## COLLABORATION

<table>
<thead>
<tr>
<th>Jascha Sohl-Dickstein</th>
<th>Sam Schoenholz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surya Ganguli</td>
<td>Greg Yang</td>
</tr>
<tr>
<td>Lechao Xiao</td>
<td>Yasaman Bahri</td>
</tr>
<tr>
<td>Jaehoon Lee</td>
<td>Roman Novak</td>
</tr>
<tr>
<td>Minmin Chen</td>
<td>Dar Gilboa</td>
</tr>
<tr>
<td>Bo Chang</td>
<td></td>
</tr>
</tbody>
</table>
OUTLINE

1. Motivation
2. Functional priors
3. Signal propagation
4. Dynamical isometry
5. Functional posteriors
6. Conclusion
OUTLINE

1. Motivation
2. Functional priors
3. Signal propagation
4. Dynamical isometry
5. Functional posteriors
6. Conclusion
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: STATISTICAL MECHANICS
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: STATISTICAL MECHANICS

Microscopic
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: STATISTICAL MECHANICS

Microscopic

\[ \{ p_i, x_i \} i=1 \ldots N \]
SIMPLICITY IN LARGE NUMBERS: STATISTICAL MECHANICS

Microscopic

\[ H = \frac{1}{2m} \sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} V(x_i) + \sum_{i<j} U(x_i - x_j) \]

\[ \{ p_i, x_i \}_{i=1}^{N} \]
SIMPPLICITY IN LARGE NUMBERS: STATISTICAL MECHANICS

H = \frac{1}{2m} \sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} V(x_i) + \sum_{i<j} U(x_i - x_j)

{p_i, x_i} \ i = 1...N

Microscopic \rightarrow N >> 1 \rightarrow \text{Macroscopic}
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: STATISTICAL MECHANICS

Microscopic \[ \{ p_i, x_i \}_{i=1}^N \]

Macroscopic \[ \{ P, V, T \} \]

\[ H = \frac{1}{2m} \sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} V(x_i) + \sum_{i<j} U(x_i - x_j) \]
SIMPPLICITY IN LARGE NUMBERS: STATISTICAL MECHANICS

Microscopic

\[ \{ p_i, x_i \}_{i=1}^{N} \]

\[ H = \frac{1}{2m} \sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} V(x_i) + \sum_{i<j} U(x_i-x_j) \]

Macroscopic

\[ \{ P, V, T \} \]

\[ PV = nRT \]

\( N \gg 1 \)
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: NEURAL NETWORKS
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: NEURAL NETWORKS

Microscopic
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: NEURAL NETWORKS

Microscopic

\[ \Theta = \{ W_{i,j}^l, b_i^l \}^{l=1...L}_{i,j=1...N} \]
SIMPLICITY IN LARGE NUMBERS: NEURAL NETWORKS

Microscopic

$$\Theta = \{W^l_{ij}, b^l_i\}_{i,j=1,...,N}^{l=1,...,L}$$

$$\partial_t \Theta = -\nabla_\Theta \mathcal{L}$$
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: NEURAL NETWORKS

\[ \Theta = \{ W^l_{ij}, b^l_i \}_{i,j=1}^{L} \]

\[ \partial_t \Theta = -\nabla_\Theta \mathcal{L} \]
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: NEURAL NETWORKS

\( \Theta = \{ W_{ij}^l, b_i^l \}_{i,j=1...N}^{l=1...L} \)

\( \partial_t \Theta = - \nabla_{\Theta} \mathcal{L} \)
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: NEURAL NETWORKS

\[ \Theta = \{ W^l_{ij}, b^l_i \}_{i,j=1}^{L} \]

\[ \partial_t \Theta = -\nabla_\Theta \mathcal{L} \]

\[ \Sigma^l_{ab} = \frac{1}{N} \sum_{i=1}^{N} x^l_{ia} x^l_{ib} \]

?
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

SIMPLICITY IN LARGE NUMBERS: NEURAL NETWORKS

Microscopic

Macroscopic

\[ \Theta = \{ W^l_{ij}, b^l \}_{i,j=1}^{l=1...L} \]

\[ \partial_t \Theta = -\nabla_{\Theta} \mathcal{L} \]

\[ \Sigma^l_{ab} = \frac{1}{N} \sum_{i=1}^{N} x^l_{ia} x^l_{ib} \quad (?) \]

\[ \partial_t \Sigma^l_{ab} \approx 0 \quad (?) \]
OVERPARAMETERIZED MODELS PERFORM BETTER

MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

The novelty of this work can be summarized as follows:

- **Generalization** through a sequence of experiments of increasing level of nuance:
  - We first define sensitivity metrics for fully-connected neural networks in tasks where such a prior may not be justified (e.g. weather forecasting).
  - Care should be taken when extrapolating our findings to classification functions that favor robustness (see observed correlation with generalization as an expression of a universal prior on (natural) image models).

See (left). This observation rules out underfitting as the reason for poor generalization in low-capacity models. See (right).

Figure 1: Published as a conference paper at ICLR 2018

1.2 S

2 puts our work in context of related research studying complexity, generalization, or sensitivity.

Evaluation of sensitivity metrics on trained neural networks in a very large-scale experiment.

Generalization gap -axis) has training loss many orders of magnitude higher than models that generalize worse.

$0.4 \ 0.6 \ 0.8$

Number of weights

Generalization gap

$0.4 \ 0.6 \ 0.8$

Accuracy

$0.65 \ 0.7 \ 0.75 \ 0.8 \ 0.85$

Number of weights

Number of weights

Accuracy

$0.65 \ 0.7 \ 0.75 \ 0.8 \ 0.85$

$6 \ 7 \ 8 \ 9$

#Channels

$6 \ 7 \ 8 \ 9$

#Channels
MOTIVATION: WHY STUDY OVERPARAMETERIZED MODELS?

OVERPARAMETERIZED MODELS PERFORM BETTER

![Graphs showing the performance of ResNet18 and two layer ReLU networks](image)

Figure 1: Over-parametrization phenomenon.

Left panel: Training pre-activation ResNet18 architecture of different sizes on CIFAR-10 dataset. We observe that even when the network is large enough to completely fit the training data, the test error continues to decrease for larger networks.

Middle panel: Training fully connected feedforward network with single hidden layer on CIFAR-10. We observe the same phenomena as the one observed in ResNet18 architecture.

Right panel: Unit capacity captures the complexity of a hidden unit and unit impact captures the impact of a hidden unit on the output of the network, and are important factors in our capacity bound (Theorem 1). We observe empirically that both unit capacity and unit impact shrink with a rate faster than $1/p_h$ where $h$ is the number of hidden units. Please see Supplementary Section A for experiments settings.

To study and analyze this phenomenon more carefully, we need to simplify the architecture making sure that the property of interest is preserved after the simplification. We therefore chose two layer ReLU networks since as shown in the left and middle panel of Figure 1, it exhibits the same behavior with over-parametrization as the more complex pre-activation ResNet18 architecture. In this paper we prove a tighter generalization bound (Theorem 2) for two layer ReLU networks. Our capacity bound, unlike existing bounds, correlates with the test error and decreases with the increasing number of hidden units. Our key insight is to characterize complexity at a unit level, and as we see in the right panel in Figure 1, these unit level measures shrink at a rate faster than $1/p_h$ for each hidden unit, decreasing the overall measure as the network size increases. When measured in terms of layer norms, our generalization bound depends on the Frobenius norm of the top layer and the Frobenius norm of the difference of the hidden layer weights with the initialization, which decreases with increasing network size (see Figure 2).

The closeness of learned weights to initialization in the over-parametrized setting can be understood by considering the limiting case as the number of hidden units go to infinity, as considered in Bengio et al. [5] and Bach [2]. In this extreme setting, just training the top layer of the network, which is a convex optimization problem for convex losses, will result in minimizing the training error, as the randomly initialized hidden layer has all possible features. Intuitively, the large number of hidden units here represent all possible features and hence the optimization problem involves just picking the right features that will minimize the training loss. This suggests that as we over-parametrize the networks, the optimization algorithms need to do less work in tuning the weights of the hidden units to find the right solution. Dziugaite and Roy [6] indeed have numerically evaluated a PAC-Bayes measure from the initialization used by the algorithms and state that the Euclidean distance to the initialization is smaller than the Frobenius norm of the parameters. Nagarajan and Kolter [18] also make a similar empirical observation on the significant role of initialization, and in fact prove an initialization dependent generalization bound for linear networks. However they do not prove a similar generalization bound for neural networks. Alternatively, Liang et al. [15] suggested a Fisher-Rao metric based complexity measure that correlates with generalization behavior in larger networks but they also prove the capacity bound only for linear networks.

Contributions:

Our contributions in this paper are as follows.

• We empirically investigate the role of over-parametrization in generalization of neural networks on 3 different datasets (MNIST, CIFAR10 and SVHN), and show that the existing complexity measures increase with the number of hidden units - hence do not explain the generalization behavior with over-parametrization.

• We prove tighter generalization bounds (Theorems 2 and 5) for two layer ReLU networks. Our proposed complexity measure actually decreases with the increasing number of hidden units, and can potentially explain the effect of over-parametrization on generalization of neural networks.

• We provide a matching lower bound for the Rademacher complexity of two layer ReLU networks. Our lower bound considerably improves over the best known bound given in Bartlett et al. [4], and to our knowledge is the first such lower bound that is bigger than the Lipschitz of the network class.
1. Motivation
2. Functional priors
3. Signal propagation
4. Dynamical isometry
5. Functional posteriors
6. Conclusion
THE SINGLE HIDDEN LAYER CASE

Fully connected, single hidden layer [Radford Neal ’94]

Inputs: \( \mathbf{x}_a \in \mathbb{R}^{N_0} \) with input index \( a \)

Parameters: \( W^l_{ij} \in \mathbb{R}^{N_{l-1} \times N_l} \quad b^l_i \in \mathbb{R}^{N_l} \)

Prior: \( W^l_{ij} \sim \mathcal{N}(0, \sigma_w^2 / N_{l-1}) \quad b^l_i \sim \mathcal{N}(0, \sigma_b^2) \)

Network:

\[
\begin{align*}
    z^1_{ia} &= \sum_j W^1_{ij} x_{ja} + b^1_i \\
    y_{ia} &= \sum_j W^2_{ij} \phi(z^1_{ja}) + b^2_i
\end{align*}
\]
THE SINGLE HIDDEN LAYER CASE

Fully connected, single hidden layer [Radford Neal ‘94]

Inputs: \( \mathbf{x}_a \in \mathbb{R}^{N_0} \) with input index \( a \)

Parameters: \( W^l_{ij} \in \mathbb{R}^{N_{l-1} \times N_l} \) \( b^l_i \in \mathbb{R}^{N_l} \)

Prior: \( W^l_{ij} \sim \mathcal{N}(0, \sigma^2_w / N_{l-1}) \) \( b^l_i \sim \mathcal{N}(0, \sigma^2_b) \)

Network:

Weighted sum of Gaussians

\[
\begin{align*}
  z^1_{ia} &= \sum_j W^1_{ij} x_{ja} + b^1_i \\
  y_{ia} &= \sum_j W^2_{ij} \phi(z^1_{ja}) + b^2_i
\end{align*}
\]

\[
(z^1_{ia}, z^1_{ib})^T \sim \mathcal{N}(0, \Sigma^1_{ab} \delta_{i,j})
\]
**WIDE NEURAL NETWORKS ARE GAUSSIAN PROCESSES**

**THE SINGLE HIDDEN LAYER CASE**

Fully connected, single hidden layer [Radford Neal ’94]

Inputs: $\mathbf{x}_a \in \mathbb{R}^{N_0}$ with input index $a$

Parameters: $W^l_{ij} \in \mathbb{R}^{N_{l-1} \times N_l}$, $b^l_i \in \mathbb{R}^{N_l}$

Prior: $W^l_{ij} \sim \mathcal{N}(0, \sigma_w^2 / N_{l-1})$, $b^l_i \sim \mathcal{N}(0, \sigma_b^2)$

Network:

$z^1_{ia} = \sum_j W^1_{ij} x_{ja} + b^1_i$  

$y^2_{ia} = \sum_j W^2_{ij} \phi(z^1_{ja}) + b^2_i$

Weighted sum of Gaussians

Sum of i.i.d. random variables

\[ \Sigma^0_{ab} = \frac{1}{N_0} \sum_i x_{ia} x_{ib} \]

\[ \Sigma_{ab}^{1} = \frac{1}{N_0} \sum_i x_{ia} x_{ib} \]

\[ \Sigma_{ab}^{2} = \frac{1}{N_0} \sum_i x_{ia} x_{ib} \]

\[ (z^1_{ia}, z^1_{jb})^T \sim \mathcal{N}(0, \Sigma^1_{ab} \delta_{ij}) \]

\[ (y^2_{ia}, y^2_{jb})^T \xrightarrow{N_1 \to \infty} \mathcal{N}(0, \Sigma^2_{ab} \delta_{ij}) \]
THE SINGLE HIDDEN LAYER CASE

Infinitely wide neural networks are Gaussian Processes

\[
\begin{align*}
  z_{ia}^1 &= \sum_j W_{ij}^1 x_{ja} + b_i^1 \\
  y_{ia} &= \sum_j W_{ij}^2 \phi(z_{ja}^1) + b_i^2
\end{align*}
\]

\[
\begin{align*}
  (z_{ia}^1, z_{jb}^1)^T &\sim \mathcal{N}(0, \Sigma_{ab}^1 \delta_{ij}) \\
  (y_{ia}, y_{jb})^T &\xrightarrow{N_1 \to \infty} \mathcal{N}(0, \Sigma_{ab}^2 \delta_{ij})
\end{align*}
\]

Completely defined by a compositional kernel
WIDE NEURAL NETWORKS ARE GAUSSIAN PROCESSES

THE SINGLE HIDDEN LAYER CASE

Infinitely wide neural networks are Gaussian Processes

\[ z_{ia} = \sum_j W_{ij}^1 x_{ja} + b_i^1 \]
\[ (z_{ia}^1, z_{ib}^1)^T \sim N(0, \Sigma_{ab}^1 \delta_{ij}) \]

\[ y_{ia} = \sum_j W_{ij}^2 \phi(z_{ja}^1) + b_i^2 \]
\[ (y_{ia}, y_{jb})^T \xrightarrow{N_1 \to \infty} N(0, \Sigma_{ab}^2 \delta_{ij}) \]

Completely defined by a compositional kernel

\[ \Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2 \]

\[ \Sigma^2 = \sigma_w^2 \mathbb{E}_{z \sim N(0, \Sigma^1)} [\phi(z) \phi(z)^T] + \sigma_b^2 \]

Significant simplification
THE SINGLE HIDDEN LAYER CASE

Infinitely wide neural networks are Gaussian Processes

\[ z_{ia} = \sum_j W_{ij}^1 x_{ja} + b_i^1 \]

\[ y_{ia} = \sum_j W_{ij}^2 \phi(z_{ja}^1) + b_i^2 \]

\[ (z_{ia}^1, z_{jb}^1)^T \sim \mathcal{N}(0, \Sigma_{ab}^1 \delta_{ij}) \]

\[ (y_{ia}, y_{jb})^T \xrightarrow{N_1 \to \infty} \mathcal{N}(0, \Sigma_{ab}^2 \delta_{ij}) \]

Completely defined by a compositional kernel

\[ \Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2 \]

\[ \Sigma^2 = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^1)} [\phi(z) \phi(z)^T] + \sigma_b^2 \]

Significant simplification

Draws from ReLU-GP
WIDE NEURAL NETWORKS ARE GAUSSIAN PROCESSES

DEEP NETWORKS

Extension to deep networks

$$z_{ia}^1 = \sum_j W_{ij}^1 x_{ja} + b_i^1 \quad \Rightarrow \quad \Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2$$
WIDE NEURAL NETWORKS ARE GAUSSIAN PROCESSES

DEEP NETWORKS

Extension to deep networks

\[ z_{ia}^1 = \sum_j W_{ij}^1 x_{ja} + b_{i}^1 \]

\[ z_{ia}^2 = \sum_j W_{ij}^2 \phi(z_{ja}^1) + b_{i}^2 \]

\[ \Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2 \]
WIDE NEURAL NETWORKS ARE GAUSSIAN PROCESSES

DEEP NETWORKS

Extension to deep networks

\[
\begin{align*}
    z_{ia}^1 &= \sum_j W_{ij}^1 x_{ja} + b_i^1 \\
    z_{ia}^2 &= \sum_j W_{ij}^2 \phi(z_{ja}^1) + b_i^2
\end{align*}
\]

\[
\begin{align*}
    \Sigma^1 &= \sigma_w^2 \Sigma^0 + \sigma_b^2 \\
    \Sigma^2 &= \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^1)} [\phi(z)\phi(z)^T] + \sigma_b^2
\end{align*}
\]

AD, RF, YS (’16)
BP, SL, MR, JSD, SG (’16)
SSS, JG, SG, JSD (’17)
JL, YB et al. (’18)
RN, XLC et al. (’18)
WIDE NEURAL NETWORKS ARE GAUSSIAN PROCESSES

DEEP NETWORKS

Extension to deep networks

\[ z_{ia}^1 = \sum_j W_{ij}^1 x_{ja} + b_i^1 \]
\[ z_{ia}^2 = \sum_j W_{ij}^2 \phi(z_{ja}^1) + b_i^2 \]
\[ \Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2 \]
\[ \Sigma^2 = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^1)} [\phi(z)\phi(z)^T] + \sigma_b^2 \]
WIDE NEURAL NETWORKS ARE GAUSSIAN PROCESSES

DEEP NETWORKS

Extension to deep networks

\[ z_{i1} = \sum_{j} W_{ij}^1 x_{ja} + b_{i}^1 \]

\[ z_{i2}^2 = \sum_{j} W_{ij}^2 \phi(z_{ja}^1) + b_{i}^2 \]

\[ \ldots \]

\[ z_{i}^l = \sum_{j} W_{ij}^l \phi(z_{ja}^{l-1}) + b_{i}^l \]

\[ \Sigma^1 = \sigma^2_w \Sigma^0 + \sigma^2_b \]

\[ \Sigma^2 = \sigma^2_w \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^1)} [\phi(z) \phi(z)^T] + \sigma^2_b \]

\[ \ldots \]

\[ \Sigma^l = \sigma^2_w \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^{l-1})} [\phi(z) \phi(z)^T] + \sigma^2_b \]

AD, RF, YS ('16)
BP, SL, MR, JSD, SG ('16)
SSS, JG, SG, JSD ('17)
JL, YB et al. ('18)
RN, XLC et al. ('18)
WIDE NEURAL NETWORKS ARE GAUSSIAN PROCESSES

DEEP NETWORKS

Extension to deep networks

\[ z_{ia}^1 = \sum_j W_{ij}^1 x_{ja} + b_i^1 \]

\[ z_{ia}^2 = \sum_j W_{ij}^2 \phi(z_{ja}^1) + b_i^2 \]

\[ \vdots \]

\[ z_{ia}^l = \sum_j W_{ij}^l \phi(z_{ja}^{l-1}) + b_i^l \]

\[ \Sigma^1 = \sigma_w^2 \Sigma^0 + \sigma_b^2 \]

\[ \Sigma^2 = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^1)} [\phi(z)\phi(z)^T] + \sigma_b^2 \]

\[ \Sigma^l = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^{l-1})} [\phi(z)\phi(z)^T] + \sigma_b^2 \]

AD, RF, YS ('16)
BP, SL, MR, JSD, SG ('16)
SSS, JG, SG, JSD ('17)
JL, YB et al. ('18)
RN, XLC et al. ('18)
WIDE NEURAL NETWORKS ARE GAUSSIAN PROCESSES

DEEP NETWORKS

Extension to deep networks

\[ z_{1a}^{1} = \sum_{j} W_{ij}^{1} x_{ja} + b_{i}^{1} \]

\[ z_{2a}^{2} = \sum_{j} W_{ij}^{2} \phi(z_{ja}^{1}) + b_{i}^{2} \]

\[ \vdots \]

\[ z_{ia}^{l} = \sum_{j} W_{ij}^{l} \phi(z_{ja}^{l-1}) + b_{i}^{l} \]

Neural network induces \textit{dynamical system} over kernels

\[ C \rightarrow \Sigma^{1} = \sigma_{w}^{2} \Sigma^{0} + \sigma_{b}^{2} \]

\[ C \rightarrow \Sigma^{2} = \sigma_{w}^{2} \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^{1})} [\phi(z)\phi(z)^{T}] + \sigma_{b}^{2} \]

\[ \vdots \]

\[ C \rightarrow \Sigma^{l} = \sigma_{w}^{2} \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^{l-1})} [\phi(z)\phi(z)^{T}] + \sigma_{b}^{2} \]

Understanding prior equivalent to studying dynamics
1. Motivation
2. Functional priors
3. Signal propagation
4. Dynamical isometry
5. Functional posteriors
6. Conclusion
FORWARD SIGNAL PROPAGATION

DYNAMICS OF SIGNAL PROPAGATION

\[ \sum^1 \xrightarrow{C} \sum^2 \xrightarrow{C} \ldots \xrightarrow{C} \sum^l \xrightarrow{C} \ldots \]
Dynamics converge to **universal** fixed point

- Independent of inputs $\Rightarrow$ pathological
DYNAMICS OF SIGNAL PROPAGATION

$\Sigma^1 \xrightarrow{C} \Sigma^2 \xrightarrow{C} \ldots \xrightarrow{C} \Sigma^l \xrightarrow{C} \ldots \xrightarrow{C} \Sigma^*$

Dynamics converge to **universal** fixed point

- Independent of inputs $\Rightarrow$ pathological

Rate of convergence determined by behavior near fixed point
**FORWARD SIGNAL PROPAGATION**

**DYNAMICS OF SIGNAL PROPAGATION**

\[ \Sigma^1 \xrightarrow{C} \Sigma^2 \xrightarrow{C} \ldots \xrightarrow{C} \Sigma^l \xrightarrow{C} \ldots \xrightarrow{C} \Sigma^* \]

Dynamics converge to **universal** fixed point

- Independent of inputs \( \Rightarrow \) pathological

Rate of convergence determined by behavior near fixed point

\[ \epsilon^l = \Sigma^* - \Sigma^l \quad \Rightarrow \quad \epsilon^{l+1} = \left. \frac{\partial C(\Sigma)}{\partial \Sigma} \right|_{\Sigma^*} \epsilon^l \]
Dynamics converge to universal fixed point

- Independent of inputs ⇒ pathological

Rate of convergence determined by behavior near fixed point

\[ \epsilon^l = \Sigma^* - \Sigma^l \quad \Rightarrow \quad \epsilon^{l+1} = \left. \frac{\partial C(\Sigma)}{\partial \Sigma} \right|_{\Sigma^*} \epsilon^l \]

\[ \lambda_{\text{max}} > 1 \quad \text{Unstable fixed point} \]

\[ \lambda_{\text{max}} \leq 1 \quad \text{Stable fixed point} \]
DYNAMICS OF SIGNAL PROPAGATION

\[ \sum^1 \xrightarrow{C} \sum^2 \xrightarrow{C} \ldots \xrightarrow{C} \sum^l \xrightarrow{C} \ldots \xrightarrow{C} \sum^* \]

Dynamics converge to **universal** fixed point

- Independent of inputs \( \Rightarrow \) pathological

Rate of convergence determined by behavior near fixed point

\[ \varepsilon^l = \sum^* - \sum^l \quad \Rightarrow \quad \varepsilon^{l+1} \approx \lambda_{\text{max}}^l \]

\[ \lambda_{\text{max}} > 1 \quad \text{Unstable fixed point} \]

\[ \lambda_{\text{max}} \leq 1 \quad \text{Stable fixed point} \]
Dynamics converge to **universal** fixed point

- Independent of inputs $\Rightarrow$ pathological

Rate of convergence determined by behavior near fixed point

Exponential convergence

$$\epsilon^l \approx e^{-l/\xi}$$

with rate

$$\xi = -1 / \log \lambda_{\text{max}}$$

$$\epsilon^{l+1} \approx \lambda^l_{\text{max}}$$

$\iff$

- $\lambda_{\text{max}} > 1$ Unstable fixed point
- $\lambda_{\text{max}} \leq 1$ Stable fixed point
Dynamics converge to universal fixed point

- Independent of inputs \( \Rightarrow \) pathological

Rate of convergence determined by behavior near fixed point

**Exponential convergence**

\[ \epsilon^l \approx e^{-l/\xi} \]

with rate

\[ \xi = -1 / \log \lambda_{\text{max}} \]

\[ \epsilon^{l+1} \approx \lambda^l_{\text{max}} \]

\[ \lambda_{\text{max}} > 1 \quad \text{Unstable fixed point} \]

\[ \lambda_{\text{max}} \leq 1 \quad \text{Stable fixed point} \]

How can we adjust the hyperparameters to delay convergence?
The fixed point satisfies

\[ \Sigma^* = \mathcal{C}(\Sigma^*) = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^*)} [\phi(z)\phi(z)^\top] + \sigma_b^2 \]
The fixed point satisfies

\[ \Sigma^* = \mathcal{C}(\Sigma^*) = \sigma^2_w \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^*)} [\phi(z) \phi(z)^\top] + \sigma^2_b \]

One solution is perfect correlation,

\[ \Sigma^* = q^* \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \]
The fixed point satisfies

\[ \Sigma^* = \mathcal{C}(\Sigma^*) = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^*)} [\phi(z) \phi(z)^\top] + \sigma_b^2 \]

One solution is perfect correlation,

\[ \Sigma^* = q^* \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \]

Is this fixed point stable?
The fixed point satisfies

$$\Sigma^* = C(\Sigma^*) = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^*)} [\phi(z)\phi(z)^\top] + \sigma_b^2$$

One solution is perfect correlation,

$$\Sigma^* = q^* \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Is this fixed point stable?

- Depends on hyperparameters
The fixed point satisfies

$$\Sigma^* = C(\Sigma^*) = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^*)} [\phi(z)\phi(z)\top] + \sigma_b^2$$

One solution is perfect correlation,

$$\Sigma^* = q^* \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Is this fixed point stable?

- Depends on hyperparameters
The fixed point satisfies

$$
\Sigma^* = C(\Sigma^*) = \sigma_w^2 \mathbb{E}_{z \sim \mathcal{N}(0, \Sigma^*)} [\phi(z)\phi(z)^\top] + \sigma_b^2
$$

One solution is perfect correlation,

$$
\Sigma^* = q^* \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}
$$

Is this fixed point stable?

- Depends on hyperparameters

\[ \lambda_{\text{max}} = 1, \quad \xi = \infty \]
For **deep** signal propagation, initialize on the “edge of chaos”
THE EDGE OF CHAOS

For deep signal propagation, initialize on the “edge of chaos”

- Analyze the fixed point and set $\lambda_{\text{max}} = 1$
The Edge of Chaos

For **deep** signal propagation, initialize on the "edge of chaos"

- Analyze the fixed point and set $\lambda_{\text{max}} = 1$

\[
\Sigma^* = q^* \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}
\]

\[
\lambda_{\text{max}} \left( \frac{\partial C(\Sigma)}{\partial \Sigma} \bigg|_{\Sigma^*} \right) = \chi(\sigma_w, \sigma_b)
\]
THE EDGE OF CHAOS

For **deep** signal propagation, initialize on the “edge of chaos”
• Analyze the fixed point and set \( \lambda_{\text{max}} = 1 \)

\[
\Sigma^* = q^* \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}
\]

\[
\lambda_{\text{max}} \left( \frac{\partial C(\Sigma)}{\partial \Sigma} \bigg|_{\Sigma^*} \right) = \chi(\sigma_w, \sigma_b)
\]

\[
\chi(\sigma_w, \sigma_b) = \sigma_w^2 \int dz \frac{e^{-\frac{1}{2}z^2}}{\sqrt{2\pi}} \phi'(\sqrt{q^*}z)^2
\]
Given a loss $\mathcal{L}$, back-propagation gives

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^l} = \delta_i^l \phi(z_j^{l-1})$$

$$\delta_i^l = \frac{\partial \mathcal{L}}{\partial z_i^l}$$

$$\delta_i^l = \phi'(z_i^l) \sum_j \delta_j^{l+1} W_{ji}^{l+1}$$
Given a loss $\mathcal{L}$, back-propagation gives

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^l} = \delta_i^l \phi(z_{ji}^{l-1}) \quad \delta_i^l = \frac{\partial \mathcal{L}}{\partial z_i^l}$$

Backpropagated gradients scale like

$$\mathbb{E}[(\delta_i^l)^2] = \mathbb{E}[(\delta_{i}^{l+1})^2] \sigma_w^2 \mathbb{E}[\phi'(z_{i}^{l+1})^2]$$

Gradients are propagated backwards through the network, with each layer's gradient contributing to the loss at the previous layer.
BACKPROPAGATED GRADIENTS

Given a loss $\mathcal{L}$, back-propagation gives

$$
\frac{\partial \mathcal{L}}{\partial W_{ij}^l} = \delta_i^l \phi(z_{ji}^{l-1}) \quad \delta_i^l = \frac{\partial \mathcal{L}}{\partial z_i^l} \quad \delta_i^l = \phi'(z_i^l) \sum_j \delta_j^{l+1} W_{ji}^{l+1}
$$

Gradients scale like

$$
\mathbb{E}[(\delta_1^l)^2] = \mathbb{E}[(\delta_1^{l+1})^2] \sigma_w^2 \mathbb{E}[\phi'(z_1^{l+1})^2] \chi(\sigma_w, \sigma_b)
$$

$$
\mathbb{E}[(\delta_1^1)^2] = \mathbb{E}[(\delta_1^{L})^2] \chi(\sigma_w, \sigma_b)^L
$$
Given a loss $\mathcal{L}$, back-propagation gives

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^l} = \delta_i^l \phi(z_{ji}^{l-1}) \quad \delta_i^l = \frac{\partial \mathcal{L}}{\partial z_i^l}$$

**Backpropagated Gradients**

Gradients scale like

$$\mathbb{E}[(\delta_1^l)^2] = \mathbb{E}[(\delta_1^{l+1})^2] \sigma_w^2 \mathbb{E}[\phi'(z_1^{l+1})^2]$$

$$\mathbb{E}[(\delta_1^L)^2] = \mathbb{E}[(\delta_1^L)^2] \chi(\sigma_w, \sigma_b)^L$$

Gradients explode/vanish unless

$$\chi(\sigma_w, \sigma_b) = 1$$
Given a loss $\mathcal{L}$, back-propagation gives

$$
\frac{\partial \mathcal{L}}{\partial W_{ij}^l} = \delta_i^l \phi(z_j^{l-1})
$$

Gradients scale like

$$
\mathbb{E}[(\delta_1^l)^2] = \mathbb{E}[(\delta_1^{l+1})^2] \sigma_w^2 \mathbb{E}[\phi'(z_1^{l+1})^2] \\
\chi(\sigma_w, \sigma_b)
$$

Gradients explode/vanish unless

$$
\chi(\sigma_w, \sigma_b) = 1
$$
Critical initialization:

In order for signals to propagate forward and backward through a deep network, the initialization hyperparameters should lie on the critical line.
CRITICAL INITIALIZATION

PREDICTING TRAINABLE DEPTH

\[ \epsilon^l \approx e^{-l/\xi} \]

\[ \xi(\sigma_w) = -1/\log \lambda_{\text{max}}(\sigma_w) \]

Graph showing the training accuracy of MNIST as a function of depth and \( \sigma^2_w \), with a phase transition indicating ordered and chaotic regions.
The stability of the fixed point is determined by whether the quantities \( e \) and \( b \) are met, i.e. \( e < 0 \) and \( b > 0 \). The set of eigenvalues is non-negative and sum to one, their Fourier convolutional neural networks decouple into independently-evolving Fourier modes that evolve near the fixed point at zero-frequency coefficient is equal to one; see Figure. Since the modes of eqn. (2.11) describes a multi-dimensional covariance iteration map (see also the SM). To further elucidate that for \( d = o.d. \), then eqn. (2.9) accurately describes the linearized mean field theory of convolutional neural networks. To see this, note eqn. (2.10) shows that this is also the same as in the fully-connected case. Together, we expect signals corresponding to a specific Fourier mode to govern fully-connected networks (Poole et al. 2016), we can immediately conclude that the convergence to the fixed point.

We now assume that the conditions for a stable fixed point are met, i.e. \( c \) and \( o.d. \) connected case. In particular, based on the results of (Schoenholz et al. 2016), it is convenient to additionally assume that for \( c \) in which all pixels approach zero, the rate of which means it also belongs to the set of eigenvalues is a specific structure. In particular, the set of eigenvalues is non-negative and sum to one, their Fourier convolutional neural networks decouple into independently-evolving Fourier modes that evolve near the fixed point at zero-frequency coefficient is equal to one; see Figure. Since the modes of eqn. (2.11) describes a multi-dimensional covariance iteration map (see also the SM). To further elucidate that for \( d = o.d. \), then eqn. (2.9) accurately describes the linearized mean field theory of convolutional neural networks. To see this, note eqn. (2.10) shows that this is also the same as in the fully-connected case. Together, we expect signals corresponding to a specific Fourier mode to govern fully-connected networks (Poole et al. 2016), we can immediately conclude that the convergence to the fixed point.

We now assume that the conditions for a stable fixed point are met, i.e. \( c \) and \( o.d. \) connected case. In particular, based on the results of (Schoenholz et al. 2016), it is convenient to additionally assume that for \( c \) in which all pixels approach zero, the rate of which means it also belongs to the set of eigenvalues is a specific structure. In particular, the set of eigenvalues is non-negative and sum to one, their Fourier convolutional neural networks decouple into independently-evolving Fourier modes that evolve near the fixed point at zero-frequency coefficient is equal to one; see Figure. Since the modes of eqn. (2.11) describes a multi-dimensional covariance iteration map (see also the SM). To further elucidate that for \( d = o.d. \), then eqn. (2.9) accurately describes the linearized mean field theory of convolutional neural networks. To see this, note eqn. (2.10) shows that this is also the same as in the fully-connected case. Together, we expect signals corresponding to a specific Fourier mode to govern fully-connected networks (Poole et al. 2016), we can immediately conclude that the convergence to the fixed point.

We now assume that the conditions for a stable fixed point are met, i.e. \( c \) and \( o.d. \) connected case. In particular, based on the results of (Schoenholz et al. 2016), it is convenient to additionally assume that for \( c \) in which all pixels approach zero, the rate of which means it also belongs to the set of eigenvalues is a specific structure. In particular, the set of eigenvalues is non-negative and sum to one, their Fourier convolutional neural networks decouple into independently-evolving Fourier modes that evolve near the fixed point at zero-frequency coefficient is equal to one; see Figure. Since the modes of eqn. (2.11) describes a multi-dimensional covariance iteration map (see also the SM). To further elucidate that for \( d = o.d. \), then eqn. (2.9) accurately describes the linearized mean field theory of convolutional neural networks. To see this, note eqn. (2.10) shows that this is also the same as in the fully-connected case. Together, we expect signals corresponding to a specific Fourier mode to govern fully-connected networks (Poole et al. 2016), we can immediately conclude that the convergence to the fixed point.

We now assume that the conditions for a stable fixed point are met, i.e. \( c \) and \( o.d. \) connected case. In particular, based on the results of (Schoenholz et al. 2016), it is convenient to additionally assume that for \( c \) in which all pixels approach zero, the rate of which means it also belongs to the set of eigenvalues is a specific structure. In particular, the set of eigenvalues is non-negative and sum to one, their Fourier convolutional neural networks decouple into independently-evolving Fourier modes that evolve near the fixed point at zero-frequency coefficient is equal to one; see Figure. Since the modes of eqn. (2.11) describes a multi-dimensional covariance iteration map (see also the SM). To further elucidate that for \( d = o.d. \), then eqn. (2.9) accurately describes the linearized mean field theory of convolutional neural networks. To see this, note eqn. (2.10) shows that this is also the same as in the fully-connected case. Together, we expect signals corresponding to a specific Fourier mode to govern fully-connected networks (Poole et al. 2016), we can immediately conclude that the convergence to the fixed point.

We now assume that the conditions for a stable fixed point are met, i.e. \( c \) and \( o.d. \) connected case. In particular, based on the results of (Schoenholz et al. 2016), it is convenient to additionally assume that for \( c \) in which all pixels approach zero, the rate of which means it also belongs to the set of eigenvalues is a specific structure. In particular, the set of eigenvalues is non-negative and sum to one, their Fourier convolutional neural networks decouple into independently-evolving Fourier modes that evolve near the fixed point at zero-frequency coefficient is equal to one; see Figure. Since the modes of eqn. (2.11) describes a multi-dimensional covariance iteration map (see also the SM). To further elucidate that for \( d = o.d. \), then eqn. (2.9) accurately describes the linearized mean field theory of convolutional neural networks. To see this, note eqn. (2.10) shows that this is also the same as in the fully-connected case. Together, we expect signals corresponding to a specific Fourier mode to govern fully-connected networks (Poole et al. 2016), we can immediately conclude that the convergence to the fixed point.

We now assume that the conditions for a stable fixed point are met, i.e. \( c \) and \( o.d. \) connected case. In particular, based on the results of (Schoenholz et al. 2016), it is convenient to additionally assume that for \( c \) in which all pixels approach zero, the rate of which means it also belongs to the set of eigenvalues is a specific structure. In particular, the set of eigenvalues is non-negative and sum to one, their Fourier convolutional neural networks decouple into independently-evolving Fourier modes that evolve near the fixed point at zero-frequency coefficient is equal to one; see Figure. Since the modes of eqn. (2.11) describes a multi-dimensional covariance iteration map (see also the SM). To further elucidate that for \( d = o.d. \), then eqn. (2.9) accurately describes the linearized mean field theory of convolutional neural networks. To see this, note eqn. (2.10) shows that this is also the same as in the fully-connected case. Together, we expect signals corresponding to a specific Fourier mode to govern fully-connected networks (Poole et al. 2016), we can immediately conclude that the convergence to the fixed point.
1. Motivation
2. Functional priors
3. Signal propagation
4. Dynamical isometry
5. Functional posteriors
6. Conclusion
Study the \textbf{end-to-end Jacobian}

\[ J = \frac{\partial z^L}{\partial z^0} = \prod_l D_l^{W_l} \]

\[ D_{ij}^l = \phi'(z_i^l) \delta_{ij} \]
Study the end-to-end Jacobian

\[ J = \frac{\partial z^L}{\partial z^0} = \prod_l D^l W^l \]

Diagonal Matrix

\[ D_{ij}^l = \phi'(z_i^l) \delta_{ij} \]

A few relations that make this interesting

\[ \delta^0 = J \delta^L \]

Gradients

\[ f(x + \delta) \approx f(x) + J^T \delta \]

Linear Response

\[ G = J^T J \]

Induced Metric
Study the end-to-end Jacobian

\[ J = \frac{\partial z^L}{\partial z^0} = \prod_l D^l W^l \]

A few relations that make this interesting

\[ \delta^0 = J \delta^L \]

Gradients

\[ f(x + \delta) \approx f(x) + J^T \delta \]

Linear Response

\[ G = J^T J \]

Induced Metric

We have worked out behavior of gradients on average:

Criticality \[ \Leftrightarrow \mathbb{E}[\text{tr}(J^T J)] = \chi(\sigma_w, \sigma_b)^L = 1 \]
Study the end-to-end Jacobian

\[
J = \frac{\partial z^L}{\partial z^0} = \prod_l D^l W^l
\]

A few relations that make this interesting

\[
\delta^0 = J \delta^L
\]

Gradients

\[
f(x + \delta) \approx f(x) + J^T \delta
\]

Linear Response

\[
G = J^T J
\]

Induced Metric

We have worked out behavior of gradients on average:

\[
\text{Criticality} \iff \mathbb{E}[^{\text{tr}(J^T J)}] = \chi(\sigma_w, \sigma_b)^L = 1
\]

But what is a good prior for the whole spectrum?
DYNAMICAL ISOMETRY

Study the end-to-end Jacobian

\[ J = \frac{\partial z^L}{\partial z^0} = \prod_l D^l W^l \]

Diagonal Matrix

A few relations that make this interesting

\[ \delta^0 = J \delta^L \quad f(x + \delta) \approx f(x) + J^T \delta \quad G = J^T J \]

Gradients
Linear Response
Induced Metric

We have worked out behavior of gradients on average:

Criticality  \iff  \mathbb{E}[\text{tr}(J^T J)] = \chi(\sigma_w, \sigma_b)^L = 1

But what is a good prior for the whole spectrum?

- **Isometry**: all singular values \( \approx 1 \)
Using tools from random matrix theory (free probability), can compute spectrum analytically:
Using tools from random matrix theory (free probability), can compute spectrum analytically:

- **Gaussian W, any f**
- **Orthogonal W, ReLU**
- **Orthogonal W, tanh, $\sigma_w \gg 1$**
- **Orthogonal W, tanh, $\sigma_w \sim 1+1/L$**

*Computation of the spectrum can be achieved using techniques from random matrix theory.*
Not every point on the critical line is equally favorable for gradient propagation. For activation functions that are linear near the origin, dynamical isometry (i.e. well-conditioned Jacobians) can be achieved with small bias variance.
THE BENEFITS OF A BETTER PRIOR

4000-layer CNN on MNIST

- **Critical initialization**
4000-layer CNN on MNIST
1. Introduction

Deep convolutional neural networks (CNNs), have been crucial to the success of deep learning. Architectures based on CNNs have achieved unprecedented accuracy in domains such as computer vision and natural language processing.

In recent years, state-of-the-art methods in computer vision have utilized increasingly deep convolutional network architectures (CNNs), demonstrating empirically that they enable efficient training of extremely deep architectures.

A critical question is how to train extremely deep CNNs. In this work, we demonstrate that it is possible to train vanilla CNNs with ten thousand layers (bottom) for depths greater than one thousand layers (top).

We prove it is possible to train vanilla CNNs with 10,000 layers using no architectural tricks. Initial orthogonal convolution kernels and an appropriate initialization scheme can achieve dynamical isometry in the sense that it is norm-preserving.

The performance of deep convolutional networks has improved as these networks have been made ever deeper. For example, some of the best-performing models on ImageNet [2012] have employed hundreds or even thousands of layers. A variety of pathologies such as vanishing/exploding gradients make training such deep networks challenging. While residual connections and batch normalization or residual connections simply by using an appropriate initialization scheme, we can train vanilla CNNs with 10,000 layers using no architectural tricks.

The benefits of a better prior

THE BENEFITS OF A BETTER PRIOR

4000-layer CNN on MNIST

- Dynamical Isometry
- Critical initialization

MNIST

CIFAR-10

Extremely deep CNNs can be trained without the use of batch normalization or residual connections simply by using an appropriate initialization scheme.
OUTLINE

1. Motivation
2. Functional priors
3. Signal propagation
4. Dynamical isometry
5. Functional posteriors
6. Conclusion
WIDE, DEEP, NETWORKS EVOLVE AS LINEAR MODELS

Consider a FC neural network, $f(x; \theta(t))$
Consider a FC neural network, $f(x; \theta(t))$

As the width of the network grows, parameters move less during gradient descent.
WIDE, DEEP, NETWORKS EVOLVE AS LINEAR MODELS

Consider a FC neural network, $f(x; \theta(t))$

As the width of the network grows, parameters move less during gradient descent

Motivates a linear approximation,

$$f(x; \theta(t)) \approx f(x; \theta(0)) + \sum_{\alpha} \frac{\partial f(x; \theta(0))}{\partial \theta_{\alpha}(0)} (\theta(t) - \theta(0)) + O((\theta(t) - \theta(0))^2)$$

This becomes exact as $N \rightarrow \infty$
WIDE, DEEP, NETWORKS EVOLVE AS LINEAR MODELS

Consider a FC neural network, \( f(x; \theta(t)) \)

As the width of the network grows, parameters move less during gradient descent

Motivates a linear approximation,

\[
\begin{align*}
    f_t(x) & \approx f_0(x) + J_0(x) \omega(t) \\
    \omega(t) & = \theta(t) - \theta(0)
\end{align*}
\]

This becomes exact as \( N \rightarrow \infty \)
We consider two-class classification on CIFAR-10 (horses vs. cats) as in Novak et al. (2018b) with full-batch gradient descent with dataset size \(|D|\) = 20,000. For simplicity, we consider random 15,000 training points. We observe that the empirical kernel dynamics for finite width networks. Moreover, as the neural network trains the change in the loss and accuracy for training and test points agree well between original and linearized model. Bottom right pane shows the dynamics for finite width networks.

In the case of a MSE loss, the output distribution remains invariant with the number of Monte Carlo samples. For both NNGP and tangent kernels we observe exact and experimental dynamics are nearly identical. The red region indicates the analytic prediction of the output distribution from an ensemble of 100 trained neural networks (NNs). Black lines indicate the time evolution of the predictive output distribution throughout training. For simplicity, we consider random 15,000 training points. We observe that empirical kernel dynamics for individual draw of functions, see SM Figure S15.

In Figure 4, we show that the output distribution remains invariant with the number of Monte Carlo samples.

Fully Connected, N=2048, Single Output, MSE Loss, Gradient Descent

Wide, deep, networks evolve as linear models
WIDE, DEEP, NETWORKS EVOLVE AS LINEAR MODELS

Fully Connected, N=1024, 10-Class, Cross Entropy Loss, Momentum

![Graphs showing training output, test class 3 logit, loss, and accuracy over time.](image)
WIDE, DEEP, NETWORKS EVOLVE AS LINEAR MODELS

CNN, C=256, 2-Class, MSE Loss, GD

Figure 7.

We have investigated gradient descent dynamics for related problems.

Figure 8.

Thanks to...

ACKNOWLEDGEMENTS

GROUP NAME OUTPUT SIZE BLOCK TYPE

AVG CONV CONV CONV CONV

-POOL -POOL -POOL -POOL -POOL

<table>
<thead>
<tr>
<th>CHANNEL SIZE</th>
<th>CHANNEL SIZE</th>
<th>CHANNEL SIZE</th>
<th>CHANNEL SIZE</th>
<th>CHANNEL SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

Novak 2016

Sensitivity analysis was performed on the CIFAR-10 datasets.

The number of realistic neural networks may be operating in the limits we studied.

Empirical investigation revealed that this agrees very accurately described by a linearized model in the infinite width regime.
WIDE, DEEP, NETWORKS EVOLVE AS LINEAR MODELS

Wide Resnet (10-layers), C=1024, 2-Class, Cross Entropy Loss, Momentum

Figure 8. Train output and Test output for a Wide Resnet (10-layers) with 2-Class, Cross-entropy Loss, Momentum. The neural network and the linearized model are compared. The loss and accuracy curves for training and testing points are shown. The linear model closely follows the neural network, indicating that wide, deep, networks evolve as linear models under gradient descent.
**IMPLICATIONS FOR THE POSTERIOR**

For MSE Loss,

\[
\partial_t f_t(X) = -\Theta(X, X)(f_t(X) - Y) \quad \Theta(X, Y) = \frac{1}{M} J_0(X) J_0(Y)^T
\]

Neural Tangent Kernel
IMPLICATIONS FOR THE POSTERIOR

For MSE Loss,

\[ \partial_t f_t(X) = -\Theta(X, X)(f_t(X) - Y) \]
\[ \Theta(X, Y) = \frac{1}{M} J_0(X) J_0(Y)^T \]

This allows us to compute the "posterior" after \( t \) steps of GD,

\[ \mu(x) = \Theta(x, X) \Theta^{-1}(I - e^{-\eta \Theta t}) Y \]
\[ \Sigma(x) = K(x, x) - 2\Theta(x, X) \Theta^{-1}(I - e^{-\eta \Theta t}) K(x, X)^T \]
\[ + \Theta(x, X) \Theta^{-1}(I - e^{-\eta \Theta t}) \Theta(x, X)^T \]

Neural Tangent Kernel

NNGP Kernel
IMPLICATIONS FOR THE POSTERIOR

FC Network, N=8192, MNIST, MSE Loss

Figure 3. We consider two-class classification on CIFAR-10 (horses vs. cars) as in Novak et al. (2018b) with full-batch gradient descent with dataset size $|D|=128$ and 64 inputs drawn from the mean denoted in solid lines. Training was performed for inputs interpolated between two training points (denoted with black dots). The blue region indicates the analytic prediction of the output distribution from an ensemble of 100 trained neural networks (NNs). The output is computed for inputs interpolated between two training points (denoted with black dots) and the other to be zero.

Figure 4. Exact and experimental dynamics are nearly identical for network outputs and are similar for individual weights and planes.

Wide neural networks of any depth evolve as linear models under gradient descent for network outputs, and are similar for individual weights and planes. We observe that empirical kernel distributions capture the true distribution throughout training. In Figure 3, we provide empirical support showing that the training dynamics of wide neural networks are well captured by linearized models. We consider fully-connected, convolutional, and wide ResNet architectures trained with small so that the continuous time approximation holds well. Moreover, as the neural network trains the change in the input distribution from an ensemble of 100 trained neural networks (NNs). The interpolation is given by $\alpha(x) = \frac{1}{2} \tanh(x) + \frac{1}{2}$ for $x \in [-2, 2]$. In Figure 4, we show the dynamics of RMSE between the two models on test datapoints or parameters. First two panes in the second row show well between original and linearized model. Bottom right pane shows the dynamics of RMSE between the two models on test datapoints or parameters. First two panes in the second row show well between original and linearized model. Bottom right pane shows the dynamics of RMSE between the two models on test datapoints or parameters. First two panes in the second row show well between original and linearized model. Bottom right pane shows the dynamics of RMSE between the two models on test datapoints or parameters.
CONCLUSIONS

Overparameterized models are simple!

The prior over functions can be computed analytically

Properties of the prior are intimately related to trainability

Wide neural networks are almost linear models

Overall, a powerful framework is emerging for theoretically analyzing overparameterized neural networks