ETH zürich

On Exact Computation with an Infinitely Wide Neural Net

Aurelien Lucchi

1. Recall NTK

NTK

- Notation: Denote by $f(\boldsymbol{\theta}, \boldsymbol{x}) \in \mathbb{R}$ the output of a neural network
  - $\boldsymbol{\theta} \in \mathbb{R}^N$ is all the parameters in the network
  - $\boldsymbol{x} \in \mathbb{R}^d$ is the input

NTK

- Notation: Denote by $f(\boldsymbol{\theta}, \boldsymbol{x}) \in \mathbb{R}$ the output of a neural network
  - $\boldsymbol{\theta} \in \mathbb{R}^N$ is all the parameters in the network
  - $\boldsymbol{x} \in \mathbb{R}^d$ is the input
- NTK is defined using the *gradient* of the output of the randomly initialized net with respect to its parameters, i.e.,

$$\ker(\boldsymbol{x}, \boldsymbol{x}') = \mathop{\mathbb{E}}_{\boldsymbol{\theta} \sim \mathcal{W}} \left\langle \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x}')}{\partial \boldsymbol{\theta}} \right\rangle . \tag{1}$$

- Generalization to convolutional neural nets: *Convolutional Neural Tangent Kernel (CNTK)*.

Contributions

- **Theory:** Non-asymptotic proof that the NTK captures the behavior of a fully-trained wide neural net
  - weaker condition than previous proofs!
- **Empirically:**
  - New dynamic programming algorithm to compute CNTKs for ReLU activation (namely, to compute $\ker(\boldsymbol{x}, \boldsymbol{x}')$)
  - Evaluate the performance of fully-trained infinitely wide nets: performance on CIFAR-10 is within 5% of the performance in the finite case

Gradient flow & NTK

- Training by minimizing the squared loss:
  $\ell(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(f(\boldsymbol{\theta}, \boldsymbol{x}_i) - y_i\right)^2$.
- Optimize using gradient flow: $\frac{\mathbf{d}\boldsymbol{\theta}(t)}{\mathbf{d}t} = -\nabla\ell(\boldsymbol{\theta}(t))$

Gradient flow & NTK

- Training by minimizing the squared loss: $\ell(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(f(\boldsymbol{\theta}, \boldsymbol{x}_i) - y_i\right)^2$.
- Optimize using gradient flow: $\frac{\mathbf{d}\boldsymbol{\theta}(t)}{\mathbf{d}t} = -\nabla \ell(\boldsymbol{\theta}(t))$

Lemma

Let $\boldsymbol{u}(t) = (f(\boldsymbol{\theta}(t), \boldsymbol{x}_i))_{i\in[n]} \in \mathbb{R}^n$ be the network outputs on all $\boldsymbol{x}_i$'s at time t, and $\boldsymbol{y} = (y_i)_{i\in[n]}$ be the desired outputs. Then $\boldsymbol{u}(t)$ follows the following evolution, where $\boldsymbol{H}(t)$ is an $n \times n$ positive semidefinite matrix whose (i, j)-th entry is $\left\langle \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_j)}{\partial \boldsymbol{\theta}} \right\rangle$:

$$\frac{\mathbf{d}\boldsymbol{u}(t)}{\mathbf{d}t} = -\boldsymbol{H}(t) \cdot (\boldsymbol{u}(t) - \boldsymbol{y}). \tag{2}$$

Limit behavior

- In the infinite limit width, matrix $\boldsymbol{H}(t)$ remains *constant* during training i.e. $\boldsymbol{H}(t) = \boldsymbol{H}(0)$.
- Under a random initialization of θ, as the width goes to infinity, $\boldsymbol{H}(0)$ converges in probability to a deterministic kernel matrix $\boldsymbol{H}^* = \mathrm{ker}(\cdot, \cdot)$ (i.e. the NTK)
- $\implies$ Equation (2) becomes

$$\frac{\mathbf{d}\boldsymbol{u}(t)}{\mathbf{d}t} = -\boldsymbol{H}^* \cdot (\boldsymbol{u}(t) - \boldsymbol{y}). \tag{3}$$

Limit behavior

- In the infinite limit width, matrix $\boldsymbol{H}(t)$ remains *constant* during training i.e. $\boldsymbol{H}(t) = \boldsymbol{H}(0)$.
- Under a random initialization of θ, as the width goes to infinity, $\boldsymbol{H}(0)$ converges in probability to a deterministic kernel matrix $\boldsymbol{H}^* = \ker(\cdot, \cdot)$ (i.e. the NTK)
- $\implies$ Equation (2) becomes

$$\frac{\mathbf{d}\boldsymbol{u}(t)}{\mathbf{d}t} = -\boldsymbol{H}^* \cdot (\boldsymbol{u}(t) - \boldsymbol{y}). \tag{3}$$

- Remark: Above dynamics is identical to the dynamics of *kernel regression* under gradient flow, for which at time $t \to \infty$ the final prediction function is:

$$f^*(\boldsymbol{x}) = (\ker(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, \ker(\boldsymbol{x}, \boldsymbol{x}_n)) \cdot (\boldsymbol{H}^*)^{-1} \boldsymbol{y}. \tag{4}$$

2. Main results

Network

- Define L-hidden-layer fully-connected neural network:

$$\begin{aligned} \boldsymbol{f}^{(h)}(\boldsymbol{x}) &= \boldsymbol{W}^{(h)} \boldsymbol{g}^{(h-1)}(\boldsymbol{x}) \in \mathbb{R}^{d_h} \\ \boldsymbol{g}^{(h)}(\boldsymbol{x}) &= \sqrt{\frac{c_\sigma}{d_h}} \sigma\left(\boldsymbol{f}^{(h)}(\boldsymbol{x})\right) \in \mathbb{R}^{d_h}, \qquad h = 1, 2, \ldots, L, \end{aligned} \tag{5}$$

where $\boldsymbol{W}^{(h)} \in \mathbb{R}^{d_h \times d_{h-1}}$, $\sigma : \mathbb{R} \to \mathbb{R}$,
$c_\sigma = \left(\mathbb{E}_{z \sim \mathcal{N}(0,1)}\left[\sigma(z)^2\right]\right)^{-1}$.

Network

- Last layer of the neural network is

$$f^{(L+1)}(\boldsymbol{x}) = \boldsymbol{W}^{(L+1)} \cdot \boldsymbol{g}^{(L)}(\boldsymbol{x})$$

$$= \boldsymbol{W}^{(L+1)} \cdot \sqrt{\frac{c_\sigma}{d_L}} \sigma \left(\boldsymbol{W}^{(L)} \cdot \sqrt{\frac{c_\sigma}{d_{L-1}}} \sigma \left(\boldsymbol{W}^{(L-1)} \cdots \sqrt{\frac{c_\sigma}{d_1}} \sigma \left(\boldsymbol{W}^{(1)} \boldsymbol{x} \right) \right) \right),$$

where $\boldsymbol{\theta} = \left(\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(L+1)} \right)$

NTK (Lee et al. (2018))

- $\boldsymbol{f}^{(h)}(\boldsymbol{x}) = \boldsymbol{W}^{(h)}\boldsymbol{g}^{(h-1)}(\boldsymbol{x}) \in \mathbb{R}^{d_h}$
- Recall from Lee et al. (2018) that in the infinite width limit, the pre-activations $\boldsymbol{f}^{(h)}(\boldsymbol{x})$ at every hidden layer $h \in [L]$ has all its coordinates tending to i.i.d. centered Gaussian processes of covariance $\Sigma^{(h-1)} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined recursively as:

$$\begin{aligned}
\Sigma^{(0)}(\boldsymbol{x}, \boldsymbol{x}') &= \boldsymbol{x}^\top \boldsymbol{x}', \\
\boldsymbol{\Lambda}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') &= \begin{pmatrix} \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \\ \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}') \end{pmatrix} \in \mathbb{R}^{2\times 2}, \qquad (6) \\
\Sigma^{(h)}(\boldsymbol{x}, \boldsymbol{x}') &= c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Lambda}^{(h)})} \left[\sigma(u)\,\sigma(v)\right].
\end{aligned}$$

NTK

- To give the formula of NTK, we also need to define a derivative covariance:

$$\dot{\Sigma}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \underset{(u,v)\sim\mathcal{N}\left(\mathbf{0},\mathbf{\Lambda}^{(h)}\right)}{\mathbb{E}} \left[\dot{\sigma}(u)\dot{\sigma}(v)\right]. \tag{7}$$

- Final NTK expression for the fully-connected neural network is

$$\boxed{\Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{h=1}^{L+1} \left(\Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \prod_{h'=h}^{L+1} \dot{\Sigma}^{(h')}(\boldsymbol{x}, \boldsymbol{x}') \right), \tag{8}}$$

where we let $\dot{\Sigma}^{(L+1)}(\boldsymbol{x}, \boldsymbol{x}') = 1$ for convenience.

NTK at initialization

Theorem (1) Convergence to the NTK at initialization)

Fix $\epsilon > 0$ *and* $\delta \in (0,1)$. *Suppose* $\sigma(z) = \max(0, z)$ *and* $\min_{h\in[L]} d_h \geq \Omega(\frac{L^6}{\epsilon^4}\log(L/\delta))$. *Then for any inputs* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{d_0}$ *such that* $\|\boldsymbol{x}\| \leq 1$, $\|\boldsymbol{x}'\| \leq 1$, *with probability at least* $1-\delta$ *we have:*

$$\left|\left\langle \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x}')}{\partial \boldsymbol{\theta}} \right\rangle - \Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\right| \leq (L+1)\epsilon.$$

NTK at initialization

Theorem (1) Convergence to the NTK at initialization)

Fix $\epsilon > 0$ *and* $\delta \in (0, 1)$. *Suppose* $\sigma(z) = \max(0, z)$ *and* $\min_{h\in[L]} d_h \geq \Omega(\frac{L^6}{\epsilon^4} \log(L/\delta))$. *Then for any inputs* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{d_0}$ *such that* $\|\boldsymbol{x}\| \leq 1$, $\|\boldsymbol{x}'\| \leq 1$, *with probability at least* $1 - \delta$ *we have:*

$$\left| \left\langle \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x}')}{\partial \boldsymbol{\theta}} \right\rangle - \Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \right| \leq (L + 1)\epsilon.$$

Improvements upon previous results (Jacot et al. 2018):

- Non-asymptotic bound on the required layer widths, previous results are asymptotic, (i.e. all $d_h \to \infty$)
- Only requires $\min_{h\in[L]} d_h$ to be sufficiently large, which is the weakest notion of limit.

Equivalence wide neural net and kernel regression

- Can further incorporate the training process and show the equivalence between a fully-trained sufficiently wide neural net and the kernel regression solution

Equivalence wide neural net and kernel regression

- Can further incorporate the training process and show the equivalence between a fully-trained sufficiently wide neural net and the kernel regression solution

- Final output of the neural network be $f_{nn}(\boldsymbol{\theta}, \boldsymbol{x}) = \kappa f(\boldsymbol{\theta}, \boldsymbol{x})$. and $f_{nn}(\boldsymbol{x}_{te}) = \lim_{t\to\infty} f_{nn}(\boldsymbol{\theta}(t), \boldsymbol{x}_{te})$ be the prediction of the neural network at the end of training

Equivalence wide neural net and kernel regression

Theorem (2) Equivalence between trained net and kernel regression)

Suppose $\sigma(z) = \max(0, z)$, $1/\kappa = \mathrm{poly}(1/\epsilon, \log(n/\delta))$ *and* $d_1 = d_2 = \cdots = d_L = m$ *with* $m \geq \mathrm{poly}(1/\kappa, L, 1/\lambda_0, n, \log(1/\delta))$. *Then for any* $\boldsymbol{x}_{te} \in \mathbb{R}^d$ *with* $\|\boldsymbol{x}_{te}\| = 1$, *with probability at least* $1 - \delta$ *over the random initialization, we have*

$$|f_{nn}(\boldsymbol{x}_{te}) - f_{ntk}(\boldsymbol{x}_{te})| \leq \epsilon.$$

Equivalence wide neural net and kernel regression

Theorem (2) Equivalence between trained net and kernel regression)

Suppose $\sigma(z) = \max(0, z)$, $1/\kappa = \mathrm{poly}(1/\epsilon, \log(n/\delta))$ *and* $d_1 = d_2 = \cdots = d_L = m$ *with* $m \geq \mathrm{poly}(1/\kappa, L, 1/\lambda_0, n, \log(1/\delta))$. *Then for any* $\boldsymbol{x}_{te} \in \mathbb{R}^d$ *with* $\|\boldsymbol{x}_{te}\| = 1$, *with probability at least* $1-\delta$ *over the random initialization, we have*

$$|f_{nn}(\boldsymbol{x}_{te}) - f_{ntk}(\boldsymbol{x}_{te})| \leq \epsilon.$$

- **Remark:** Result can be extended to (exponentially many) finite testing points using a union bound.

Proof Idea Theorem 2

- First use a generic argument to show that the perturbation on the prediction can be reduced to the perturbation on kernel value at the initialization and during training.

Proof Idea Theorem 2

- First use a generic argument to show that the perturbation on the prediction can be reduced to the perturbation on kernel value at the initialization and during training.
- Initialization: Theorem 1 guarantees a small perturbation on kernel value.
- Training: use high level proof idea from Du et al. 2018 to reduce the perturbation on the kernel value to the perturbation on the gradient of each prediction with respect to weight matrices. Then we adopt technical lemmas from Allen-Zhu et al. 2018 to obtain bounds on the perturbation of the gradient.

3. Proofs

NTK Derivation

- **Goal:** Compute the limit of $\left\langle \frac{\partial f(\boldsymbol{\theta},\boldsymbol{x})}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta},\boldsymbol{x}')}{\partial \boldsymbol{\theta}} \right\rangle$ at random initialization in the infinite width limit.
- Recall: $\boldsymbol{f}^{(h)}(\boldsymbol{x}) = \boldsymbol{W}^{(h)}\boldsymbol{g}^{(h-1)}(\boldsymbol{x}) \in \mathbb{R}^{d_h}$

NTK Derivation

- **Goal:** Compute the limit of $\left\langle \frac{\partial f(\boldsymbol{\theta},\boldsymbol{x})}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta},\boldsymbol{x}')}{\partial \boldsymbol{\theta}} \right\rangle$ at random initialization in the infinite width limit.
- Recall: $\boldsymbol{f}^{(h)}(\boldsymbol{x}) = \boldsymbol{W}^{(h)}\boldsymbol{g}^{(h-1)}(\boldsymbol{x}) \in \mathbb{R}^{d_h}$
- Partial derivative w.r.t. particular weight $\boldsymbol{W}^{(h)}$:

$$\frac{\partial f(\boldsymbol{\theta},\boldsymbol{x})}{\partial \boldsymbol{W}^{(h)}} = \mathbf{b}^{(h)}(\boldsymbol{x}) \cdot \left(\boldsymbol{g}^{(h-1)}(\boldsymbol{x})\right)^{\top}, \qquad h = 1, 2, \ldots, L+1,$$

where

$$\mathbf{b}^{(h)}(\boldsymbol{x}) = \begin{cases} 1 \in \mathbb{R}, & h = L+1, \\ \sqrt{\frac{c_\sigma}{d_h}} \boldsymbol{D}^{(h)}(\boldsymbol{x}) \left(\boldsymbol{W}^{(h+1)}\right)^{\top} \mathbf{b}^{(h+1)}(\boldsymbol{x}) \in \mathbb{R}^{d_h}, & h = 1, \ldots, L, \end{cases}$$

$$\boldsymbol{D}^{(h)}(\boldsymbol{x}) = \operatorname{diag}\left(\dot{\sigma}\left(\boldsymbol{f}^{(h)}(\boldsymbol{x})\right)\right) \in \mathbb{R}^{d_h \times d_h}, \qquad h = 1, \ldots, L.$$

NTK Derivation

Then, for any $h \in [L+1]$, we can compute

$$\left\langle \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x})}{\partial \boldsymbol{W}^{(h)}}, \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x}')}{\partial \boldsymbol{W}^{(h)}} \right\rangle = \left\langle \mathbf{b}^{(h)}(\boldsymbol{x}) \cdot \left(\boldsymbol{g}^{(h-1)}(\boldsymbol{x})\right)^{\top}, \mathbf{b}^{(h)}(\boldsymbol{x}') \cdot \left(\boldsymbol{g}^{(h-1)}(\boldsymbol{x}')\right)^{\top} \right\rangle$$

$$= \left\langle \boldsymbol{g}^{(h-1)}(\boldsymbol{x}), \boldsymbol{g}^{(h-1)}(\boldsymbol{x}') \right\rangle \cdot \left\langle \mathbf{b}^{(h)}(\boldsymbol{x}), \mathbf{b}^{(h)}(\boldsymbol{x}') \right\rangle .$$

- Using CLT,

$$\left\langle \boldsymbol{g}^{(h-1)}(\boldsymbol{x}), \boldsymbol{g}^{(h-1)}(\boldsymbol{x}') \right\rangle \to \Sigma^{(h-1)}\left(\boldsymbol{x}, \boldsymbol{x}'\right) .$$

- Inductively, can show that

$$\left\langle \mathbf{b}^{(h)}(\boldsymbol{x}), \mathbf{b}^{(h)}(\boldsymbol{x}') \right\rangle \to \prod_{h'=h}^{L} \dot{\Sigma}^{(h')}(\boldsymbol{x}, \boldsymbol{x}').$$

Proof Theorem 2

- Recall: Final output of the neural network be $f_{nn}(\boldsymbol{\theta}, \boldsymbol{x}) = \kappa f(\boldsymbol{\theta}, \boldsymbol{x})$. and $f_{nn}(\boldsymbol{x}_{te}) = \lim_{t \to \infty} f_{nn}(\boldsymbol{\theta}(t), \boldsymbol{x}_{te})$ be the prediction of the neural network at the end of training.
- We define $\text{ker}_t(\boldsymbol{x}_{te}, \boldsymbol{X}) \in \mathbb{R}^n$ as

$$[\text{ker}_t(\boldsymbol{x}_{te}, \boldsymbol{X})]_i = \left\langle \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_{te})}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}} \right\rangle$$

Proof Theorem 2

- Recall: Final output of the neural network be $f_{nn}(\boldsymbol{\theta}, \boldsymbol{x}) = \kappa f\left(\boldsymbol{\theta}, \boldsymbol{x}\right)$. and $f_{nn}(\boldsymbol{x}_{te}) = \lim_{t\to\infty} f_{nn}(\boldsymbol{\theta}(t), \boldsymbol{x}_{te})$ be the prediction of the neural network at the end of training.
- We define $\mathrm{ker}_t(\boldsymbol{x}_{te}, \boldsymbol{X}) \in \mathbb{R}^n$ as

$$[\mathrm{ker}_t\left(\boldsymbol{x}_{te}, \boldsymbol{X}\right)]_i = \left\langle \frac{\partial f\left(\boldsymbol{\theta}(t), \boldsymbol{x}_{te}\right)}{\partial \boldsymbol{\theta}}, \frac{\partial f\left(\boldsymbol{\theta}(t), \boldsymbol{x}_i\right)}{\partial \boldsymbol{\theta}} \right\rangle$$

- Let $\mathrm{ker}_{ntk}\left(\boldsymbol{x}_{te}, \boldsymbol{X}\right) \in \mathbb{R}^n$ be the kernel values between $\boldsymbol{x}_{te}$ and each training data (i.e., $[\mathrm{ker}_{ntk}(\boldsymbol{x}_{te}, \boldsymbol{X})]_i = \Theta^{(L)}(\boldsymbol{x}_{te}, \boldsymbol{x}_i)$). Then, the prediction of kernel regression is

$$f_{ntk}(\boldsymbol{x}_{te}) = \mathrm{ker}_{ntk}(\boldsymbol{x}_{te}, \boldsymbol{X})^\top \underbrace{\left(\boldsymbol{H}^*\right)^{-1}}_{[\boldsymbol{H}^*]_{i,j}=\Theta^{(L)}(\boldsymbol{x}_i, \boldsymbol{x}_j)} \boldsymbol{y}. \tag{9}$$

Proof Theorem 2

- First prove a lemma to reduce the prediction perturbation bound to the kernel perturbation bound.

Lemma (Kernel Value Perturbation $\Rightarrow$ Prediction Perturbation)

Fix $\epsilon_{\boldsymbol{H}} \leq \frac{1}{2}\lambda_0$. *Suppose* $|f_{nn}(\boldsymbol{\theta}(0), \boldsymbol{x}_i)| \leq \epsilon_{init}$ *for* $i = 1, \ldots, n$ *and* $|f_{nn}(\boldsymbol{\theta}(0), \boldsymbol{x}_{te})| \leq \epsilon_{init}$ *and* $\|\boldsymbol{u}_{nn}(0) - \boldsymbol{y}\|_2 = O(\sqrt{n})$.
Furthermore, if for all $t \geq 0$
$\|\mathrm{ker}_{ntk}(\boldsymbol{x}_{te}, \boldsymbol{X}) - \mathrm{ker}_t(\boldsymbol{x}_{te}, \boldsymbol{X})\|_2 \leq \epsilon_{test}$ *and*
$\|\boldsymbol{H}^* - \boldsymbol{H}(t)\|_2 \leq \epsilon_{\boldsymbol{H}}$, *then we have*

$$|f_{ntk}(\boldsymbol{x}_{te}) - f_{nn}(\boldsymbol{x}_{te})| \leq O\left(\epsilon_{init} + \frac{\sqrt{n}}{\lambda_0}\epsilon_{test} + \frac{\sqrt{n}}{\lambda_0^2}\log\left(\frac{n}{\epsilon_{\boldsymbol{H}}\lambda_0\kappa}\right)\epsilon_{\boldsymbol{H}}\right).$$

Proof Theorem 2

- Use $\beta(t)$ to denote this parameter at time t trained by gradient flow and $f_{ntk}\left(\boldsymbol{x}_{te}, \beta(t)\right)$ be the predictor for $\boldsymbol{x}_{te}$ at time t.
- Then

$$f_{ntk}(\boldsymbol{x}_{te}) = \int_{t=0}^{\infty} \frac{df_{ntk}(\beta(t), \boldsymbol{x}_{te})}{dt} dt$$

- Now we take a closer look at the time derivative:

$$\frac{df_{ntk}(\beta(t), \boldsymbol{x}_{te})}{dt} = -\frac{\kappa^2}{n} \mathrm{ker}_{ntk}(\boldsymbol{x}_{te}, \boldsymbol{X})^\top \left(\boldsymbol{u}_{ntk}(t) - \boldsymbol{y}\right)$$

where $u_{ntk,i}(t) = f_{ntk}(\beta(t), \boldsymbol{x}_i)$ and $\boldsymbol{u}_{ntk}(t) \in \mathbb{R}^n$ with $[\boldsymbol{u}_{ntk}(t)]_i = u_{ntk,i}(t)$.

Proof Theorem 2

- Similarly, for the NN predictor:

$$\frac{df_{nn}(\boldsymbol{\theta}(t), \boldsymbol{x}_{te})}{dt} = -\frac{\kappa^2}{n}\mathrm{ker}_t(\boldsymbol{x}_{te}, \boldsymbol{X})^\top (\boldsymbol{u}_{nn}(t) - \boldsymbol{y})$$

Proof Theorem 2

- Similarly, for the NN predictor:

$$\frac{df_{nn}(\boldsymbol{\theta}(t), \boldsymbol{x}_{te})}{dt} = -\frac{\kappa^2}{n}\mathrm{ker}_t(\boldsymbol{x}_{te}, \boldsymbol{X})^\top (\boldsymbol{u}_{nn}(t) - \boldsymbol{y})$$

- Next, we analyze the difference between the NN predictor and NTK predictor via this integral form

$$\begin{aligned} &|f_{nn}(\boldsymbol{x}_{te}) - f_{ntk}(\boldsymbol{x}_{te})| \\ \leq &\epsilon_{init} + \kappa^2 \epsilon_{test} \int_{t=0}^{\infty} \|\boldsymbol{u}_{nn}(t) - \boldsymbol{y}\|_2 \, dt \\ &+ \kappa^2 \max_{0 \leq t \leq \infty} \|\mathrm{ker}_{ntk}(\boldsymbol{x}_{te}, \boldsymbol{X})\|_2 \int_{t=0}^{\infty} \|\boldsymbol{u}_{nn}(t) - \boldsymbol{u}_{ntk}(t)\|_2 \, dt \end{aligned}$$

Proof Theorem 2

- Next, we observe that $\boldsymbol{u}_{nn}(t) \to \boldsymbol{y}$ and $\boldsymbol{u}_{ntk}(t) \to \boldsymbol{y}$ with linear convergence rate (Allen-Zhu et al. 2018), thus:

$$\|\boldsymbol{u}_{nn}(t) - \boldsymbol{y}\|_2 \leq \exp(-\frac{\kappa^2}{2}\lambda_0 t) \underbrace{\|\boldsymbol{u}_{nn}(0) - \boldsymbol{y}\|_2}_{O(\sqrt{n})}$$

.

Concluding the proof

By previous Lemma, the problem now reduces to

i) Choose κ small enough to make $\epsilon_{init} = O(\epsilon)$

ii) Show when the width is large enough then $\epsilon_{\boldsymbol{H}}$ and ϵ_{test} are both $O(\epsilon)$.

Concluding the proof

By previous Lemma, the problem now reduces to

i) Choose κ small enough to make $\epsilon_{init} = O(\epsilon)$

ii) Show when the width is large enough then $\epsilon_{\boldsymbol{H}}$ and ϵ_{test} are both $O(\epsilon)$.

How?

i) Use result by Daniely et al., 2016 to choose $\kappa = O\left(\frac{\epsilon}{\log(n/\delta)}\right)$ to make $\epsilon_{init} = O(\epsilon)$ with probability $1 - \delta$.

ii) Use Theorem 1 and additional lemma that bounds kernel perturbation

Proof Theorem 1

- Infinite limit of the blue and red terms can be made more precise.

Theorem

Let $\sigma(z) = \max(0, z), z \in \mathbb{R}$, *if* $[\boldsymbol{W}^{(h)}]_{ij} \overset{i.i.d.}{\sim} N(0,1)$, $\forall h \in [L+1], i \in [d^{h+1}], j \in [d^h]$, *there exist constants* c_1,c_2, *such that if* $\min_{h\in[L]} d_h \geq c_1 \frac{L^2 \log(\frac{L}{\delta})}{\epsilon^4}, \epsilon \leq \frac{c_2}{L}$, *then for any fixed* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{d_0}$, $\|\boldsymbol{x}\|, \|\boldsymbol{x}'\| \leq 1$,*we have w.p.* $1-\delta$, $\forall 0 \leq h \leq L$, $\forall (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) \in \{(\boldsymbol{x}, \boldsymbol{x}), (\boldsymbol{x}, \boldsymbol{x}'), (\boldsymbol{x}', \boldsymbol{x}')\}$,

$$\left| \mathbf{g} h \boldsymbol{x}^{(2)\top} \mathbf{g} h \boldsymbol{x}^{(1)} - \boldsymbol{\Sigma}^{(h)}(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) \right| \leq \frac{\epsilon^2}{2},$$

$$\textit{and} \quad \left| \left\langle \mathbf{b}^{(h)}(\boldsymbol{x}^{(1)}), \mathbf{b}^{(h)}(\boldsymbol{x}^{(2)}) \right\rangle - \prod_{h'=h}^{L} \dot{\Sigma}^{(h')}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) \right| < 3L\epsilon.$$

4. Experiments

Experiments on CIFAR-10 dataset

- Two CNN architectures: with and without global average pooling (GAP)
- Also test CNTKs with only 2,000 training data

Experiments on CIFAR-10 dataset

- Two CNN architectures: with and without global average pooling (GAP)
- Also test CNTKs with only 2,000 training data

L	CNN-V	CNTK-V	CNTK-V-2K	CNN-GAP	CNTK-GAP	CNTK-GAP-
3	59.97%	64.47%	40.94%	63.81%	70.47%	49.71%
4	60.20%	65.52%	42.54%	80.93%	75.93%	51.06%
6	64.11%	66.03%	43.43%	83.75%	76.73%	51.73%
11	69.48%	65.90%	43.42%	82.92%	**77.43%**	51.92%
21	75.57%	64.09%	42.53%	83.30%	77.08%	52.22%

- CNN-V represents vanilla CNN and CNTK-V represents the kernel corresponding to CNN-V.
- CNN-GAP represents CNN with GAP and CNTK-GAP represents the kernel corresponding to CNN-GAP.

Experiments on CIFAR-10 dataset

- CNTKs are very powerful kernel, $+10\%$ compared to previous kernels
- For both CNN and CNTK, depth can affect the classification accuracy
- The global average pooling operation can significantly increase the classification accuracy by $8\% - 10\%$ for both CNN and CNT
- **Still 5% - 6% performance gap between CNTKs & CNNs**