Architectural Bias on Representations

Jeremy Bernstein

jbernstein@mit.edub

6.7960 :: Lecture 13

Aside: Steepest Descent (HW2)

$$\boxed{\operatorname{argmin}_{\Delta W} \; \mathrm{Tr}(G^T \Delta W) + \frac{\lambda}{2} \|\Delta W\|_*^2}$$

spectral norm

<u>Solution</u>

for $G = U \Sigma V^T$

reduced SVD

$$\Delta W = -\frac{\mathrm{Tr}(\Sigma)}{\lambda} U V^T$$

"steepest descent under the spectral norm"

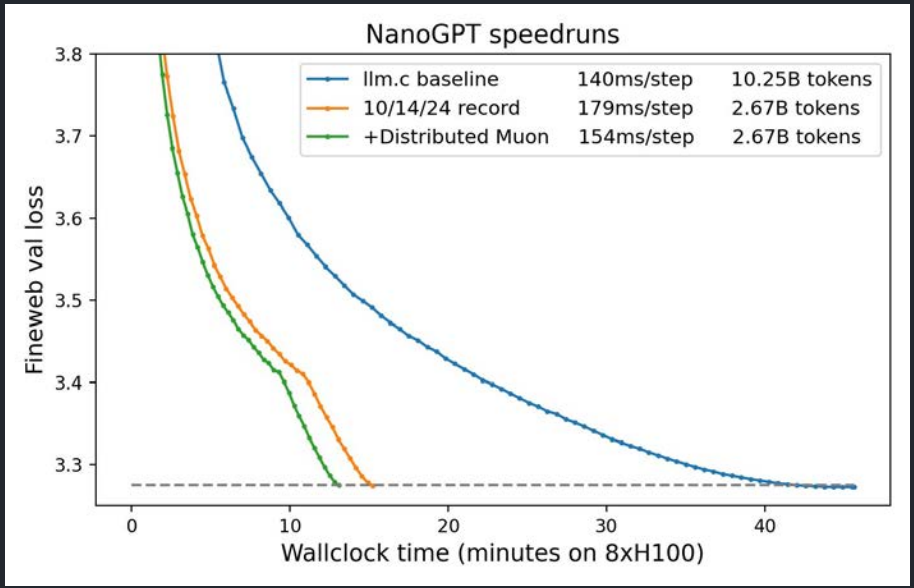
Neural Network Speedrunning

@kellerjordan0

$$\Delta W = -\frac{\mathrm{Tr}(\Sigma)}{\lambda} UV^T$$

"steepest descent under the spectral norm"

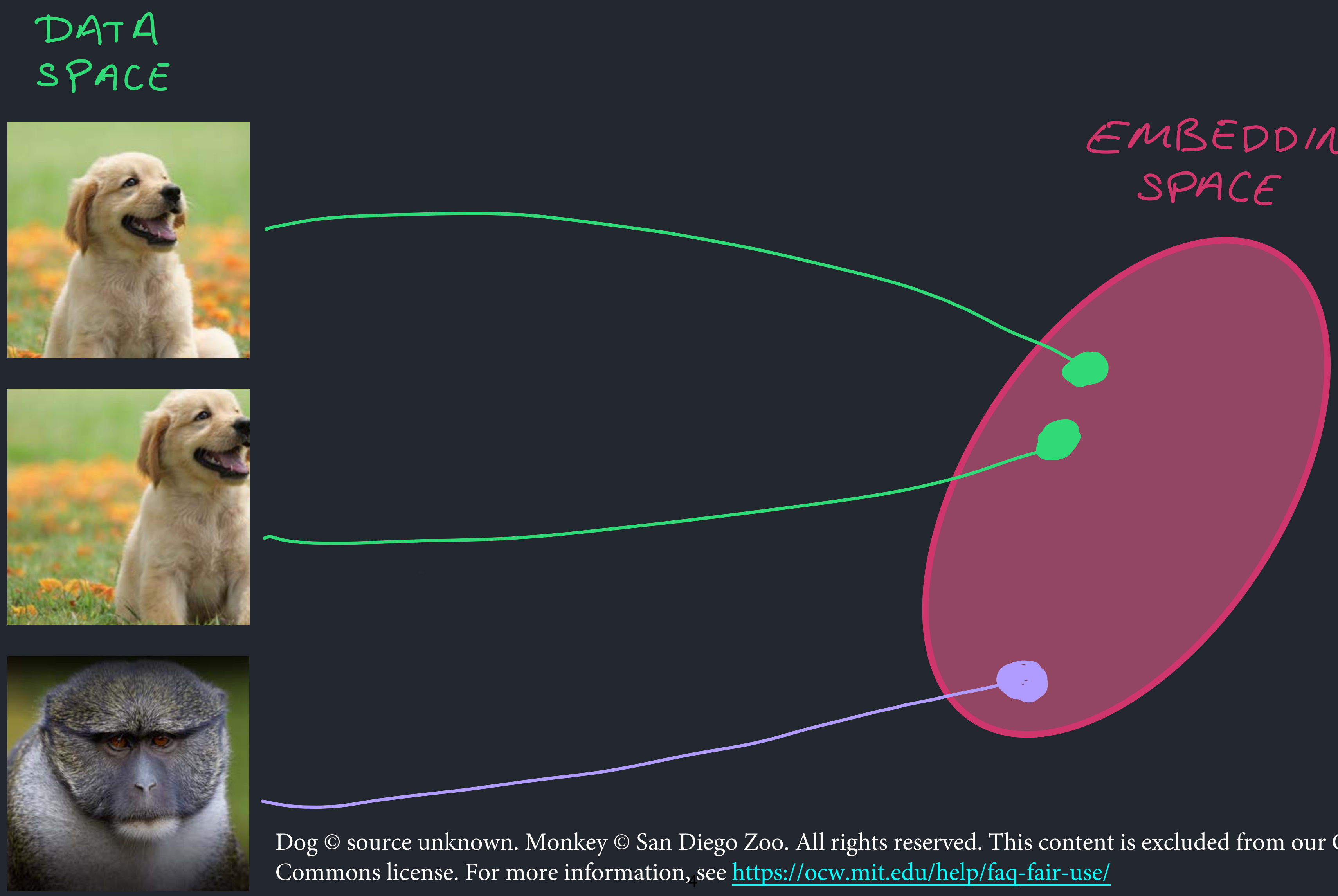
Add some tricks: momentum
low precision
fast computation of UV^T via iteration



"Muon optimizer" trains nanoGPT to 3.28 val loss on "Fineweb" dataset in < 15 minutes

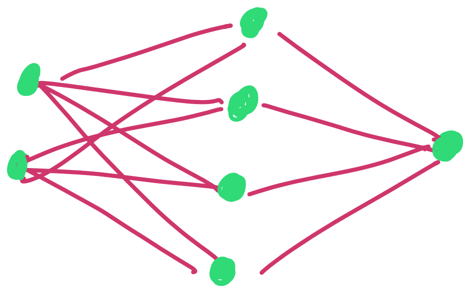
Similarity-Based Representation Learning (Lecture 12)

training objective that maps similar data to nearby embeddings

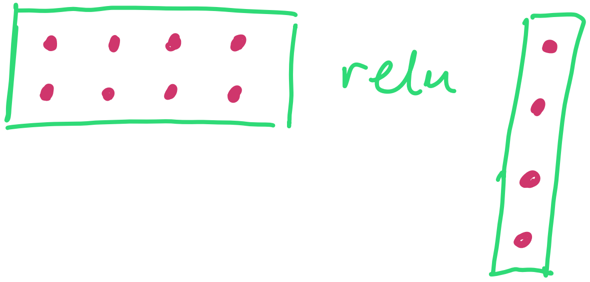


Perspectives on Neural Computation (Lecture 7)

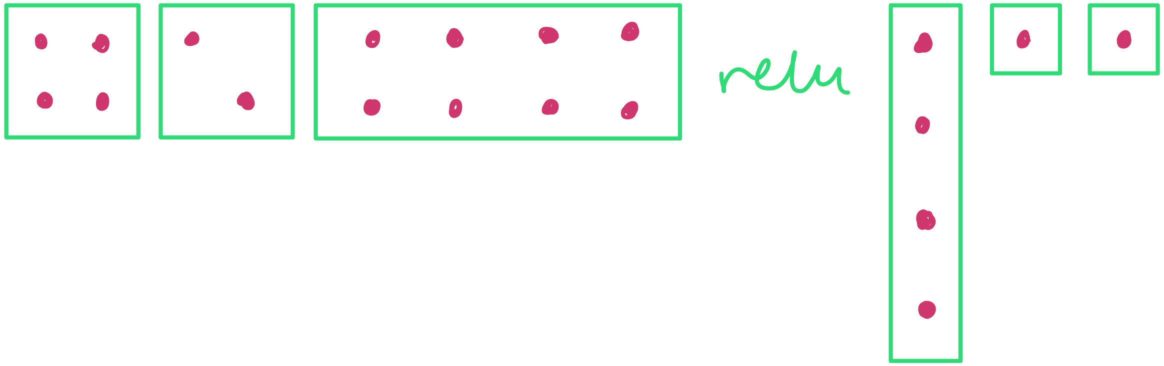
#1 NEURAL PERSPECTIVE



#2 TENSOR PERSPECTIVE

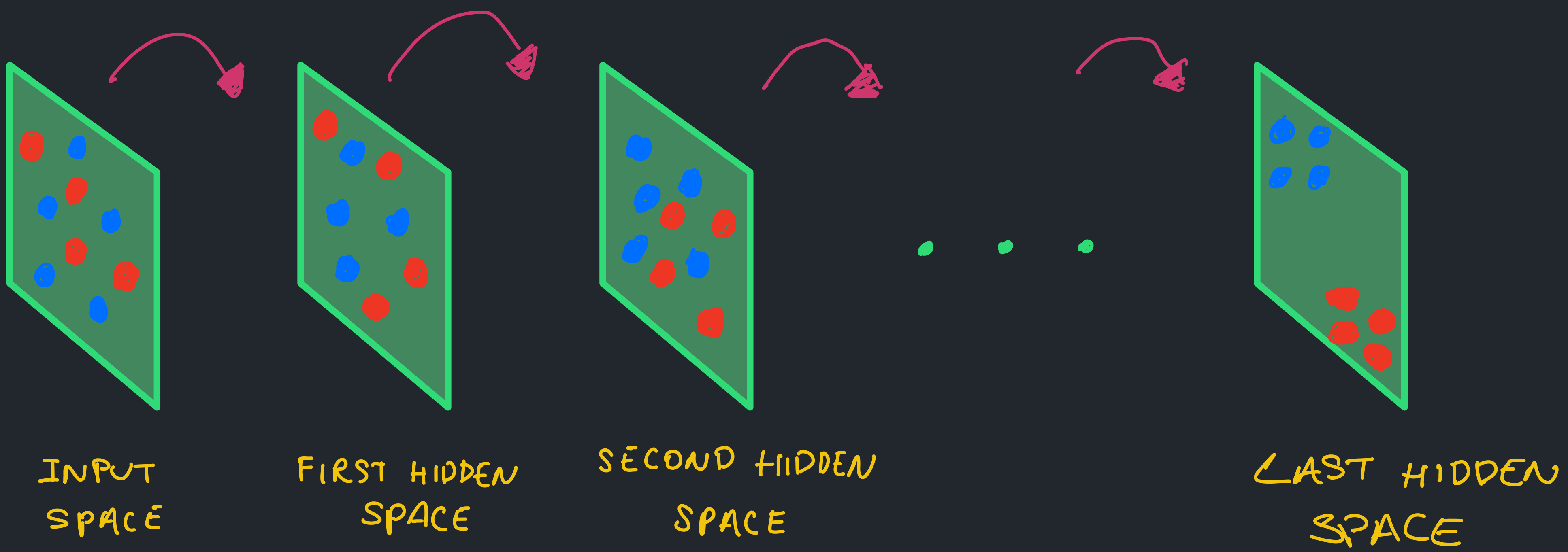


#3 SPECTRAL PERSPECTIVE



A More Abstract Perspective

A neural net is a map through a sequence of vector spaces.

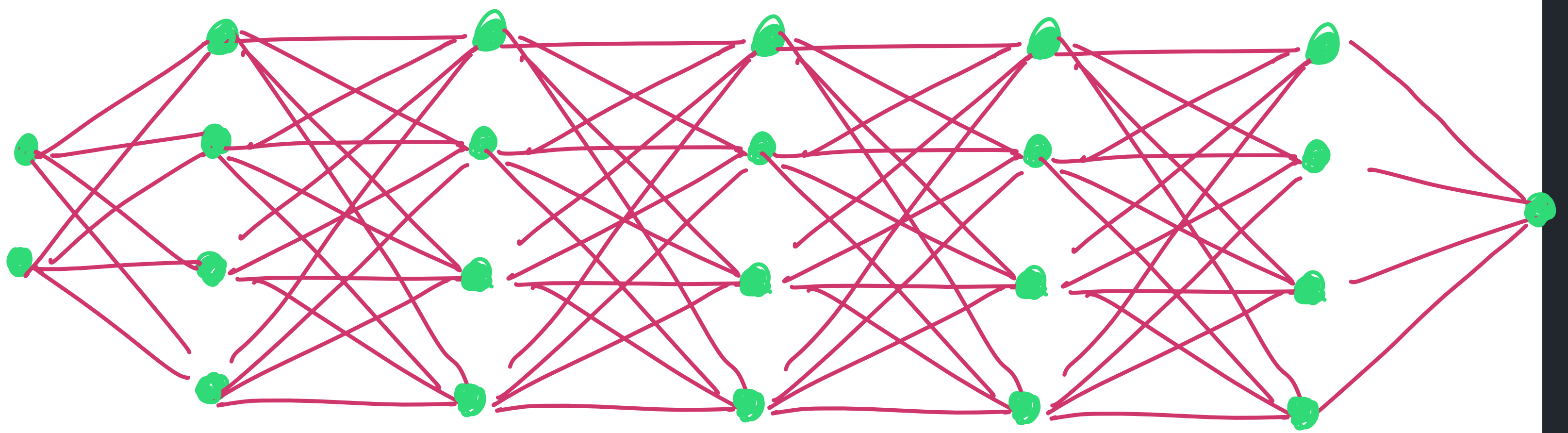


Want a good representation of the data at the final layer

e.g. linearly separable

This Lecture

We will develop an advanced tool that shows that a neural architecture (even without training) already expresses an opinion about data similarity.



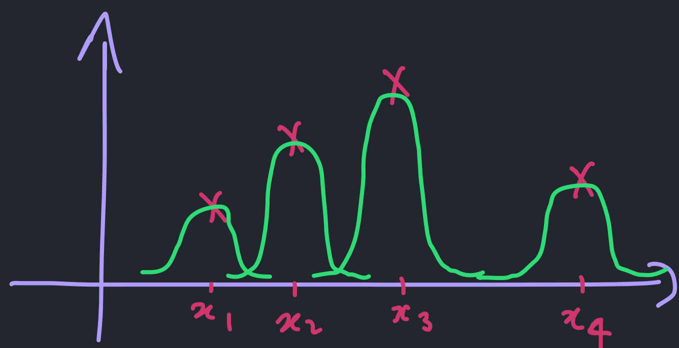
A Journey to the Past

Pretend you've never heard of deep learning, and you want to fit some data.

How would you do it?

Function Space Construction #1

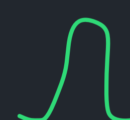
Place a "bump function" on each datapoint:



k is called the "kernel"

Formally, we consider functions of the form

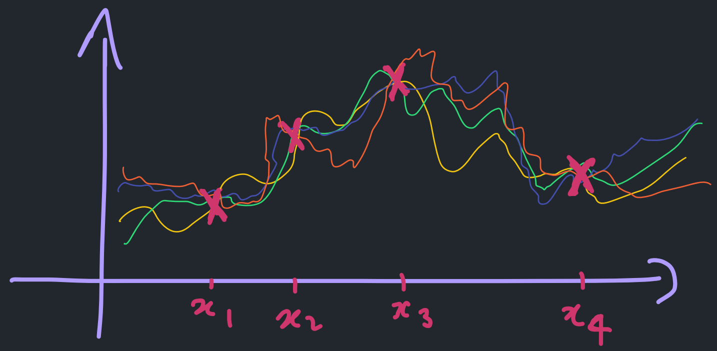
$$f(x) = \sum_{i=1}^{n} \alpha_i \, k(x, x_i)$$

where:
- $k(x, x_i)$ is a bump  centred on x_i
- the α_i are weights

The freedom to choose the number of bumps n, the bump centres x_i and the weights α_i leads to a rich function space called a "reproducing kernel Hilbert space".

Function Space Construction #2

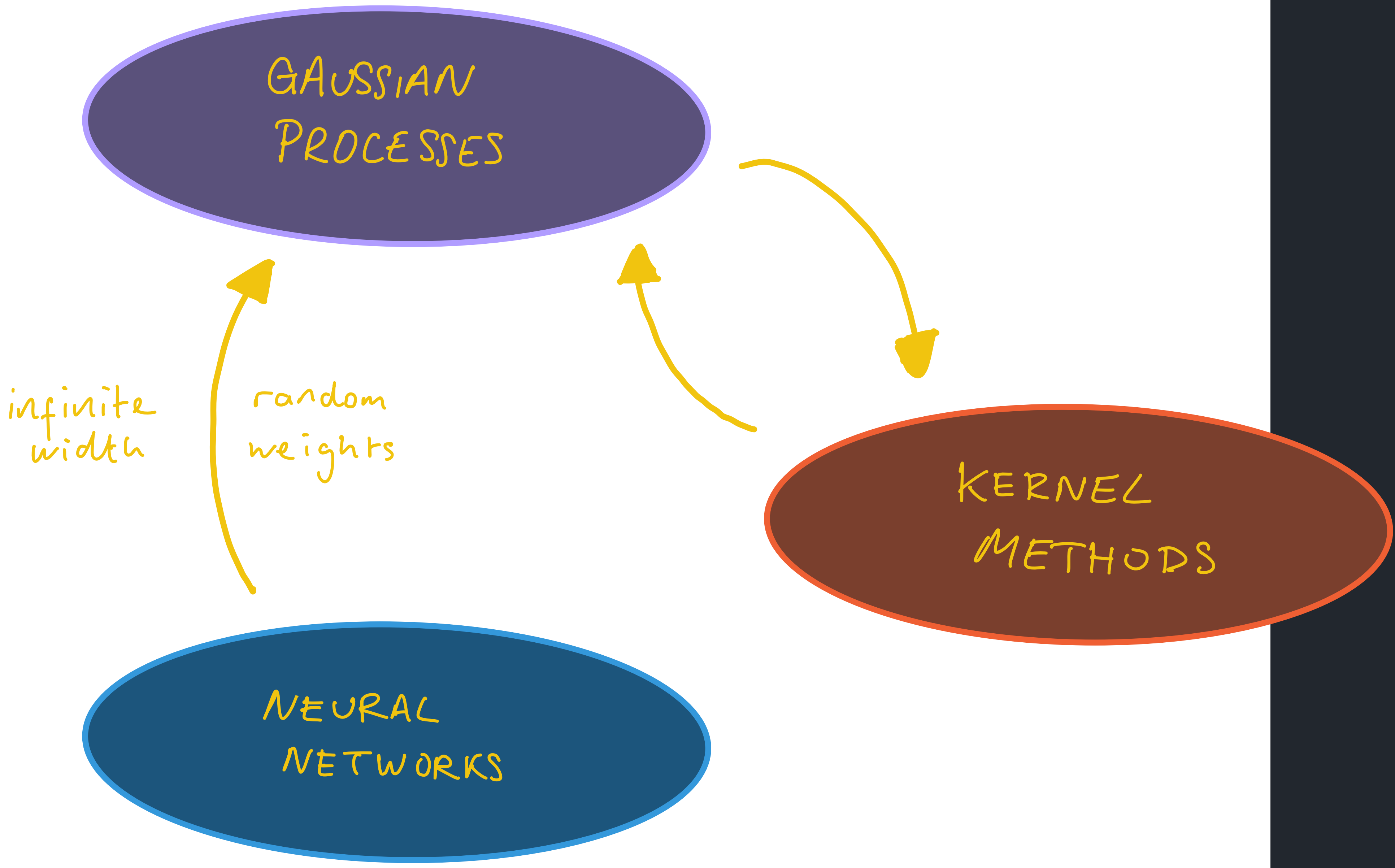
Draw random functions consistent with the data



We will look at a special way of drawing random functions that uses Gaussian random variables

→ it is called a Gaussian process (GP)

Correspondences between Function Spaces

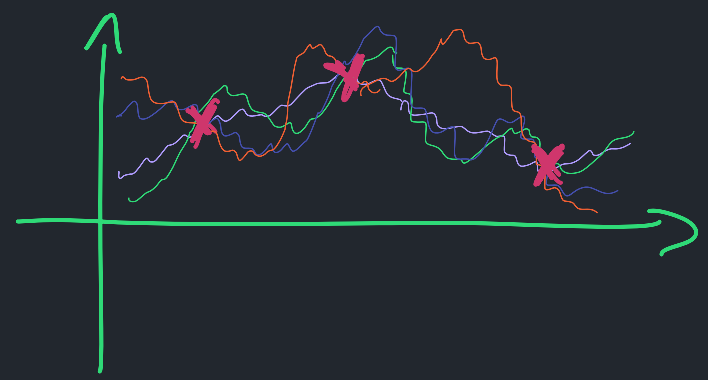


GAUSSIAN PROCESSES

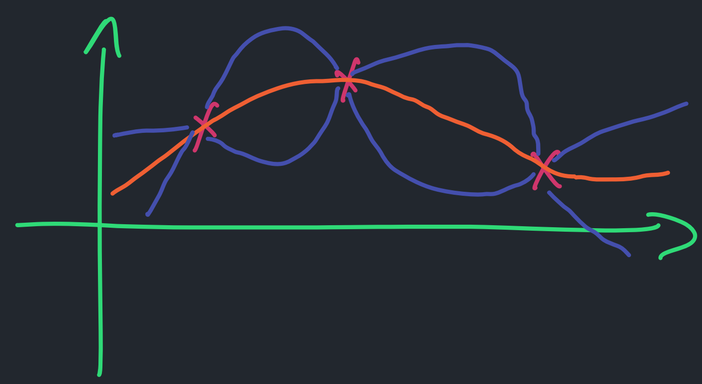
What is a Gaussian Process Pictorially?

Given data

A Gaussian process gives us a distribution of consistent functions

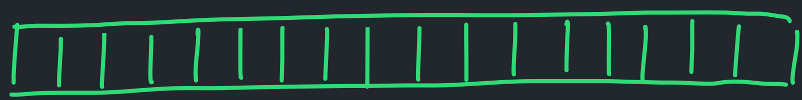


Along with a formula for the mean and standard deviation of this distribution



What is a Gaussian Process Informally?

Sample a Gaussian vector:

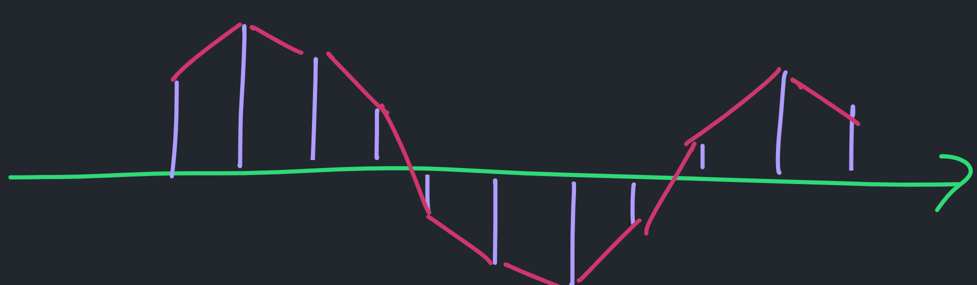


$$\sim N(\underset{\text{mean}}{0}, \underset{\text{covariance}}{\Sigma})$$

We can construct a function by plotting the components of this vector

Idea: set up Σ such that consecutive components are correlated

e.g. $\Sigma_{ij} \sim \exp -(i-j)^2$



This leads to more "continuous looking" functions.

What is a Gaussian Process Formally?

Consider an input space X

Let $f(x)$ be a random variable for every $x \in X$

Informally, think of f as an infinite-dimensional random vector indexed by $x \in X$.

If for every finite collection of inputs

$x_1, x_2, \ldots, x_n$

the associated finite-dimensional random vector

$\boxed{f(x_1) \mid f(x_2) \mid \cdots \mid f(x_n)}$ is Gaussian

then f is a "Gaussian process" on X.

Covariance Functions

A Gaussian process generalises

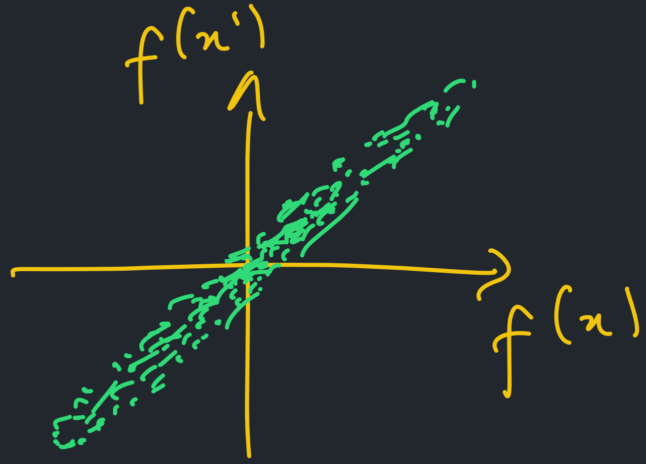
finite-dimensional Gaussian vectors $\longrightarrow$ infinite-dimensional functions

The covariance matrix Σ_{ij} for $i, j = 1, \dots, n$ $\longrightarrow$ covariance function $\Sigma(x, x')$ for $x, x' \in X$

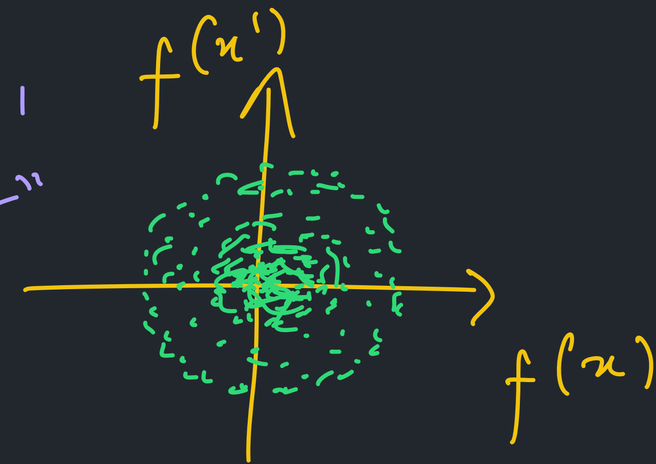
Covariance Functions

Typically, we want a covariance function that is large for "nearby" points and small otherwise

x and x' "nearby"



x and x' "distant"



$\longrightarrow$ choice of covariance function encodes what we mean by "nearby"

e.g. squared exponential $\qquad \Sigma(x, x') = e^{-(x-x')^2}$

inner product $\qquad \Sigma(x, x') = \langle x, x' \rangle$

Conditioning on Data

Suppose we are given $f(x_1), \dots, f(x_n)$

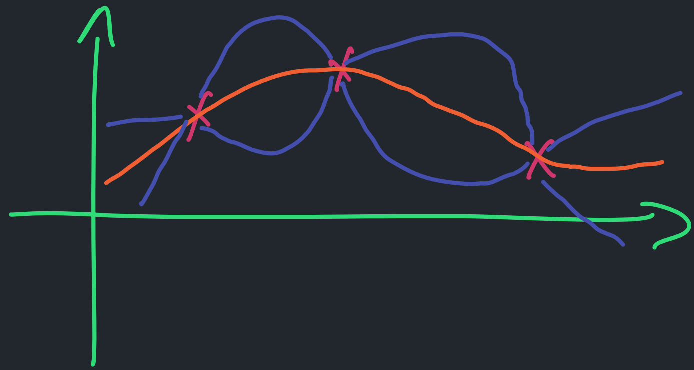
And we want to predict $f(x_*)$

Since f is a Gaussian process, then this vector is Gaussian

$f(x_1)$	$f(x_2)$	$\dots$	$f(x_n)$	$f(x_*)$

Can prove: $f(x_*)$ given $f(x_1), \dots, f(x_n)$ is $\mathcal{N}(\mu, \sigma^2)$

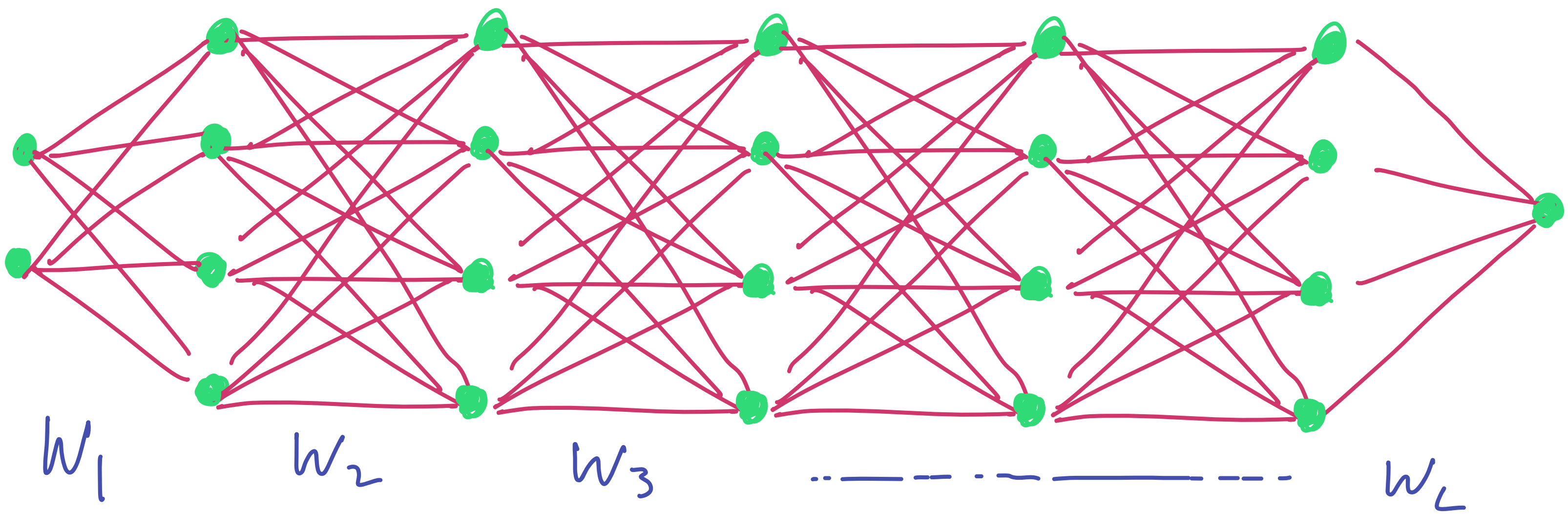
The mean $\mu(x_*)$ and standard deviation $\sigma(x_*)$ have simple closed form formulae.



NN-GP CORRESPONDENCE

Random Weights ⟹ Random Functions

Consider a neural network:



If we randomly sample the weights,
we get a random function.

Inspecting the Random Functions

To inspect the distribution of random functions

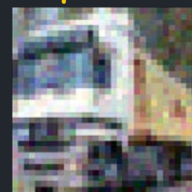
Pick two inputs x and x'

Sample 1000 random networks $f_1, \dots, f_{100}$

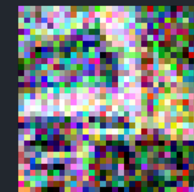
Plot scatterplot of $\begin{bmatrix} f_1(x_1), & f_1(x_2) \\ f_2(x_1), & f_2(x_2) \\ \vdots & \\ f_{1000}(x_1), & f_{1000}(x_2) \end{bmatrix}$

3 Layer MLP, Width 1000

input 2

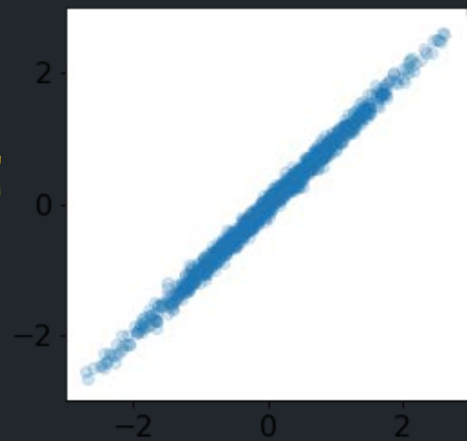


input 3



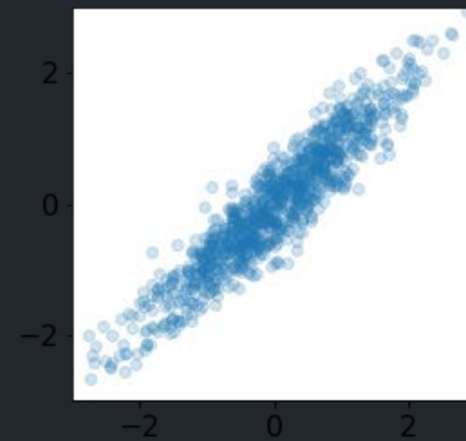
input 1

output 1



output 2

output 1



output 3

Observations:

- joint distribution of pairs of outputs seems Gaussian
- covariance depends on similarity of inputs

Neural Network – Gaussian Process Correspondence

If we sample iid the weights of an NN

then as the width $\longrightarrow \infty$

The joint distribution of any finite collection of network outputs $f(x_1) \quad f(x_2) \quad \dots \quad f(x_n)$ is Gaussian.

The covariance function depends on the architecture and non-linearity.

Proof Sketch

Main tool: multivariate central limit theorem

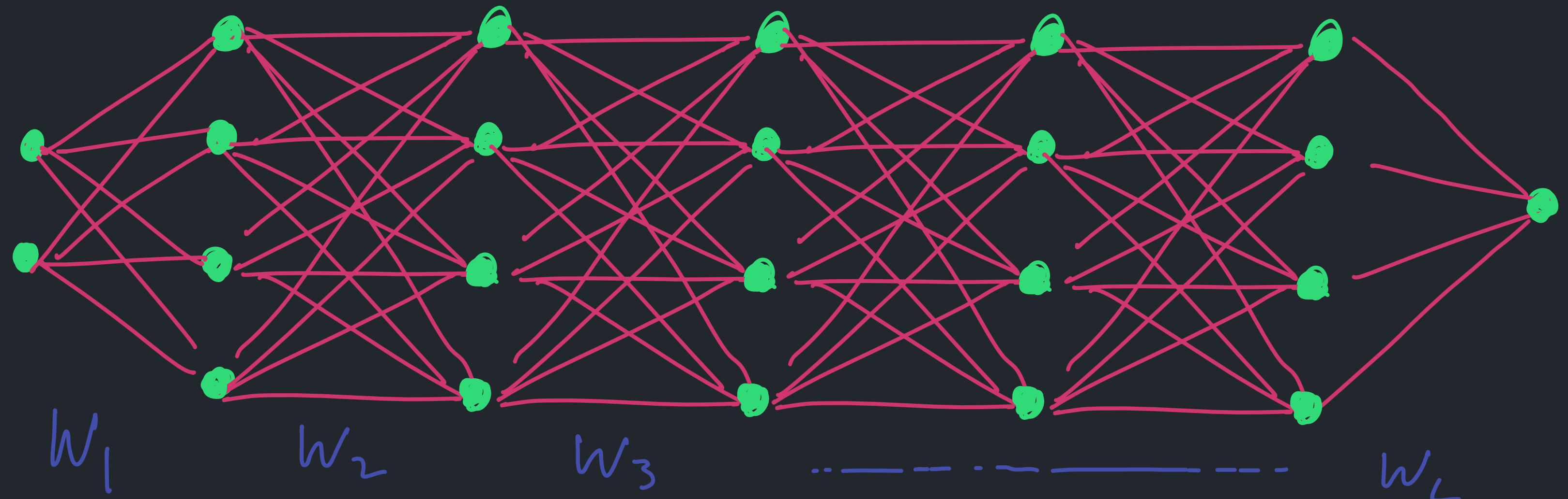
Step 1: for a fixed input and fixed layer, the activations are iid random variables

→ prove by induction on depth using MV-CLT

Step 2: for any collection of k inputs $x_1, \dots, x_k$ the network outputs $f(x_1), \dots, f(x_k)$ are Gaussian

→ prove by writing the vector $f(x_1), \dots, f(x_k)$ as a sum over iid vectors from the penultimate layer and apply the MV-CLT.

Example: MLPs with ReLU



set non-linearity to $\quad \phi(x) = \sqrt{2}\,\mathrm{relu}(x)$

sample weights iid $\quad \mathcal{N}\left(0, \frac{1}{\text{fan-in}}\right)$

for inputs $x, x' \in \mathbb{R}^d$:

$$\mathbb{E} f(x) = 0$$

$$\mathbb{E} f(x) f(x') = \underbrace{h \circ \dots \circ h}_{L-1 \text{ times}} \left(\frac{x^T x'}{d}\right)$$

where $h(t) = \frac{1}{\pi}\left[\sqrt{1-t^2} + t\,(\pi - \arccos t)\right]$ "compositional arccosine kernel"

Natural Questions

How does $\Sigma(x, x')$ depend on:

- choice of architecture?
- choice of weight distribution?

Can this inform:

- architecture design?
- weight regularisation strategies?

References

- Bayesian Learning for Neural Networks
  Neal
- Kernel Methods for Deep Learning
  Cho & Saul
- Deep Neural Networks as Gaussian Processes
  Lee, Bahri, Novak et al.