Lifting Layers: Analysis and Applications

Peter Ochs

Mathematical Optimization Group
Saarland University

— 03.02.2019 —

joint work: Tim Meinhardt, Laura Leal-Taixe, and Michael Moeller

Supervised Learning:

- **Goal**: learn (parametric) estimator $\mathcal{N}(\cdot;\theta)\colon \mathcal{X} \to \mathcal{Y}$ satisfying

$$\mathcal{N}(x;\theta) \approx y \quad \text{for } \mathbb{P}\text{-a.e. pair } (x,y) \in \mathcal{X} \times \mathcal{Y}\,.$$

- Minimize the **expected error** of a loss function $\mathcal{L}$:

$$\min_{\theta\in\mathbb{R}^N} \mathbb{E}_{(x,y)}\left[\mathcal{L}(\mathcal{N}(x;\theta),y)\right] = \min_{\theta\in\mathbb{R}^N} \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(\mathcal{N}(x;\theta),y)\, d\mathbb{P}(x,y)\,.$$

- We **optimize the empirical risk** on a finite training set $X \times Y$:

$$\min_{\theta\in\mathbb{R}^N} \sum_{(x,y)\in X\times Y} \mathcal{L}(\mathcal{N}(x;\theta),y)$$

Supervised Learning
is an **interpolation/regression** problem.

Low Dimensional Interpolation

Low Dimensional Interpolation: (A simplified view)

- Consider **1D setting**: $\mathcal{X} \times \mathcal{Y} = [-1, 1] \times \mathbb{R}$ and $X = \{x_1, \ldots, x_M\}$.
- **Interpolation** using **splines**.
- **Kernel Method**: Minimization of regularized empirical risk (over RKHS $\mathcal{H}_k$)

$$\min_{\mathcal{N} \in \mathcal{H}_k} \sum_{(x,y) \in X \times Y} \mathcal{L}(\mathcal{N}(x), y) + \|\mathcal{N}\|$$

yields estimator of the form

$$\mathcal{N}(x; \theta) = \sum_{i=1}^{M} \theta_i k(x, x_i)$$

with positive definite kernel $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

- Close relation to spline interpolation.

High Dimensional Interpolation

High Dimensional Interpolation:

- Parametric estimator is a **Neural Network** $\mathcal{N}(x;\theta)$.

- $\mathcal{N}(x;\theta)$ has special composite structure (architecture):

$$\mathcal{N}(x;\theta) = W^L\left(\dots\sigma\left(W^1\left(\sigma\left(W^0x\right)\right)\right)\dots\right)\,,\quad \theta := \left(W^0,\dots,W^L\right).$$

- Common building blocks are:
  - fully connected layer or convolution layer: affine mappings W^i.
  - ReLU activation function: $\sigma(x) = \max(x,0)$ coordinate-wise.
    (alternatives: leaky ReLUs, parameterized ReLUs, or maxout units, ...)

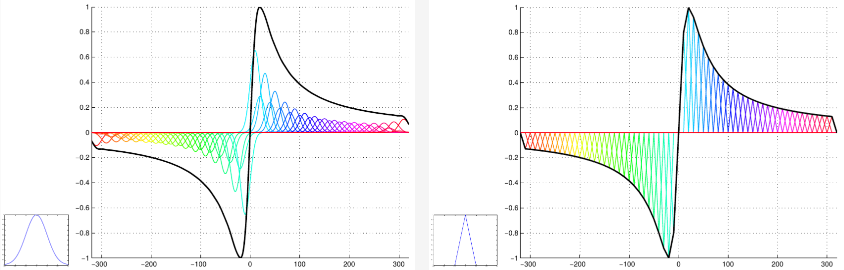
$\rightsquigarrow$ usually, they discard some information.

Learn Activation Functions

Learn Activation Functions: [CP15]

- Decouple parameters for each layer $l = 1, \ldots, L$ (and node i):

$$\sigma_i^l(x;\theta) = \sum_{j=1}^{m} \theta_{i,j}^l \underbrace{\varphi\left(\frac{|x-\mu_j|}{\Delta\mu}\right)}_{k(x,\mu_j)}$$



[CP15]

[CP15] [Y. Chen and T. Pock: *Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration*, TPAMI 2015.]

[SR14] [U. Schmidt and S. Roth: *Shrinkage Fields for Effective Image Restoration*, CVPR 2014.]

A Representer Theorem

A Representer Theorem: [Unser18]

- **Goal:** Optimize the shape of the activation function.

- Choice of regularization:
  - favor simple solutions
  - weakly differentiable (compatible with backpropagation)
  - locally linear (work best in practice)

$\rightsquigarrow$ Penalize second derivative ("sparse" second derivative) $TV^{(2)}$.

- Solution of regularized interpolation problem (in $BV^{(2)}$) is a **piecewise-linear function with max.** $M-2$ **adaptive knots**.

- Classic interpolation (Sobolev regularization) requires M knots $x_1, \ldots x_M$.

[Unser18] [M. Unser: *A Representer Theorem for Deep Neural Networks*, 2018.]

Learn Activation Function

Learn Activation Function:

- Learned activation in [CP15] with $\varphi(x) = \max(1 - |x|, 0)$ is an approximation where knots are fixed equidistantly.

$$\sigma(x;\theta) = \sum_{j=1}^{m} \theta_j \underbrace{\varphi\Big(\frac{|x-\mu_j|}{\Delta\mu}\Big)}_{k(x,\mu_j)}$$

- **Example**: $\mu_1 = -1, \mu_2 = 1$ and $\Delta\mu = 1$:

$$\sigma(x;\theta) = \theta_1 \max(x,0) - \theta_2 \min(x,0)$$

- We **lift** the information in different channels:

$$\sigma(x;\theta) = \ell(x) = \begin{pmatrix} \max(x,0) \\ \min(x,0) \end{pmatrix}$$

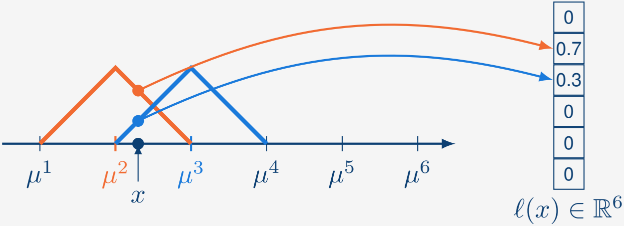
[CP15] [Y. Chen and T. Pock: *Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration*, TPAMI 2015.]

Novel Lifting Layer:

- For equidistant centers $\mu_1 < \ldots < \mu_m$ with distance $\Delta\mu$

$$\ell(x) = \begin{pmatrix} \varphi\left(\frac{|x-\mu_1|}{\Delta\mu}\right) \\ \vdots \\ \varphi\left(\frac{|x-\mu_m|}{\Delta\mu}\right) \end{pmatrix} \in \mathbb{R}^m$$

- Example with hat-function φ.



- **(Left-)inverse lifting** $\ell^\dagger : \mathbb{R}^m \to \mathbb{R}$: $\quad \ell^\dagger(z) = \sum_{i=1}^m z_i \mu^i$.

Motivation and Contribution

Contribution:

Novel non-linear layer
with **favorable properties**
and **good practical performance**.

Motivation by Functional Lifting in Optimization:

- Make non-convex problems convex in higher dimensional 'lifted' space.

Properties of our Lifting Layer:

- Naturally, yields linear splines.
- Does not discard information. It is lifted to different channels.
- "Tight" convex approximation of non-convex loss function.
- Good test accuracy in several experiments.

Properties of Lifting Layer

Properties of Lifting Layer in simple network architectures:

- Fully connected layer $z \mapsto \langle \theta, z \rangle$, $\theta \in \mathbb{R}^m$, composed with lifting layer

$$\mathcal{N}_\theta(x) := \langle \theta, \ell(x) \rangle = \sum_{i=1}^{m} \theta_i \varphi\Big(\frac{|x - \mu_i|}{\Delta\mu}\Big)$$

  yields, for example, a **linear spline** (continuous piecewise linear function).

- Splines are known to have remarkable approximation properties.

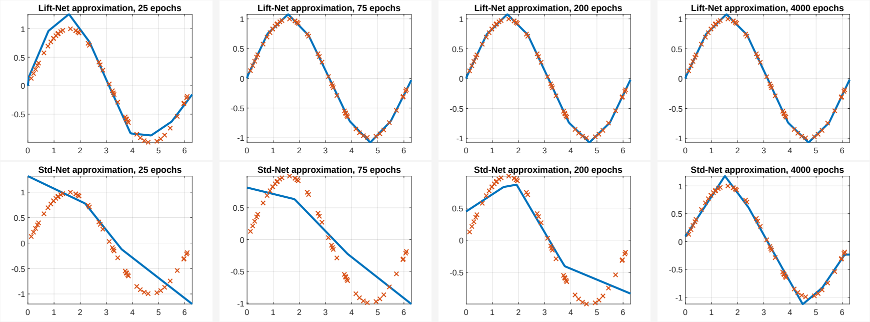
- If $\mathcal{L}$ is convex, then finding the best linear spline fit is a **convex problem**:

$$\min_\theta \sum_{i=1}^{N} \mathcal{L}(\langle \theta, \ell(x_i) \rangle \,; y_i)$$

- **Example (not true for ReLUs):** $\theta \mapsto (\max(\theta, 0) - 1)^2$ is non-convex.

Experiment ($1D$ regression):

- Fit values $y_i = \sin(x_i)$ from input data x_i sampled uniformly in $[0, 2\pi]$.

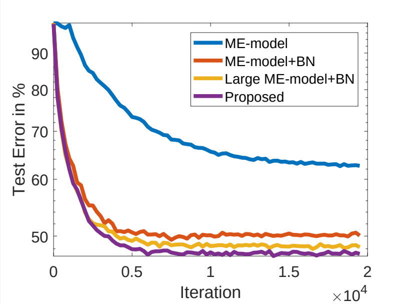


increasing number of epochs

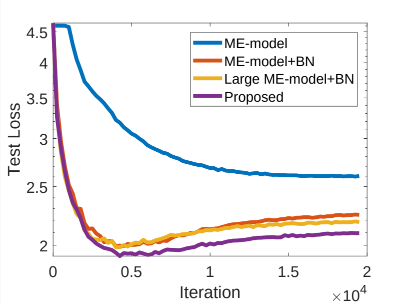
- **Top row**: lifting-based architecture $\mathcal{N}_\theta(x) = \langle \theta, \ell_9(x) \rangle$ **(Lift-Net)**.
- **Bottom row**: standard design architecture $\text{fc}_1(\max(0, \text{fc}_9(x)))$ **(Std-Net)**.

Experiment (Image Classification): **CIFAR-100**

- "Deep MNIST for expert model" (ME-model) by TensorFlow
- ME-model+BN = ME-model + batch normalization
- We replace ReLUs by lifting layers with $L = 3$.



(c) CIFAR-100 Test Error



(d) CIFAR-100 Test Loss

Experiment (Image Denoising): **BSD68 dataset**

- 16 blocks each with 46 convolution filters of size 3×3, batch normalization, lifting layer with $L = 3$.
- same training pipeline as for the DnCNN-S architecture.

Reconstruction PSNR in $[dB]$:

σ	noisy	BM3D	WNNM	EPLL	MLP	CSF	TNRD	DnCNN-S	Our
15	24.80	31.07	31.37	31.21	-	31.24	31.42	**31.72**	**31.72**
25	20.48	28.57	28.83	28.68	28.96	28.74	28.92	**29.21**	**29.21**
50	14.91	25.62	25.87	25.67	26.03	-	25.97	26.21	**26.23**

[BM3D] [Dabov et al.: *Image denoising by sparse 3-d transform-domain collaborative filtering*, 2007.]

[WNNM] [Gu et al.: *Weighted nuclear norm minimization with application to image denoising*, 2014.]

[EPLL] [Zoran, Weiss: *From learning models of natural image patches to whole image restoration*, 2011.]

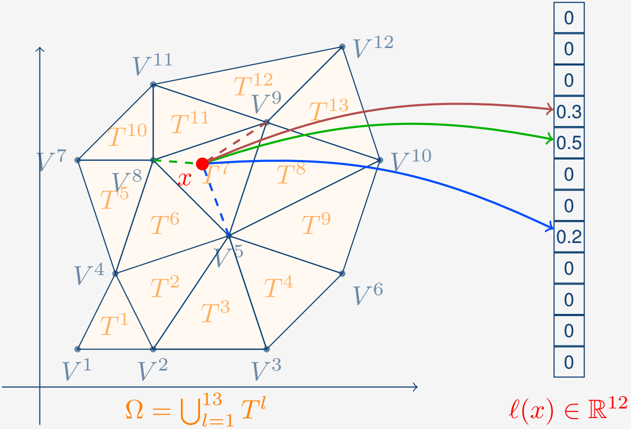
[MLP] [Burger et al.: *Image denoising: Can plain neural networks compete with BM3D?*, 2012]

[CSF] [Schmidt, Roth: *Shrinkage fields for effective image restoration*, 2014.]

[TNRD] [Chen, Pock: *On learning optimized reaction diffusion processes for effective image restoration*, 2015.]

[DnCNN-S] [Zhang et al.: *Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising*, 2017.]

Generalization (Vector-valued Lifting):



Lifting the Output

Lifting the Output:

$$x_i \longrightarrow (\mathcal{N}_\theta(x_i), y_i) \xrightarrow{\mathcal{L}_i} \mathbb{R}$$

Lifting the Output

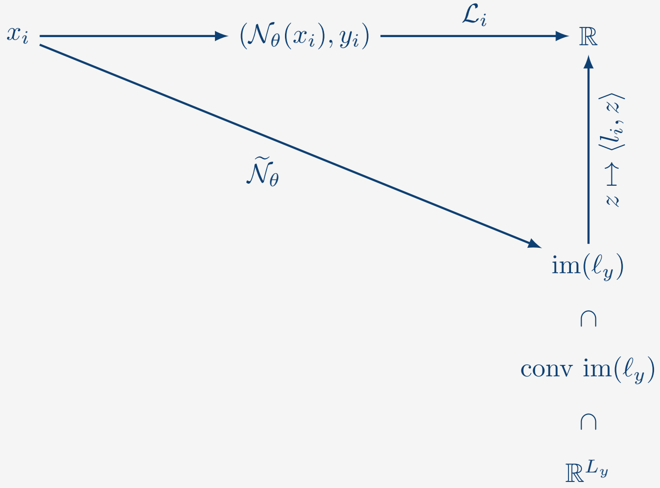
Lifting the Output (lift the loss function):

$$x_i \longrightarrow (\mathcal{N}_\theta(x_i), y_i) \xrightarrow{\mathcal{L}_i} \mathbb{R}$$

$$\mathbb{R} \xleftarrow{z \mapsto \langle l_i, z \rangle} \mathrm{im}(\ell_y)$$

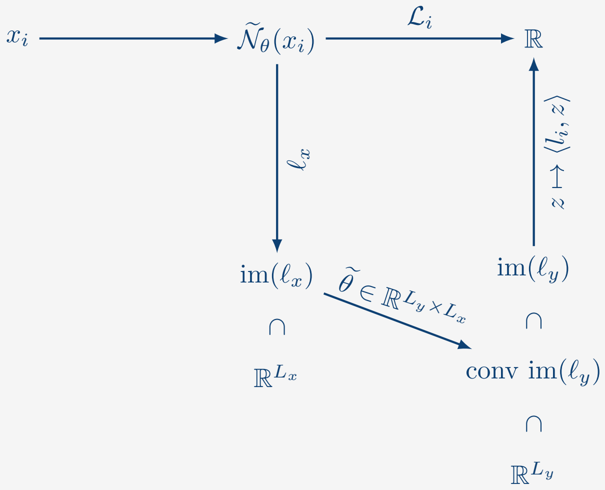
$$\mathrm{im}(\ell_y) \cap \mathrm{conv}\ \mathrm{im}(\ell_y) \cap \mathbb{R}^{L_y}$$

Lifting the Output (try to predict lifted point):

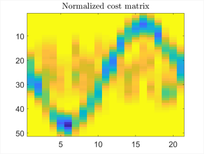


Lifting the Output

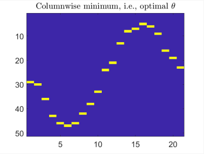
Lifting the Output (efficient approximation $\rightsquigarrow$ analytic solution for $\widetilde{\theta}$):



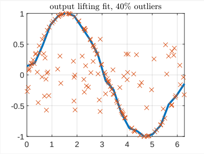
Experiment (Lifting the Output): Robust fitting by truncated linear loss



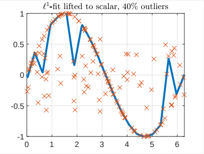
(a) Cost matrix c



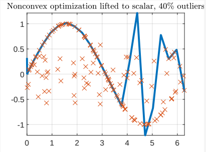
(b) Optimal θ



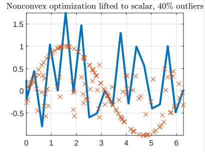
(c) **Resulting fit**



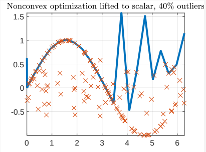
(d) Best ℓ^1 fit



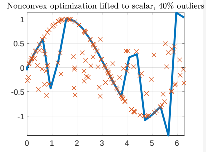
(e) Non-convex fit 1



(f) Non-convex fit 2



(g) Non-convex fit 3

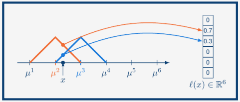


(h) Non-convex fit 4

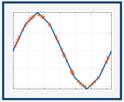
- (c) Our lifting yields a convex optimization problem.
- (d) Convex ℓ^1-loss function.
- (e)-(f) Direct optimization of truncated linear loss.

Summary

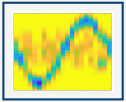
Summary:



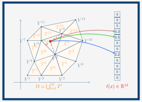
Introduce novel type of non-linear layer for neural networks: **Lifting Layer**



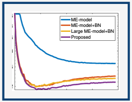
Favorable theoretical properties.



The lifting seems to act “**convexifying**”.



Vector-valued lifting.



Good performance for **Machine Learning and Computer Vision** problems.