Deep Learning Limitations and New Frontiers

Alexander Amini

T-shirts!

MIT 6.S191
THINK
DEEPER
Introduction To
Deep Learning

MIT Introduction To
Deep Learning
Google
IBM
Tencent
AI Lab
NVIDIA
Microsoft

Final Class Project

Option 1: Proposal Presentation

- **Groups of 3 or 4**
- Present a novel deep learning research idea or application
- 1 slide, 1 minute
- List of example proposals on website: [CLICK HERE]
- Presentations on **Friday, Feb 2**
- Submit groups by **TODAY at 9pm** to be eligible
- Submit slide by **Thursday 9pm** to be eligible

- Judged by a panel of industry judges
- Top winners are awarded:

1x NVIDIA Titan Xp
MSRP: $1200

2x NVIDIA Titan TX2
MSRP: $600

3x Google Home
MSRP: $300

Final Class Project

Option 1: Proposal Presentation

- **Groups of 3 or 4**
- Present a novel deep learning research idea or application
- 1 slide, 1 minute
- List of example proposals on website: CLICK HERE
- Presentations on **Friday, Feb 2**
- Submit groups by **TODAY at 9pm** to be eligible
- Submit slide by **Thursday 9pm** to be eligible

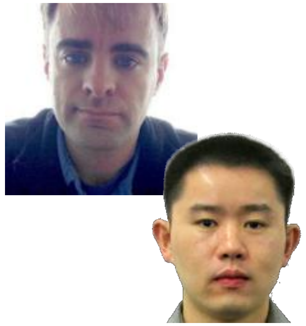
Option 2: Write a 1-page review of a deep learning paper (single spaced)

- Suggested papers listed on website: CLICK HERE
- Grade is based on clarity of writing and technical communication of main ideas

Thursday: Deep Learning in Industry

Urs Muller
Chief Architect Autonomous Driving
- End-to-End Learning for Self Driving Cars

NVIDIA



D Sculley
- Issues with Image Classification

Shanqing Cai
- Faster TensorFlow Development with TF Debugger and Eager Mode

Google

Industry sponsors **recruitment booths** setup in front of class

NVIDIA Google IBM Tencent

Friday: Project Presentations

Lisa Amini
Director of IBM Research Cambridge & Acting Director of MIT/IBM AI Lab

IBM Research

Lin Ma
Technical Lead, Manager
- Computer Vision Meets Social Networks

Tencent AI Lab

Afternoon Session:

Final Project Presentations
Pizza and Awards!

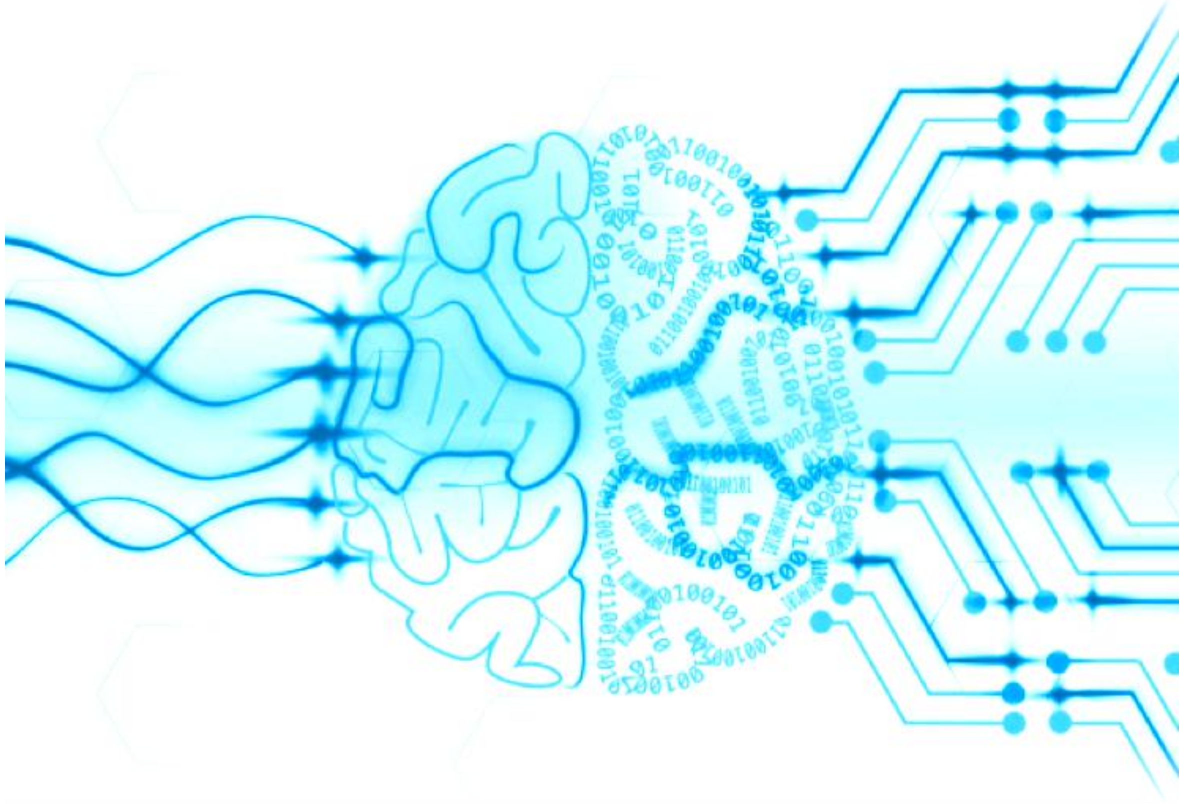
So far in 6.S191…

So far in 6.S191…

Data

- Signals
- Images
- Sensors

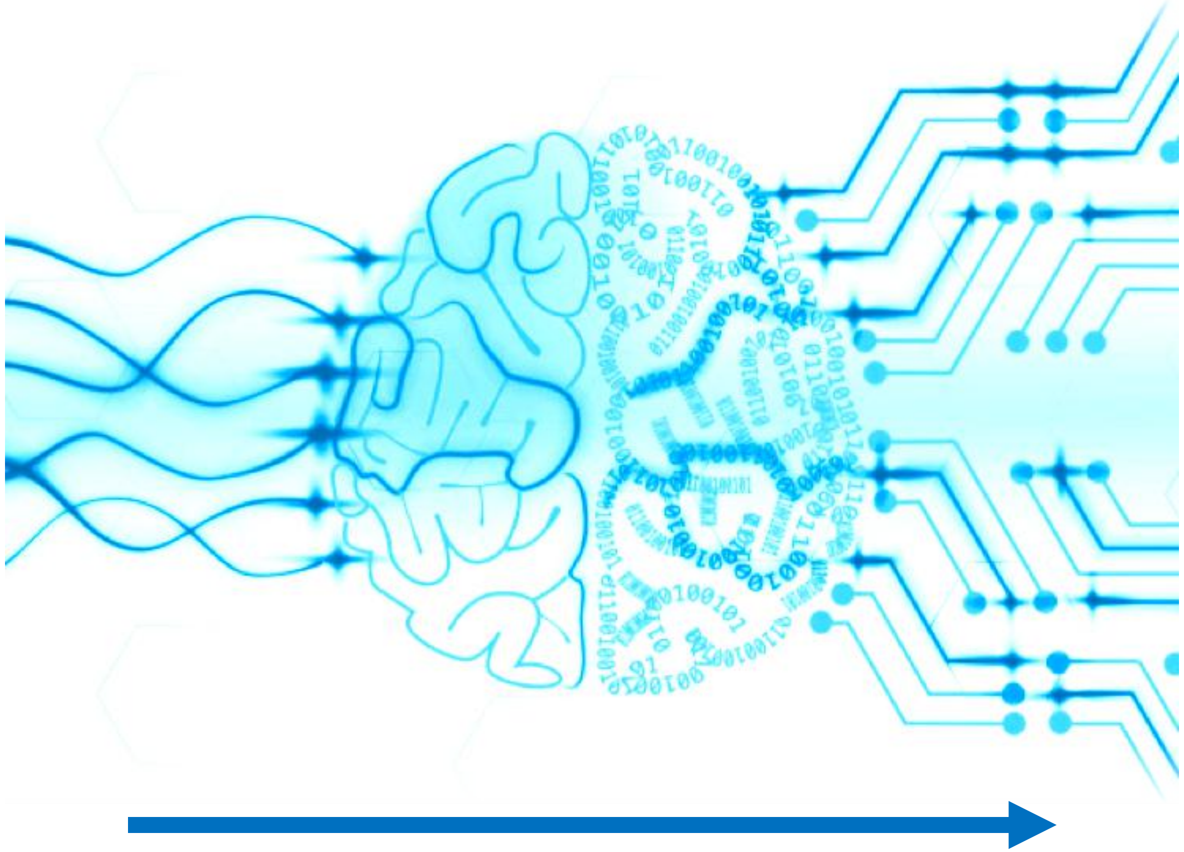
…



So far in 6.S191…

Data
- Signals
- Images
- Sensors

…



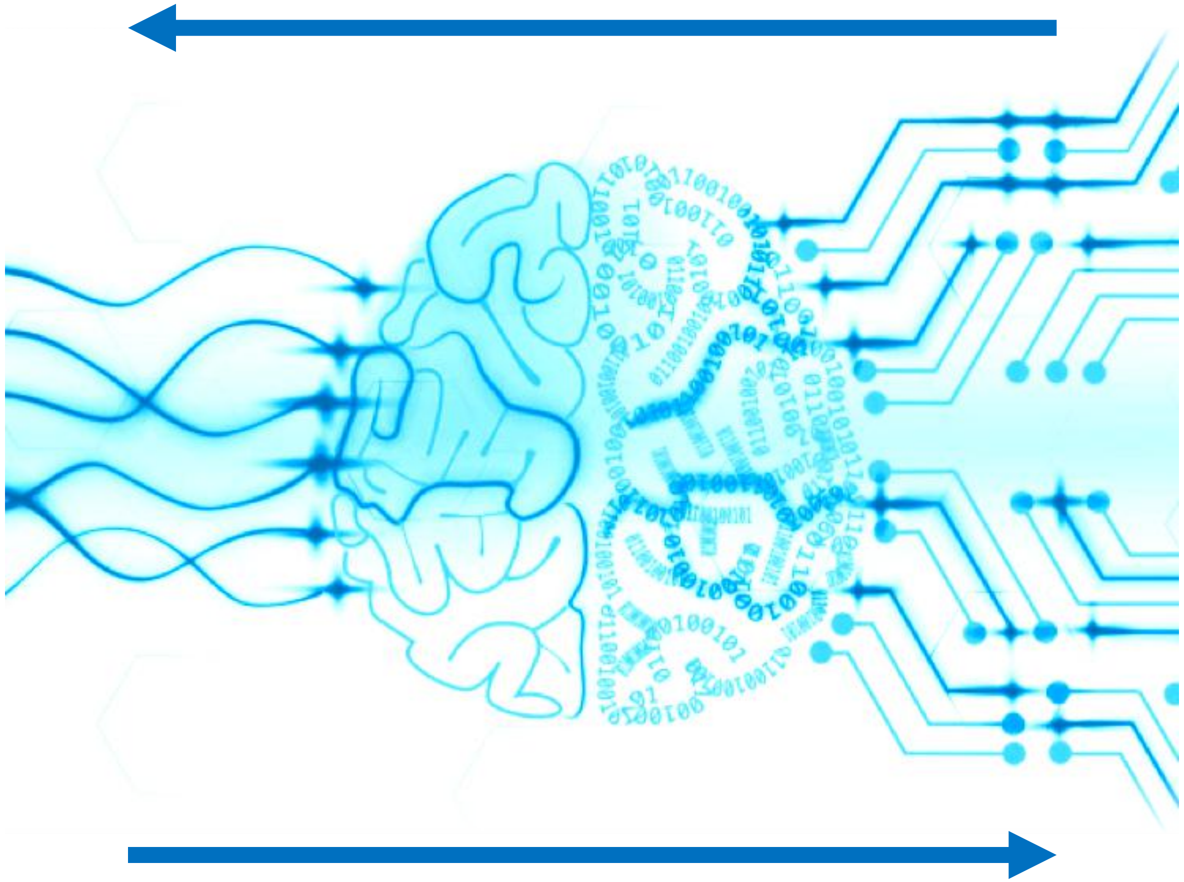
Decision
- Prediction
- Detection
- Action

…

So far in 6.S191…

Data
- Signals
- Images
- Sensors

…



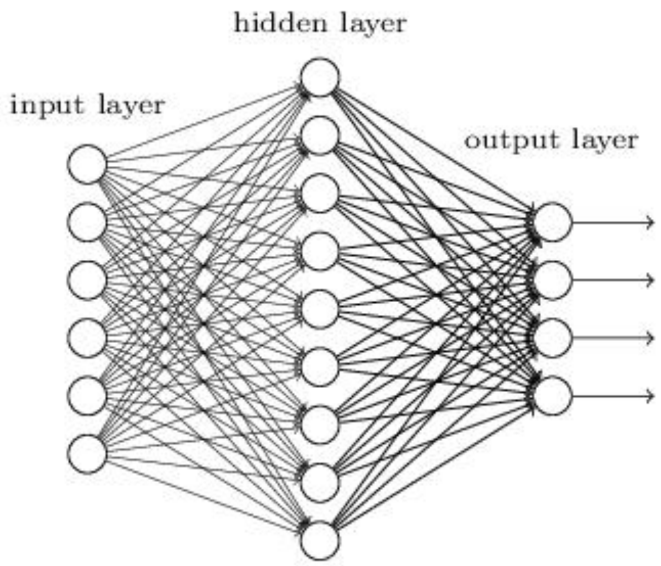
Decision
- Prediction
- Detection
- Action

…

Power of Neural Nets

Universal Approximation Theorem

A feedforward network with a single layer is sufficient to approximate, to an arbitrary precision, any continuous function.



Hornik, K., et al. "Multilayer feedforward networks are universal approximators." (1989)

Power of Neural Nets

Universal Approximation Theorem

A feedforward network with a single layer is sufficient to approximate, to an arbitrary precision, any continuous function.

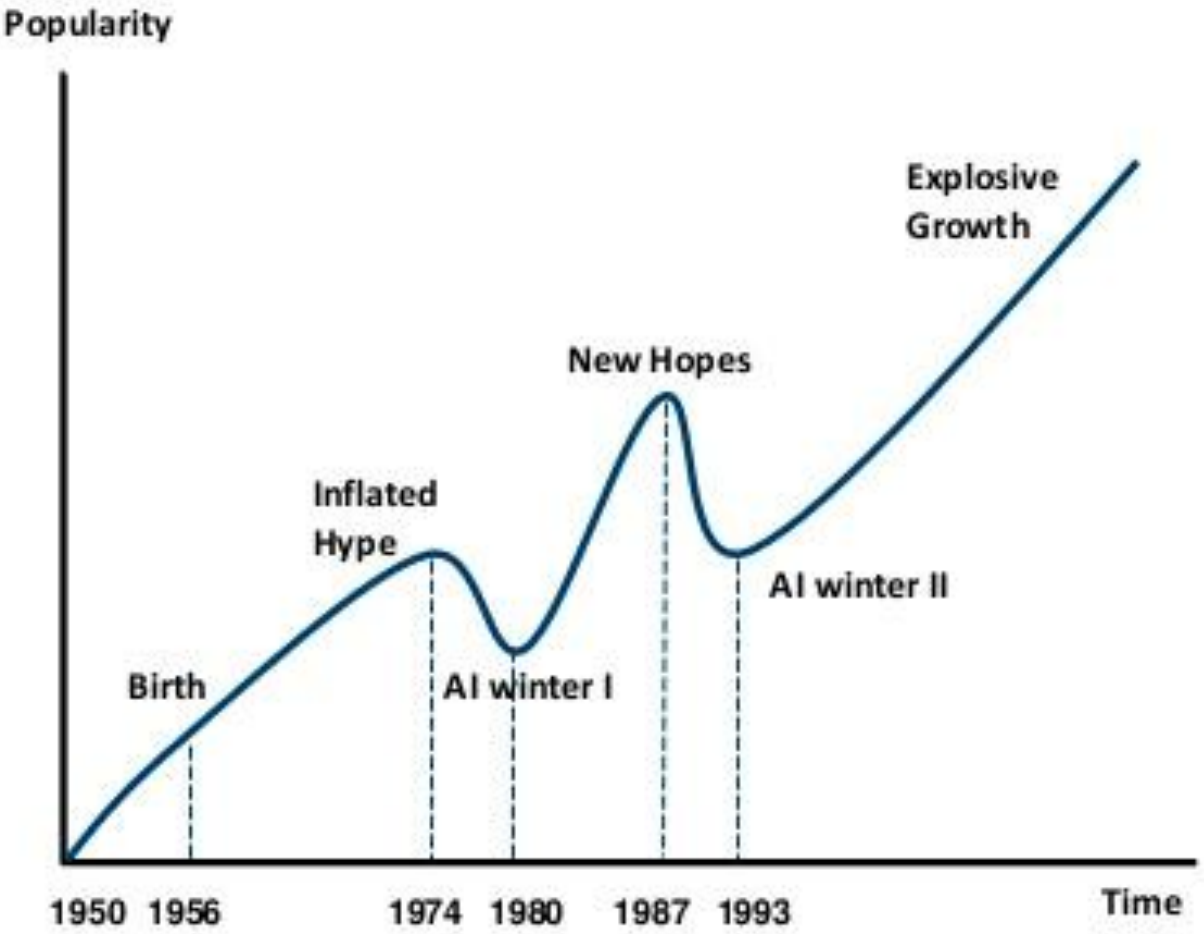
Caveats:

The number of hidden units may be infeasibly large

The resulting model may not generalize

Hornik, K., et al. *Neural Networks.* (1989)

History of Artificial Intelligence Hype



Limitations

Rethinking Generalization

"Understanding Deep Neural Networks requires rethinking generalization"

dog

banana

dog

tree

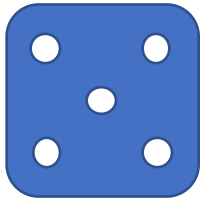
Zhang et al. *ICLR.* (2017)

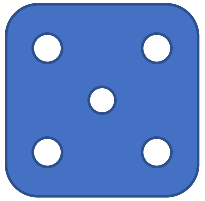
Rethinking Generalization

"Understanding Deep Neural Networks requires rethinking generalization"




dog banana dog tree



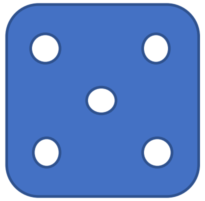
Zhang et al. *ICLR.* (2017)

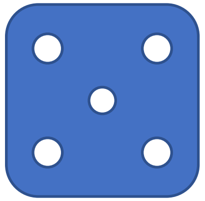
Rethinking Generalization

"Understanding Deep Neural Networks requires rethinking generalization"




dog | banana | dog | tree



banana | dog | tree | dog

Zhang et al. *ICLR.* (2017)

Rethinking Generalization

"Understanding Deep Neural Networks requires rethinking generalization"

~~dog~~

banana

~~banana~~

dog

~~dog~~

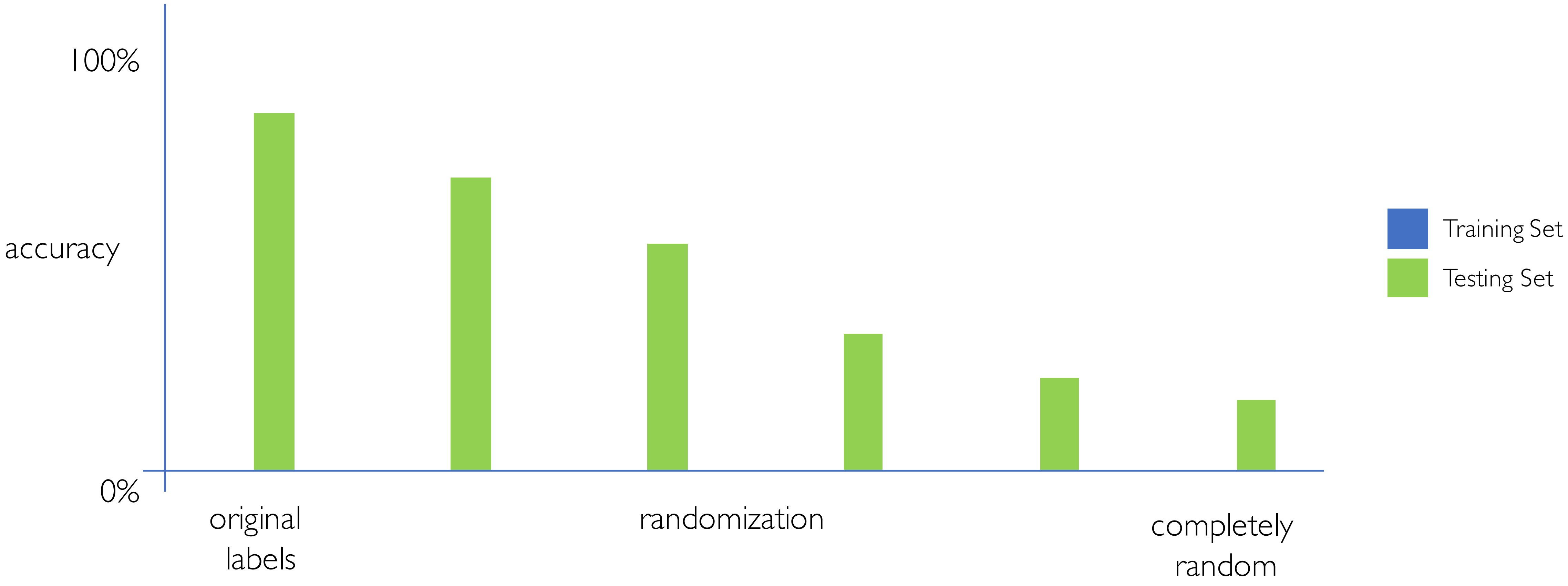
tree

~~tree~~

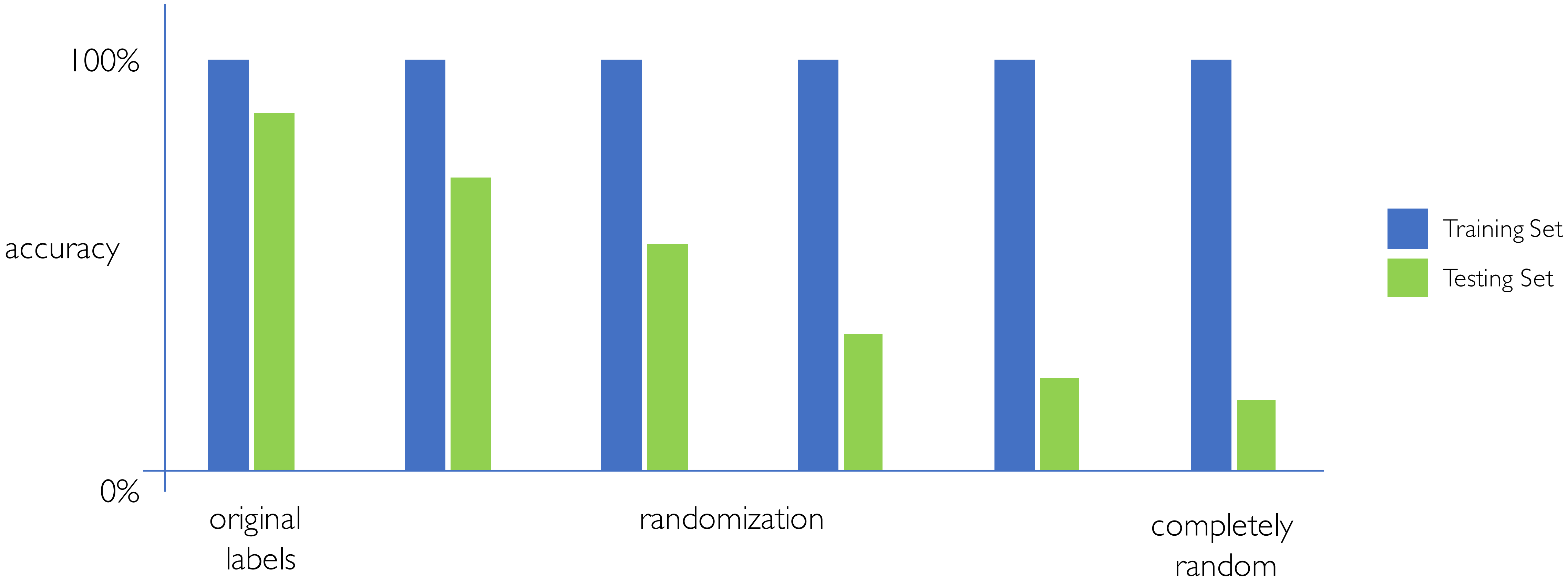
dog

Zhang et al. *ICLR.* (2017)

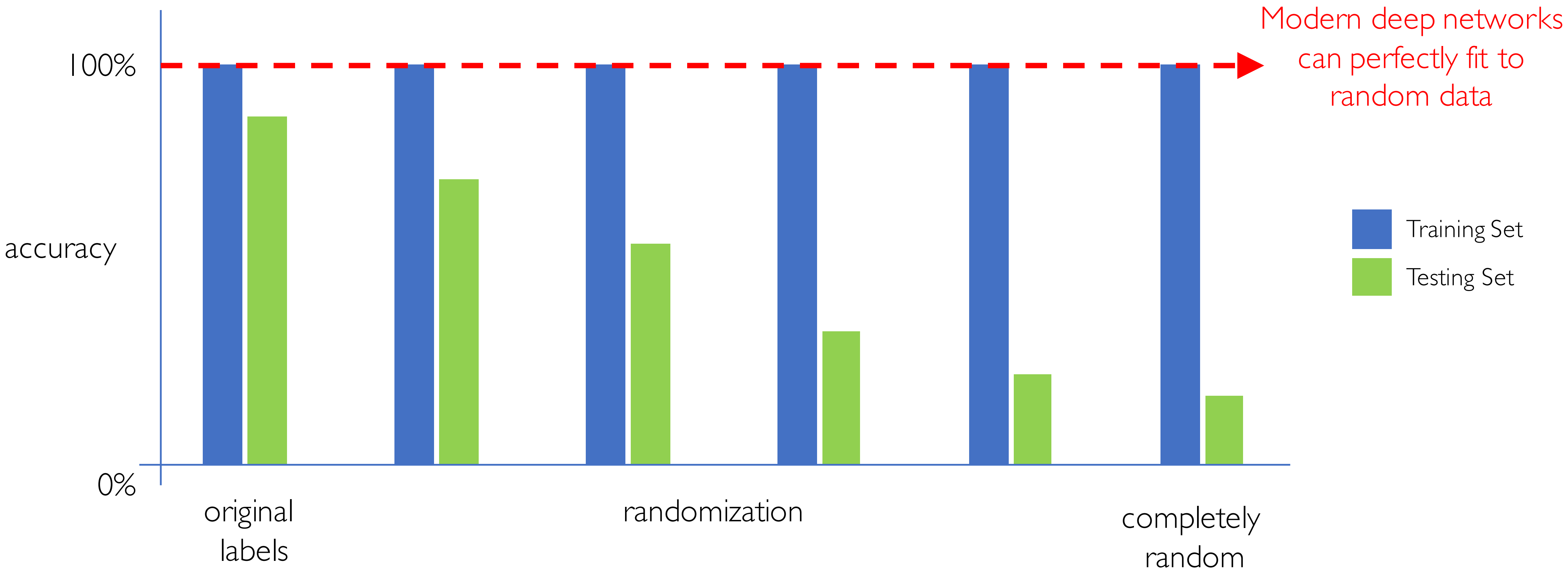
Capacity of Deep Neural Networks



Capacity of Deep Neural Networks

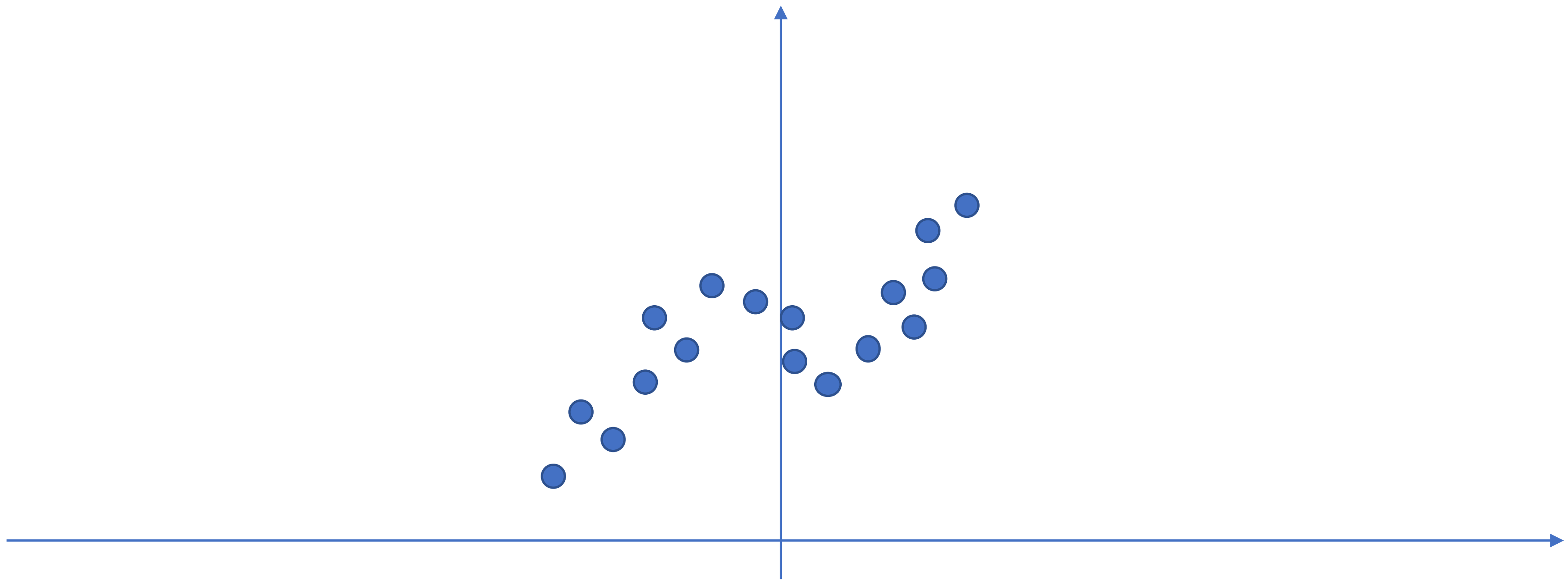


Capacity of Deep Neural Networks



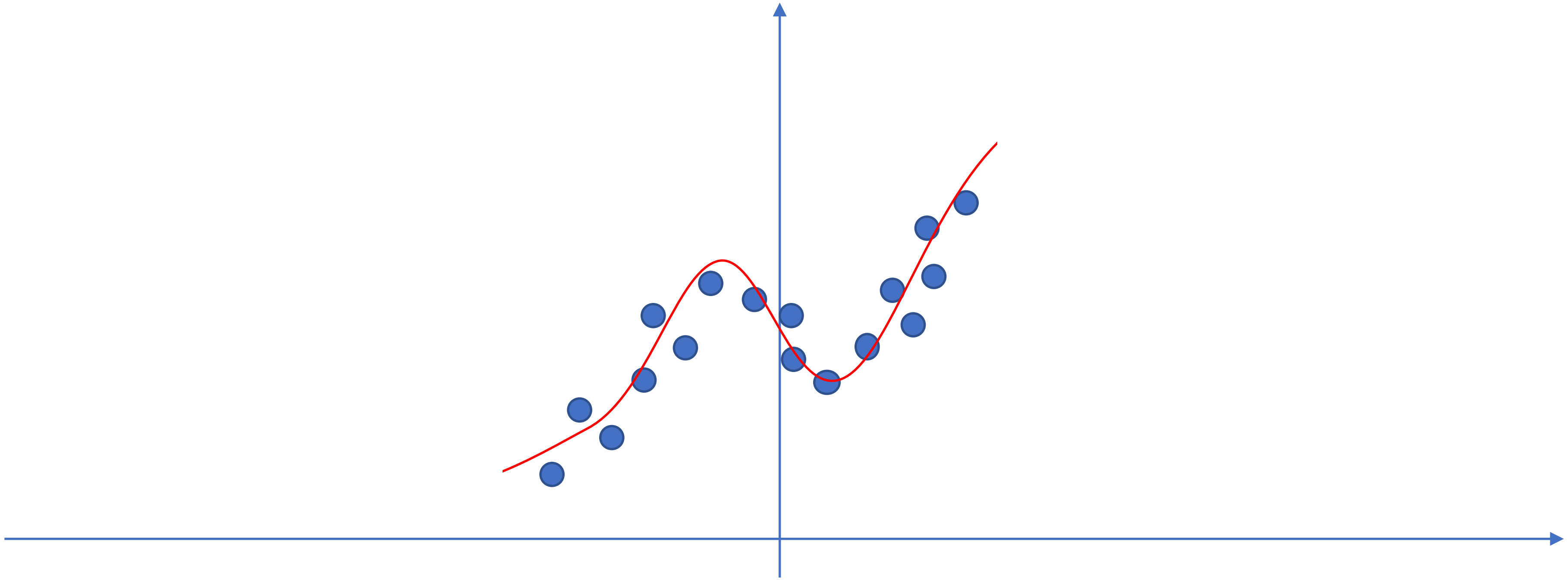
Function Approximators

*Neural networks are **excellent** function approximators*



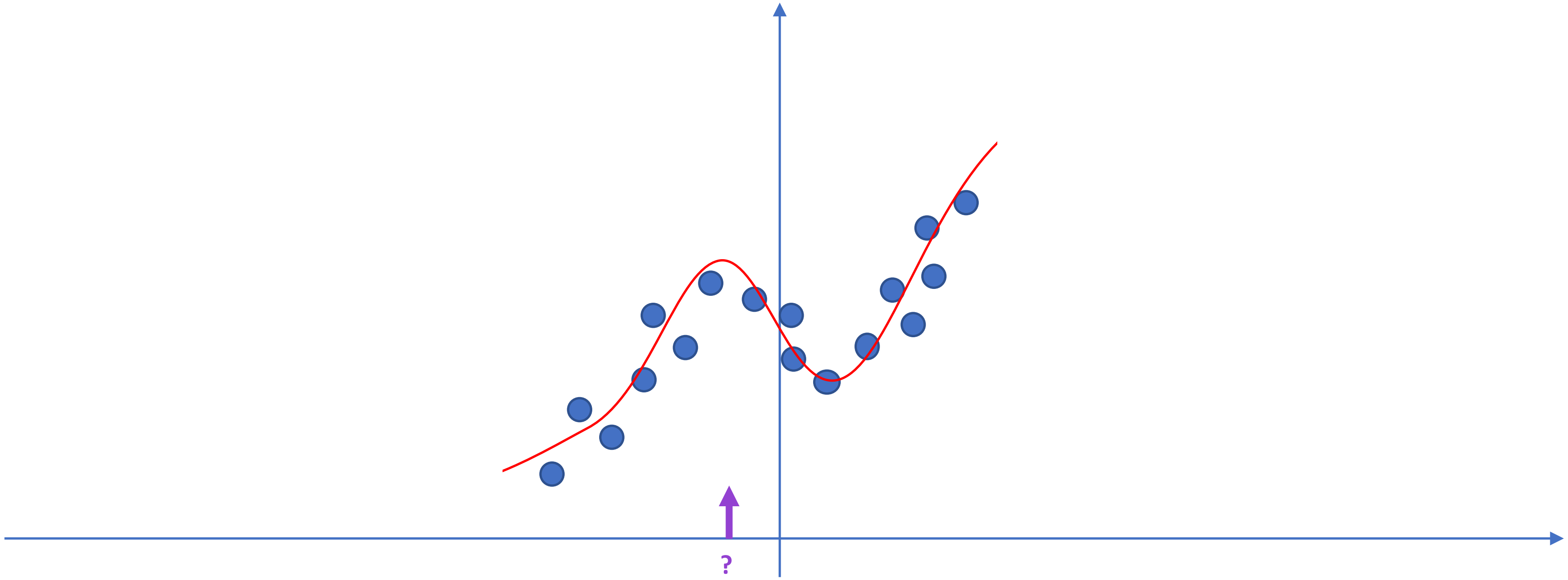
Function Approximators

Neural networks are ***excellent*** *function approximators*



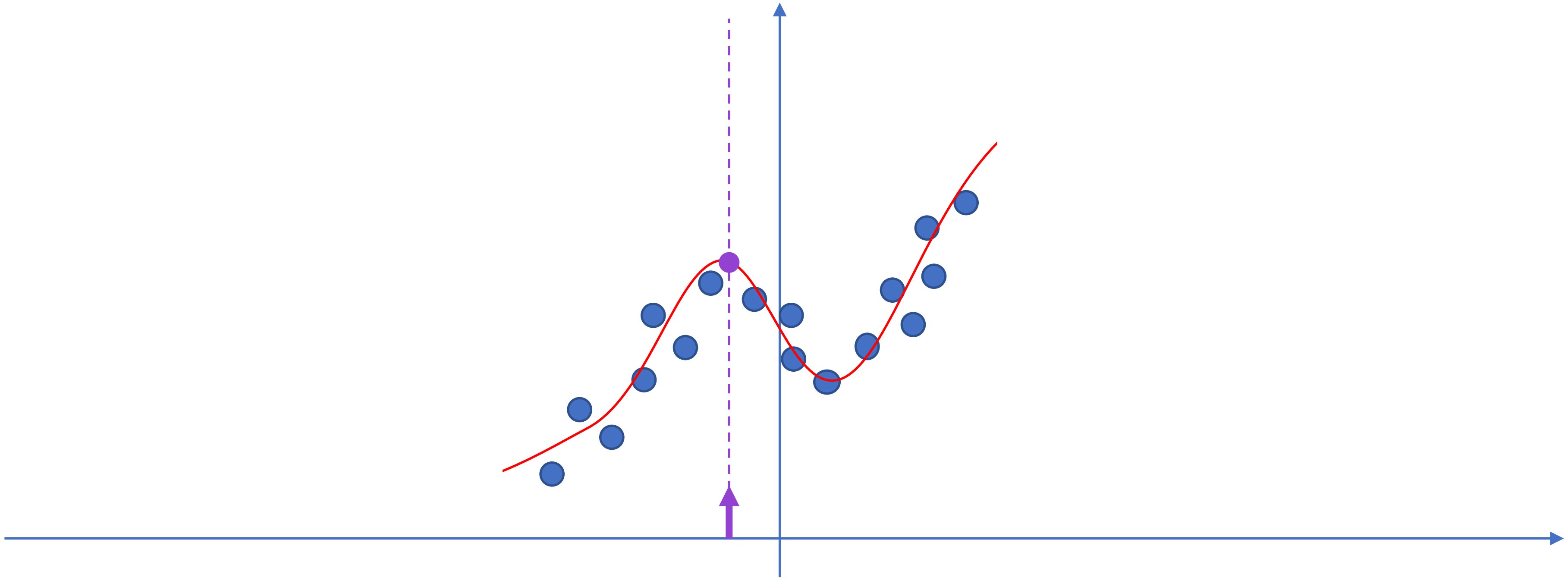
Function Approximators

Neural networks are ***excellent*** *function approximators*



Function Approximators

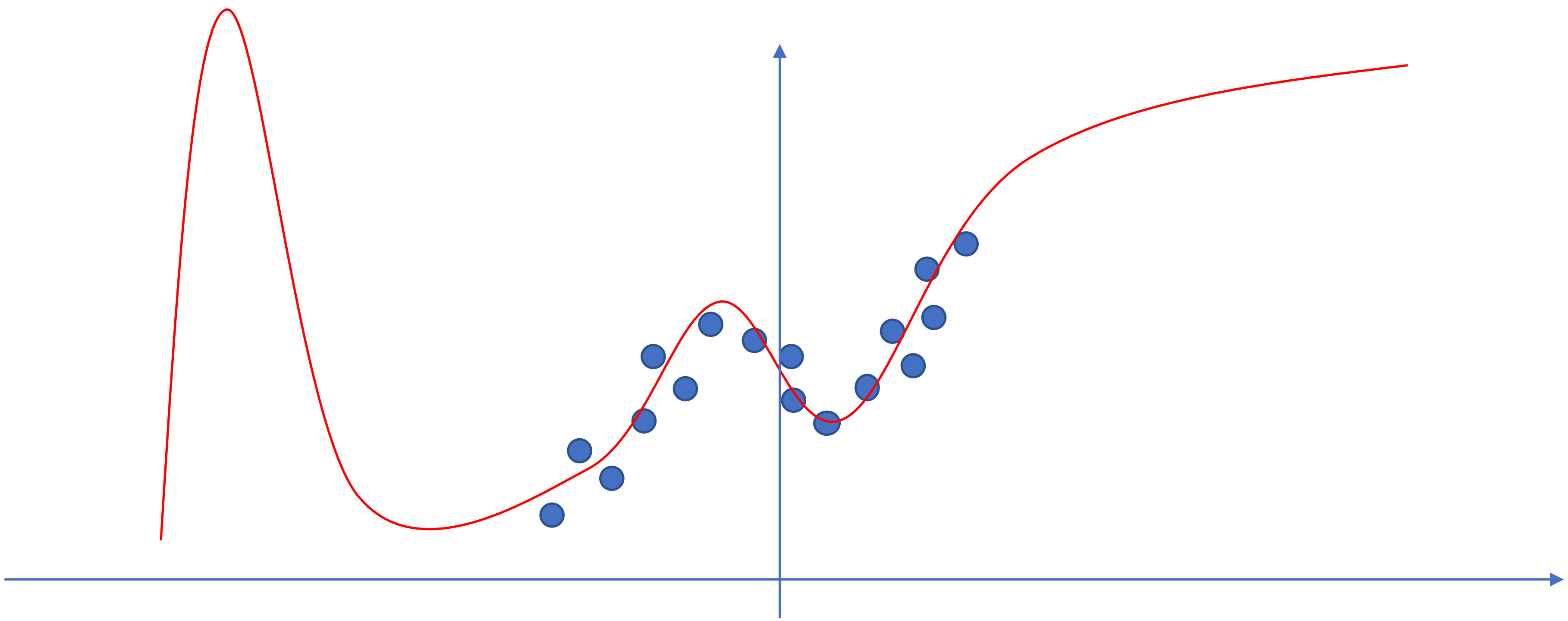
*Neural networks are **excellent** function approximators*



Function Approximators

Neural networks are ***excellent*** *function approximators*

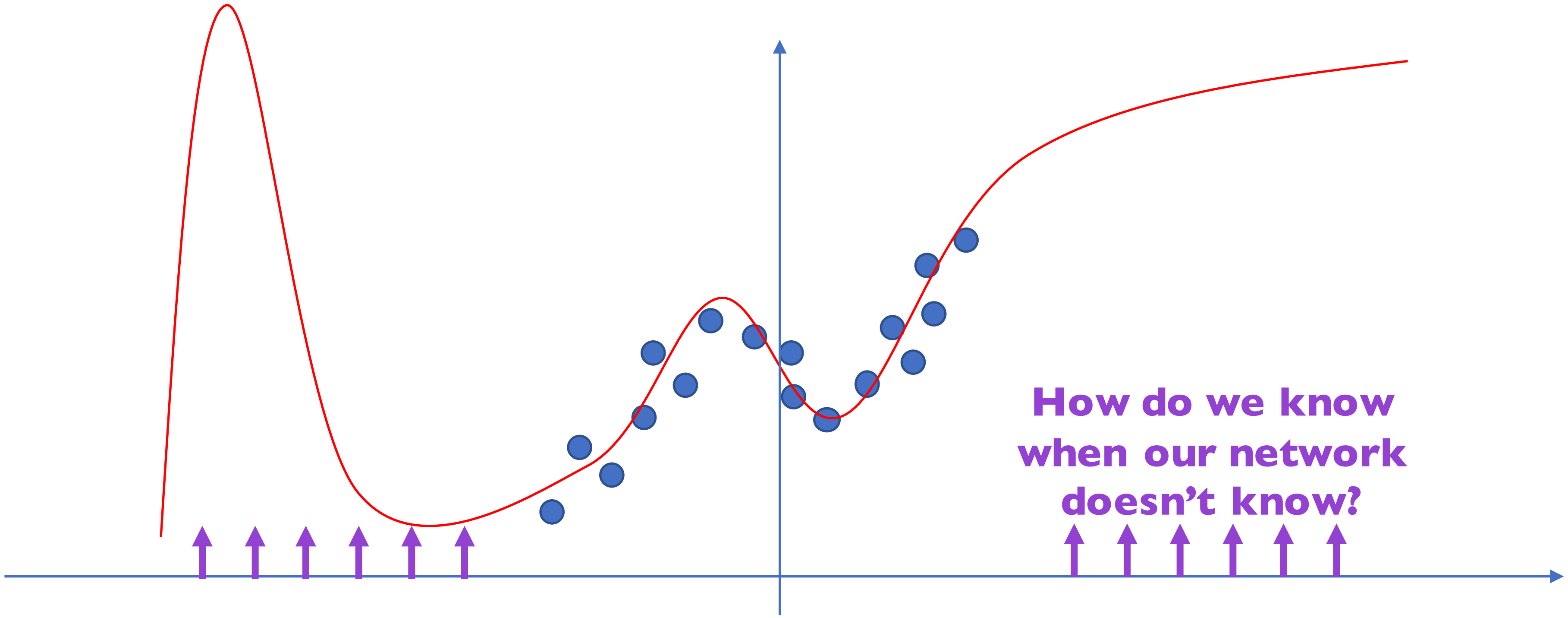
…when they have training data



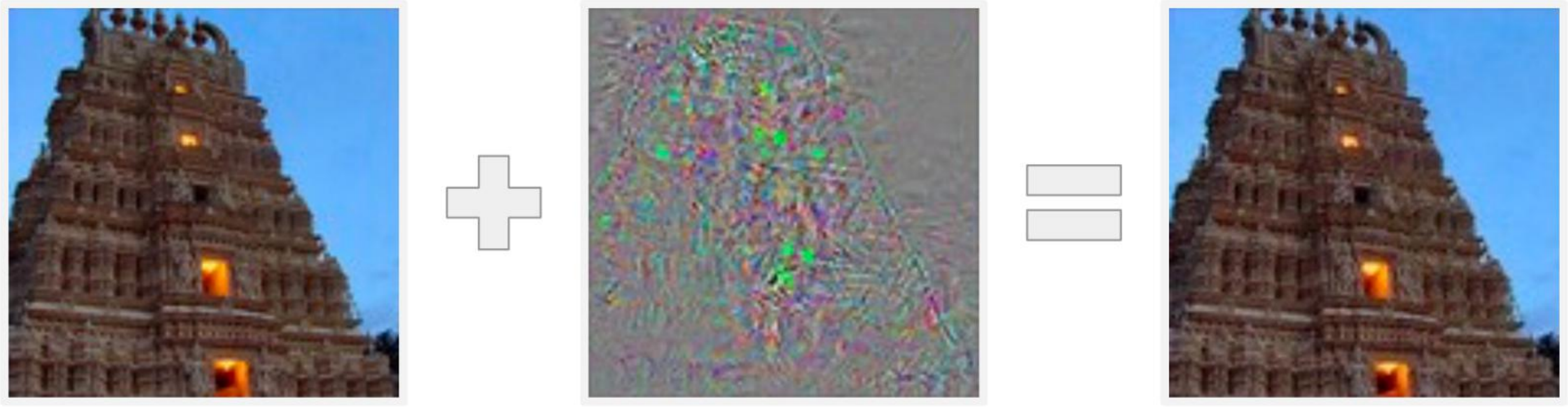
Function Approximators

Neural networks are ***excellent*** *function approximators*

...when they have training data



Adversarial Attacks on Neural Networks



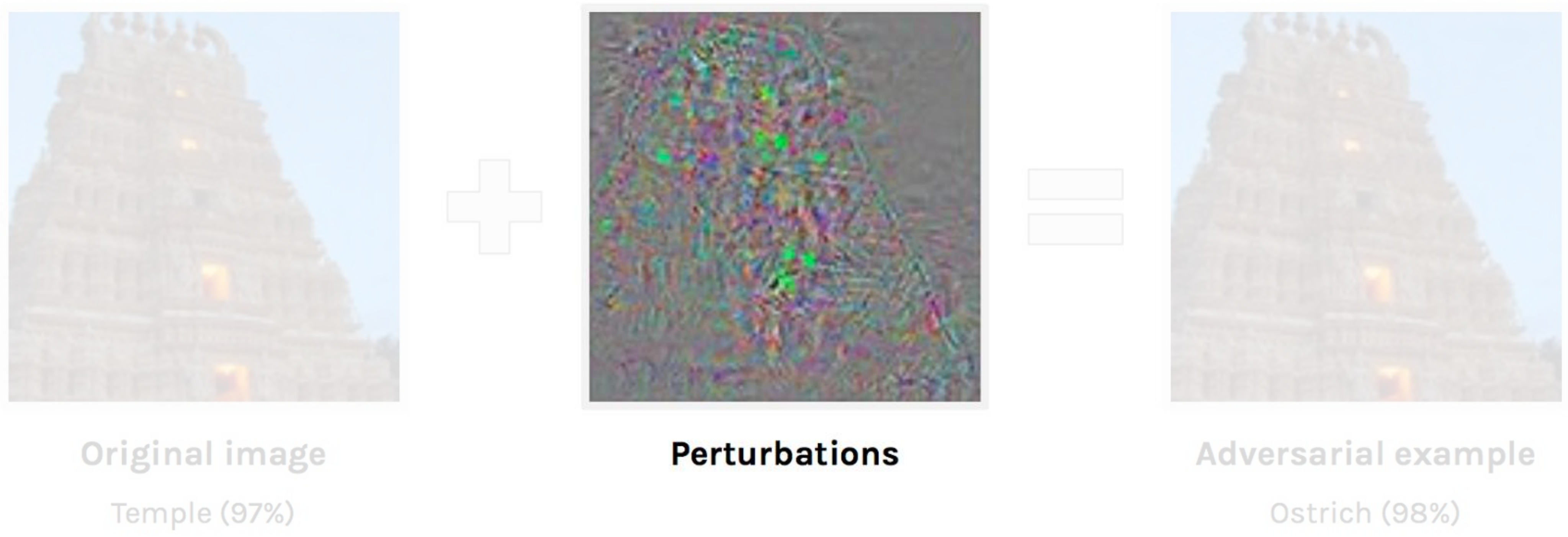
Original image — Temple (97%)

Perturbations

Adversarial example — Ostrich (98%)

Despois. "Adversarial examples and their implications". 2017.

Adversarial Attacks on Neural Networks



Adversarial Attacks on Neural Networks

Remember:

We train our networks with gradient descent

$$\theta \leftarrow \theta - \eta \frac{\partial J(\theta, x, y)}{\partial \theta}$$

"How does a small change in weights increase our loss"

Adversarial Attacks on Neural Networks

Remember:

We train our networks with gradient descent

$$\theta \leftarrow \theta - \eta \frac{\partial J(\theta, x, y)}{\partial \theta}$$

"How does a small change in weights decrease our loss"

Adversarial Attacks on Neural Networks

Remember:

We train our networks with gradient descent

$$\theta \leftarrow \theta - \eta \frac{\partial J(\theta, x, y)}{\partial \theta}$$

Fix your image x,
and true label y

"How does a small change in weights decrease our loss"

Adversarial Attacks on Neural Networks

Adversarial Image:

Modify image to increase error

$$x \leftarrow x + \eta \frac{\partial J(\theta, x, y)}{\partial x}$$

"How does a small change in the input increase our loss"

Goodfellow et al. 2014

Adversarial Attacks on Neural Networks

Adversarial Image:

Modify image to increase error

$$x \leftarrow x + \eta \frac{\partial J(\theta, x, y)}{\partial x}$$

"How does a small change in the input increase our loss"

Adversarial Attacks on Neural Networks

Adversarial Image:

Modify image to increase error

$$x \leftarrow x + \eta \frac{\partial J(\theta, x, y)}{\partial x}$$

Fix your weights θ, and true label y

"How does a small change in the input increase our loss"

Neural Network Limitations…

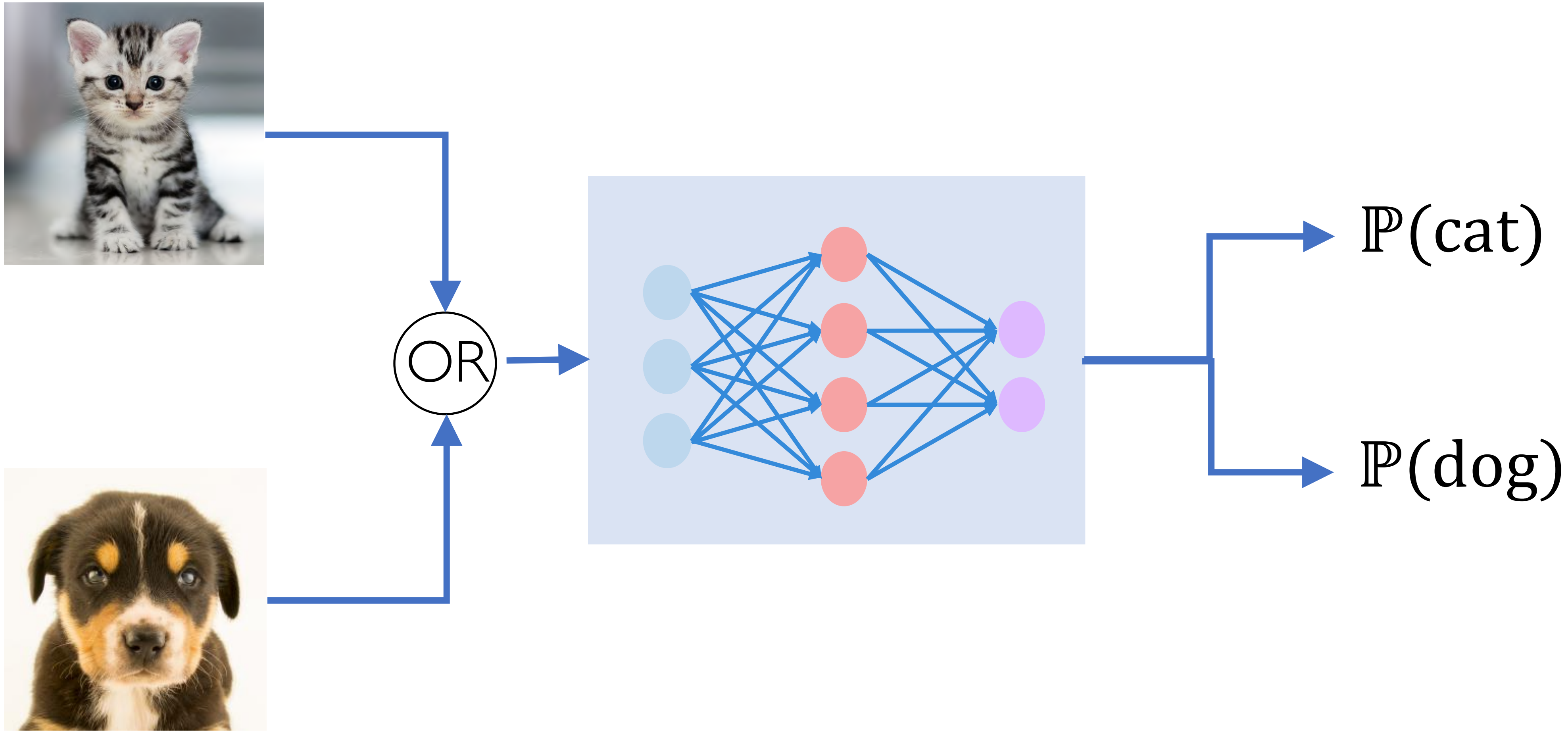
- Very **data hungry** (eg. often millions of examples)
- **Computationally intensive** to train and deploy (tractably requires GPUs)
- Easily fooled by **adversarial examples**
- Poor at **representing uncertainty** (how do you know what the model knows?)
- Uninterpretable **black boxes**, difficult to trust
- **Finicky to optimize**: non-convex, choice of architecture, learning parameters
- Often require **expert knowledge** to design, fine tune architectures

Neural Network Limitations…

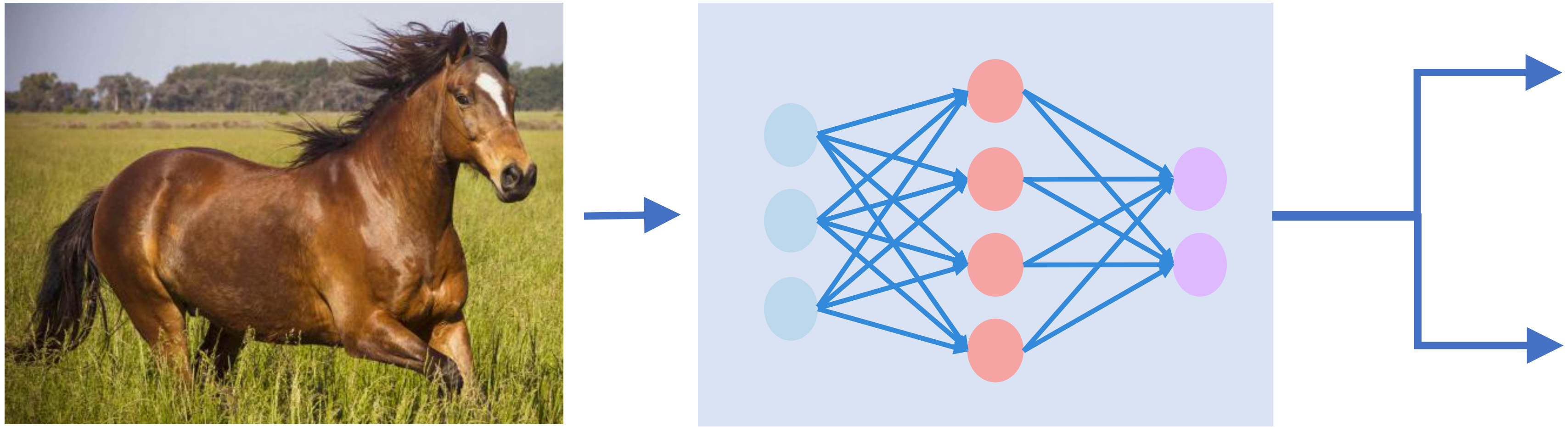
- Very **data hungry** (eg. often millions of examples)
- **Computationally intensive** to train and deploy (tractably requires GPUs)
- Easily fooled by **adversarial examples**
- Poor at **representing uncertainty** (how do you know what the model knows?)
- Uninterpretable **black boxes**, difficult to trust
- **Finicky to optimize**: non-convex, choice of architecture, learning parameters
- Often require **expert knowledge** to design, fine tune architectures

New Frontiers 1:
Bayesian Deep Learning

Why Care About Uncertainty?



Why Care About Uncertainty?



$$\mathbb{P}(\text{cat}) = 0.2$$

$$\mathbb{P}(\text{dog}) = 0.8$$

Remember:

$$\mathbb{P}(\text{cat}) + \mathbb{P}(\text{dog}) = 1$$

Bayesian Deep Learning for Uncertainty

Network tries to learn output, $\boldsymbol{Y}$, directly from raw data, $\boldsymbol{X}$

Find mapping, f, parameterized by weights $\boldsymbol{\theta}$ such that

$$\min \mathcal{L}(\boldsymbol{Y}, f(X; \boldsymbol{\theta}))$$

Bayesian neural networks aim to learn a posterior over weights, $\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Y})$:

$$\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Y}) = \frac{\mathbb{P}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})}{\mathbb{P}(\boldsymbol{Y}|\boldsymbol{X})}$$

Bayesian Deep Learning for Uncertainty

Network tries to learn output, $\boldsymbol{Y}$, directly from raw data, $\boldsymbol{X}$

Find mapping, f, parameterized by weights $\boldsymbol{\theta}$ such that

$$\min \mathcal{L}(\boldsymbol{Y}, f(X; \boldsymbol{\theta}))$$

Bayesian neural networks aim to learn a posterior over weights, $\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Y})$:

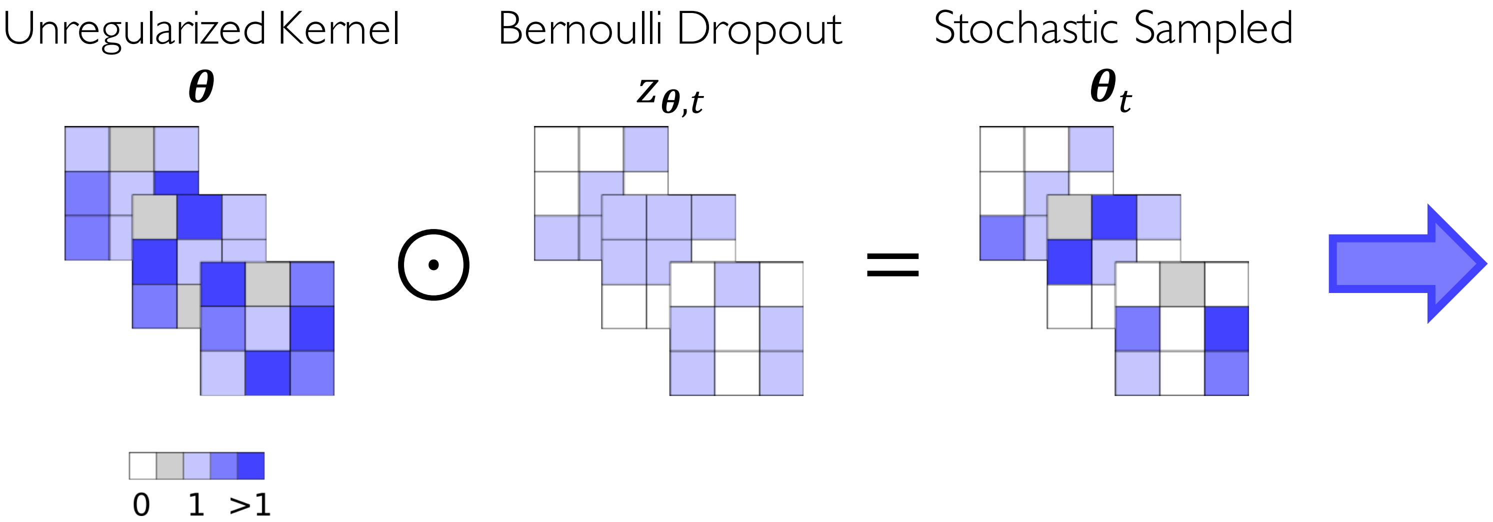
Intractable! $$\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Y}) = \frac{\mathbb{P}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})}{\mathbb{P}(\boldsymbol{Y}|\boldsymbol{X})}$$

Approximate the posterior $\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Y})$ by sampling

Elementwise Dropout for Uncertainty

Evaluate T stochastic forward passes through the network $\{\boldsymbol{\theta}_t\}_{t=1}^{T}$

Dropout as a form of stochastic sampling $\quad z_{w,t} \sim Bernoulli(p) \quad \forall\, w \in \boldsymbol{\theta}$



$$\mathbb{E}(\widehat{\boldsymbol{Y}}|\boldsymbol{X}) = \frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{X}|\boldsymbol{\theta}_t)$$

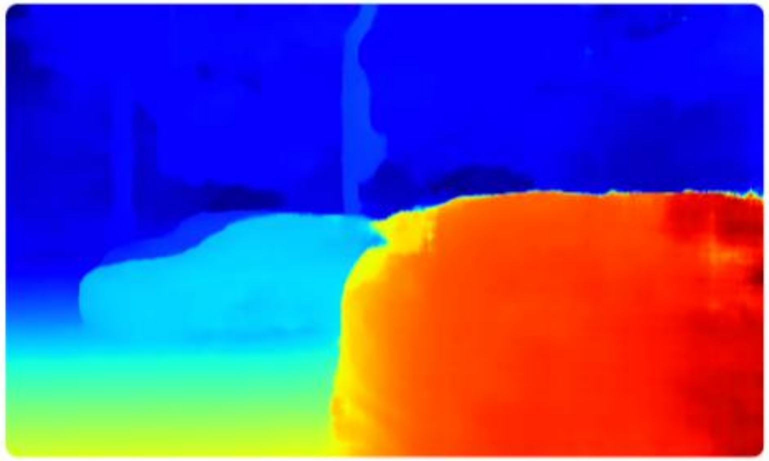
$$Var(\widehat{\boldsymbol{Y}}|\boldsymbol{X}) = \frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{X})^2 - \mathbb{E}(\widehat{\boldsymbol{Y}}|\boldsymbol{X})^2$$

Gal and Ghahramani, *ICML*, 2016.

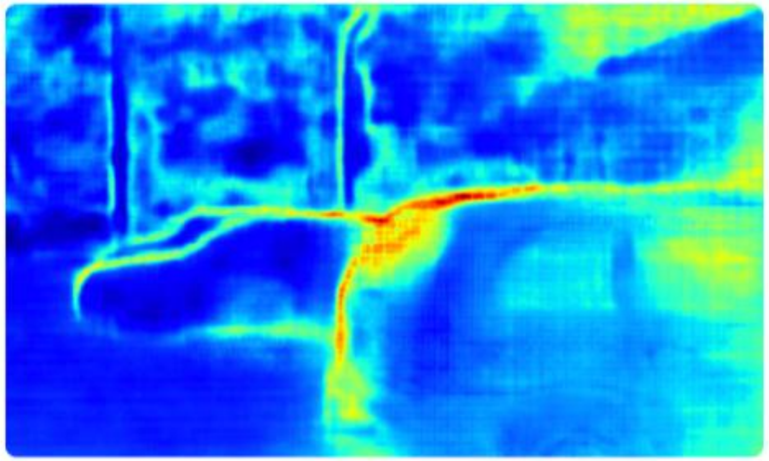
Amini et al., *NIPS Workshop on Bayesian Deep Learning*, 2017.

Model Uncertainty Application

Input image



Predicted Depth



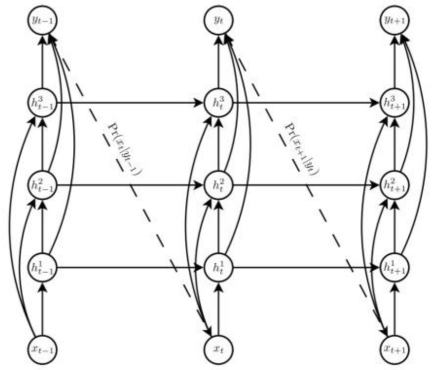
Model Uncertainty

Kendall, Gal, *NIPS* 2017.

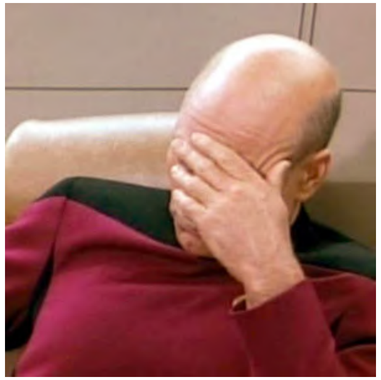
New Frontiers II:
Learning to Learn

Motivation

Standard deep neural networks are optimized for **a single task**



Complexity of models increases



Greater the need for specialized engineers

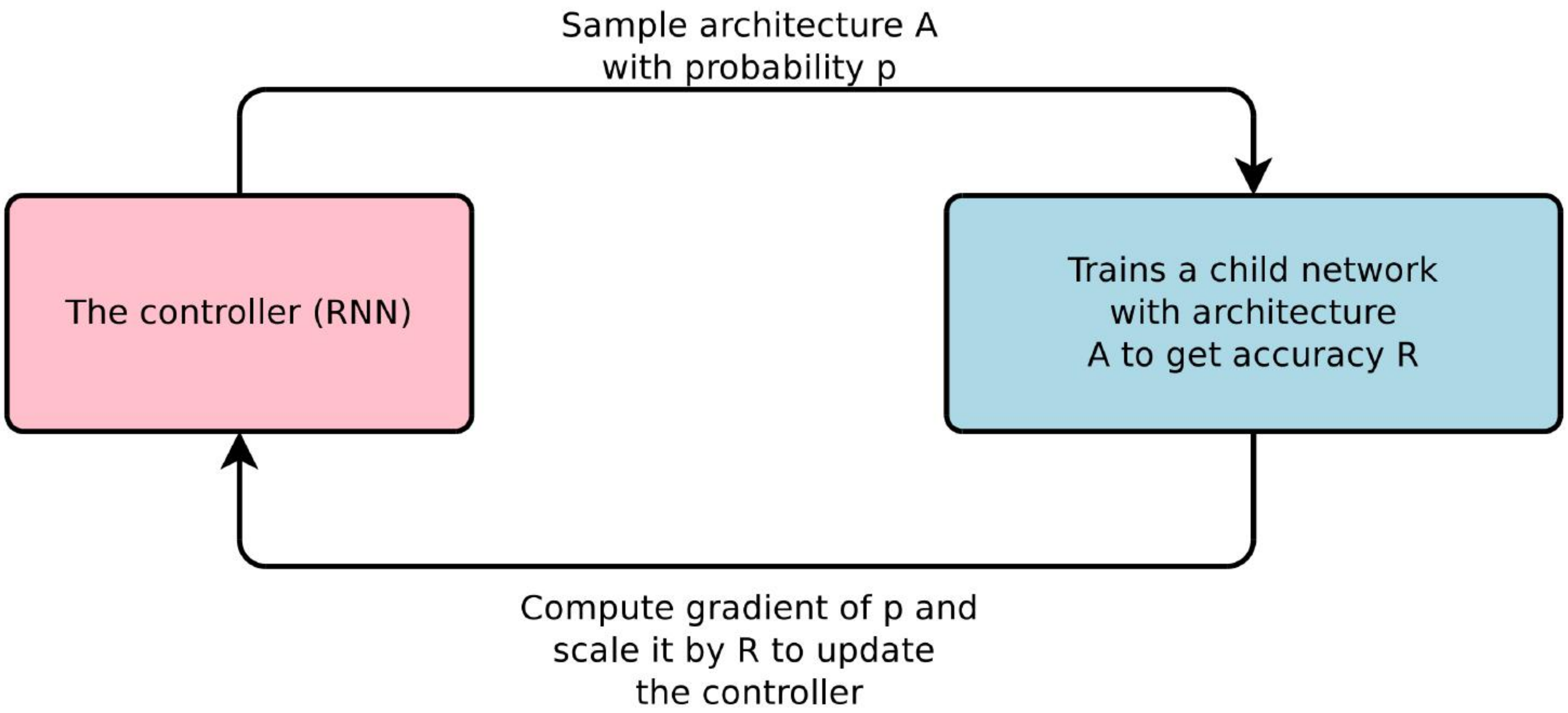
Often require **expert knowledge** to build an architecture for a given task

Possible Solution

AutoML: Learning to Learn

Build a learning algorithm that learns which model to use to solve a given problem

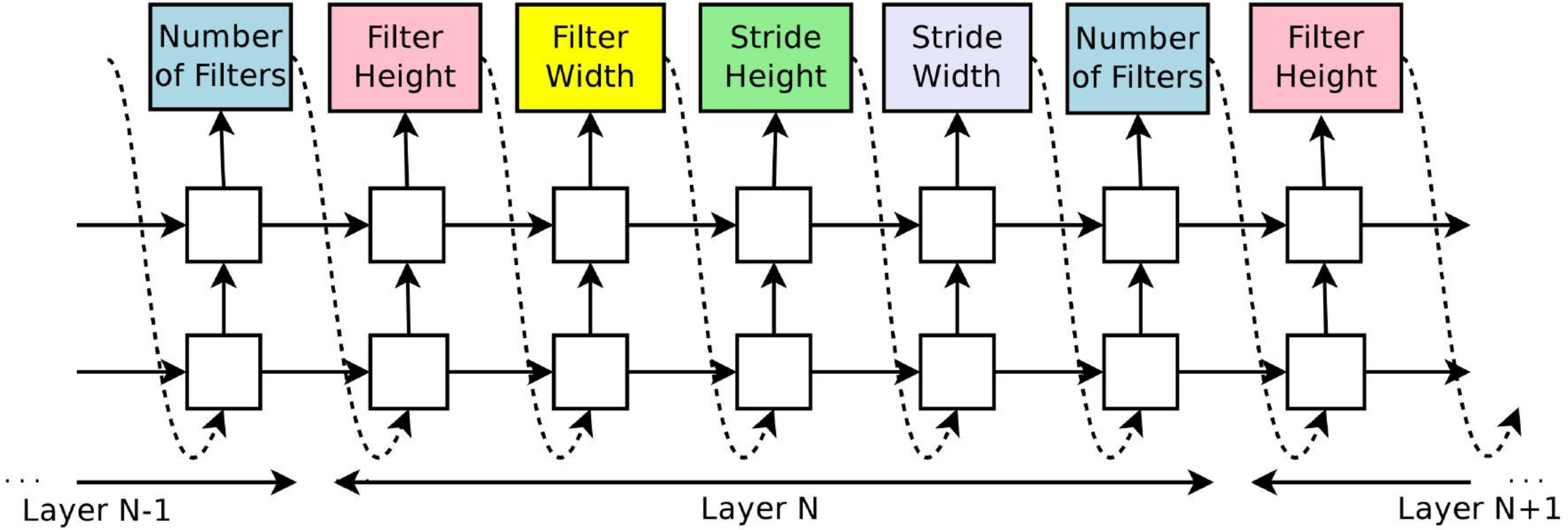
AutoML: Learning to Learn



Zoph and Le, *ICLR* 2017.

Model Controller

At each step, the model samples a brand new network



The Child Network

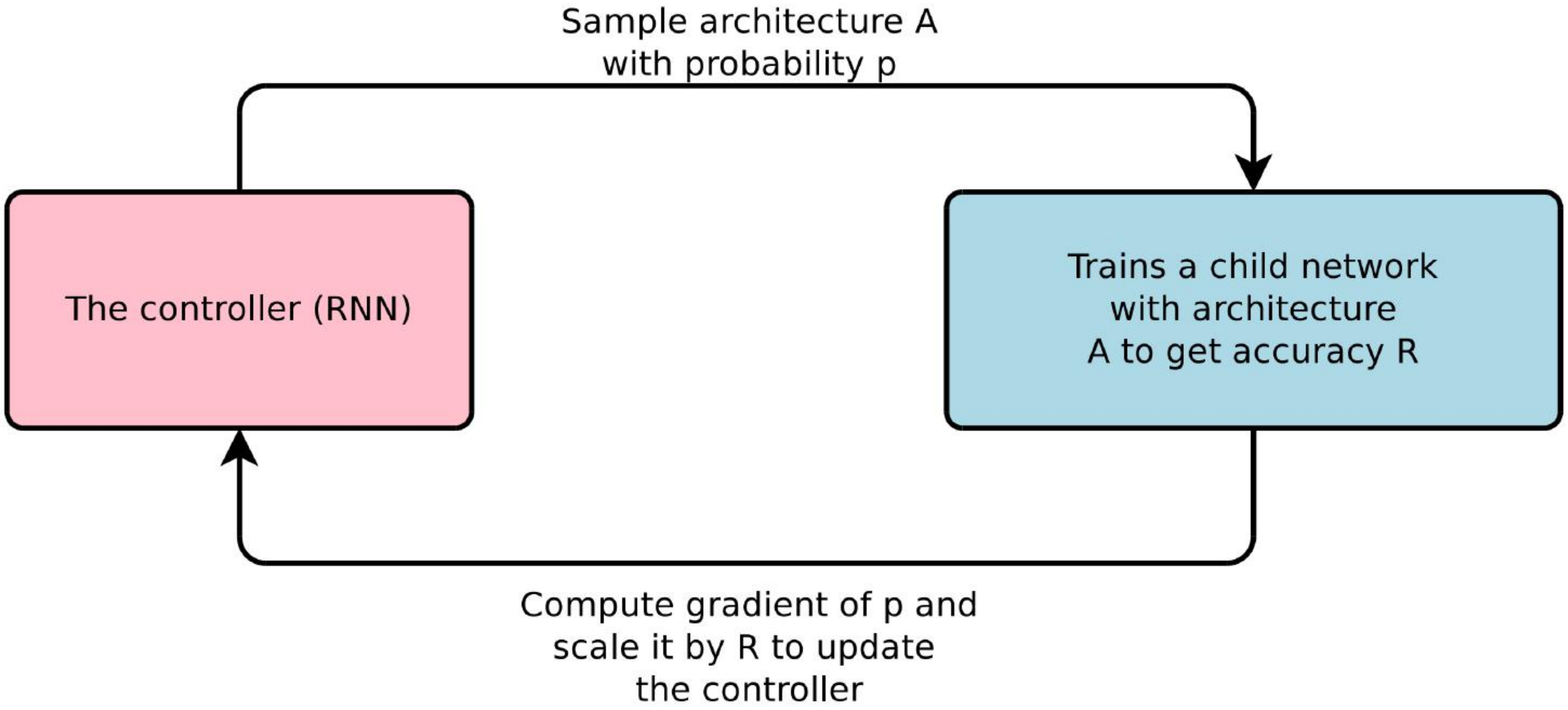
Training Data → Sampled network from RNN → Predicted Labels

Compute final accuracy on this dataset

Update RNN controller based on the accuracy of the child network after training

Zoph and Le, *ICLR* 2017.

Learning to Learn: A level deeper



Zoph and Le, *ICLR* 2017.

This Spawns a Very Powerful Idea

- Design an AI algorithm that can build new models capable of solving a task
- Reduces the need for experienced engineers to design the networks
- Makes deep learning more accessible to the public

The connection to Artificial General Intelligence:
the ability to intelligently reason about how we learn

Questions?