Neural Tangent Kernel

Neural Tangent Kernel Formula

L-layer NN. For h = 1,…,L:

$$\Sigma^{(0)}(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}',$$

$$\boldsymbol{\Lambda}^{(h)}(\boldsymbol{x},\boldsymbol{x}') = \begin{pmatrix} \Sigma^{(h-1)}(\boldsymbol{x},\boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x},\boldsymbol{x}') \\ \Sigma^{(h-1)}(\boldsymbol{x}',\boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}',\boldsymbol{x}') \end{pmatrix} \in \mathbb{R}^{2\times 2},$$

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

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

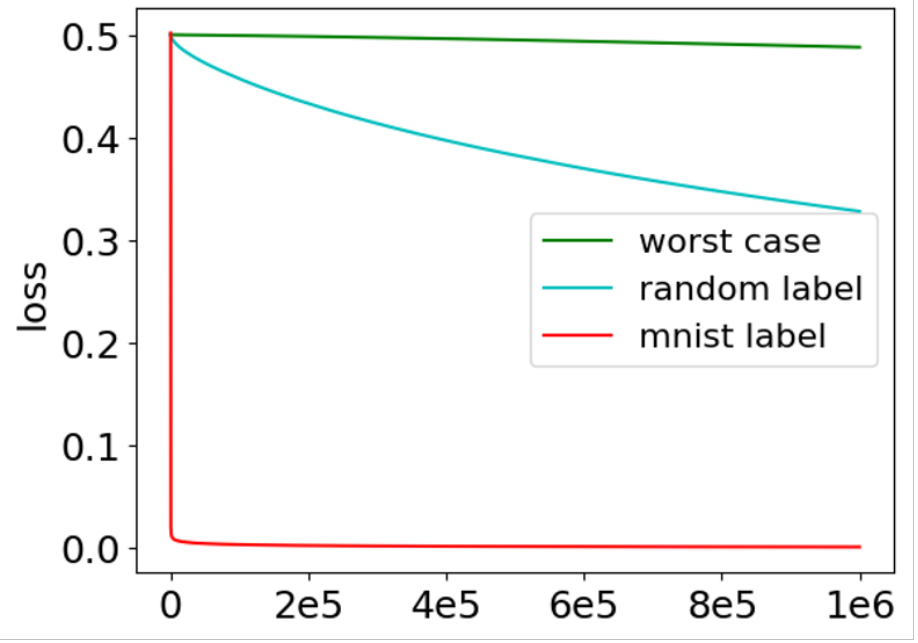
L-layer recursion.
Encodes NN's architecture.

Dependency on the derivative:
Gradient decent algorithm.

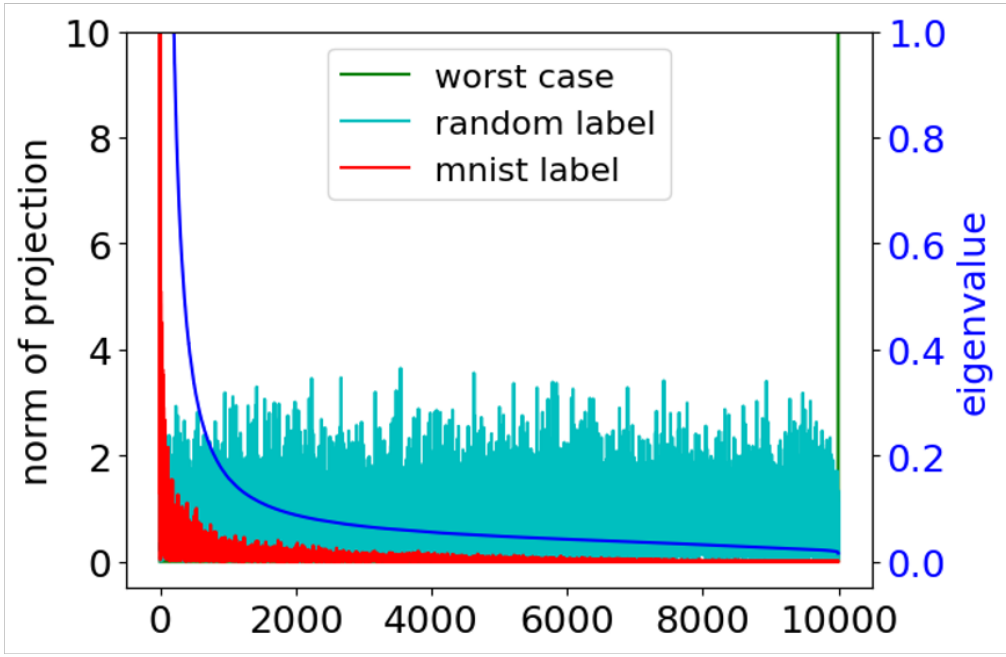
Final output:

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

What determines the convergence rate?



Convergence Rate



Projections

Neural Tangent Kernel

Recipe for designing new kernels

$$f_{\mathrm{NN}}\left(\theta_{\mathrm{NN}}, x\right) \Rightarrow k\left(x, x'\right) = \mathbb{E}_{\theta_{\mathrm{NN}} \sim \mathcal{W}}\left[\left\langle \frac{\partial f_{\mathrm{NN}}\left(\theta_{\mathrm{NN}}, x\right)}{\partial \theta_{\mathrm{NN}}}, \frac{\partial f_{\mathrm{NN}}\left(\theta_{\mathrm{NN}}, x'\right)}{\partial \theta_{\mathrm{NN}}} \right\rangle\right]$$

Transform a neural network of any architecture to a kernel!

Fully-connected NN → Fully-connected NTK

Convolutional NN → Convolutional NTK

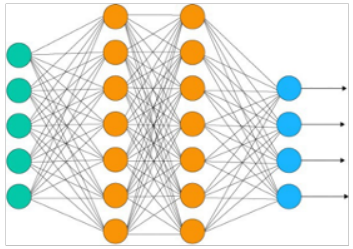
Graph NN → Graph NTK

.....

Fully-Connect NTK

$$\begin{pmatrix} -0.1 \\ 0.2 \\ \cdots \\ 0.9 \end{pmatrix}$$

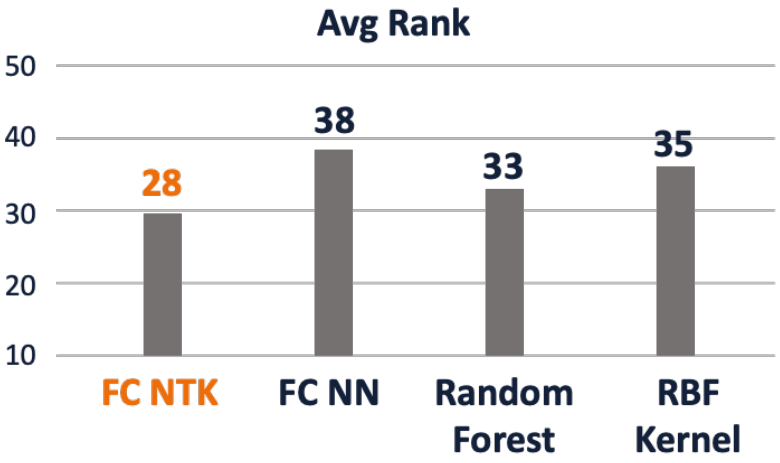
Features



FC NN

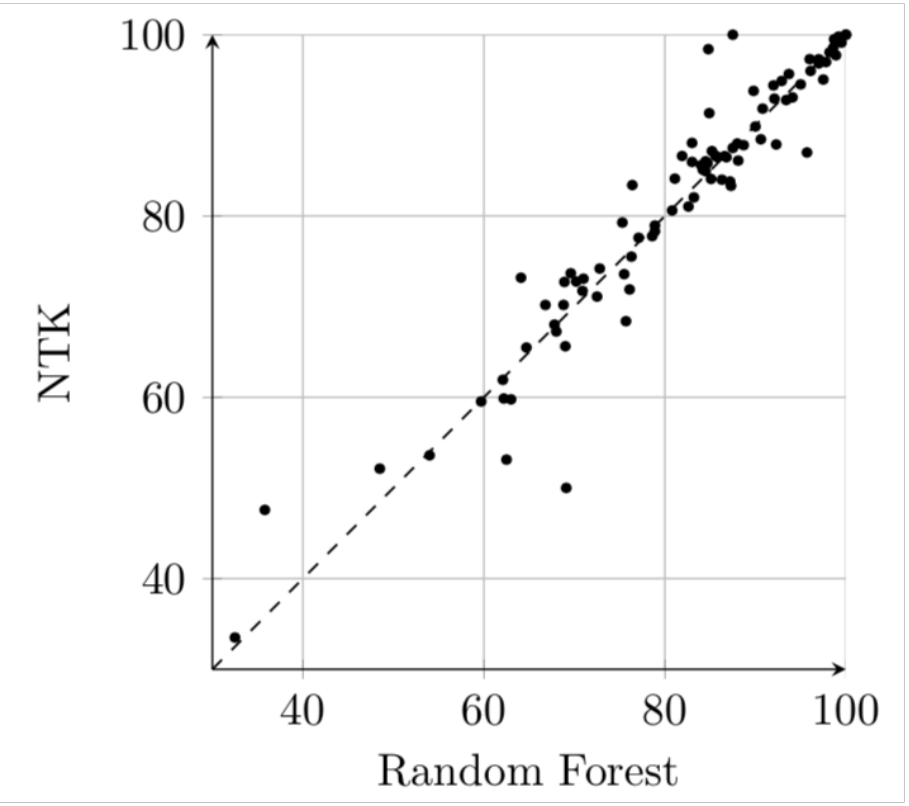
$$k\left(\begin{pmatrix} -0.1 \\ 0.2 \\ \cdots \\ 0.9 \end{pmatrix}, \begin{pmatrix} -0.3 \\ 0.5 \\ \cdots \\ -0.8 \end{pmatrix}\right)$$

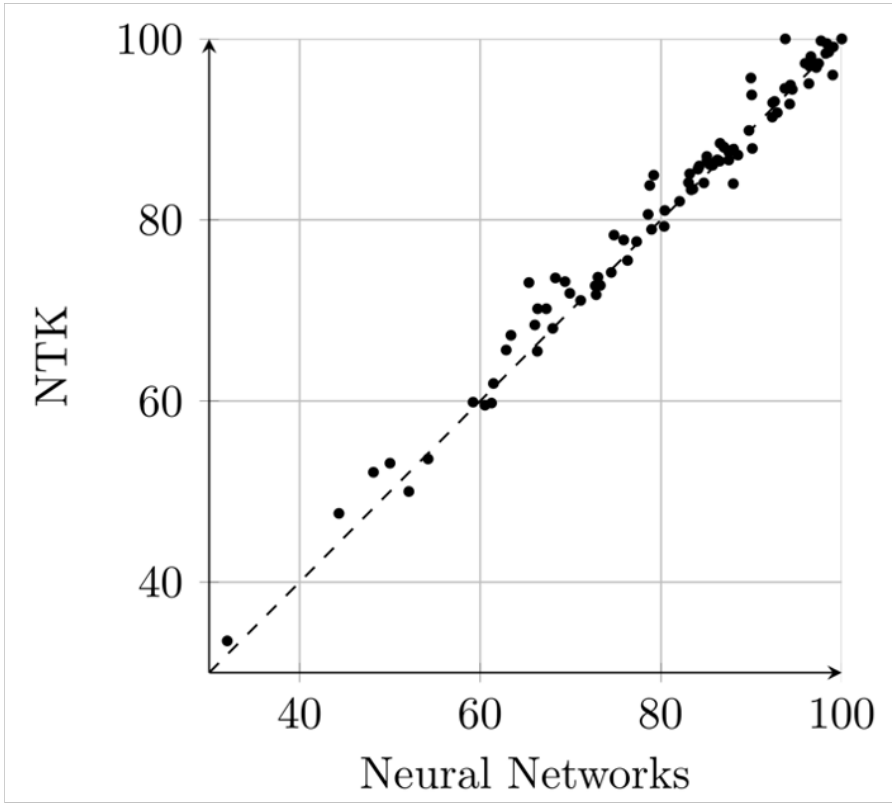
FC NTK



Classifier	Avg Acc	P95	PMA
FC NTK	82%	72%	96%
FC NN	81%	60%	95%
Random Forest	82%	68%	95%
RBF Kernel	81%	72%	94%

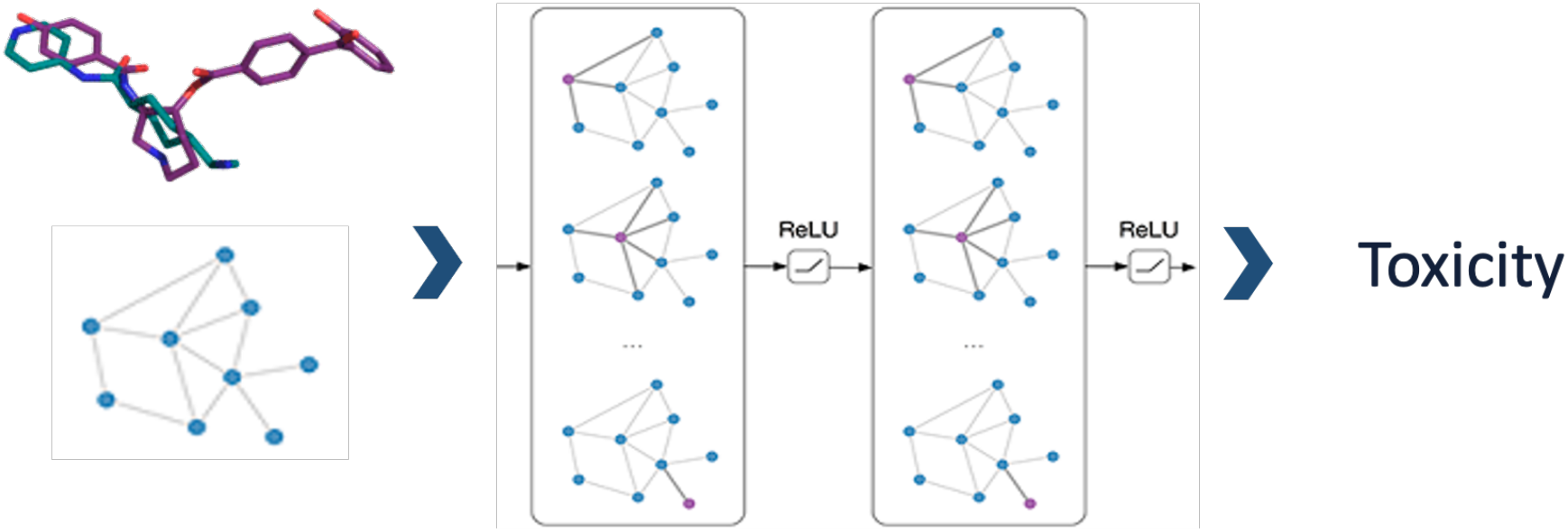
Pairwise Comparisons





Classification Accuracy

Graph Neural Network

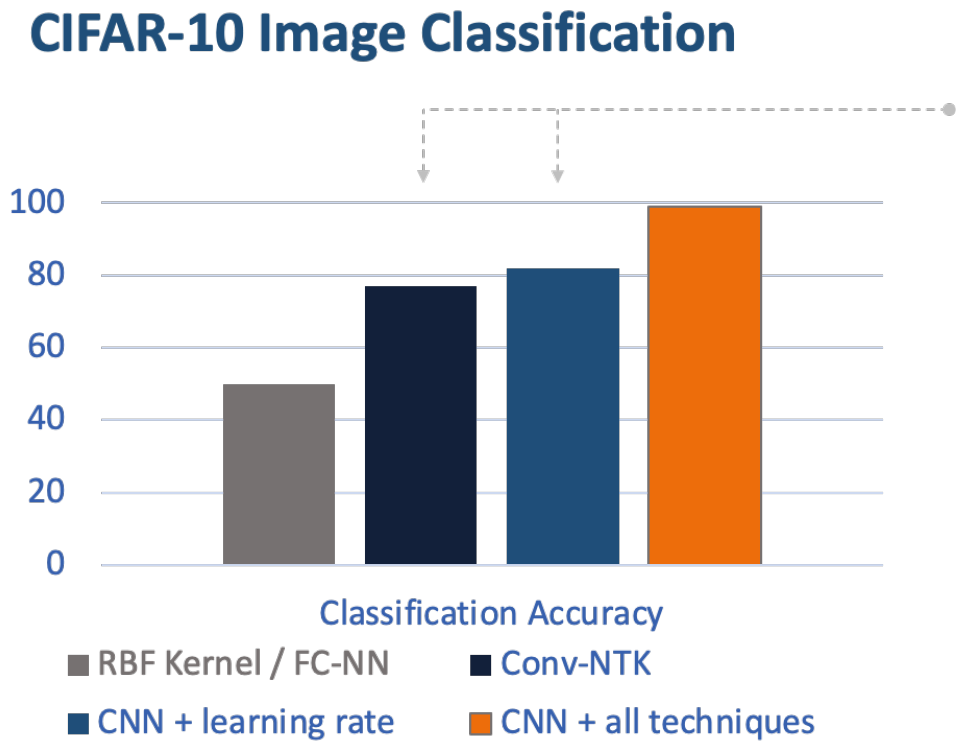


Graph Neural Tangent Kernel

Graph **Graph NN** **Graph NTK**

	Method	COLLAB	IMDB-B	IMDB-M	PTC
GNN	GCN	79%	74%	51%	64%
	GIN	80%	75%	52%	65%
GK	WL	79%	74%	51%	60%
	GNTK	84%	77%	53%	68%

Gap between NN and NTK



Open Problems:

Why there is a gap:
finite-width?
learning rate?

Understanding techniques:
batch-norm
dropout
data-augmentation
…

Deep Learning Generalization

Measure of Generalization

Generalization: difference in performance on train vs. test.

$$\frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i), y_i) - \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$$

Assumption $(x_i, y_i)\ i.i.d. \sim \mathcal{D}$

Problems with the theoretical idealization

Data is not identically distributed:

- Images (Imagenet) are scraped in slightly different ways
- Data has systematic bias (e.g., patients are tested based on symptoms they exhibit)
- Data is result of interaction (reinforcement learning)
- Domain / distribution shift

Meta Theorem of Generalization

Meta theorem of generalization: with probability $1 - \delta$ over the choice of a training set of size n, we have

$$\sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) - \mathbb{E}_{(x,y)\sim D} \left[\ell(f(x), y) \right] \right| = O\left(\sqrt{\frac{\text{Complexity}(\mathcal{F}) + \log(1/\delta)}{n}} \right)$$

Some measures of complexity:

- (Log) number of elements
- VC (Vapnik-Chervonenkis) dimension
- Rademacher complexity
- PAC-Bayes
- …

Classical view of generalization

Decoupled view of generalization and optimization:

- Optimization: find a global minimum: $\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{m} \ell(f(x_i), y_i)$
- Generalization: how well does the global optimizer generalize

Practical implications: to have a good generalization, make sure $\mathcal{F}$ is not too "complex".

Strategies:

- **Direct capacity control:** bound the size of the network / amount of connections, clip the weights, etc.
- **Regularization:** add a penalty term for "complex" predictors: weight decay (ℓ_2 norm), dropout, etc.