An overview of empirical studies in NNs

Levent Sagun, EPFL

Mostly on the landscape

1. Shape of the landscape
2. Dynamics on the landscape

Loss function fixes the landscape

1. Take an iid dataset and split into two parts $\mathcal{D}_{train}$ & $\mathcal{D}_{test}$
2. Form the loss using only $\mathcal{D}_{train}$

$$\mathcal{L}_{train}(\theta) = \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} \ell(y, f(\theta; x))$$

3. Find: $\theta^* = \arg\min \mathcal{L}_{train}(\theta)$
4. *...and hope that it will work on* $\mathcal{D}_{test}$

Loss function fixes the landscape

1. Take an iid dataset and split into two parts $\mathcal{D}_{train}$ & $\mathcal{D}_{test}$
2. Form the loss using only $\mathcal{D}_{train}$

$$\mathcal{L}_{train}(\theta) = \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} \ell(y, f(\theta; x))$$

3. Find: $\theta^* = \arg\min \mathcal{L}_{train}(\theta)$
4. *...and hope that it will work on* $\mathcal{D}_{test}$

- N : number of parameters $\theta \in \mathbb{R}^N$
- P : number of examples in the *training* set $|\mathcal{D}_{train}|$

Observation 1

GD vs SGD

Moving on the fixed landscape

1. Take an iid dataset and split into two parts $\mathcal{D}_{train}$ & $\mathcal{D}_{test}$
2. Form the loss using only $\mathcal{D}_{train}$

$$\mathcal{L}_{train}(\theta) = \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} \ell(y, f(\theta; x))$$

3. Find: $\theta^* = \arg\min \mathcal{L}_{train}(\theta)$
4. *...and hope that it will work on* $\mathcal{D}_{test}$

- N : number of parameters $\theta \in \mathbb{R}^N$
- P : number of examples in the *training* set $|\mathcal{D}_{train}|$

GD is bad use SGD

"Stochastic gradient learning in neural networks"

Léon Bottou, 1991

- The total gradient (3) converges to a *local minimum* of the cost function. The algorithm then cannot escape this local minimum, which is sometimes a poor solution of the problem.

  In practical situations, the gradient algorithm may get stuck in an area where the cost is extremely ill conditionned, like a deep ravine of the cost function. This situation actually is a local minimum in a subspace defined by the largest eigenvalues of the Hessian matrix of the cost.

  The stochastic gradient algorithm (4) usually is able to escape from such bothersome situations, thanks to its random behavior (Bourrely, 1989).

GD is bad use SGD

Bourrely, 1988

5 CONCLUSION

It has been shown that the difficulty in parallel learning is due to the fact that the parallel algorithm does not really use the stochastic algorithm. Two solutions are presently proposed to prevent the system from falling into a local minimum.

1) Add momentum to the algorithm such that it can "roll past" a local minimum. Thus the algorithm then becomes:

$$W_{t+1} = (1-\alpha)\, W_t - \varepsilon\, \alpha\, f(W_t, X_t)$$

where f is the error gradient Q relative to W

2) One can add a random "noise" to the gradient calculations. One method of performing this task is to calculate the gradients in an approximate manner. This variation could be modelled as a type of 'Brownian motion', using a temperature function (similar to simulated annealing). This temperature could be lowered relative to the remaining system error. For example, the variation in gradients could follow a Gaussian distribution. Thus, for example:

$$W_{t+1} = W_t - \varepsilon\, N\left(f(W_t, X_t),\ k\sqrt{Temp} \right)$$

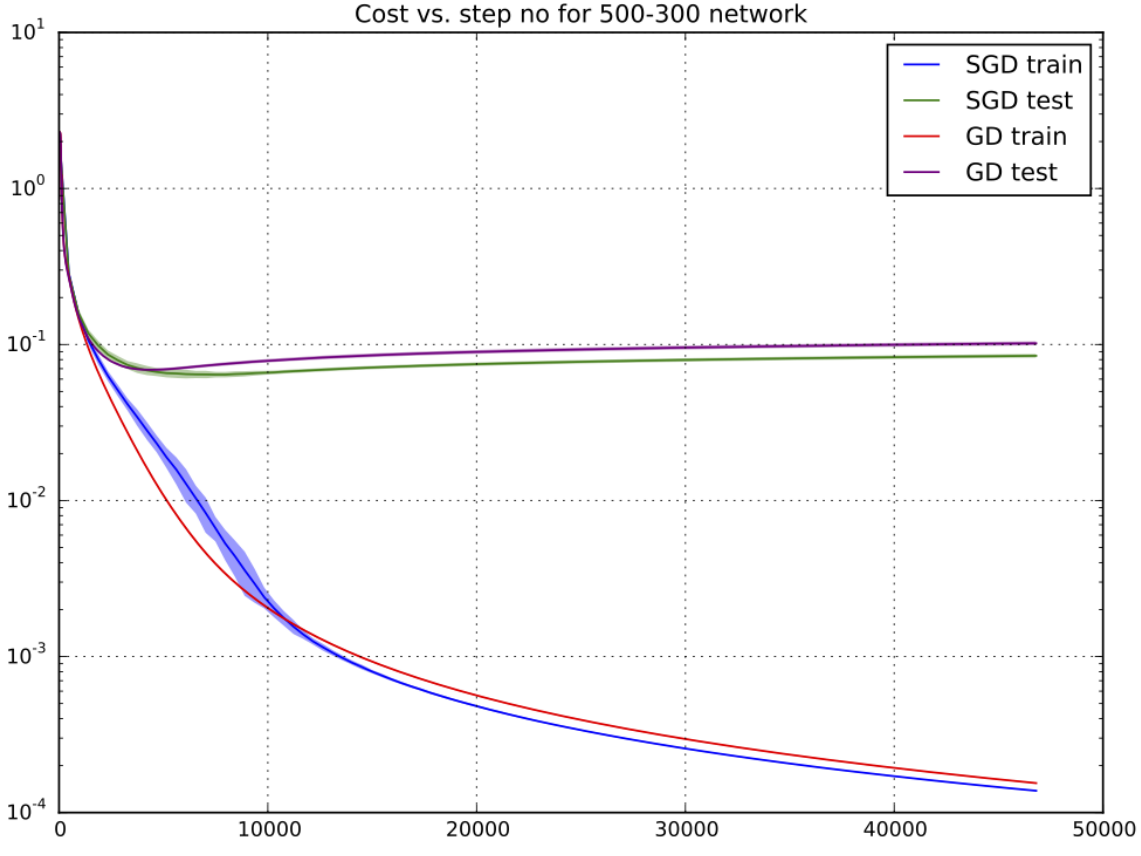
where f is the error gradient Q relative to W

and N is a function giving a Gaussian random variable.

Both of these approaches are presently under research.

GD is the same as SGD

Fully connected network on MNIST: $N \sim 450\text{K}$



Different regimes depending on N

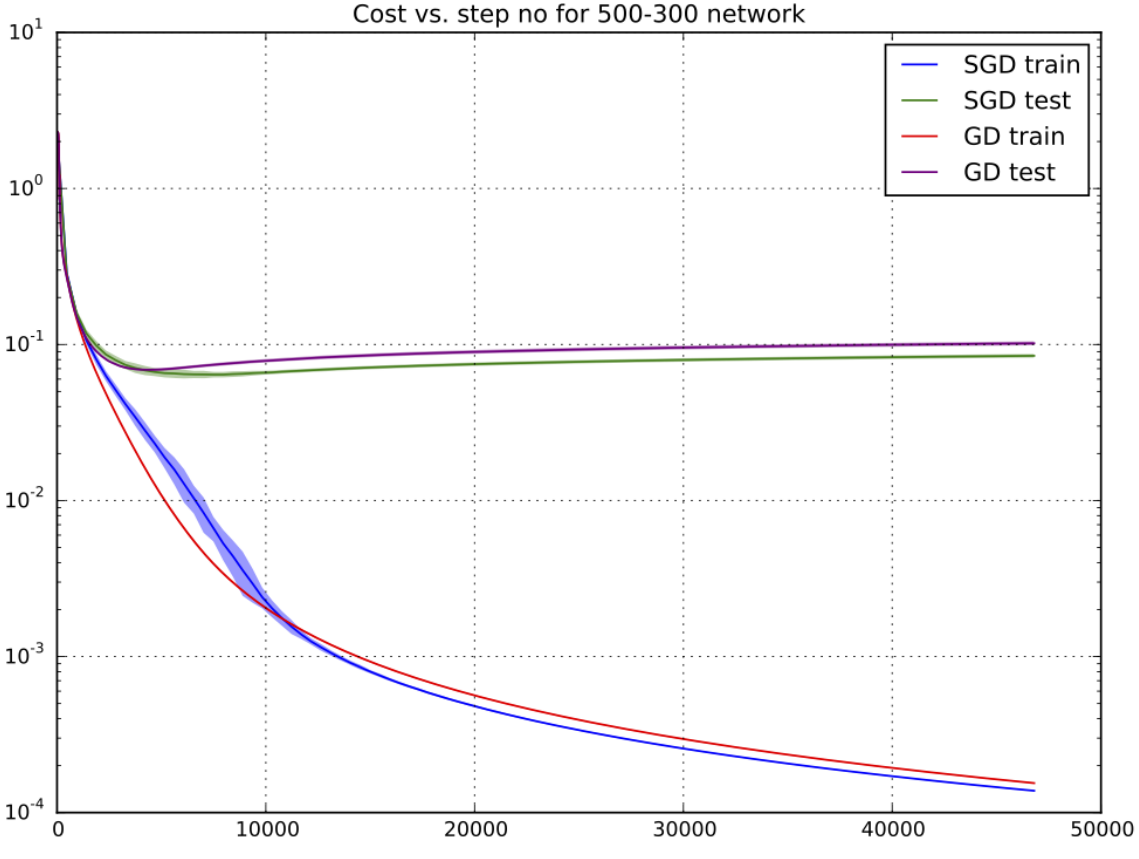
Bourrely, 1988

Evaluation of computational time and learning time is achieved by training the network for the handwritten numbers recognition task. The network is designed as follows : 400 input units (a 20x20 grid), 5 hidden units and 10 output units. It must perform a classification task: for each input number, the network must activate the correct output unit (0 to 9). In addition, it must overcome distortions such as vertical and horizontal translations, scaling, rotation and random white noise.

Numbers are coded on matrices of 20x20 real grey-levels. Table 1

GD is the same as SGD

Fully connected network on MNIST: $N \sim 450\text{K}$



Average number of mistakes: SGD 174, GD 194

Recent theoretical results

Is it really the case that in the large N limit, GD $=$ SGD?

Recent theoretical results

Is it really the case that in the large N limit, GD $=$ SGD?

Mean Field approach to 1 hidden layer NNs:

- Mei, Montanari, Nguyen 2018
- Sirignano, Spiliopoulos 2018
- Rotskoff, Vanden-Eijnden 2018
- Chizat, Bach 2018

Recent theoretical results

Is it really the case that in the large N limit, GD $=$ SGD?

Mean Field approach to 1 hidden layer NNs:

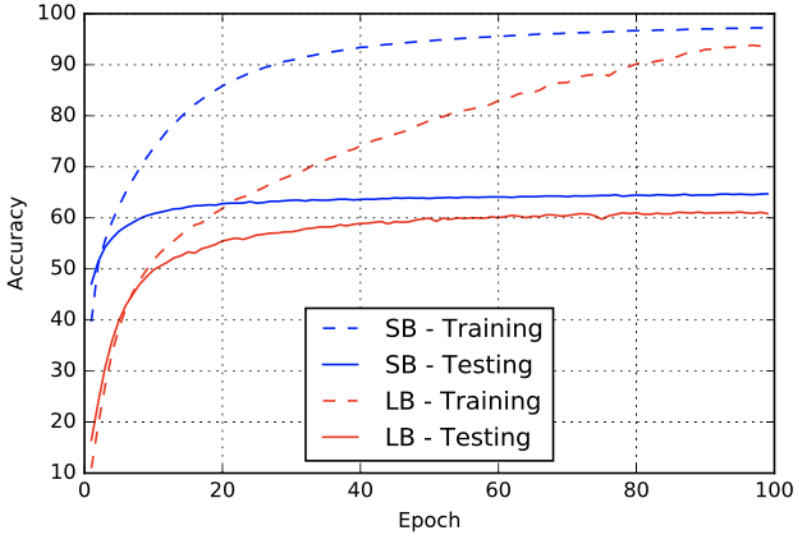
- Mei, Montanari, Nguyen 2018
- Sirignano, Spiliopoulos 2018
- Rotskoff, Vanden-Eijnden 2018
- Chizat, Bach 2018

"when initialized correctly and in the many-particle limit the gradient flow converges to global minimizers"

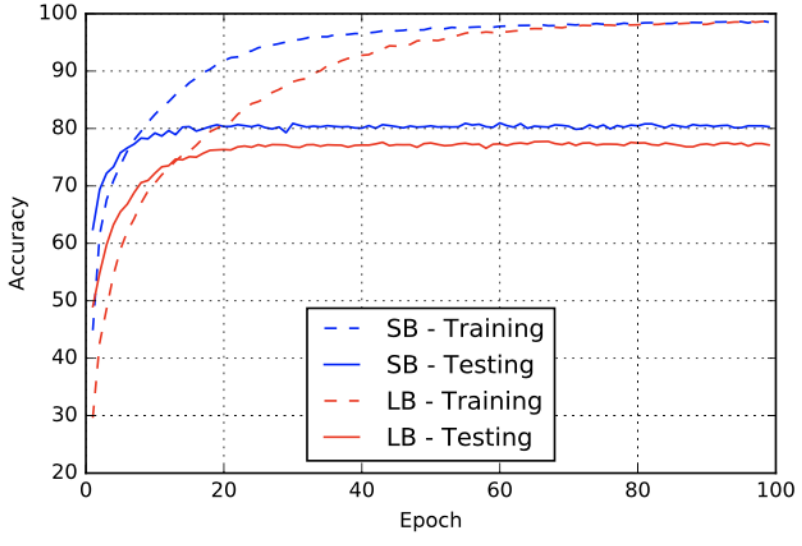
Regime where SGD is really special?

Where common wisdom may be true (Keskar et. al. 2016):
$\rightarrow$ Similar training error, but gap in the test error.

fully connected, TIMIT $N = 1.2$M



conv-net, CIFAR10 $N = 1.7$M

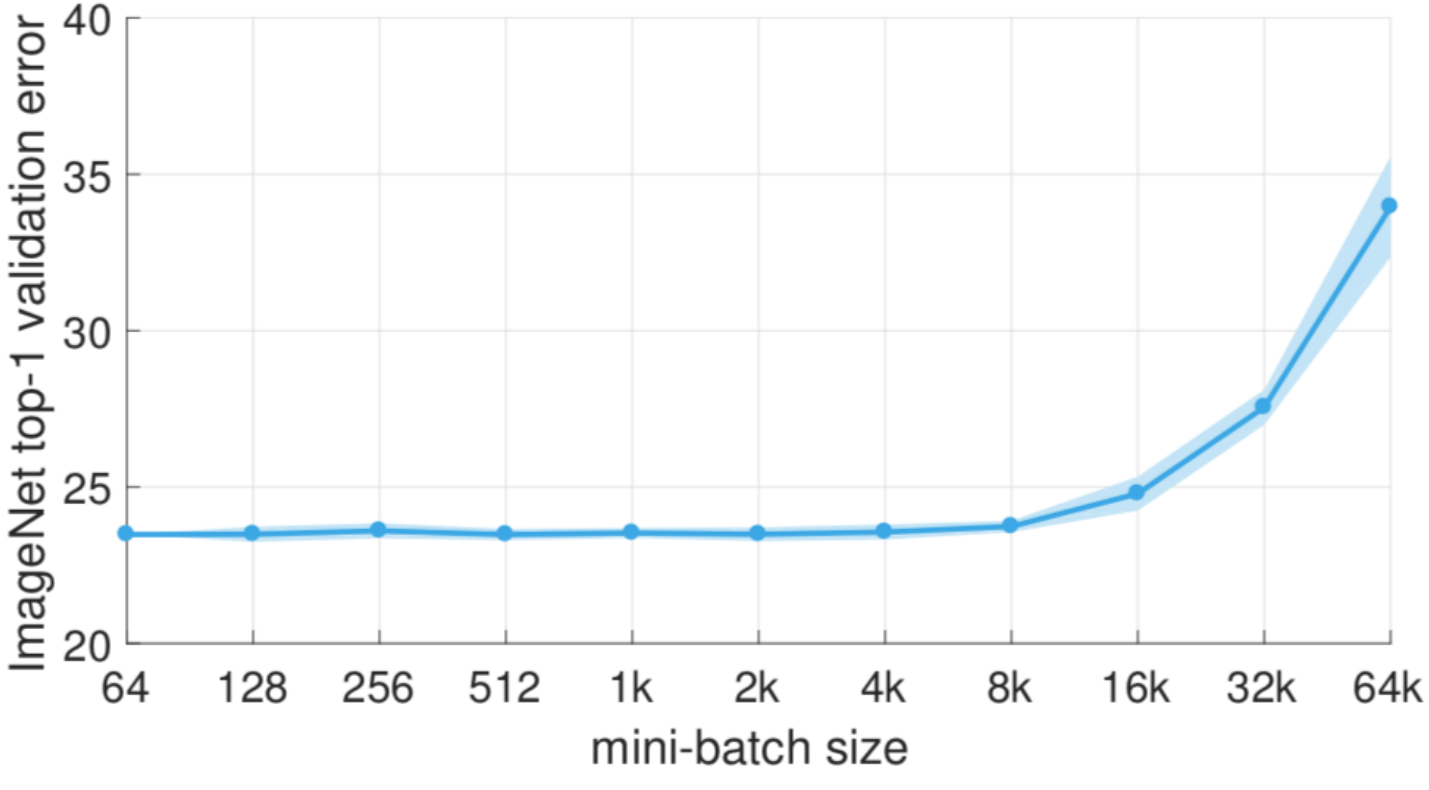


The 'generalization gap' can be filled

- Jastrzębski et. al. 2018
- Goyal et. al. 2018
- Shallue and Lee et. al. 2018
- McCandlish et. al. 2018
- Smith et. al. 2018

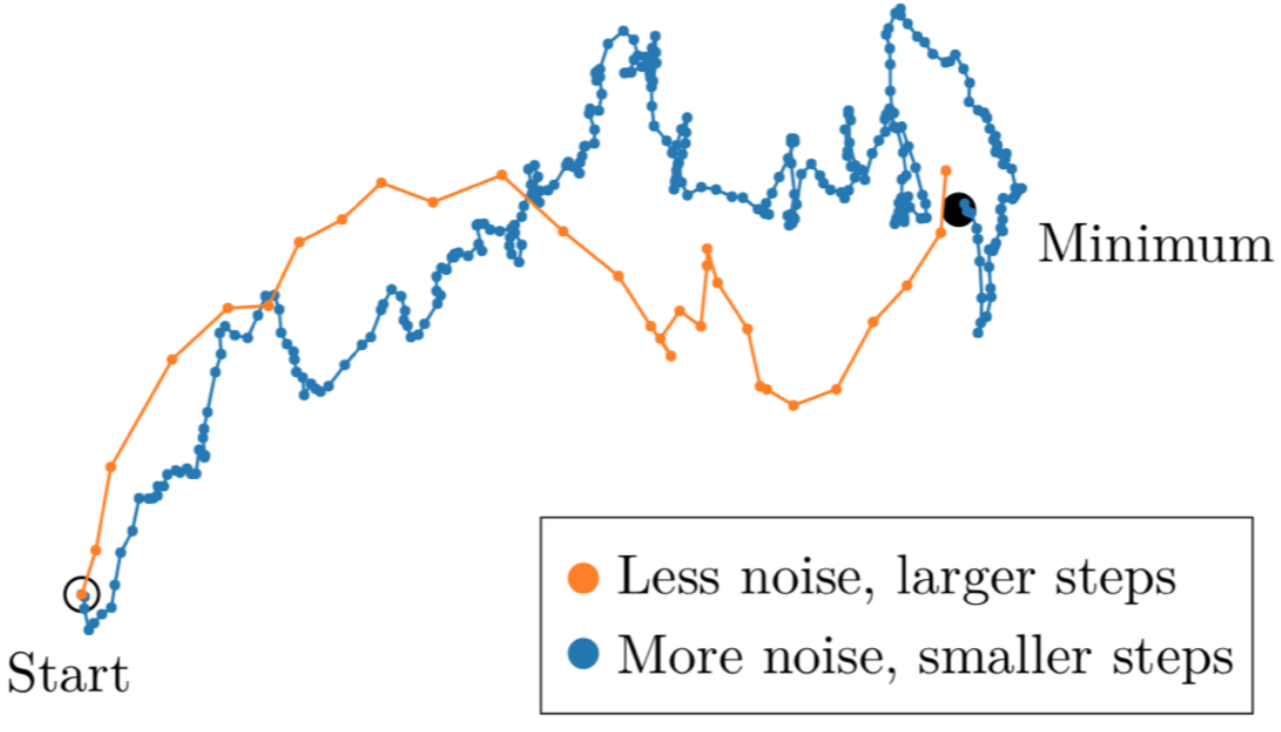
The 'generalization gap' can be filled

- Jastrzębski et. al. 2018
- Goyal et. al. 2018
- Shallue and Lee et. al. 2018
- McCandlish et. al. 2018
- Smith et. al. 2018



The 'generalization gap' can be filled

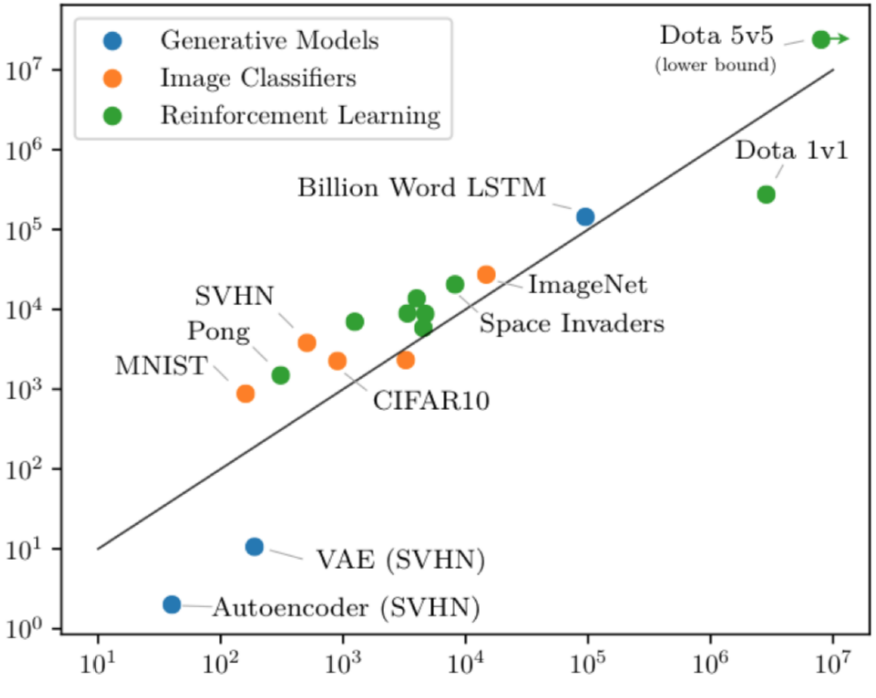
- Jastrzębski et. al. 2018
- Goyal et. al. 2018
- Shallue and Lee et. al. 2018
- McCandlish et. al. 2018
- Smith et. al. 2018



The 'generalization gap' can be filled

- Jastrzębski et. al. 2018
- Goyal et. al. 2018
- Shallue and Lee et. al. 2018
- McCandlish et. al. 2018
- Smith et. al. 2018

Gradient noise scale measures the variation of the gradients between different training examples



Critical batch size is the maximum batch size above which scaling efficiency decreases significantly

The 'generalization gap' can be filled

- Jastrzębski et. al. 2018
- Goyal et. al. 2018
- Shallue and Lee et. al. 2018
- McCandlish et. al. 2018
- Smith et. al. 2018

where we have established that $(\boldsymbol{g}^{(S)}(\boldsymbol{\theta}_k) - \boldsymbol{g}(\boldsymbol{\theta}_k))$ is an additive zero mean Gaussian random noise with variance $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = (1/S)\mathbf{C}(\boldsymbol{\theta})$. Hence we can rewrite (3) as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \boldsymbol{g}(\boldsymbol{\theta}_k) + \frac{\eta}{\sqrt{S}} \boldsymbol{\epsilon} \ , \tag{4}$$

where $\boldsymbol{\epsilon}$ is a zero mean Gaussian random variable with covariance $\mathbf{C}(\boldsymbol{\theta})$.

The 'generalization gap' can be filled

- Jastrzębski et. al. 2018
- Goyal et. al. 2018
- Shallue and Lee et. al. 2018
- McCandlish et. al. 2018
- Smith et. al. 2018

where we have established that $(\boldsymbol{g}^{(S)}(\boldsymbol{\theta}_k) - \boldsymbol{g}(\boldsymbol{\theta}_k))$ is an additive zero mean Gaussian random noise with variance $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = (1/S)\mathbf{C}(\boldsymbol{\theta})$. Hence we can rewrite (3) as

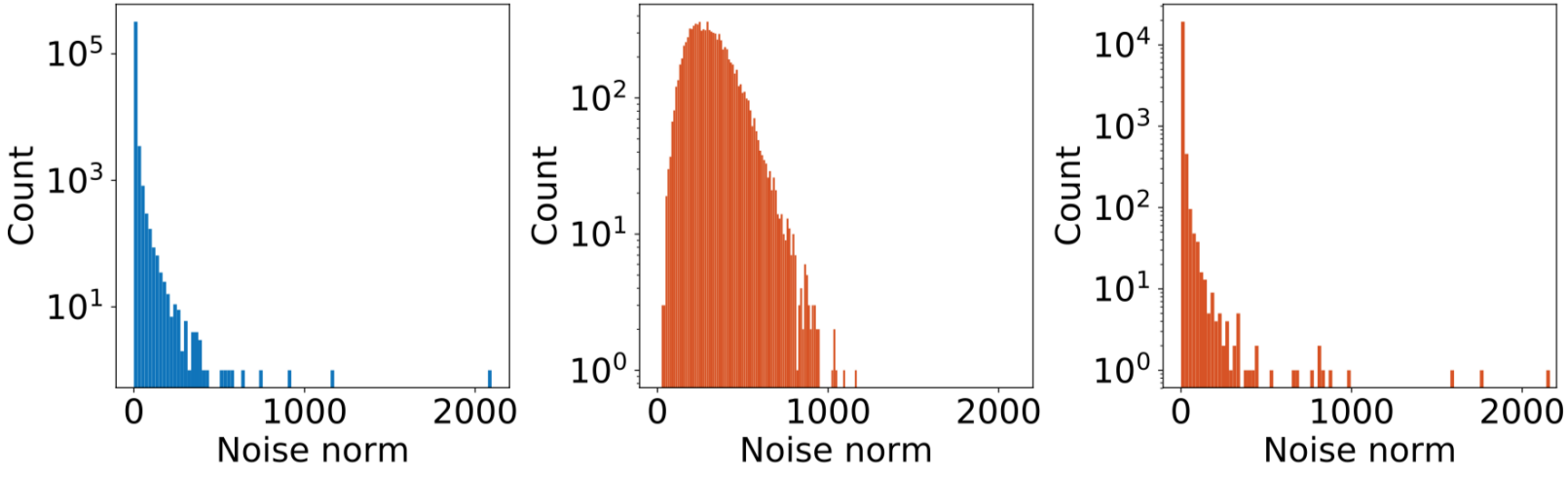
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \boldsymbol{g}(\boldsymbol{\theta}_k) + \frac{\eta}{\sqrt{S}} \boldsymbol{\epsilon} \ , \tag{4}$$

where $\boldsymbol{\epsilon}$ is a zero mean Gaussian random variable with covariance $\mathbf{C}(\boldsymbol{\theta})$.

But the noise is not Gaussian!

The 'generalization gap' can be filled

But the noise is not Gaussian!



(a) Real (b) Gaussian (c) α-stable

Simsekli, Sagun, Gurbuzbalaban 2019

Large batch allows parallel training

Accurate, Large Minibatch SGD:
Training ImageNet in 1 Hour

Priya Goyal Piotr Dollár Ross Girshick Pieter Noordhuis
Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He

Facebook

Abstract

Deep learning thrives with large neural networks and large datasets. However, larger networks and larger datasets result in longer training times that impede research and development progress. Distributed synchronous SGD offers a potential solution to this problem by dividing

Large batch allows parallel training

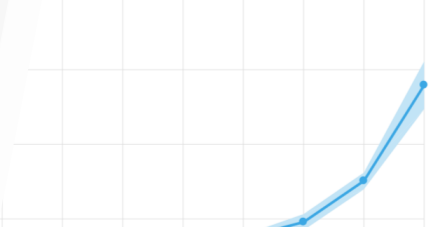
Now anyone can train Imagenet in 18 minutes

Written: 10 Aug 2018 by Jeremy Howard

Note from Jeremy: I'll be teaching Deep Learning for Coders at the University of San Francisco starting in October; if you've got at least a year of coding experience, you can apply here.

A team of fast.ai alum Andrew Shaw, DIU researcher Yaroslav Bulatov, and I have managed to train Imagenet to 93% accuracy in just 18 minutes, using 16 public AWS cloud instances, each with 8 NVIDIA V100 GPUs, running the fastai and PyTorch libraries. This is a new speed record for training Imagenet to this accuracy on publicly available infrastructure, and is 40% faster than Google's DAWNBench record on their proprietary TPU Pod cluster. Our approach uses the same number of processing units as Google's benchmark (128) and costs around $40 to run.

Noordhuis
g Jia Kaiming He



Large batch allows parallel training

Now an-
minu

Written: 10 ,

Note from Jer
San Francisco
experience, you

A team of fast.ai alum
managed to train Imag
cloud instances, each wit
libraries. This is a new spe
publicly available infrastruc
on their proprietary TPU Pod
processing units as Google's be

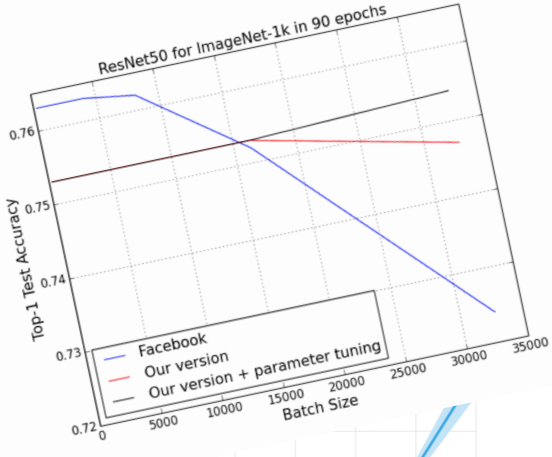
ImageNet Training in Minutes

Yang You[1], Zhao Zhang[2], Cho-Jui Hsieh[3], James Demmel[1], Kurt Keutzer[1]
UC Berkeley[1], TACC[2], UC Davis[3]
{youyang, demmel, keutzer}@cs.berkeley.edu; zzhang@tacc.utexas.edu; chohsieh@ucdavis.edu

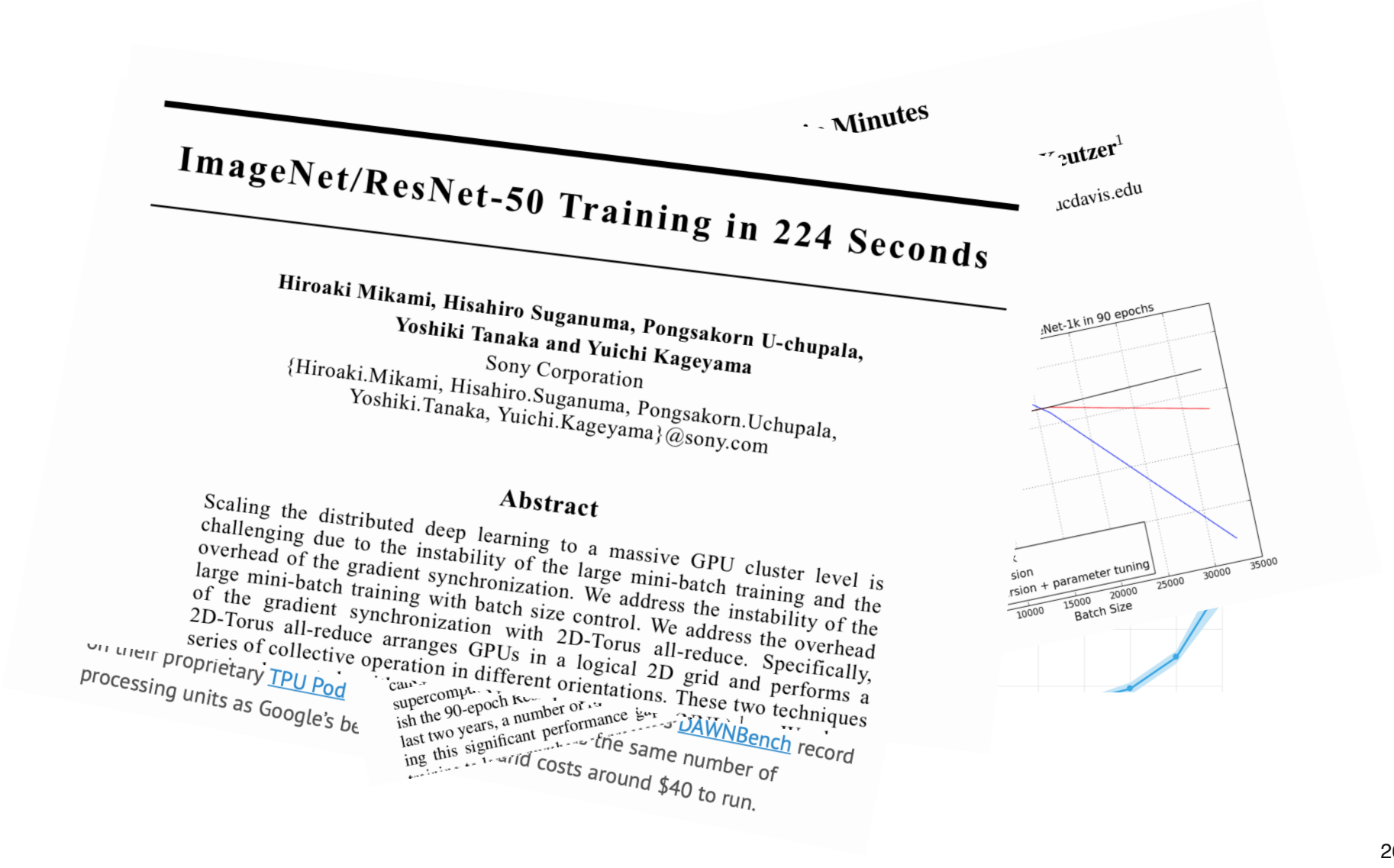
Abstract

Since its creation, the ImageNet-1k benchmark set has played a significant role as a benchmark for ascertaining the accuracy of different deep neural net (DNN) models on the classification problem. Moreover, in recent years it has also served as the principal benchmark for assessing different approaches to DNN training. Finishing a 90-epoch ImageNet-1k training with ResNet-50 on a NVIDIA M40 GPU takes 14 days. This training requires 10^{18} single precision operations in total. On the other hand, the world's current fastest supercomputer can finish 2×10^{17} single precision operations per second. If we can make full use of the computing capability of the fastest supercomputer for DNN training, we should be able to finish the 90-epoch ResNet-50 training in five seconds. Over the last two years, a number of researchers have focused on closing this significant performance gap through scaling DNN

DAWNBench record
the same number of
ard costs around $40 to run.



Large batch allows parallel training



Minutes

ImageNet/ResNet-50 Training in 224 Seconds

Hiroaki Mikami, Hisahiro Suganuma, Pongsakorn U-chupala,
Yoshiki Tanaka and Yuichi Kageyama
Sony Corporation
{Hiroaki.Mikami, Hisahiro.Suganuma, Pongsakorn.Uchupala,
Yoshiki.Tanaka, Yuichi.Kageyama}@sony.com

Abstract

Scaling the distributed deep learning to a massive GPU cluster level is challenging due to the instability of the large mini-batch training and the overhead of the gradient synchronization. We address the instability of the large mini-batch training with batch size control. We address the overhead of the gradient synchronization with 2D-Torus all-reduce. Specifically, 2D-Torus all-reduce arranges GPUs in a logical 2D grid and performs a series of collective operation in different orientations. These two techniques

processing units as Google's be

TPU Pod

DAWNBench

Net-1k in 90 epochs

sion
rsion + parameter tuning

Batch Size

Observation 2

A look at the bottom of the loss

Different kinds of minima

- Continuing with Keskar et al (2016): LB $\rightarrow$ *sharp,* SB $\rightarrow$ *wide...*
- Also see Jastrzębski et. al. (2018), Chaudhari et. al. (2016)...
- Older considerations Pardalos et. al. (1993)
- Sharpness depends on parametrization: Dinh et. al. (2017)

Different kinds of minima

- Continuing with Keskar et al (2016): LB $\rightarrow$ *sharp,* SB $\rightarrow$ *wide...*
- Also see Jastrzębski et. al. (2018), Chaudhari et. al. (2016)...
- Older considerations Pardalos et. al. (1993)
- Sharpness depends on parametrization: Dinh et. al. (2017)

$$\tilde{H}_{\Lambda,f}(R) \equiv f^{-1}\Big\{\int S_\Lambda(R-R')f[H(R')]dR'\Big\} \tag{2}$$

where R is a multidimensional vector representing all the coordinates in the molecule.

One of the simplest and most useful forms for S_Λ is a Gaussian

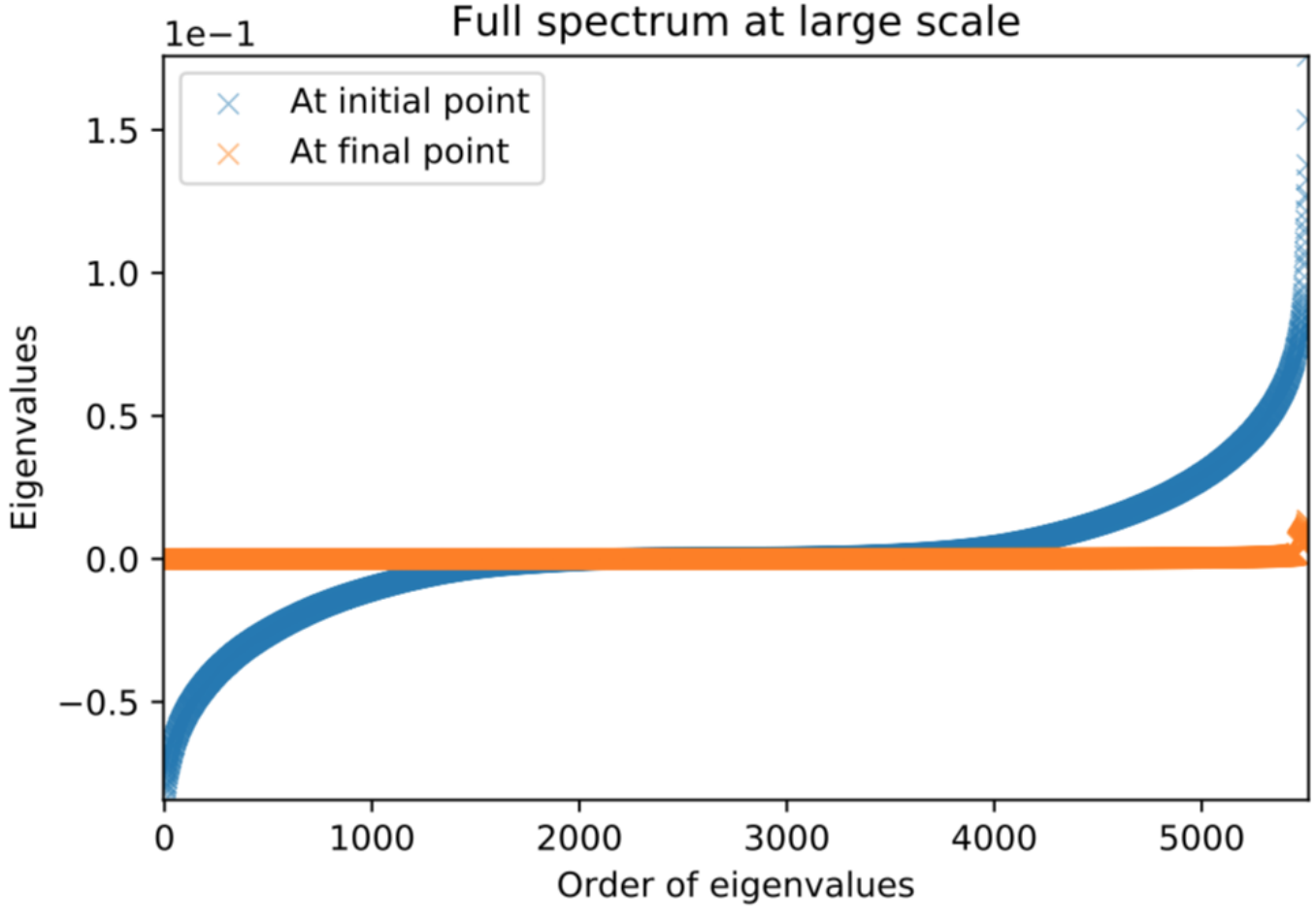
$$\begin{aligned} S_\Lambda(R) &\equiv C(\Lambda)e^{-R\Lambda^{-2}R} \\ C(\Lambda) &\equiv \pi^{-d/2}Det^{-1}(\Lambda) \end{aligned} \tag{3}$$

where d is the total dimensionality of R. The function f included in (2) allows for non-linear averaging. Two choices motivated by physical considerations are $f(x) = x$ and $f(x) = e^{-x/k_BT}$. These choices correspond respectively to the "diffusion equation" and "effective energy" methods which are described below. Wu [77] has presented a general discussion of transformations of the form of (2).

A highly smoothed $\tilde{H}_{\Lambda,f}$ (from which all high spatial-frequency components have been removed) will in most cases have fewer local minima than the unsmoothed ("bare") function, so it will be much easier to identify its global minimum. If the strong spatial-scaling hypothesis is correct, the position of this minimum can then be iteratively tracked by local-minimization as Λ decreases. As $\Lambda \rightarrow 0$, the position will approach the global minimizer of the bare objective function.

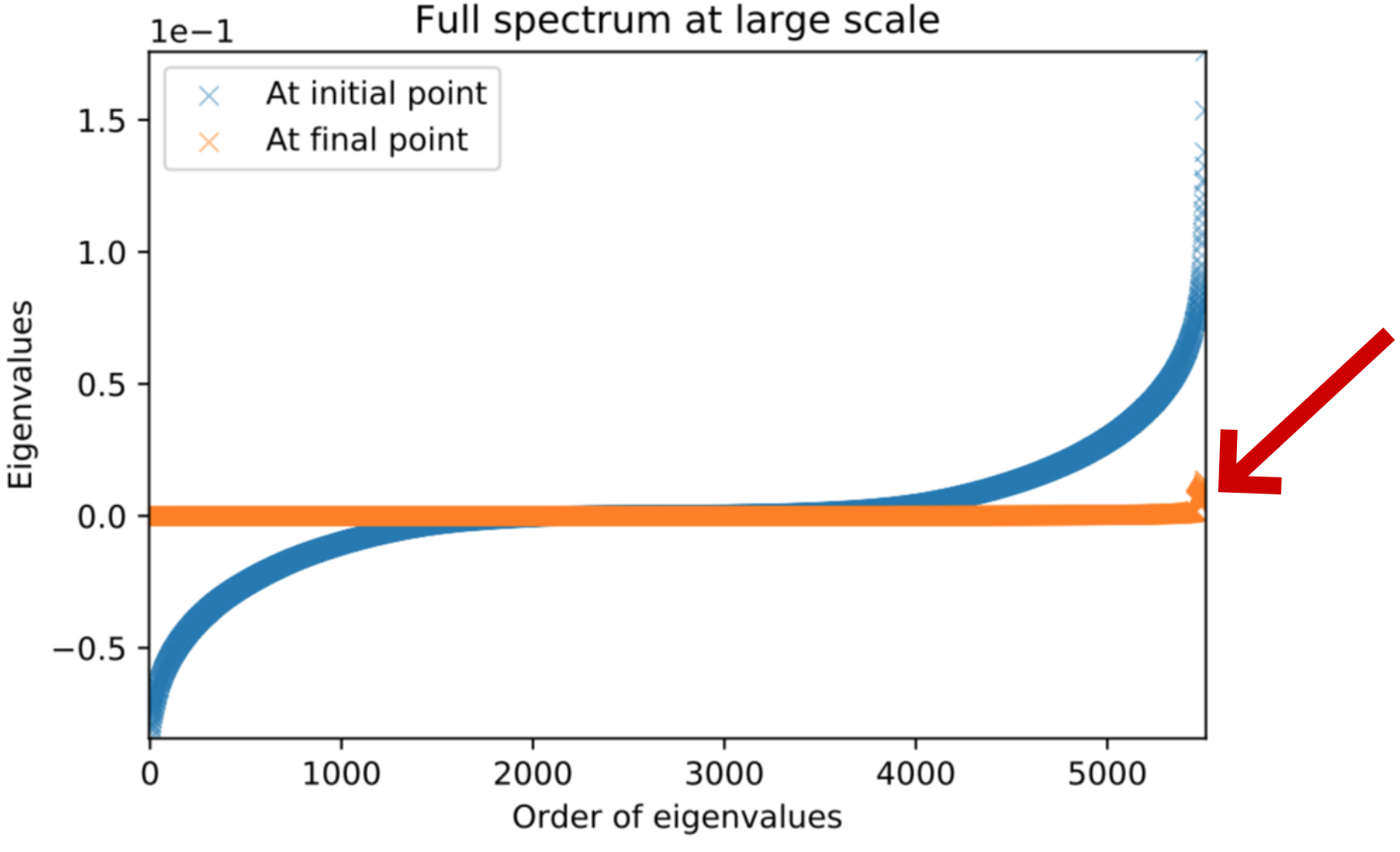
A look through the local curvature

Eigenvalues of the Hessian at the beginning and at the end



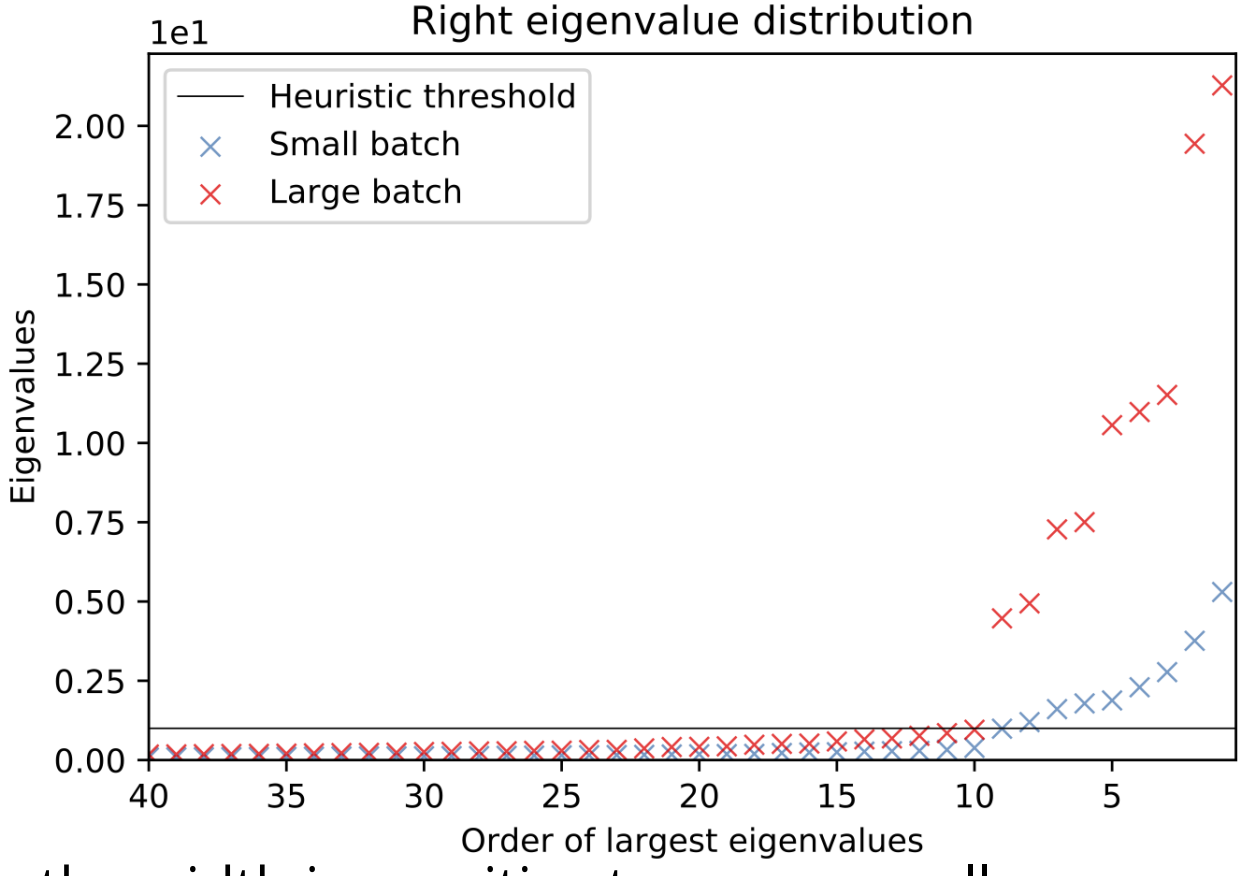
A look through the local curvature

Eigenvalues of the Hessian at the beginning and at the end



A look through the local curvature

Increasing the batch-size leads to larger outlier eigenvalues:

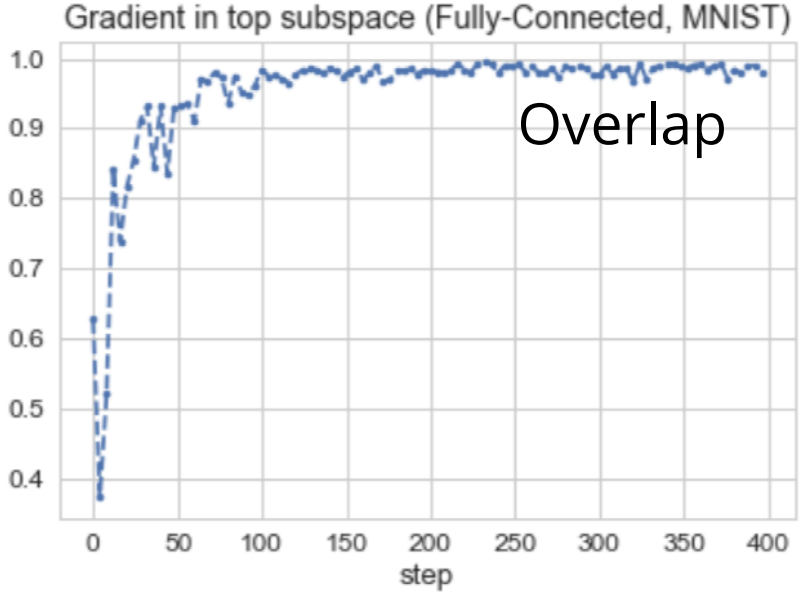


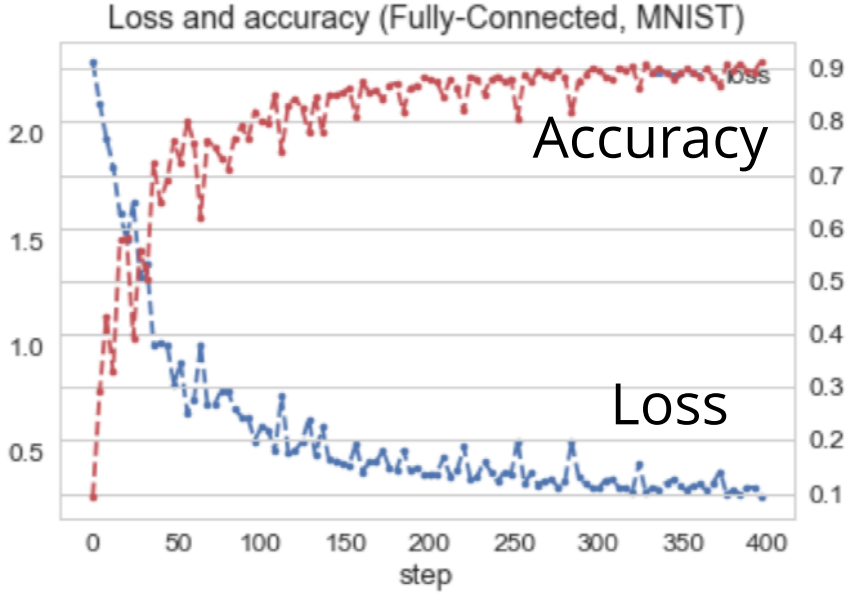
$\rightarrow$ the width is sensitive to a very small space only

$\rightarrow$ small chance for barriers in such a flat landscape

Gradients live in the top eigenspace

Gur-Ari, Roberts, Dyer 2018





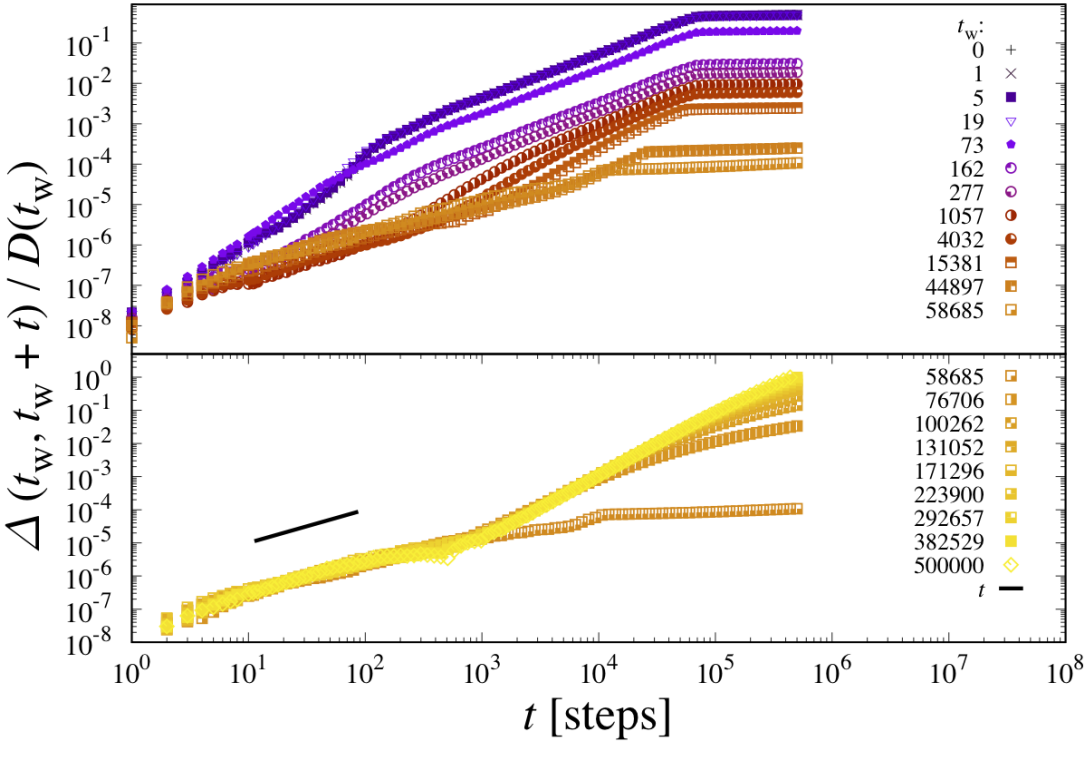
Attempt to understand this analytically: Vladimir Kirilin

More on the lack of barriers

1. Freeman and Bruna 2017: barriers of order $1/N$
2. Baity-Jesi et. al. 2018: no barrier crossing in SGD dynamics
3. Xing et. al. 2018: no barrier crossing in SGD dynamics
4. Garipov et. al. 2018: no barriers between solutions
5. Draxler et. al. 2018: no barriers between solutions

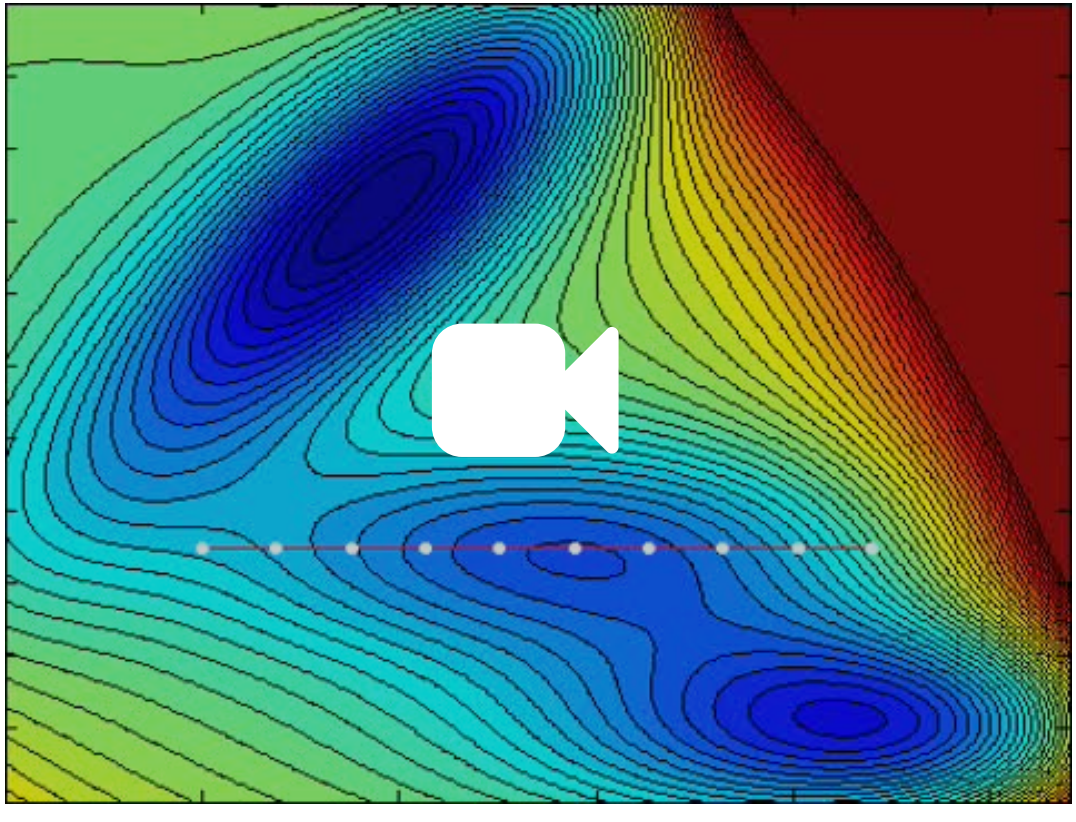
More on the lack of barriers

1. Freeman and Bruna 2017: barriers of order $1/N$
2. Baity-Jesi et. al. 2018: no barrier crossing in SGD dynamics
3. Xing et. al. 2018: no barrier crossing in SGD dynamics
4. Garipov et. al. 2018: no barriers between solutions
5. Draxler et. al. 2018: no barriers between solutions



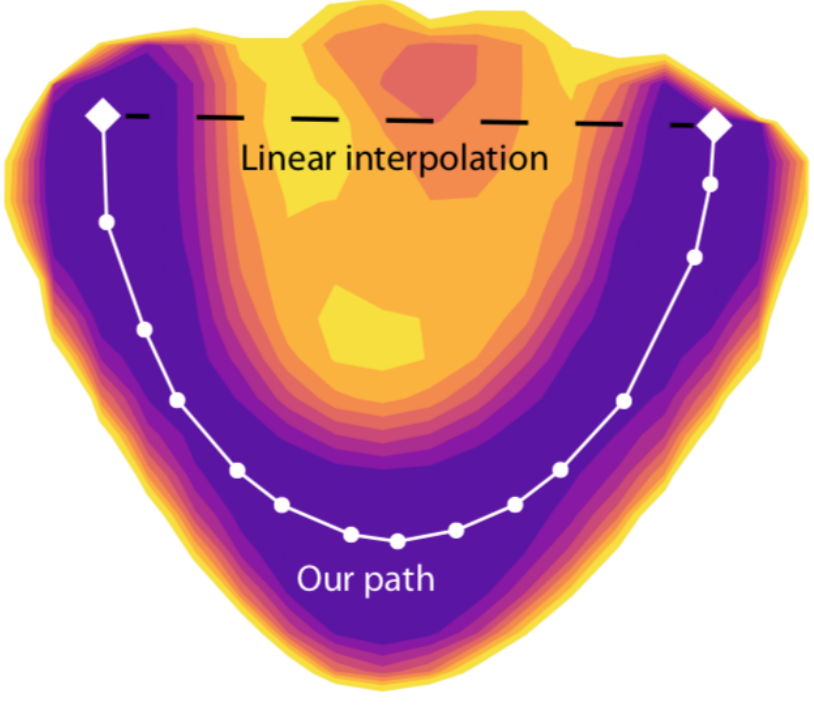
More on the lack of barriers

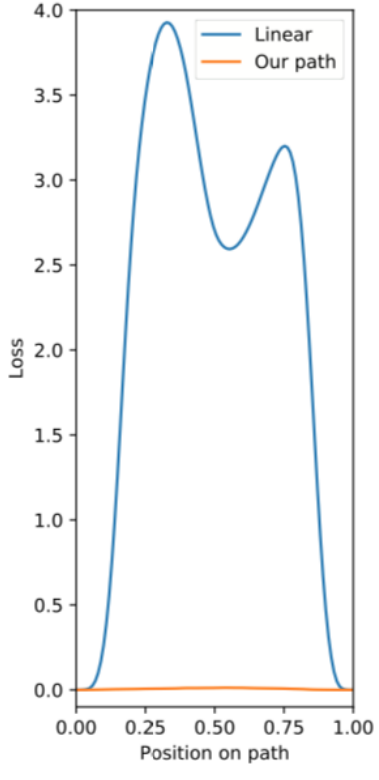
1. Freeman and Bruna 2017: barriers of order $1/N$
2. Baity-Jesi et. al. 2018: no barrier crossing in SGD dynamics
3. Xing et. al. 2018: no barrier crossing in SGD dynamics
4. Garipov et. al. 2018: no barriers between solutions
5. Draxler et. al. 2018: no barriers between solutions



More on the lack of barriers

1. Freeman and Bruna 2017: barriers of order $1/N$
2. Baity-Jesi et. al. 2018: no barrier crossing in SGD dynamics
3. Xing et. al. 2018: no barrier crossing in SGD dynamics
4. Garipov et. al. 2018: no barriers between solutions
5. Draxler et. al. 2018: no barriers between solutions



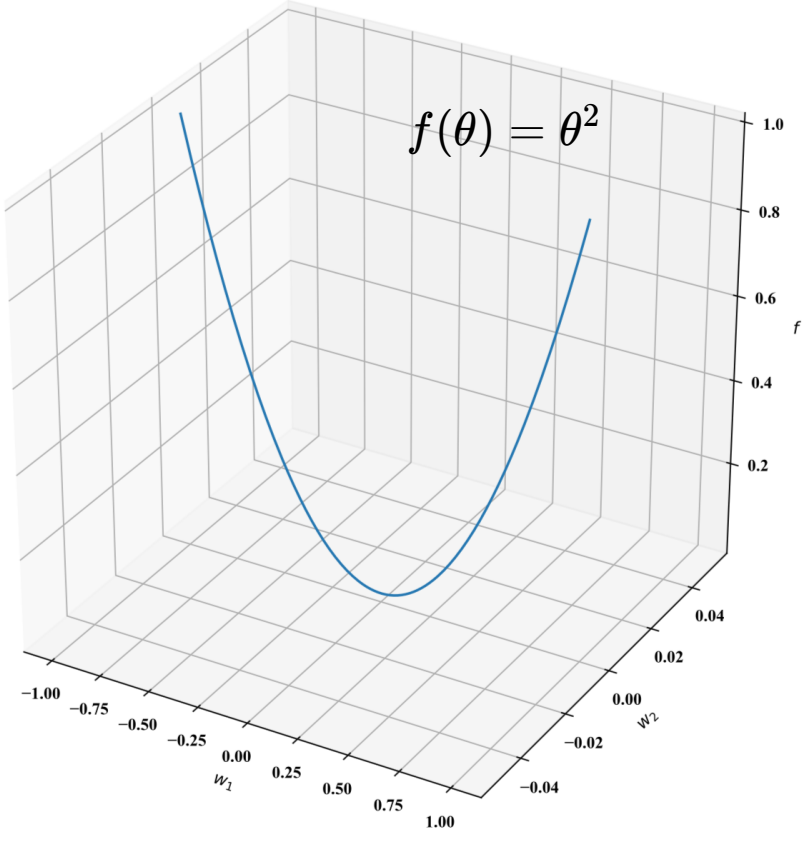


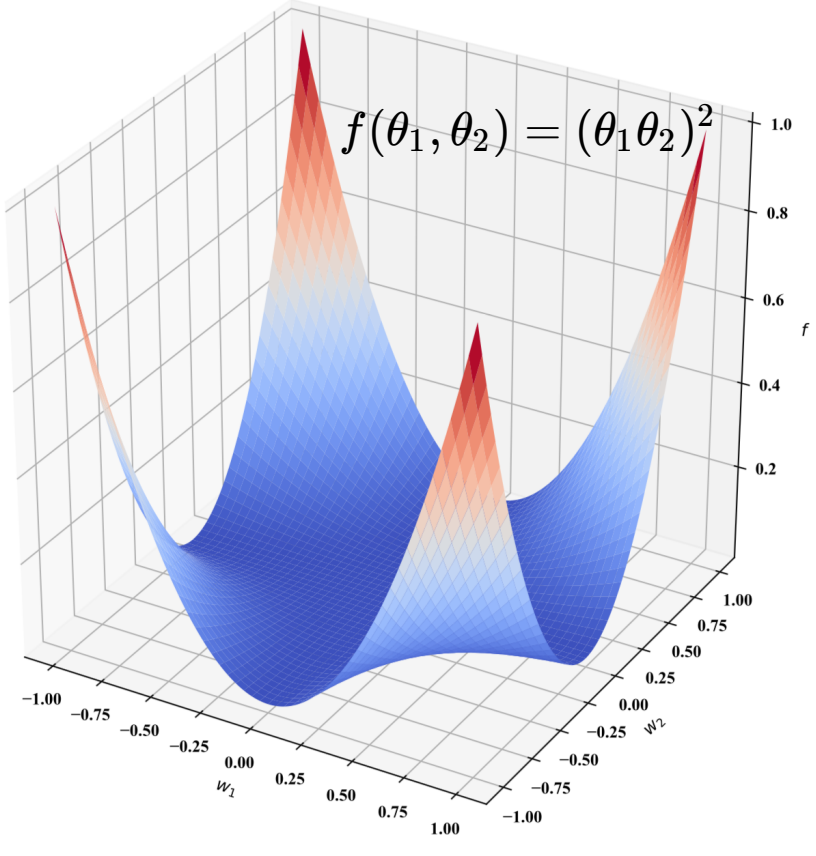
A toy model

Lessons from observations

Observation 1: captures wide/sharp discussion

Observation 2: captures flatness





Defining over-parametrization

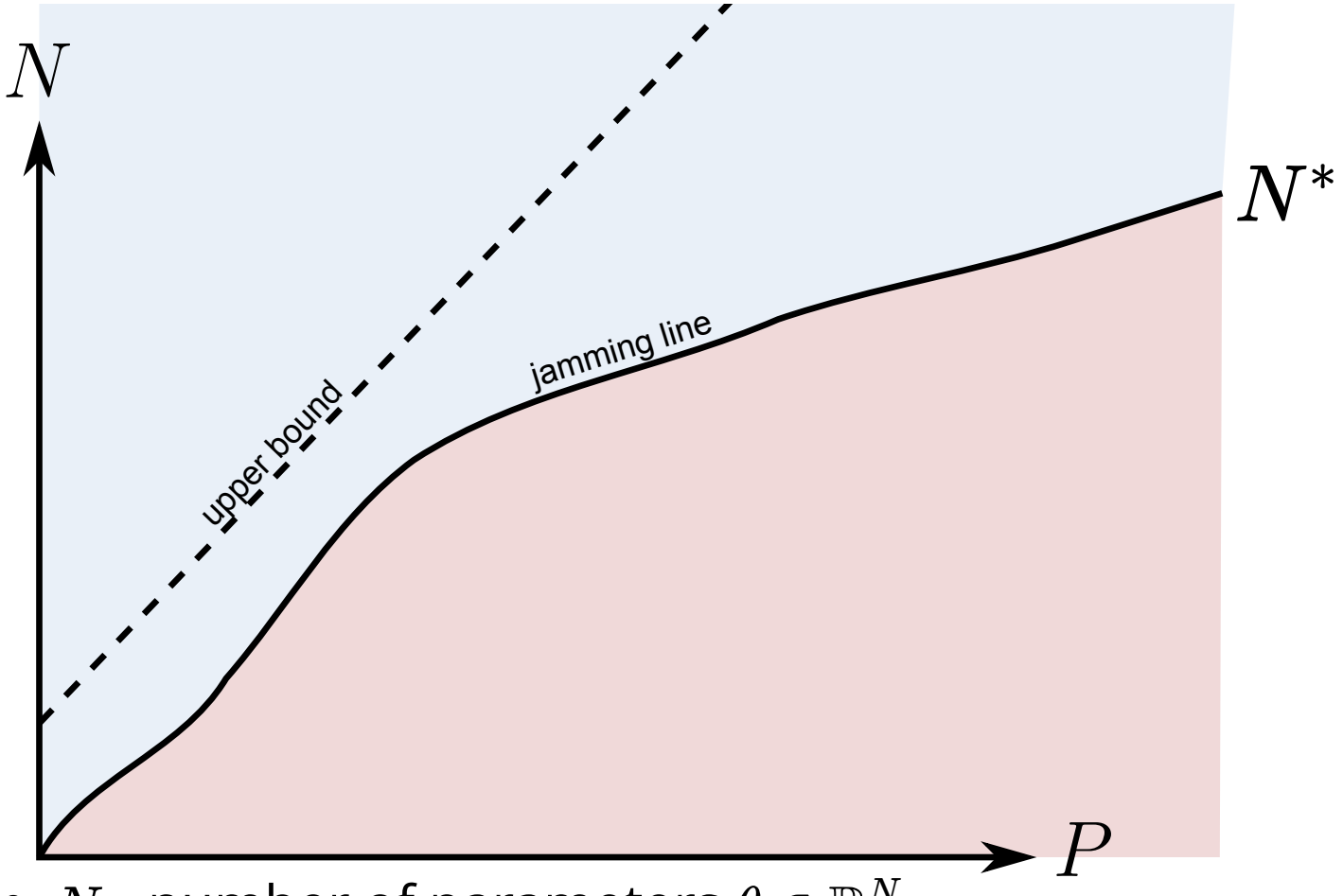
Puzzles with partial answers

1. For large N the dynamics don't get stuck
   $\rightarrow$ When is the training landscape *nice*?

2. Often $N >> P$, yet it doesn't it overfit
   $\rightarrow$ Relationship of the landscape with generalization?

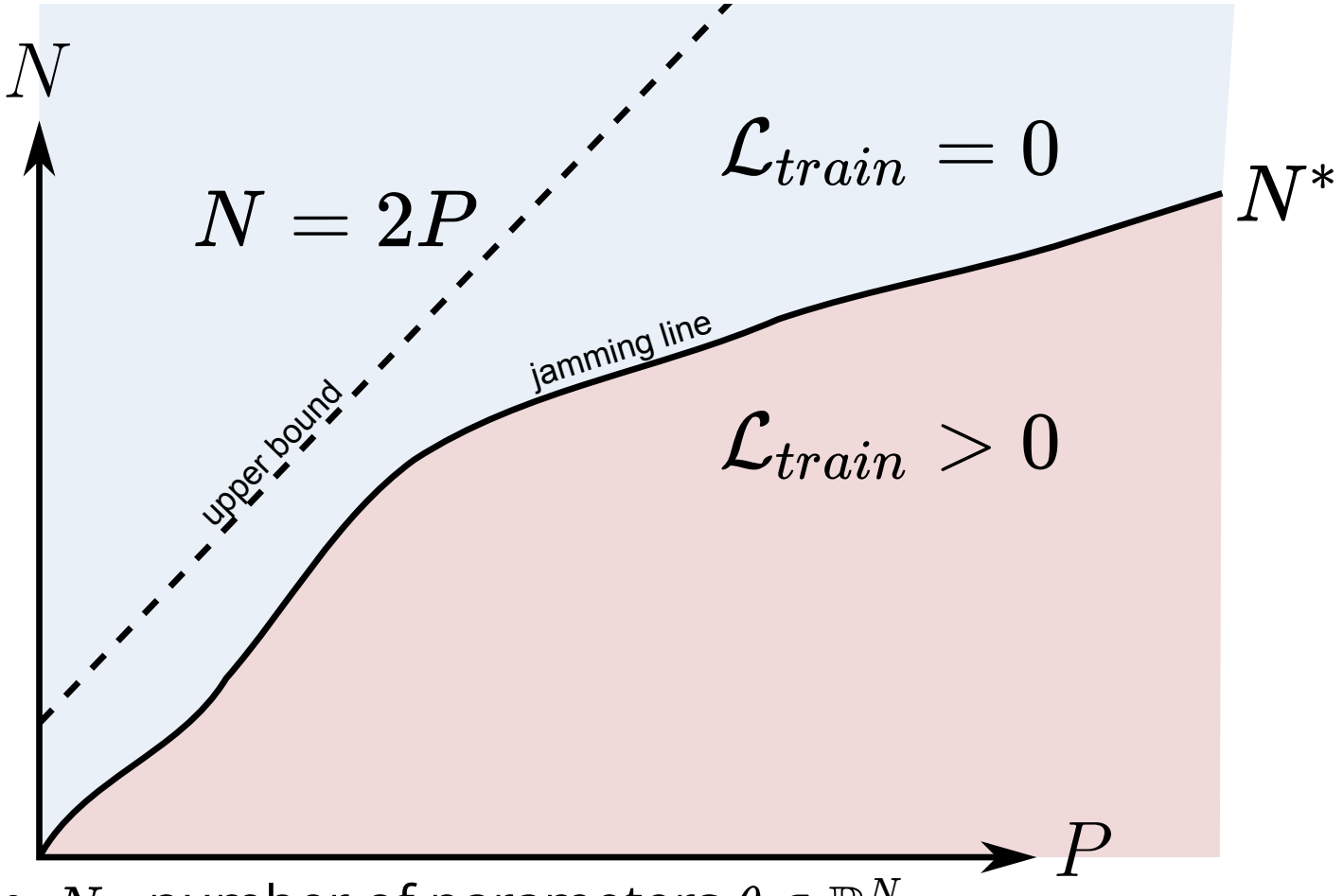
- N : number of parameters $\theta \in \mathbb{R}^N$
- P : number of examples in the *training* set $|\mathcal{D}_{train}|$

Sharp transition to OP in NNs



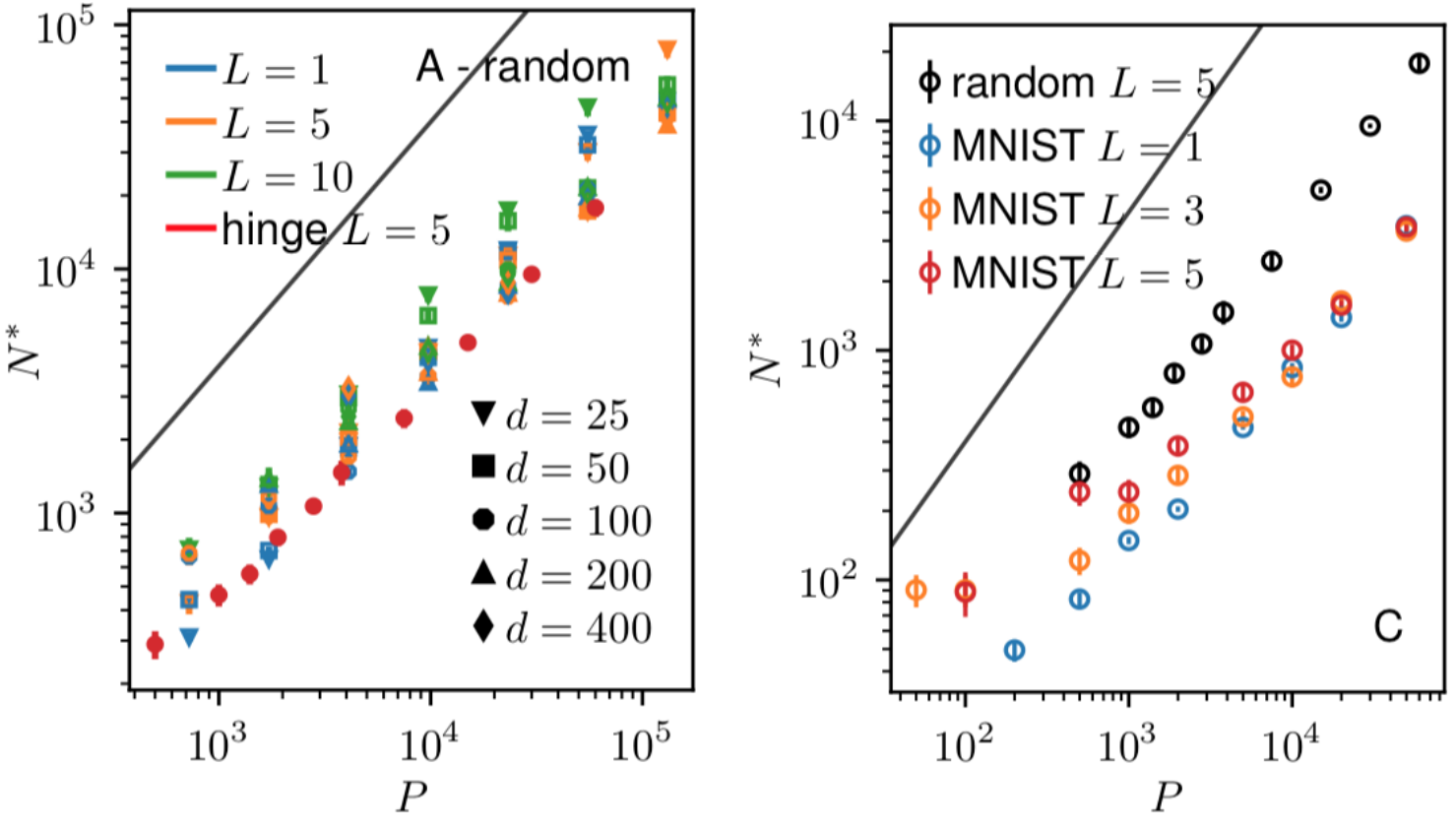
- N : number of parameters $\theta \in \mathbb{R}^N$
- P : number of examples in the *training* set $|\mathcal{D}_{train}|$
- N^* : critical number of parameters that fits $\mathcal{D}_{train}$

Sharp transition to OP in NNs



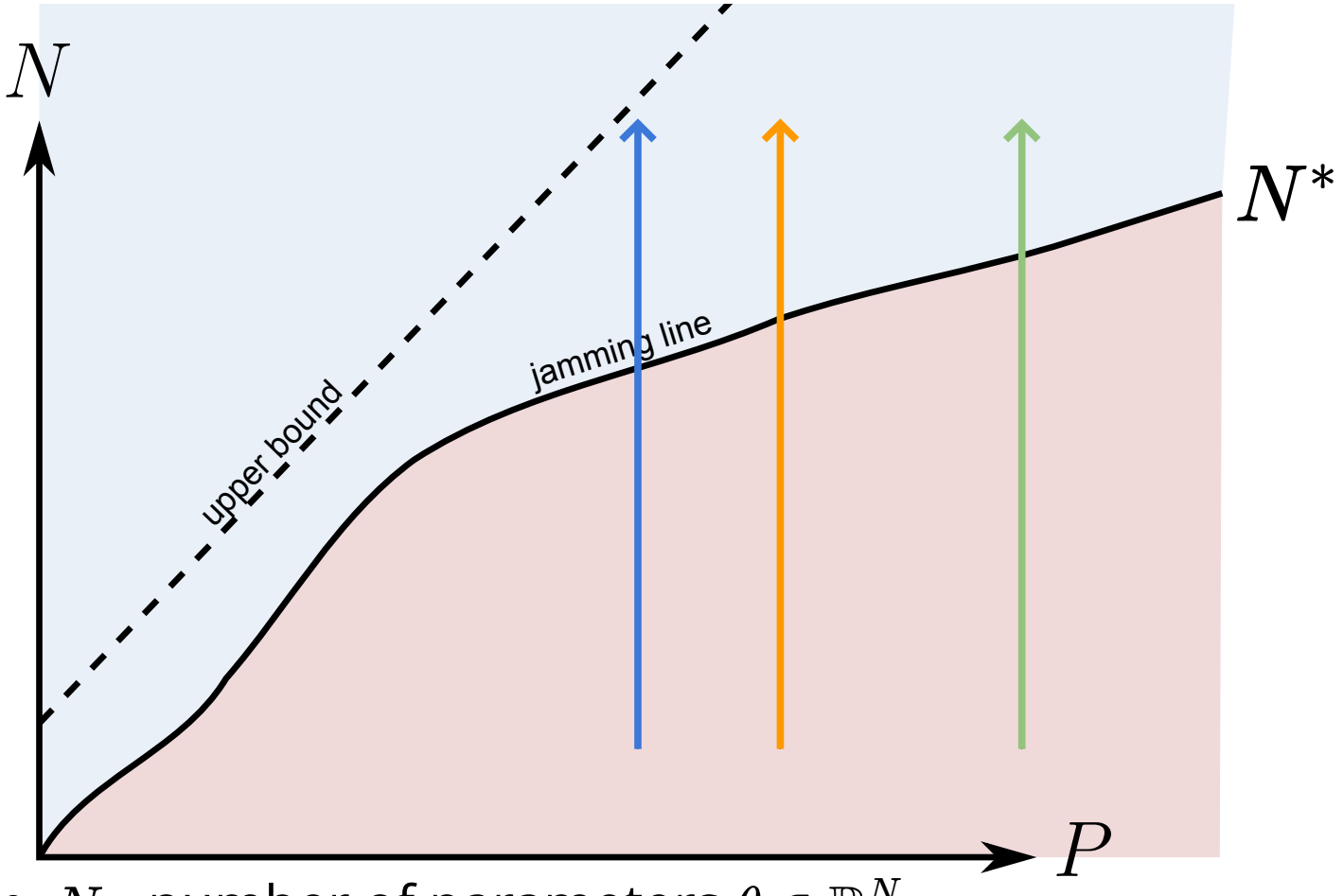
- N : number of parameters $\theta \in \mathbb{R}^N$
- P : number of examples in the *training* set $|\mathcal{D}_{train}|$
- N^* : critical number of parameters that fits $\mathcal{D}_{train}$

Sharp transition to OP in NNs



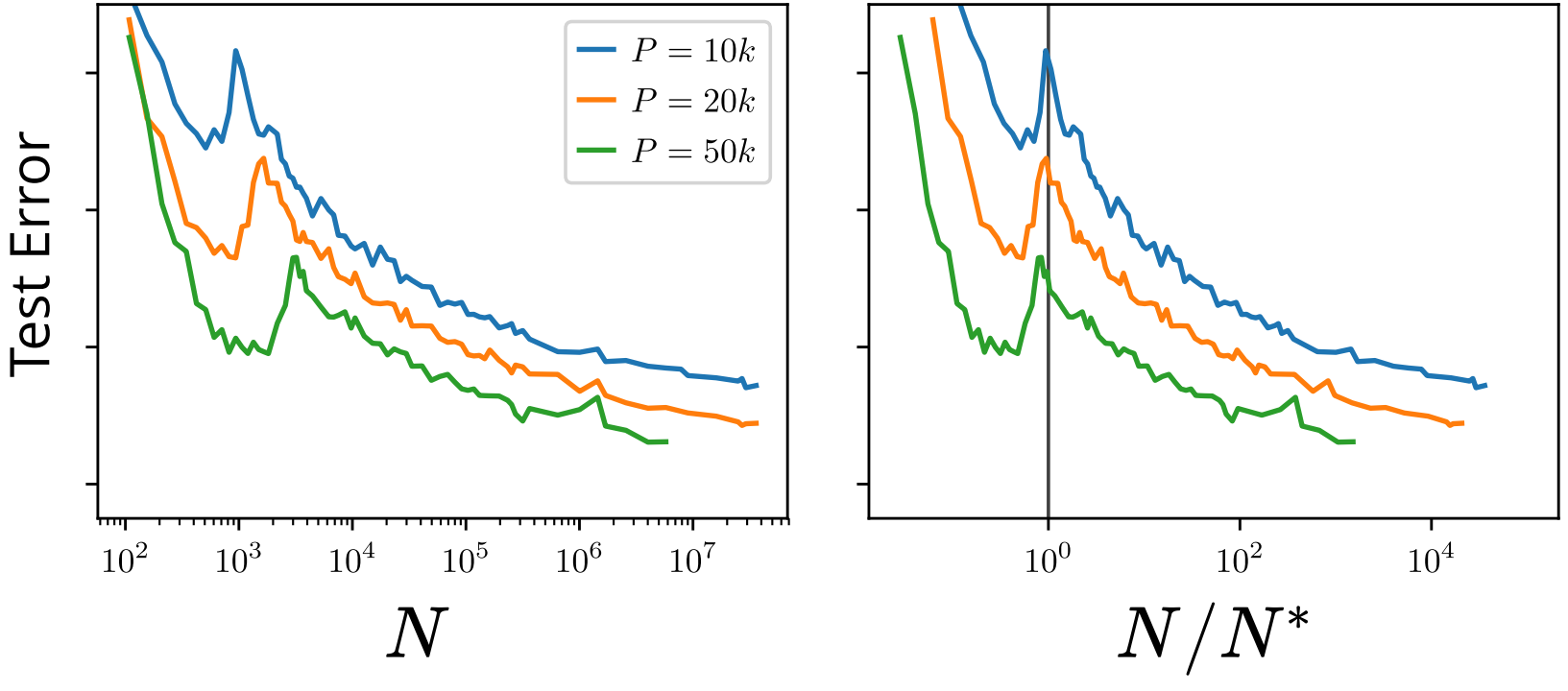
- N : number of parameters $\theta \in \mathbb{R}^N$
- P : number of examples in the *training* set $|\mathcal{D}_{train}|$
- N^* : critical number of parameters that fits $\mathcal{D}_{train}$

Sharp transition to OP in NNs



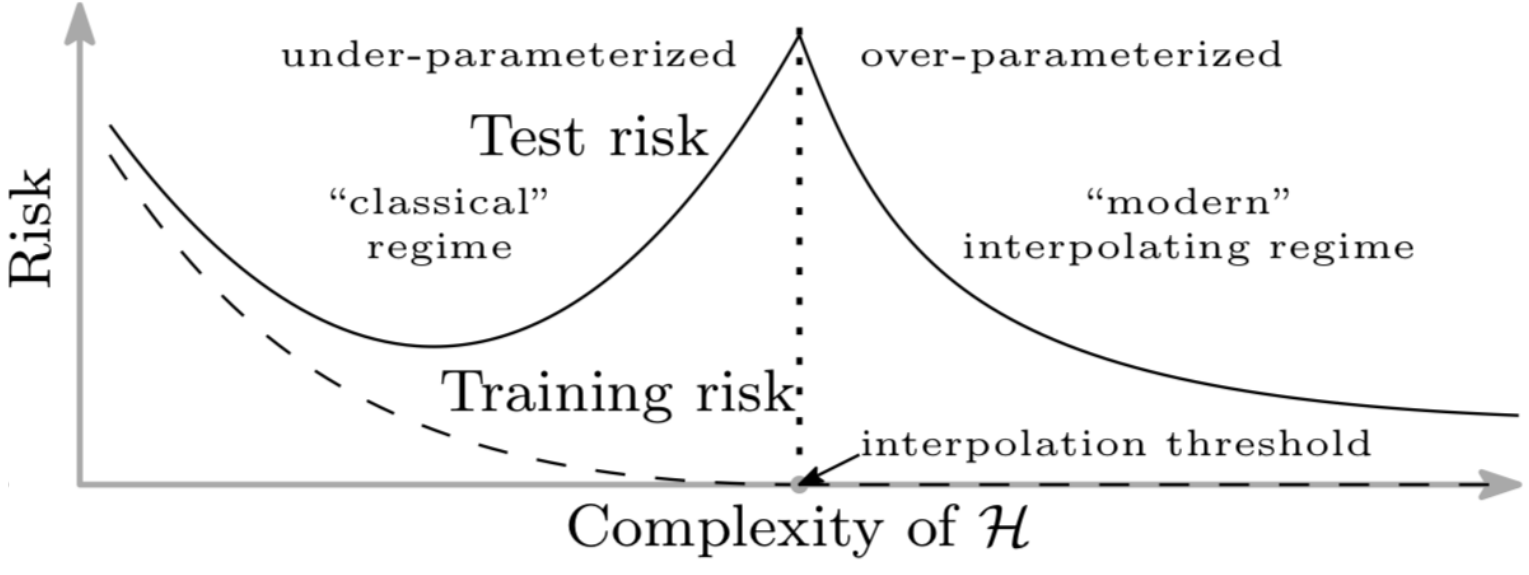
- N : number of parameters $\theta \in \mathbb{R}^N$
- P : number of examples in the *training* set $|\mathcal{D}_{train}|$
- N^* : critical number of parameters that fits $\mathcal{D}_{train}$

Jamming is linked to Generalization



Jamming is linked to Generalization

Belkin et. al. December 31, 2018



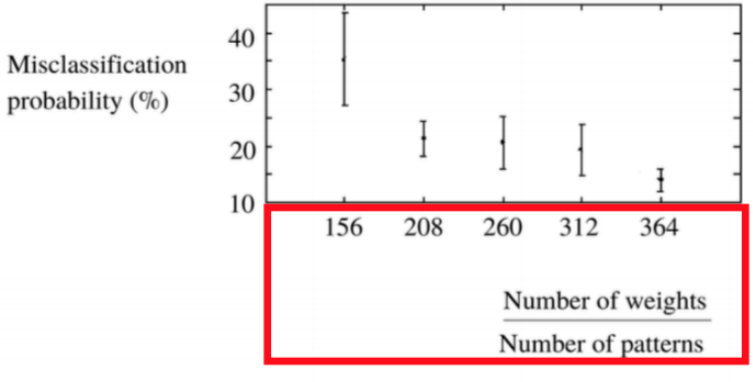
Jamming is linked to Generalization

Similar observations 20 years ago

Neural networks with many parameters, trained on small data sets, sometimes generalize well.

Eg: Face recognition (Lawrence *et al*, 1996)

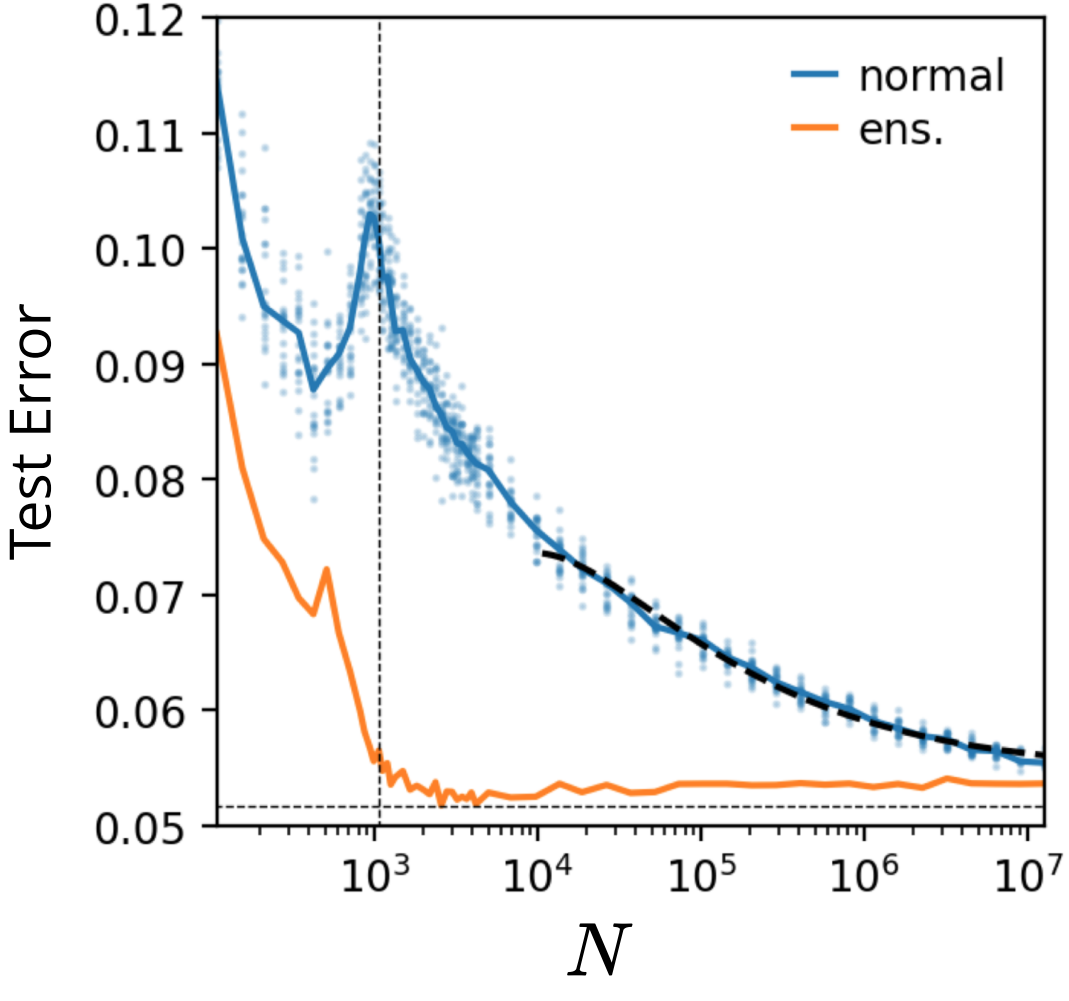
$m = 50$ training patterns.



Behnam Neyshabur's slide - also see Neyshabur et. al. 2018

Ensembling improves generalization

Key is *reducing fluctuations*



Open Questions (with partial answers)

- What controls the dynamics of SGD?
- How is it linked to generalization?
- Is the problem essentially convex?
- What's the role of the algorithm?
- What's the role of priors on performance?
- Is ensembling after jamming the best one can get?

Thank You!