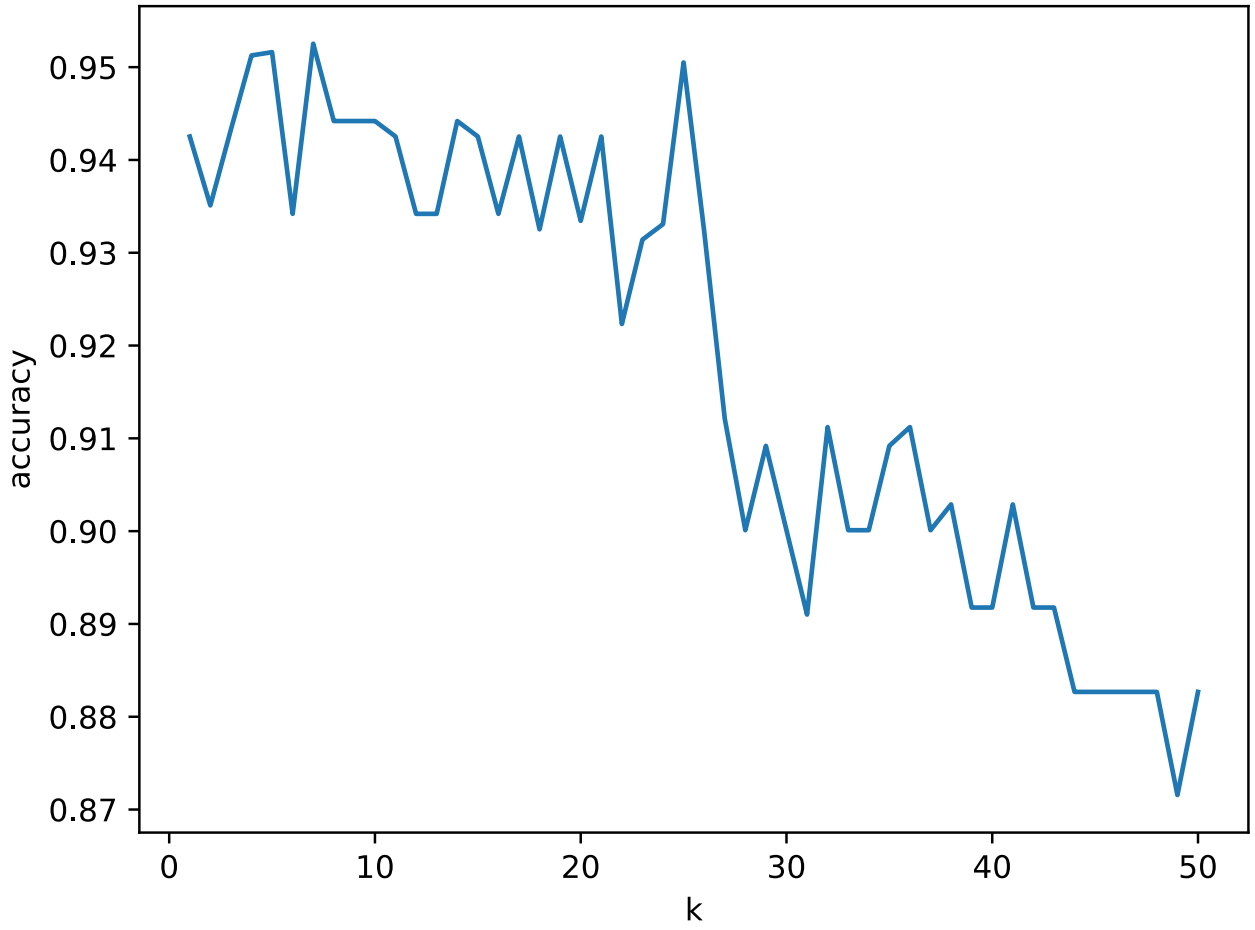
Can we tune hyperparameters to improve the accuracy of our algorithm?

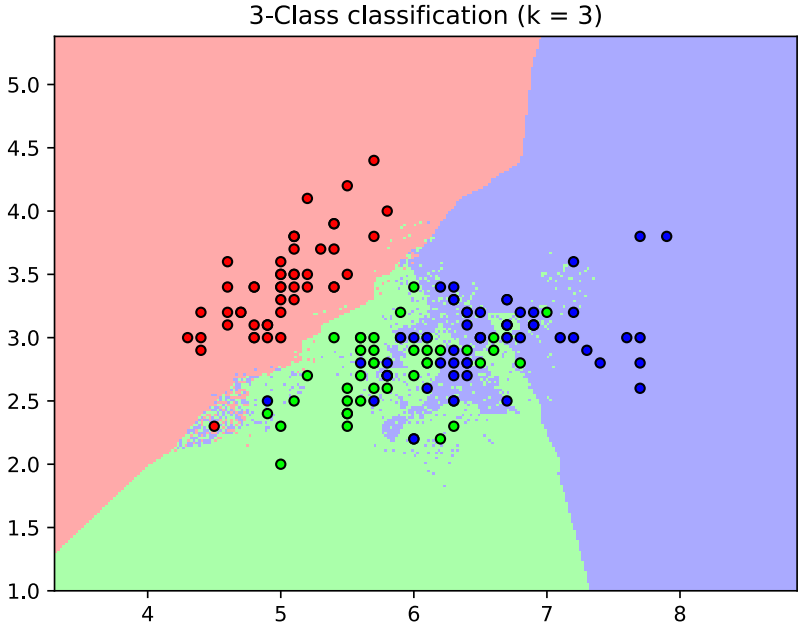


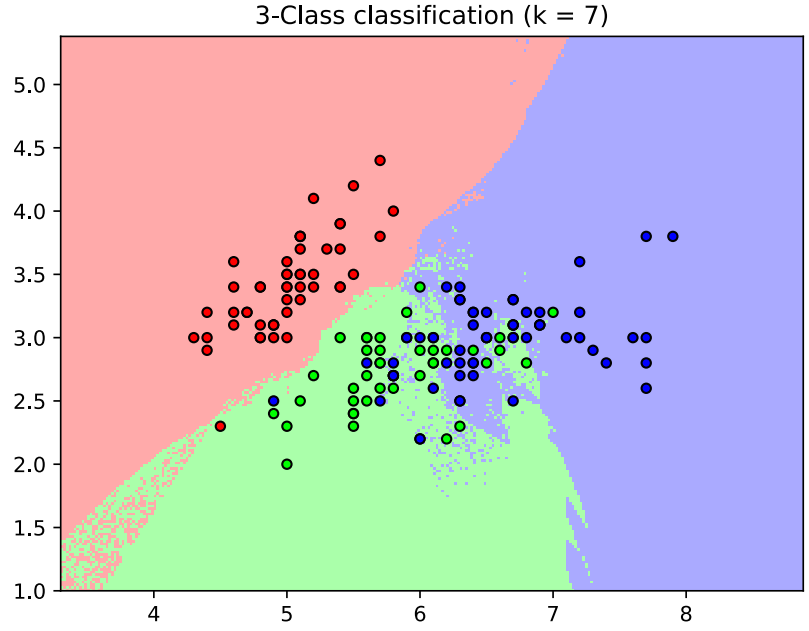

```
for j in 1…k
    use {1,…,k} \ {j} as training set
    evaluate performance on fold j
```

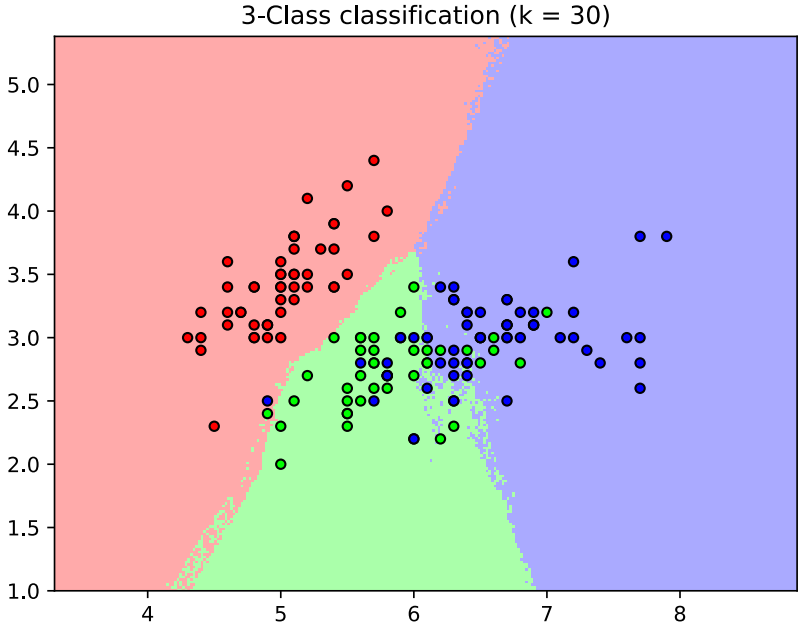
Take average of the k performances to estimate accuracy of the model.

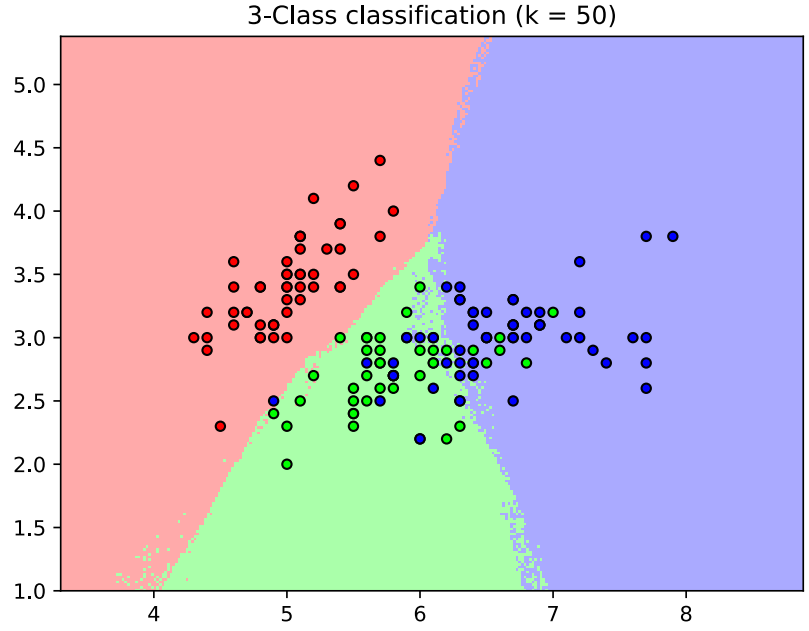
Repeat this process for each model you want to test (generally grid search over hyperparameter space).


0.95
0.94
0.93
0.92
0.91
0.90
0.89
0.88
0.87
accuracy
0
10
20
30
40
50
k


3-Class classification (k = 3)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8


3-Class classification (k = 7)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8


3-Class classification (k = 30)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8


3-Class classification (k = 50)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8

SUMMARY

- KNN is one of the simplest machine learning algorithms, but it can be very accurate
- Think carefully about scaling!
- In general not suitable for very high-dimensional feature spaces
- Lazy-learner, no real training happens (the data **is** the algorithm)

CHALLENGE – BOSTON HOUSE PRICES

- There are **14** attributes in each case of the dataset. They are: CRIM - per capita crime rate by town

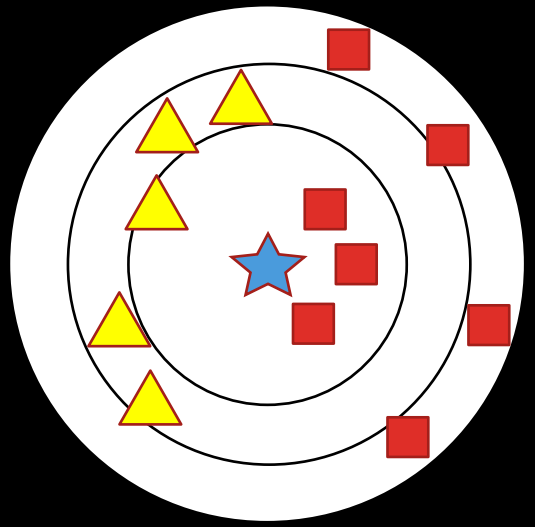
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of black citizens by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

CLASS 2: RANDOM FORESTS

Last Time

- Explored k-Nearest Neighbors algorithm



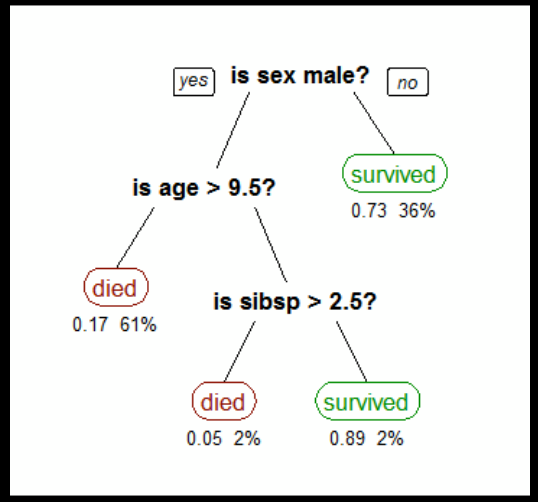
- Works well for low-dimensional feature spaces (flower identification)
- Not so good on high-dimensional problems (Boston Housing Prices)

THE BIAS-VARIANCE TRADE-OFF

- Bias: missing important relations between features and targets (outputs)
- Variance: overfitting – finding relationships that aren't there (modelling noise)
- As you increase complexity of the model,

DECISION TREES

- Building blocks of random forests
- Simple learning algorithms - produces output from several inputs.
- Deep trees are needed to deal with complex problems, but deep trees tend to overfit
- Low bias, high variance
- Is there a way to fix this?



Wikipedia (Stephen Millborrow)

TREE BAGGING

Say we have a training set of n feature sets X ={x_1,…,x_n} and n targets Y = {y_1,…,y_n}

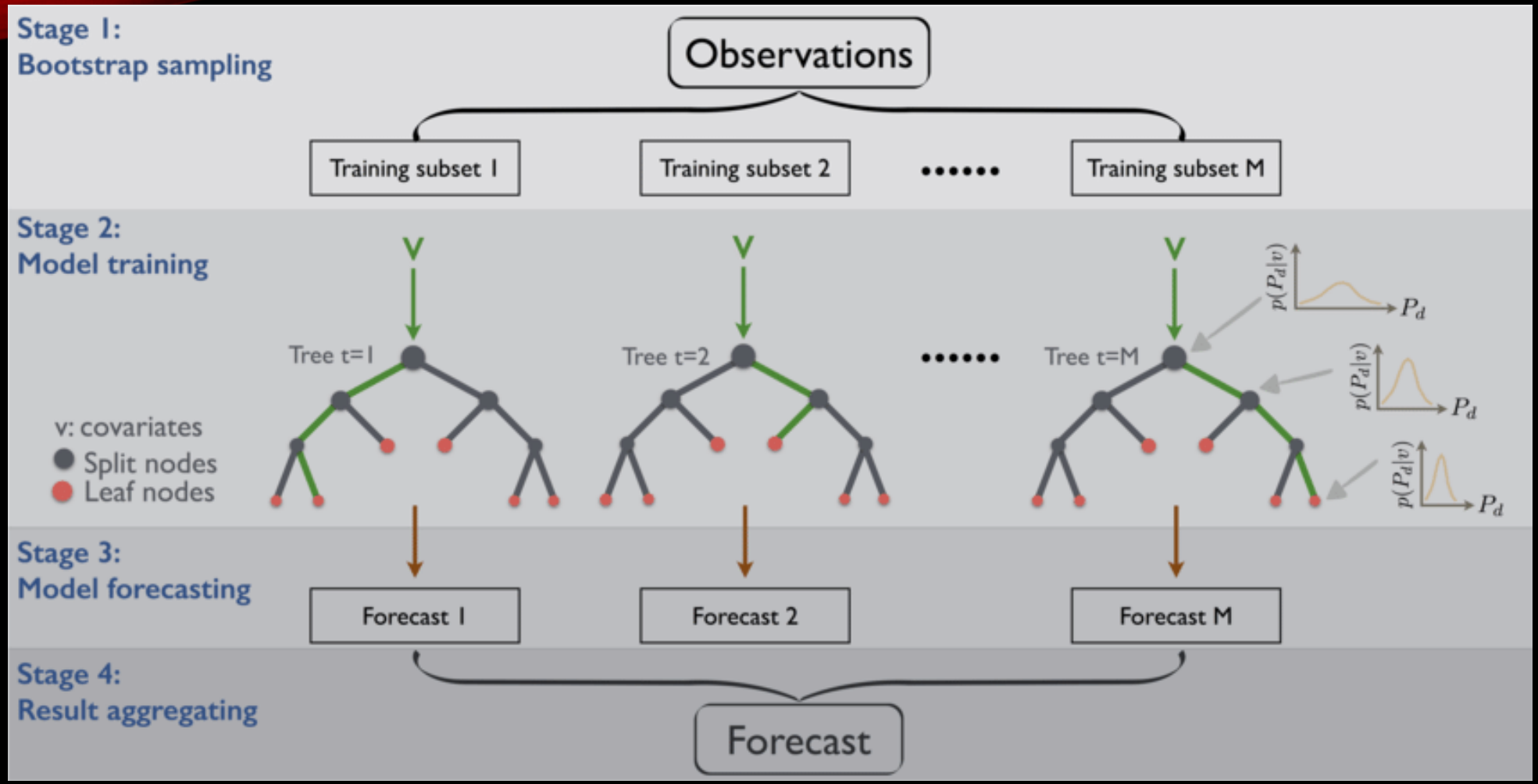
```
for b in 1,…,B:
      sample (with replacement) n training examples from X and
      Y. Call these X_b and Y_b.

      create a decision tree T_b that has been trained on {X_b,Y_b}
```

Classification

Take majority output from all of the trees

Regression

Take an average over all of the trees


Stage 1:
Bootstrap sampling
Observations
Training subset 1
Training subset 2
Training subset M
Stage 2:
Model training
Tree t=1
Tree t=2
Tree t=M
v: covariates
Split nodes
Leaf nodes
Stage 3:
Model forecasting
Forecast 1
Forecast 2
Forecast M
Stage 4:
Result aggregating
Forecast

He et al. 2016

- Why do we boostrap?
  - **Decorrelates** the trees
  - Reduces sensitivity to noise
  - Decreases variance without large increase in bias.

- How do we know how many trees we need?
  - Cross validation!
  - Or... Out-of-bag error

- How do you actually train a tree?

RANDOM FOREST HYPERPARAMETERS

- n_estimators: number of trees in the forest
- max_features: The number of features to consider when looking for the best split
- max_depth: maximum depth of the trees
- Bootstrap: whether to do boostrap sampling or not

FEATURE IMPORTANCE

- Permute the values of each feature among the training data
- Recalculate OOB error
- Repeat over all trees
- Average over all trees, divide by std dev to get importance score

- Tries to extract ‘real’ information from the statistics!

OPTIMISED KNN

BOSTON HOUSE PRICES – TAKE 2

CHALLENGE: BREAST CANCER DATASET

- Download the set with

```
from sklearn.datasets import load_breast_cancer
X, y = load_breast_cancer(return_X_y = True)
```

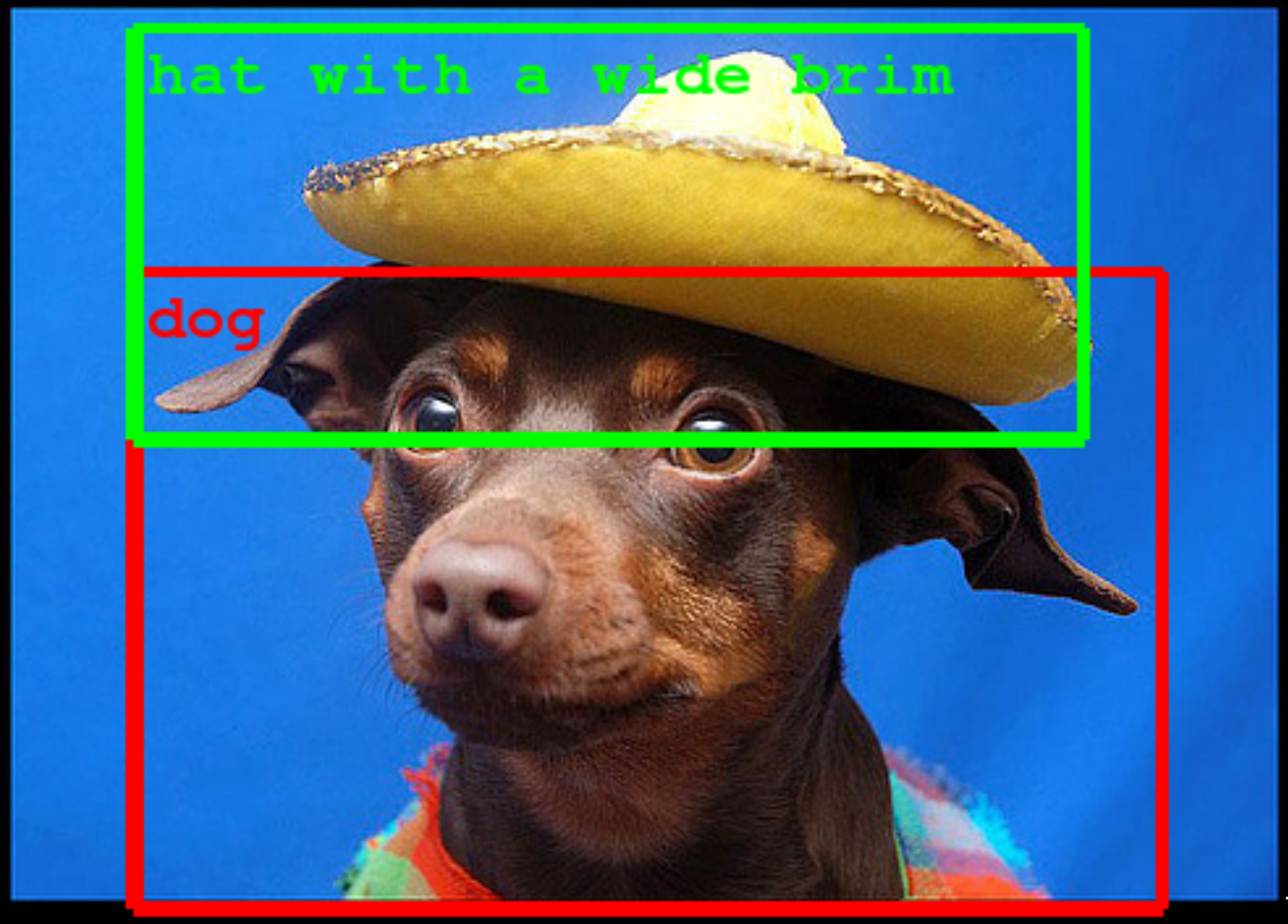
Create a random forest algorithm that predicts whether a tumour is benign (0) or malignant (1) as accurately as possible.

What are the most important features of the tumour?

RANDOM FORESTS - SUMMARY

Random forests are tolerant to feature scaling! – no need to transform data (though it's still good practice)

CLASS 3 – INTRODUCTION TO NEURAL NETWORKS AND DEEP LEARNING



Taken from the Google Blog.

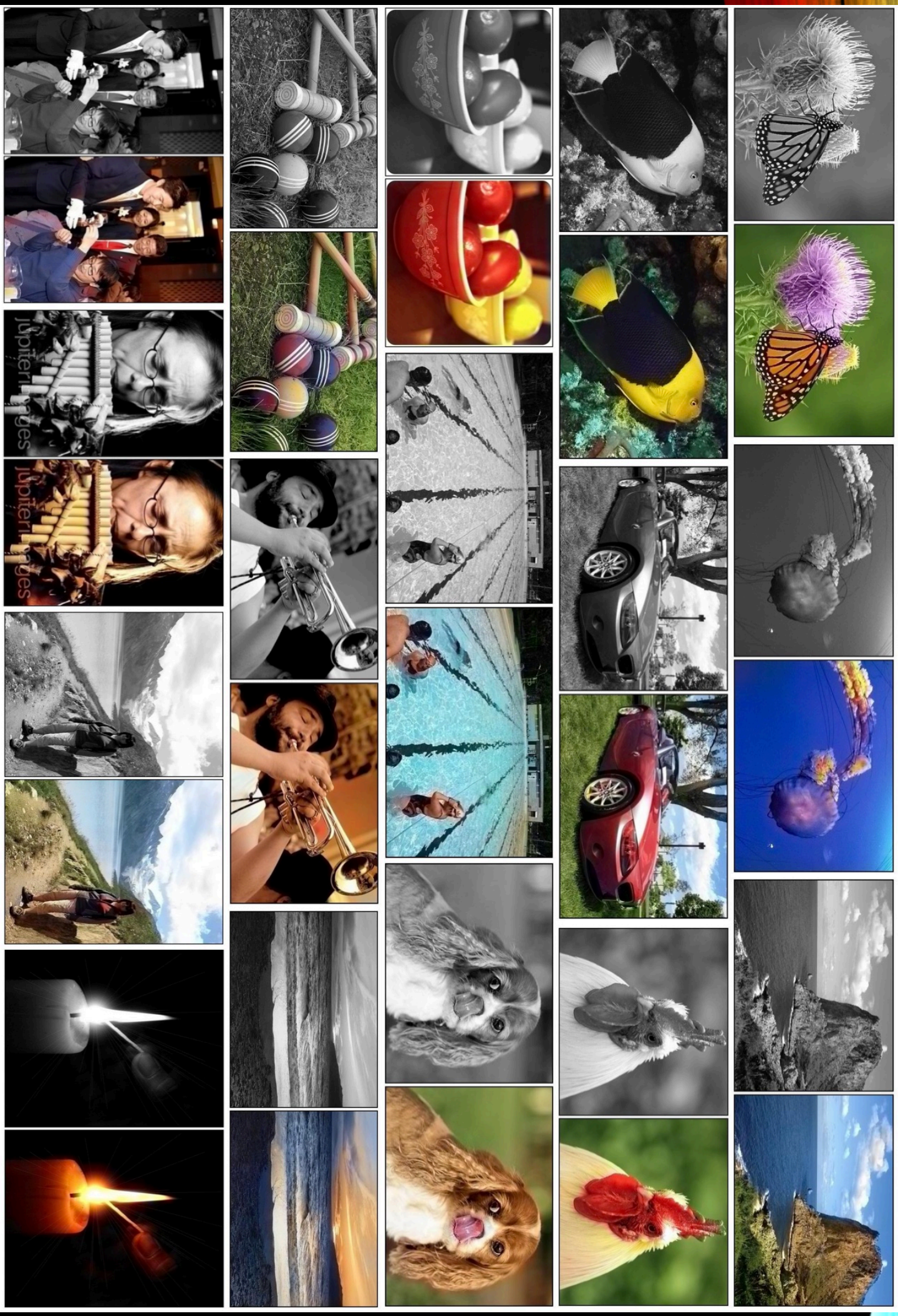


Image taken from Richard Zhang, Phillip Isola and Alexei A. Efros.

WHAT IS A NEURAL NETWORK?

- Collection of **nodes** with connections between them.

- Each connection has:
  - Weight
  - Bias
  - Activation function

A mostly complete chart of

Neural Networks

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

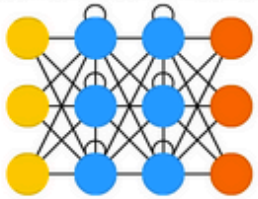
Perceptron (P)

Feed Forward (FF)

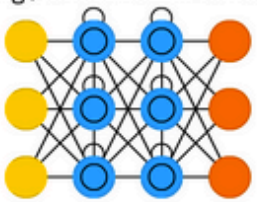
Radial Basis Network (RBF)

Deep Feed Forward (DFF)

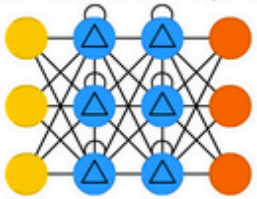
Recurrent Neural Network (RNN)



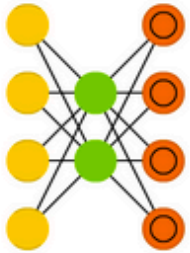
Long / Short Term Memory (LSTM)



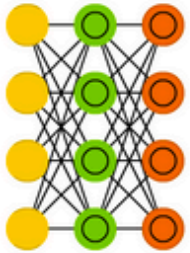
Gated Recurrent Unit (GRU)



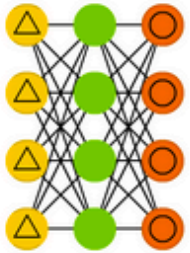
Auto Encoder (AE)



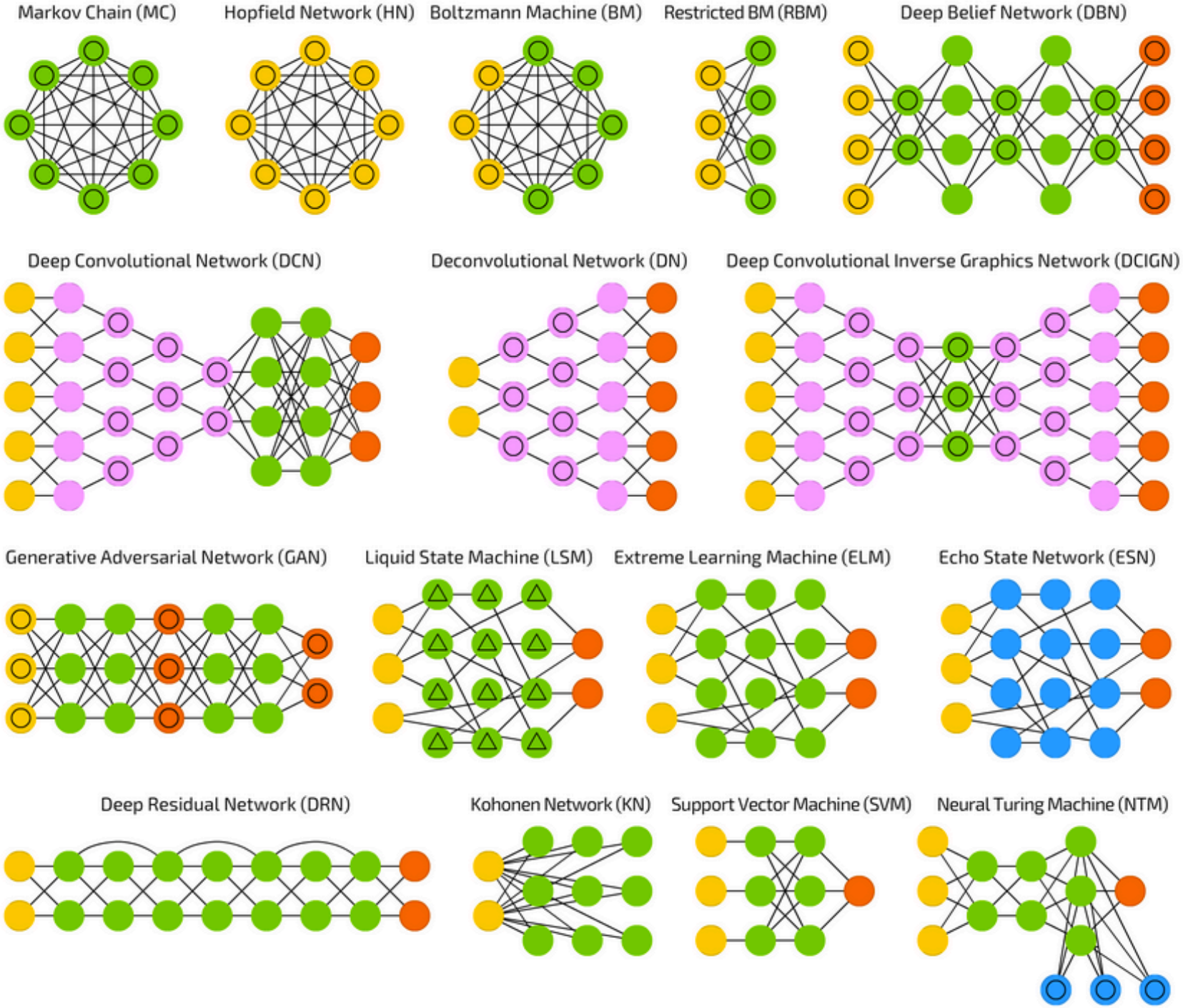
Variational AE (VAE)



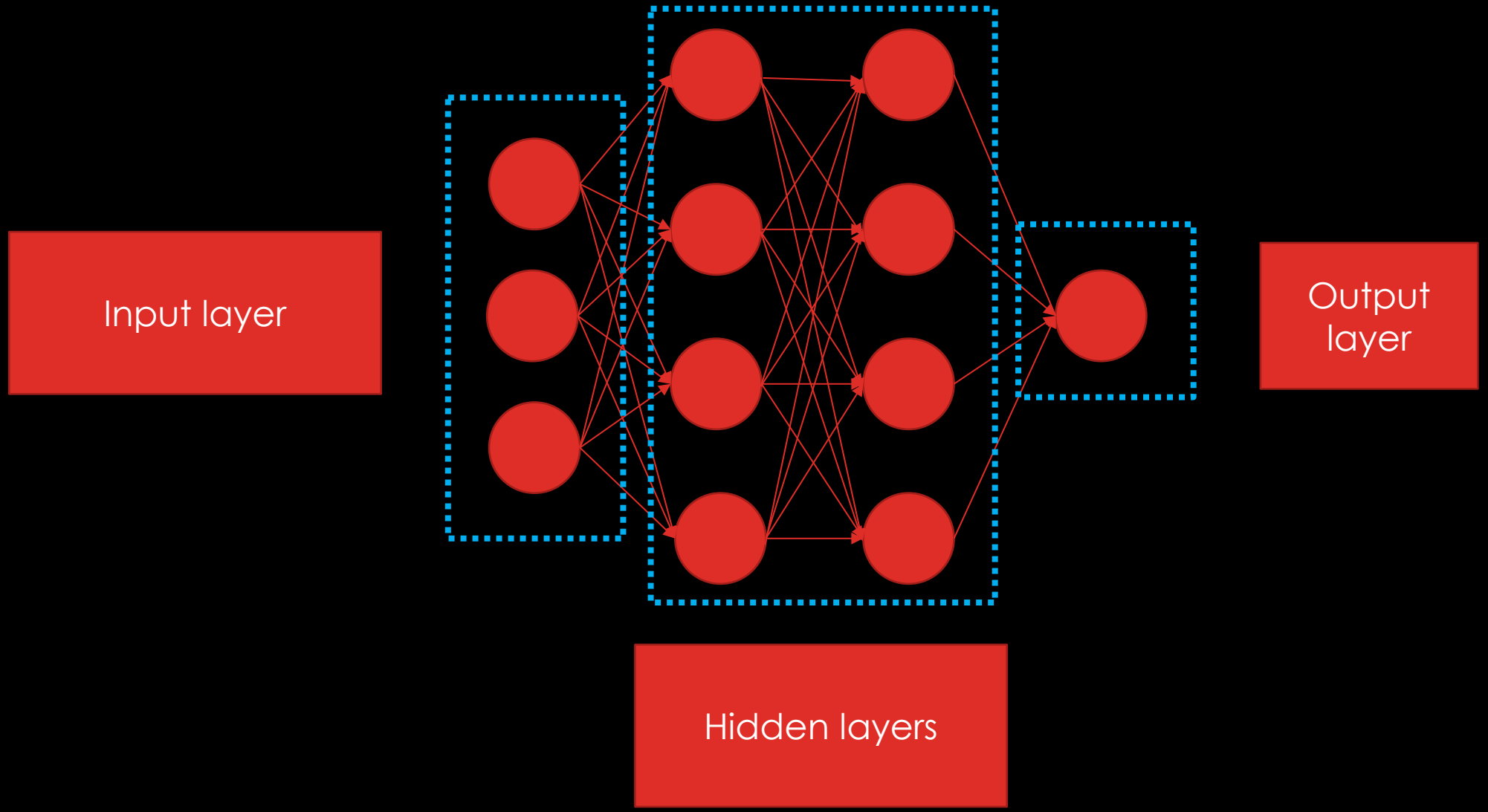
Denoising AE (DAE)

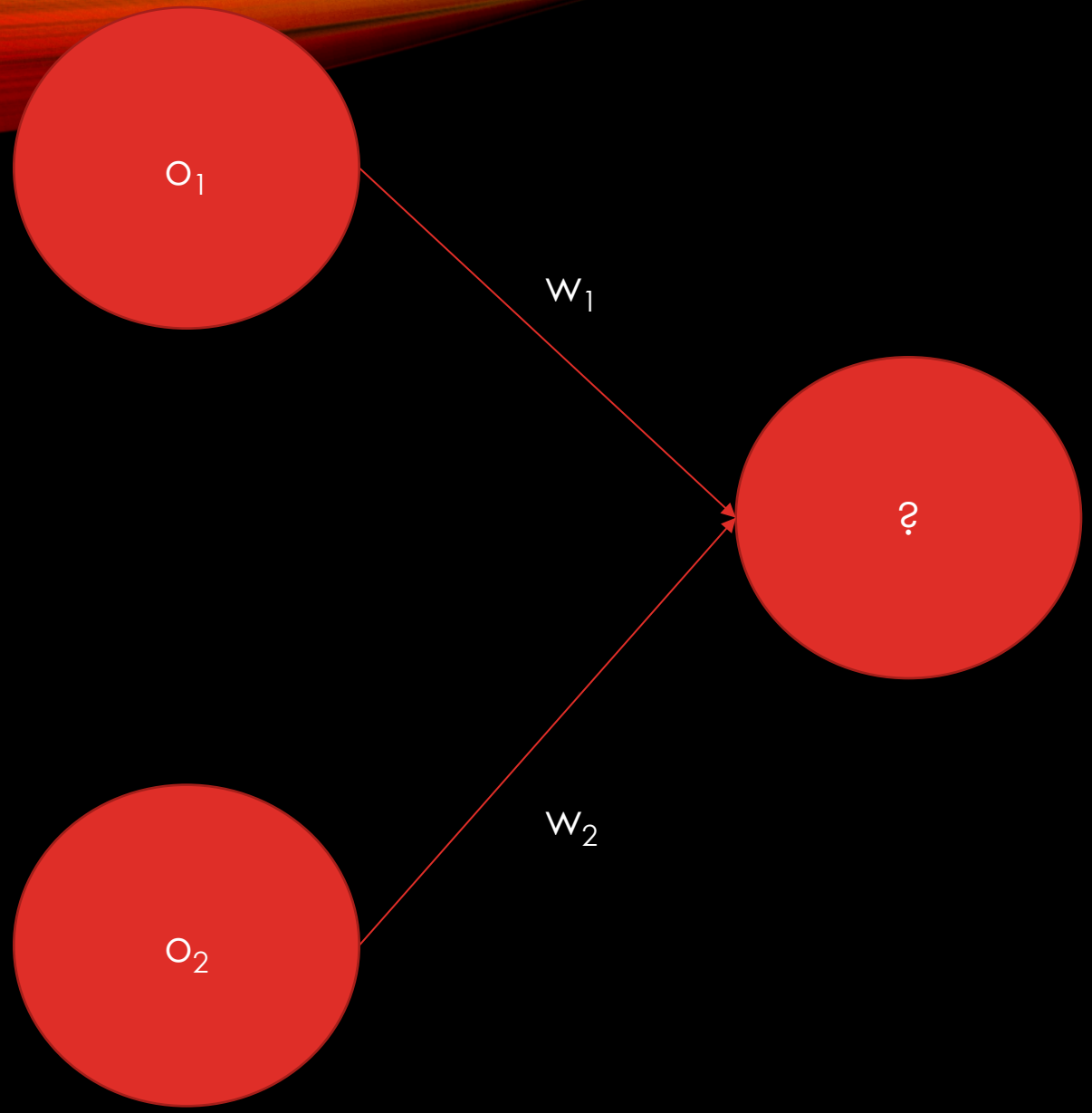


Sparse AE (SAE)


Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

FEED FORWARD NEURAL NETWORKS





$$\text{output} = \varphi(w_1 o_1 + w_2 o_2 + b)$$

ACTIVATION FUNCTIONS

- Introduce non-linearity into the model
- Best option in general: ReLU
  - "Almost linear"
  - Preserves the easy optimisation of linear models while allowing for non-linearity.

- We can build a universal function approximator from ReLU functions (!)

UNIVERSAL APPROXIMATION THEOREM

Formal statement [edit]

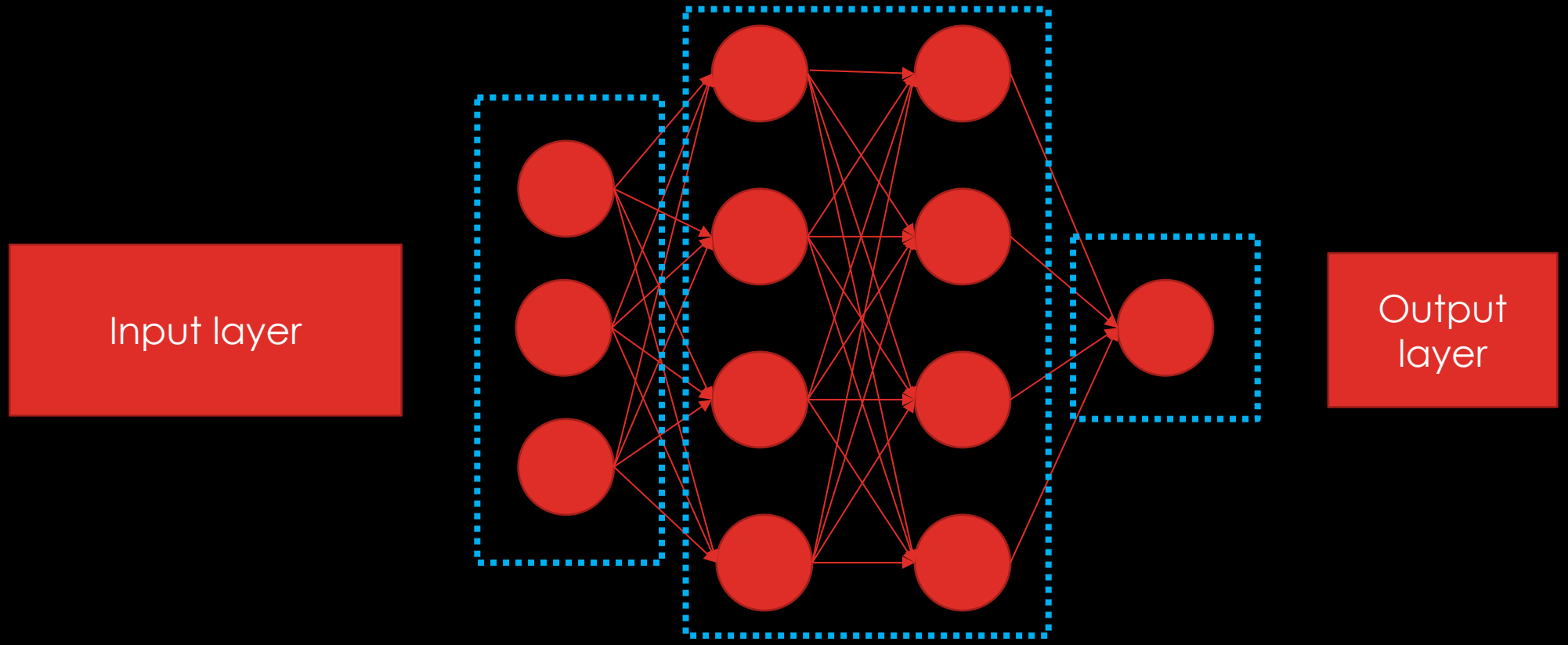
The theorem[2][3][4][5] in mathematical terms:

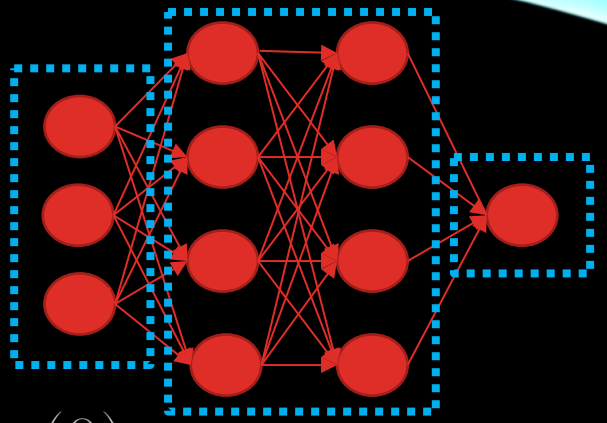
> Let $\varphi(\cdot)$ be a nonconstant, bounded, and continuous function. Let I_m denote the m-dimensional unit hypercube $[0,1]^m$. The space of continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\varepsilon > 0$ and any function $f \in C(I_m)$, there exist an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$, where $i = 1, \cdots, N$, such that we may define:
>
> $$F(x) = \sum_{i=1}^{N} v_i \varphi\left(w_i^T x + b_i\right)$$
>
> as an approximate realization of the function f where f is independent of φ; that is,
>
> $$|F(x) - f(x)| < \varepsilon$$
>
> for all $x \in I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$.

This still holds when replacing I_m with any compact subset of $\mathbb{R}^m$.

https://en.wikipedia.org/wiki/Universal_approximation_theorem

FEED FORWARD NEURAL NETWORKS





More mathematically:

- Write layers as vectors: $\vec{y}^{(0)}, \ldots, \vec{y}^{(n)}$ - we only know $\vec{y}^{(0)}$ to begin with
- Write biases as vectors: $\vec{b}^{(0)}, \ldots, \vec{b}^{(n-1)}$
- Write weights as **matrices:** $W^{(0)}, \ldots, W^{(n-1)}$

- Calculate the node values iteratively:

$$\vec{y}^{(i+1)} = \varphi\left(W^{(i)}\vec{y}^{(i)} + \vec{b}^{(i)}\right)$$

TRAINING - BACKPROPAGATION

HOW DO WE PREVENT OVERFITTING?

- Limit number of epochs
- Dropout: set a fraction of your nodes to zero output
- Reduce complexity of your architecture
- Get more data!

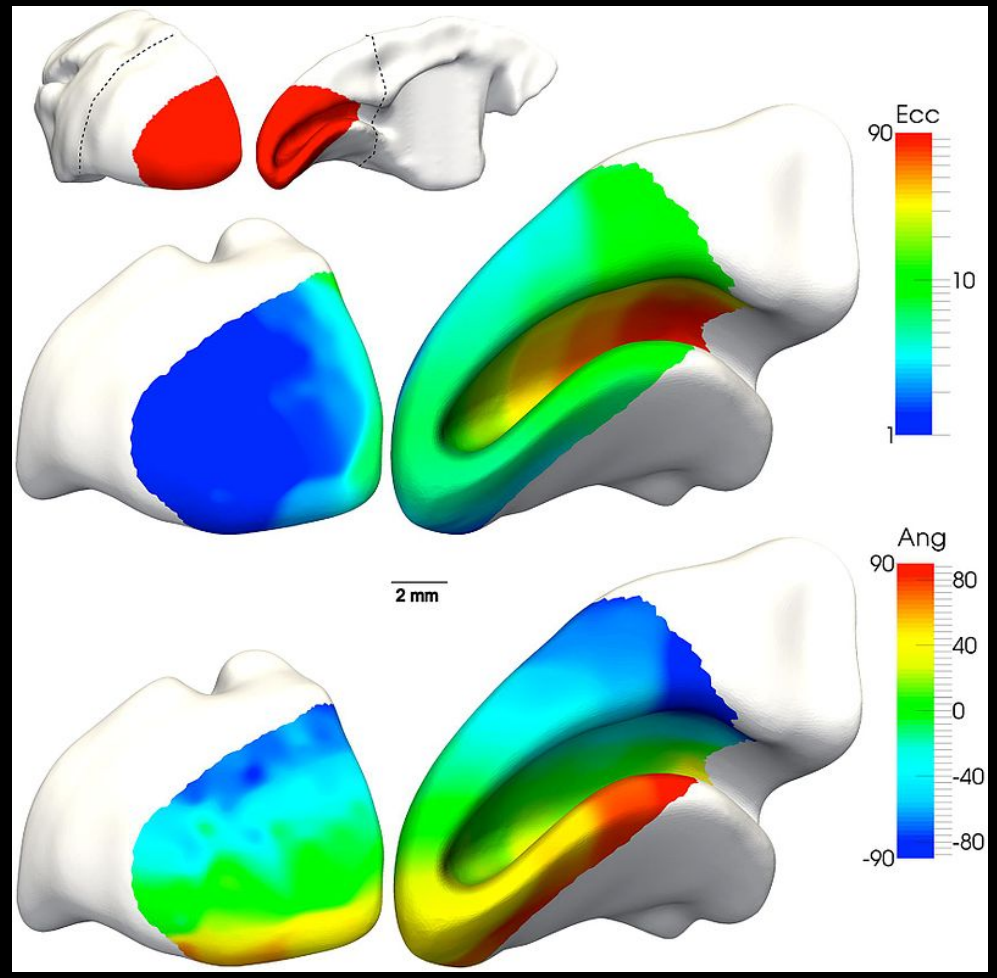
GENERAL GOOD PRACTICE

- How is your data going to arrive at the neural network? What kind of preprocessing do you need to do?
- Look at popular recent architectures – what works well?
- Think about the problem you're trying to solve: is a feedforward network really the best option for image classification?
- Beware of overfitting!
- Think about generalisability and edge cases
- Always train on a GPU if you have one available

NEXT TIME:

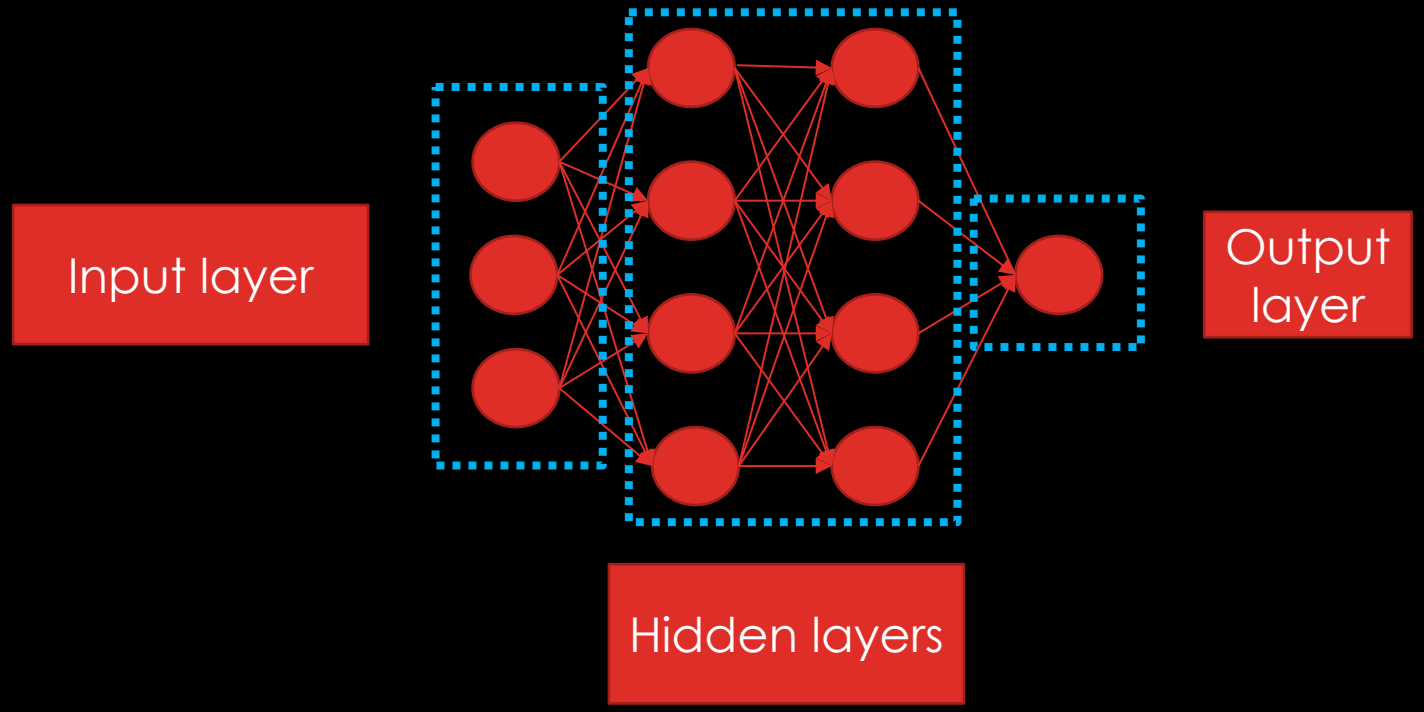
- Convolutional Neural Networks
- Derivative-free optimisation
- Natural Language Processing
- Backpropagation in depth
- Bayesian methods
- …?

CHAPTER 4: CONVOLUTIONAL NEURAL NETWORKS



Solomon SG and Rosa MGP (2014) *A simpler primate brain: the visual system of the marmoset monkey.*

LAST TIME: FULLY CONNECTED FEEDFORWARD NEURAL NETS



- Takes in a vector
- Outputs a vector (or a scalar)

- Not great for images – dimensionality problem

STRUCTURE OF AN IMAGE

```
array([[ 63,  45,  43, ..., 108, 102, 103],
       [ 20,   0,   0, ...,  55,  50,  57],
       [ 21,   0,   8, ...,  50,  50,  42],
       ...,
                                    7,  20],
array([[ 62,  46,  48, ..., 132, 125, 124],   34,  34],
       [ 20,   0,   8, ...,  88,  83,  87],   84,  72]]
       [ 24,   7,  27, ...,  84,  84,  73],
       ...,
       [170, 153, 161, ..., 133,  31,  34],
                            148,  62,  53],
                            184, 118,  92]]
array([[ 59,  43,  50, ..., 158, 152, 148],
       [ 16,   0,  18, ..., 123, 119, 122],
       [ 25,  16,  49, ..., 118, 120, 109],
       ...,
       [208, 201, 198, ..., 160,  56,  53],
       [180, 173, 186, ..., 184,  97,  83],
       [177, 168, 179, ..., 216, 151, 123]],
```

https://www.cs.toronto.edu/~kriz/cifar.html

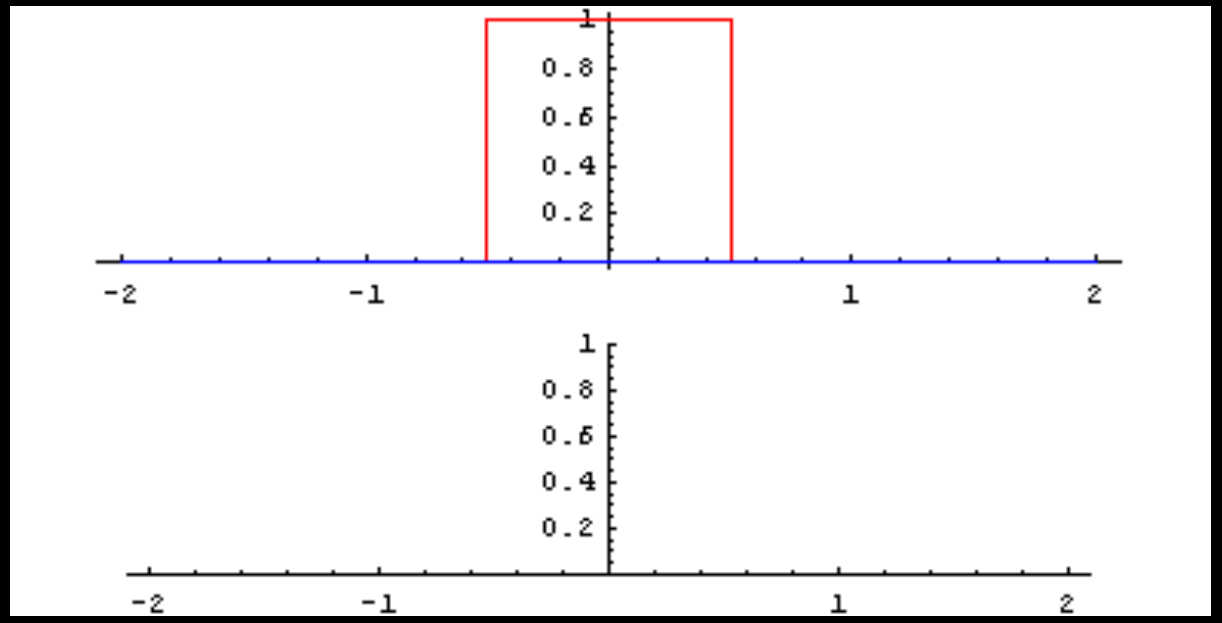
CONVOLUTIONAL NEURAL NETWORKS

- Specifically designed for image recognition
- Takes in multi-dimensional arrays (tensors) rather than just vectors
- Cares about **neighbors** in inputs, rather than treating each feature completely separately.

CONVOLUTION

$$(x * y)(t) = \sum_{t'=-\infty}^{\infty} x(t)y(t - t')$$

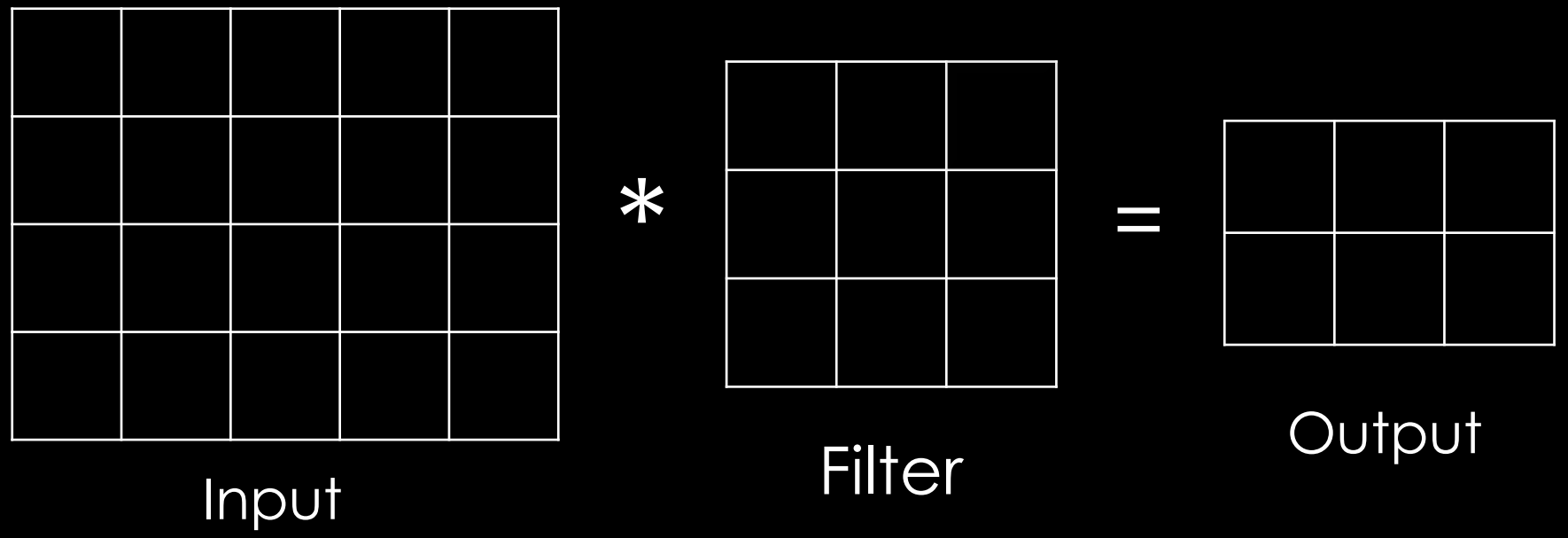
- x: input
- y: filter



https://giphy.com/gifs/processing-signal-kcchao-wunmqxMZ8XtsY

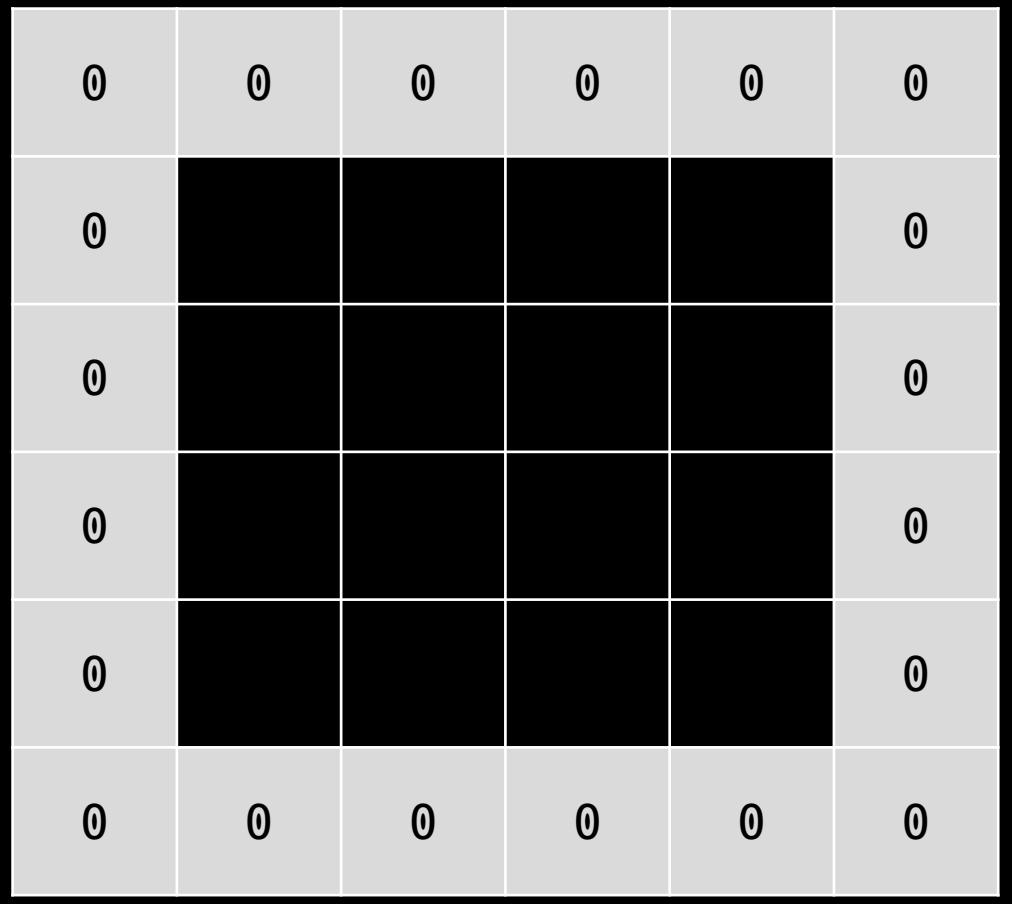
CONVOLUTION OF ARRAYS

$$(A * B)(i, j) = \sum_m \sum_n A(m, n) B(i - m, j - n)$$



https://www.deeplearningbook.org/

ZERO PADDING



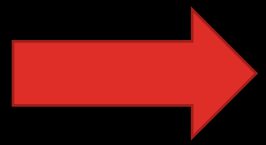
Avoids reduction in array size with each convolution

POOLING

- Cluster of neuron outputs in one layer becomes a single output in the next layer.

- Reduces risk of overfitting

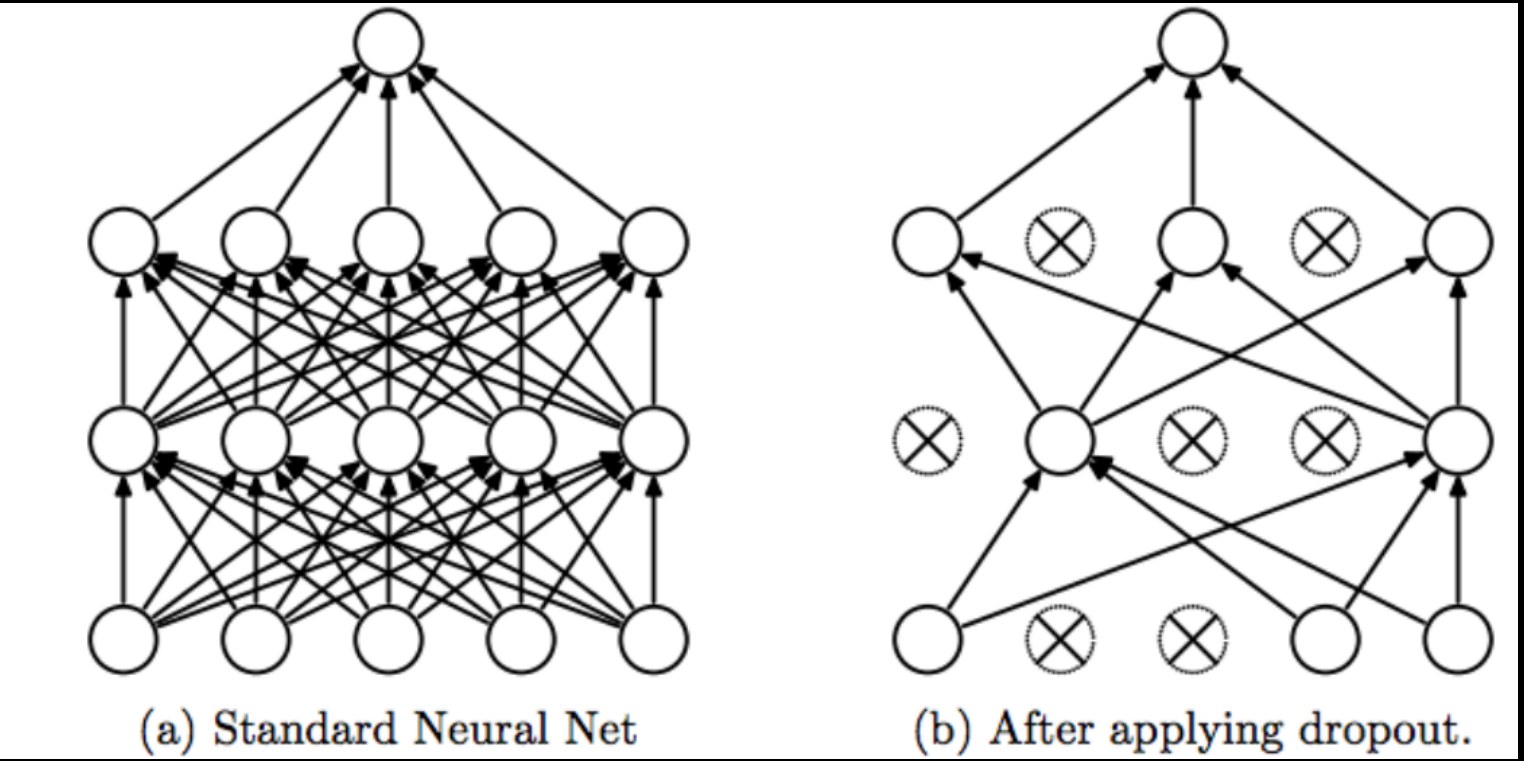
4	2	7	1
3	8	2	5
9	2	2	4
10	2	3	6



8	7
10	6

Max pooling

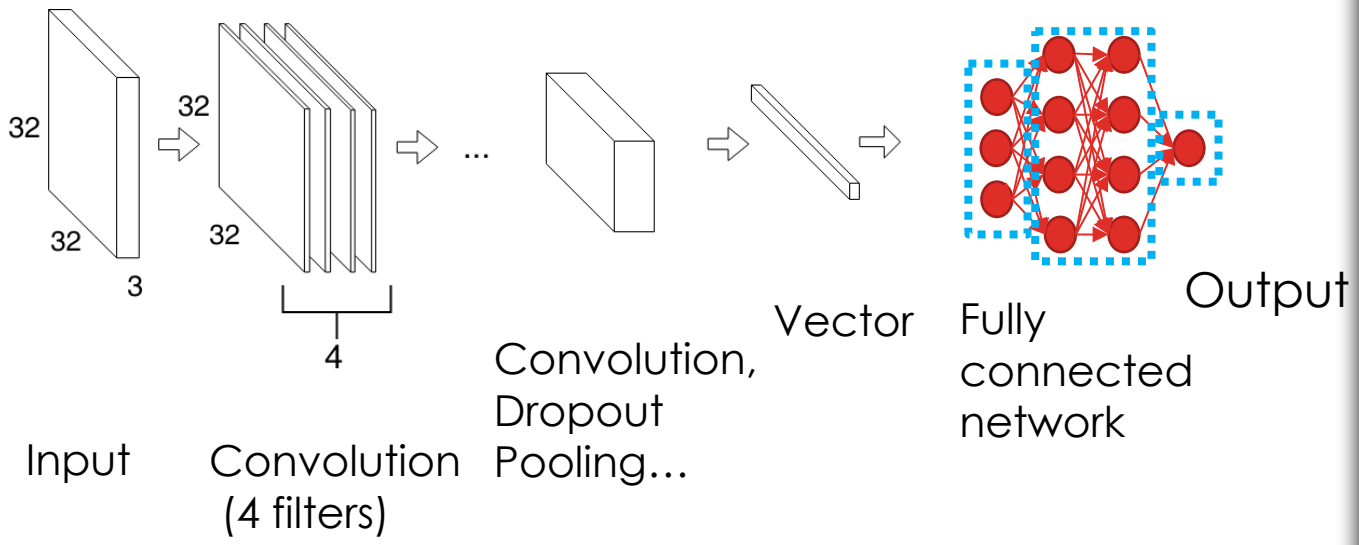
DROPOUT



- In a given layer, set a proportion p of the neurons equal to zero.
- General rule of thumb is p ~ 0.2, but you can cross-validate this.
- Reduces overfitting – fewer parameters to train on
- Stops the network caring too much about some neurons.

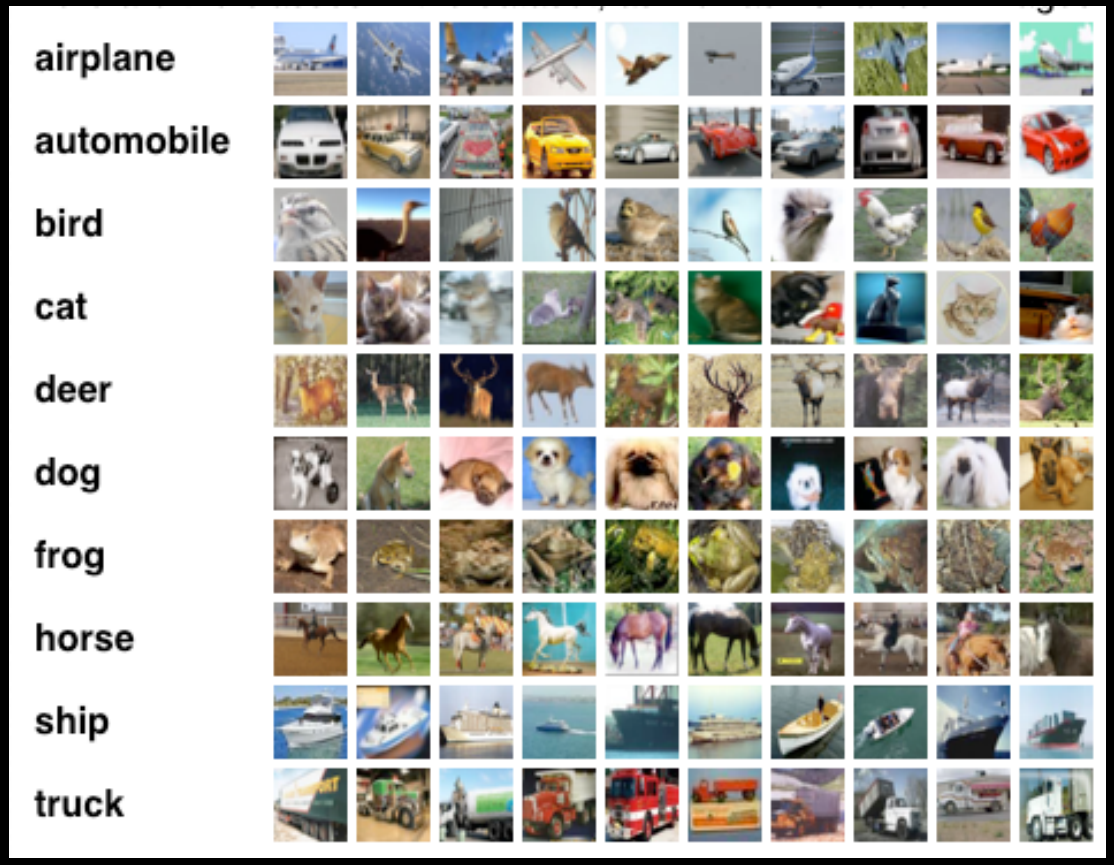
CNN ARCHITECTURE

- CNNs can be extremely deep
- Deeper networks can pick up more complex features in the image
- Remember to zero pad
- As you go deeper, increase the number of filters in the each convolution layer



CIFAR10 DATASET

- 60,000 32x32 color (RGB) images
- 10 categories



https://www.cs.toronto.edu/~kriz/cifar.html