Mathematical
Institute

Data classes for which DNNs can overcome the curse of dimensionality and Attention modules.

Theories of Deep Learning: C6.5,
Lecture / Video 4

Prof. Jared Tanner
Mathematical Institute
University of Oxford

Oxford
Mathematics

Curse of dimensionality (Yarotsky 16')

Exponential in d/m growth in number of weights.

Mathematical Institute

Yarotsky 16' results show exponential approximation in depth, but the overall number of weights is $\mathcal{O}(\epsilon^{-d/m})$. Recall

$$\|f\|_{W_m^\infty([0,1]^d)} = \max_{|s|\leq m} \text{esssupp}_{x\in[0,1]^d} |D^s f(x)|.$$

Theorem (Yarotsky 16')

For any d, m and $\epsilon \in (0,1)$, there is a ReLU network with depth at most $c(1+\ln(1/\epsilon))$ and at most $c\epsilon^{-d/m}(1+\log(1/\epsilon))$ weights (width $\mathcal{O}(\epsilon^{-d/m})$), for c a function of d, m, that can approximate any function from $F_{d,m}$ within absolute error ϵ.

`https://arxiv.org/pdf/1610.01145.pdf`

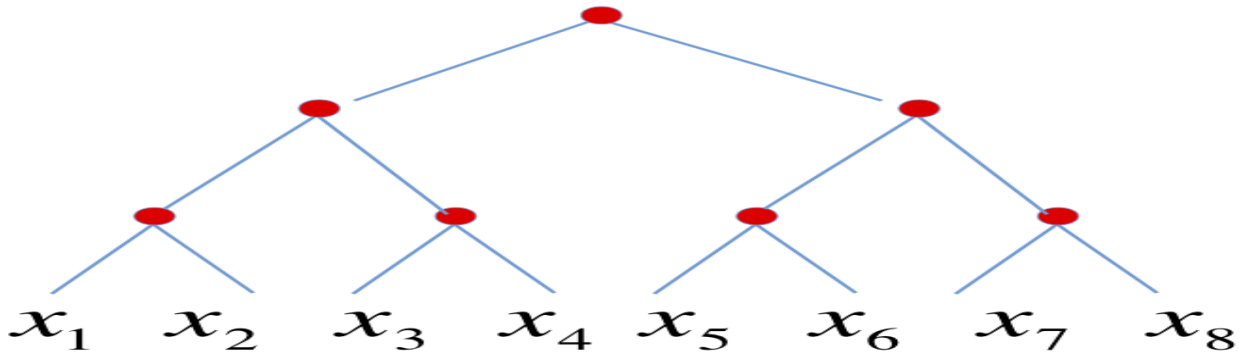
To avoid curse of dimensionality need $m \sim d$ or more structure in the function F to be approximated; e.g. compositional structure.

Compositional structured functions (Poggio et al. 17')

Extending the compositional nature of Yarotsky dimensionally

UNIVERSITY OF OXFORD
Mathematical Institute

Consider functions with a binary tree hierarchical structure:



where $x \in \mathbb{R}^8$ and
$f(x) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$
Let $W_m^{n,2}$ be the class of all compositional functions $f(\cdot)$ of n variables with binary tree structure and constituent functions $h(\cdot)$ of 2 variables with m bounded derivatives.

`https://arxiv.org/pdf/1611.00740.pdf`

Compositional structured functions (Poggio et al. 17')

Each constituent function is a map from $\mathbb{R}^2 \to \mathbb{R}$

UNIVERSITY OF OXFORD
Mathematical Institute

The set $W_m^{n,2}$ of of all compositional functions $f(\cdot)$ of n variables with binary tree structure and constituent functions $h(\cdot)$ of 2 variables with m bounded derivatives can be effectively approximated using a DNN with a rate dictated by the ability to approximate functions $\mathbb{R}^2 \to \mathbb{R}$; e.g. effectively locally $d = 2$.

Theorem (Poggio 17')

Let $f(\cdot) \in W_m^{n,2}$ and consider a DNN with the same binary compositional tree structure and an activation $\sigma(\cdot)$ which is infinitely differentiable, and not a polynomial. The function $f(\cdot)$, can be approximated by ϵ with a number of weights that is $\mathcal{O}\left((n-1)\epsilon^{-2/m}\right)$.

`https://arxiv.org/pdf/1611.00740.pdf`

Compositional structured functions (Poggio et al. 17')

Compositional functions $W_m^{n,2}$ compared to shallow NNs and Yarotsky(16')

UNIVERSITY OF OXFORD
Mathematical Institute

The set $W_m^{n,2}$ of of all compositional functions $f(\cdot)$ of n variables with binary tree structure are effectively $d = 2$ in the DNN approximation requirements, but are much richer than $d = 2$.

Functions can be approximated within ϵ with a DNN from $\mathcal{O}(In(1/\epsilon))$ layers with a number of weights:

- $\mathcal{O}(\epsilon^{-d/m})$ for general locally smooth functions (Yarotsky 16'),
- $\mathcal{O}\left((n-1)\epsilon^{-2/m}\right)$ for $f(\cdot) \in W_m^{n,2}$, binary tree structure and constituent functions in $C_m[0,1]^2$.
- $\mathcal{O}(\epsilon^{-d/m})$ for shallow NNs is best possible for $f(\cdot) \in W_m^n$ which have non-binary hierarchical tree structures.

`https://arxiv.org/pdf/1611.00740.pdf`

Definition of local effective dimensionality (Poggio et al. 17')

Local dimensionality determined by approximation rate ϵ^{-d}.

UNIVERSITY OF OXFORD
Mathematical Institute

Definition (Poggio 17')

The effective dimensionality of a function class W is said to be d if for every $\epsilon > 0$, any function within W can be approximated within an accuracy ϵ by a DNN at rate ϵ^{-d}.

In the prior slide we had examples of complex compositional functions with effective dimensionality 2. These could be extended naturally to local *effective dimensionality* d_{eff} and *local smoothness* m_{eff} for rate $\epsilon^{-d_{eff}/m_{eff}}$.

Restriction to a data class decreases d_{eff} and localisation can increase the smoothness m_{eff} substantially.
`https://arxiv.org/pdf/1611.00740.pdf`

Intrinsic dimensionality of sub-manifolds (Hein et al. 05')

MNIST exemplar dataset classes are approximately under 15 dimensional

University of Oxford
Mathematical Institute

Estimates of dimensionality within MNIST digit classes using three approaches: the reference below, and two others building on local linear embedding.

Table 7. Number of samples and estimated intrinsic dimensionality of the digits in MNIST.

1	2	3	4	5
7877	6990	7141	6824	6903
8/7/7	13/12/13	14/13/13	13/12/12	12/12/12
6	7	8	9	0
6876	7293	6825	6958	6903
11/11/11	10/10/10	14/13/13	12/11/11	12/11/11

https://icml.cc/Conferences/2005/proceedings/papers/037_Intrinsic_HeinAudibert.pdf

A manifold model can explicitly represent the data through:

$$X = f(CF/\sqrt{d}) \in \mathbb{R}^{p,n}$$

where:

- $F \in \mathbb{R}^{d,n}$ are the d features used to represent the data
- $C \in \mathbb{R}^{p,d}$ combines the $d < n < p$ features
- $f(\cdot)$ is an entrywise locally smooth nonlinear function.

This data model is the same as a generative adversarial network (GAN) and is similar to dictionary learning and subspace clustering models where C is typically sparse.

`https://hal-cea.archives-ouvertes.fr/cea-02529246/document`

Additional approximation theory resources

Review articles and courses elsewhere

Further references for the approximation theory perspective of deep learning include:

- Telgarsky's "Deep Learning Theory" course, lectures 1-11:
  `http://mjt.cs.illinois.edu/courses/dlt-f20/`
- Matthew Hirn's "Mathematics of Deep Learning" course: lectures 20-24.
  `https://matthewhirn.com/teaching/spring-2020-cmse-890-002/`
- DNN Approximation Theory by Elbrachter et al. (19')
  `https://www.mins.ee.ethz.ch/pubs/files/deep-it-2019.pdf`

Transformers and Attention, the building blocks of natural language generative models.

Natural language processing (NLP) involves decomposing text into a finite set of tokens.

"Tokens can be thought of as pieces of words. Before the API processes the request, the input is broken down into tokens. These tokens are not cut up exactly where the words start or end - tokens can include trailing spaces and even sub-words. Here are some helpful rules of thumb for understanding tokens in terms of lengths:"

1 token $=$ 4 chars in English

1 token $=$ 3/4 words

100 tokens $=$ 75 words"

`https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them`

UNIVERSITY OF OXFORD
Mathematical Institute

Each token is then embedded in a word embedding dimension and sequentially each token in the context window is embedded as rows in a matrix $X \in \mathbb{R}^{d_{ctx} \times d_{emb}}$.

NLP networks can have tasks such as "next token prediction" where text is grown, or "translation" from one language or sentiment to another, or combined in multi-modal systems with other networks (e.g. vision) for such tasks as automatic caption generation.

By far the dominant method for NPL are attention / transformer modules.

`https://arxiv.org/abs/2005.14165`

Attention mechanism (Vaswani 17'), equations

Key and Query quadratic form to highlight relations

The input matrix X is then mapped to a query, key, and value matrices: $Q = XW_Q$, $K = XW_K$, and $V = XW_V$. Layer-norm is applied to the key and query matrices where each row of Q and K are normalized to have unit ℓ^2 length. These normalized query and key matrices are then compared, further normalized by scalar $d^{-\alpha}$ and typically a softmax applied row-wise

$$A = \text{softmax}\left(QK^T d^{-\alpha}\right)$$

so that each row of A has non-negative entries that sum to 1. This "self"-attention matrix is then applied to the value matrix, e.g. AV. The scaling α is typically $1/2$, but also sometimes 1.

`https://arxiv.org/abs/1706.03762`

`https://arxiv.org/abs/1607.06450`

Attention mechanism (Vaswani 17'), diagram

Key and Query quadratic form to highlight relations

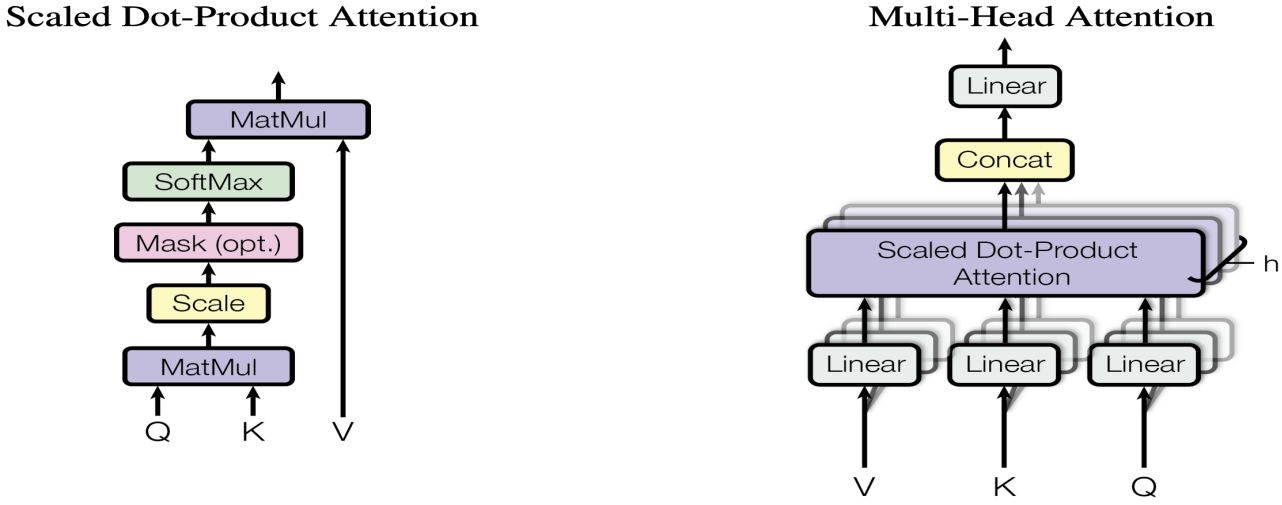


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Typically the attention mechanism applied on the left is computed within the same layer for multiple values of W_Q, W_K, and W_V with each output called a "head" and these "heads" then concatonated. `https://arxiv.org/abs/1706.03762`

Transformer mechanism (Vaswani 17'), diagram

Encoder-decoder structure using attention

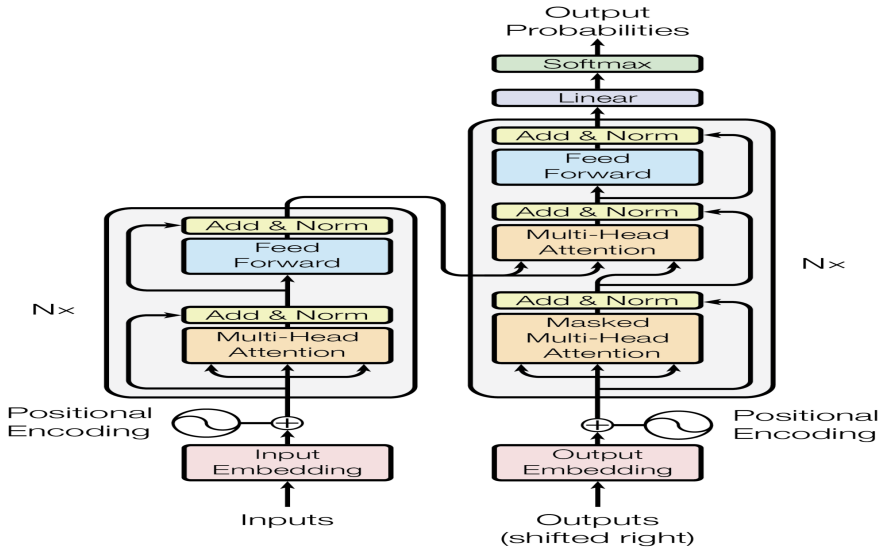


Figure 1: The Transformer - model architecture.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

The concatonated heads are then acted on by a fully connected layer W_O applied. Additional fully connected layers with ReLU are often included, as are skip connections.

`https://arxiv.org/abs/1706.03762`

UNIVERSITY OF OXFORD
Mathematical Institute

Omitting the notation to denote a layer:
Input $X \in \mathbb{R}^{d_{ctx} \times d_{emb}}$

$$Q_i = LN(XW_{Q_i}), \quad K_i = LN(XW_{K_i}), \quad \text{and} \quad V_i = XW_{V_i}$$

for $i = 1, \ldots, n_h$ where $LN(Z)$ normalizes each row of the matrix Z to unit ℓ^2 length. Form $h_i = \text{softmax}(Q_i K_i^T / d_{emb}^{1/2}) V_i$ for $i = 1, \ldots, n_h$ and concatonate on to p of one another $H = [h_1 \ h_2 \ \ldots h_{n_h}]$ and form $W_O H$ then output

$$X + ReLU(W_1 ReLU(W_2 H))$$

which is then the input for the next layer.
`https://arxiv.org/abs/1706.03762`

GPT3 architecture (Brown 20'), diagram

GPT: Generative Pre-Trained Transformers

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Each layer of GPT3 (and earlier variants) is a transformer block.
Energy usage to train and apply such large models is very costly.
`https://arxiv.org/abs/2005.14165`

Vision Transformer - ViT (Dosovitskiy 21'), diagram

Patches of images embedded and acting as token; with position embedding

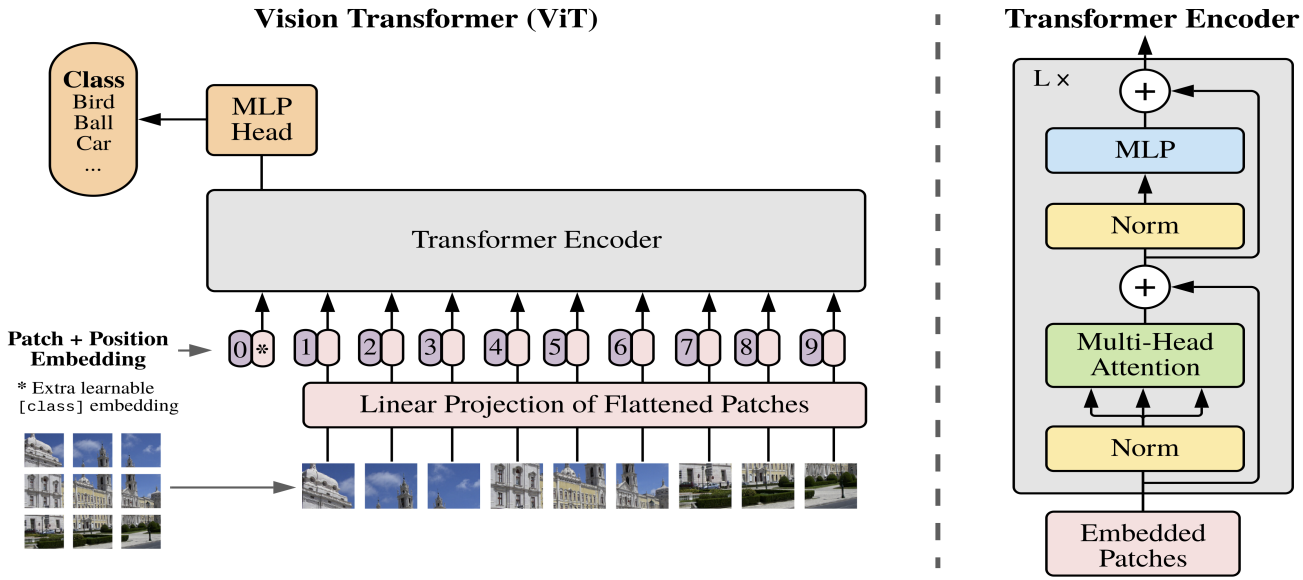


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

https://arxiv.org/pdf/2010.11929

Vision Transformer - ViT (Dosovitskiy 21'), formulae

Positional embedding included

The MLP contains two layers with a GELU non-linearity.

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

$$\mathbf{z'}_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

Inductive bias. We note that Vision Transformer has much less image-specific inductive bias than CNNs. In CNNs, locality, two-dimensional neighborhood structure, and translation equivariance are baked into each layer throughout the whole model. In ViT, only MLP layers are local and translationally equivariant, while the self-attention layers are global. The two-dimensional neighborhood structure is used very sparingly: in the beginning of the model by cutting the image into patches and at fine-tuning time for adjusting the position embeddings for images of different resolution (as described below). Other than that, the position embeddings at initialization time carry no information about the 2D positions of the patches and all spatial relations between the patches have to be learned from scratch.

`https://arxiv.org/pdf/2010.11929`

Expressivity of deep nets prior to 2016, for reference.

Mathematical Institute

Prior to the approximation rate results from Telgarsky 15' and Yarotsky 16', there were qualitative geometric results showing showing potential for exponential expressivity:

- On the number of response regions of deep feedforward networks with piecewise linear activations (Pascanu et al. 14')
  `https://arxiv.org/pdf/1312.6098.pdf`
- On the expressive power of deep neural networks (Raghu et al. 16')
  `https://arxiv.org/abs/1606.05336`
- Trajectory growth lower bounds for random sparse deep ReLU networks (Price et al. 19')
  `https://arxiv.org/abs/1911.10651`

ReLU hyperplane arrangement

Partition of the input domain $\mathbb{R}^{n_0}$: one layer

University of Oxford
Mathematical Institute

The action of ReLU to an affine transform is a linearly increasing function orthogonal to hyperplanes; let $W \in \mathbb{R}^{n_1 \times n_0}$ then:

$$H_i := \{x \in \mathbb{R}^{n_0} : \ W_i x + b_i = 0\} \quad \forall i \in [n_1]$$

where W_i is the i^{th} row of W.
The normals to these hyperplanes partition the input dimension n_0, and if W is in general position (all subsets of rows are maximal rank), then the number of partitions is:

$$\sum_{j=0}^{n_0} \binom{n_1}{j}$$

`https://arxiv.org/pdf/1312.6098.pdf`

The number of partitions in one layer is lower bounded by

$$\sum_{j=0}^{n_0} \binom{n_1}{j} \geq n_1^{\min\{n_0, n_1/2\}}$$

and each hidden layers can further subdivide these regions:

Theorem (Pascanu et al. 14')

An L layer DNN with ReLU activation, input $\mathbb{R}^{n_0}$, and hidden layers of width $n_1, n_2, \ldots, n_L$ partitions the input space into at least

$$\Pi_{\ell=0}^{L} n_\ell^{\min\{n_0, n_\ell/2\}}$$

This shows an exponential dependence on depth L.

`https://arxiv.org/pdf/1312.6098.pdf`

ReLU hyperplane arrangement

Partition of the input domain $\mathbb{R}^{n_0}$: plot Pascanu et al. 14'

Mathematical Institute

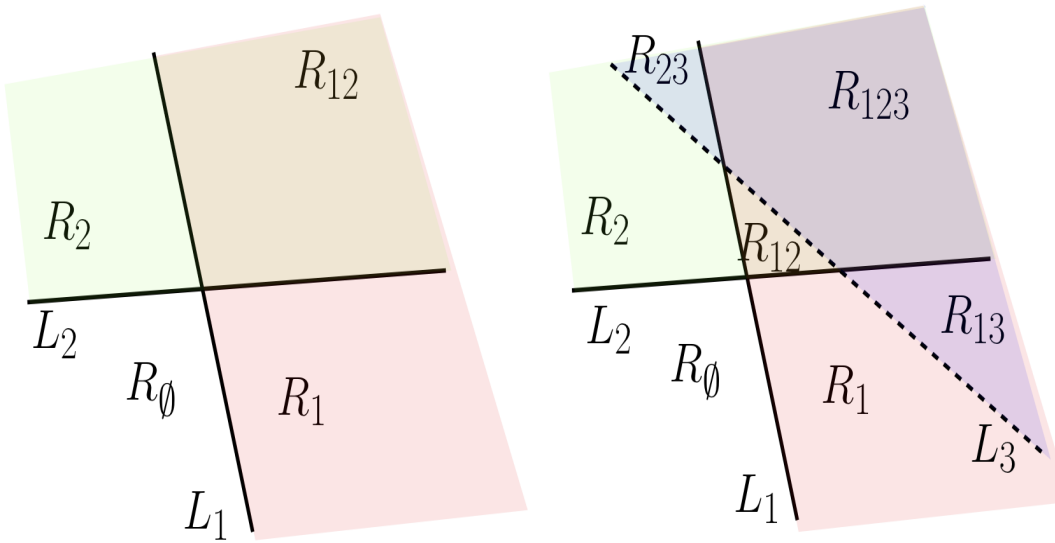


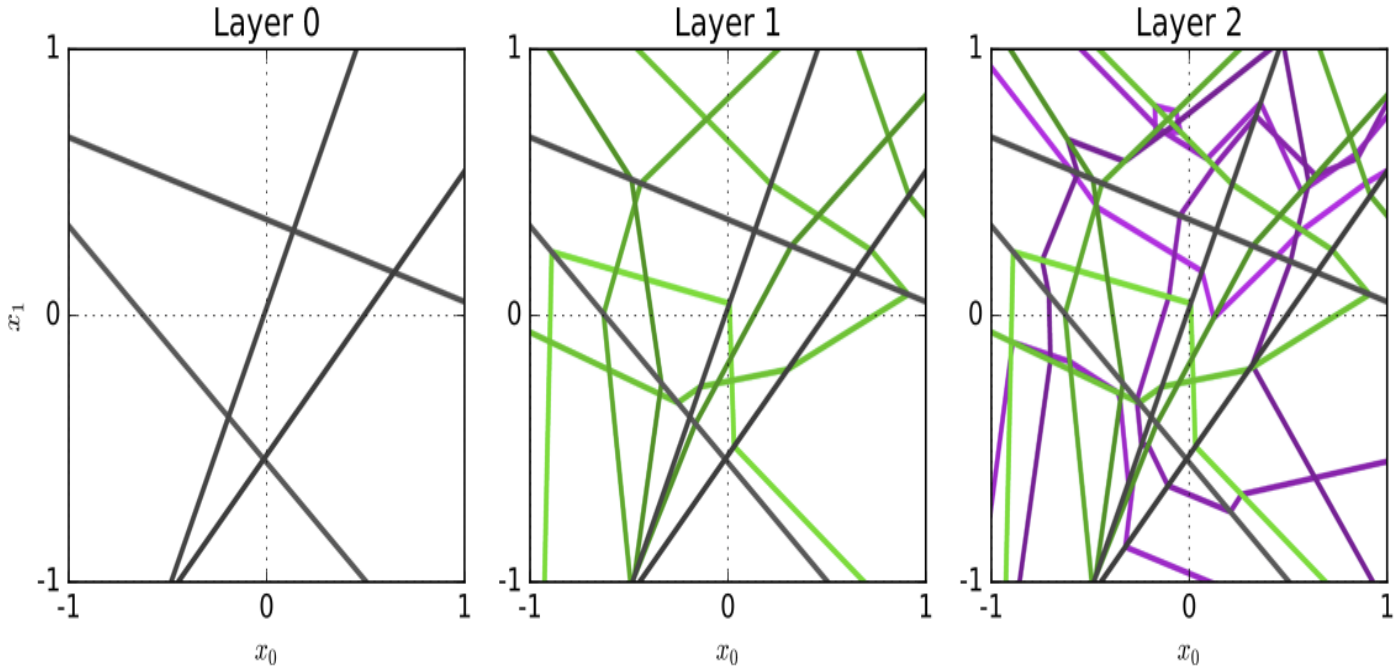
Figure 2: Induction step of the hyperplane sweep method for counting the regions of line arrangements in the plane.

`https://arxiv.org/pdf/1312.6098.pdf`

ReLU hyperplane arrangement

Partition of the input domain $\mathbb{R}^{n_0}$: plot Raghu et al. 16'

Mathematical Institute



https://arxiv.org/abs/1606.05336

This "activation region" perspective is a useful intuition for ReLU, but lacks the quantitative convergence rates we observed in more recent approximation theory results of Yarotsky 16'.

Random initialisations and DNNs

DNNs are typically first trained from random values

A random network $f_{NN}(x; \mathcal{P}, \mathcal{Q})$ denotes a deep neural network:

$$h^{(d)} = W^{(d)} z^{(d)} + b^{(d)}, \qquad z^{(d+1)} = \phi(h^{d}), \qquad d = 0, \ldots, L-1,$$

which takes as input the vector x, and is parameterised by random weight matrices $W^{(d)}$ with entries sampled iid from the distribution $\mathcal{P}$, and bias vectors $b^{(d)}$ with entries drawn iid from distribution $\mathcal{Q}$.

While our goal is always to train a network, DNNs typically start as random networks which influence their ability to be trained.

Popular choices are Gaussian, $\mathcal{P} = \mathcal{N}(0, \sigma_w^2)$, or uniform, $\mathcal{P} = \mathcal{U}(-C_w, C_w)$ initialisations.
(*Note, for random networks we use $\phi(\cdot)$ as the nonlinear activation and σ to denote variance.)

Raghu et al. 16' introduced the notion of trajectory length

$$l(x(t)) = \int_t \left|\left| \frac{dx(t)}{dt} \right|\right| dt.$$

as a measure of expressivity of a DNN. In particular, they considered passing a simple geometric object $x(t)$, such as a line $x(t) = tx_0 + (1-t)x_1$ for $x_0, x_1 \in \mathbb{R}^k$ and measure the expected length of the output of the random DNN at layer d:

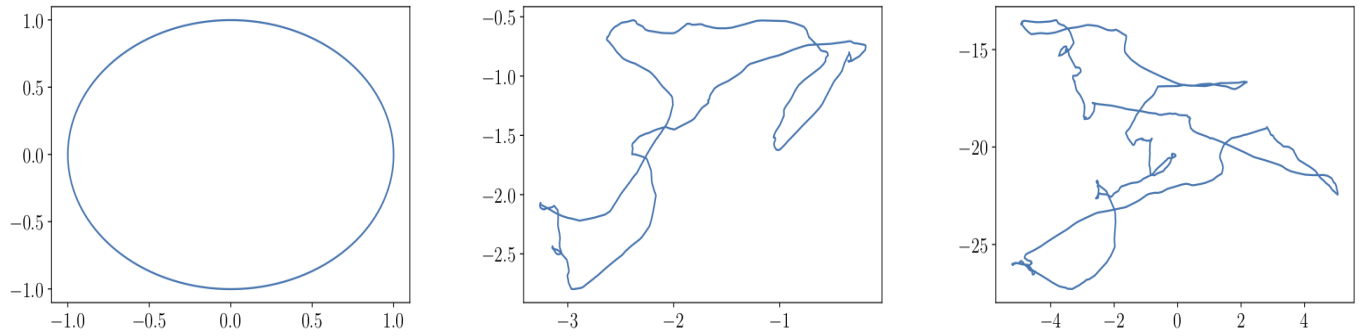
$$\frac{\mathcal{E}\left[\ell(z^{(d)})\right]}{\ell(x(t))}$$

`https://arxiv.org/abs/1606.05336`

Example of circle passed through a random DNN

Complexity of output increasing with depth

A circle passed through a random DNN and the pre-activation output $h^{(d)}$ at layers $d = 6$ and 12.



(a) Input (b) Layer 6 (c) Layer 12

Figure 1: A circular trajectory, passed through a ReLU network with $\sigma_w = 2$. The plots show the pre-activation trajectory at different layers projected down onto 2 dimensions.

DNNs can be used to *generative* data, GANs, and there one might consider the complexity of the manifold the GAN can generate as a measure of expressivity.

Random DNN: expected path length lower bound

Path length bound dependence on σ_w.

Mathematical Institute

Consider random DNNs of width n and depth L with weights and bias are drawn i.i.d. $W^{(\ell)}(i,j) \sim \mathcal{N}(0, \sigma_w^2/n)$, $b^{(\ell)}(j) \sim \mathcal{N}(0, \sigma_b^2)$

Theorem (Raghu et al. 16')

Consider as input a one dimensional trajectory $x(t)$ with arc-length $\ell(x(t)) = \int_t \left\| \frac{dx(t)}{dt} \right\| dt$ and let $z^{(L)}(t)$ be the output of the Gaussian random feedforward network with ReLu activations, then

$$\frac{\mathcal{E}\left[\ell(z^{(L)})\right]}{\ell(x(t))} \geq \mathcal{O}\left(\left(\frac{\sigma_w}{(\sigma_w^2+\sigma_b^2)^{1/4}} \cdot \frac{n^{1/2}}{(n+(\sigma_w^2+\sigma_b^2)^{1/2})^{1/2}}\right)^L\right).$$

https://arxiv.org/abs/1606.05336

Exponential growth of path length with depth.

Empirical experiments for htanh activation

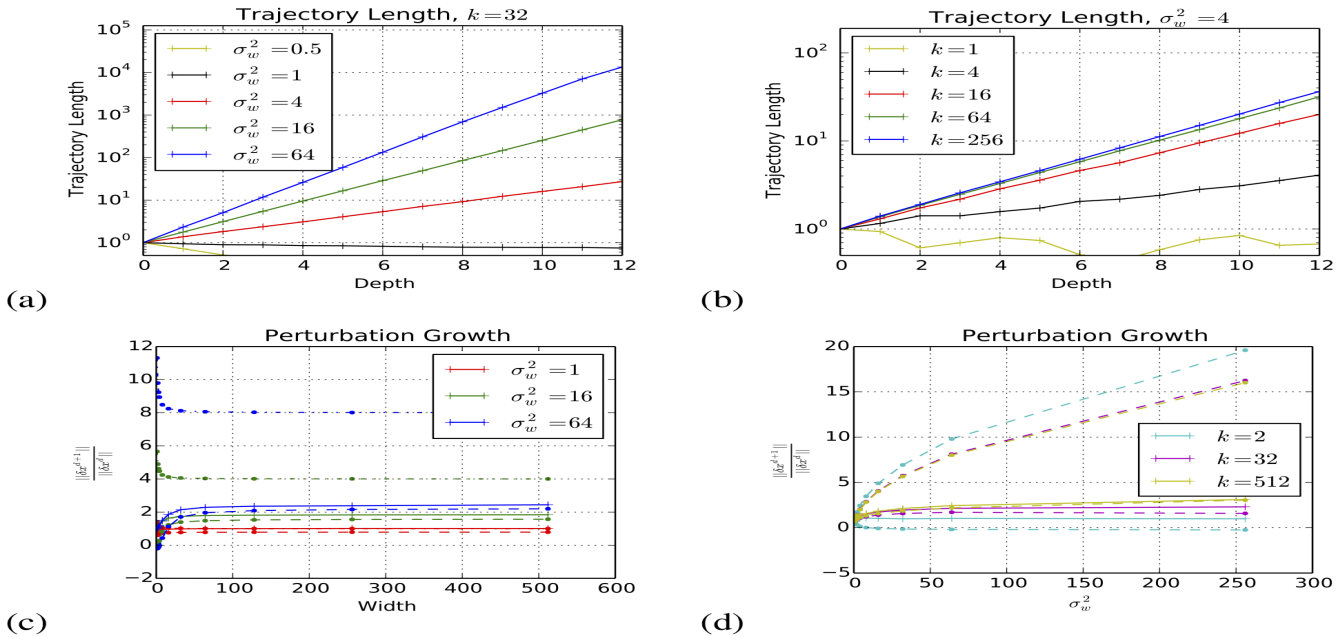


Figure 2: The exponential growth of trajectory length with depth, in a random deep network with hard-tanh nonlinearities. A circular trajectory is chosen between two random vectors. The image of that trajectory is taken at each layer of the network, and its length measured. *(a,b)* The trajectory length vs. layer, in terms of the network width k and weight variance σ_w^2, both of which determine its growth rate. *(c,d)* The average ratio of a trajectory's length in layer $d+1$ relative to its length in layer d. The solid line shows simulated data, while the dashed lines show upper and lower bounds (Theorem 1). Growth rate is a function of layer width k, and weight variance σ_w^2.

https://arxiv.org/pdf/1611.08083.pdf

UNIVERSITY OF OXFORD
Mathematical Institute

Theorem (Price et al. 19')

Let $f_{NN}(x; \alpha, \mathcal{P}, \mathcal{Q})$ be a random sparse net with layers of width n. Then, if $\mathrm{E}[|u^T w_i|] \geq M\|u\|$, where w_i is the i^{th} row of $W \in \mathcal{P}$, and u and M are constants, then

$$\mathrm{E}[l(z^{(L)}(t))] \geq \left(\frac{M}{2}\right)^L \cdot l(x(t))$$

for $x(t)$ a 1-dimensional trajectory in input space.

Exponential growth with depth for random initialisations such as Gaussian, uniform, and discrete; e.g. for Gaussian $M = \sigma_w\sqrt{2/\pi}$.
`https://arxiv.org/abs/1911.10651`

Observed growth rate (solid) and bounds (dashed)

Empirical experiments showing dependence on σ_w and sparsity α

University of Oxford
Mathematical Institute

Price et al. 19' also extended the results to have all but α fraction of the entries in W equal to 0.

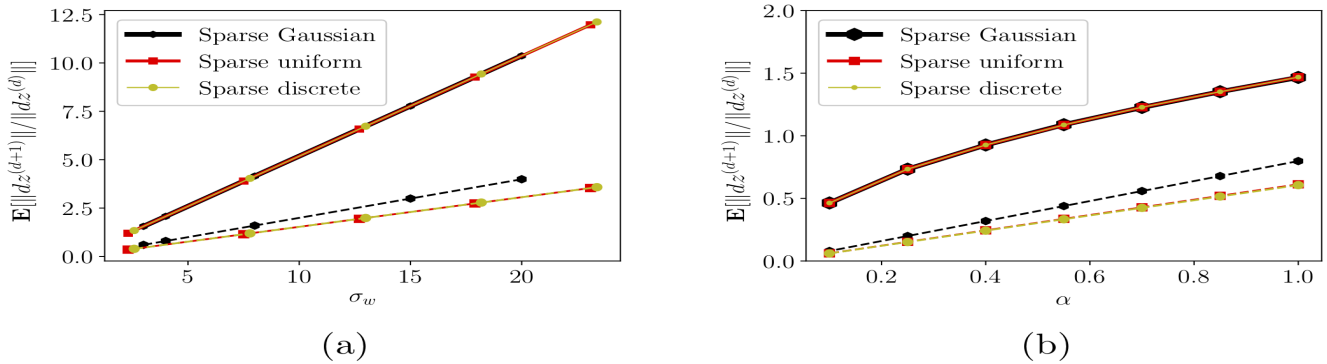


Figure 3: Expected growth factor, that is, the expected ratio of the length of any very small line segment in layer $d+1$ to its length in layer d. Figure 3a shows the dependence on the variance of the weights' distribution, and Figure 3b shows the dependence on sparsity.

Unless σ_w or α small enough at initialisation the pre-activiation output is exponentially complex.

`https://arxiv.org/abs/1911.10651`