



“Optimization is  
all you need!”

# Deep Bootstrap Framework

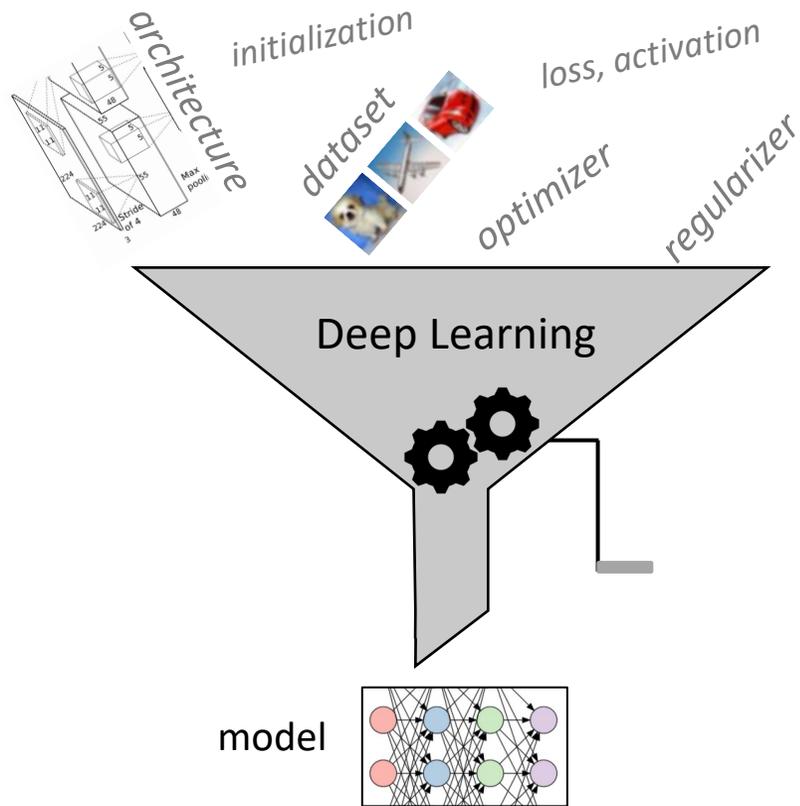
*Rethinking Generalization to  
Understand Deep Learning*

**Preetum Nakkiran**  
Harvard → UCSD

Behnam Neyshabur  
Google

Hanie Sedghi  
Google Brain

# Motivation

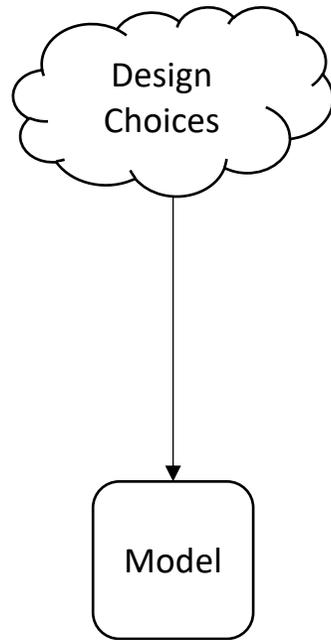


Deep Learning:  
accepts inputs (design choices),  
produces output (model)

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

- Advances in DL are unpredictable.
- Every advance = new choice of inputs
- Surprised by which choices work!

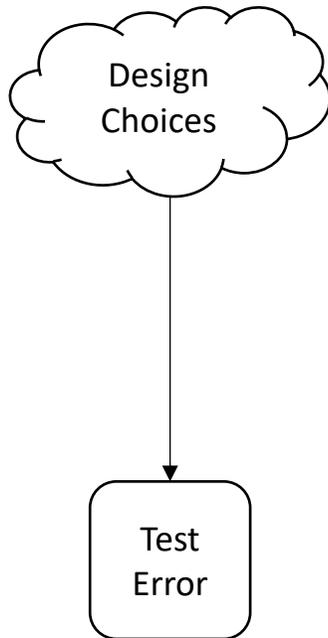
# Motivation



Deep Learning:  
accepts inputs (design choices),  
produces output (model)

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

- Advances in DL are unpredictable.
- Every advance = new choice of inputs
- Surprised by which choices work!



**Setup:** Supervised classification.

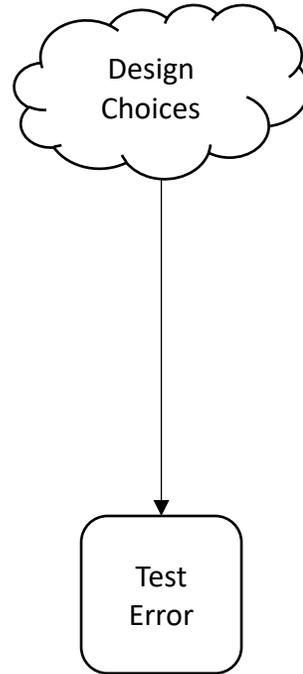
Distribution  $(x, y) \sim D$

Given: iid samples from  $D$

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

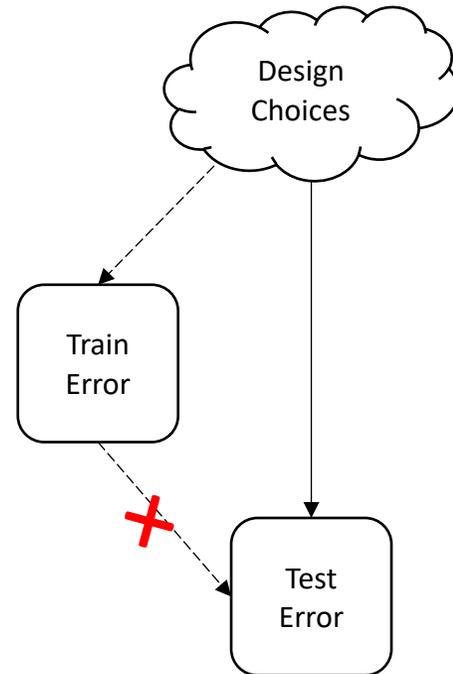
Measure: *test error*  $\Pr_{x,y \sim D} [f(x) \neq y]$

# Generalization Frameworks



# Generalization Frameworks

*“Uniform  
Convergence  
Framework”*



Any “big enough”  
network can have  
Train Error  $\approx 0$

[Zhang et al. 2016]

**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  $f_t$

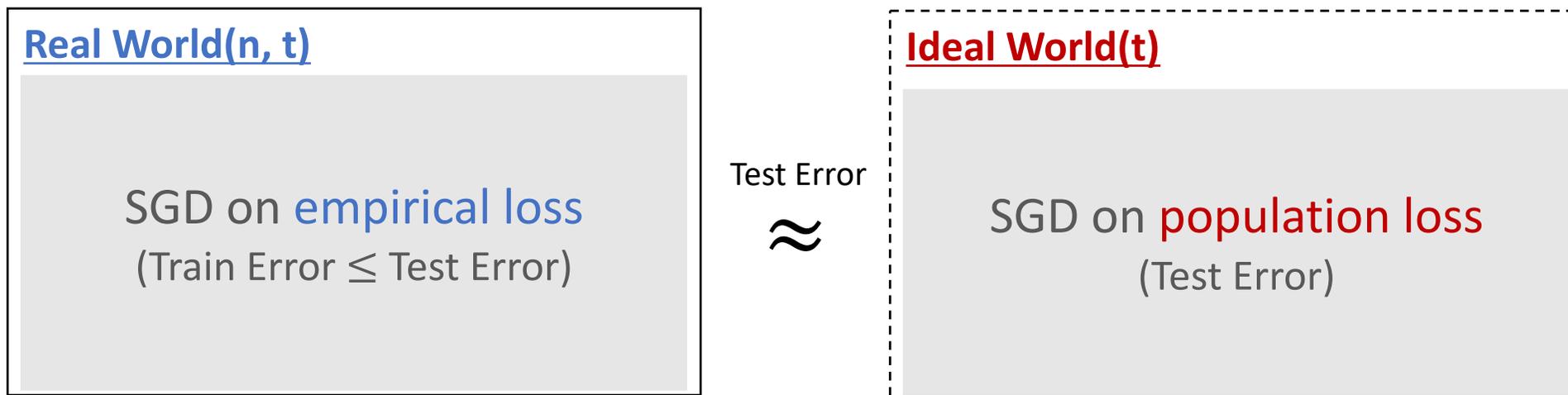
### Ideal World(t)

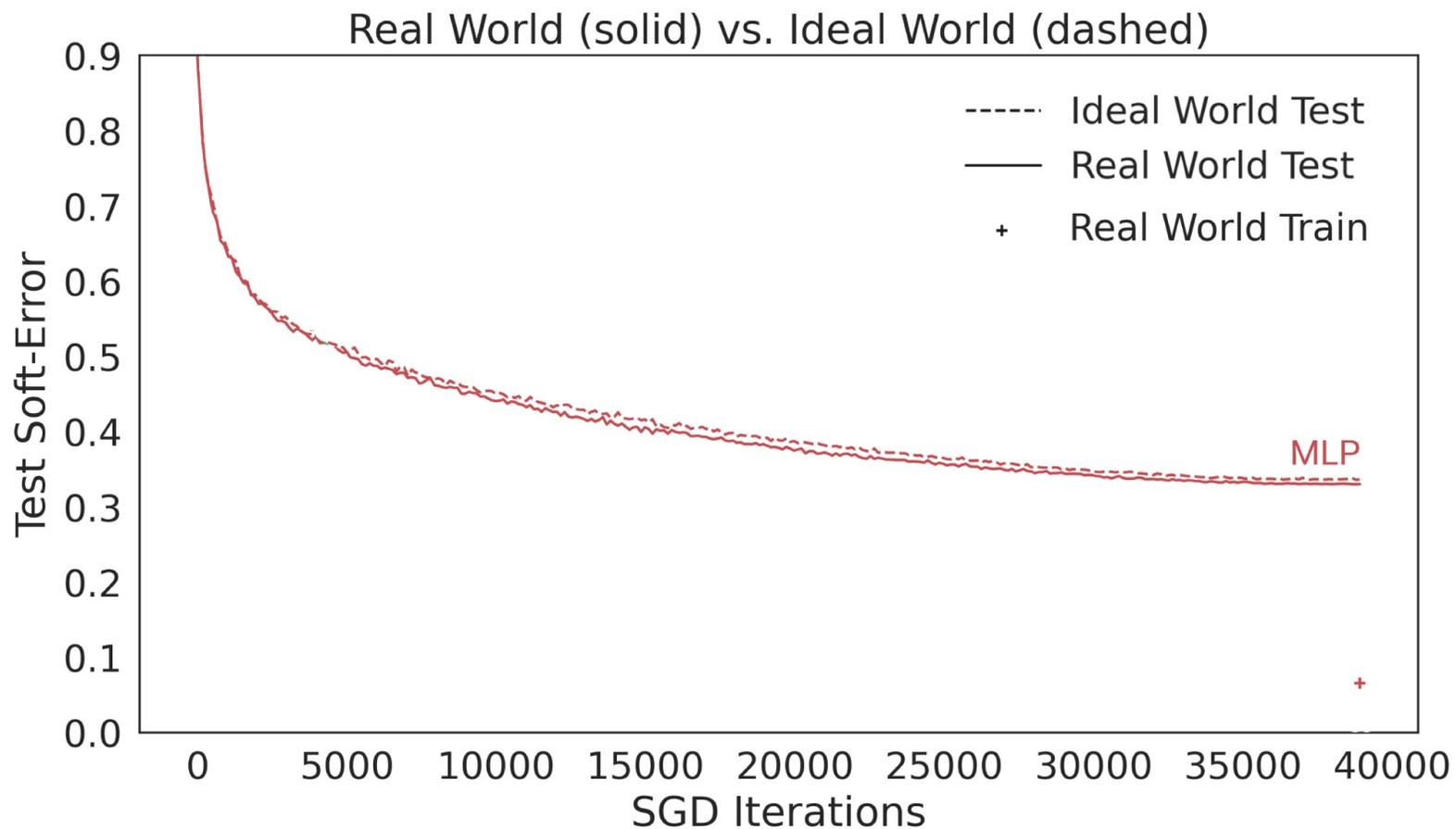
- Initialize architecture  $f_0$  from  $\mathcal{F}$
- For  $t$  steps:
  - Sample minibatch from  $D$
  - Gradient step on minibatch
- Output  $f_t^{\text{iid}}$

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

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

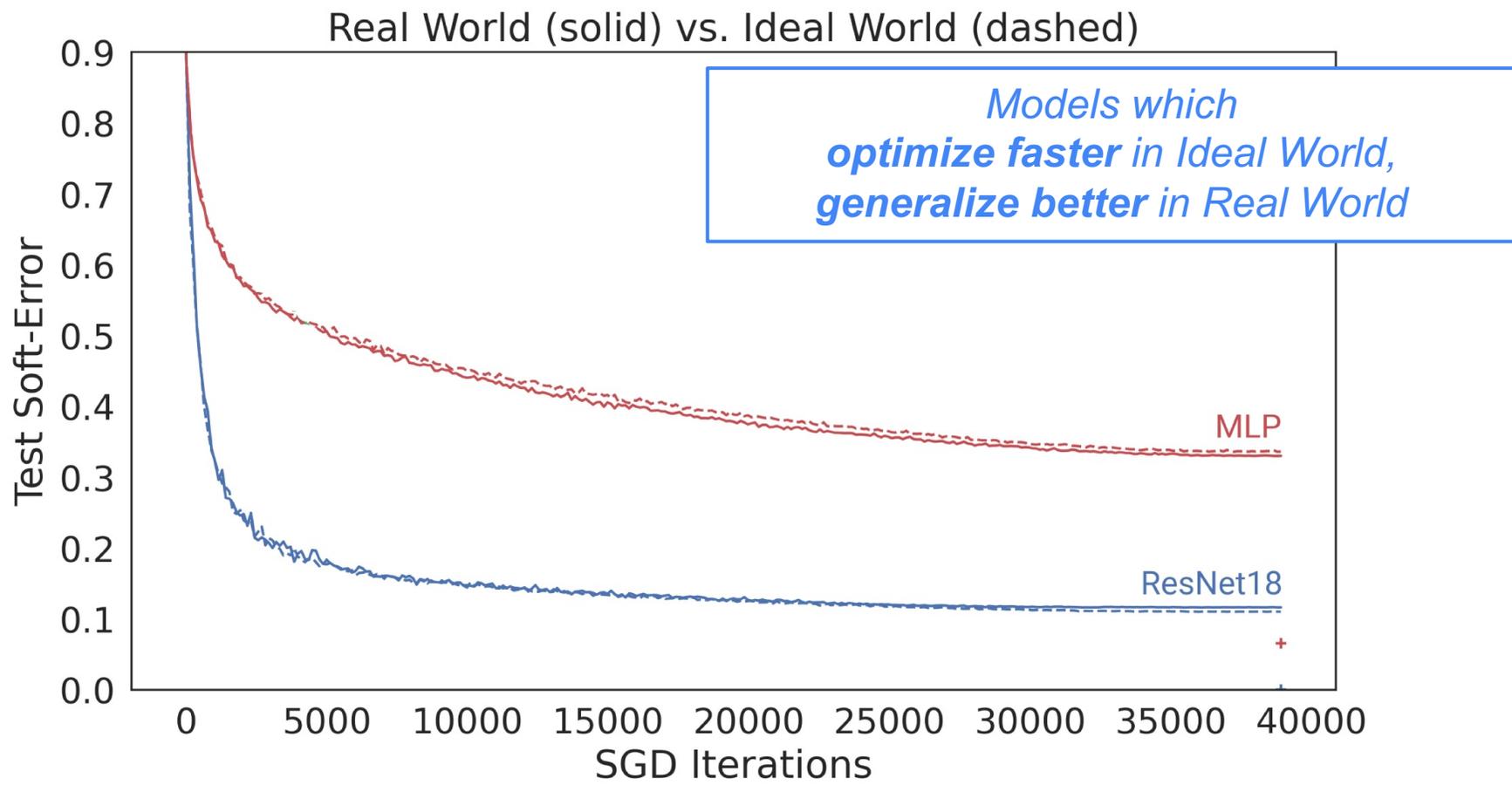
Then, for all steps  $t \in \mathbb{N}$  define:





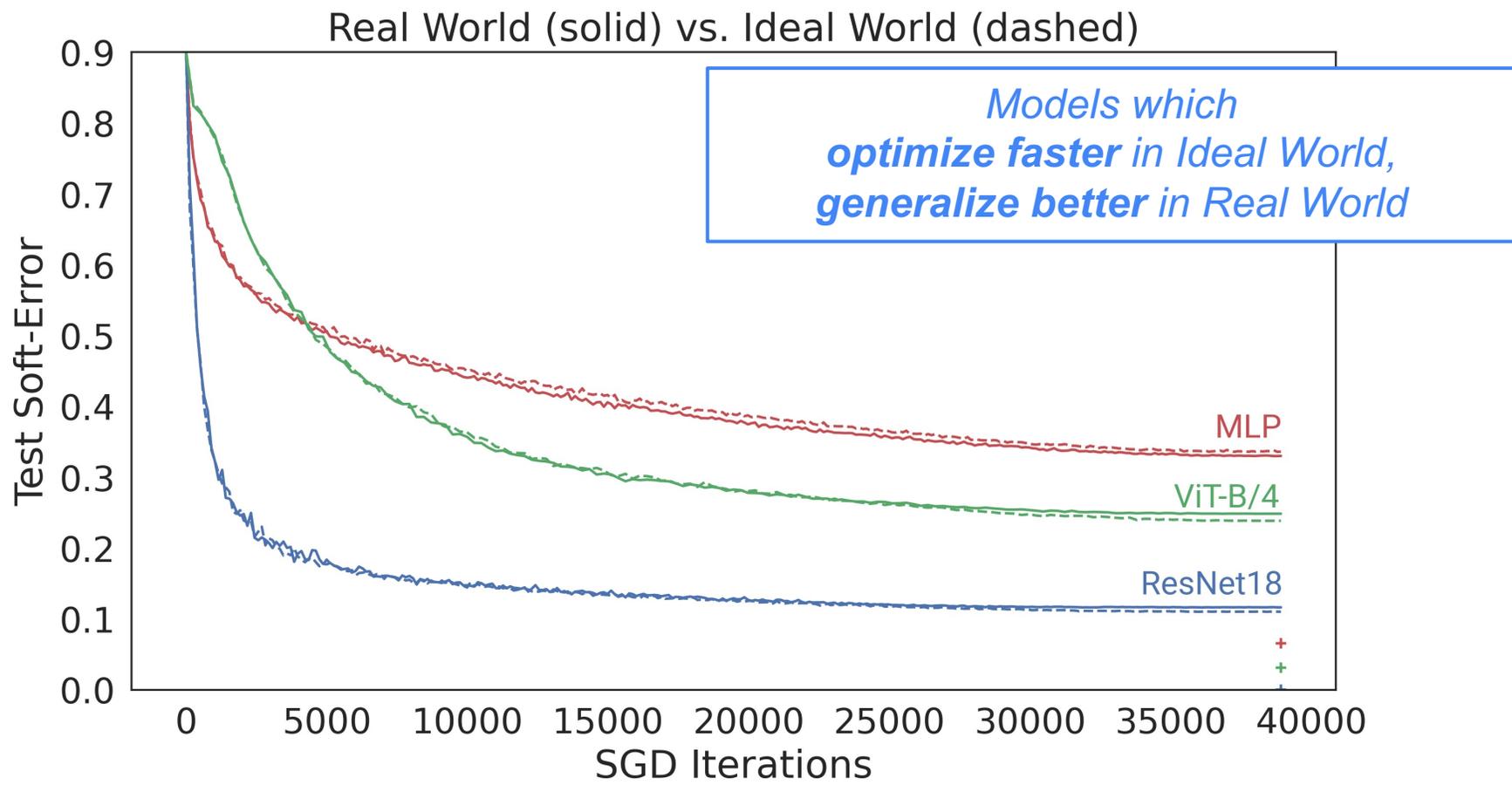
**Real World:** 50K samples, 100 epochs.

**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.

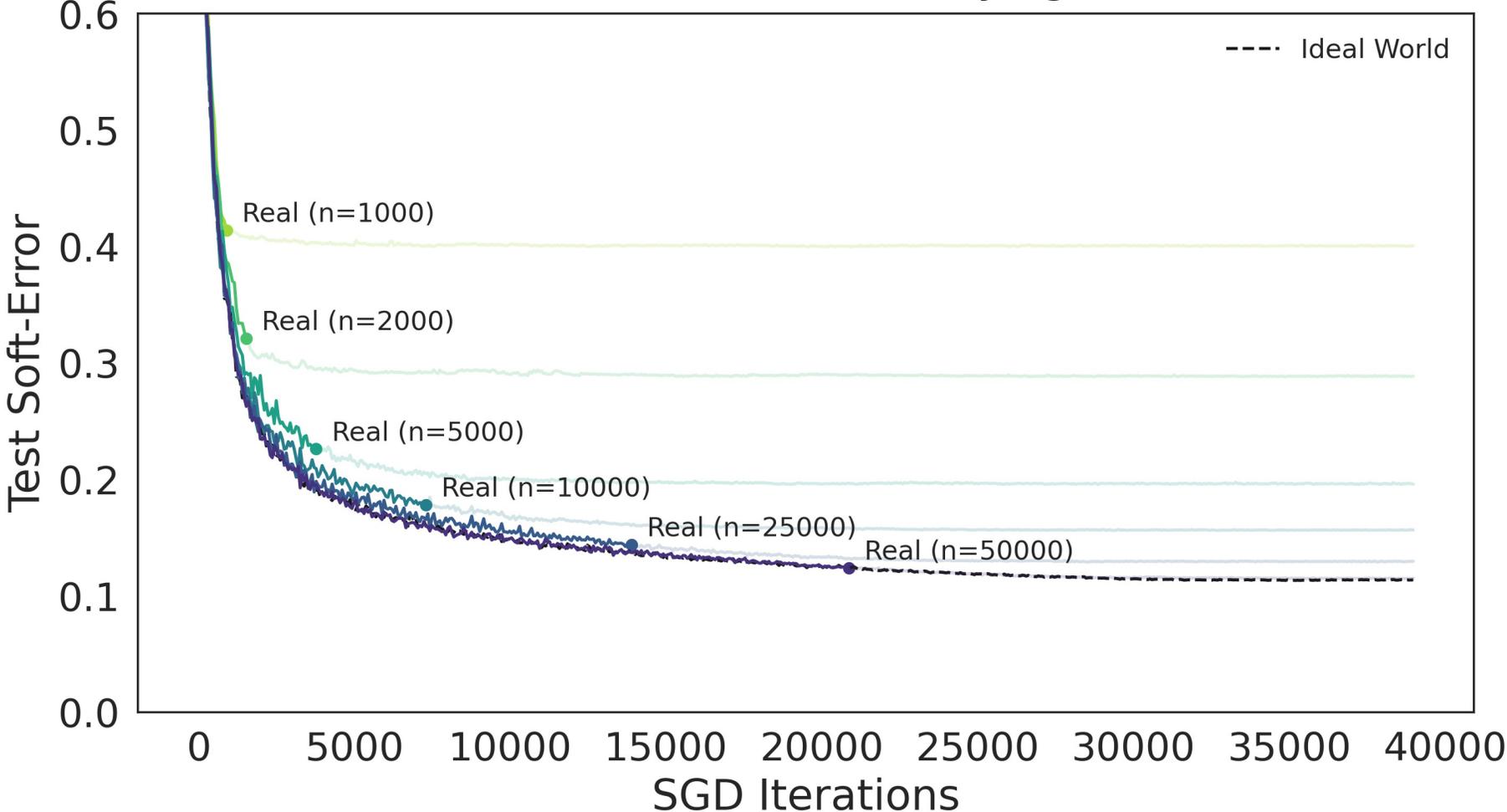
$T(n)$ : “Stopping time”. Real World time to converge on  $n$  samples (< 1% train error)

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

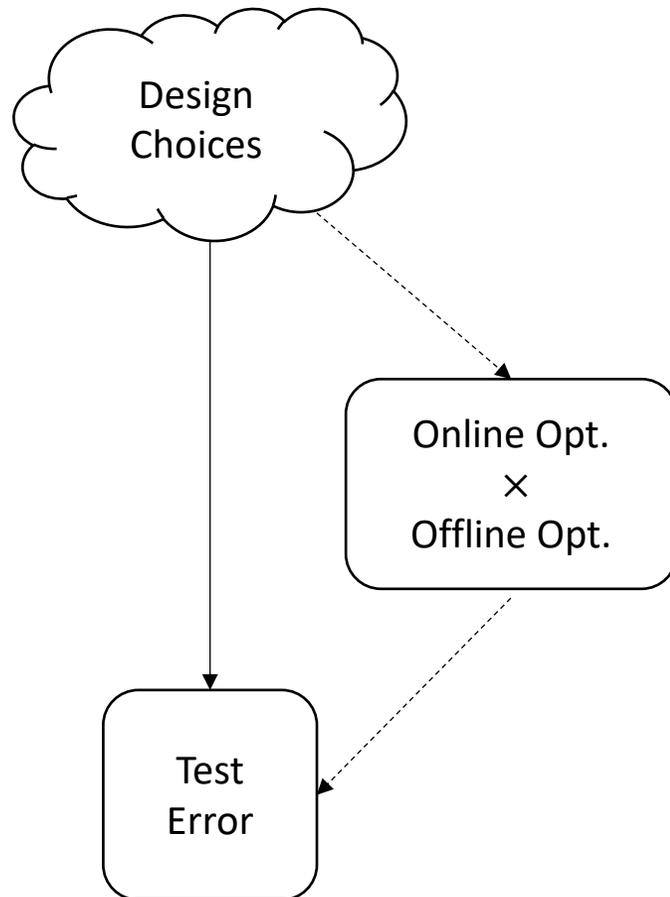
**LHS: Generalization**

**RHS: Optimization**

(Online optimization & Empirical Optimization)

## Deep Bootstrap:

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

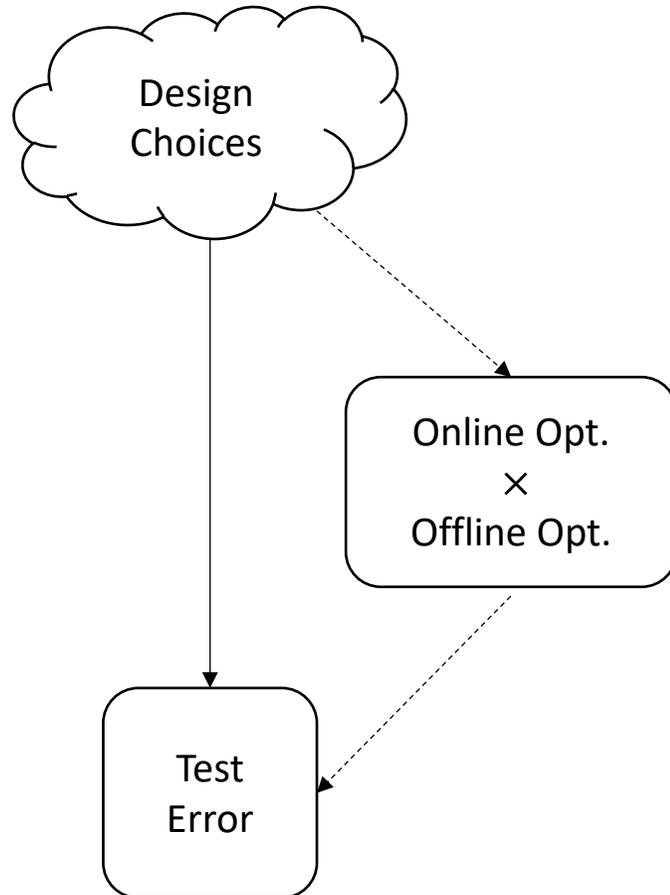


### Empirically verified for varying:

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

**Main Claim:** *Bootstrap error  $\epsilon(n, \mathcal{D}, \mathcal{F}, t)$  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)

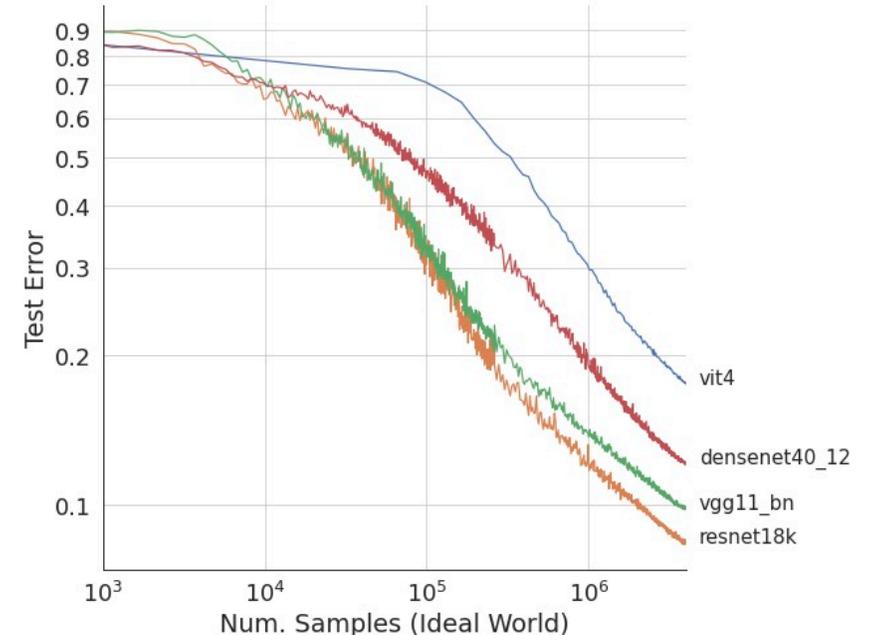
$\tilde{L}(t)$ : Test error after  $t$  online SGD steps (Ideal World)

Deep Bootstrap:

$$L(n) \approx \tilde{L}(T(n))$$

NB: Scaling exponents multiply

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

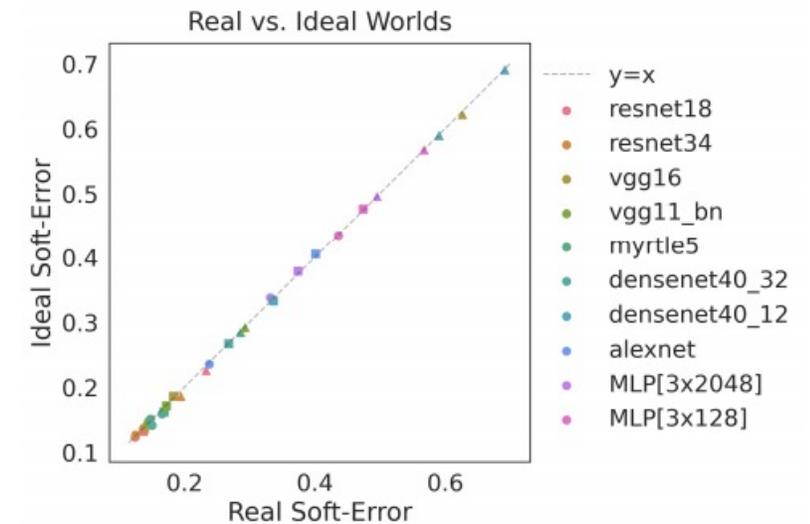


# Validation: Summary of Experiments

- **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



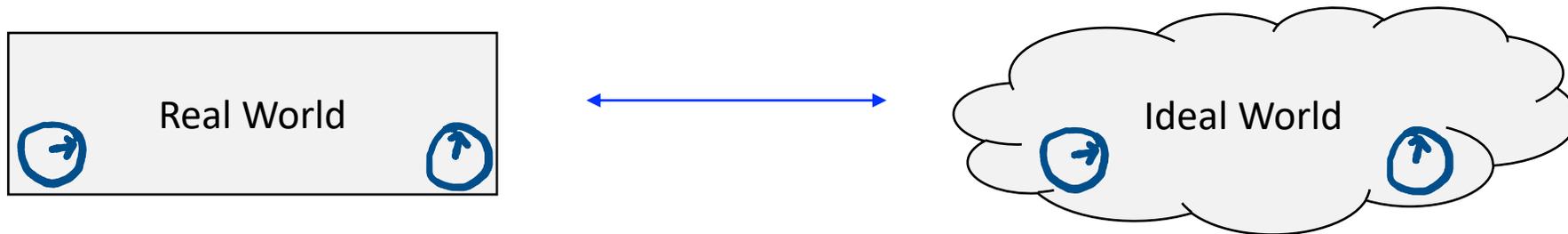
Samples from CIFAR-5m



(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



# Effect of Pretraining

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

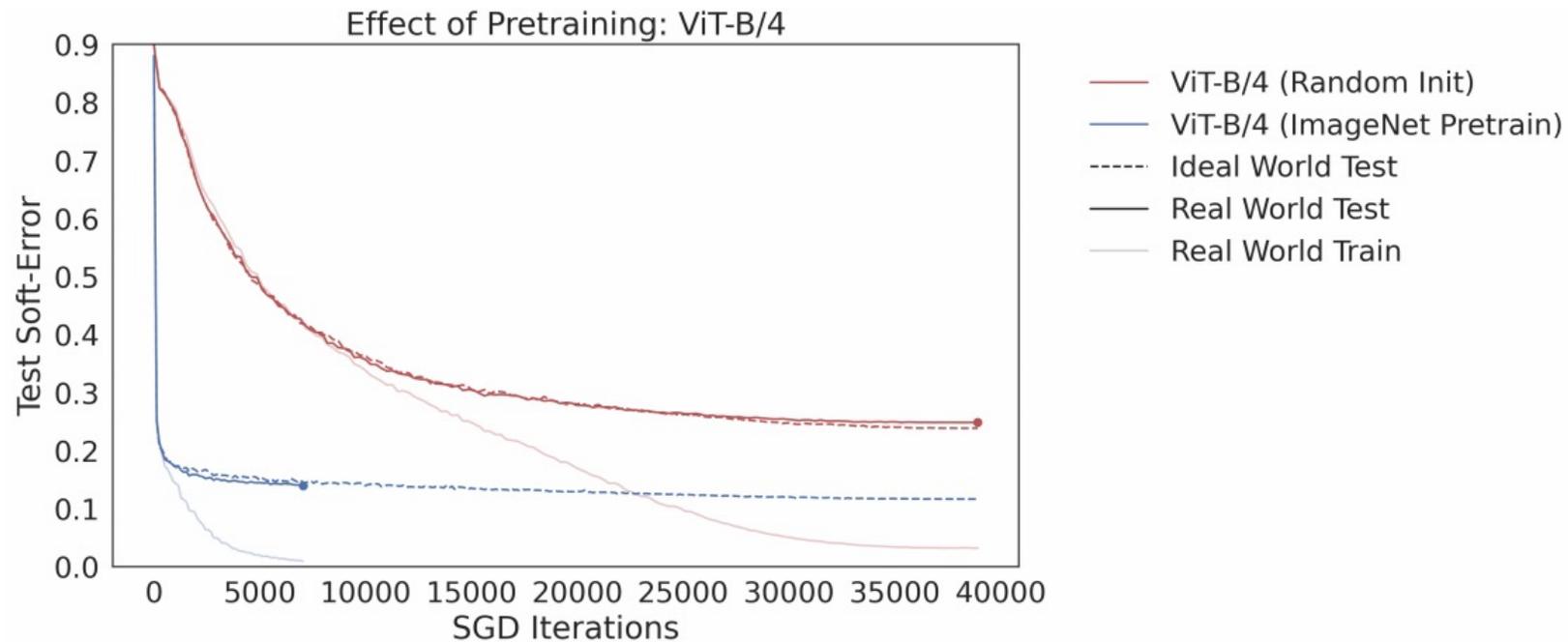


Figure 13: Real vs. Ideal Worlds for Vision Transformer on CIFAR-5m, with and w/o pretraining.

# Effect of Data Aug

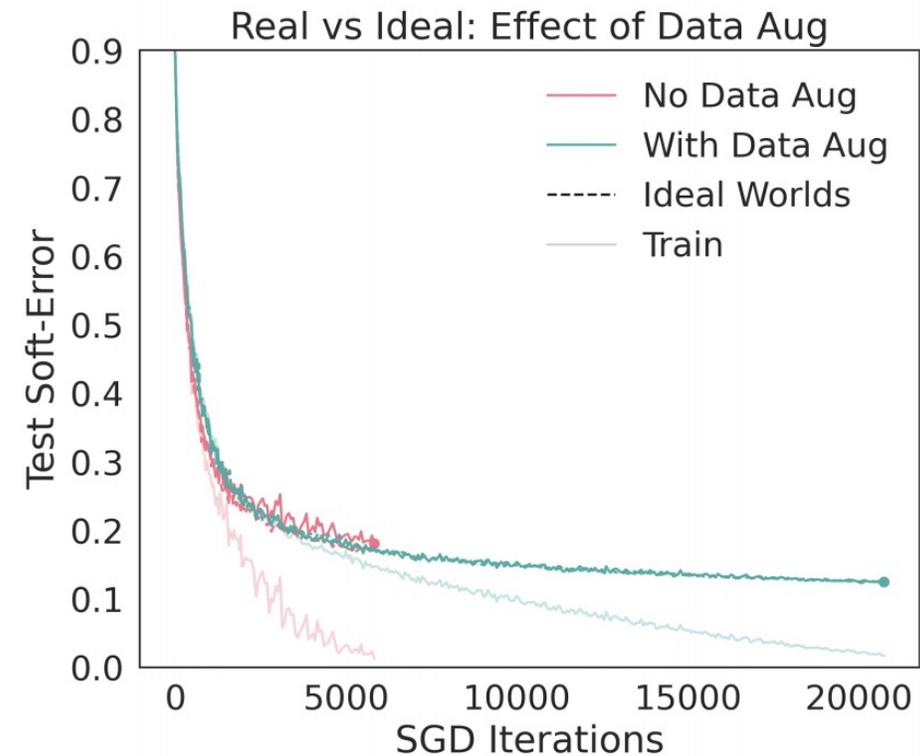
Data-aug in the Ideal World =  
Augment each sample once

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)



see "Affinity and Diversity"  
of [Gontijo-Lopes et al.]

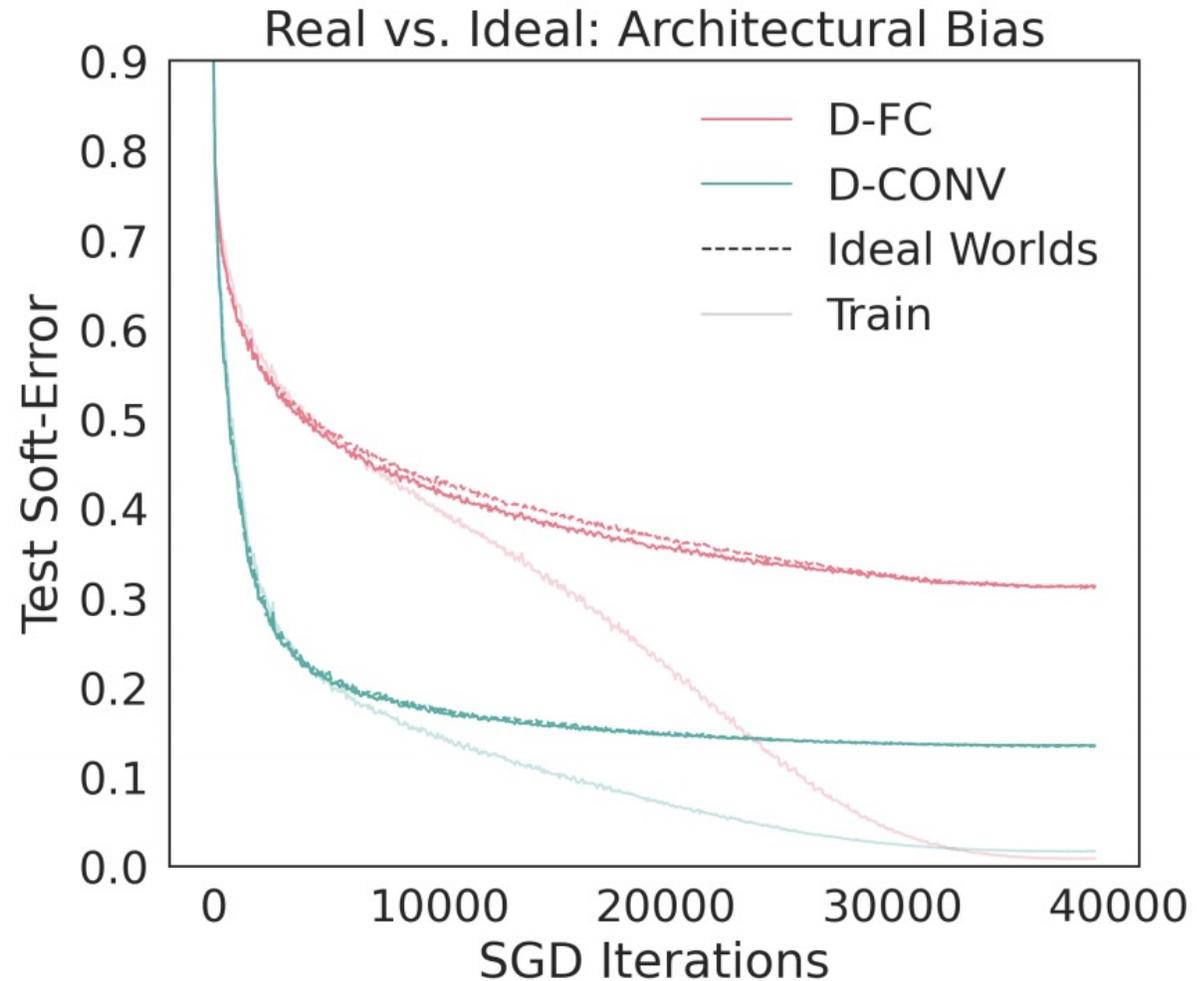
# Implicit Bias $\rightarrow$ Explicit Optimization

Two archs from [Neyshabur 2020]:  
D-CONV (convnet)  $\subset$  D-FC (mlp)

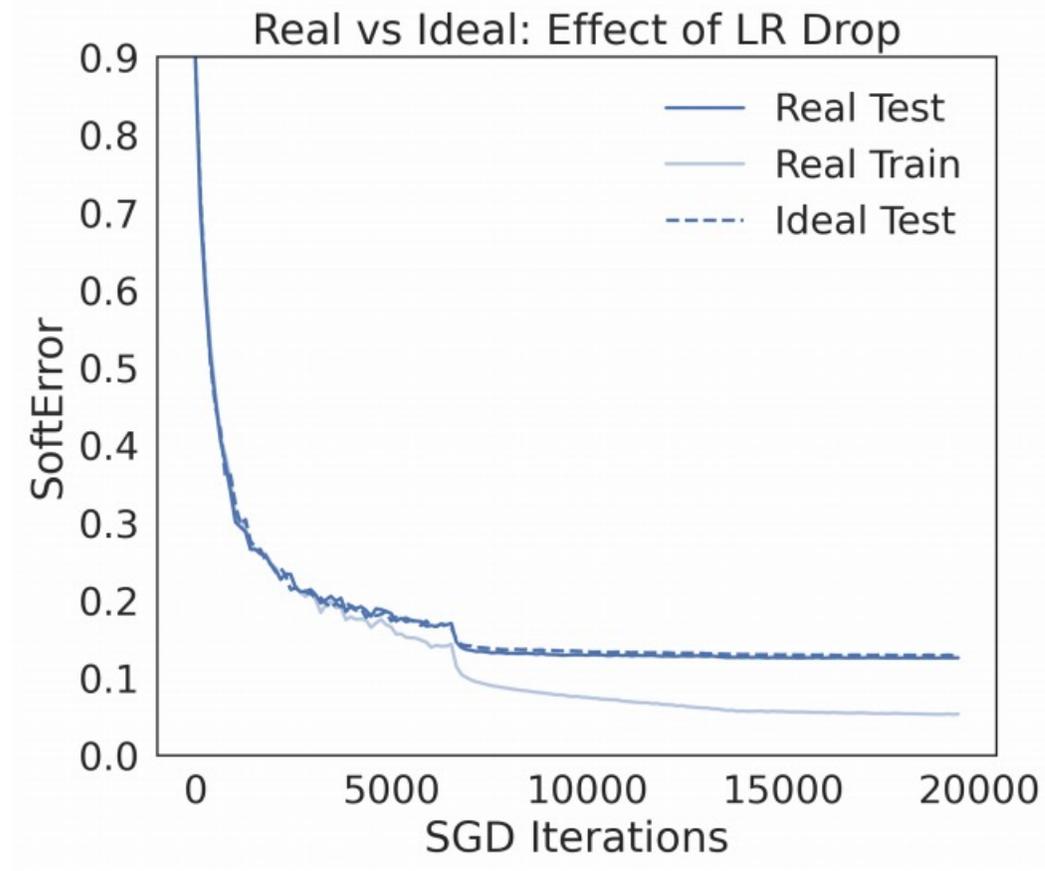
Both train to 0 Train Error, but  
convnet generalizes better.

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]

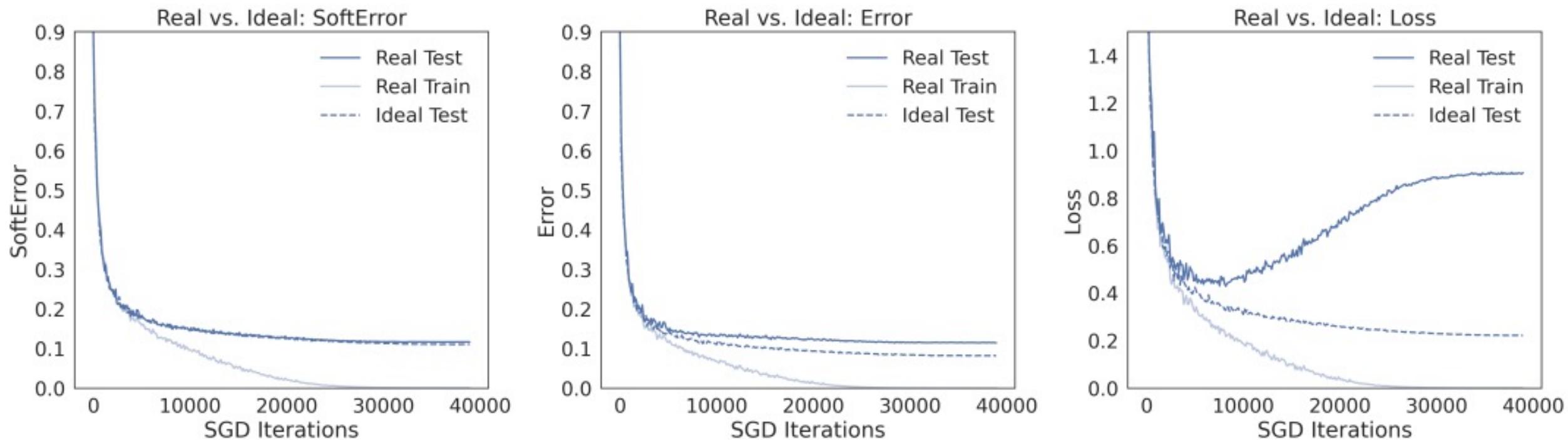
- Train on randomly-labeled inputs.
- 0% train error, 90%/trivial test error.

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)

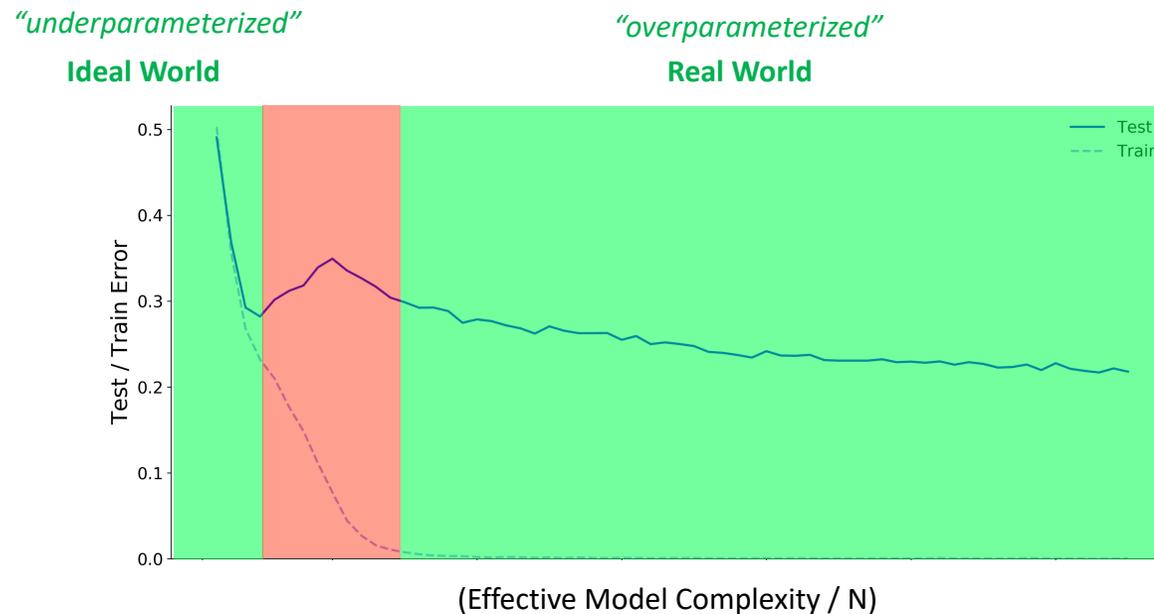
# Conclusions

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

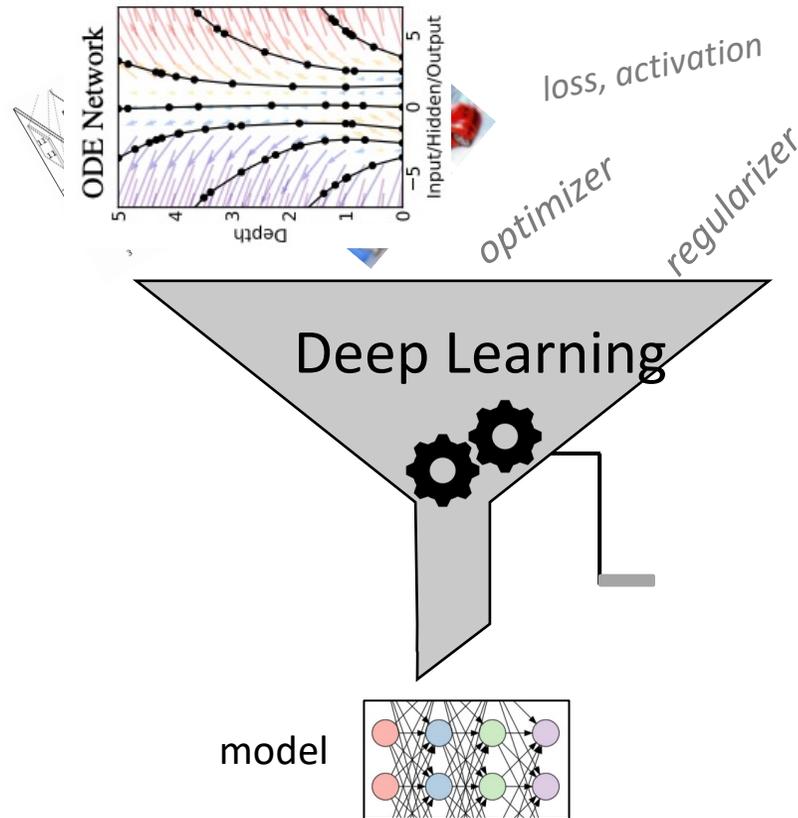
Two regimes in practice:

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!*

**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

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

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

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

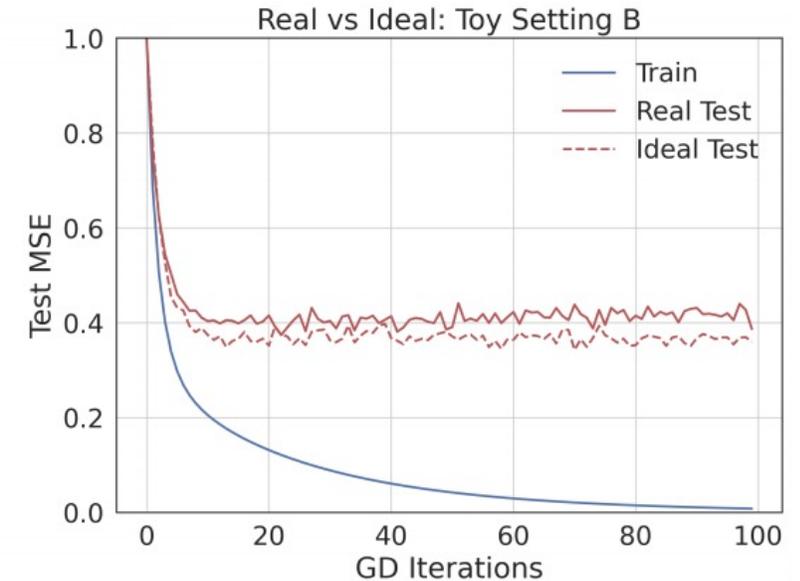
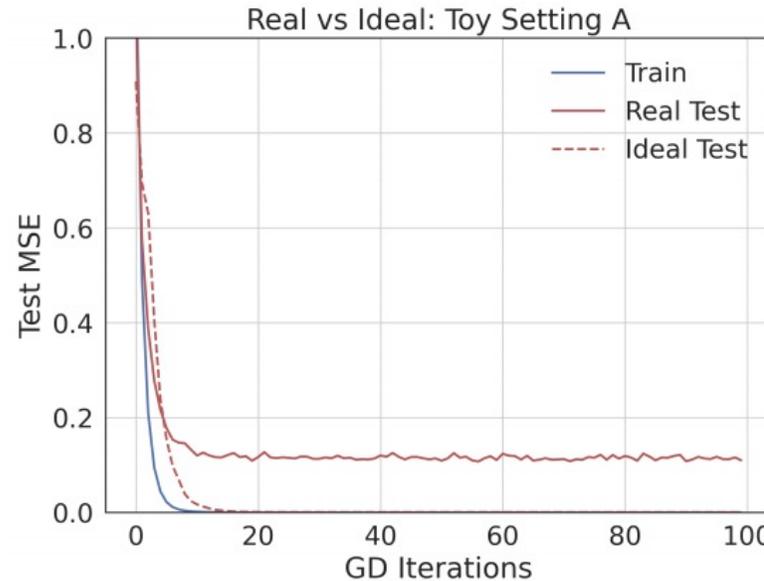
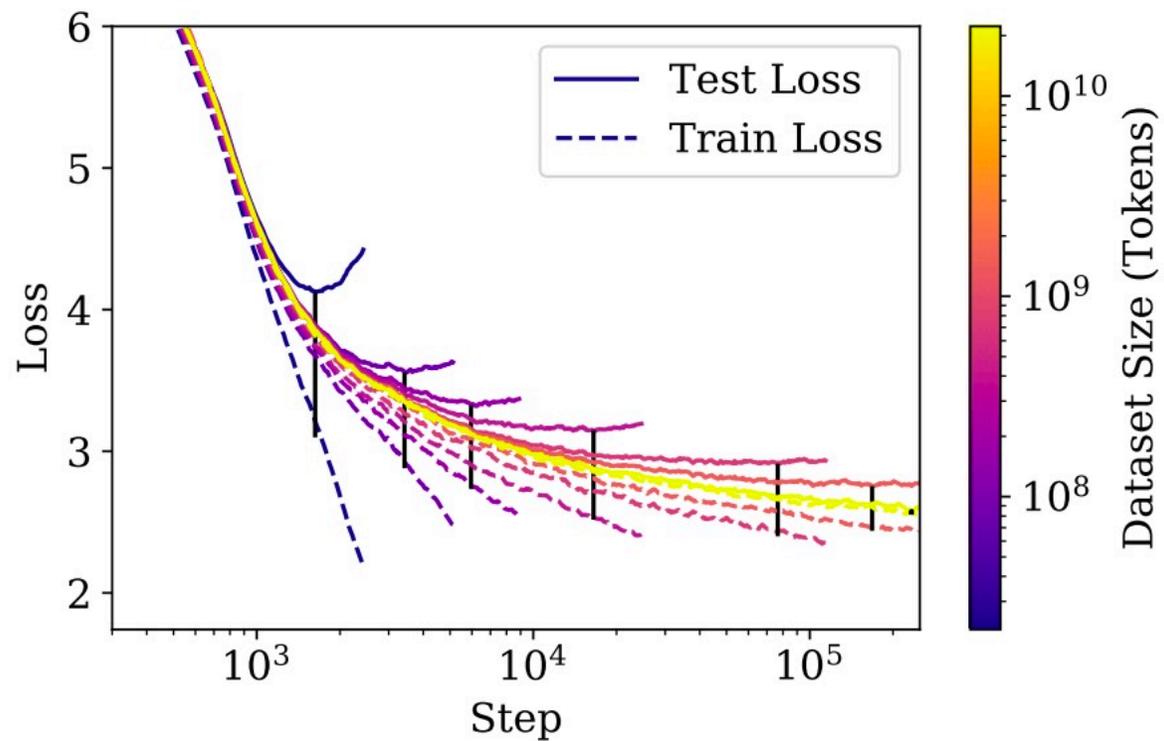


Figure 7: **Toy Example.** Examples of settings with large and small bootstrap error.

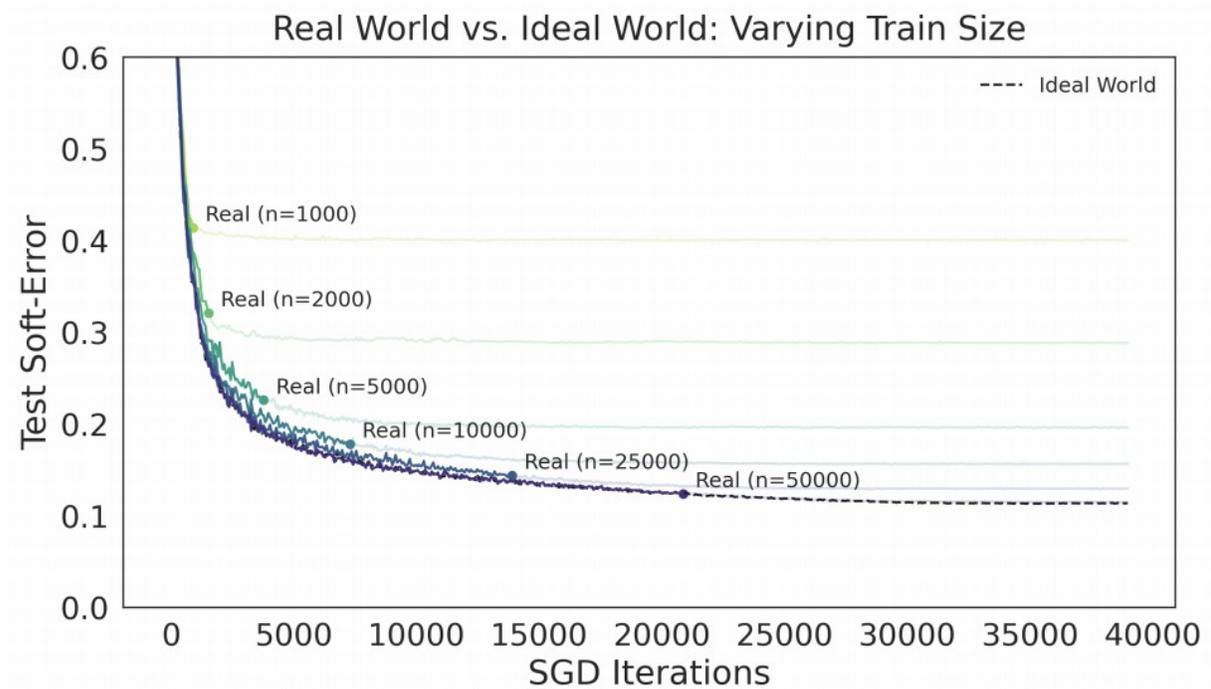
- **Setting A.** Linear activation  $\sigma(x) = x$ . With  $n = 20$  train samples.
- **Setting B.** Sign activation  $\sigma(x) = \text{sgn}(x)$ . With  $n = 100$  train samples.

## GPT-3 Learning Curves

[Kaplan et al 2020]



## ResNet18 Curves

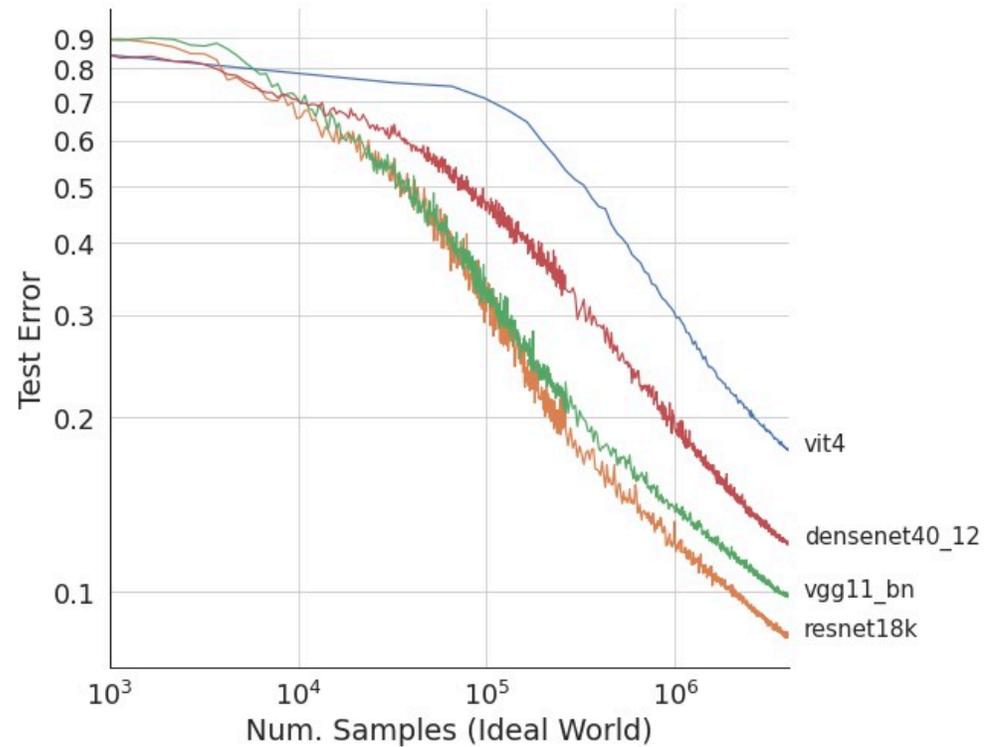


# Scaling Laws in Ideal World

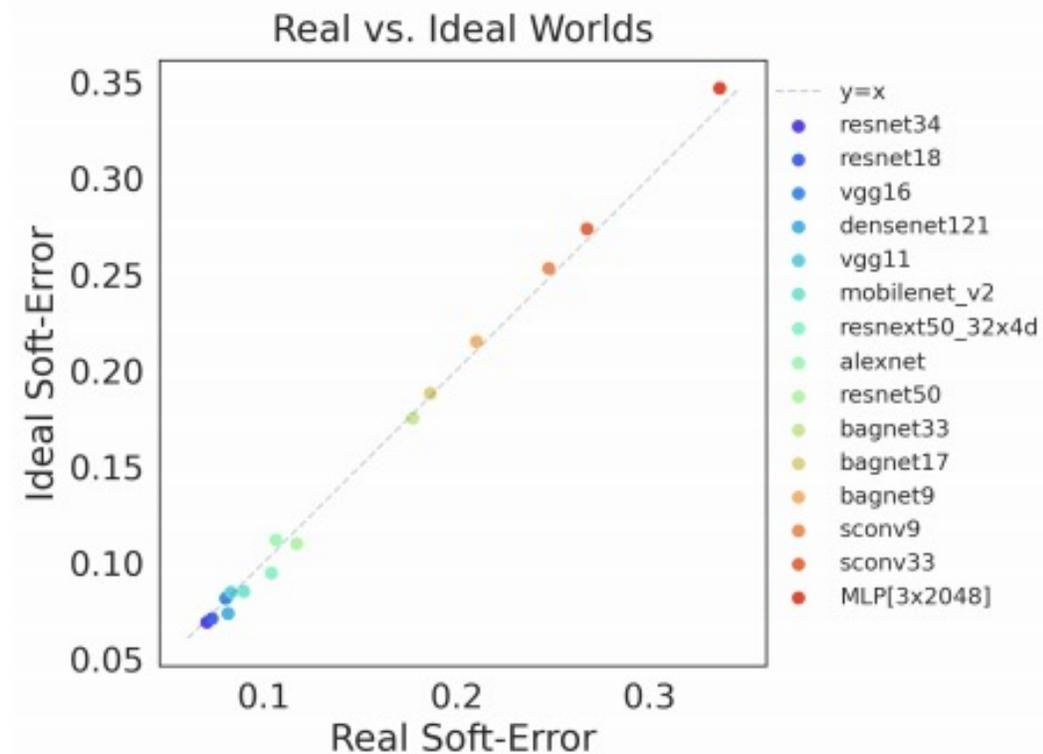
$L(t)$  : Ideal-world learning curve

Empirically: power law

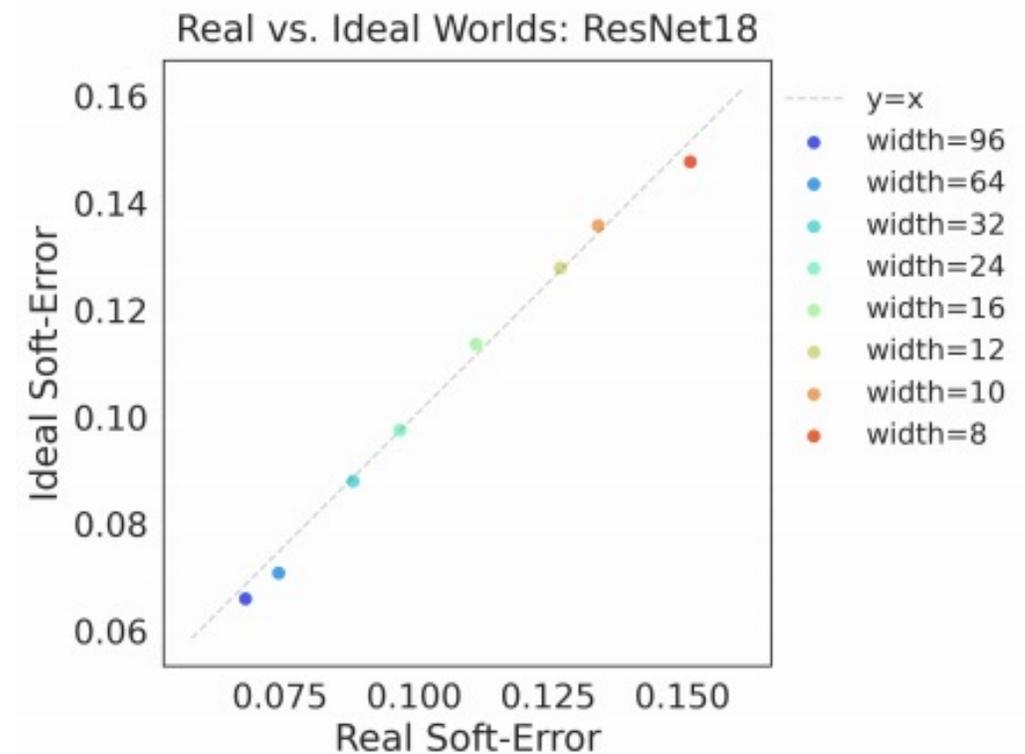
$$L(t) \sim t^{-\alpha}$$



# ImageNet Experiments



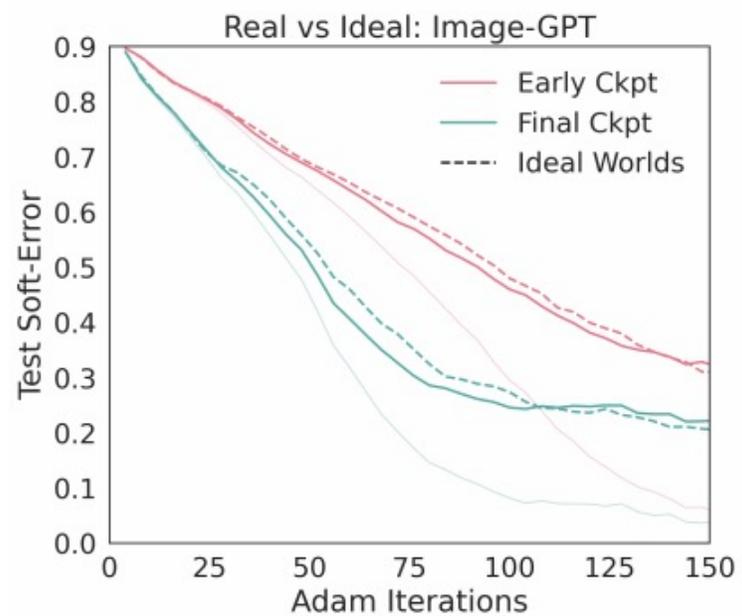
(a) Standard architectures.



(b) ResNet-18s of varying width.

Figure 3: **ImageNet-DogBird**. Real World models trained on 10K samples.

# Effect of Pretraining



(b) Pretrain: Image-GPT ( $n = 2K$ ).

# When Data-Aug Hurts

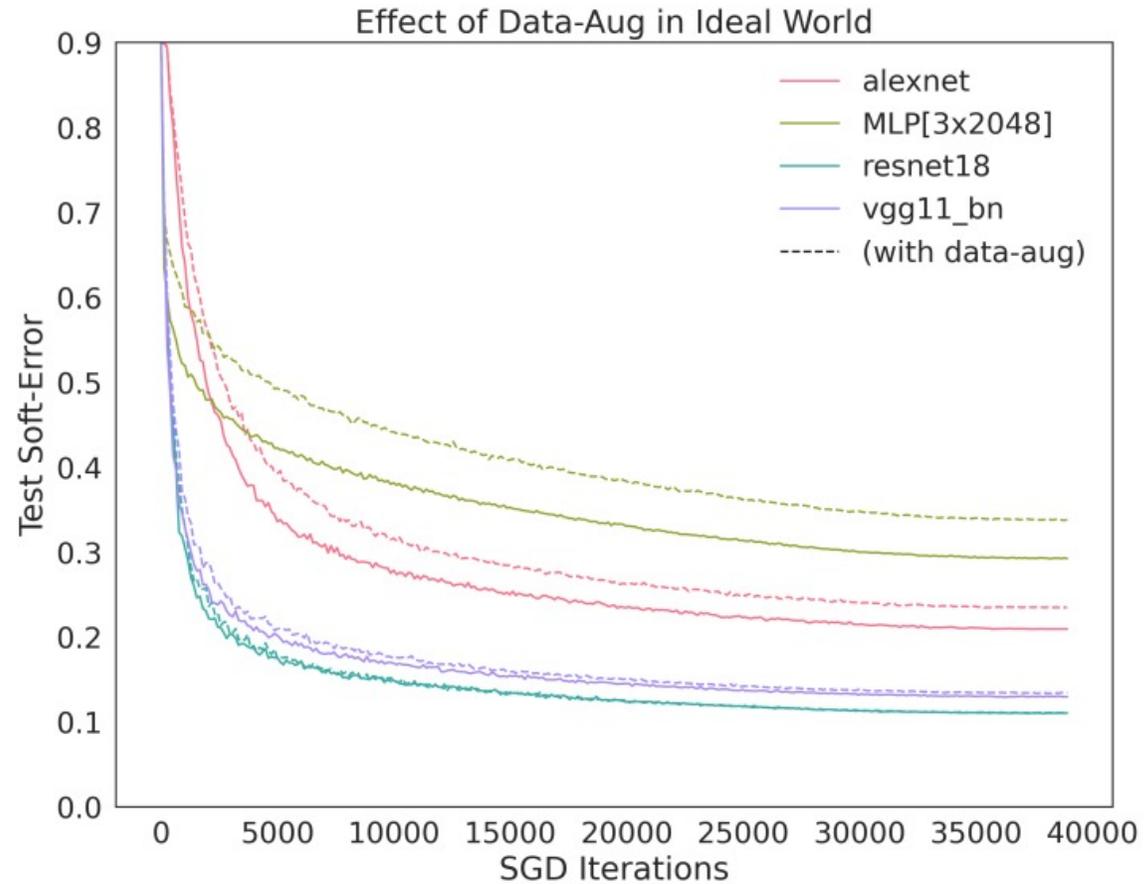


Figure 10: Effect of Data Augmentation in the Ideal World.

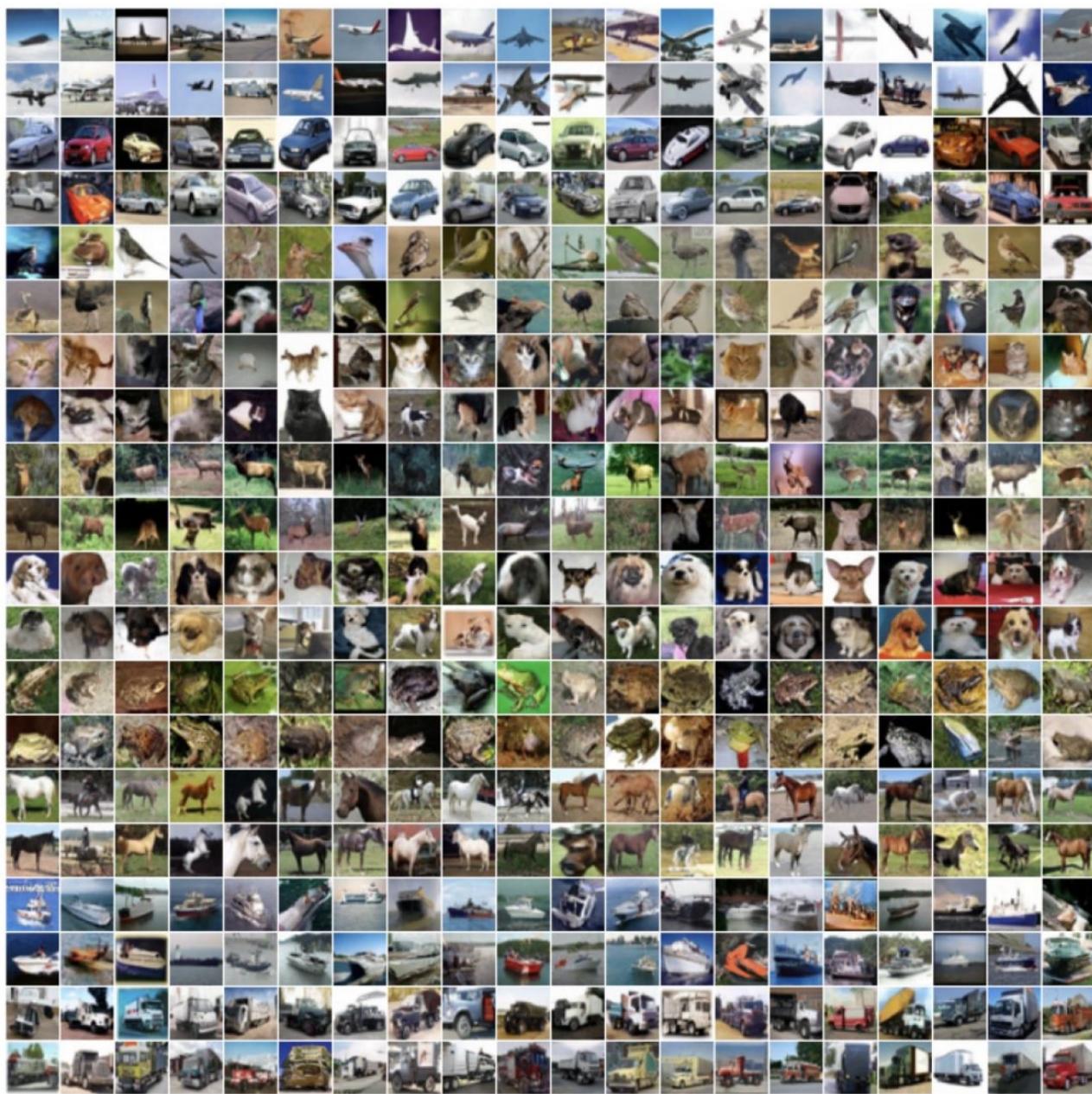


Figure 17: **CIFAR-5m Samples.** Random samples from each class (by row).



Figure 18: **CIFAR-10 Samples.** Random samples from each class (by row).

<b>Trained On</b>	<b>Test Error On</b>	
	CIFAR-10	CIFAR-5m
<b>CIFAR-10</b>	0.032	0.091
<b>CIFAR-5m</b>	0.088	0.097

Table 2: WRN28-10 + cutout on CIFAR-10/5m

norwegian elkhound



lhasa



wire-haired fox terrier



norwich terrier



basset



brittany spaniel



english springer



irish terrier



german short-haired pointer



flat-coated retriever



english springer



italian greyhound



silky terrier



cocker spaniel



bald\_eagle



goose



jacamar



great grey owl



albatross



hummingbird



bustard



goose



water ouzel



ptarmigan



hummingbird



european gallinule



vulture



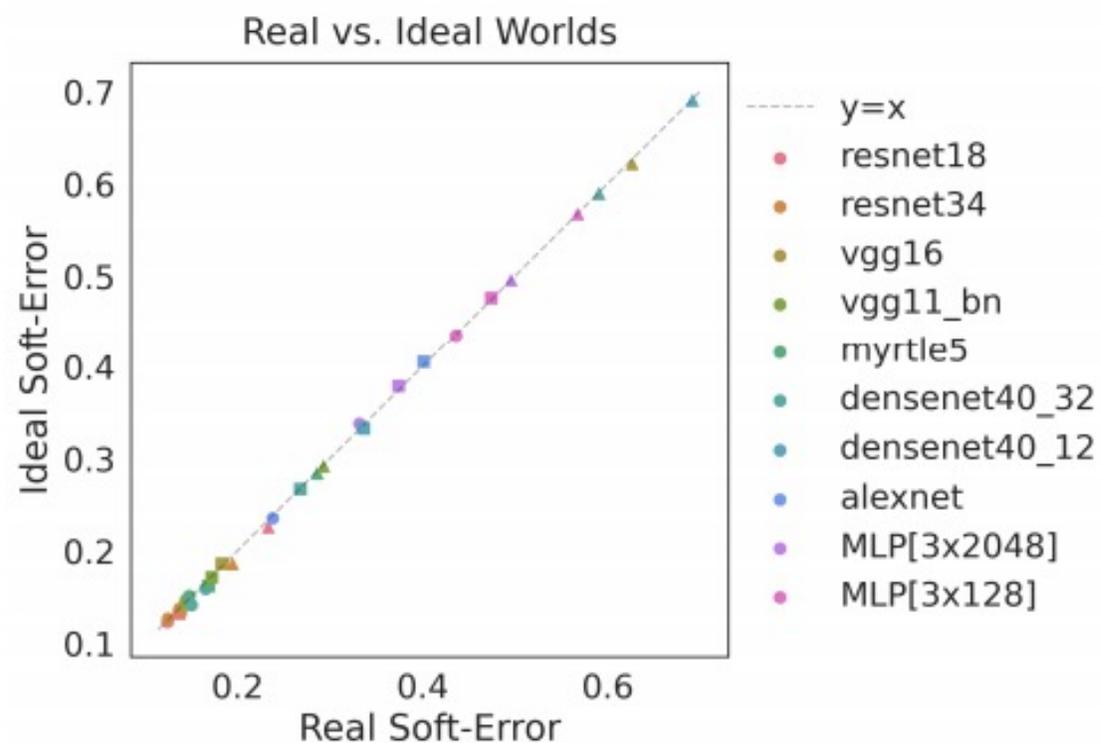
jay



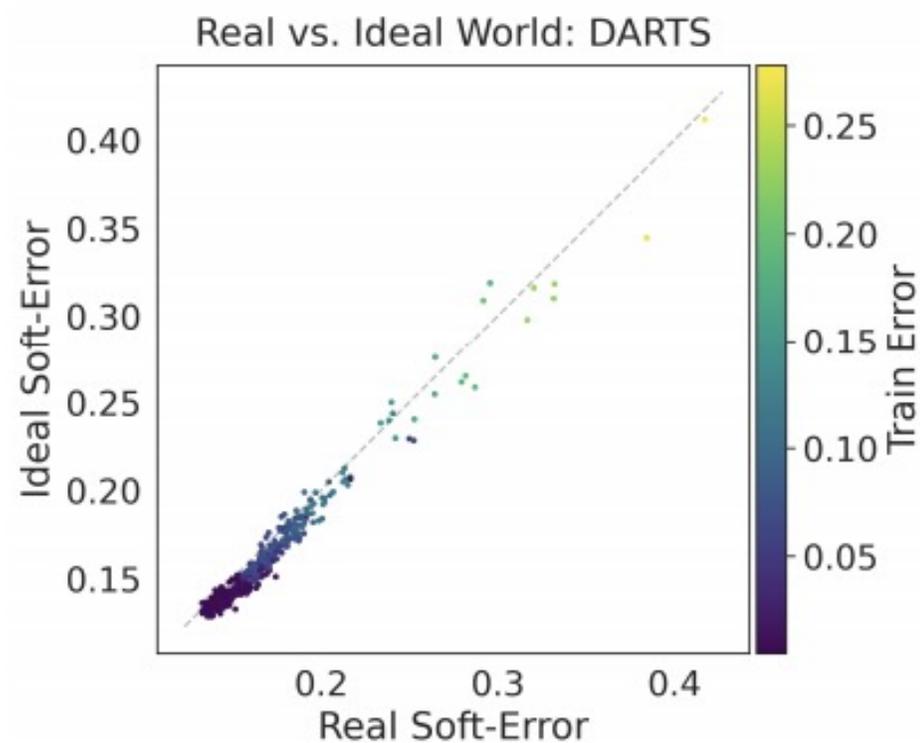
american egret



# CIFAR-5m Experiments



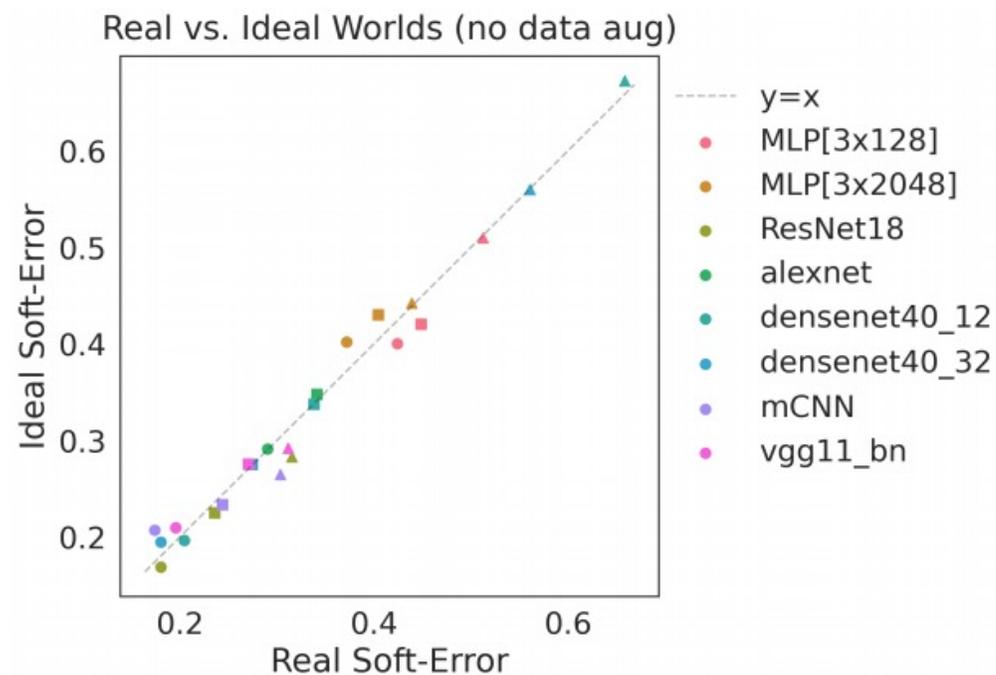
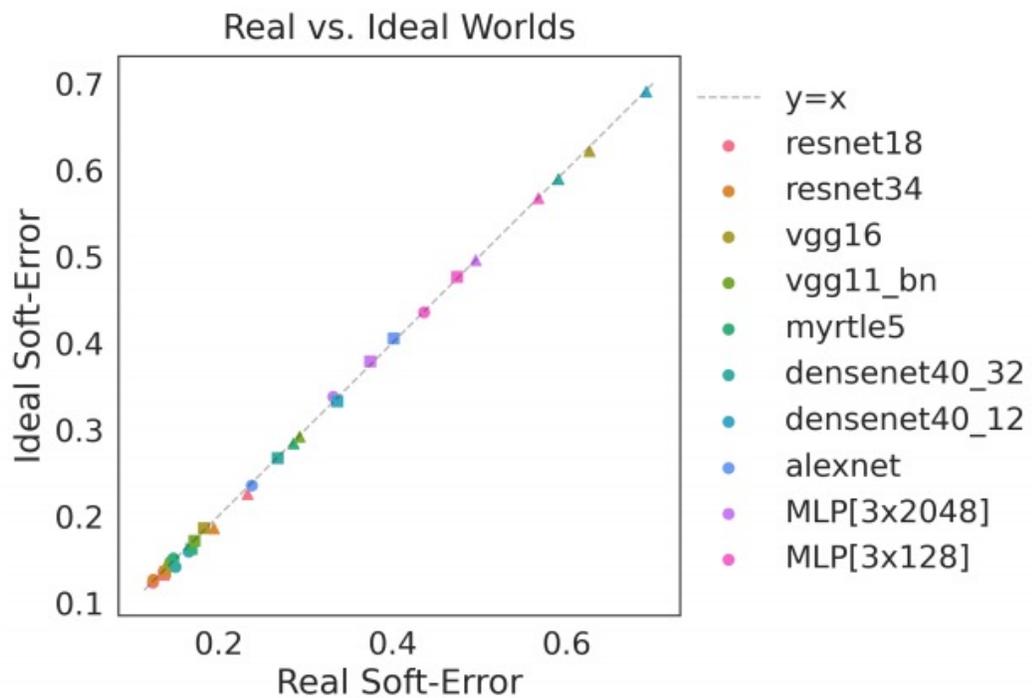
(a) Standard architectures.



(b) Random DARTS architectures.

Figure 2: **Real vs Ideal World: CIFAR-5m.** SGD with 50K samples. (a): Varying learning-rates 0.1 (●), 0.01 (■), 0.001 (▲). (b): Random architectures from DARTS space (Liu et al., 2019).

# ImageNet Experiments



# Validation: Summary of Experiments

- **CIFAR-5m:** 5-million synthetic samples from a generative model trained on CIFAR-10
  - Realistic: Training WRN on  $n=50K$  from CIFAR-5m yields 91.2% test acc on CIFAR-10
- **ImageNet-DogBird:** 155K images by collapsing ImageNet categories.
  - Real World:  $n=10K$  for 120 epochs
  - Ideal World:  $n=155K$  for  $< 8$  epochs (approximation of  $n = \infty$ )
- **Various archs:** convnets, ResNets, MLPs, Image-GPT, Vision-Transformer

***“Deep Bootstrap”***



$$\text{RealWorld}(N, T = \infty) \approx \text{RealWorld}(N, T_N) \approx_{\epsilon} \text{RealWorld}(\infty, T_N)$$

***Practice: Real World***  
*(trained as long as possible)*

***Real World***  
*(stopped at  $T_N$  : when Train Error  $\approx 1\%$ )*

***Ideal World***  
*(stopped at  $T_N$ )*

Learning curves:

$L(n)$ : 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)

Then:

$$L(n) \approx \tilde{L}(T(n))$$

# Classical Framework (ERM)

**Classical Framework:** Finite data, need to understand *generalization gap*

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

*“Good models are those with small generalization gap”*

## Obstacles:

1. Hard: Decades of work, little progress.
2. Large models can fit train sets → trivializes framework