Games for Computation and Learning. Kernel Flows: from learning kernels from data into the abyss

Houman Owhadi

AFOSR, August 15, 2018

DARPA EQUiPS / AFOSR award no FA9550-16-1-0054
Computational Information Games, 2015-2018

AFOSR. Grant number FA9550-18-1-0271.
Games for Computation and Learning, 2018-2021.

Team

Houman Owhadi, PI

Caltech Prof. of Computing and Mathematical Sciences
Multiscale analysis, Game Theory, Probability Theory, Uncertainty Quantification. PSAAP, AFOSR, ExMatEx.

Clint Scovel

Caltech research associate. Machine Learning. Uncertainty Quantification. Former LANL senior scientist. PSAAP, AFOSR.

Florian Schäfer

Caltech graduate student (third year). Compression, inversion and approximate PCA of dense kernel matrices. Universal Solvers.

Gene Ryan Yoo

Caltech graduate student (first year). PDE denoising. Learning kernels from data and Kernel Flows.

Collaborators

Peter Schröder

Joel Tropp

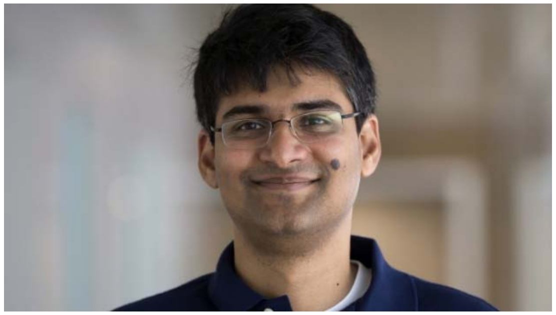
Mathieu Desbrun

Max Budninskiy

Lei Zhang

Tim Sullivan

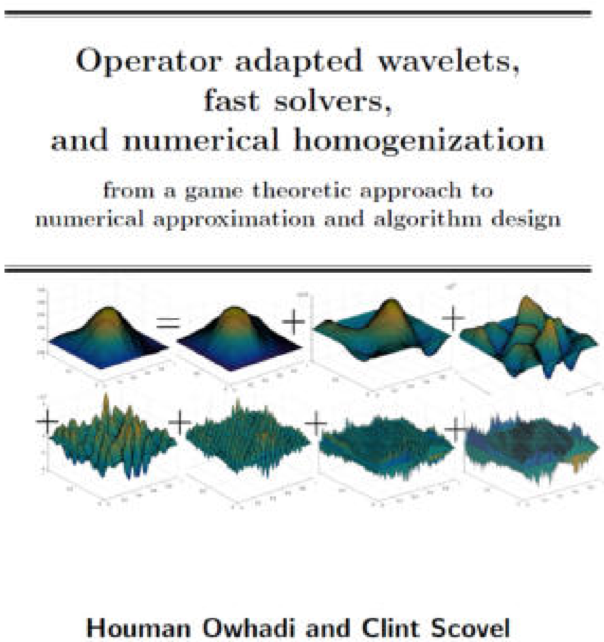
Animashree Anandkumar



Vikram Gavini

Phani Motamarri

Publications



Journal

- Kernel Flows: from learning kernels from data into the abyss. H. Owhadi and G. R. Yoo, arXiv:1808.04475, 2018.
- Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity, arXiv:1706.02205, 2017. Schäfer, Sullivan, Owhadi.
- De-noising by thresholding operator adapted wavelets. G. R. Yoo and H. Owhadi, 2018 [arXiv:1805.10736]. To appear in Statistics and Computing.
- Fast eigenpairs computation with operator adapted wavelets and hierarchical subspace correction. H. Xie, L. Zhang and H. Owhadi, 2018. [arXiv:1806.00565]
- Universal Scalable Robust Solvers from Computational Information Games and fast eigenspace adapted Multiresolution Analysis, 2017. arXiv:1703.10761. H. Owhadi and C. Scovel.
- Gamblets for opening the complexity-bottleneck of implicit schemes for hyperbolic and parabolic ODEs/PDEs with rough coefficients. arXiv:1606.07686. H. Owhadi and L. Zhang. Journal of Computational Physics, Volume 347, pages 99-128, 2017.
- Multigrid with rough coefficients and Multiresolution operator decomposition from Hierarchical Information Games. H. Owhadi. SIAM Review, 59(1), 99149, 2017. arXiv:1503.03467
- Bayesian Numerical Homogenization. H. Owhadi. SIAM Multiscale Modeling & Simulation, 13(3), 812828, 2015. arXiv:1406.6668

Book

- Operator adapted wavelets, fast solvers, and numerical homogenization from a game theoretic approach to numerical approximation and algorithm design. H. Owhadi and C. Scovel, 2018. Under contract to appear in **Cambridge Monographs on Applied and Computational Mathematics**

DARPA EQUiPS / AFOSR award no FA9550-16-1-0054
(Computational Information Games)

Play Games → Solve Comp. Math Problems

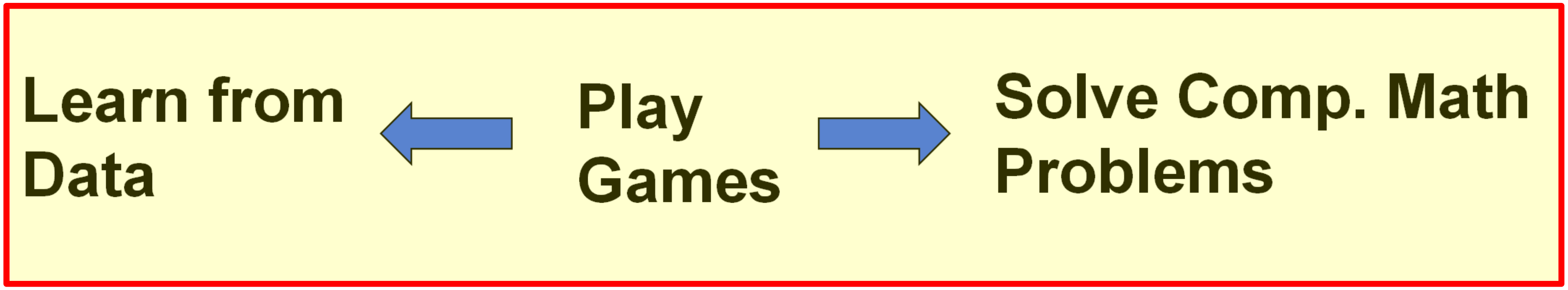
Interplays between Game Theory and Numerical Approximation

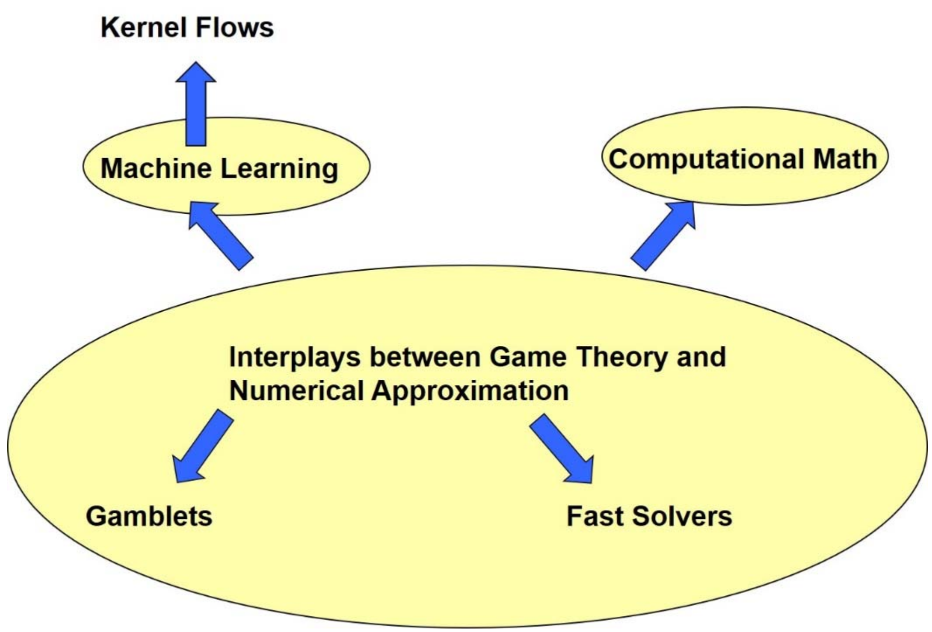
Gamblets (operator adapted wavelets)

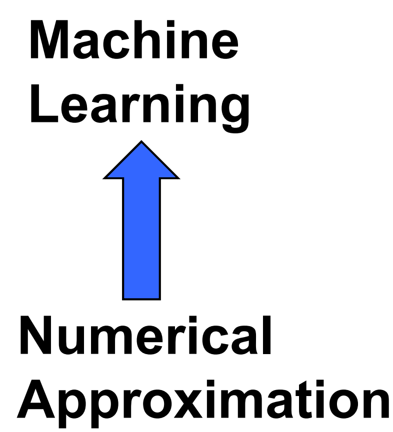
Fast Solvers

Machine Learning → Numerical Approximation

AFOSR. Grant number FA9550-18-1-0271.
Games for Computation and Learning, 2018-2021.



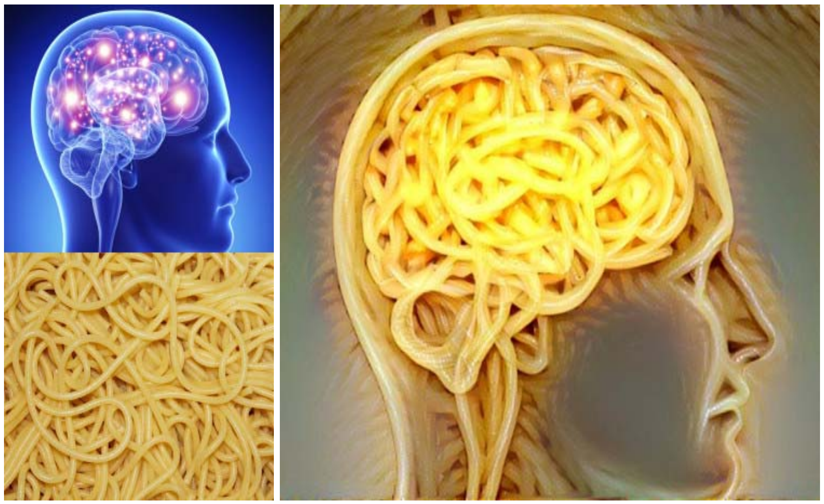




Deep Learning

Impressive results

https://deepart.io/
https://deepdreamgenerator.com/



A Neural Algorithm of Artistic Style, Gatys et al, 2015

BUT **It is "alchemy"**

- We don't know why algorithms work or why they don't (no theory)
- Algorithms are developed through trial and error
- Some results are hard to replicate (many hyperparameters)
- Finding good architectures relies on guesswork
- Very deep networks (more 40 layers) are difficult to train with backpropagation
- Algorithms are not robust to adversarial examples

AI researchers allege that machine learning is alchemy

By Matthew Hutson | May. 3, 2018 , 11:15 AM

Ali Rahimi, a researcher in artificial intelligence (AI) at Google in San Francisco, California, took a swipe at his field last December—and received a 40-second ovation for it. Speaking at an AI conference, Rahimi charged that machine learning algorithms, in which computers learn through trial and error, have become a form of "alchemy." Researchers, he said, do not know why some algorithms work and others don't, nor do they have rigorous criteria for choosing one AI architecture over another. Now, in a paper presented on 30 April at the International Conference on Learning Representations in Vancouver, Canada, Rahimi and his collaborators document examples of what they see as the alchemy problem and offer prescriptions for bolstering AI's rigor.

"There's an anguish in the field," Rahimi says. "Many of us feel like we're operating on an alien technology."

Science Mag, May 2018

"Machine learning has become alchemy"
Ali Rahimi
NIPS 2017 Test of Time Award

Questions

Can the interface between NA and Game theory offer some insights?

Is there an approach that

- **Is amenable to some degree of analysis?**
- **Produces a network without guesswork?**
  (plug and play, no tweaking of hyperparameters, no guessing of the architecture)
- **Enables the training of very deep networks?**
  (50,000 layers or more) and the exploration of their properties
- **Provides some insight on developing a rigorous theory for deep learning?**

Initial results

Interface between Game Theory and NA → **Deep Learning**

Gene Ryan Yoo

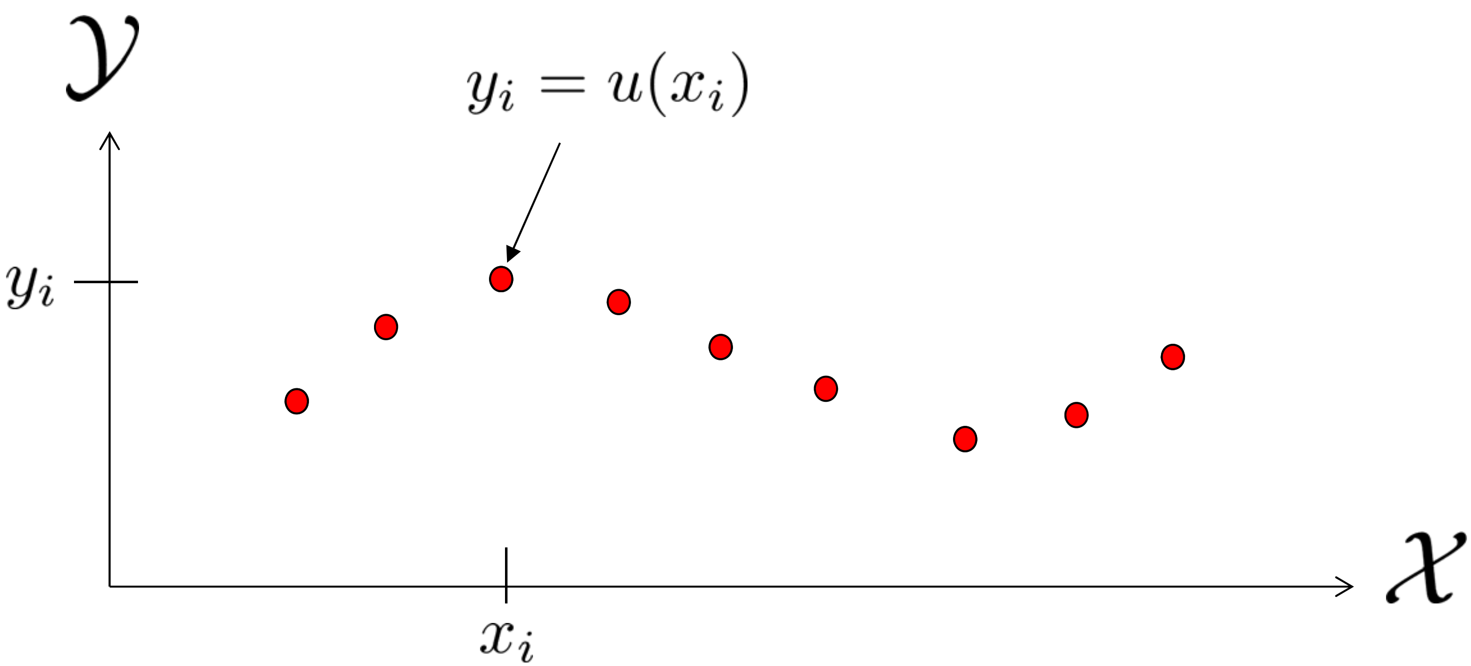
- Kernel Flows: from learning kernels from data into the abyss. H. Owhadi and G. R. Yoo, arXiv:1808.04475, 2018.

Learning is solving an interpolation problem

$$\mathcal{X} \xrightarrow{\quad u \quad} \mathcal{Y}$$

u : Unknown

Given $y_i = u(x_i)$ for $i = 1, \ldots, N$, approximate u

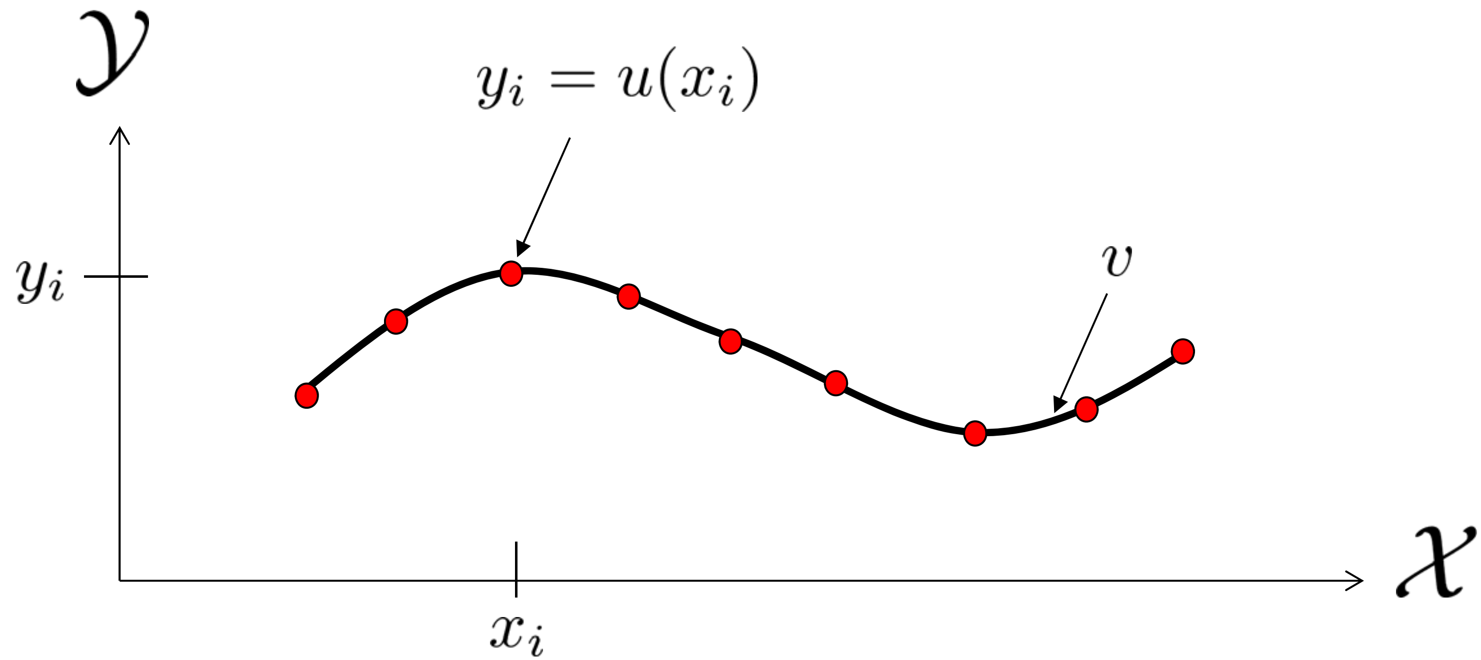


Solution: Kriging/GPR/SVM

Given kernel K approximate $u(x)$ with

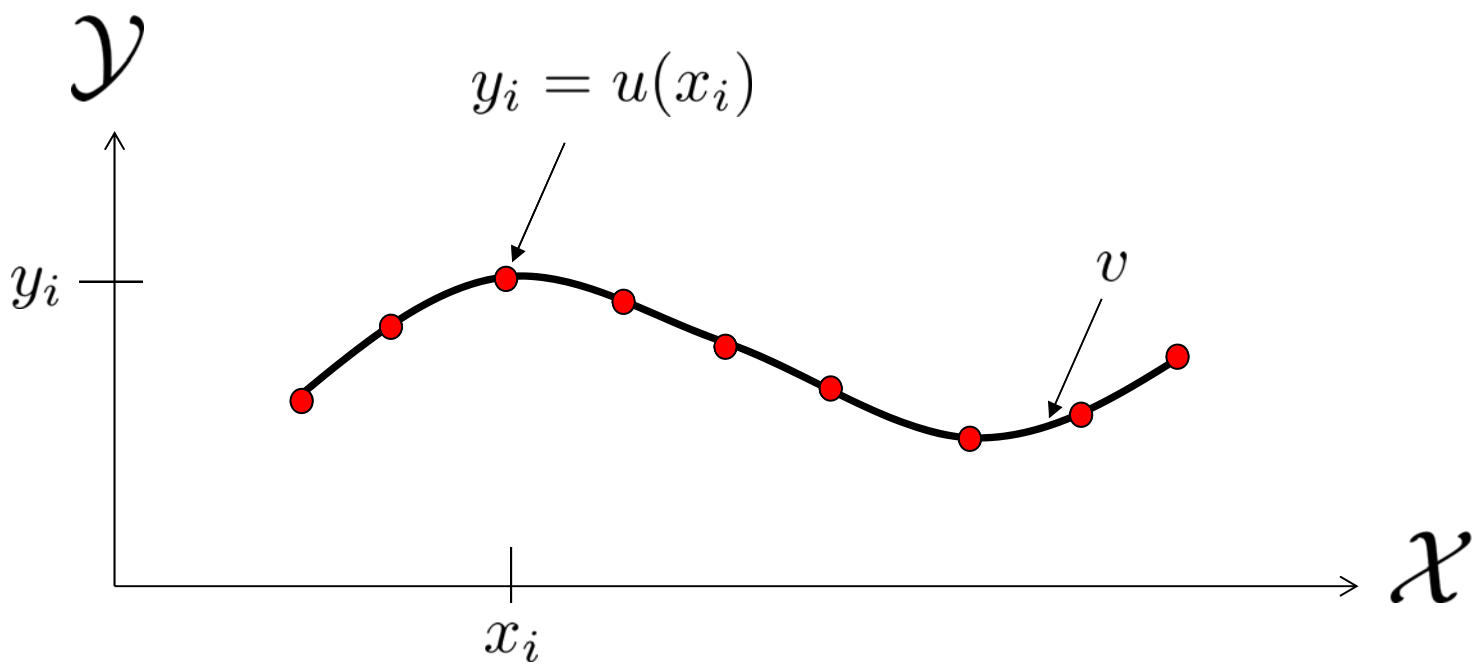
$$v(x) = \sum_i c_i K(x_i, x)$$

c such that $v(x_i) = y_i$ for all i



BUT

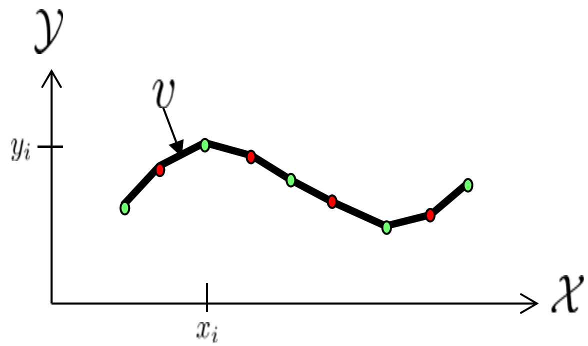
- **What if N is large?**
- **Which kernel do we pick?**



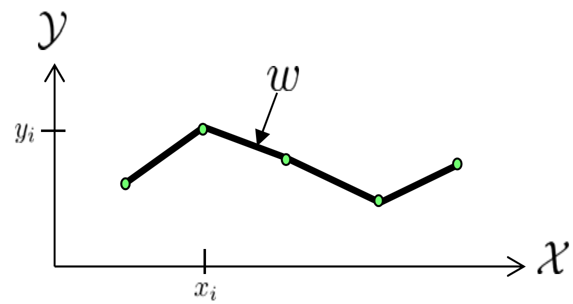
Premise

A kernel K is good if the number of interpolation points can be halved without significant loss in accuracy

v: Interpolate with K and N points



w: Interpolate with K and $N/2$ points



$$\rho = \frac{\|v - w\|^2}{\|v\|^2}$$

$$\|v\|^2 = \sup_\phi \frac{(\int \phi(x) v(x)\, dx)^2}{\int \phi(x) K(x,x') \phi(x')\, dx\, dx'}$$

Good kernel ⟺ Small ρ

Kernel Flow

Learns kernels of the form

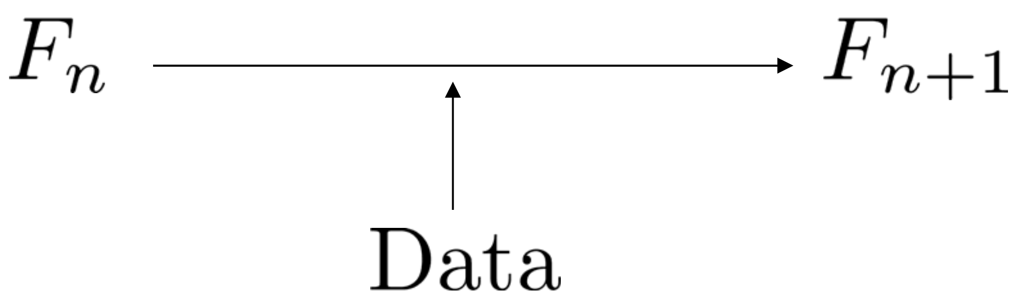
$$K_n(x, x') = K_1(F_n(x), F_n(x'))$$

K_1: kernel (e.g. $K_1(x, x') = e^{-\frac{|x-x'|^2}{\gamma^2}}$)

F_n : Flow in input space

$$F_n : \mathcal{X} \to \mathcal{X}$$

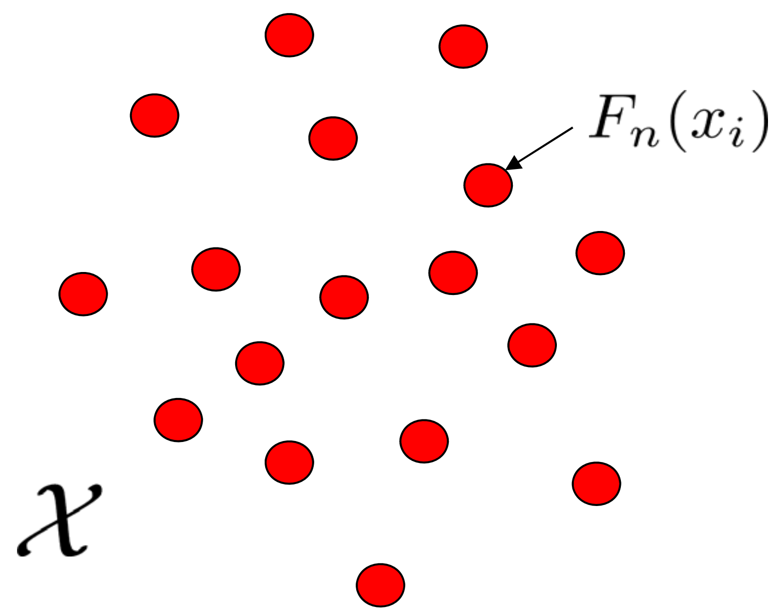
$$F_1 = I_d$$



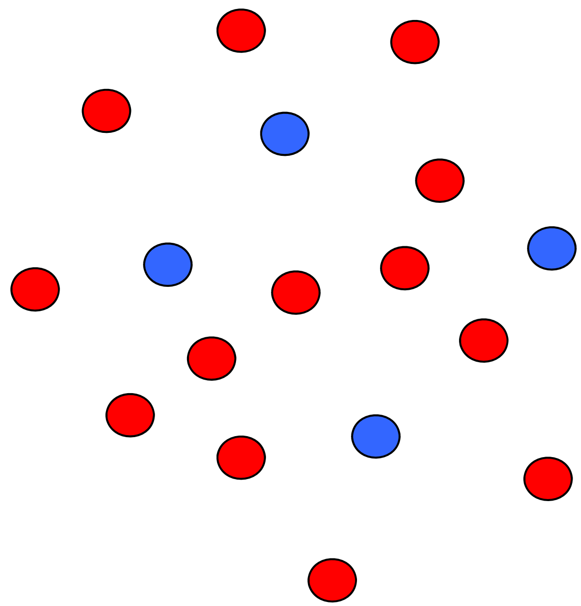
Step $n \to n+1$

Assume F_n known

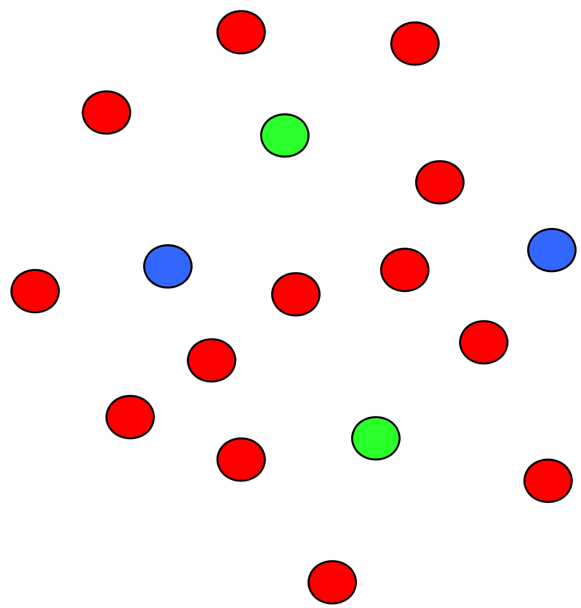
Images of the N training points under F_n



Select N_f at random out of N



Select $N_f/2$ at random out of N_f

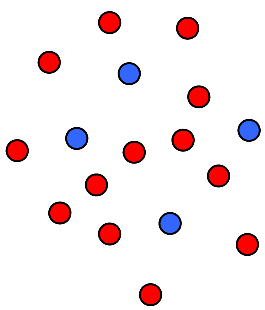


Player I

Selects the values/labels of the blue points $F_n(x_i)$ to be y_i (training labels)

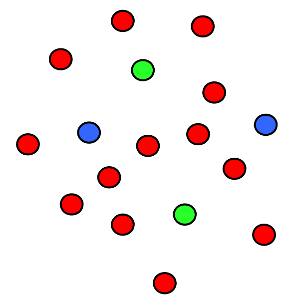
Player II

Sees values/labels y_i of the $N_c = N_f/2$ green points must predict the values of the blue points



Max

Min

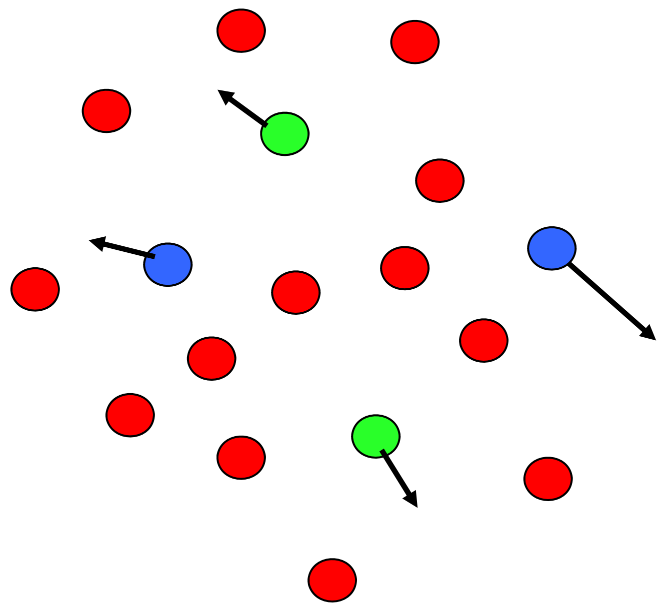


$$\rho$$

ρ: Relative error in $\|\cdot\|$ norm

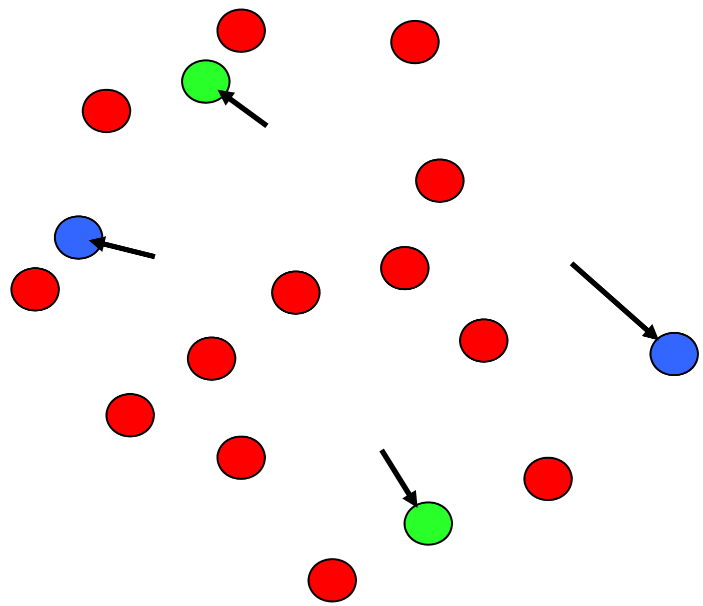
$\|\cdot\|$: RKHS norm associated with K_1

Move the N_f points in the gradient descent direction of ρ

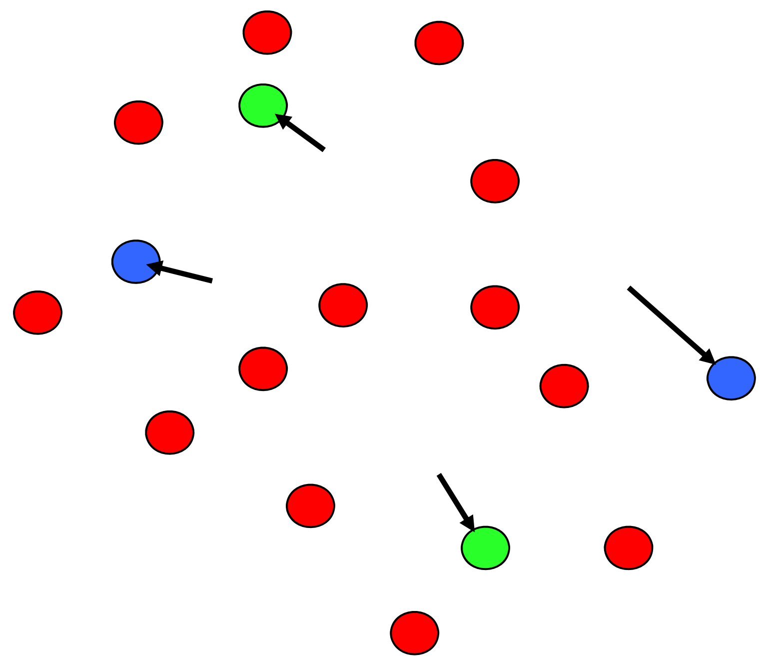


Rig the game in favor of Player II

Move the N_f points in the gradient descent direction of ρ



Move the remaining $N - N_f$ points
via interpolation with kernel K_1

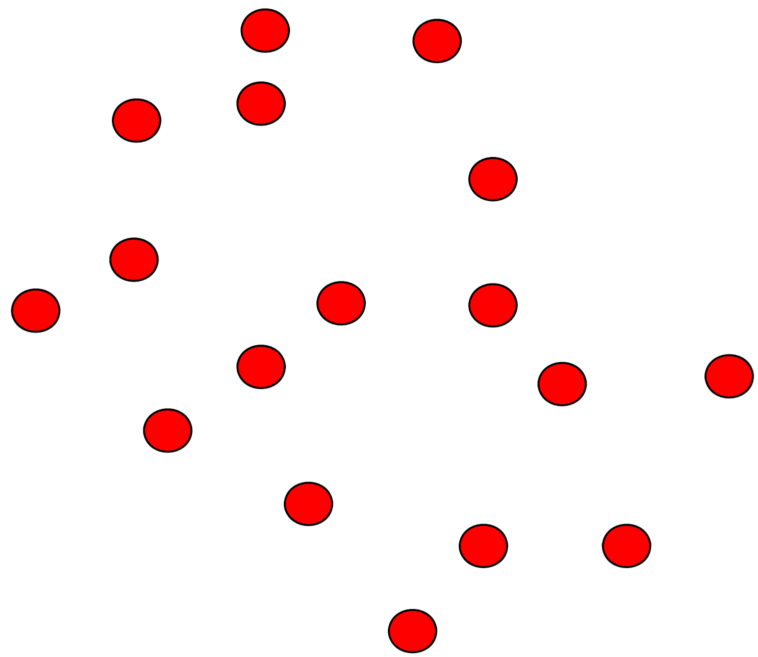


Move any point x
via interpolation with kernel K_1 ⟹ F_{n+1}

Repeat

F_{n+1} known

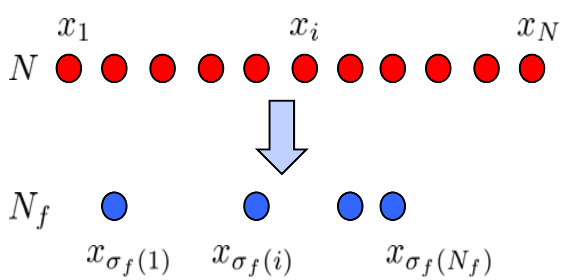
Images of the N training points under F_{n+1}

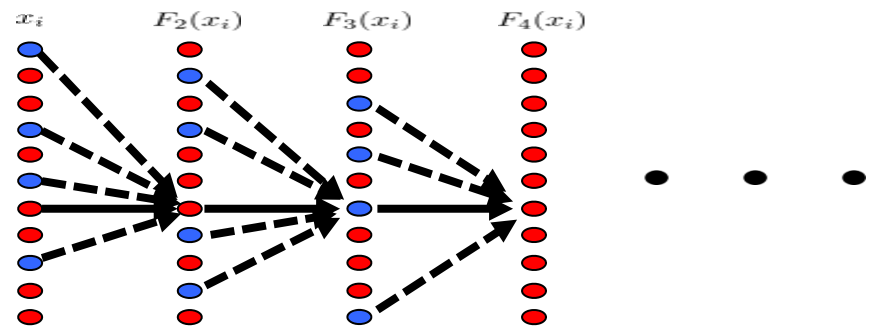


Kernel Flow

Produces a deep hierarchical kernel of the form

$$K_n(x,x') = K_{n-1}\big(x + \epsilon \sum_{i=1}^{N_f} c_i K_{n-1}(x_{\sigma_f(i)}, x), x' + \epsilon \sum_{i=1}^{N_f} c_i K_{n-1}(x_{\sigma_f(i)}, x')\big)$$





Randomized bottomless network

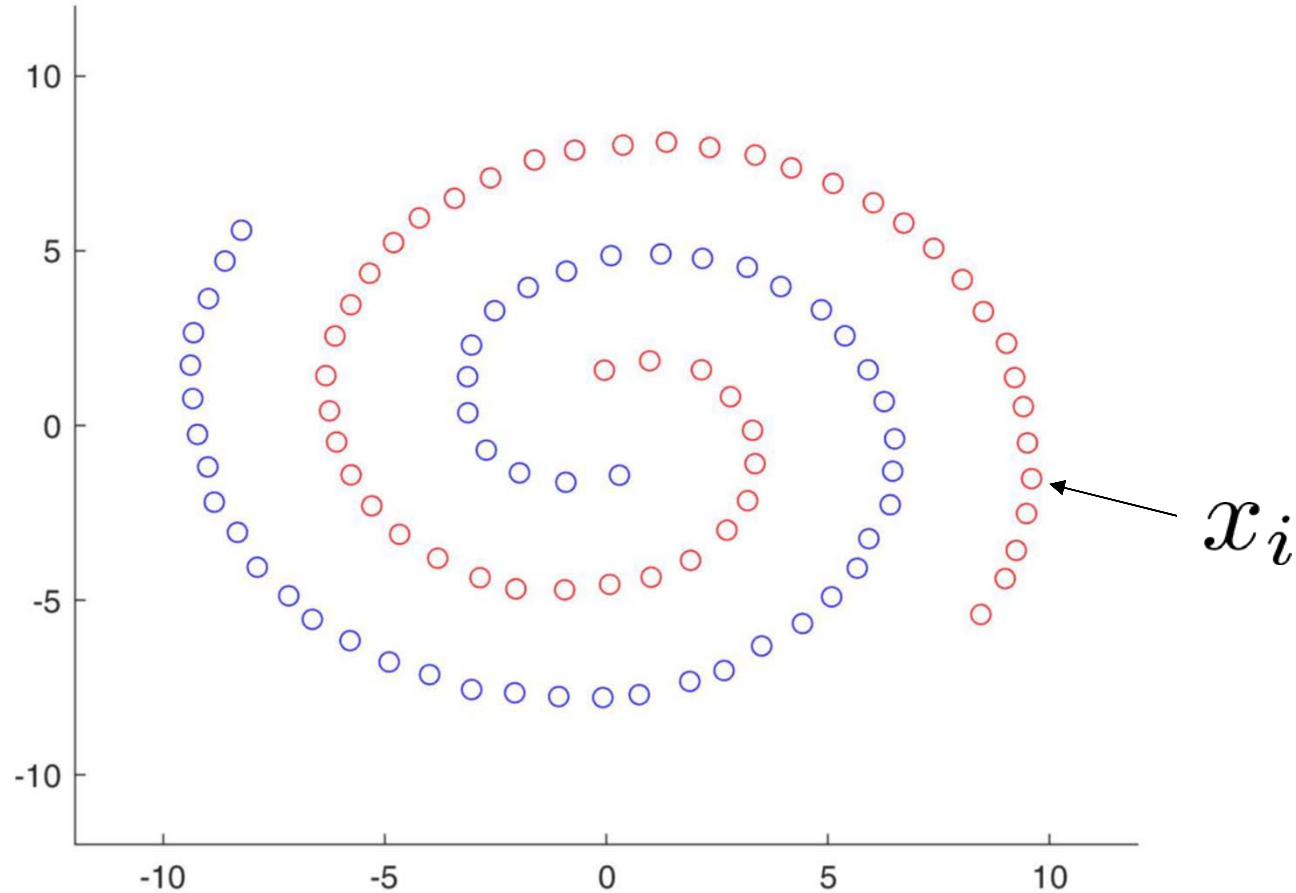
and a flow of the form

$$F_{n+1} = (I_d + \epsilon G_{n+1}) \circ F_n$$

$$G_{n+1}(x) = \sum_{i=1}^{N_f} c_i K_1(F_n(x_{\sigma_f(i)}), x)$$

Identified as the steepest gradient descent direction of ρ.

Application: Swiss Roll Cheesecake



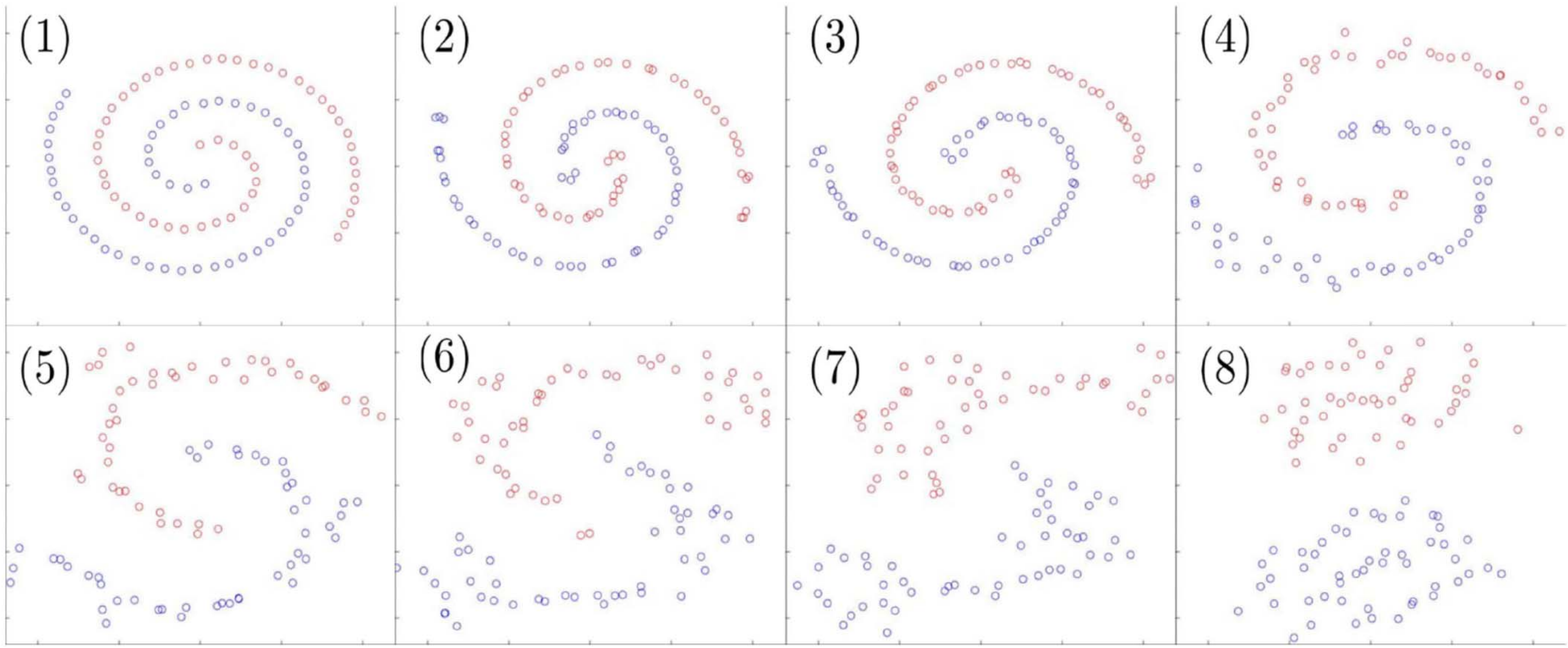
$N = 100$ data points $x_i \in \mathbb{R}^2$

$y_i = -1$ if point at x_i is red

$y_i = +1$ if point at x_i is blue

Objective:
Visualize $n \to F_n(x_i)$

$F_n(x_i)$ Gaussian Kernel, $N_f = N$

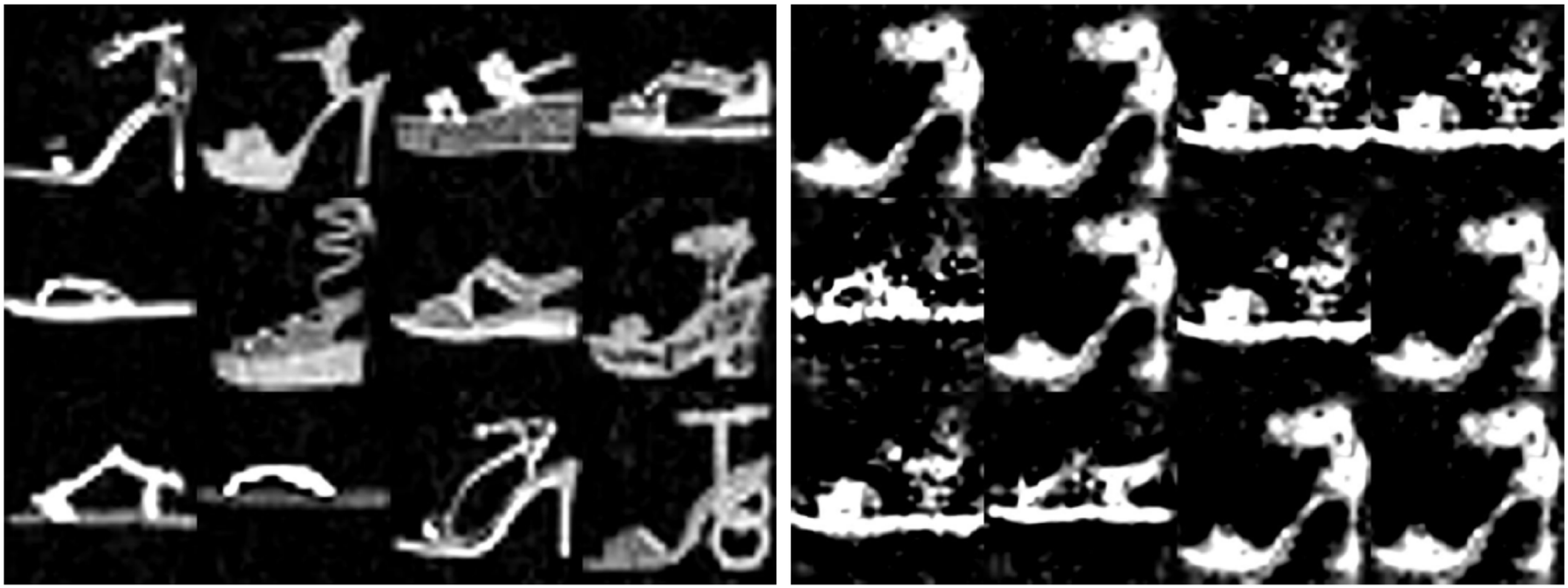


Application to Fashion-MNIST

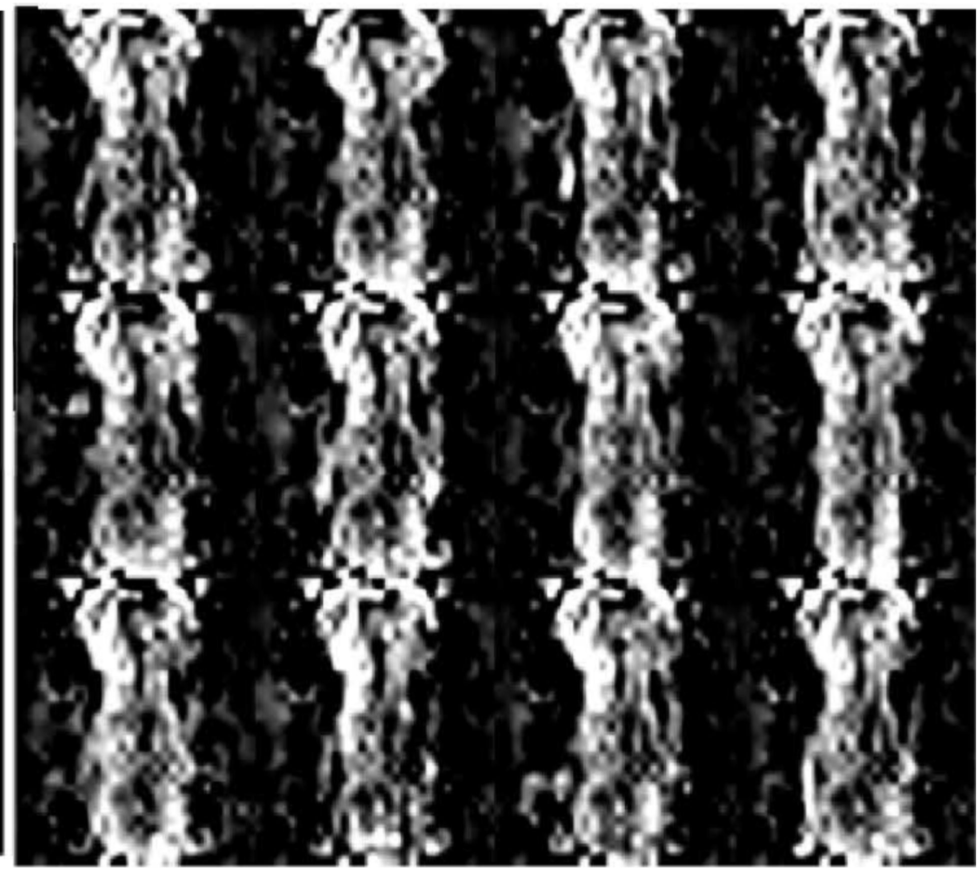
$$N = 60000$$

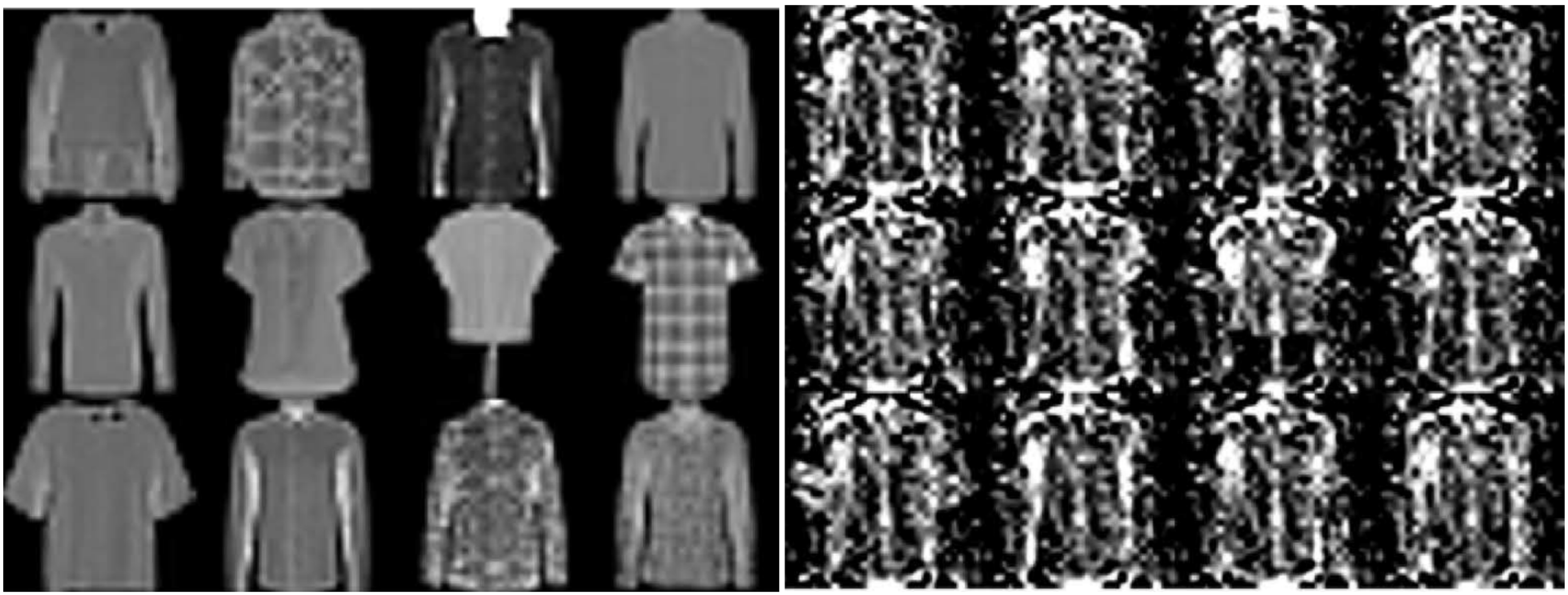
$$N_f = 600$$

12000 layers, large steps

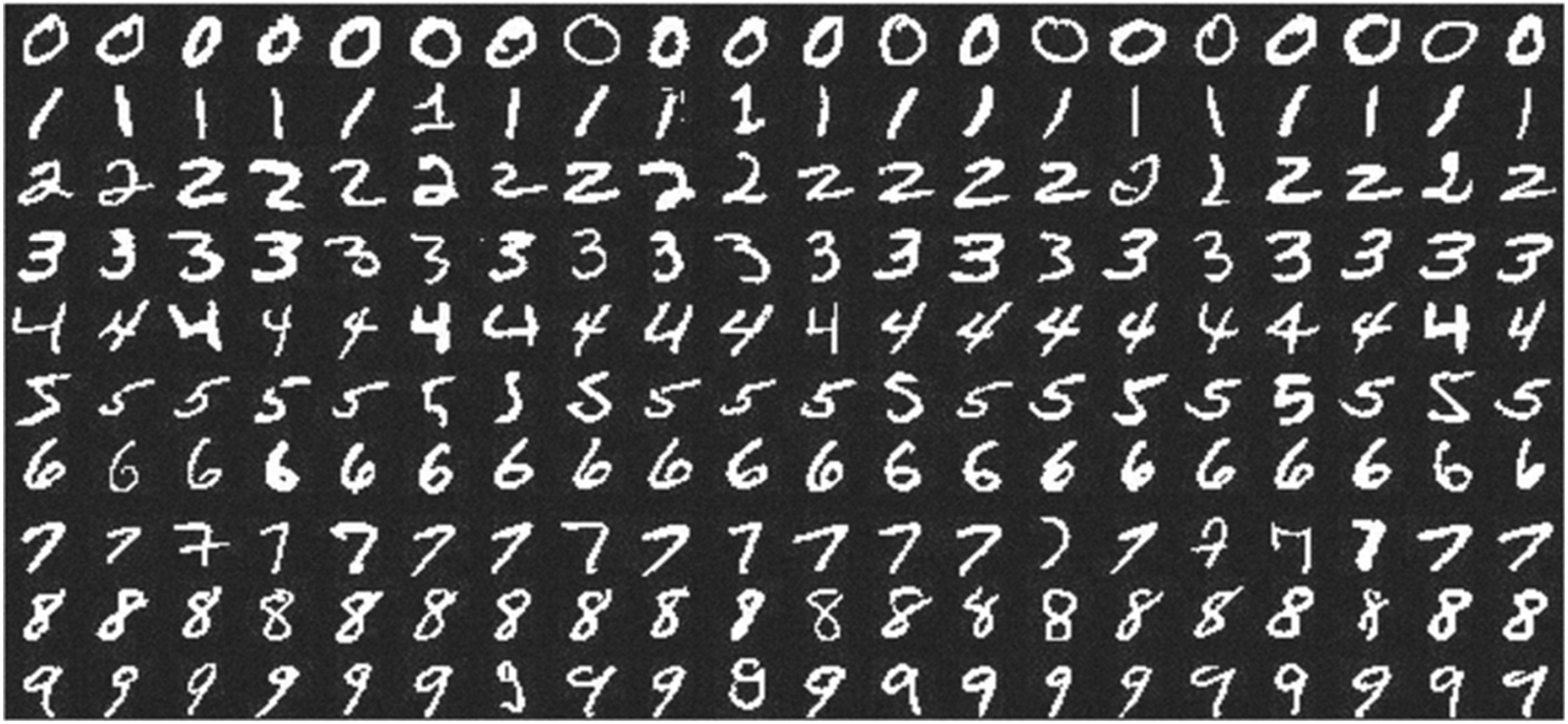


50000 layers, small steps



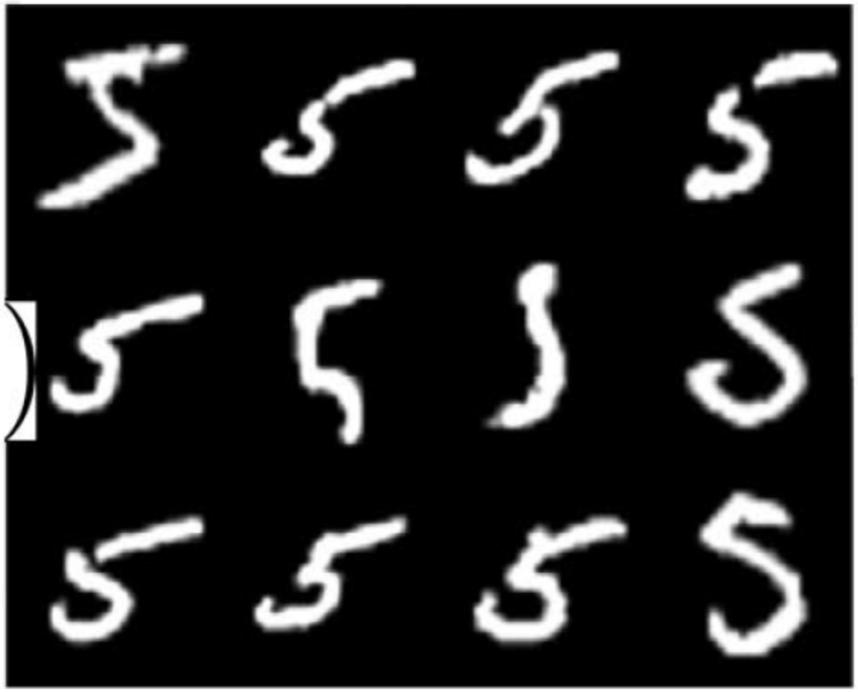
Application to MNIST



$N = 60000$

$N_f = 600$

12000 layers

(1)

(2)


(3)

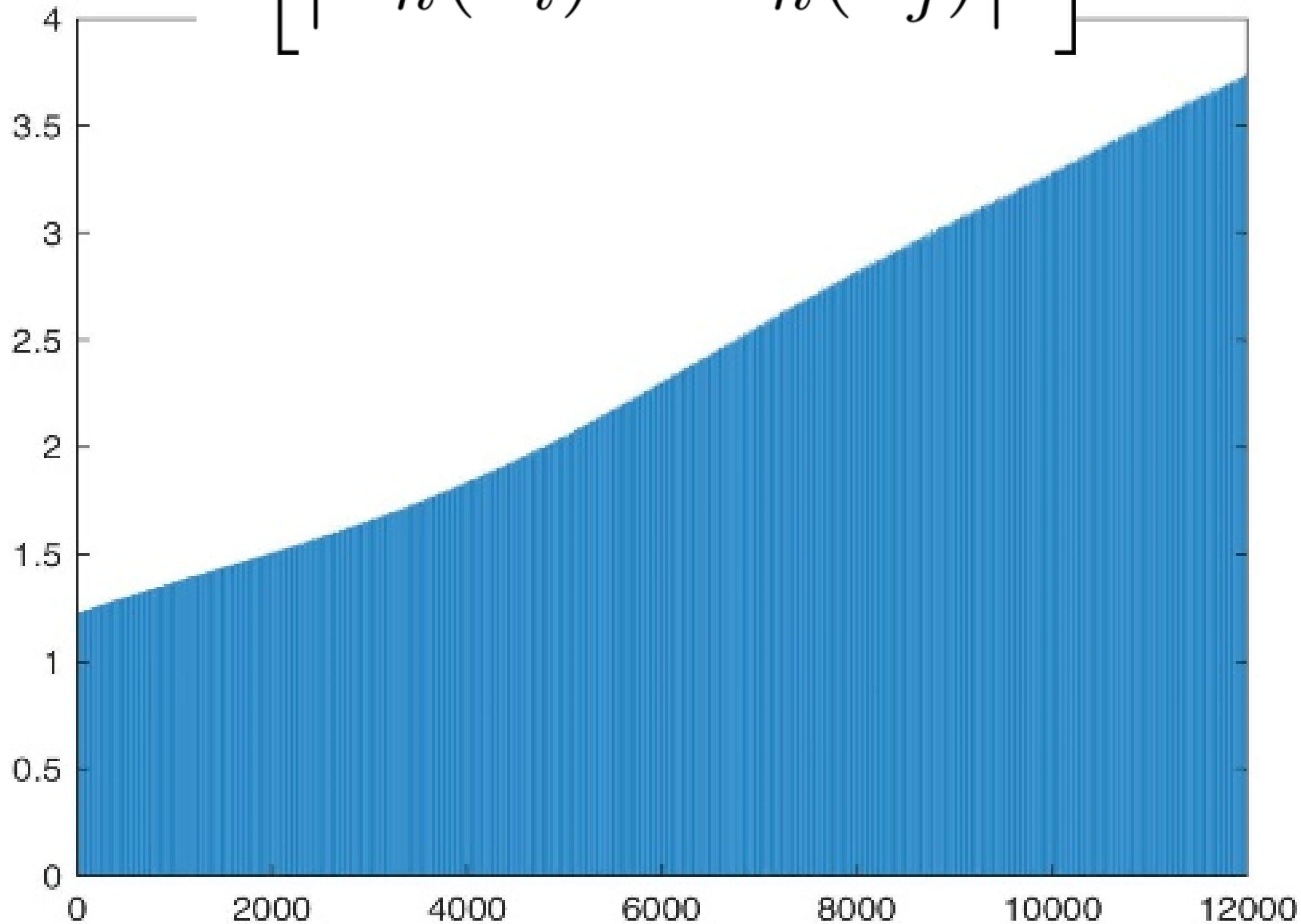
(4)

MNIST

Average distance, inter-class

$$y_i \neq y_j$$

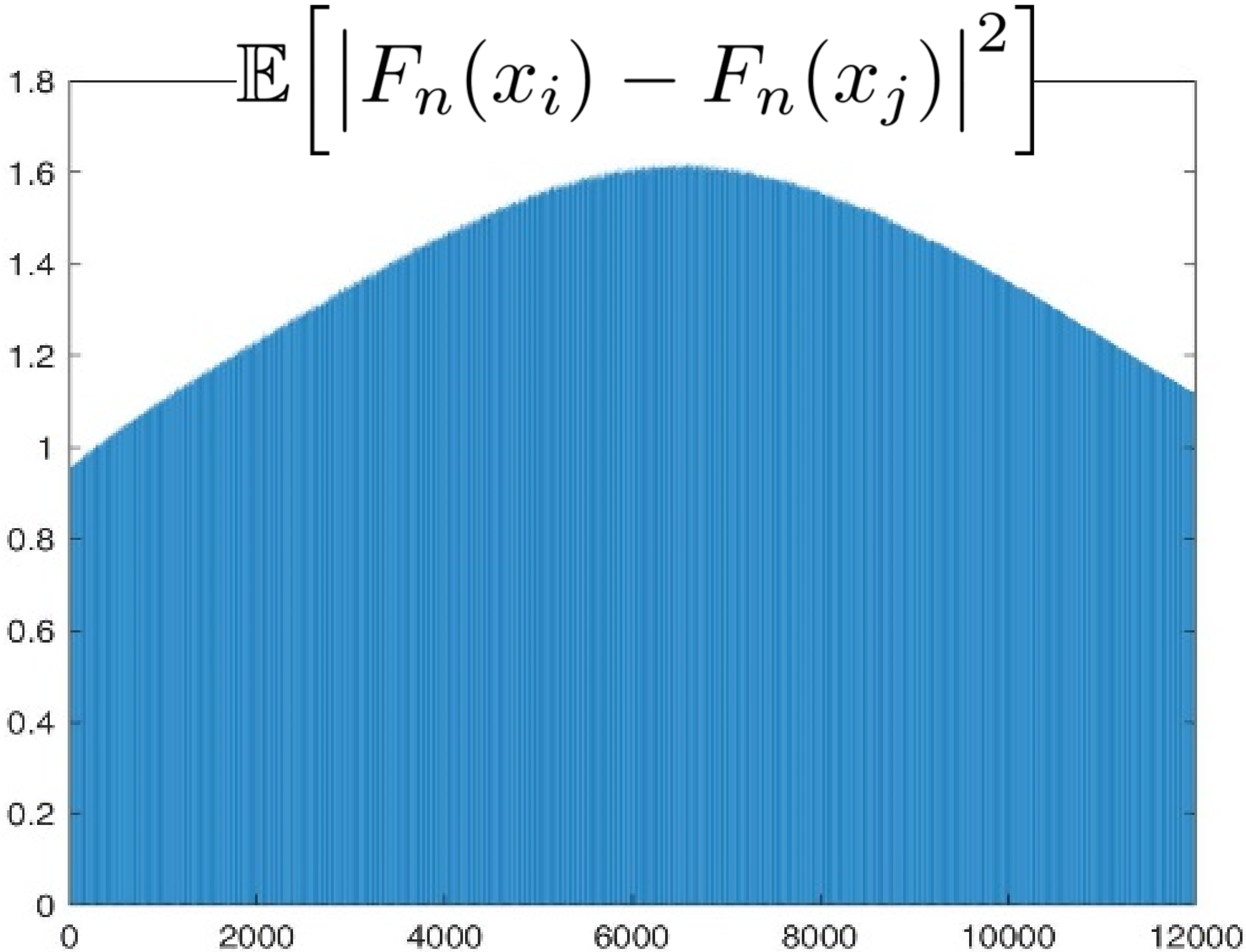
$$\mathbb{E}\left[\left|F_n(x_i) - F_n(x_j)\right|^2\right]$$



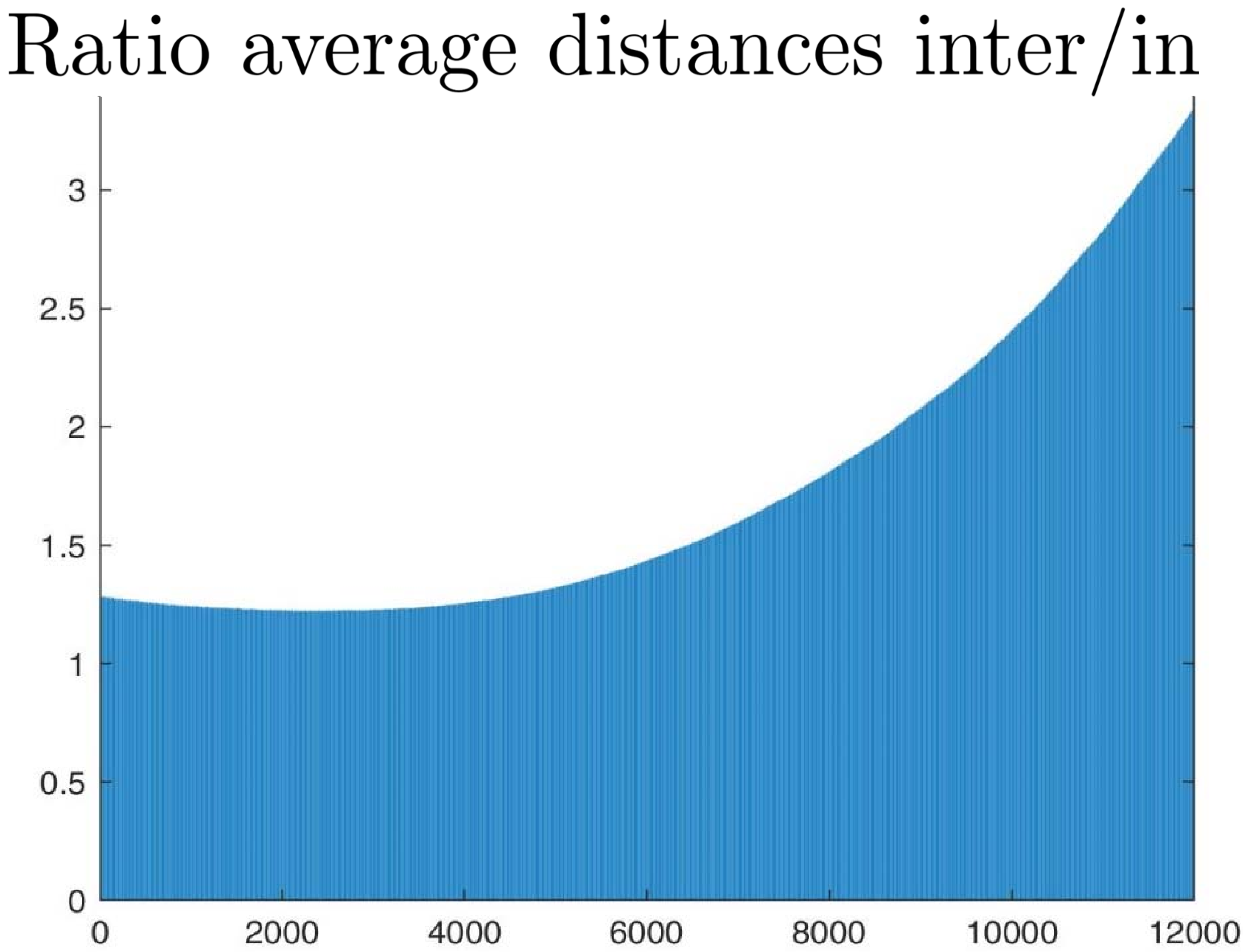
MNIST

Average distance, in-class

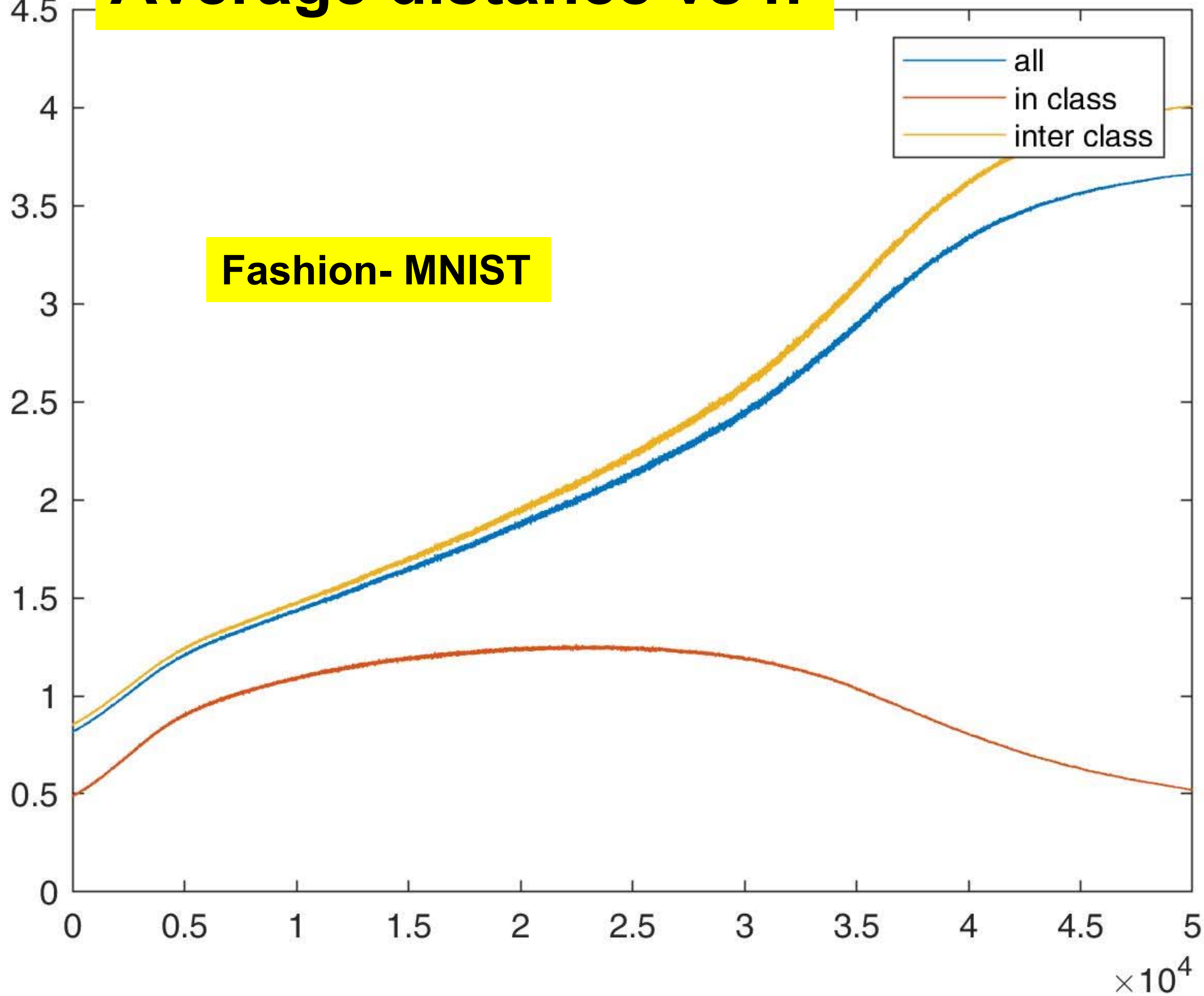
$y_i = y_j$



MNIST


Ratio average distances inter/in
3
2.5
2
1.5
1
0.5
0
0
2000
4000
6000
8000
10000
12000

Average distance vs n



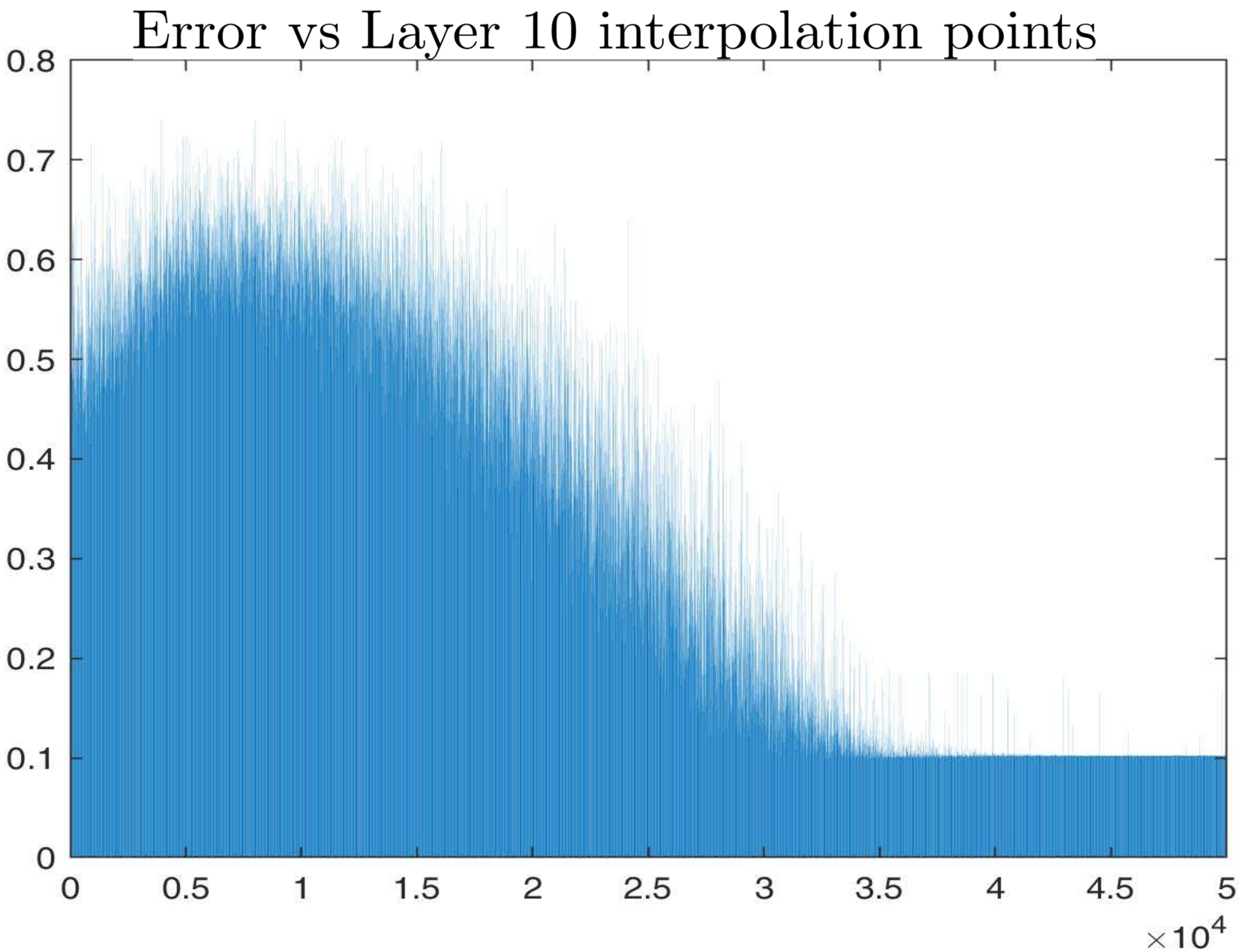
Classify 10000 test points

Use kernel K_n
and N_I interpolation points
selected at random

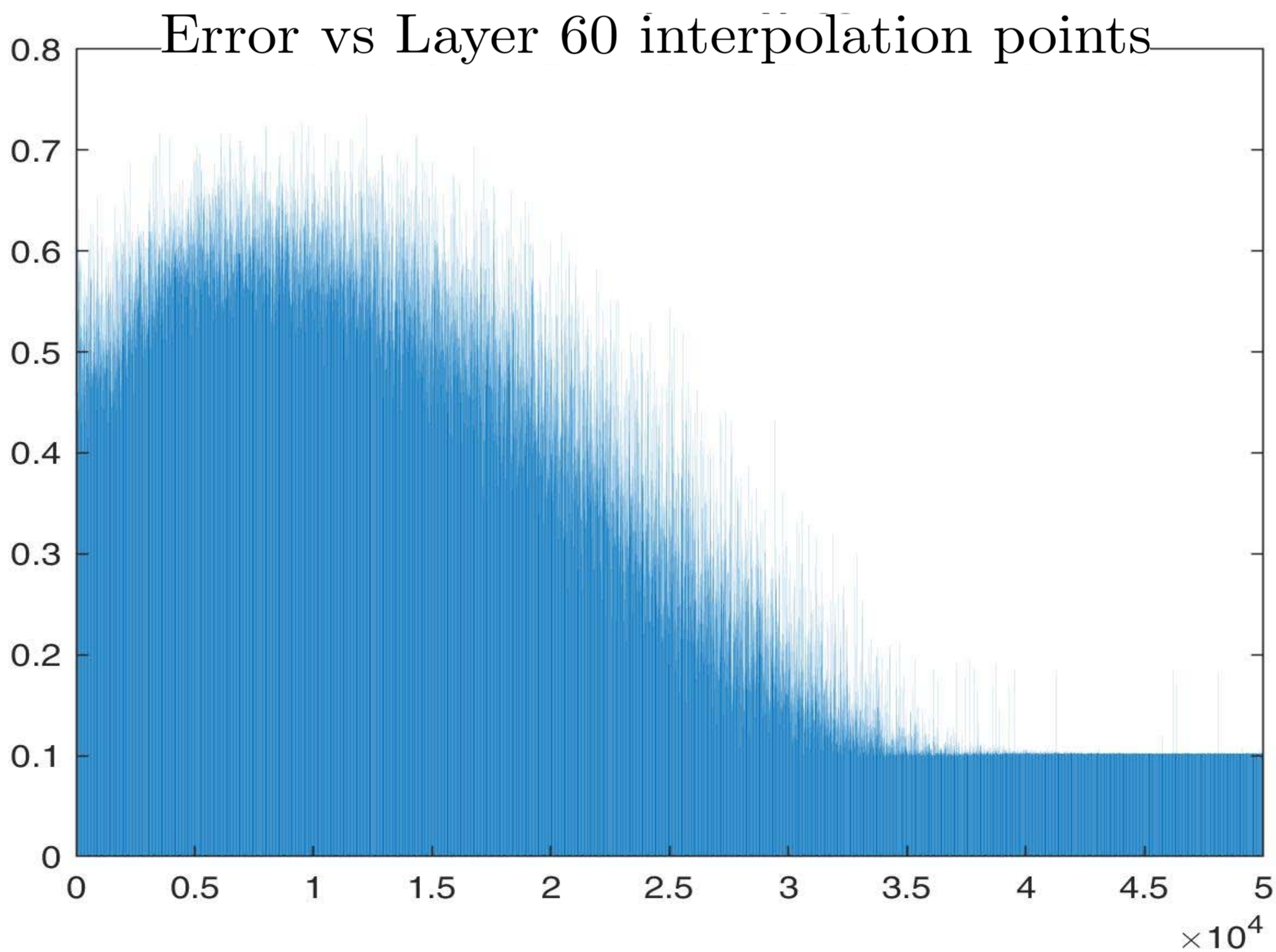
$$N_I = 6000, 600, 60, 10$$

$N_I = 10$ ⟺ Interpolate with only 1 point per class

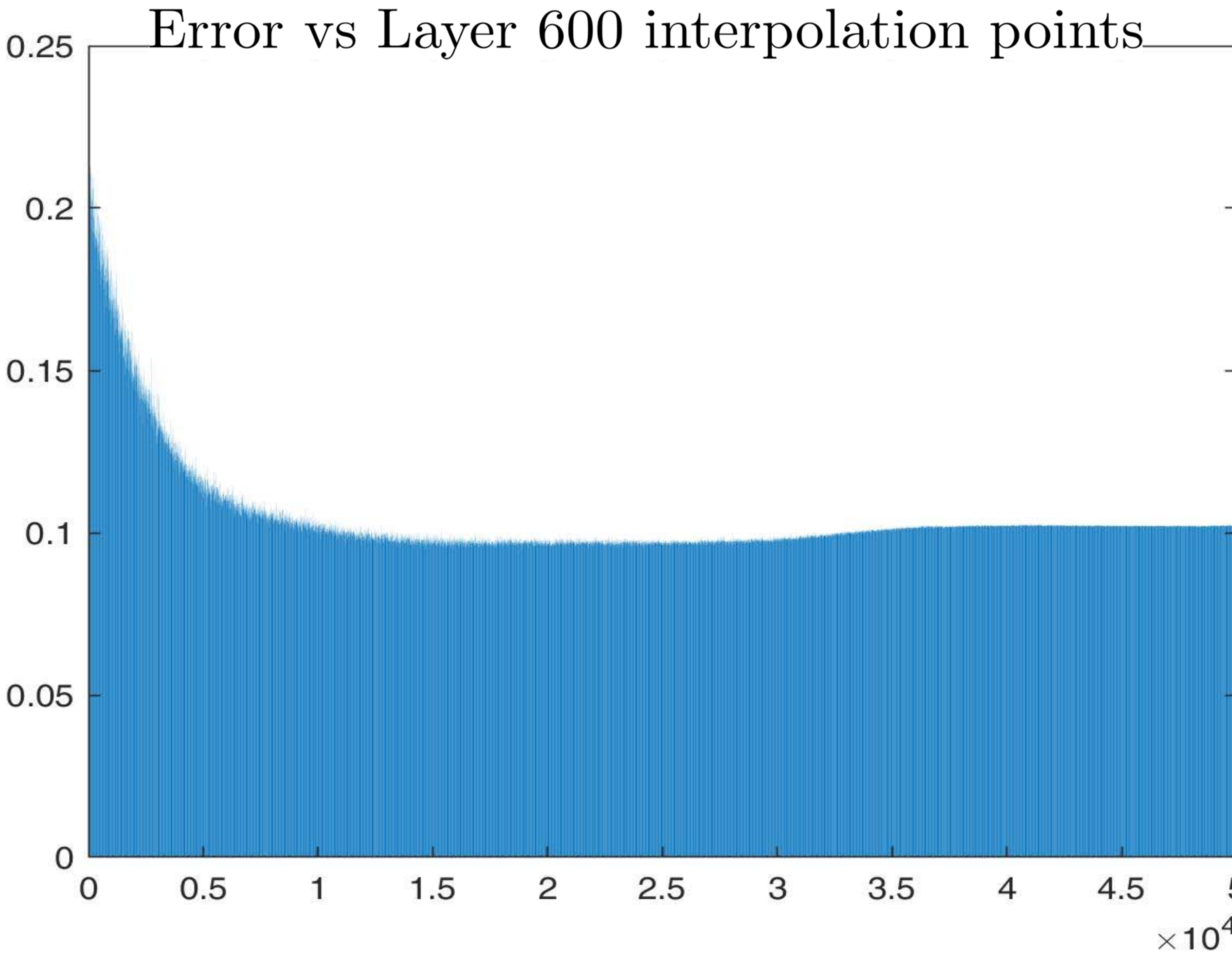
Fashion-MNIST Test Error vs layer



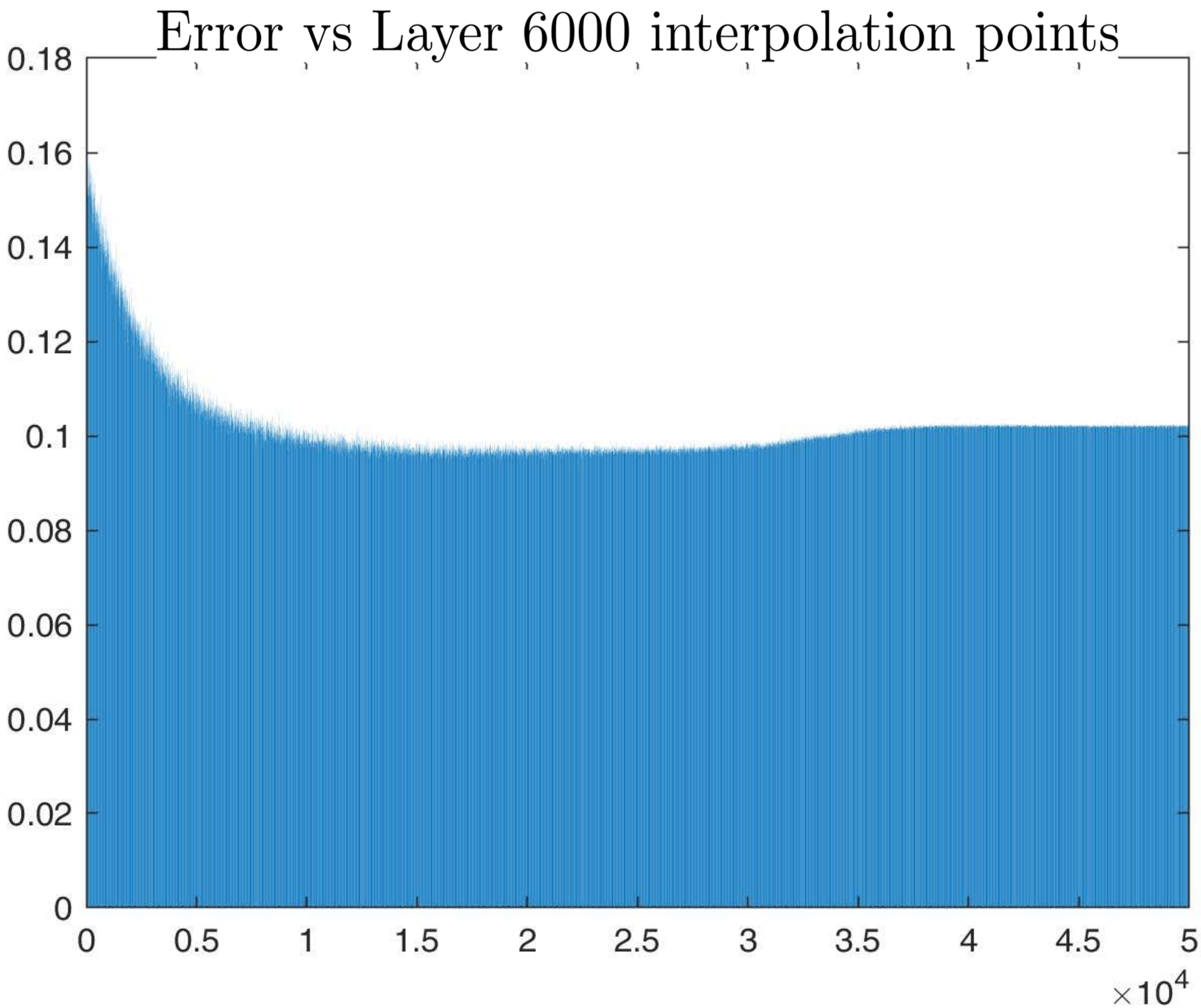
Fashion-MNIST Test Error vs layer



Fashion-MNIST Test Error vs layer



Fashion-MNIST Test Error vs layer



Fashion MNIST

For $15000 \leq n \leq 25000$

9.7% average error with K_n and 600 interpolation points

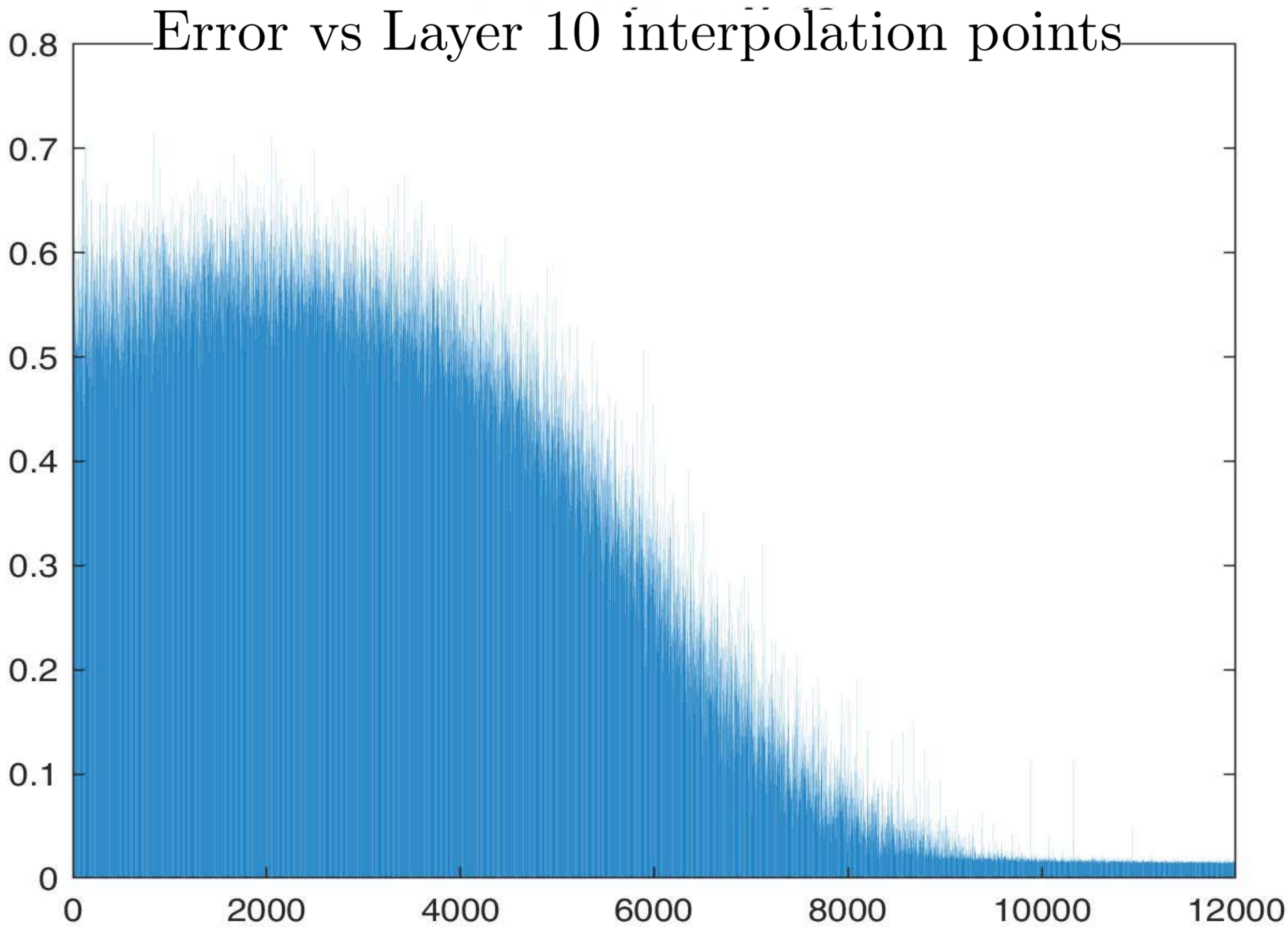
N_I	Average error	Min error	Max error	Standard Deviation
6000	0.096809	0.094	0.1001	6.997×10^{-4}
600	0.097026	0.0945	0.1003	6.7479×10^{-4}
60	0.44911	0.175	0.7337	0.09377
10	0.44959	0.1457	0.726	0.093017

For $49900 \leq n \leq 50000$

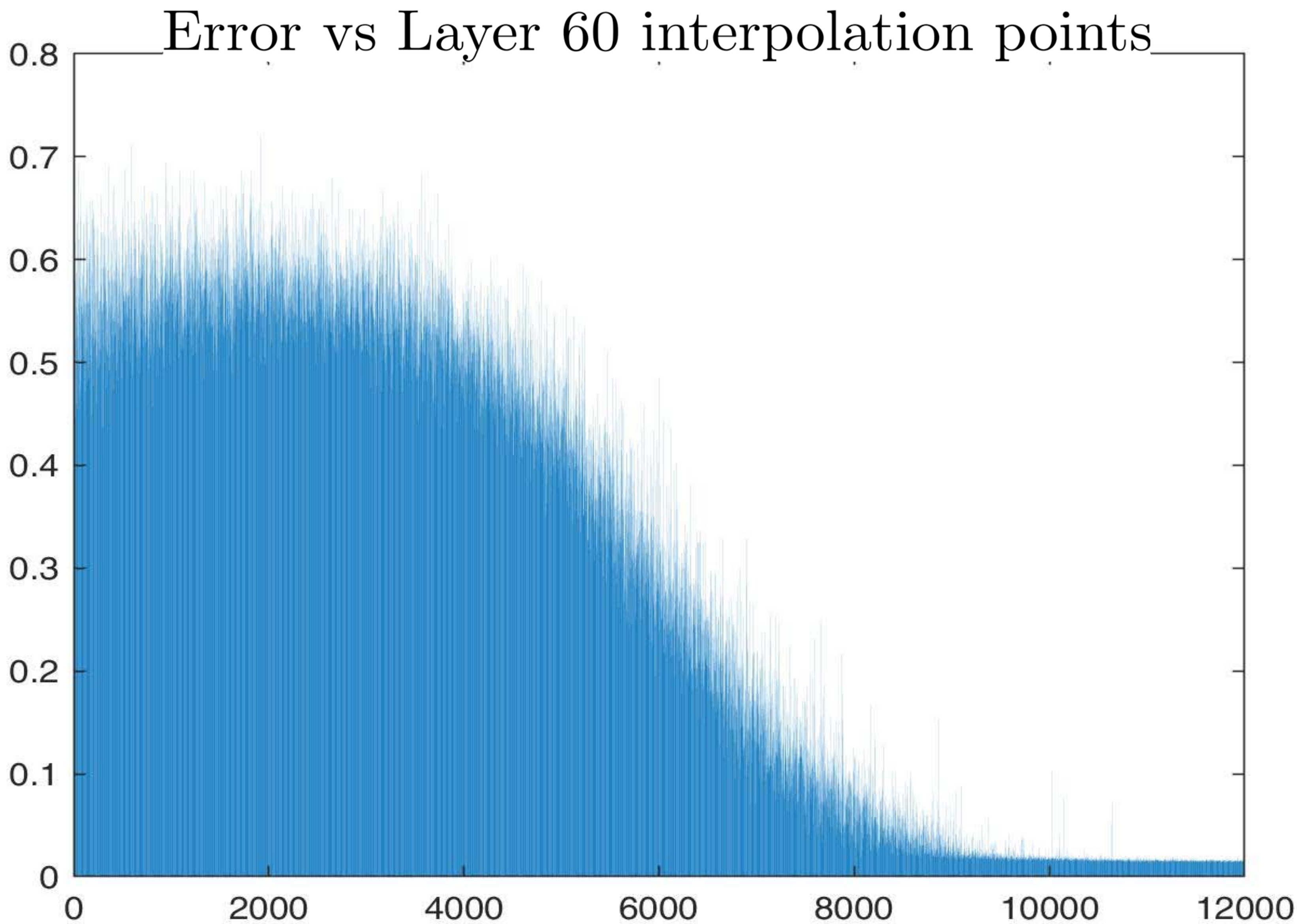
10% average error with K_n and 10 interpolation points

N_I	Average error	Min error	Max error	Standard Deviation
6000	0.10207	0.1016	0.1024	1.7049×10^{-4}
600	0.10217	0.1017	0.1023	9.1034×10^{-5}
60	0.10222	0.1018	0.1026	1.8225×10^{-4}
10	0.10223	0.1018	0.1028	1.9253×10^{-4}

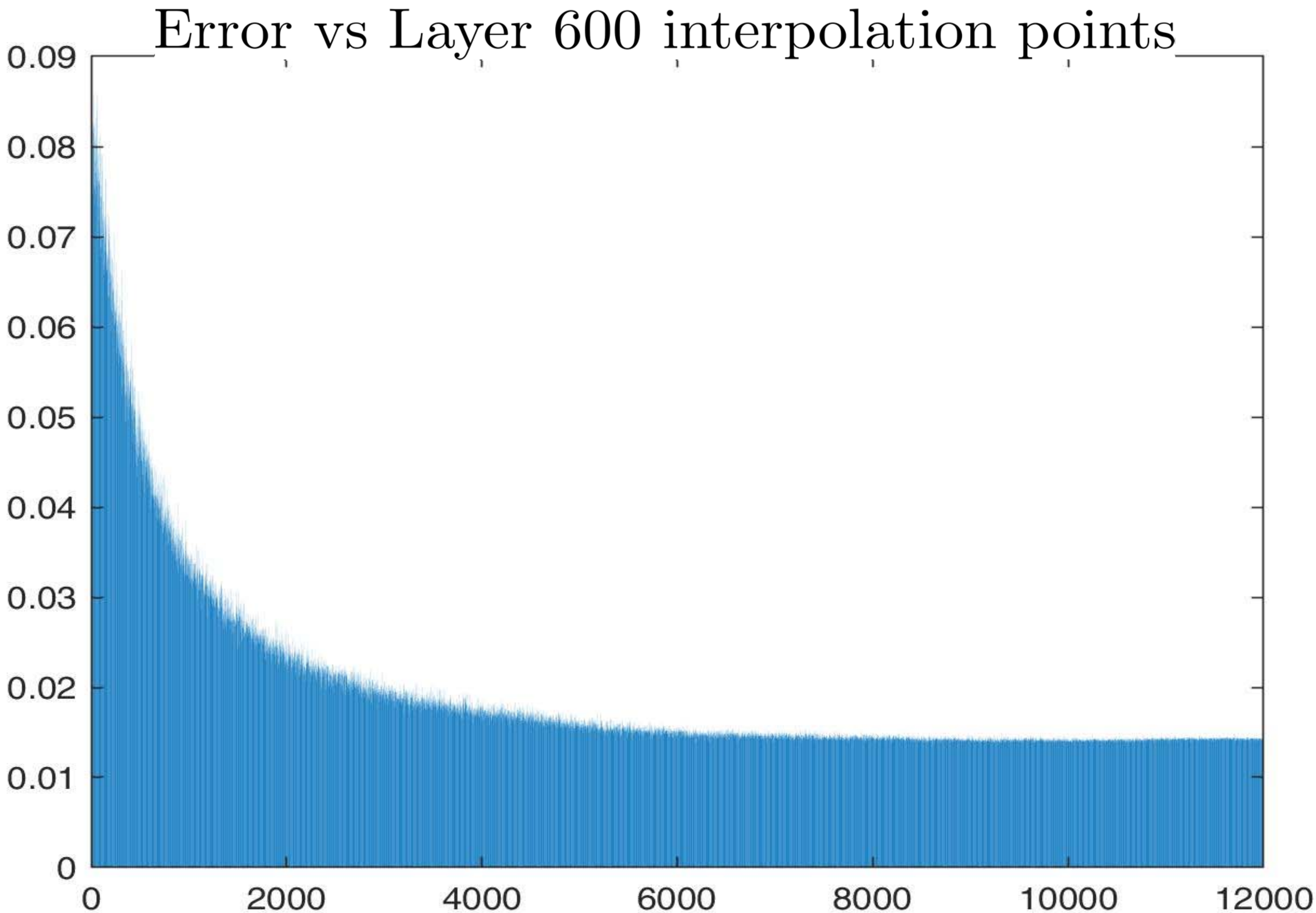
MNIST Test Error vs layer



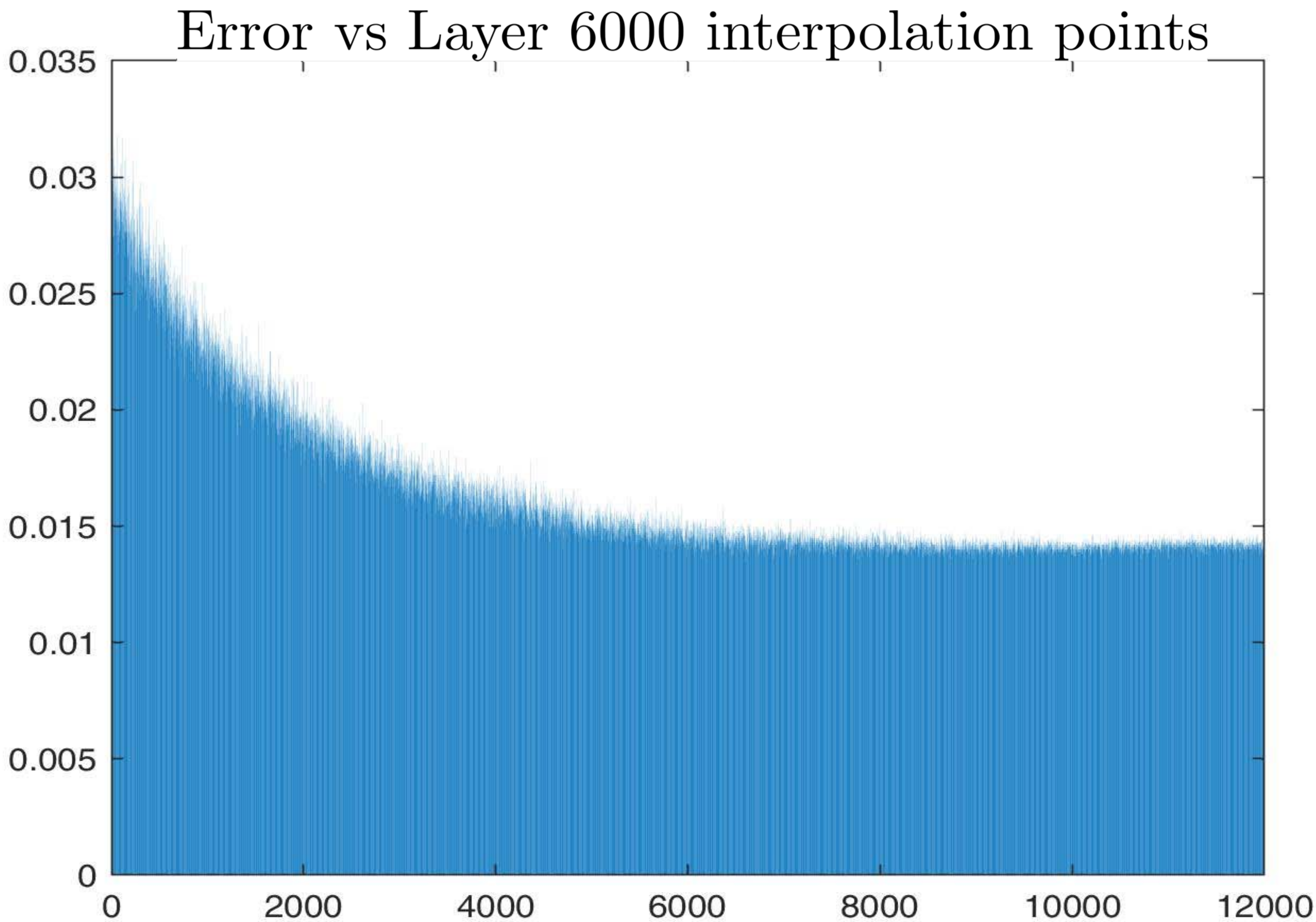
MNIST Test Error vs layer



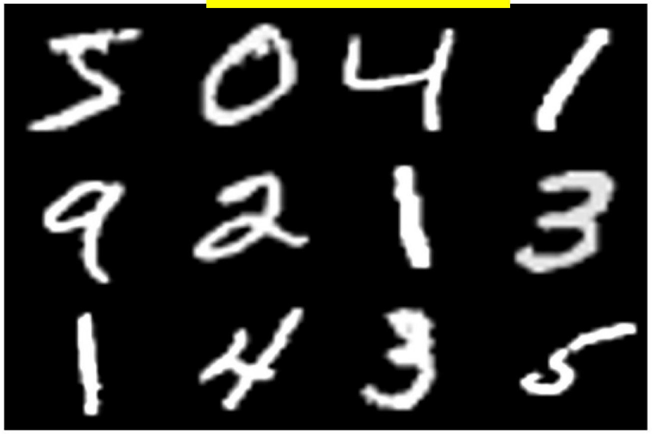
MNIST Test Error vs layer



MNIST Test Error vs layer



MNIST



$N = 60000$

10000 test points

$N_f = 600$

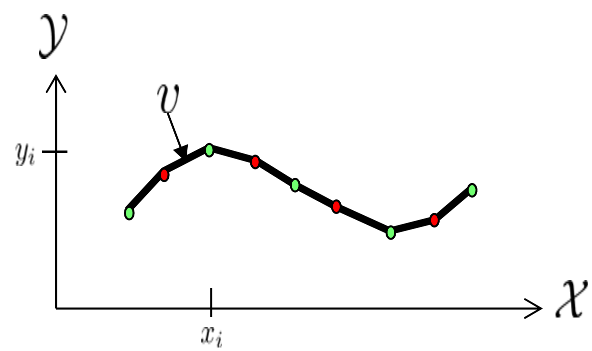
$n = 12000$

1.5% average error with K_n and 10 interpolation points

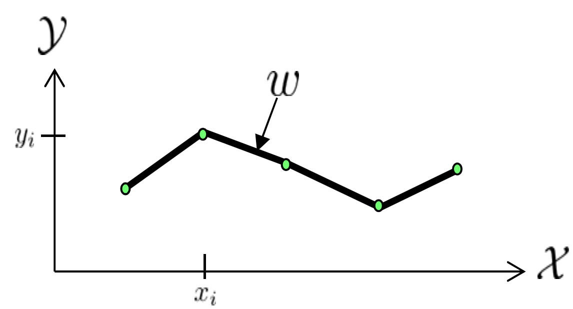
N_I	Average error	Min error	Max error	Standard Deviation
6000	0.014061	0.0137	0.0144	1.3036×10^{-4}
600	0.014127	0.0139	0.0144	1.0945×10^{-4}
60	0.014916	0.0137	0.0169	6.2669×10^{-4}
10	0.014839	0.0132	0.0163	6.473×10^{-4}

Premise A kernel K is good if the number of interpolation points can be halved without significant loss in accuracy

v: Interpolate with K and N points



w: Interpolate with K and $N/2$ points



$$\rho = \frac{\|v-w\|^2}{\|v\|^2}$$

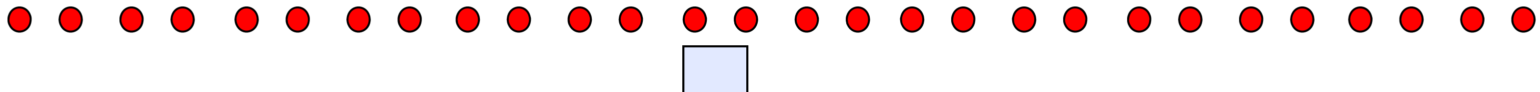
$$\|v\|^2 = \sup_\phi \frac{(\int \phi(x) v(x)\, dx)^2}{\int \phi(x) K(x,x') \phi(x')\, dx\, dx'}$$

Good kernel ⟺ Small ρ

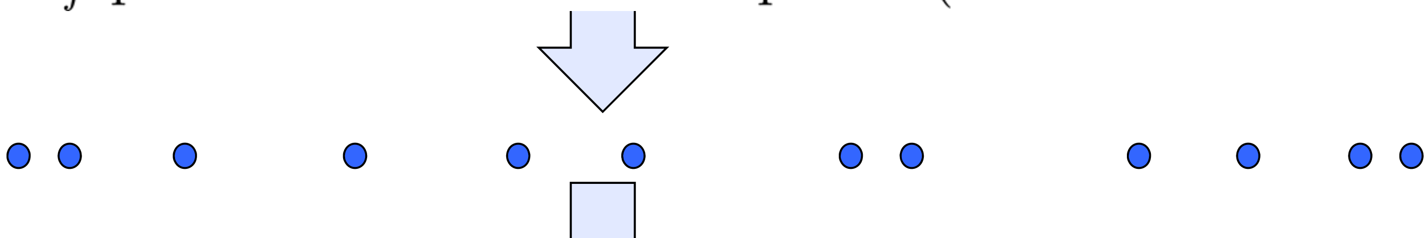
Parametric version

Step $n \to n+1$

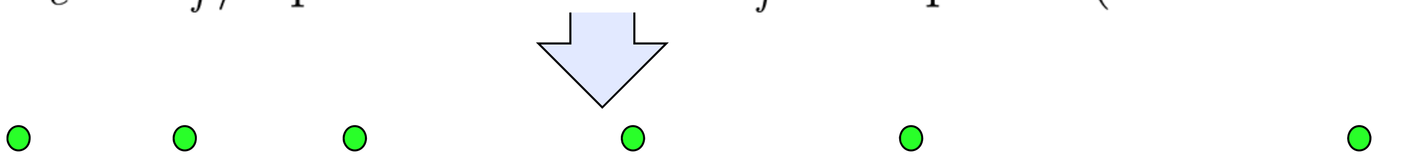
$K(\alpha)$: parametrized family of kernels



Select N_f points out of the N data points (at random uniformly)



Select $N_c = N_f/2$ points out of the N_f data points (at random uniformly)



v: Kriging with N_f points

w: Kriging with N_c points

$$\rho = \frac{\|v-w\|^2}{\|v\|^2}$$

$$\boxed{\alpha \to \alpha - \epsilon \nabla_\alpha \rho(\alpha)}$$

$\nabla_\alpha \rho(\alpha)$: Computable using the gamblet machinery

Application: Recovery of the coefficients of a PDE

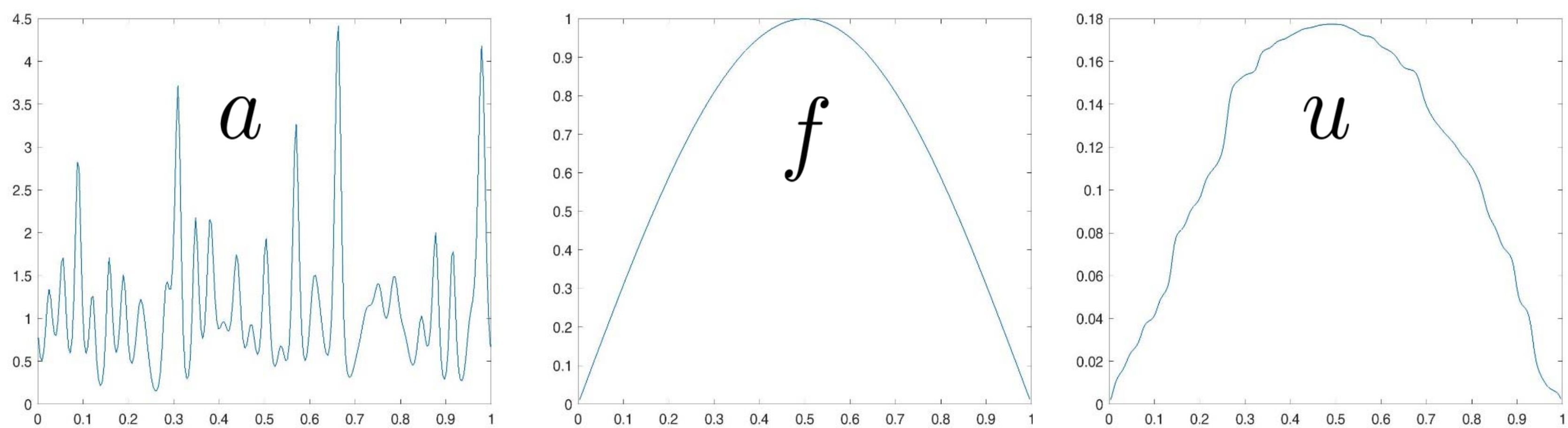
$$(1)\quad \begin{cases} -\operatorname{div}(a\nabla u) = f, & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \end{cases} \qquad f \in L^2(\Omega)$$

a, u, f: unknown

You see $(y_i = u(x_i))_{1\le i\le N}$

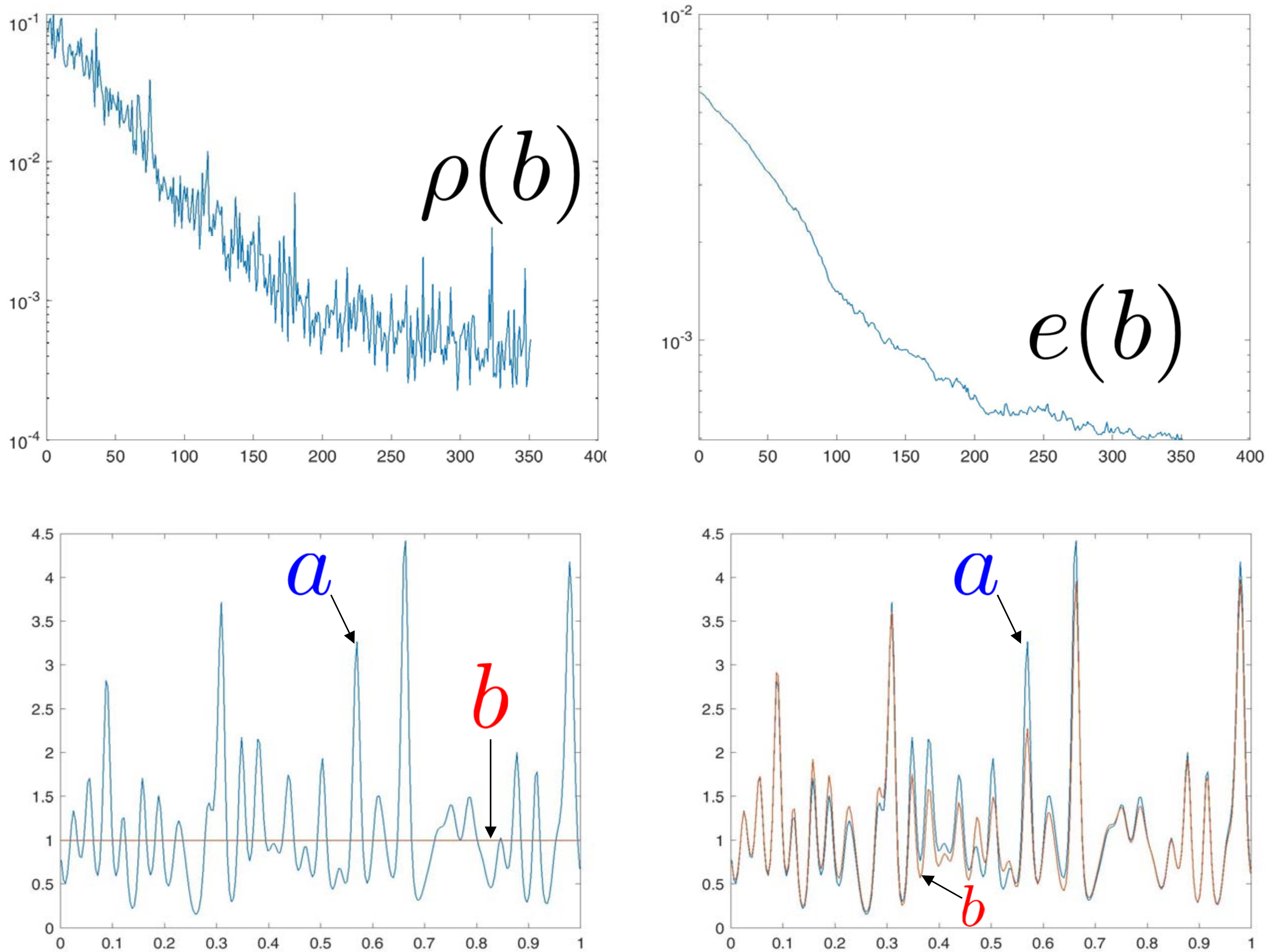
You want to recover a

G_b: Green's function of (1) with $a = b$



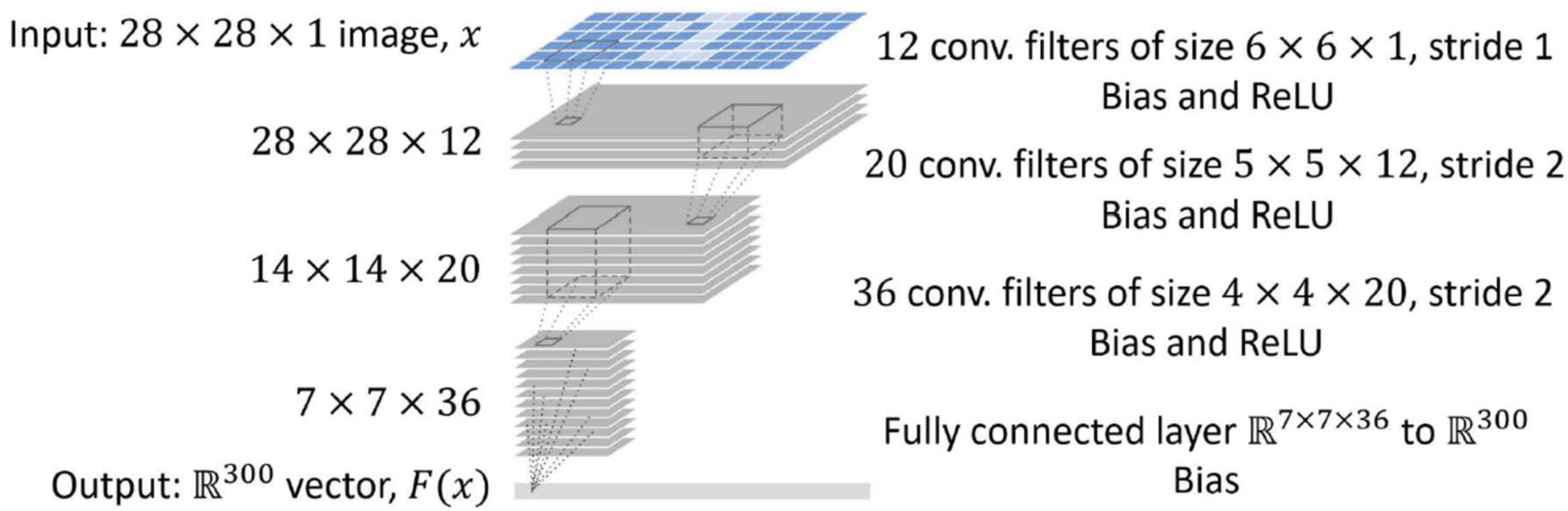
Implementation of the algorithm

$e(b) = \|u - v_b\|_{L^2(\Omega)}$ recovery error



Kernels parametrized by weights of a CNN

MNSIT



$$K(x, x') = K_1(F(x), F(x'))$$

$$K_1(x, x') = e^{-\frac{|x - x'|^2}{\gamma^2}}$$

Training

Step $n \to n+m$

Select $N_f = 500$ points out of the N data points (at random uniformly)

Select $N_c = 250$ points out of the N_f data points (at random uniformly)

v: Kriging with N_f points

w: Kriging with N_c points

$$\rho = \frac{\|v-w\|^2}{\|v\|^2}$$

$$e_2 = \|v-w\|^2_{L^2}$$

Minimize ρ or e_2 with respect to weights of the network

Interpolate training data with
$N_I = 6000, 600, 60, 10$ points selected at random

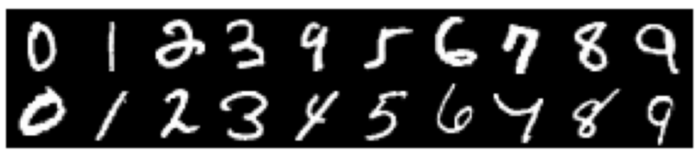
Observations

Works better than minimizing Relative Entropy + Dropout

Gives state of the art test accuracies
(compared to CNNs not using data augmentation)

Interpolation with 10 points is more sensitive to bad samples

Bad



Good

Minimizing e_2 gives slightly better results than ρ
Results with ρ are slightly more stable/robust

MNIST

Training by minizing ρ

N_I	Average error	Min error	Max error	Standard Deviation
6000	0.575%	0.42%	0.72%	0.052%
600	0.628%	0.48%	0.83%	0.062%
60	0.728%	0.51%	1.23%	0.103%
10	1.05%	0.58%	4.81%	0.375%

Training by minizing e_2

N_I	Average error	Min error	Max error	Standard Deviation
6000	0.646%	0.51%	0.78%	0.046%
600	0.676%	0.56%	0.82%	0.047%
60	0.850%	0.58%	3.98%	0.357%
10	4.434%	0.97%	18.91%	2.320%

Fashion MNIST

Training by minizing ρ

N_I	Average error	Min error	Max error	Standard Deviation
6000	8.526%	8.17%	8.96%	0.120%
600	8.810%	8.36%	9.29%	0.140%
60	11.677%	9.32%	18.03%	1.437%
10	36.642%	23.44%	53.56%	4.900%

Training by minizing e_2

N_I	Average error	Min error	Max error	Standard Deviation
6000	8.561%	8.23%	8.97%	0.135%
600	8.724%	8.31%	9.26%	0.161%
60	9.677%	8.77%	11.48%	0.486%
10	15.261%	10.00%	32.69%	3.196%

Thank you

DARPA EQUiPS / AFOSR award no FA9550-16-1-0054
(Computational Information Games)

AFOSR. Grant number FA9550-18-1-0271.
Games for Computation and Learning, 2018-2021.