# Master's Thesis Notes

Shreyas Kalvankar

## Contents



# 1 Formalization of the Neural Tangent Kernel (NTK)

## 1.1 Setup and Assumptions

We consider a supervised learning setting with data $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. A neural network $f : \mathbb{R}^d \times \Theta \to \mathbb{R}$ is parameterized by $\theta \in \Theta \subseteq \mathbb{R}^p$, where $p$ is the number of parameters.

**Assumption 1.1** (Model and training)**.**

1. (*Differentiability*) The network $f(x; \theta)$ is differentiable in $\theta$.
2. (*Random initialization*) We initialize parameters $\theta_0$ i.i.d. with zero mean and variance scaled according to layer input dimensions (e.g. NTK or Xavier/He schemes), so that activations and gradients remain well-behaved as depth/width grow [4, 2, 3].
3. (*Training*) We train $f$ by gradient descent on the squared loss:

$$L(\theta) = \frac{1}{2} \sum_{i=1}^n \big(f(x_i; \theta) - y_i\big)^2.$$

## 1.2 Linearization around Initialization

A first-order Taylor expansion of $f(x; \theta)$ around $\theta_0$ gives

$$f(x; \theta) \;\approx\; f(x; \theta_0) + \nabla_\theta f(x; \theta_0)^\top (\theta - \theta_0).$$

- $f(x; \theta_0)$ is the network output at initialization (a bias term).

- $\phi(x) := \nabla_\theta f(x;\theta_0)$ is the feature vector induced at initialization.

Thus, locally, the network behaves as a linear model in $\theta$:

$$f(x;\theta) \approx f(x;\theta_0) + \phi(x)^\top(\theta - \theta_0).$$

### 1.3 Neural Tangent Kernel

**Definition 1.2** (Neural Tangent Kernel [2])**.** Given initialization $\theta_0$, the Neural Tangent Kernel (NTK) is

$$K(x,x') \;=\; \nabla_\theta f(x;\theta_0)^\top \nabla_\theta f(x';\theta_0).$$

The NTK captures how parameter updates couple the outputs of $x$ and $x'$.

*Remark* 1.3.

- $K(x,x')$ is positive semidefinite [2].
- In the infinite-width limit, under common initializations, $K(x,x')$ converges almost surely to a deterministic kernel depending only on architecture and activation [2, 3].
- For finite but large width, $K$ is still a random kernel due to random initialization, but it concentrates around its infinite-width expectation. Fluctuations vanish at rate $O(1/\sqrt{m})$ as width $m \to \infty$ [2, 3].

### 1.4 NTK Characterization (one hidden layer, no biases)

Consider a width-$m$ one-hidden-layer network

$$f(x) \;=\; \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r\, \sigma(w_r^\top x),$$

with parameters $\theta = \{(a_r, w_r)\}_{r=1}^m$, activation $\sigma$, and random initialization

$$a_r \sim \mathcal{N}(0,\sigma_a^2), \qquad w_r \sim \mathcal{N}\Big(0, \frac{\sigma_w^2}{d} I_d\Big),$$

independently across $r$.

#### Finite-width NTK at initialization

By definition,

$$K_m(x,x') \;=\; \nabla_\theta f(x)^\top \nabla_\theta f(x') \;=\; \sum_{r=1}^{m} \Bigg[ \underbrace{\nabla_{a_r} f(x)\, \nabla_{a_r} f(x')}_{\text{output-weight part}} + \underbrace{\nabla_{w_r} f(x)^\top \nabla_{w_r} f(x')}_{\text{hidden-weight part}} \Bigg].$$

Compute the gradients:

$$\nabla_{a_r} f(x) = \frac{1}{\sqrt{m}} \sigma(w_r^\top x), \qquad \nabla_{w_r} f(x) = \frac{1}{\sqrt{m}}\, a_r\, \sigma'(w_r^\top x)\, x.$$

Hence,

$$\boxed{K_m(x,x') = \frac{1}{m}\sum_{r=1}^{m} \sigma(w_r^\top x)\, \sigma(w_r^\top x') + \frac{1}{m}\sum_{r=1}^{m} a_r^2\, \sigma'(w_r^\top x)\, \sigma'(w_r^\top x')\, x^\top x'}.$$

**Infinite-width limit**

The two sums are empirical averages of i.i.d. terms. Since $a_r$ and $w_r$ are independent with finite moments, the (strong) law of large numbers gives, almost surely,

$$\frac{1}{m}\sum_{r=1}^{m}\sigma(w_r^\top x)\,\sigma(w_r^\top x') \;\longrightarrow\; \mathbb{E}_w\big[\sigma(w^\top x)\,\sigma(w^\top x')\big],$$

$$\frac{1}{m}\sum_{r=1}^{m}a_r^2\,\sigma'(w_r^\top x)\,\sigma'(w_r^\top x') \;\longrightarrow\; \sigma_a^2\,\mathbb{E}_w\big[\sigma'(w^\top x)\,\sigma'(w^\top x')\big].$$

Thus, in the infinite-width limit, the empirical NTK converges almost surely to a deterministic kernel [2, 3].

$$\boxed{K_\infty(x,x') = \mathbb{E}_w\big[\sigma(w^\top x)\,\sigma(w^\top x')\big] \;+\; \sigma_a^2\,x^\top x'\,\mathbb{E}_w\big[\sigma'(w^\top x)\,\sigma'(w^\top x')\big]}$$

with $w \sim \mathcal{N}(0, \frac{\sigma_w^2}{d} I_d)$.

## 1.5 One Hidden Layer: Adding Biases

If we allow per-neuron biases, we get minimal changes from subsection 1.4. Consider

$$f(x) \;=\; \frac{1}{\sqrt{m}}\sum_{r=1}^{m}a_r\,\sigma\big(w_r^\top x + b_r\big), \quad a_r \sim \mathcal{N}(0,\sigma_a^2),\; w_r \sim \mathcal{N}\Big(0, \tfrac{\sigma_w^2}{d} I_d\Big),\; b_r \sim \mathcal{N}(0,\sigma_b^2),$$

independently across $r$. Let $u_r(x) := w_r^\top x + b_r$.

**Finite width** Gradients are

$$\nabla_{a_r} f(x) = \tfrac{1}{\sqrt{m}}\sigma(u_r(x)), \quad \nabla_{w_r} f(x) = \tfrac{1}{\sqrt{m}}\,a_r\,\sigma'(u_r(x))\,x, \quad \nabla_{b_r} f(x) = \tfrac{1}{\sqrt{m}}\,a_r\,\sigma'(u_r(x)).$$

Thus

$$\boxed{K_m(x,x') = \frac{1}{m}\sum_{r=1}^{m}\sigma(u_r(x))\,\sigma(u_r(x')) + \frac{1}{m}\sum_{r=1}^{m}a_r^2\,\sigma'(u_r(x))\,\sigma'(u_r(x'))\,\big(x^\top x' + 1\big)}.$$

**Infinite width (preactivation covariance picks up $\sigma_b^2$).** With $(U,V)$ jointly Gaussian:

$$\mathrm{Var}(U) = \tfrac{\sigma_w^2}{d}\|x\|^2 + \sigma_b^2, \quad \mathrm{Var}(V) = \tfrac{\sigma_w^2}{d}\|x'\|^2 + \sigma_b^2, \quad \mathrm{Cov}(U,V) = \tfrac{\sigma_w^2}{d}x^\top x' + \sigma_b^2,$$

we have

$$\boxed{K_\infty(x,x') = \mathbb{E}\big[\sigma(U)\sigma(V)\big] + \sigma_a^2\,\big(x^\top x' + 1\big)\,\mathbb{E}\big[\sigma'(U)\sigma'(V)\big]}.$$

### 1.6 Extension to Deep Fully–Connected Networks

**No biases.** Let $n_0 = d,\ n_1, \dots, n_{L-1}$ be layer widths and define

$$\alpha^{(0)}(x) = x, \qquad \tilde{\alpha}^{(\ell+1)}(x) = W^{(\ell)} \alpha^{(\ell)}(x), \qquad \alpha^{(\ell+1)}(x) = \sigma\big(\tilde{\alpha}^{(\ell+1)}(x)\big),$$

for $\ell = 0, \dots, L-2$, with entries $W^{(\ell)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}\Big(0, \frac{\sigma_w^2}{n_\ell}\Big)$. The scalar output is

$$f_\theta(x) = \frac{1}{\sqrt{n_{L-1}}} \sum_{r=1}^{n_{L-1}} a_r\, \alpha_r^{(L-1)}(x), \qquad a_r \sim \mathcal{N}(0, \sigma_a^2).$$

Define the activation covariance

$$\Sigma^{(\ell)}(x, x') := \mathbb{E}\big[\alpha_r^{(\ell)}(x)\, \alpha_r^{(\ell)}(x')\big], \qquad q^{(\ell)}(x) := \Sigma^{(\ell)}(x, x),$$

with base $\Sigma^{(0)}(x, x') = \frac{1}{d}\, x^\top x'$. Then the forward (NNGP) recursion is

$$\boxed{\Sigma^{(\ell+1)}(x, x') = \mathbb{E}_{f \sim \mathcal{GP}\left(0,\ \sigma_w^2\, \Sigma^{(\ell)}\right)}\big[\sigma\big(f(x)\big)\, \sigma\big(f(x')\big)\big]}.$$

Define the derivative–correlation kernel

$$\boxed{\dot{\Sigma}^{(\ell+1)}(x, x') := \mathbb{E}_{f \sim \mathcal{GP}\left(0,\ \sigma_w^2\, \Sigma^{(\ell)}\right)}\big[\sigma'\big(f(x)\big)\, \sigma'\big(f(x')\big)\big]}.$$

The limiting NTK satisfies

$$\boxed{\Theta_\infty^{(1)}(x, x') = \Sigma^{(1)}(x, x'), \qquad \Theta_\infty^{(\ell+1)}(x, x') = \Theta_\infty^{(\ell)}(x, x')\, \dot{\Sigma}^{(\ell+1)}(x, x') + \Sigma^{(\ell+1)}(x, x')}.$$

For a dataset $\{x_i\}$:

$$\Theta^{(\ell+1)} = \Theta^{(\ell)} \odot \dot{\Sigma}^{(\ell+1)} + \Sigma^{(\ell+1)}, \qquad \Theta^{(1)} = \Sigma^{(1)}.$$

Setting $L = 2$ recovers the two-term one-layer kernel.

### 1.7 Training Dynamics under NTK

We train with squared loss

$$L(\theta) \;=\; \frac{1}{2} \sum_{i=1}^{n} \big(f(x_i; \theta) - y_i\big)^2,$$

and consider *gradient flow* in parameter space, i.e. the continuous-time limit of gradient descent as the step size $\eta \to 0$:

$$\frac{d\theta_t}{dt} \;=\; -\nabla_\theta L(\theta_t).$$

Let $f_t(x_i) := f(x_i; \theta_t)$. By the chain rule,

$$\frac{d}{dt} f_t(x_i) \;=\; \nabla_\theta f(x_i; \theta_t)^\top \frac{d\theta_t}{dt} \;=\; -\nabla_\theta f(x_i; \theta_t)^\top \nabla_\theta L(\theta_t).$$

Compute the parameter gradient of the loss:

$$\nabla_\theta L(\theta_t) \;=\; \sum_{j=1}^{n} \big(f_t(x_j) - y_j\big)\, \nabla_\theta f(x_j; \theta_t).$$

Substituting gives

$$\frac{d}{dt}f_t(x_i) \;=\; -\sum_{j=1}^{n} \underbrace{\nabla_\theta f(x_i;\theta_t)^\top \nabla_\theta f(x_j;\theta_t)}_{=:\,K_t(x_i,x_j)} \big(f_t(x_j)-y_j\big).$$

Stacking $f_t = (f_t(x_1),\dots,f_t(x_n))$ yields the vector ODE

$$\frac{d}{dt}f_t \;=\; -\,K_t\,(f_t - y),$$

where $[K_t]_{ij} = K_t(x_i,x_j)$ is the (time–dependent) NTK matrix.

**Constant–kernel (NTK) regime.** In the infinite–width limit (or under a lazy–training approximation), the kernel remains essentially constant during training, $K_t \approx K_0 =: K$ [2]. The ODE reduces to

$$\frac{d}{dt}f_t \;=\; -\,K\,(f_t - y).$$

Let $r_t := f_t - y$. Then $\dot r_t = -K r_t$ with solution $r_t = e^{-Kt} r_0$, i.e.

$$f_t \;=\; y + e^{-Kt}\,(f_0 - y).$$

The convergence rate along eigenvector $v_j$ of $K$ is exponential with rate $\lambda_j$, the corresponding eigenvalue.

### 1.8 Lazy Training Regime

Training is in the *lazy regime* if parameter updates stay small relative to initialization:

$$\|\theta_t - \theta_0\| \ll \|\theta_0\|.$$

Then $\phi(x)$ and the NTK remain essentially constant and training is equivalent to kernel regression with fixed kernel $K$. When $\|\theta_t - \theta_0\|$ is not negligible, $\phi(x)$ evolves, yielding adaptive feature learning beyond NTK.

### 1.9 Finite Depth/Width Corrections to the NTK (Hanin–Nica)

The classical NTK result with *fixed* depth and width $\to \infty$ yields a deterministic, fixed kernel throughout training [2, 3]. In contrast, Hanin and Nica [1] analyze fully–connected ReLU networks at *finite* depth and width and show that when depth $d$ and widths $\{n_\ell\}$ grow *together*, the NTK exhibits substantial stochasticity at initialization and evolves non–trivially during training.

**Setup.** Let the network have input dimension $n_0$, hidden widths $n_1,\dots,n_{d-1}$, output dimension $n_d = 1$, ReLU activations, zero biases at init (but biases are trainable), and standard variance–preserving scalings. Define the "inverse temperature"

$$\beta \;:=\; \sum_{\ell=1}^{d-1} \frac{1}{n_\ell}\,, \qquad \text{(equal widths } n_\ell = n \text{ give } \beta = d/n\text{)}.$$

**Fluctuations of the NTK at initialization.** Denote the (on–diagonal) NTK by $K_N(x,x)$ for input $x$. Hanin–Nica prove

$$\mathbb{E}\big[K_N(x,x)\big] \;=\; d\left(\tfrac{1}{2} + \frac{\|x\|_2^2}{n_0}\right),$$

and show that the normalized second moment scales as

$$\frac{\mathbb{E}\big[K_N(x,x)^2\big]}{\mathbb{E}\big[K_N(x,x)\big]^2} \;\simeq\; \exp(5\,\beta)\,\big(1 + O(\textstyle\sum_\ell n_\ell^{-2})\big).$$

In particular, for $n_\ell = n$ this ratio is $\simeq \exp(5d/n)$, so when $d/n$ is bounded away from 0 the standard deviation is of the same order as the mean: the NTK is *not* concentrated (hence not deterministic) even if $d, n \to \infty$ jointly with $d/n = \Theta(1)$.

**Training–time evolution at initialization.** For squared loss and a single–example SGD step on $x$, the mean update of $K_N(x,x)$ at $t = 0$ satisfies

$$\frac{\mathbb{E}\left[\Delta K_N(x,x)\right]}{\mathbb{E}[K_N(x,x)]} \;\asymp\; \frac{d\,\beta}{n_0}\,\exp(5\beta)\,\Big(1 + O(\textstyle\sum_\ell n_\ell^{-2})\Big),$$

which, for equal widths, becomes $\asymp \frac{d^2}{n\,n_0}\exp(5d/n)$. Thus, unlike the fixed–depth infinite–width setting, the NTK generically *evolves* (data–dependently) when depth and width co–scale.

*Remark* 1.4 (Weak feature learning regime)*.* The results suggest a regime with $0 < \beta \ll 1$ (e.g. $0 < d/n \ll 1$) where training remains numerically stable while $K_N$ still evolves, enabling *weak* feature learning beyond the strictly lazy NTK limit. This gives a concrete knob ($\beta$) to interpolate between kernel–like behavior and feature adaptation.

### Notes

- The constant–kernel ODE in §1 (*gradient flow under fixed* $K$) exactly matches the fixed–depth, infinite–width limit. When $d$ and $n$ co–scale, $K_t$ becomes stochastic and time–varying, so the ODE becomes

$$\frac{d}{dt} f_t \;=\; -\,K_t\,(f_t - y), \qquad K_t \text{ random and evolving},$$

  with fluctuations and drift controlled by $\beta$.

- Practically, this could help explain empirical gaps between NTK predictions and real networks, and motivates experiments in the small–$\beta$ region to observe "weak" feature learning.

## References


[1] Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel, 2019. URL https://arxiv.org/abs/1909.05989.

[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks, 2020. URL https://arxiv.org/abs/1806.07572.

[3] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent ∗. *Journal of Statistical Mechanics: Theory and Experiment*, 2020 (12):124002, December 2020. ISSN 1742-5468. doi: 10.1088/1742-5468/abc62b. URL http://dx.doi.org/10.1088/1742-5468/abc62b.

[4] Radford M. Neal. *Priors for Infinite Networks*, pages 29–53. Springer New York, New York, NY, 1996. ISBN 978-1-4612-0745-0. doi: 10.1007/978-1-4612-0745-0_2. URL https://doi.org/10.1007/978-1-4612-0745-0_2.