UNIVERSITY OF CAMBRIDGE

Generative Adversarial Networks

Petar Veličković

Artificial Intelligence Group
Department of Computer Science and Technology, University of Cambridge, UK

Introduction

- In this talk, I will guide you through a high level overview of **Generative Adversarial Networks** (GANs)—*one of the most popular ideas to hit deep learning in the past decade*.
- This will involve a journey through the essentials of how neural networks normally work. . .
- . . . followed by a dive into *turning horses into zebras*, *generating fake celebrities*, and anecdotes about how **pints can lead to wonderful science**.

Motivation: notMNIST

- Which characters do you see? (*How* did you conclude this?)

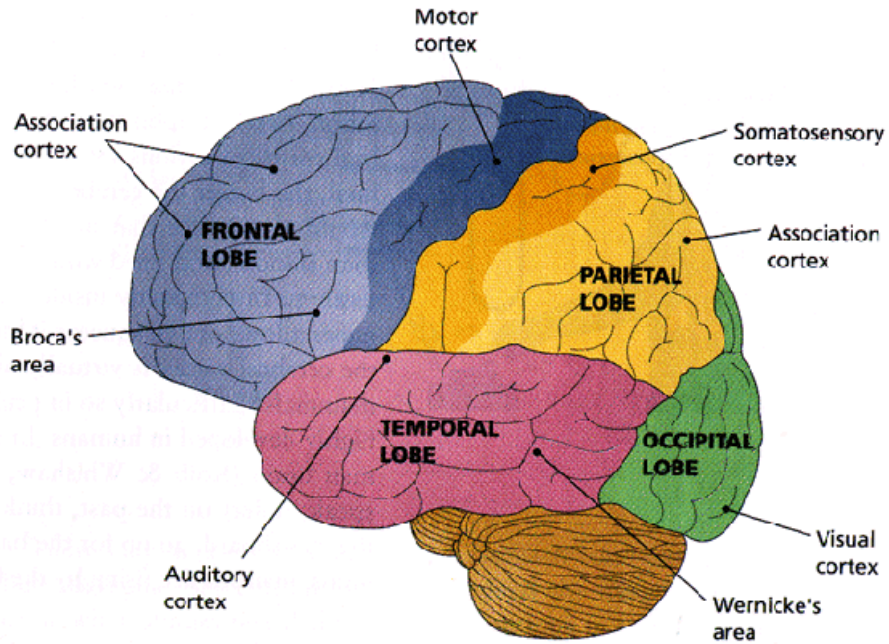
- Imagine someone asked you to write a program that recognises characters from arbitrary glyphs. . .

Intelligent systems

- Although the previous task was likely *simple* to you, you (probably) couldn't turn your thought process into a concise sequence of instructions for a program!
- Unlike a *"dumb"* program (that just blindly executes preprogrammed instructions), you've been exposed to a lot of `A` characters during your lifetimes, and eventually *"learnt"* the complex features making something an `A`!
- Desire to design such systems (capable of *generalising* from past experiences) is the essence of *machine learning*!
  - How many such systems do we know from nature?

Specialisation in the brain

- We know that *different parts* of the brain perform *different tasks*:

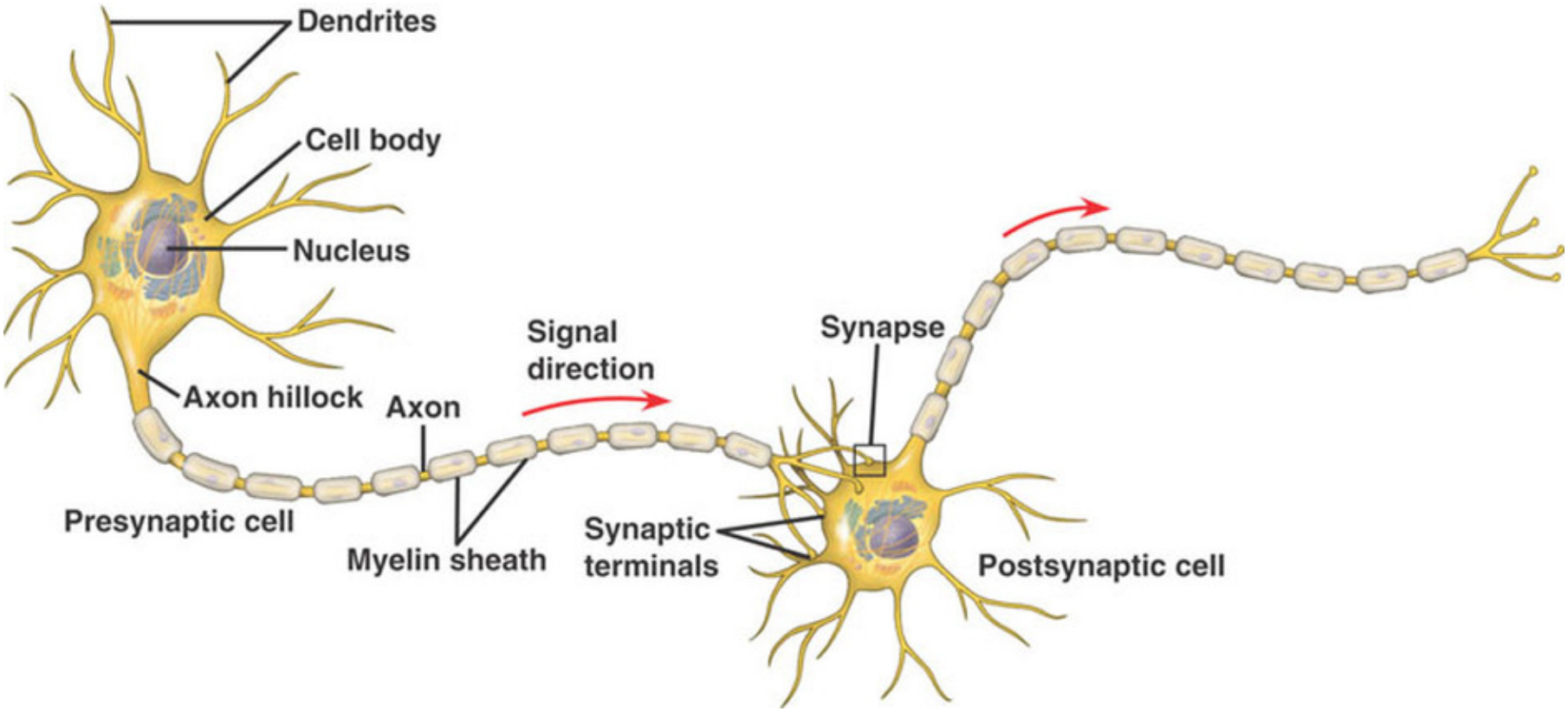


- There is increasing evidence that the brain:
  - Learns from *exposure to data*;
  - Is *not* preprogrammed!

Brain & data

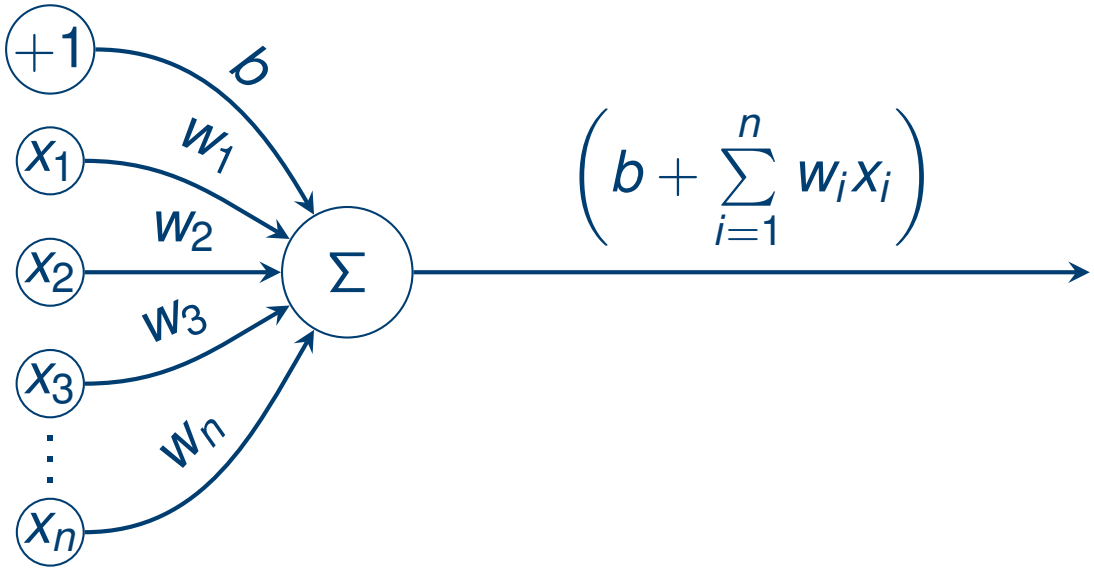
- The majority of what we know about the brain comes from studying *brain damage*:
  - Rerouting visual inputs into the auditory region of baby ferrets makes this region capable of dealing with visual input!
  - As far as we know (for now), the modified region works equally good as the visual cortex of healthy ferrets!
- If there are no major biological differences in learning to process different kinds of input. . .
- $\implies$ the brain likely uses a *general learning algorithm*, capable of adapting to a wide spectrum of inputs.
- We'd very much like to capture this algorithm!

A *real* neuron!



An *artificial* neuron!

Within this context sometimes also called a *perceptron* (. . .)



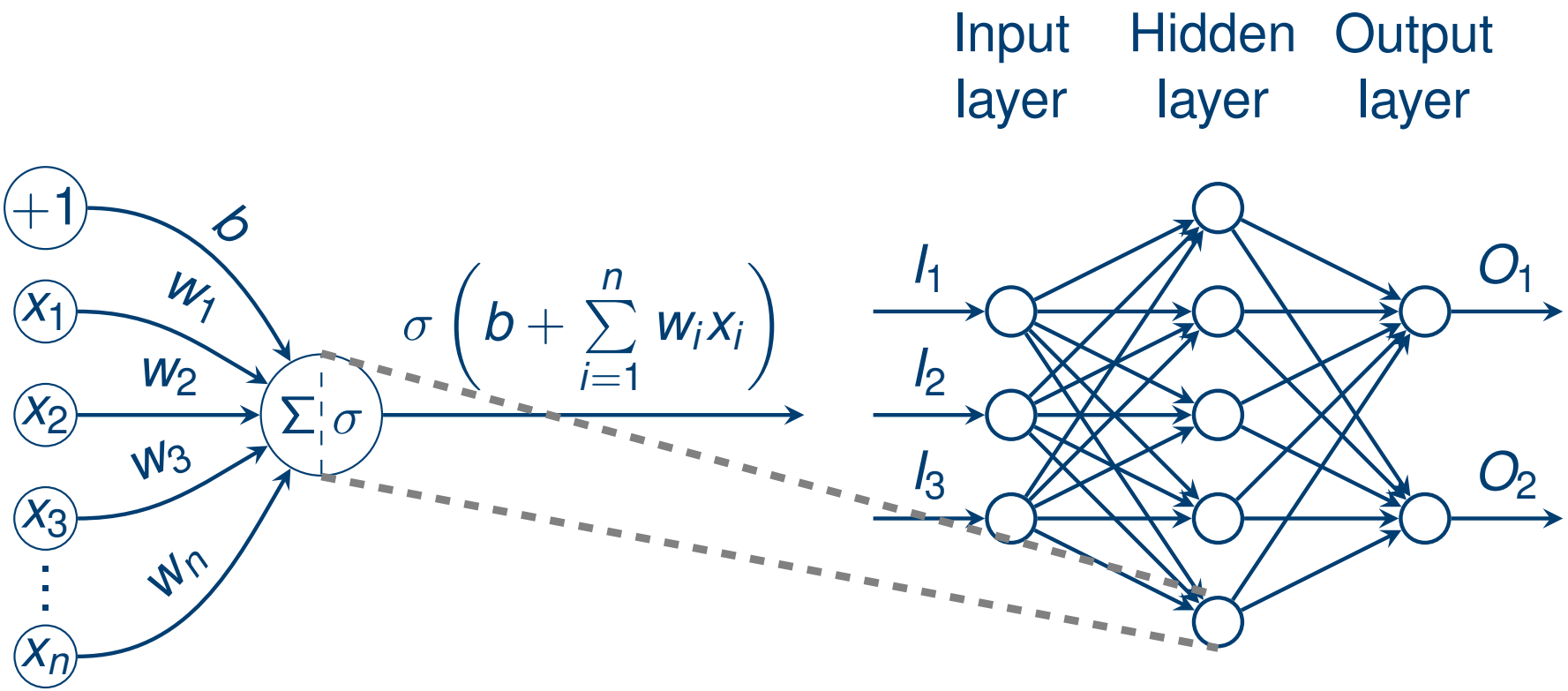
We *train* the neuron by exposing it to many $(\vec{x}, y)$ examples, and then adjusting $\vec{w}$ and b to make its output close to y when given $\vec{x}$ as input.

Neural networks and deep learning

- It is easy to extend a single neuron to a *neural network*—simply connect outputs of neurons to inputs of other neurons.
- Typically we organise neural networks in a sequence of *layers*, such that a single layer only processes output from the previous layer.
- By extension from a single neuron, we train the neural network by exposing it to many input-output pairs, and appropriately adjusting the parameters of all the neurons.

Multilayer perceptrons

The most potent feedforward architecture allows for full connectivity between layers—sometimes also called a *multilayer perceptron*.



Neural network depth

- I'd like to highlight a specific parameter: *the number of hidden layers*, i.e. *the network's depth*.
- What do you think, how many hidden layers are *sufficient* to learn any "useful" function?

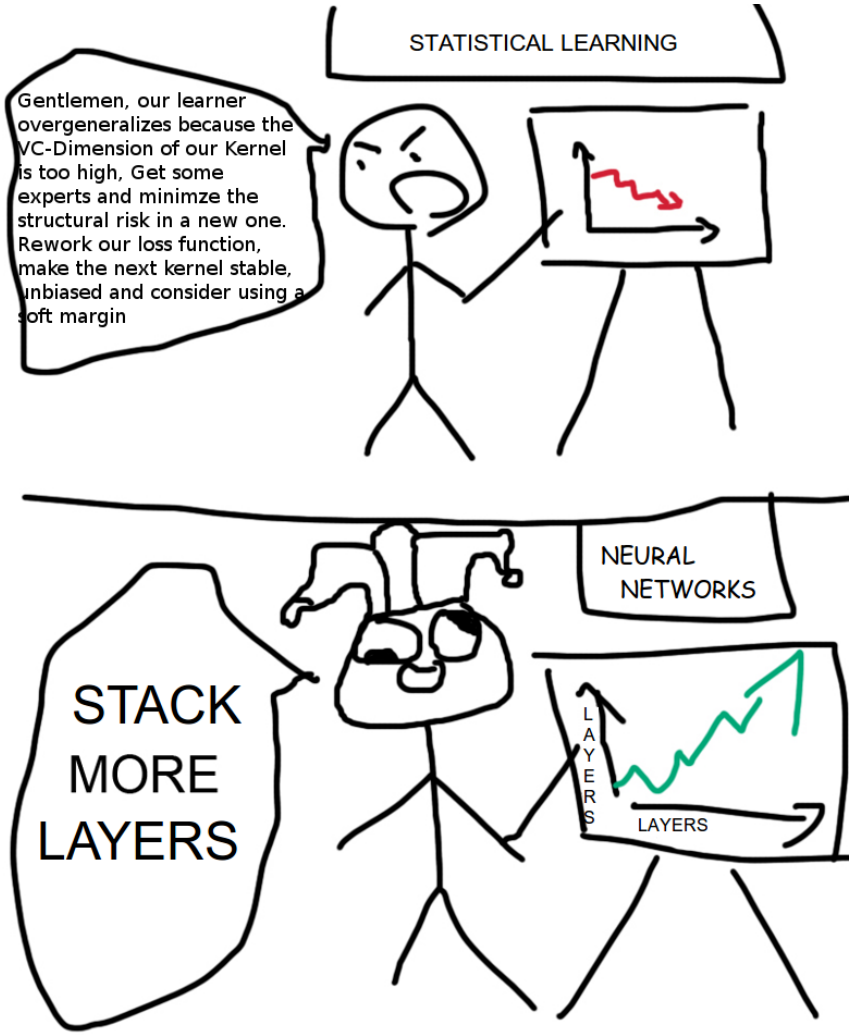
Neural network depth

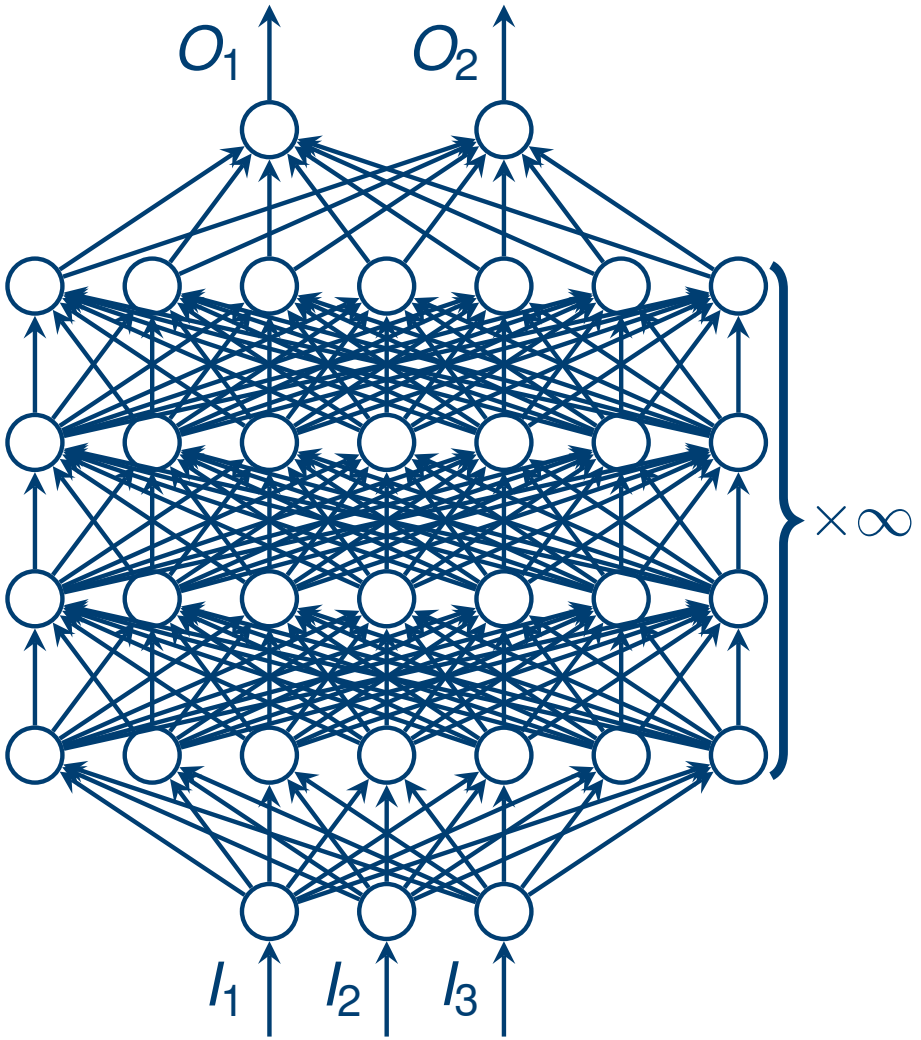
- I'd like to highlight a specific parameter: *the number of hidden layers*, i.e. *the network's depth*.
- What do you think, how many hidden layers are *sufficient* to learn any "useful" function?
- One! (*Cybenko's theorem*, 1989.)

Neural network depth

- I'd like to highlight a specific parameter: *the number of hidden layers*, i.e. *the network's depth*.
- What do you think, how many hidden layers are *sufficient* to learn any "useful" function?
- One! (*Cybenko's theorem*, 1989.)
- However, the number of neurons in this layer would need to be **astronomical** for almost all problems we care about.
- *We must go deeper. . .*
  - Every network with $>$ 1 hidden layer is considered *deep*!
  - Today's *state-of-the-art* networks often have over 150 layers.

Deep neural networks





Quiz: What do we have here?

DeepBlue vs. AlphaGo

- Main idea (roughly) the same: *assume that a grandmaster is only capable of thinking k steps ahead*—then generate a (near-)optimal move when considering $k' > k$ steps ahead.
  - DeepBlue does this exhaustively, AlphaGo sparsely (discarding many "highly unlikely" moves).
- One of the key issues: when *stopping exploration*, how do we determine the *advantage* that player 1 has?

DeepBlue: Gather a team of *chess experts*, and define a function $f : \mathit{Board} \to \mathbb{R}$, to define this advantage.

AlphaGo: Feed the raw state of the board to a deep neural network, and have it *learn* the advantage function *by itself*.

- This highlights an important *paradigm shift* brought about by deep learning...

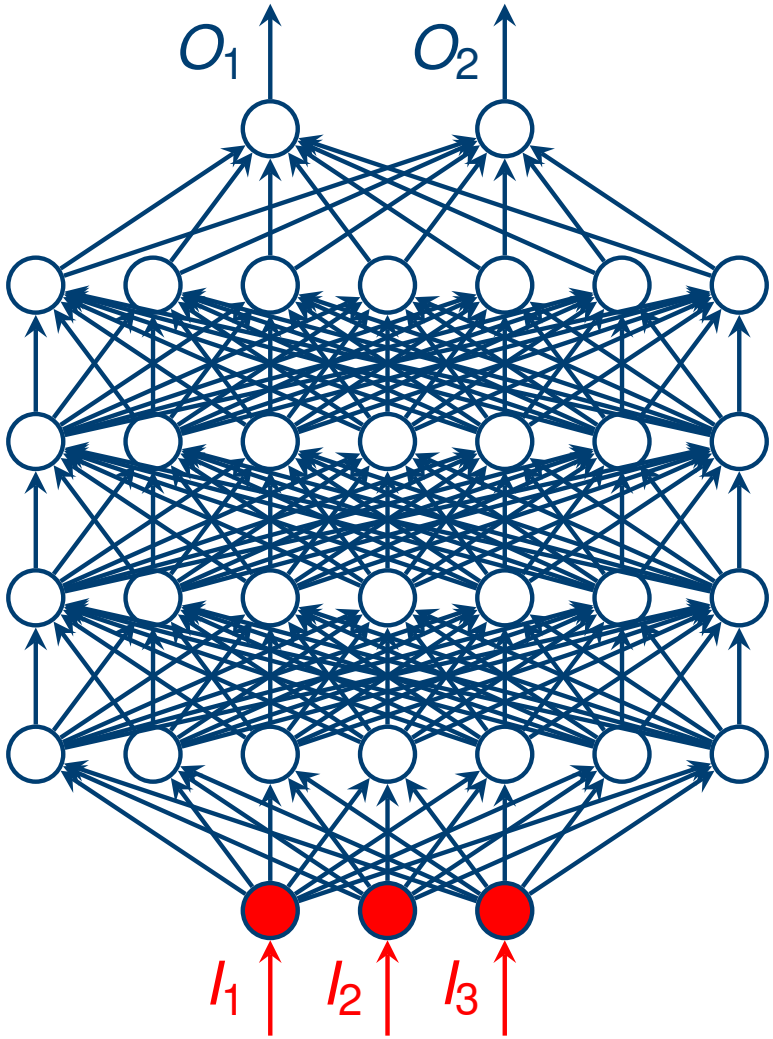
Feature engineering

- Historically, machine learning problems were tackled by defining a set of *features* to be manually extracted from raw data, and given as inputs for *"shallow"* models.
  - Many scientists built *entire PhDs* focusing on features of interest for just one such problem!
  - Generalisability: very small (often zero)!
- With deep learning, the network *learns the best features by itself*, directly from *raw data*!
  - For the first time connected researchers from fully distinct areas, e.g. *natural language processing* and *computer vision*.
  - $\Longrightarrow$ a person capable of working with deep neural networks may readily apply their knowledge to create state-of-the-art models in virtually any domain (assuming a large dataset)!

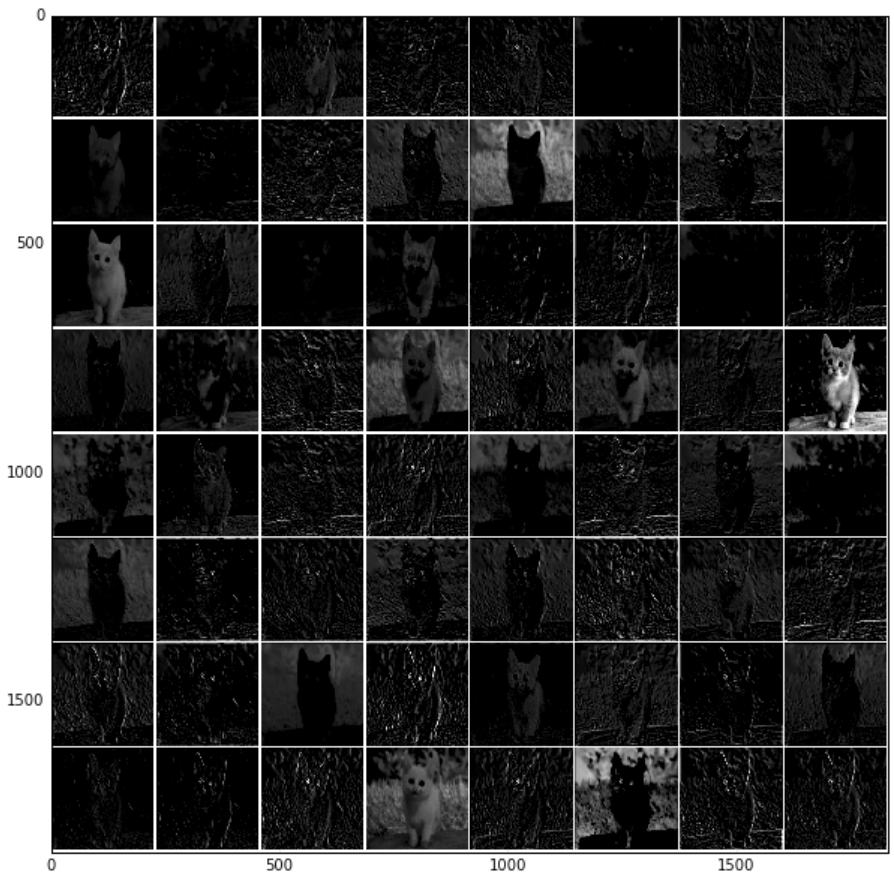
Representation learning

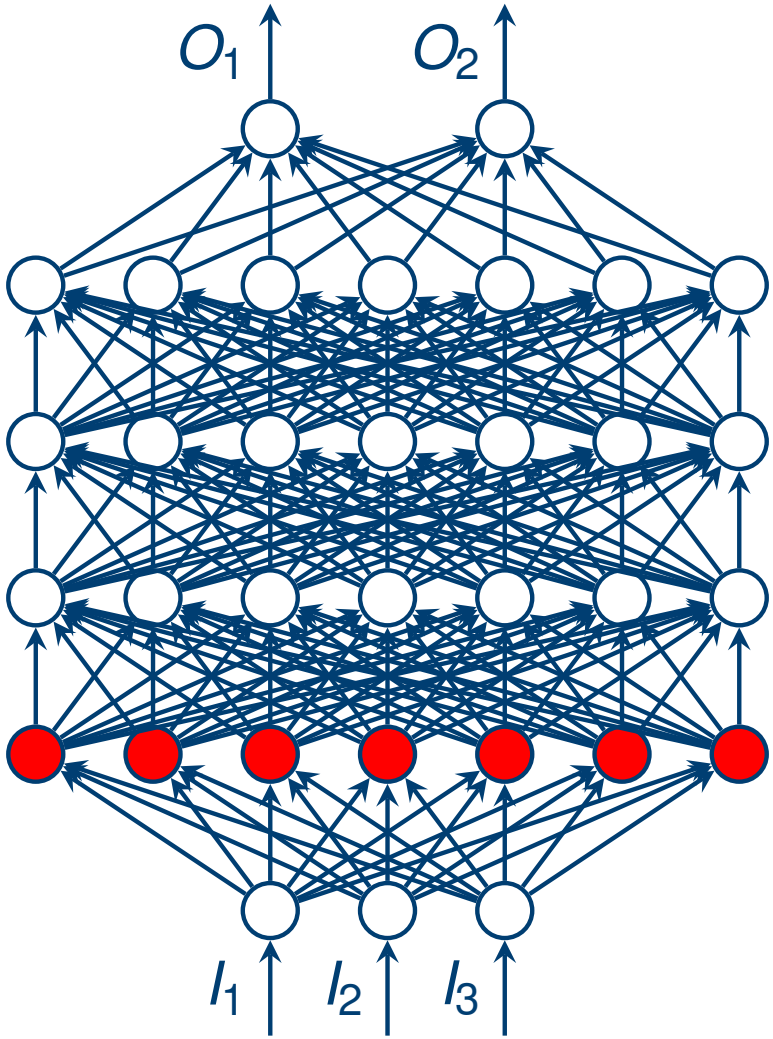
- As inputs propagate through the layers, the network captures more complex *representations* of them.
- It will be extremely valuable for us to be able to reason about these representations!
- Typically, models that deal with *images* will tend to have the best visualisations.

Passing data through the network: *Input*

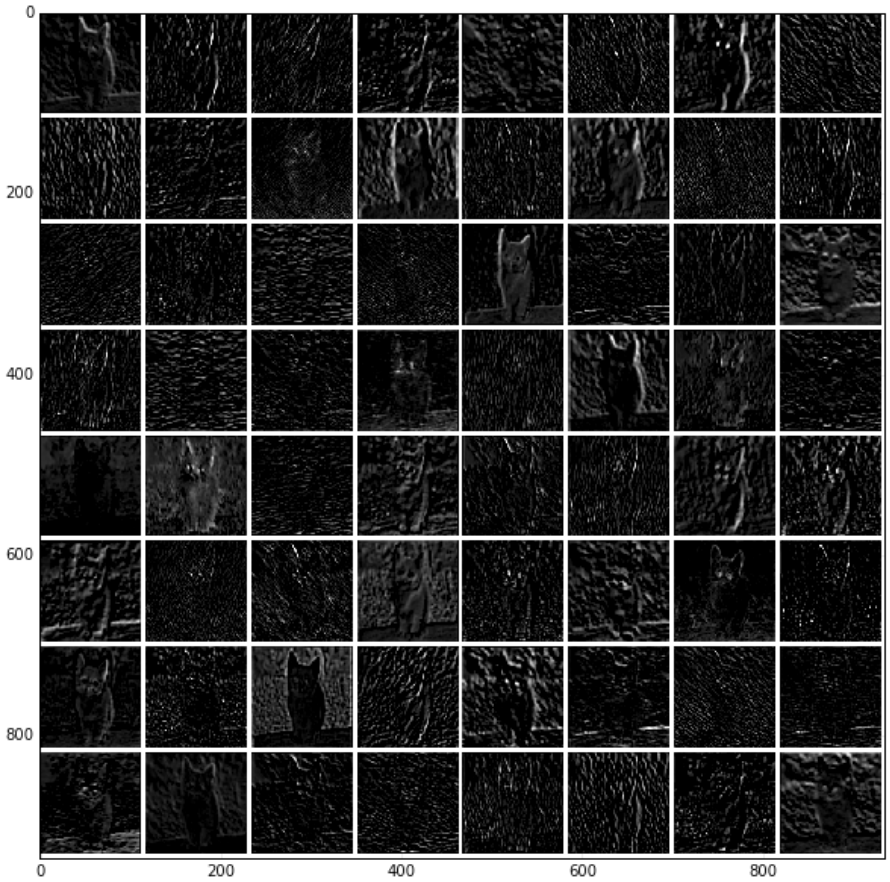


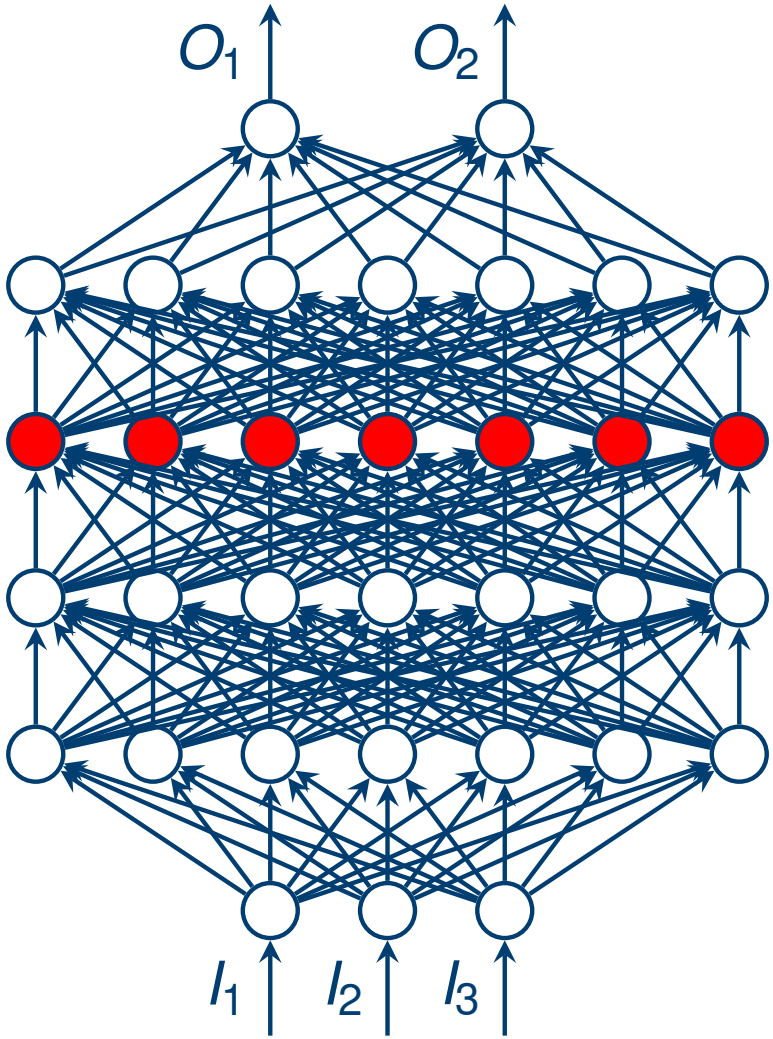
Passing data through the network: *Shallow layer*



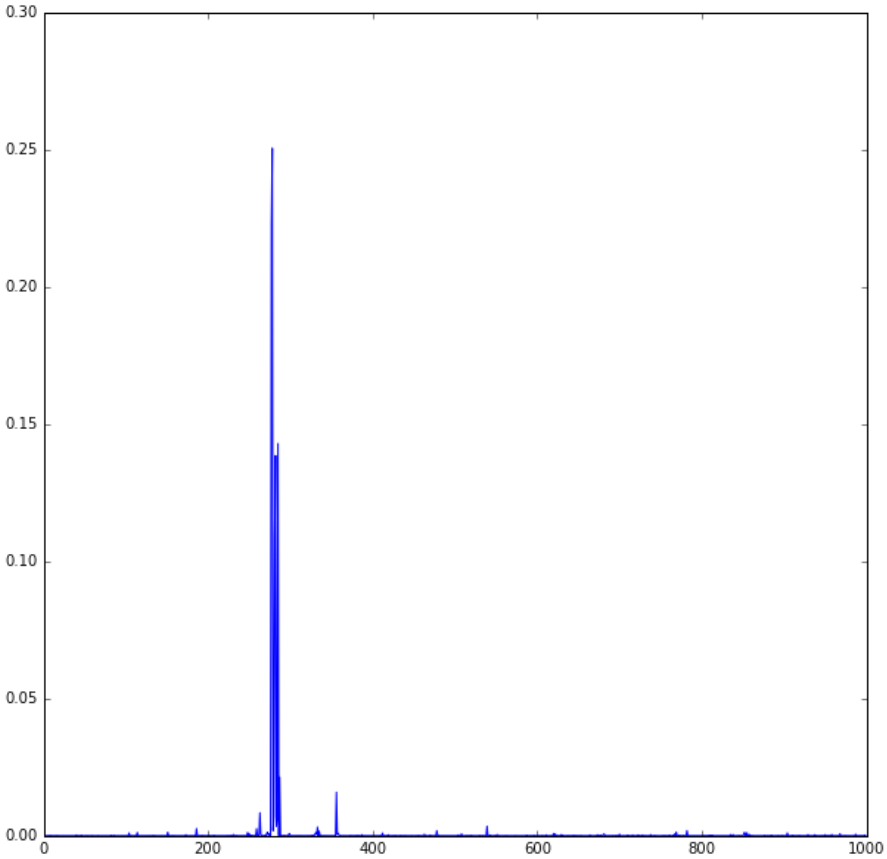


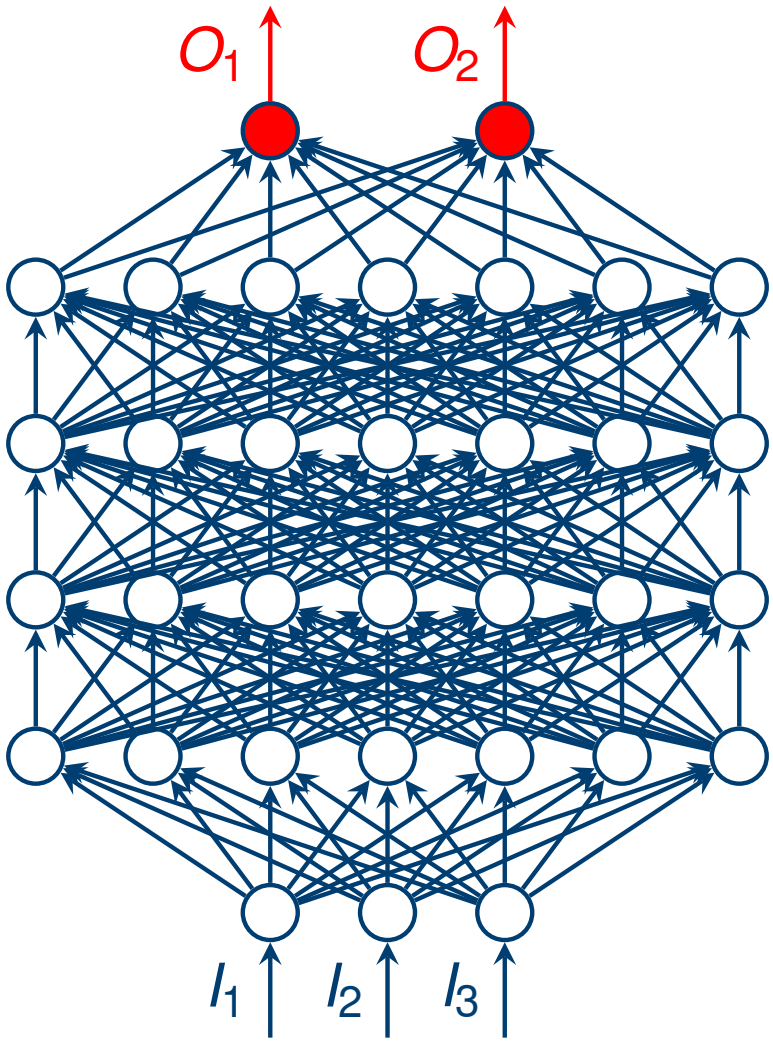
Passing data through the network: *Deep layer*



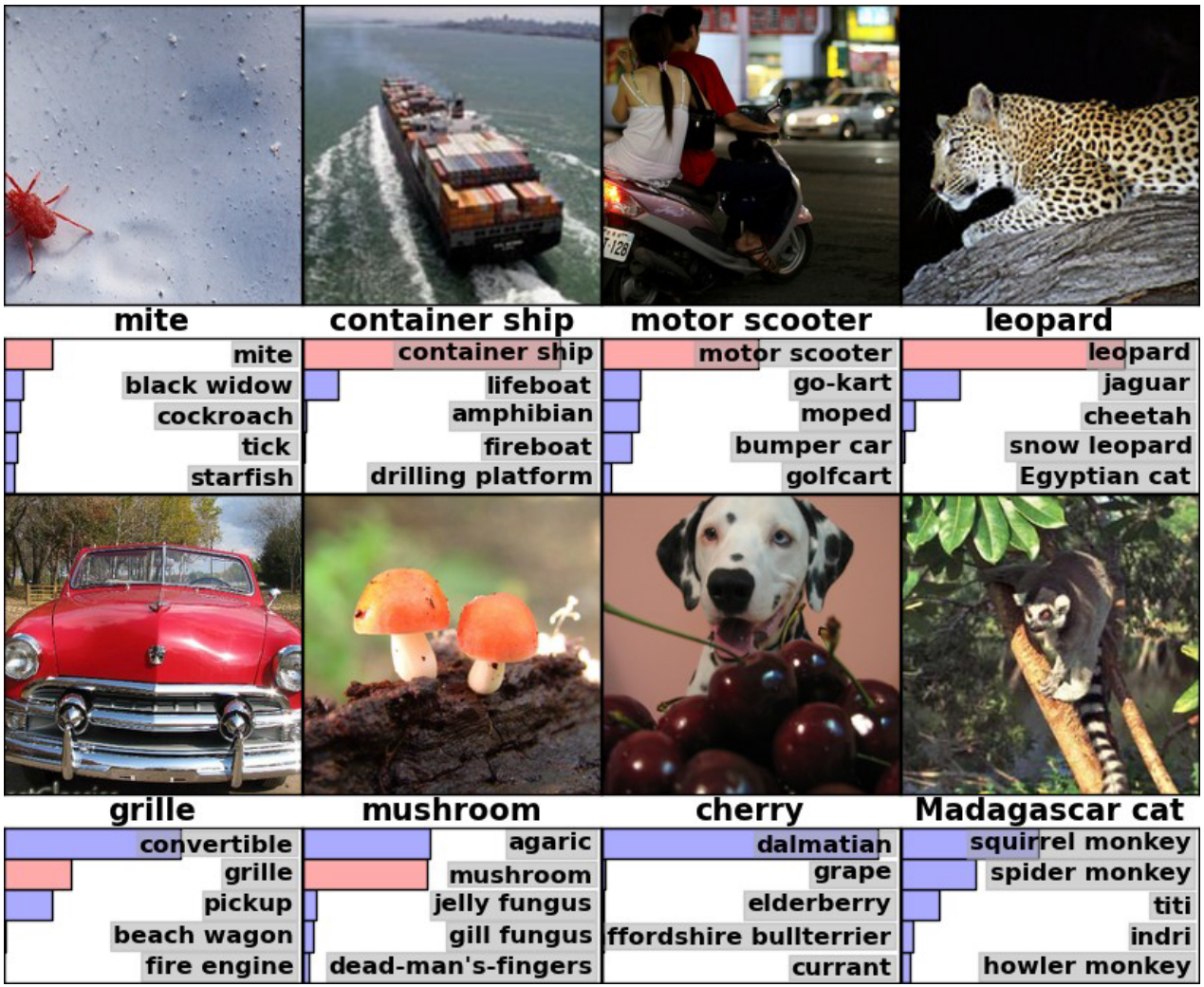


Passing data through the network: *Output*

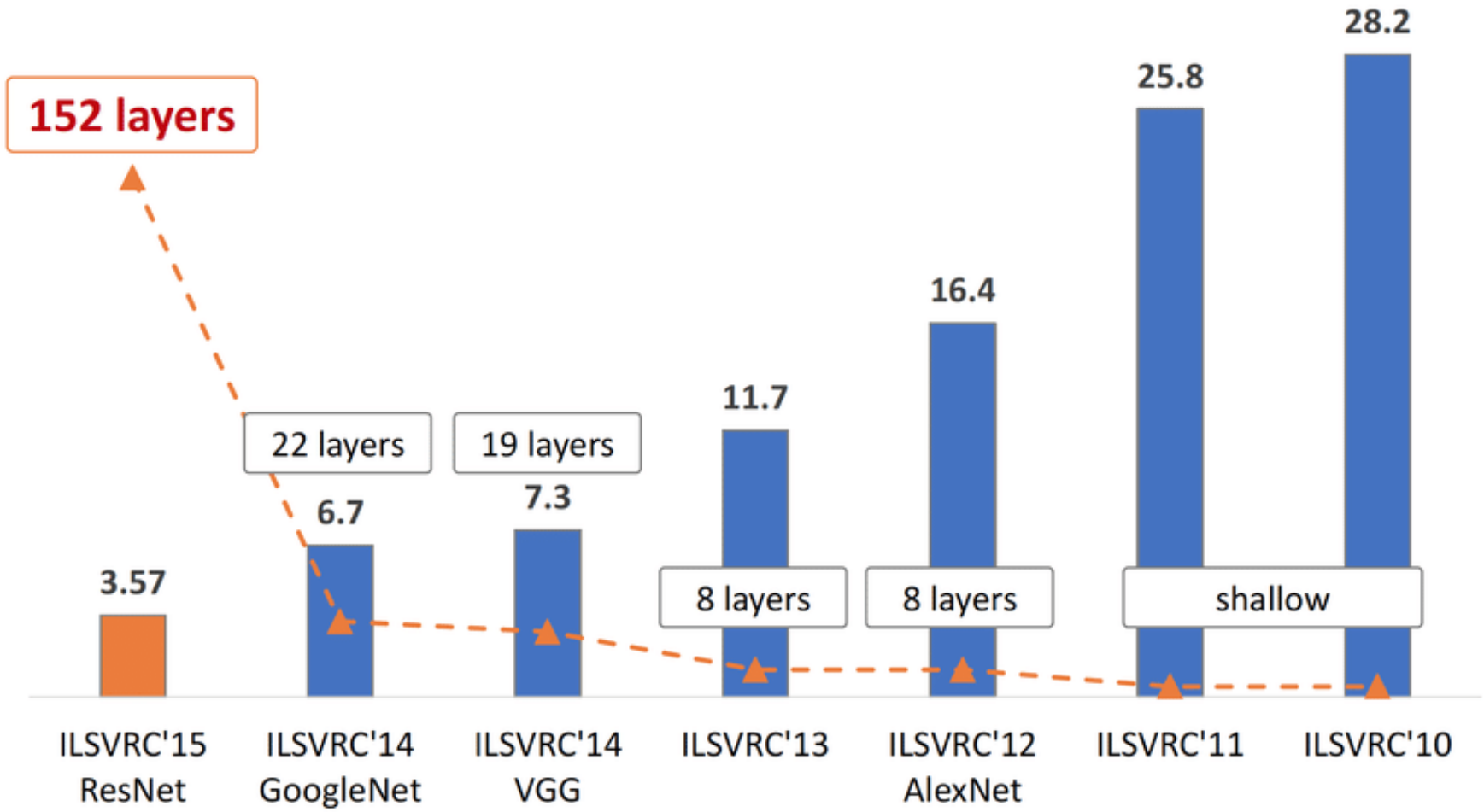




We're good at detecting what's in an image. . .



. . . even exceeding human performance. . .

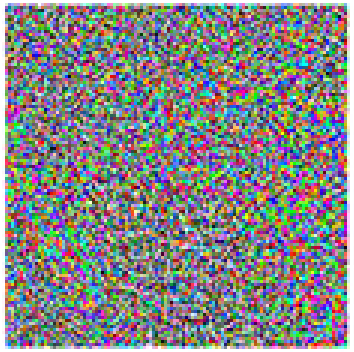


. . . or are we???

x

"panda"
57.7% confidence

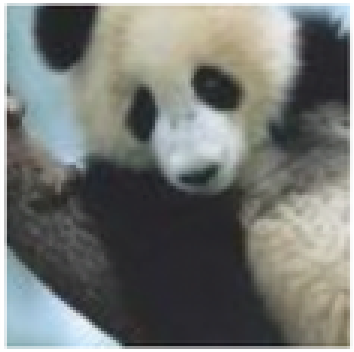
$+ .007 \times$



$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$



$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

The importance of *understanding* data

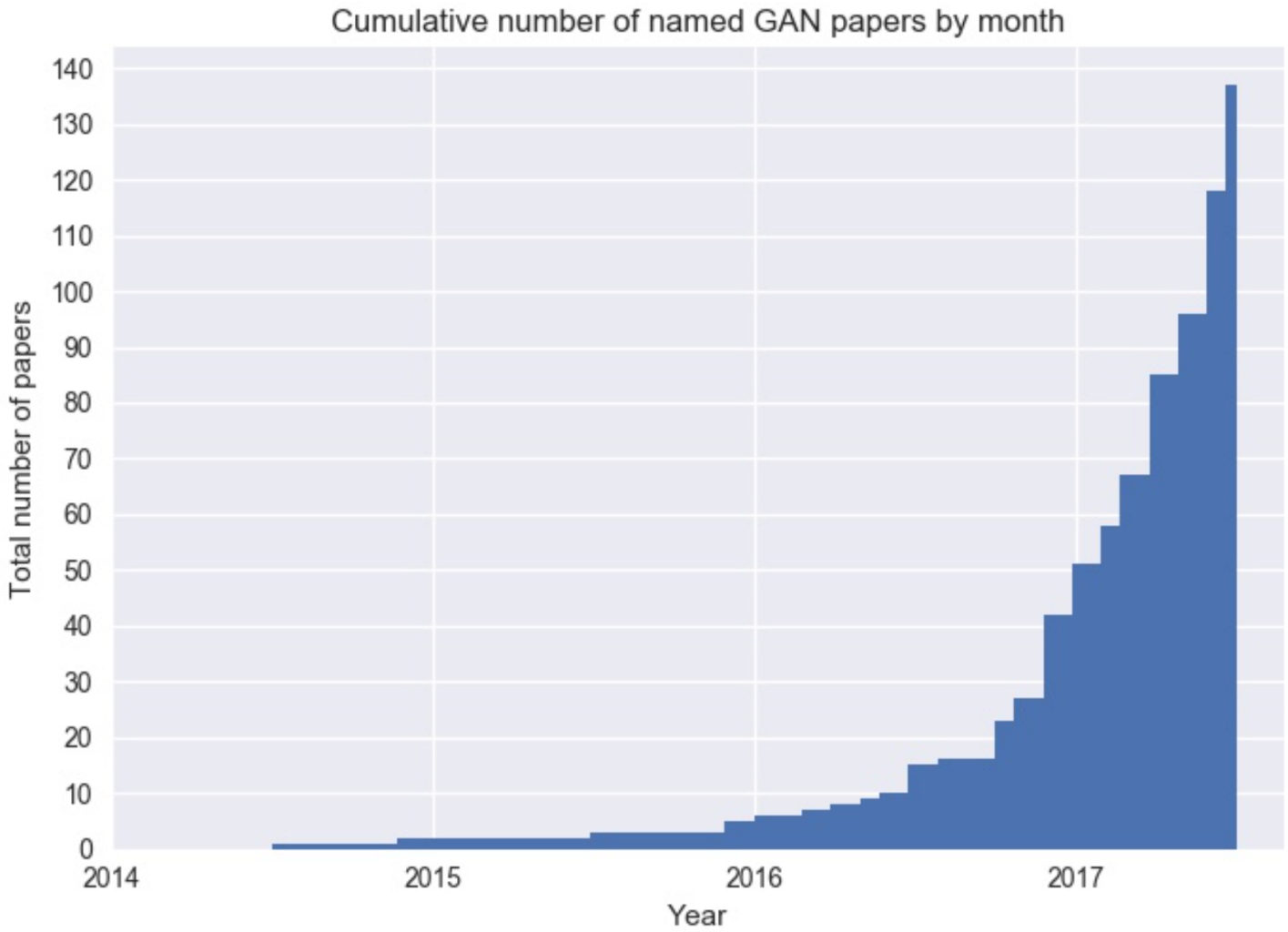
Generative modelling

“What I cannot create, I do not understand.” —Richard Feynman

Generative Adversarial Networks

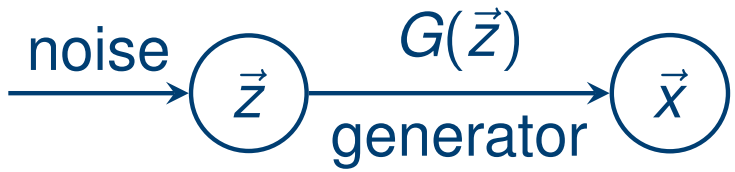
- *"The most important one, in my opinion, is adversarial training (also called GAN for **Generative Adversarial Networks**). This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion."*
  —Yann LeCun
  (Director of Research, *Facebook AI Research*)

GANs are **everywhere**



The **generator**

- Imagine that we had a neural network, G, capable of generating new data from random inputs $\vec{z}$...



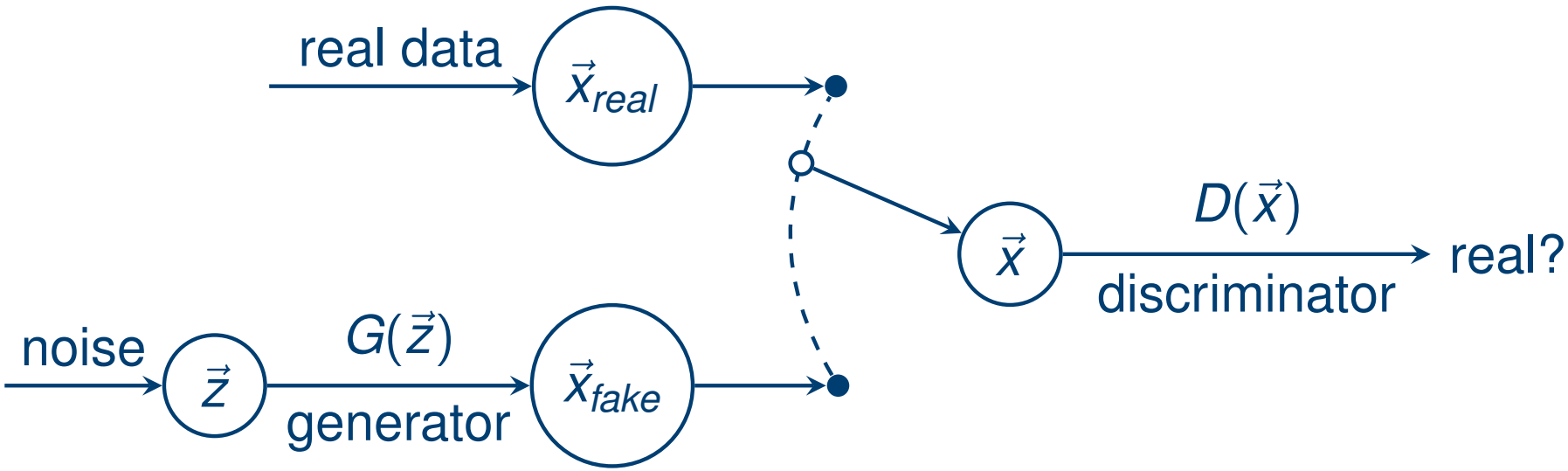
  We will call this neural network the **generator**.

- What we want to do is make the output of this generator match the properties of **real data**.
- But how can we check if a synthetic input has properties similar to real data?

The **discriminator**

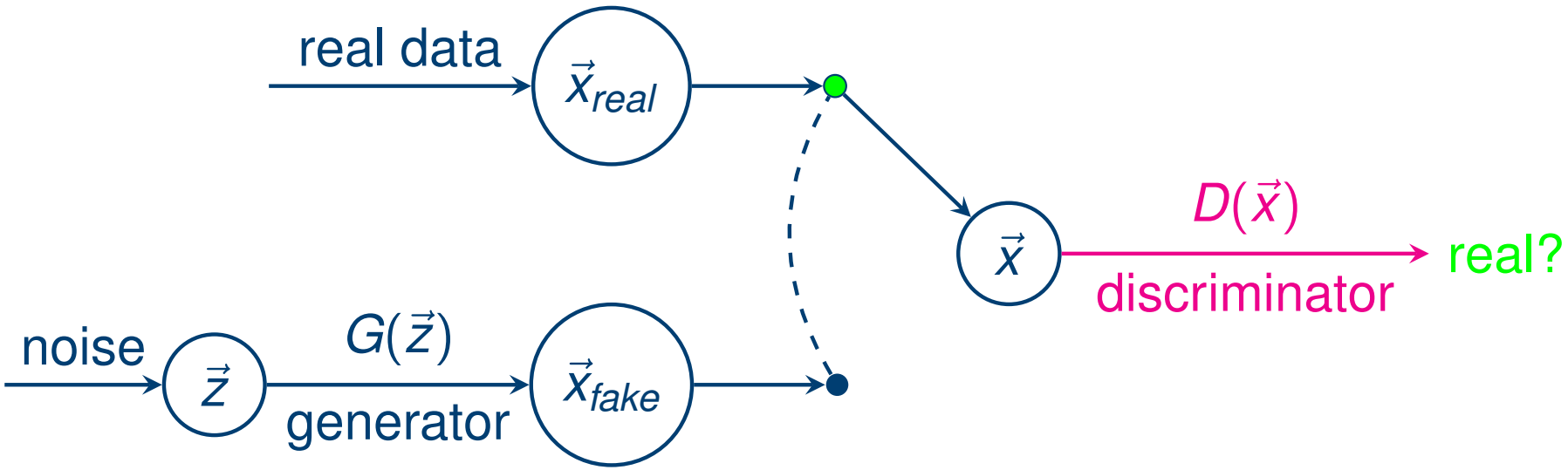
- Checking for properties by looking over full data (e.g. individual image pixels) is **hard**.
- We can utilise a *neural network* to extract underlying features from inputs, and examine those.
- If these features are “good enough”, the network should be capable of *telling real and synthetic inputs apart*!
- Call this network the **discriminator**, $D(\vec{x})$.
- Essentially, a *binary classifier*, telling whether $\vec{x}$ is real or synthetic. **N.B.** The discriminator effectively **specifies the *error*** that we want the generator to minimise!

The GAN framework (Goodfellow *et al.*, 2014)



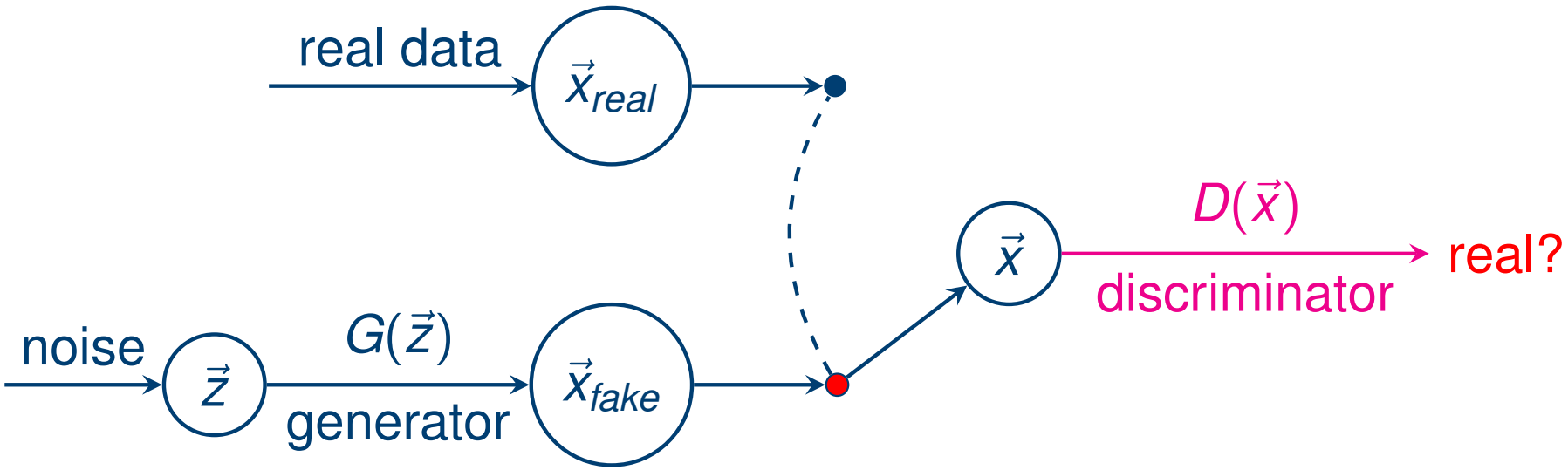
Two neural networks *playing a game*. . .
Alternate updating their weights; hopefully they *improve* together!

The GAN framework—update step 1



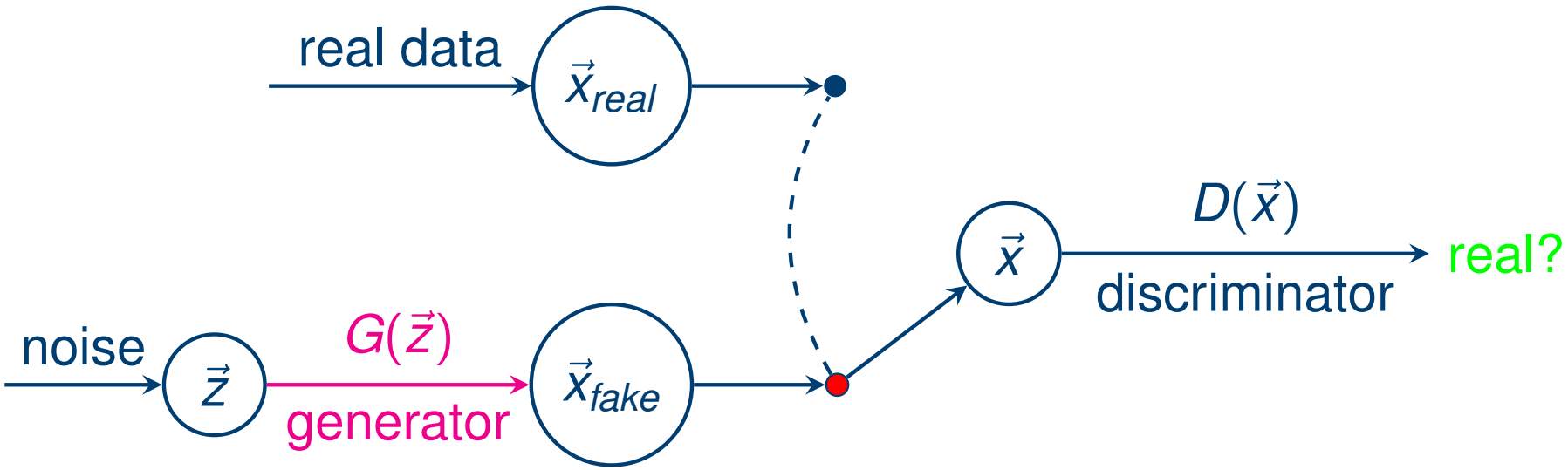
Train discriminator to say '**real**' on real data.

The GAN framework—update step 2



Train discriminator to say **'fake'** on fake data.

The GAN framework—update step 3



Train generator to make discriminator say **'real'** on fake data.

The desired final outcome

$$\min_G \max_D V(D,G) = \mathbb{E}_{\vec{x}\sim p_{data}(\vec{x})}[\log D(\vec{x})] + \mathbb{E}_{\vec{z}\sim p_\theta(\vec{z})}[\log(1 - D(G(\vec{z})))]$$

- Assuming everything goes well, concluding the training process we obtain **two** very useful networks!
  - The **generator**, $G(\vec{z})$, becomes capable of generating extremely useful synthetic examples.
  - The **discriminator**, $D(\vec{x})$, becomes a high-quality **feature extractor** from data.
- Let's see how good these synthetic examples from $G(\vec{z})$ actually get...

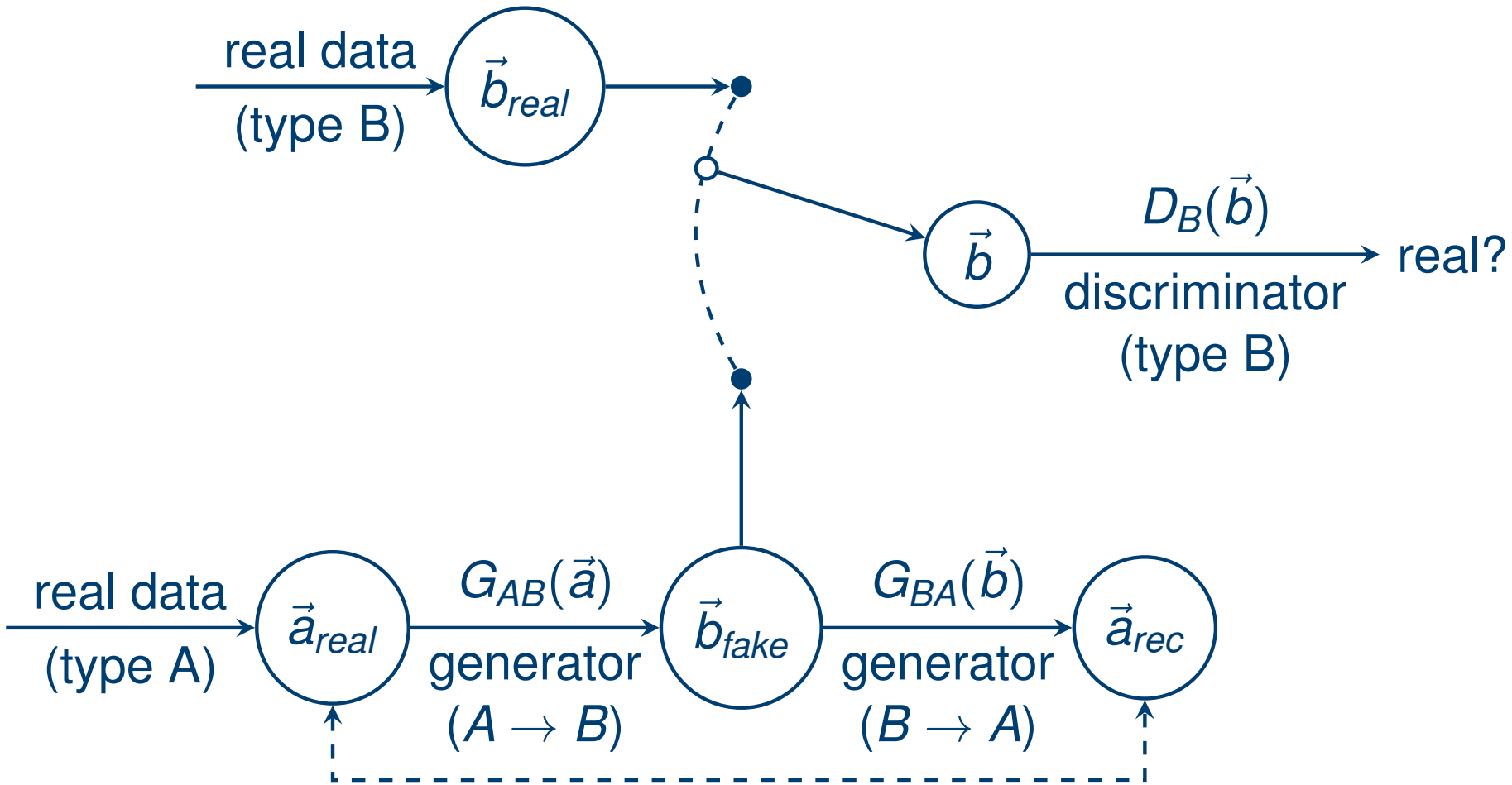
Generating fake celebrities (*Karras et al., 2018*)

A step further—domain transfer

- Similar ideas can be applied in order to *transform data from one type to another*...
- We learn **two generators**, G_{AB} and G_{BA}, to transfer inputs from type A to type B, and vice-versa.
- We also learn the **two discriminators**, D_A and D_B.
- Besides the usual GAN game, we also can enforce *cycle consistency*: $G_{BA}(G_{AB}(\vec{a})) \sim \vec{a}$.
  - This means that if we transform an input of type A to type B, then transform the result back into type A, we should be able to recover the original input.

We arrive at the remarkably powerful *CycleGAN* model...

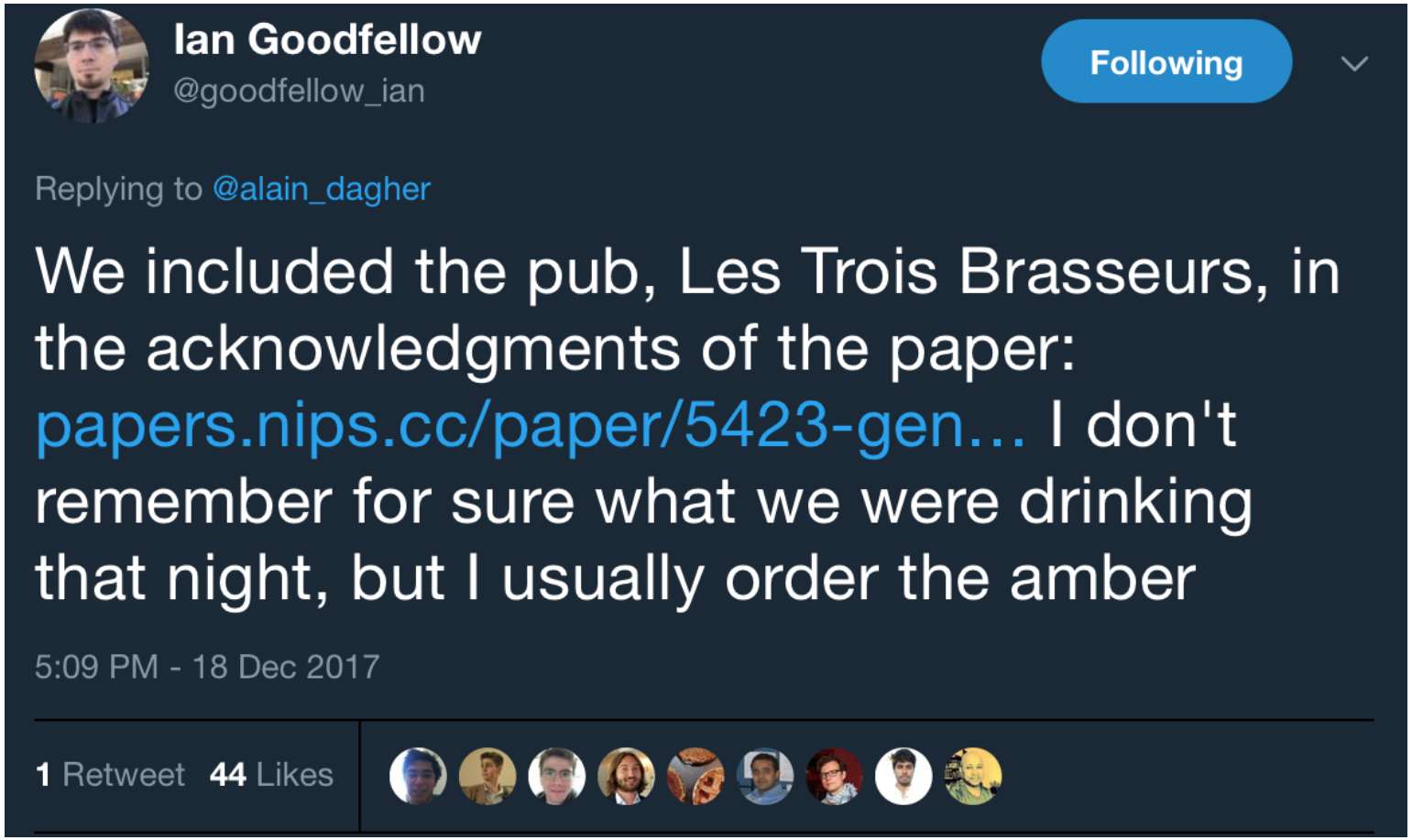
Domain transfer: *CycleGAN* (*Zhu et al., 2017*)



Domain transfer: *CycleGAN* (*Zhu et al., 2017*)

Where did this all begin?

A true *pint of science*. . .



. . . credited in the paper.

Acknowledgments

We would like to acknowledge Patrice Marcotte, Olivier Delalleau, Kyunghyun Cho, Guillaume Alain and Jason Yosinski for helpful discussions. Yann Dauphin shared his Parzen window evaluation code with us. We would like to thank the developers of Pylearn2 [11] and Theano [6, 1], particularly Frédéric Bastien who rushed a Theano feature specifically to benefit this project. Arnaud Bergeron provided much-needed support with LaTeX typesetting. We would also like to thank CIFAR, and Canada Research Chairs for funding, and Compute Canada, and Calcul Québec for providing computational resources. Ian Goodfellow is supported by the 2013 Google Fellowship in Deep Learning. Finally, we would like to thank Les Trois Brasseurs for stimulating our creativity.

An overview of historical deep learning ideas

- Initially, we needed to extract hand-crafted features before applying a machine learning model to them.
  - **Deep neural networks** can perform feature extraction by themselves.
- Then, we needed to decide on how to estimate the *error* of our network's prediction.
  - **GANs** use a neural network (the *discriminator*) to compute this!
- We need to figure out a correct way to reduce this error.
  - **Learn how to learn**?

Thank you!

Questions?

petar.velickovic@cst.cam.ac.uk