



Neural Tangent Kernel: Intro

19 июля-8 августа 2021

**СОВРЕМЕННЫЕ МЕТОДЫ ТЕОРИИ
ИНФОРМАЦИИ, ОПТИМИЗАЦИИ И
УПРАВЛЕНИЯ**

*By Petyushko Alexander
2021-08-03*

Intro: NNGP/NTK/MFT

- **Neural Network Gaussian Process (NNGP)**
 - Explicit connection to Bayesian Neural Nets (BNN)
 - Output of NN with $m \rightarrow \infty$ is approximated with Gaussian Process based on second moment of inputs
 - No any training (and its dynamic) is included
- **Neural Tangent Kernel (NTK):**
 - SGD training dynamics is considered
 - Only linearization regime: weights during process are not changing much
 - Have explicit analytical formulations for the output
 - Have practical proof of concept even for CNN, but for small datasets / shallow networks
- **Mean Field Theory (MFT):**
 - Higher order training dynamics is considered (not only first order Taylor)
 - No any practical solutions for number of layers more than 2 (only existence theorems)

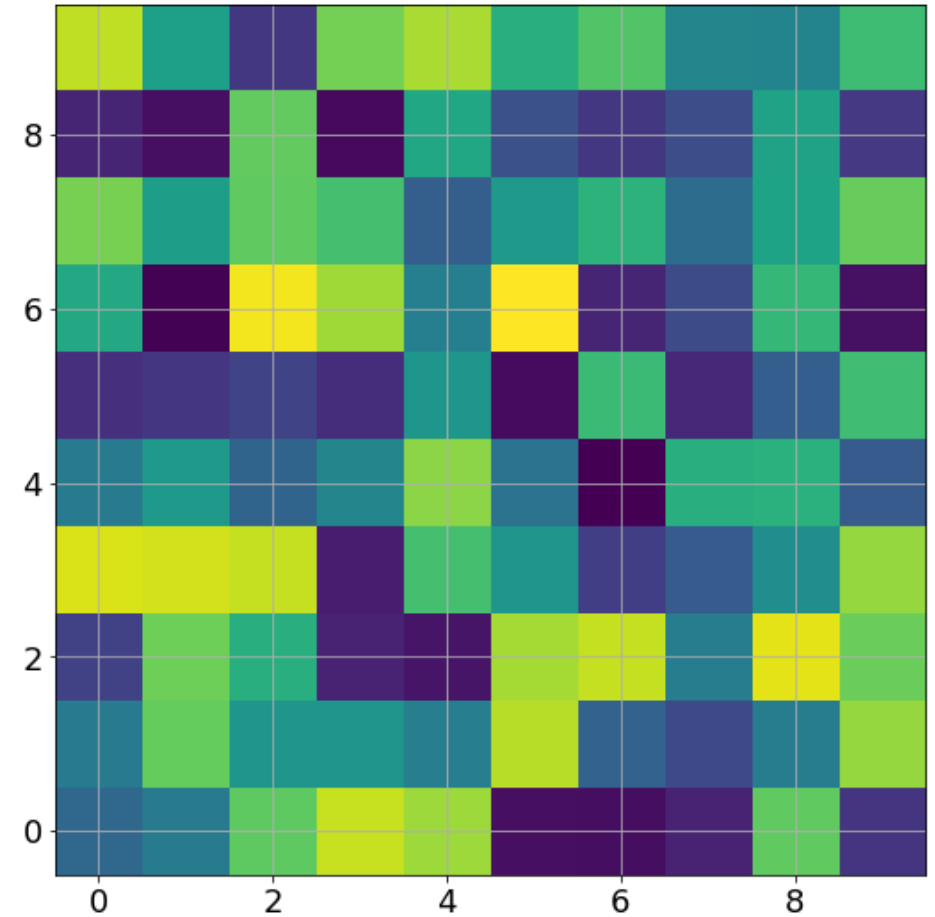
Main points of the original paper¹

- Behavior of DNN during SGD is described by a related **neural tangent kernel (NTK)**
- NTK only depends on the depth of the NN, activation function and initialization variance
- Values of DNN outside the training set are described by NTK
- Behavior of wide DNN is close to the theoretical limit
- **Key property (Lazy training)**: training loss is decreasing to 0 with minimal movement of weights from their initialization (*linear dynamics*)
 - In contrast to “**Mean-field**” regime, where weights move according to *non-linear dynamics*

$$f(x; \theta) \approx f(x; \theta_0) + \langle \theta - \theta_0, \nabla_{\theta} f(x; \theta_0) \rangle$$

Lazy regime

- Dynamics of NN weights during training
 - yes, it is a *gif*¹
 - $W \in \mathbb{R}^{10 \times 10}$



Definitions

- Training set: $(X, Y) = \{(x_i, y_i)\}_{i=1}^N, x \in \mathbb{R}^d$
- Neural Net: $f: \mathbb{R}^d \rightarrow \mathbb{R}^c$ (output is logits), parametrized by weights θ
- $f(x) = (z_1, \dots, z_c)^T$
- Loss: $L: \mathbb{R}^c \rightarrow \mathbb{R}$
- Initialization: i.i.d. Gaussians $N(0,1)$
- Signal propagation on the layer l : Wx is always multiplied by $\frac{1}{\sqrt{n_{l-1}}}$
 - This is not standard parametrization!
 - In standard parametrization $\sigma^2 \sim \frac{1}{n_{l-1}}$, but Wx without any multiplication factor

SGD as the PDE

- SGD process: $\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} L$
- **PDE:** $\partial_t \theta_t = -\nabla_{\theta_t} L = -\partial_z L \times \partial_{\theta_t} f$

Linear regression intuition

- **Regression task:** $L = \frac{1}{N} \sum_{i=1}^N (f(x_i) - y_i)^2$
- **Linear model:** $f(x) = w^T x$
- **Solution to Regression task:** $w = (X^T X)^{-1} X^T Y$
 - As a consequence of $\partial_w L = 0$
- Suppose that we are solving this task by SGD (as PDE)
- **Dynamics** of f : $\partial_t f = -\partial_f L \times \partial_w f = \frac{2}{N} \sum -\partial_w f \times (f - y) \times \partial_w f$
 - For linear regression in matrix form, $\dot{f} = -XX^T(f - Y)$



Arbitrary function f for MSE loss (NTK ODE solution)

- Linear Regression

- $w = (X^T X)^{-1} X^T Y$
 - $\dot{f} = -XX^T(f - Y)$

- For arbitrary function f we'll have $\dot{f} = -\nabla_{\theta} f \nabla_{\theta} f^T (f - Y) = -\Theta(f - Y)$, where $\Theta = \nabla_{\theta} f \nabla_{\theta} f^T$

- And the solution is $f(X) = e^{-\Theta t}(f_0(X) - Y) + Y$

- For arbitrary point x , if $w = \theta - \theta_0$:

- $f(x) \approx f_0(x) + \nabla_{\theta} f_0(x)(\theta - \theta_0)$, $\dot{w} = -\nabla_{\theta} f_0(X)^T (f_0(X) + \nabla_{\theta} f_0(X)w - Y)$

- $\Rightarrow w = -\nabla_{\theta} f_0(X)^T \Theta^{-1}(I - e^{-\Theta t})(f_0(X) - Y)$, $\lim_{t \rightarrow \infty} w_t = -\nabla_{\theta} f_0(X)^T \Theta^{-1}(f_0(X) - Y)$

- $\Rightarrow f(x) \approx f_0(x) + \nabla_{\theta} f_0(x)w$,

- $\Rightarrow \lim_{t \rightarrow \infty} f_t(x) = f_0(x) - \nabla_{\theta} f_0(x) \nabla_{\theta} f_0(X)^T \Theta^{-1}(f_0(X) - Y) = \nabla_{\theta} f_0(x) \nabla_{\theta} f_0(X)^T \Theta^{-1} Y$,

- if we assume $f_0(X) = 0$

Remember it

NTK: iterative solution¹

- Taylor decomposition: $f(\theta, x) \approx f(\theta_0, x) + \langle \nabla_{\theta} f(\theta_0, x), \theta - \theta_0 \rangle$, where $\theta_0 = \theta(0)$
- Let $u_i(t) = f(\theta_t, x_i)$, where $\theta_t = \theta(t)$, $\phi(x) = \nabla_{\theta} f(\theta_0, x)$
- Then the loss for regression becomes:
 - $L(\theta) = \frac{1}{2} \sum_{i=1}^N (f(\theta, x_i) - y_i)^2 \approx \frac{1}{2} \sum_{i=1}^N (\langle \phi(x_i), \theta - \theta_0 \rangle + u_i(0) - y_i)^2$
- In combination with SGD updating rule $\theta(k+1) = \theta(k) - \eta \nabla_{\theta} L(\theta(k))$ we can derive:
 - $\theta(k) = \theta_0 - ZH^{-1}(I - (I - \eta H)^k)(u(0) - Y)$, where $Z = (\phi(x_1), \dots, \phi(x_N))$, $H = Z^T Z$
 - If $0 < \eta \leq 1/\|H\|$, then $\lim_{k \rightarrow \infty} \theta(k) = \theta^* = \theta_0 - ZH^{-1}(u(0) - Y)$
 - And the solution is $f(\theta^*, x) \approx f(\theta_0, x) + \langle \nabla_{\theta} f(\theta_0, x), -ZH^{-1}(u(0) - Y) \rangle$
- Suppose that on initialization outputs are close to zero: $f(\theta_0, x_i) = u_i(0) \approx 0$
 - $\Rightarrow \underline{f(\theta^*, x) \approx \phi(x)^T ZH^{-1}Y}$

Reproducing Kernel Hilbert Space (RKHS)¹

- **RKHS:** $f, g \in RKHS$, $\|f - g\|$ is small $\Leftrightarrow |f(x) - g(x)|$ is small for all x
- **Representer theorem:**
 - If positive-definite real-valued kernel $k: X \times X \rightarrow \mathbb{R}$
 - If $f^* = \operatorname{argmin}_{f \in RKHS} H_k \left[\frac{1}{N} \sum L(x_i, y_i, f(x_i)) \right]$,
 - Then $f^*(\cdot) = \sum_{i=1}^N \alpha_i k(\cdot, x_i)$

[1] https://en.wikipedia.org/wiki/Reproducing_kernel_Hilbert_space

[2] https://en.wikipedia.org/wiki/Representer_theorem

NTK: RKHS solution¹

- We can do it more elegantly by using RKHS
- If $f^* = \operatorname{argmin}_{f \in \text{RKHS}} H_k \left[\frac{1}{N} \sum L(x_i, y_i, f(x_i)) \right]$, then by **Representer Th.**² $f^*(\cdot) = \sum_{i=1}^N \alpha_i k(\cdot, x_i)$
 - Then, $L = \|Y - \Theta \alpha\|^2 \rightarrow \min$, where $\Theta_{ij} = k(x_i, x_j) \Rightarrow \alpha = \Theta^{-1} Y$
 - And the solution is $f^*(x) = k(x, X)^T \Theta^{-1} Y$
- In our case $L(\theta) = \frac{1}{2} \sum_{i=1}^N (\langle \phi(x_i), \theta - \theta_0 \rangle - y_i)^2$, it means that we have linearization of f with the kernel $k(x, x') = \langle \phi(x), \phi(x') \rangle$, where $\phi(x) = \nabla_{\theta} f(\theta_0, x)$
 - Note, that $f(x) = \langle f(\cdot), k(\cdot, x) \rangle$ and $k(\cdot, x) = \phi(x)$
 - Positive definiteness: $z[\nabla_{\theta} f \nabla_{\theta} f^T] z^T = [z \nabla_{\theta} f][z \nabla_{\theta} f]^T = \|[z \nabla_{\theta} f]\|^2 \geq 0$
- Then $f^*(x) = k(x, X)^T \Theta^{-1} Y$, where $\Theta = k(X, X) = \left[\nabla_{\theta} f(\theta_0, x_i) \nabla_{\theta} f(\theta_0, x_j)^T \right]_{i,j=1}^N$

Remember!
 $\nabla_{\theta} f_0(x) \nabla_{\theta} f_0(X)^T \Theta^{-1} Y$
- The same as $f(\theta^*, x) = \phi(x)^T Z H^{-1} Y$, where $\phi(x)^T Z = \phi(x)^T (\phi(x_1), \dots, \phi(x_N)) = k(x, X)$, and $H = Z^T Z = \Theta$

NTK

- Define $\Theta(t): \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^c \times \mathbb{R}^c$ as $\mathbb{E}_{\theta}[\partial_{\theta_t} f(x) \partial_{\theta_t} f(x')^T]$
- **NB:** This is not Hessian Matrix! $H = \left[\frac{\partial^2 f}{\partial \theta_i \partial \theta_j} \right]_{i,j}$
- **Theorem1:** when the width of the layers tend to infinity $n_l \rightarrow \infty$, then $\Theta(0) \rightarrow \Theta_{\infty}$, and this Θ_{∞} **depends only on:**
 - DNN depth
 - Non-linearity σ
 - Variance of the initialization of θ
- **Theorem2:** for any t , when the width of the layers tend to infinity $n_l \rightarrow \infty$, then $\Theta(t) \rightarrow \Theta(0)$

NTK: formulas

$$\Sigma^{(1)}(x, x') = \frac{1}{n_0} x^T x' + \beta^2$$

$$\Sigma^{(L+1)}(x, x') = \mathbb{E}_{f \sim \mathcal{N}(0, \Sigma^{(L)})} [\sigma(f(x)) \sigma(f(x'))] + \beta^2$$

$$\Theta_{\infty}^{(1)}(x, x') = \Sigma^{(1)}(x, x')$$

$$\Theta_{\infty}^{(L+1)}(x, x') = \Theta_{\infty}^{(L)}(x, x') \dot{\Sigma}^{(L+1)}(x, x') + \Sigma^{(L+1)}(x, x'),$$

$$\dot{\Sigma}^{(L+1)}(x, x') = \mathbb{E}_{f \sim \mathcal{N}(0, \Sigma^{(L)})} [\dot{\sigma}(f(x)) \dot{\sigma}(f(x'))],$$

Note:

- Variation during training of individual activations in the hidden layers shrinks as their width grows
- Overall variation of activations is significant, which allows the parameters of the lower layers to learn

Specific bounds for NTK

- Previously we have seen only limit theorems
- Later¹ it was expanded to more useful estimations layer width (here L means number of layers in MLP):

- *Initialization:*

Theorem 3.1 (Convergence to the NTK at initialization). Fix $\epsilon > 0$ and $\delta \in (0, 1)$. Suppose $\sigma(z) = \max(0, z)$ and $\min_{h \in [L]} d_h \geq \Omega(\frac{L^{14}}{\epsilon^4} \log(L/\delta))$. Then for any inputs $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{d_0}$ such that $\|\mathbf{x}\| \leq 1, \|\mathbf{x}'\| \leq 1$, with probability at least $1 - \delta$ we have:

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

Note: $\epsilon \sim d^{-\frac{1}{4}}$

- *Training convergence* (need some small positive multiplier to have the initial output near 0; here $n=N$):

$$f_{nn}(\boldsymbol{\theta}, \mathbf{x}) = \kappa f(\boldsymbol{\theta}, \mathbf{x})$$

$$f_{nn}(\mathbf{x}_{te}) = \lim_{t \rightarrow \infty} f_{nn}(\boldsymbol{\theta}(t), \mathbf{x}_{te})$$

$$f_{ntk}(\mathbf{x}_{te}) = (\ker_{ntk}(\mathbf{x}_{te}, \mathbf{X}))^\top (\mathbf{H}^*)^{-1} \mathbf{y}$$

Theorem 3.2 (Equivalence between trained net and kernel regression). Suppose $\sigma(z) = \max(0, z)$, $1/\kappa = \text{poly}(1/\epsilon, \log(n/\delta))$ and $d_1 = d_2 = \dots = d_L = m$ with $m \geq \text{poly}(1/\kappa, L, 1/\lambda_0, n, \log(1/\delta))$. Then for any $\mathbf{x}_{te} \in \mathbb{R}^d$ with $\|\mathbf{x}_{te}\| = 1$, with probability at least $1 - \delta$ over the random initialization, we have

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

CNTK¹

- Let input image of size $P \times Q$, C_l - the number of channels on layer l , convolutional filters of size $q \times q$, c_σ - variance of $\sigma(x)$
- Can have the **GAP** layer at the end, but **no MaxPooling** layers

CNTK formula. We let \mathbf{x}, \mathbf{x}' be two input images.

- For $\alpha = 1, \dots, C^{(0)}$, $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$, define

$$\mathbf{K}_{(\alpha)}^{(0)}(\mathbf{x}, \mathbf{x}') = \mathbf{x}_{(\alpha)} \otimes \mathbf{x}'_{(\alpha)} \text{ and } \left[\boldsymbol{\Sigma}^{(0)}(\mathbf{x}, \mathbf{x}') \right]_{ij, i' j'} = \sum_{\alpha=1}^{C^{(0)}} \text{tr} \left(\left[\mathbf{K}_{(\alpha)}^{(0)}(\mathbf{x}, \mathbf{x}') \right]_{\mathcal{D}_{ij, i' j'}} \right).$$

- For $h \in [L]$,
 - For $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$, define

$$\boldsymbol{\Lambda}_{ij, i' j'}^{(h)}(\mathbf{x}, \mathbf{x}') = \begin{pmatrix} \left[\boldsymbol{\Sigma}^{(h-1)}(\mathbf{x}, \mathbf{x}) \right]_{ij, ij} & \left[\boldsymbol{\Sigma}^{(h-1)}(\mathbf{x}, \mathbf{x}') \right]_{ij, i' j'} \\ \left[\boldsymbol{\Sigma}^{(h-1)}(\mathbf{x}', \mathbf{x}) \right]_{i' j', ij} & \left[\boldsymbol{\Sigma}^{(h-1)}(\mathbf{x}', \mathbf{x}') \right]_{i' j', i' j'} \end{pmatrix} \in \mathbb{R}^{2 \times 2}.$$

- Define $\mathbf{K}^{(h)}(\mathbf{x}, \mathbf{x}'), \dot{\mathbf{K}}^{(h)}(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^{P \times Q \times P \times Q}$: for $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$,

$$\left[\mathbf{K}^{(h)}(\mathbf{x}, \mathbf{x}') \right]_{ij, i' j'} = \frac{c_\sigma}{q^2} \cdot \mathbb{E}_{(u, v) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_{ij, i' j'}^{(h)}(\mathbf{x}, \mathbf{x}'))} [\sigma(u) \sigma(v)], \quad (11)$$

$$\left[\dot{\mathbf{K}}^{(h)}(\mathbf{x}, \mathbf{x}') \right]_{ij, i' j'} = \frac{c_\sigma}{q^2} \cdot \mathbb{E}_{(u, v) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_{ij, i' j'}^{(h)}(\mathbf{x}, \mathbf{x}'))} [\dot{\sigma}(u) \dot{\sigma}(v)]. \quad (12)$$

- Define $\boldsymbol{\Sigma}^{(h)}(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^{P \times Q \times P \times Q}$: for $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$,

$$\left[\boldsymbol{\Sigma}^{(h)}(\mathbf{x}, \mathbf{x}') \right]_{ij, i' j'} = \text{tr} \left(\left[\mathbf{K}^{(h)}(\mathbf{x}, \mathbf{x}') \right]_{\mathcal{D}_{ij, i' j'}} \right).$$

1. First, we define $\boldsymbol{\Theta}^{(0)}(\mathbf{x}, \mathbf{x}') = \boldsymbol{\Sigma}^{(0)}(\mathbf{x}, \mathbf{x}')$.
2. For $h = 1, \dots, L-1$ and $(i, j, i', j') \in [P] \times [Q] \times [P] \times [Q]$, we define

$$\left[\boldsymbol{\Theta}^{(h)}(\mathbf{x}, \mathbf{x}') \right]_{ij, i' j'} = \text{tr} \left(\left[\dot{\mathbf{K}}^{(h)}(\mathbf{x}, \mathbf{x}') \odot \boldsymbol{\Theta}^{(h-1)}(\mathbf{x}, \mathbf{x}') + \mathbf{K}^{(h)}(\mathbf{x}, \mathbf{x}') \right]_{\mathcal{D}_{ij, i' j'}} \right).$$

3. For $h = L$, we define $\boldsymbol{\Theta}^{(L)}(\mathbf{x}, \mathbf{x}') = \dot{\mathbf{K}}^{(L)}(\mathbf{x}, \mathbf{x}') \odot \boldsymbol{\Theta}^{(L-1)}(\mathbf{x}, \mathbf{x}') + \mathbf{K}^{(L)}(\mathbf{x}, \mathbf{x}')$.
4. The final CNTK value is defined as $\text{tr}(\boldsymbol{\Theta}^{(L)}(\mathbf{x}, \mathbf{x}'))$.

CNTK: results

- Still **5-6% performance gap** between best CNTK and best CNN
- For some depths, CNTK provides better results than CNN

Depth	CNN-V	CNTK-V	CNTK-V-2K	CNN-GAP	CNTK-GAP	CNTK-GAP-2K
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%

Table 1: Classification accuracies of CNNs and CNTKs on the CIFAR-10 dataset. 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. CNTK-V-2K and CNTK-GAP-2K represent training CNTKs with only 2,000 training data.

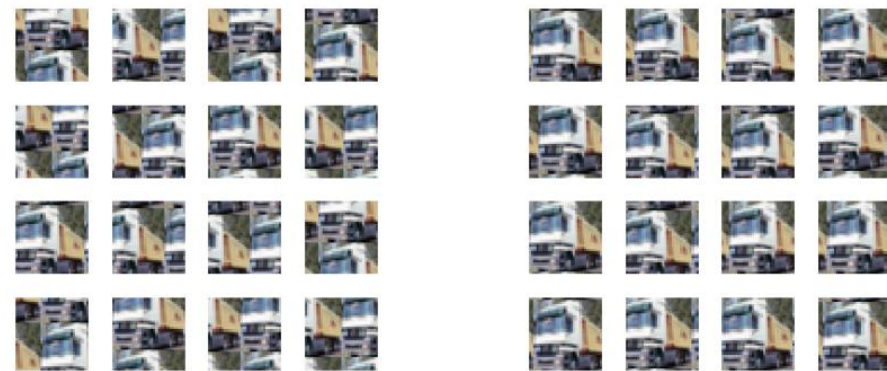
CNTK: augmentation added¹

- CNTK+GAP is equivalent to CNTK with full translation data augmentation with wrap-around at the boundary, but such a translation could produce unrealistic images => Local Average Pooling (LAP) layer was proposed (small translation, e.g. by a few pixels)
- Instead of input augmentation, **augmented kernel** (under group a group G) is proposed: $\mathbf{K}^{\mathcal{G}}(x, x') = \mathbb{E}_{g \in \mathcal{G}} \mathbf{K}(g(x), x') \implies \Theta_{\text{GAP}}(x, x') = \frac{1}{PQ} \Theta_{\text{FC}}^{\mathcal{G}}(x, x') \quad \mathcal{G} = \{\mathcal{T}_{i,j}\}_{(i,j) \in [P] \times [Q]}$
- **Equivariant** kernel under a group G : $\forall g \in G \Rightarrow K(g(x), g(x')) = K(x, x')$

Theorem 4.1. Given a group \mathcal{G} and a kernel \mathbf{K} such that \mathbf{K} is equivariant under \mathcal{G} , then the prediction of augmented kernel $\mathbf{K}^{\mathcal{G}}$ with dataset (\mathbf{X}, \mathbf{y}) is equal to that of kernel \mathbf{K} and augmented dataset $(\mathbf{X}_{\mathcal{G}}, \mathbf{y}_{\mathcal{G}})$. Namely, for any $\mathbf{x}' \in \mathbb{R}^{P \times Q \times C}$, $\sum_{i=1}^N \alpha_i \mathbf{K}^{\mathcal{G}}(\mathbf{x}', \mathbf{x}_i) = \sum_{i \in [N], g \in \mathcal{G}} \tilde{\alpha}_{i,g} \mathbf{K}(\mathbf{x}', g(\mathbf{x}_i))$ where $\alpha = (\mathbf{K}_{\mathbf{X}}^{\mathcal{G}})^{-1} \mathbf{y}$, $\tilde{\alpha} = (\mathbf{K}_{\mathbf{X}_{\mathcal{G}}})^{-1} \mathbf{y}_{\mathcal{G}}$.

Corollary 4.1. For $\mathcal{G} = \{\mathcal{T}_{i,j}\}_{(i,j) \in [P] \times [Q]}$, for any given dataset D , the prediction of Σ_{GAP} (or Θ_{GAP}) with dataset D is equal to the prediction of Σ_{FC} (or Θ_{FC}) with augmented dataset $D_{\mathcal{T}}$.

CNTK+LAP results



(a) GAP

(b) LAP with $c = 4$

$$\mathcal{T}_{\Delta_i, \Delta_j} = (\Delta_i, \Delta_j) \in [-c, c] \times [-c, c]$$

- **+4%** in comparison to CNTK+GAP w/o preprocessing
- **+11%** in comparison to CNTK+GAP w/ preprocessing
 - Matching AlexNet on CIFAR-10
- **Now the strongest classifier which is not the trained CNN**

$c \backslash d$	5	8	11	14
0	66.55 (69.87)	66.27 (69.87)	65.85 (69.37)	65.47 (68.90)
4	77.06 (79.08)	77.14 (78.96)	77.06 (78.98)	76.52 (78.74)
8	79.24 (80.95)	79.25 (81.03)	78.98 (80.94)	78.65 (80.35)
12	80.11 (81.34)	79.79 (81.28)	79.29 (81.14)	79.13 (80.91)
16	79.80 (81.21)	79.71 (81.40)	79.74 (81.09)	79.42 (81.00)
20	79.24 (80.67)	79.27 (80.88)	79.30 (80.76)	78.92 (80.39)
24	78.07 (79.88)	78.16 (79.79)	78.14 (80.06)	77.87 (80.07)
28	76.91 (78.69)	77.33 (79.20)	77.65 (79.56)	77.65 (79.74)
32	76.79 (78.53)	77.39 (79.13)	77.63 (79.51)	77.63 (79.74)

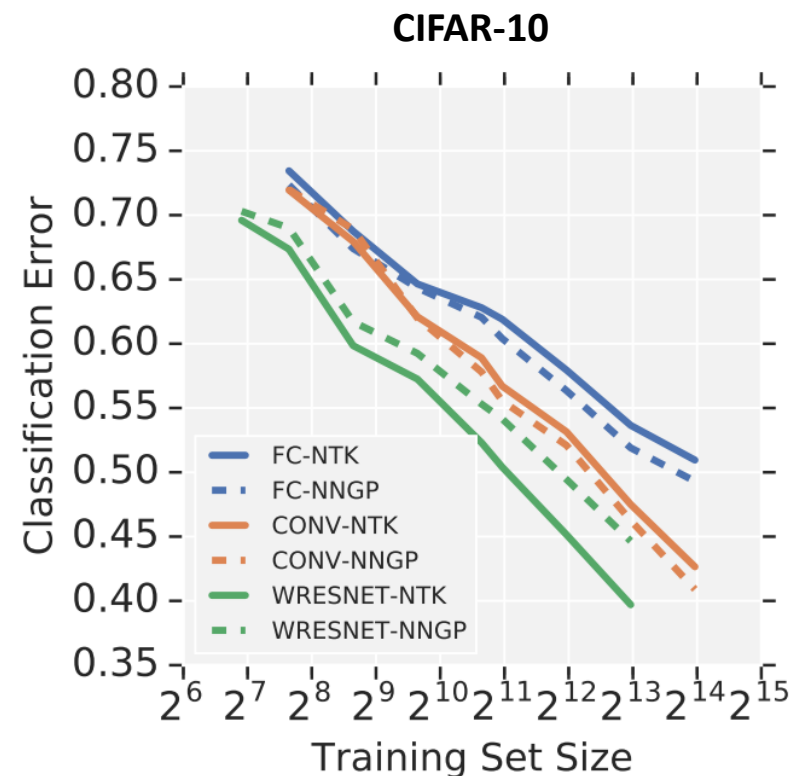
Table 1: Test accuracy of CNTK on CIFAR-10.

$c \backslash d$	5	8	11	14
4	84.63 (86.64)	84.07 (86.23)	83.29 (85.53)	82.57 (84.81)
8	86.36 (88.32)	85.80 (87.81)	85.01 (87.08)	84.57 (86.53)
12	86.74 (88.35)	86.20 (87.90)	85.60 (87.36)	84.95 (86.99)
16	86.77 (88.36)	86.17 (87.85)	85.60 (87.44)	84.92 (86.98)
20	86.17 (87.77)	85.71 (87.50)	85.14 (87.07)	84.59 (86.84)

Table 5: Test accuracy of additional pre-processing + CNTK on CIFAR-10.

Comparison of architectures under NTK regime

- FC < Vanilla CONV < ResNet
- NTK is more precise than NNGP
- Vanilla CONV is better than FC
- ResNet is better than Vanilla CONV



NTK and NNGP connection

- **Tl;dr** linearized NN behaves as the Gaussian dependent on the NTK Θ

$$f_t^{\text{lin}}(x) \equiv f_0(x) + \nabla_{\theta} f_0(x)|_{\theta=\theta_0} \omega_t$$

$$\mathcal{K}^{i,j}(x, x') = \lim_{\min(n_1, \dots, n_L) \rightarrow \infty} \mathbb{E} \left[f_0^i(x) \cdot f_0^j(x') \right]$$

Corollary 1. For every test points in $x \in \mathcal{X}_T$, and $t \geq 0$, $f_t^{\text{lin}}(x)$ converges in distribution as width goes to infinity to a Gaussian with mean and covariance given by⁹

$$\mu(\mathcal{X}_T) = \Theta(\mathcal{X}_T, \mathcal{X}) \Theta^{-1} \left(I - e^{-\eta \Theta t} \right) \mathcal{Y}, \quad (14)$$

$$\begin{aligned} \Sigma(\mathcal{X}_T, \mathcal{X}_T) = & \mathcal{K}(\mathcal{X}_T, \mathcal{X}_T) + \Theta(\mathcal{X}_T, \mathcal{X}) \Theta^{-1} \left(I - e^{-\eta \Theta t} \right) \mathcal{K} \left(I - e^{-\eta \Theta t} \right) \Theta^{-1} \Theta(\mathcal{X}, \mathcal{X}_T) \\ & - \left(\Theta(\mathcal{X}_T, \mathcal{X}) \Theta^{-1} \left(I - e^{-\eta \Theta t} \right) \mathcal{K}(\mathcal{X}, \mathcal{X}_T) + h.c. \right). \end{aligned} \quad (15)$$

h.c. means “Hermitian conjugate”

Therefore, over random initialization, $\lim_{t \rightarrow \infty} \lim_{n \rightarrow \infty} f_t^{\text{lin}}(x)$ has distribution

$$\begin{aligned} & \mathcal{N} \left(\Theta(\mathcal{X}_T, \mathcal{X}) \Theta^{-1} \mathcal{Y}, \right. \\ & \left. \mathcal{K}(\mathcal{X}_T, \mathcal{X}_T) + \Theta(\mathcal{X}_T, \mathcal{X}) \Theta^{-1} \mathcal{K} \Theta^{-1} \Theta(\mathcal{X}, \mathcal{X}_T) - \left(\Theta(\mathcal{X}_T, \mathcal{X}) \Theta^{-1} \mathcal{K}(\mathcal{X}, \mathcal{X}_T) + h.c. \right) \right). \end{aligned} \quad (16)$$

NTK: another estimations

- **Tl;dr** linearized NN behaves as the Gaussian dependent on the NTK Θ

Theorem 2.1 (Informal). *Let $n_1 = \dots = n_L = n$ and assume $\lambda_{\min}(\Theta) > 0$. Applying gradient descent with learning rate $\eta < \eta_{\text{critical}}$ (or gradient flow), for every $x \in \mathbb{R}^{n_0}$ with $\|x\|_2 \leq 1$, with probability arbitrarily close to 1 over random initialization,*

$$\sup_{t \geq 0} \|f_t(x) - f_t^{\text{lin}}(x)\|_2, \sup_{t \geq 0} \frac{\|\theta_t - \theta_0\|_2}{\sqrt{n}}, \sup_{t \geq 0} \|\hat{\Theta}_t - \hat{\Theta}_0\|_F = \mathcal{O}(n^{-\frac{1}{2}}), \text{ as } n \rightarrow \infty. \quad (17)$$

Note, $\epsilon \sim n^{-\frac{1}{2}}$

NTK: what about CE-loss¹

- **Tldr** no close form solution, only analytical ODE. And numerical solving (disaster...)

$$\ell(f, y) = - \sum_i y^i \log \sigma(f^i), \quad \sigma(f^i) \equiv \frac{\exp(f^i)}{\sum_j \exp(f^j)} \quad \frac{\partial \ell}{\partial \hat{y}^i} = \sigma(\hat{y}^i) - y^i$$

$$\begin{aligned} \dot{f}_t^i(x) &= \nabla_{\theta} f_t^i(x) \frac{d\theta}{dt} = -\eta \nabla_{\theta} f_t^i(x) \sum_j \sum_{(z,y) \in \mathcal{D}} \left[\nabla_{\theta} f_t^j(z)^T \frac{\partial \ell(f_t, y)}{\partial \hat{y}^j} \right] \\ &= -\eta \sum_{(z,y) \in \mathcal{D}} \sum_j \nabla_{\theta} f_t^i(x) \nabla_{\theta} f_t^j(z)^T \left(\sigma(f_t^j(z)) - y^j \right) \end{aligned}$$

Let $\hat{\Theta}^{ij}(x, \mathcal{X}) = \nabla_{\theta} f^i(x) \nabla_{\theta} f^j(\mathcal{X})^T$. The above is

$$\begin{aligned} \dot{f}_t(\mathcal{X}) &= -\eta \hat{\Theta}_t(\mathcal{X}, \mathcal{X}) (\sigma(f_t(\mathcal{X})) - \mathcal{Y}) \\ \dot{f}_t(x) &= -\eta \hat{\Theta}_t(x, \mathcal{X}) (\sigma(f_t(\mathcal{X})) - \mathcal{Y}) \end{aligned}$$

NTK: about parametrization

- **Tldr** NTK parametrization can be standardized!

Parameterization	Standard (naive)	NTK	Standard (improved)
Layer equation, $x^{l+1} =$	$W^l x^l + b^l$	$\frac{\sigma_w}{\sqrt{sN^l}} W^l x^l + \sigma_b b^l$	$\frac{1}{\sqrt{s}} W^l x^l + b$
Weight shape, $W^l \in$	$\mathcal{R}^{sN^{l+1} \times sN^l}$		
W initialization, $W_{ij}^l \sim$	$\mathcal{N}\left(0, \frac{\sigma_w^2}{sN^l}\right)$	$\mathcal{N}(0, 1)$	$\mathcal{N}\left(0, \frac{\sigma_w^2}{N^l}\right)$
b initialization, $b_i^l \sim$	$\mathcal{N}(0, \sigma_b^2)$	$\mathcal{N}(0, 1)$	$\mathcal{N}(0, \sigma_b^2)$
NTK, $s \rightarrow \infty, \Theta^{l+1} =$	diverges	$\sigma_w^2 K^l + \sigma_b^2 + \sigma_w^2 \Theta^l$	$N^l K^l + 1 + \sigma_w^2 \Theta^l$

- But why we needed this factor $\frac{1}{\sqrt{m}}$ anyway?

NTK: about parametrization

- But why we needed this factor $\frac{1}{\sqrt{m}}$ anyway?
- Suppose output of some layer is $z = z(X, \theta) \in \mathbb{R}^{d \times m}$, where d =input dimension, m =layer width
- NNGP kernel $K_z = \mathbb{E}_\theta [z_i z_i^T]$, NTK kernel $\Theta_z = \mathbb{E}_\theta \left[\frac{\partial z_i}{\partial \theta} \frac{\partial z_i^T}{\partial \theta} \right]$, for i -th neuron: $z_i \in \mathbb{R}^d$
- Let $y = \frac{1}{\sqrt{m}} z W$, where $W \in \mathbb{R}^{m \times m}$, $y \in \mathbb{R}^{d \times m}$
- Then $K_y = \mathbb{E}_{W, \theta} [y_i y_i^T] = \frac{1}{m} \mathbb{E}_{W, \theta} [z_i W W^T z_i^T]$, by i.i.d $N(0,1)$: $\frac{1}{m} \mathbb{E}_W [W W^T] = I$
- $\Rightarrow K_y = \mathbb{E}_\theta [z_i z_i^T] = K_z$, and **no dependency on $m \rightarrow \infty$!**
- The same calculations for $\Theta_y = K_z + \Theta_z$
- Note:
 - NTK regime: $\dot{\theta} \rightarrow 0, \Theta = \text{const}$ if $m \rightarrow \infty$
 - Std regime: $\dot{\theta} \neq 0, \Theta \rightarrow \infty$ if $m \rightarrow \infty$

NTK: about classification

- **How to use MSE loss for classification?**
- **Approach 1:** one-hot encoding $y = (0, \dots, 1, \dots, 0)$
- **Approach 2:** zero-centered one-hot encoding $y = (-\frac{1}{C}, \dots, \frac{C-1}{C}, \dots, -\frac{1}{C})$, where C =number of classes
- And then.. MSE loss!

NTK: sort of Lipschitz smoothness property¹

- NTK has the Holder smoothness property
- Let $\Theta(x, x') = \langle \Phi_n(x), \Phi_n(x') \rangle$, where $n = L$ (number of layers)
- Then Holder smoothness property is:

Lemma 11 (Norm and smoothness of Φ_n). *We have $\|\Phi_n(x)\| \leq \sqrt{n+1}\|x\|$, and*

$$\|\Phi_n(x) - \Phi_n(x')\| \leq (n+1)\|x - x'\| + O(n^{5/4})\sqrt{\|x\|\|x - x'\|}.$$

NTH¹: Neural Tangent Hierarchy

- NTK vs DNN: about 5% GAP²
 - Because in fact NTK is changing along training due to the finite width effect
- Solution: **NTH** - infinite hierarchy of ODE, which can grasp the learning data-dependent features
- NTH is (here $n = N$ the number of samples and $K^{(2)} = \Theta$):

Theorem 2.3. Under Assumptions 2.1 and 2.2, there exists an infinite family of operators $K_t^{(r)} : \mathcal{X}^r \mapsto \mathbb{R}$ for $r \geq 2$, the continuous time gradient descent dynamic is given by an infinite hierarchy of ordinary differential equations, i.e., the NTH,

$$\partial_t(f_\alpha(t) - y_\alpha) = -\frac{1}{n} \sum_{\beta=1}^n K_t^{(2)}(x_\alpha, x_\beta)(f_\beta(t) - y_\beta), \quad (2.1)$$

and for any $r \geq 2$,

$$\partial_t K_t^{(r)}(x_{\alpha_1}, x_{\alpha_2}, \dots, x_{\alpha_r}) = -\frac{1}{n} \sum_{\beta=1}^n K_t^{(r+1)}(x_{\alpha_1}, x_{\alpha_2}, \dots, x_{\alpha_r}, x_\beta)(f_\beta(t) - y_\beta). \quad (2.2)$$

There exists a deterministic family (independent of m) of operators $\mathfrak{K}^{(r)} : \mathcal{X}^r \mapsto \mathbb{R}$ for $2 \leq r \leq p+1$ and $\mathfrak{K}^{(r)} = 0$ if r is odd, such that with high probability with respect to the random initialization, there exist some constants $\mathbb{C}, \mathbb{C}' > 0$ such that

$$\left\| K_0^{(r)} - \frac{\mathfrak{K}^{(r)}}{m^{r/2-1}} \right\|_\infty \lesssim \frac{(\ln m)^{\mathbb{C}}}{m^{(r-1)/2}}, \quad (2.3)$$

and for $0 \leq t \leq m^{\frac{p}{2(p+1)}} / (\ln m)^{\mathbb{C}'}$,

$$\|K_t^{(r)}\|_\infty \lesssim \frac{(\ln m)^{\mathbb{C}}}{m^{r/2-1}}. \quad (2.4)$$

NTH¹: Neural Tangent Hierarchy

- In [2] it has been experimentally noted that change of NTK is about $O(\frac{1}{m})$, where m is the layer width, while by theoretical estimations it should be $O(\frac{1}{\sqrt{m}})$
- Here we have the theoretical proof of this experimental fact

Corollary 2.4. *Under Assumptions 2.1 and 2.2, the NTK $K_t^{(2)}(\cdot, \cdot)$ varies at a rate of order $O(1/m)$: with high probability with respect to the random initialization, there exist some constants $C, C' > 0$ such that for $0 \leq t \leq m^{\frac{p}{2(p+1)}} / (\ln m)^{C'}$, it holds*

$$\|\partial_t K_t^{(2)}\|_\infty \lesssim \frac{(1+t)(\ln m)^C}{m}.$$

- Convergence guarantees (L -number of layers) if $m = O(n^3 2^{O(L)})$:

$$\sum_{\beta=1}^n (f_\beta(t) - y_\beta)^2 \lesssim n e^{-\frac{\lambda t}{2n}}$$

[1] Huang, J., & Yau, H. T. (2019). Dynamics of deep neural networks and neural tangent hierarchy.

[2] Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., & Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent.

NTK: Finite-width NN and NTK change¹

- More rigorous estimation of change of NTK in form of $O(\frac{1}{\sqrt{m}})$
- **Theorem 1:** If $\frac{L}{m} > 0$, then $\Theta_0(x, x)$ is not approximately deterministic at initialization: $\frac{\mathbb{E}[\Theta_0(x, x)]^2}{\mathbb{E}[\Theta_0(x, x)^2]} \sim e^{\frac{5L}{m}} (1 + O(\frac{L}{m^2}))$
- **Theorem 2:** If $\frac{L^2}{md} > 0$, then $\Theta_t(x, x)$ is evolving during training: $\frac{\mathbb{E}[\partial_t \Theta_t(x, x)]^2}{\mathbb{E}[\Theta_t(x, x)^2]} \sim \frac{L^2}{md} e^{\frac{5L}{m}} (1 + O(\frac{L}{m^2}))$

NTK and ResNet¹

- Previous theoretical results for ResNets:
 - No spurious local optima (all local extremums are close to global)
 - Training requires weaker conditions compared with MLP
- More tight requirements on the width of the layer (exponential => polynomial):
 - Convergence guarantees $m = O(n^3 L^2)$: $\sum_{\beta=1}^n (f_{\beta}(t) - y_{\beta})^2 \lesssim n e^{-\frac{\lambda t}{2n}}$
 - => For ResNet we can build much more layers (and it is really so!) than for usual DNN

[1] Li, Y., Luo, T., & Yip, N. K. (2020). Towards an Understanding of Residual Networks Using Neural Tangent Hierarchy (NTH).

[2] Hardt, M., & Ma, T. (2016). Identity matters in deep learning.

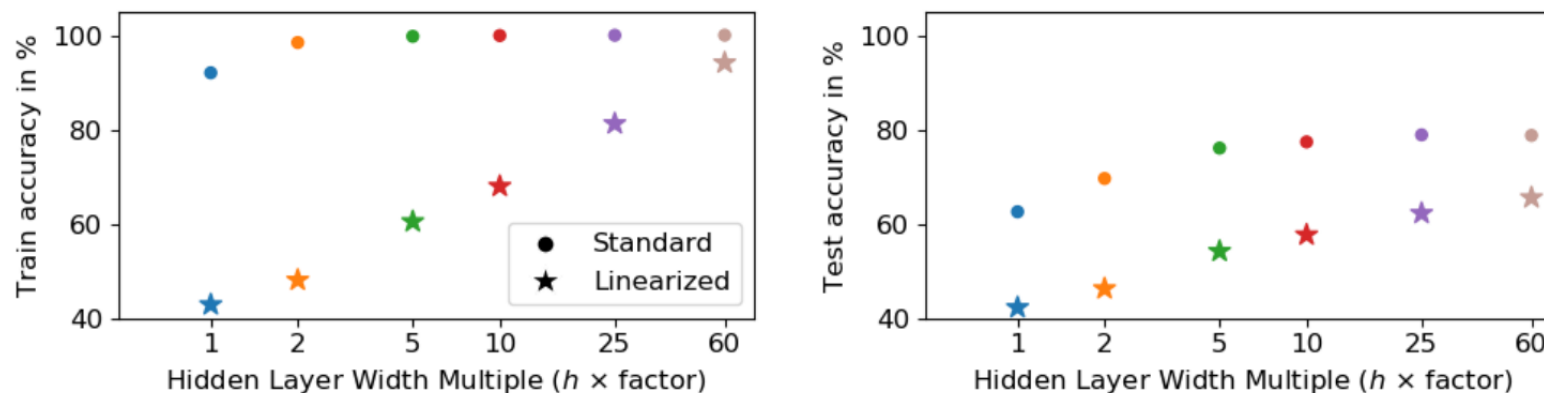
[3] Du, S., Lee, J., Li, H., Wang, L., & Zhai, X. (2019, May). Gradient descent finds global minima of deep neural networks.

NTK: real current gap with finite width NN¹

- Finite width matters

Architecture	Dataset	Standard	Linearized	Generalization gap
LeNet	MNIST	99.2	93.2	6.0
LeNet×60	MNIST	99.4	98.9	0.5
LeNet	CIFAR-10	62.5	42.3	20.2
LeNet×60	CIFAR-10	78.8	65.6	13.2
VGG-11×6 [3]	CIFAR-10	89.7	61.7*	28.0
ResNet-18×8 [3]	CIFAR-10	91.0	56.7*	34.3
21 layer CNN-V [1]	CIFAR-10	75.6	64.1 [†]	11.5
21 layer CNN-GAP [1]	CIFAR-10	83.3	77.1 [†]	6.2
AlexNet	ImageNet (subset)	53.8	35.2	18.6
AlexNet×4	ImageNet (subset)	57.0	39.2	17.8

CIFAR-10, LeNet



NTK: training time prediction¹

- We can estimate loss for the lazy training regime using NTK:

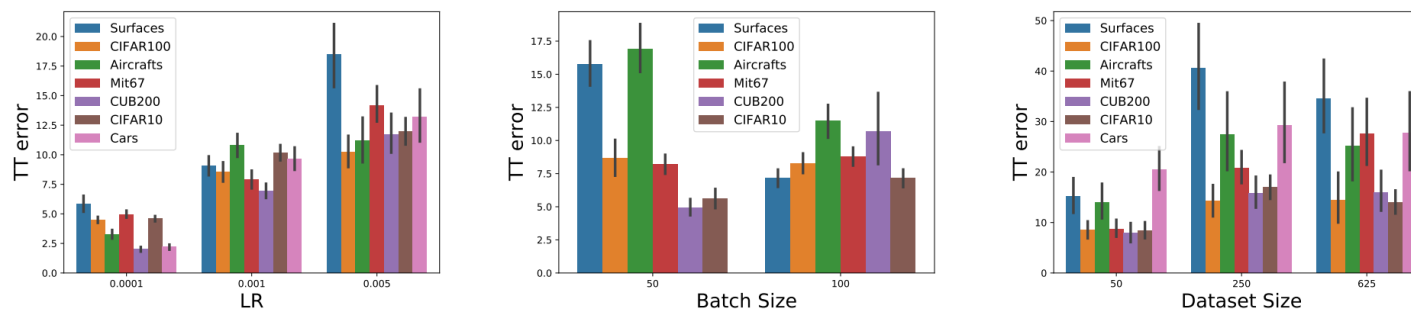
$$L_t = (\mathcal{Y} - f_0(\mathcal{X}))^T e^{-2\eta\Theta t} (\mathcal{Y} - f_0(\mathcal{X}))$$

- And even more efficiently estimate Θ

Proposition 2 Let $S = \nabla_w f_w(\mathcal{X})^T \nabla_w f_w(\mathcal{X})$ be the second moment matrix of the gradients and let $S = U\Sigma U^T$ be the uncentered PCA of the gradients, where $\Sigma = \text{diag}(\lambda_1, \dots, \lambda_n, 0, \dots, 0)$ is a $D \times D$ diagonal matrix, $n \leq \min(N, D)$ is the rank of S and λ_i are the eigenvalues sorted in descending order. Then we have

$$L_t = \sum_{k=1}^D e^{-2\eta\lambda_k t} (\delta\mathbf{y} \cdot \mathbf{v}_k)^2, \quad (5)$$

where $\lambda_k \mathbf{v}_k = (g_i \cdot \mathbf{u}_k)_{i=1}^N$ is the N -dimensional vector containing the value of the k -th principal component of gradients g_i and $\delta\mathbf{y} := \mathcal{Y} - f_0(\mathcal{X})$.



NTK: text processing and attention¹

- NTK with attentions layers
 - For CV and text processing tasks

Attention layer:

queries $Q(x) := xW^Q$, keys $K(x) := xW^K$, and values $V(x) := xW^V$ as usual. The attention layer output is then

$$f(x) := \zeta\left(\frac{Q(x)K(x)^\top}{\sqrt{d}}\right)V(x) = \zeta(G(x))V(x), \quad (1)$$

where ζ is the row-wise softmax function.

Table 1. Overview of the discussed kernels. The d column refers to the d^{-1} and $d^{-1/2}$ scaling of the $Q(x)K(x)^\top$. $(\tilde{\kappa}, \tilde{\Theta})$ denote the input and (κ, Θ) the output NNGP and NTK kernels. NNGP and NTK columns are stated as updates for full $d^s \times d^s$ covariance blocks

KERNEL	d	NNGP	NTK
VANILLA	1	$\zeta(\tilde{\kappa}^{xx})\tilde{\kappa}^{xx'}\zeta(\tilde{\kappa}^{x'x'})^\top$	$2\kappa^{xx'} + \zeta(\tilde{\kappa}^{xx})\tilde{\Theta}^{xx'}\zeta(\tilde{\kappa}^{x'x'})^\top$
	$\frac{1}{2}$	$\tilde{\kappa}^{xx'}\ \tilde{\kappa}^{xx'}\ _F^2$	$4\kappa_{ab}^{xx'} + \langle \tilde{\kappa}^{xx'}, 2\tilde{\kappa}_{ab}^{xx'}\tilde{\Theta}^{xx'} + \tilde{\Theta}_{ab}^{xx'}\tilde{\kappa}^{xx'} \rangle_F$
RANDOM POSITIONAL ENCODING	1	$\zeta(\mathcal{I}_I \circ \tilde{\kappa}^{xx})[\mathcal{I}_I \circ \tilde{\kappa}^{xx'}]\zeta(\mathcal{I}_I \circ \tilde{\kappa}^{x'x'})^\top$	$2\kappa^{xx'} + \zeta(\mathcal{I}_I \circ \tilde{\kappa}^{xx})[\mathcal{I}_I \circ \tilde{\Theta}^{xx'}]\zeta(\mathcal{I}_I \circ \tilde{\kappa}^{x'x'})^\top$
	$\frac{1}{2}$	$\mathcal{I}_I \circ \tilde{\kappa}^{xx'}\ \mathcal{I}_I \circ \tilde{\kappa}^{xx'}\ _F^2$	$4\kappa_{ab}^{xx'} + \langle \mathcal{I}_I \circ \tilde{\kappa}^{xx'}, 2[\mathcal{I}_I \circ \tilde{\kappa}_{ab}^{xx'}]\mathcal{I}_I \circ \tilde{\Theta}^{xx'} + [\mathcal{I}_I \circ \tilde{\Theta}_{ab}^{xx'}]\mathcal{I}_I \circ \tilde{\kappa}^{xx'} \rangle_F$
STRUCTURED POSITIONAL ENCODING	1	$\zeta(\mathcal{I} \circ \tilde{\kappa}^{xx})[\mathcal{I}^\dagger \circ \tilde{\kappa}^{xx'}]\zeta(\mathcal{I} \circ \tilde{\kappa}^{x'x'})^\top$	$2\kappa^{xx'} + \zeta(\mathcal{I} \circ \tilde{\kappa}^{xx})[\mathcal{I}^\dagger \circ \tilde{\Theta}^{xx'}]\zeta(\mathcal{I} \circ \tilde{\kappa}^{x'x'})^\top$
	$\frac{1}{2}$	$\mathcal{I} \circ \tilde{\kappa}^{xx'}\langle \mathcal{I}^\dagger \circ \tilde{\kappa}^{xx'}, \mathcal{I} \circ \tilde{\kappa}^{xx'} \rangle_F$	$4\kappa_{ab}^{xx'} + \langle \mathcal{I}^\dagger \circ \tilde{\kappa}^{xx'}, [\mathcal{I} \circ \tilde{\kappa}_{ab}^{xx'}]\mathcal{I} \circ \tilde{\Theta}^{xx'} + [\mathcal{I} \circ \tilde{\Theta}_{ab}^{xx'}]\mathcal{I} \circ \tilde{\kappa}^{xx'} \rangle_F$ $+ \langle \mathcal{I} \circ \tilde{\kappa}^{xx'}, [\mathcal{I} \circ \tilde{\kappa}_{ab}^{xx'}]\mathcal{I}^\dagger \circ \tilde{\Theta}^{xx'} \rangle_F$
RESIDUAL	–	$\alpha\tilde{\kappa}^{xx'} + (1 - \alpha)R\tilde{\kappa}^{xx'}R^\top$	$2(1 - \alpha)\kappa^{xx'} + \alpha\tilde{\Theta}^{xx'} + (1 - \alpha)R\tilde{\Theta}^{xx'}R^\top$
LAYERNORM	–	$\tilde{\kappa}_{ab}^{xx'}[\tilde{\kappa}_{aa}^{xx}\tilde{\kappa}_{bb}^{xx'}]^{-1/2}$	$\tilde{\Theta}_{ab}^{xx'}[\tilde{\Theta}_{aa}^{xx}\tilde{\Theta}_{bb}^{xx'}]^{-1/2}$

Table 2. CIFAR-10 test accuracies of attention kernels and existing NNGP/NTK alternatives. The standard 50K/10K train/test

KERNEL	NNGP	NTK
FLATTEN	65.54	66.27
GAP (YU ET AL., 2020)	77.85	77.39
LAP (YU ET AL., 2020)	80.36	79.71
STRUCT	80.55	79.93
RESIDUAL	80.72	80.10

Table 3. IMDb sentiment classification, test accuracies of simple NNGP/NTK models on the 25K/25K train/test split using GloVe word embeddings (Pennington et al. (2014); 840B.300d).

KERNEL	NNGP	NTK
GAP-ONLY	–	84.98
GAP-FCN	85.82	85.80
STRUCT	86.09	86.09

Neural Tangents: NTK in Python

- Learning path: JAX¹ => STAX² => Neural Tangents³
 - **JAX**: composable functions transformations (e.g., grad or jit)
 - **STAX**: neural net specification library
 - **NT**: NTK! (+NNGP)

Example of STAX CONV NN definition

```
from neural_tangents import stax

def ConvolutionalNetwork(depth, W_std=1.0, b_std=0.0):
    layers = []
    for _ in range(depth):
        layers += [stax.Conv(1, (3, 3), W_std, b_std, padding='SAME'), stax.Relu()]
    layers += [stax.Flatten(), stax.Dense(1, W_std, b_std)]
    return stax.serial(*layers)
```

[1] <https://github.com/google/jax>

[2] <https://github.com/google/jax/blob/master/jax/experimental/stax.py>

[3] <https://github.com/google/neural-tangents>

Neural Tangents: what can be estimated

NTK+MSE

```
from neural_tangents import stax
init_fn, apply_fn, kernel_fn = stax.serial(stax.Dense(512), stax.Relu(), stax.Dense(10))
from jax.numpy.linalg import inv
y_test = kernel_fn(x_test, x_train).ntk @ inv(kernel_fn(x_train, x_train).ntk) @ y_train
```

```
import neural_tangents as nt
predictor = nt.predict.gradient_descent_mse(kernel_fn(x_train, x_train), y_train,
fx_train, fx_test = predictor(training_time, fx_train, fx_test)
```

NTK+ODE

```
import neural_tangents as nt
predictor = nt.predict.gradient_descent_mse(kernel_fn(x_train, x_train), y_train,
fx_train, fx_test = predictor(training_time, fx_train, fx_test)
```

NTK+Monte-Carlo

```
from jax import random
from jax.experimental import stax
import neural_tangents as nt

init_fn, apply_fn = stax.serial(stax.Dense(64), stax.BatchNorm(), stax.Sigmoid, stax.Dense(1))
kernel_fn = nt.monte_carlo_kernel_fn(init_fn, apply_fn, key=random.PRNGKey(1), n_samples=128)
kernel = kernel_fn(x_train, x_train)
```

Neural Tangents: what is supported and what is not

The following layers do *not* have a known closed-form solution for infinite network covariances, and networks with them have to be estimated empirically (provided with out implementation via `nt.monte_carlo_kernel_fn`) or using other approximations (not currently implemented):

- Sigmoid, Tanh,¹¹ Swish,¹² Softmax, LogSoftMax, Softplus, MaxPool.

The following layers⁷ are currently implemented, with translation rules

- serial
- parallel
- FanOut
- FanInSum
- Dense
- Conv⁸ with arbitrary filter shapes, strides, and padding⁹
- Relu
- LeakyRelu
- Abs
- ABRelu¹⁰,
- Erf
- Identity
- Flatten
- AvgPool
- GlobalAvgPool
- GlobalSelfAttention (Hron et al., 2019)
- LayerNorm







The following is in our near-term plans:

- SumPool
- Dropout
- FanInConcat
- Exp, Elu, Selu, Gelu

Challenging problems

1. Million/Billion-sampled training dataset scale (ImageNet)
2. Online Augmentation of any type (not just translation)
3. Other tasks than just regression / classification
4. CE and other losses (not just MSE)
5. Modern layers (BN, MaxPool) and architectures (Vision Transformer)

Compute requirements of infinite NNs

Architecture → Dataset size ↓	Fully-connected	CNNs	CNNs w/ pooling
$O(100)$			
$O(10,000)$	CIFAR10: $O(0.1)$ GPU-hours	CIFAR10: $O(1)$ GPU-hours	CIFAR10: $O(1000)$ GPU-hours
$O(1,000,000)$			

Thanks!