The Gaussian world is not enough

How and what do NN learn from their data?

Sebastian **Goldt** (SISSA, Trieste)

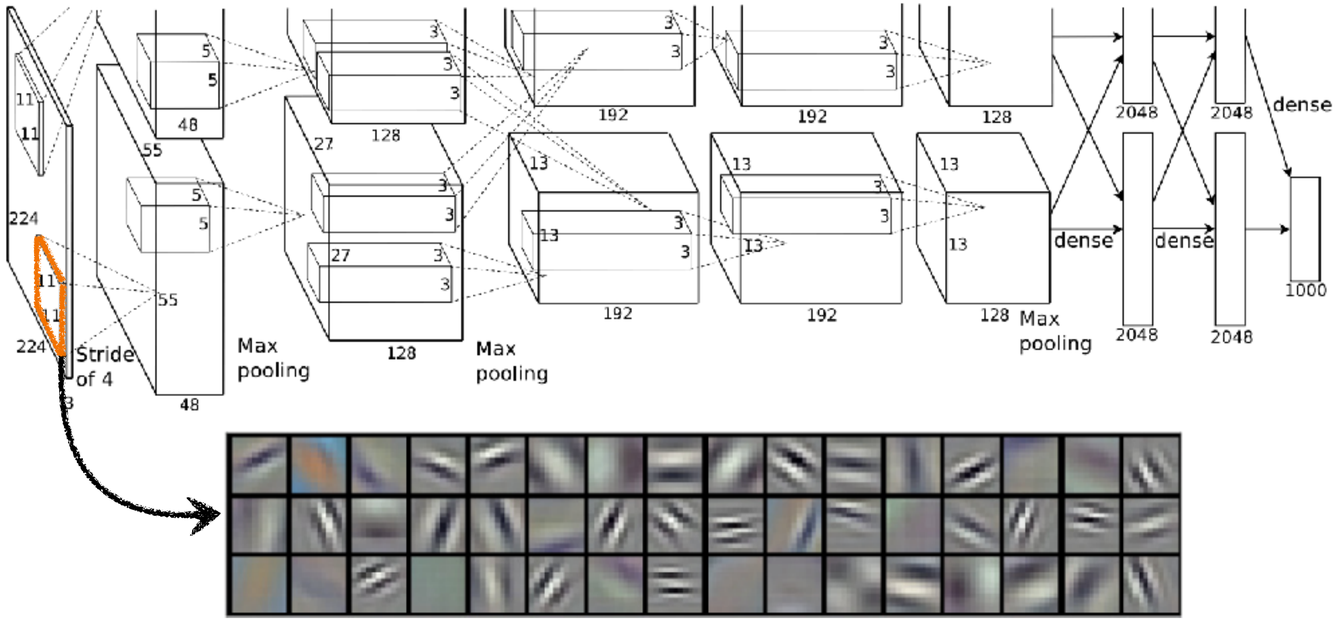
joint work w/ L. Bardone, F. Gerace, A. Laio, A. Ingrosso,
W. Redman, R. Rende, M. Refinetti & E. Székely

Statistical physics & machine learning back together, again @ Cargèse — Aug 9th, 2023

Neural networks are great at feature learning

But what do they actually learn from their data?

AlexNet on ImageNet Krizhevsky, Sutskever, Hinton. NeurIPS 2012

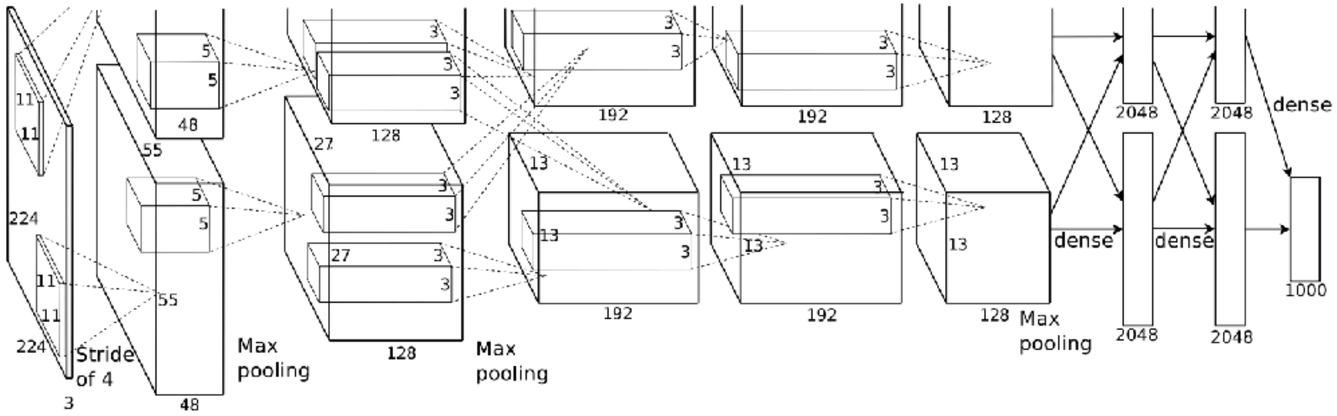


ImageNet + AlexNet + SGD ≈ Gabor filters

What do neural networks learn from data?

Data statistics shape the features learnt by neural networks

AlexNet on ImageNet Krizhevsky, Sutskever, Hinton. NeurIPS 2012



Data-dependent features » hand-crafted features

Efficient coding: receptive fields are adapted to environmental statistics

(Attneave ’54, Barlow ’61, etc.)

Finding a dictionary for images

Olshausen & Field (1996)

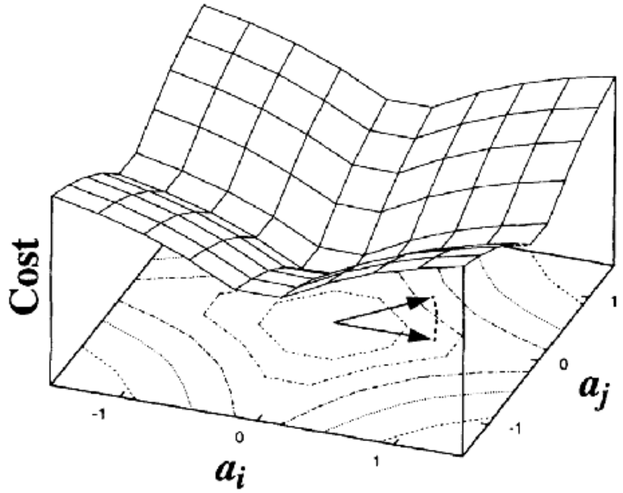
$$I(x,y) = \sum_i a_i \phi_i(x,y)$$

Coefficients *Basis functions*

Cost function for a **linear** model:

E = -[preserve information] + λ [sparseness of a_i]

$$-\sum_{x,y}\left(I(x,y) - \sum_i a_i \phi_i(x,y)\right)^2$$



Finding a dictionary for images

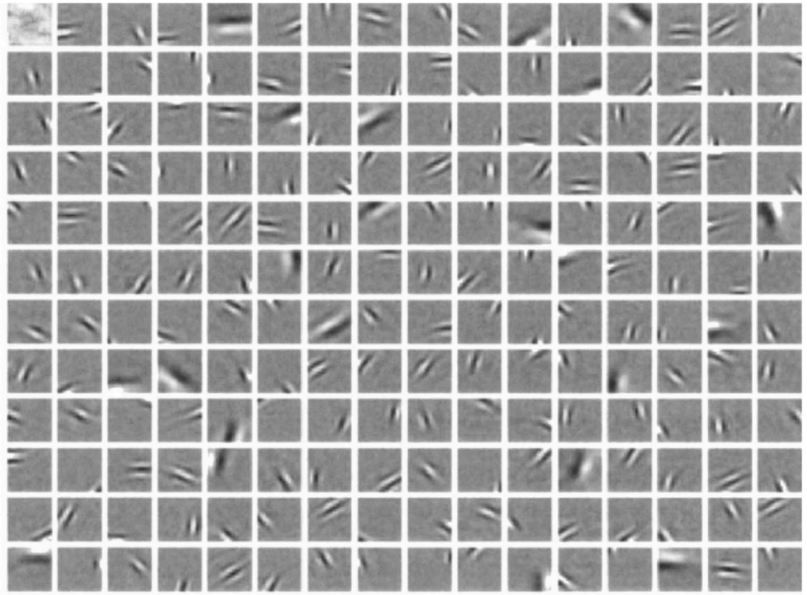
Olshausen & Field (1996)

$$I(x,y) = \sum_i a_i \phi_i(x,y)$$

Coefficients *Basis functions*

Cost function for a **linear** model:

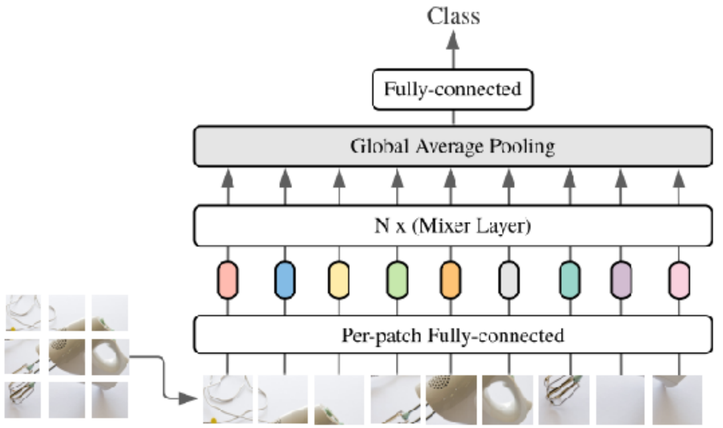
$$E = -\sum_{x,y}\left(I(x,y) - \sum_i a_i \phi_i(x,y)\right)^2 - \lambda \sum_i S\left(\frac{a_i}{\sigma}\right)$$

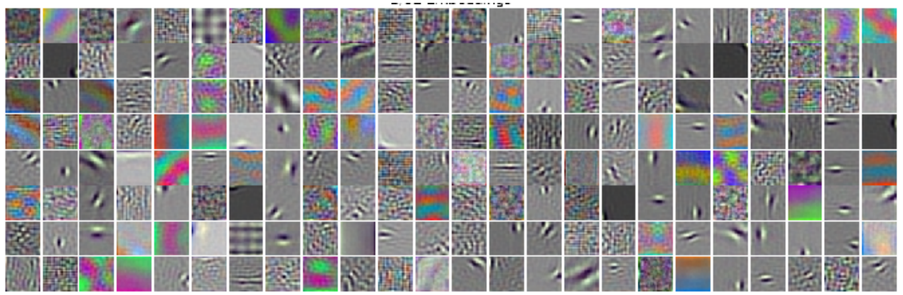


Embeddings in MLP mixer

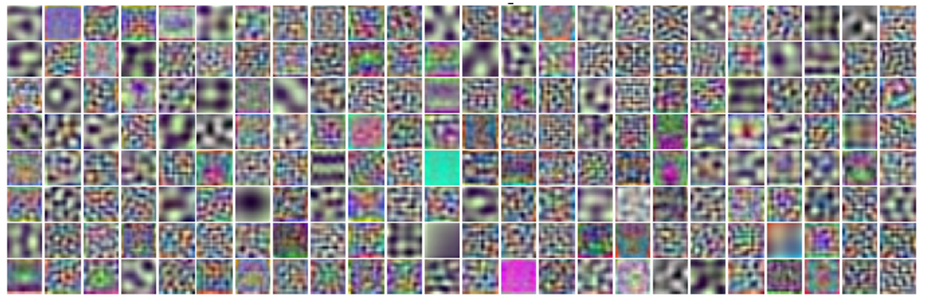
One final (recent!) example

- MLP mixer (Tolstikhin et al. NeurIPS ’21), an image classifier with mostly fully-connected layers.
  - (but the filters need to be large enough)





32x32 linear filters of the first layer (embeddings)

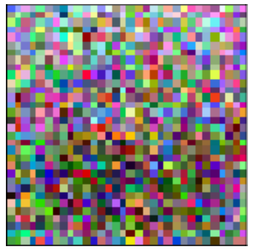


16x16 linear filters of the first layer (embeddings)

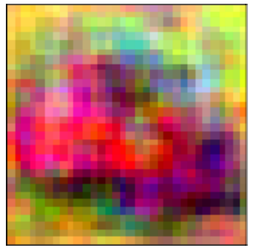
The Gaussian world…

Statistical physics provides Gaussian models of data

- **Classic approach**: random fn (“teacher”) of i.i.d. inputs
  - Gardner & Derrida (1989), Seung, Sompolinsky, and Tishby (1992)
  - “Gaussian design” in high-dim statistics (Donoho & Tanner, 2009)



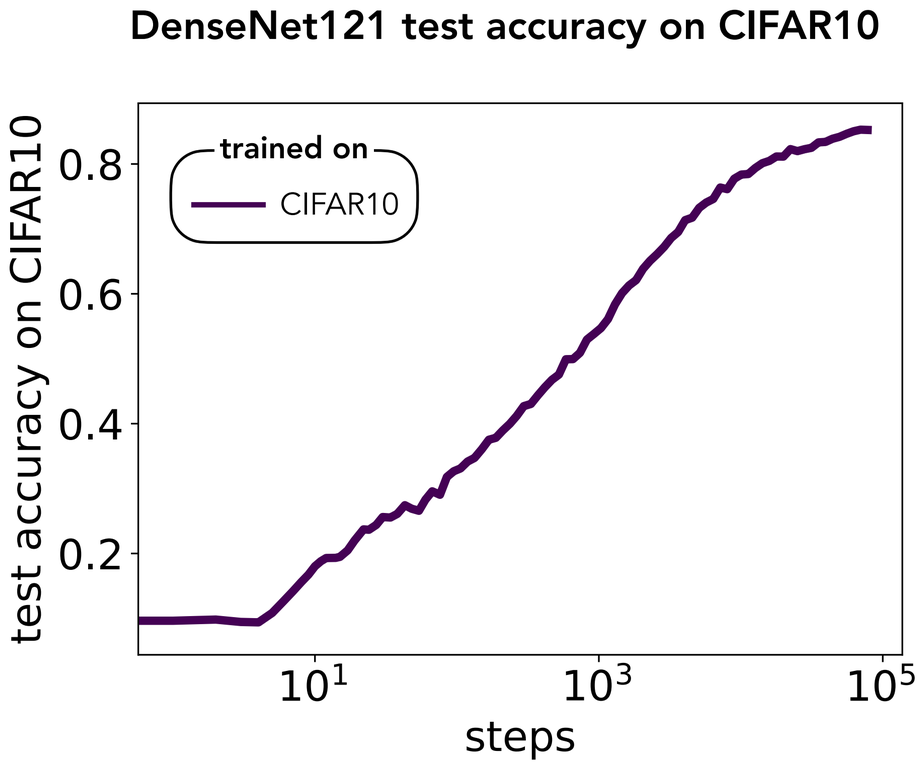
- **Gaussian equivalence** ⇒ extend theory to correlated inputs
  - Mei & Montanari (2022), Goldt et. al (2020, 2022), Gerace et al. (2020), Hu & Lu (2020), Bordelon et al. (2020), Loureiro et al. (2022), etc.
    (cf. Yue’s, Hugo’s, and Yatin’s talk!)

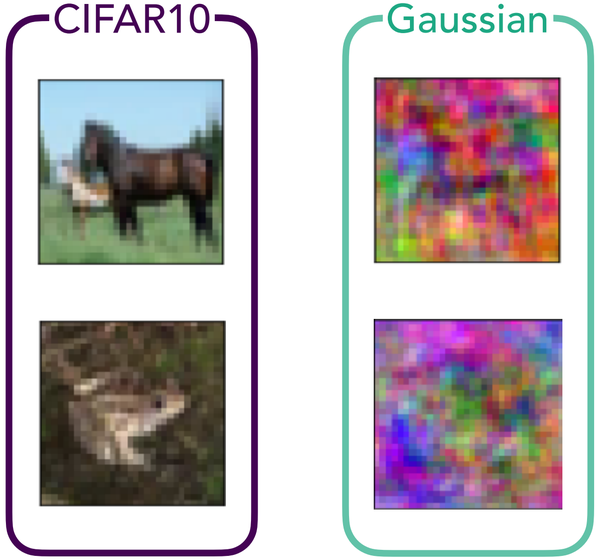


- Generated **many insights**!

The Gaussian world… is not enough

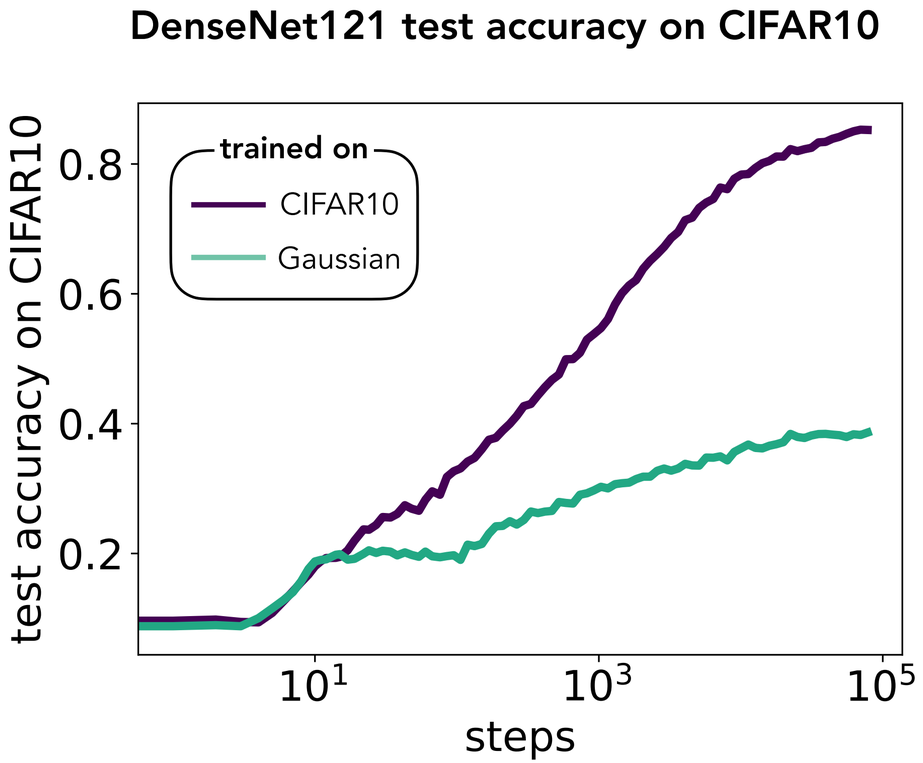
Higher-order cumulants are key to learning in deep convolutional networks

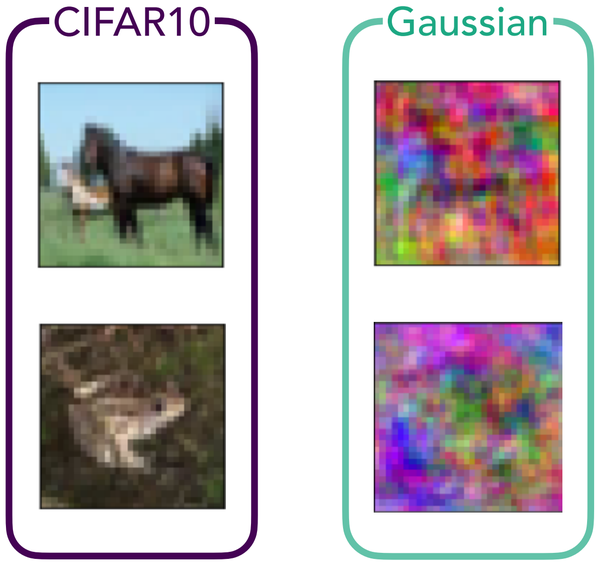




The Gaussian world… is not enough

Higher-order cumulants are key to learning in deep convolutional networks

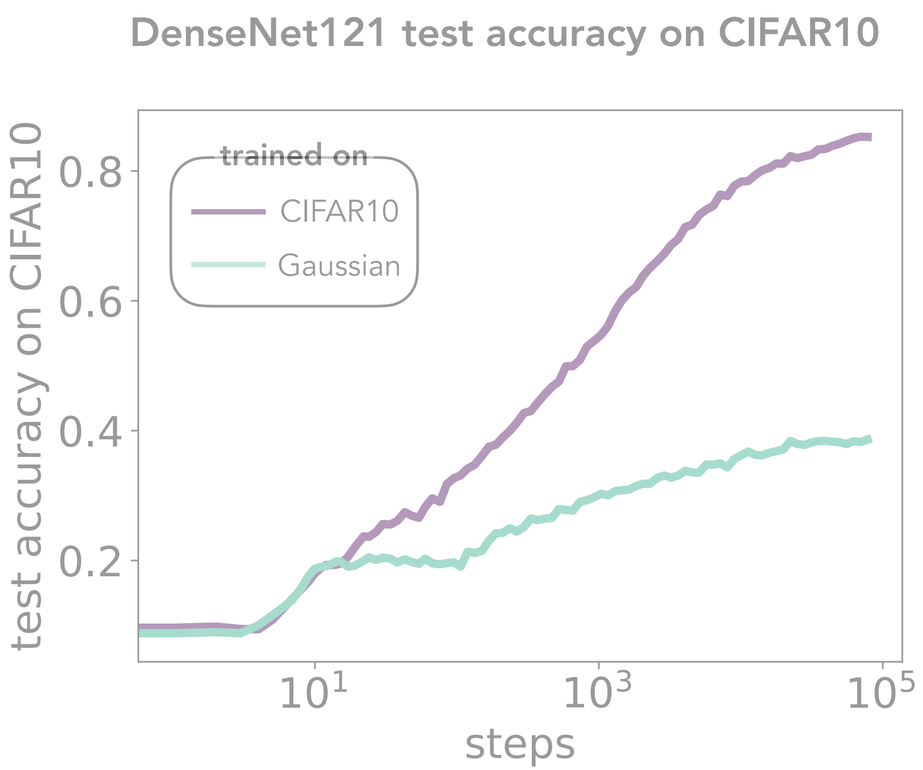




- HOCs improve generalisation (even in two-layer NN!)
- But we do not have theory for how!

The Gaussian world… is not enough

Higher-order cumulants are key to learning in deep convolutional networks

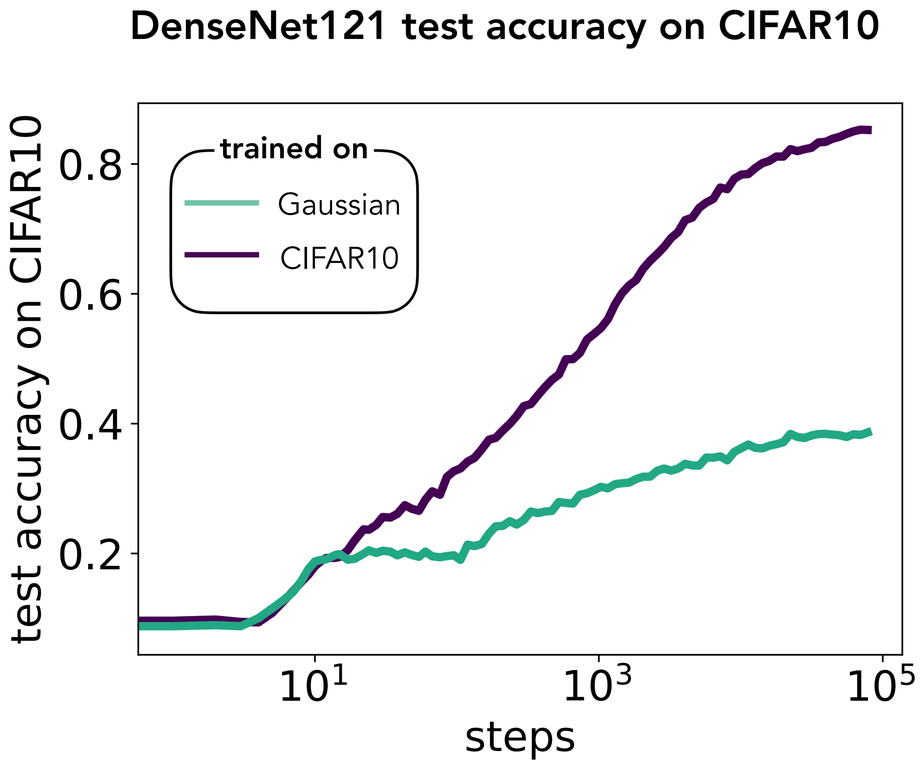


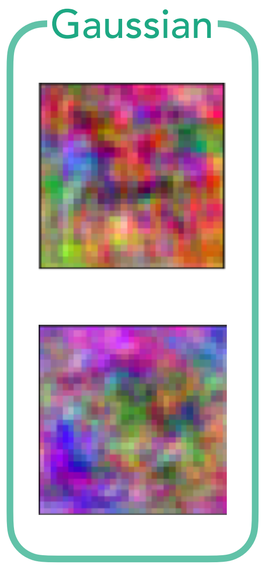
Three questions:

- 🤔 How do NN learn about HOCs?
- 🤔 How efficient are NN at learning from HOCs?
- 🤔 How do HOCs shape learnt representations?

NN learn distributions of increasing complexity

A simplicity bias in neural networks

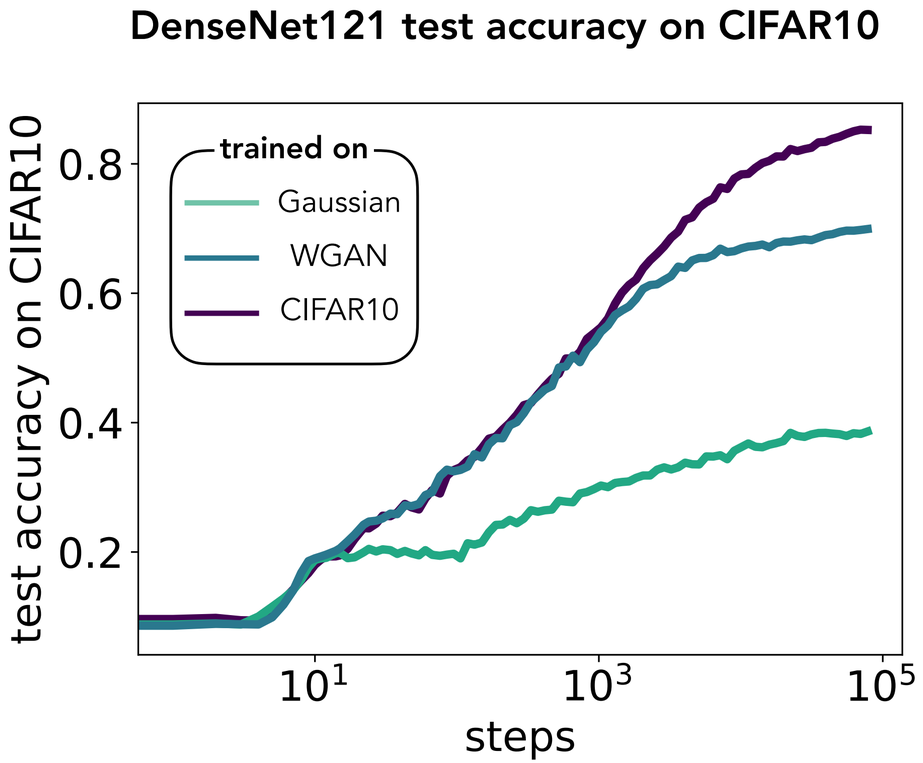


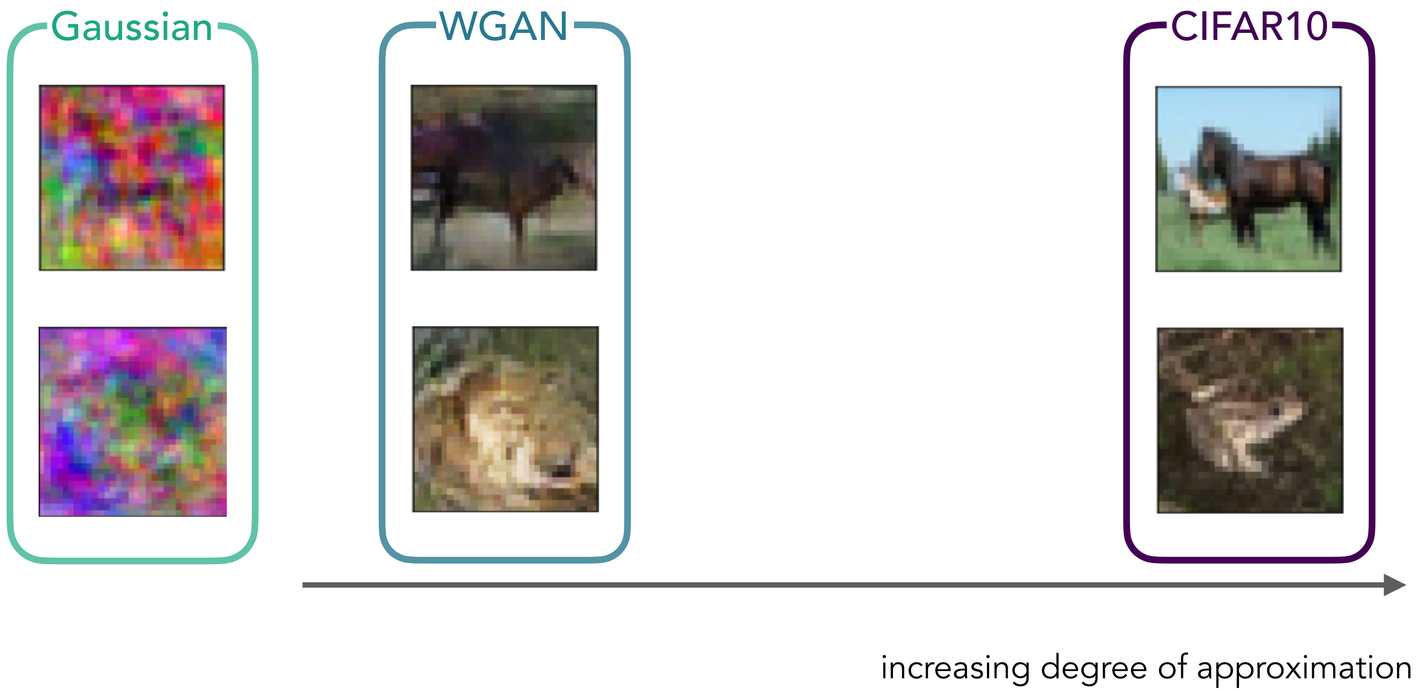


increasing degree of approximation

NN learn distributions of increasing complexity

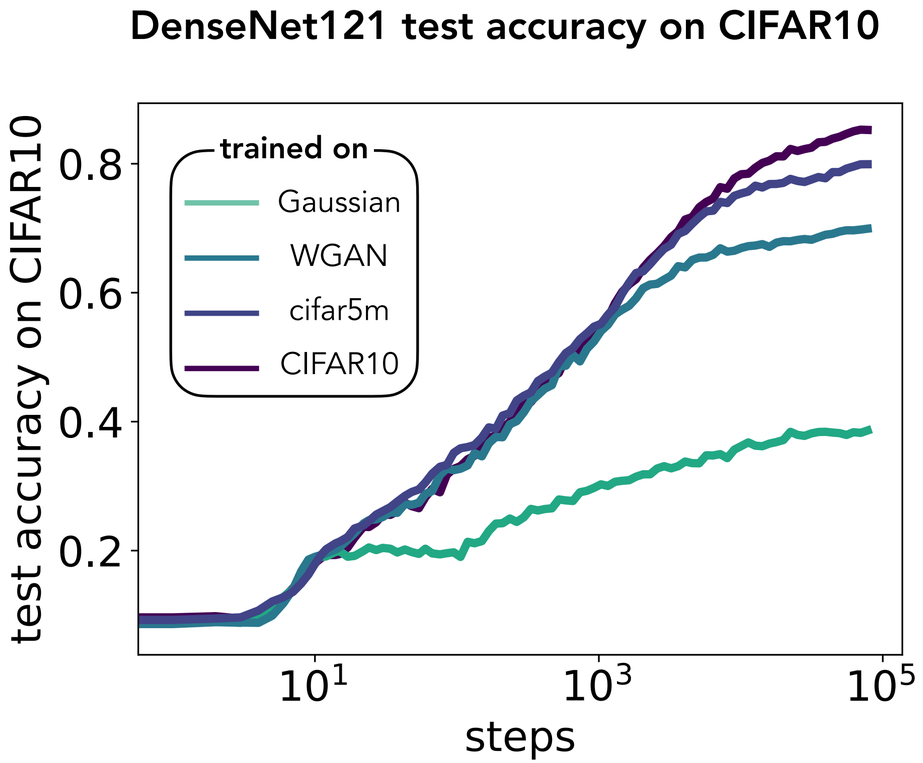
A simplicity bias in neural networks

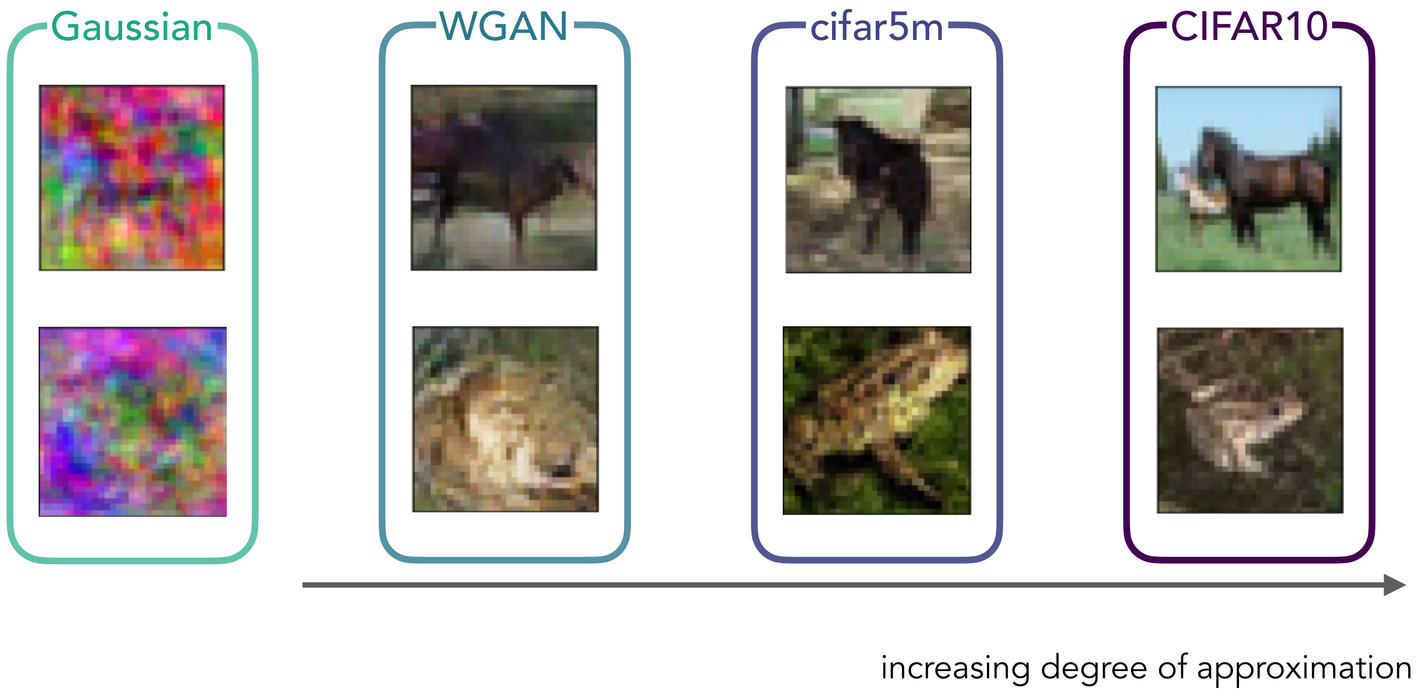




NN learn distributions of increasing complexity

A simplicity bias in neural networks

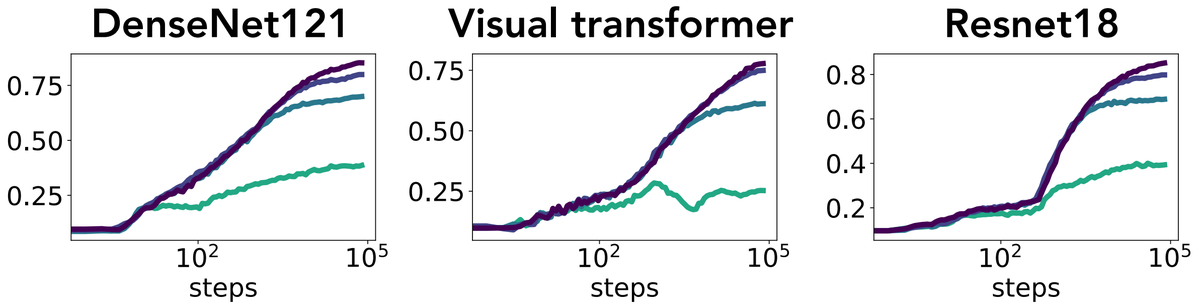


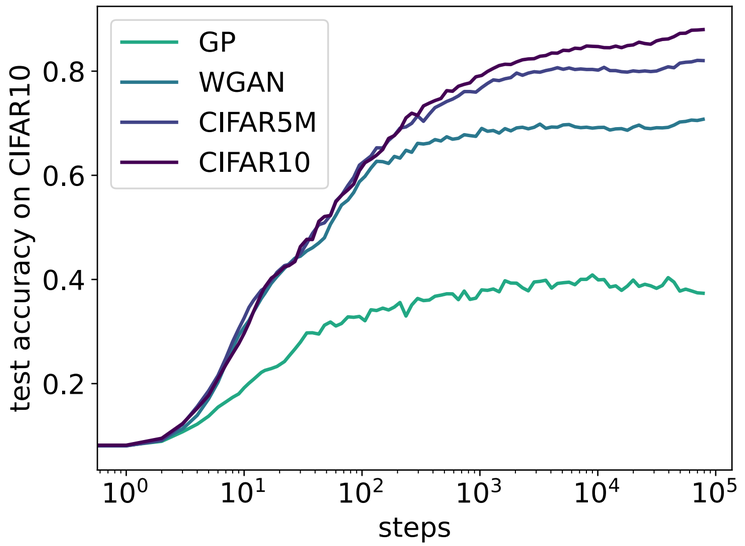


NN learn distributions of increasing complexity

Different architectures, same story

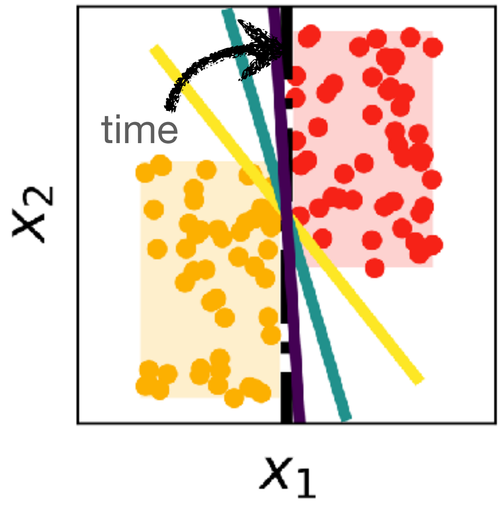
- Behaviour is **robust across architectures** for CIFAR10.
- Even a R**esnet18 pre-trained on ImageNet** learns distributions of increasing complexity (but faster)
- So what do you actually “learn” from pre-training? Is it “just” about the optimisation?

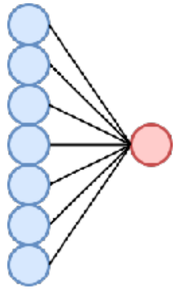




A day in the life... of a perceptron

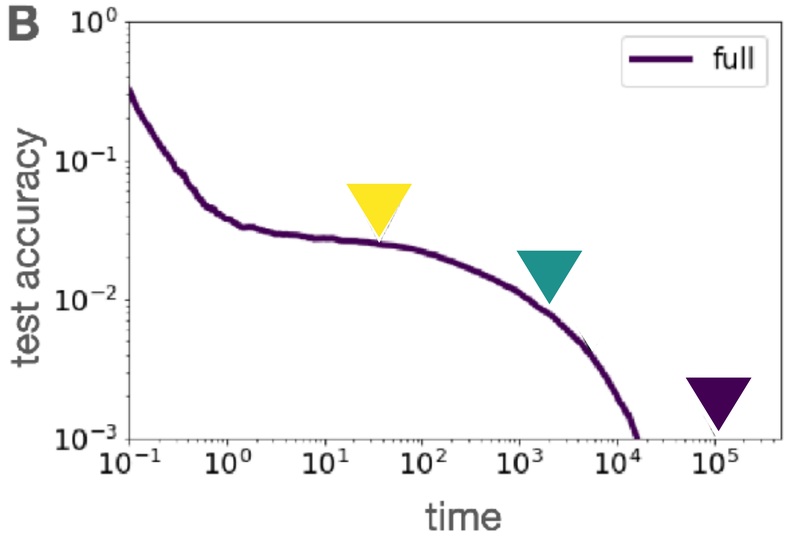
A perceptron learns to take increasingly higher-order statistics into account





$$\hat{y} = \sigma(\lambda), \quad \lambda \equiv w_i x^i / \sqrt{D},$$

GF: $\tau \dot{w}_i = -\mathbb{E} \nabla_{w_i} L(w)$

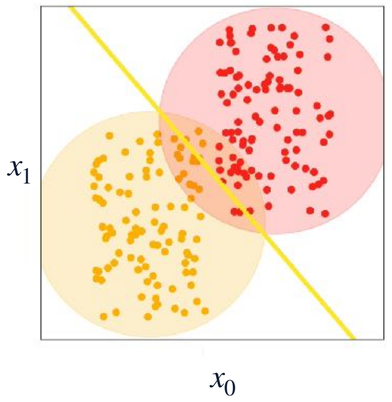


▶	**0th order** in w	$w^{(0)} \propto m$	distance btw **means** m
▶	**1st order**	$w^{(1)} \propto (\kappa_w)^{-1} m$	Fisher discriminant: use **covariance**
▶	**3th order**	$w_i^{(c)} \propto -(\kappa_w)_{ij} \kappa_w^{j,k,l,m} w_k^{(1)} w_l^{(1)} w_m^{(1)}$	first non-Gaussian correction

A complementary point of view

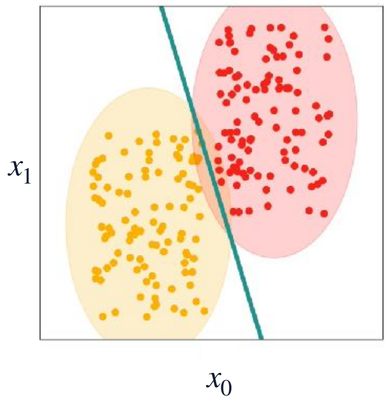
A perceptron “sees” distributions of increasing complexity

mean



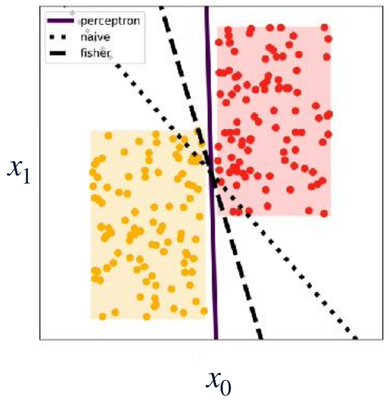
$w^{(0)} \propto m$

mean & covariance

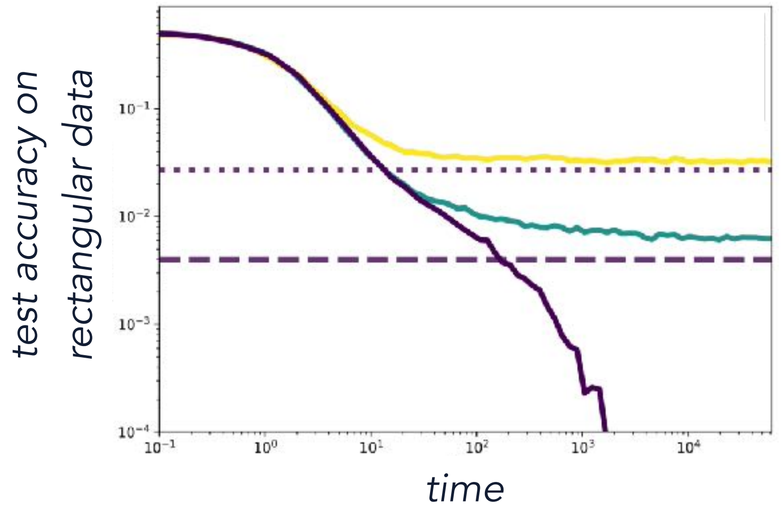


$w^{(1)} \propto \kappa_{\mathrm{w}}^{-1} m$

higher-order cumulants



etc…



Refinetti, Ingrosso & Goldt — ICML ’23

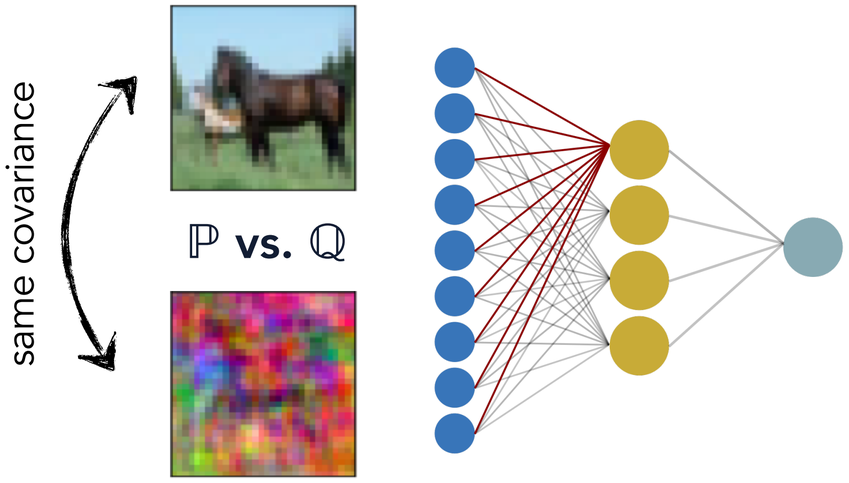
Take-away #1: Neural nets learn
distributions of increasing complexity

But **how much data** do you need to
learn from higher-order cumulants?

How many samples to detect data structure?

The second moment method for contiguity

Learning HOCs as an hypothesis test:



Likelihood ratio testing:

$$L(X) \equiv \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(Y)$$

(an optimal statistical test)

Second moment for contiguity: (Le Cam '60)

$$\|L\|^2 \equiv \mathbb{E}_{X\sim\mathbb{Q}}\, L(X)^2 \to \infty\,?$$

How many samples to detect data structure?

The low-degree likelihood ratio

Likelihood ratio testing:

$$L(X) \equiv \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(Y)$$

(an optimal statistical test)

Second moment for contiguity: (Le Cam '60)

$$\|L\|^2 \equiv \mathbb{E}_{X\sim\mathbb{Q}}\, L(X)^2 \to \infty\,?$$

Low-degree likelihood ratio (LDLR): Hopkins & Steurer (2017)

Orthogonal projection of L into subspace of polynomials of degree at most D:

$$L^{\leq D} \equiv \mathscr{P}^{\leq D} L$$

Conjecture (informal): For "nice" $\mathbb{P}, \mathbb{Q}$, if there exists $D(n) \geq (\log n)^{1+\epsilon}$ for which $L^{\leq D}$ remains bounded, no polynomial-time algorithm strongly distinguishes $\mathbb{P}, \mathbb{Q}$.

Review by Kunisky, Wein & Bandeira (2022)

How many samples to detect data structure?

Gaussian data models

Gaussian additive models (Wigner spiked matrix)

under $\mathbb{P}$, observe $\mathbf{X} = \beta uu^\top + \mathbf{W} \in \mathbb{R}^{n \times n}$

under $\mathbb{Q}$, observe $\mathbf{X} = \mathbf{W}$

Review by Kunisky, Wein & Bandeira (2022)

Can be easily extended to the **Spiked Wishart model:**

under $\mathbb{P}$, observe $\mathbf{X} = (\mathbf{x}^\mu)$

under $\mathbb{Q}$, observe $\mathbf{x}^\mu = \mathbf{w}$

$$\mathbf{x}^\mu = \sqrt{\frac{\beta}{d}} g^\mu \mathbf{u} + \mathbf{w} \qquad g^\mu \sim \mathcal{N}(0,1)$$

LDLR recovers the BBP transition: **linear sample complexity.**

How many samples to detect data structure?

From spiked covariances to spiked cumulants

Spiked cumulant model:

Székely*, Bardone*, Gerace, Goldt — *arXiv:2309.xxxxx*

under $\mathbb{P}$, observe $\mathbf{X} = (\mathbf{x}^\mu)$

$$\mathbf{x}^\mu = S\tilde{\mathbf{x}}^\mu \qquad \tilde{\mathbf{x}}^\mu = \sqrt{\frac{\beta}{d}} g^\mu \mathbf{u} + \mathbf{w} \qquad g^\mu \sim \text{Laplace}$$

whitening matrix

How many samples do you need to distinguish $\mathbb{P}$ and $\mathbb{Q}$?

How many samples to detect data structure?

From spiked covariances to spiked cumulants

Spiked cumulant model:

Székely*, Bardone*, Gerace, Goldt — *arXiv:2309.xxxxx*

under $\mathbb{P}$, observe $\quad \mathbf{X} = (\mathbf{x}^\mu) \qquad \mathbf{x}^\mu = S\tilde{\mathbf{x}}^\mu \qquad \tilde{\mathbf{x}}^\mu = \sqrt{\frac{\beta}{d}} g^\mu \mathbf{u} + \mathbf{w} \qquad g^\mu \sim \text{Rademacher}$

LDLR analysis of spiked cumulant model: Let $\varepsilon > 0, \quad D(n) \asymp \log^{1+\varepsilon}(n)$. Consider $n, d \to \infty$ with $n \asymp d^\theta$.

lower bound: $\|L_{n,d}^{\leq D(n)}\|^2 \geq \left(\frac{1}{\lfloor D(n)/4 \rfloor} \left(\frac{\beta^2 \kappa_4^g}{(1+\beta)^2} \right)^2 \frac{n}{d^2} \right)^{\lfloor D(n)/4 \rfloor}$

upper bound: $\|L_{n,d}^{\leq D(n)}\|^2 \leq C \left(\frac{\beta}{1+\beta} D(n)^8 \left(\frac{n}{d^2} \right)^{1/4} \right)^{D(n)}$

$$\lim_{n,d\to\infty} \|L_{n,d}^{\leq D(n)}\| = \begin{cases} 0 & 0 < \theta < 2 \\ +\infty & \theta > 2 \end{cases}$$

Lorenzo Bardone
(SISSA)

A simple model for images

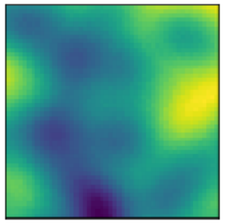
Modelling images using translation invariance

A. Ingrosso

Start from translation-invariant Gaussians:

$$\mathbb{E}z_{ij} = 0$$

$$\mathbb{E}z_{1k}z_{1l} = \exp\left(-|k-l|/\xi\right)$$

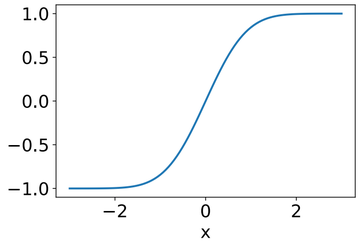


Edges are the 'Independent Components' of Natural Scenes.

Anthony J. Bell and Terrence J. Sejnowski
Computational Neurobiology Laboratory
The Salk Institute
10010 N. Torrey Pines Road
La Jolla, California 92037
tony@salk.edu, terry@salk.edu

A simple model for images

Modelling images using translation invariance and sharp edges



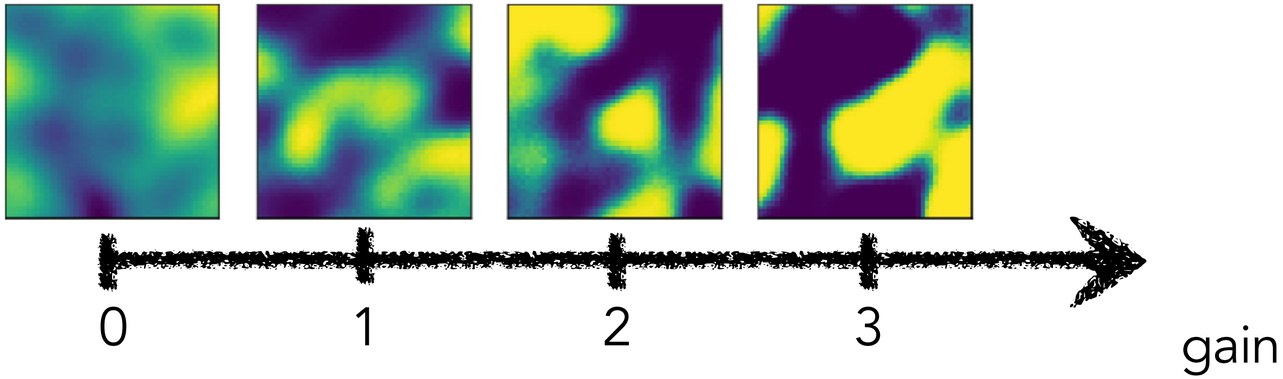
Start from translation-invariant Gaussians:

$$\mathbb{E}z_{ij} = 0$$

$$\mathbb{E}z_{1k}z_{1l} = \exp\left(-|k-l|/\xi\right)$$

and pass them through a **saturating non-linearity:**

$$x_{ij} = \frac{1}{Z}\mathrm{erf}\left(gz_{ij}\right)$$

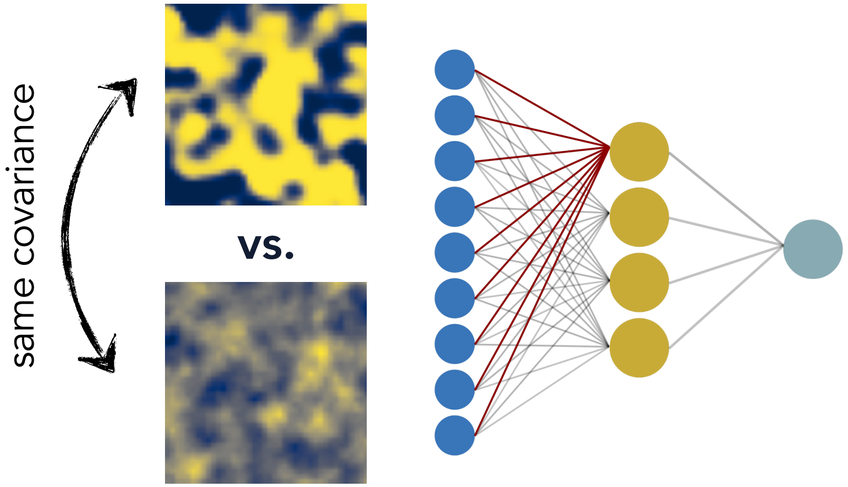


2LNN detect data structure efficiently

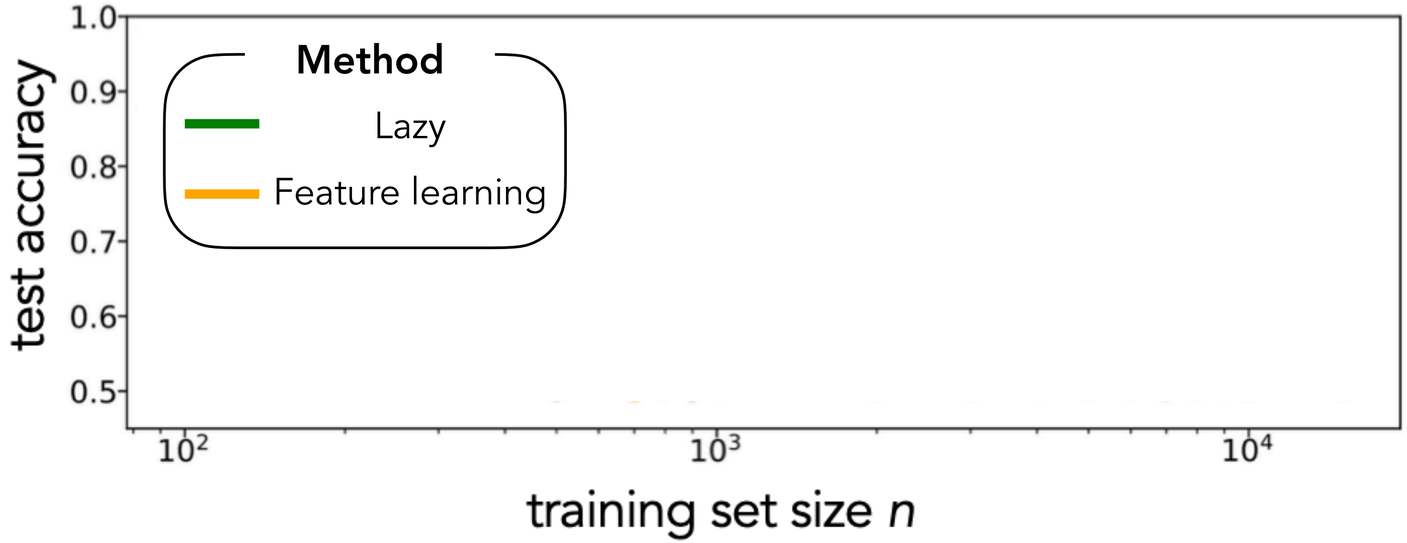
More efficiently than random features, at least

Eszter Székely!

Learning HOCs as an hypothesis test:



Lazy vs feature learning:

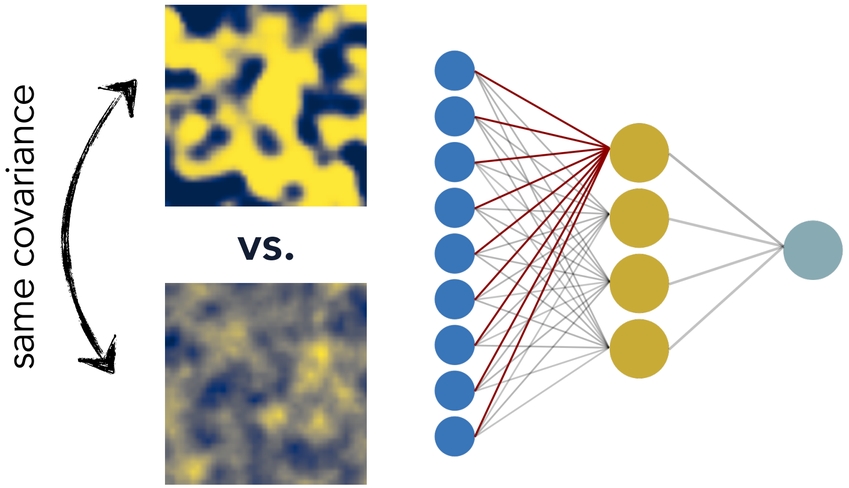


Székely*, Bardone*, Gerace, Goldt — *arXiv:2309.xxxxx*

2LNN detect data structure efficiently

More efficiently than random features, at least

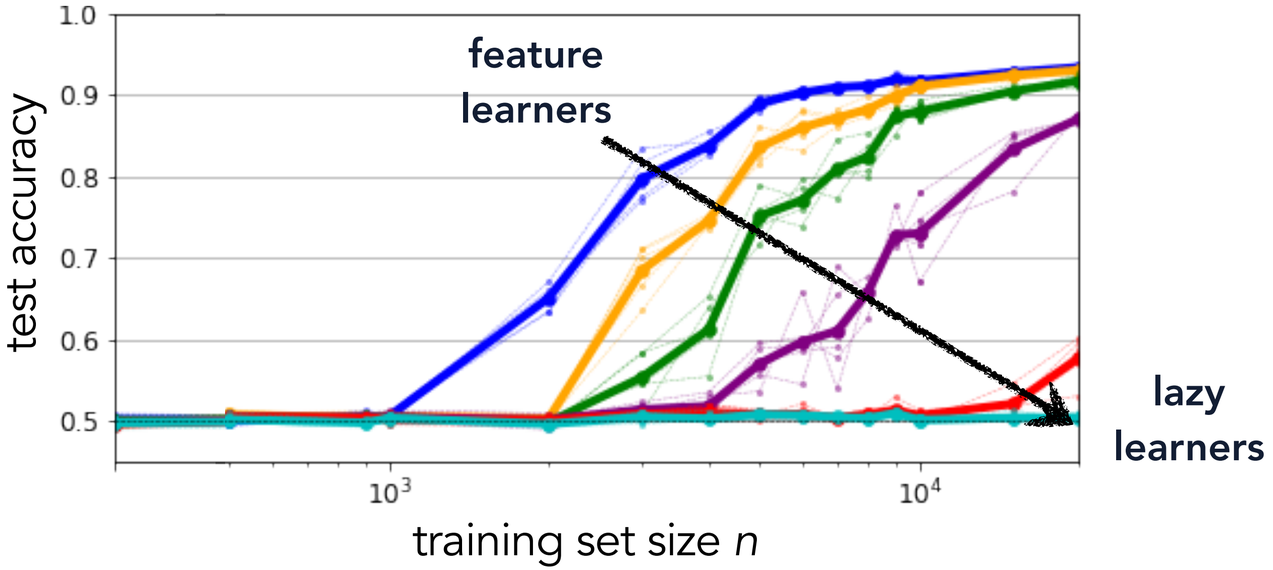
Learning HOCs as an hypothesis test:



Interpolating between lazy and feature learning:

$$\mathscr{L}(\theta) = \frac{1}{\alpha^2} \sum_{i}^{n} \left(y_i - \alpha\, \tilde{f}(x_i; \theta) \right)$$

Chizat, Oyallon, Bach
NeurIPS 2019



Székely*, Bardone*, Gerace, Goldt — *arXiv:2309.xxxxx*

Take-away #2: Neural nets are **efficient** at learning from higher-order cumulants

But how do cumulants shape the **representations** that they learn?

A tale of two mixtures

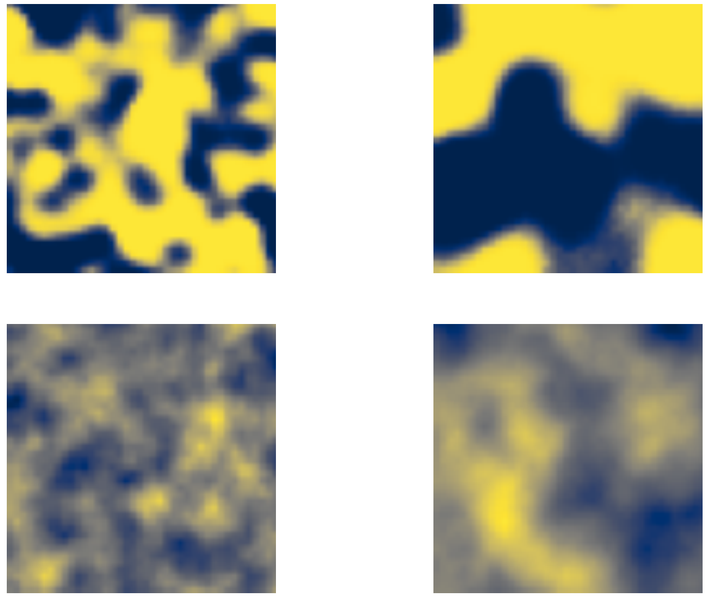
Ingrosso & Goldt
arXiv:2202.00565, *PNAS 2022*

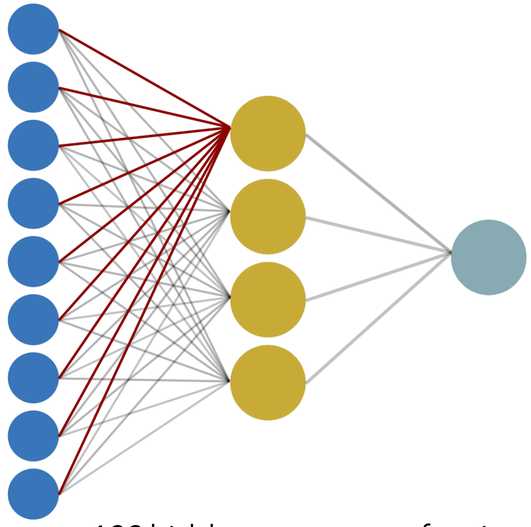
Non-Gaussian, translation-invariant inputs yield localised receptive fields

Start from translation-invariant Gaussians: $\mathbb{E}z_{ij}^{\pm} = 0$, $\mathbb{E}z_{1k}^{\pm}z_{1l}^{\pm} = \exp\left(|k-l|/\xi^{\pm}\right)$

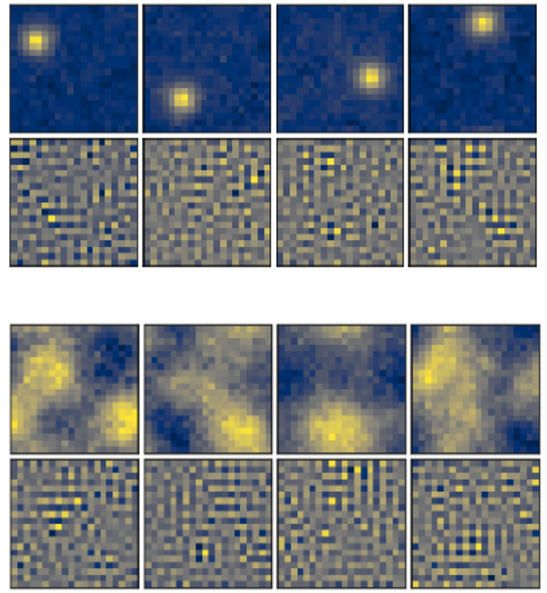
to generate **binary mixture** of **non-linear Gaussian processes** $x_{ij}^{\pm} = \frac{1}{Z}\mathrm{erf}\left(gz_{ij}^{\pm}\right)$

short-range correlations **vs.** **long-range** correlations





100 hidden neurons, erf activation, online SGD, quadratic error



localised

oscillatory

low-freq

high-freq

A tale of two mixtures

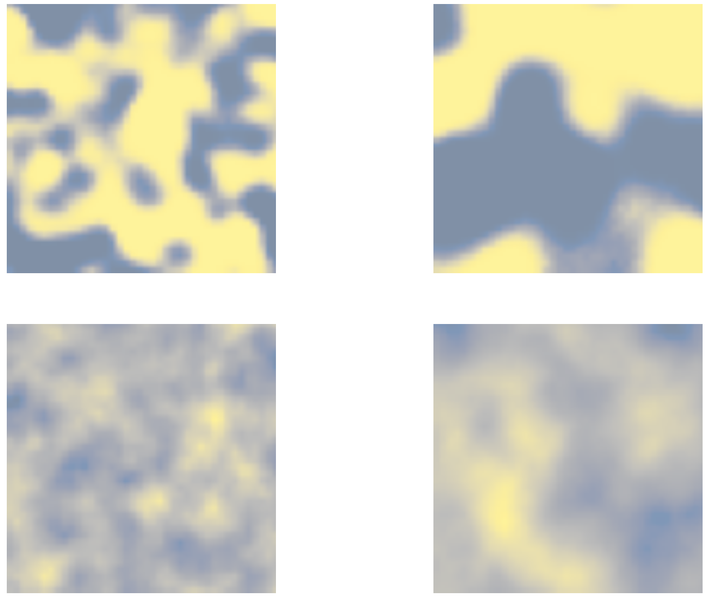
Ingrosso & Goldt
arXiv:2202.00565, *PNAS 2022*

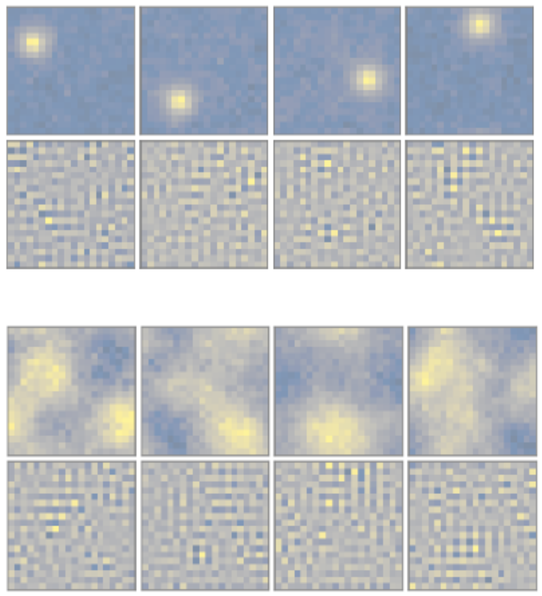
Non-Gaussian, translation-invariant inputs yield localised receptive fields

Start from translation-invariant Gaussians: $\mathbb{E}z_{ij}^{\pm} = 0$, $\mathbb{E}z_{1k}^{\pm}z_{1l}^{\pm} = \exp\left(|k-l|/\xi^{\pm}\right)$

to generate **binary mixture** of **non-linear Gaussian processes** $x_{ij}^{\pm} = \frac{1}{Z}\mathrm{erf}\left(gz_{ij}^{\pm}\right)$

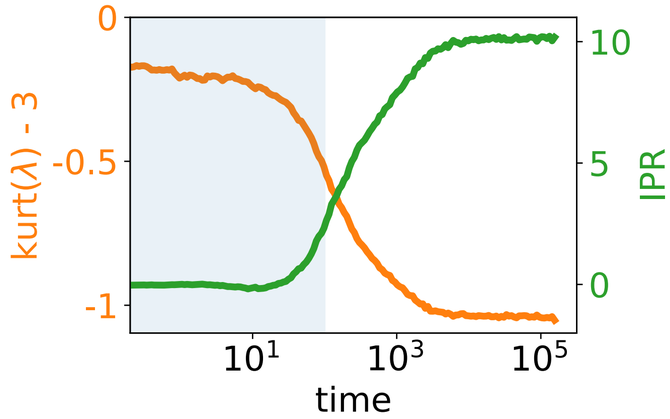
short-range correlations **vs.** **long-range** correlations





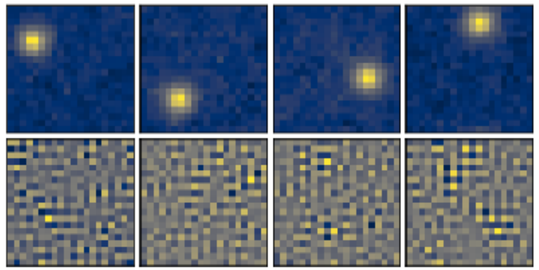
Receptive fields form beyond the Gaussian regime:

$\lambda \propto w \,.\, x$ $\qquad \mathrm{kurt}(\lambda) = \mathbb{E}\lambda^4 - 3\left(\mathbb{E}\lambda^2\right)^2$

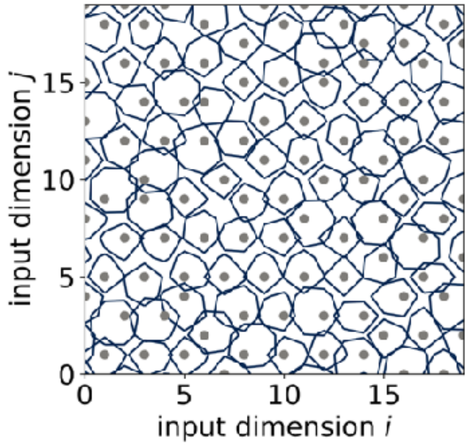


Data-driven emergence of convolutions

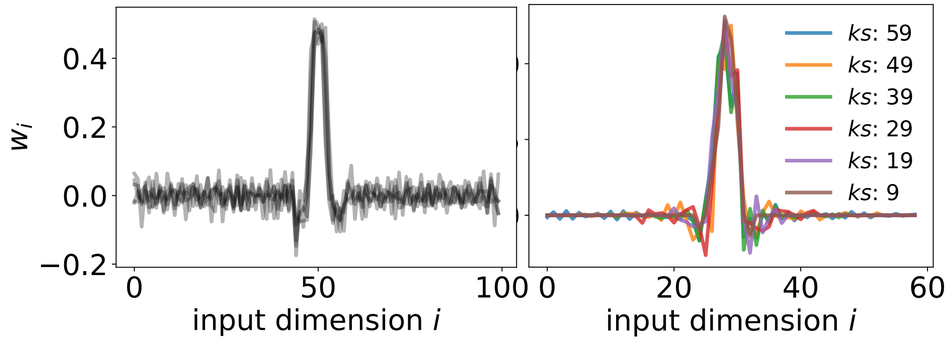
Two-layer networks learn a convolution with one filter directly from data



Translation invariance
+ non-Gaussian inputs
= **localised receptive fields**



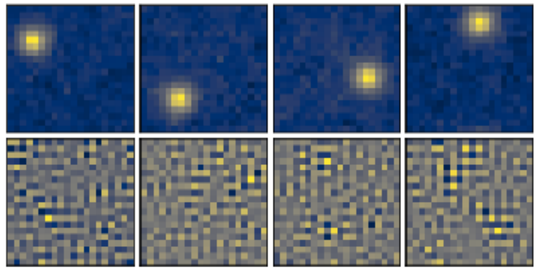
Receptive fields
tesselate input space



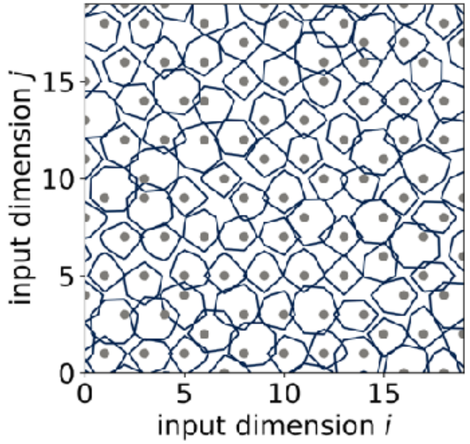
Receptive fields
resemble **convolutions**

Data-driven emergence of convolutions

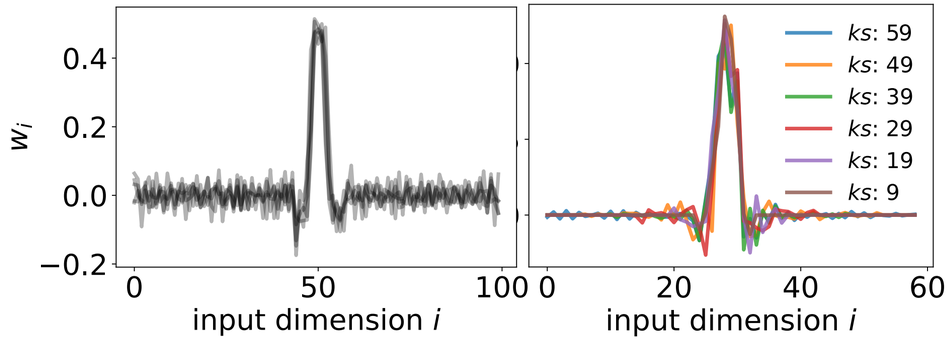
Receptive fields tesselate input space and resemble convolutional filters



Localised receptive fields



tesselate input space



and **resemble convolutions**

More localised RF:
via efficient coding (Olshausen & Field ’96)
or unsupervised learning (Harsh et al. ’20,
Farrell et al. ’21, Benna & Fusi ’21

Learning convolutions:
Neyshabur (NeurIPS 2020) using learnt regularisation;
Pellegrini & Biroli (ICML 2022) using pruning

What's going on?

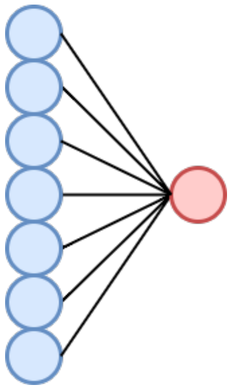
FC networks trained on images
do **not** learn convolutions!

(and they perform worse…)

(e.g. Urban et al. '18, d'Ascoli et al., '20, Neyshabur, '21)

How do HOCs generate localised RF?

The learning dynamics of a single neuron



$$y = \sigma(wx)$$

$$\langle \dot{\boldsymbol{w}} \rangle = -\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \sum_{\mu=1}^{2} \left[c_2^{\mu} C^{\mu} \boldsymbol{w} + c_4 T^{\mu} \boldsymbol{w}^{\otimes 3} \right]$$

$$\left\langle x_i^{\mu} x_j^{\mu} \right\rangle \qquad \left\langle x_i^{\mu} x_j^{\mu} x_k^{\mu} x_{\ell}^{\mu} \right\rangle = C_{ij}^{\mu} C_{k\ell}^{\mu} + C_{ik}^{\mu} C_{j\ell}^{\mu} + C_{i\ell}^{\mu} C_{jk}^{\mu} + \Delta T_{ijk\ell}^{\mu}$$

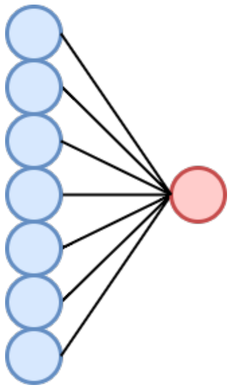
Two-pixel correlations

Four-pixel correlations

Gaussian and non-Gaussian contribution

How do HOCs generate localised RF?

The learning dynamics of a single neuron



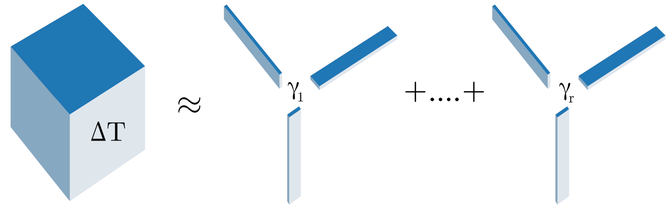
$$y = \sigma(wx)$$

$$\langle \dot{\boldsymbol{w}} \rangle = -\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \sum_{\mu=2}^{M} \left(c_2^{\mu} + c_4 q^{\mu} \right) C^{\mu} \boldsymbol{w} + \frac{c_4}{M} \sum_{\mu=1}^{M} \Delta T^{\mu} \boldsymbol{w}^{\otimes 3}$$

Gaussian

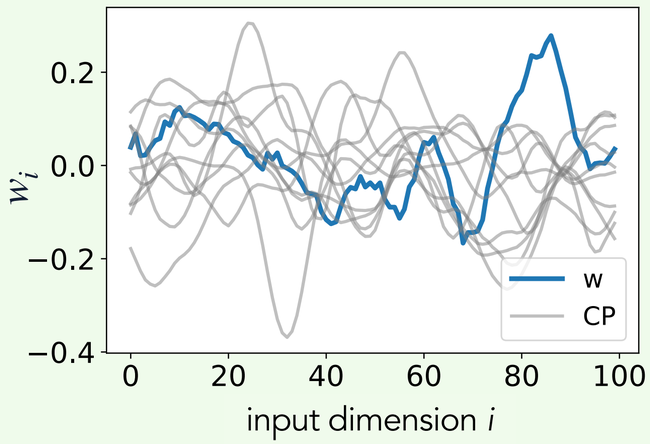
Non-Gaussian

Tensor decomposition



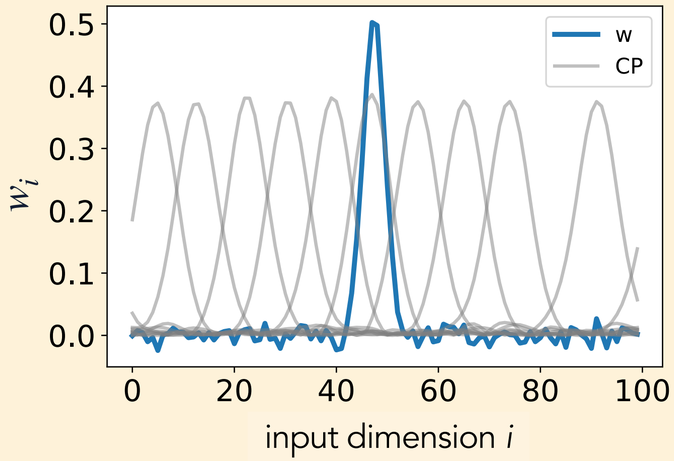
Blurry images

- CP factors oscillate
- weight vector is not localised.



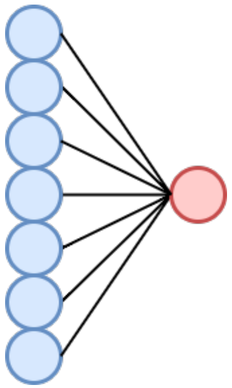
Sharp edges

- CP factors localise
- and so does the weight



How do HOCs generate localised RF?

The learning dynamics of a single neuron



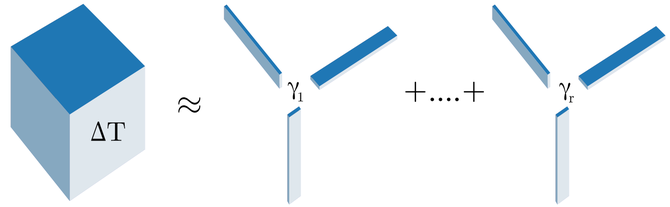
$$y = \sigma(wx)$$

$$\langle \dot{\boldsymbol{w}} \rangle = -\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \sum_{\mu=2}^{M} \left(c_2^{\mu} + c_4 q^{\mu} \right) C^{\mu} \boldsymbol{w} + \frac{c_4}{M} \sum_{\mu=1}^{M} \Delta T^{\mu} \boldsymbol{w}^{\otimes 3}$$

Gaussian

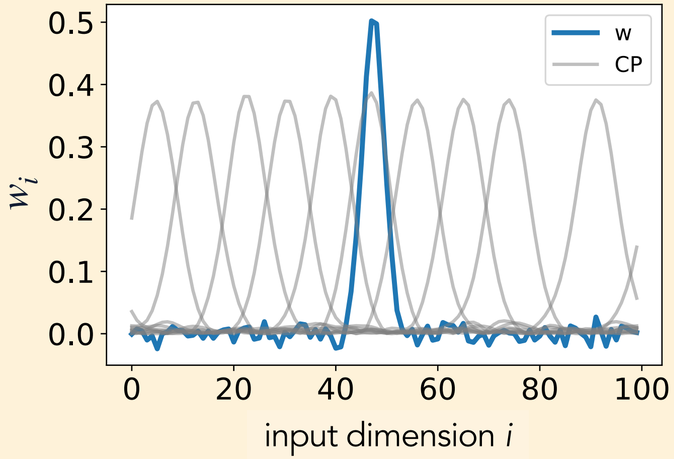
Non-Gaussian

Tensor decomposition

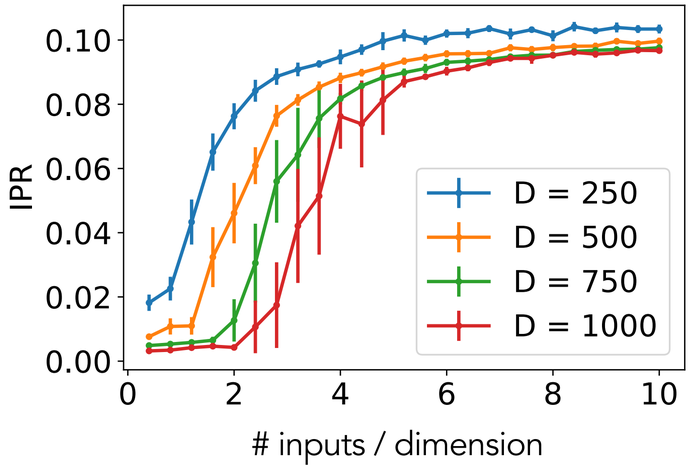


Sharp edges

- CP factors localise
- and so does the weight

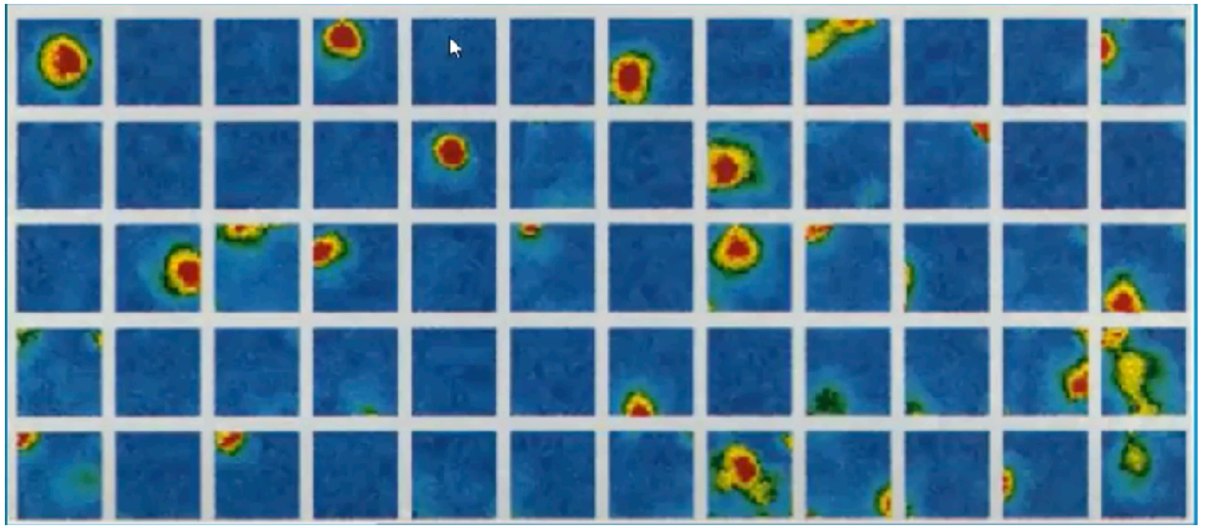


A phase transition?



Place cells, place cells everywhere

Rat CA1 cortex shows similar activation patterns



Rat CA1 place cells from simultaneous recordings, Wilson and McNaughton

Challenge: can we “learn” place cells in recurrent neural networks?

What about other
data modalities?

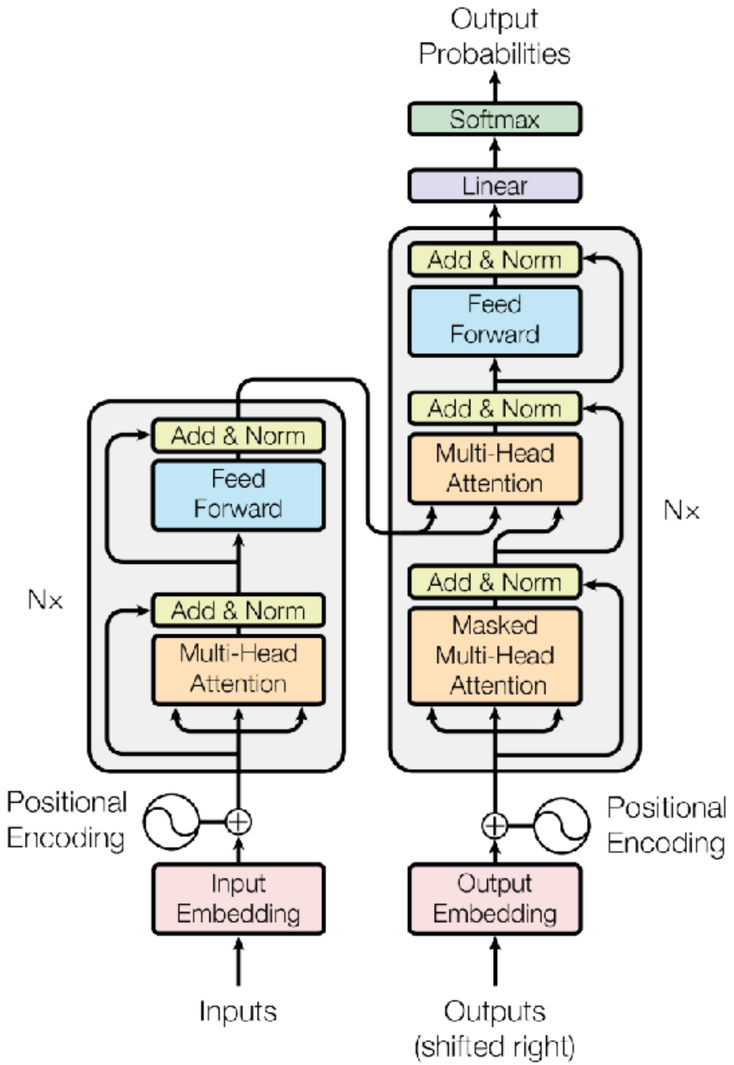
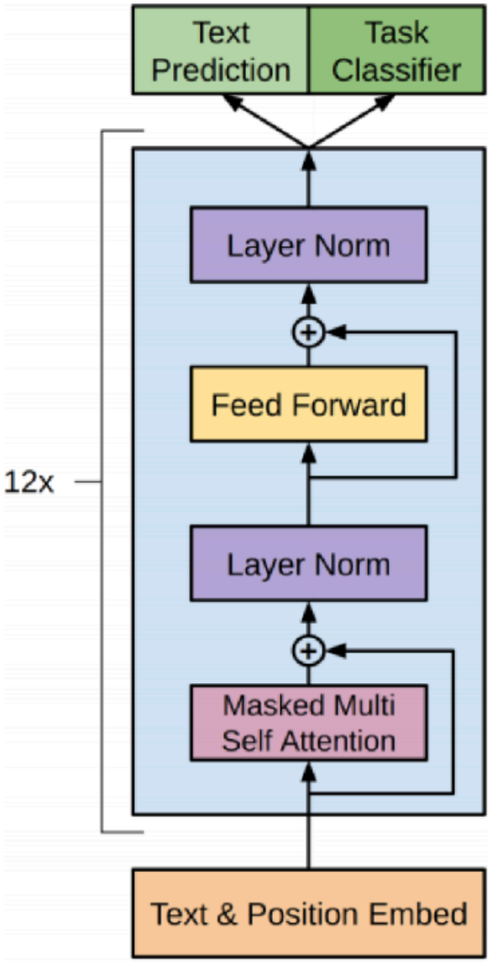
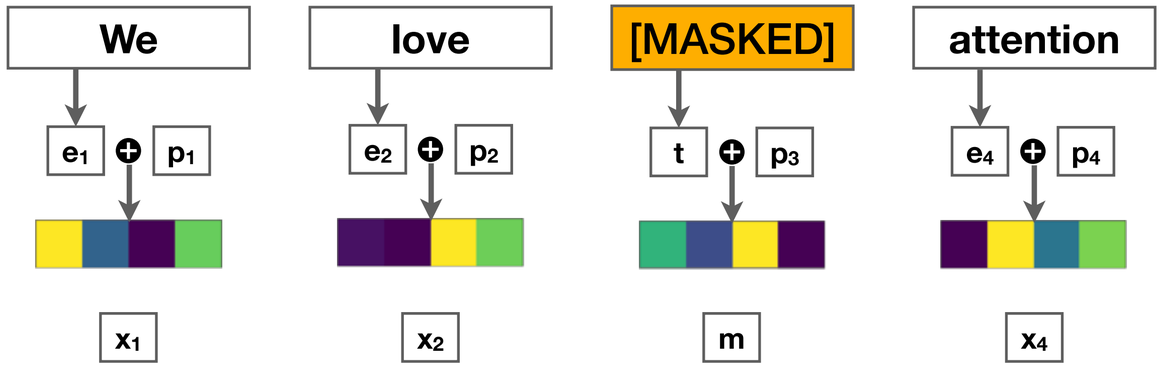


Figure 1: The Transformer - model architecture.


Text Prediction
Task Classifier
Layer Norm
Feed Forward
Layer Norm
Masked Multi Self Attention
12x
Text & Position Embed

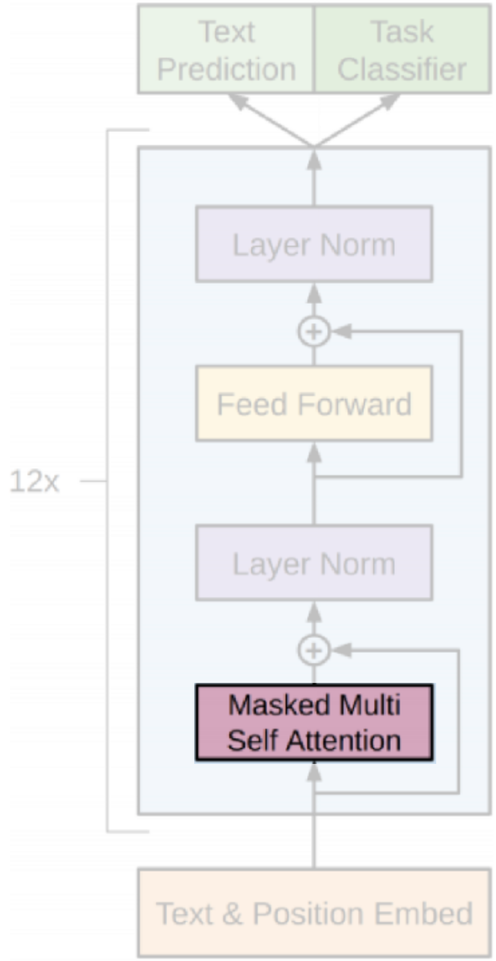
What does a transformer do?

Self-attention uses data-dependent filters to predict missing tokens



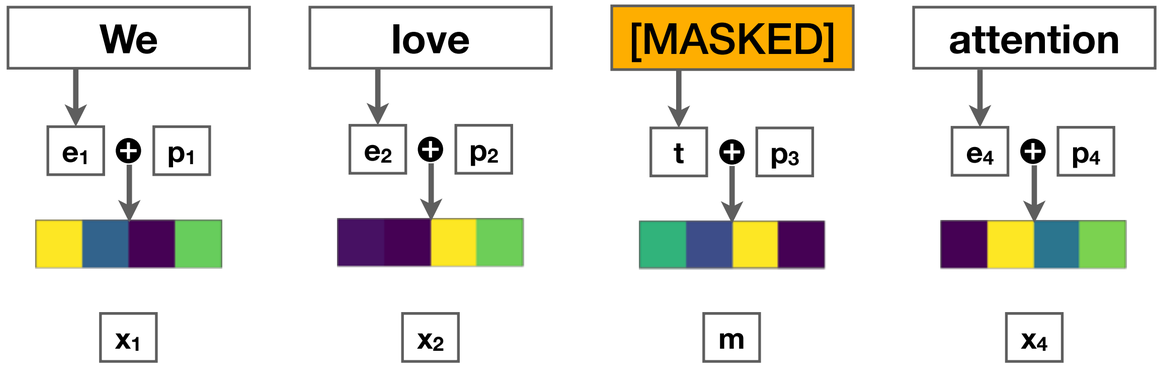
$$\mathbf{y}_3 \propto \sum_{j=1}^{L} \alpha(\mathbf{x}_j, \mathbf{x}_3)\, \mathbf{x}_j$$

Queries $\mathbf{q}_j = W_Q \mathbf{x}_j$ Keys $\mathbf{k}_j = W_K \mathbf{x}_j$ Values $\mathbf{v}_j = W_V \mathbf{x}_j$



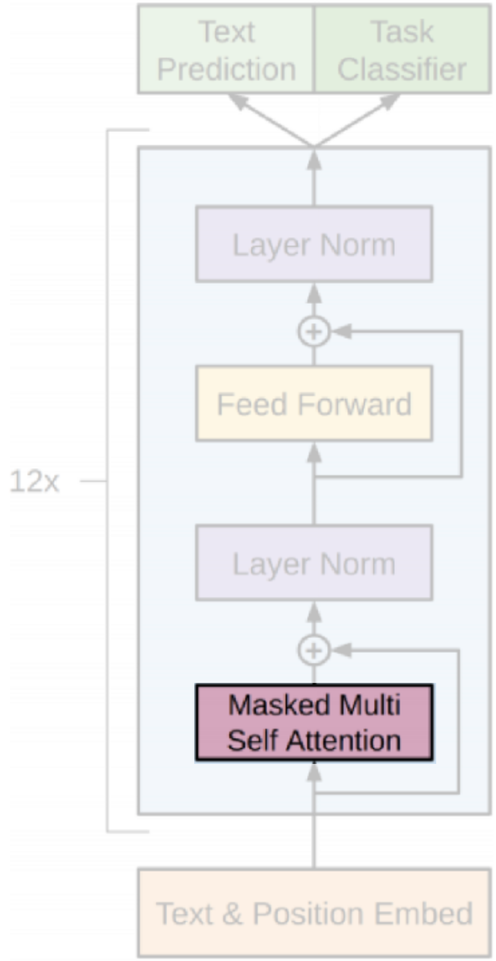
What does a transformer do?

Self-attention uses data-dependent filters to predict missing tokens



$$\mathbf{y}_3 \propto \sum_{j=1}^{L} \exp\left(\mathbf{k}_j^\top \mathbf{q}_3\right) \mathbf{v}_j$$

Queries $\mathbf{q}_j = W_Q \mathbf{x}_j$ Keys $\mathbf{k}_j = W_K \mathbf{x}_j$ Values $\mathbf{v}_j = W_V \mathbf{x}_j$



What does a transformer do?

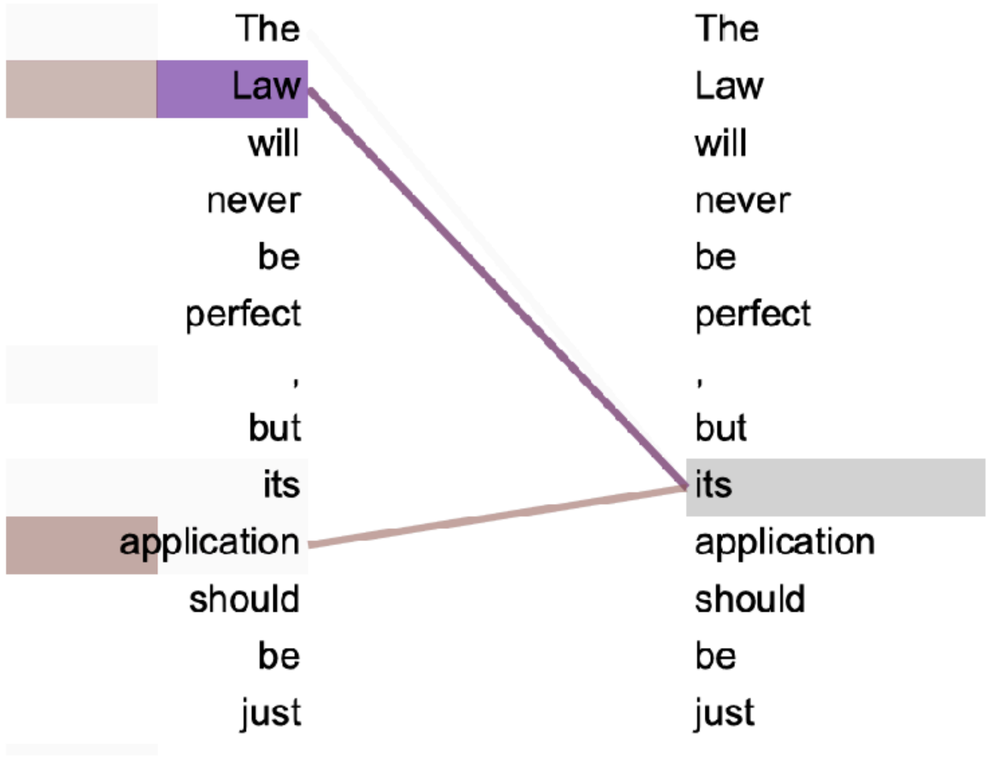
Self-attention uses **data-dependent filters** to predict missing tokens

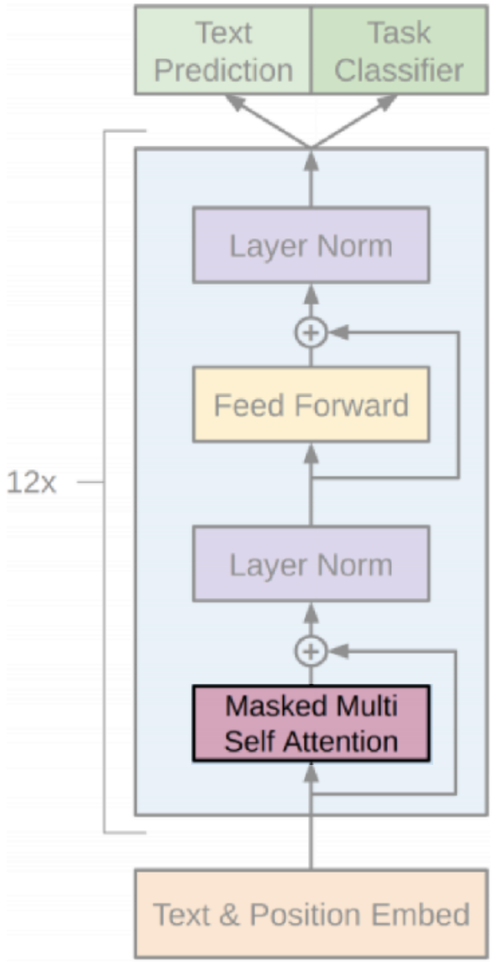
$$\mathbf{y}_3 \propto \sum_{j=1}^{L} \exp\left(\mathbf{k}_j^\top \mathbf{q}_3\right) \mathbf{v}_j$$

Queries $\mathbf{q}_j = W_Q \mathbf{x}_j$

Keys $\mathbf{k}_j = W_K \mathbf{x}_j$

Values $\mathbf{v}_j = W_V \mathbf{x}_j$





(Factorised) SA learns inverse Potts models

Masked language modelling = pseudo-likelihood (Federica's talk)

- Model sequences using a generalised Potts model with q colours.

$$\mathscr{H}(\mathbf{s}) = -\frac{1}{2}\sum_{i,j=1}^{L} J_{ij}^{\star}\mathbf{s}_i^T V^{\star}\mathbf{s}_j$$

- *Factorise* the self-attention mechanism:

$$\mathbf{y}_3 \propto \sum_{j=1}^{L} \exp\left(\mathbf{k}_j^{\top}\mathbf{q}_3\right)\mathbf{v}_j$$

attention -> position

value -> input

- Factored SA + MLM objective = pseudo-likelihood for inverse Potts!

  Besag ('75), Ravikumar et al. ('10), Cocco & Monasson ('11), Ricci-Tersenghi ('12), Ekeberg et al. ('13), Decelle & Ricci-Tersenghi ('14)

Riccardo Rende

Federica Gerace

Alessandro Laio

Rende, Gerace, Laio & Goldt — arXiv:2304.07235

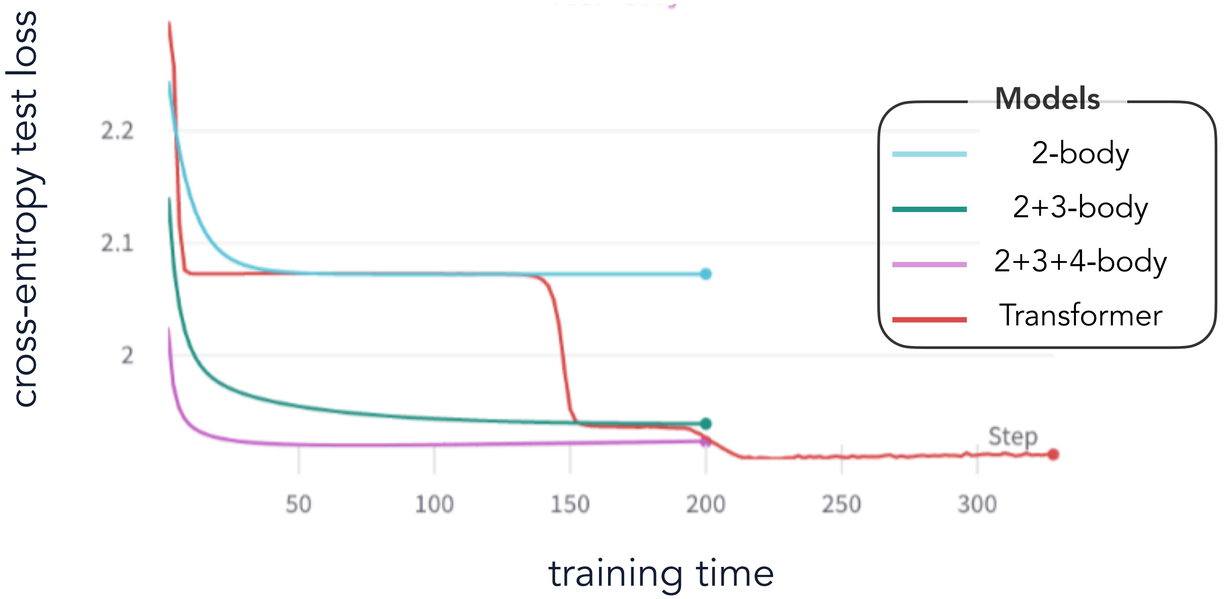
Learning many-body interactions

One step at a time

- Can extend the Hamiltonian to include higher-order interactions (cf. *p*-spin model)

$$\mathscr{H}(\mathbf{s}) \propto -\sum_{i,j=1}^{L} J_{ij}^{(2)} s_i s_j - \sum_{i,j,k=1}^{L} J_{ijk}^{(3)} s_i s_j s_k - \dots$$

- Train a simplified transformer:
  several layers of self-attention followed by point-wise square non-linearity



Rende, Gerace, Laio & Goldt — *forthcoming*

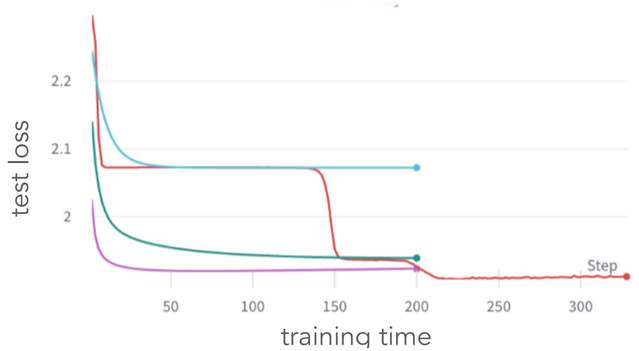
Concluding perspectives

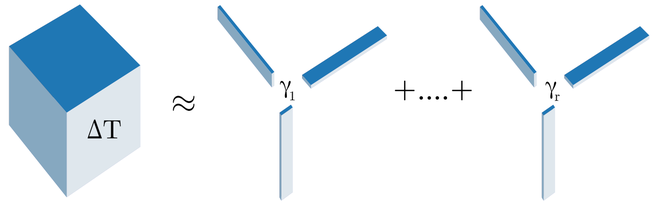
What do neural networks learn from their data?

- Higher-order data statistics are key to understanding feature learning

- Neural networks learn **distributions of increasing complexity,** efficiently.

- **Data symmetries** shape principal components of HOCs, which determine **neural representations**.





Acknowledgements

L. Bardone
SISSA

F. Gerace
SISSA

A. Laio
SISSA

A. Ingrosso
ICTP

W. Redman
UCSB

M. Refinetti
ENS Paris → G Research

R. Rende
SISSA

E. Székely
SISSA

The view from SISSA

datascience.sissa.it

@sebastiangoldt