CHALMERS
UNIVERSITY OF TECHNOLOGY

MACHINE LEARNING AND NEURAL NETWORKS

PART TWO

Bastiaan Bruinsma

March 1, 2022

Chalmers University of Technology

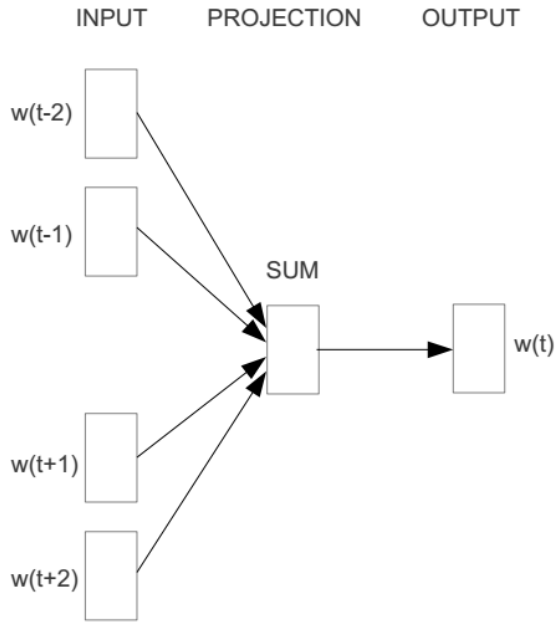
The Programme

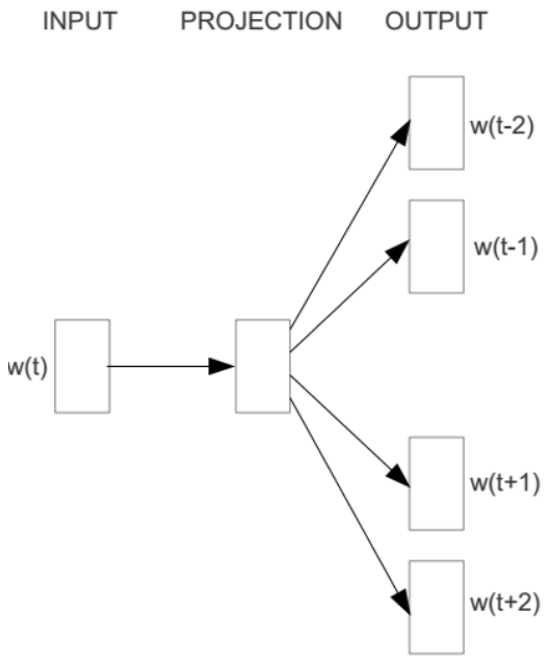
- Today
  - How do we define Machine Learning?
  - Neural Networks and their role
  - Training Neural Networks
  - Explainability
  - Convolutional Neural Networks

- Tomorrow
  - Shallow and Deep Networks
  - Deep Q-Learning
  - Autoencoders
  - Natural Language Processing
  - Generative Adversary Networks

Shallow and deep neural networks

Shallow neural networks is a term used to describe neural networks that usually have only one hidden layer (Word2Vec)



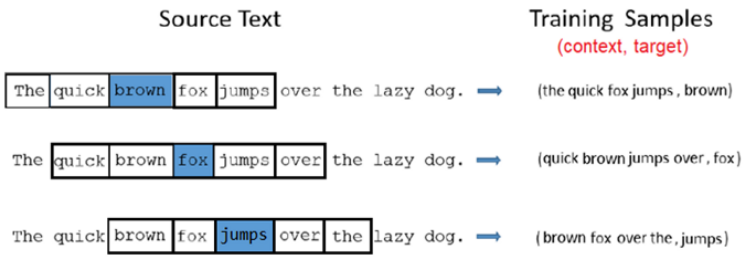


CBOW: takes the context as the input and tries to predict the word corresponding to the context

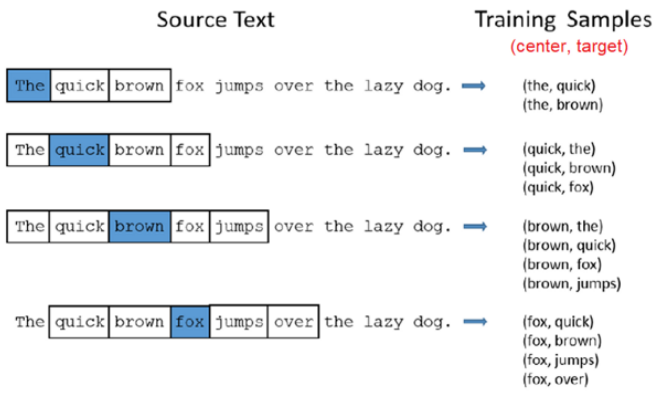
Skip Gram: predict the context for a given target word

Shallow and deep neural networks

Shallow neural networks: word2vec



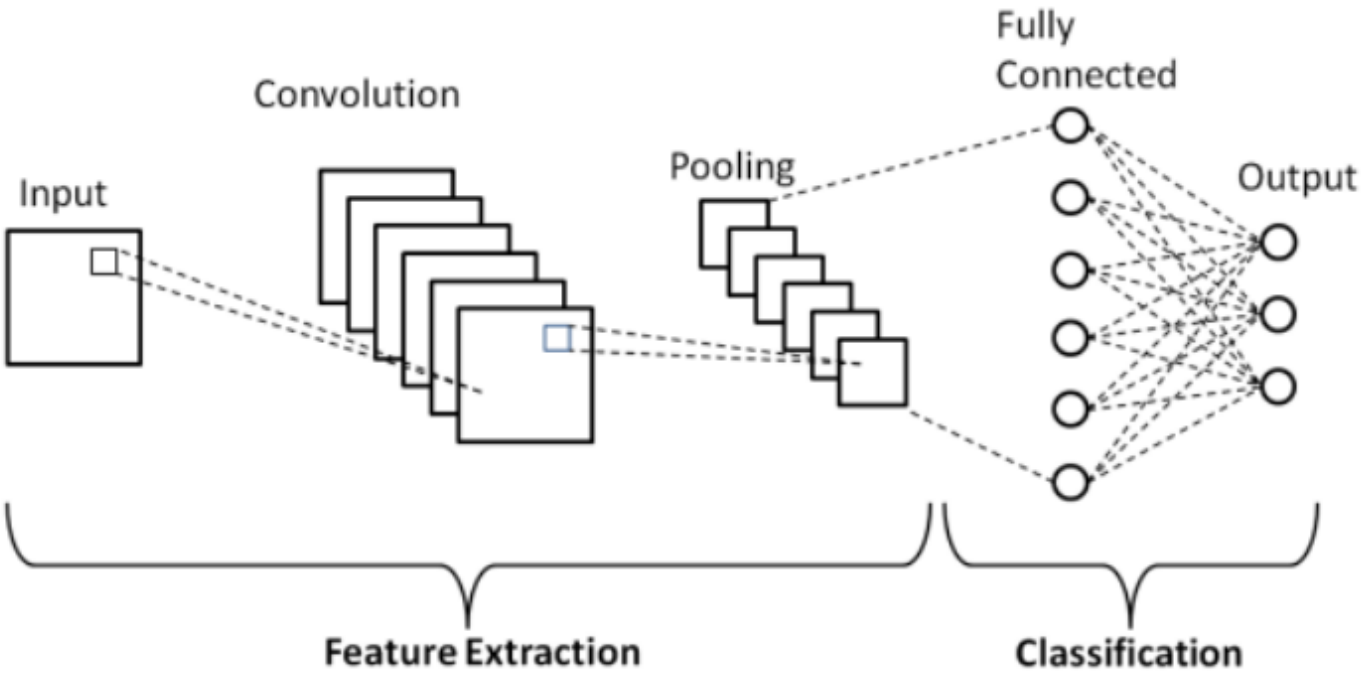
CBOW



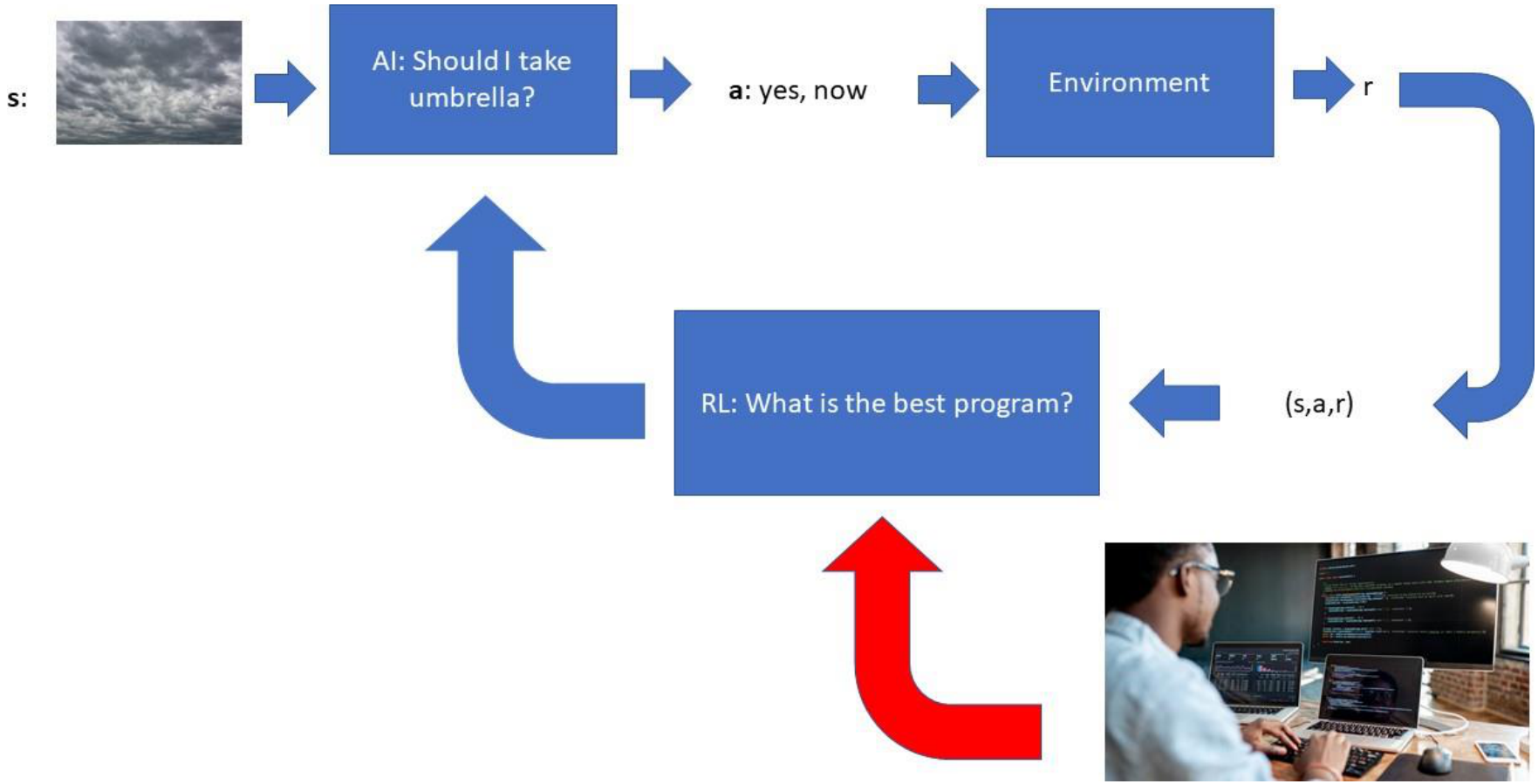
Skipgram

Deep neural networks

- Deep NN which have several hidden layers, often of various types
- Example: CNN



Reinforcement learning



Q Learning

How is the best program selected?

1. $Q(s, a)$ is the maximum expected total reward
2. Exploration: Take a random action and improve the estimate of $Q(s, a)$ by the result
3. Exploitation: take the best action a maximizing $Q(s, a)$

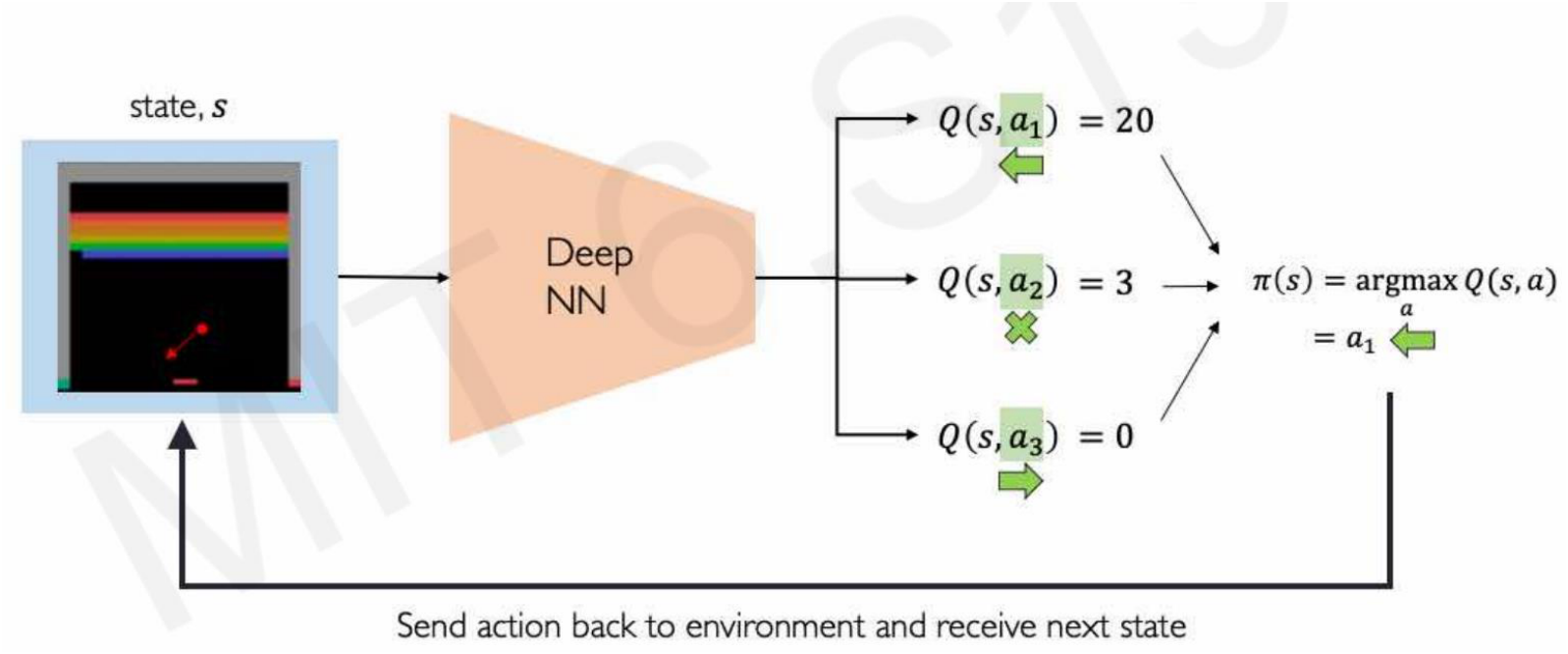
Breakout

Breakout (and other Atari vintage games) can be viewed as MDPs:

- States: a state consists of the last 4 screens (for representing moving objects). The screens are resized to 84x84 and converted to grayscalewith 256 gray levels
- Actions: 18 actions on an Atari controller (8 directions + no direction, combined with the red button)
- Reward: changes to the score of the game
- Maybe we could do table-based Q-learning? Well, there are $256^{84 \times 84 \times 4}$ states…

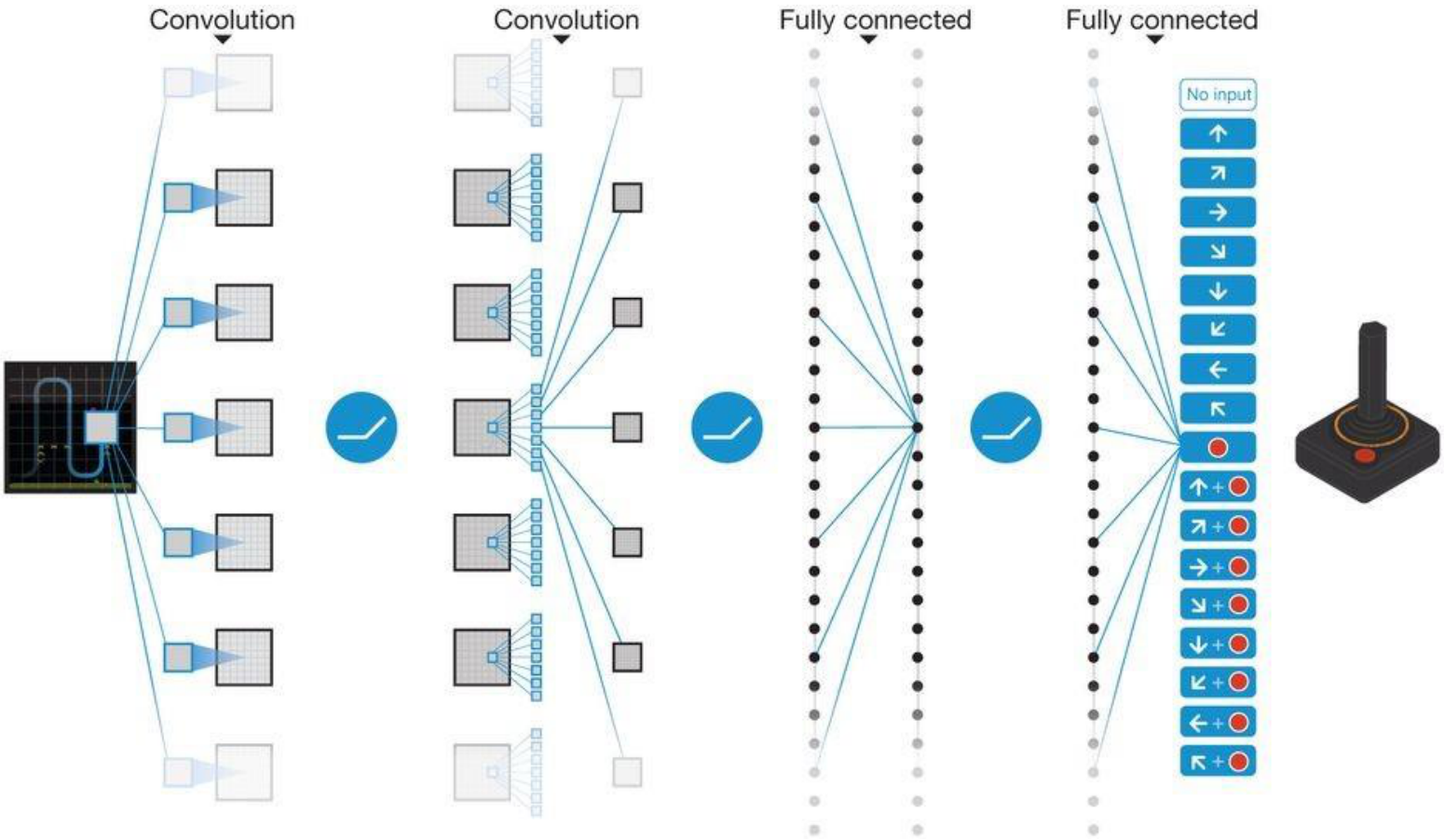
Deep Q-learning

Idea: Could we approximate the Q-values for each action with a neural network?

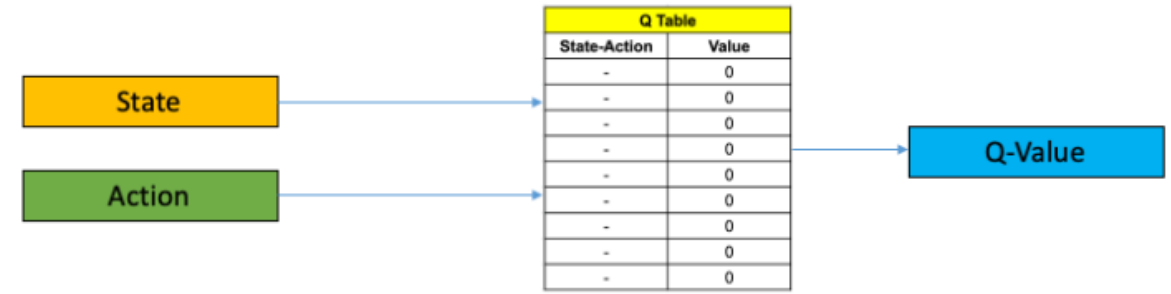


Deep Q-learning

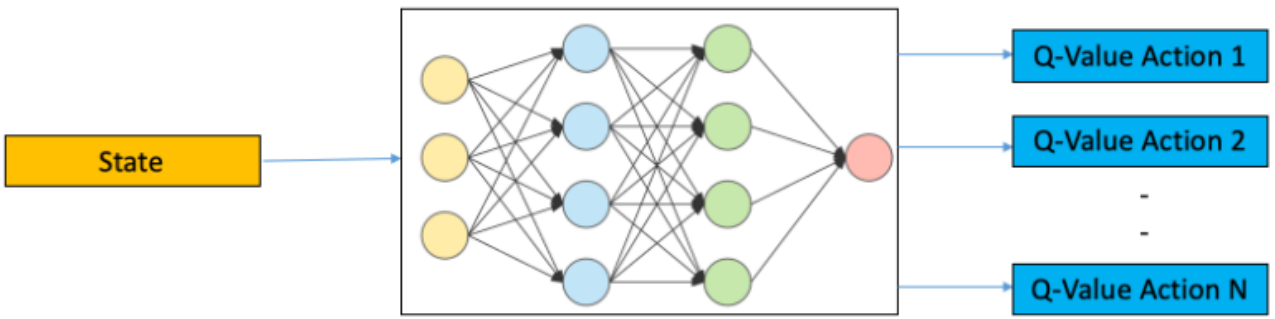
Maybe a convolutional network since we are working with images?



Deep Q-learning

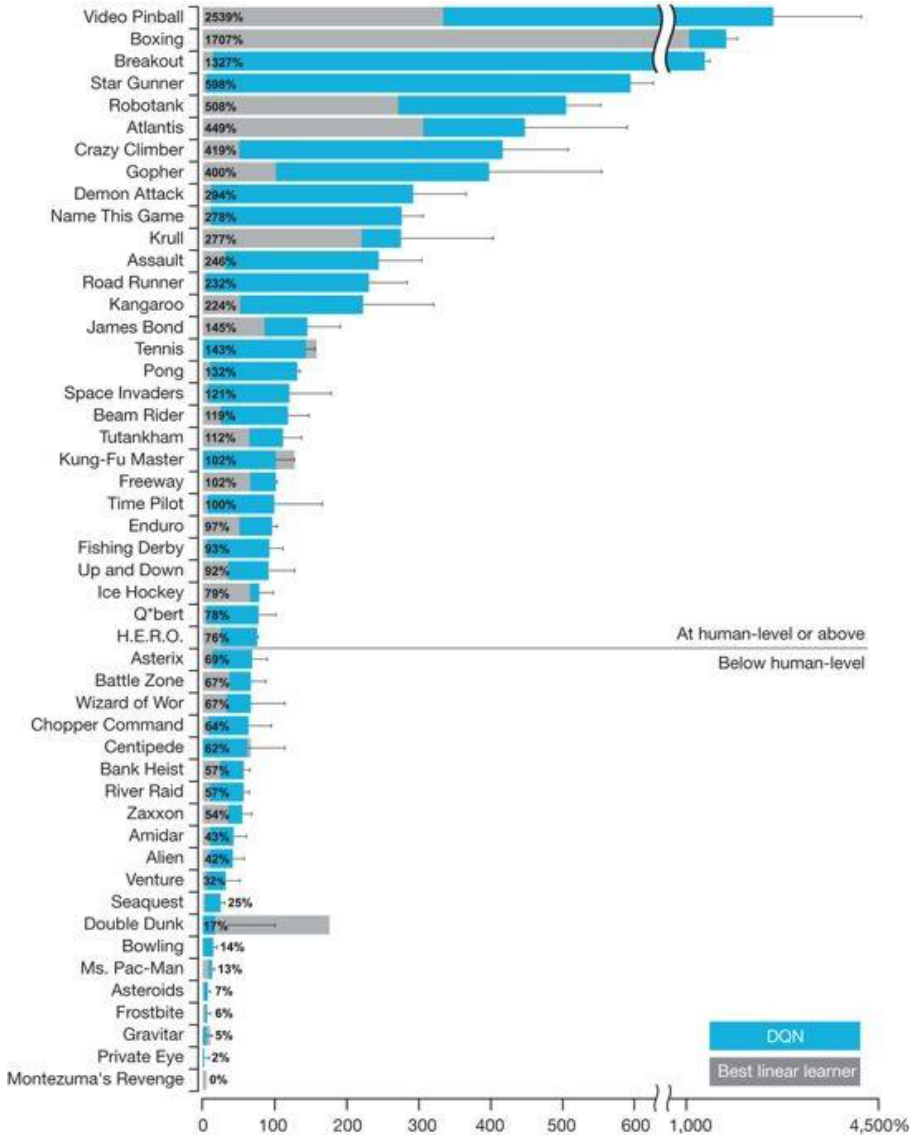


Q Learning

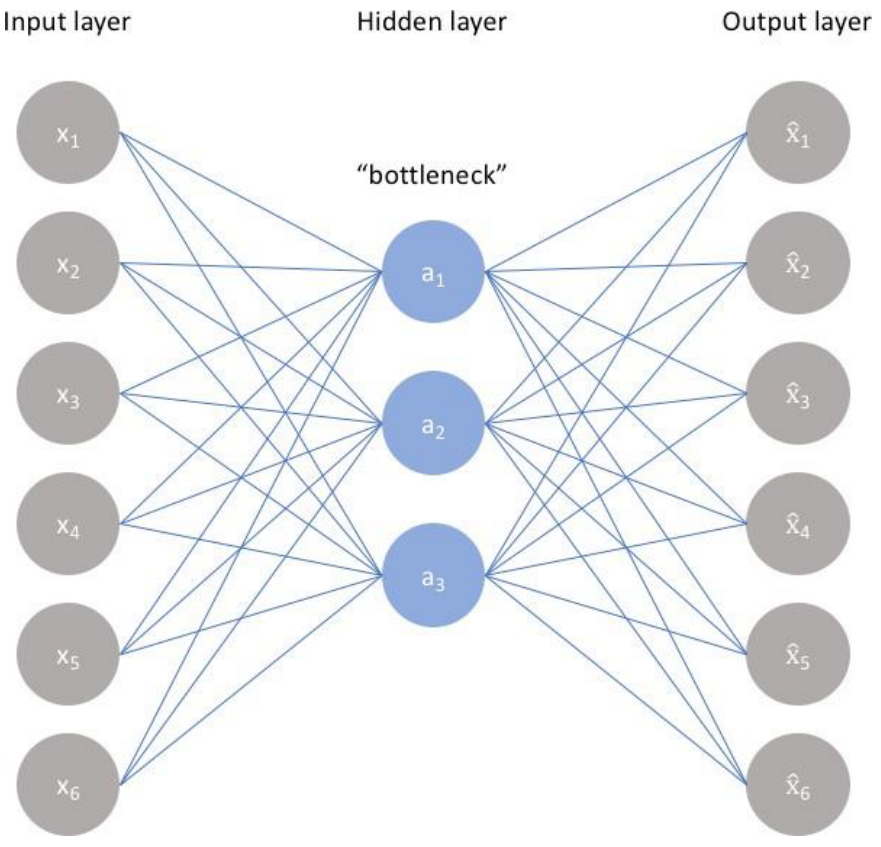


Deep Q Learning

Performance on other Atari games

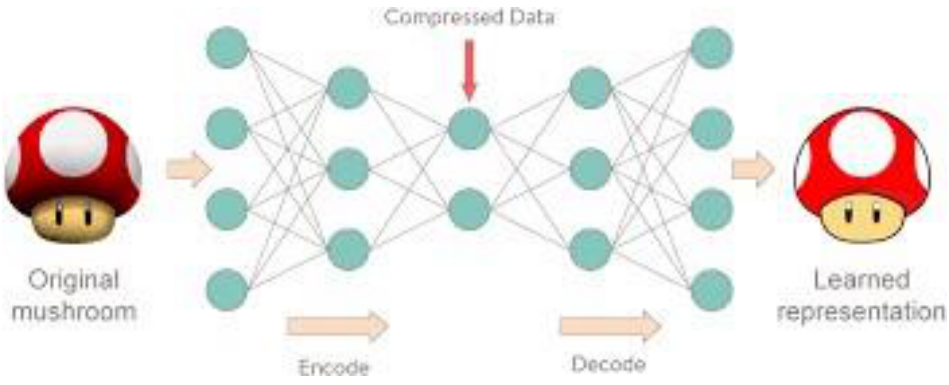


Autoencoders



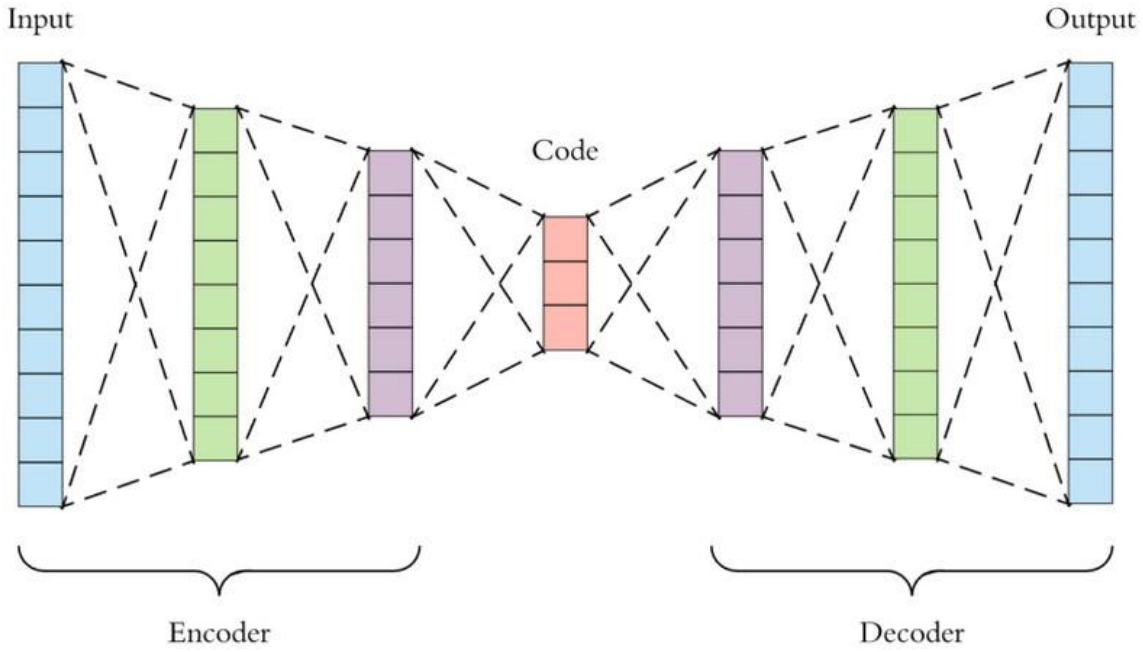
The goal of the training process is to make a neural network that computes the identity function (output = input). Since there is a small layer in the middle of the autoencoder (a *bottleneck*), the task is not trivial. Leads to (lossy) compression.

Autoencoders



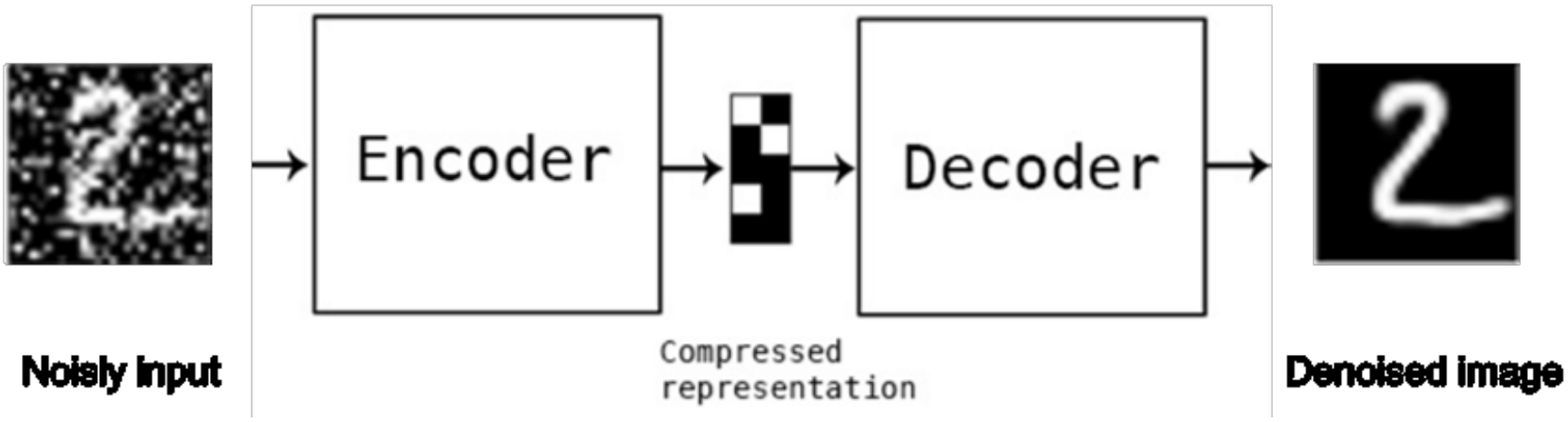
Good to use deep networks with convolution for compressing images

Compression



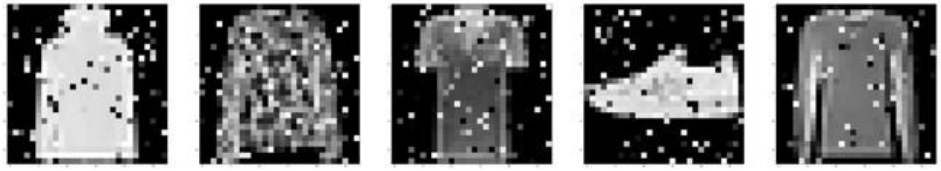
Suppose we want to store 1000 images compacty. Then we can rst train an autoencoder on the images. Then we can compute 1000 codes by using the encoder. Finally we can store the code + the decoder

Denoising autoencoders



Denoising autoencoders

Input images from fashion MNIST.

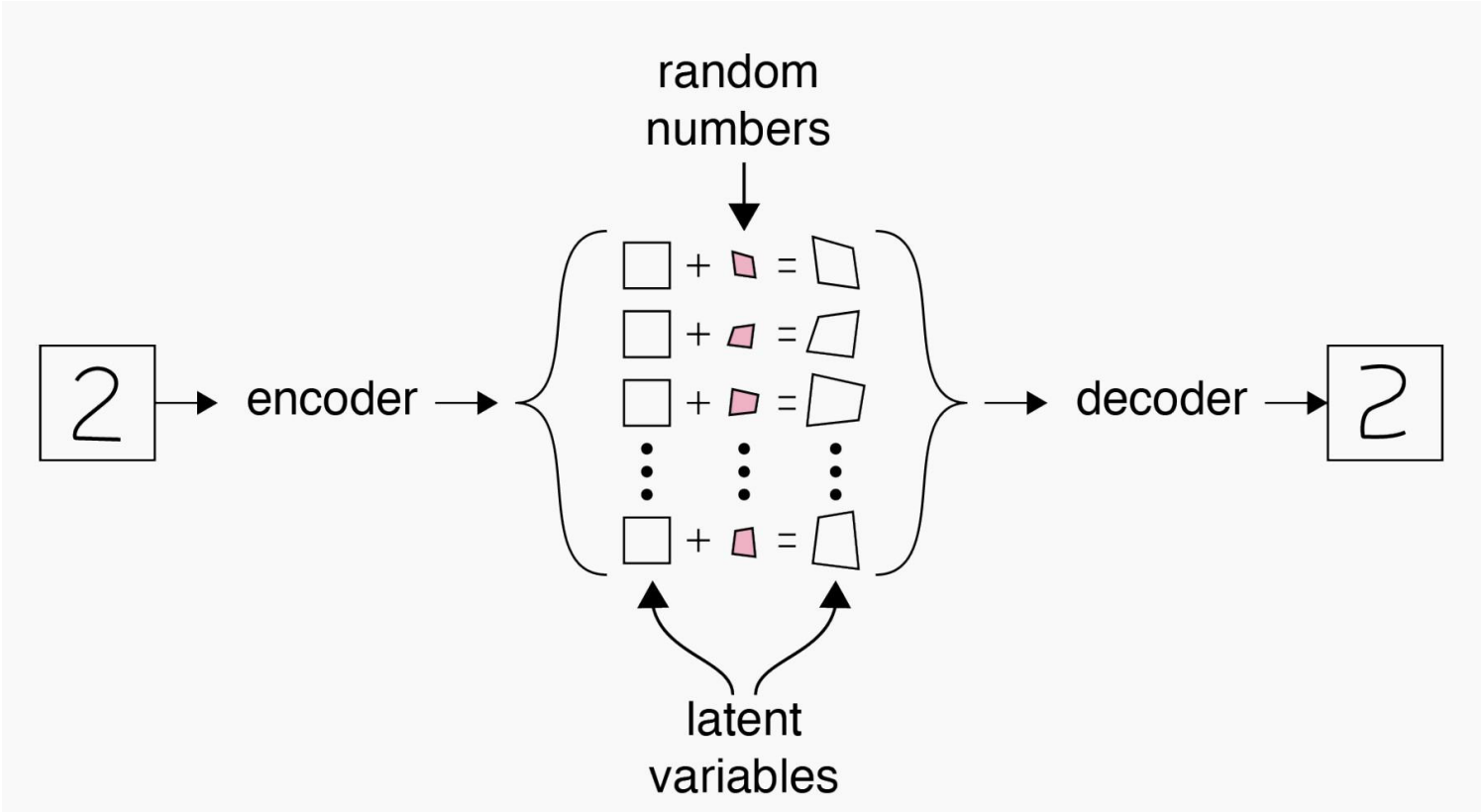


Input images with salt and pepper noise.

Output from denoising network

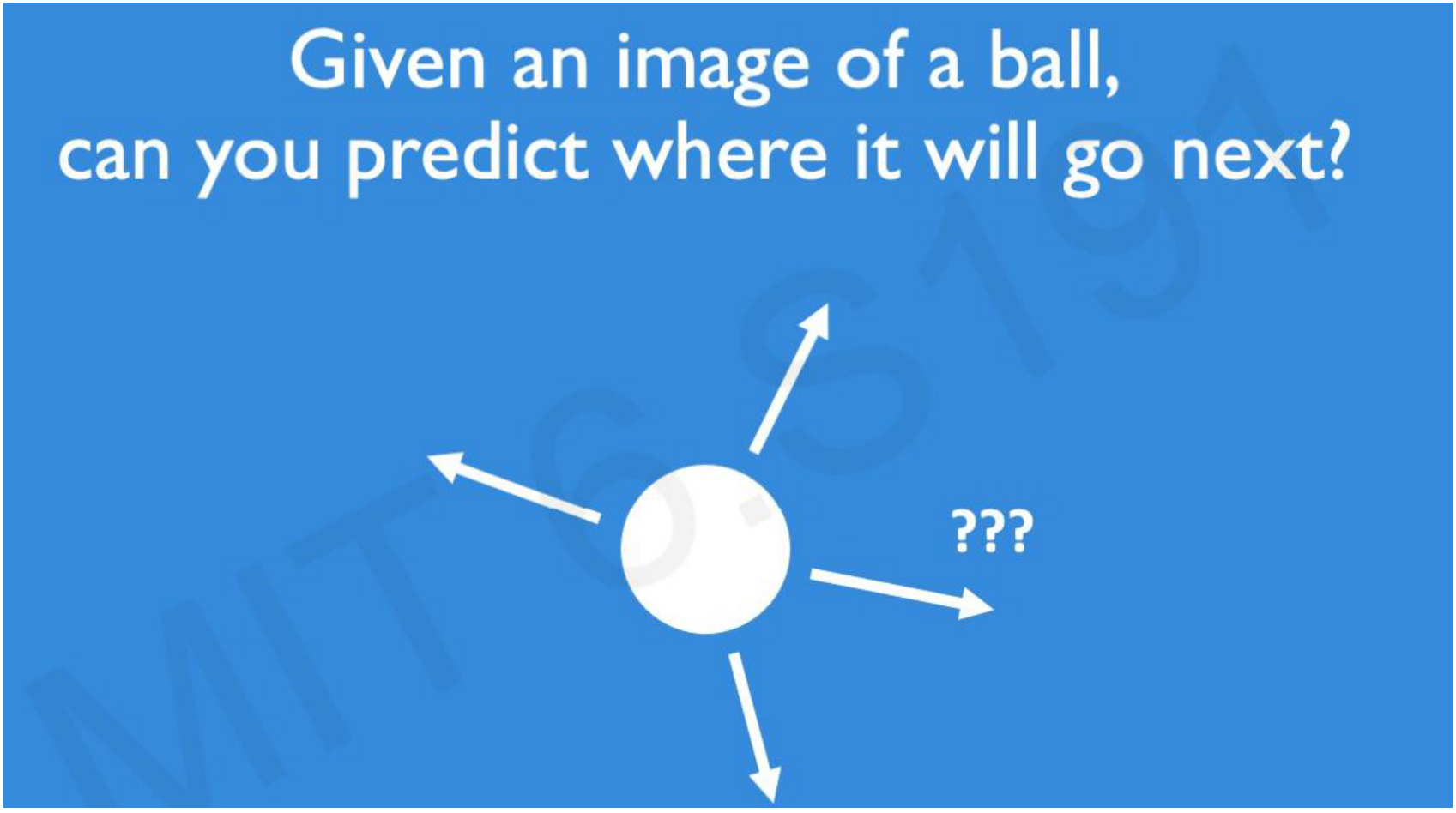
Salt and pepper noise is a type of impulse noise where random pixels get replaced by extremely dark or bright values

Variational autoencoders

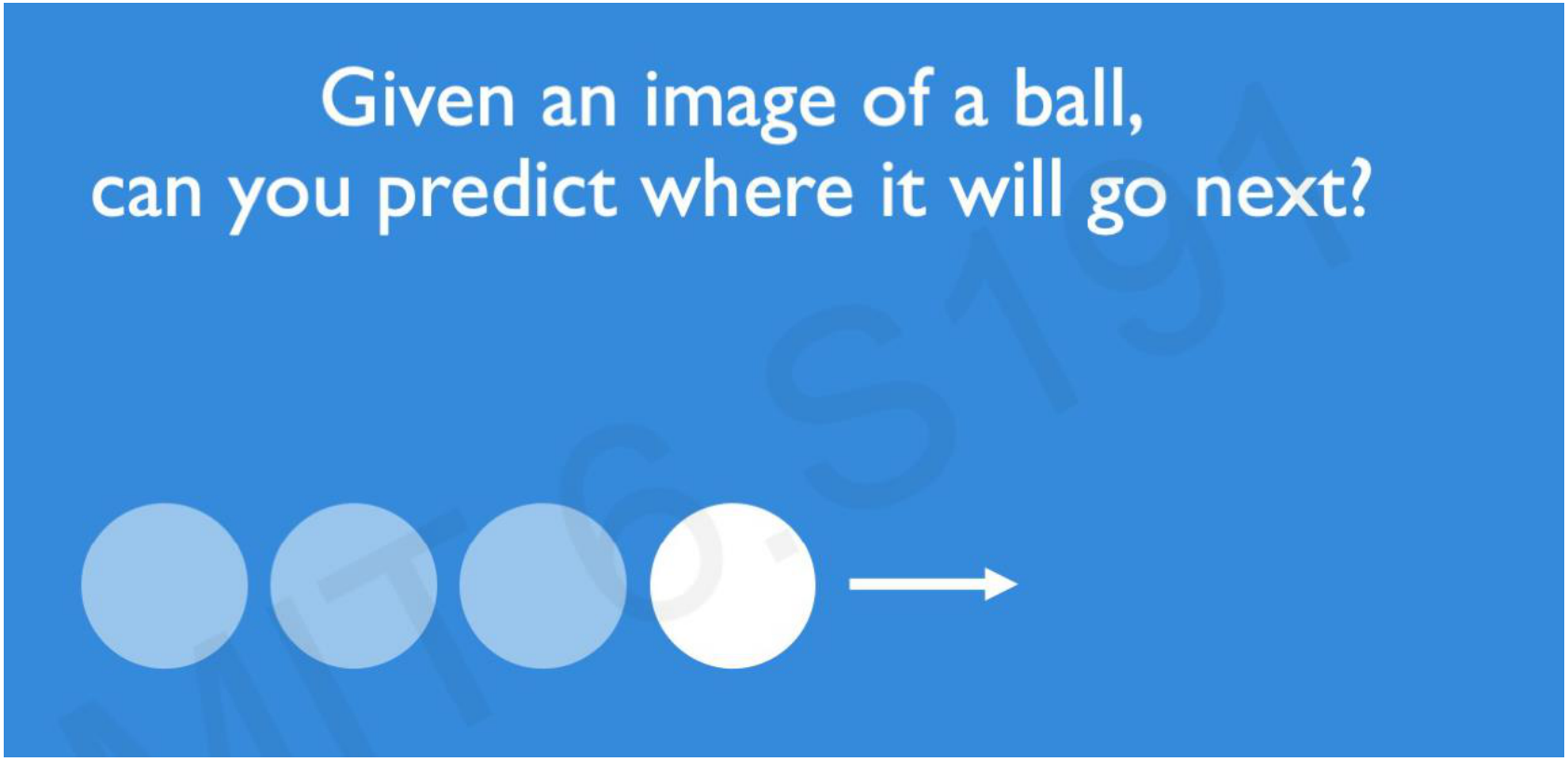


Add some noise at the bottleneck to generate variations of the original image

Recurrent neural networks



Impossible to know without looking at the previous frame(s)


Given an image of a ball,
can you predict where it will go next?

Sequence prediction tasks

Sequence prediction is a problem that involves using historical sequence information to predict the next value or values in the sequence.
Here it also helps to take earlier data points into account (for example, in sentiment analysis):

Text	Positive?
i am good	✓
i am bad	✗
this is very good	✓
this is not bad	✓
i am bad not good	✗
i am not at all happy	✗
this was good earlier	✓
i am not at all bad or sad right now	✓

Sequence prediction tasks

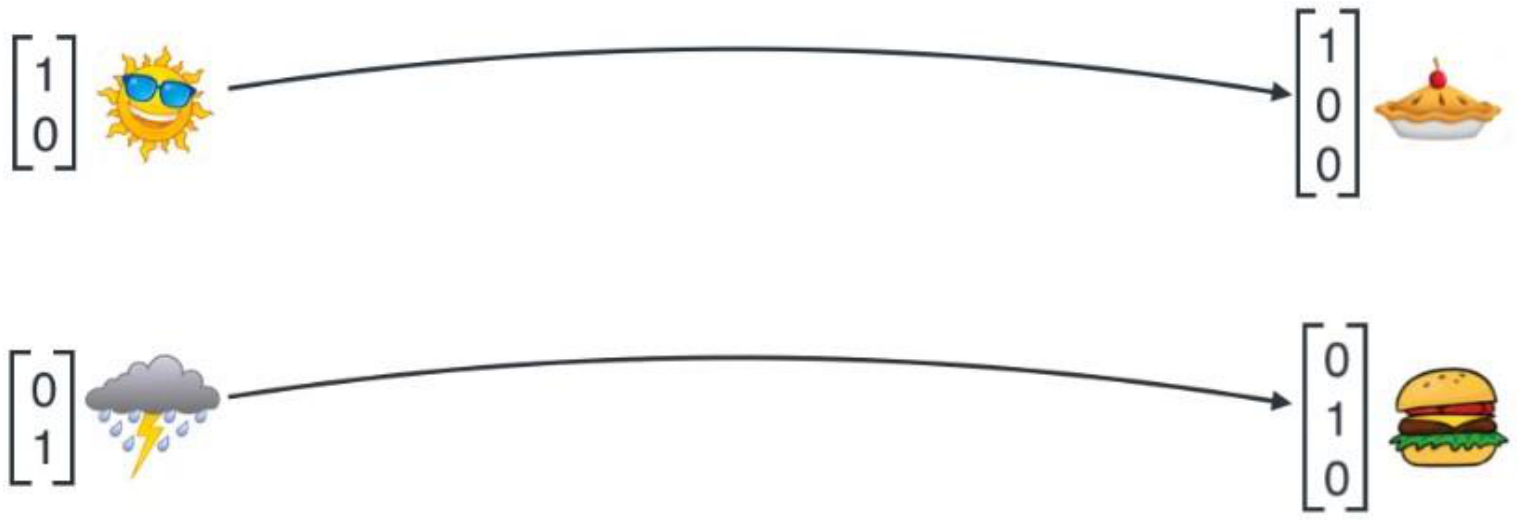
Here it also helps to take earlier data points into account:

- Sentiment Analysis
- Speech Tecognition
- Language Transition
- Stock Prediction

What are we having for dinner tonight?

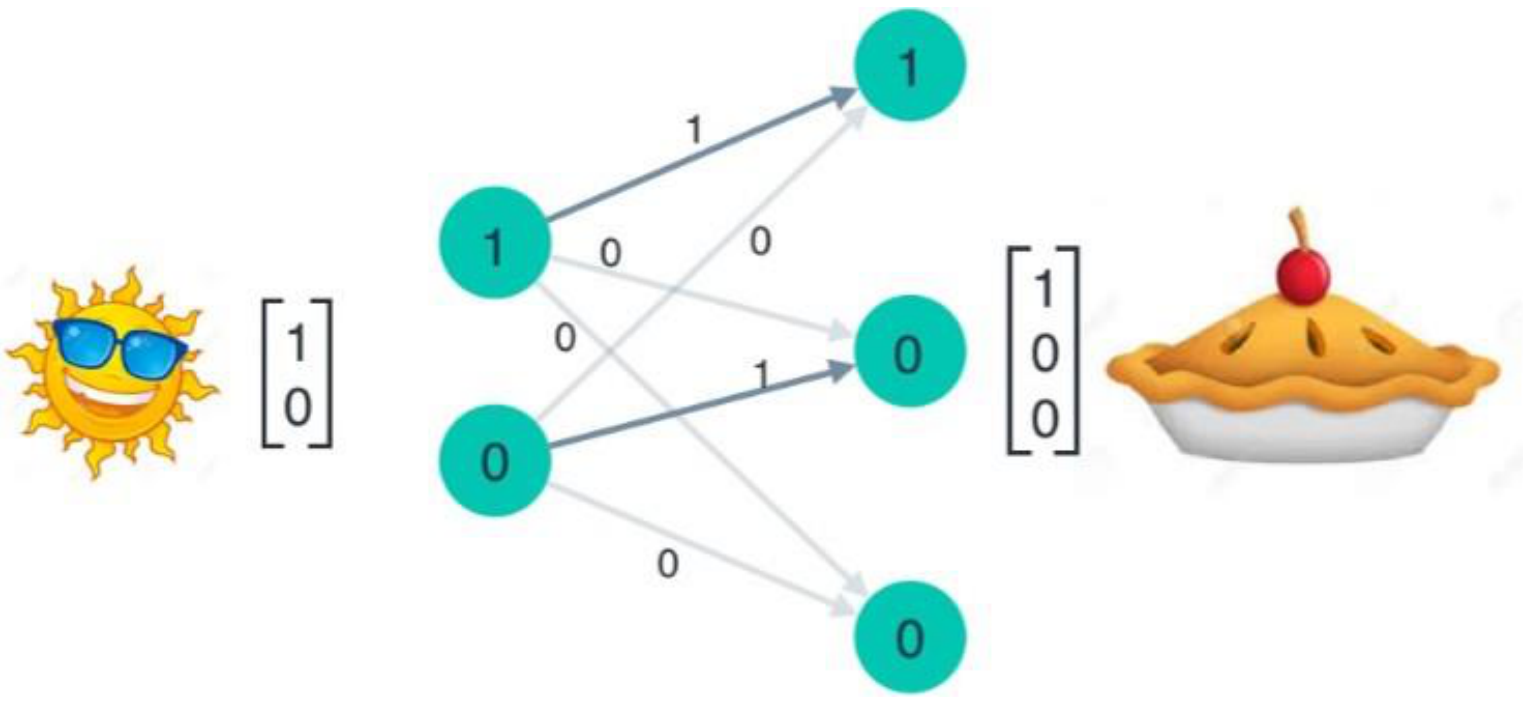
Alice, Bob, and Carol take turns cooking dinner. All three have di erent policies for deciding what to cook

When Alice cooks

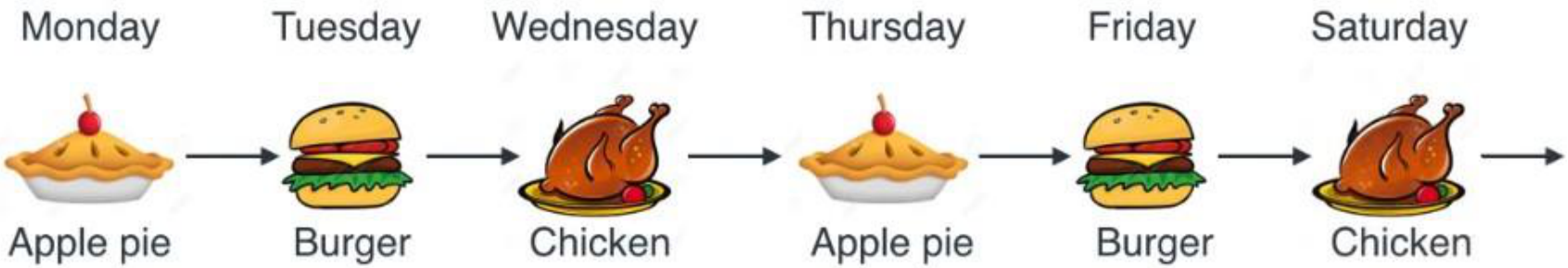


Alice's decision depends on the weather only

Alice's neural network

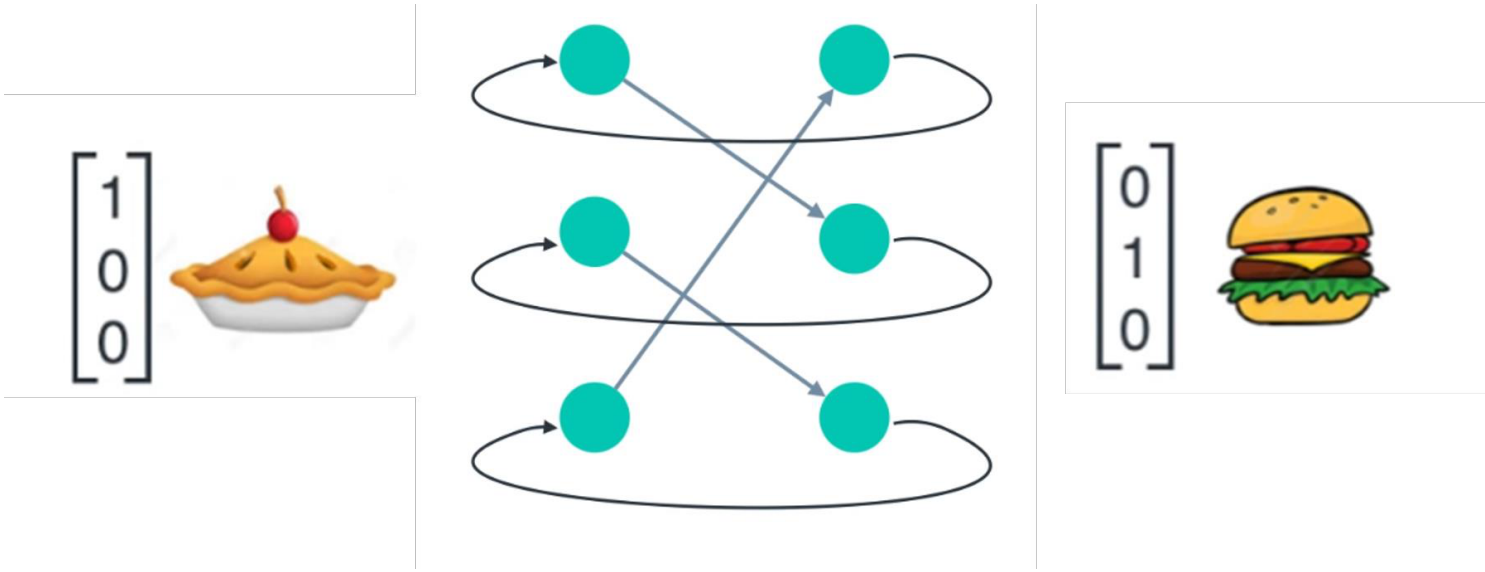


When Bob cooks



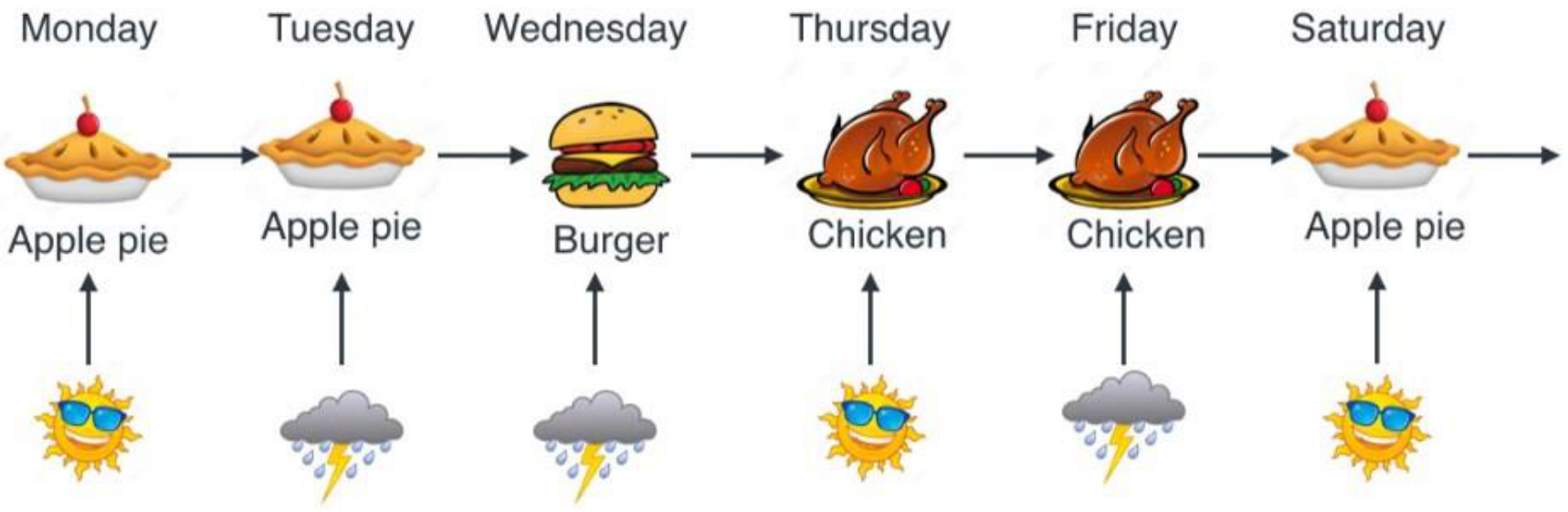
Bob's decision depends on yesterday's dinner only. He always takes the next item on the circular list [Apple pie, Burger, Chicken]

Bob's neural network



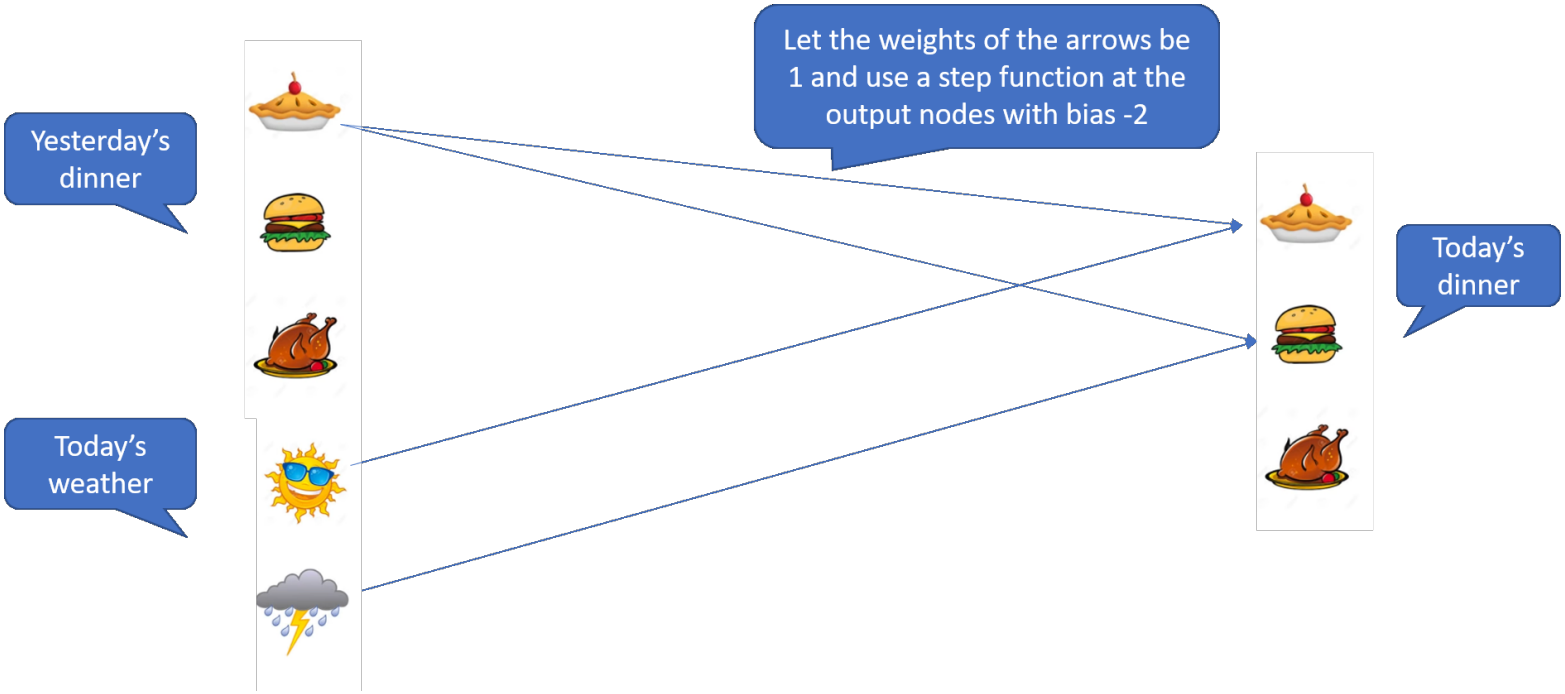
The output nodes give Bob's decision for today's dinner and also feed into the input nodes (with a delay of one time unit)

When Carol cooks



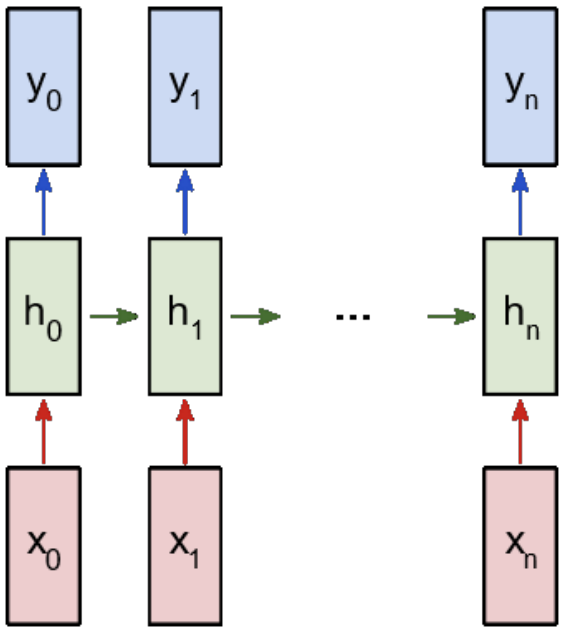
Carol's decision depends both on the weather today and on yesterday's dinner. If it is sunny, she repeats yesterday's dinner. If it is rainy, she takes the next item on the list

Carol's neural network



Several arrows are omitted for clarity. Arrows (with a time delay) from the three output nodes to the top three input nodes are also omitted.

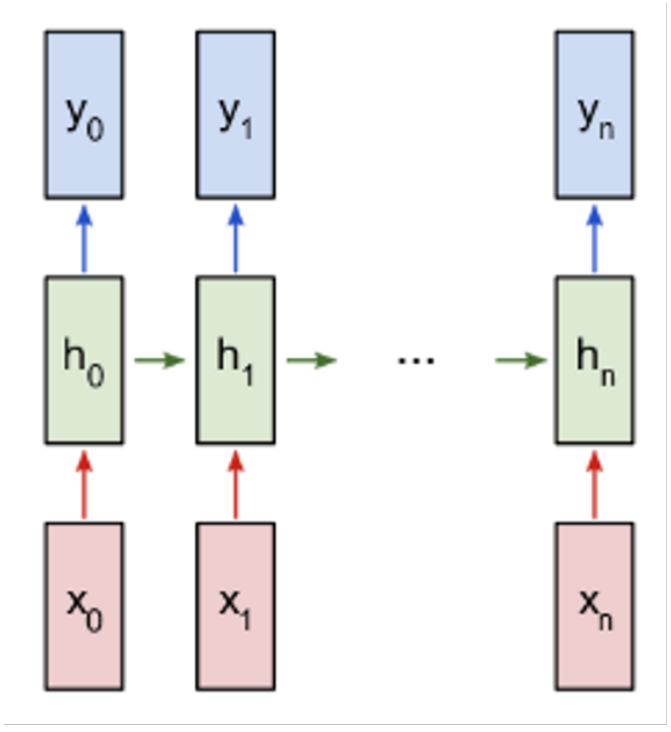
Hidden states

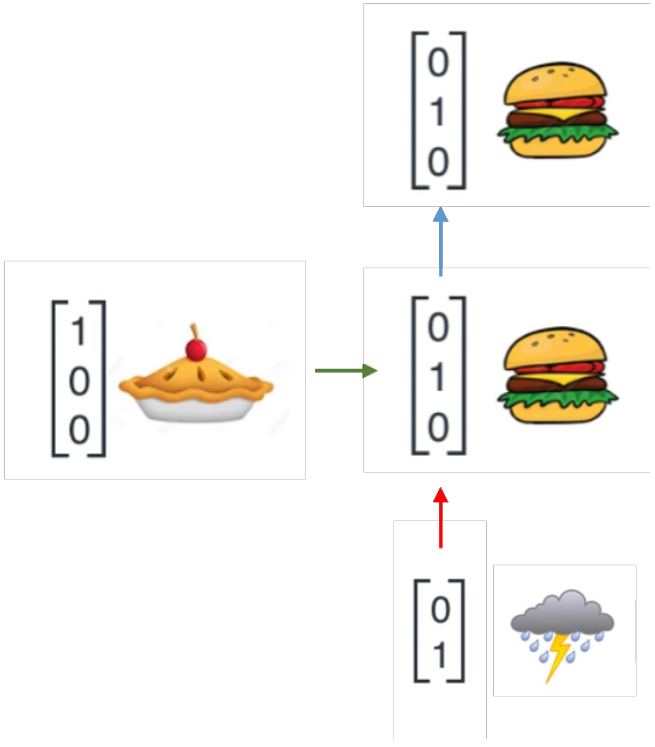


Let us consider a "many to many" RNN with inputs $x_0, x_1, \ldots, x_n$ what wants to produce outputs $y_0, y_1, \ldots, y_n$. These x_i and y_i are *vectors* and can have arbitrary dimensions. RNNs work by iteratively updating a hidden state h which is a vector that can also have an arbitrary dimension. At any given step t:

- The next hidden state h_t is calculated using the previous hidden state h_{t-1} and the next input x_t
- The next output y_t is calculated using h_t

Hidden states





Components of an RNN

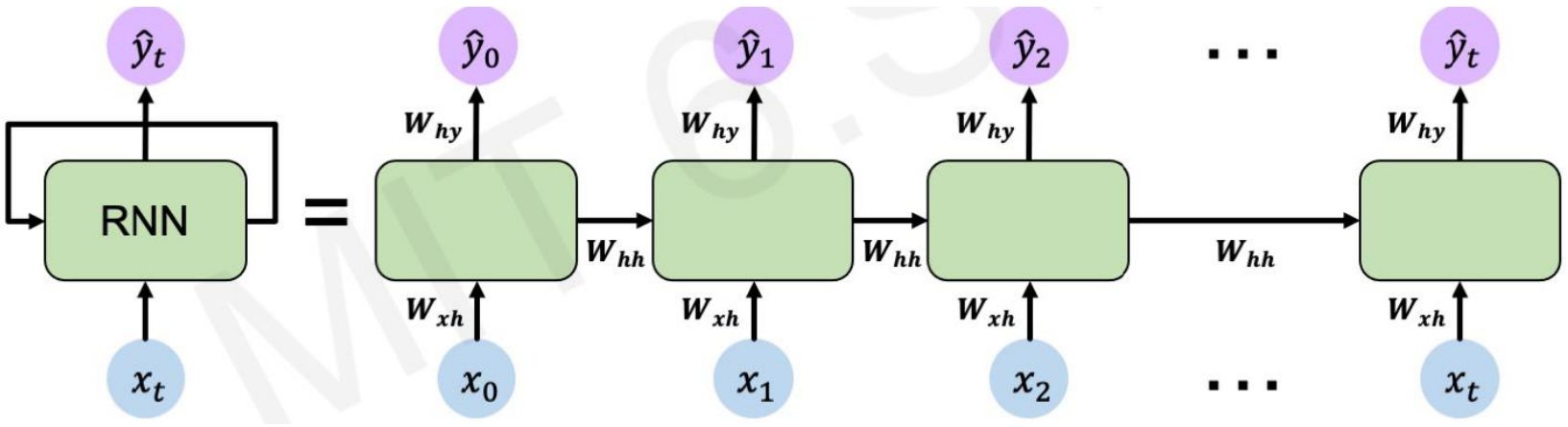
Here's what makes a RNN recurrent: *it uses the same weights for each step*. That is, a typical vanilla RNN uses only 3 sets of weights to perform its calculations:

- We use W_{xh} for all $x_t \rightarrow h_t$ links
- We use W_{hh} for all $h_{t-1} \rightarrow h_t$ links
- We use W_{hy} for all $h_t \rightarrow y_t$ links

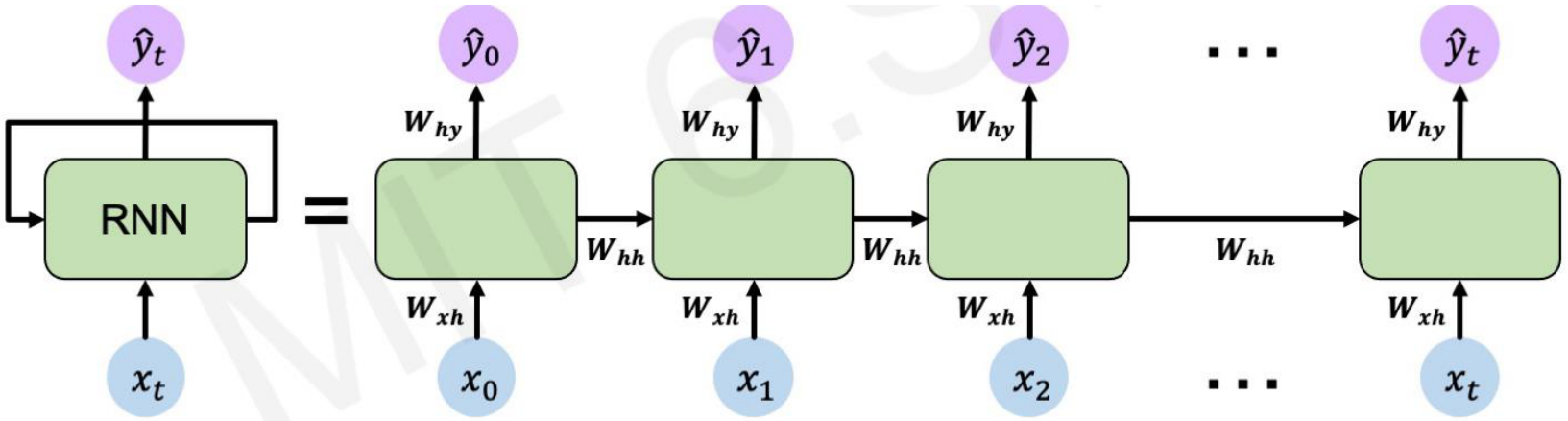
We will also use two biases for our RNN:

- We add b_h when calculating h_t
- We add b_y when calculating y_t

Components of an RNN



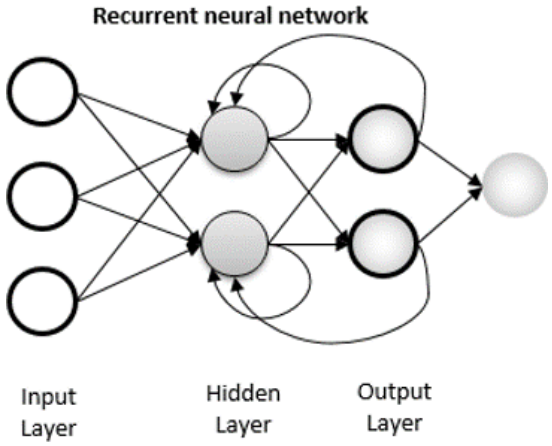
Computations of an RNN



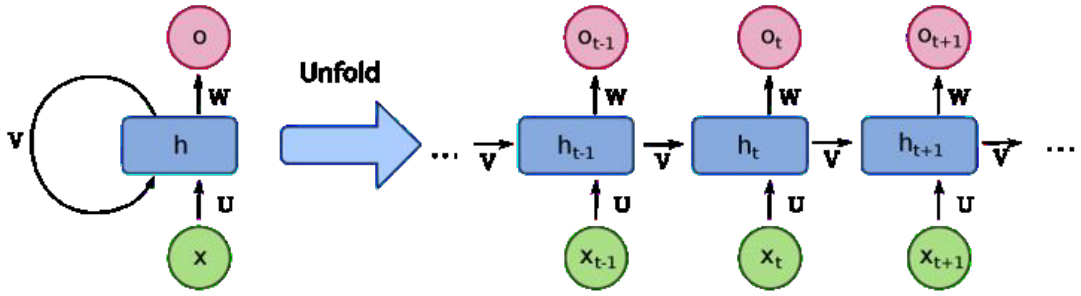
$$h_t = tanh(W_{xhx_t} + W_{hh}h_{t-1} + b_h)$$
$$y_t = W_{hy}h_t + b_y)$$

Recurrent networks

- Recurrent networks contain loops



- The output of a recurrent network depends on the input, but also on time, which proceeds in discrete steps



Input and output sizes

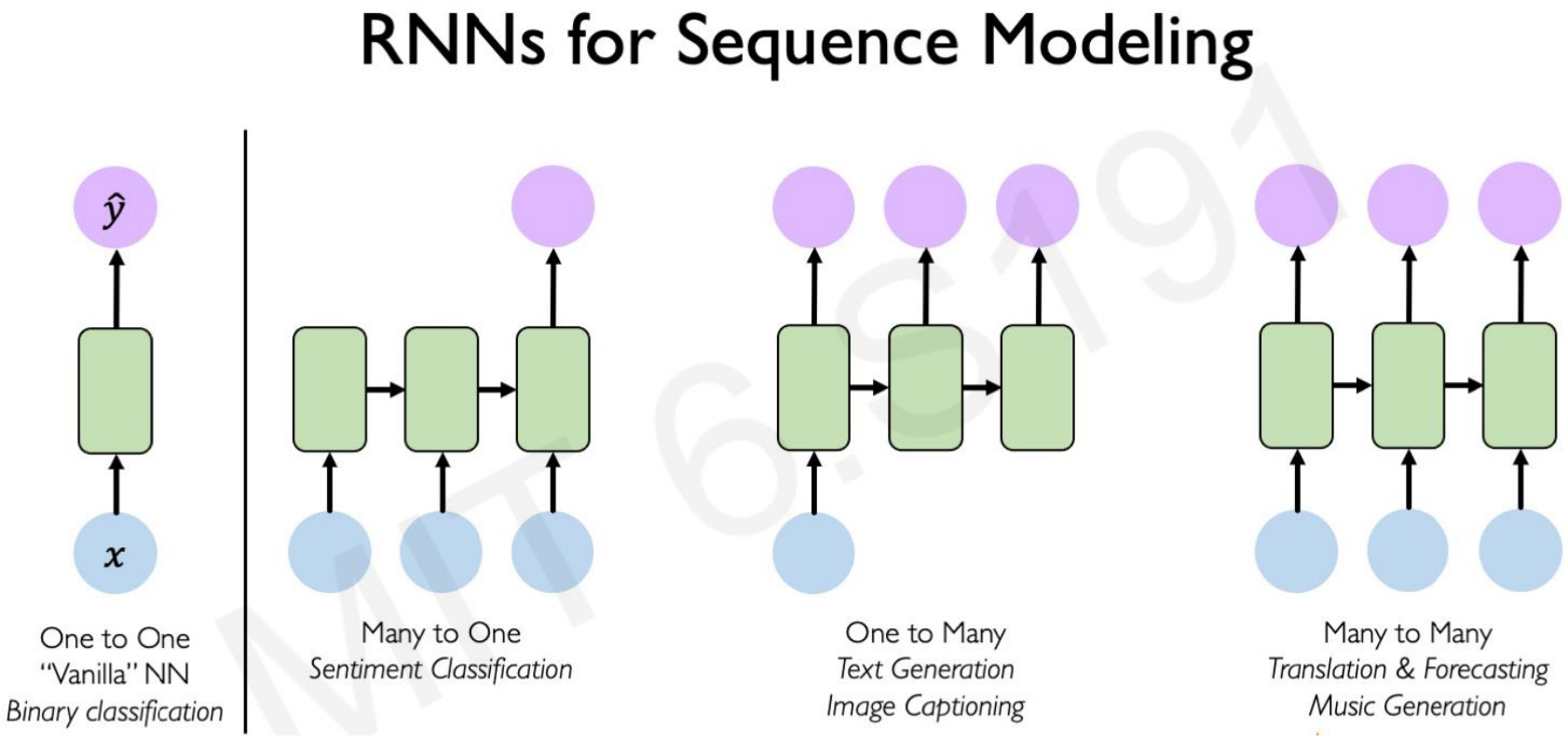
- One issue with vanilla neural nets (and also CNNs) is that they only work with pre-determined sizes: they take fixed-size inputs and produce fixed-size outputs
- What about a network for translating French sentences into English sentences? How many input and output nodes should it have?
- Recurrent neural networks (RNNs) are useful because they let us have variable-length sequences as both inputs and outputs

Sequence Modeling: Design Criteria

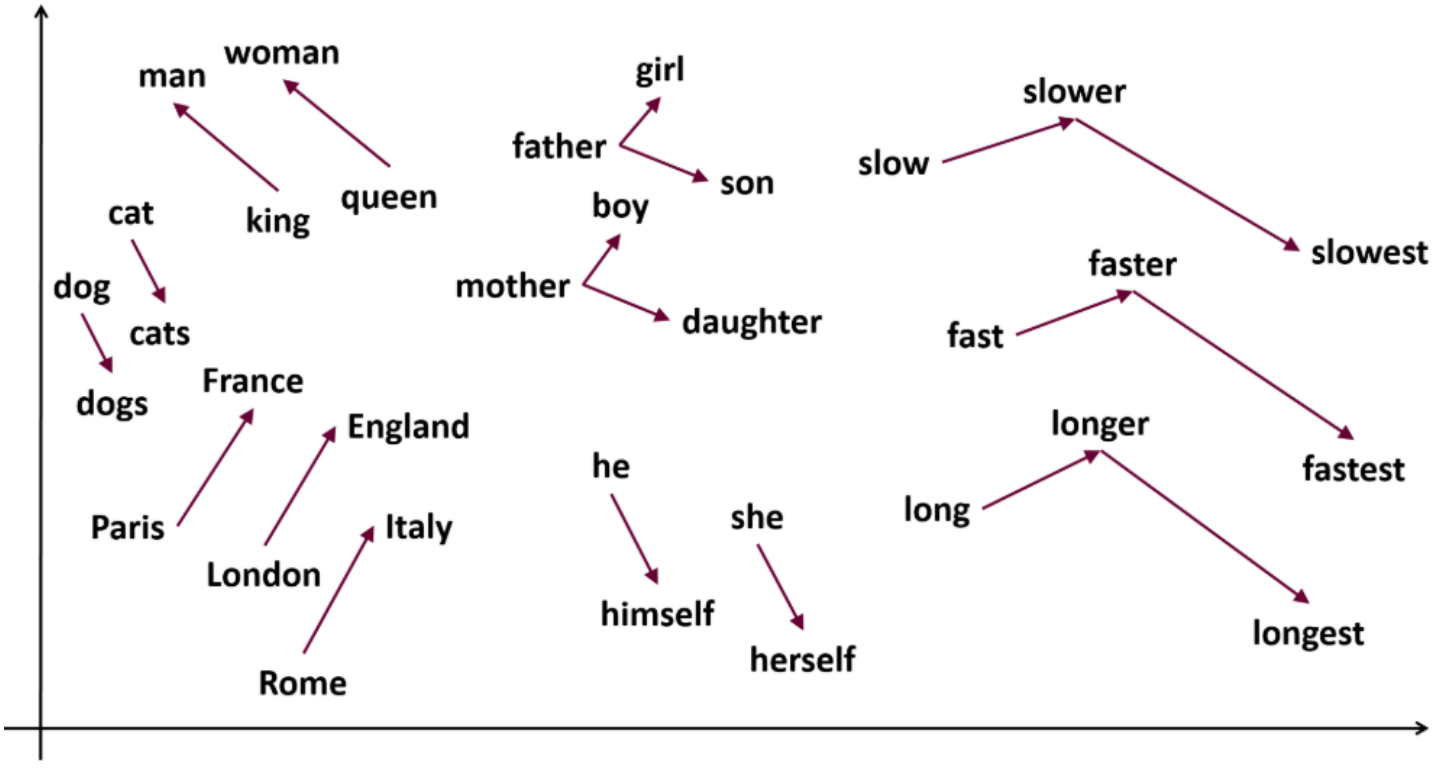
To model sequences, we need to:

- Handle *variable-length* sequences
- Track *long-term* dependencies
- Maintain information abour *order*
- *Share paramenters* across the sequence

Recurrent Neural Networks meet these sequences modeling design criteria


RNNs for Sequence Modeling
$\hat{y}$
x
One to One
"Vanilla" NN
Binary classification
Many to One
Sentiment Classification
One to Many
Text Generation
Image Captioning
Many to Many
Translation & Forecasting
Music Generation

Natural Language Processing - Word2vec



Alexander Amini and Ava Soleimany, MIT 6.S191: Introduction to Deep Learning

Natural Language Processing

IBM's Watson computer beat two of the most successful human contestants on the long-running US game show Jeopardy!, which requires participants to provide a question in response to general knowledge clues.

Google Neural Machine Translation

Google Översätt

Google Neural Machine Translation

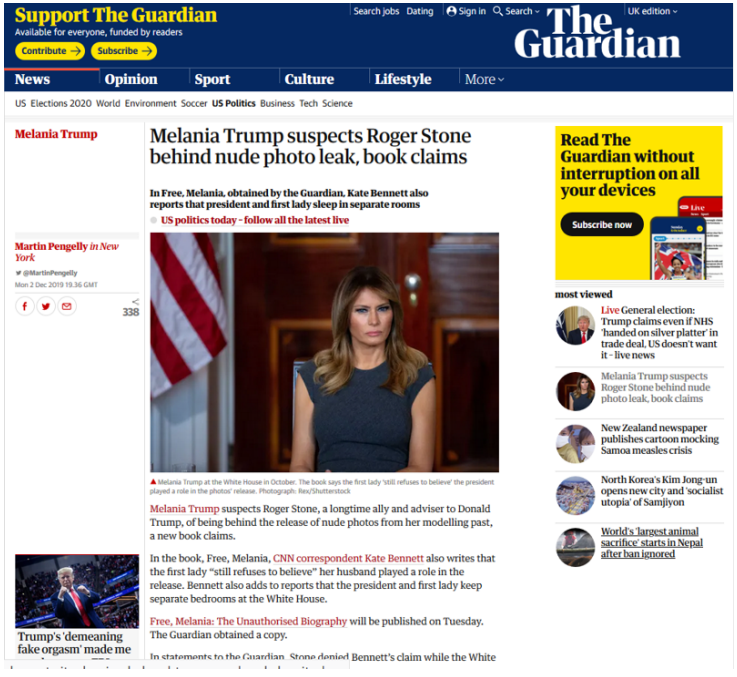
Google Översätt

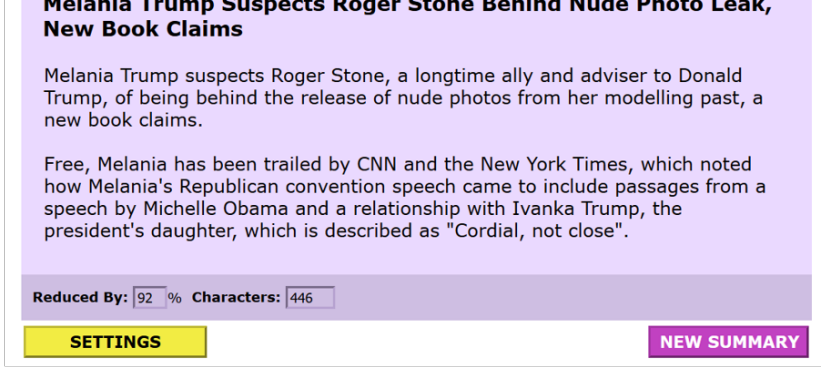
Natural Language Processing

Debater was rated more informative than the human on stage

Natural Language Processing

Natural Language Processing

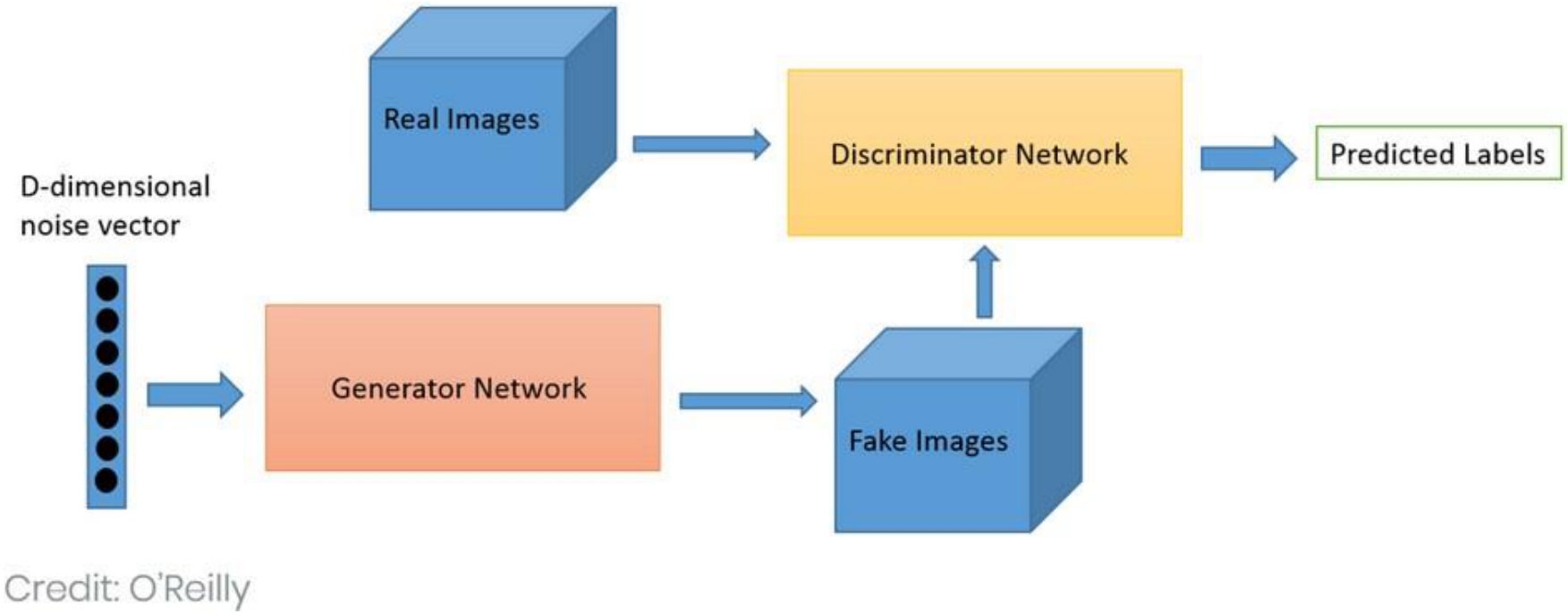




Generative Adversary Networks (GANs)

You can think of a GAN as the opposition of a counterfeiter and a cop in a game of cat and mouse, where *the counterfeiter is learning to pass false notes*, and *the cop is learning to detect them*. Both are dynamic; i.e. the cop is in training, too, and each side comes to learn the others methods in a constant escalation

Generative Adversary Networks (GANs)



Credit: O'Reilly

A generator ("the artist") learns to create images that look real, while a discriminator ("the art critic") learns to tell real images apart from fakes

Generative Adversary Networks (GANs)

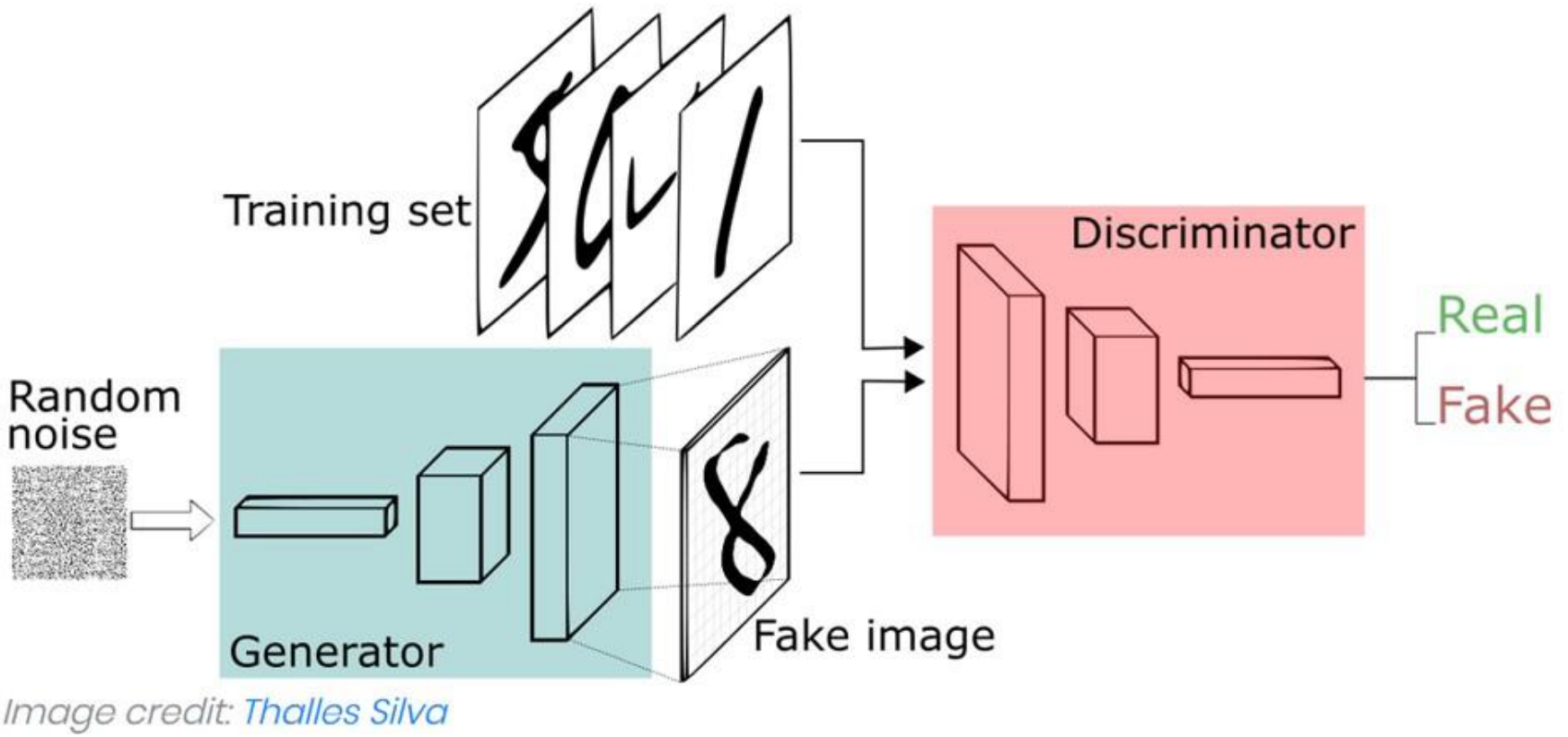
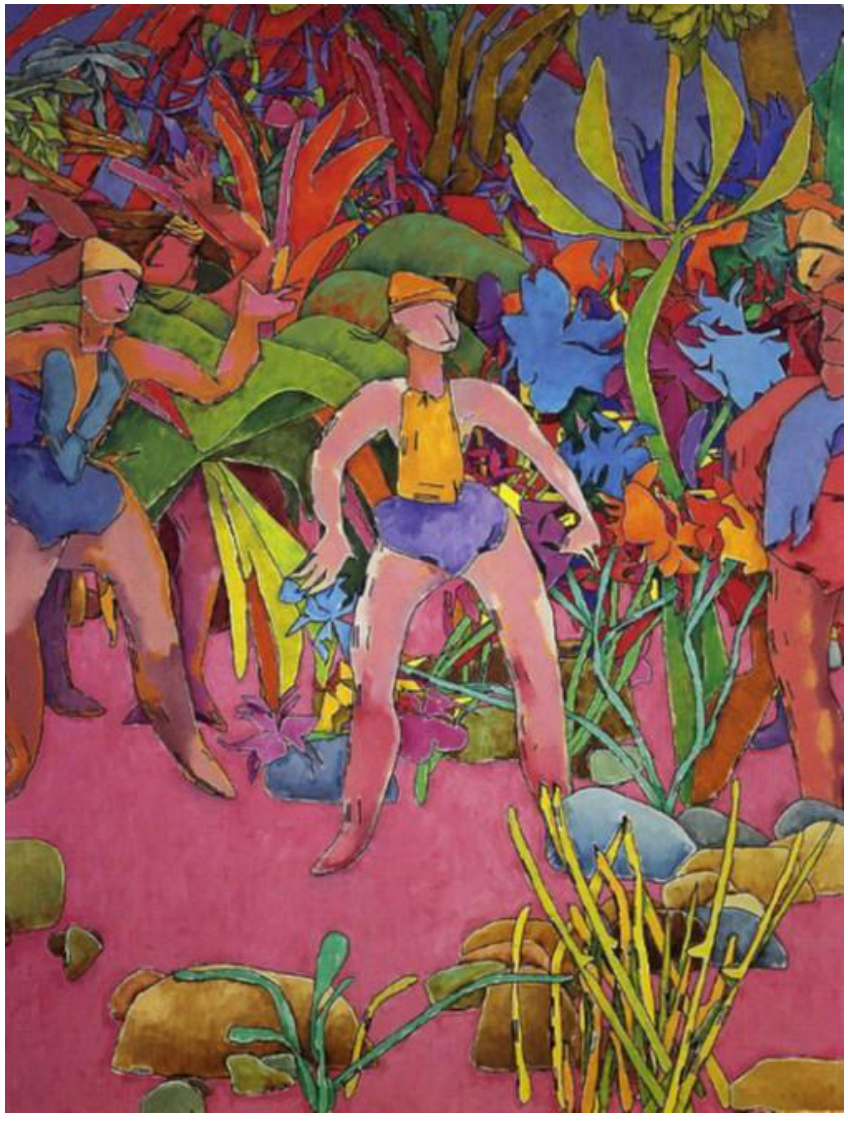


Image credit: Thalles Silva

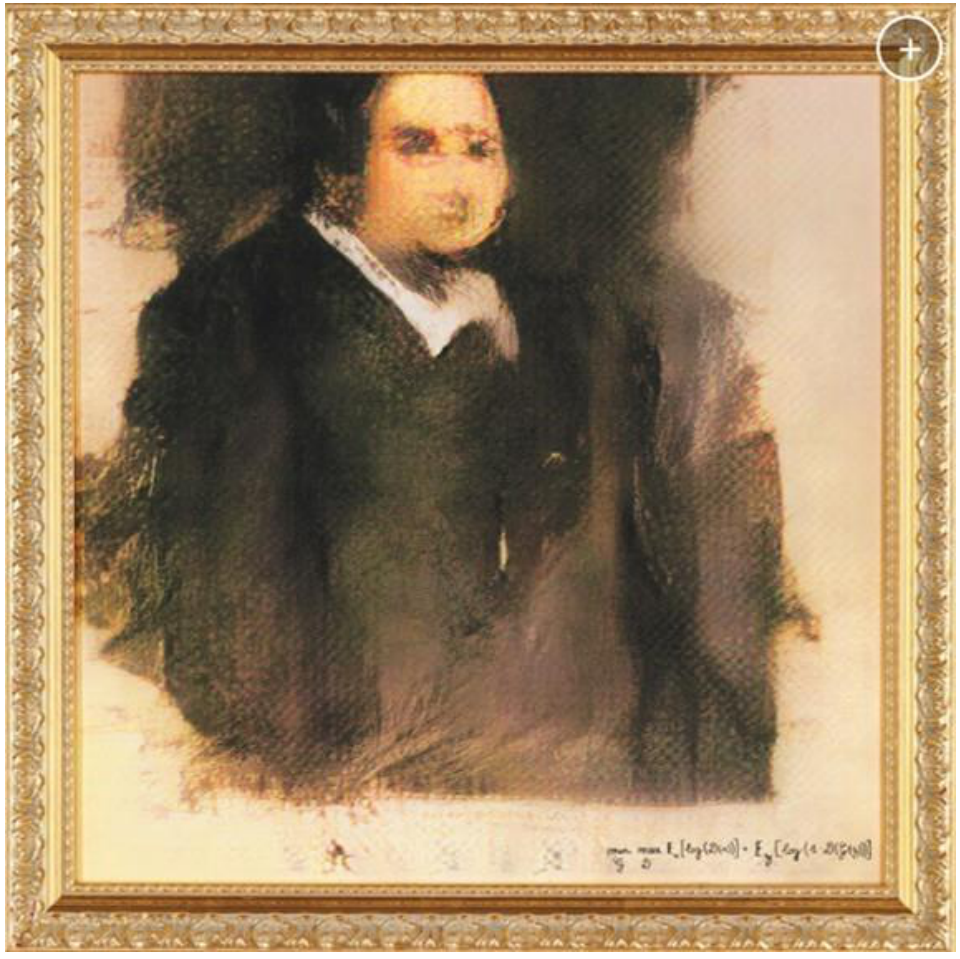
GAN Art

GAN Art



GAN Art

GAN Art



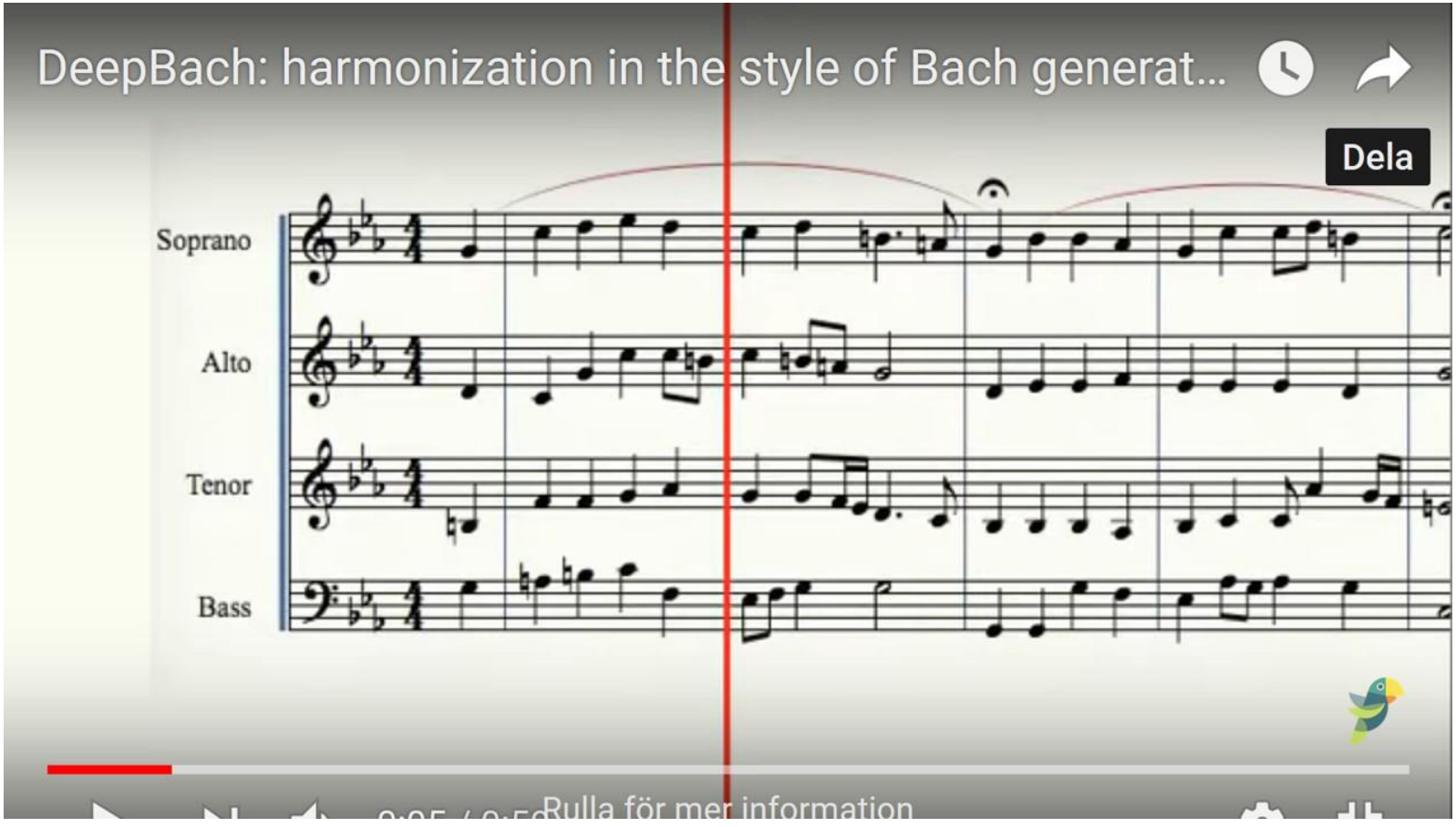
Trained on works by Dutch masters. Sold for $432,500 at Christie's in 2018

GAN photos

GAN photos

GAN photos

GAN Music


DeepBach: harmonization in the style of Bach generat...
Dela
Soprano
Alto
Tenor
Bass
Rulla för mer information

GAN Poetry

Night sky and fields of black

A flat cracked surface and a building

She reflects an image in a glass

She does not see, she does not watch

CHALMERS
UNIVERSITY OF TECHNOLOGY

MACHINE LEARNING AND NEURAL NETWORKS

PART TWO

Bastiaan Bruinsma

March 1, 2022

Chalmers University of Technology