Neural Redshift: Random Networks are not Random Functions

Damien Teney	Idiap Research Institute
Armand Nicolicioiu	ETH Zurich
Valentin Hartmann	EPFL
Ehsan Abbasnejad	University of Adelaide

Why do neural networks generalize so well?

It’s not SGD

GD without stochasticity works too

Stochastic training is not necessary for generalization, Geiping et al. 2021

Models from **gradient-free** methods also generalize, e.g. rejection sampling

Loss landscapes are all you need: NN generalization can be explained without the implicit bias of grad. descent, Chiang et al. 2022

There is a simplicity bias even in **untrained** language models

The no free lunch theorem, Kolmogorov complexity, and the role of inductive biases in machine learning, Goldblum et al. 2023

Why do neural networks generalize so well?

Not all neural networks generalize well

Some applications need **special architectures**, e.g. sine activations in INRs (NeRF)

Implicit neural representations with periodic activation functions, Sitzmann et al. 2020

Tabular datasets often work better with **decision trees**

Why do tree-based models still outperform deep learning on tabular data, Grinsztajn et al. 2022

Inductive biases

Some properties of common architectures
make them well suited to **most real-world data**

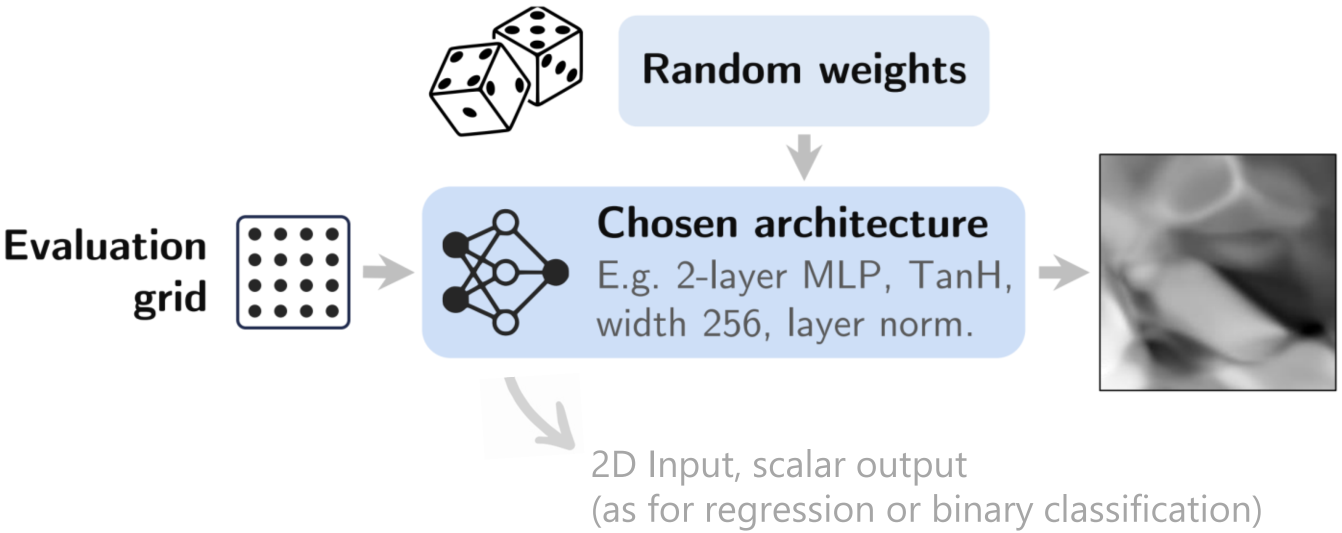
What are these properties?
What gives neural networks these properties?

How to measure these properties?

Existing work looks at models during/after training,

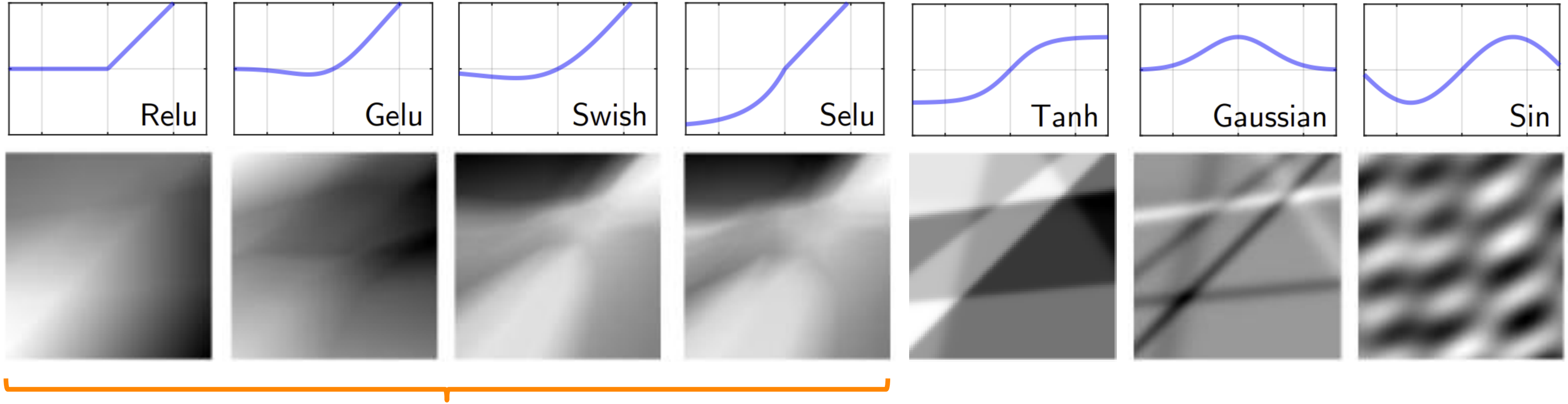
which confounds effects of architectures/optimization. (simplicity bias, spectral bias ...)

We examine random-weight untrained MLPs.



Different activation ⟹ different function ‘shape’

Examples of functions implemented by random-weight, 2D-input networks:



Popular activations
(simplicity bias, smooth functions)

Why should we care about random networks?

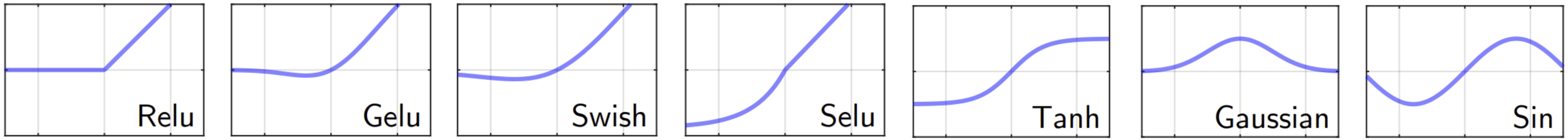
📖 Prior work showed that (S)GD training acts like Bayesian inference.
Random models reflect the **prior distribution over functions**.

➡️ Among the many solutions that fit the training data,
those close to the prior will be favored.

Deep learning generalizes because the parameter-function map is biased towards simple functions, Valle-Perez et al. 2018

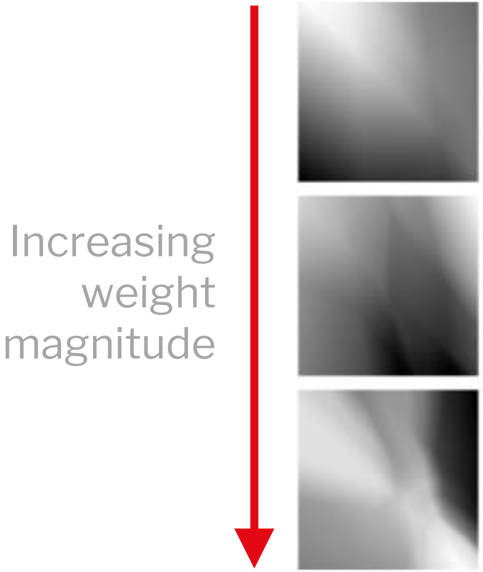
Neural networks are a priori biased towards boolean functions with low entropy, Mingard et al. 2019

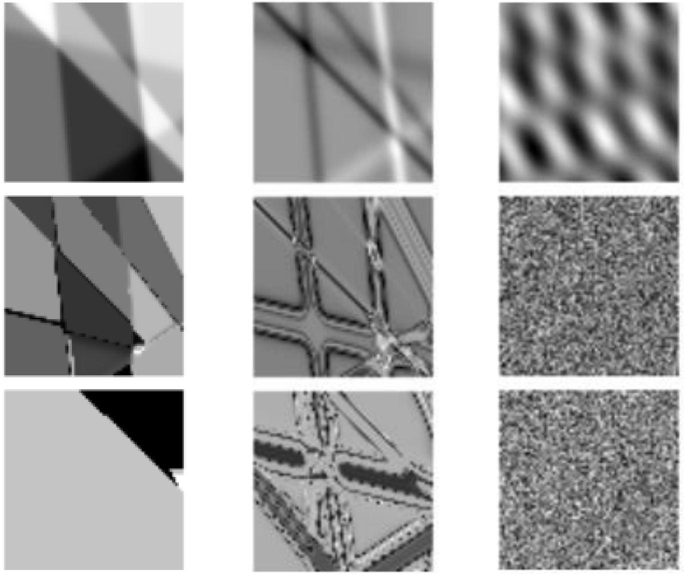
Larger weights/activations ⟹ higher complexity



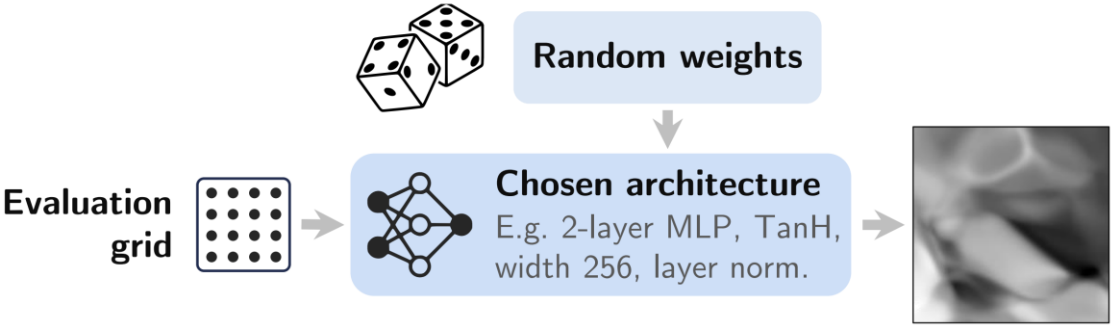
ReLU-like activations:
no/weak sensitivity to weight magnitude

Other activations:
strong sensitivity

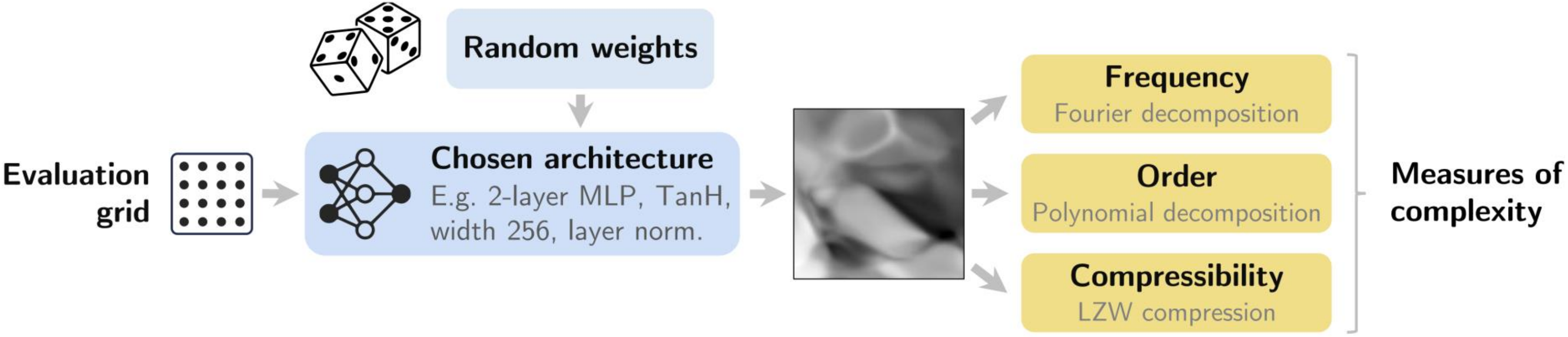




How to quantify these properties?

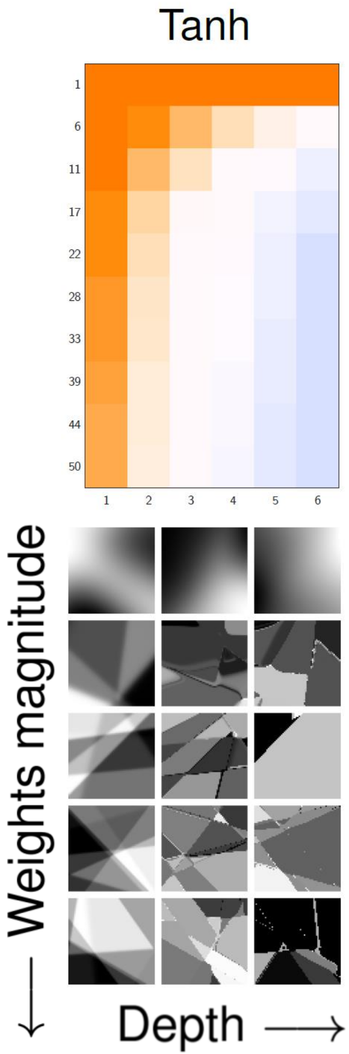


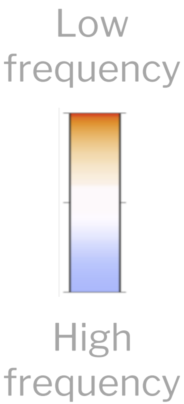
How to quantify these properties?

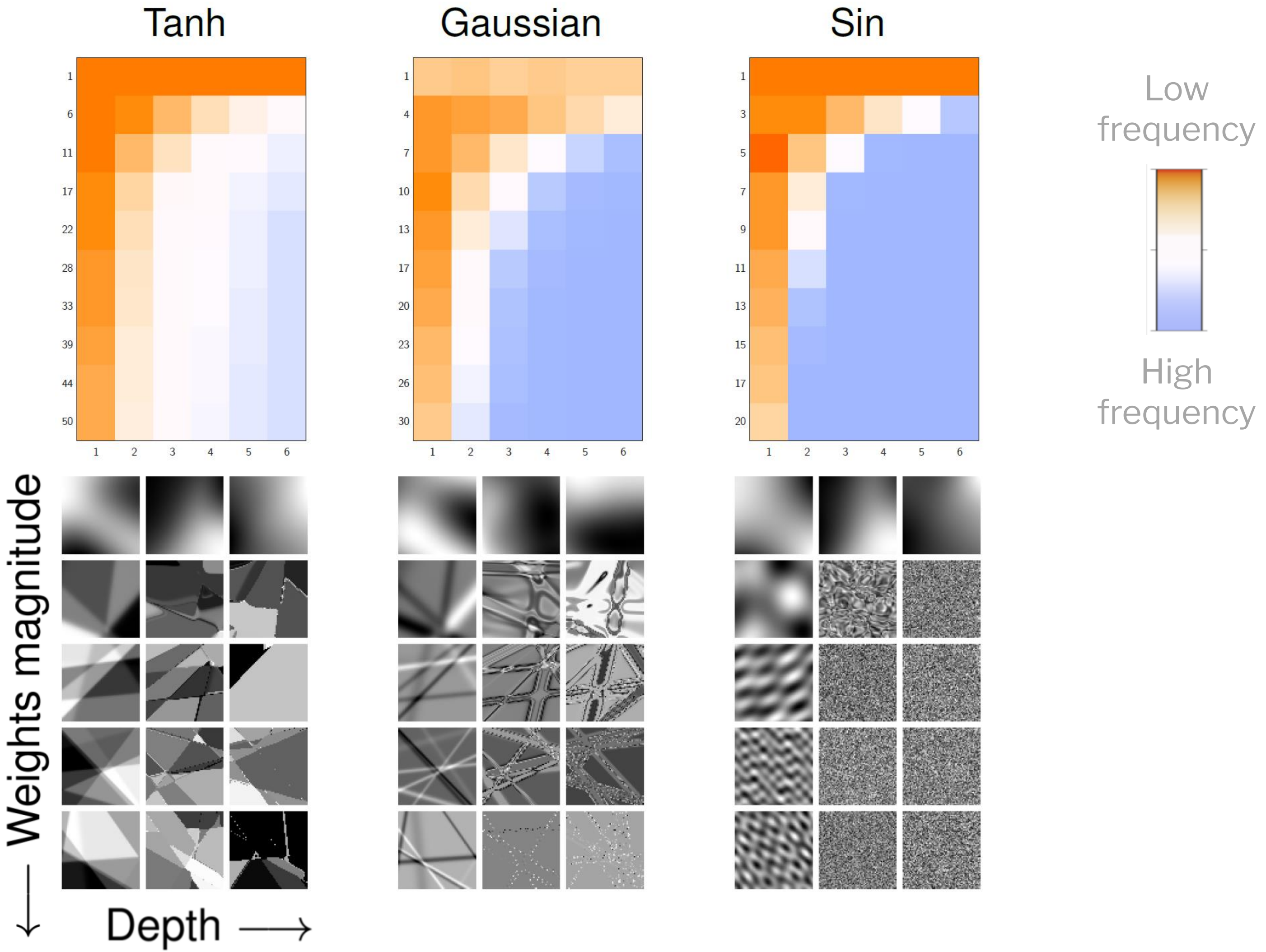


Quantifiable characterizations of inductive biases

- high frequency in Fourier decomposition
- high order in decomposition in polynomial basis
- non-compressibility (dictionary size with LZ compression)

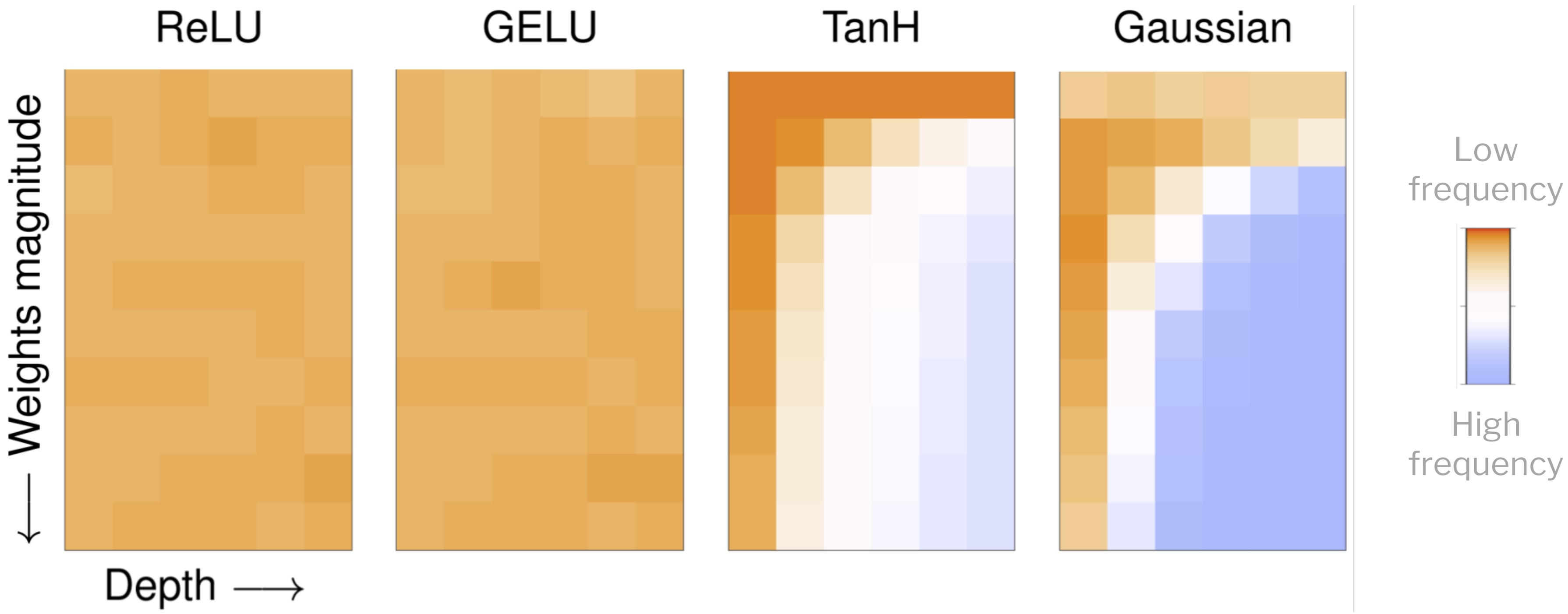

Tanh
1
6
11
17
22
28
33
39
44
50
1 2 3 4 5 6
Weights magnitude
Depth


Low frequency
High frequency

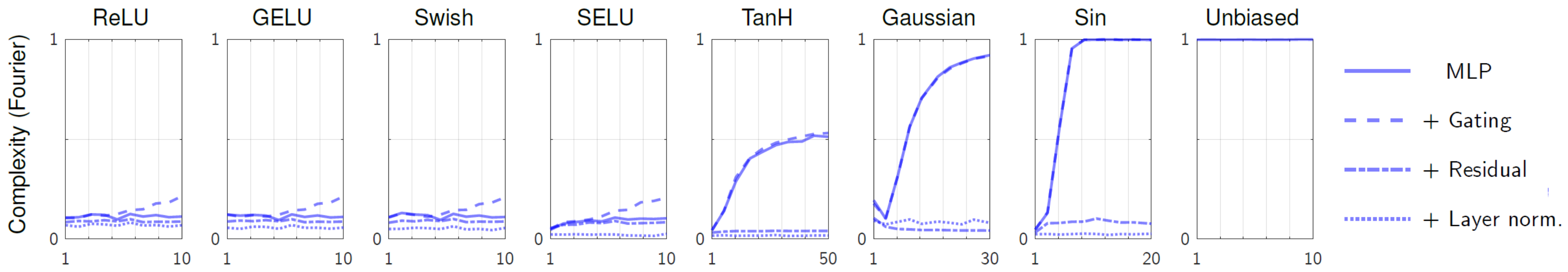


Tanh
Gaussian
Sin
Low frequency
High frequency
Weights magnitude
Depth

The strong **simplicity bias** is unique to ReLU-like activations

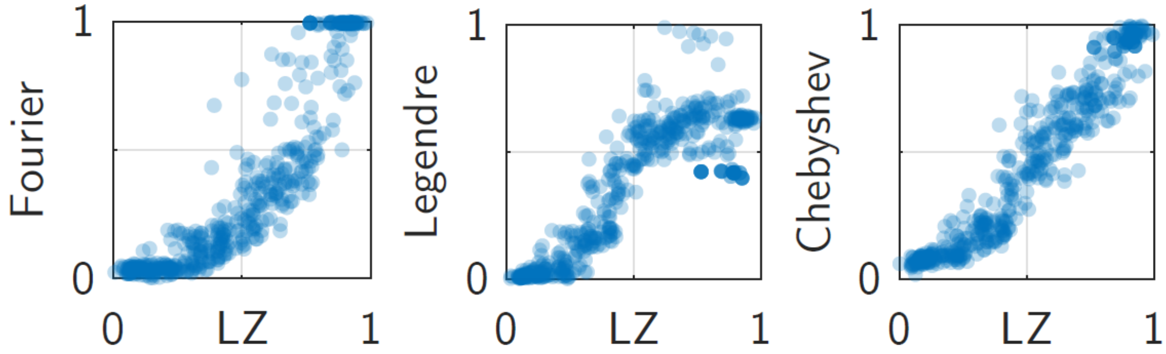


Impact of other components



Lower complexity	No impact	**Higher** complexity
ReLU-like activations	Width	Other activations
Layer normalization	Bias magnitudes	Depth
Residual connections		Multiplicative interactions

Different complexity measures are correlated



Despite measuring different proxies of complexity:

- frequency (Fourier)
- polynomial order (Legendre, Chebyshev)
- compressibility (LZ)

Is this relevant after training?

📝 We correlated complexity at initialization with generalization in **trained models**.

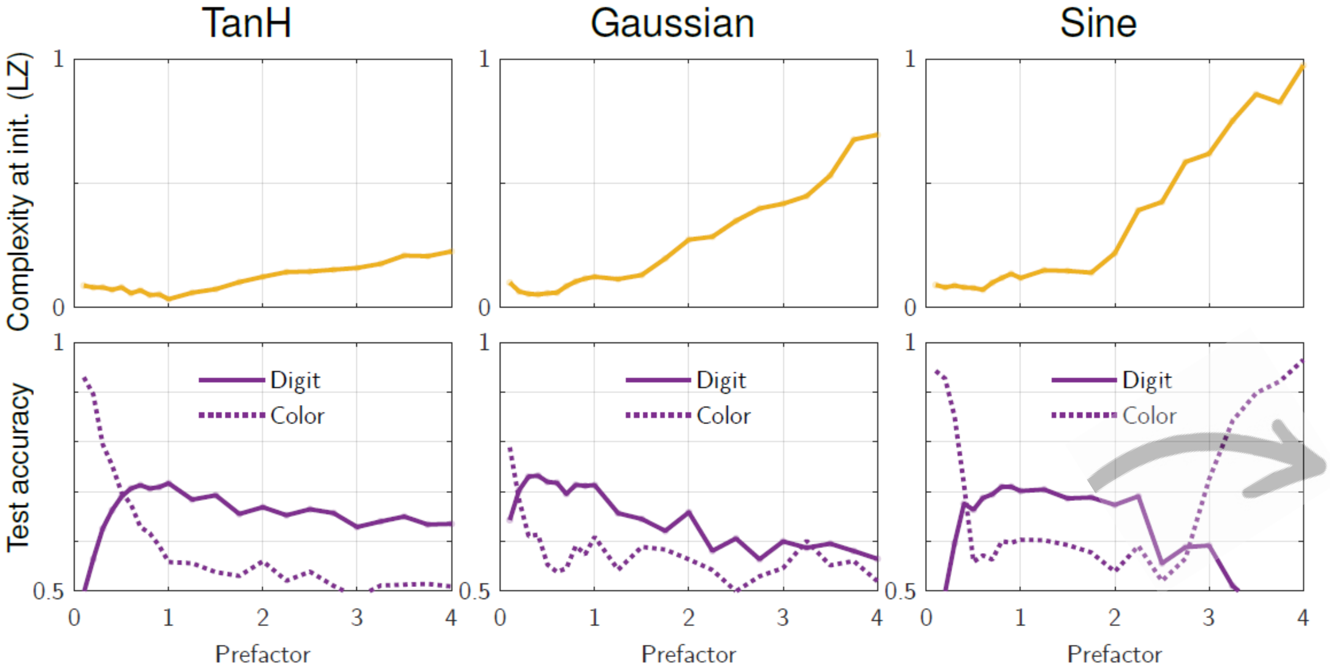
📈 **Generalization** occurs when the architecture's preferred complexity matches the target function's.

❗ In some cases, a bias towards **higher complexity is desirable**.

For example: learning INRs, parity function, avoiding shortcut learning.

Mitigating shortcut learning (Colored-MNIST)

We tweak the preferred complexity with a fixed prefactor before the activations.

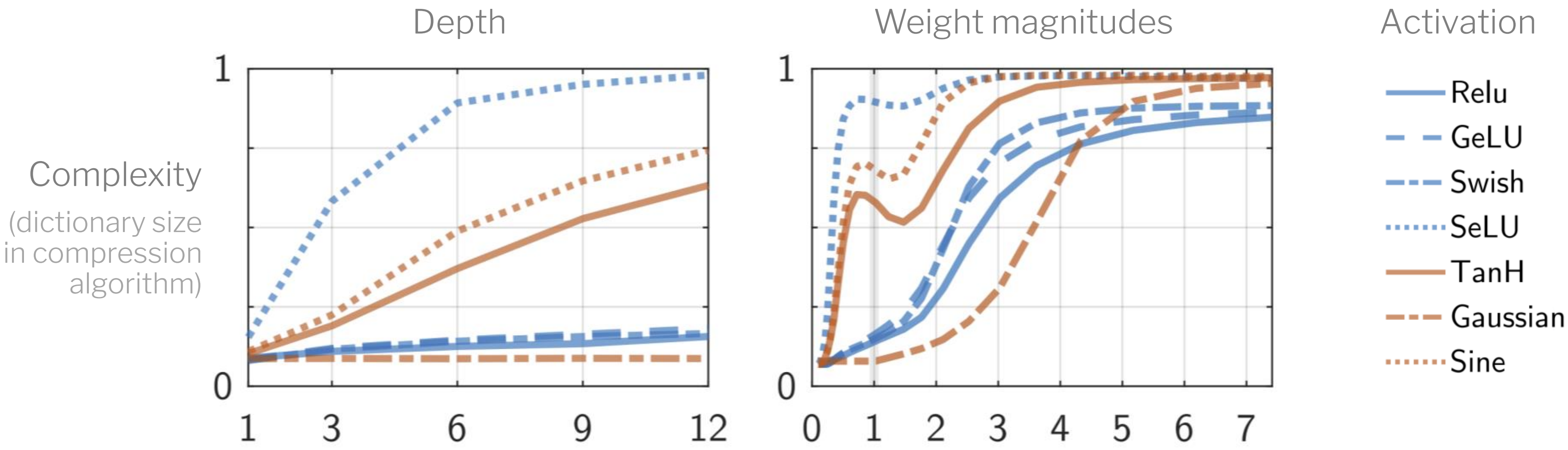


Sweet spot to learn the digit (task-specific!)

Transformers are biased towards compressible sequences

We greedily sample sequences from an **untrained** GPT-2 architecture.

Similar interventions cause an increase in sequence complexity.



Transformers seem to **inherit inductive biases from their building blocks** via

via mechanisms similar to those in simple models.

Take-aways

⭐ Fresh explanations for the **success of deep learning** independent from gradient-based training.

🔍 The '**simplicity bias**' is not a universal property of all architectures,
it can be explained without gradient descent
but is not always desirable.
Can cause shortcut learning, prevent learning complex patterns, ...

💡 The findings suggest possibilities for nudging inductive biases
and controlling the functions implemented by trained models.
E.g. via reparameterization, learning activations, ...