

Mental Models for Neural Networks

Joint Mathematics Meetings, Washington DC,

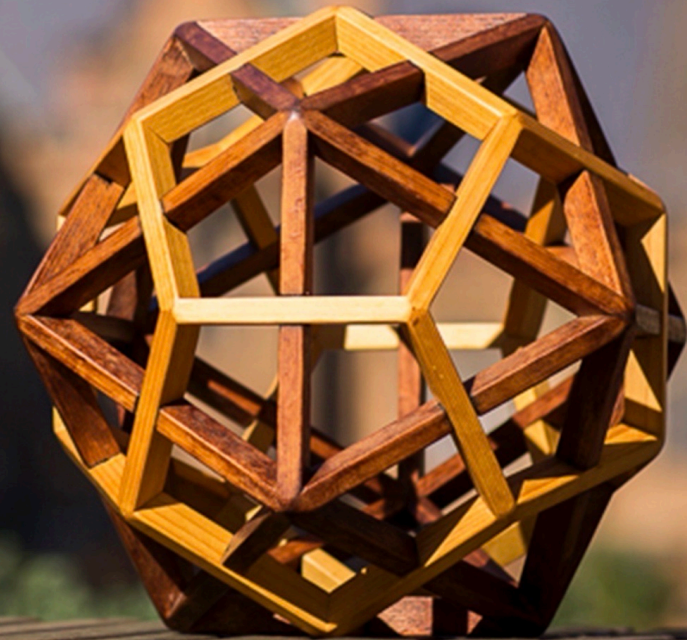
Colloquium Lectures, Lecture 2

Geordie Williamson



THE UNIVERSITY OF
SYDNEY

**Mathematical
Research
Institute**

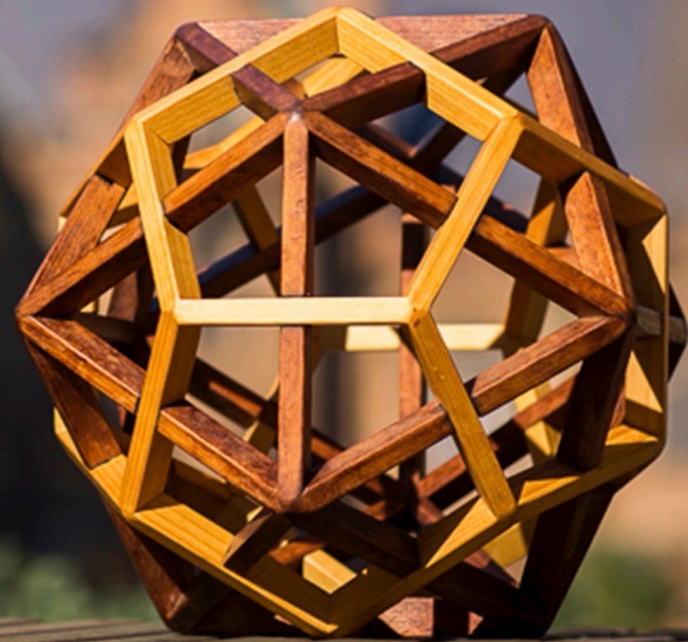




THE UNIVERSITY OF
SYDNEY

**Mathematical
Research
Institute**

- *International visitor program
(Visits of 4-12 weeks to work with an
Australian based researcher.)*
- *Special semesters and focus periods
(e.g. quantum computing, optimal
transport, computational number theory
and magma...)*



It seems important nowadays to have some mental picture of what goes on inside neural networks.

It seems important nowadays to have some mental picture of what goes on inside neural networks.

Some fundamentals are straightforward,
as I hope to explain.

It seems important nowadays to have some mental picture of what goes on inside neural networks.

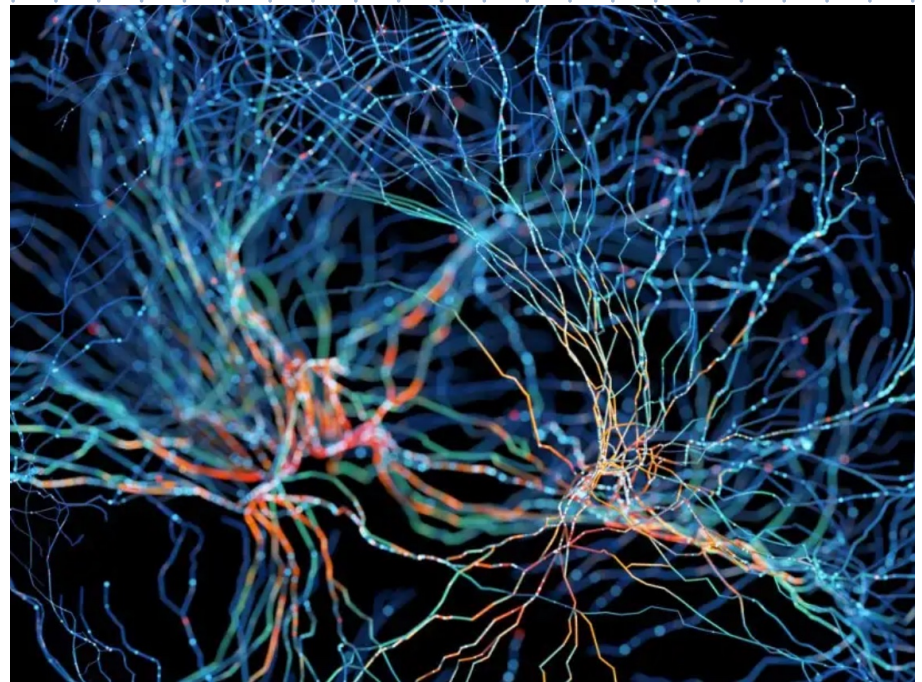
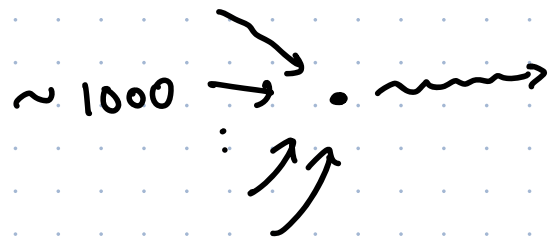
Some fundamentals are straightforward,
as I hope to explain.

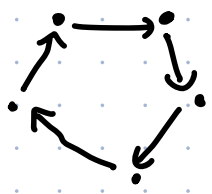
WARNING: I am not an expert!

(I welcome feedback on errors, omissions,...)

Very simplistic picture of brain:

- $\sim 86 \times 10^9$ neurons
- relatively sparse

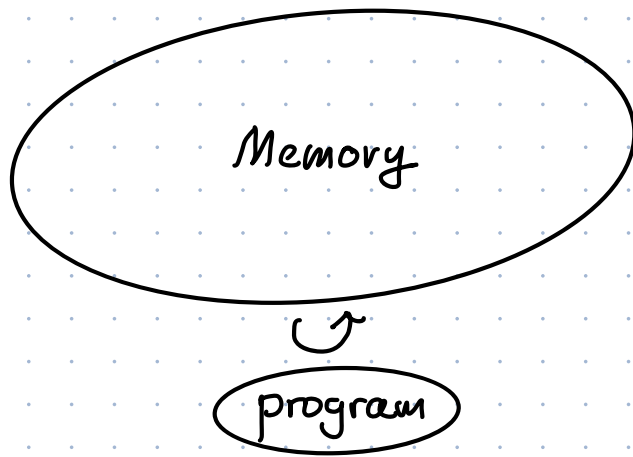


- assumed local and classical $\implies$ parallel
- great variation in neuron function
- many loops (recurrence) 

Changing conception of computing

Old Picture

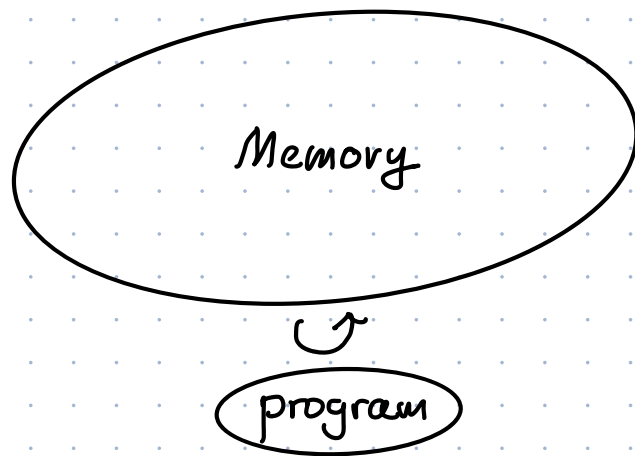
("Turing machine")



Changing conception of computing

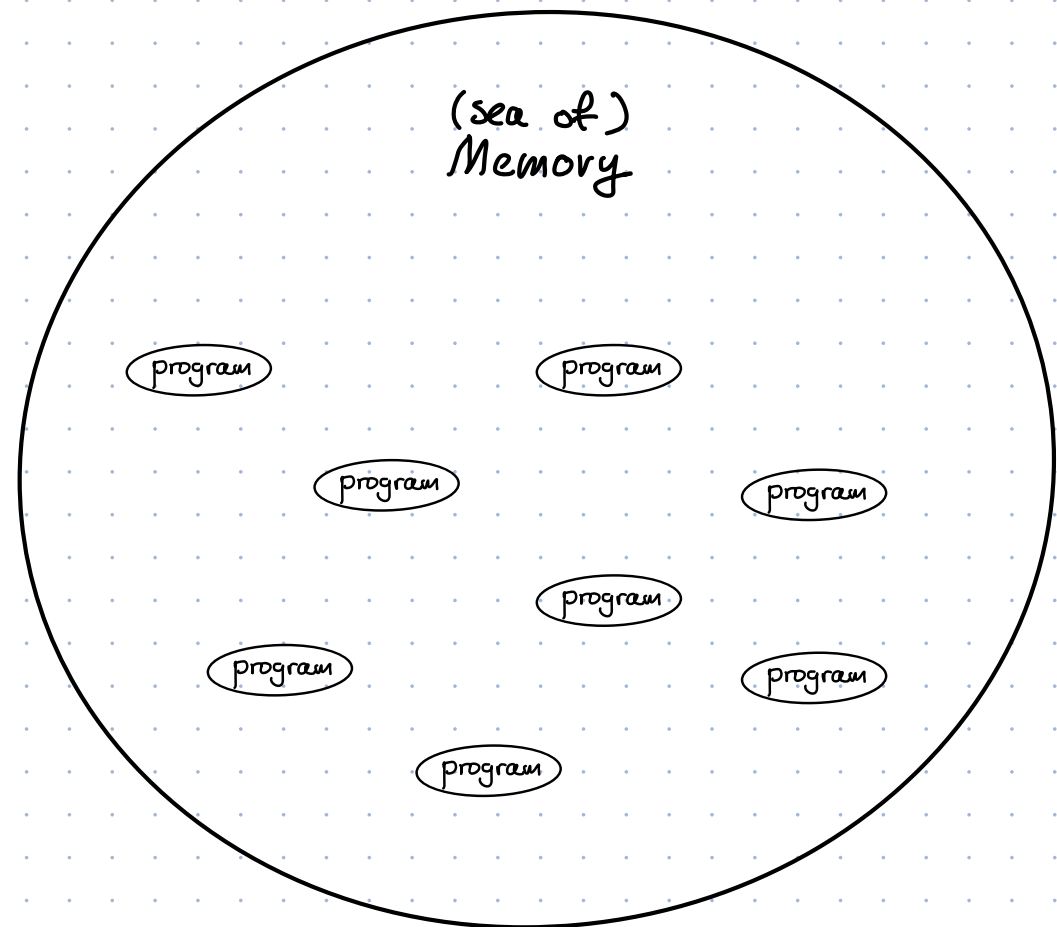
Old Picture

("Turing machine")



Emerging Picture

(last 30 years)

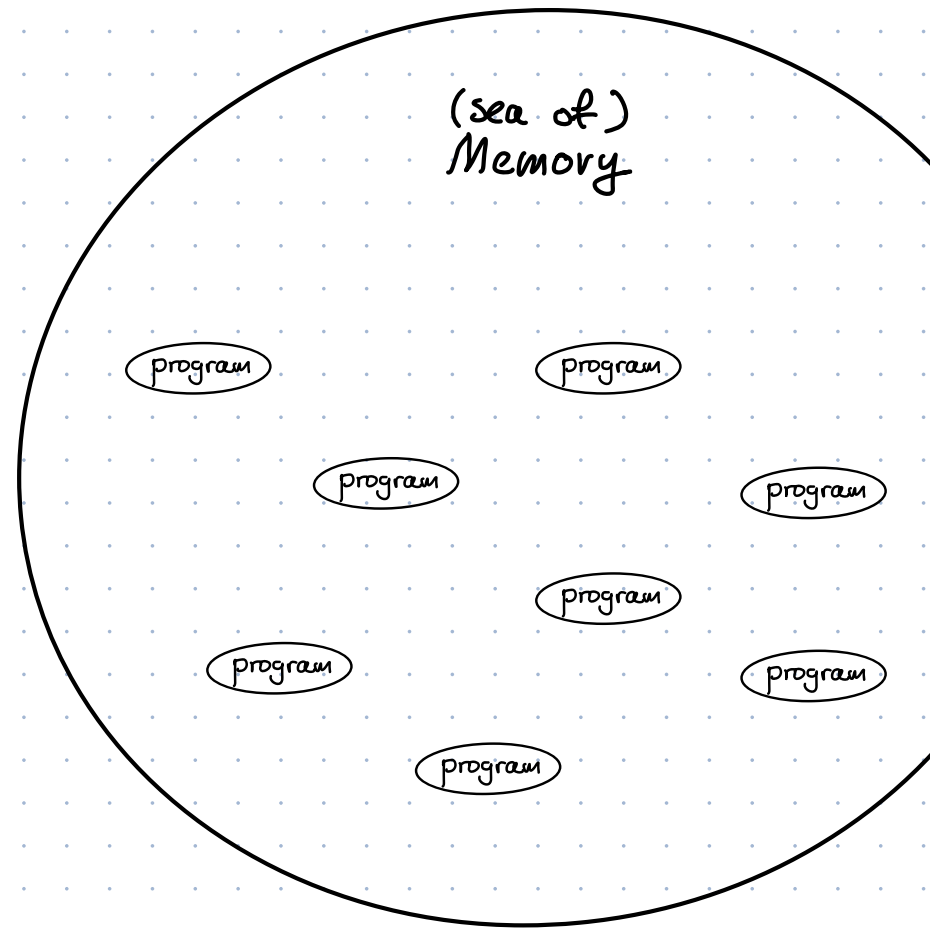


Changing conception of computing

Contributions to new paradigm:

- Networks and internet
- Slowing of Moore's law / multiprocessing
- Parallelism, GPUs, SIMD.

Emerging Picture
(last 30 years)

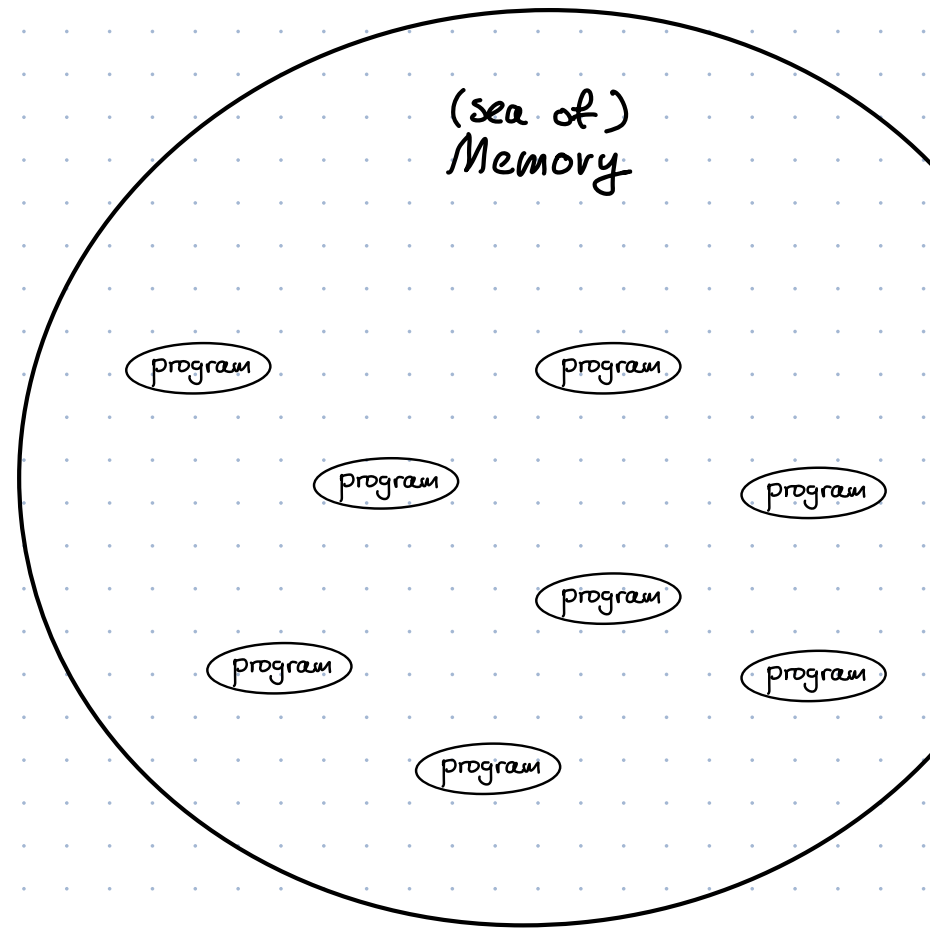


Changing conception of computing

Contributions to new paradigm:

- Networks and internet
- Slowing of Moore's law / multiprocessing
- Parallelism, GPUs, SIMD.
- Still big gap between what happens on computer and biological systems.

Emerging Picture
(last 30 years)



What is a neural network?

Typically one thinks of a program as being a discrete object.

What is a neural network?

Typically one thinks of a program as being a discrete object.

A neural network is a program that depends on parameters

$\theta \in \mathbb{R}^n$ that one can train by gradient descent.

What is a neural network?

Typically one thinks of a program as being a discrete object.

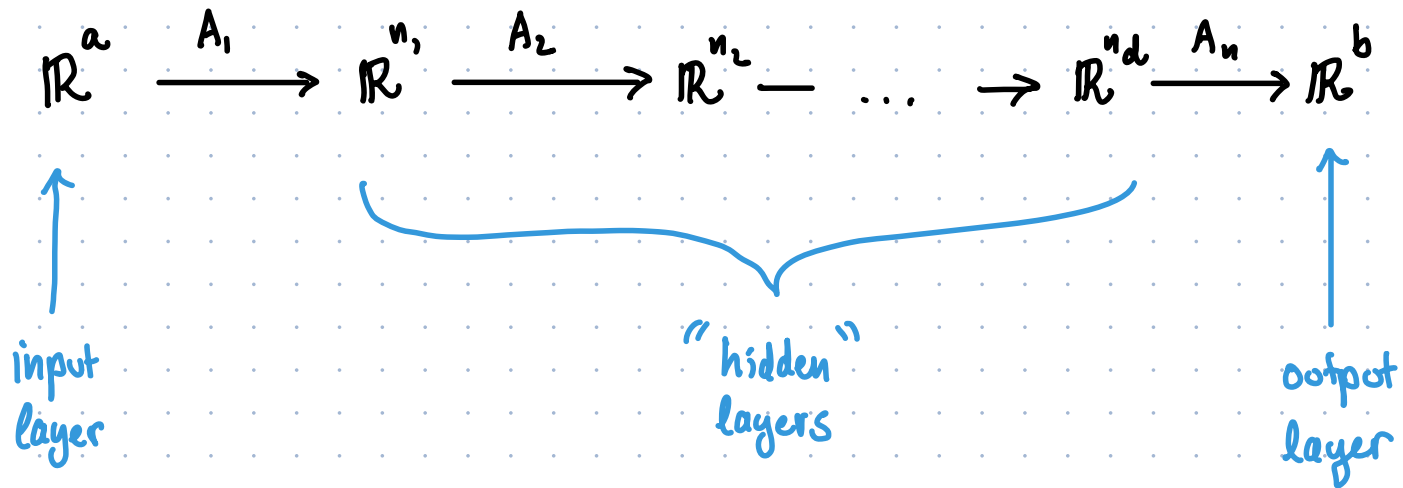
A neural network is a program that depends on parameters

$\theta \in \mathbb{R}^n$ that one can train by gradient descent.

Typically one discusses a specific model, but one

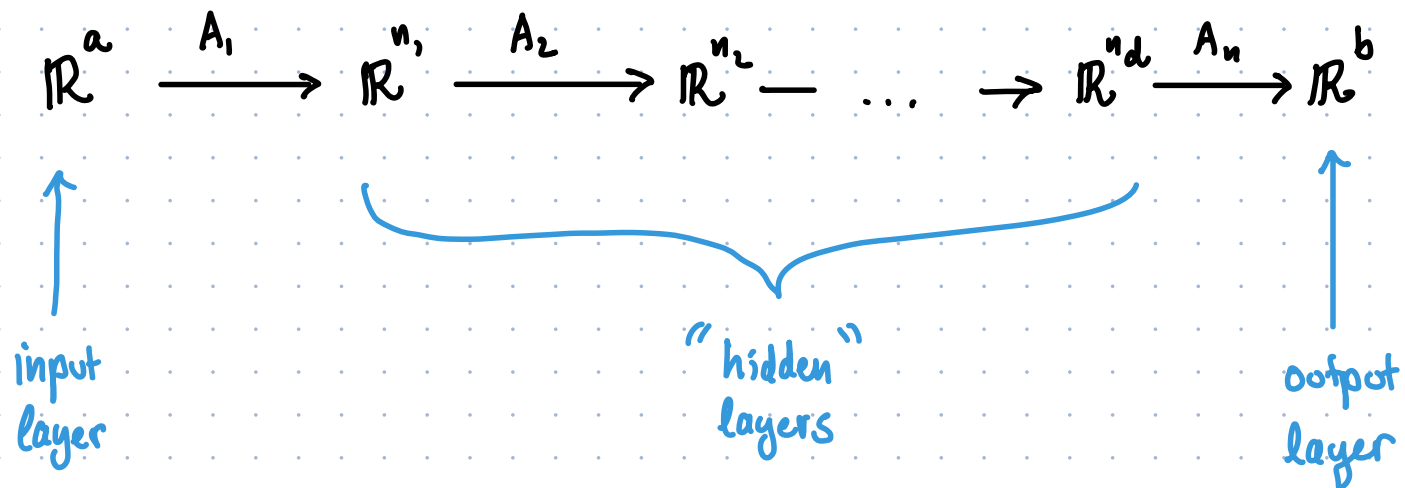
should keep general picture in mind.

What is a neural network?



A_i affine linear (i.e. $A_i(v) = M_i(v) + b_{i+1}$) "bias"

What is a neural network?



A_i affine linear (i.e. $A_i(v) = M_i(v) + b_{i+1}$) "bias"

$$f : \mathbb{R}^a \longrightarrow \mathbb{R}^b : v \mapsto A_n(\phi(\dots \phi(A_2(\phi(A_1, v))) \dots))$$

ϕ = coordinate-wise application of non-linearity

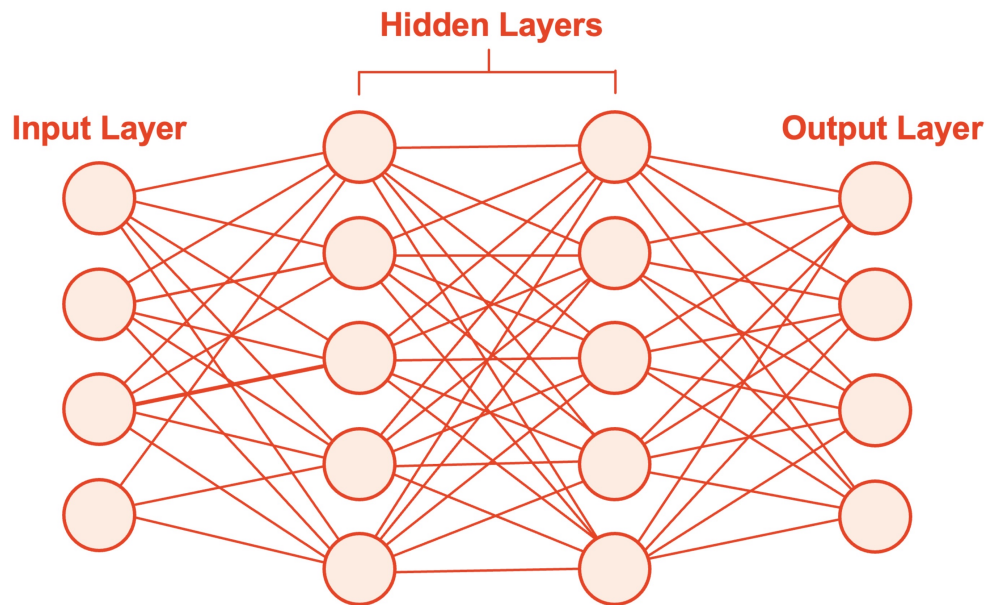
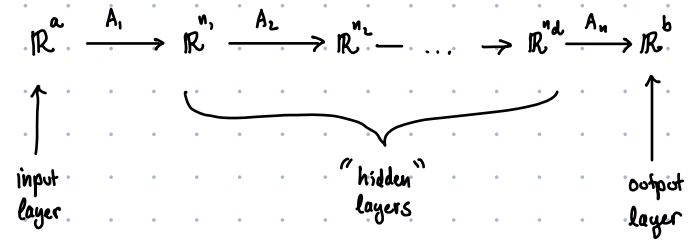
e.g.

$$\text{ReLU}(x) = \max(0, x)$$

or

$$\tanh(x)$$

What is a neural network?

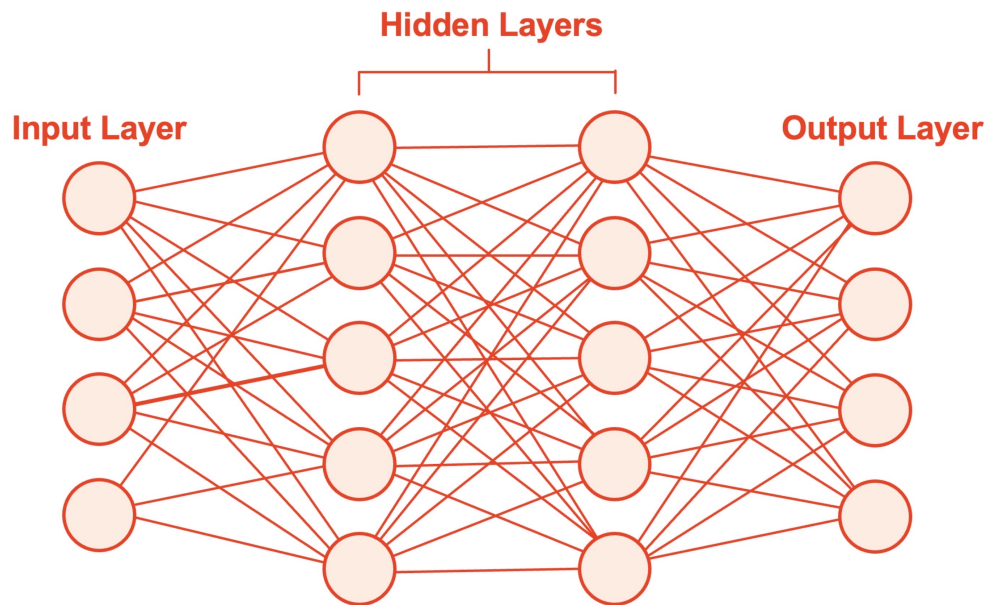
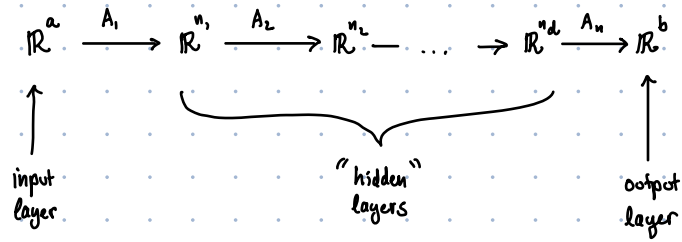


Alternatively, can think of

as network of functions

$$x \rightarrow \max(ax + b, c)$$

What is a neural network?



Alternatively, can think of

as network of functions

$$x \rightarrow \max(ax + b, c)$$

- local computation
- NOT sparse
- all "neurons" compute similar functions.

What is a neural network?

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$

→ large collection of functions parametrized by $A_1, \dots, A_{d+1}$.

(MANY real numbers, typically between -1 and 1.)

What is a neural network?

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$

→ large collection of functions parametrized by $A_1, \dots, A_{d+1}$.

(MANY real numbers, typically between -1 and 1.)

→ functions are "eventually linear in all directions"

→ typically only care about values on bounded regions $B^a \subset \mathbb{R}^a$ (e.g. unit cube)

What is a neural network?

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$

PARAMETERS
 $\cap$
 $\mathbb{R}^{\text{big}}$

all continuous
functions $B^a \rightarrow \mathbb{R}^b$
 $\cap$
 $\mathbb{R}^a$

What is a neural network?

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$

PARAMETERS
 $\cap$
 $\mathbb{R}^{\text{big}}$

all continuous
functions $B^a \rightarrow \mathbb{R}^b$
 $\cap$
 $\mathbb{R}^a$

What is a neural network?

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$

PARAMETERS
 $\cap$
 $\mathbb{R}^{\text{big}}$

all continuous
functions $B^a \rightarrow \mathbb{R}^b$
 $\cap$
 $\mathbb{R}^a$

functions arising
from neural network

What is a neural network?

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$

As n_i get larger, we can approximate
any continuous function.

(Universal approximation theorem.)

PARAMETERS
 $\cap$
 $\mathbb{R}^{\text{big}}$

all continuous
functions $B^a \rightarrow \mathbb{R}^b$
 $\cap$
 $\mathbb{R}^a$

functions arising
from neural network

What is a neural network?

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$

As n_i get larger, we can approximate
any continuous function.

(Universal approximation theorem.)

PARAMETERS
 $\cap$
 $\mathbb{R}^{\text{big}}$

all continuous
functions $B^a \rightarrow \mathbb{R}^b$
 $\cap$
 $\mathbb{R}^a$

functions arising
from neural network

Typically
many to one
"overparameterized"

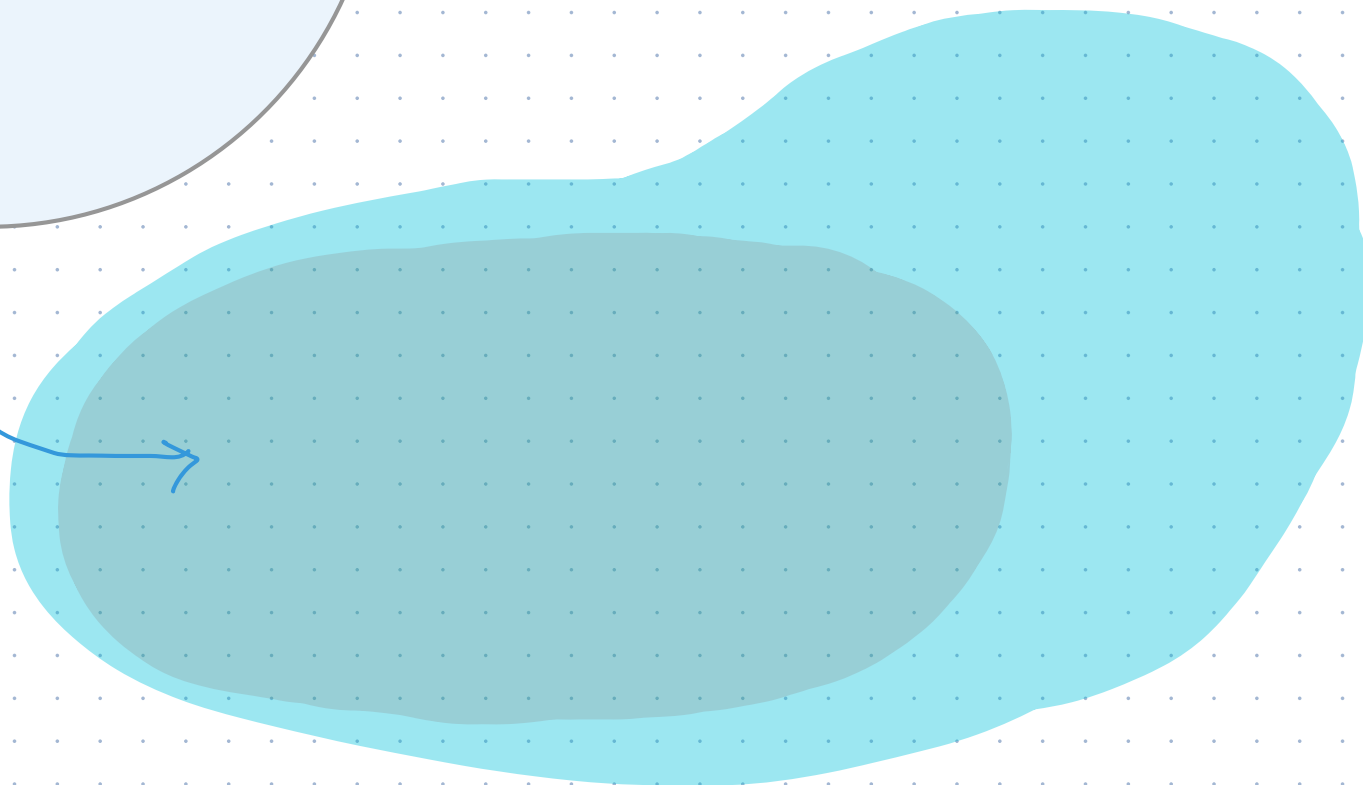
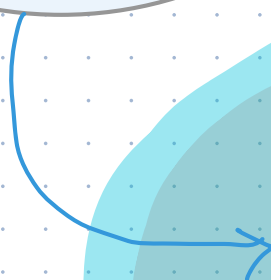
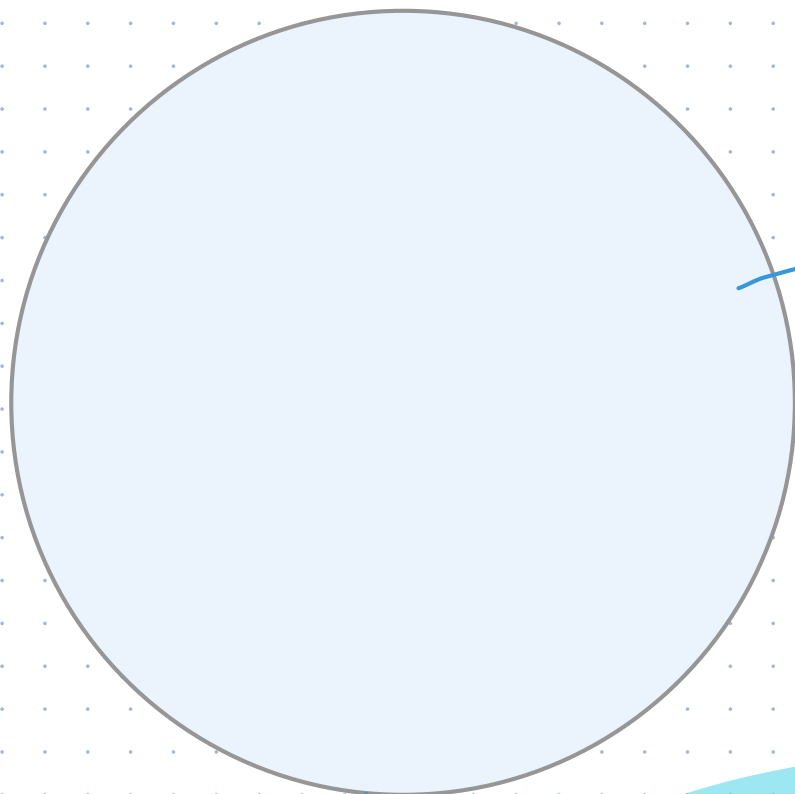
Training

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$

differentiable
loss function

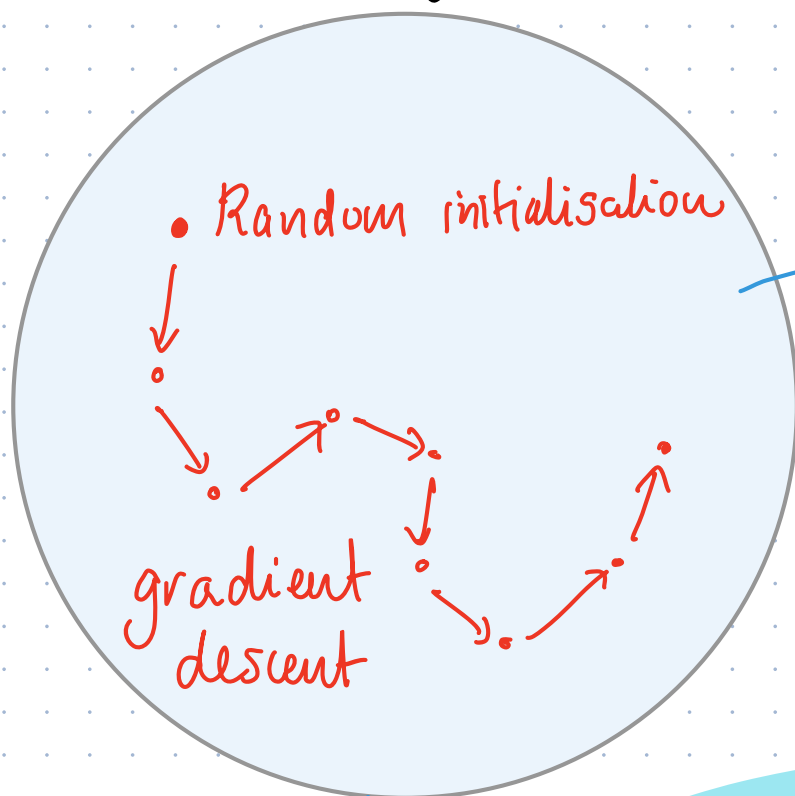
$\mathbb{R}$



Training

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$



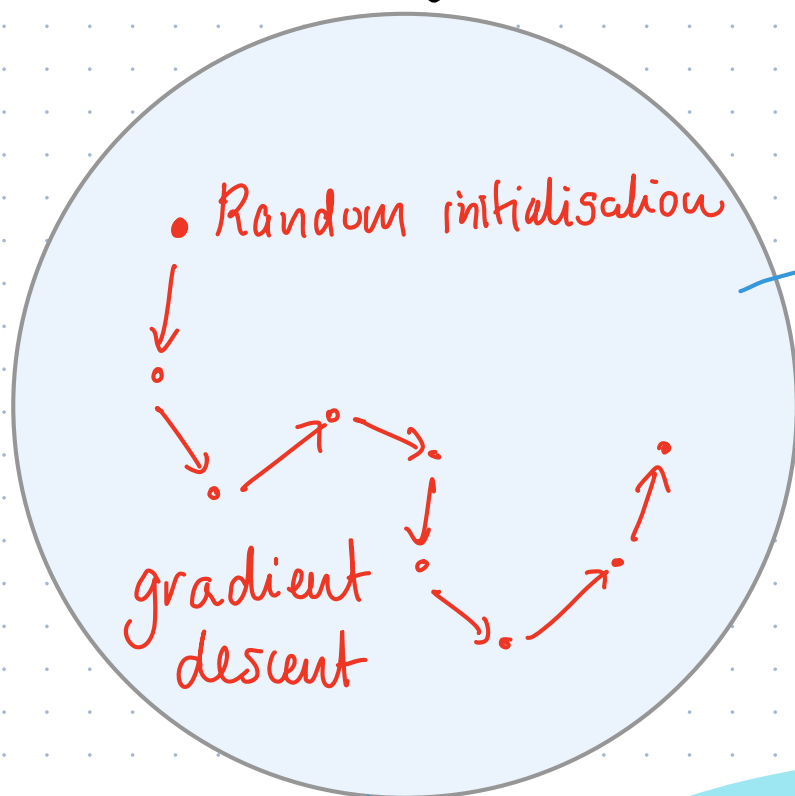
differentiable
loss function

$\mathbb{R}$

Training

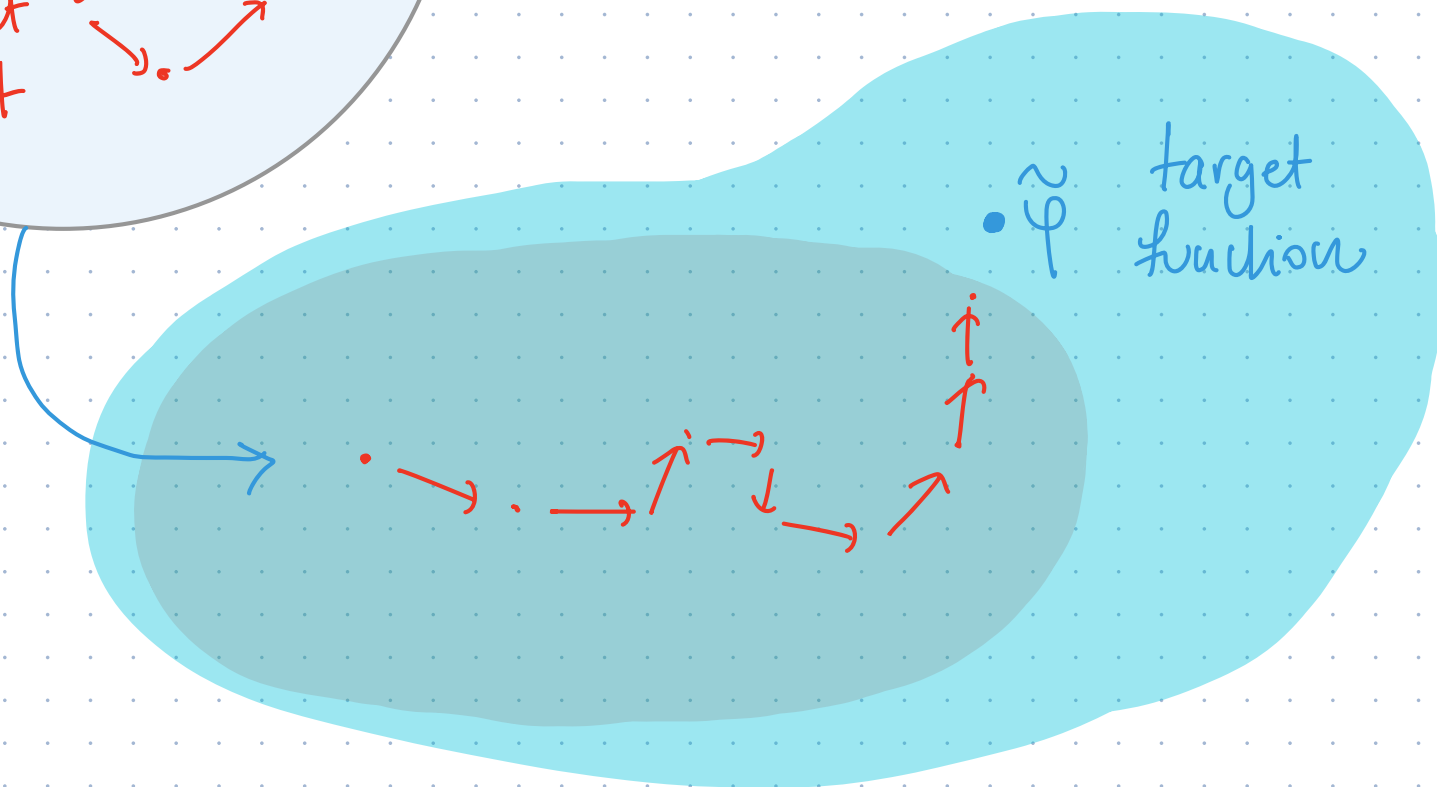
$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

$$A_i(v) = M_i(v) + b_i$$



differentiable
loss function

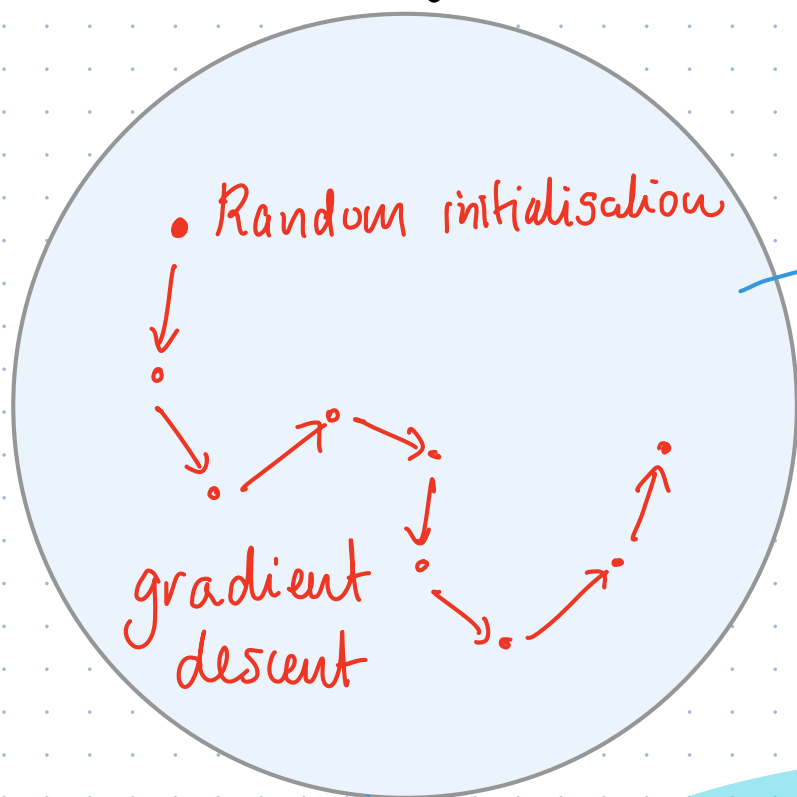
$\mathbb{R}$



Training

$$\mathbb{R}^a \xrightarrow{A_1} \mathbb{R}^{n_1} \xrightarrow{A_2} \mathbb{R}^{n_2} \dots \rightarrow \mathbb{R}^{n_d} \xrightarrow{A_{d+1}} \mathbb{R}^b$$

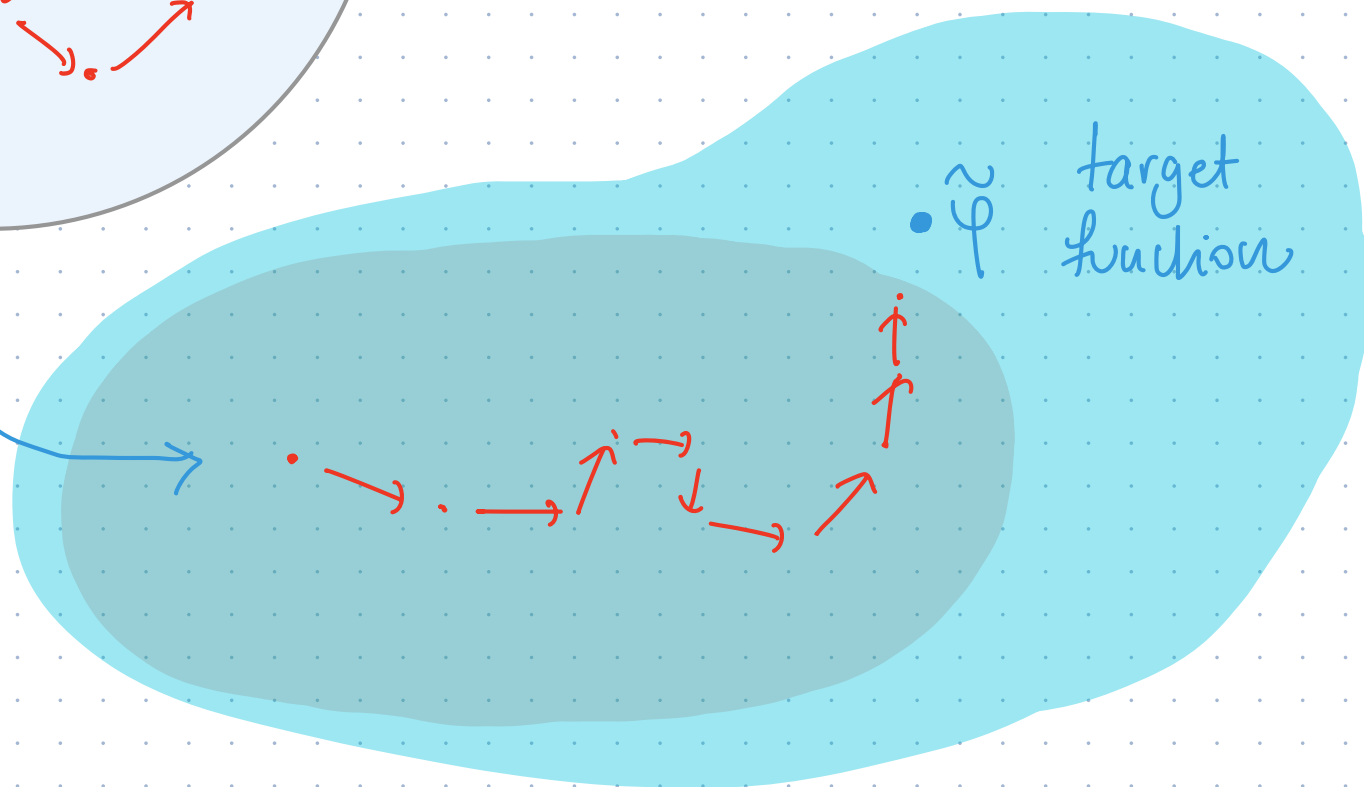
$$A_i(v) = M_i(v) + b_i$$



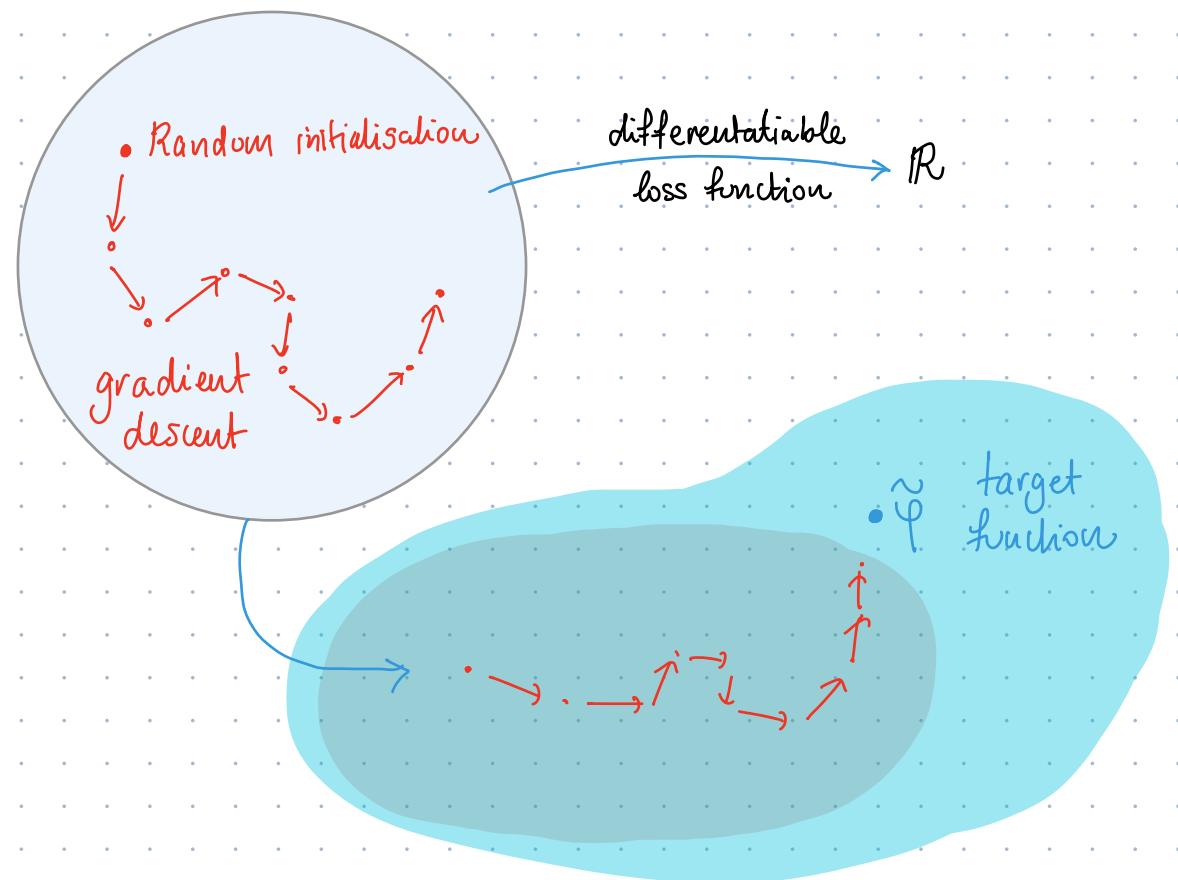
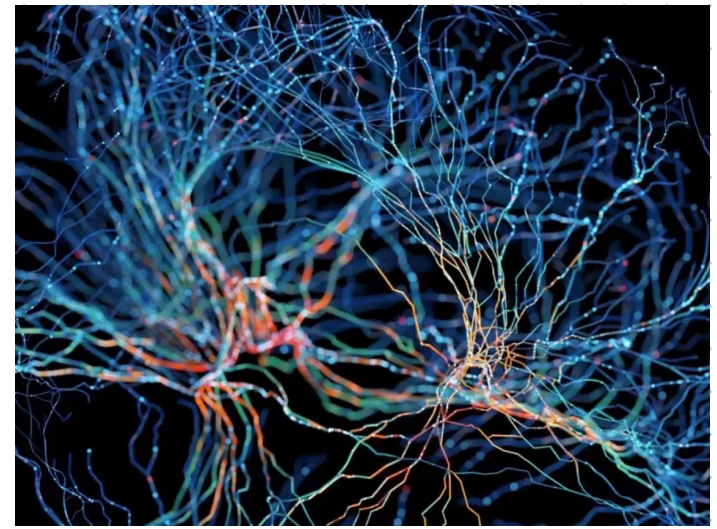
differentiable
loss function

$\mathbb{R}$

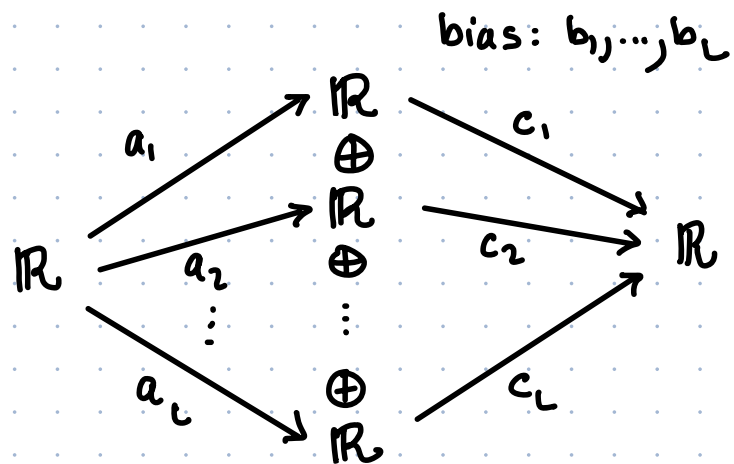
MANY
mysteries in
training dynamics!



Remark: No evidence that
gradient descent occurs
in the brain!



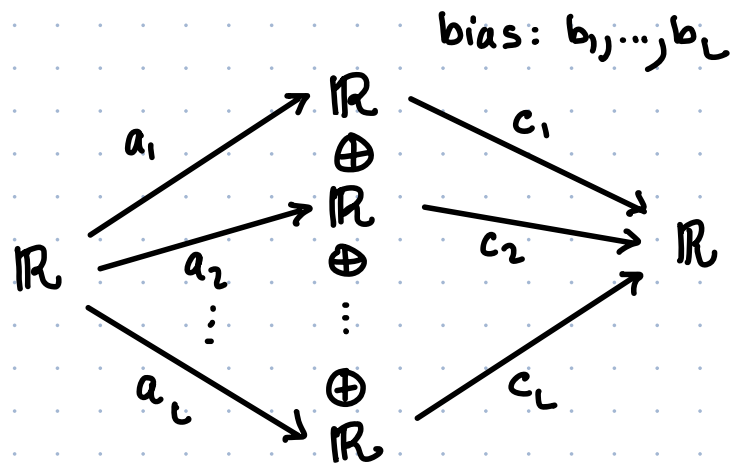
What do randomly initialised neural networks look like?



$a_i, b_i, c_i \in U(-1, 1)$
(uniform distribution)

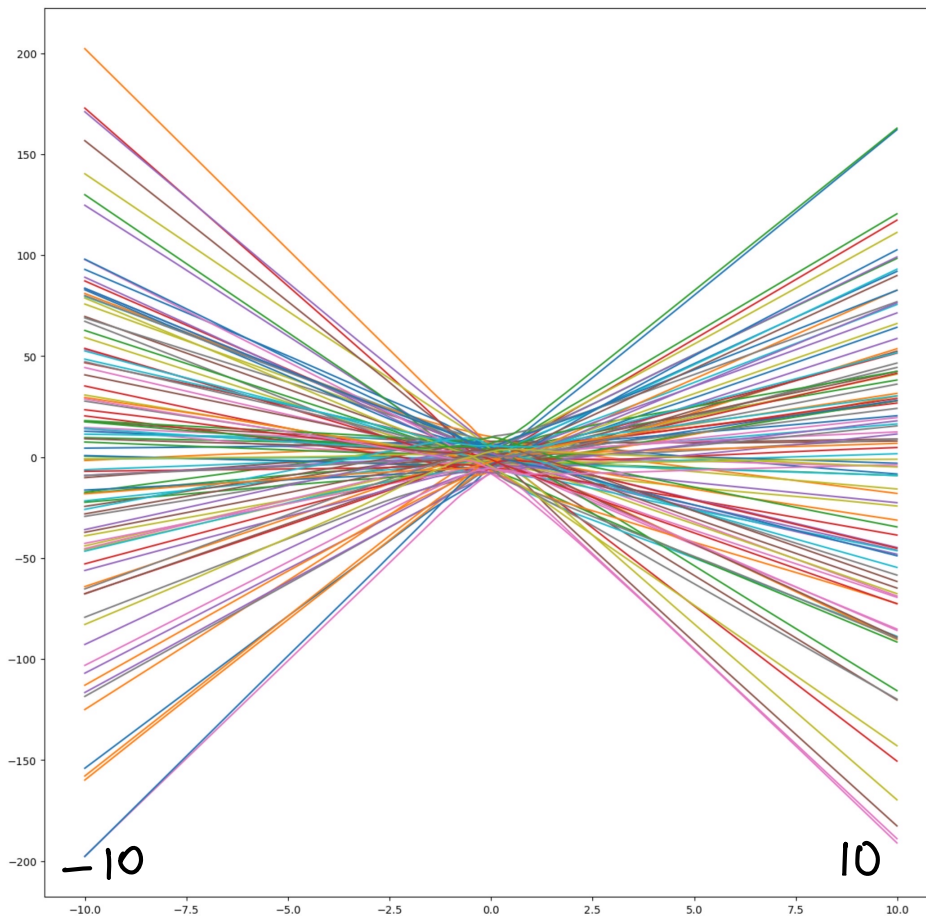
$$\mu(x) = \frac{1}{L} \sum_{i=1}^L c_i \phi(a_i x + b_i)$$

What do randomly initialised neural networks look like?

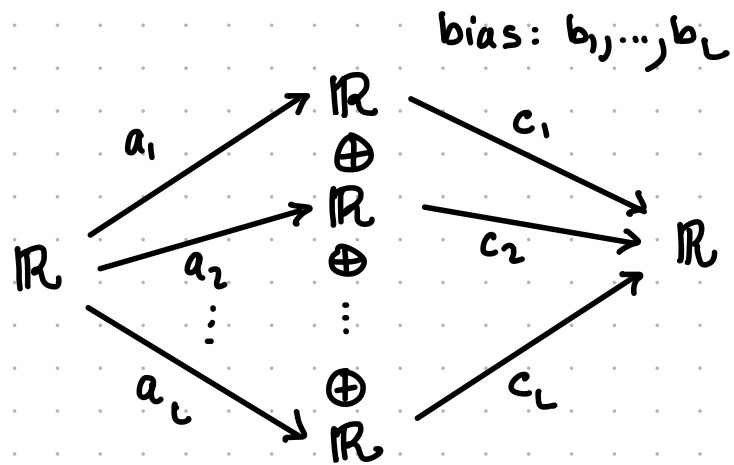


$a_i, b_i, c_i \in U(-1, 1)$
(uniform distribution)

$$\mu(x) = \frac{1}{L} \sum_{i=1}^L c_i \phi(a_i x + b_i)$$

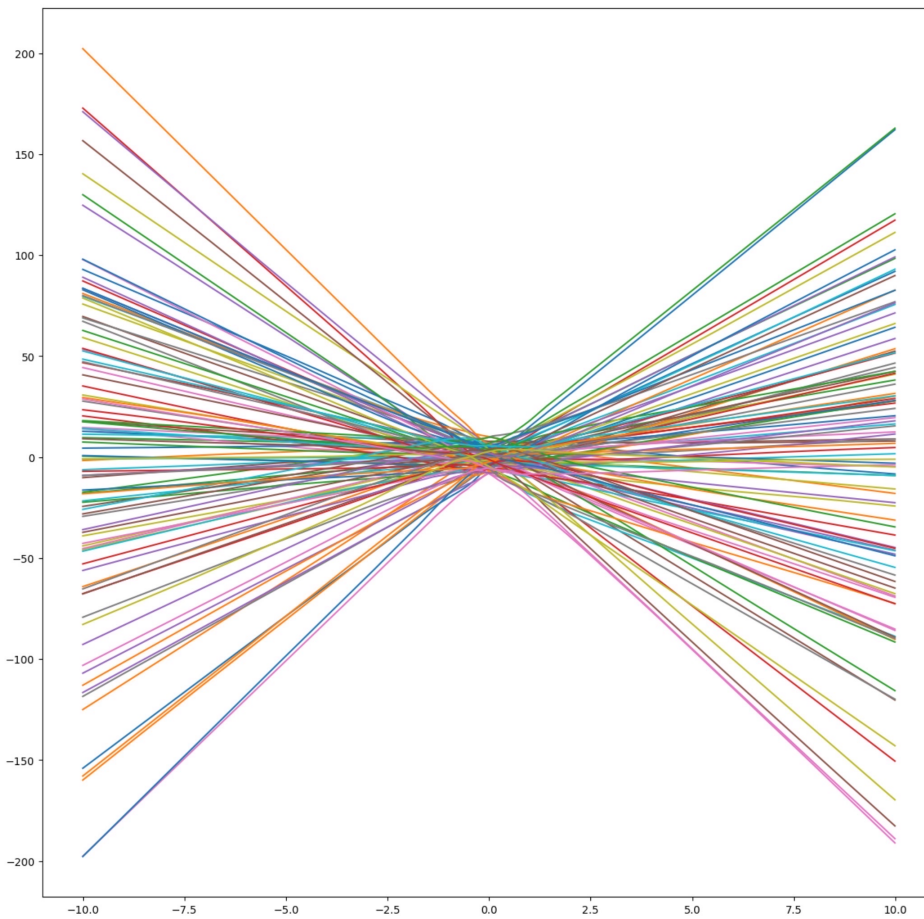


What do randomly initialised neural networks look like?

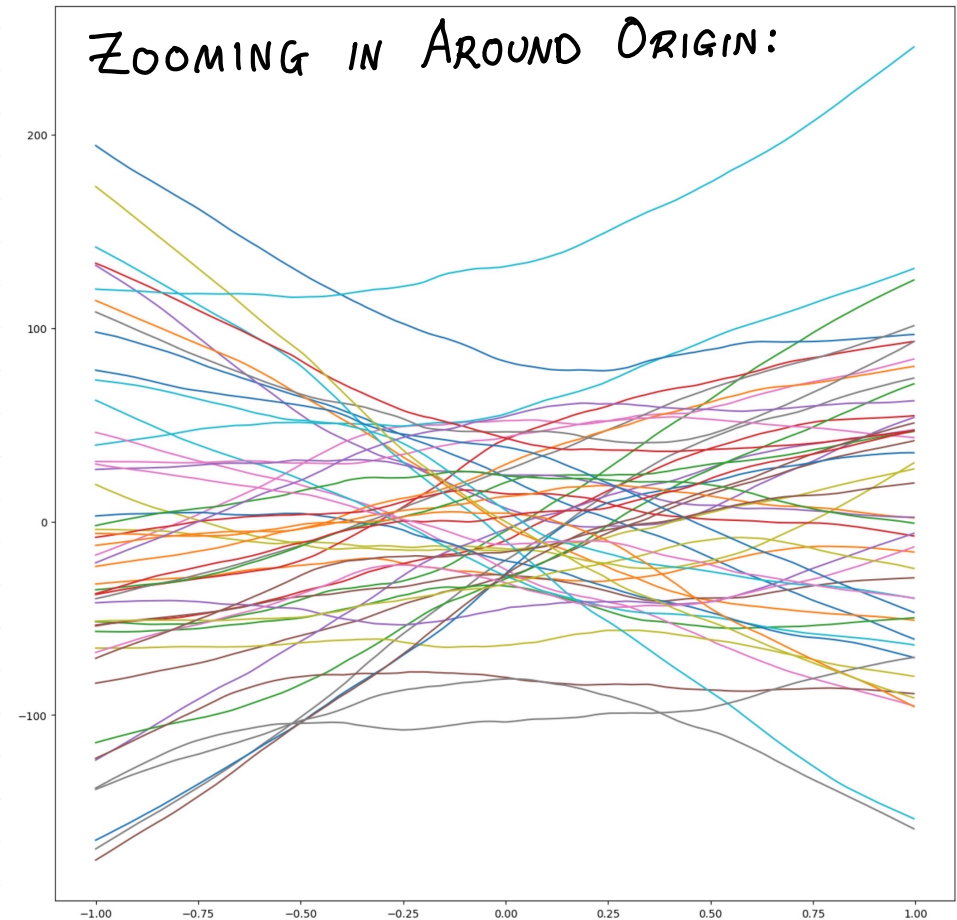


$a_i, b_i, c_i \in U(-1, 1)$
(uniform distribution)

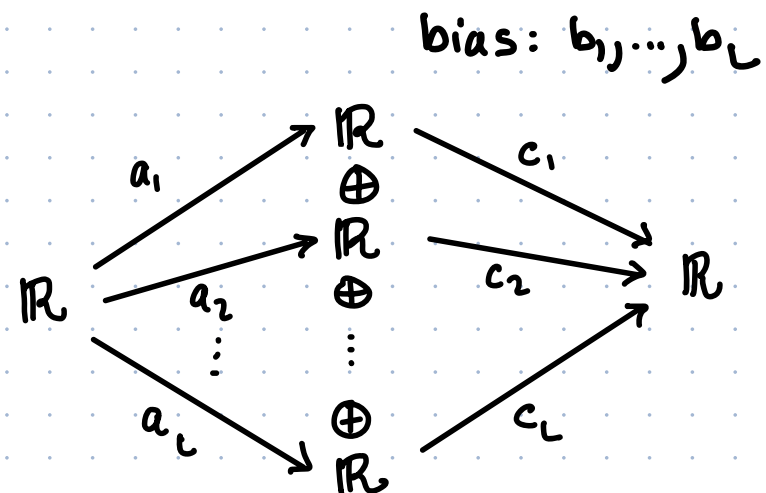
$$\mu(x) = \frac{1}{L} \sum_{i=1}^L c_i \phi(a_i x + b_i)$$



ZOOMING IN AROUND ORIGIN:



What do randomly initialised neural networks look like?



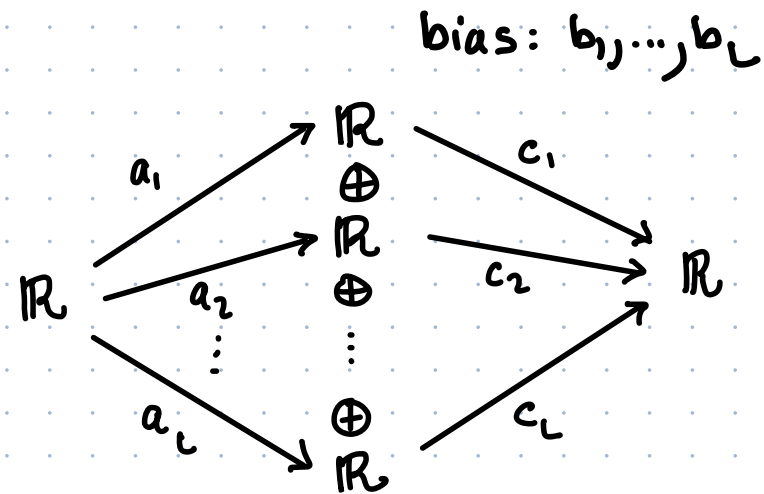
$a_i, b_i, c_i \in U(-1, 1)$
(uniform distribution)

$$\mu(x) = \frac{1}{L} \sum_{i=1}^L c_i \phi(a_i x + b_i)$$

For x fixed and $L \rightarrow \infty$, $\mu(x)$ becomes Gaussian w mean 0.

As $L \rightarrow \infty$, initializations look like "random functions".
(around the origin)

What do randomly initialised neural networks look like?



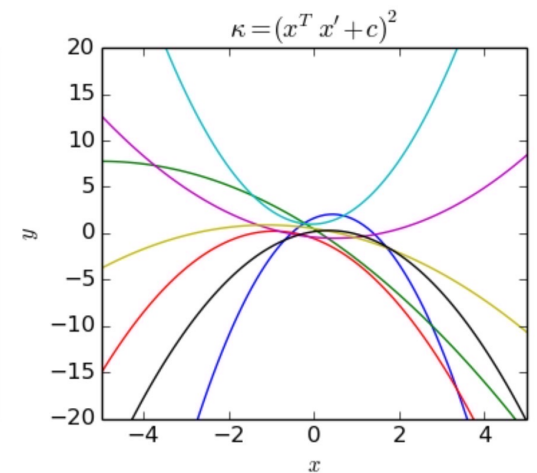
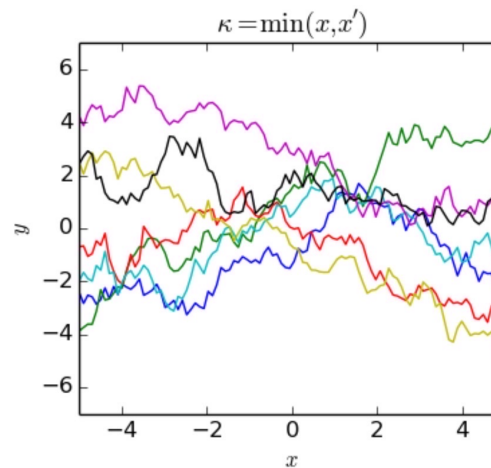
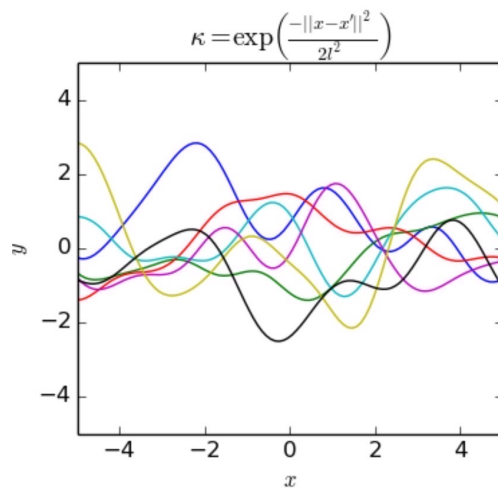
$$a_i, b_i, c_i \in U(-1, 1)$$

(uniform distribution)

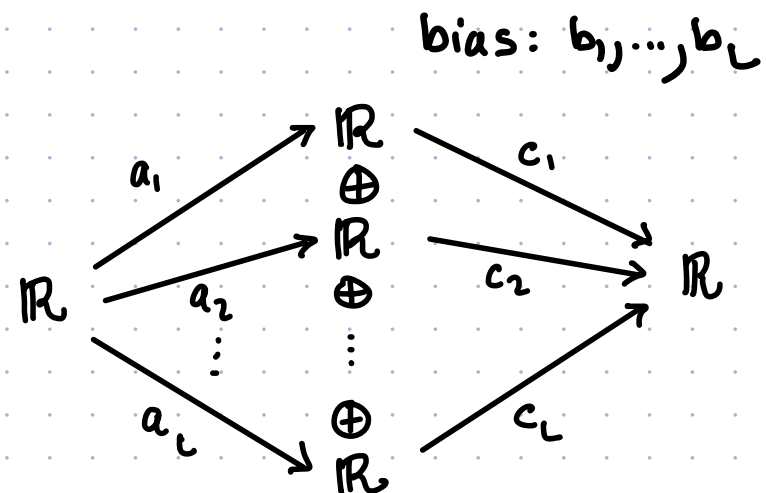
$$\mu(x) = \frac{1}{L} \sum_{i=1}^L c_i \phi(a_i x + b_i)$$

As $L \rightarrow \infty$, initializations look like "random functions".

Gaussian
processes



What do randomly initialised neural networks look like?



$a_i, b_i, c_i \in U(-1, 1)$
(uniform distribution)

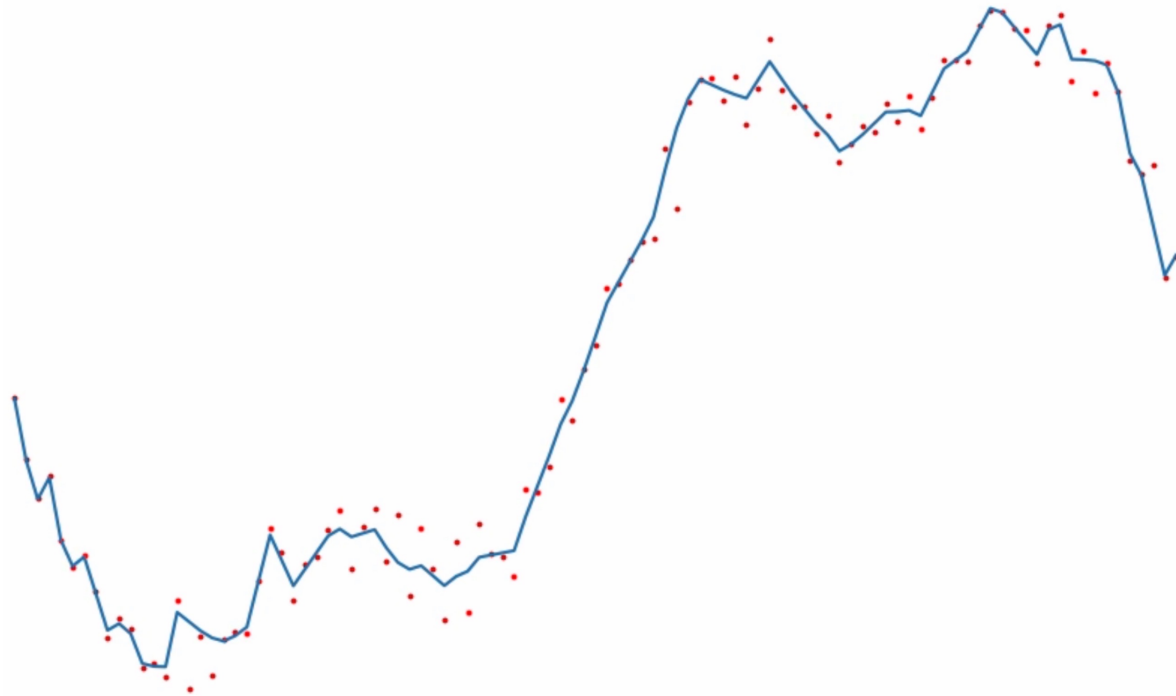
$$\mu(x) = \frac{1}{L} \sum_{i=1}^L c_i \phi(a_i x + b_i)$$

As $L \rightarrow \infty$, initializations look like "random functions".

Neal, Bayesian Learning for Neural Networks, 1994. (shallow case)

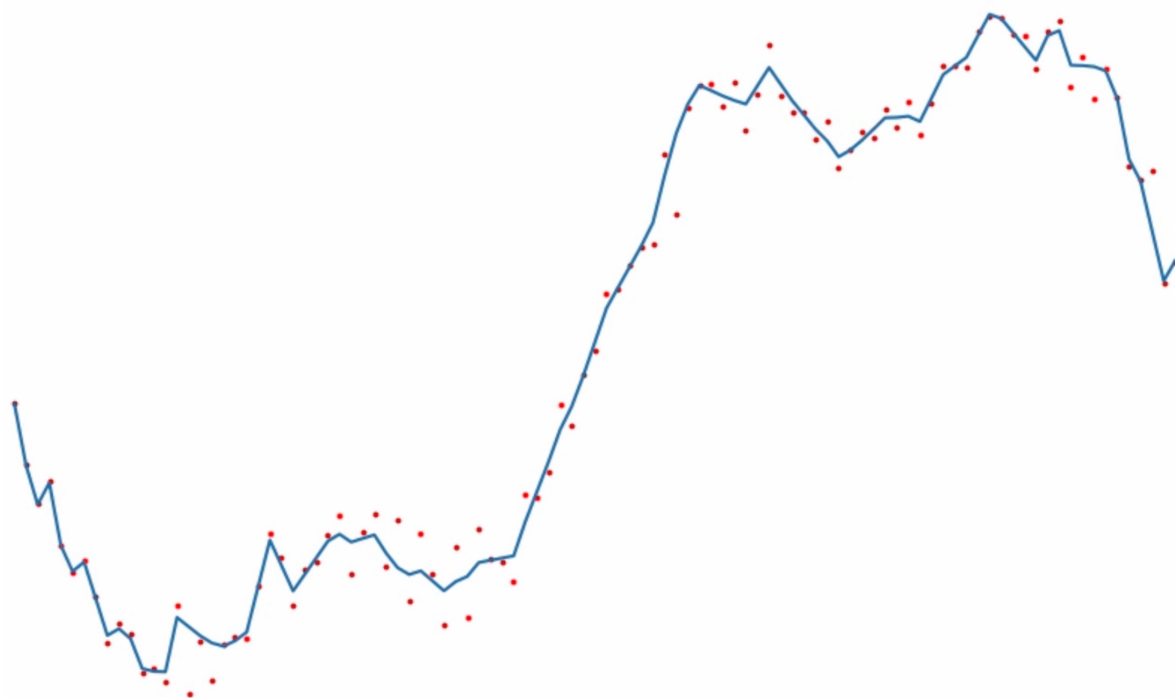
Lee et al., Deep Neural Networks as Gaussian Processes, 2018. (deep case)

Why not just use Gaussian processes,
or some other form of non-linear regression?



"... supervised neural networks are nothing more than
non-linear curve fitting devices."

Mackay, Information Theory, Inference, and Learning Algorithms, 2003



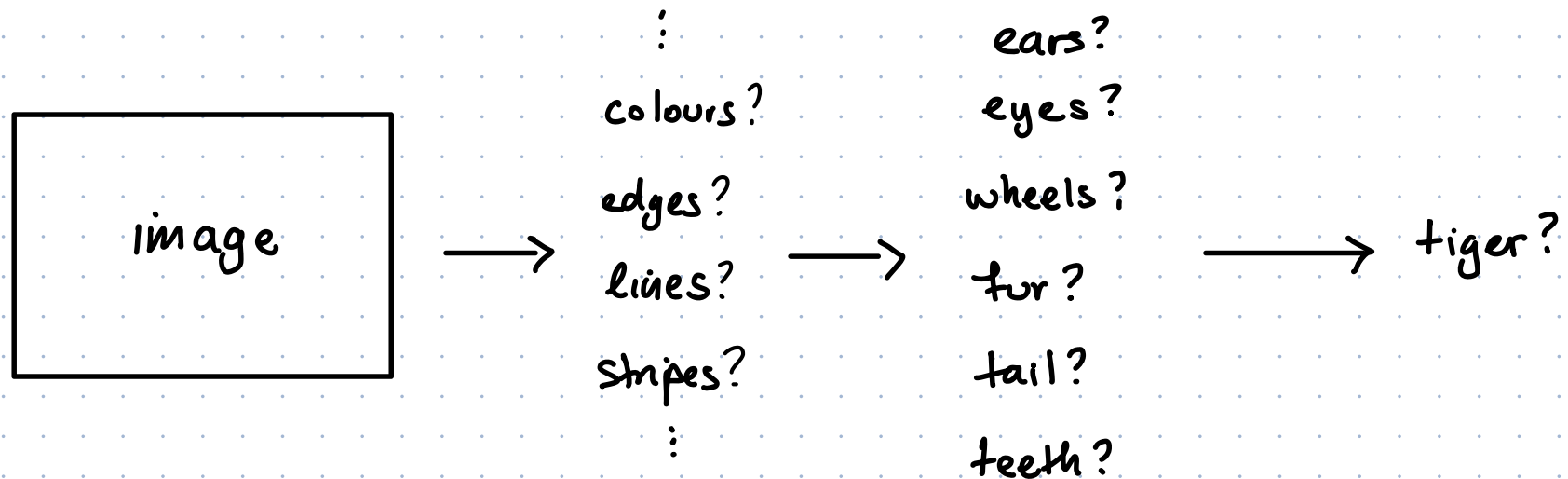
The importance of depth: $\mathbb{R}^n \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}^p$ "shallow" $\mathbb{R}^n \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}^m \rightarrow \dots \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}^p$
many layers = "deep"

"Feature learning" slogan: intermediate layers learn "features",
steadily simplifying task as we pass through neural network.

The importance of depth: $\mathbb{R}^n \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}$ "shallow" $\mathbb{R}^n \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}^m \rightarrow \dots \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}$
many layers = "deep"

"Feature learning" slogan: intermediate layers learn "features",

steadily simplifying task as we pass through neural network.



The importance of depth: $\mathbb{R}^n \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}$ "shallow" $\mathbb{R}^n \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}^m \rightarrow \dots \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}$
many layers = "deep"

"Feature learning" slogan: intermediate layers learn "features",
steadily simplifying task as we pass through neural network.

"As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, ..."

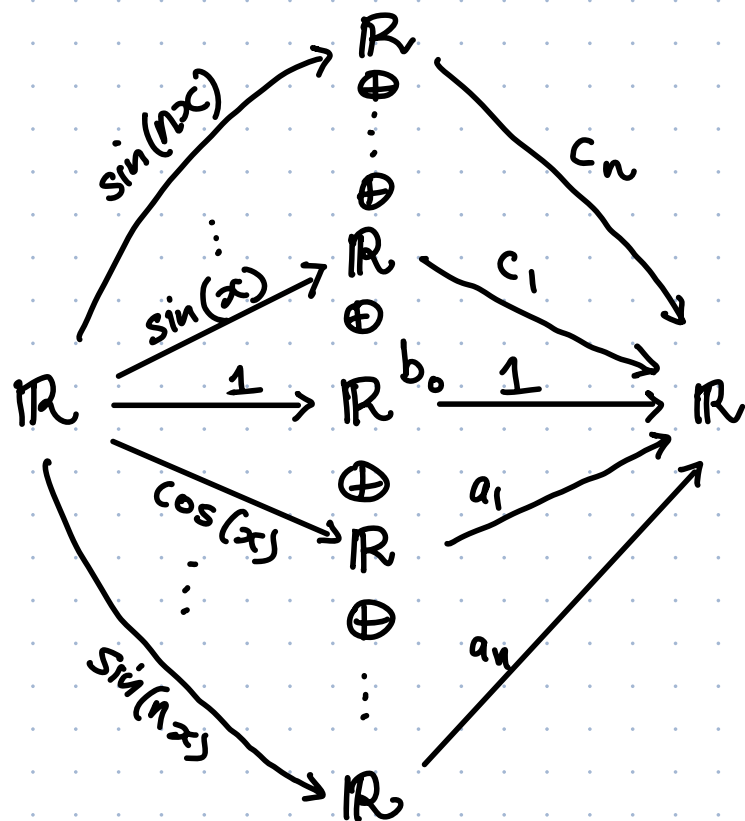
— Rumelhart, Hinton, Williams
Learning representations by back-propagating errors, 1986.

The importance of depth: $\mathbb{R}^n \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}$ "shallow" $\mathbb{R}^n \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}^m \rightarrow \dots \rightarrow \mathbb{R}^m \rightarrow \mathbb{R}$
many layers = "deep"

"Feature learning" slogan: intermediate layers learn "features",
simplifying task as we pass through neural network.

Open Problem: Although there is overwhelming
empirical evidence for feature learning,
our theoretical understanding is very limited.

Useful Analogy: Approximation via series or formulas



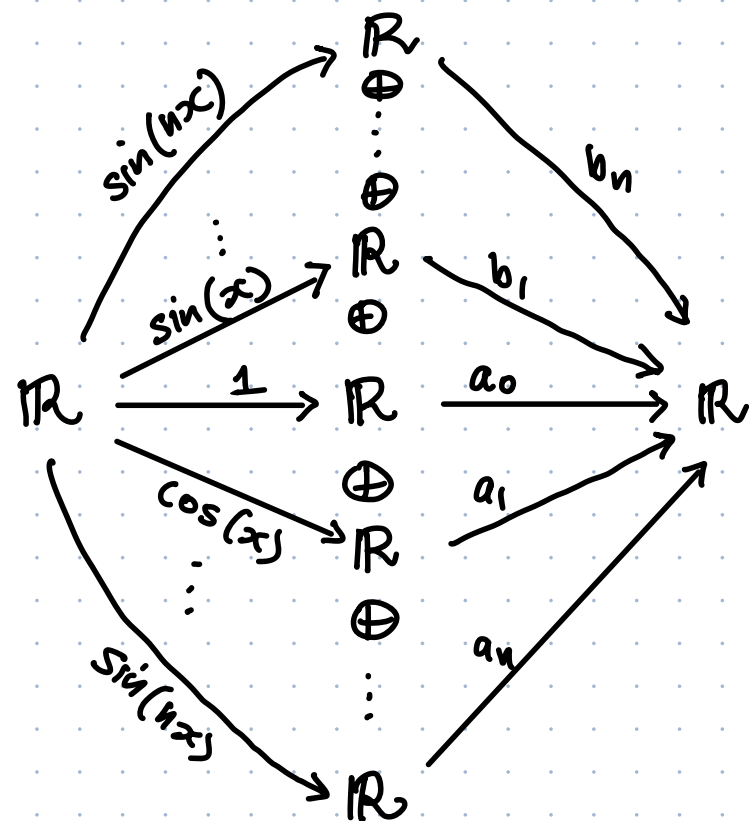
$$f \sim b_0 + \sum_{m=1}^n a_m \cos(mx) + c_m \sin(mx)$$

Here neural net training

||

approximation via Fourier series.

Useful Analogy: Approximation via series or formulas



$$f \sim a_0 + \sum a_n \cos(nx) + b_n \sin(nx)$$

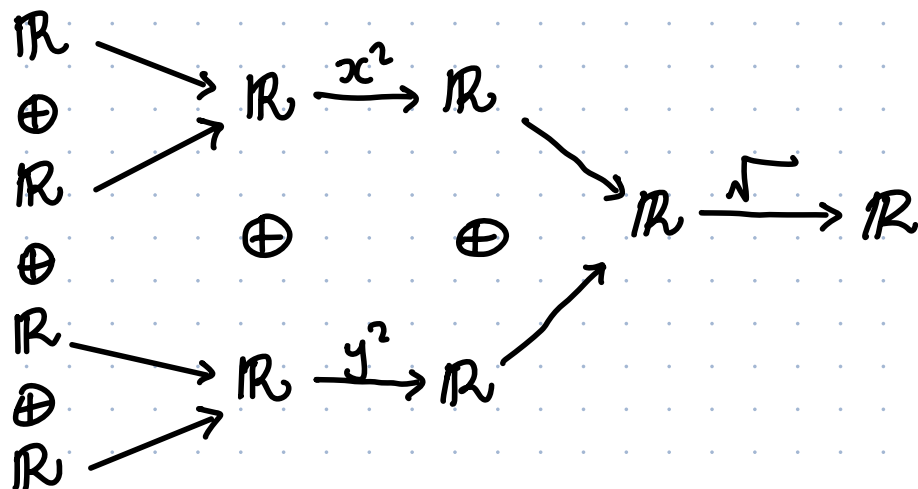
Here neural net training

||

approximation via Fourier series.

$$f \sim \sqrt{(x_1 - x_2)^2 + (x_3 - x_4)^2}$$

approximation of
compositional expressions



Useful Analogy: Approximation via series or formulas

shallow

SS

approximation in
classical regimes

deep

SS

approximation of
compositional expressions

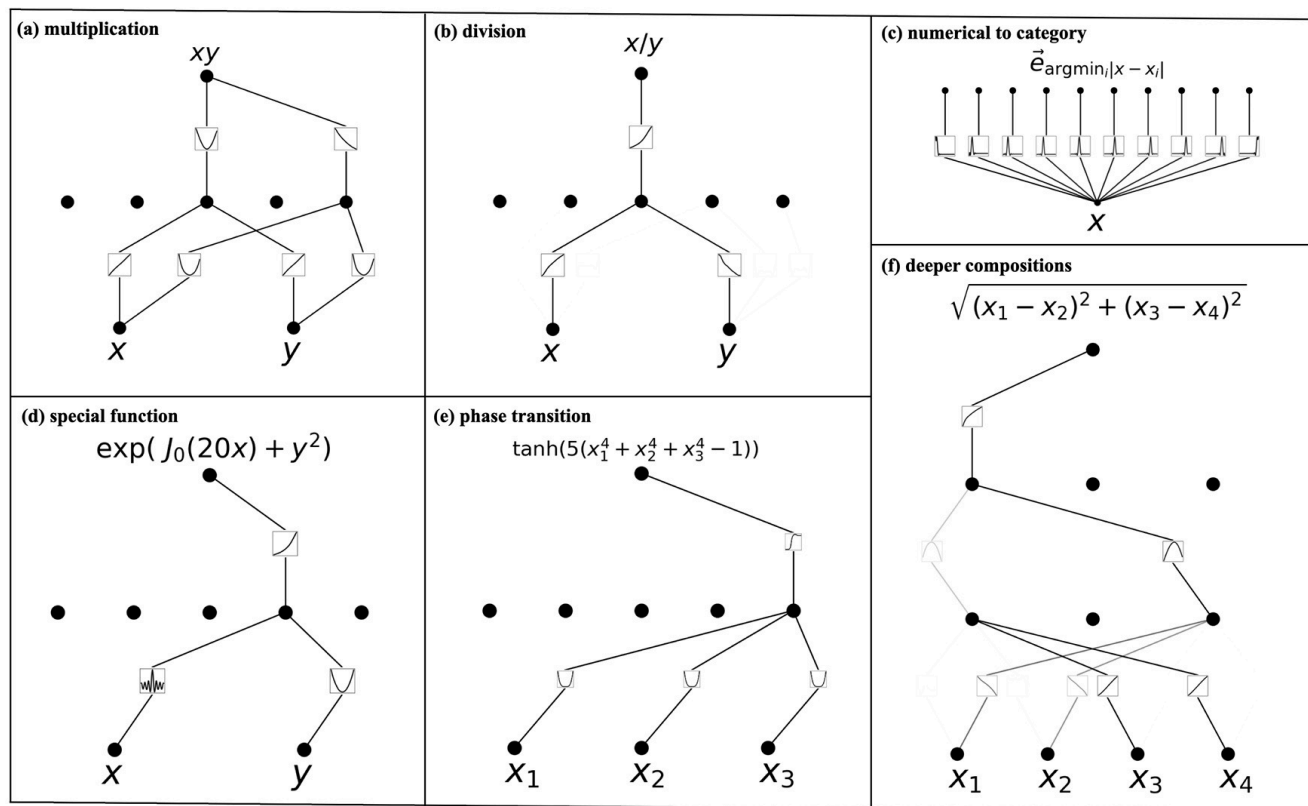
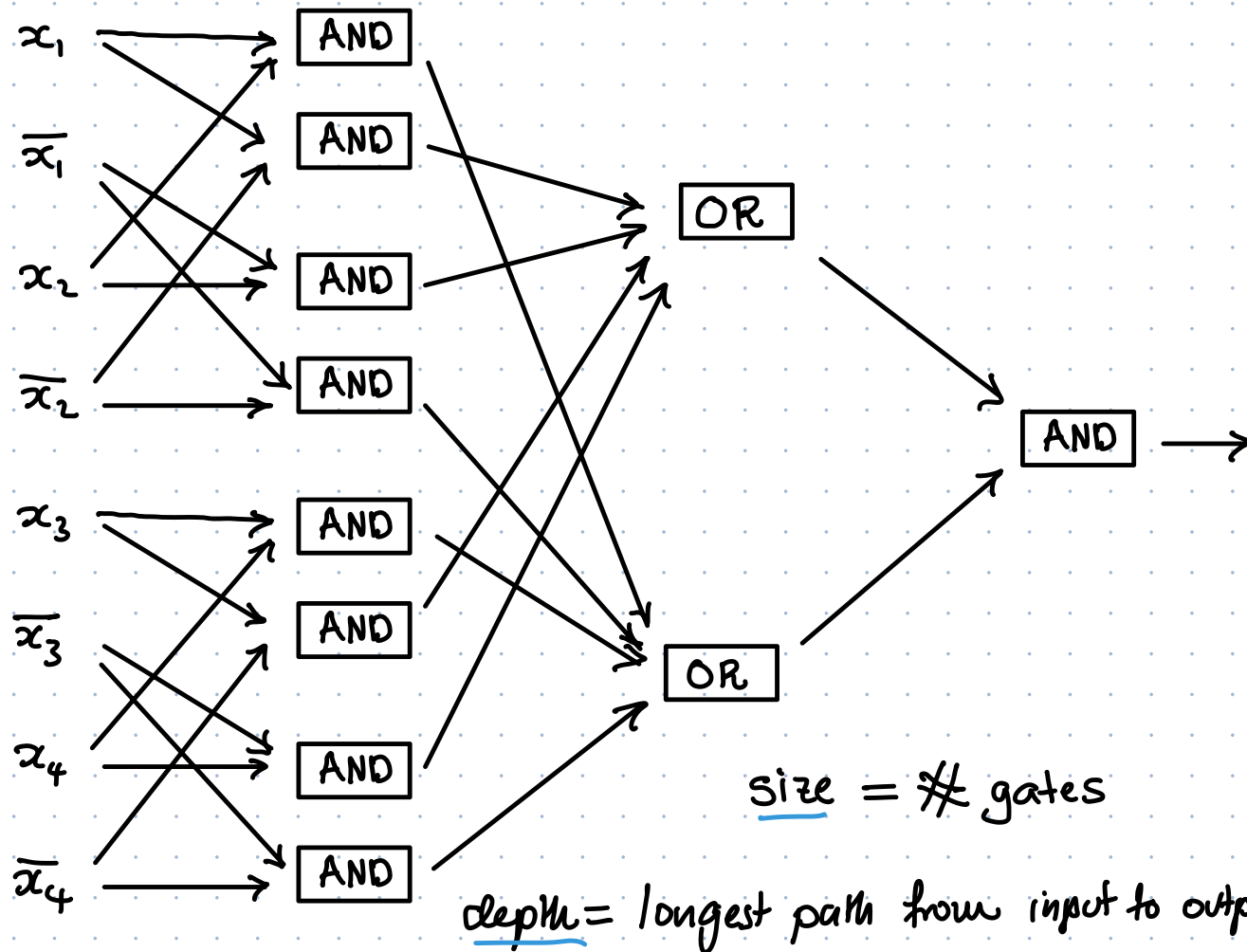


Figure 4.1: KANs are interpretable for simple symbolic tasks

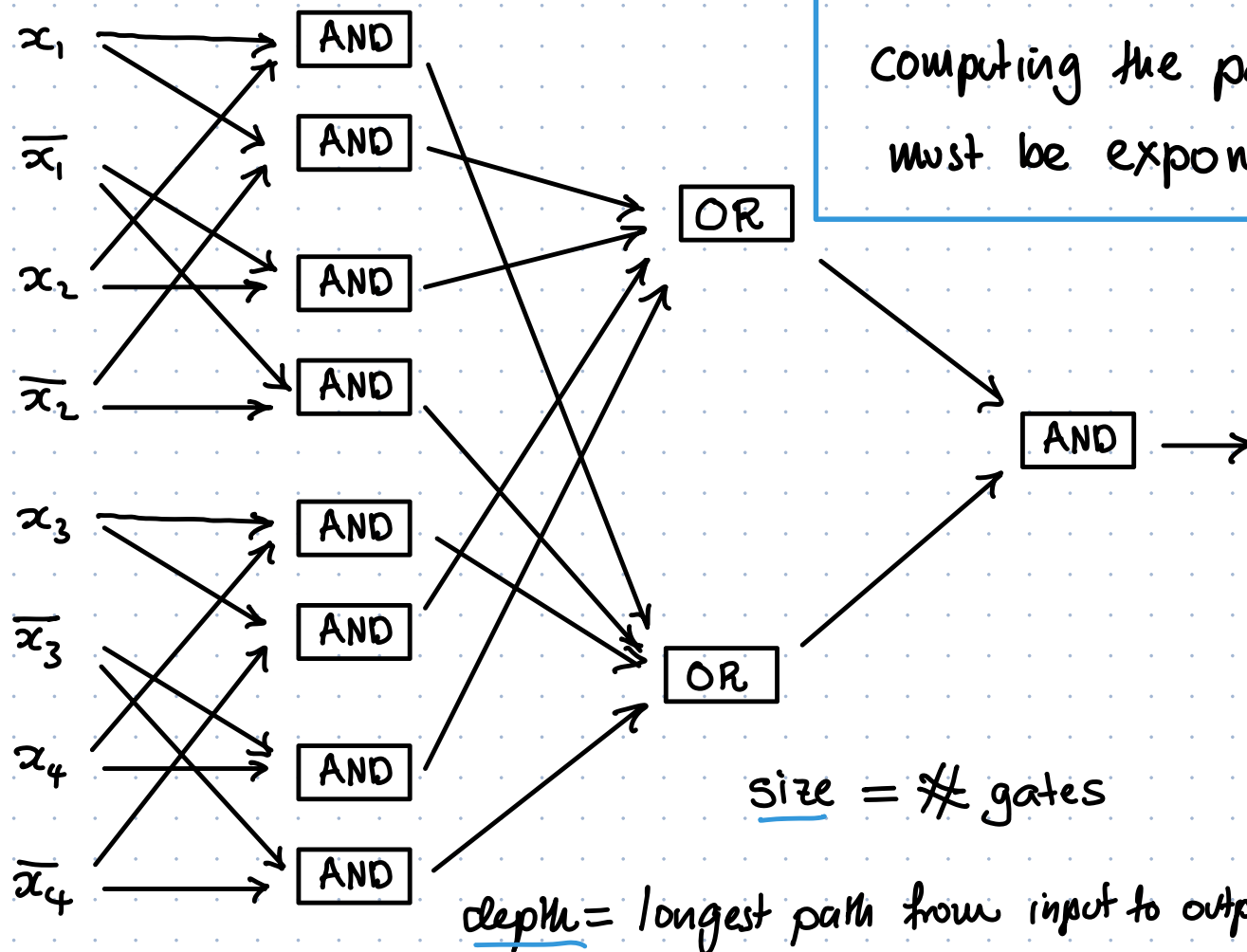
Useful Analogy: Circuit Complexity

A typical circuit:



Useful Analogy: Circuit Complexity

A typical circuit:



Håstad's Thm: "shallow circuits
computing the parity function
must be exponentially large."

Håstad, Computational Limitations for Small Depth Circuits, PhD thesis, 1986.

The overparametrized regime

"An over-complex model overfits the data and generalizes poorly."

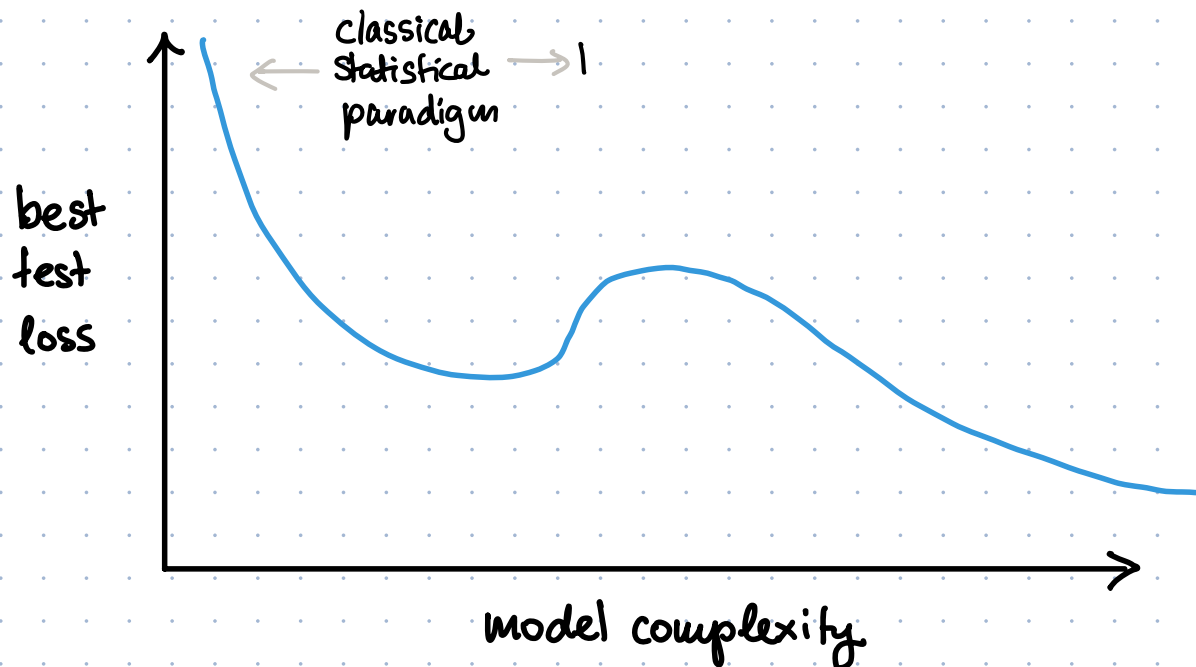
Mackay, Information Theory, Inference, and Learning Algorithms, 2003

The overparametrized regime

"An over-complex model overfits the data and generalizes poorly."

Mackay, Information Theory, Inference, and Learning Algorithms, 2003

Experience with deep learning directly contradicts this heuristic!



Belkin, Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation, 2021.

Transformers

Transformers are a remarkable architecture for "next token prediction".

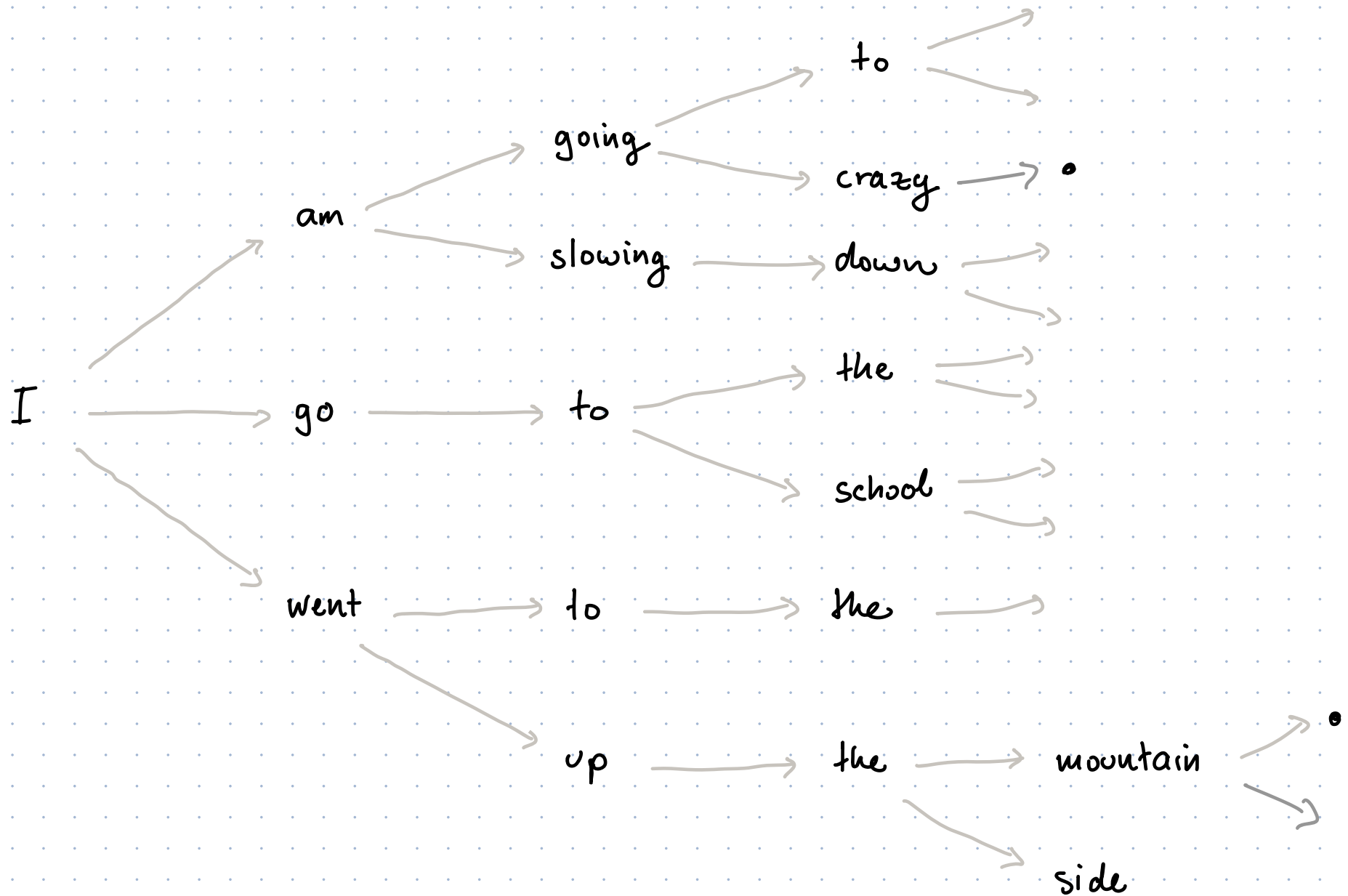
They strike a balance between ensuring sparsity whilst keeping great flexibility.

They are difficult to explain!

Best explanation: "Attention in transformers, step by step"

3Blue1Brown on youtube.

Language as a stochastic process



Language as a stochastic process

1. Zero-order approximation (symbols independent and equiprobable).

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZL-HJQD.

2. First-order approximation (symbols independent but with frequencies of English text).

OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL.

3. Second-order approximation (digram structure as in English).

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TU-COOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE.

4. Third-order approximation (trigram structure as in English).

IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE.

5. First-order word approximation. Rather than continue with tetragram, $\dots$, n -gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.

REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

6. Second-order word approximation. The word transition probabilities are correct but no further structure is included.

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

Language as a stochastic process

1. Zero-order approximation (symbols independent and equiprobable).

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD.

2. First-order approximation (symbols independent but with frequencies of English text).

OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL.

3. Second-order approximation (digram structure as in English).

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TU COOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE.

4. Third-order approximation (trigram structure as in English).

IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE.

5. First-order word approximation. Rather than continue with tetragram, $\dots$, n -gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.

REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

6. Second-order word approximation. The word transition probabilities are correct but no further structure is included.

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

"It would be interesting if further approximations could be constructed, but the labour involved becomes enormous at the next stage."

— Shannon, 1948

Scaling laws (Kaplan et al. 2020)

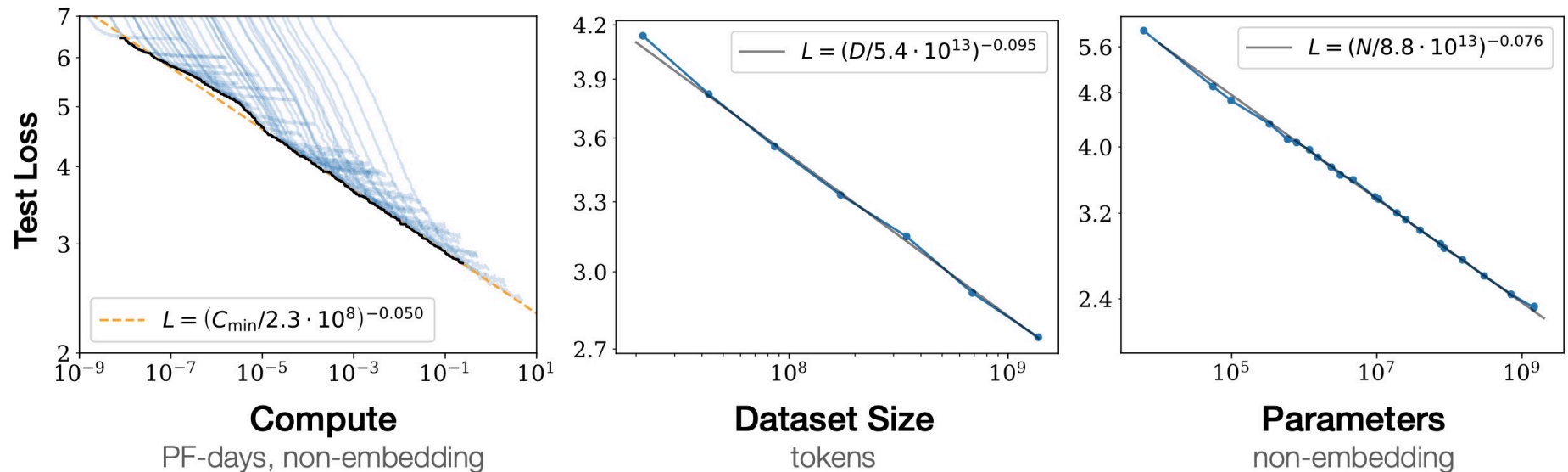


Figure 1 Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute² used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Basic idea

Want a map

sequences of tokens $\xrightarrow{\mathbb{f}}$ $\mathbb{R}^T$

so that

$$\mathbb{f}(t_1, \dots, t_{m-1}) = (\dots, \dots, 0.5, 0.01, \dots)$$

↑
probability that next
token is t_m

← set of tokens

← position t_m

Basic idea

Want a map

sequences of tokens $\xrightarrow{\mathbb{f}}$ $\mathbb{R}^T$

set of tokens

so that

$$\mathbb{f}(t_1, \dots, t_{m-1}) = (\dots, \dots, 0.5, 0.01, \dots)$$

position t_m

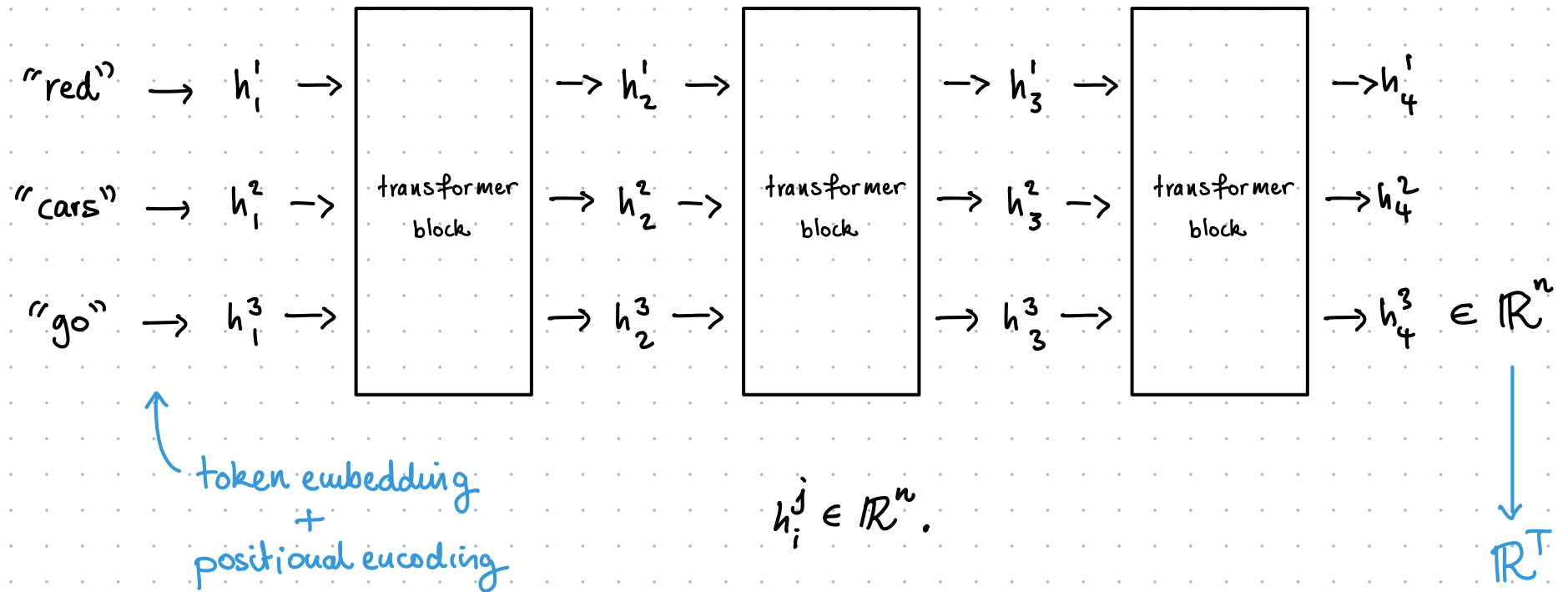
↑
probability that next
token is t_m

Assume this factors as

$$\text{sequences of tokens} \rightarrow \mathbb{R}^n \xrightarrow{\text{linear}} \mathbb{R}^T \xrightarrow{\text{softmax}} \mathbb{R}^T$$

this is the difficult bit!

$$(x_i) \mapsto \frac{1}{\sum \exp(x_i)} (\exp(x_i))$$



Guiding Principle: "transformers learn circuits"

This is a very compelling idea, and there is evidence for this principle in small algorithmic tasks.

see e.g. Poggio, Fraser Compositional Sparsity of Learnable Functions, 2024.

In-context learning Empirically, transformers can adapt to context:

maison, house, chien, dog, chat → cat

maison, Haus, chien, Hund, chat → Katze

In-context learning Empirically, transformers can adapt to context:

maison, house, chien, dog, chat $\rightarrow$ cat

Beautiful idea:

- 1) Fix some class $\mathcal{F}$ of functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$ (e.g. linear, ...)
- 2) Train transformer on $(x_1, f(x_1), x_2, f(x_2), \dots, x_m, f(x_m))$
over different choices of $f \in \mathcal{F}$.

In-context learning Empirically, transformers can adapt to context:

maison, house, chien, dog, chat $\rightarrow$ cat

Beautiful idea:

- 1) Fix some class $\mathcal{F}$ of functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$ (e.g. linear, ...)
- 2) Train transformer on $(x_1, f(x_1), x_2, f(x_2), \dots, x_m, f(x_m))$
over different choices of $f \in \mathcal{F}$.
- 3) Exhibits "meta learning": learns to learn function!
- 4) In some settings provably learns a regression algorithm!

In-context learning Empirically, transformers can adapt to context:

maison, house, chien, dog, chat $\rightarrow$ cat

Beautiful idea:

- 1) Fix some class $\mathcal{F}$ of functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$ (e.g. linear, ...)
- 2) Train transformer on $(x_1, f(x_1), x_2, f(x_2), \dots, x_m, f(x_m))$
over different choices of $f \in \mathcal{F}$.
- 3) Exhibits "meta learning": learns to learn function!
- 4) In some settings provably learns a regression algorithm!

Garg et al., What can transformers learn in context? (2023)

Zhang et al., Trained transformers learn linear models in context (2023)

Zhang et al., Training dynamics of in-context learning in linear Attention (2025)

Transformers on small algorithmic tasks

Consider tokens $0, 1, 2, \dots, 96 \in \mathbb{Z}/97\mathbb{Z}$.

Can a transformer learn $(x, y) \rightarrow x + y \bmod 97$?

Transformers on small algorithmic tasks

Consider tokens $0, 1, 2, \dots, 96 \in \mathbb{Z}/97\mathbb{Z}$.

Can a transformer learn $(x, y) \rightarrow x + y \bmod 97$?

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Transformers on small algorithmic tasks

Consider tokens $0, 1, 2, \dots, 96 \in \mathbb{Z}/97\mathbb{Z}$.

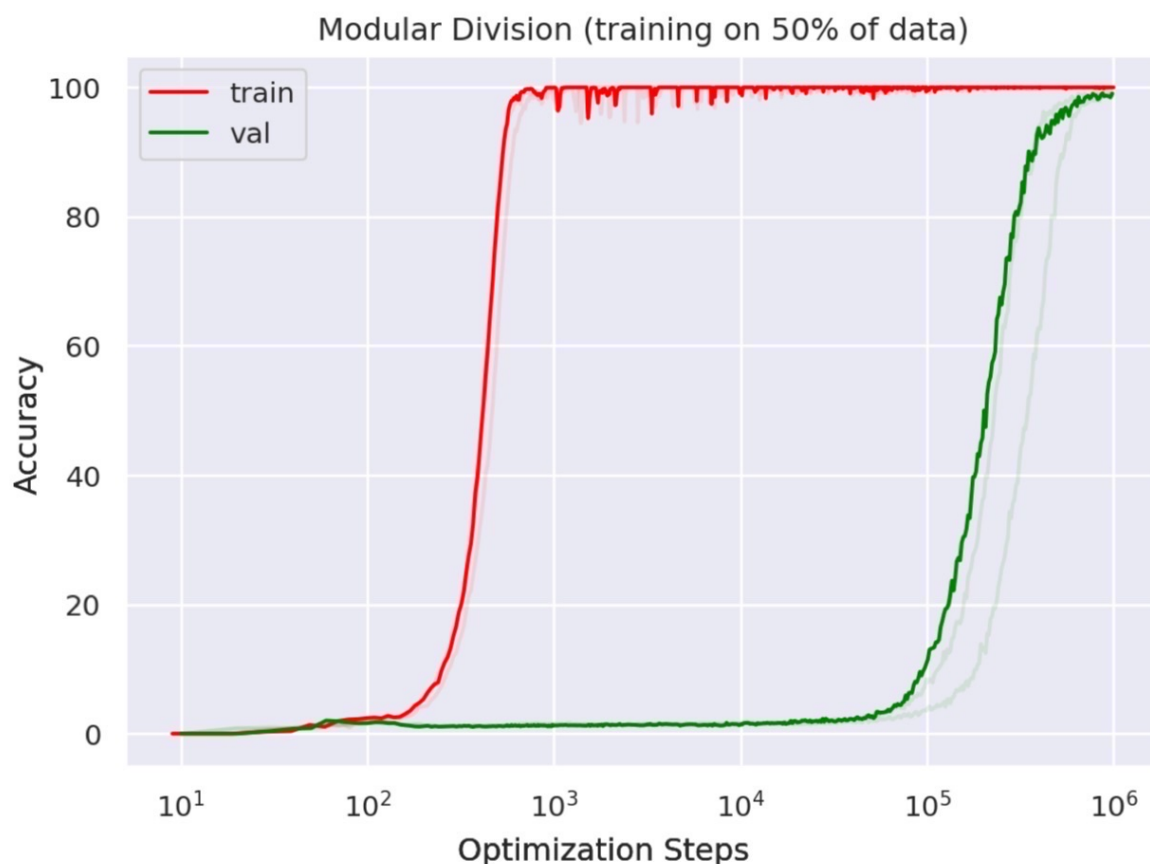
Can a transformer learn $(x, y) \rightarrow x + y \bmod 97$?

	0	1	2	3	4
0	0	?	2	?	4
1	?	2	3	?	?
2	2	?	4	0	1
3	?	4	?	1	2
4	4	0	?	2	?

Transformers on small algorithmic tasks

Consider tokens $0, 1, 2, \dots, 96 \in \mathbb{Z}/97\mathbb{Z}$.

Can a transformer learn $(x, y) \rightarrow x + y \bmod 97$?

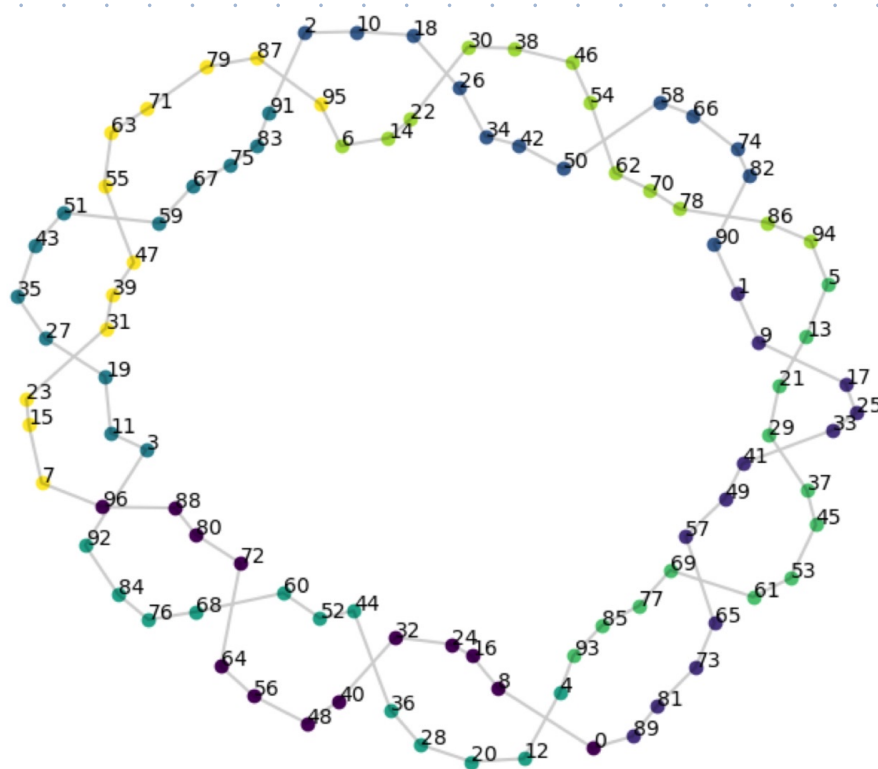


From Power et. al.

Transformers on small algorithmic tasks

Consider tokens $0, 1, 2, \dots, 96 \in \mathbb{Z}/97\mathbb{Z}$.

Can a transformer learn $(x, y) \rightarrow x + y \bmod 97$?



← dimensional
reduction (TSNE)
of output layer.

Power et. al. Grokking: Generalization
Beyond Overfitting on Small Algorithmic
Data Sets, 2022.

Nanda et. al (2023): Transformer learns Fourier transform
on $(\mathbb{Z}/97\mathbb{Z}, +)$!

Transformers on small algorithmic tasks

Consider tokens $0, 1, 2, \dots, 96 \in \mathbb{Z}/97\mathbb{Z}$.

Can a transformer learn $(x, y) \rightarrow x+y \bmod 97$?

Nanda et. al (2023): Transformer (appears to)
learn Fourier transform
on $(\mathbb{Z}/97\mathbb{Z}, +)$!

Thank you !