Deep learning theory for computer vision

Volkan Cevher, **Fanghui Liu**, **Grigorios G. Chrysos**

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)
Switzerland

at Conference on Computer Vision and Pattern Recognition 2023

License Information for CVPR Tutorial Slides

Acknowledgements

- LIONS group members (current & alumni): https://lions.epfl.ch
  - Quoc Tran Dinh, Fabian Latorre, Ahmet Alacaoglu, Maria Vladarean, Chaehwan Song, Ali Kavis, Mehmet Fatih Sahin, Thomas Sanchez, Thomas Pethick, Igor Krawczuk, Leello Dadi, Paul Rolland, Junhong Lin, Marwa El Halabi, Baran Gozcu, Quang Van Nguyen, Yurii Malitskyi, Armin Eftekhari, Ilija Bogunovic, Yen-Huan Li, Anastasios Kyrillidis, Ya-Ping Hsieh, Bang Cong Vu, Kamal Parameswaran, Jonathan Scarlett, Luca Baldassarre, Bubacarr Bah, Grigorios Chrysos, Stratis Skoulakis, Fanghui Liu, Kimon Antonakopoulos, Andrej Janchevski, Pedro Abranches, Luca Viano, Zhenyu Zhu, Yongtao Wu, Wanyun Xie, Alp Yurtsever.
  - EE-556 (Mathematics of Data): Course material

- Many talented faculty collaborators
  - Panayotis Mertikopoulos, Georgios Piliouras, Kfir Levy, Francis Bach, Joel Tropp, Madeleine Udell, Stephen Becker, Suvrit Sra, Mark Schmidt, Larry Carin, Michael Kapralov, Martin Jaggi, David Carlson, Adrian Weller, Adish Singla, Lorenzo Rosasco, Alessandro Rudi, Stefanie Jegelka, Panos Patrinos, Andreas Krause, Niao He, Bernhard Schölkopf, Olivier Fercoq...

- Many talented collaborators
  - Francesco Locatello, Chris Russell, Matthaeus Kleindessner, Puya Latafat, Andreas Loukas, Yu-Guan Hsieh

Today: “Basic” robust machine learning

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})$$

- A seemingly simple optimization formulation
- Critical in machine learning with many applications
  - Adversarial examples and training
  - Generative adversarial networks
  - Robust reinforcement learning

Warm up: Flexibility of the template

$$\Phi^\star = \min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}) \quad (\text{argmin}, \text{argmax} \to \mathbf{x}^\star, \mathbf{y}^\star)$$

Warm up: Flexibility of the template

$$\Phi^\star = \min_{\mathbf{x}\in\mathcal{X}} \underbrace{\max_{\mathbf{y}:\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})}_{f(\mathbf{x})} \quad (\text{argmin}, \text{argmax} \to \mathbf{x}^\star, \mathbf{y}^\star)$$

$$f^\star = \min_{\mathbf{x}:\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad (\text{argmin} \to \mathbf{x}^\star)$$

Warm up: Flexibility of the template

$$\Phi^\star = \min_{\mathbf{x}\in\mathcal{X}} \underbrace{\max_{\mathbf{y}:\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})}_{f(\mathbf{x})} \quad (\text{argmin}, \text{argmax} \to \mathbf{x}^\star, \mathbf{y}^\star)$$

$$f^\star = \min_{\mathbf{x}:\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad (\text{argmin} \to \mathbf{x}^\star)$$

◦ (eula) In the sequel,

- the set $\mathcal{X}$ is convex
- all convergence characterizations are with feasible iterates $\mathbf{x}^k \in \mathcal{X}$
- L-smooth means $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x}-\mathbf{y}\|, \forall \mathbf{x},\mathbf{y} \in \mathcal{X}$
- ∇ may refer to the generalized subdifferential

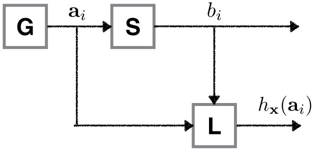
Warm up: Flexibility of the template

$$\Phi^\star = \min_{\mathbf{x}\in\mathcal{X}} \underbrace{\max_{\mathbf{y}:\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})}_{f(\mathbf{x})} \quad (\text{argmin}, \text{argmax} \to \mathbf{x}^\star, \mathbf{y}^\star)$$

$$f^\star = \min_{\mathbf{x}:\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad (\text{argmin} \to \mathbf{x}^\star)$$

- (eula) In the sequel,
  - the set $\mathcal{X}$ is convex
  - all convergence characterizations are with feasible iterates $\mathbf{x}^k \in \mathcal{X}$
  - L-smooth means $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x}-\mathbf{y}\|, \forall \mathbf{x},\mathbf{y}\in\mathcal{X}$
  - ∇ may refer to the generalized subdifferential

A deep learning optimization problem in supervised learning



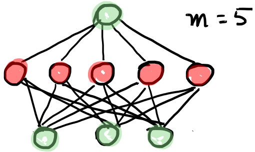
Definition (Optimization formulation)

The "deep-learning" problem with a neural network $h_{\mathbf{x}}(\mathbf{a})$ is given by

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

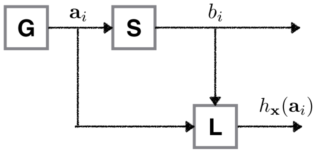
where $\mathcal{X}$ denotes the constraints and L is a loss function.

- A single hidden layer neural network with params $\mathbf{x} := [\mathbf{X}_1, \mathbf{X}_2, \mu_1, \mu_2]$



$$h_{\mathbf{x}}(\mathbf{a}) := \Big[\ \mathbf{X}_2\ \Big]\ \underbrace{\overset{\text{activation}}{\sigma}\left(\Big[\ \overset{\text{weight}}{\mathbf{X}_1}\ \Big] \Big[\overset{\text{input}}{\mathbf{a}}\Big] + \Big[\overset{\text{bias}}{\mu_1}\Big] \right) + \Big[\overset{\text{bias}}{\mu_2}\Big]}_{\text{hidden layer = learned features}}$$

A deep learning optimization problem in supervised learning



Definition (Optimization formulation)

The "deep-learning" problem with a neural network $h_{\mathbf{x}}(\mathbf{a})$ is given by

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

where $\mathcal{X}$ denotes the constraints and L is a loss function.

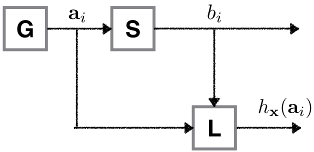
Adversarial Training

Let $h_{\mathbf{x}} : \mathbb{R}^n \to \mathbb{R}$ be a model with parameters $\mathbf{x}$ and let $\{(\boldsymbol{a}_i, \mathbf{b}_i)\}_{i=1}^n$, with $\boldsymbol{a}_i \in \mathbb{R}^p$ and $\mathbf{b}_i$ be the corresponding labels. The adversarial training optimization problem is given by

$$\min_{\mathbf{x}} \left\{ \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} \underbrace{\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), \mathbf{b}_i) \right]}_{=: f_i(\mathbf{x})} \right\}.$$

Note that L is not continuously differentiable due to ReLU, max-pooling, etc.

A deep learning optimization problem in supervised learning



Definition (Optimization formulation)

The "deep-learning" problem with a neural network $h_{\mathbf{x}}(\mathbf{a})$ is given by

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

where $\mathcal{X}$ denotes the constraints and L is a loss function.

Example objectives in different tasks

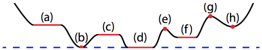
- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \le \epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\delta}\right), \mathbf{b}_i\right)\right]\right\}$ Adversarial training [44].
- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_2 \le \epsilon} L(h_{\mathbf{x}+\boldsymbol{\delta}}\left(\mathbf{a}_i\right), \mathbf{b}_i)\right]\right\}$ ϵ-stability training [10], Sharpness-aware minimization [29].
- $\min_{\mathbf{x}} \max_{\mathbf{b}^c \in [C]} \frac{1}{n_c} \sum_{i=1}^{n_c} \left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\| \le \epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\delta}\right), \mathbf{b}_i^c\right)\right]$ Class fairness [67].

Basic questions on solution concepts

◦ Consider the finite sum setting

$$\boxed{f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} f_j(\mathbf{x}) \right\}.}$$

◦ Goal: Find $\mathbf{x}^\star$ such that $\nabla f(\mathbf{x}^\star) = 0$.

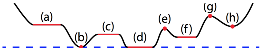


Basic questions on solution concepts

◦ Consider the finite sum setting

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} f_j(\mathbf{x}) \right\}.$$

◦ Goal: Find $\mathbf{x}^\star$ such that $\nabla f(\mathbf{x}^\star) = 0$.



1. Does SGD converge with probability 1? [8, 71, 55, 61]
2. Does SGD avoid non-minimum points with probability 1? [51, 31, 61]
3. How fast does SGD converge to local minimizers? [31, 32, 61]
4. Can SGD converge to global minimizers? [42, 45, 34, 89, 37, 64, 53, 25, 97, 47, 72]

Vanilla (Minibatch) SGD

Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\boldsymbol{x}^0$, step size α_k

1. **For** $k = 0, 1, \ldots$:
   obtain the (minibatch) stochastic gradient $\mathbf{g}^k$
   update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \gamma_k \mathbf{g}^k$

Perturbed Stochastic Gradient Descent [30]

Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\boldsymbol{x}^0$, step size α_k

1. **For** $k = 0, 1, \ldots$:
   sample noise ξ uniformly from unit sphere
   update $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k - \alpha_k(\mathbf{g}^k + \xi)$

*Stochastic Gradient Langevin Dynamics [80]

Input: Stochastic gradient oracle g, initial point $\boldsymbol{x}^0$, step size α_k

1. **For** $k = 0, 1, \ldots$
   sample noise ξ standard Gaussian
   update $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^l - \alpha_k \mathbf{g}^k + \sqrt{2\alpha_k}\xi$

Solving the outer problem: Gradient computation

Adversarial Training

Let $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$ be a model with parameters $\mathbf{x}$ and let $\{(\boldsymbol{a}_i, \mathbf{b}_i)\}_{i=1}^n$, with $\boldsymbol{a}_i \in \mathbb{R}^p$ and $\mathbf{b}_i$ be the corresponding labels. The adversarial training optimization problem is given by

$$\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} \underbrace{\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\| \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), \mathbf{b}_i) \right]}_{=: f_i(\mathbf{x})} \right\}.$$

Note that L is not continuously differentiable due to ReLU, max-pooling, etc.

Question

How can we compute the following stochastic gradient (i.e., $\mathbb{E}_i \nabla_{\mathbf{x}} f_i(\mathbf{x}) = \nabla_{\mathbf{x}} f_i(\mathbf{x})$ for $i \sim \mathrm{Uniform}\{1, \ldots, n\}$):

$$\nabla_{\mathbf{x}} f_i(\mathbf{x}) := \nabla_{\mathbf{x}} \left(\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\| \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), \mathbf{b}_i) \right)?$$

◦ **Challenge:** It involves differentiating with respect to a maximization.

Danskin's Theorem (1966): How do we compute the gradient?

Theorem ([21])

Let $\mathcal{S}$ be compact set, $\Phi : \mathbb{R}^p \times \mathcal{S}$ be continuous such that $\Phi(\cdot, \mathbf{y})$ is differentiable for all $\mathbf{y} \in \mathcal{S}$, and $\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y})$ be continuous on $\mathbb{R}^p \times \mathcal{S}$. Define

$$f(\mathbf{x}) := \max_{\mathbf{y} \in \mathcal{S}} \Phi(\mathbf{x}, \mathbf{y}), \qquad \mathcal{S}^\star(\mathbf{x}) := \arg\max_{\mathbf{y} \in \mathcal{S}} \Phi(\mathbf{x}, \mathbf{y}).$$

Let $\gamma \in \mathbb{R}^p$, and $\|\gamma\|_2 = 1$. The directional derivative $D_\gamma f(\bar{\mathbf{x}})$ of f in the direction γ at $\bar{\mathbf{x}}$ is given by

$$D_\gamma f(\bar{\mathbf{x}}) = \max_{\mathbf{y} \in \mathcal{S}^\star(\bar{\mathbf{x}})} \langle \gamma, \nabla_{\mathbf{x}} \Phi(\bar{\mathbf{x}}, \mathbf{y}) \rangle.$$

An immediate consequence

If $\boldsymbol{\delta}^\star \in \arg\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\| \le \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), \mathbf{b}_i)$ is unique, then we have

$$\nabla_{\mathbf{x}} f_i(\mathbf{x}) = \nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}^\star), \mathbf{b}_i)\,.$$

Optimized perturbations are typically not unique!

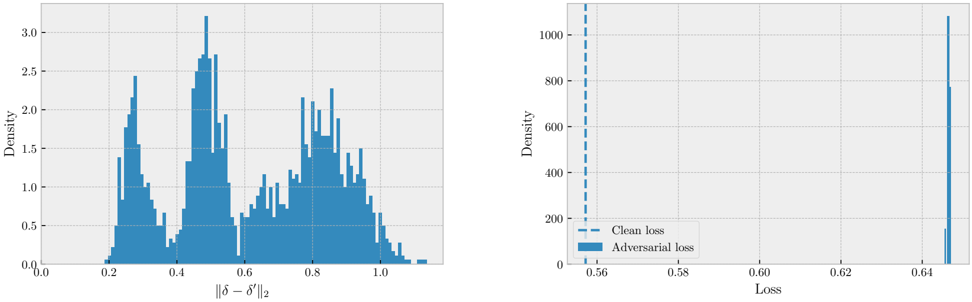


Figure: (*left*) Pairwise ℓ_2-distances between "optimized" perturbations with different initializations are bounded away from zero. (*right*) The losses of multiple perturbations on the same sample concentrate around a value much larger than the clean loss.

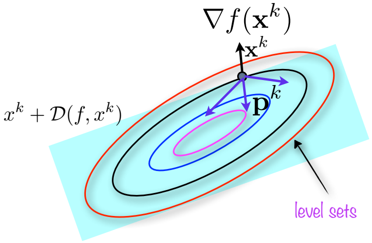
Theoretical foundations

	unique $\delta^\star$	non-unique $\delta^\star$
$\nabla_{\mathbf{x}}\Phi(\mathbf{x}, \delta^\star)$	$\nabla_{\mathbf{x}} f(\mathbf{x})$	descent direction [58]

Published as a conference paper at ICLR 2018

TOWARDS DEEP LEARNING MODELS RESISTANT TO ADVERSARIAL ATTACKS

Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu*
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{madry,amakelov,ludwigs,tsipras,avladu}@mit.edu



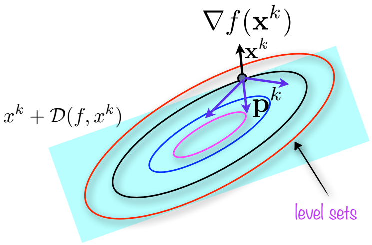
Theoretical foundations ?

	unique $\delta^\star$	non-unique $\delta^\star$
$\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \delta^\star)$	$\nabla_{\mathbf{x}} f(\mathbf{x})$	descent direction [58]

Published as a conference paper at ICLR 2018

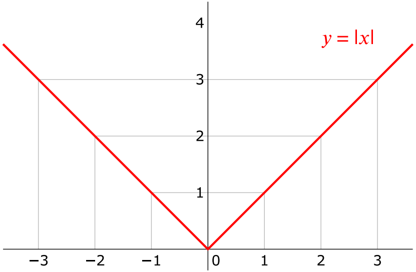
Towards Deep Learning Models Resistant to Adversarial Attacks

Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu*
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
`{madry,amakelov,ludwigs,tsipras,avladu}@mit.edu`



A counterexample

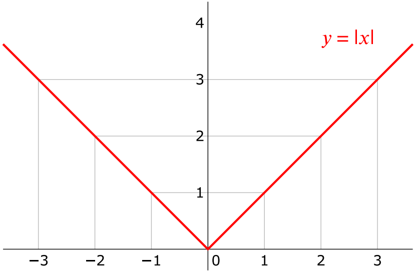
$$f(\mathbf{x}) := \max_{\boldsymbol{\delta}\in[-1,1]} \mathbf{x}\boldsymbol{\delta} = |\mathbf{x}| \, .$$



- We have $\mathcal{S} := [-1, 1]$ and $\Phi(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{x}\boldsymbol{\delta}$.
- At $\mathbf{x} = 0$, we have $\mathcal{S}^\star(0) = [-1, 1]$.
- We can choose $\delta = 1 \in \mathcal{S}^\star(0)$: $\Phi(\mathbf{x}, 1) = \mathbf{x}$.

A counterexample

$$f(\mathbf{x}) := \max_{\boldsymbol{\delta}\in[-1,1]} \mathbf{x}\boldsymbol{\delta} = |\mathbf{x}|\,.$$



- We have $\mathcal{S} := [-1, 1]$ and $\Phi(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{x}\boldsymbol{\delta}$.
- At $\mathbf{x} = 0$, we have $\mathcal{S}^\star(0) = [-1, 1]$.
- We can choose $\delta = 1 \in \mathcal{S}^\star(0)$: $\Phi(\mathbf{x}, 1) = \mathbf{x}$.
  - $-\nabla_{\mathbf{x}}\Phi(0, 1) = -1 \neq 0$.
  - Is -1 a descent direction at $\mathbf{x} = 0$?

Our understanding [Latorre, Krawczuk, Dadi, Pethick, Cevher, ICLR (2023)]

- The corollary in [58] is false (it is subtle!).
- We constructed a counter example & proposed an alternative way (DDi) of computing "the gradient":

	unique $\delta^\star$	non-unique $\delta^\star$
$\nabla_{\mathbf{x}}\Phi(\mathbf{x}, \delta^\star)$	$\nabla_{\mathbf{x}} f(\mathbf{x})$	could be ascent direction!

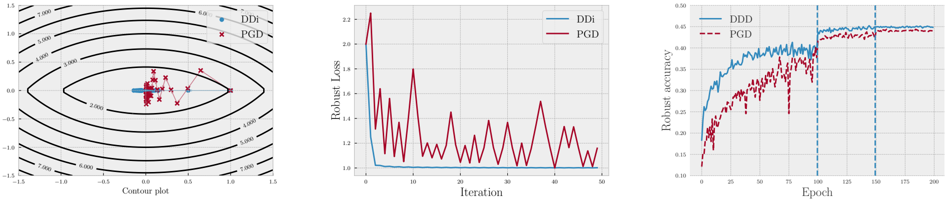


Figure: Left and middle pane: comparison DDi and PGD ([58]) on a synthetic problem. Right pane: DDi vs PGD on CIFAR10.

Comparison with the state-of-the-art

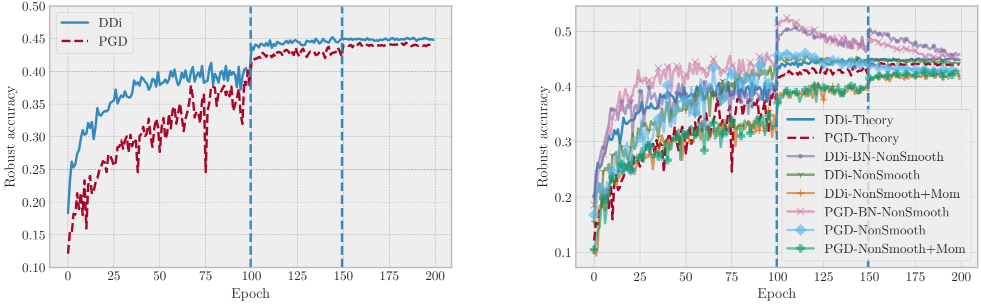
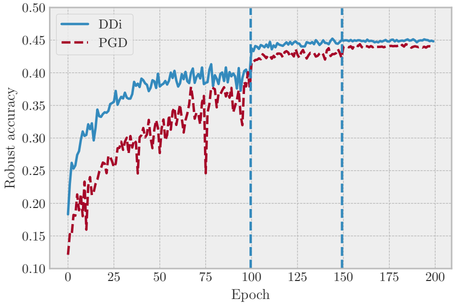


Figure: (left) PGD vs DDi on CIFAR10, in a setting covered by theory. (right) An ablation testing the effect of adding back the elements not covered by theory (BN,ReLU,momentum).

Comparison with the state-of-the-art



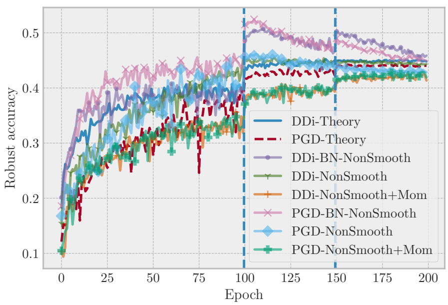
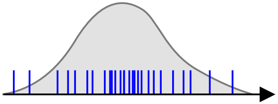


Figure: (left) PGD vs DDi on CIFAR10, in a setting covered by theory. (right) An ablation testing the effect of adding back the elements not covered by theory (BN,ReLU,momentum).

DDi + Graduate Student Descent may improve things (performance or catastrophic overfitting)?

Learning without concentration

- We can minimize $W_1\left(\hat{\mu}_n, h_{\mathbf{x}}\#\mathsf{p}_\Omega\right)$ with respect to $\mathbf{x}$.
- Figure: Empirical distribution (blue), $\hat{\mu}_n = \sum_{i=1}^n \delta_i$



A plug-in empirical estimator

Using the triangle inequality for Wasserstein distances we can upper bound in the follow way,

$$W_1(\mu^\natural, h_{\mathbf{x}}\#\mathsf{p}_\Omega) \leq W_1(\mu^\natural, \hat{\mu}_n) + W_1(\hat{\mu}_n, h_{\mathbf{x}}\#\mathsf{p}_\Omega), \tag{1}$$

where $\hat{\mu}_n$ is the empirical estimator of $\mu^\natural$ obtained from n independent samples from $\mu^\natural$.

Theorem (Slow convergence of empirical measures in 1-Wasserstein [79, 26])

Let $\mu^\natural$ be a measure defined on $\mathbb{R}^p$ and let $\hat{\mu}_n$ be its empirical measure. Then the $\hat{\mu}_n$ converges, in the worst case, at the following rate,

$$W_1(\mu^\natural, \hat{\mu}_n) \gtrsim n^{-1/p}. \tag{2}$$

Remarks:
- Using an empirical estimator in high-dimensions is terrible in the worst case.
- However, it does not directly say that $W_1\left(\mu^\natural, h_{\mathbf{x}}\#\mathsf{p}_\Omega\right)$ will be large.
- So we can still proceed and hope our parameterization interpolates harmlessly.

Duality of 1-Wasserstein

○ How do we get a sub-gradient of $W_1\left(\hat{\mu}_n, h_{\mathbf{x}}\#\mathrm{p}_\Omega\right)$ with respect to $\mathbf{x}$?

Theorem (Kantorovich-Rubinstein duality)

$$W_1(\mu,\nu) = \sup_{\mathtt{d}}\{\langle \mathtt{d},\mu\rangle - \langle \mathtt{d},\nu\rangle : \mathtt{d} \textit{ is 1-Lipschitz}\} \tag{3}$$

Remark: ○ $\mathtt{d}$ is the "dual" variable. In the literature, it is commonly referred to as the "discriminator."

Inner product is an expectation

$$\langle \mathtt{d},\mu\rangle = \int \mathtt{d}\mathrm{d}\mu = \int \mathtt{d}(\mathbf{a})\mathrm{d}\mu(\mathbf{a}) = \boldsymbol{E}_{\mathbf{a}\sim\mu}\left[\mathtt{d}(\mathbf{a})\right]. \tag{4}$$

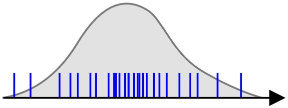
Kantorovich-Rubinstein duality applied to our objective

$$W_1\left(\hat{\mu}_n, h_{\mathbf{x}}\#\omega\right) = \sup\left\{\boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n}[\mathtt{d}(\mathbf{a})] - \boldsymbol{E}_{\mathbf{a}\sim h_{\mathbf{x}}\#\omega}[\mathtt{d}(\mathbf{a})] : \mathtt{d} \text{ is 1-Lipschitz}\right\} \tag{5}$$

Another minimax example: Generative adversarial networks (GANs)

◦ Ingredients:

- fixed *noise* distribution p_Ω (e.g., normal)
- target distribution $\hat{\mu}_n$ (natural images)
- $\mathcal{X}$ parameter class inducing a class of functions (generators)
- $\mathcal{Y}$ parameter class inducing a class of functions (dual variables)



Wasserstein GANs formulation [3]

Define a parameterized function $\mathrm{d}_{\mathbf{y}}(\mathbf{a})$, where $\mathbf{y} \in \mathcal{Y}$ such that $\mathrm{d}_{\mathbf{y}}(\mathbf{a})$ is 1-Lipschitz. In this case, the Wasserstein GAN training problem is given by

$$\min_{\mathbf{x}\in\mathcal{X}} \left(\max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n} \left[\mathrm{d}_{\mathbf{y}}(\mathbf{a})\right] - \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathrm{p}_\Omega} \left[\mathrm{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right] \right). \tag{6}$$

This problem is already captured by the template $\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})$. Note that the original problem is a direct non-smooth minimization problem and the Rubinstein-Kantarovic duality results in the minimax template.

Remarks:
◦ Cannot solve in a manner similar to adversarial training a la Danskin. Need a direct approach.
◦ Scalability, mode collapse, catastrophic forgetting. Heuristics galore!
◦ Enforce Lipschitz constraint weight clipping, gradient penalty, spectral normalization [3, 36, 62].

Abstract minmax formulation

Minimax formulation

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}), \tag{7}$$

where

- Φ is differentiable and nonconvex in $\mathbf{x}$ and nonconcave in $\mathbf{y}$,
- The domain is unconstrained, specifically $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{Y} = \mathbb{R}^n$.

◦ Key questions:

1. Where do the algorithms converge?
2. When do the algorithm converge?

Solving the minimax problem: Solution concepts

- Consider the unconstrained setting:

$$\boxed{\Phi^\star = \min_{\mathbf{x}} \max_{\mathbf{y}} \Phi(\mathbf{x}, \mathbf{y})}$$

- Goal: Find an LNE point $(\mathbf{x}^\star, \mathbf{y}^\star)$.

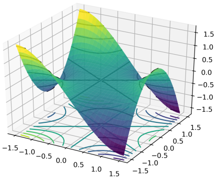


Figure: The monkey saddle $\Phi(x, y) = x^3 - 3xy^2$.

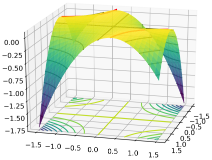


Figure: The weird saddle $\Phi(x, y) = -x^2y^2 + xy$.

Definition (Local Nash Equilibrium)

A pure strategy $(\mathbf{x}^\star, \mathbf{y}^\star)$ is called a local Nash equilibrium if

$$\Phi(\mathbf{x}^\star, \mathbf{y}) \leq \Phi(\mathbf{x}^\star, \mathbf{y}^\star) \leq \Phi(\mathbf{x}, \mathbf{y}^\star) \qquad \text{(LNE)}$$

for all $\mathbf{x}$ and $\mathbf{y}$ within some neighborhood of $\mathbf{x}^\star$ and $\mathbf{y}^\star$, i.e., $\|\mathbf{x} - \mathbf{x}^\star\| \leq \varepsilon$ and $\|\mathbf{y} - \mathbf{y}^\star\| \leq \varepsilon$ for some $\varepsilon > 0$.

Necessary conditions

Through a Taylor expansion around $\mathbf{x}^\star$ and $\mathbf{y}^\star$ one can show that a LNE implies

$$\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}), -\nabla_{\mathbf{y}} \Phi(\mathbf{x}, \mathbf{y}) = 0;$$
$$\nabla_{\mathbf{xx}} \Phi(\mathbf{x}, \mathbf{y}), -\nabla_{\mathbf{yy}} \Phi(\mathbf{x}, \mathbf{y}) \succeq 0.$$

Abstract minmax formulation

Minimax formulation

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}), \tag{8}$$

where

- Φ is differentiable and nonconvex in $\mathbf{x}$ and nonconcave in $\mathbf{y}$,
- The domain is unconstrained, specifically $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{Y} = \mathbb{R}^n$.

◦ Key questions:

1. Where do the algorithms converge?
2. When do the algorithm converge?

A buffet of negative results [22]

"Even when the objective is a Lipschitz and smooth differentiable function, deciding whether a min-max point exists, in fact even deciding whether an approximate min-max point exists, is NP-hard. More importantly, an approximate local min-max point of large enough approximation is guaranteed to exist, but finding one such point is PPAD-complete. The same is true of computing an approximate fixed point of the (Projected) Gradient Descent/Ascent update dynamics."

Basic algorithms for minimax

∘ Given $\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})$, define $V(\mathbf{z}) = [\nabla_{\mathbf{x}}\Phi(\mathbf{x},\mathbf{y}), -\nabla_{\mathbf{y}}\Phi(\mathbf{x},\mathbf{y})]$ with $\mathbf{z} = [\mathbf{x},\mathbf{y}]$.

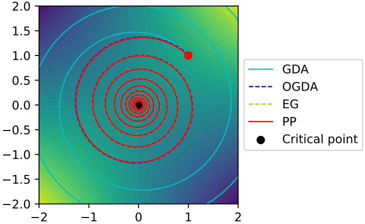


Figure: Trajectory of different algorithms for a simple bilinear game $\min_x \max_y xy$.

∘ (In)Famous algorithms

- Gradient Descent Ascent (GDA)
- Proximal point method (PPM) [69, 35]
- Extra-gradient (EG) [48]
- Optimistic GDA (OGDA) [94, 59]
- Reflected-Forward-Backward-Splitting (RFBS) [15]

∘ EG and OGDA are approximations of the PPM

- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^k)$.
- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^{k+1})$.
- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^k - \alpha V(\mathbf{z}^{k-1}))$.
- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha[2V(\mathbf{z}^k) - V(\mathbf{z}^{k-1})]$.
- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(2\mathbf{z}^k - \mathbf{z}^{k-1})$.

Where do the algorithms converge?

- Recall: Given $\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})$, define $V(\mathbf{z}) = [\nabla_{\mathbf{x}}\Phi(\mathbf{x},\mathbf{y}), -\nabla_{\mathbf{y}}\Phi(\mathbf{x},\mathbf{y})]$ with $\mathbf{z} = [\mathbf{x},\mathbf{y}]$.
- Given $V(\mathbf{z})$, define stochastic estimates of $V(\mathbf{z},\zeta) = V(\mathbf{z}) + U(\mathbf{z},\zeta)$, where
  - $U(\mathbf{z},\zeta)$ is a bias term,
  - We often have unbiasedness: $\boldsymbol{E}U(\mathbf{z},\zeta) = 0$,
  - The bias term can have bounded moments,
  - We often have bounded variance: $P(\| U(\mathbf{z},\zeta) \| \geq t) \leq 2\exp -\frac{t^2}{2\sigma^2}$ for $\sigma > 0$.
- An abstract template for generalized Robbins-Monro schemes, dubbed as $\mathcal{A}$:

$$\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha_k V(\mathbf{z}^k, \zeta^k).$$

The dessert section in the buffet of negative results: [41]

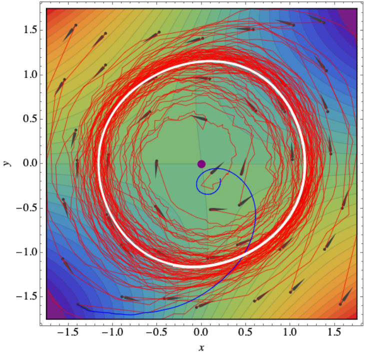
1. Bounded trajectories of $\mathcal{A}$ always converge to an internally chain-transitive (ICT) set.
2. Trajectories of $\mathcal{A}$ may converge with arbitrarily high probability to spurious attractors that contain no critical point of Φ.

Minimax is more difficult than just optimization [41]

- Internally chain-transitive (ICT) sets characterize the convergence of dynamical systems [9].
  - For optimization, {attracting ICT} $\equiv$ {solutions}
  - For minimax, {attracting ICT} $\equiv$ {solutions} $\cup$ {spurious sets}

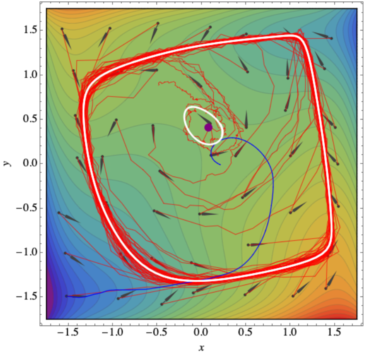
- "Almost" bilinear $\neq$ bilinear:

$$\Phi(x,y) = xy + \epsilon\phi(x), \phi(x) = \frac{1}{2}x^2 - \frac{1}{4}x^4$$



- The "forsaken" solutions:

$$\Phi(y,x) = y(x-0.5) + \phi(y) - \phi(x), \phi(u) = \frac{1}{4}u^2 - \frac{1}{2}u^4 + \frac{1}{6}u^6$$

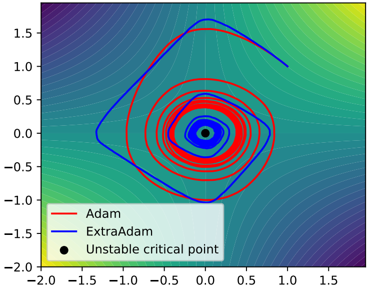


Minimax is more difficult than just optimization [41]

- Internally chain-transitive (ICT) sets characterize the convergence of dynamical systems [9].
  - For optimization, {attracting ICT} $\equiv$ {solutions}
  - For minimax, {attracting ICT} $\equiv$ {solutions} $\cup$ {spurious sets}

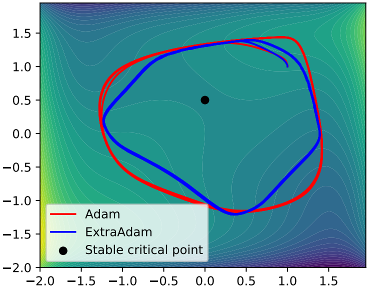
- "Almost" bilinear $\neq$ bilinear:

$$\Phi(x,y) = xy + \epsilon\phi(x), \phi(x) = \frac{1}{2}x^2 - \frac{1}{4}x^4$$



- The "forsaken" solutions:

$$\Phi(y,x) = y(x-0.5)+\phi(y)-\phi(x), \phi(u) = \frac{1}{4}u^2-\frac{1}{2}u^4+\frac{1}{6}u^6$$



When do the algorithms converge?

Assumption (weak Minty variational inequality)

For some $\rho \in \mathbb{R}$, *weak MVI implies*

$$\langle V(\mathbf{z}), \mathbf{z} - \mathbf{z}^\star \rangle \geqslant \rho \|V(\mathbf{z})\|^2, \quad \textit{for all } \mathbf{z} \in \mathbb{R}^n. \tag{9}$$

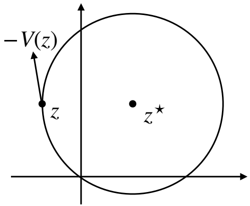
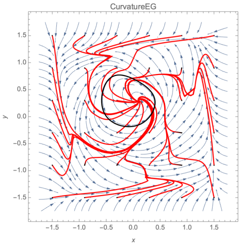


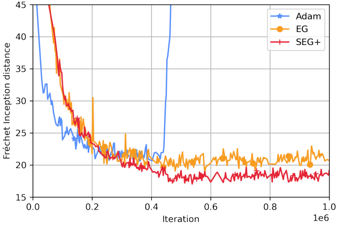
Figure: The operator $V(z)$ is allowed to point away from the solution by some amount when ρ is negative.

- A variant EG+ converges when $\rho > -\frac{1}{8L}$
  - Diakonikolas, Daskalakis, Jordan, AISTATS 2021.
- It still cannot handle the examples of [41].

- Complete picture under weak MVI (ICLR'22 and '23)
  - Pethick, Lalafat, Patrinos, Fercoq, and Cevher.
  - constrained and regularized settings with $\rho > -\frac{1}{2L}$
  - matching lower bounds
  - stochastic variants handling the examples of [41]
  - adaptive variants handling the examples of [41]



GANs with SEG+ [68]



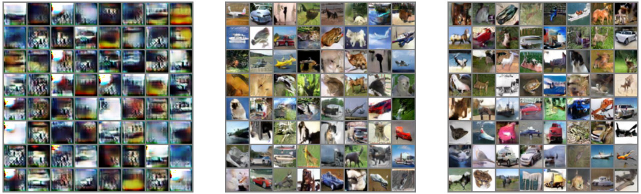


Figure: A performance comparison of GAN training by Adam, EG with stochastic gradients, and SEG+.

Robustness of the worst-performing class [67]

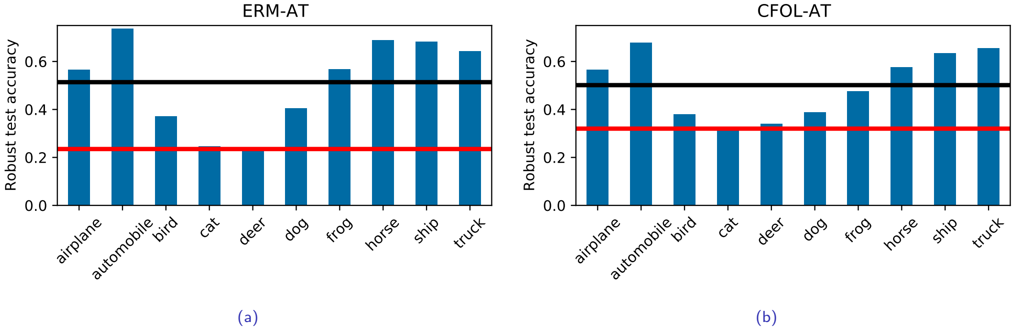


Figure: Robust test accuracy of (a) Empirical Risk Minimization and (b) the class focused online learning.

Code: `https://github.com/LIONS-EPFL/class-focused-online-learning-code`

Out of the frying pan into the fire

Original Formulation of Adversarial Training (I)

$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L(\mathbf{x}, \mathbf{a}+\boldsymbol{\delta}, \mathbf{b})\right]$$

Original Formulation of Adversarial Training (I)

$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L(\mathbf{x}, \mathbf{a}+\boldsymbol{\delta}, \mathbf{b})\right]$$

which loss L?

Original Formulation of Adversarial Training (II)

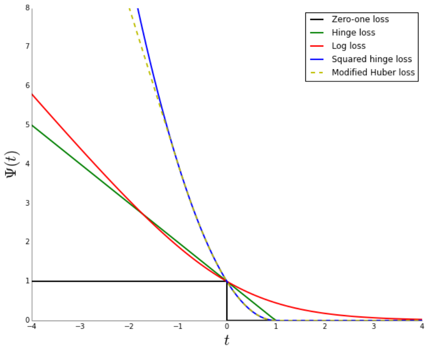
$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{01}(\mathbf{x}, \mathbf{a}+\boldsymbol{\delta}, \mathbf{b})\right]$$

Original Formulation of Adversarial Training (II)

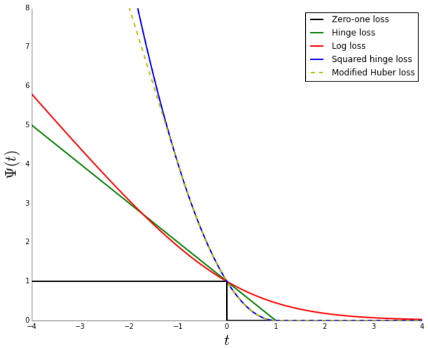
$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{01}(\mathbf{x}, \mathbf{a}+\boldsymbol{\delta}, \mathbf{b})\right]$$

$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{\mathsf{CE}}(\mathbf{x}, \mathbf{a}+\boldsymbol{\delta}, \mathbf{b})\right]$$

Surrogate-based optimization for Risk Minimization



Surrogate-based optimization for Risk Minimization

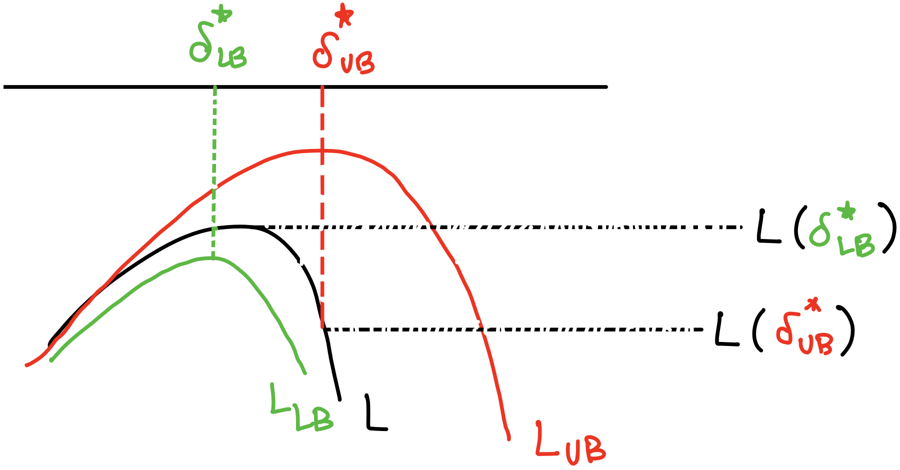


$$\mathbb{E}\left[L_{01}(\mathbf{x}^\star, \mathbf{a}, \mathbf{b})\right] \leq \min_{\mathbf{x}} \mathbb{E}\left[L_{\mathsf{CE}}(\mathbf{x}, \mathbf{a}, \mathbf{b})\right]$$

Adversary maximizes an upper bound (I)

$$L_{01}(\mathbf{x}, \mathbf{a} + \boldsymbol{\delta}^\star, \mathbf{b}) \leq \max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{\mathsf{CE}}(\mathbf{x}, \mathbf{a} + \boldsymbol{\delta}, \mathbf{b})$$

Adversary maximizes an upper bound (II)



Suppose $\mathbf{b}_1 = 1$, $c = 4$:

$$h_{\mathbf{x}}(\mathbf{a} + \boldsymbol{\delta}_A) = (0.26, 0.24, 0.25, 0.25)$$
$$h_{\mathbf{x}}(\mathbf{a} + \boldsymbol{\delta}_B) = (0.49, 0.51, 0, 0)$$

Why maximizing Cross-Entropy leads to weak adversaries

Suppose $\mathbf{b}_1 = 1$, $c = 4$:

$$h_{\mathbf{x}}(\mathbf{a} + \boldsymbol{\delta}_A) = (0.26, 0.24, 0.25, 0.25)$$
$$h_{\mathbf{x}}(\mathbf{a} + \boldsymbol{\delta}_B) = (0.49, 0.51, 0, 0)$$

$$L_{\mathsf{CE}}(\mathbf{x}, \mathbf{a} + \boldsymbol{\delta}_A, \mathbf{b}) = 1.38$$
$$L_{\mathsf{CE}}(\mathbf{x}, \mathbf{a} + \boldsymbol{\delta}_B, \mathbf{b}) = 1.18$$

Adversary's problem can be "solved" without using surrogates

Theorem (Reformulation of the Adversary's problem)

$$\boldsymbol{\delta}^\star \in \arg\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\le\epsilon} \max_{j\neq \mathbf{b}} h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta})_j - h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta})_{\mathbf{b}} \Rightarrow$$

$$\boldsymbol{\delta}^\star \in \arg\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\le\epsilon} L_{01}(\mathbf{x}, \mathbf{a}+\boldsymbol{\delta}, \mathbf{b})$$

Bilevel Optimization (BETA) [Robey,* Latorre,* Pappas, Hassani, Cevher(2023)][1]

$$\min_{\mathbf{x}\in\mathbf{X}} \frac{1}{n}\sum_{i=1}^{n} L_{\mathsf{CE}}(\mathbf{x}, \mathbf{a}_i + \boldsymbol{\delta}^\star_{i,j^\star}, \mathbf{b}_i)$$

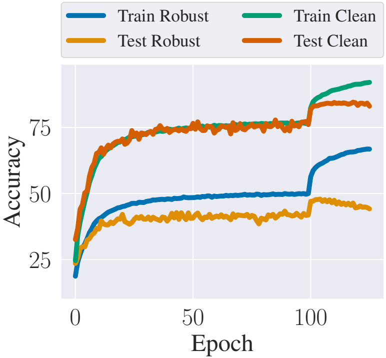
such that $\boldsymbol{\delta}^\star_{i,j} \in \arg\max_{\boldsymbol{\delta}:\,\|\boldsymbol{\delta}\|\leq\epsilon} h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta})_j - h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta})_{\mathbf{b}_i}$

$$j^\star \in \arg\max_{j\in[K]-\{\mathbf{b}_i\}} h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}_{i,j^\star})_j - h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}_{i,j^\star})_{\mathbf{b}_i}$$

[1] `https://infoscience.epfl.ch/record/302995` or `https://tinyurl.com/33yup77v`

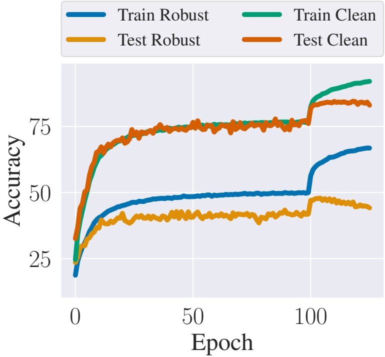
Practical Consequences of the Bilevel Formulation (I)

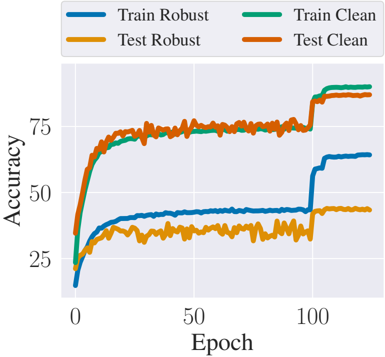
Figure: Learning curves of PGD^{10}-AT (Left) and BETA^{10}-AT



Practical Consequences of the Bilevel Formulation (I)

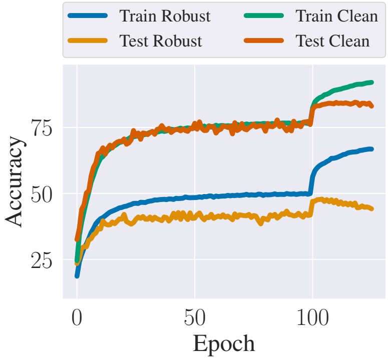
Figure: Learning curves of PGD^{10}-AT (Left) and BETA^{10}-AT (Right). Robust accuracy estimated with PGD^{20}

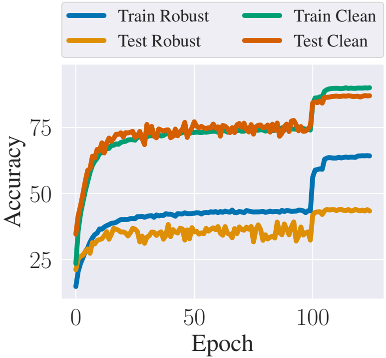




Practical Consequences of the Bilevel Formulation (I)

Figure: Learning curves of PGD^{10}-AT (Left) and BETA^{10}-AT (Right). Robust accuracy estimated with PGD^{20}





No Robust Overfitting occurs!

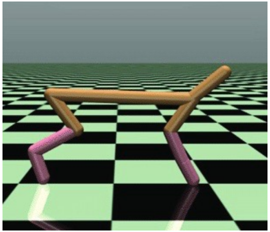
Practical Consequences of the Bilevel Formulation (I)

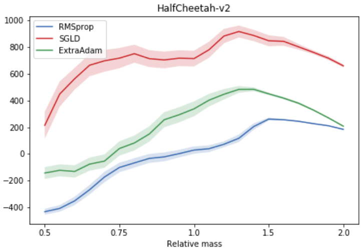
Training algorithm	Test accuracy					
	Clean		BETA^{10}		APGD	
	Best	Last	Best	Last	Best	Last
FGSM	81.96	75.43	40.30	0.04	41.56	0.00
PGD^{10}	83.71	83.21	43.64	40.21	44.36	42.62
TRADES^{10}	81.64	81.42	44.31	40.97	43.34	41.33
MART^{10}	78.80	77.20	44.81	41.22	45.00	42.90
BETA-AT^{5}	87.02	86.67	42.62	42.61	41.44	41.02
BETA-AT^{10}	85.37	85.30	44.54	44.36	44.32	44.12
BETA-AT^{20}	82.11	81.72	**46.91**	**45.90**	**45.27**	**45.25**

Figure: Adversarial performance on CIFAR-10.

Take home messages

- Even the simplified view of robust & adversarial ML is challenging
- $\min$-$\max$-type has spurious attractors with no equivalent concept in $\min$-type
- Not all step-size schedules are considered in our work: Possible to "converge" under some settings
- Other successful attempts[1] consider "mixed Nash" concepts[2]



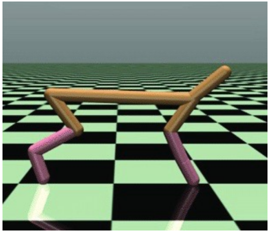


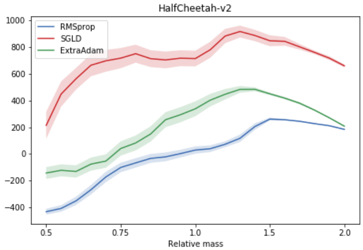
- Existing theory and methods for adversarial training is wrong!

[1] Y-P. Hsieh, C. Liu, and V. Cevher, "Finding mixed Nash equilibria of generative adversarial networks," International Conference on Machine Learning, 2019.
[2] K. Parameswaran, Y-T. Huang, Y-P. Hsieh, P. Rolland, C. Shi, V. Cevher, "Robust Reinforcement Learning via Adversarial Training with Langevin Dynamics," NeurIPS, 2020.

Take home messages

- Even the simplified view of robust & adversarial ML is challenging
- $\min$-$\max$-type has spurious attractors with no equivalent concept in $\min$-type
- Not all step-size schedules are considered in our work: Possible to "converge" under some settings
- Other successful attempts[1] consider "mixed Nash" concepts[2]



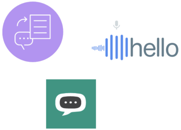


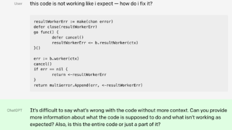
- Existing theory and methods for adversarial training is wrong! ... SAM too...

[1] Y-P. Hsieh, C. Liu, and V. Cevher, "Finding mixed Nash equilibria of generative adversarial networks," International Conference on Machine Learning, 2019.
[2] K. Parameswaran, Y-T. Huang, Y-P. Hsieh, P. Rolland, C. Shi, V. Cevher, "Robust Reinforcement Learning via Adversarial Training with Langevin Dynamics," NeurIPS, 2020.

Break

Over-parameterization: more parameters than training data



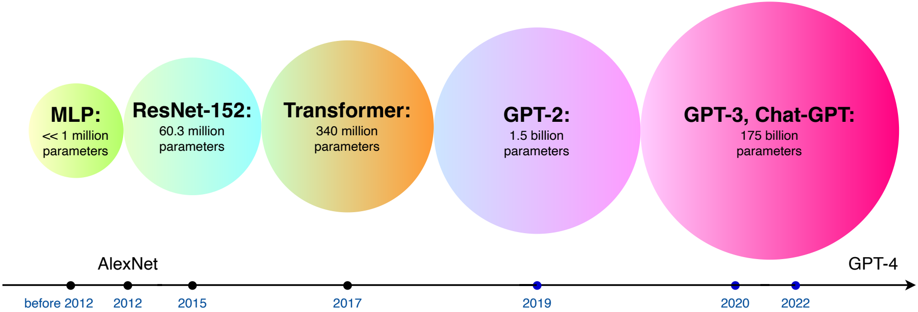


User this code is not working like i expect — how do i fix it?

```
resultWorkerErr := make(chan error)
defer close(resultWorkerErr)
go func() {
	defer cancel()
	resultWorkerErr <- b.resultWorker(ctx)
}()

err := b.worker(ctx)
cancel()
if err == nil {
	return <-resultWorkerErr
}
return multierror.Append(err, <-resultWorkerErr)
```

ChatGPT It's difficult to say what's wrong with the code without more context. Can you provide more information about what the code is supposed to do and what isn't working as expected? Also, is this the entire code or just a part of it?



Over-parameterization: more parameters than training data

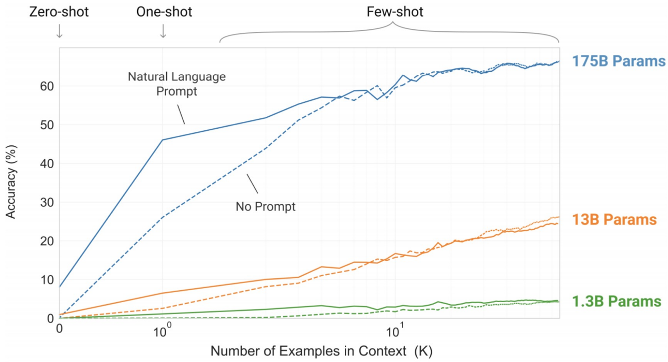


Figure: Larger models make increasingly efficient use of in-context information: source from Open AI.

Recall DNNs: the good in fitting ...

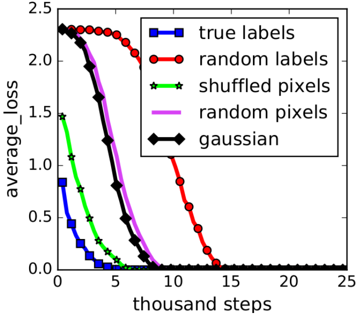
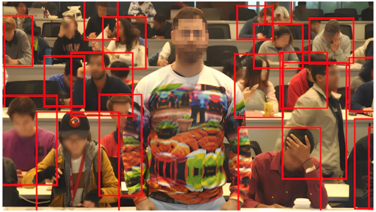


Figure: DNN Training curves on CIFAR10, from [90]

- A gap between theory and practice:
  - DNNs can fit random labels
  - SGD: zero training error and low test error

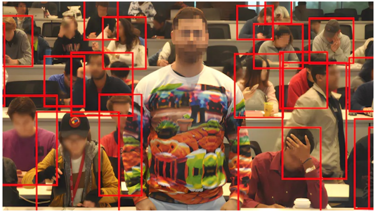
Recall DNNs: the bad in robustness...



(a) Invisibility [83]

(b) Stop sign classified as 45 mph sign [28]

Recall DNNs: the bad in robustness...

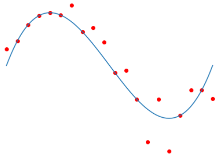


(a) Invisibility [83]

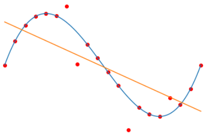
(b) Stop sign classified as 45 mph sign [28]

the ugly in over-parameterization?

A toy example: curve fitting

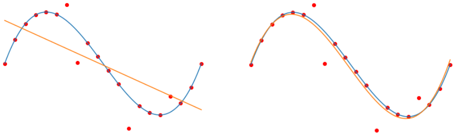


A toy example: curve fitting



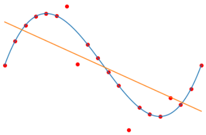
(a) under-fitting

A toy example: curve fitting

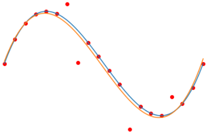


(a) under-fitting (b) sweet spot

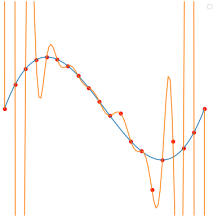
A toy example: curve fitting



(a) under-fitting

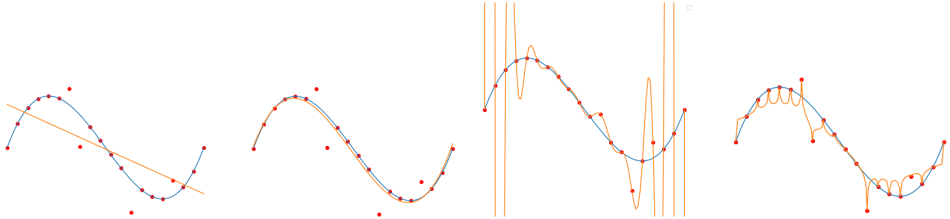


(b) sweet spot



(c) overfitting

A toy example: curve fitting



(a) under-fitting (b) sweet spot (c) overfitting (d) benign overfitting [6]

Figure: Test performance on curve fitting: source from Open AI.

Recall: the formulation of FCNN

$$h^{(0)}(\mathbf{a}) = \mathbf{a},$$

$$h^{(l)}(\mathbf{a}) = \underset{\text{activation}}{\overset{\downarrow}{\sigma}} \left(\left[\overset{\text{weight}}{\overset{\downarrow}{\mathbf{X}_l}} \right] \left[\overset{\text{input features}}{\overset{\downarrow}{h^{(l-1)}(\mathbf{a})}} \right] \right), \qquad (L\text{-Layer NN})$$

$$h_{\mathbf{x}}(\mathbf{a}) = h^{(L)}(\mathbf{a}) = \frac{1}{\alpha} \sigma \left(\mathbf{X}_L h^{(L-1)}(\mathbf{a}) \right), \qquad \mathbf{x} := [\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_L].$$

- Elements of NN architectures we will discuss in the sequel:
  - Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m \times p}$, $\mathbf{X}_L \in \mathbb{R}^{1 \times m}$, $\mathbf{X}_l \in \mathbb{R}^{m \times m}$ for $l = 2, 3, \cdots, L-1$ (weights).
  - Initialization: $\mathbf{X}_1 \sim \mathcal{N}(0, \beta_1^2)$, $\mathbf{X}_L \sim \mathcal{N}(0, \beta_L^2)$, $\mathbf{X}_l \sim \mathcal{N}(0, \beta^2)$ for $l = 2, 3, \cdots, L-1$ (weights).
  - Activation function ReLU: $\sigma(\cdot) = \max(\cdot, 0) : \mathbb{R} \to \mathbb{R}$.
  - Without loss of generality, we will avoid the bias variables in the sequel.

Initialization in deep ReLU NNs

∘ Initialization: $\mathbf{X}_1 \sim \mathcal{N}(0, \beta_1^2)$, $\mathbf{X}_L \sim \mathcal{N}(0, \beta_L^2)$, $\mathbf{X}_l \sim \mathcal{N}(0, \beta^2)$ for $l = 2, 3, \cdots, L-1$ (weights).

Table: Some commonly used initializations in neural networks.

Initialization name	β_1^2	β^2	β_L^2	α
LeCun [50]	$\frac{1}{p}$	$\frac{1}{m}$	$\frac{1}{m}$	1
He [38]	$\frac{2}{p}$	$\frac{2}{m}$	$\frac{2}{m}$	1
NTK [2]	$\frac{2}{m}$	$\frac{2}{m}$	1	1
Xavier [33]	$\frac{2}{m+p}$	$\frac{1}{m}$	$\frac{2}{m+1}$	1
Mean-field [60]	1	1	1	m
E et al. [27]	1	1	β_c^2	1

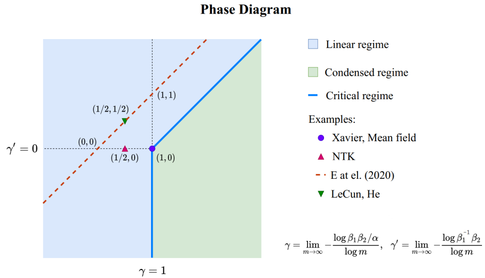


Figure: Phase diagram of two-layer ReLU NNs at infinite-width limit in [56].

Lazy training

Definition (Lazy-training (Linear) regime [56])

Define an L-layer fully-connected ReLU NN via (L-Layer NN). After training time t, as $m \to \infty$, if the following condition holds

$$\sup_{t \in [0,+\infty)} \frac{\|\boldsymbol{X}_l(t) - \boldsymbol{X}_l(0)\|_2}{\|\boldsymbol{X}_l(0)\|_2} \to 0, \quad \forall l \in [L] \,.$$

then the NN training dynamics falls into the lazy-training regime.

Remarks:
- ∘ In this regime, training h and h_0 is equivalent if taking Taylor expansion.
- ∘ Which conditions allow for lazy training to occur ?

Lazy training: a consequence of overparametrization or scaling?

Theorem (Lazy training for two-layer ReLU networks [18], modified version)

Two layer networks $h(\mathbf{a}, \{\mathbf{x}, \mathbf{v}\}) : \mathbf{a} \mapsto \alpha(m) \sum_{j=1}^{m} v_j \mathit{ReLU}(\mathbf{x}_j^\top \mathbf{a})$ *with Gaussian initialization* $v_i, \mathbf{x}_i \sim \mathcal{N}(0, \beta^2)$ *will fall within the lazy regime as long as*

$$\lim_{m \to \infty} m\beta = \infty \,.$$

Remarks:
- The loss changes a lot but the neural network output changes little.
- Other conditions for deep neural networks can be found here [18, 7].

Lazy training regime: visualization

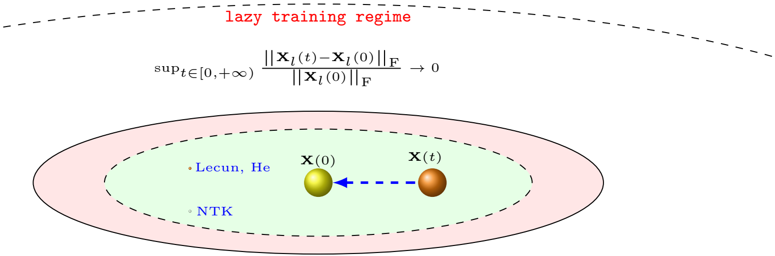
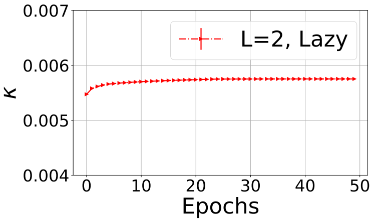
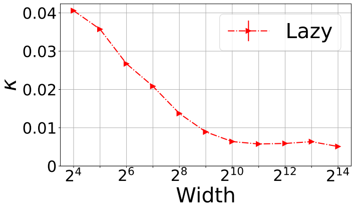


Figure: Training dynamics of two-layer ReLU NNs under different initializations [46, 19, 57].

Lazy training regime: experiments

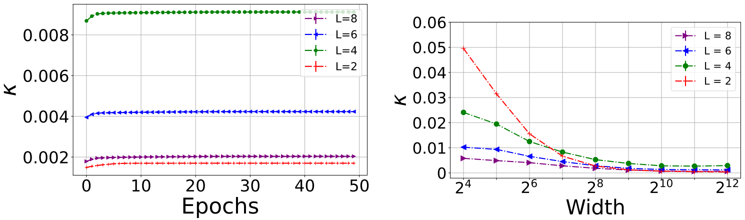
$$\text{lazy training ratio } \kappa := \frac{\sum_{l=1}^{L} \|\boldsymbol{X}_l(t) - \boldsymbol{X}_l(0)\|_{\mathrm{F}}}{\sum_{l=1}^{L} \|\boldsymbol{X}_l(0)\|_{\mathrm{F}}}$$





Lazy training regime: experiments

$$\text{lazy training ratio } \kappa := \frac{\sum_{l=1}^{L} \|\boldsymbol{X}_l(t) - \boldsymbol{X}_l(0)\|_{\mathrm{F}}}{\sum_{l=1}^{L} \|\boldsymbol{X}_l(0)\|_{\mathrm{F}}}$$



Non-lazy training regime: visualization

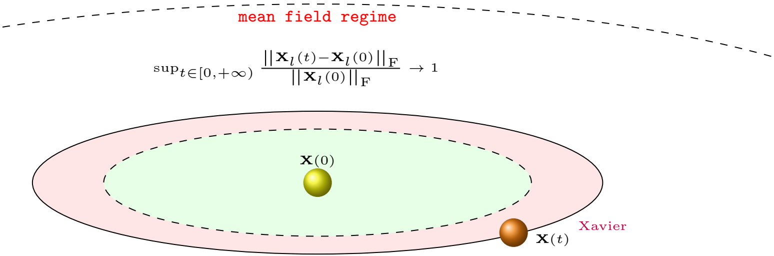


Figure: Training dynamics of two-layer ReLU NNs under different initializations [46, 19, 57].

Non-lazy training regime: visualization

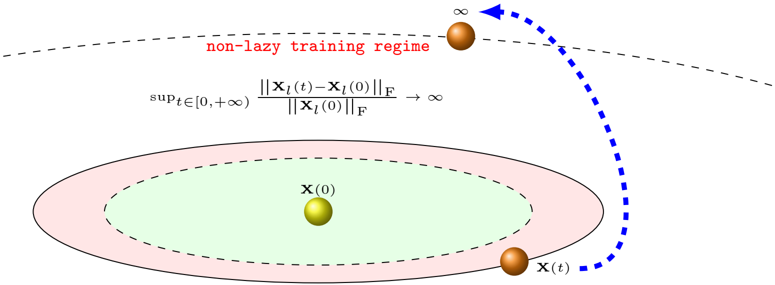


Figure: Training dynamics of two-layer ReLU NNs under different initializations [46, 19, 57].

Neural Tangent Kernel [46]

◦ Define feature mapping $\mathbf{a} \mapsto \frac{\partial h}{\partial \mathbf{x}}(\mathbf{a}, \mathbf{x}_0)$, the (empirical) neural tangent kernel is defined as

$$\Theta(\mathbf{a}_i, \mathbf{a}_j) := \langle \nabla_{\mathbf{x}} h(\mathbf{a}_i, \mathbf{x}), \nabla_{\mathbf{x}} h(\mathbf{a}_j, \mathbf{x}) \rangle \,, \forall i, j \in [n] \,.$$

Training dynamics

Under NTK initialization and large enough width, we have

$$\lim_{\text{width}\to\infty} \Theta_{\mathbf{x}(0)}(\mathbf{a}_i, \mathbf{a}_j) = \mathbb{E}_{\mathbf{x}}[\Theta_{\mathbf{x}(0)}(\mathbf{a}_i, \mathbf{a}_j)] =: K_\infty \,.$$

Under the squared loss, the dynamics of $h(\mathbf{a}, \mathbf{x})$ is equivalent to kernel regression

$$\dot{h}(\mathbf{a}, \mathbf{x}(t)) = \nabla_{\mathbf{x}} h(\mathbf{a}, \mathbf{x}) \dot{\mathbf{x}}(t) = K_\infty(\mathbf{a}, \mathbf{a}_i)(h(\mathbf{a}, \mathbf{x}(t)) - b) \,.$$

Neural Tangent Kernel [46]

◦ Define feature mapping $\mathbf{a} \mapsto \frac{\partial h}{\partial \mathbf{x}}(\mathbf{a}, \mathbf{x}_0)$, the (empirical) neural tangent kernel is defined as

$$\Theta(\mathbf{a}_i, \mathbf{a}_j) := \langle \nabla_{\mathbf{x}} h(\mathbf{a}_i, \mathbf{x}), \nabla_{\mathbf{x}} h(\mathbf{a}_j, \mathbf{x}) \rangle, \forall i, j \in [n].$$

Training dynamics

Under NTK initialization and large enough width, we have

$$\lim_{\text{width} \to \infty} \Theta_{\mathbf{x}(0)}(\mathbf{a}_i, \mathbf{a}_j) = \mathbb{E}_{\mathbf{x}}[\Theta_{\mathbf{x}(0)}(\mathbf{a}_i, \mathbf{a}_j)] =: K_\infty.$$

Under the squared loss, the dynamics of $h(\mathbf{a}, \mathbf{x})$ is equivalent to kernel regression

$$\dot{h}(\mathbf{a}, \mathbf{x}(t)) = \nabla_{\mathbf{x}} h(\mathbf{a}, \mathbf{x}) \dot{\mathbf{x}}(t) = K_\infty(\mathbf{a}, \mathbf{a}_i)(h(\mathbf{a}, \mathbf{x}(t)) - b).$$

Remarks:
- ◦ NTK stays unchanged during training
- ◦ General loss functions: equivalence between infinite NNs and kernel methods [17]
  - e.g., NN trained by soft margin loss vs. SVM trained by subgradient descent

Convolutional neural tangent kernel

- Convolutional neural networks (CNNs) [4]
  - without global average pooling (GAP)
  - with GAP, without training the first/last year

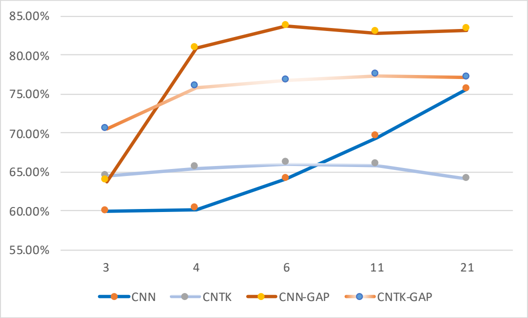


Figure: Classification accuracies of CNNs and CNTK on the CIFAR-10 dataset [4].

Remarks:
- This performance is below the accuracy of finite width networks ($> 98\%$).
- NTK for general architectures, e.g., RNNs [1], GNNs [24, 49], PNNs [82]

Optimization and generalization by NTK

Theorem (optimization and generalization [2, 13])

For a DNN with a large enough width trained by (S)GD, under proper data assumption and step-size η,

- *global convergence*

$$L(\mathbf{x}(t)) \leq [1 - \eta\lambda_{\min}(K_\infty)]^t L(\mathbf{x}(0))\,, \qquad whp.$$

where $\lambda_{\min}(K_\infty)$ is the minimum eigenvalue of K_∞.

Optimization and generalization by NTK

Theorem (optimization and generalization [2, 13])

For a DNN with a large enough width trained by (S)GD, under proper data assumption and step-size η,

- *global convergence*

$$L(\mathbf{x}(t)) \leq [1 - \eta\lambda_{\min}(K_\infty)]^t L(\mathbf{x}(0)) \,, \quad whp.$$

  where $\lambda_{\min}(K_\infty)$ is the minimum eigenvalue of K_∞.

- *generalization guarantee*

$$\textit{generalization error} \lesssim \mathcal{O}\left(\frac{1}{\sqrt{\lambda_{\min}(K_\infty)}}\right) + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) , whp.$$

Optimization and generalization by NTK

Theorem (optimization and generalization [2, 13])

For a DNN with a large enough width trained by (S)GD, under proper data assumption and step-size η,

- *global convergence*

$$L(\mathbf{x}(t)) \le [1 - \eta \lambda_{\min}(K_\infty)]^t L(\mathbf{x}(0)) \,, \quad whp.$$

  where $\lambda_{\min}(K_\infty)$ is the minimum eigenvalue of K_∞.

- *generalization guarantee*

$$\text{generalization error } \lesssim \mathcal{O}\left(\frac{1}{\sqrt{\lambda_{\min}(K_\infty)}} \right) + \mathcal{O}\left(\frac{1}{\sqrt{n}} \right) , whp.$$

Remarks: ◦ The objective is "almost convex".

◦ The minimum eigenvalue of NTK plays an important role!

How much overparametrization of fully-connected NNs is enough?

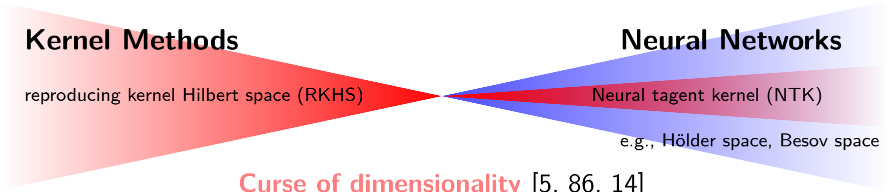
Reference	Number of parameters	Depth L	Result
[53]	$\tilde{\Omega}(\text{poly}(n))$	1	(S)GD global convergence
[2, 96]	$\tilde{\Omega}(\text{poly}(n, L))$	Any L	(S)GD global convergence
[23]	$\tilde{\Omega}(n^8 2^{O(L)})$	Any L	(S)GD global convergence
[97]	$\tilde{\Omega}(n^8 L^{12})$	Any L	(S)GD global convergence
[47]	$\tilde{\Omega}(n)$ (Training last layer)	Any L	(S)GD global convergence
[11]	$\tilde{\Omega}(n)$ (Training all layers)	Any L	(S)GD global convergence

Table: Summary of results on overparametrization. Minimum number of parameters required as a function of data size n and depth L. [11]: smooth activation function; Lipschitz concentrated data; a loose pyramidal topology.

Remarks: ◦ Practical datasets are structured: the width need no be large for a good approximation [63]

Function space: from kernel methods to neural networks

- Feature mapping $\mathbf{a} \mapsto \frac{\partial h}{\partial \mathbf{x}}(\mathbf{a}, \mathbf{x}_0)$: captures the first-order approximation of NN's training.

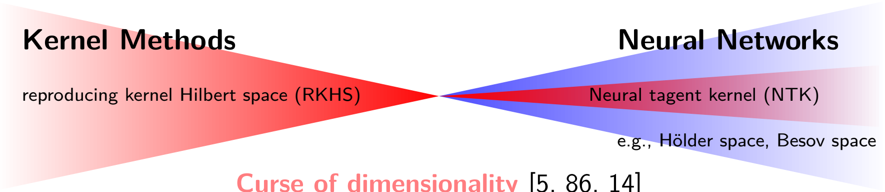


Curse of dimensionality [5, 86, 14]

efficiently approximate non-smooth functions?

Function space: from kernel methods to neural networks

- Feature mapping $\mathbf{a} \mapsto \frac{\partial h}{\partial \mathbf{x}}(\mathbf{a}, \mathbf{x}_0)$: captures the first-order approximation of NN's training.



Curse of dimensionality [5, 86, 14]

efficiently approximate non-smooth functions?

What can linearized neural networks actually say about CV tasks?

What can we benefit from NTK for computer vision?

- efficient algorithm
  - fine-tuning: gradient as features [63, 87]
  - efficient training in low-dimensional spaces [52], neural networks pruning [54]
  - robustness: generate adversarial examples [75], black-box generalization attack [88]
  - small-scale dataset [66], dataset distillation [65]
  - image denoising [73]
  - neural architecture search in a "train-free" fashion [92, 16]

What can we benefit from NTK for computer vision?

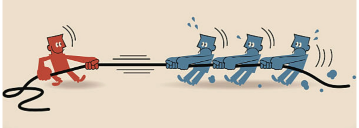
- efficient algorithm
  - fine-tuning: gradient as features [63, 87]
  - efficient training in low-dimensional spaces [52], neural networks pruning [54]
  - robustness: generate adversarial examples [75], black-box generalization attack [88]
  - small-scale dataset [66], dataset distillation [65]
  - image denoising [73]
  - neural architecture search in a "train-free" fashion [92, 16]
- understanding (and beyond)
  - adversarial training [75]
  - spectral bias [70]
  - hyperparameter transfer [85]

What can we benefit from NTK for computer vision?

- efficient algorithm
  - fine-tuning: gradient as features [63, 87]
  - efficient training in low-dimensional spaces [52], neural networks pruning [54]
  - robustness: generate adversarial examples [75], black-box generalization attack [88]
  - small-scale dataset [66], dataset distillation [65]
  - image denoising [73]
  - neural architecture search in a "train-free" fashion [92, 16]
- understanding (and beyond)
  - adversarial training [75]
  - spectral bias [70]
  - hyperparameter transfer [85]

Over-parameterization helps or hurts robustness?

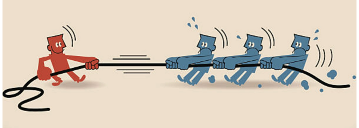
Helps! [12]



Hurts! [81, 43]

Over-parameterization helps or hurts robustness?

Helps! [12]

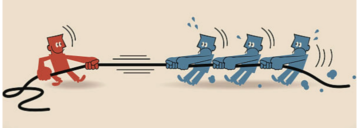


Hurts! [81, 43]

- initialization (e.g., lazy training, non-lazy training)
- architecture (e.g., width, depth)

Over-parameterization helps or hurts robustness?

Helps! [12]



Hurts! [81, 43]

- initialization (e.g., lazy training, non-lazy training)
- architecture (e.g., width, depth)

Definition (perturbation stability [93])

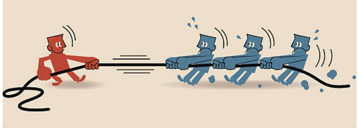
The perturbation stability of a ReLU DNN $h_{\mathbf{x}}(\mathbf{a})$ is:

$$\mathscr{P}(h, \epsilon) = \mathbb{E}_{\mathbf{a}, \hat{\mathbf{a}}, \mathbf{x}} \left\| \nabla_{\mathbf{a}} h_{\mathbf{x}}(\mathbf{a})^\top (\mathbf{a} - \hat{\mathbf{a}}) \right\|_2, \quad \forall \boldsymbol{a} \sim \mathcal{D}_A, \quad \hat{\mathbf{a}} \sim \text{Unif}(\mathbb{B}(\epsilon, \mathbf{a}))\,.$$

where ϵ is the perturbation radius.

Over-parameterization helps or hurts robustness?

Helps! [12]



Hurts! [81, 43]

- initialization (e.g., lazy training, non-lazy training)
- architecture (e.g., width, depth)

Definition (perturbation stability: lazy training regime)

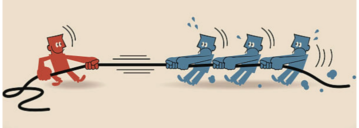
The perturbation stability of a ReLU DNN $h_{\mathbf{x}}(\mathbf{a})$ is

$$\mathscr{P}(h,\epsilon) = \mathbb{E}_{\mathbf{a},\hat{\mathbf{a}},\mathbf{x}(0)} \left\| \nabla_{\mathbf{a}} h_{\mathbf{x}}(\mathbf{a})^\top (\mathbf{a}-\hat{\mathbf{a}}) \right\|_2, \quad \forall \boldsymbol{a} \sim \mathcal{D}_A, \quad \hat{\mathbf{a}} \sim \text{Unif}(\mathbb{B}(\epsilon,\mathbf{a}))\,.$$

where ϵ is the perturbation radius.

Over-parameterization helps or hurts robustness?

Helps! [12]



Hurts! [81, 43]

- initialization (e.g., lazy training, non-lazy training)
- architecture (e.g., width, depth)

Definition (perturbation stability: non-lazy training regime)

The perturbation stability of a ReLU DNN $h_{\mathbf{x}}(\mathbf{a})$ is

$$\mathscr{P}(h, \epsilon) = \mathbb{E}_{\mathbf{a}, \hat{\mathbf{a}}} \left\| \nabla_{\mathbf{a}} h_{\mathbf{x}}(\mathbf{a})^\top (\mathbf{a} - \hat{\mathbf{a}}) \right\|_2, \quad \forall \mathbf{a} \sim \mathcal{D}_A, \quad \hat{\mathbf{a}} \sim \text{Unif}(\mathbb{B}(\epsilon, \mathbf{a}))\,.$$

where ϵ is the perturbation radius.

Main results (Lazy-training regime)

Theorem: perturbation stability $\lesssim \mathrm{Func}(m, L, \beta)$

Assumption	Initialization	Our bound for $\mathscr{P}(f,\epsilon)/\epsilon$	Trend of width m [1]	Trend of depth L [1]
$\|\boldsymbol{x}\|_2 = 1$	Lecun initialization	$\left(\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}\right)\left(\frac{\sqrt{2}}{2}\right)^{L-2}$	$\nearrow \searrow$	$\searrow$
	He initialization	$\sqrt{\frac{L^3 m}{d}} e^{-m/L^3} + \sqrt{\frac{1}{d}}$	$\nearrow \searrow$	$\nearrow$
	NTK initialization	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + 1$	$\nearrow \searrow$	$\nearrow$

[1] The larger perturbation stability means worse average robustness.

◦ Takeaway messages: **the good (width), the bad (depth), the ugly (initialization)**

Main results (Lazy-training regime)

Theorem: perturbation stability $\lesssim \mathrm{Func}(m, L, \beta)$

Assumption	Initialization	Our bound for $\mathscr{P}(f,\epsilon)/\epsilon$	Trend of width m [1]	Trend of depth L [1]
$\\|\boldsymbol{x}\\|_2 = 1$	Lecun initialization	$\left(\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}\right)(\frac{\sqrt{2}}{2})^{L-2}$	$\nearrow \searrow$	$\searrow$
	He initialization	$\sqrt{\frac{L^3 m}{d}} e^{-m/L^3} + \sqrt{\frac{1}{d}}$	$\nearrow \searrow$	$\nearrow$
	NTK initialization	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + 1$	$\nearrow \searrow$	$\nearrow$

[1] The larger perturbation stability means worse average robustness.

- Takeaway messages: **the good (width), the bad (depth), the ugly (initialization)**
  - width **helps** robustness in the over-parameterized regime

Main results (Lazy-training regime)

Theorem: perturbation stability $\lesssim \mathrm{Func}(m, L, \beta)$

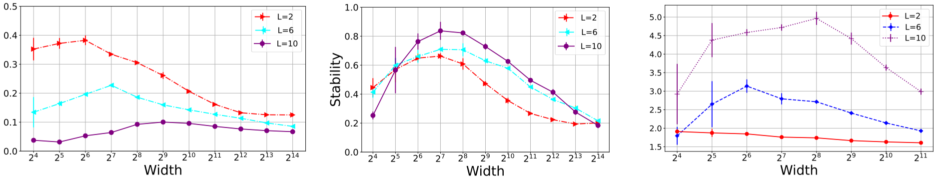
Assumption	Initialization	Our bound for $\mathscr{P}(f,\epsilon)/\epsilon$	Trend of width m [1]	Trend of depth L [1]
$\|\boldsymbol{x}\|_2 = 1$	Lecun initialization	$\left(\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}\right)(\frac{\sqrt{2}}{2})^{L-2}$	$\nearrow \searrow$	$\searrow$
	He initialization	$\sqrt{\frac{L^3 m}{d}} e^{-m/L^3} + \sqrt{\frac{1}{d}}$	$\nearrow \searrow$	$\nearrow$
	NTK initialization	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + 1$	$\nearrow \searrow$	$\nearrow$

[1] The larger perturbation stability means worse average robustness.

- Takeaway messages: **the good (width), the bad (depth), the ugly (initialization)**
  - width **helps** robustness in the over-parameterized regime
  - depth **helps** robustness in Lecun initialization but **hurts** robustness in He/NTK initialization

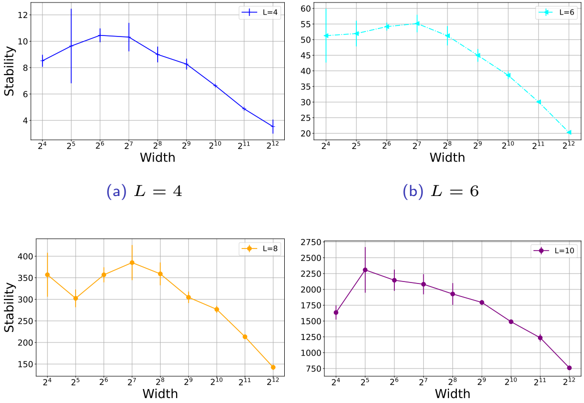
Experiments: lazy training experiment for FCNN

Metrics	Ours (NTK initialization)	[81]	[43]
$\mathscr{P}(\boldsymbol{f}, \epsilon)/\epsilon$	$\sqrt{\frac{L^3 m}{p} e^{-m/L^3} + 1}$	$L^2 m^{1/3} \sqrt{\log m} + \sqrt{mL}$	$2^{\frac{3L-5}{2}} \sqrt{L}$



(a) LeCun initialization (b) He initialization (c) NTK initialization

Experiments: lazy training experiment for CNN



(a) $L = 4$ (b) $L = 6$

(c) $L = 8$ (d) $L = 10$

Figure: Relationship between the *perturbation stability* and width of CNN under He initialization for different depths of $L = 4, 6, 8$ and 10. More experimental results on ResNet can be found in [93].

Main results (Non-lazy training regime)

A sufficient condition for DNNs

For large enough m and $m \gg p$, w.h.p, DNNs fall into non-lazy training regime if $\alpha \gg (m^{3/2} \sum_{i=1}^{L} \beta_i)^L$.

Remarks: ◦ $L = 2$, $\alpha = 1$, $\beta_1 = \beta_2 = \beta \sim \frac{1}{m^c}$ with $c > 1.5$

Main results (Non-lazy training regime)

A sufficient condition for DNNs

For large enough m and $m \gg p$, w.h.p, DNNs fall into non-lazy training regime if $\alpha \gg (m^{3/2} \sum_{i=1}^{L} \beta_i)^L$.

Remarks: ◦ $L = 2$, $\alpha = 1$, $\beta_1 = \beta_2 = \beta \sim \frac{1}{m^c}$ with $c > 1.5$

Theorem (non-lazy training regime for two-layer NNs)

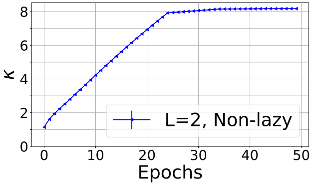
Under this setting with $m \gg n^2$ and standard assumptions, then

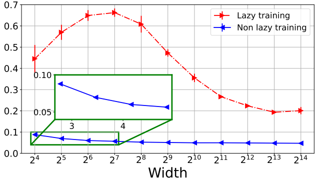
$$\textit{perturbation stability} \leq \widetilde{\mathcal{O}}\left(\frac{n}{m^{c+1.5}}\right), \ whp.$$

Remarks: ◦ Width helps robustness in the over-parameterized regime in both lazy/non-lazy training regime

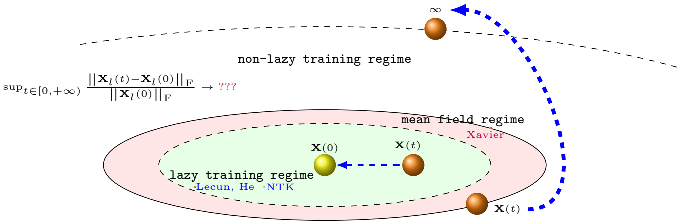
Experiment: Non-lazy training regime

$$\text{lazy training ratio } \kappa := \frac{\sum_{l=1}^{L} \|\boldsymbol{X}_l(t) - \boldsymbol{X}_l(0)\|_{\mathrm{F}}}{\sum_{l=1}^{L} \|\boldsymbol{X}_l(0)\|_{\mathrm{F}}}$$





	good	bad	ugly
neural networks	performance	analysis	over-parameterization
generalization	benign overfitting	catastrophic overfitting	model complexity
robustness	width	depth	initialization



Break

Neural Architecture Search (NAS) [95]

- An architecture has a significant impact on the performance and the inductive bias of the model [39, 77, 78, 20].

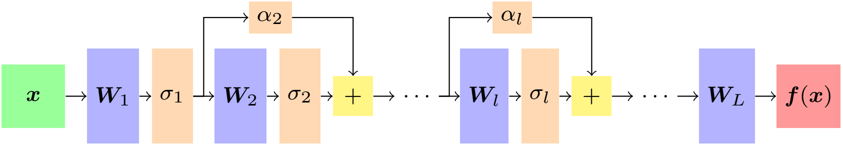
Neural Architecture Search (NAS) [95]

- An architecture has a significant impact on the performance and the inductive bias of the model [39, 77, 78, 20].
- Manually designed architectures require domain expertise and might not be optimal.

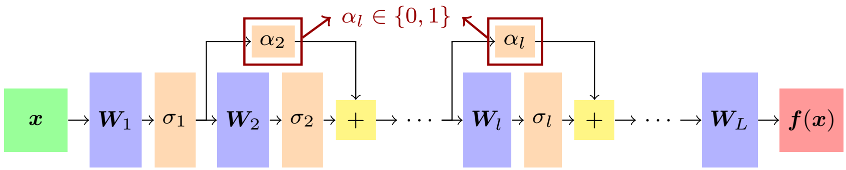
Neural Architecture Search (NAS) [95]

- An architecture has a significant impact on the performance and the inductive bias of the model [39, 77, 78, 20].
- Manually designed architectures require domain expertise and might not be optimal.
- Instead, we can define a search procedure to select the architecture.

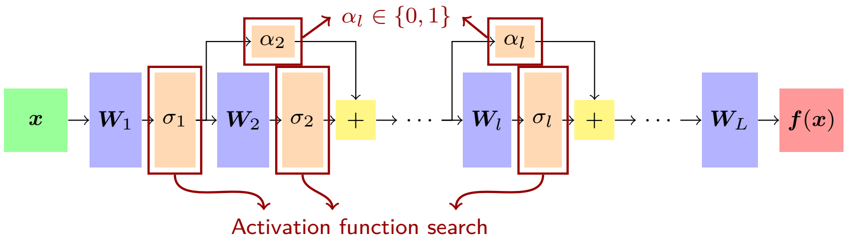
Towards a principled Neural Architecture Search (NAS) [92]



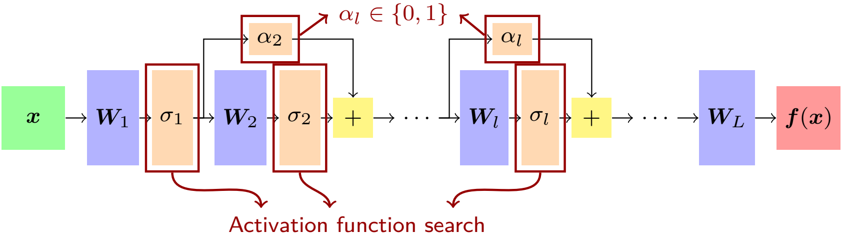
Towards a principled Neural Architecture Search (NAS) [92]



Towards a principled Neural Architecture Search (NAS) [92]



Towards a principled Neural Architecture Search (NAS) [92]



- Generalization error (of the unified architecture) with respect to the minimum eigenvalue $\lambda_{\min}$ of NTK:

$$\text{generalization error} \lesssim \mathcal{O}\left(\frac{1}{\sqrt{\lambda_{\min}}}\right), whp.$$

Towards a principled Neural Architecture Search (NAS) [92]

○ Beyond the depth L, the minimum eigenvalue is also affected by the constants $\beta_1, \beta_2, \beta_3$ that are only determined by the activation function. Specifically, $f_{\text{lower}}(\beta_3) \leq \lambda_{\min} \leq f_{\text{upper}}(\beta_1, \beta_2)$.

σ	Trend of lower bound	ReLU	LeakyReLU	Sigmoid	Tanh	Swish
$\beta_3(\sigma)$	$\nearrow$	1	> 1	$\leq 1/16$	≤ 1	$1/2$

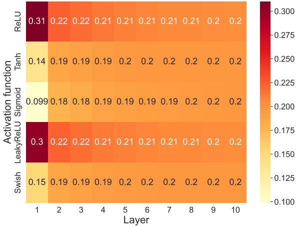


Figure: Probability of selecting activation per layer numerically.

Towards a principled Neural Architecture Search (NAS) [92]

◦ Beyond the depth L, the minimum eigenvalue is also affected by the constants $\beta_1, \beta_2, \beta_3$ that are only determined by the activation function. Specifically, $f_{\text{lower}}(\beta_3) \leq \lambda_{\min} \leq f_{\text{upper}}(\beta_1, \beta_2)$.

σ	Trend of lower bound	ReLU	LeakyReLU	Sigmoid	Tanh	Swish
$\beta_3(\sigma)$	$\nearrow$	1	> 1	$\leq 1/16$	≤ 1	$1/2$

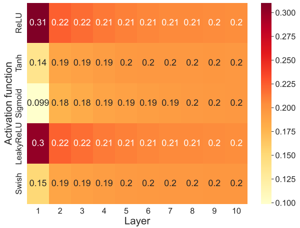


Figure: Probability of selecting activation per layer numerically.

Overall insights:

- The depth L and the skip connections via α_l affect significantly the bounds.
- The first activation function σ_1 is more important than the rest activation functions.

Train-free Neural Architecture Search (NAS) [92]

Algorithm Eigen-NAS Algorithm

Require: Search space $\mathcal{S}$, training data $\mathcal{D}_{tr} = \{(\boldsymbol{x}_i, y_i)_{i=1}^{N}\}$, validation data $\mathcal{D}_{val} = \{(\boldsymbol{x}_j, y_j)_{j=1}^{N_v}\}$.

Initialize max_iteration $= M$

Initialize candidate set $\mathcal{C} = []$

for search_iteration in $1, 2, \ldots,$max_iteration **do**

Randomly sample architecture s from search space $\mathcal{S}$.

Compute $Eigen :=$ minimum eigenvalue of NTK.

$\mathcal{C}$.append$(s, Eigen)$

update $\mathcal{C}$ to kept top-K best architectures

end for

$s^\star = best_s(\mathcal{C}, \mathcal{D}_{tr}, \mathcal{D}_{val})$ # Choose the best architecture based on validation error after training 20 epochs.

Output $s^\star$

Extrapolation

Let us assume training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{X}|}$ and any direction $\boldsymbol{v} \in \mathbb{R}^d$ that satisfies $\|\boldsymbol{v}\|_2 = \max\{\|\boldsymbol{x}_i\|^2\}$. Let $\boldsymbol{x} = (t+h)\boldsymbol{v}$ with $t > 1$ and $h > 0$ be the extrapolation data points.

Theorem (N-layer MLP [84])

The output $f(\boldsymbol{x})$, when f is a trained N-layer ReLU-NN, follows a linear function with respect to h.

Extrapolation

Let us assume training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{X}|}$ and any direction $\boldsymbol{v} \in \mathbb{R}^d$ that satisfies $\|\boldsymbol{v}\|_2 = \max\{\|\boldsymbol{x}_i\|^2\}$. Let $\boldsymbol{x} = (t+h)\boldsymbol{v}$ with $t > 1$ and $h > 0$ be the extrapolation data points.

Theorem (N-layer MLP [84])

The output $f(\boldsymbol{x})$, when f is a trained N-layer ReLU-NN, follows a linear function with respect to h.

Theorem (N-degree PN [82])

The output $f(\boldsymbol{x})$, when f is a trained N-degree PN, follows a γ-degree ($\gamma \leq N$) function with respect to h.

Extrapolation

Let us assume training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{X}|}$ and any direction $\boldsymbol{v} \in \mathbb{R}^d$ that satisfies $\|\boldsymbol{v}\|_2 = \max\{\|\boldsymbol{x}_i\|^2\}$. Let $\boldsymbol{x} = (t+h)\boldsymbol{v}$ with $t > 1$ and $h > 0$ be the extrapolation data points.

Theorem (N-layer MLP [84])

The output $f(\boldsymbol{x})$, when f is a trained N-layer ReLU-NN, follows a linear function with respect to h.

Theorem (N-degree PN [82])

The output $f(\boldsymbol{x})$, when f is a trained N-degree PN, follows a γ-degree ($\gamma \leq N$) function with respect to h.

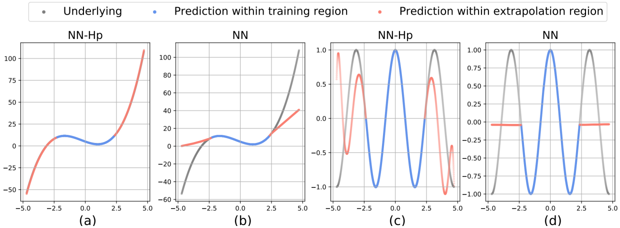


Figure: (a) and (b): fitting $f_\rho(x) = x^3 + x^2 - 10x + 5$. (c) and (d): fitting $f_\rho(x) = \cos(2x)$.

Extrapolation – experimental validation

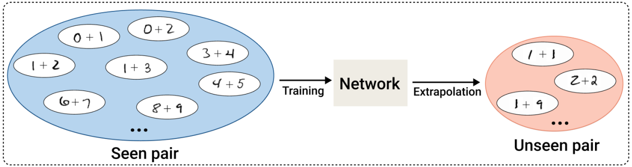
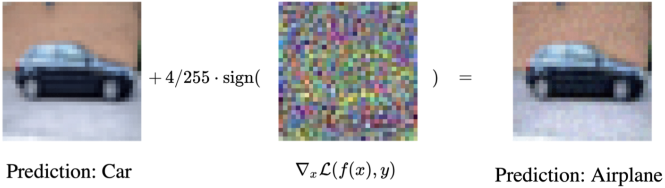


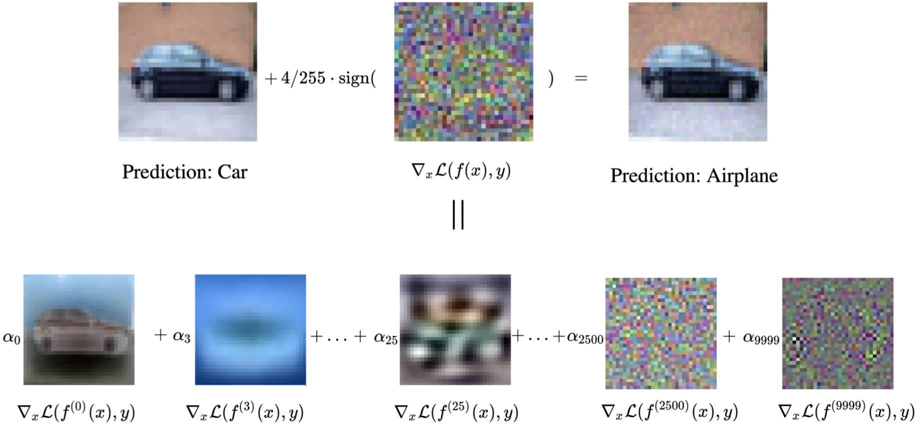
Table: Experimental evaluation of visual addition.

Method	Accuracy (Rounding)
NN (Dense)	0.436 ± 0.065
PNN (Dense)	$\mathbf{0.554 \pm 0.011}$
NN (Conv)	0.617 ± 0.103
PNN (Conv)	$\mathbf{0.825 \pm 0.109}$

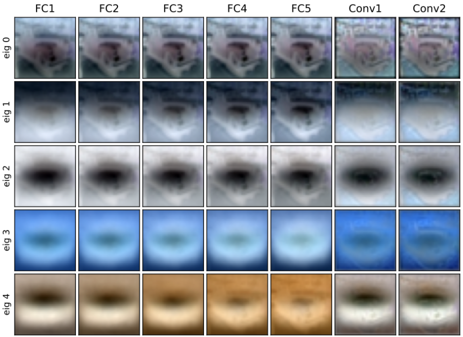
Visualizing the components of adversarial perturbations [75]



Visualizing the components of adversarial perturbations [75]



Visualizing the features [75]



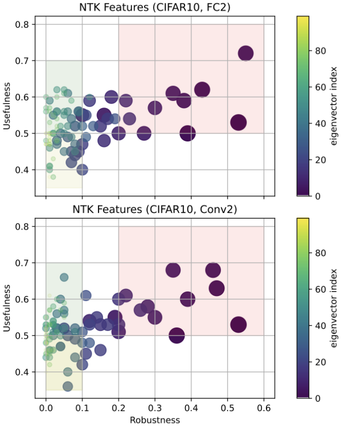
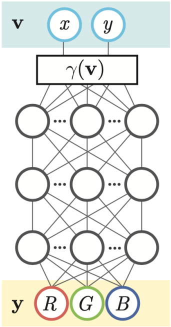


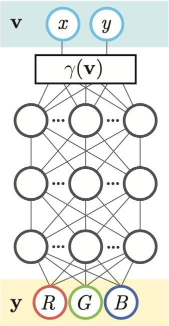
Figure: (Left) Top 5 features for 7 different kernel architectures for a car image. (Right) Features according to their robustness (x-axis) and usefulness (y-axis).

The role of positional encodings in implicit representations [74]



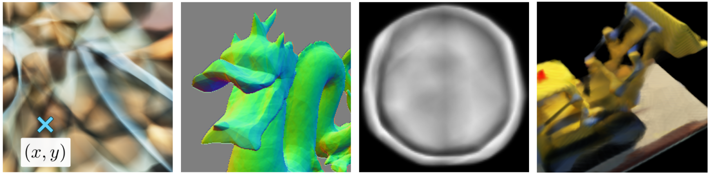
(a) Coordinate-based MLP

The role of positional encodings in implicit representations [74]

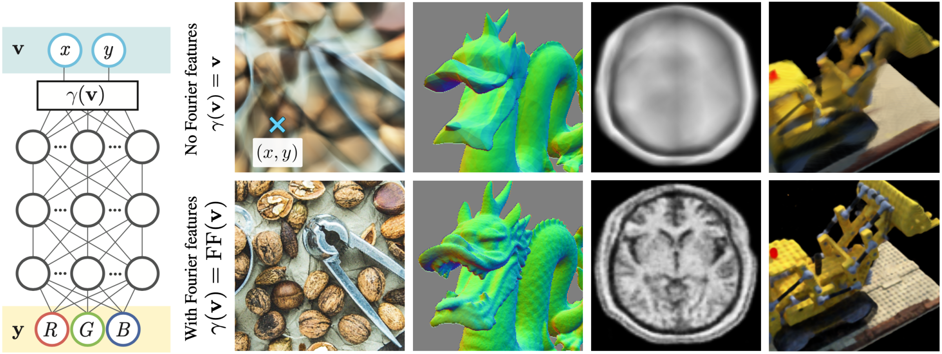


(a) Coordinate-based MLP

No Fourier features
$\gamma(\mathbf{v}) = \mathbf{v}$



The role of positional encodings in implicit representations [74]



(a) Coordinate-based MLP

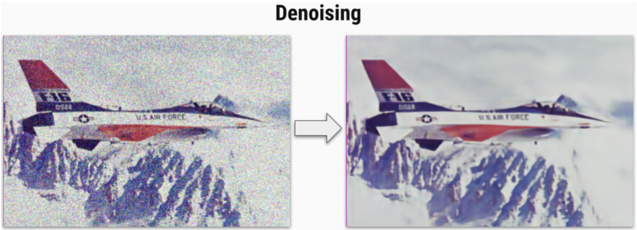
(b) Image regression $(x,y) \rightarrow$ RGB

(c) 3D shape regression $(x,y,z) \rightarrow$ occupancy

(d) MRI reconstruction $(x,y,z) \rightarrow$ density

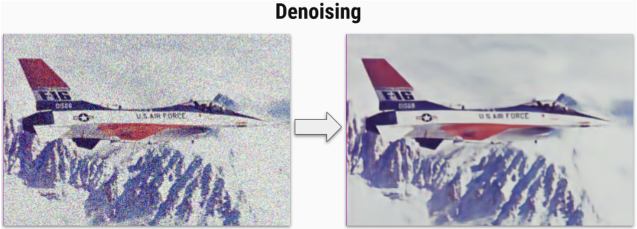
(e) Inverse rendering $(x,y,z) \rightarrow$ RGB, density

Denoising with Deep Image Prior (DIP) [76]



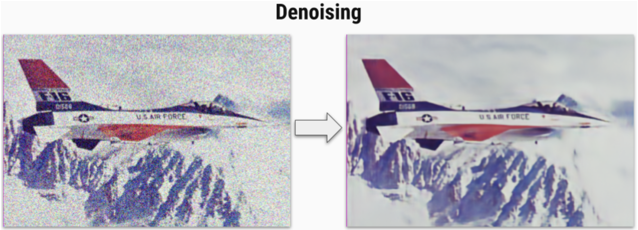
- Inverse problems have immense applications in imaging tasks.
- Deep Image Prior (DIP) does not require training with massive data. The noisy (input) image is sufficient.

Denoising with Deep Image Prior (DIP) [76]



- Inverse problems have immense applications in imaging tasks.
- Deep Image Prior (DIP) does not require training with massive data. The noisy (input) image is sufficient.
- *How* does DIP work?

Denoising with Deep Image Prior (DIP) [76]



- Inverse problems have immense applications in imaging tasks.
- Deep Image Prior (DIP) does not require training with massive data. The noisy (input) image is sufficient.
- *How* does DIP work?
- *Why* does DIP work?

The Neural Tangent Link Between DIP and Non-Local Filters [73]

- From DIP to NTK [second part of the tutorial].

The Neural Tangent Link Between DIP and Non-Local Filters [73]

- From DIP to NTK [second part of the tutorial].
- Link between first-order approximation of DIP network and non-local filters.

The Neural Tangent Link Between DIP and Non-Local Filters [73]

- From DIP to NTK [second part of the tutorial].
- Link between first-order approximation of DIP network and non-local filters.
- Compute the NTK Gram matrix instead of learning. Use directly that version for denoising.

The Neural Tangent Link Between DIP and Non-Local Filters [73]

- From DIP to NTK [second part of the tutorial].
- Link between first-order approximation of DIP network and non-local filters.
- Compute the NTK Gram matrix instead of learning. Use directly that version for denoising.
- Use insights from the derivation to explain why the optimizer is crucial and why DIP works primarily with Adam and not SGD.

Beyond linear layers

- Analysis typically assumes fully-connected layers, which are far from practice, e.g., in convolutional networks.

Beyond linear layers

- Analysis typically assumes fully-connected layers, which are far from practice, e.g., in convolutional networks.
- Analysis of contemporary components, e.g. layer normalization, remains elusive.

Beyond linear layers

- Analysis typically assumes fully-connected layers, which are far from practice, e.g., in convolutional networks.
- Analysis of contemporary components, e.g. layer normalization, remains elusive.
- The assumptions on the components are often restrictive and do not reflect their utilization in actual applications.

Analysis of Transformers

- No complete analysis of the self-attention block **with** the softmax [40].

Analysis of Transformers

- No complete analysis of the self-attention block **with** the softmax [40].
- Simplifying assumptions in the transformer block, e.g., width of the layers.

Analysis of Transformers

- No complete analysis of the self-attention block **with** the softmax [40].
- Simplifying assumptions in the transformer block, e.g., width of the layers.
- Little insight into what is special about transformers when compared to other architectures.

Learning regime

- There is an incomplete list of theoretical insights for guiding practical implementations.

Learning regime

- There is an incomplete list of theoretical insights for guiding practical implementations.
- Most of the properties, e.g., inductive bias, is derived on the lazy regime, which might not reflect what happens in practice.

Learning regime

- There is an incomplete list of theoretical insights for guiding practical implementations.
- Most of the properties, e.g., inductive bias, is derived on the lazy regime, which might not reflect what happens in practice.
- We currently lack any evidence on whether lazy regime is realistic and under which cases.

Learning regime

- There is an incomplete list of theoretical insights for guiding practical implementations.
- Most of the properties, e.g., inductive bias, is derived on the lazy regime, which might not reflect what happens in practice.
- We currently lack any evidence on whether lazy regime is realistic and under which cases.
- The theoretical analysis on the non-lazy regime seems to be much more diverse, which creates a requirement for a more thorough taxonomy.

Practical considerations

- How can the theoretical insights extend beyond classification to problems relevant to vision? For instance, (conditional) generation, dense reconstruction or tracking?

Practical considerations

- How can the theoretical insights extend beyond classification to problems relevant to vision? For instance, (conditional) generation, dense reconstruction or tracking?
- Can tight generalization bounds be used for guiding practical implementations [91]?

Practical considerations

- How can the theoretical insights extend beyond classification to problems relevant to vision? For instance, (conditional) generation, dense reconstruction or tracking?
- Can tight generalization bounds be used for guiding practical implementations [91]?
- How can we relax the existing theoretical tools to reflect practical implementations (e.g., having a finite width)?

Thanks for your attention!

Q & A

References I

[1] Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard Baraniuk.
The recurrent neural tangent kernel.
In *International Conference on Learning Representations*, 2021.
(Cited on page 78.)

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song.
A convergence theory for deep learning via over-parameterization.
In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.
(Cited on pages 68, 79, 80, 81, and 82.)

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou.
Wasserstein generative adversarial networks.
In *International conference on machine learning*, pages 214–223. PMLR, 2017.
(Cited on page 26.)

[4] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang.
On exact computation with an infinitely wide neural net.
Advances in Neural Information Processing Systems, 32, 2019.
(Cited on page 78.)

References II

[5] Francis Bach.
Breaking the curse of dimensionality with convex neural networks.
Journal of Machine Learning Research, 18(1):629–681, 2017.
(Cited on pages 83 and 84.)

[6] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler.
Benign overfitting in linear regression.
Proceedings of the National Academy of Sciences, 117(48):30063–30070, 2020.
(Cited on pages 62, 63, 64, 65, and 66.)

[7] Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin.
Deep learning: a statistical viewpoint.
Acta numerica, 30:87–201, 2021.
(Cited on page 70.)

[8] Michel Benaïm.
Dynamics of stochastic approximation algorithms.
In Jacques Azéma, Michel Émery, Michel Ledoux, and Marc Yor, editors, *Séminaire de Probabilités XXXIII*, volume 1709 of *Lecture Notes in Mathematics*, pages 1–68. Springer Berlin Heidelberg, 1999.
(Cited on pages 12 and 13.)

References III

[9] Michel Benaïm and Morris W. Hirsch.
Asymptotic pseudotrajectories and chain recurrent flows, with applications.
Journal of Dynamics and Differential Equations, 8(1):141–176, 1996.
(Cited on pages 32 and 33.)

[10] Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher.
Adversarially robust optimization with gaussian processes.
In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 5765–5775, 2018.
(Cited on page 10.)

[11] Simone Bombari, Mohammad Hossein Amani, and Marco Mondelli.
Memorization and optimization in deep neural networks with minimum over-parameterization.
In *Advances in Neural Information Processing Systems*, 2022.
(Cited on page 82.)

[12] Sebastien Bubeck and Mark Sellke.
A universal law of robustness via isoperimetry.
In *Advances in Neural Information Processing Systems*, 2021.
(Cited on pages 88, 89, 90, 91, and 92.)

References IV

[13] Yuan Cao and Quanquan Gu.
Generalization bounds of stochastic gradient descent for wide and deep neural networks.
In *Advances in neural information processing systems*, 2019.
(Cited on pages 79, 80, and 81.)

[14] Michael Celentano, Theodor Misiakiewicz, and Andrea Montanari.
Minimum complexity interpolation in random features models.
arXiv preprint arXiv:2103.15996, 2021.
(Cited on pages 83 and 84.)

[15] Volkan Cevher and Bang Cong Vu.
A reflected forward-backward splitting method for monotone inclusions involving lipschitzian operators.
Set-Valued and Variational Analysis, pages 1–12, 2020.
(Cited on page 30.)

[16] Wuyang Chen, Xinyu Gong, and Zhangyang Wang.
Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective.
In *International Conference on Learning Representations*, 2021.
(Cited on pages 85, 86, and 87.)

References V

[17] Yilan Chen, Wei Huang, Lam Nguyen, and Tsui-Wei Weng.
On the equivalence between neural network and support vector machine.
Advances in Neural Information Processing Systems, 34:23478–23490, 2021.
(Cited on pages 76 and 77.)

[18] Lenaic Chizat, Edouard Oyallon, and Francis Bach.
On lazy training in differentiable programming.
Advances in Neural Information Processing Systems, 32, 2019.
(Cited on page 70.)

[19] Lenaic Chizat, Edouard Oyallon, and Francis Bach.
On lazy training in differentiable programming.
arXiv preprint arXiv:1812.07956, 2019.
(Cited on pages 71, 74, and 75.)

[20] Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou.
P-nets: Deep polynomial neural networks.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7325–7335, 2020.
(Cited on pages 103, 104, and 105.)

References VI

[21] J. Danskin.
The theory of max-min, with applications.
SIAM Journal on Applied Mathematics, 14(4):641–664, 1966.
(Cited on page 15.)

[22] Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis.
The complexity of constrained min-max optimization.
arXiv preprint arXiv:2009.09623, 2020.
(Cited on page 29.)

[23] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai.
Gradient descent finds global minima of deep neural networks.
In *International Conference on Machine Learning*, pages 1675–1685, 2019.
(Cited on page 82.)

[24] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu.
Graph neural tangent kernel: Fusing graph neural networks with graph kernels.
In *Advances in neural information processing systems*, 2019.
(Cited on page 78.)

References VII

[25] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh.
Gradient descent provably optimizes over-parameterized neural networks.
arXiv preprint arXiv:1810.02054, 2018.
(Cited on pages 12 and 13.)

[26] Richard Mansfield Dudley.
The speed of mean glivenko-cantelli convergence.
The Annals of Mathematical Statistics, 40(1):40–50, 1969.
(Cited on page 24.)

[27] Weinan E, Chao Ma, and Lei Wu.
A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics.
Science China Mathematics, 2020.
(Cited on page 68.)

[28] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song.
Robust physical-world attacks on deep learning visual classification.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.
(Cited on pages 60 and 61.)

References VIII

[29] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur.
Sharpness-aware minimization for efficiently improving generalization.
In *International Conference on Learning Representations*, 2021.
(Cited on page 10.)

[30] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan.
Escaping from saddle points—online stochastic gradient for tensor decomposition.
In *Conference on Learning Theory*, pages 797–842, 2015.
(Cited on pages 12 and 13.)

[31] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan.
Escaping from saddle points — Online stochastic gradient for tensor decomposition.
In *COLT '15: Proceedings of the 28th Annual Conference on Learning Theory*, 2015.
(Cited on pages 12 and 13.)

[32] Saeed Ghadimi and Guanghui Lan.
Stochastic first- and zeroth-order methods for nonconvex stochastic programming.
SIAM Journal on Optimization, 23(4):2341–2368, 2013.
(Cited on pages 12 and 13.)

References IX

[33] Xavier Glorot and Yoshua Bengio.
Understanding the difficulty of training deep feedforward neural networks.
In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
(Cited on page 68.)

[34] Guang-Bin Huang and H. A. Babri.
Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions.
IEEE Transactions on Neural Networks, 9(1):224–229, 1998.
(Cited on pages 12 and 13.)

[35] Osman Güler.
On the convergence of the proximal point algorithm for convex minimization.
SIAM J. Control Opt., 29(2):403–419, March 1991.
(Cited on page 30.)

[36] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville.
Improved training of wasserstein gans.
In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.
(Cited on page 26.)

References X

[37] Moritz Hardt and Tengyu Ma.
Identity matters in deep learning.
arXiv preprint arXiv:1611.04231, 2016.
(Cited on pages 12 and 13.)

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.
In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.
(Cited on page 68.)

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Deep residual learning for image recognition.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
(Cited on pages 103, 104, and 105.)

[40] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak.
Infinite attention: NNGP and NTK for deep attention networks.
In *International Conference on Machine Learning*, 2020.
(Cited on pages 133, 134, and 135.)

References XI

[41] Ya-Ping Hsieh, Panayotis Mertikopoulos, and Volkan Cevher.
The limits of min-max optimization algorithms: Convergence to spurious non-critical sets.
arXiv preprint arXiv:2006.09065, 2020.
(Cited on pages 31, 32, 33, and 34.)

[42] Guang-Bin Huang.
Learning capability and storage capacity of two-hidden-layer feedforward networks.
IEEE Transactions on Neural Networks, 14(2):274–281, 2003.
(Cited on pages 12 and 13.)

[43] Hanxun Huang, Yisen Wang, Sarah Monazam Erfani, Quanquan Gu, James Bailey, and Xingjun Ma.
Exploring architectural ingredients of adversarially robust deep neural networks.
In *Advances in Neural Information Processing Systems*, 2021.
(Cited on pages 88, 89, 90, 91, 92, and 96.)

[44] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári.
Learning with a strong adversary.
arXiv preprint arXiv:1511.03034, 2015.
(Cited on page 10.)

References XII

[45] S. . Huang and Y. . Huang.
Bounds on the number of hidden neurons in multilayer perceptrons.
IEEE Transactions on Neural Networks, 2(1):47–55, 1991.
(Cited on pages 12 and 13.)

[46] Arthur Jacot, Franck Gabriel, and Clément Hongler.
Neural tangent kernel: Convergence and generalization in neural networks.
In *Advances in neural information processing systems*, pages 8571–8580, 2018.
(Cited on pages 71, 74, 75, 76, and 77.)

[47] Kenji Kawaguchi and Jiaoyang Huang.
Gradient descent finds global minima for generalizable deep neural networks of practical sizes.
In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 92–99. IEEE, 2019.
(Cited on pages 12, 13, and 82.)

[48] Galina M Korpelevich.
The extragradient method for finding saddle points and other problems.
Matecon, 12:747–756, 1976.
(Cited on page 30.)

References XIII

[49] Sanjukta Krishnagopal and Luana Ruiz.
Graph neural tangent kernel: Convergence on large graphs.
arXiv preprint arXiv:2301.10808, 2023.
(Cited on page 78.)

[50] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller.
Efficient backprop.
In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.
(Cited on page 68.)

[51] Jason D. Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I. Jordan, and Benjamin Recht.
First-order methods almost always avoid strict saddle points.
Mathematical Programming, 176(1):311–337, February 2019.
(Cited on pages 12 and 13.)

[52] Tao Li, Lei Tan, Zhehao Huang, Qinghua Tao, Yipeng Liu, and Xiaolin Huang.
Low dimensional trajectory hypothesis is true: Dnns can be trained in tiny subspaces.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022.
(Cited on pages 85, 86, and 87.)

References XIV

[53] Yuanzhi Li and Yingyu Liang.
Learning overparameterized neural networks via stochastic gradient descent on structured data.
In *Advances in Neural Information Processing Systems*, pages 8157–8166, 2018.
(Cited on pages 12, 13, and 82.)

[54] Tianlin Liu and Friedemann Zenke.
Finding trainable sparse networks through neural tangent transfer.
In *International Conference on Machine Learning*, pages 6336–6347. PMLR, 2020.
(Cited on pages 85, 86, and 87.)

[55] Lennart Ljung.
Analysis of recursive stochastic algorithms.
IEEE Transactions on Automatic Control, 22(4):551–575, August 1977.
(Cited on pages 12 and 13.)

[56] Tao Luo, Zhi-Qin John Xu, Zheng Ma, and Yaoyu Zhang.
Phase diagram for two-layer relu neural networks at infinite-width limit.
Journal of Machine Learning Research, 2021.
(Cited on pages 68 and 69.)

References XV

[57] Tao Luo, Zhi-Qin John Xu, Zheng Ma, and Yaoyu Zhang.
Phase diagram for two-layer relu neural networks at infinite-width limit.
Journal of Machine Learning Research, 22(71):1–47, 2021.
(Cited on pages 71, 74, and 75.)

[58] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu.
Towards deep learning models resistant to adversarial attacks.
In *ICLR '18: Proceedings of the 2018 International Conference on Learning Representations*, 2018.
(Cited on pages 17, 18, and 21.)

[59] Yura Malitsky and Matthew K Tam.
A forward-backward splitting method for monotone inclusions without cocoercivity.
SIAM Journal on Optimization, 30(2):1451–1472, 2020.
(Cited on page 30.)

[60] Song Mei, Andrea Montanari, and Phan-Minh Nguyen.
A mean field view of the landscape of two-layers neural networks.
Proceedings of the National Academy of Sciences (PNAS), 2018.
(Cited on page 68.)

References XVI

[61] Panayotis Mertikopoulos, Nadav Hallak, Ali Kavis, and Volkan Cevher.
On the almost sure convergence of stochastic gradient descent in non-convex problems.
pages 1117–1128, 2020.
(Cited on pages 12 and 13.)

[62] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida.
Spectral normalization for generative adversarial networks.
arXiv preprint arXiv:1802.05957, 2018.
(Cited on page 26.)

[63] Fangzhou Mu, Yingyu Liang, and Yin Li.
Gradients as features for deep representation learning.
In *International Conference on Learning Representations*, 2020.
(Cited on pages 82, 85, 86, and 87.)

[64] Quynh Nguyen and Matthias Hein.
Optimization landscape and expressivity of deep cnns.
In *International conference on machine learning*, pages 3730–3739. PMLR, 2018.
(Cited on pages 12 and 13.)

References XVII

[65] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee.
Dataset distillation with infinitely wide convolutional networks.
Advances in Neural Information Processing Systems, 34:5186–5198, 2021.
(Cited on pages 85, 86, and 87.)

[66] Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard.
What can linearized neural networks actually say about generalization?
In *Advances in Neural Information Processing Systems*, pages 8998–9010, 2021.
(Cited on pages 85, 86, and 87.)

[67] Thomas Pethick, Grigorios G Chrysos, and Volkan Cevher.
Revisiting adversarial training for the worst-performing class.
Transactions on Machine Learning Research, 2023.
(Cited on pages 10 and 36.)

[68] Thomas Pethick, Olivier Fercoq, Puya Latafat, Panagiotis Patrinos, and Volkan Cevher.
Solving stochastic weak minty variational inequalities without increasing batch size.
In *The Eleventh International Conference on Learning Representations*, 2023.
(Cited on page 35.)

References XVIII

[69] R. Tyrrell Rockafellar.
Convex Analysis.
Princeton Univ. Press, Princeton, NJ, 1970.
(Cited on page 30.)

[70] Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman.
The convergence rate of neural networks for learned functions of different frequencies.
In *Advances in Neural Information Processing Systems*, 2019.
(Cited on pages 85, 86, and 87.)

[71] Grégory Roth and William H. Sandholm.
Stochastic approximations with constant step size and differential inclusions.
SIAM Journal on Control and Optimization, 51(1):525–555, 2013.
(Cited on pages 12 and 13.)

[72] Chaehwan Song, Ali Ramezani-Kebrya, Thomas Pethick, Armin Eftekhari, and Volkan Cevher.
Subquadratic overparameterization for shallow neural networks.
In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
(Cited on pages 12 and 13.)

References XIX

[73] Julián Tachella, Junqi Tang, and Mike Davies.
The neural tangent link between cnn denoisers and non-local filters.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8618–8627, 2021.
(Cited on pages 85, 86, 87, 126, 127, 128, and 129.)

[74] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng.
Fourier features let networks learn high frequency functions in low dimensional domains.
In *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547, 2020.
(Cited on pages 120, 121, and 122.)

[75] Nikolaos Tsilivis and Julia Kempe.
What can the neural tangent kernel tell us about adversarial robustness?
In *Advances in Neural Information Processing Systems*, 2022.
(Cited on pages 85, 86, 87, 117, 118, and 119.)

[76] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky.
Deep image prior.
2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9446–9454, 2018.
(Cited on pages 123, 124, and 125.)

References XX

[77] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
In *Advances in Neural Information Processing Systems*, 2017.
(Cited on pages 103, 104, and 105.)

[78] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He.
Non-local neural networks.
pages 7794–7803, 2018.
(Cited on pages 103, 104, and 105.)

[79] Jonathan Weed, Francis Bach, et al.
Sharp asymptotic and finite-sample rates of convergence of empirical measures in wasserstein distance.
Bernoulli, 25(4A):2620–2648, 2019.
(Cited on page 24.)

[80] Max Welling and Yee W Teh.
Bayesian learning via stochastic gradient langevin dynamics.
In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.
(Cited on pages 12 and 13.)

References XXI

[81] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu.
Do wider neural networks really help adversarial robustness?
In *Advances in Neural Information Processing Systems*, 2021.
(Cited on pages 88, 89, 90, 91, 92, and 96.)

[82] Yongtao Wu, Zhenyu Zhu, Fanghui Liu, Grigorios G Chrysos, and Volkan Cevher.
Extrapolation and spectral bias of neural nets with hadamard product: a polynomial net study.
In *Advances in Neural Information Processing Systems*, 2022.
(Cited on pages 78, 113, 114, and 115.)

[83] Zuxuan Wu, Ser-Nam Lim, Larry S Davis, and Tom Goldstein.
Making an invisibility cloak: Real world adversarial attacks on object detectors.
In *European Conference on Computer Vision*, pages 1–17. Springer, 2020.
(Cited on pages 60 and 61.)

[84] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon Shaolei Du, Ken-Ichi Kawarabayashi, and Stefanie Jegelka.
How neural networks extrapolate: From feedforward to graph neural networks.
In *International Conference on Learning Representations*, 2021.
(Cited on pages 113, 114, and 115.)

References XXII

[85] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao.
Tuning large neural networks via zero-shot hyperparameter transfer.
In *Advances in Neural Information Processing Systems*, pages 17084–17097, 2021.
(Cited on pages 85, 86, and 87.)

[86] Gilad Yehudai and Ohad Shamir.
On the power and limitations of random features for understanding neural networks.
arXiv preprint arXiv:1904.00687, 2019.
(Cited on pages 83 and 84.)

[87] Yaodong Yu, Alexander Wei, Sai Praneeth Karimireddy, Yi Ma, and Michael Jordan.
Tct: Convexifying federated learning using bootstrapped neural tangent kernels.
In *Advances in Neural Information Processing Systems*, pages 30882–30897, 2022.
(Cited on pages 85, 86, and 87.)

[88] Chia-Hung Yuan and Shan-Hung Wu.
Neural tangent generalization attacks.
In *International Conference on Machine Learning*, pages 12230–12240. PMLR, 2021.
(Cited on pages 85, 86, and 87.)

References XXIII

[89] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie.
Small relu networks are powerful memorizers: a tight analysis of memorization capacity.
In *Advances in Neural Information Processing Systems*, pages 15558–15569, 2019.
(Cited on pages 12 and 13.)

[90] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals.
Understanding deep learning requires rethinking generalization.
arXiv preprint arXiv:1611.03530, 2016.
(Cited on page 59.)

[91] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals.
Understanding deep learning requires rethinking generalization.
In *International Conference on Learning Representations*, 2017.
(Cited on pages 140, 141, and 142.)

[92] Zhenyu Zhu, Fanghui Liu, Grigorios G Chrysos, and Volkan Cevher.
Generalization properties of nas under activation and skip connection search.
Advances in Neural Information Processing Systems, 35:23551–23565, 2022.
(Cited on pages 85, 86, 87, 106, 107, 108, 109, 110, 111, and 112.)

References XXIV

[93] Zhenyu Zhu, Fanghui Liu, Grigorios G Chrysos, and Volkan Cevher.
Robustness in deep learning: The good (width), the bad (depth), and the ugly (initialization).
In *Advances in Neural Information Processing Systems*, 2022.
(Cited on pages 90 and 97.)

[94] Martin Zinkevich.
Online convex programming and generalized infinitesimal gradient ascent.
In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.
(Cited on page 30.)

[95] Barret Zoph and Quoc V Le.
Neural architecture search with reinforcement learning.
2017.
(Cited on pages 103, 104, and 105.)

[96] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu.
Gradient descent optimizes over-parameterized deep relu networks.
Machine Learning, 109(3):467–492, 2020.
(Cited on page 82.)

References XXV

[97] Difan Zou and Quanquan Gu.
An improved analysis of training over-parameterized deep neural networks.
In *Advances in Neural Information Processing Systems*, pages 2055–2064, 2019.
(Cited on pages 12, 13, and 82.)