# What Can ResNet Learn Efficiently, Going Beyond Kernels?

**NIPS'19, Citation: 77**

Zeyuan Allen-Zhu, Yuanzhi Li

Microsoft Research AI, Carnegie Mellon University

# Motivation

- In many **practical tasks**, **neural networks give much better generalization error compared to kernels**, although both methods can achieve zero training error.
- For example, ResNet achieves 96% test accuracy on the CIFAR-10 data set, but NTKs achieve 77% and random feature kernels achieve 85%. This gap becomes larger on more complicated data sets.

# Problem Formulation

## Can neural networks(like ResNet) efficiently and distribution-freely learn a concept class, with better generalization than kernel methods?

In other words, can kernel method fully represent the learning capacity of a neural network?

# PAC Learnability

- **How to measure the learnability of a model with a given target function?** That is, does a model learn the target function(concept class) or not?
- We can see the **target function(concept class)** as a generator $y_i = g(x_i)$ that **generates the dataset** $\{\mathcal{X}, \mathcal{Y}\},\ x_i \in \mathcal{X}, y_i \in \mathcal{Y}\ \forall i \in N$, the features are drawn from an arbitrary distribution $\mathcal{X} \in \mathcal{D}$, and the labels are **binary** $y_i \in \{0, 1\}$.
- With **Hoeffding's inequality**, for a given training dataset $\{\mathcal{X}, \mathcal{Y}\}$ with $N$ samples, we can guarantee that

$$Pr(||f(\mathcal{X}) - \mathbb{E}[f(X)]||_2^2 > \epsilon) \leq 2e^{-2\epsilon^2 N}$$

where $X \in \mathcal{D}$ and $Y = f(X)$ are random variables. $\epsilon$ is the error.

# PAC Learnability

## Formal Definition

Let $\mathcal{C}$ be a class of **boolean functions** $f : \{0,1\}^n \to \{0,1\}$. We say that $\mathcal{C}$ **is PAC-learnable if there exists an algorithm** $\mathcal{L}$ **such that for every** $f \in \mathcal{C}$**, for any probability distribution** $\mathcal{D}$, for any (where $0 < \epsilon \leq \frac{1}{2}$), for any $\delta$ (where $0 \leq \delta < 1$) algorithm $\mathcal{L}$ on input $\epsilon$ and $\delta$ and a set of random examples picked from any probability distribution $\mathcal{D}$ **outputs at least with a probability** $1 - \delta$**, concept** $h$ **such that error** $(h, f) \leq \epsilon$.

# Main Idea

- For neural networks with **ReLU** activations, we show without any distributional assumption, a **three-layer residual network (ResNet)** can (improperly) learn a **concept class that includes three-layer ResNets of smaller size and smooth activations**, and the generalization error is also small if polynomially many training examples are given while the network is trained by SGD.
- Then prove that for **some $\delta \in (0,1)$, with $N = O(\delta^{-2})$ training samples**, neural networks can efficiently **achieve generalization error $\delta$ for this concept class over any distribution**; in contrast, there exists a simple distributions such that **any kernel method cannot have generalization error better than $\sqrt{\delta}$ for this class cannot have generalization error better than $\sqrt{\delta}$ for this class**.
- Also prove a computation complexity advantage of neural networks with respect to linear regression over arbitrary feature mappings as well.

# Target Function

We wish to learn a **concept class given by target functions** that can be written as

$$\mathcal{H}(x) = \mathcal{F}(x) + \alpha \mathcal{G}(\mathcal{F}(x))$$

where $\alpha \in [0, 1)$ and $\mathcal{G} : \mathbb{R}^k \to \mathbb{R}^k$, $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^k$ are two functions that can be written as two-layer networks with smooth activations.

# Function Complexity

Measure the complexity of any infinite-order smooth function $\phi : \mathbb{R} \to \mathbb{R}$. Suppose $\phi(z) = \sum_{i=0}^{\infty} c_i z_i$ is its Taylor expansion.

$$C_\phi \stackrel{def}{=} C^* \sum_{i=0}^{\infty} (i+1)|c_i|$$

where $C^*$ is a sufficiently large constant (e.g., $10^4$).

# Theorem 1

Consider a **single-skip three-layer ResNet with ReLU** activation, defined as a function $\text{out} : \mathbb{R}^d \to \mathbb{R}^k$

$$\text{out}(x) = A(\sigma(Wx + b_1) + \sigma(U\sigma(Wx + b_1) + b_2))$$

Where $\sigma$ is the ReLU function, $W \in \mathbb{R}^{m \times d}$ and $U \in \mathbb{R}^{m \times m}$ are the hidden weights, $A \in \mathbb{R}^{k \times m}$ is the output weight, and $b_1, b_2 \in \mathbb{R}^m$ are two bias vectors.

For any distribution over $x$, for every $\delta \in ((\alpha C_{\mathcal{G}})^4, 1)$ with probability at least 0.99, SGD efficiently learns a network $\text{out}(x)$ satisfying

$$\mathbb{E}_{(x,y)\sim\mathcal{D}} \frac{1}{2}||\text{out}(x) - y||_2^2 \leq \delta \quad \text{using} \quad N = \tilde{O}(\frac{C_{\mathcal{F}}^2}{\delta^2}) \quad \text{samples}$$

The **running time of SGD** is **polynomial in** $\text{poly}(C_{\mathcal{G}}, C_{\mathcal{F}}, \alpha^{-1})$. In other words, ResNet is capable of **achieving population risk** $\alpha^4$

# Theorem 3

Given (Mercer) kernels $K_1, \ldots, K_k : \mathbb{R}^{d \times d} \to \mathbb{R}$ and training examples $\{(x^{(i)}, y^{(i)})\}_{i \in [N]}$ from $\mathcal{D}$, a kernel method tries to learn a function $\mathfrak{K}\colon \mathbb{R}^d \to \mathbb{R}^k$ where each

$$\mathfrak{K}_j(x) = \sum_{n \in [N]} K_j(x, x^{(n)}) \cdot w_{j,n} \tag{3.1}$$

**Theorem 3** (kernel, sketched). *For every constant $k \geq 2$, for every sufficiently large $d \geq 2$, there exist concept classes consisting of functions $\mathcal{H}(x) = \mathcal{F}(x) + \alpha \mathcal{G}(\mathcal{F}(x))$ with complexities $C_{\mathcal{F}}, C_{\mathcal{G}}$ and $\alpha \in (0, \frac{1}{C_{\mathcal{G}}})$ such that, letting*

$$N_{\mathsf{res}} \text{ be the sample complexity from Theorem 1 to achieve } \alpha^{3.9} \text{ population risk,}$$

*then there exists simple distributions $\mathcal{D}$ over $(x, \mathcal{H}(x))$ such that, for at least 99% of the functions $\mathcal{H}$ in this concept class, even given $N = O\big((N_{\mathsf{res}})^{k/2}\big)$ training samples from $\mathcal{D}$, any function $\mathfrak{K}(x)$ of the form (3.1) has to suffer population risk*

$$\mathbb{E}_{(x,y)\sim\mathcal{D}} \tfrac{1}{2} \|\mathfrak{K}(x) - y\|_2^2 > \alpha^2 \quad \text{even if the label } y = \mathcal{H}(x) \text{ has no error.}$$

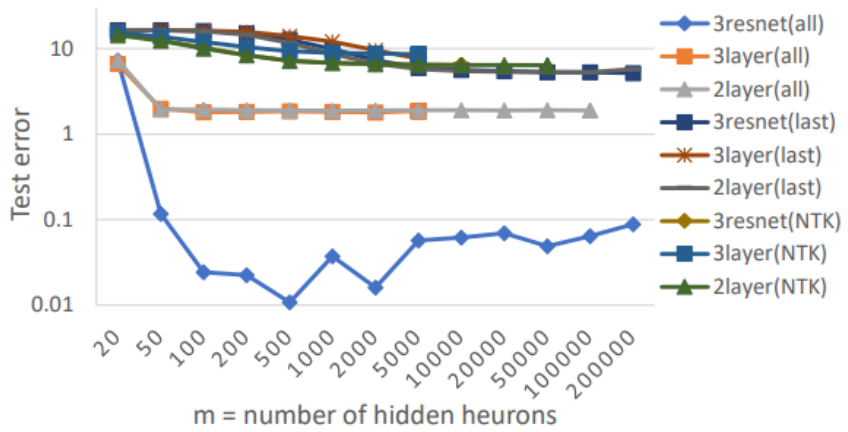
# Intuition

- **Neural networks (trained by SGD)** achieve **population risk $\alpha^{3.9}$ using $N_{res}$ samples** for **any distribution over $x$**, while **kernel** methods **cannot** achieve any population risk **better than $\alpha^2$ for some simple distributions even with $N = (N_{res})^{k/2} \gg N_{res}$ samples**.
- **Kernel method tries to learn everything in one shot**. This unavoidably requires the **sample complexity to be at least $\Omega(d^k)$**. Intuitively, as the kernel method tries to learn $\mathcal{G}(\mathcal{F})$ from scratch, this means that it has to **take into account all $\Omega(d^k)$ many possible choices of $\mathcal{G}(\mathcal{F})$ (recall that $\mathcal{G}$ is a degree $k$ polynomial over dimension $d$)**.
- On the other hand, a kernel method with $N$ **samples only has $N$-degrees of freedom (for each output dimension)**. This means, if $N \ll o(d^k)$, kernel method simply **does not have enough degrees of freedom to distinguish between different $\mathcal{G}(\mathcal{F})$**, so has to pay $\Omega(\alpha^2)$ in population risk

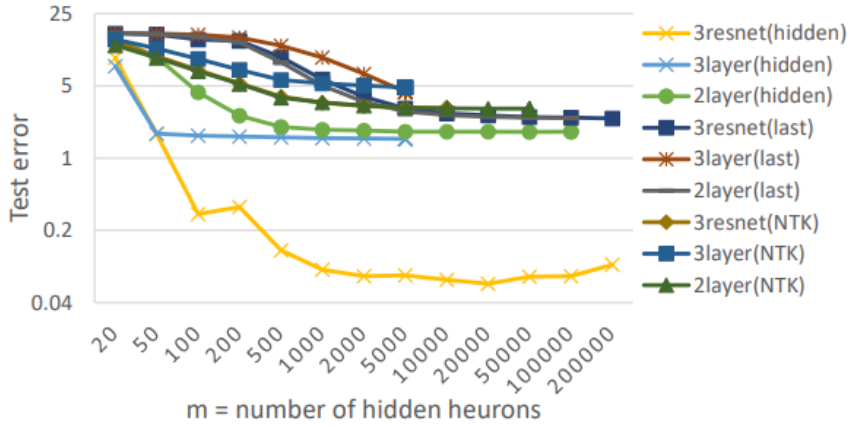
# Experiment

- Generate **feature vectors** $x \in \{-1, 1\}^{30}$ that are **uniformly sampled at random**, and labels are generated from a **target function** $\mathcal{H}(x) = \mathcal{F}(x) + \alpha\mathcal{G}(\mathcal{F}(x)) \in \mathbb{R}^{15}$ satisfying $\mathcal{F}(x) = (x_1x_2, x_3x_4, ..., x_{29}x_{30}),\ \mathcal{F} : \mathbb{R}^{30} \to \mathbb{R}^{15}$ and $\mathcal{G}_i(y) = (-1)^i y_1y_2y_3y_4,\ \mathcal{G} : \mathbb{R}^{15} \to \mathbb{R}^{15}$ for all $i = 1, 2, ..., 15$. In other words, $\mathcal{F}$ is a **degree-2 parity function** over **30 dimensions**, and $\mathcal{G}$ is a **degree-4 parity function** over **15 dimensions**.
- SGD optimizer of pytorch, with **momentum 0.9**, **mini-batch size 50**. We carefully run each algorithm with respect to learning rates and **weight decay parameters** in the set $\{10^{-k}, 2 \cdot 10^{-k}, 5 \cdot 10^{-k} : k \in \mathbb{Z}\}$, and present the **best one in terms of testing accuracy**. In each parameter setting, we run **SGD for 800 epochs**, and **decrease the learning rate by 10 on epoch 400**.

# Experiment 1




(a) $N = 500$, train all layers vs. kernel methods

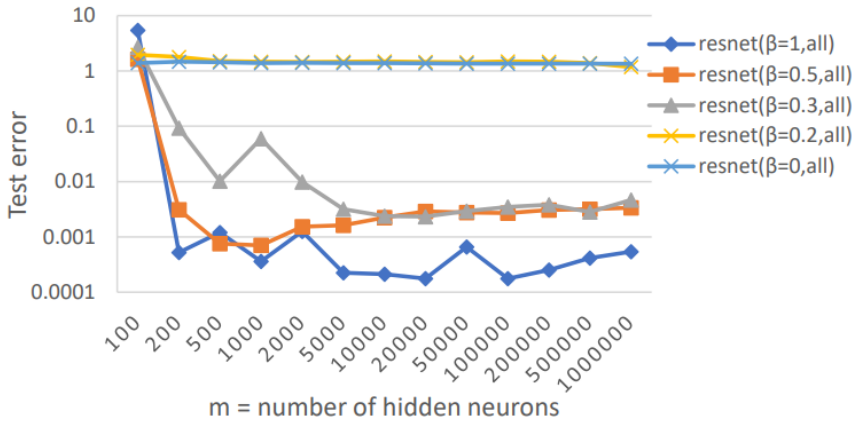



(b) $N = 1000$, train hidden layers vs. kernel methods

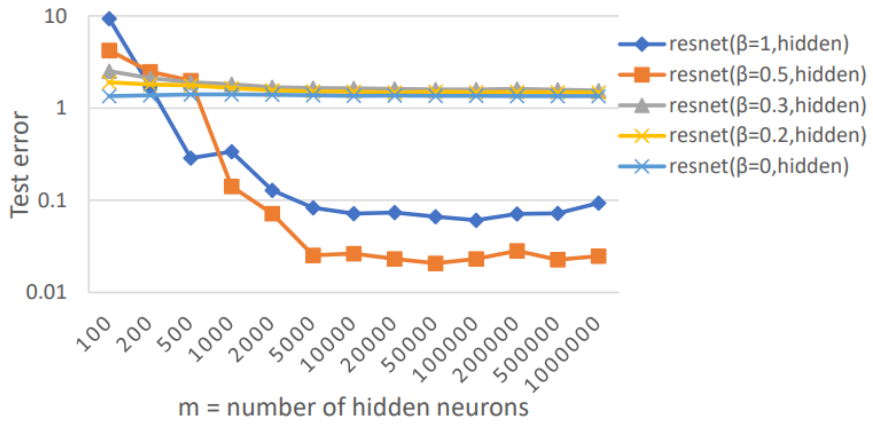
Figure 2: Performance comparison. `3resnet` stands for our three-layer ResNet and `3layer/2layer` stands for three and two-layer fully connected networks. `(all)` stands for training all layers, `(hidden)` stands for training only hidden layers, `(last)` stands for training only the last output layer, and `(NTK)` stands for training all layers in the *neural tangent kernel* [21]. We emphasize that `(last)` is a kernel method and corresponds to the *conjugate kernel* [12]. Experiment setup is in Section 8.1.

Denote $N$ as the number of training samples. Let $\alpha = 0.3$ and $k = 15$ so that test error $k\alpha^2 = 1.35$ is a threshold for detecting whether the trained model has successfully learned $\alpha\mathcal{G}(\mathcal{F}(x))$ or not

# Experiment 2




(a) training all layers of `3resnet`




(b) training hidden layers of `3resnet`

Figure 3: Sensitivity test on $\alpha$. Using the same choice of $\mathcal{F}(x)$ and $\mathcal{G}(y)$ from Section 8.1, we choose target function $\mathcal{H}(x) = \beta\mathcal{F}(x) + \alpha\mathcal{G}(\mathcal{F}(x))$ with $\alpha = 0.3$ and varying $\beta \in [0, 1]$.

when $\alpha < \beta$, the base signal is larger than the composite signal, so indeed ResNet can perform hierachical learning; in contrast, when $\alpha > \beta$, learning the composite signal becomes practically impossible.