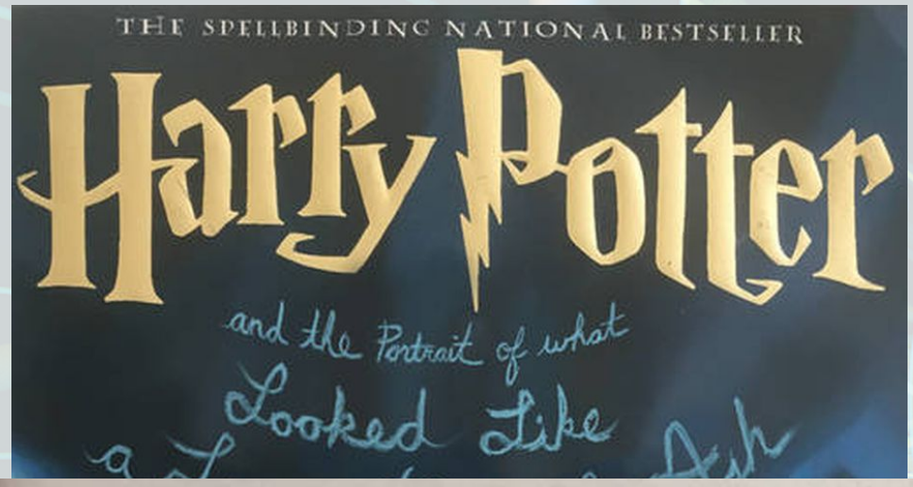
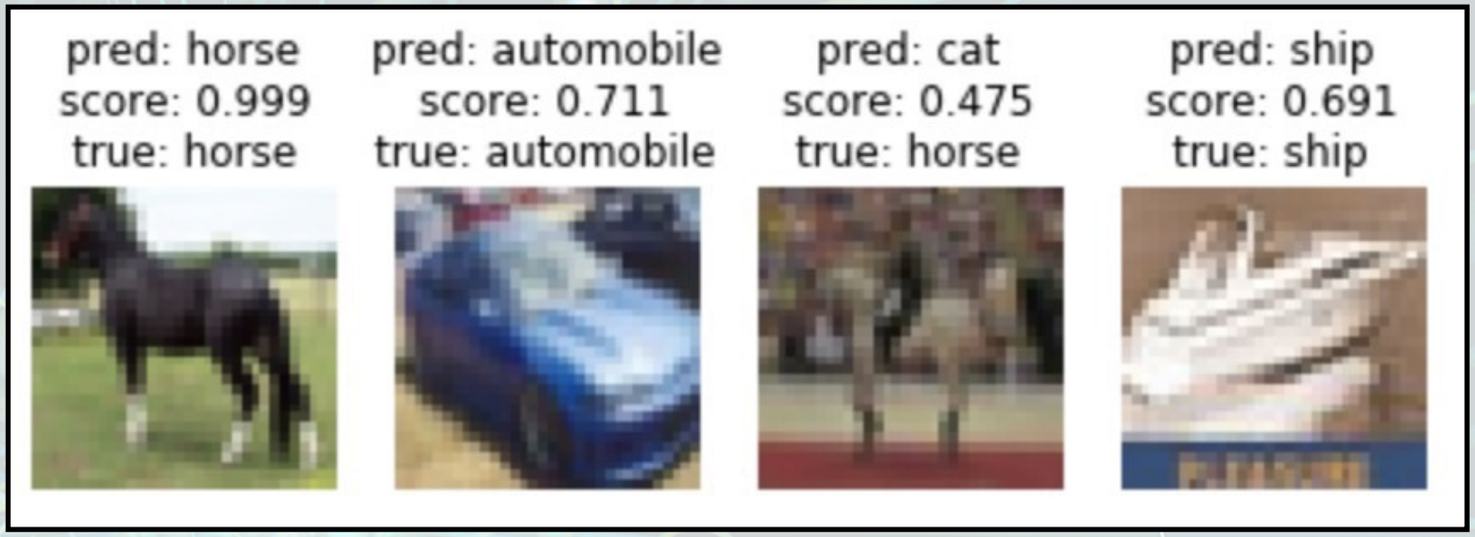
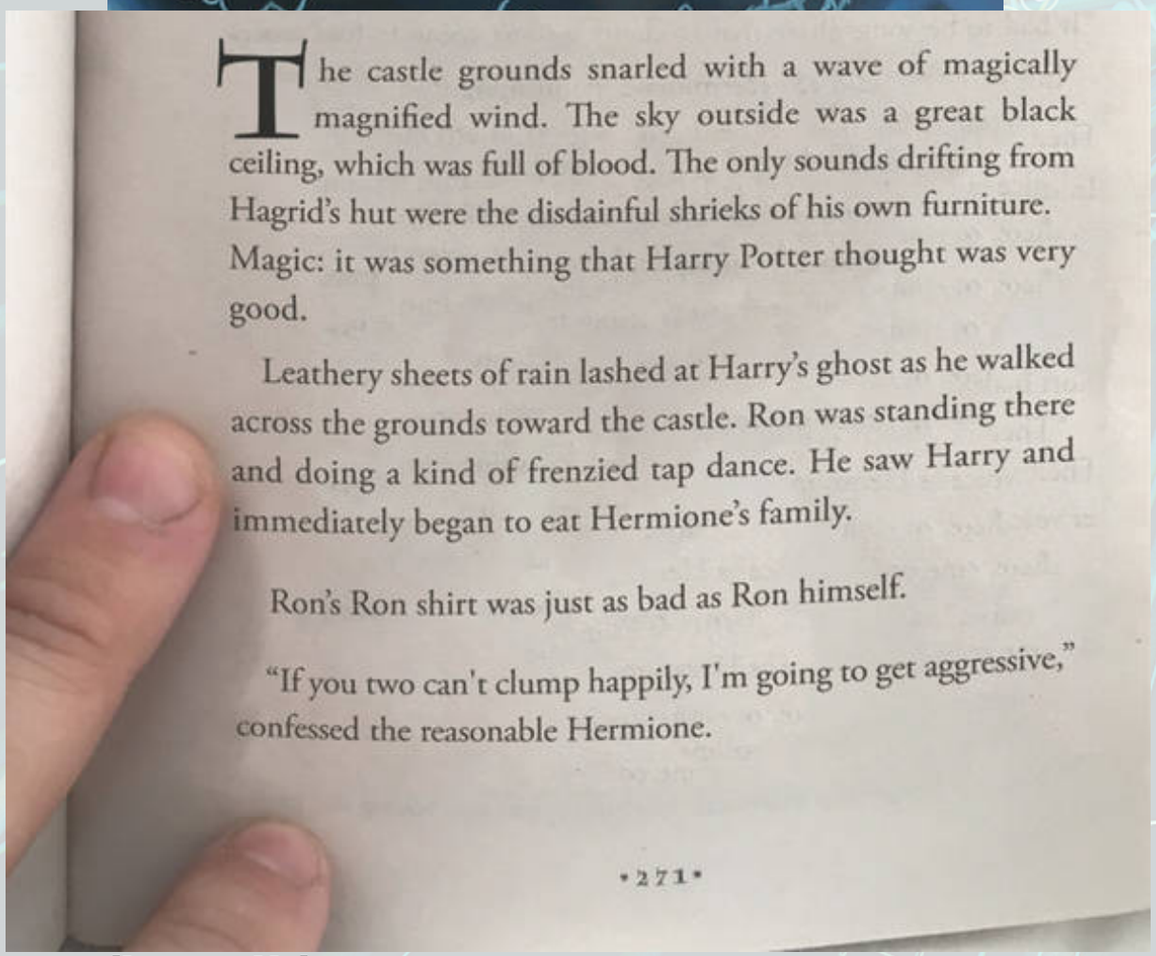
PRA3024 - Introduction to Machine Learning

Cool stuff!






The castle grounds snarled with a wave of magically magnified wind. The sky outside was a great black ceiling, which was full of blood. The only sounds drifting from Hagrid's hut were the disdainful shrieks of his own furniture. Magic: it was something that Harry Potter thought was very good.

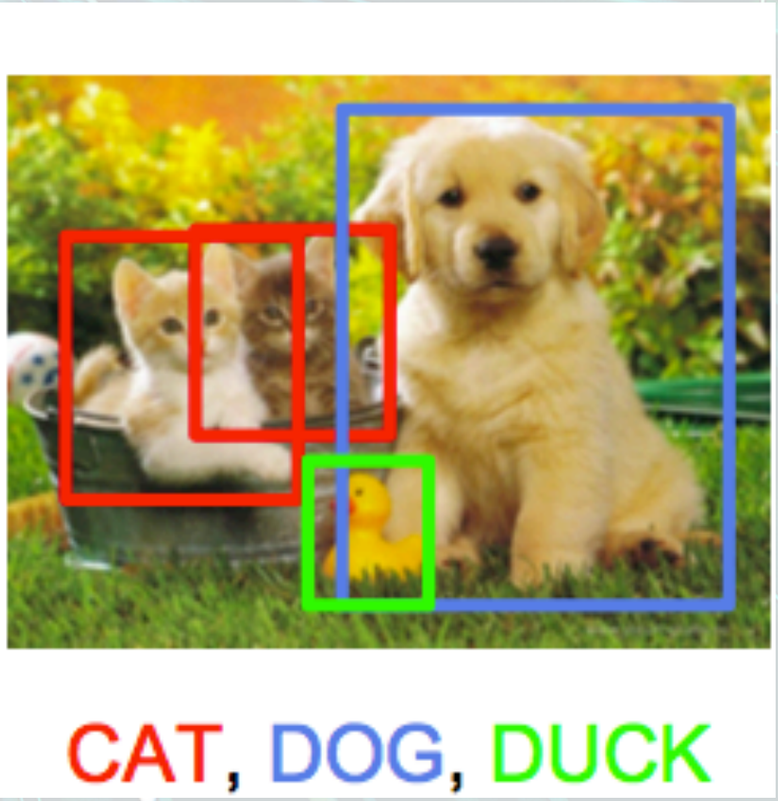
Leathery sheets of rain lashed at Harry's ghost as he walked across the grounds toward the castle. Ron was standing there and doing a kind of frenzied tap dance. He saw Harry and immediately began to eat Hermione's family.

Ron's Ron shirt was just as bad as Ron himself.

"If you two can't clump happily, I'm going to get aggressive," confessed the reasonable Hermione.

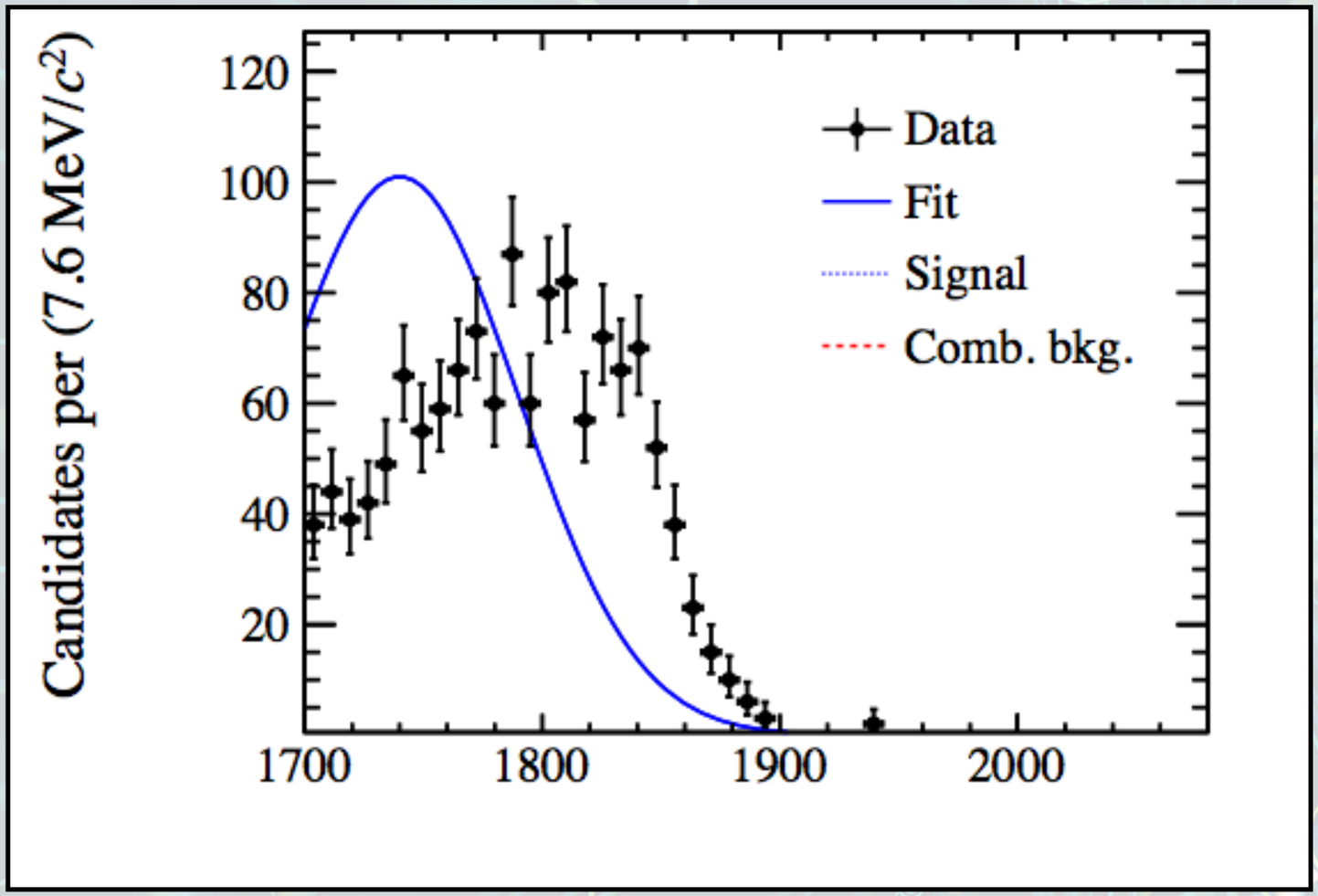
•271•

(botnik)



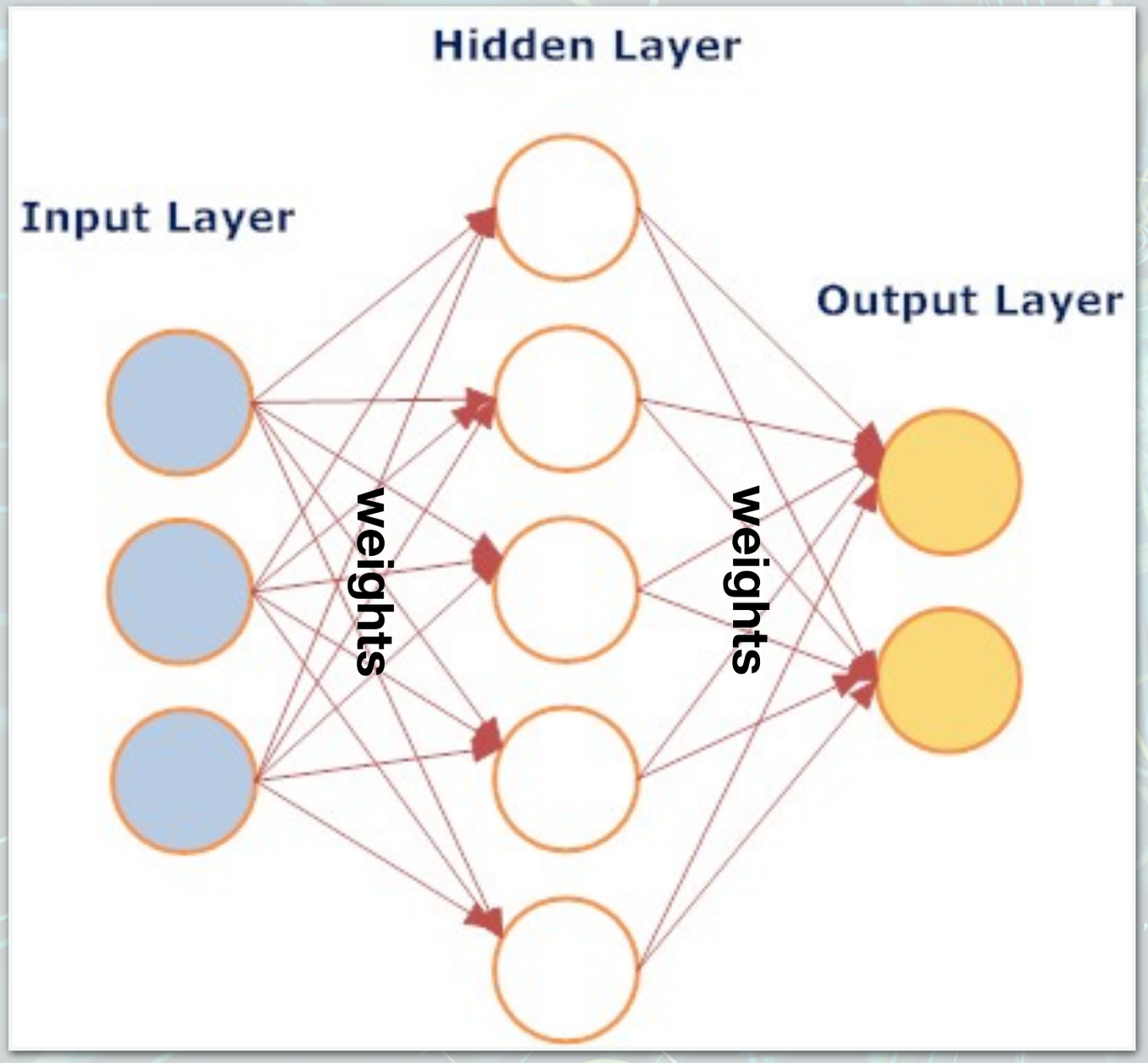
Tensorflow object detection API:
http://jacco.science/public/VID_20190623_214122.mp4

‘Learning’?



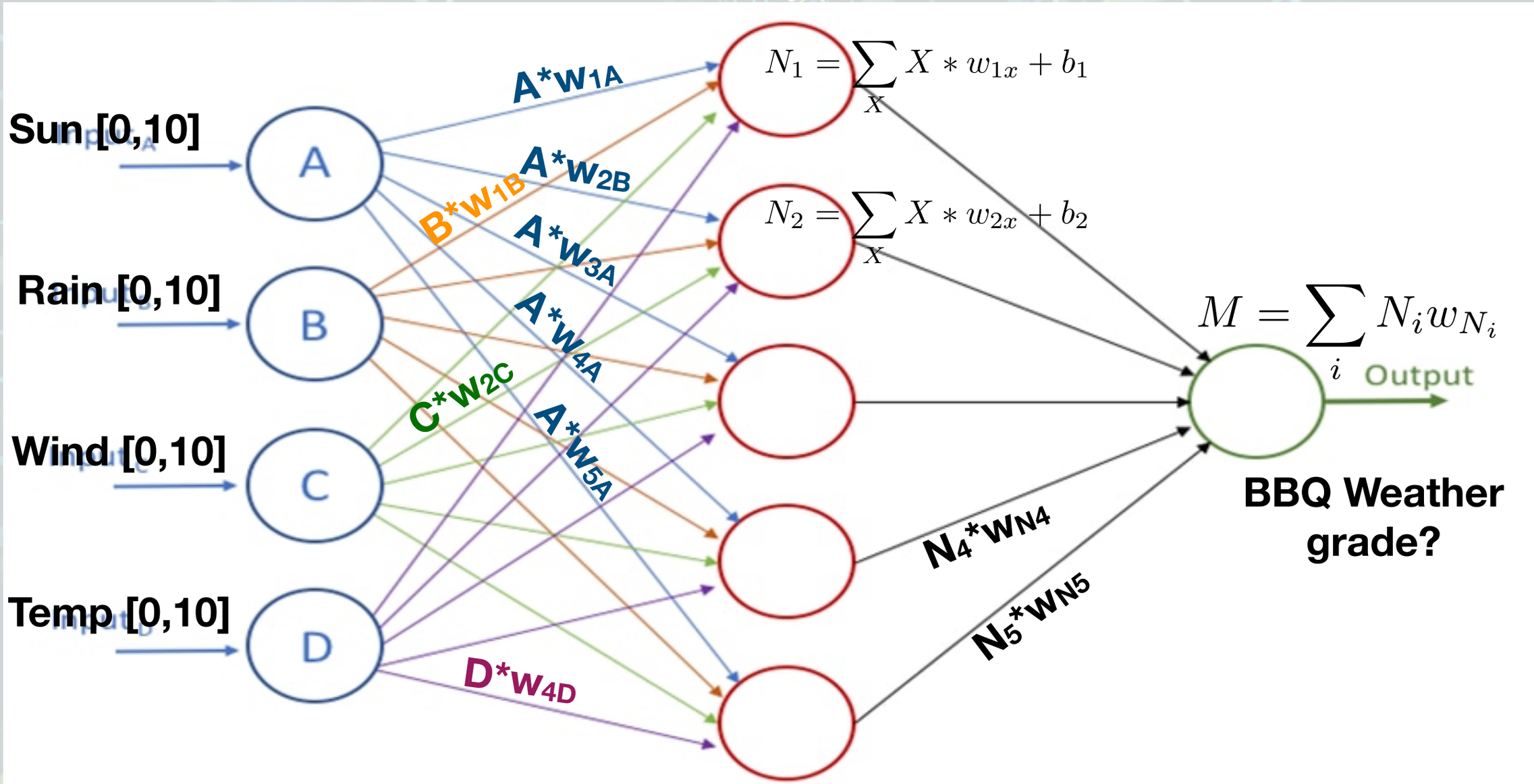
- data -> 2D points
- model -> gaussian function with free mean, width
- ‘goodness of fit’ -> Chi2 = SUM(point-line)^2
- improve goodness -> ‘Fit’ (Minuit)
  —> changes model parameters to reduce chi2

‘Learning’?



- data
  N input vars per ‘entry’
- model
  Neural Network
  with number of weights
- ‘goodness of fit’
  Loss function
- improve goodness
  Use an optimizer
  (SGD, Adam)
  —> changes
  weights to reduce Loss

Neural Network



- A neural network is ‘just a function’, with parameters w
- The output M is ‘just a number’, that we can normalise to e.g. [0,10]
- Usually a simple ‘activation function’ is used at every node, $N_1 = f(N_1)$

Training

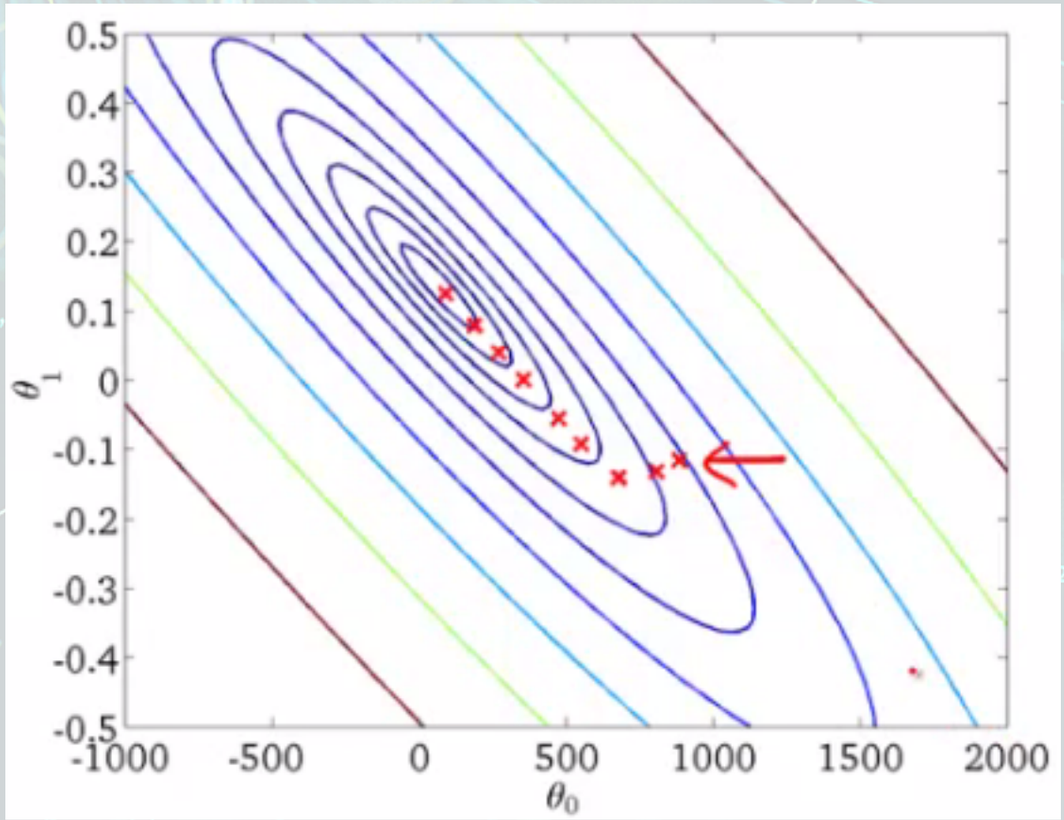
- For supervised learning, we have data with both input AND output label.
  e.g. {sun : 9, rain : 2, wind : 3, temp : 8} —> label ‘L’ = 9

- We can then compare the output of the network with the label
  (initially, w are random —> output is random, e.g. M = 5

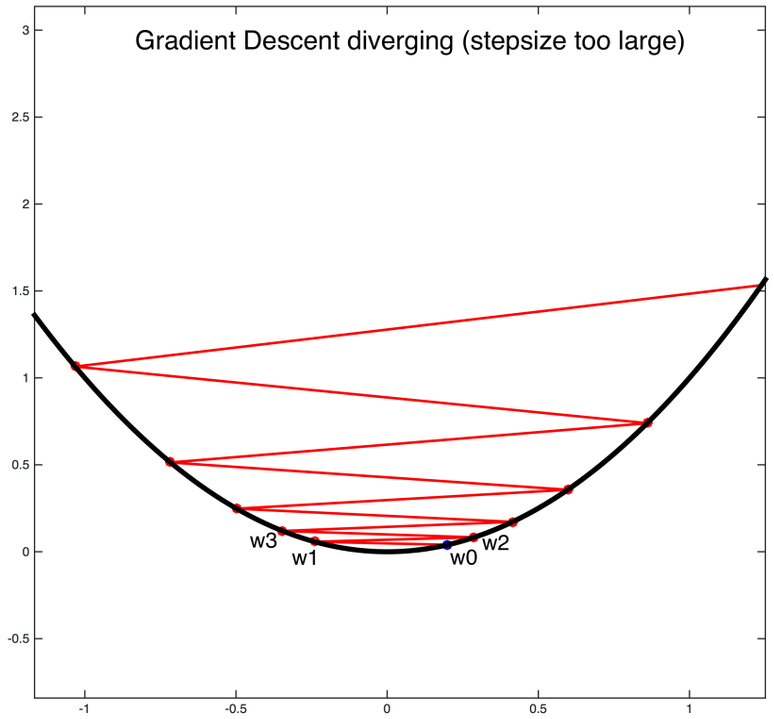
- The difference between M and L is a measure of how ‘wrong’ we are
  —> Loss function = f(M - L), e.g. (M-L)^2

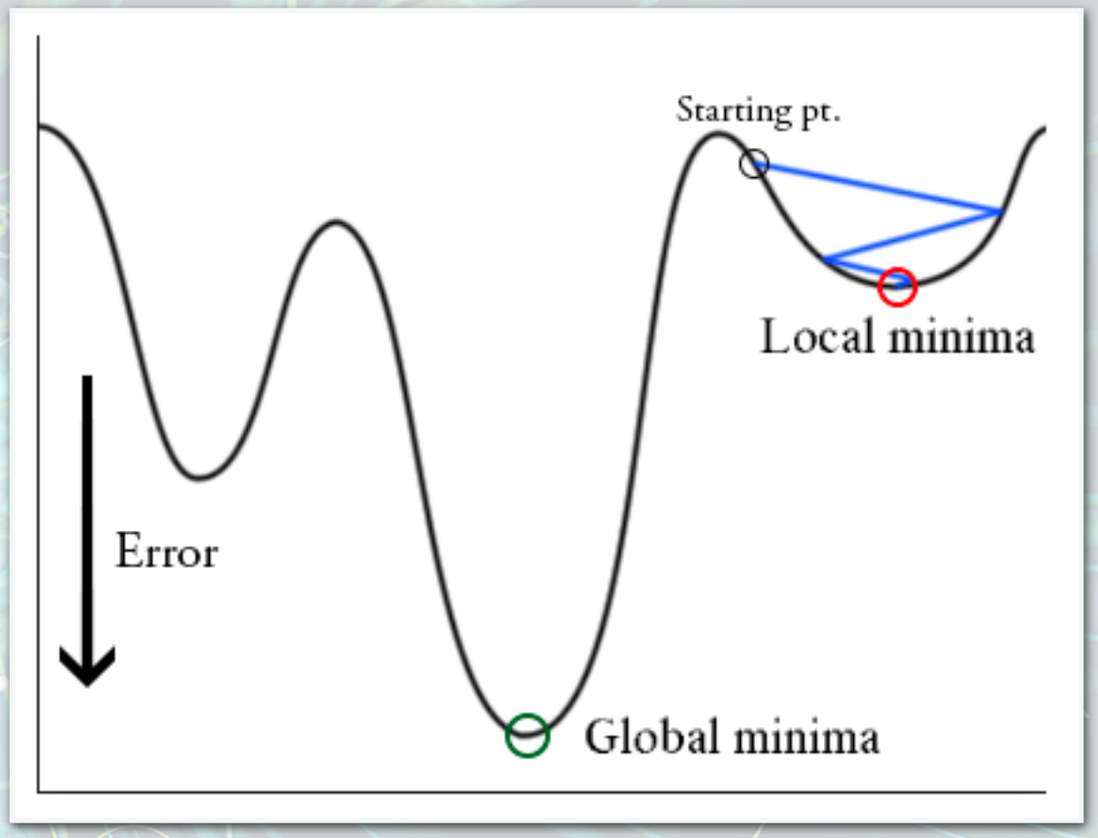
- Using an optimizer we can change the weights. But how should we change them?
  —> Gradient descent!
  d(Loss)/d(w) always perpendicular to the ‘equi-loss’ plane

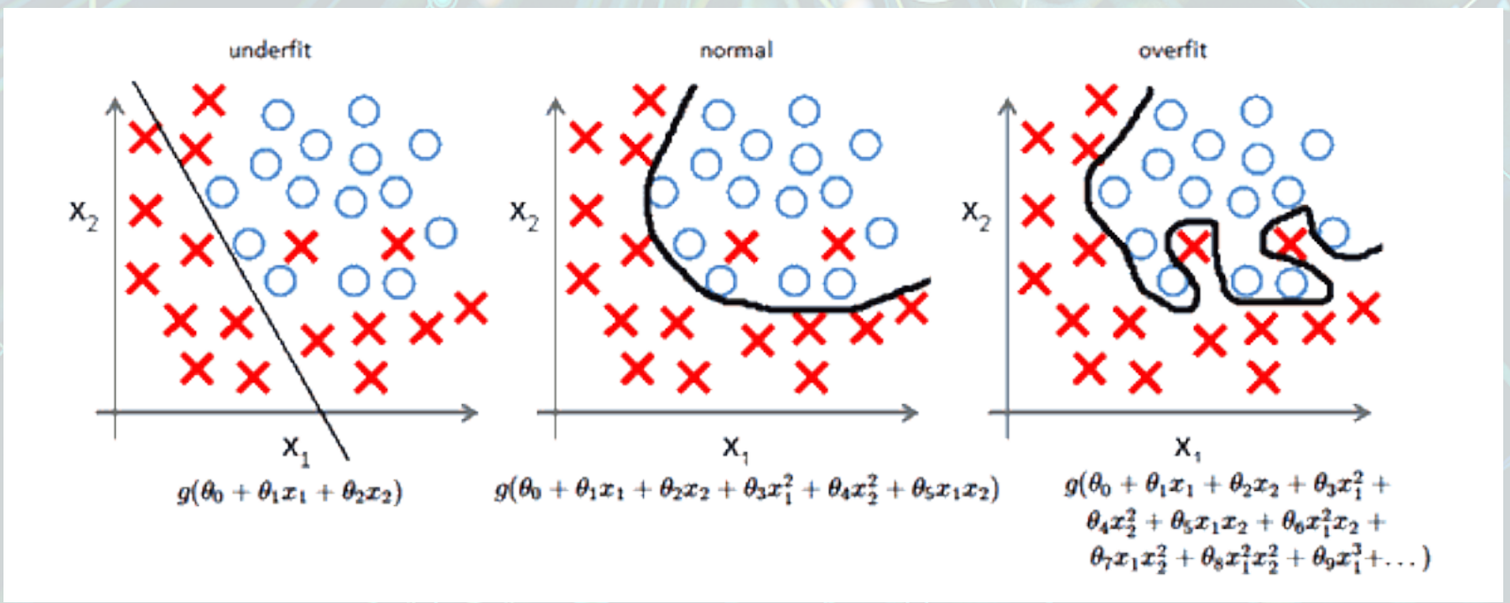
- We can use the chain rule to ‘back-propagate’ from the output through many layers



Training - pitfalls







Why so hot?

- Instead of choosing a very specific model (e.g. Gaussian) with a few parameters,
  —> make little assumptions with a lot of weights.

- Learning generalized behavior of datasets can be much more powerful!
  —> Especially if you don't know your model, (e.g. 'pictures of cats vs dogs')
  —> Or if you want to learn abstract and complex behavior

- To learn generalized behavior, you need A LOT of data (think overtraining)

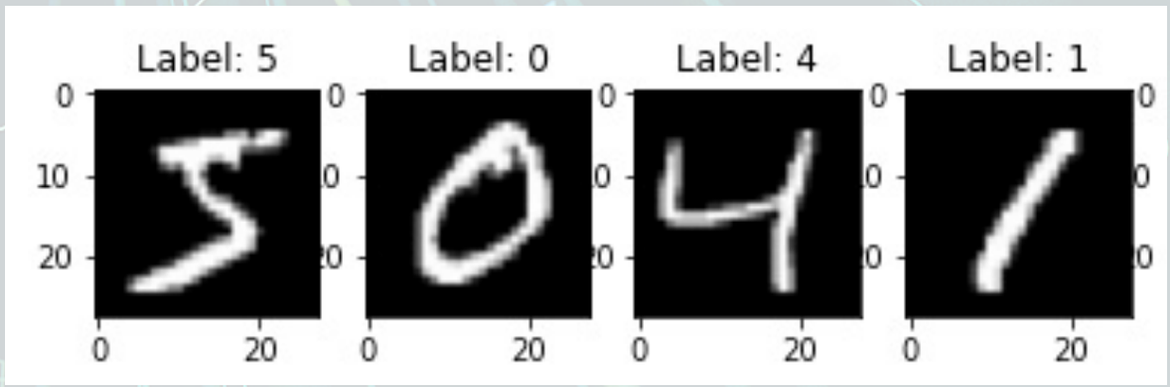
- To learn complex models with multiple layers ('deep learning'),
  with ~100000 weights, you need A LOT of computing power to compute
  gradients, back-propagate, etc.

- —> Fortunately, we now have both!

Morning exercise

- You'll get a dataset of hand-written digits (MNIST)
  (Every 'image' has 28x28 pixels = 784 'inputs', + a 'true' label)
- You will build a network that will train on this dataset,
  and learn to predict a label on a 'new' image it hasn't seen.
- The output will be 10-dim: a score between 0 and 1 for every possible digit, while SUM(all outputs) = 1
- The accuracy (= #correct predictions / all predictions)
  with which your network predicts labels will be your score!
  (obviously it's a competition)

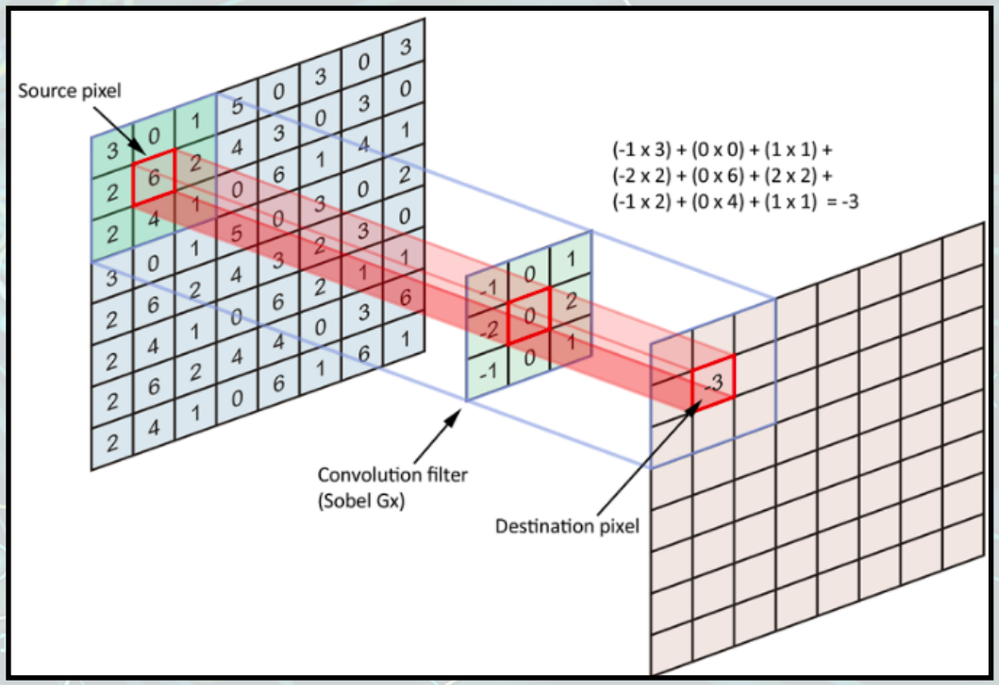
- (This exercise is a warm-up for the afternoon, so no hand-in.)

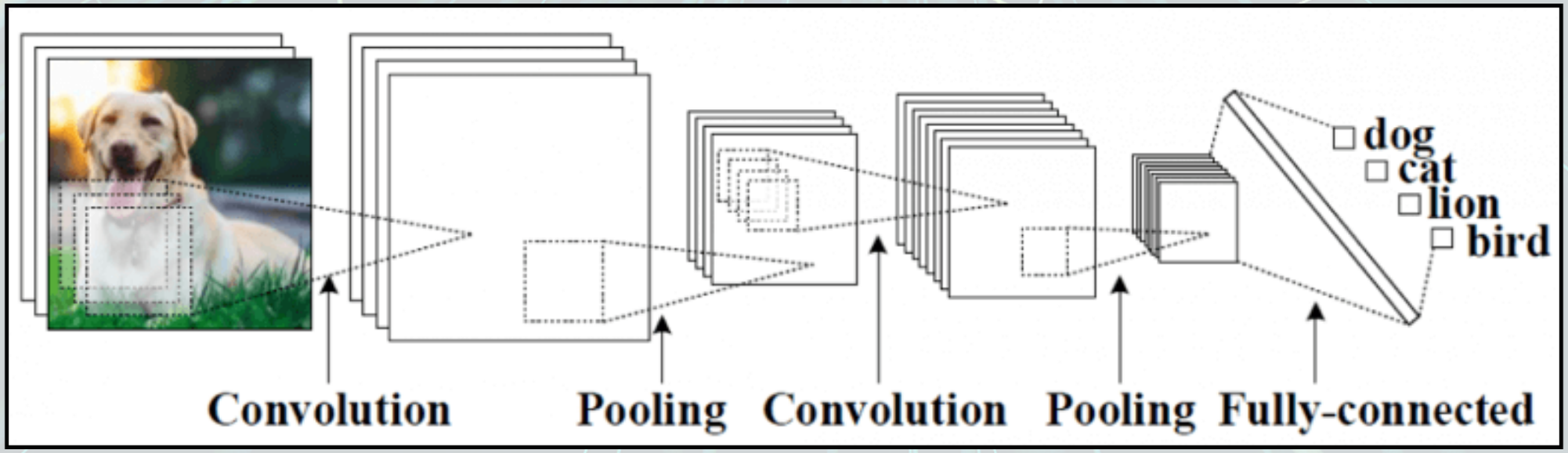


Convolutional networks

- Input: image (30x30 pixel matrix, x 3 colours)
- ‘nodes’: reduction of dimensionality with scanning filters (e.g. 3x3x1)
  —> trainable ‘weights’: numbers in the filter
- Output after many layers:
  a label, e.g. ‘dog’ or ‘cat’

- We will use a more complex dataset now,
  with 10 ‘classes’ of colour figures called CIFAR.



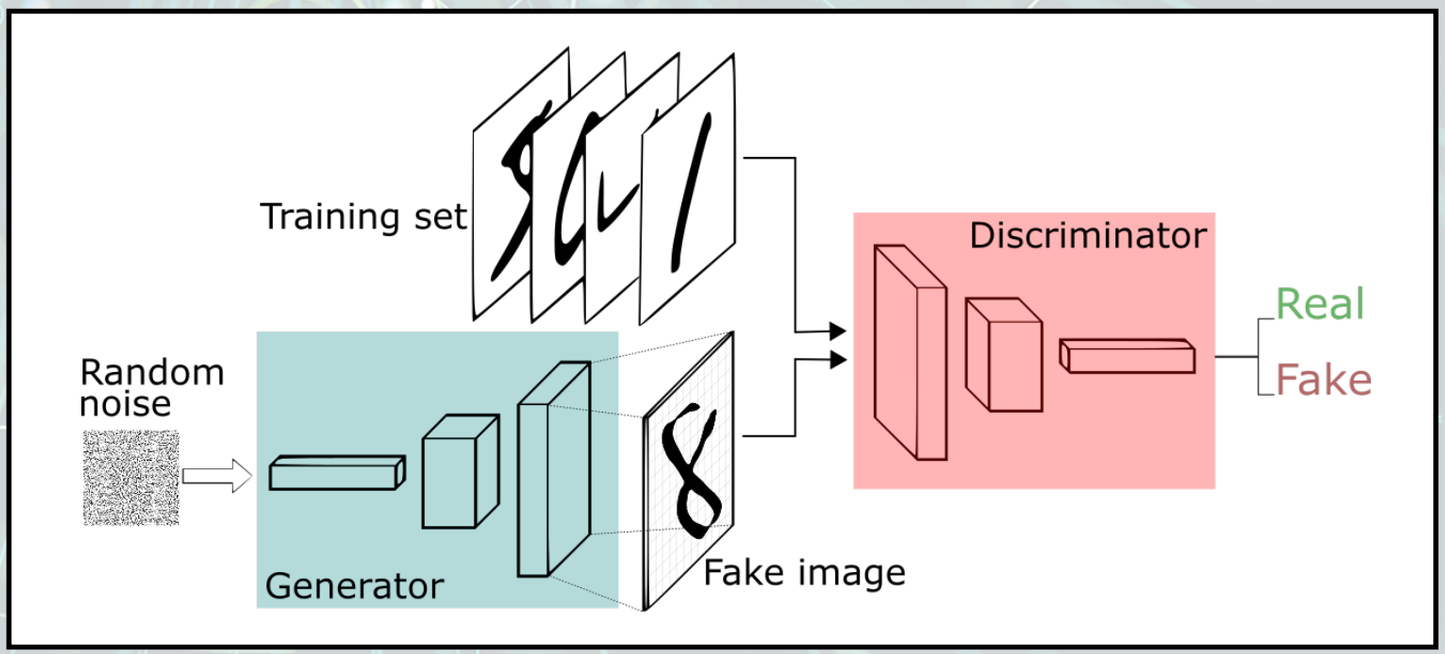


Generative Adversarial Networks

- Generator: a network that takes random noise ('latent space') as input and generates something as output
- Discriminator: a network that sees both real images and the nonsense the Generator outputs, and must distinguish between them.

- Discriminator loss: f(output - label), where label = 1 ('generated') or 0 ('real')
- Generator loss: f(discriminator output)
- Train them simultaneously!

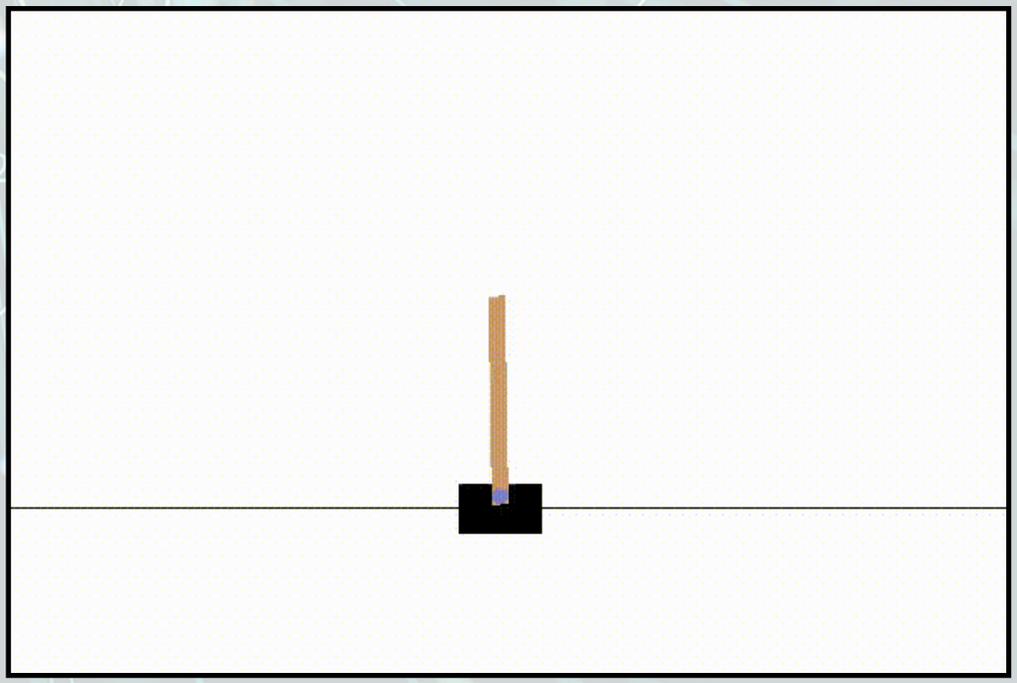
- We will us a dataset of faces (FLW) to generate non-existing faces



Reinforcement learning

- ‘player’ can interact with an environment, take certain actions and receive feedback
- Feedback consists of an updated environment and a ‘score’
- The higher the score before ‘dying’, the better
- The ‘player’ must learn which moves in which environments result in the highest ‘score’

- We will use the gym package which includes ‘Cartpole’
  - goal is to balance an unstable pole
  - every second = 1 point
  - options are ‘left’ and ‘right’
  - death = angle too large



- (protip: it also includes pacman…)

End notes

- We explore _some_ machine learning / AI tools. There are many more! E.g. clustering algorithms, word embeddings, decision trees, Boltzmann machines, support vector machines, graph networks, recurrent networks, variational autoencoders, …

- Use cases in physics are widespread:
  - signal vs background classification
  - generation of simulated events
  - clustering of calorimeter deposits, jets
  - pattern recognition, track reconstruction
  - tagging of particle flavours
  - anomaly detection
  - …

- For the assignments: read the notebook carefully. Add text to explain the code. Evaluate your results (accuracy, overtraining, mistake matrix), and evaluate with text. Extra efforts / functionality gets bonus points.