

Theory and Deep Learning

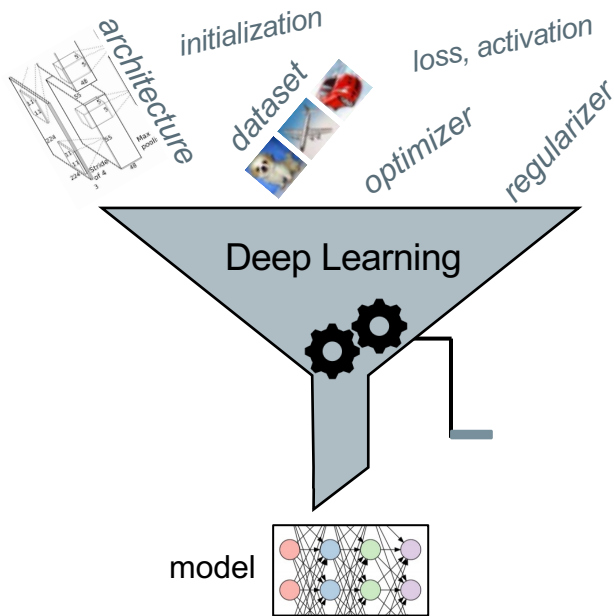
Preetum Nakkiran
UCSD

Open Question: What is Deep Learning?

Open Question: What is Deep Learning?

Deep Learning (informal):

A set of *ingredients* that can be combined to solve a certain *learning problems*



Why Study Deep Learning?

1. ~~Mathematical beauty~~

2. ~~Theoretical depth~~

yet...

Why Study Deep Learning?

“Theory for Deep Learning”

1. Science: “something out there we don’t understand...”

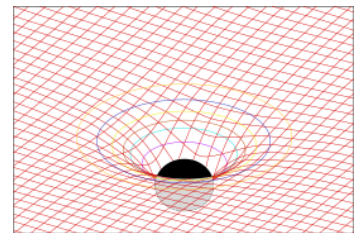
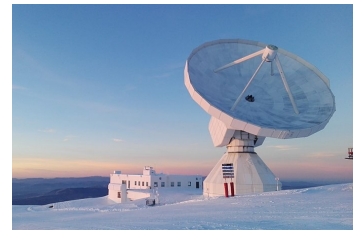
- consistently *surprised* by DL...

“Deep Learning for Theory”

2. Theory: “...what does it teach us in general?”

“What can Deep Learning teach us about learning?”

- **new** framework of learning, deep mathematics..
- what can we learn, deeply or otherwise?
- what object generalizes deep learning?



Why Study Deep Learning?

“Theory for Deep Learning”

1. Science: “something out there we don’t unders

- consistently *surprised* by DL...

“Deep Learning for Theory”

2. Theory: “...what does it teach us in general?”

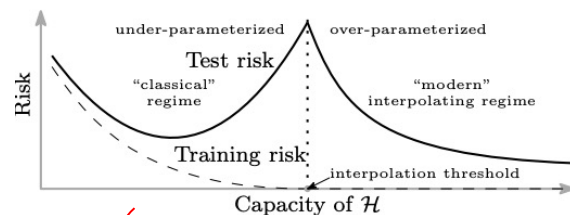
“What can Deep Learning teach us ab

- **new** framework of learning, deep n
- what can we learn, deeply or other
- what object generalizes deep learn

Reconciling modern machine learning practice
and the bias-variance trade-off

Mikhail Belkin^a, Daniel Hsu^b, Siyuan Ma^a, and Soumik Mandal^a

[stat.ML] 10 Sep 2019



Surprises in High-Dimensional Ridgeless Least Squares
Interpolation

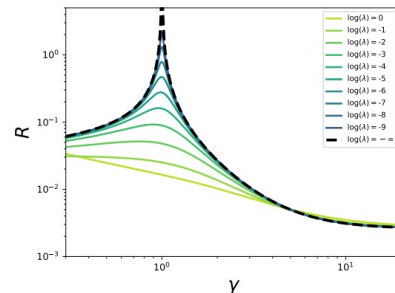
Trevor Hastie

Andrea Montanari*

Saharon Rosset

Ryan J. Tibshirani*

[math.ST] 7 Dec 2020



Why Study Deep Learning?

“Theory for Deep Learning”

1. Science: “something out there we don’t understand...”

- consistently *surprised* by DL...

“Deep Learning for Theory”

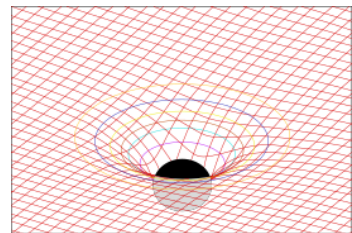
2. Theory: “...what does it teach us in general?”

“What can Deep Learning teach us about learning?”

- **new** framework of learning, deep mathematics?
- what can we learn, deeply or otherwise?
- what object generalizes deep learning?

3. Money: “*notice me NSF*”

4. Frustration: “*why won’t my students do anything else*”



Why Study Deep Learning?

Many things we don't fully understand
(linear regression, 1-nearest-
neighbors, decision trees, SAT solvers...)

Q: Why study Deep Learning specifically?

A: Richness (← surprisal)

- New domains, emergent behaviors...



ARTICLE

doi:10.1038/nature24270

Mastering the game of Go without human knowledge

David Silver^{1*}, Julian Schrittwieser^{1*}, Kai
Thomas Hubert¹, Lucas Baker¹, Matthew
George van den Driessche¹, Thore Graepel¹

IT WILL CHANGE EVERYTHING': AI MAKES GIGANTIC LEAP IN SOLVING PROTEIN STRUCTURES

The New

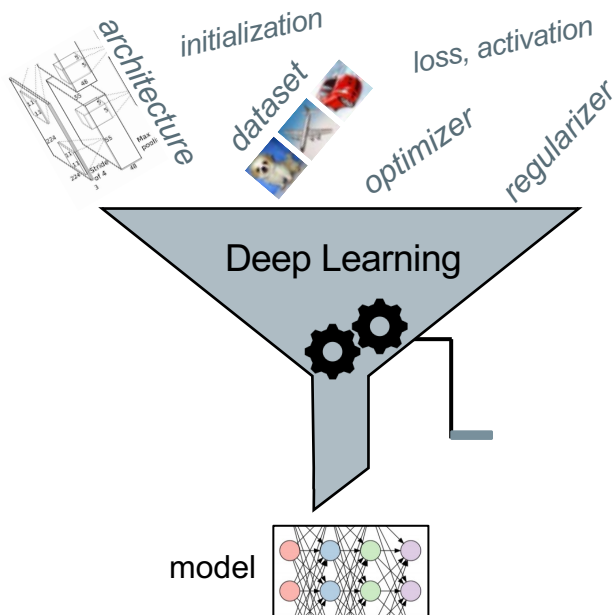
DeepMind's program for determining the 3D shapes
of proteins stands to transform biology, say scientists.

Meet GPT-3. It Has Learned to Code (and Blog and Argue).

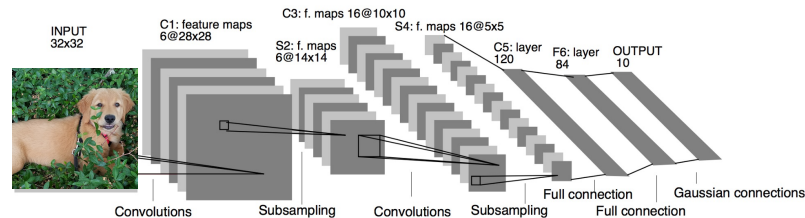
The latest natural-language system generates tweets, pens poetry,
summarizes emails, answers trivia questions, translates
languages and even writes its own computer programs.



Advances are *Unpredictable*

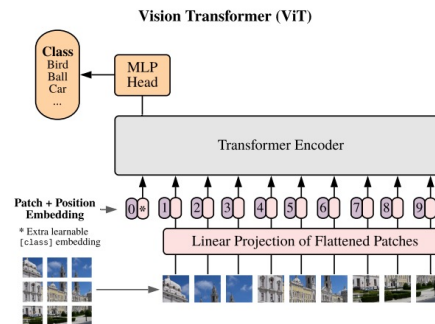


~1998-2020: ConvNets dominate vision



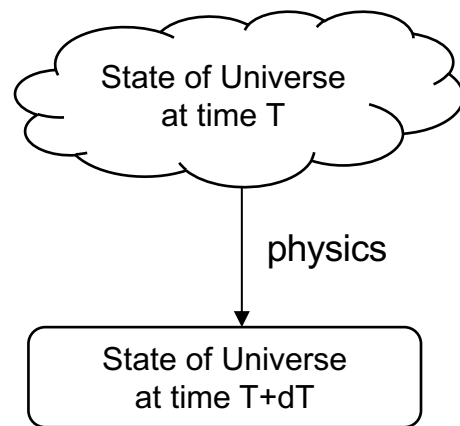
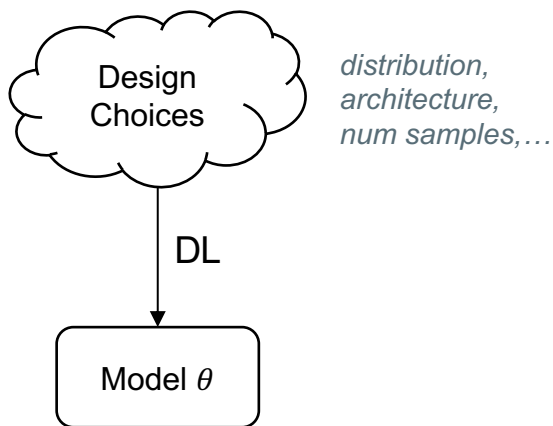
[LeCun et al 1998]

2020: *Transformers* (from NLP) dominate vision



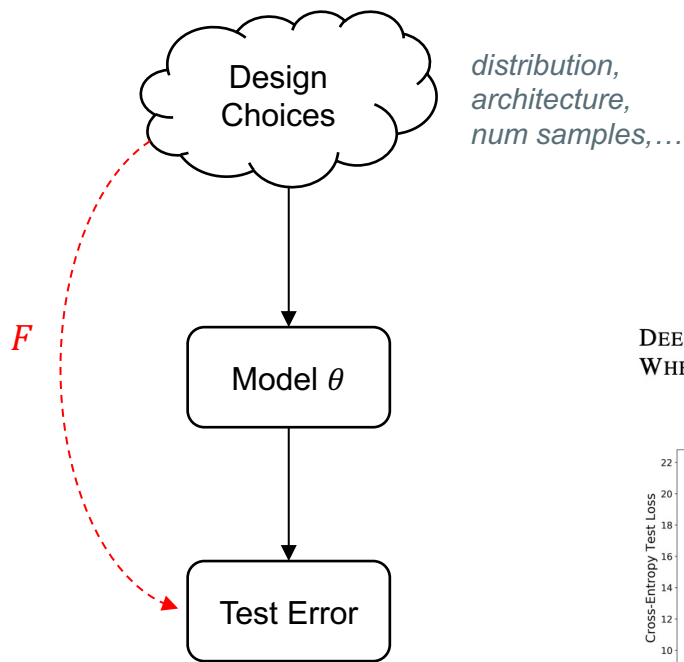
[Dosovitskiy et al 2020]

What does it mean to “Understand DL”?



“ How does what we **do**
affect what we **get?** ”

What does it mean to “Understand DL”?



“Generalization”

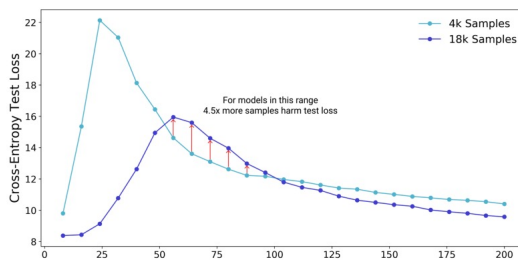
= Understanding **F**

= **Identifying structure** in **F**

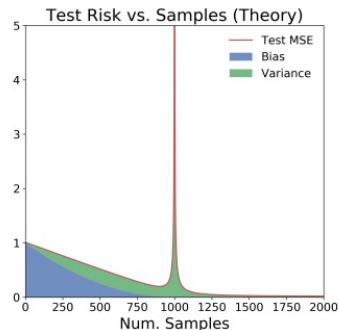
Examples of Structure:

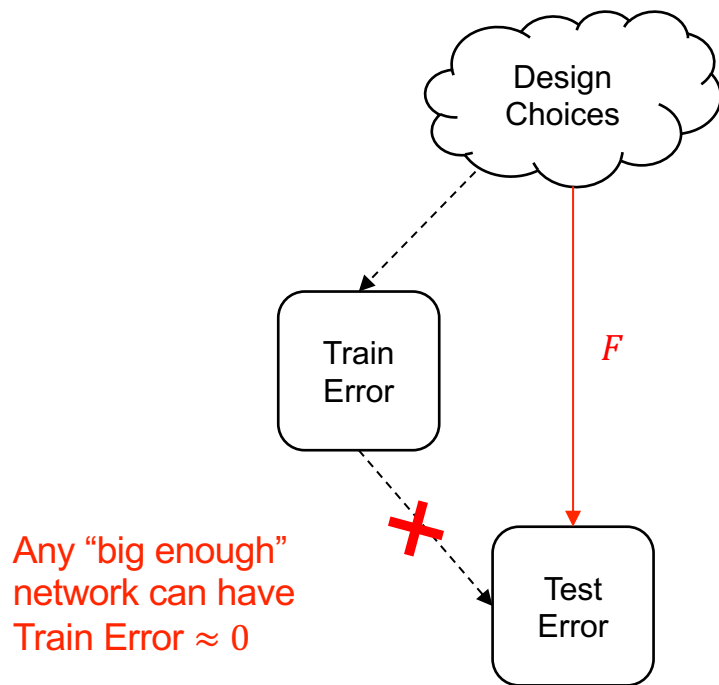
- Monotonicity in N (samples)

DEEP DOUBLE DESCENT:
WHERE BIGGER MODELS AND MORE DATA HURT



More Data Can Hurt for Linear Regression:
Sample-wise Double Descent



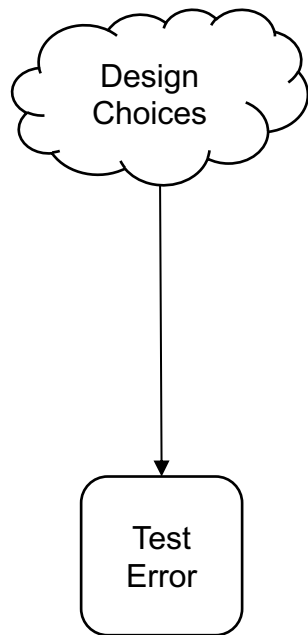


Examples of Structure:

- Factorization of F
(ex: ERM framework)

How to Study Deep Learning?

Obstacles to Mathematical Rigor



“Generalization Theorem:

Deep neural nets with design choices X ,
on **distribution** $Y \in \mathcal{Y}$, have test error

$$F(X, Y) \leq G(X, Y)$$

for some explicit G ”

Not even too hard. Too ill-defined!

1. Can’t define “deep neural nets”

(big enough to include practice,
small enough to exclude P/poly)

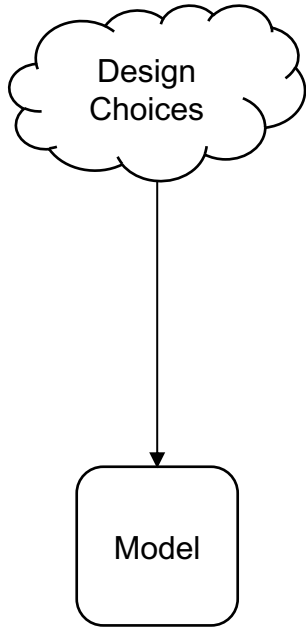
*“natural architectures
barrier”*

SGD on a “universal architecture” can
simulate P/poly [Abbe, Sandon 2020]

2. Can’t define the tasks they solve
(Vision, NLP,...)

*“natural distributions
barrier”*

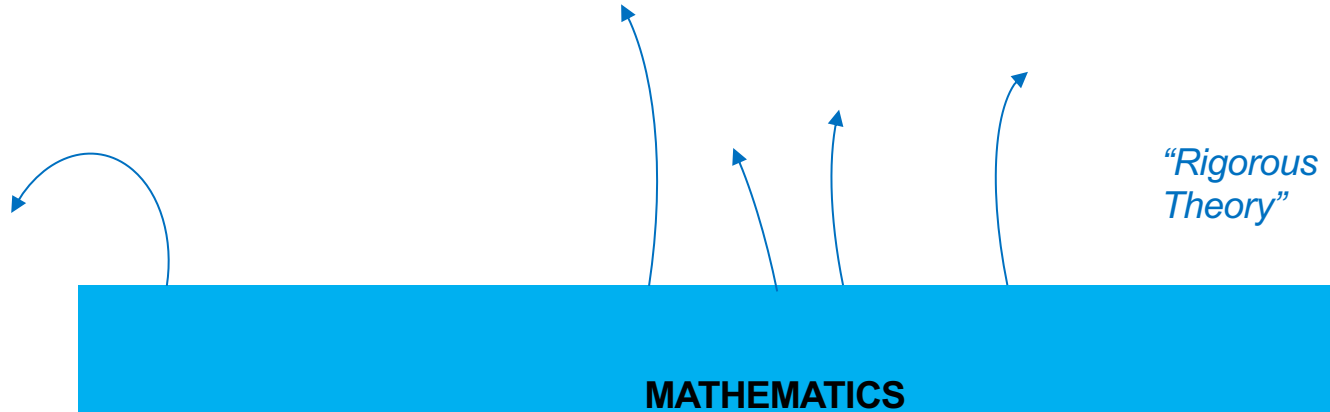
The Two Cultures



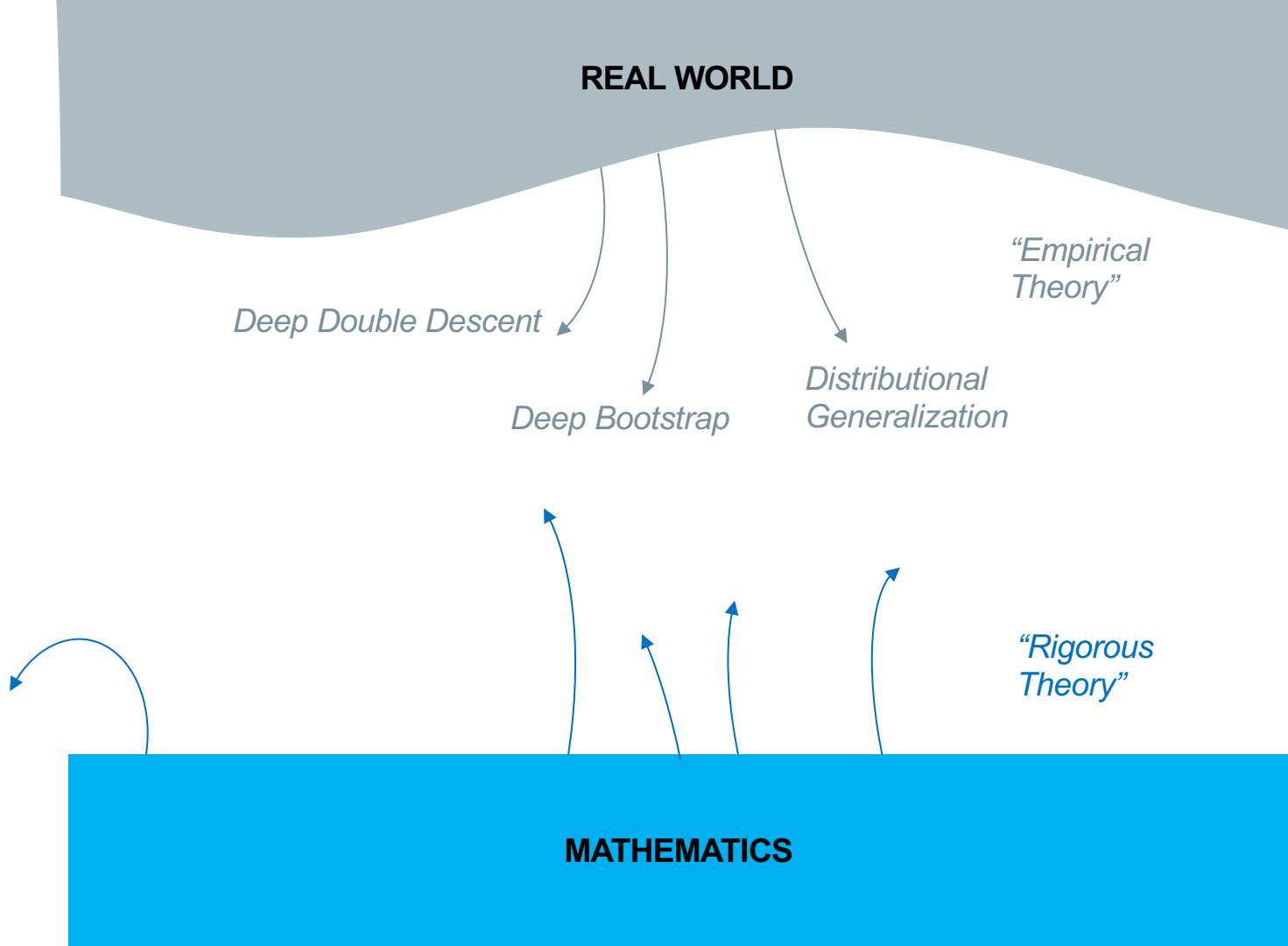
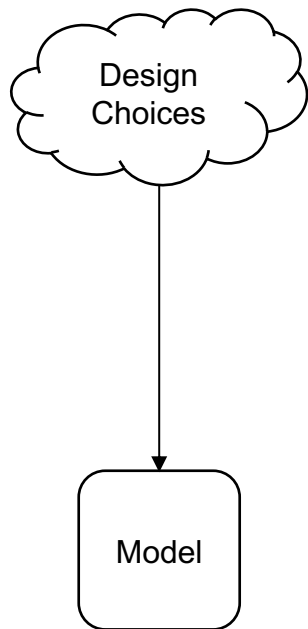
Empirical Theory in Physics:

- Kepler's **Laws**
- Ideal Gas **Law**
- Hooke's **Law**
- ...

characterize first; prove later!



The Two Cultures



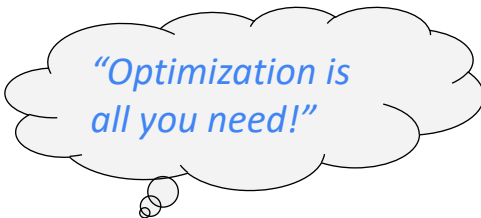
Opportunities for Theory

- Very young area:
Right definitions / abstractions
("do for Deep Learning what the Turing Machine did for computation")
- New conceptual frameworks
- "Reductions" / barriers
- New models of learning?
- ***Asking the right questions!***

Fun Open Problems (informal):

1. What problem is DL solving?
(*what is it "learning"?*)
2. Lower-bounds: Things we can't learn with DL?
(*want to solve, no existing soln, believe tractable*)
3. Minimal set of assumptions for DL?
(*"OWF" for DL theory*)
4. Axiomatic definition of DL, in terms of behaviors?

"Dedekind cuts" are to "axioms of the real line" as "deep neural networks" are to XXXX
5. What would learning look like in an Alien World, without: (Time, Space, Sample) constraints?



*“Optimization is
all you need!”*

Deep Bootstrap Framework

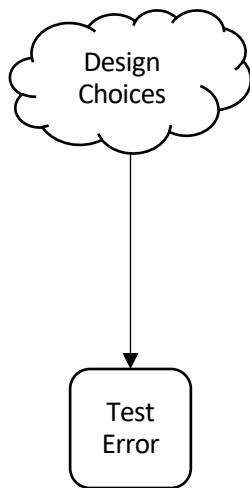
*Rethinking Generalization to
Understand Deep Learning*

Preetum Nakkiran
Harvard → UCSD

Behnam Neyshabur
Google

Hanie Sedghi
Google Brain

Appears in ICLR2021:
<https://arxiv.org/abs/2010.08127>



Setup: Supervised classification.

Distribution $(x, y) \sim D$

Given: iid samples from D

Do: SGD* on NN to minimize *train error*

Measure: *test error* $\Pr_{x,y \sim D} [f(x) \neq y]$

Main Idea: compare **Real World** vs. **Ideal World**

Fix distribution D , architecture \mathcal{F} , num samples n .

Then, for all steps $t \in \mathbb{N}$ define:

Real World(n, t)



Main Idea: compare **Real World** vs. **Ideal World**

Fix distribution D , architecture \mathcal{F} , num samples n .

Then, for all steps $t \in \mathbb{N}$ define:

Real World(n, t)

- Sample train set $S \sim D^n$
- Initialize architecture f_0 from \mathcal{F}
- For t steps:
 - Sample minibatch **from S**
 - Gradient step on minibatch
- Output f_t

Ideal World(t)



Main Idea: compare **Real World** vs. **Ideal World**

Fix distribution D , architecture \mathcal{F} , num samples n .

Then, for all steps $t \in \mathbb{N}$ define:

Real World(n, t)

- Sample train set $S \sim D^n$
- Initialize architecture f_0 from \mathcal{F}
- For t steps:
 - Sample minibatch **from S**
 - Gradient step on minibatch
- Output f_t

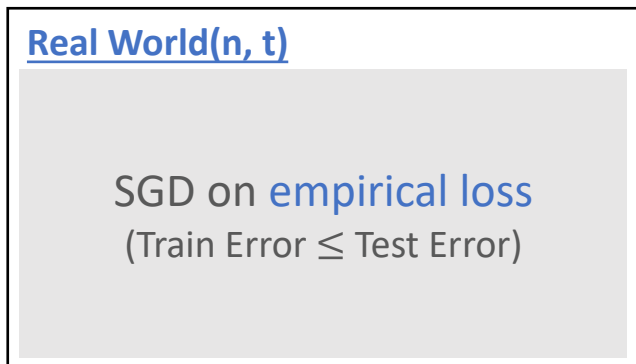
Ideal World(t)

- Initialize architecture f_0 from \mathcal{F}
- For t steps:
 - Sample minibatch **from D**
 - Gradient step on minibatch
- Output f_t^{iid}

Main Idea: compare **Real World** vs. **Ideal World**

Fix distribution D , architecture \mathcal{F} , num samples n .

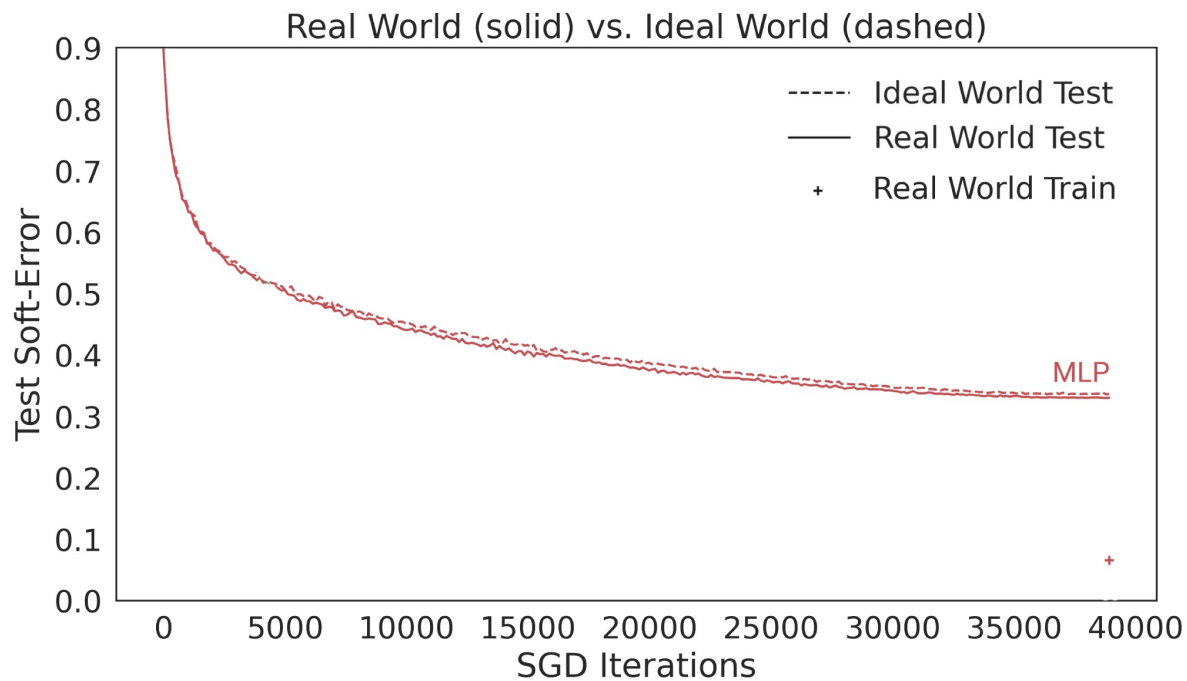
Then, for all steps $t \in \mathbb{N}$ define:



Test Error

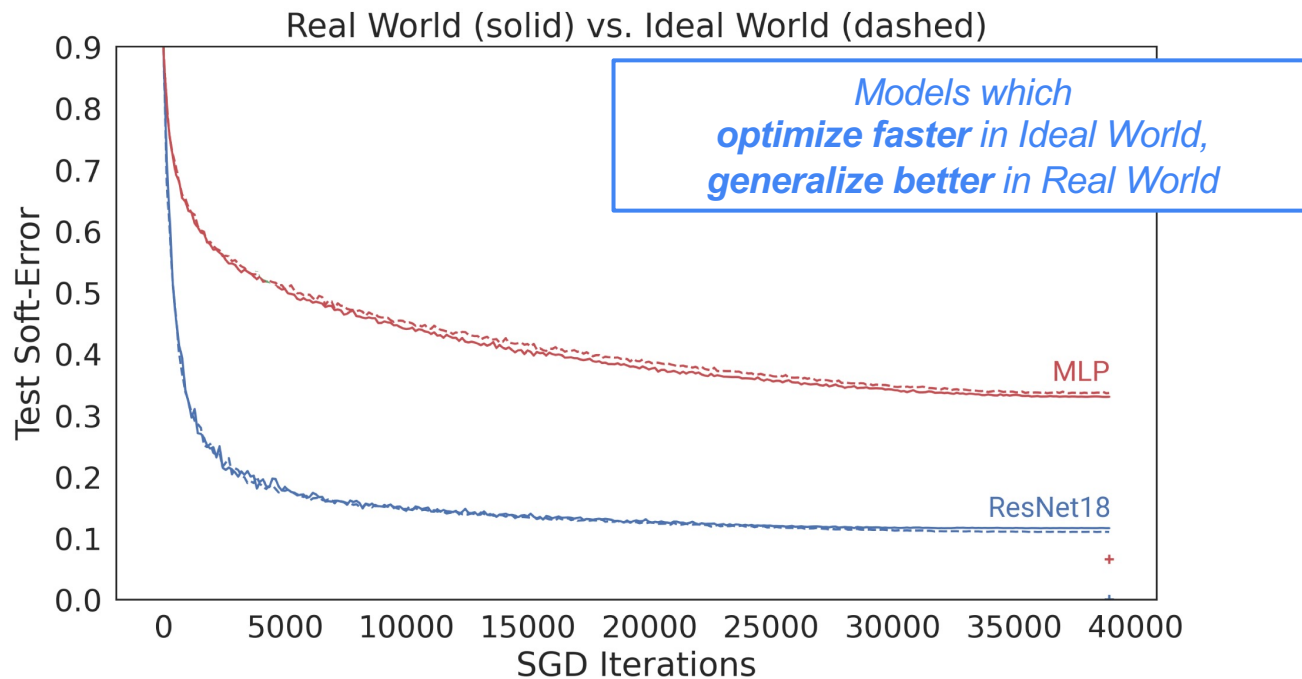
\approx





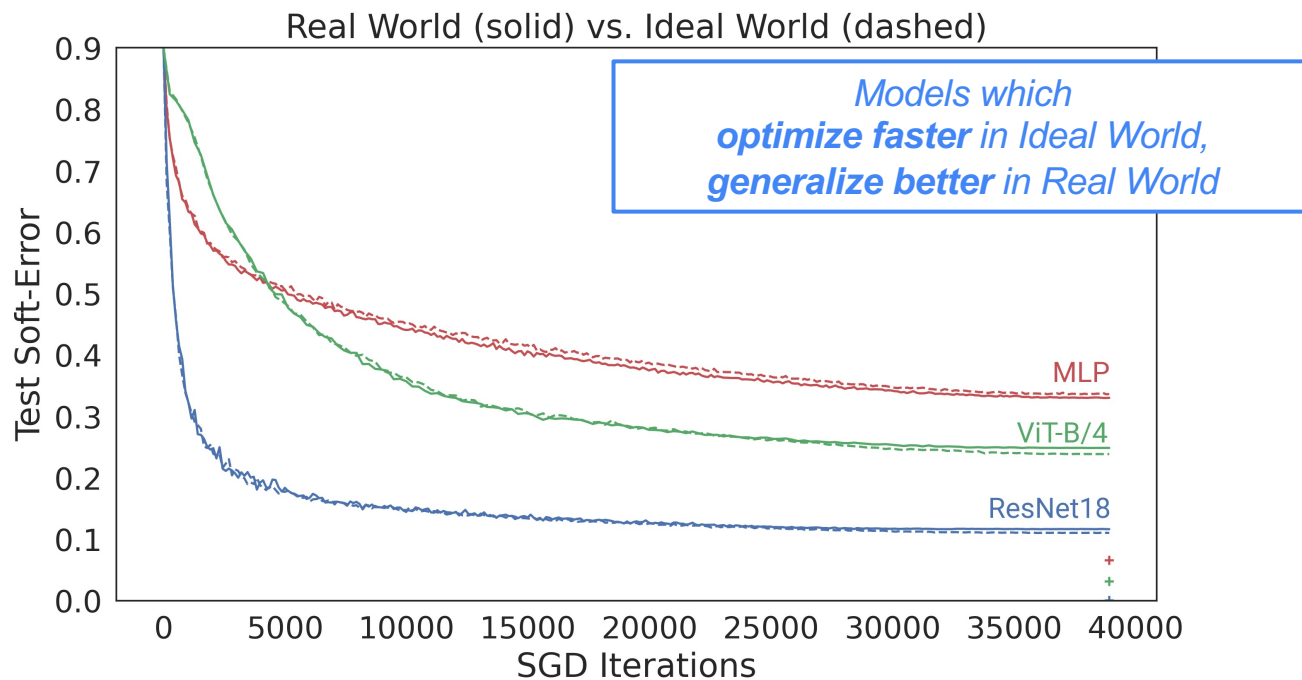
Real World: 50K samples, 100 epochs.

Ideal World: 5M samples, 1 epoch.



Real World: 50K samples, 100 epochs.

Ideal World: 5M samples, 1 epoch.



Real World: 50K samples, 100 epochs.

Ideal World: 5M samples, 1 epoch.

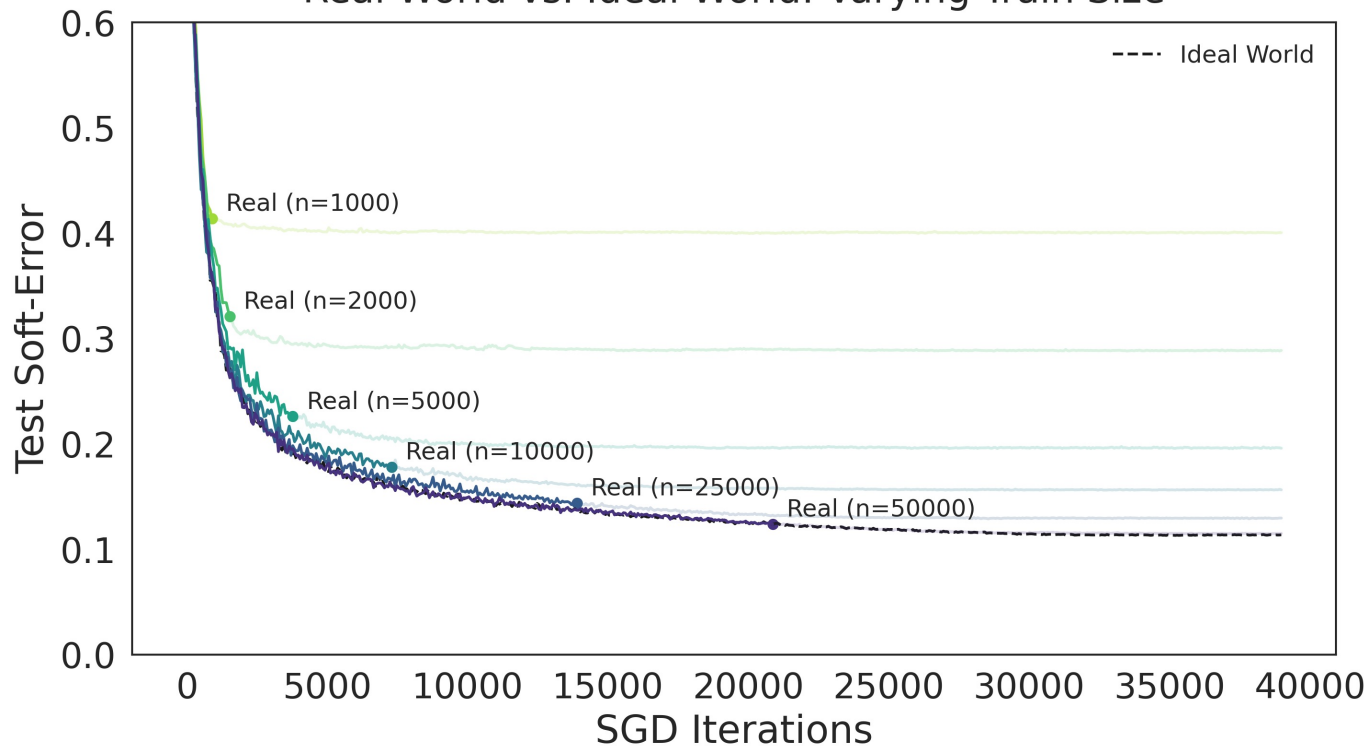
$T(n)$: “Stopping time”. Real World time to converge on n samples (< 1% train error)

Deep Bootstrap:

$$\forall t \leq T(n): \quad \text{RealWorld}(n, t) \approx_{\epsilon} \text{IdealWorld}(t)$$

*“SGD on deep nets behaves similarly
whether trained on **re-used samples** or **fresh samples**
...up until the Real World has converged”*

Real World vs. Ideal World: Varying Train Size



Deep Bootstrap:

$$\text{FinalError}(n) \approx_{\epsilon} \text{IdealWorld}(T(n))$$

$T(n)$: Time to converge on n samples

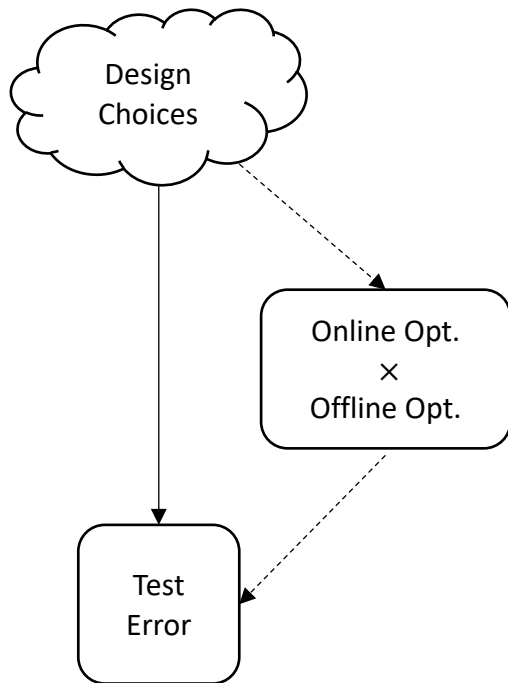
LHS: Generalization

RHS: Optimization

(Online optimization & Empirical Optimization)

Deep Bootstrap:

$$\text{FinalError}(n) \approx_{\epsilon} \text{IdealWorld}(T(n))$$

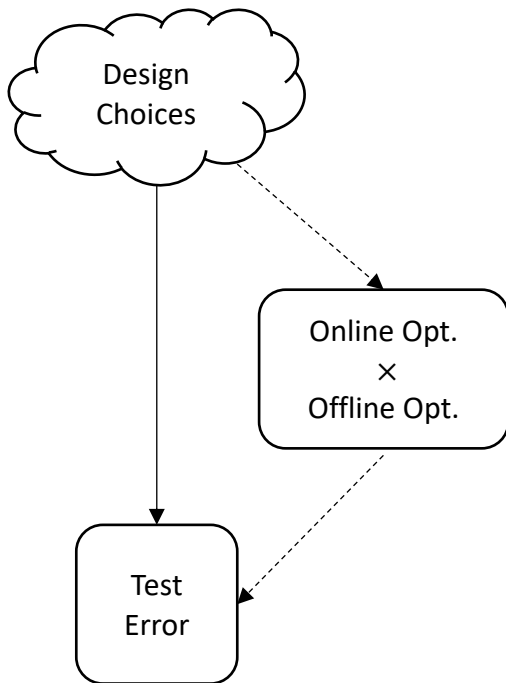


Empirically verified for varying:

- Architectures
- Model size
- Data size
- Optimizers (SGD/Adam/etc)
- Pretraining
- Data-augmentation
- Learning rate
- ...

Deep Bootstrap:

$$\text{FinalError}(n) \approx_{\epsilon} \text{IdealWorld}(T(n))$$



Good design choices:

1. **Optimize quickly** in online setting
(large models, skip-connections, pretraining,...)
2. **Don't optimize too** quickly on finite samples
(regularization, data-aug,...)

ERM decomposition:
$$\text{TestError}(f_t) = \text{TrainError}(f_t) + \underbrace{[\text{TestError}(f_t) - \text{TrainError}(f_t)]}_{\text{Generalization gap}}$$

Our decomposition:
$$\text{TestError}(f_t) = \underbrace{\text{TestError}(f_t^{\text{iid}})}_{\text{A: Online Learning}} + \underbrace{[\text{TestError}(f_t) - \text{TestError}(f_t^{\text{iid}})]}_{\substack{\text{B: Bootstrap error} \\ \varepsilon(n, \mathcal{D}, \mathcal{F}, t)}}$$

Main Claim: *Bootstrap error $\varepsilon(n, \mathcal{D}, \mathcal{F}, t)$ is small for realistic $(n, \mathcal{D}, \mathcal{F})$, and all $t \leq T(n)$*

Where “stopping time” $T(n) := \text{time when Real World reaches TrainError} \leq 1\%$.

$L(n)$: Test error on n samples (Real World, trained to convergence)

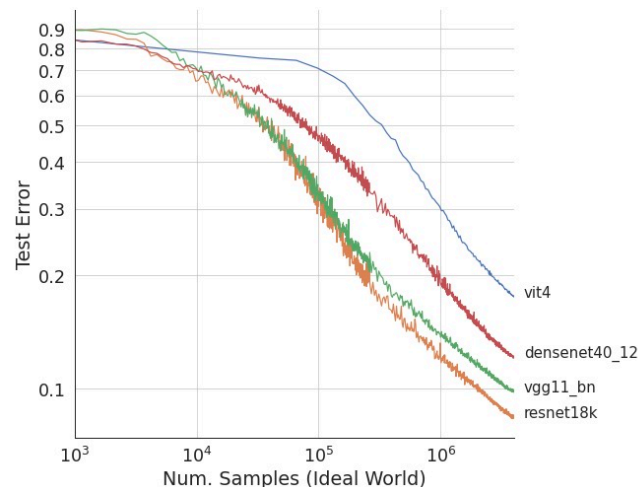
$T(n)$: Time to converge on n samples (Real World SGD steps)

$\tilde{L}(t)$: Test error after t online SGD steps (Ideal World)

Deep Bootstrap: $L(n) \approx \tilde{L}(T(n))$

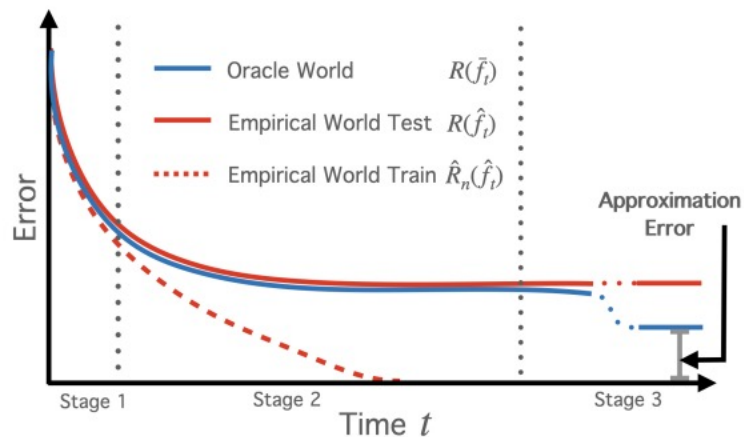
Assuming $T(n) \sim \Theta(n)$,
(Learning curve exponent) \approx (Online optimization exponent)

NB: Scaling exponents
multiply



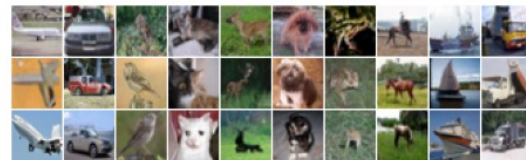
Recent proof in Kernel setting [authors redacted for anon] :

THE THREE STAGES OF LEARNING DYNAMICS IN HIGH-DIMENSIONAL KERNEL METHODS

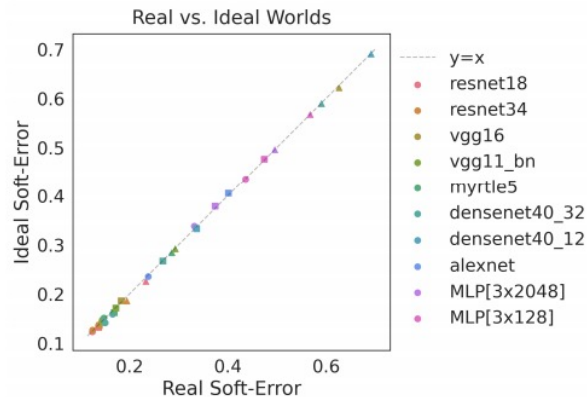


Validation: Summary of Experiments

- **CIFAR-5m**: 5-million synthetic samples from a generative model trained on CIFAR-10
- **ImageNet-DogBird**: 155K images by collapsing ImageNet categories. Binary task.
- **Varying settings**: {archs, opt, LR,...}
convnets, ResNets, MLPs, Image-GPT, Vision-Transformer



Samples from CIFAR-5m

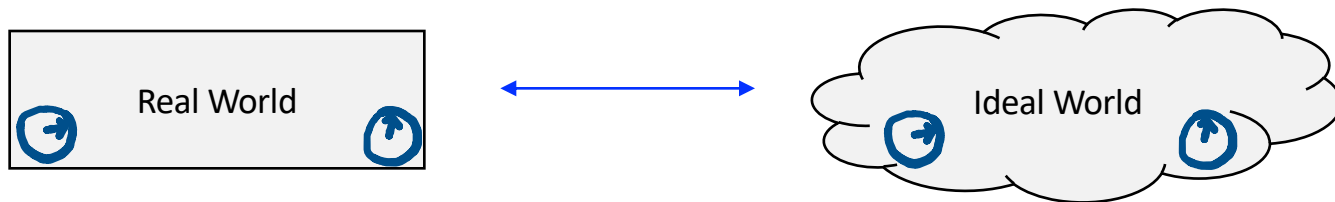


(a) Standard architectures.

Figure 2: **Real vs Ideal World: CIFAR-5m**. SGD w 0.1 (●), 0.01 (■), 0.001 (▲). (b): Random architecture

Implications:

Deep Learning through the Bootstrap Lens

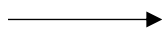


Alternate Perspectives

Generalization Perspective:

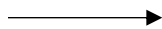
“ConvNets *generalize better* than MLPs”

“Pretraining helps *generalization*”



Optimization Perspective:

“ConvNets *optimize faster* than MLPs”



“Pretraining helps *optimization*”
(a la *preconditioning*)

Effect of Pretraining

Pretrained models generalize better (Real)
“because” they optimize faster (Ideal)

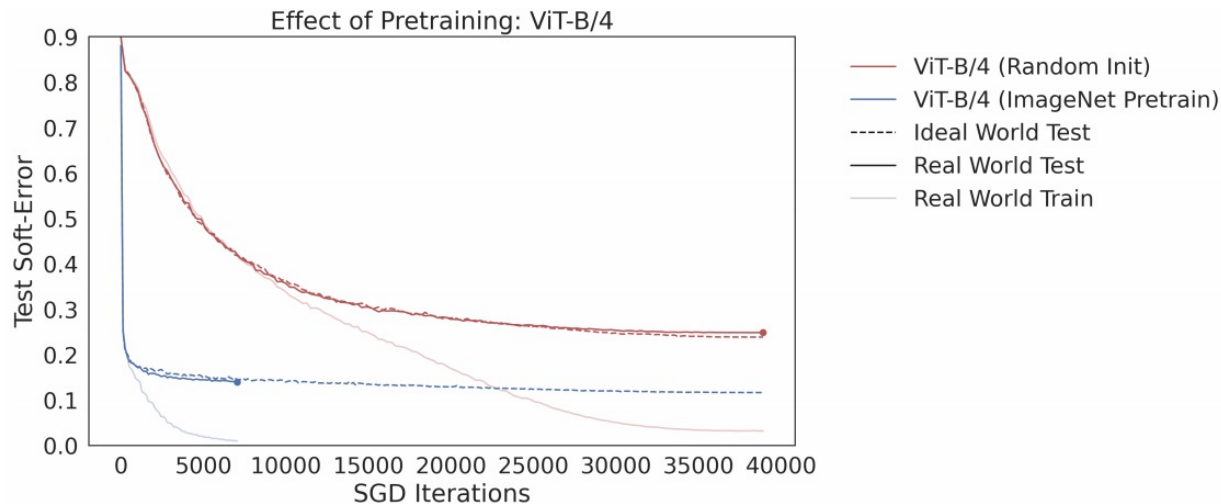


Figure 13: Real vs. Ideal Worlds for Vision Transformer on CIFAR-5m, with and w/o pretraining.

Effect of Data Aug

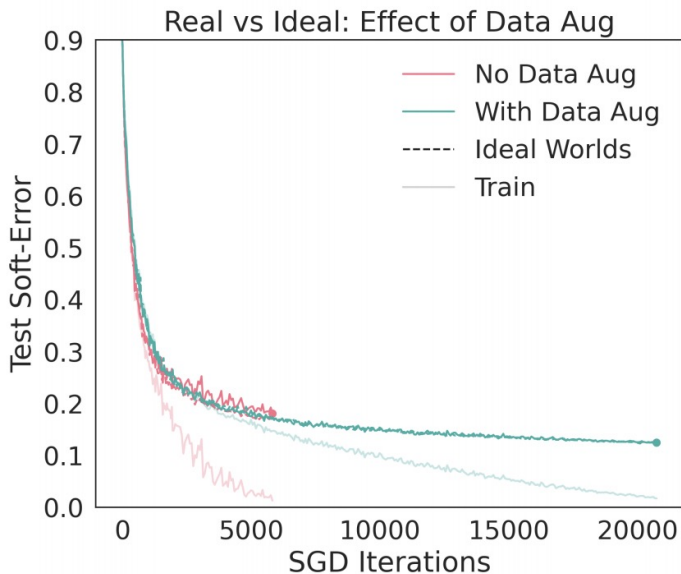
Data-aug in the Ideal World =
Augment each sample once

Two potential effects:

1. Ideal World Optimization Speed
- 2. Real World Convergence Speed**

Good data-augs:

1. Don't hurt learning in Ideal World
2. Decelerate optimization in Real World (train for longer)



Implicit Bias \rightarrow Explicit Optimization

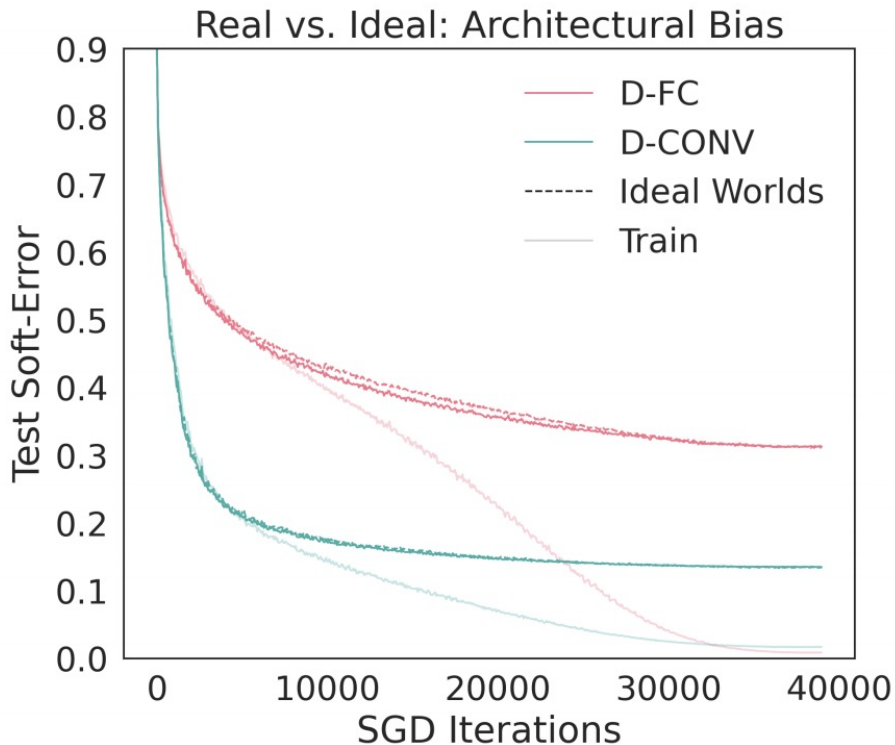
Two archs from [Neyshabur 2020]:

D-CONV (convnet) \subset D-FC (mlp)

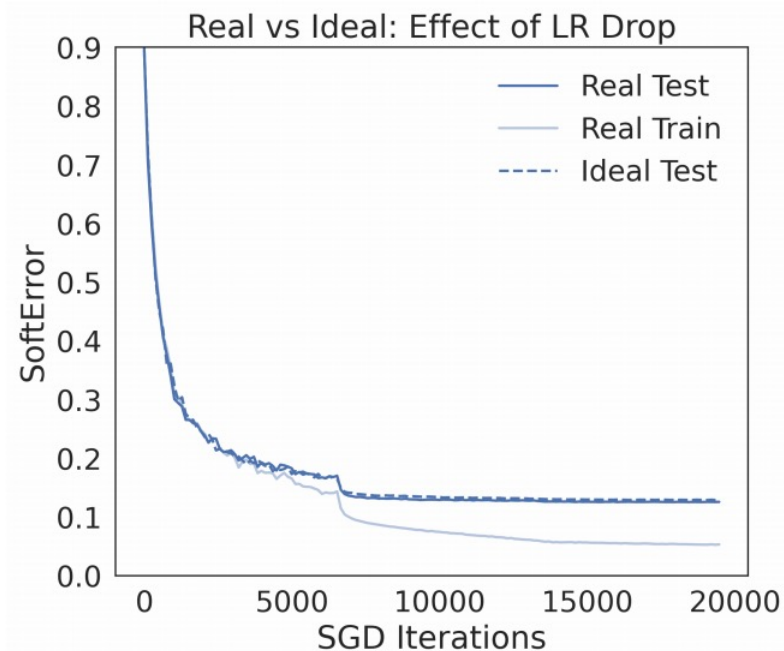
Both train to 0 Train Error, but
convnet generalizes better.

Traditionally: due to “implicit bias”
of SGD on the convnet.

Our view: due to better
optimization in the Ideal World



Effect of Learning Rate



Random Labels (Thought Experiment)

“Understanding deep learning requires rethinking generalization”
[Zhang et al. 2016]

- Train on randomly-labeled inputs.
- 0% train error, 90%/trivial test error.

Here:

- Real World: Test Error \gg Train Error
- Real World Test \approx Ideal World Test

Details

Choice of Metric Matters!

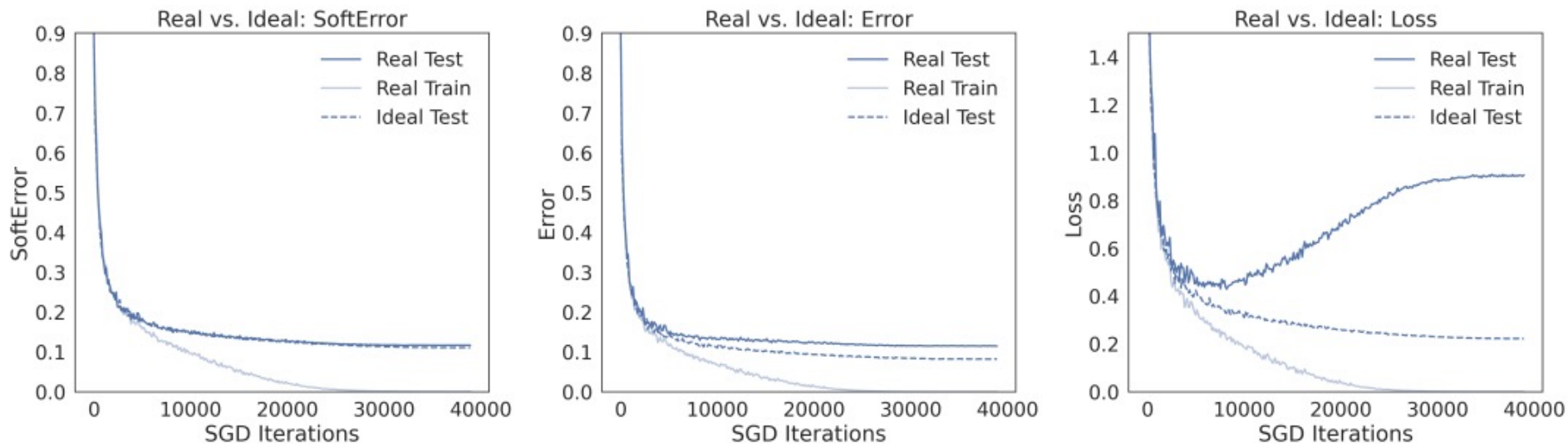


Figure 6: SoftError vs. Error vs. Loss: ResNet-18.

Why Soft-Error?

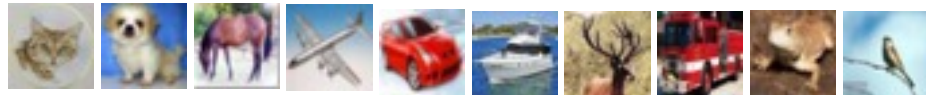
Want: $F(\text{RealWorld}) - F(\text{IdealWorld}) \rightarrow 0$ as $(\text{model}, \text{data}) \rightarrow \infty$.

This doesn't happen for $F = \text{TestError}$, if:

(1) Bayes risk $\neq 0$.

(2) Take overparameterized limit: $(\text{model}, \text{data}) \rightarrow \infty$, $\text{model} \gg \text{data}$

Digression



Setup: Take CIFAR-10 train set, apply label noise: cats \rightarrow dog w.p. 30%.
Train a ResNet to 0 train error. What happens on test samples?

Result: Cats \rightarrow dog w.p. $\sim 30\%$ on test set! (other classes unaffected)

Surprising because:

- Not close to Bayes-optimal classifier
- Ideal World won't do this
- (unless we consider randomized softmax instead of argmax)

“Distributional Generalization”
[Nakkiran, Bansal 2020]

When Bootstrap Fails

1. Near Double-Descent region (Real World has pathology)
 - Or any setting with non-monotonic Soft-Error
2. Very small number of samples
3. Potentially: weird distributions / architectures / optimizers?
(seems to work in any setting with “real data”, regardless of model)

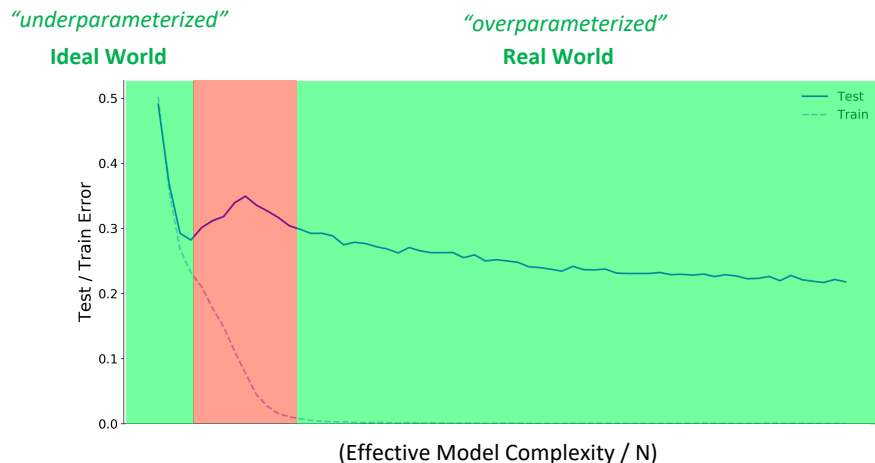
Conclusions

Assuming bootstrap claim: Reduces *generalization* to *optimization*.

Hope: Refocus attention on online optimization aspects of deep learning
(some modern models actually in “Ideal World”)

Connects *overparametrized* and *underparameterized* regimes:

Models which fit their train sets “behave like” models trained on infinite data



A Practical Mystery

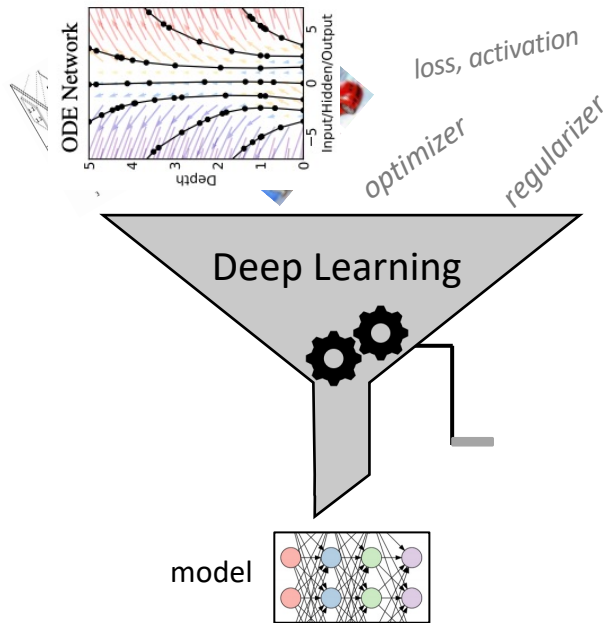
Two regimes in practice:

1. Effectively infinite data (e.g. train on internet, 1B+ samples)
want architectures which optimize quickly
2. Small finite data (e.g. 50K samples)
want architectures which generalize well

Mystery: Why do we use the same architectures in both regimes?

Deep Bootstrap: Not a coincidence...

Conclusions



Many *diverse* choices in deep learning “work” (generalize)

Want theory of generalization that applies to all.

Deep Bootstrap:

“Any choice that works for online optimization will work for offline generalization.”

Speculation: Holds much more generically than deep learning...

Open Problems

Mysteries of Online Learning:

- (essentially all mysteries in ML remain)
- Why do certain architectures optimize faster on certain distributions?
- How to characterize interaction between: {architecture, optimizer, task}?
- Why does pretraining act as a preconditioner?
- Out-of-distribution robustness
- Why do we “learn representations”?
- ...

Beyond Test Error: Similarities between Real & Ideal World

- Similar behavior under distribution shift?
- Similar representations?
- Similar transfer performance

Ideal World as a Testbed:

- To compare optimizers
- Calibration/uncertainty/ensembling

Open Problems

Limits of Deep Bootstrap:

- Understanding when it applies/fails (theory & practice)
- Ex: Holds for even “simple” models on real data (logistic regression on FMNIST)
- Fails for certain contrived settings
- Does it hold more generically?
 - “Exotic” optimizers (ODEs, gradient-free, higher-order, Hebbian, etc)
 - Other settings: generative models, etc.
 - Other behaviors: ensembling over random init \sim ensembling over random train sets.

“Does generalization always reduce to optimization, in natural settings”?

Thanks!

preetum@ucsd.edu

Extras

Why Soft-Error?

Want: $F(\text{RealWorld}) - F(\text{IdealWorld}) \rightarrow 0$ as $(\text{model}, \text{data}) \rightarrow \infty$.

This doesn't happen for $F = \text{TestError}$, if Bayes risk $\neq 0$.

Suppose Real World takes overparameterized limit: $(\text{model}, \text{data}) \rightarrow \infty$

Ideal World converges to *Bayes-optimal classifier*:

$$\lim_{T \rightarrow \infty, S \rightarrow \infty} \tilde{f}_{S,T}(x) = \operatorname{argmax}_y p(y | x)$$

Real World converges to *optimal sampler*:

$$\lim_{N \rightarrow \infty, S \rightarrow \infty} f_{S,N}(x) \sim p(y|x)$$

“Distributional Generalization”
[Nakkiran, Bansal 2020]

What about Non-Deep Learning?

- Not true for well-specified linear regression!
- Can be contrived to be true for **misspecified** regression

$$x \sim \mathcal{N}(0, V)$$

$$y := \sigma(\langle \beta^*, x \rangle)$$

$$f_\beta(x) := \langle \beta, x \rangle$$

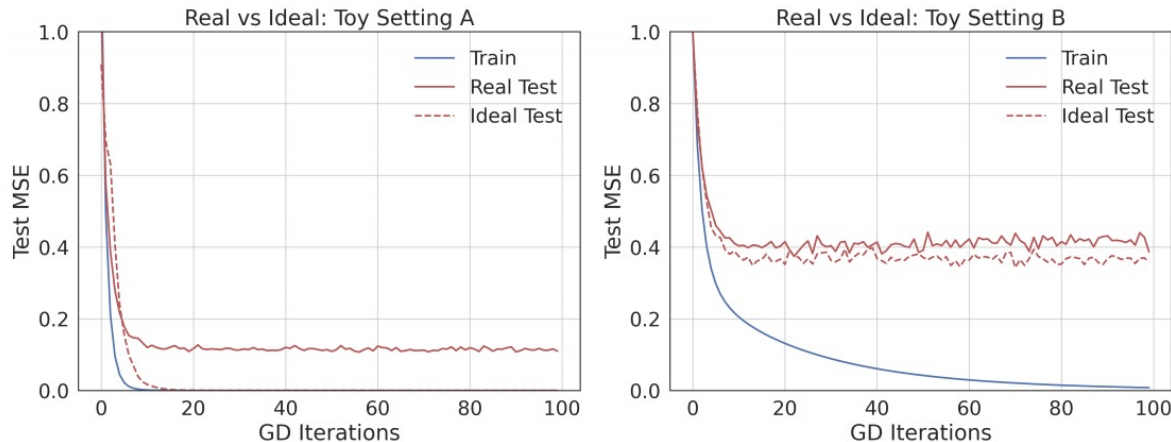


Figure 7: **Toy Example.** Examples of settings with large and small bootstrap error.

- **Setting A.** Linear activation $\sigma(x) = x$. With $n = 20$ train samples.
- **Setting B.** Sign activation $\sigma(x) = \text{sgn}(x)$. With $n = 100$ train samples.