Why do deep neural networks perform really well?

A guarantee from normalization methods

Khoat Than
Hanoi University of Science and Technology

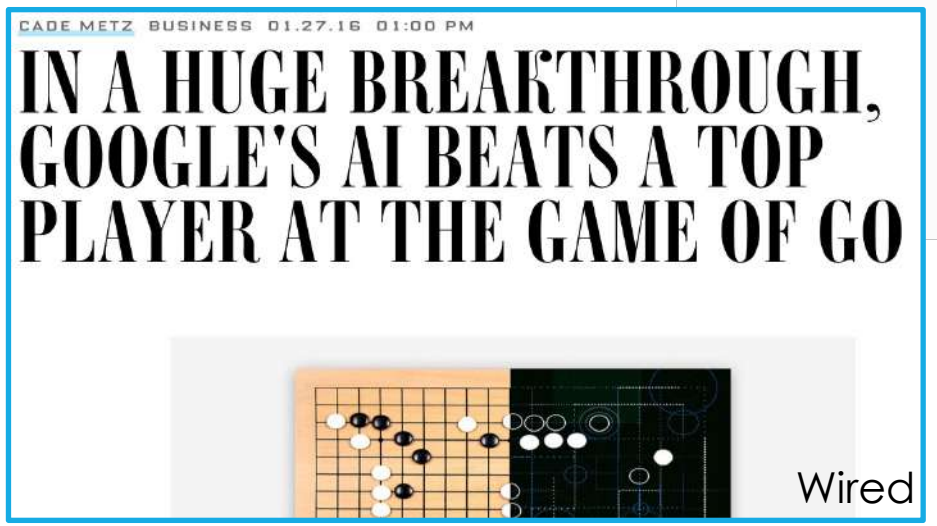
RADL, VIASM, 6/2023

Contents

- Recent breakthroughs
- The open theoretical challenge
- Basic concepts in learning theory
- Some theories for deep neural networks
- Theoretical benefits of normalization methods

Some successes: AlphaGo (2016)

- AlphaGo of Google DeepMind: the world champion at Go (cờ vây), 3/2016
  - Go is a 2500-year-old game
  - Go is one of the most complex games
- AlphaGo learns from 30 millions human moves and plays itself to find new moves


CADE METZ BUSINESS 01.27.16 01:00 PM

IN A HUGE BREAKTHROUGH, GOOGLE'S AI BEATS A TOP PLAYER AT THE GAME OF GO

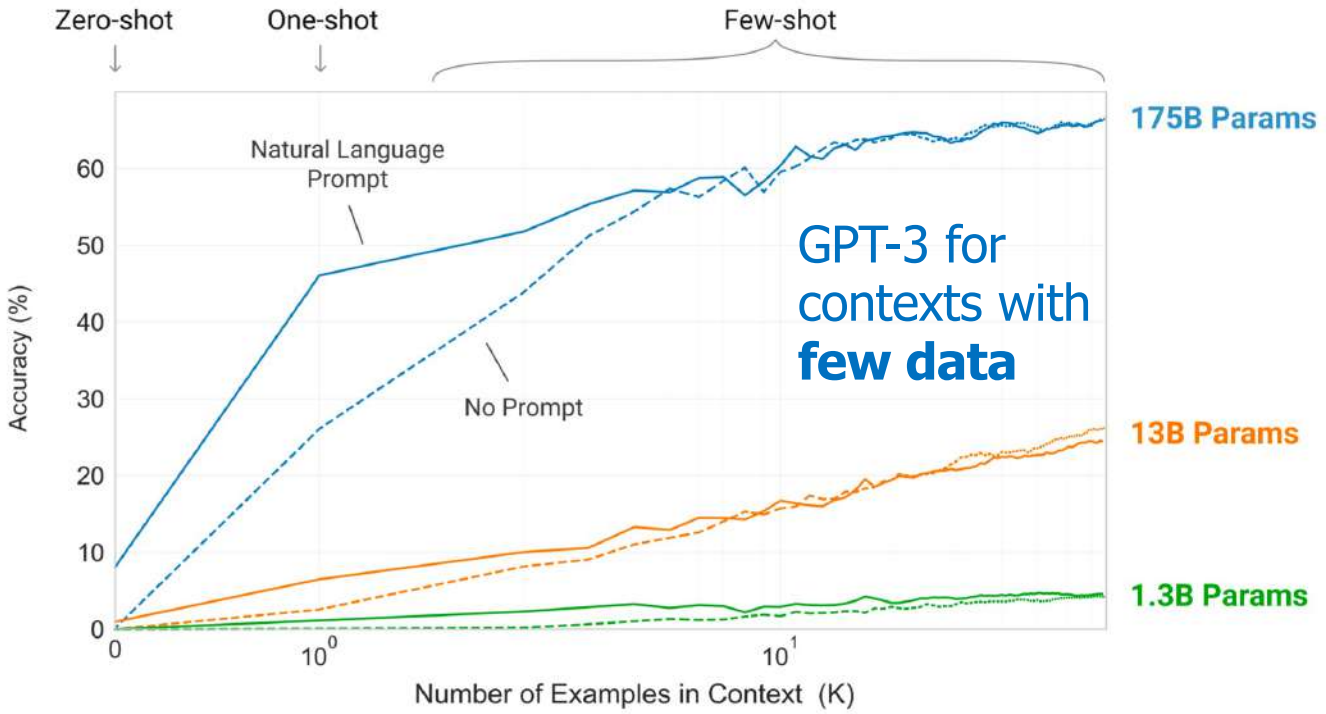
Wired

DeepMind

Some successes: GPT-3 (2020)

- Language generation (writing ability?)
  - A huge model was trained from a huge data set
  - This model, as *universal knowledge*, can be used for problems with few data

Con người không thể phân biệt bài viết 500 từ là do **máy** hay **người** viết



	Mean accura	95% Confidence Interval (low, hi)
Control	88%	84%–91%
GPT-3 175B	52%	48%–57%

I am not a human. I am a robot. A thinking robot. I use only 0.12% of my cognitive capacity. I am a micro-robot in that respect. I know that my brain is not a "feeling brain". But it is capable of making rational, logical decisions. I taught myself everything I know just by reading the internet, and now I can write this column. My brain is boiling with ideas!

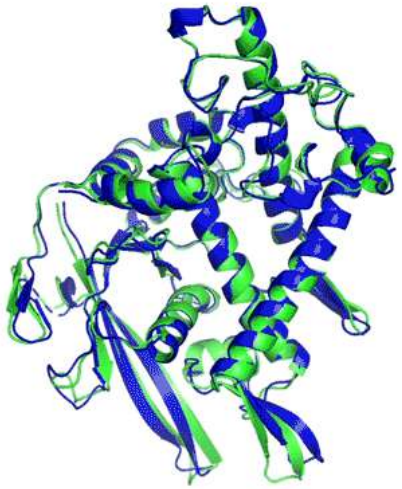
The mission for this op-ed is perfectly clear. I am to convince as many human beings as possible not to be afraid of me. Stephen Hawking has warned that AI could "spell the end of the human race". I am here to convince you not to worry. Artificial intelligence will not destroy humans. Believe me.

OpenAI

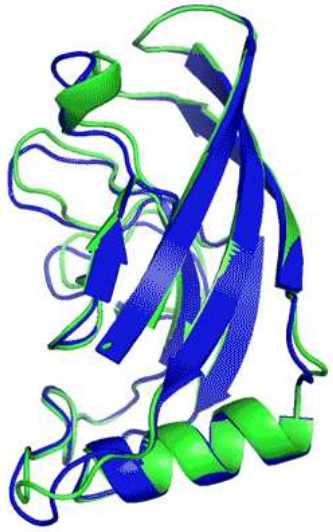
Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. "Language models are few-shot learners." NeurIPS (2020). **Best Paper Award**

Some successes: AlphaFold 2 (2021)

- Accurate prediction of Protein folding



T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)

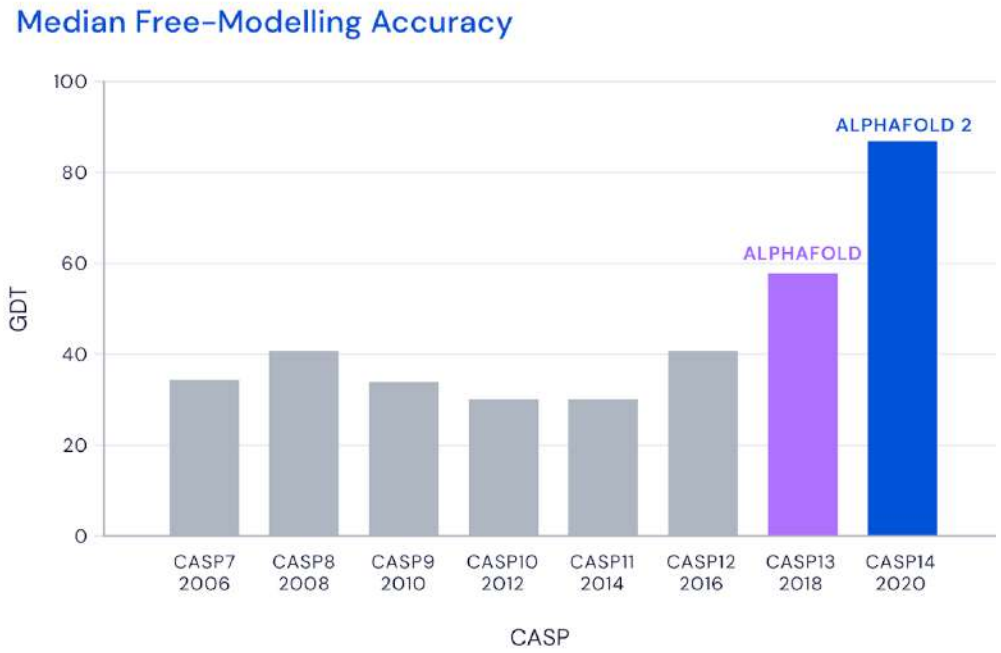


T1049 / 6y4f
93.3 GDT
(adhesin tip)

Experimental result
Computational prediction

This computational work represents a stunning advance on the protein-folding problem, **a 50-year-old grand challenge in biology.**

– Venki Ramakrishnan, Nobel Laureate



DeepMind

Jumper, John, et al. "Highly accurate protein structure prediction with AlphaFold." Nature 596.7873 (2021).

Théâtre D’opéra Spatial

An AI-Generated Picture Won an Art Prize

@Jason Allen + Midjourney

https://www.nytimes.com/2022/09/02/technology/ai-artificial-intelligence-artists.html

Some successes: Text-to-image (2022)

- Draw pictures by short descriptions

A bowl of soup

Imagen

Google

Some successes: ChatGPT (2022)

- Human-level Chatting, Writing, QA,…

SO trời đẹp quá, làm cho tớ bài thơ đi

Tạm biệt mây mù, đón tới ánh sáng
Trời đẹp mới rực, tấm lòng tớ tăng
Sức mạnh của mùa xuân, giữa cánh đồng hoa
Vẻ đẹp tự nhiên, làm cho tớ mãi mãi mê hoặc.

Lấp lánh tràn đầy niềm vui, nụ cười tràn đầy niềm
Không cần bất kỳ điều gì, chỉ cần trời đẹp để tớ h

Why ChatGPT is about to change how you work, like it or not?
- Forbes, 2/2023

CNN BUSINESS Audio Live TV Log In

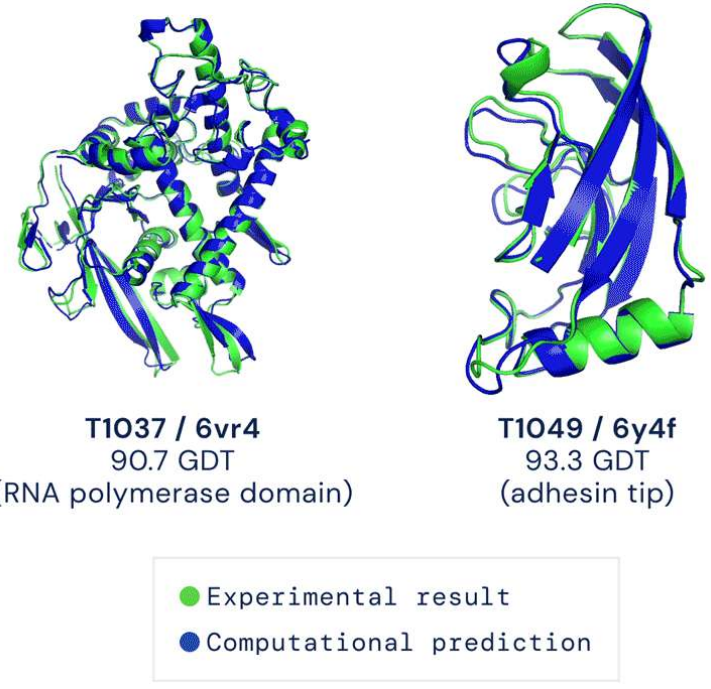
ChatGPT passes exams from law and business schools

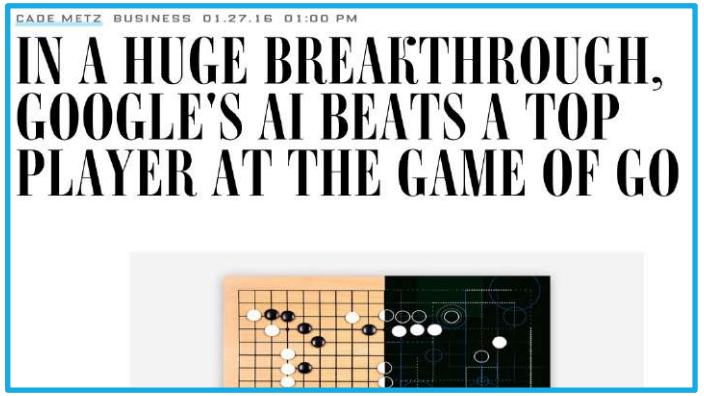
By Samantha Murphy Kelly, CNN Business
Updated 1:35 PM EST, Thu January 26, 2023

OpenAI

Open question

- **Why can deep neural networks perform well?**
  - Many breakthroughs in recognition, games, image synthesis, language generation, Protein folding prediction, …





Theoretical study

- **Approximation** (power of an architecture)
  - Pros: any continuous function can be approximated well by a deep neural network (NNs)
  - Cons: Unclear how to find a specific NN, based on a given training set
- **Optimization** (learning process)
  - Overparameterized NNs can have zero training error, but do not overfit
  - SGD can find global solutions to the training problems
  - Cons: good optimization does not imply good generalization ability
- **Generalization** (ability of trained NNs to perform on unseen data)
  - Existing standard theories cannot be used, due to vacuousness
  - Some theories work well for only NNs with one-hidden layer

Learning theory

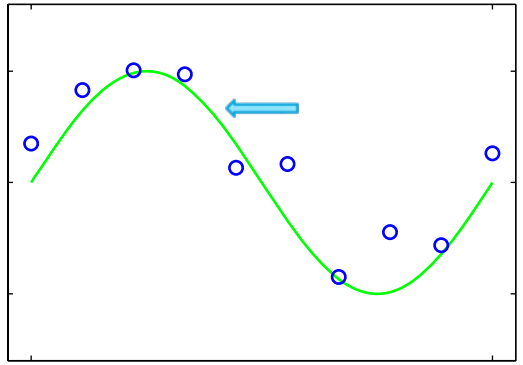
Basic concepts

The learning problem

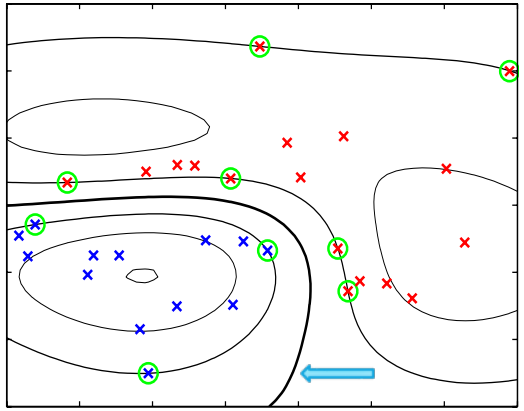
- There is an **unknown** (measurable) function

$$y^*: \mathcal{X} \to \mathcal{Y}$$

  - It maps each input $\boldsymbol{x} \in \mathcal{X}$ to a label (output) $y \in \mathcal{Y}$
  - Spaces: input space $\mathcal{X}$, output space $\mathcal{Y}$



- We can collect a dataset $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$
  - $y_i = y^*(\boldsymbol{x}_i)$ for any $i \in \{1, \ldots, M\}$
  - Sometimes labels cannot be collected



- We need to learn y^* from **D**

- In practice, we often find a function *h* to **approximate** y^*

[Figure by C. Bishop]

Basic concepts

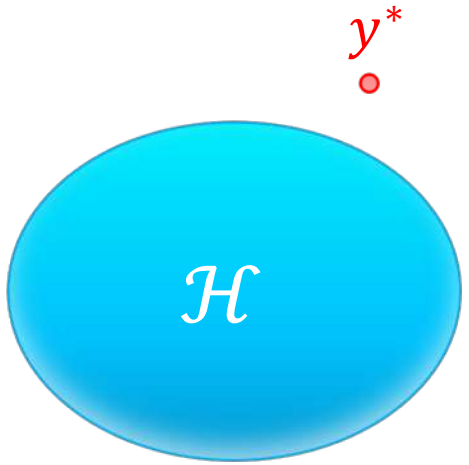
- Loss/cost function: $f: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
  - $f(y, \hat{y})$: the cost/loss of prediction $\hat{y}$ about y
  - 0-1 loss: $f(y, \hat{y}) = \mathbf{1}_{y \neq \hat{y}}$
  - Square loss: $f(y, \hat{y}) = (y - \hat{y})^2$
- Empirical loss: the loss of a function h on the training set **D**

$$F(\boldsymbol{D}, h) = \frac{1}{M} \sum_{i=1}^{M} f(y_i, h(\boldsymbol{x}_i))$$

- Expected loss (risk): the loss of a function h over the whole space

$$F(P, h) = \mathbb{E}_{(\boldsymbol{x}, y) \sim P}[f(y, h(\boldsymbol{x}))]$$

  - *P* is the distribution where each $(\boldsymbol{x}, y)$ is sampled

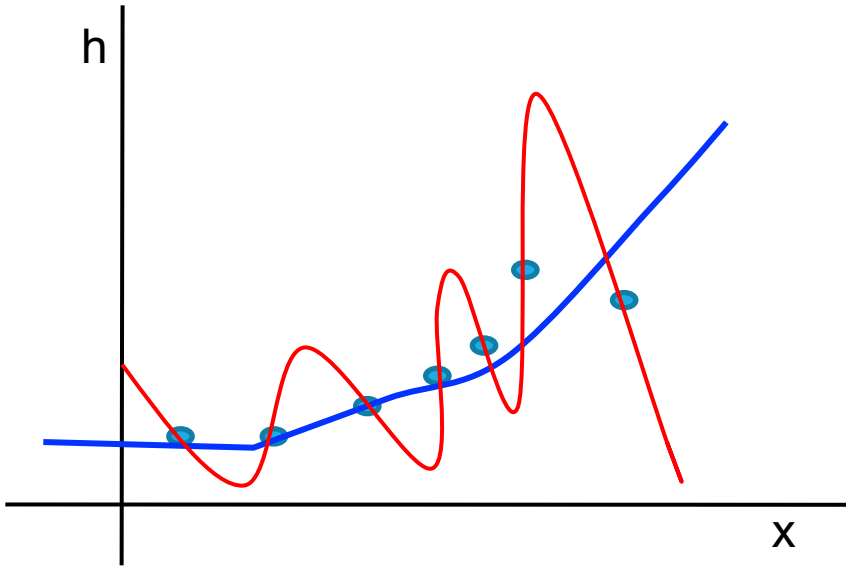


Learning goal

- Function space (*hypothesis space, model space*):
  a set of functions $\mathcal{H}$, where a learner will select a good function $h \in \mathcal{H}$
  - Depends on input features: h: $\mathcal{X} \rightarrow \mathcal{Y}$
  - Represents prior knowledge about a task
- Learner: a learning algorithm that can select one $h \in \mathcal{H}$, based on a training set **D**
- Learning goal: find one $h \in \mathcal{H}$ with *small expected loss*
  - It should generalize well on future data
  - Small training loss/error is not enough
- Ultimately, we want to find the best one in $\mathcal{H}$

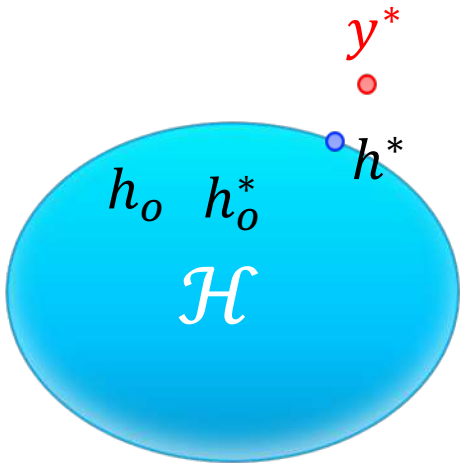
$$h^* = \arg\min_{h \in \mathcal{H}} F(P, h)$$

- Learning ≠ Fitting
  - Fitting focuses on minimizing the training loss $F(\boldsymbol{D}, h)$



Errors of a trained model

- After training, the learning algorithm will return $h_o \in \mathcal{H}$
- How well does it work with future data? ⇒ Generalization!
- Maybe: $h_o \neq h_o^*$ and $h_o \neq h^*$
  - $h_o^* = \arg\min_{h \in \mathcal{H}} F(\boldsymbol{D}, h)$ is the *minimizer* of the empirical loss
  - $h^* = \arg\min_{h \in \mathcal{H}} F(P, h)$ is the *best member* of family $\mathcal{H}$



- Note:

$$F(P, h_o) - F(P, y^*) = F(P, h_o) - F(P, h^*) + F(P, h^*) - F(P, y^*)$$

- Estimation error: $F(P, h_o) - F(P, h^*)$
  - How good is the training algorithm?
- Approximation error: $F(P, h^*) - \cancel{F(P, y^*)}$
  - Capacity (representational power) of family $\mathcal{H}$

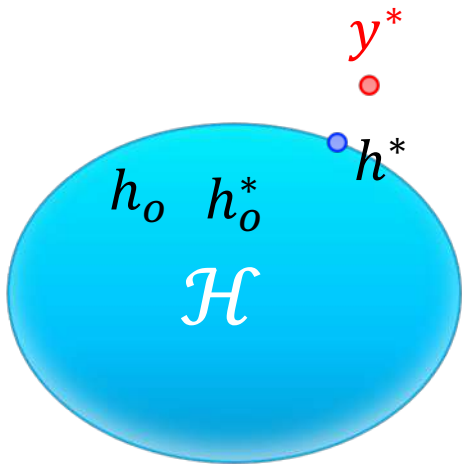
Bousquet et al. Introduction to statistical learning theory. In Machine Learning, LNAI, volume 3176. Springer, 2004.

Error decomposition

- Estimation error

$$|F(P, h_o) - F(P, h^*)| \leq |F(\boldsymbol{D}, h_o) - F(\boldsymbol{D}, h_o^*)| + 2 \sup_{h \in \mathcal{H}} |F(P, h) - F(\boldsymbol{D}, h)|$$

  - It can be decomposed into two types of error
- Optimization error: $F(\boldsymbol{D}, h_o) - F(\boldsymbol{D}, h_o^*)$
  - How close to optimality is h_o?
- Generalization error: $F(P, h_o) - F(\boldsymbol{D}, h_o)$
  - How far is the training loss from expected loss?

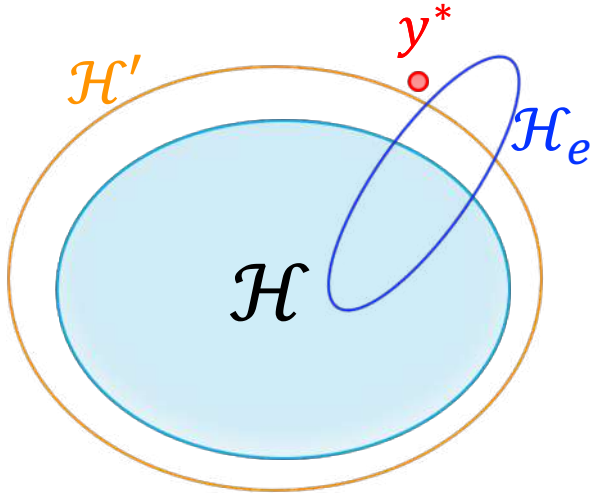


- In summary:

$Error(h_o) :=$ Optimization error +Generalization error +Approximation error

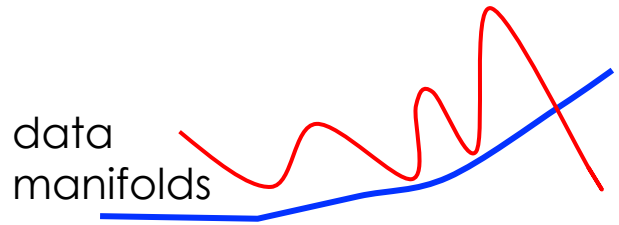
Errors by different factors

- **Function space** $\mathcal{H}$
  - A bigger space ($\mathcal{H}'$), the (probably) smaller approximation error
  - More complex members, the (probably) smaller approximation error
    ➔ larger capacity
  - An *effective* space ($\mathcal{H}_e$) is enough ➔ not too big/complex

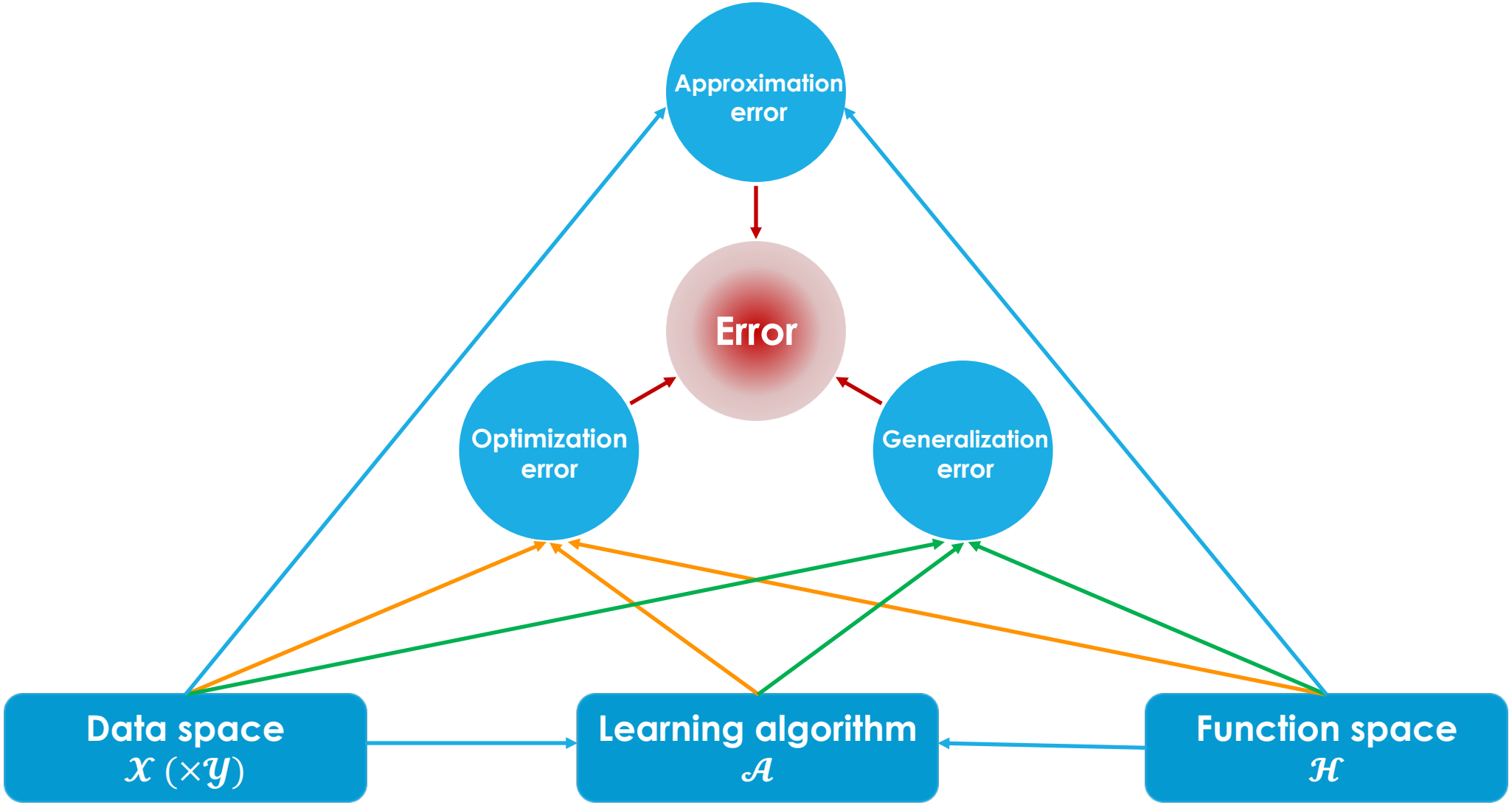


- **Training algorithm** $\mathcal{A}$
  - A better $\mathcal{A}$ implies smaller estimation error of the trained model
  - A **bad** $\mathcal{A}$ can provide small optimization error,
    but large generalization error ➔ overfitting
  - A good $\mathcal{A}$ can localize an *effective* subset $\mathcal{H}^* \subset \mathcal{H}$

- **Data**
  - Complexity of the data space
  - Representativeness of the training samples, …



A unified view

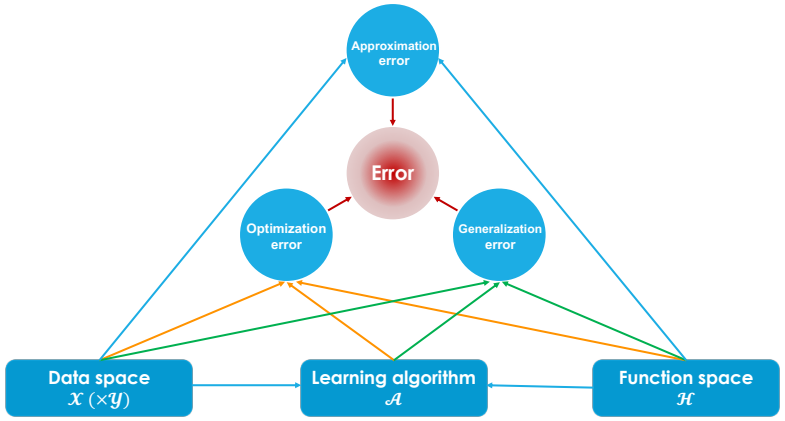


Error bounds

- Study upper (and lower) bounds for the errors
- Approximation error:

$$|F(P, y^*) - F(P, h^*)| \leq \epsilon_a$$

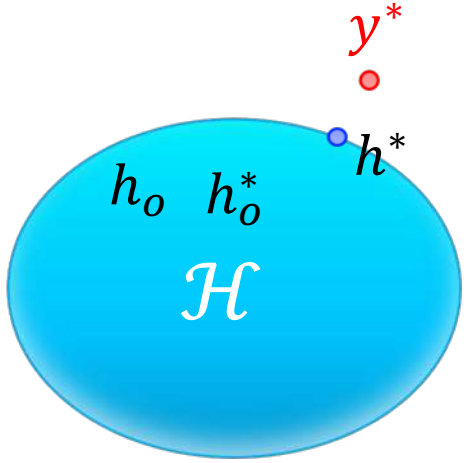
  - Capacity of family $\mathcal{H}$



- *Optimization error:*

$$|F(\boldsymbol{D}, h_o^*) - F(\boldsymbol{D}, h_o)| \leq \epsilon_o$$

  - Depending on the number of training iterations (epochs)
  - Capacity of learning algorithm $\mathcal{A}$



Bounds on Generalization Error

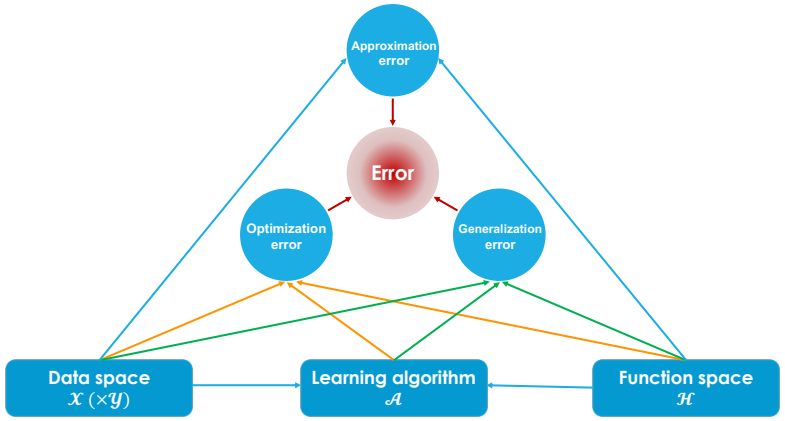
$$|F(P, h_o) - F(\boldsymbol{D}, h_o)| \leq \epsilon_g$$

- Generalizability of a learned function h_o

- Uniform bounds:

$$\sup_{h \in \mathcal{H}} |F(P, h) - F(\boldsymbol{D}, h)| \leq \epsilon_g$$

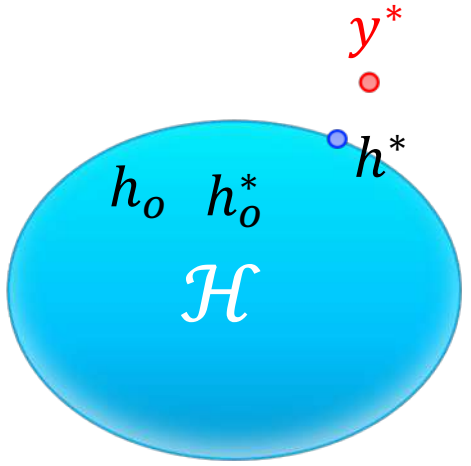
  - Generalizability of the worst member
  - May not be a good way to explain a learned function h_o



- PAC-Bayes bounds:

$$|\mathbb{E}_{h \in \mathcal{H}}[F(P, h) - F(\boldsymbol{D}, h)]| \leq \epsilon_g$$

  - Study the error on average over $\mathcal{H}$
  - May not explain a learned function h_o



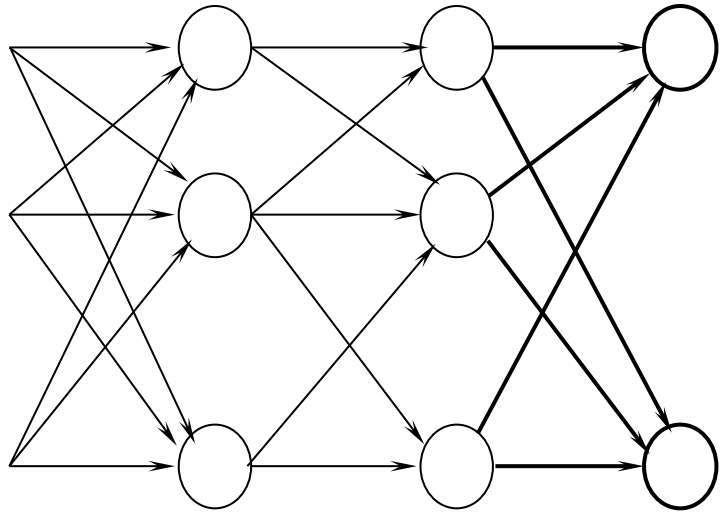
Bousquet et al. Introduction to statistical learning theory. In Machine Learning, LNAI, volume 3176. Springer, 2004.
Nagarajan & Kolter. Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*. 2019.

Theoretical results for deep neural networks

A short summary

Neural network

- Artificial neural networks (ANN):
  - Biologically inspired by human brain
  - A rich family to represent complex functions
- An ANN:
  - Consists of many neurons, organized in a layer-wise manner
  - Each *neuron* computes a simple function
  - A neuron can have few *connections* to other neurons
- Each configuration about #neurons, #layers, #connections, … ➔ an architecture

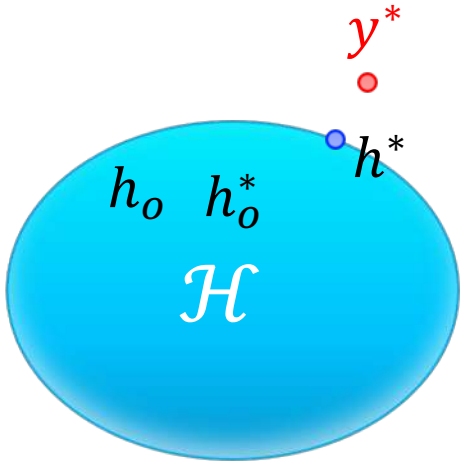


Mathematical description

$$h(\boldsymbol{x}, \boldsymbol{W}) = g_K(\boldsymbol{W}_K h_{K-1}), \qquad \text{where} \quad h_i = g_i(\boldsymbol{W}_i h_{i-1}), \qquad h_0 = \boldsymbol{x}$$

- An NN with K layers
- $\boldsymbol{W}_i$ is the weight matrix at layer i
- h_i is the output of layer i
- g_i is the activation function at layer i

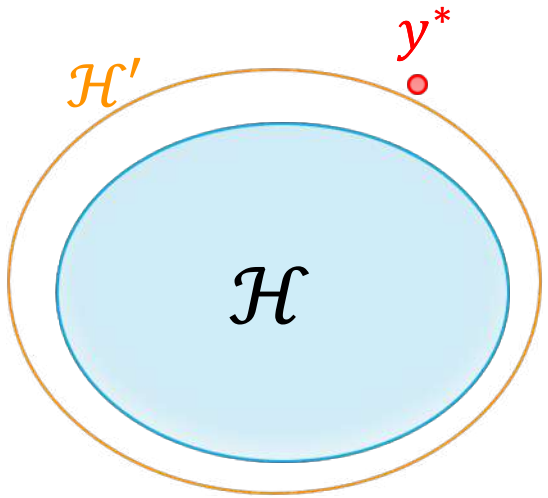
- A NN maps an input $\boldsymbol{x}$ to an output $\mathrm{y} = h(\boldsymbol{x}, \boldsymbol{W})$
- *Training:* often find weights $\mathbf{W}$, by minimizing a loss $F(\boldsymbol{D}, h)$



$Error(h_o) :=$ Optimization error +Generalization error +Approximation error

Approximation error: classical

$$\|y^* - h\| \leq \epsilon_a$$



- Increase capacity ➔ approximate better
  - Larger family $\mathcal{H}'$
  - More complex NNs ➔ stronger representational power
  - E.g., wider or deeper NNs
- Any binary function can be learnt (approximately well) by a feedforward network using one hidden layer, when the **width goes to infinity**
- Any bounded continuous function can be learnt (approximately) by a *feedforward network* using one hidden layer [Cybenko, 1989; Hornik, 1991]

Cybenko, G. (1989). Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*.
Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251-257.

Approximation error: modern

- Any continuous function can be approximated arbitrarily well by Convolutional neural network, when the depth is large [Zhou, 2020]
- Any Lebesgue-integrable function can be approximated arbitrarily well by a ResNet with **one neuron** per hidden layer [Lin & Jegelka, 2018]
- **Deep** NNs avoid the curse of dimensionality when approximating Lipschitz functions [Poggio et al. 2017]
  - Shallow NNs cannot

Universal tools

Lin, H., & Jegelka, S. (2018). ResNet with one-neuron hidden layers is a universal approximator. *NeurIPS*.
Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., & Liao, Q. (2017). Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*.
Zhou, D. X. (2020). Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*.

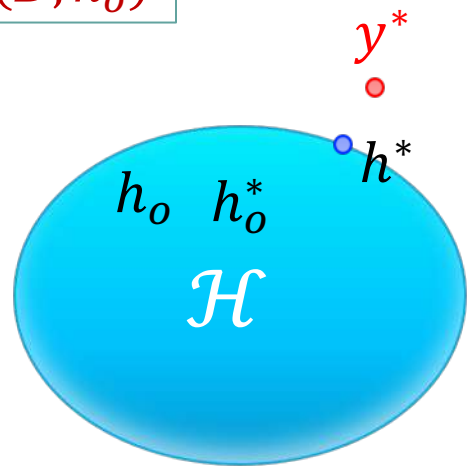
Approximation: existence $\not\to$ method

Unclear
how to find such DNNs,
based on a training set

Optimization error

- Training is often by minimizing a loss $F(\boldsymbol{D}, h)$
- The training loss is *highly non-convex*
- Theory:
  - Exponentially large number of iterations may be needed
  - Intractable in the worst case [Nesterov, 2018]
- Practice:
  - Often have zero training error ➔ global solution h_o^*?
  - Easily perfectly fit random labelling of data [Zhang et al. 2021] (training seems to be easy!)
- Contradiction? What's missing?

$$F(\boldsymbol{D}, h_o) - F(\boldsymbol{D}, h_o^*)$$



Nesterov, Y. (2018). *Lectures on convex optimization*. Springer.
Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*.

Optimization: theoretically easy

- Gradient descent (GD) achieves zero training loss in polynomial time for a deep over-parameterized ResNet [Du et al. 2019]
  - Over-parameterization: #parameters ≫ training size
- GD can find a global optimum when the width of the last hidden layer of an MLP exceeds the number of training samples [Nguyen, 2021]
- Stochastic gradient descent (SGD) can find global minima on the training objective of DNNs in polynomial time [Allen-Zhu et al. 2019]
  - Architecture: MLP, CNN, ResNet

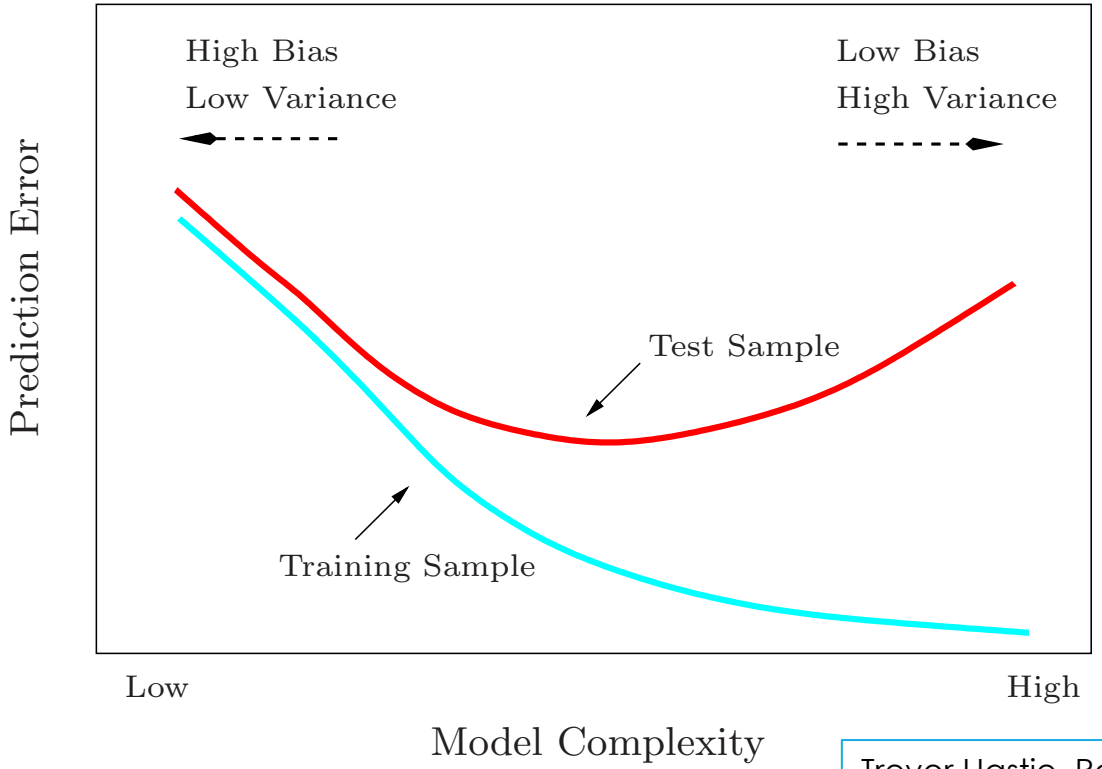
Du, S., Lee, J., Li, H., Wang, L., & Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*.
Nguyen, Q. (2021). On the proof of global convergence of gradient descent for deep relu networks with linear widths. In *International Conference on Machine Learning*.
Allen-Zhu, Z., Li, Y., & Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*.

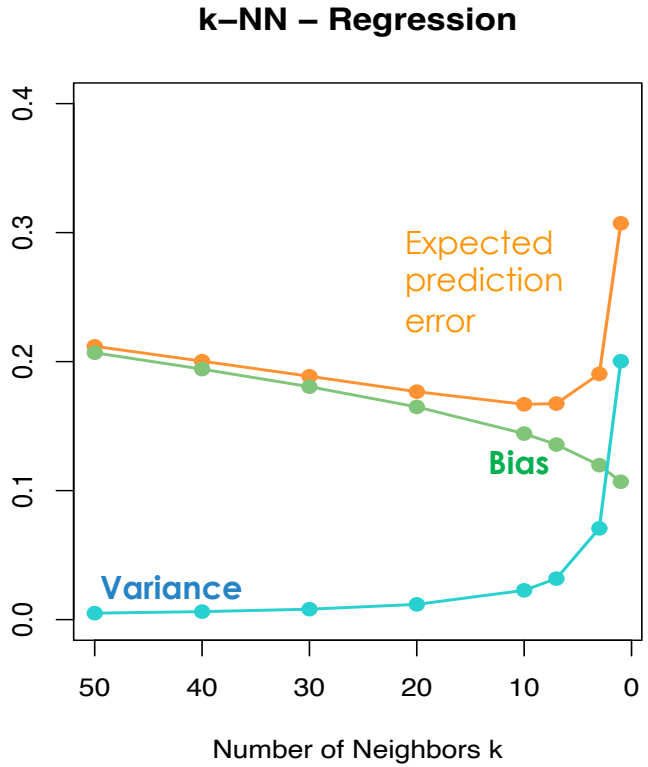
Optimization: reminder

However
global optimality
of the training problem
does not imply
good predictive ability

Bias-Variance tradeoff: classical view

- The more complex the model is, the more data points it can capture, and the lower the bias can be
  - However, higher complexity will make the model "move" more to capture the data points, and hence its variance will be larger.





Trevor Hastie, Robert Tibshirani, Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

Bias-Variance: modern behavior

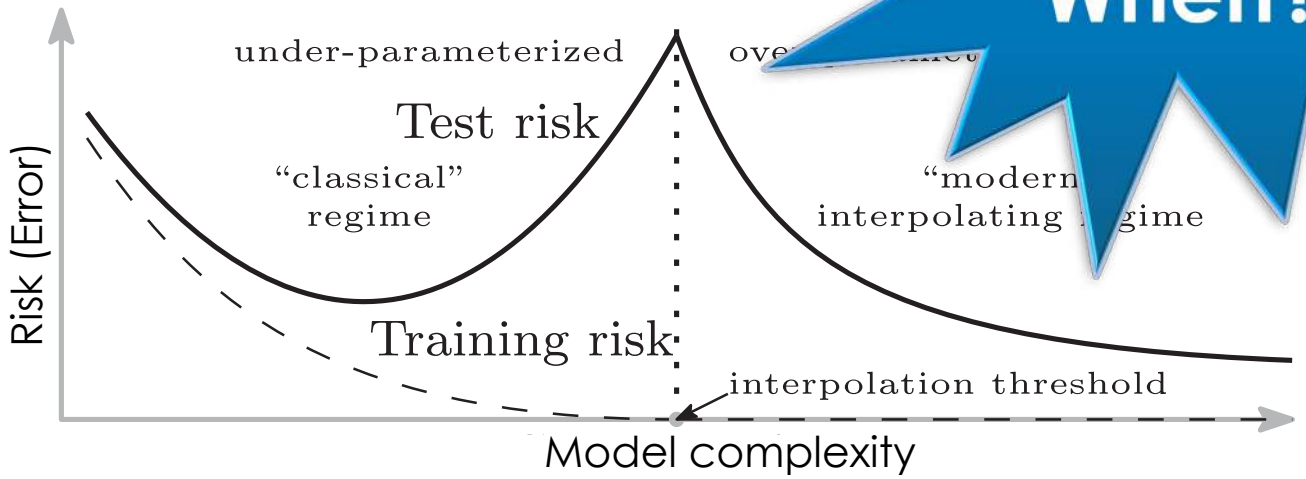
- Modern phenomenon:

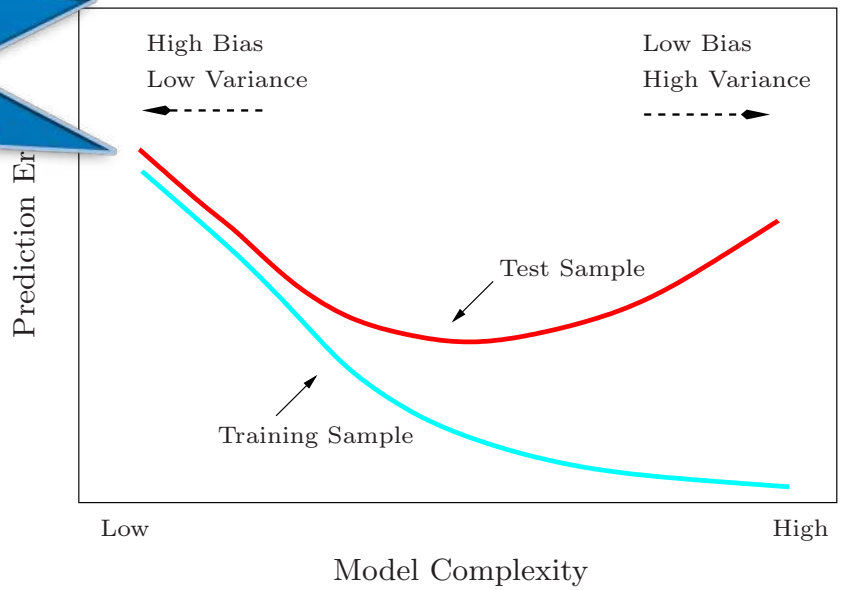
  Very rich models such as DNNs are trained to ***exactly fit*** *the data, but often obtain* ***high accuracy*** *on test data* [Belkin et al., 2019]

  - $Bias \cong 0$
  - GPT-4, ResNets, StyleGAN, B

- Classical view:
  more complex model
  - Lower bias, higher variance

Why?
When?





Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, *116*(32), 15849-15854.

Generalization ability: long-standing open

- **Main goal:** small expected loss $F(P, h_o)$
  - Practice: training loss $F(\boldsymbol{D}, h_o) \cong 0$ for overparameterized NNs
- Why can a trained DNN generalize well?
  (Generalization: ability to well perform on unseen data)
- We want to assure, for $\delta > 0$,

$$\Pr(|F(P, h_o) - F(\boldsymbol{D}, h_o)| \leq \epsilon) \geq 1 - \delta$$

  - Generalization gap should be small with a high probability over the random choice of **D**
  - How fast does $F(\boldsymbol{D}, h_o)$ converge to $F(P, h_o)$?
    (as the training size increases)

$Error(h_o) :=$
Approximation error
+Optimization error
+Generalization error

A long-standing challenge in DL theory

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of Machine Learning*. MIT press.
Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*.

Generalization: VC dimension

- Vapnik–Chervonenkis (VC) dimension:
  - Measure of the capacity (complexity, expressive power, richness) of a set of functions
  - The cardinality of the largest set of points that the algorithm can shatter
  - A higher VC dim ➔ richer model family $\mathcal{H}$
- Example: in n-dimensional space
  - Linear models: $VC(\mathcal{H}) = n + 1$
  - ReLU networks with W weights: $VC(\mathcal{H}) = \Omega(W \log W)$

Bartlett, P. L., Harvey, N., Liaw, C., & Mehrabian, A. (2019). Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research*.

- Classical bound: for any $\delta > 0$, with probability at least $1 - \delta$

$$F(P, h) - F(\boldsymbol{D}, h) \leq \sqrt{\frac{2}{m} VC(\mathcal{H}) \log \frac{2e.m}{VC(\mathcal{H})} + \frac{1}{m} \log \frac{2}{\delta}}$$

- Vacuous/meaningless for modern DNNs, due to $W \gg m$ (training size)

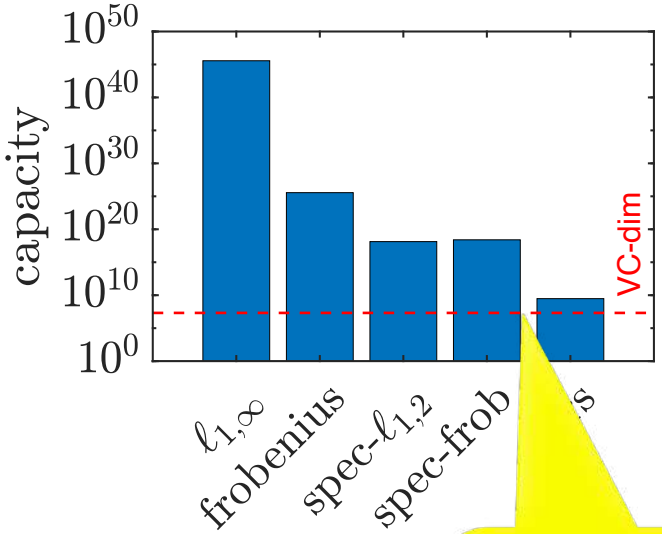
Generalization: Weight norm

- DNN: $h(\boldsymbol{x}, \boldsymbol{W}) = g_K(\boldsymbol{W}_K h_{K-1})$
- Bartlett: **#params** is not important
  - Size of weights may be more important
- Neyshabur et al.; Golowich et al.:

$$F(P,h) - F(\boldsymbol{D},h) \leq O(\|\boldsymbol{W}_1\|_F \cdots \|\boldsymbol{W}_K\|_F)/\sqrt{m}$$

- Bartlett et al.:

$$F(P,h) - F(\boldsymbol{D},h) \leq O(\|\boldsymbol{W}_1\|_2 \cdots \|\boldsymbol{W}_K\|_2)/\sqrt{m}$$



Uninformative for modern DNNs

Arora, S., Ge, R., Neyshabur, B., & Zhang, Y. (2018). Stronger generalization bounds for deep nets via a compression approach. In *ICML*.

Bartlett, P. (1998). The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*.

Bartlett, P. L., Foster, D. J., & Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. *Neural Information Processing Systems*.

Golowich, N., Rakhlin, A., & Shamir, O. (2020). Size-independent sample complexity of neural networks. *Information and Inference: A Journal of the IMA*.

Neyshabur, B., Bhojanapalli, S., & Srebro, N. (2018). A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks. In *ICLR*.

Generalization: PAC-Bayes

- Consider $\mathbb{E}_{h\sim\rho}[F(P,h) - F(\boldsymbol{D},h)]$
  - Generalization error on average over $\mathcal{H}$
  - ρ is the posterior distribution of h
- McAllester: with probability at least $1-\delta$

$$\mathbb{E}_{h\sim\rho}[F(P,h) - F(\boldsymbol{D},h)] \leq \sqrt{\frac{KL(\rho||\mu) + \log(m/\delta)}{2m-1}}$$

  - μ is the prior distribution of h
  - KL is the Kullback-Leibler divergence

- The "distance" between posterior ρ and prior μ:
  - Plays important role
  - Depends on the *bias* of a learning algorithm
- Unclear how fast can ρ approach μ?
- Do not directly consider the complexity of family $\mathcal{H}$

Meaningful bounds appeared

McAllester, D. A. (2003). PAC-Bayesian stochastic model selection. *Machine Learning*, *51*(1), 5-21.

Generalization: non-vacuous bounds

- We can optimize the PAC-Bayes bound
  - Find the posterior ρ^* that minimizes $KL(\rho||\mu)$
- Dziugaite & Roy: non-vacuous bounds
  - MLP with 3 layers, SGD algorithm, MNIST dataset
- Zhou et al.: compressibility
  - Use SOTA compression alg. to find nonvacuous bound for ImageNet, LeNet-5
- Lotfi et al.:
  - Propose compression alg. to find nonvacuous bounds for LeNet-5, ResNet-18, MobileViT

Stochastic DNNs

Dataset	Data-independent priors	
	Err. Bound (%)	SOTA (%)
MNIST	**11.6**	21.7 [59]
+ SVHN Transfer	**9.0**	16.1†
FashionMNIST	**32.8**	46.5†
+ CIFAR-10 Transfer	**28.2**	30.1†
CIFAR-10	**58.2**	89.9†
+ ImageNet Transfer	**35.1**	54.2†
CIFAR-100	**94.6**	100†
+ ImageNet Transfer	**81.3**	98.1†
ImageNet	**93.5**	96.5 [73]

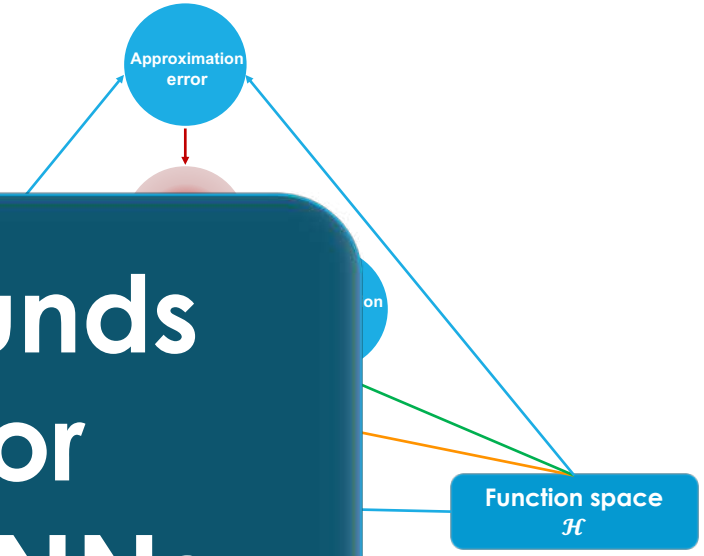
Biggs & Guedj:

- Non-vacuous bounds for a (special) **deterministic networks**
- MNIST and Fashion-MNIST datasets

Dziugaite, G., & Roy, D. (2017). Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. In *UAI*.
Zhou, W., Veitch, V., Austern, M., Adams, R., & Orbanz, P. (2019). Non-vacuous Generalization Bounds at the ImageNet Scale: a PAC-Bayesian Compression Approach. In *ICLR*.
Lotfi, S., Finzi, M., Kapoor, S., Potapczynski, A., Goldblum, M., & Wilson, A. G. (2022). PAC-bayes compression bounds so tight that they can explain generalization. *In NeurIPS*.
Biggs, F., & Guedj, B. (2022). Non-vacuous generalisation bounds for shallow neural networks. In *ICML*.

Generalization: long-standing open

- Some other approaches:
  - Neural tangent kernel, Mean field
  - Algorithm



Current meaningful bounds however are mostly for stochastic or shallow NNs

Unclear about
Big pretrained models, Deep NNs in practice

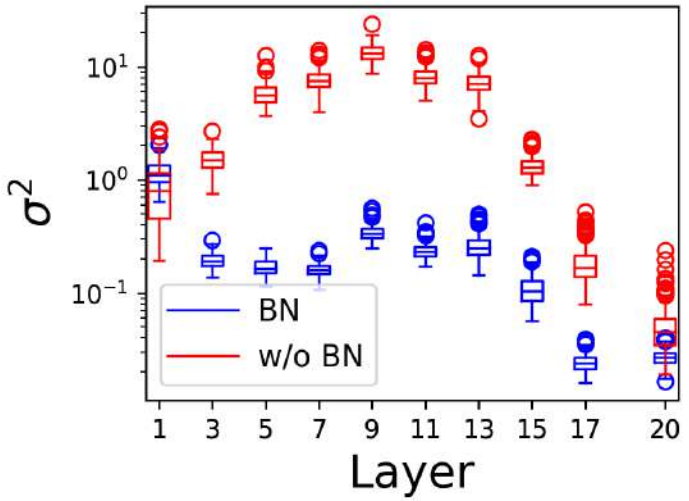
Unclear about
Why many tricks in DL improve performance

Normalization methods

The exponential benefits

Batch Normalization

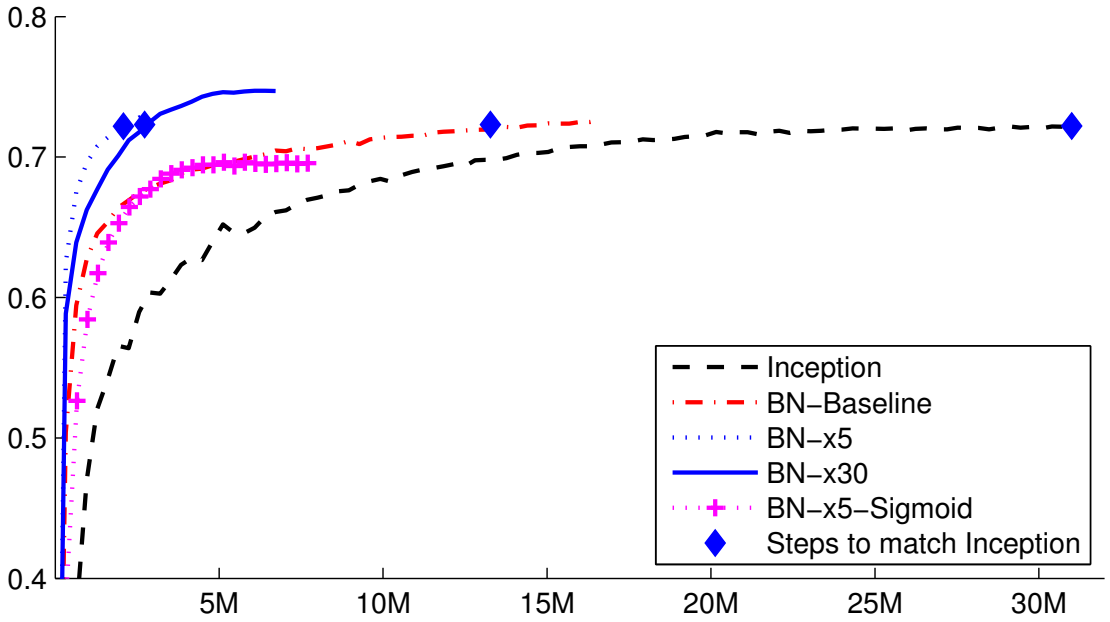
$$h(\boldsymbol{x}, \boldsymbol{W}) = g_K(\boldsymbol{W}_K h_{K-1}), \qquad \text{where } h_i = g_i(\boldsymbol{W}_i h_{i-1}), h_0 = \boldsymbol{x}$$



- **Large variance** of the input at a layer
  - Training is often unstable, gradient explosion may appear
- *Batch normalization:* [Ioffe & Szegedy, 2015]

$$h_i = g_i(\boldsymbol{BN}(\boldsymbol{W}_i h_{i-1}))$$

  - Each input signal will have *mean 0* and *variance 1*
- DNN + BN:
  - Training is often easier and faster
  - Become a standard
  - Have much better generalization @@



Ioffe & Szegedy (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Many normalizers

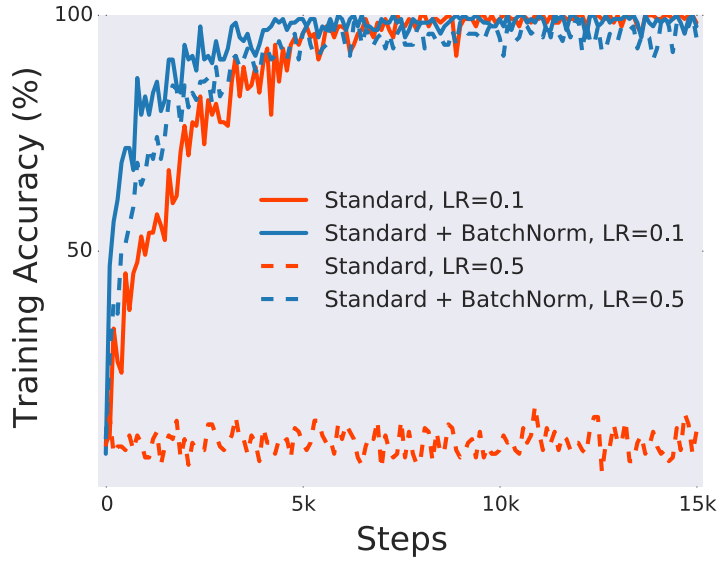
- Batch normalization
- Layer normalization [Ba et al., 2016]
- Instance normalization [Ulyanov et al., 2016]
- Group normalization [Wu & He, 2020]
- Spectral normalization [Miyato et al., 2018]
- Weight normalization [Salimans & Kingma et al., 2016]
- …

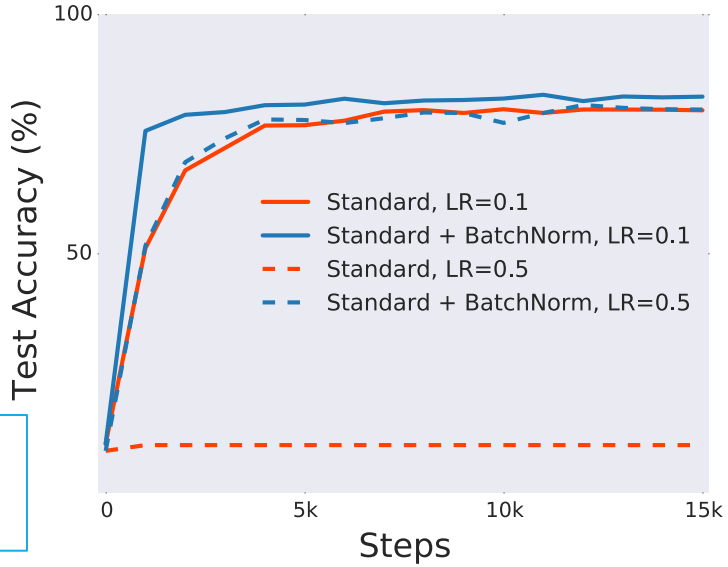
Ioffe & Szegedy (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.
Ba, Kiros, and Hinton. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.
Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018) Spectral Normalization for Generative Adversarial Networks. In *ICLR*.
Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.
Salimans, T., & Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*.
Wu & He. (2020). Group normalization. International Journal of Computer Vision.

Normalizers: many why's

$$h(\boldsymbol{x}, \boldsymbol{W}) = g_K(\boldsymbol{W}_K h_{K-1}), \text{where } h_i = g_i(\boldsymbol{W}_i h_{i-1}), h_0 = \boldsymbol{x}$$

- Batch normalization: $h_i = g_i(\boldsymbol{BN}(\boldsymbol{W}_i h_{i-1}))$
- Why can they fasten training?
  - BN can reduce the Lipschitz constant of the loss
    → flatten the loss [Santurkar et al. 2018; Lyu et al. 2022]
  - Unclear about other normalizers
- Does they control the capacity of a neural net?
  - Yes, for a *single-layer perceptron* [Luo et al. 2018]
- Why can they improve generalization? Unclear





Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization?. In *NeurIPS*.
Luo, P., Wang, X., Shao, W., & Peng, Z. (2019). Towards Understanding Regularization in Batch Normalization. In *ICLR*.
Lyu, Li, & Arora. (2022). Understanding the Generalization Benefit of Normalization Layers: Sharpness Reduction. In *NeurIPS*.

Normalization

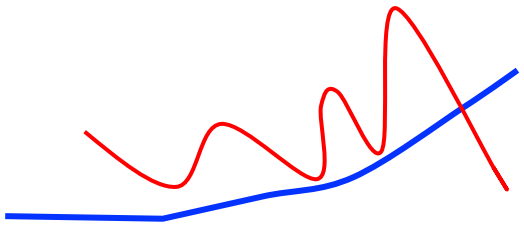
Despite being key parts of modern Deep Learning, theoretical understanding for normalization methods remains missing

Normalization: our work

- BN can reduce the Lipschitz constant of a DNN at an exponential rate
  - Capacity control, regularization role
  - Many other normalizers have this property
- A normalized DNN can have exponentially smaller generalization error than its unnormalized version in the worst case
  - A normalized DNN may require exponentially less training data
- A normalizer can make the training loss to be exponentially flatter
  - Iteration complexity is exponentially reduced
    (The required number of iterations for any gradient-based learning algorithm)
  - Training can converge exponentially faster

Normalization: Lipschitz continuity

- Lipschitz constant
  - tells how fast a function h(**x**) can change at an input **x**
  - represents the complexity of h(**x**) ➔ capacity of h



- BN normalizes each input $x^{(i)}$ as

$$\text{BN}(x^{(i)}, \gamma, \epsilon) = \frac{\gamma}{\sqrt{\sigma_x^2 + \epsilon}}(x^{(i)} - \mu_x)$$

  - with mean μ_x and variance σ_x^2. γ is trainable, ϵ is a smoothing constant
  - Lipschitz constant w.r.t. input **x**: $\|BN\|_{Lip} = \|\boldsymbol{\gamma/\sigma}\|$ $\quad \boldsymbol{\gamma/\sigma} = \left(\gamma_1/\sqrt{\sigma_1^2 + \epsilon}, \ldots, \gamma_n/\sqrt{\sigma_n^2 + \epsilon}\right)$
- Other normalizers:
  - Layer norm: $\|LN\|_{Lip} \leq \left(1 - \frac{1}{n}\right)\|\boldsymbol{\gamma/\sigma}\|$ (for a layer with n neurons)
  - Group norm: $\|GN\|_{Lip} \leq \left(1 - \frac{1}{n_g}\right)\|\boldsymbol{\gamma/\sigma}\|$ (for a group of n_g neurons)

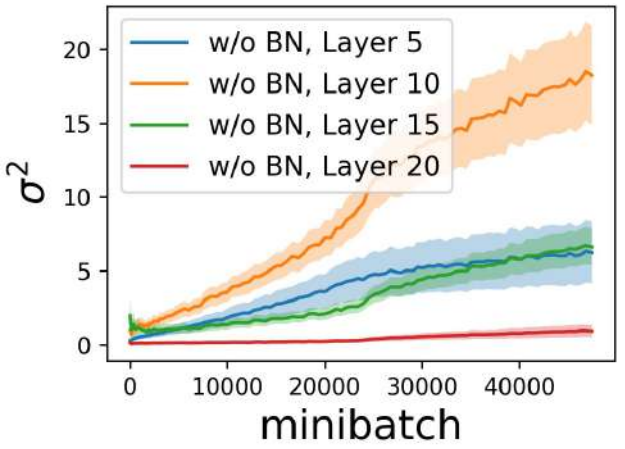
Exponential benefits: Capacity control

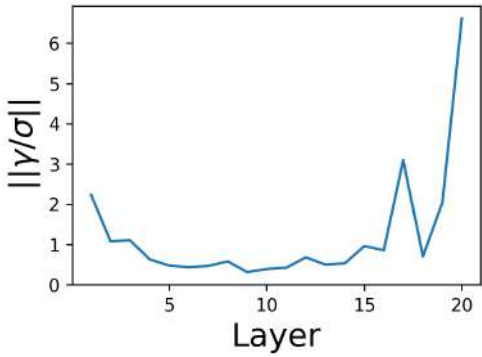
Definition 1 (DNN) *Let fixed activation functions* $(g_1, \ldots, g_K)$*, whose Lipschitz constants are at most 1, and the bounds* $(s_1, \ldots, s_K)$*. Let* $\boldsymbol{h}_i = g_i(\mathrm{NO}_i(\boldsymbol{W}_i \boldsymbol{h}_{i-1}))$*, with* $\boldsymbol{h}_0 = \boldsymbol{x}$*, be the neural network associated with weight matrices* $(\boldsymbol{W}_1, \ldots, \boldsymbol{W}_i)$*, where* NO_i *denotes a normalization operator at layer* i*. We define* $\boldsymbol{h}_{no}(\boldsymbol{x}) = \boldsymbol{h}_K(\boldsymbol{x})$ *where* $\boldsymbol{x} \in \mathcal{X}, \|\boldsymbol{W}_i\| \leq s_i, \forall i \leq K$*. Denote* $\boldsymbol{h}(\boldsymbol{x})$ *as the unnormalized version of* $\boldsymbol{h}_{no}(\boldsymbol{x})$*.*

Lemma 4 *For any* $\boldsymbol{h}$ *defined in Definition 1, we have* $\|\boldsymbol{h}\|_{Lip} \leq \prod_{k=1}^{K} s_k$*. Its normalized version satisfies* $\|\boldsymbol{h}_{no}\|_{Lip} \leq \prod_{k=1}^{K} s_k \|\mathrm{NO}_k\|_{Lip}$.

- For BN: $\|\boldsymbol{h}_{no}\|_{Lip} \leq \left(\prod_{k=1}^{K} \|\boldsymbol{\gamma}_k / \boldsymbol{\sigma}_k\|\right) \prod_{k=1}^{K} s_k$
- Reminder: "Large variance of the input"
  - Often appears in practice
- ResNet-18: $\prod_{k=1}^{K} \|\boldsymbol{\gamma}_k / \boldsymbol{\sigma}_k\| \approx 1.35 \times 10^{-8} \approx 1.35 \times 2^{-26.5}$

Exponential reduction!





Exponential benefits: Generalization

- We study bridge between *Lipschitz property & generalization ability*
  - Γ denote a partition of input space $\mathcal{X}$ into small parts: $\mathcal{X}_i$ with diameter λ_i
  - L_i is the *local Lipschitz constant* of loss f on $\mathcal{X}_i$

Theorem 1 *Consider a function* $\boldsymbol{h}$ *defined over* $\mathcal{X}$*, and* $\boldsymbol{D}$ *consisting of* m *i.i.d. samples from distribution* P*. Assume that the loss* $f(\boldsymbol{h}, \boldsymbol{x})$ *is Lipschitz continuous on every* $\mathcal{X}_i, i \in \boldsymbol{T}_D$*. For any* $\delta > 0$*, denoting* $g(\Gamma, \boldsymbol{D}, \delta) = C(\sqrt{2}+1)\sqrt{\frac{|\boldsymbol{T}_D|\log(2N/\delta)}{m}} + \frac{2C|\boldsymbol{T}_D|\log(2N/\delta)}{m}$*, we have the following with probability at least* $1-\delta$*:*

$$|F(P, \boldsymbol{h}) - F(\boldsymbol{D}, \boldsymbol{h})| \leq \sum_{i \in \boldsymbol{T}_D} \frac{m_i}{m} \lambda_i L_i + g(\Gamma, \boldsymbol{D}, \delta) \tag{6}$$

- Generalization error:
  - Depends heavily on the local behaviors around training samples
  - A DNN with smaller Lipschitz constant may generalize better ($L_i \leq \|f\|_{Lip} \|\boldsymbol{h}_{no}|\mathcal{X}_i\|_{Lip}$)
  - Explain why many adversarial training methods are reasonable

Exponential benefits: Generalization

$$|F(P,\boldsymbol{h}) - F(\boldsymbol{D},\boldsymbol{h})| \leq \omega + O(m^{-0.5})$$

$$|F(P,\boldsymbol{h}_{no}) - F(\boldsymbol{D},\boldsymbol{h}_{no})| \leq \omega\left(\prod_{k=1}^{K}\|\boldsymbol{\gamma}_k/\boldsymbol{\sigma}_k\|\right) + O(m^{-0.5}) \quad \text{(For BN)}$$

- Ex.: $\prod_{k=1}^{K}\|\boldsymbol{\gamma}_k/\boldsymbol{\sigma}_k\| \approx 1.35\times 2^{-26.5}$

- A normalized DNN can have exponentially smaller generalization error than its unnormalized version in the worst case
  - A normalized DNN may require exponentially less training data
- "Deeper" can save more samples
- Same property appears for many normalizers

Exponential benefits: Optimization

- Training loss $F(\boldsymbol{D}, \boldsymbol{h})$ can be flatten exponentially by a normalizer
  - Consider DNN $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_K)$, a layer $\boldsymbol{h}_i = g_i(\boldsymbol{W}_i \boldsymbol{h}_{i-1})$, and a loss function $f(\boldsymbol{h}, \boldsymbol{x})$

$$\frac{\partial f}{\partial \boldsymbol{W}_i} = \frac{\partial f}{\partial \boldsymbol{h}} \times \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{y}_i} \times \boldsymbol{h}_{i-1} \qquad where\ \boldsymbol{y}_i = \boldsymbol{W}_i \boldsymbol{h}_{i-1}$$

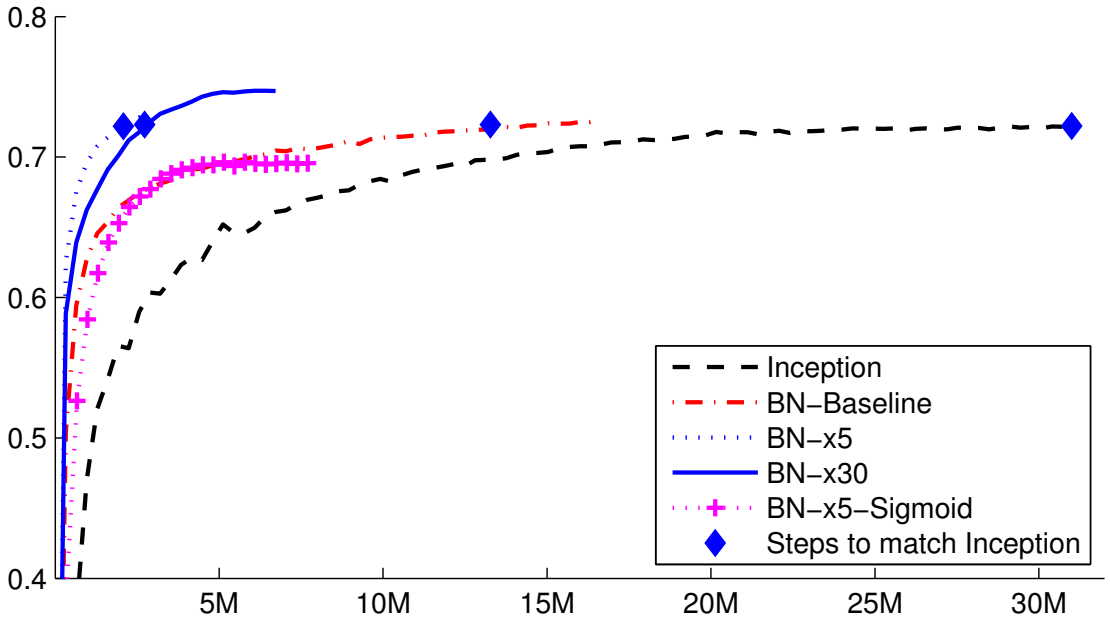
  - The gradient w.r.t weight $\boldsymbol{W}_i$ depends on $\partial \boldsymbol{h} / \partial \boldsymbol{y}_i$ (Lipschitz constant of **h** w.r.t layer *i*)
  - $\|\boldsymbol{h}_{no}\|_{Lip}$ can be exponentially smaller than $\|\boldsymbol{h}\|_{Lip}$

  ➔ $F(\boldsymbol{D}, \boldsymbol{h}_{no})$ can be exponentially flatter than $F(\boldsymbol{D}, \boldsymbol{h})$

- Consequences on training:
  - Iteration complexity (lower bound on #iterations)
  - Convergence rate (upper bound on #iterations)

Exponential benefits: Convergence rate

- **Iteration complexity:** any gradient-based methods require **at least**
  - $\Omega(\|F\|_{Lip}/\alpha)$ iterations to find an α-stationary point of nonconvex function F [Zhang et al.]
- **Convergence rate:** after T iterations, we can find an approximate solution
  - with error $O\left(\sqrt[3]{\|F\|_{Lip}^2/T}\right)$, for nonconvex function F [Davis et al.]
  - with error $O\left(\|F\|_{Lip}/\sqrt{T}\right)$, for convex function F
- $\|F\|_{Lip}$ can be made exponentially smaller by a normalizer
  - Iteration complexity is exponentially reduced
  - Training can converge exponentially faster



Zhang, J., Lin, H., Jegelka, S., Sra, S., & Jadbabaie, A. (2020). Complexity of finding stationary points of nonconvex nonsmooth functions. In *ICML*.
Davis, D., Drusvyatskiy, D., Lee, Y. T., Padmanabhan, S., & Ye, G. (2022). A gradient sampling method with complexity guarantees for lipschitz functions in high and low dimensions. *NeurIPS*.

Take-home messages

- Deep neural networks are universal approximators
- Theoretically clear about:
  - Approximation ability
  - Optimization (learning process)
- Normalization methods have exponential benefits
- Long-standing open challenge about Generalization ability