$u_i = f(\theta, x_i)$

prediction

$$\frac{d\,u(t)}{dt} = -H^*(u(t) - y)$$

$$H_{ij} = \lim_{\text{width}\to\infty} \mathbb{E}_{\text{init}} \left\langle \frac{\partial u_i}{\partial \theta}, \frac{\partial u_j}{\partial \theta} \right\rangle$$

Neural Tangent Kernel

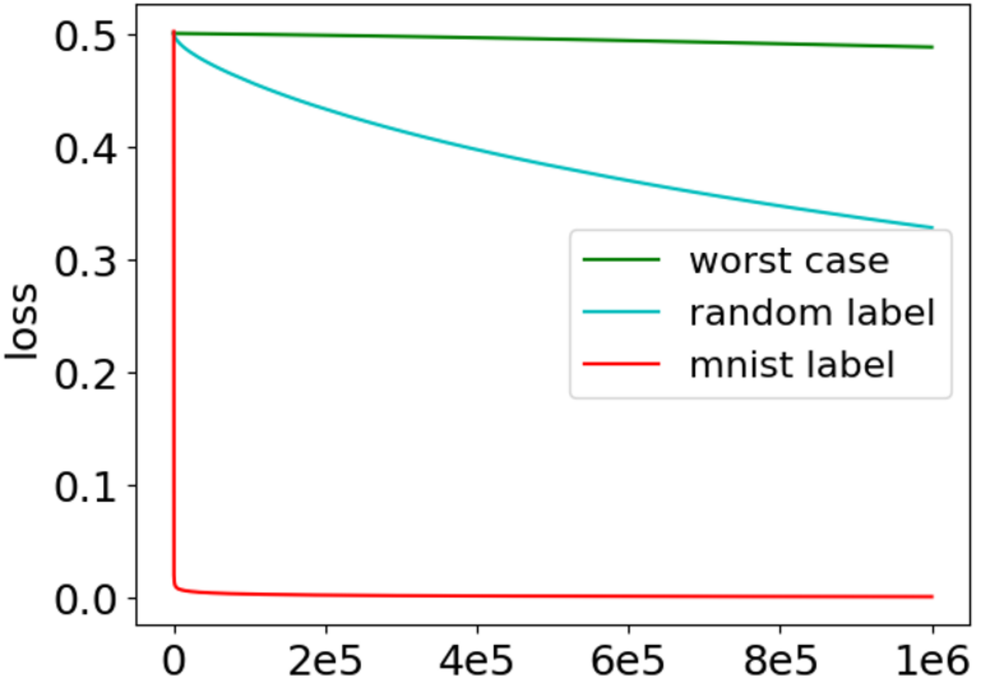
What determines the convergence rate?

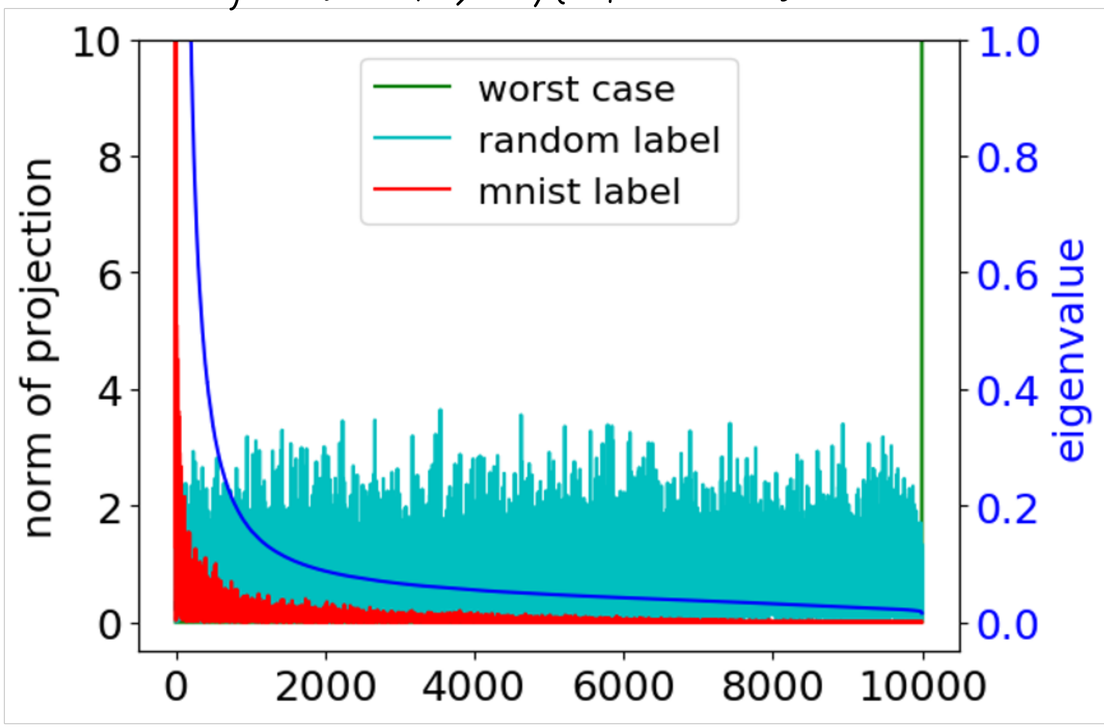
but if $y^T v_n$ large

For simplicity $u_0 \approx 0 \ll y$

$\Rightarrow u_t^T v_i - y^T v_i \approx \exp(-\sigma_i t)(-y^T v_i)$

if $y^T v_1$ large, still fast convergence





Convergence Rate

Projections

$\sigma_1 \geq \sigma_2 \cdots \sigma_n \geq 0$

$$\frac{du_t}{dt} = -H^*(u_t - y)$$

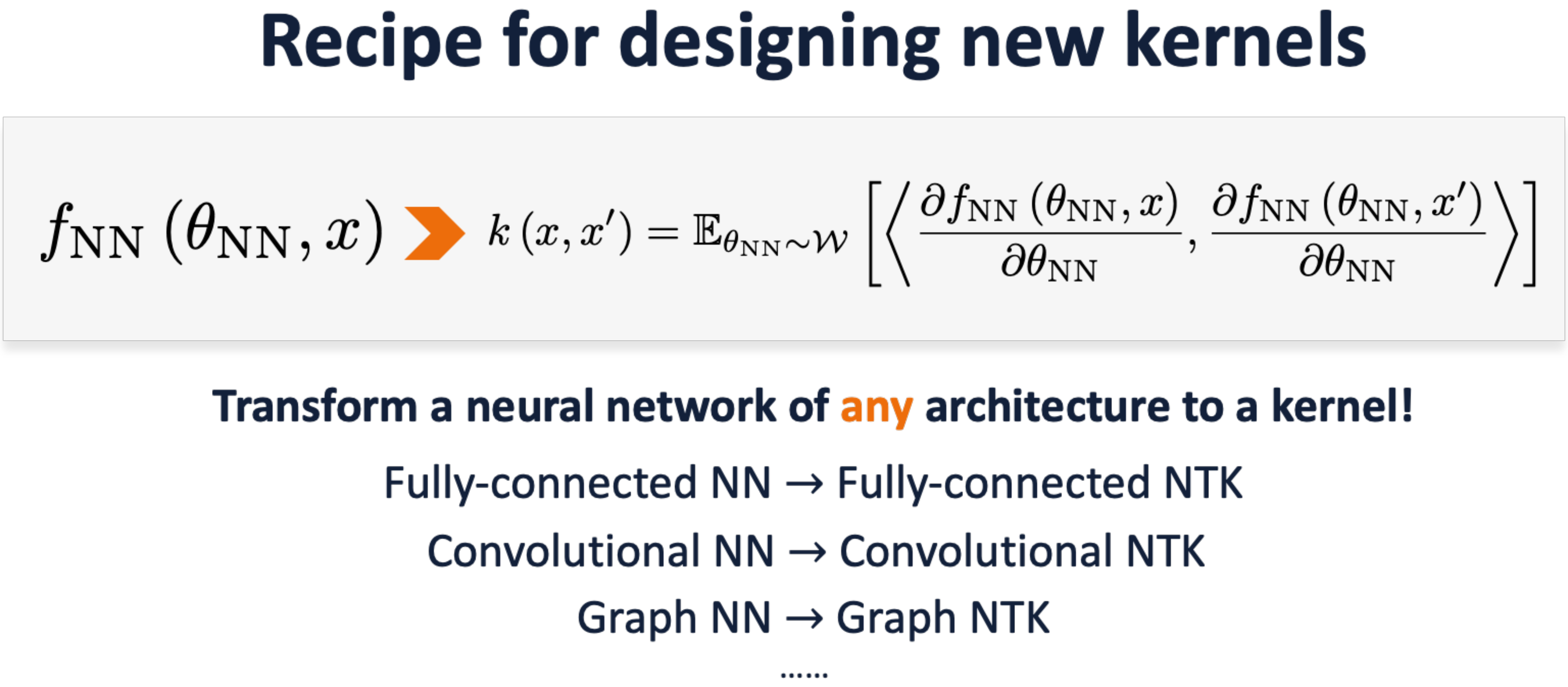
$$H^* = V \Sigma V^T,$$

σ_i: i^{th} eigenvalue v_i: i^{th} eigen vector

$$\frac{du_t^T v_i}{dt} = -\sigma_i(u_t^T v_i - y^T v_i)$$

$$u_t^T v_i - y^T v_i = \exp(-\sigma_i t)(u_0^T v_i - y^T v_i)$$

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

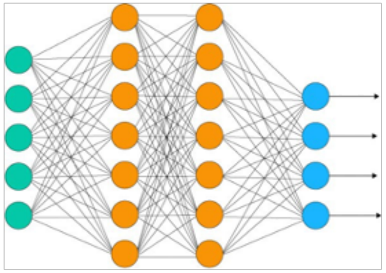
Graph NN → Graph NTK

……

Fully-Connect NTK

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

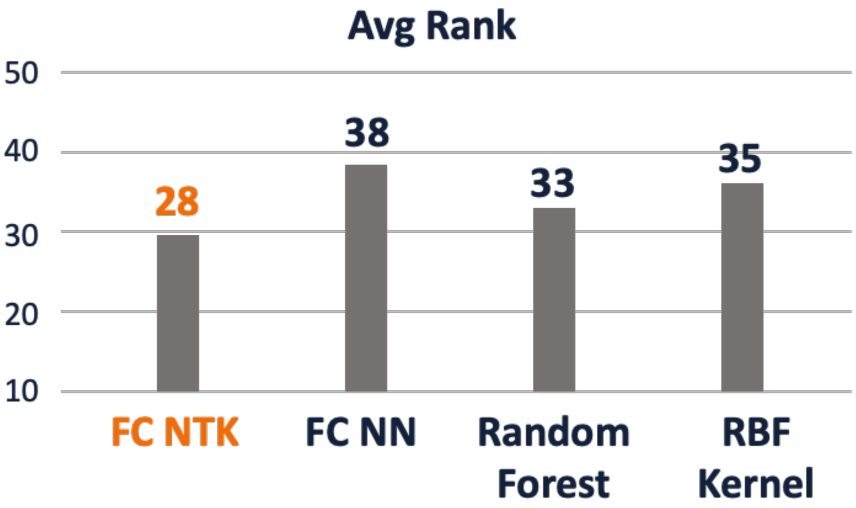
Features



FC NN

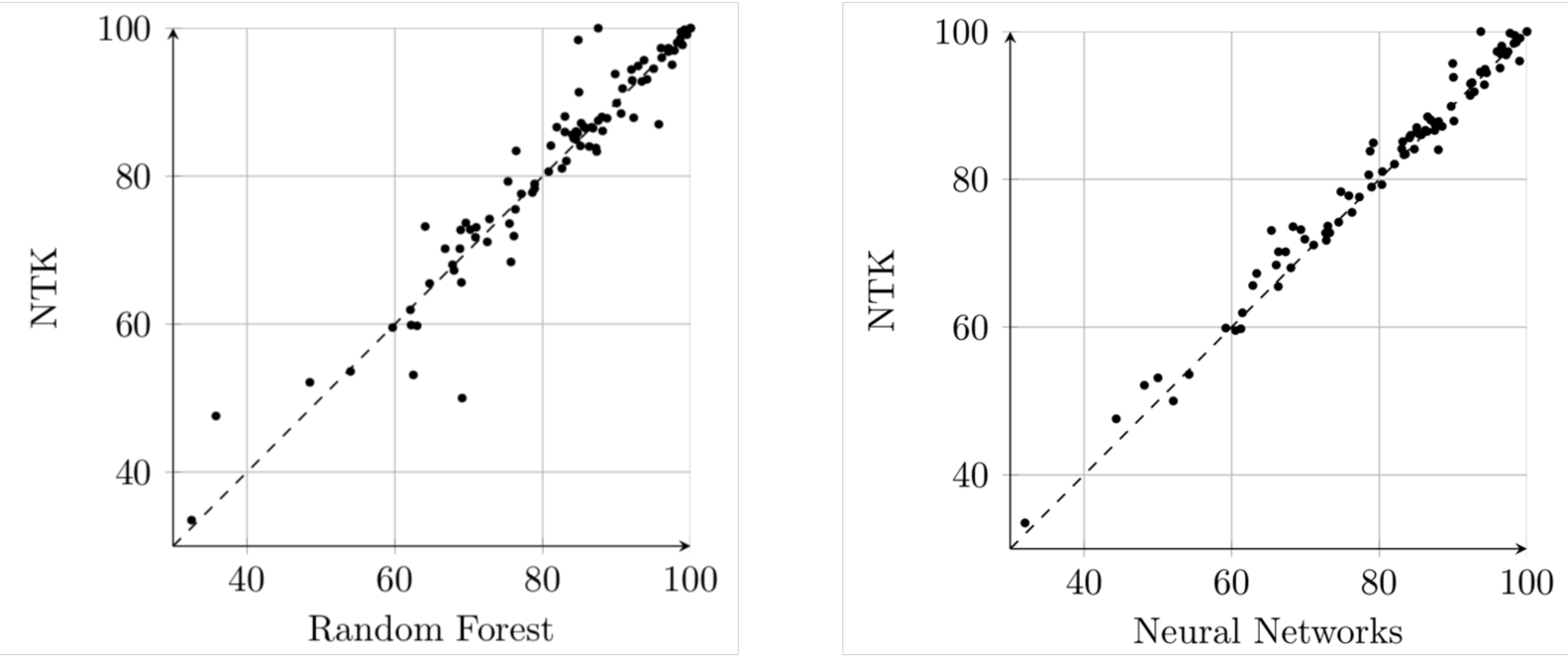
$$k\left(\begin{pmatrix} -0.1 \\ 0.2 \\ \cdots \\ 0.9 \end{pmatrix}, \begin{pmatrix} -0.3 \\ 0.5 \\ \cdots \\ -0.8 \end{pmatrix}\right)$$

FC NTK



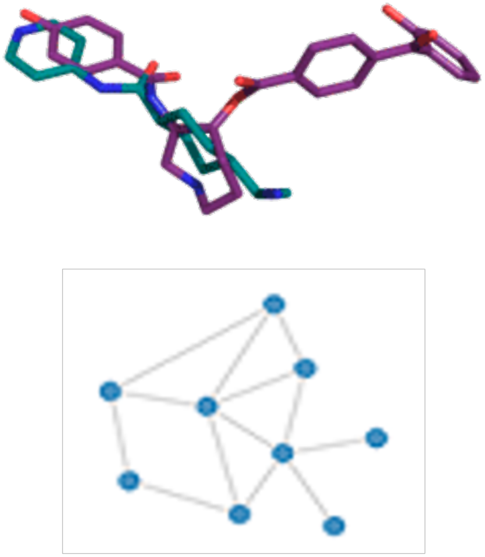
Classifier	Avg Acc	P95	PMA
FC NTK	82%	72%	96%
FC NN	81%	60%	95%
Random Forest	82%	68%	95%
RBF Kernel	81%	72%	94%

Pairwise Comparisons

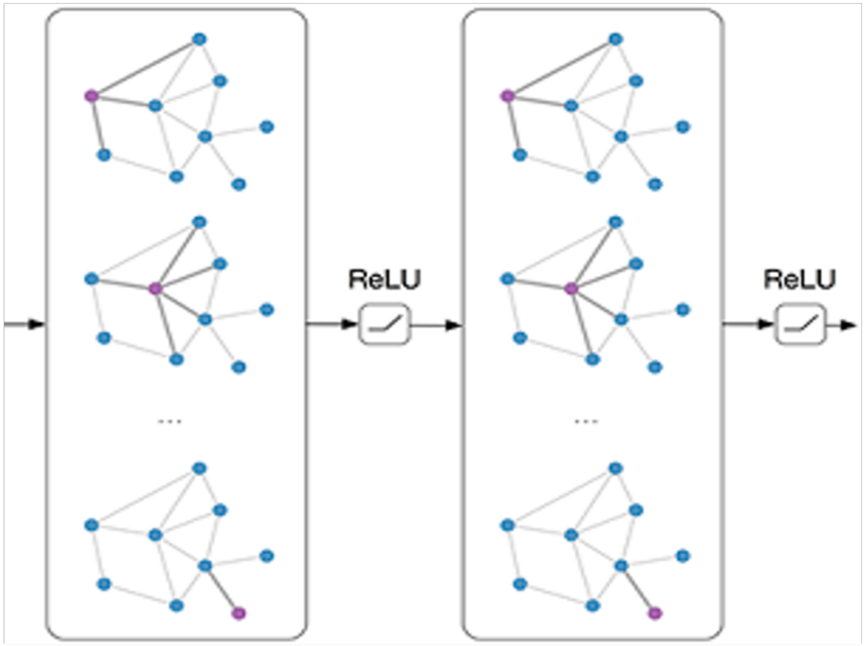


Classification
Accuracy

Graph Neural Network



Graph

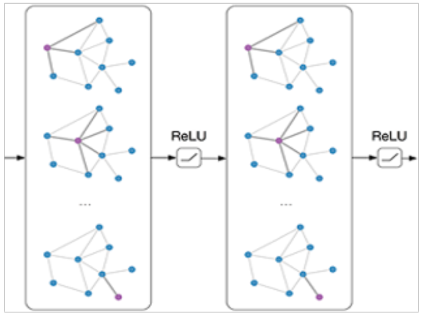


Graph Neural Network

Toxicity

Label

Graph Neural Tangent Kernel



$k($, $)$

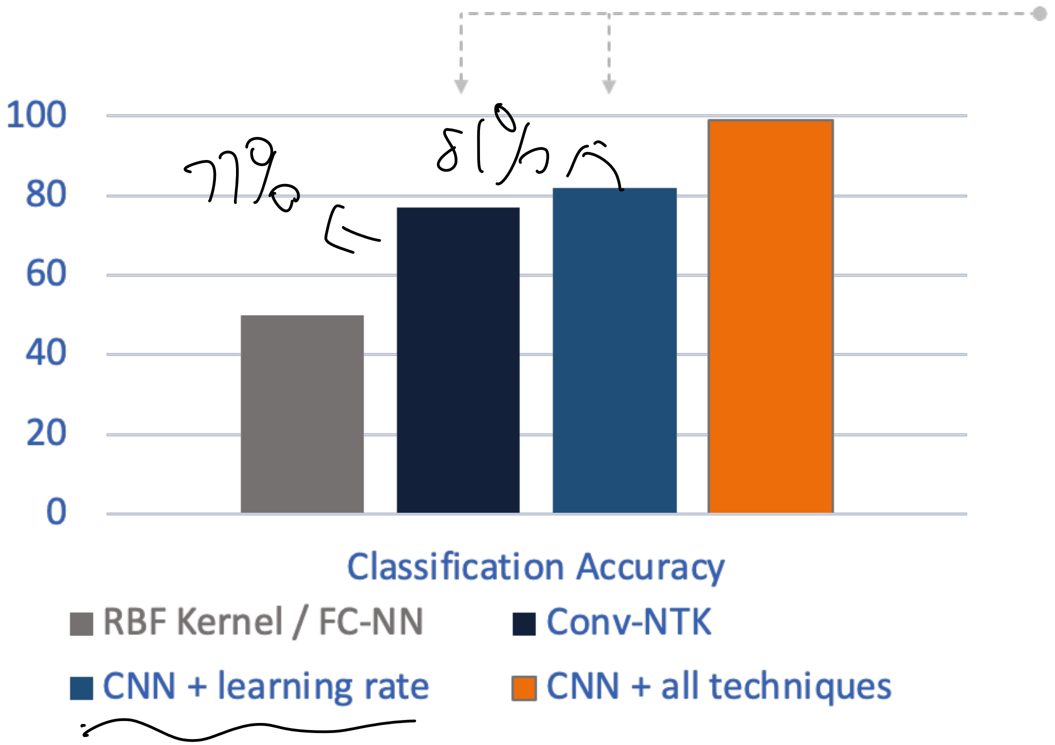
Graph **Graph NN** **Graph NTK**

	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%

What are left open?

1) CNN finite-width
2) learning rate

CIFAR-10 Image Classification



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.

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

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

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

uniform convergence

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

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

Some measures of complexity:

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

Classical view of generalization

In DL

Opt → Gen

Decoupled view of generalization and optimization:

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

Practical implications: to have a good generalization, make sure $\mathscr{F}$ 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.

Techniques for Improving Generalization

Weight Decay

L2 regularization: $\boxed{\frac{\lambda}{2}\|\theta\|_2^2}$

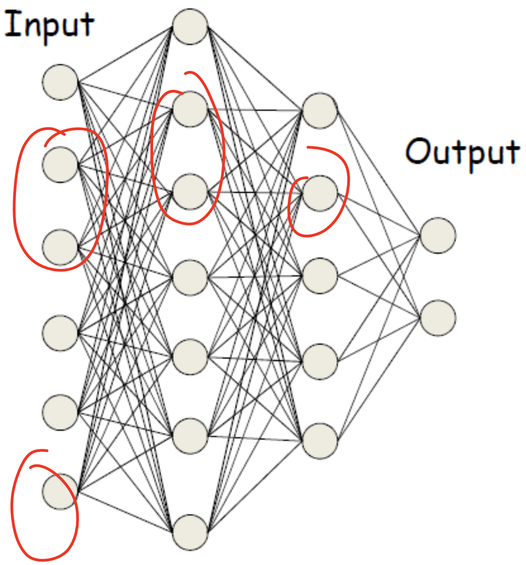
Implementation: $\theta \leftarrow (1 - \underline{\eta\lambda)\theta} - \eta\nabla f(\theta)$

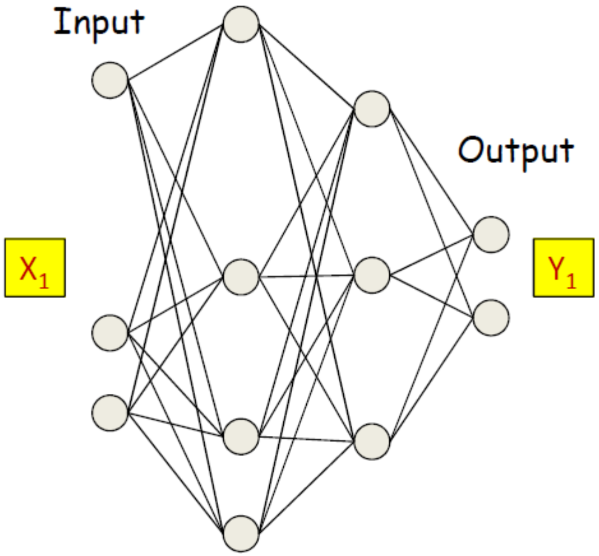
Dropout

Intuition: randomly cut off some connections and neurons.

Training: for each input, at each iteration, randomly “turn off” each neuron with a probability $1 - \alpha$

- Change a neuron to 0 by sampling a Bernoulli variable.
- Gradient only propogatd from non-zero neurons.



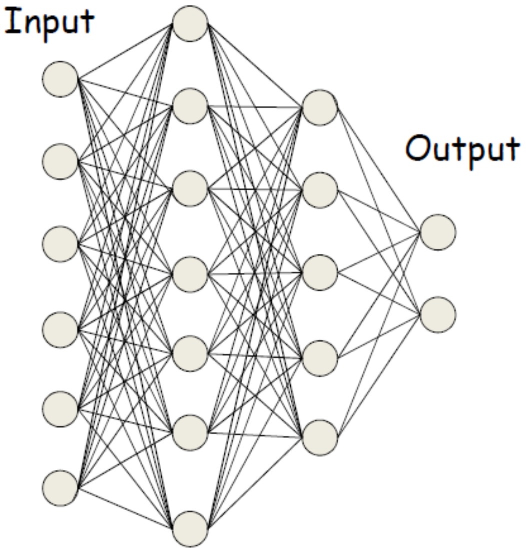


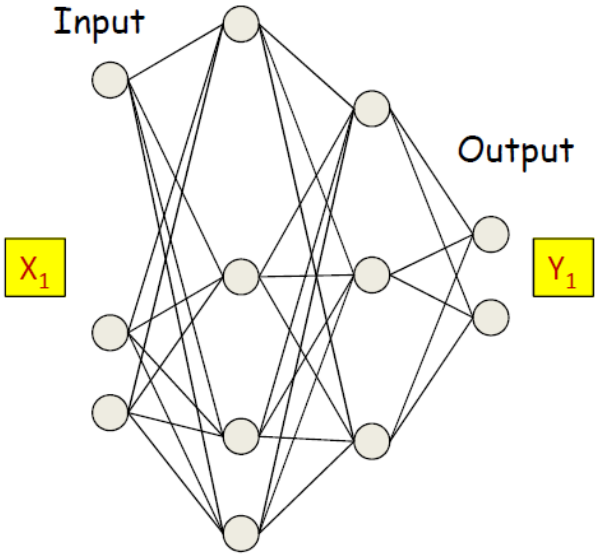
Dropout

Dropout changes the scale of the output neuron:

- $y = \text{Dropout}(\sigma(WX))$
- $\mathbb{E}[y] = \alpha\mathbb{E}[\sigma(Wx)]$

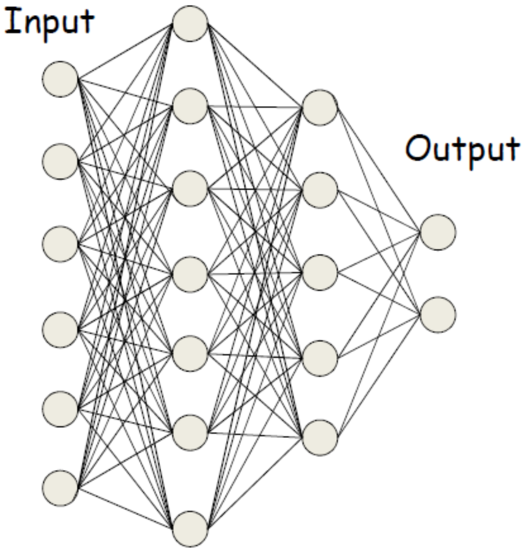
Test time: $y = \alpha\sigma(Wx)$ to match the scale

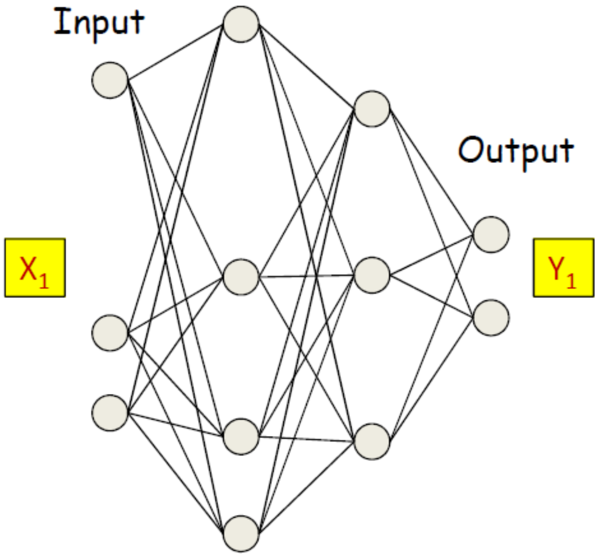




Understanding Dropout

- Dropout forces the neural network to learn redundant patterns.
- Dropout can be viewed as an implicit L2 regularizer (Wager, Wang, Liang ’13).





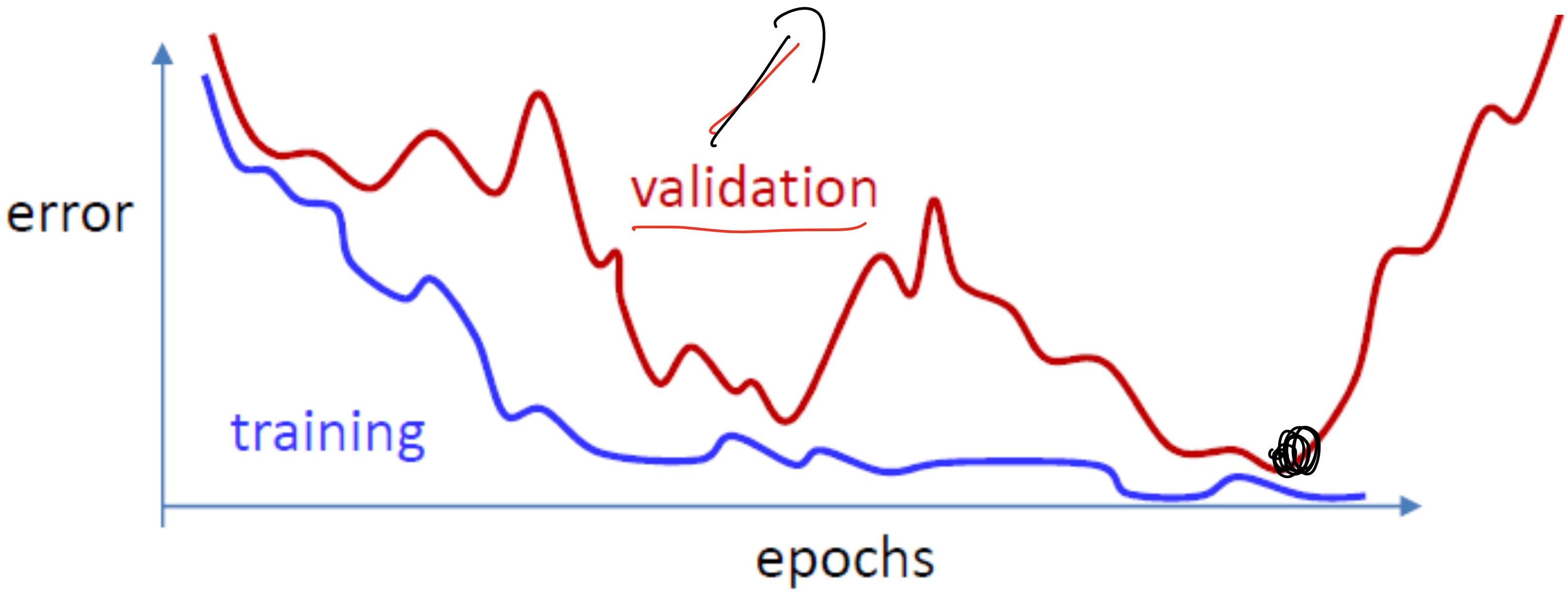
Early Stopping

($\|X\theta - y\|_2^2$ early stop at)

$\Leftrightarrow \|X\theta - y\|_2^2 + \lambda\|\theta\|_2^2$

- Continue training may lead to overfitting.
- Track performance on a held-out validation set.
- Theory: for linear models, equivalent to L2 regularization.

tune # of iterations

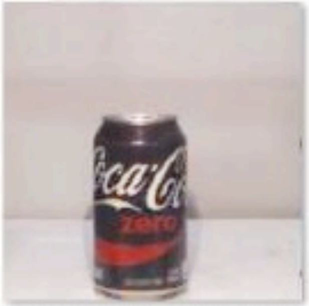


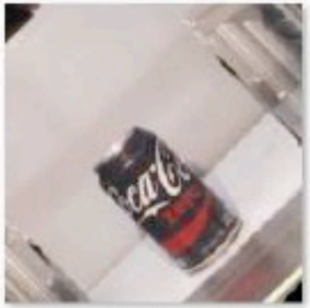
Data Augmentation

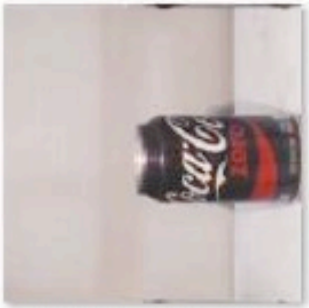
$$\sqrt{\frac{\text{Complexity}}{n}}$$

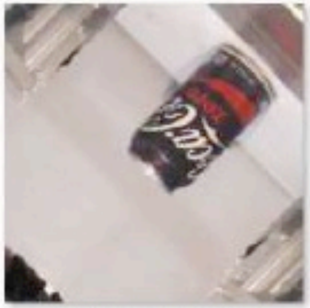
Depend on data types.

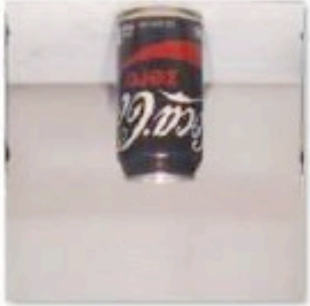
Computer vision: rotation, stretching, flipping, etc

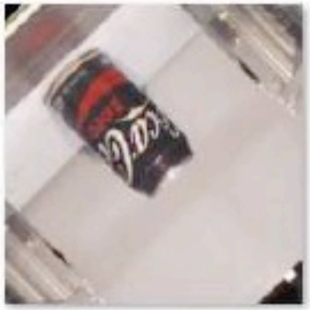

CocaColaZero1_1.png

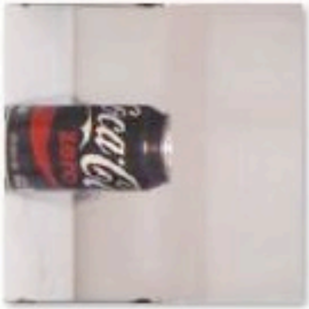

CocaColaZero1_2.png

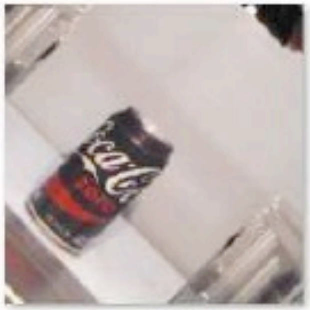

CocaColaZero1_3.png


CocaColaZero1_4.png


CocaColaZero1_5.png


CocaColaZero1_6.png


CocaColaZero1_7.png


CocaColaZero1_8.png

Mixup data augmentation

- $\hat{x} = \lambda x_i + (1 - \lambda) x_j$
- $\hat{y} = \lambda y_i + (1 - \lambda) y_j$
- $\lambda \sim$ **Beta**(0.2) $\lambda \in (0, 1)$

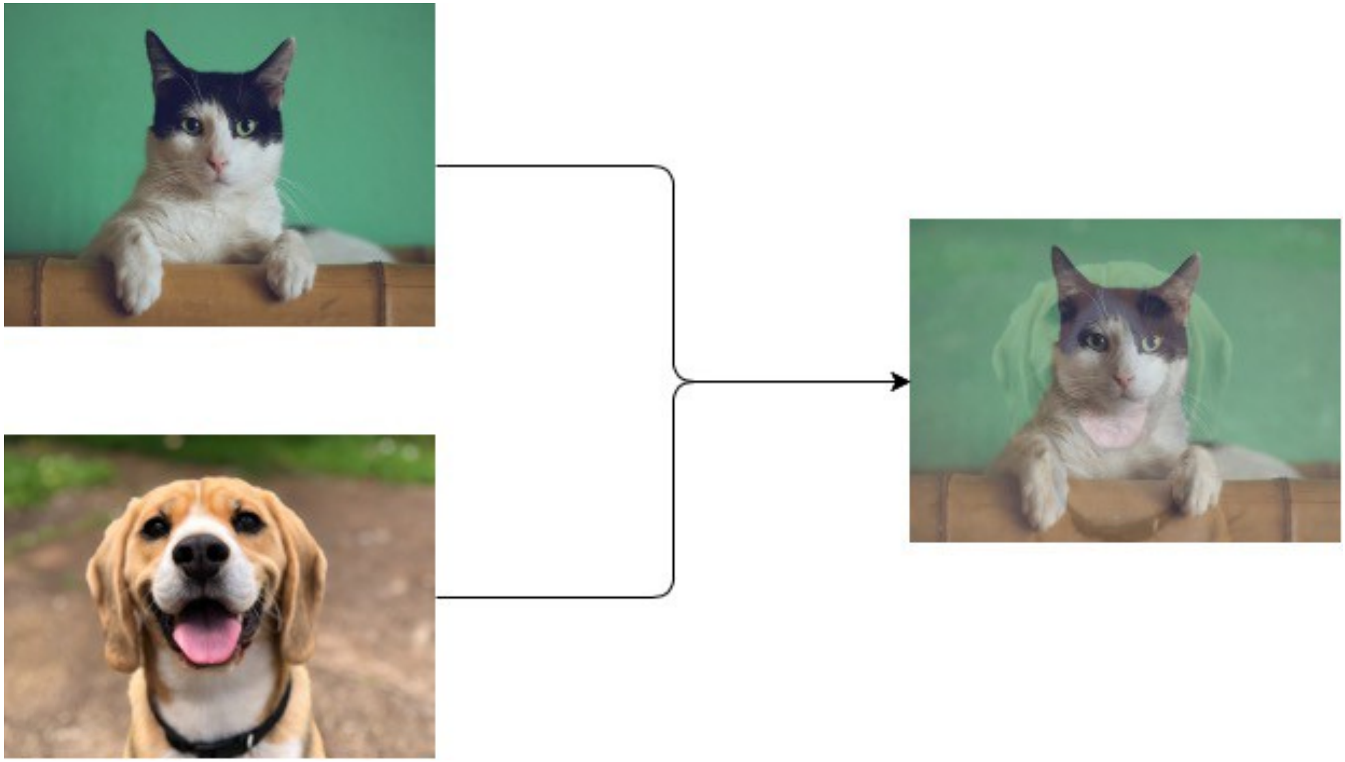
$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ one-hot

(x_i, y_i)

(x_j, y_j)

$\hat{y} = \begin{pmatrix} 0.2 \\ 0.8 \\ 0 \\ 0 \end{pmatrix}$

$(\hat{x}, \hat{y}) \longrightarrow$ Data set



Data Augmentation

Depend on data types.

Natural language processing:

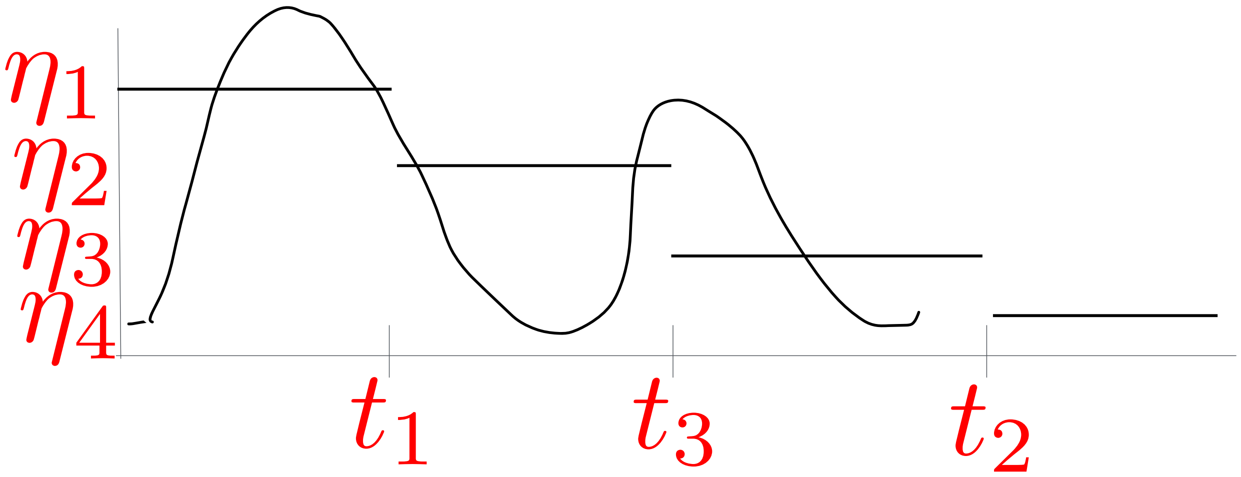
- Synonym replacement
  - *This article will focus on summarizing data augmentation in NLP.* +
  - *This write-up will focus on summarizing data augmentation in NLP.* +
- Back translation: translate the text data to some language and then translate back
  - *I have no time. -> 我没有时间. -> I do not have time.*

Learning rate scheduling

Start with large learning rate. After some epochs, use small learning rate.

SGD

Learning rate schedule

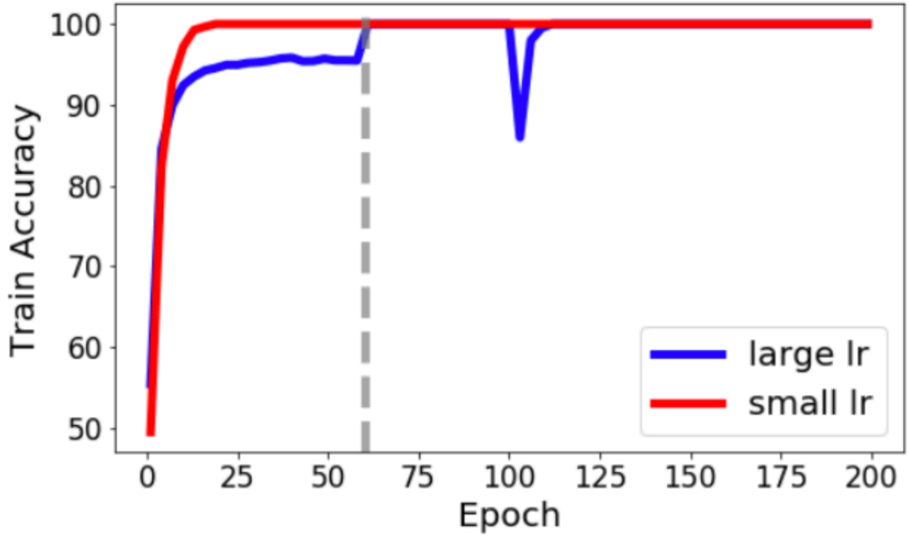


Learning rate scheduling

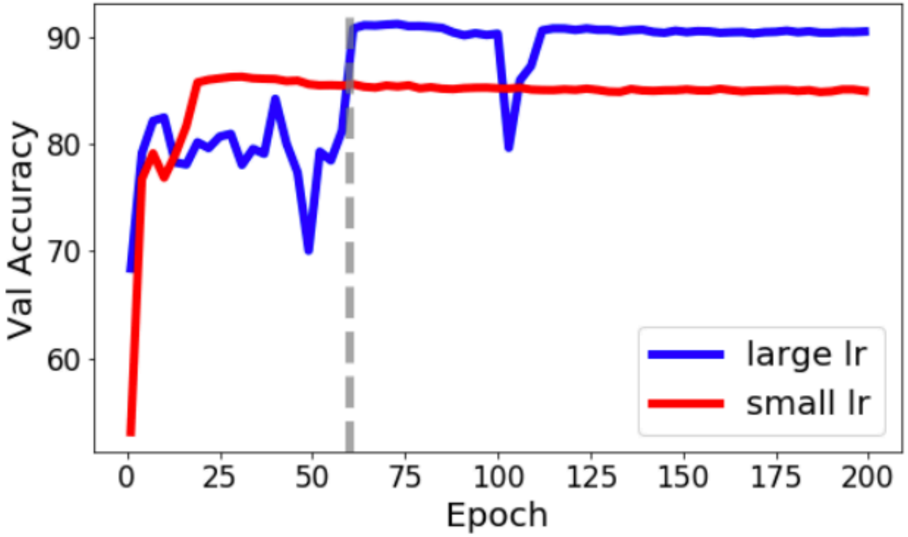
Start with large learning rate. After some epochs, use small learning rate.

Theory:

- Linear model / Kernel: large learning rate first learns eigenvectors with large eigenvalues (Nakkiran, ’20).
- Representation learning (Li et al., ‘19)



Train



Validation

Normalizations

- Batch normalization (Ioffe & Szegedy, ’15)
- Layer normalization (Ba, Kiros, Hinton, ’16)
- Weight normalization (Salimans, Kingma, ’16)
- Instant normalization (Ulyanov, Vedaldi, Lempitsky, ’16)
- Group normalization (Wu & He, ’18)
- …

Generalization Theory for Deep Learning

Basic version: finite hypothesis class

Finite hypothesis class: with probability $1 - \delta$ over the choice of a training set of size n, for a bounded loss ℓ, we have

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

Pf: for any fixed f, by Hoeffding inequality

w.p. $1 - \frac{\delta}{|F|}$, gen error $< O\left(\sqrt{\frac{\log(|F|/\delta)}{n}}\right)$

Union bound: $\text{event}_1, \dots, \text{event}_m$

$$P\left(\bigcup_i \text{event}_i\right) \le \sum P(\text{event}_i)$$

choose event_f : gen error $> \sqrt{\frac{\log(|F|/\delta)}{n}}$

$$P\left(\bigcup_{f \in F} \text{event}_f\right) \le \sum_{f \in F} P(\text{event}_f) \le \sum_{f \in F} \frac{\delta}{|F|} = \delta$$

$$\Rightarrow \quad \forall f, P\left(\text{gen error} < \sqrt{\frac{\log(|F|/\delta)}{n}}\right) \geq 1 - \delta$$

VC-Dimension

Motivation: Do we need to consider **every** classifier in $\mathscr{F}$?

Intuitively, **pattern of classifications** on the training set should suffice. (Two predictors that predict identically on the training set should generalize similarly).

Let $\mathscr{F} = \{f : \mathbb{R}^d \to \{+1, -1\}\}$ be a class of binary classifiers.

The growth function $\Pi_{\mathscr{F}} : \mathbb{N} \to \mathbb{F}$ is defined as:

$$\Pi_{\mathscr{F}}(m) = \max_{(x_1, x_2, \ldots, x_m)} \left| \left\{ (f(x_1), f(x_2), \ldots, f(x_m)) \mid f \in \mathscr{F} \right\} \right|.$$

The VC dimension of $\mathscr{F}$ is defined as:

$$\text{VCdim}(\mathscr{F}) = \max\{m : \Pi_{\mathscr{F}}(m) = 2^m\} .$$

VC-dimension Generalization bound

Theorem (Vapnik-Chervonenkis): with probability $1-\delta$ over the choice of a training set, for a bounded loss ℓ, we have

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

Examples:

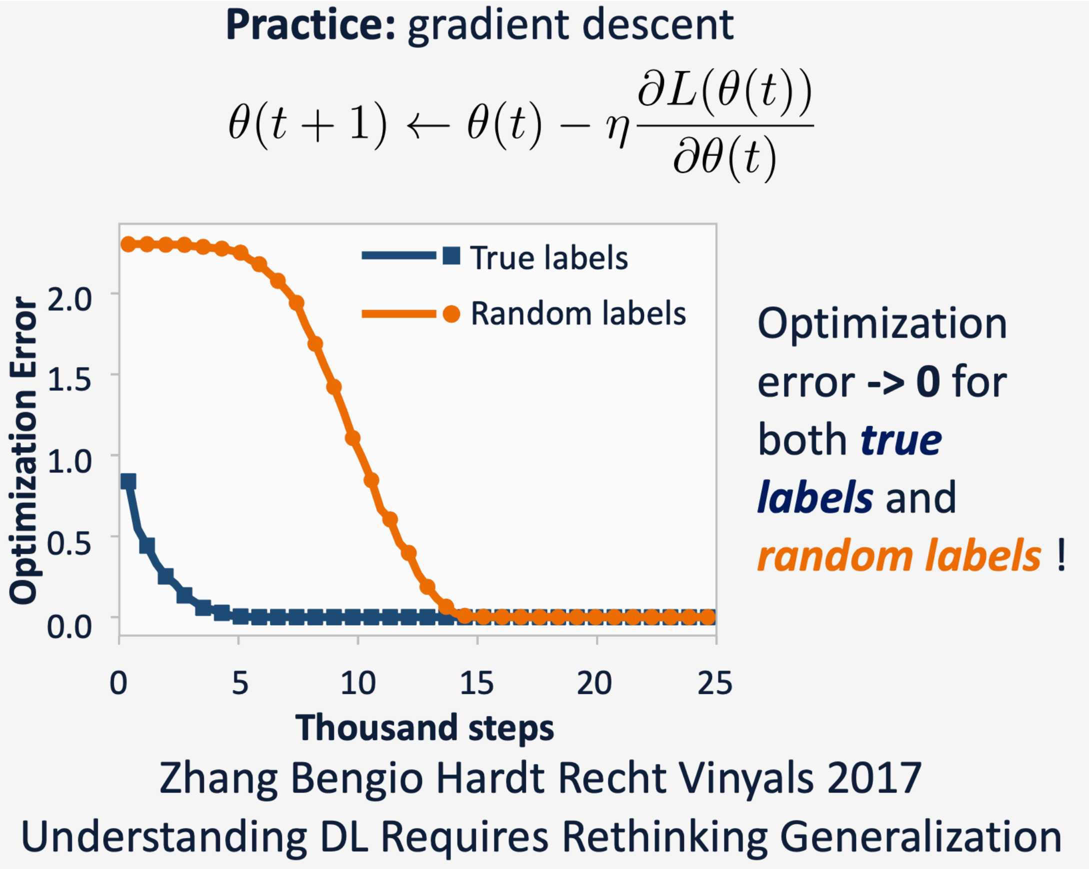
- Linear functions: VC-dim = O(dimension)
- Neural network: VC-dimension of fully-connected net with width W and H layers is $\underbrace{\widetilde{\Theta}(WH)}_{\text{tight}}$ (Bartlett et al., ’17).

Problems with VC-dimension bound

# of para > n

1. In over-parameterized regime, bound >> 1.
2. Cannot explain the random noise phenomenon:
   - Neural networks that fit random labels and that fit true labels have the same VC-dimension.

$$\sqrt{\frac{\#\text{ of para}}{n}}$$



Practice: gradient descent

$$\theta(t+1) \leftarrow \theta(t) - \eta \frac{\partial L(\theta(t))}{\partial \theta(t)}$$

Optimization error **-> 0** for both ***true labels*** and ***random labels*** !

Zhang Bengio Hardt Recht Vinyals 2017
Understanding DL Requires Rethinking Generalization

PAC Bayesian Generalization Bounds

Setup: Let P be a prior over function in class $\mathscr{F}$, let Q be the posterior (after algorithm's training).

$W_{ij}^{\ell} \sim N(0, \sigma)$

Theorem: with probability $1 - \delta$ over the choice of a training set, for a bounded loss ℓ, we have

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

$\Rightarrow$ can give bound < 1

Rademacher Complexity

Intuition: how well can a classifier class **fit random noise?**

(Empirical) **Rademacher complexity:** For a training set $S = \{x_1, x_2, \ldots, x_n\}$, and a class $\mathcal{F}$, denote:

$$\hat{R}_n(S) = \mathbb{E}_\sigma \sup_{f \in \mathcal{F}} \sum_{i=1}^{n} \sigma_i f(x_i) \; .$$

where $\sigma_i \sim$ Unif$\{+1, -1\}$ (Rademacher R.V.).

(Population) **Rademacher complexity:**

$$R_n = \mathbb{E}_S \left[\hat{R}_n(s) \right].$$

Rademacher Complexity Generalization Bound

Theorem: with probability $1 - \delta$ over the choice of a training set, for a bounded loss ℓ, we have

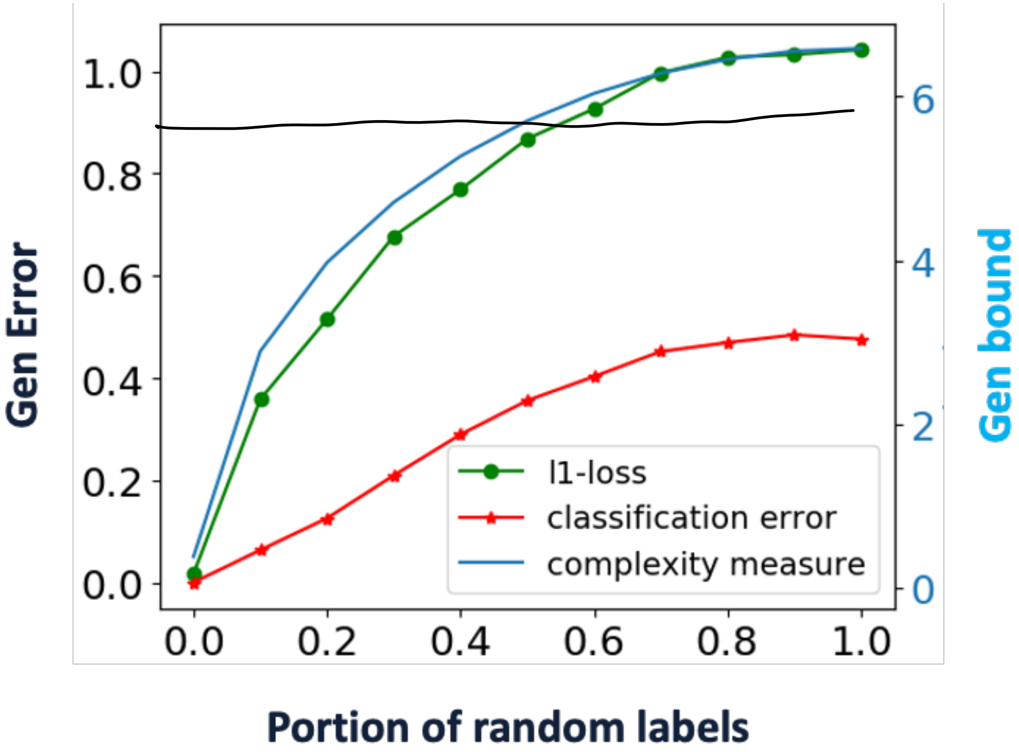
$$\sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) - \mathbb{E}_{(x,y)\sim D} \left[\ell(f(x), y) \right] \right| = O\left(\frac{\hat{R}_n}{n} + \sqrt{\frac{\log 1/\delta}{n}} \right)$$

and

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

Kernel generalization bound

Use Rademacher complexity theory, we can obtain a generalization bound $O(\sqrt{y^\top (H^*)^{-1} y/n})$ where $y \in \mathbb{R}^n$ are n labels, and $H^* \in \mathbb{R}^{n \times n}$ is the kernel (e.g., NTK) matrix.

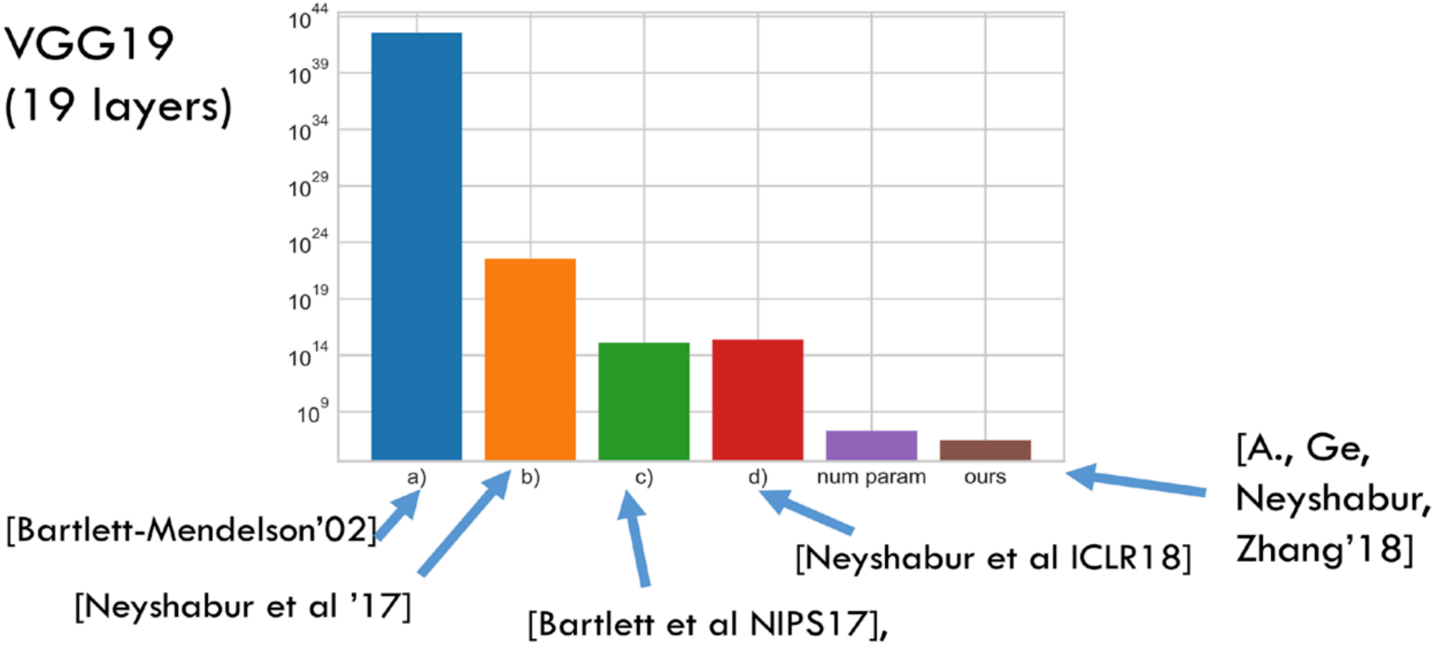


Norm-based Rademacher complexity bound

Theorem: If the activation function is σ is ρ-Lipschitz. Let $\mathcal{F} = \{x \mapsto W_{H+1}\sigma(W_h\sigma(\cdots\sigma(W_1x)\cdots)), \|W_h^T\|_{1,\infty} \leq B \forall h \in [H]\}$ then $R_n(\mathcal{S}) \leq \|X^\top\|_{2,\infty}(2\rho B)^{H+1}\sqrt{2\ln d}$ where $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d\times n}$ is the input data matrix.

Comments on generalization bounds

- When plugged in real values, the bounds are rarely non-trivial (i.e., smaller than 1)
- "*Fantastic Generalization Measures and Where to Find them*" by Jiang et al. '19 : large-scale investigation of the correlation of extant generalization measures with true generalization.



VGG19
(19 layers)

[Bartlett-Mendelson'02]

[Neyshabur et al '17]

[Bartlett et al NIPS17],

[Neyshabur et al ICLR18]

[A., Ge, Neyshabur, Zhang'18]

Image credits to Andrej Risteski

Comments on generalization bounds

- Uniform convergence may be unable to explain generalization of deep learning [Nagarajan and Kolter, '19]
  - Uniform convergence: a bound for all $f \in \mathscr{F}$
  - Exists example that 1) can generalize, 2) uniform convergence fails.

- Rates:
  - Most bounds: $1/\sqrt{n}$.
  - Local Rademacher complexity: $1/n$.