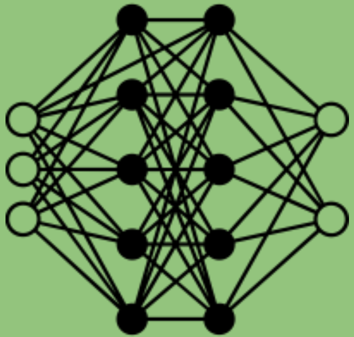
Gradient Descent in Wide, Deep Neural Networks

Yasaman Bahri

Caltech CS 159 Guest Lecture
May 13, 2021



Motivation

Why deep neural networks (DNNs) that are very wide?

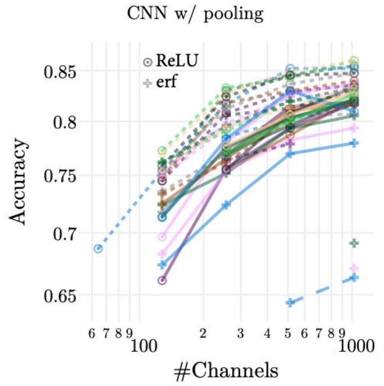
Theoretical:

- Dealing with infinitely many variables can sometimes simplify the problem.
  - For instance: the prior in function space for infinitely wide DNNs is a Gaussian process
- Mysteries surrounding generalization and overparameterization.

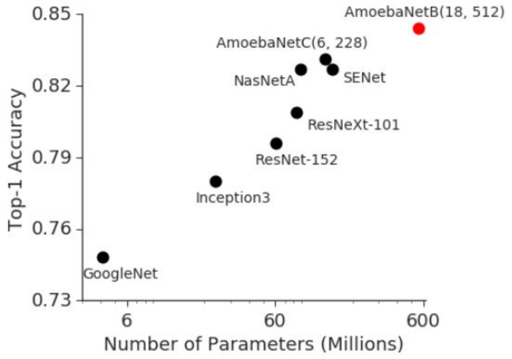
Empirical:

- Potentially of interest in practice. (Performance tends to improve as width is increased.)

This lecture will focus on the **theory of gradient descent dynamics in infinitely wide DNNs**.



Novak & Xiao, et al. ICLR 2019.

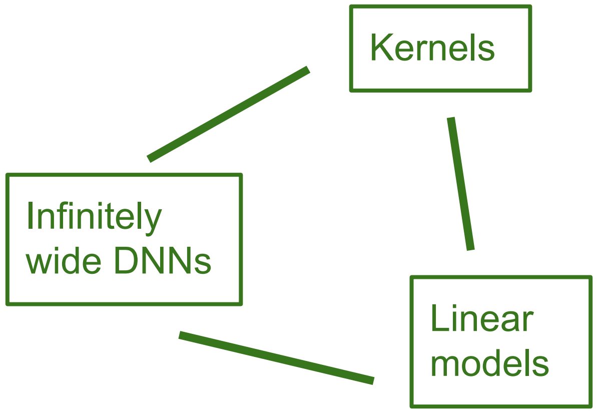


Huang, et al. NeurIPS 2019.

Outline for Today

1. Gradient descent in infinitely-wide DNNs
    a. Evolution in function space
    b. A simple example
    c. General result: “Neural Tangent Kernel” (NTK)
    d. Connection to linear models

2. Dynamics beyond NTK
    a. Sampling of other directions



Notation

Training dataset: $\mathcal{D} = \{(x^\alpha, y^\alpha)\}_{\alpha=1}^{D}$ with $\mathcal{D} \subseteq \mathbb{R}^d \times \mathbb{R}$. $D =$ number of training points.

Parameters: $\{W_{ij}^\ell, b_i^\ell\}$ with layer index $\ell = 0, ..., L$ and hidden layer widths $N_1, ...N_L$. We will use $\{\theta_\mu\}$ as shorthand for all the parameters (with $\mu = 1, ..., P$ where P is the total number of parameters).

Nonlinearity: $\phi(\cdot)$

Preactivations: $z_i^\ell(x)$

Feedforward computation: $z_i^\ell(x) = b_i^\ell + \sum_{j=1}^{N_\ell} W_{ij}^\ell \cdot \phi(z_j^{\ell-1}(x))$
(Base case: $z_i^0(x) = b_i^0 + \sum_{j=1}^{d} W_{ij}^0 x_j$.)

Network output: $f(x) \equiv z^L(x) \in \mathbb{R}$

Empirical loss: $\mathcal{L} \equiv \frac{1}{D} \sum_{\alpha \in \mathcal{D}} \ell(f(x^\alpha), y(x^\alpha))$

Focus will be on gradient descent (learning rate η) or gradient flow.

The view from function space

Suppose we are interested in gradient flow (descent), which is how we update parameters:

$$\frac{d\theta_\mu}{dt} = -\frac{\partial \mathcal{L}}{\partial \theta_\mu} = -\sum_{\alpha \in \mathcal{D}} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)} \frac{\partial f(x^\alpha)}{\partial \theta_\mu}$$

For simplicity of presentation, we stick to continuous time for now and comment on the discrete case later.

What evolution does this give rise to in function space?

$$\frac{df(x)}{dt} = \sum_\mu \frac{\partial f(x)}{\partial \theta_\mu} \frac{\partial \theta_\mu}{\partial t} = -\sum_{\alpha \in \mathcal{D}} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)} \left(\sum_\mu \frac{\partial f(x)}{\partial \theta_\mu} \frac{\partial f(x^\alpha)}{\partial \theta_\mu} \right)$$

This suggests defining a quantity, which is an inner product:

$$\Theta_t(x, x') \equiv \sum_\mu \frac{\partial f(x)}{\partial \theta_\mu} \frac{\partial f(x')}{\partial \theta_\mu}$$

What dictates function space dynamics?

$$\boxed{\frac{df(x)}{dt} = -\sum_{\alpha\in\mathcal{D}} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)} \cdot \Theta_t(x, x^\alpha)}$$

For instance, for square loss:

$$\frac{\partial \mathcal{L}}{\partial f(x^\alpha)} = \frac{1}{D}\Big(f(x^\alpha) - y(x^\alpha)\Big)$$

$$\boxed{\frac{d}{dt} \to \sum_\nu \frac{\partial \theta_\nu}{\partial t}\frac{\partial}{\partial \theta_\nu}}$$

$$\frac{d\Theta_t(x,x')}{dt} = \sum_\nu \frac{\partial \Theta_t(x,x')}{\partial \theta_\nu}\frac{\partial \theta_\nu}{\partial t}$$

$$= -\sum_{\alpha\in\mathcal{D}}\sum_{\mu,\nu} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)}\left(\frac{\partial^2 f(x)}{\partial\theta_\mu\partial\theta_\nu}\frac{\partial f(x')}{\partial\theta_\mu}\frac{\partial f(x^\alpha)}{\partial\theta_\nu} + \frac{\partial^2 f(x')}{\partial\theta_\mu\partial\theta_\nu}\frac{\partial f(x)}{\partial\theta_\mu}\frac{\partial f(x^\alpha)}{\partial\theta_\nu}\right)$$

$$\equiv -\sum_{\alpha\in\mathcal{D}} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)}\mathcal{O}_3(x,x',x^\alpha)$$

$$\frac{d\mathcal{O}_3(x,x',x'')}{dt} = \ldots$$

A set of coupled differential equations that is in general difficult to solve!

But we have identified some key quantities that influence the dynamics.

Different “parameterizations” we will work with

In a moment, we will take a closer look at the “dynamical kernel” and its evolution.

Before doing so, some comments on choices for neural network “parameterization.”

We often draw our parameters from a prior where the scale of the variance is dependent on width, e.g.: $W_{ij}^{\ell} \sim \mathcal{N}(0, \frac{\sigma_w^2}{N_\ell}), b_i^{\ell} \sim \mathcal{N}(0, \sigma_b^2)$

We can achieve a similar effect by explicitly parameterizing the network with these factors. (Will drop σ_w, σ_b for simplicity.)

“NTK” Parameterization

Weights: $W_{ij}^{\ell} \sim \mathcal{N}(0, 1)$
Factors $1/\sqrt{N_\ell}$ used explicitly in the model definition, e.g.:

$$z_i^{\ell} = b_i^{\ell} + \frac{1}{\sqrt{N_\ell}} \sum_{j=1}^{N_\ell} W_{ij}^{\ell} \cdot \phi(z_j^{\ell-1}(x))$$

The range of feasible learning rates in GD will be $\mathcal{O}(1)$.
(Deduce from the Hessian).

“Standard” Parameterization

Weights: $W_{ij}^{\ell} \sim \mathcal{N}(0, 1/N_\ell)$
No factors used explicitly in the model definition, e.g.:

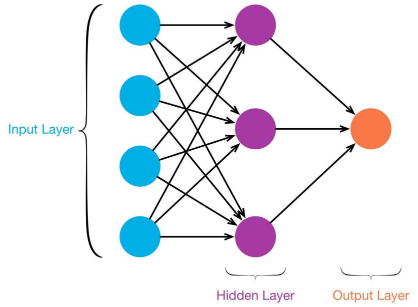
$$z_i^{\ell} = b_i^{\ell} + \sum_{j=1}^{N_\ell} W_{ij}^{\ell} \cdot \phi(z_j^{\ell-1}(x))$$

The range of feasible learning rates in GD will *depend on width.*
(Deduce this from the Hessian.)

Dynamics between these two ***in general is not equivalent*** but will not affect our infinite-width results. We work with “NTK” parameterization to make it easier to track how things *scale with width* in the next few slides.

Simple Example

$$f(x) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} W_i^1 \phi(z_i^0(x)), \qquad z_i^0(x) = \frac{1}{\sqrt{d}} \sum_{j=1}^{d} W_{ij}^0 x_j$$



$$\begin{aligned} \frac{dW_i^1}{dt} &= -\sum_{\alpha\in\mathcal{D}} \frac{\partial\mathcal{L}}{\partial f(x^\alpha)} \frac{\partial f(x^\alpha)}{\partial W_i^1} \\ &= -\frac{1}{\sqrt{N}} \sum_{\alpha\in\mathcal{D}} \frac{\partial\mathcal{L}}{\partial f(x^\alpha)} \phi(z_i^0(x^\alpha)) \\ &\propto -\frac{1}{\sqrt{N}} \sum_{\alpha\in\mathcal{D}} \Big(f(x^\alpha) - y(x^\alpha)\Big) \phi(z_i^0(x^\alpha)) \end{aligned}$$

$$\left.\frac{dW_i^1}{dt}\right|_{init} \sim \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$$

because all the other quantities are $\mathcal{O}(1)$, i.e. do not dependent on width.

Note that the initial scale is $W_i^1 \sim \mathcal{O}(1)$.

$$\frac{dW_{ij}^0}{dt} = -\frac{1}{\sqrt{N}} \sum_{\alpha\in\mathcal{D}} \frac{\partial\mathcal{L}}{\partial f(x^\alpha)} W_i^1 \phi'(z_i^0(x^\alpha)) \cdot \frac{x_j^\alpha}{\sqrt{d}}$$

$$\left.\frac{dW_{ij}^0}{dt}\right|_{init} \sim \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$$

Note that the initial scale is $W_{ij}^0 \sim \mathcal{O}(1)$.

Takeaway: the change ****per parameter**** vanishes in the infinite width limit, $N \to \infty$

The same is true for the hidden layer preactivation (per node).

Simple Example

What about the evolution of the function? Recall that:

$$\frac{df(x)}{dt} = -\sum_{\alpha \in \mathcal{D}} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)} \cdot \Theta_t(x, x^\alpha) \qquad \Theta_t(x, x') \equiv \sum_\mu \frac{\partial f(x)}{\partial \theta_\mu} \frac{\partial f(x')}{\partial \theta_\mu}$$

In this example,

$$\Theta_t(x, x') = \frac{1}{N}\sum_{i=1}^{N} \phi(z_i^0(x))\phi(z_i^0(x')) + \frac{1}{N}\sum_{i=1}^{N} (W_i^1)^2 \phi'(z_i^0(x))\phi'(z_i^0(x')) \cdot \left(\frac{1}{d}\sum_{j=1}^{d} x_j x'_j\right)$$

$$\Theta_{t=0}(x, x') \sim \mathcal{O}(1)$$

How does this dynamical variable evolve?

$$\frac{d\Theta_{t=0}(x, x')}{dt} \sim \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$$

At initialization, the update vanishes in the infinite width limit!

Main ideas for the General Case: Neural Tangent Kernel

In the previous, we focused on how quantities ***scale with neural network width***, *N*. (Since we are interested in the limit.)

We found that if we examined time evolution (rates of change) ***at initialization***, they vanished for a number of quantities.

- Of greatest interest: the (dynamical) kernel.

If we run the dynamics forward for times ~O(1), kernel would change by vanishingly small amount.

In fact, it is stronger than this, because we have not considered that ***training is happening***: the training loss itself is *decreasing*.

$$\frac{df(x)}{dt} = -\sum_{\alpha \in \mathcal{D}} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)} \cdot \Theta_t(x, x^\alpha) = -\frac{1}{D} \sum_{\alpha \in \mathcal{D}} \left(f(x^\alpha) - y(x^\alpha) \right) \cdot \Theta_t(x, x^\alpha)$$

Suppose that we approximate the kernel as constant.

As we will see, this is a linear differential equation: can solve it exactly!

We will find that the **model** approaches the **targets** exponentially fast ~ exp(-t).
So the **training loss** decreases exponentially fast.
The rapid decrease in the loss means the change in the kernel is even smaller than we expected.
Tracking the total change is ~ similar to summing a convergent geometric series.

Main Ideas for the General Case: Neural Tangent Kernel

- We've worked towards justifying that in the infinite-width limit, we can essentially just solve a ***closed*** equation:

$$\frac{df(x)}{dt} = -\frac{1}{D}\sum_{\alpha\in\mathcal{D}}\Big(f(x^\alpha) - y(x^\alpha)\Big)\cdot\Theta_0(x, x^\alpha)$$

- The dynamical variable has been replaced with its value at initialization (because it stays constant):

$$\Theta_0(x, x') \equiv \sum_\mu \frac{\partial f(x)}{\partial\theta_\mu}\frac{\partial f(x')}{\partial\theta_\mu}\Big|_{init}$$

- In general, this quantity itself is a random variable. It depends on the draws of the parameters at initialization. For instance, recall for our example:

$$\Theta_t(x, x') = \frac{1}{N}\sum_{i=1}^{N}\phi(z_i^0(x))\phi(z_i^0(x')) + \frac{1}{N}\sum_{i=1}^{N}(W_i^1)^2\phi'(z_i^0(x))\phi'(z_i^0(x'))\cdot\left(\frac{1}{d}\sum_{j=1}^{d}x_j x_j'\right)$$

- An additional fact is true: the kernel at initialization is itself a ***deterministic*** quantity, in the infinite width limit. We know that the preactivations become Gaussian processes -> we can use this to write down the expression for the kernel, in the infinite width limit (sketched on next slide).

Main Ideas for the General Case: Neural Tangent Kernel

- We sketch the form below. Recall, from a previous lecture, that in the limit of infinite width, preactivations ~ Gaussian Process:

$$z_i^\ell \sim \mathcal{N}(0, K^\ell) \text{ where } K^\ell(x, x') \text{ is the ``NNGP'' kernel}$$

(The node index is irrelevant, since all nodes are iid.)

$$\Theta_0(x, x') = \mathbb{E}_{z \sim \mathcal{GP}(0, K^0)}\left[\phi(z(x))\phi(z(x'))\right] + \sigma_w^2 \cdot \mathbb{E}_{z \sim \mathcal{GP}(0, K^0)}\left[\phi'(z(x))\phi'(z(x'))\right] \cdot \left(\frac{1}{d}\sum_{j=1}^{d} x_j x_j'\right)$$

These are just two dimensional integrals! For some nonlinearities (e.g. Relu), they can be computed analytically.

In the original paper, this deterministic quantity is the "Neural Tangent Kernel." It obeys the general recursion relation:

$$\Theta_0^\ell(x, x') = \Theta_0^{\ell-1}(x, x') \cdot \tau^\ell(x, x') + K^\ell(x, x')$$

where:

$$\tau^\ell(x, x') = \mathbb{E}_{z \sim \mathcal{GP}(0, K^\ell)}\left[\phi'(z(x))\phi'(z(x'))\right]$$

Main Ideas for the General Case: Neural Tangent Kernel

Hence, we keep track of: $K^0, K^1, ..., K^\ell \text{ and } \Theta^0, \Theta^1, ..., \Theta^{\ell-1} \rightarrow \text{to compute } \Theta^\ell$

We will return to the solution for the dynamics in a moment:

$$\frac{df(x)}{dt} = -\sum_{\alpha \in \mathcal{D}} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)} \cdot \Theta_0(x, x^\alpha)$$

(General)

$$\frac{df(x)}{dt} = -\frac{1}{D} \sum_{\alpha \in \mathcal{D}} \Big(f(x^\alpha) - y(x^\alpha) \Big) \cdot \Theta_0(x, x^\alpha)$$

(MSE)

Main Result: In the simultaneous limit (all layers -> infinity), we have:

- The Neural Tangent Kernel at initialization becomes a deterministic quantity.
- The NTK is frozen at its initial value under time evolution!
- The dynamics in function space is **entirely determined by this fixed kernel**!
  Can use it to make statements about convergence and generalization.
  For MSE, = kernel regression!

Original formulation was in continuous time; has since been extended to gradient *descent* (discrete time).

Main Ideas for the General Case: Neural Tangent Kernel

Let’s recap.

We found that the time evolution ***per parameter*** scales as a negative power of N -- based off an estimate ***at initialization*** -- so in the limit it will vanish.

- If not too much changes, this will continue to hold with time.

Similarly, the time evolution for preactivations (in hidden layers ***per node***) scales as a negative power of N - based off an estimate ***at initialization*** -- so it will vanish in the limit, if not too much changes.

However, note that the model (final output) does not stay fixed at initialization!

Infinitely many parameters changing a vanishingly small amount can have a collective effect.

Indeed, we will see that the final function changes by O(1) during training. This change is distributed over the (infinitely many) parameters.

What type of model realizes this dynamics?

So far, we have focused on the dynamics in function space, in the infinite width limit.

Can we say anything about the relationship between the learned function and the parameters?

It turns out the model dynamics is equivalent to that of its first order Taylor expansion about initialization.

The model behaves like its linearization.

$$\boxed{f_t(x) = f_0(x) + \sum_\mu \frac{\partial f_0(x)}{\partial \theta_\mu} \cdot \Big(\theta_\mu(t) - \theta_\mu(0)\Big)}$$

(We use $f_0(x)$ to denote the function at initialization.)

Note! Only the relationship between f(x) and the parameters is linear. ***The dependence on x is nonlinear.***

One can ~ view the features used by the linear model as the model gradients at initialization.

These random features were used in the construction of the Neural Tangent Kernel: $\Theta_0(x, x') = \sum_\mu \frac{\partial f_0(x)}{\partial \theta_\mu} \frac{\partial f_0(x')}{\partial \theta_\mu}$

Exact Solutions in Parameter and Function Space (MSE)

In the case of MSE, we can compute all quantities analytically.
A bit more notation used in the next few slides only:

$\mathcal{X} \equiv$ the vector of training inputs.
$\mathcal{Y} \equiv$ the vector of training labels.
$f_0(\mathcal{X}) \equiv$ the vector of model outputs on the inputs.
$\Theta_0 \equiv$ the $D \times D$ matrix from evaluating NTK on all pairs of inputs.

Evolution of parameters:

$$\boxed{\theta_\mu(t) - \theta_\mu(0) = -\frac{\partial f_0(\mathcal{X})}{\partial \theta_\mu}^T \cdot \Theta_0^{-1} \cdot \left(I_{D\times D} - e^{-\Theta_0 t}\right) \cdot \left(f_0(\mathcal{X}) - \mathcal{Y}\right)}$$

Exact Solutions in Parameter and Function Space (MSE)

The evolution of the model output on the training data:

$$f_t(\mathcal{X}) = \left(I_{D\times D} - e^{-\Theta_0 t}\right) \cdot \mathcal{Y} + e^{-\Theta_0 t} \cdot f_0(\mathcal{X})$$

As t -> infinity, the model output exactly matches the training targets. Notice that it converges exponentially fast, so the training loss will approach zero exponentially vast!

The evolution of the model output at a general point x: $f_t(x) = \mu_t(x) + \gamma_t(x) \text{ where}$

$$\boxed{\mu_t(x) = \Theta_0(x, \mathcal{X}) \cdot \Theta_0^{-1} \cdot \left(I_{D\times D} - e^{-\Theta_0 t}\right) \cdot \mathcal{Y}}$$

$$\boxed{\gamma_t(x) = f_0(x) - \Theta_0(x, \mathcal{X}) \cdot \Theta_0^{-1} \cdot \left(I_{D\times D} - e^{-\Theta_0 t}\right) \cdot f_0(\mathcal{X})}$$

Depends on the eigendecomposition of the NTK matrix (evaluated on the training data).

Exact Solutions in Parameter and Function Space (MSE)

We also know that $f_0(x)$ is (because of the infinite width limit) a draw from a Gaussian Process (NNGP).

$$f_t(x) = \mu_t(x) + \gamma_t(x) \text{ where}$$

$$\boxed{\mu_t(x) = \Theta_0(x, \mathcal{X}) \cdot \Theta_0^{-1} \cdot \left(I_{D\times D} - e^{-\Theta_0 t}\right) \cdot \mathcal{Y}}$$

$$\boxed{\gamma_t(x) = f_0(x) - \Theta_0(x, \mathcal{X}) \cdot \Theta_0^{-1} \cdot \left(I_{D\times D} - e^{-\Theta_0 t}\right) \cdot f_0(\mathcal{X})}$$

Hence, it is an affine transformation of Gaussian variables and will remain Gaussian distributed.

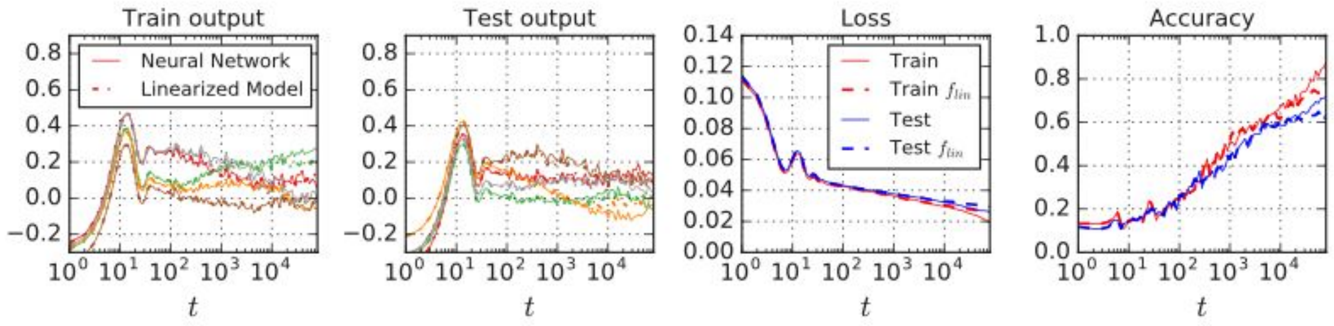
For any finite collection of test points, the model output will be a draw from a Gaussian Process with mean and covariance which we can write down.

See Ref [2] for the full expressions. It involves both the NNGP kernel and NTK.

Infinite-Width Dynamics and Linearized Models

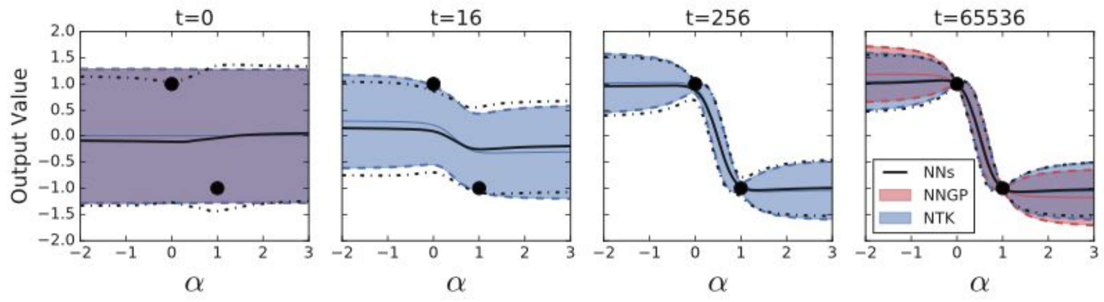
Several ways to numerically examine dynamics corresponding to NTK:

- Closed-form solutions (such as those for MSE)
- Use an ODE solver to solve the differential equation (e.g. for cross-entropy)
- Directly train the *(finite-width)* linearized model, using tools available (SGD, other optimizers, treat cross-entropy loss, architecturally difficult models)
    - Recommend the Neural Tangents Library (https://github.com/google/neural-tangents)

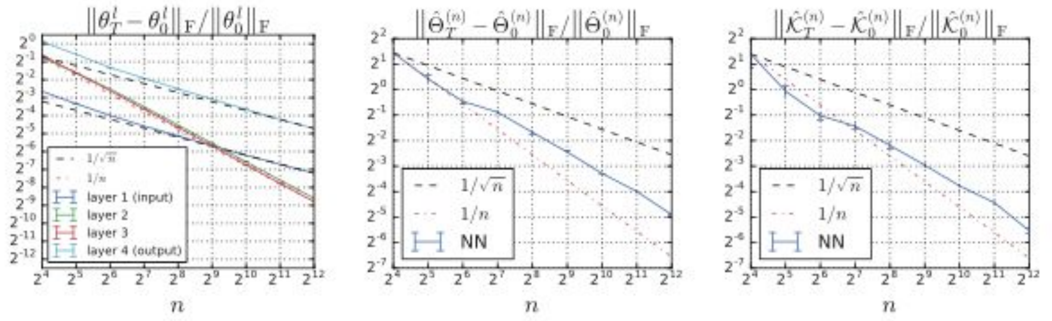


A WideResnet type model and its linearization. SGD with momentum and MSE loss on full CIFAR-10. Channel size = 1024, one block, batch size = 8.

Infinite-Width Dynamics and Linearized Models



Dynamics of mean and variance of NTK-GP and NNGP vs ensemble of 100 trained neural networks with full-batch GD. Small dataset size (128), three hidden layers of width ~8k.

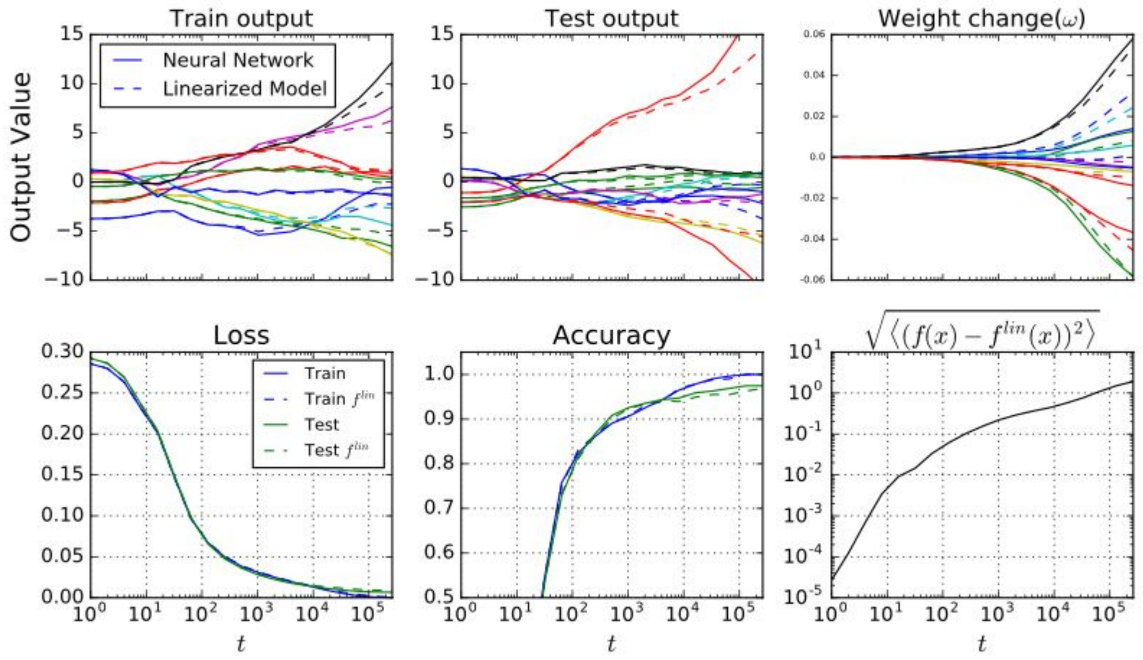


Relative Frobenius norm change after $O(2^{17})$ steps on a subset of MNIST for:

(a) parameters
(b) (finite-width) NTK
(c) (finite-width) NNGP kernel

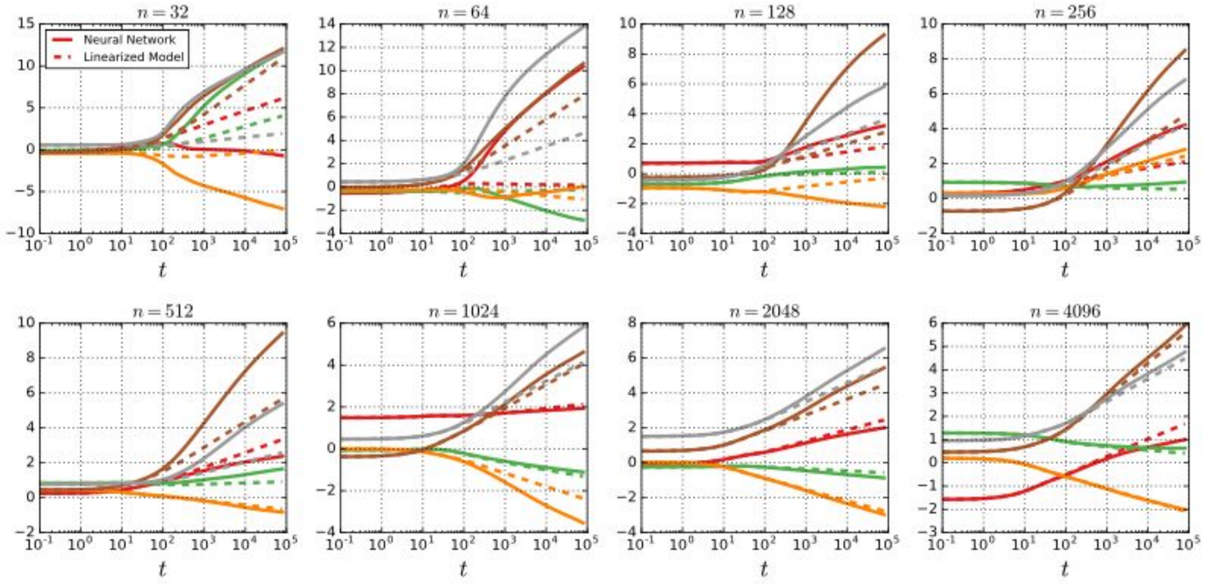
-> their change from initialization vanishes with width after training.

Infinite-Width Dynamics and Linearized Models



DNN vs its linearization. SGD (batch size 64) trained with momentum + cross-entropy loss on MNIST. Two-hidden layer Relu FC network.

Infinite-Width Dynamics and Linearized Models



Logit deviation for cross-entropy loss. Deviation happens later in training as network gets wider.
Depth = 4 fully-connected tanh network on binary CIFAR-10 task.

Remarks

- Extension to other model architectures: take a relevant dimension -> infinity.
  For instance, # of channels in convolutional neural network.

- Analysis of eigendecomposition of kernel (eigenvalues & vectors) -> insight into architectures.

- Linear model / kernel methods can be a generally useful tool even when far from NTK dynamics.
  - Dynamics may not match, but potentially other properties might correlate between DNN and its “corresponding” kernel

- Informs some of the mysteries of overparameterized neural networks:
  - Now we know that in the infinite-width limit, they correspond to linearized models.
  - Hence, only certain directions in parameter space (governed by the data) will be updated.

- Unanswered questions surrounding “feature learning.”

Partial List of References

Gradient descent in infinite-width deep NNs
[1]. Jacot, et al. “Neural Tangent Kernel: Convergence & Generalization in Neural Networks.” NeurIPS 2018.
Function space analysis, governed by fixed NTK.

[2]. Lee*, Xiao*, Schoenholz, YB, Novak, Sohl-Dickstein, Pennington. NeurIPS 2019.
NTK solution corresponds to a linearized model. Proof for additional settings (discrete time, parameterization-independent), empirical comparisons.

A number of papers on convergence of gradient descent in this limit:
[3]. Du, et al. ICML 2019; [4]. Allen-Zhu, et al. ICML 2019, and many others!

A different way to get linearized behavior:
[5]. Chizat, et al. “On Lazy Training in Differentiable Programming.” NeurIPS 2019.

NTK for convolutional networks:
[6]. Arora, et al. NeurIPS 2019.

Nice results in: [7]. Shankar, et al. “Neural Kernels without Tangents.” ICML 2020. (High-performing compositional kernel.)

Infinitely-wide deep NNs are GPs / Bayesian inference:
[8]. Neal. “Priors for Infinite Networks.” 1994. [Single-hidden layer NN]
[9]. Lee* and YB*, et al. ICLR 2018. [Deep NNs]
[10]. Matthews, et al. ICLR 2018. [Deep NNs]
And many other subsequent papers extending to other architectures (CNNs, attention-based models).

General approach for other architectures, both NNGP and NTK:
[11]. See works by G. Yang such as arxiv 1910.12478 (NeurIPS 2019) and arxiv 2006.14548.

Library for Easy NTK Computations:
https://github.com/google/neural-tangents

Dynamics for
Wide, Deep Neural Networks
“Beyond NTK”

Perturbative approaches about the infinite-width limit

“Perturbation theory” is a set of theoretical approaches typically used to compute the corrections to some solvable case, order by order in a series expansion.

For instance, **ε** ~ a power of inverse width is an expansion parameter (i.e. we take it to be small).

$$f_t^{NN}(x) = f_t^{NTK}(x) + \epsilon\, g_t^1(x) + \epsilon^2\, g_t^2(x) + \dots$$

If you are willing to only have a certain level of approximation, you can truncate at some order in the series.

Perturbative approaches about the infinite-width limit

Returning to the dynamics in function space: in general, a large set of coupled differential equations that seem hopeless!

Turns out that backing off slightly from the infinite-width limit, there is an ordering to them.

Lower equations in this “hierarchy” or tower of equations are more suppressed in the inverse width!

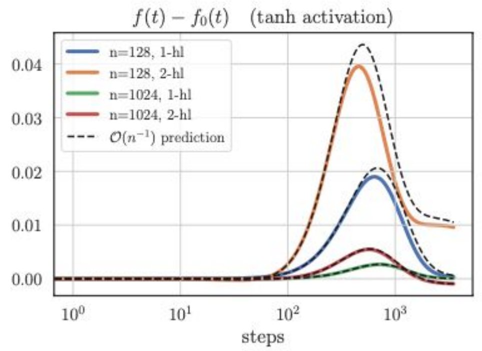
$$\frac{df(x)}{dt} = -\sum_{\alpha \in \mathcal{D}} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)} \cdot \Theta_t(x, x^\alpha)$$

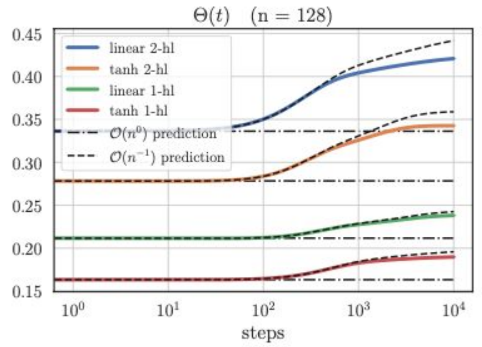
$$\frac{d\Theta_t(x, x')}{dt} = -\sum_{\alpha \in \mathcal{D}} \frac{\partial \mathcal{L}}{\partial f(x^\alpha)} \mathcal{O}_3(x, x', x^\alpha)$$

$$\frac{d\mathcal{O}_3(x, x', x'')}{dt} = \ldots$$

$$\downarrow$$

Hierarchy / tower of equations





Dyer & Gur-Ari, ICLR 2020 and Huang & Yai, ICML 2020.

Dynamics at large learning rates

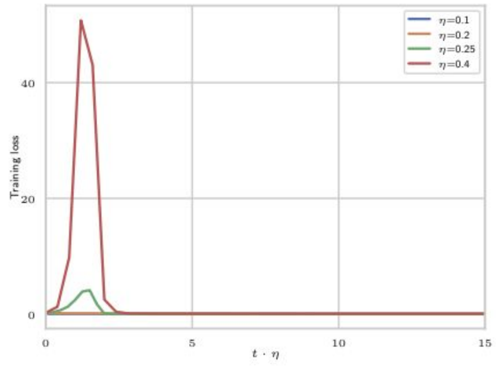
It turns out that the NTK solution is stable only up to a maximum learning rate.

This max learning rate makes sense from convex optimization:

$$\eta_{crit} = 2/\lambda_0 \text{ where } \lambda_0 \text{ is the max eigenvalue of the NTK}$$

	NTK	Catapult	Divergent
Learning rate	$\eta < 2/\lambda_0$	$2/\lambda_0 < \eta < c/\lambda_0$	$c/\lambda_0 < \eta$

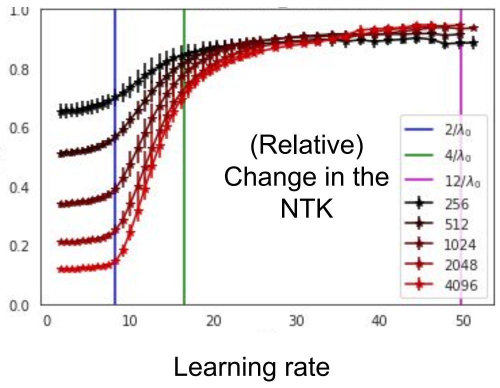
Crucially, c is a width-independent constant and $c > 2$.



Training loss vs time.
WRN 28-10 on CIFAR-10.

NTK Phase:
Loss decreases smoothly, kernel changes vanishingly small amount.

Catapult phase:
Loss rises and then drops, kernel changes by O(1) amount.



Dynamics at large learning rates

These coupled observations (change in kernel, growth & peaking in loss) are robust across:

- Neural architectures (fully-connected, convolutional, Wide Resnet, nonlinearities)
- Datasets (MNIST, CIFAR-10, CIFAR-100)
- Gradient descent setting (full-batch, SGD, decaying LR)

Is there a common underlying mechanism?

Can be understood in a single-hidden layer neural network with data!
Simplest version involves tracking two scalars (the scale of the model and NTK eigenvalue).

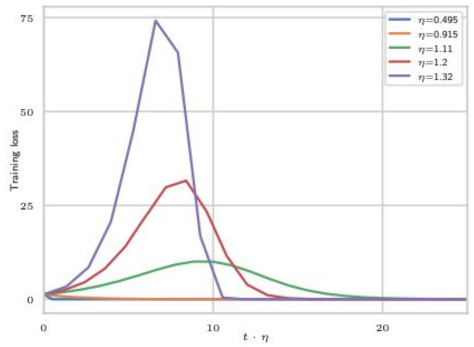
Coupled dynamical system with two scalar (f, λ) whose fixed points depends on (η, n).

$$\boxed{f_{t+1} = \left(1 - \eta\lambda_t + \frac{\eta^2 f_t^2}{n}\right) f_t}$$

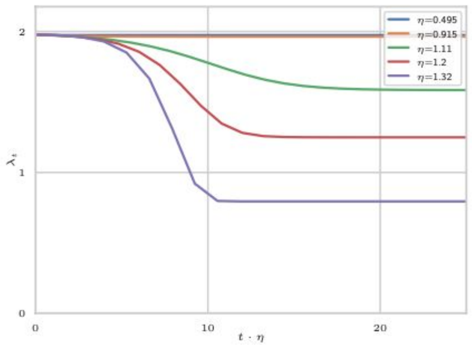
$$\boxed{\lambda_{t+1} = \lambda_t + \frac{\eta f_t^2}{n}(\eta\lambda_t - 4)}$$

The two transitions occur at $2/\lambda_0$ (NTK -> catapult phase) and $4/\lambda_0$ (catapult phase -> divergent).

Dynamics at large learning rates



Training loss vs time



Kernel vs time

Kernel changes by O(1) and adjusts dynamically to accommodate large learning rate.

$$\boxed{f_{t+1} = \left(1 - \eta\lambda_t + \frac{\eta^2 f_t^2}{n}\right) f_t}$$

$$\boxed{\lambda_{t+1} = \lambda_t + \frac{\eta f_t^2}{n}(\eta\lambda_t - 4)}$$

- Occurs at time scales t ~ log(N).
  - Not accessible by perturbative techniques (solving for a few corrections in the hierarchy).

Lewkowycz, et al. arxiv 2003.02218

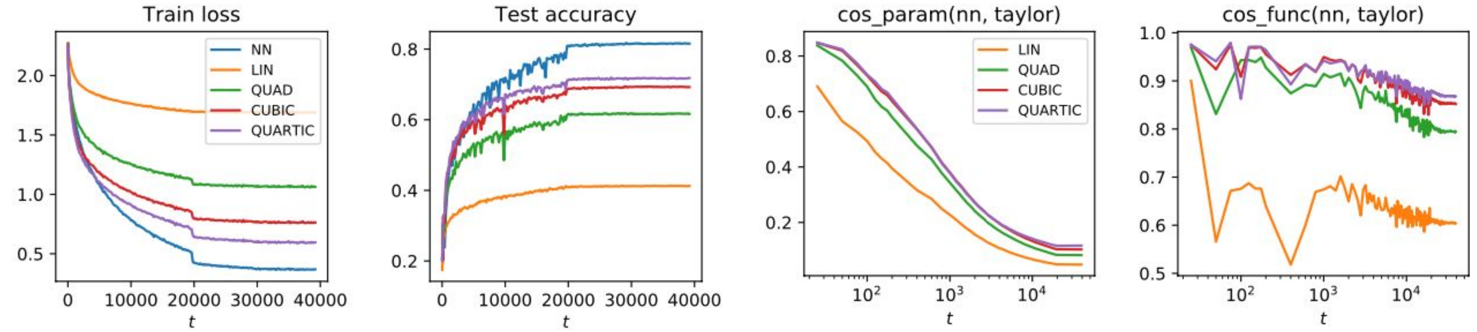
Higher-order Taylor series

Better descriptions of the model in parameter space?

Higher-order Taylor series expansion about initialization is natural.

$$f_t(x) = f_0(x) + \sum_{\mu} \frac{\partial f_0(x)}{\partial \theta_\mu} \cdot \Big(\theta_\mu(t) - \theta_\mu(0)\Big) + \frac{1}{2} \sum_{\mu,\nu} \frac{\partial^2 f_0(x)}{\partial \theta_\mu \partial \theta_\nu} \cdot \Big(\theta_\mu(t) - \theta_\mu(0)\Big) \cdot \Big(\theta_\nu(t) - \theta_\nu(0)\Big) + \ldots$$

See the Neural Tangents Library for easy tools.



CNN with depth = 4, width = 128, trained on CIFAR-10 with cross-entropy loss & learning rate decay.

Bai, et al. arxiv 2002.04010

Other settings under study

“Mean-field theory” (different from the one we discussed!)

The parameterization with width is typically different in these papers. For instance: $f(x) = \frac{1}{N}\sum_{i=1}^{N}\phi(z_i(x))$

Leads to non-trivial behavior in the infinite-width limit. See, e.g. [1, 2] and others.

[1]. Mei, et al. “A mean field view of the landscape of two-layered neural networks.” PNAS 2018.
[2]. Rotskoff, et al. “Neural networks as interacting particle systems: Asymptotic convexity of the loss scape & universal scaling of the approximation error.”

Simultaneous limits

For instance: **input dimension**, **width**, and **dataset size** -> infinity while keeping fixed ratios.

Often studied in the setting of random feature models. See, e.g. [1, 2] and others.

Other approaches: depth and width allowed to scale similarly. See e.g. [3] for some properties at initialization.

[1]. Mei & Montanari. “The generalization error of random features regression: Precise asymptotics and double descent curve.” CPAM, to appear.
[2]. Adlam & Pennington. “The Neural Tangent Kernel in high dimensions: triple descent and a multi-scale theory of generalization.” ICML 2020.
[3]. Hanin & Nica. “Finite depth and width corrections to the Neural Tangent Kernel.” ICLR 2020.

Recap & conclusion

Theoretical connections between infinitely-wide DNNs, kernels, and linear models

- Everywhere you have a DNN -> can ask about the corresponding linear model

- Helps resolve some of the mysteries of overparameterization
  - Problem is well-controlled -- effectively convex
  - Reduces the problem of generalization to one for kernels

- Informs design of compositional kernels

- Typical neural networks in large-scale image classification (where datasets are large) may be operating far from this regime -> but will depend on settings!
  - ~need to check on a per-setting basis

- DNNs are “learning a kernel” -> we saw this in the function space perspective

- Many more questions surrounding “feature learning”