Topics in Numerical Analysis II
Computational Inverse Problems

Bangti Jin (b.jin@cuhk.edu.hk)

Chinese University of Hong Kong

November 23, 2022

Outline

Review

linear (nonlinear) inverse problems

$$Ax = y$$

- Hilbert space: truncated SVD, Tikhonov regularization, iterative methods (Landweber, Kaczmarz method, stochastic gradient descent, conjugate gradient method)
- Banach space: sparsity regularization (theory + algorithm)
- other possibilities: other penalties, iterative methods in Banach space

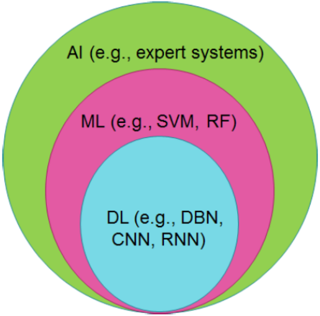
What is beyond ?

What is deep learning

- gained a lot of attention recently and impressive results, e.g., image classification, speech recognition, segmentation, natural language processing, etc.
- computationally highly efficient
- limitation of current approaches to inverse problems: expressibility of hand-crafted prior (regularizer)
- instead: learn prior from data itself to improve the image reconstruction

what is deep learning ?

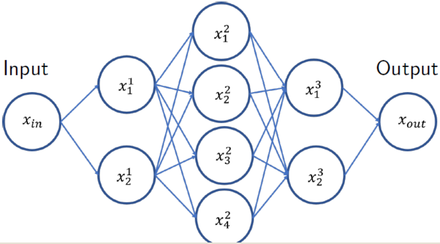
- artificial intelligence: any techniques that enables computers to mimic human intelligence
- machine learning: a subset of AI using statistical techniques to enable machine to learn from data
- deep learning: a subset of ML where complicated tasks are performed by breaking them down into several layers



convolutional neural networks (CNN)

- aim: to establish a mapping $G_\theta(x_i) = x_o$ with input / output $x_i, x_o \in X = \mathbb{R}^{n\times n}$ both being images
- construction: A fully convolutional neural network (CNN) is the composition of an affine linear operation, based on convolution, and a nonlinear function. The essential building block is

$$x_{i+1} = \sigma(b + \omega * x_i)$$

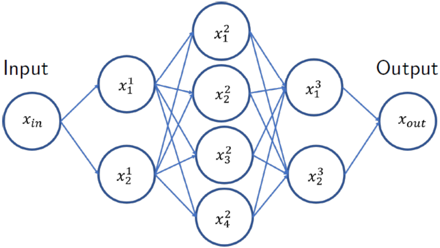


CNN (1st layer)

Each layer consists of a convolutional layer (scalar + convolution) followed by a nonlinear function

$$x_i^1 = \sigma(b_i^1 + \omega_{i,1}^1 * x_{in}), \quad i = 1, 2$$

- b: biases (scalar); ω: a convolutional filter
- σ: pointwise nonlinear function, typically $\mathrm{ReLU}(x) = \max(x, 0)$

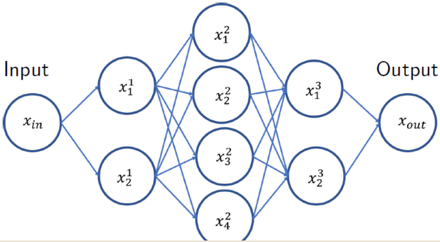


CNN (2nd layer)

Each layer consists of a convolutional layer (scalar + convolution) followed by a nonlinear function

$$x_i^2 = \sigma\left(b_i^2 + \sum_{j=1}^{2} \omega_{i,j}^2 * x_j^1\right), \quad i = 1, \ldots, 4$$

- b: biases (scalar); ω: a convolutional filter
- σ: pointwise nonlinear function, typically $\mathrm{ReLU}(x) = \max(x, 0)$

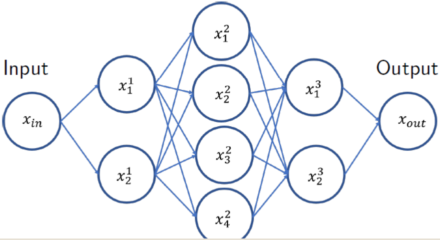


CNN (3rd layer)

Each layer consists of a convolutional layer (scalar + convolution) followed by a nonlinear function

$$x_i^3 = \sigma\left(b_i^3 + \sum_{j=1}^{4} \omega_{i,j}^3 * x_j^2\right), \quad i = 1, \dots, 2$$

- b: biases (scalar); ω: a convolutional filter
- σ: pointwise nonlinear function, typically $\mathrm{ReLU}(x) = \max(x, 0)$

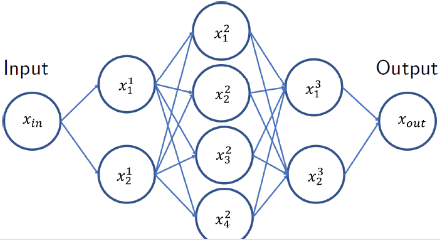


CNN (last layer)

Each layer consists of a convolutional layer (scalar + convolution) followed by a nonlinear function

$$x_o = \sigma\left(b_1^4 + \sum_{j=1}^{2} \omega_{1,j}^4 * x_j^3\right)$$

- b: biases (scalar); ω: a convolutional filter
- σ: pointwise nonlinear function, typically $\mathrm{ReLU}(x) = \max(x, 0)$



training of CNN

- denote the CNN with given parameters θ by G_θ
- With an input set $\{x^i\}$, and the desired outputs $\{x_\dagger^i\}$, seek to minimize a loss function, i.e.,

  $$\mathrm{loss}(\theta; x^i) = \|G_\theta(x^i) - x_\dagger^i\|_2^2$$

  The set of (input, output) pairs $\{(x^i, x_\dagger^i)\}$ is the training data
- training = optimization of θ, by minimizing the loss $\mathrm{loss}(\theta; \mathbf{x})$ with respect to θ over the training set

caveat:

- require a lot of training data
- training can be very expensive

the setting of general **discrete** inverse problems

$$y = A(x^\dagger) + \delta y$$

- $x^\dagger \in X$: true unknown
- $y \in Y$: measured data
- $A : X \to Y$: forward operator (maybe imperfectly known only)
- $\delta y \in Y$: data noise

parameterised reconstruction of the unknown

- given the measurement y and operator A, aim to find a mapping $A^\dagger : Y \to X$ s.t.
$$A^\dagger(y) \approx x^\dagger$$
- parameterise the mapping $A^\dagger : Y \to X$ with some parameter θ and find an optimal set of parameters θ^* s.t.
$$A^\dagger_{\theta^*}(y) \approx x^\dagger$$
- Issue: What is the input ?
- Issue: How to construct the mapping ?

gradient unrolling

given the classical minimization problem

$$x^* = \arg\min_{x \geq 0} \frac{1}{2}\|Ax - y\|_2^2 + \frac{\alpha}{2}\|x\|_2^2$$

This can be iteratively solved by a projected gradient descent

$$x_{k+1} = P_{x\geq 0}(x_k - \lambda((A^\top A + \alpha I)x_k - A^\top y))$$

which can be rewritten as

$$x_{k+1} = P_{x\geq 0}(((1 - \lambda\alpha) - \lambda A^\top A)x_k + \lambda A^\top y)$$

given the update equations

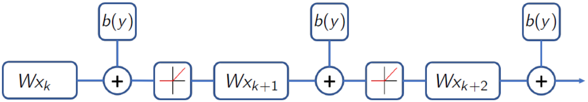
$$x_{k+1} = P_{x\geq 0}(\underbrace{((1-\lambda\alpha)-\lambda A^\top A)}_{W}\, x_k + \underbrace{\lambda A^\top y}_{b(y)})$$

Interpretation:

$$Wx_k = ((1-\lambda\alpha)-\lambda A^\top A)x_k : \text{ performs a filtering}$$

and

$$b(y) := \lambda A^\top y : \text{ data-dependent constant}$$



motivation: ISTA

given the minimization problem with a sparsity transform:

$$x^* = \arg\min_{x \geq 0} \tfrac{1}{2}\|Ax - y\|_2^2 + \alpha\|Wx\|_1$$

we can solve this with the iterative thresholding algorithm (ISTA)

$$x_{k+1} = S_{\alpha,W}(x_k - \lambda A^\top(Ax_k - y))$$

with the soft-thresholding operator S_μ,

$$S_{\alpha,W}(x) = W^{-1}S_\mu(Wx)$$

This can be written in one step as

$$x_{k+1} = W^{-1}(S_\mu W(x_k - \lambda A^\top(Ax_k - y)))$$

Consider the one-step

$$x_{k+1} = W^{-1}(S_\mu W(x_k - \lambda A^\top(Ax_k - y)))$$

we now write $a_k = Wx_k$ and apply W on both sides, we obtain

$$\begin{aligned}
Wx_{k+1} &= WW^{-1}(S_\mu W(x_k - \lambda A^\top(Ax_k - y))) \\
\Rightarrow a_{k+1} &= S_\mu W(x_k - \lambda A^\top(Ax_k - y)) \\
\Rightarrow a_{k+1} &= S_\mu(Wx_k - \lambda WA^\top(Ax_k - y)) \\
\Rightarrow a_{k+1} &= S_\mu(a_k - \lambda WA^\top(AW^{-1}a_k - y)) \\
\Rightarrow a_{k+1} &= S_\mu((a_k - \lambda WA^\top AW^{-1})a_k - \lambda WA^\top y)
\end{aligned}$$

given the update equation

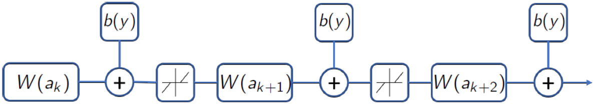
$$a_{k+1} = S_\mu((a_k - \lambda W A^\top A W^{-1}) a_k - \lambda W A^\top y)$$

note that similarly

$$W(a_k) = (I - \lambda W^\top A^\top A W^{-1}) a_k \text{ perform filtering}$$

and

$$b(y) = \lambda W A^\top y \text{ data dependent constant}$$



Interpretation: pure CNN

The general operation can be summarized as:

- iteratively filtering by W
- adding a bias $b(y)$
- applying a pointwise nonlinearity S (ReLU or soft-thresholding)

the procedure is identical with a layer in a CNN

$$x_i^{k+1} = \sigma(b_i^k + \sum_{j=1}^{m_k} \omega_{i,j}^k * x_j^k), \quad i = 1, \ldots, n$$

letting $Wx^k = \sum_{j=1}^{m_k} \omega_{i,j}^k * x_j^k$, then we have simplified

How to use CNN in inverse problems

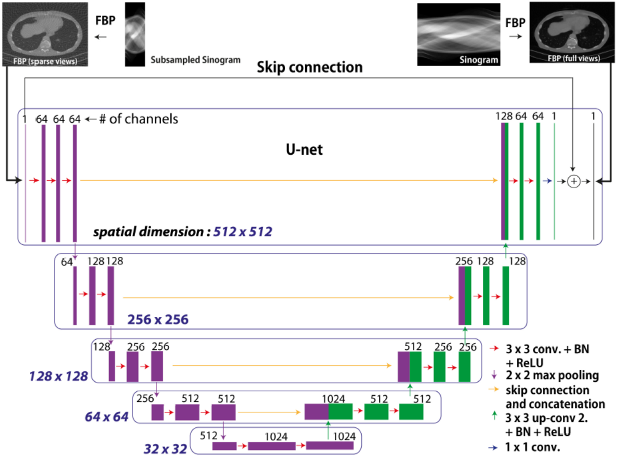
approach (i): post-processing K. H. Jin, M. T. McCann, E. Froustey IEEE Trans. Imag. Proc. 2017

- Motivated by the unrolling, use a CNN as sort of post-processing of an initial reconstruction
- given a reconstruction operator $A^\dagger : Y \to X$, such as filtered backprojection, we compute

$$x^0 = A^\dagger y$$

  and train a CNN f_θ to clean the input images (i.e., remove noise and undersampling artefacts)
- The parameters θ are given by convolution filters W and biases b

standard architecture – U-net



popularity of U-net

U-net: Convolutional networks for biomedical image segmentation

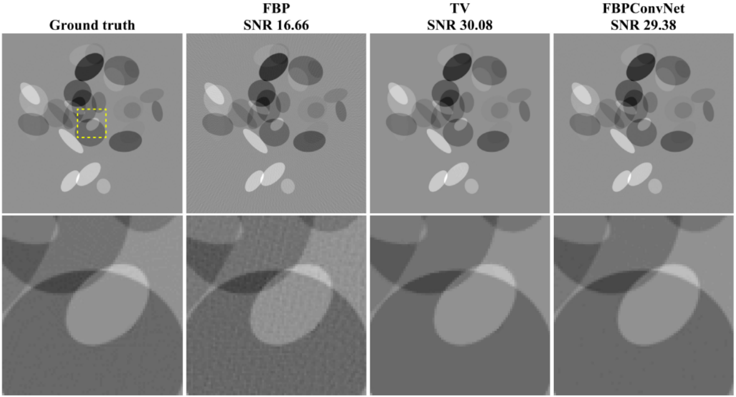
O Ronneberger, P Fischer, T Brox - International Conference on Medical …, 2015 - Sprir

… We demonstrate the application of the **u-net** to three different segmentation tasks. Th

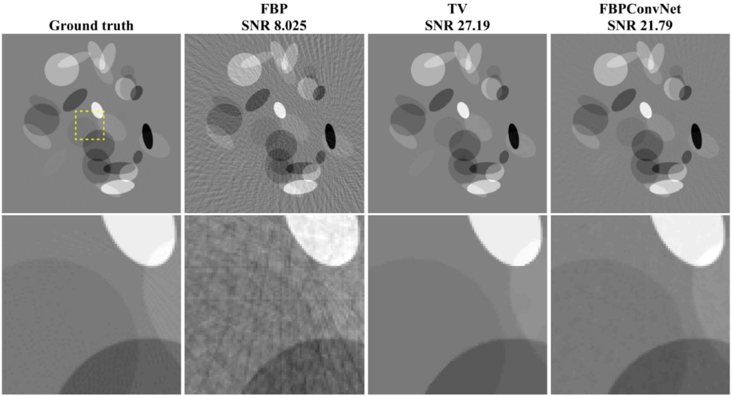
… The **u-net** (averaged over 7 rotated versions of the input data) achieves without any fı

☆ Save 99 Cite Cited by 52086 Related articles All 30 versions

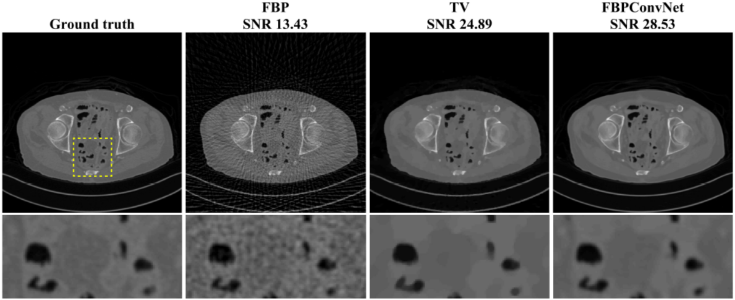
ellipse data with 143 views



ellipse data with 50 views



reconstruction on biomedical data set with 50 views



How to use CNN in inverse problems

approach (ii): learned model based reconstruction

- aim: model-based learning / reconstruction: in this approach, the forward and adjoint operators of the imaging problem are used directly in the recovery procedure.
- the general problem

$$x^* = \arg\min_{x \geq 0} \|Ax - y\|_2^2 + \alpha \mathcal{R}(x)$$

where $\mathcal{R}$ denotes some regularisation term.

- This leads to the gradient descent scheme

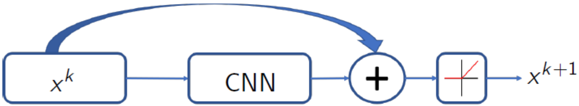
$$x^{k+1} = P_{x \geq 0}(x^k - \lambda(A^\top (Ax_k - y) + \alpha \nabla \mathcal{R}(x^k)))$$

including the model in the learned algorithm: ResNet

- without the data fit, we only have

$$x^{k+1} = P_{x\geq 0}(x^k - \lambda\alpha\nabla\mathcal{R}(x^k))$$

- This is similar to ResNet structure, with the convolutional part learning the action of $\lambda\alpha\nabla\mathcal{R}(x^k)$



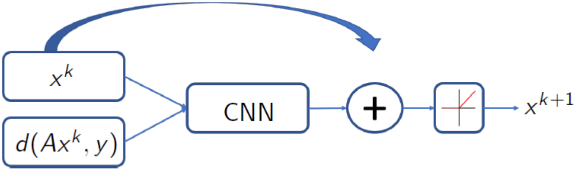
- add the model information (i.e., the data fit)

$$d(y, Ax) = \|Ax - y\|_2^2$$

- the update equation reads

$$x^{k+1} = P_{x\geq 0}(x^k - \lambda(\nabla d(y, Ax^k) + \alpha \nabla \mathcal{R}(x^k)))$$

- including the gradient of the data fit modifies the ResNet to a simple model based learned gradient descent network (each block $G^k_{\theta_k}$)



J. Adler, O. Öktem, Inverse Problems 2017; IEEE Trans. Med. Imag. 2018

The algorithm can be summarized as J. Adler, O. Öktem, Inverse Problems 2017; IEEE Trans. Med. Imag. 2018

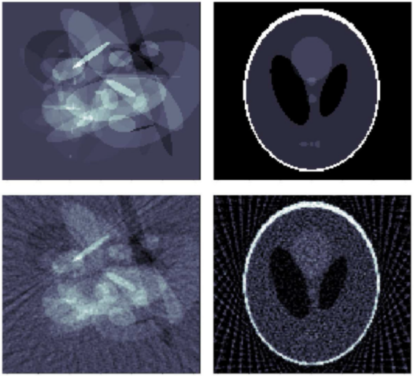
Algorithm 1 Learned gradient descent

function $G_\theta(y)$
    $x^0 \leftarrow A^T y$
    $k \leftarrow 0$
    for $k < k_{max}$ **do**
        Compute $\nabla d(y, Ax^k) = A^*(Ax^k - y)$
        $x^{k+1} \leftarrow G^k_{\theta_k}(\nabla d(y, Ax^k), x^k)$
        $k \leftarrow k + 1$
    end for
end function(**return** $x^{k_{max}}$)

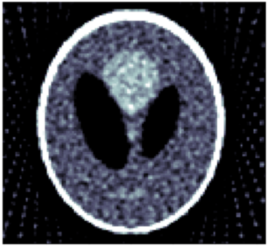
create a training set: train the network on phantoms of random ellipses by minimizing

$$\mathrm{loss}(\theta; x^K) = \|G_\theta(y) - x^\dagger\|_2^2$$

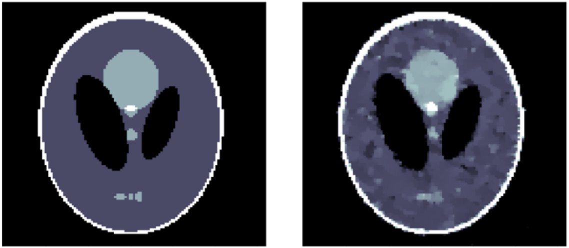
and test the performance on the Shepp Logan phantom



results by filtered back projection (FBP)



results by total variation



results by post-processing

results by learned gradient descent

results by learned primal dual

quantitative comparative results

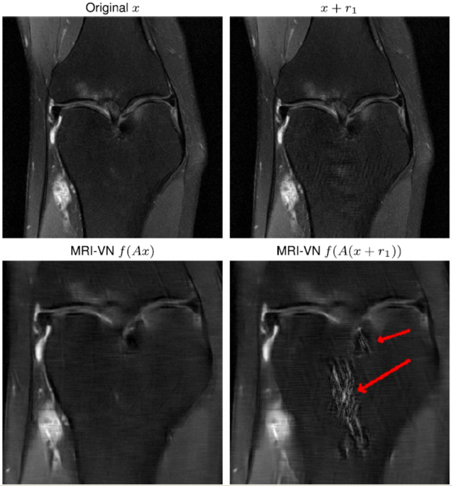
Method	PSNR (dB)	SSIM	Runtime (ms)	Parameters
FBP	19.75	0.597	4	1
TV	28.06	0.928	5 166	1
Learned U-Net	29.20	0.943	9	10^7
Learned Gradient Descent	32.02	-	-	-
Learned Primal-Dual	38.28	0.988	49	$2.4 \cdot 10^5$

The peril of training data

sensitivity of classification neural networks

K. Eykholt et al CVPR 2018

The peril of training data



The need for model adaptation

setting of the problem D. Gilton, G. Ongie, R. Willett. IEEE Trans. Comput. Imag. 2021

- Suppose we have neural network estimator $\hat{x} = f_0(y)$ trained to solve

$$y = A_0 x + \epsilon, \quad x \sim P_X, \epsilon \sim P_{N_0}$$

- concerned task:

$$y = A_1 x + \epsilon', \quad x \sim P_X, \epsilon' \sim P_{N_1}$$

  with A_1 known, or partially unknown (e.g., additional param. in A_1)

- goal: can we adapt / modify the estimator $\hat{x} = f_0(y)$ to the new task ?

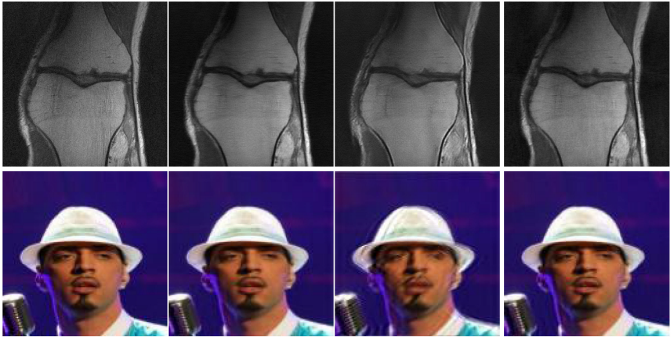
transfer learning approach

- naive approach: apply the neural network directly, with $f(\cdot;\theta_0^*,A_0)$ by $f(\cdot;\theta_0^*,A_1)$
- adaptation: estimate the image by $f(y;\theta_1^*,A_1)$, with θ_1^* solving

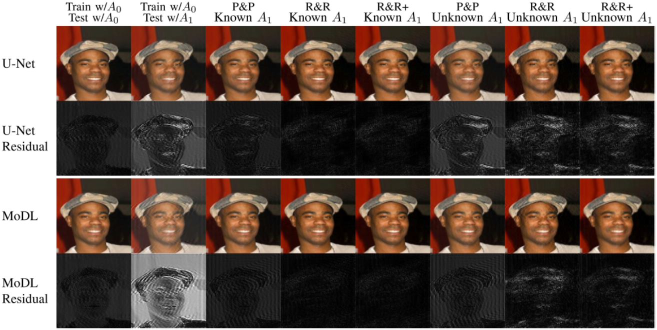
$$\min_\theta \underbrace{\|y - Af(y;\theta,A_1)\|^2}_{\text{data consistency}} + \mu \underbrace{\|\theta-\theta_0^*\|^2}_{\text{constraint param.}}$$

  with $\mu > 0$ is a tunable parameter
- with training dataset

$$(\theta_1^*, A_1^*) = \arg\min_{\theta, A\in\mathcal{A}} \frac{1}{N}\sum_{i=1}^{N} \|y_i - Af(y_i;\theta,A)\|^2 + \mu\|\theta-\theta_0^*\|^2$$



(a) Ground truth (b) No model drift (c) Model drift (d) Model drift w/model adaptation


Train w/A_0 Test w/A_0
Train w/A_0 Test w/A_1
P&P Known A_1
R&R Known A_1
R&R+ Known A_1
P&P Unknown A_1
R&R Unknown A_1
R&R+ Unknown A_1
U-Net
U-Net Residual
MoDL
MoDL Residual

Deep equilibrium model

state of the art

- modern feedforward deep networks build on the concept of layers: each network is a stack of L transformations, L being the depth (determined by model designer)
- update: backpropagation through L layers via chain rule
- requires storing all intermed. values (activations) of these layers $\Rightarrow$ memory requirement is proportional to L

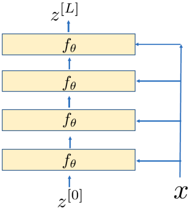
What happens if you iterate more ?

at iter 1, 10, 20, 30, 40

D. Gilton, G. Ongie, R. Willett. IEEE Trans. Comput. Imag. 2021

weight-tied deep sequence models:

$$z_{1:T}^{[i+1]} = f_\theta(z_{1:T}^{[i]}; x_{1:T}), \quad i = 0, \dots, L-1,$$
$$z_{1:T}^{[0]} = 0, G(x_{1:T}) = z_{1:T}^{[L]}$$

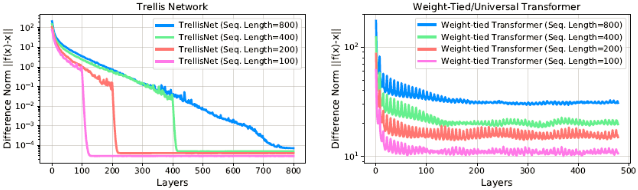


features

- reduce the model size, a form of regularization
- any deep network can be represented by a weight-tied deep network of equal depth and linear increase in width
- the network can be unrolled to any depth (improved feature extraction as depth increases)
- in practice only fixed layers, due to memory on training hardware

hypothesis: the weight-tied models converge to a fixed point, as the depth increases to infinity

$$z_{1:T}^* = \lim_{i\to\infty} z_{1:T}^{[i]} = \lim_{i\to\infty} f_\theta\left(z_{1:T}^{[i]}; x_{1:T}\right) = f_\theta\left(z_{1:T}^*; x_{1:T}\right)$$



S. Bai, J. Z. Kolter, and V. Koltun, NeurIPS 2019 eventual hidden layer values of an infinite depth network

$$z^*_{1:T} = f_\theta(z^*_{1:T}; x_{1:T})$$

deep equilibrium model (DEM): compute the fixed point $z^*_{1:T}$ directly using black-box root-finding, via implicit differentiation

- fixed point procedure

$$z^{[i+1]}_{1:T} = f_\theta(z^{[i]}_{1:T}; x_{1:T}), \quad i = 0, 1, \ldots$$

- alternative method (quasi-Newton method)

S. Bai, J. Z. Kolter, and V. Koltun, NeurIPS 2019 equilibrium equation

$$f_\theta(z^*_{1:T}; x_{1:T}) = z^*_{1:T}$$

let $(\cdot)$ be a parameter of $f_\theta(z^*_{1:T}; x_{1:T})$ (e.g., θ or $x_{1:T}$) implicit diff. $\Rightarrow$

$$\frac{\partial z^*_{1:T}}{\partial(\cdot)} = \frac{\partial f_\theta(z^*_{1:T}; x_{1:T})}{\partial(\cdot)}$$

$$\frac{\partial z^*_{1:T}}{\partial(\cdot)} = \frac{\mathrm{d} f_\theta(z^*_{1:T}; x_{1:T})}{\mathrm{d}(\cdot)} + \frac{\partial f_\theta(z^*_{1:T}; x_{1:T})}{\partial z^*_{1:T}} \frac{\partial z^*_{1:T}}{\partial(\cdot)}$$

$$\left(I - \frac{\partial f_\theta(z^*_{1:T}; x_{1:T})}{\partial z^*_{1:T}}\right) \frac{\partial z^*_{1:T}}{\partial(\cdot)} = \frac{\mathrm{d} f_\theta(z^*_{1:T}; x_{1:T})}{\mathrm{d}(\cdot)}$$

fixed point equation: $g_\theta(z^*_{1:T}) = f_\theta(z^*_{1:T}; x_{1:T}) - z^*_{1:T}$

$$J_{g_\theta}|_{z^*_{1:T}} = -\left(I - \frac{\partial f_\theta(z^*_{1:T}; x_{1:T})}{\partial z^*_{1:T}}\right)$$

$$\frac{\partial z^*_{1:T}}{\partial(\cdot)} = -(J^{-1}_{g_\theta}|_{z^*_{1:T}}) \frac{\mathrm{d} f_\theta(z^*_{1:T}; x_{1:T})}{\mathrm{d}(\cdot)}$$

do not backprop through all layers

backprop through equilibrium point

$$\frac{\partial \ell}{\partial(\cdot)} = \frac{\partial \ell}{\partial z^*_{1:T}} \frac{\partial z^*_{1:T}}{\partial(\cdot)} = -\frac{\partial \ell}{\partial z^*_{1:T}} (J^{-1}_{g_\theta}|_{z^*_{1:T}}) \frac{\mathrm{d} f_\theta(z^*_{1:T}; x_{1:T})}{\mathrm{d}(\cdot)}$$

- backward gradient through infinite stacking can be represented as one-step matrix multiplication (Jacobian)
- SGD

$$\theta^+ = \theta - \alpha \frac{\partial \ell}{\partial \theta} = \theta + \frac{\partial \ell}{\partial z^*_{1:T}} (J^{-1}_{g\theta}|_{z^*_{1:T}}) \frac{\mathrm{d} f_\theta(z^*_{1:T}; x_{1:T})}{\mathrm{d}(\cdot)}$$

it does not depend on rootfinding procedure, structure of transformation, no intermediate value reliable estimation of equilibrium point

- use any black-box root-finding method, e.g., quasi-Newton (Broyden) method for

$$g_\theta(z^*_{1:T}; x_{1:T}) = f_\theta(z^*_{1:T}; x^*_{1:T}) - z^*_{1:T} \to 0.$$

- Broyden iterations

$$z^{i+1}_{1:T} = z^i_{1:T} - \alpha B g_\theta(z^i_{1:T}; x_{1:T}), \quad i = 0, 1, \ldots$$

with step size α, $B \approx J^{-1}_{g_\theta}|_{z^i_{1:T}}$ (Jacobian approx.)

overall computational strategy:

- forward pass: $z_{1:T}^* = \text{RootFind}(g_\theta; x_{1:T})$
- backward pass (implicit diff.):

$$\frac{\partial \ell}{\partial (\cdot)} = -\frac{\partial \ell}{\partial z_{1:T}^*}(J_{g_\theta}^{-1}|_{z_{1:T}^*})\frac{\mathrm{d} f_\theta(z_{1:T}^*; x_{1:T})}{\mathrm{d}(\cdot)}$$

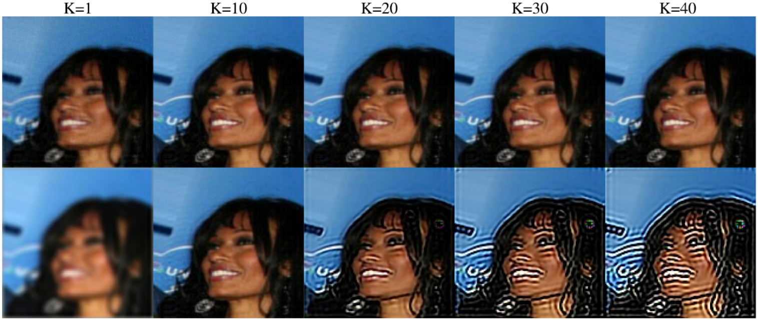
- accelerating DEM: to compute $-\frac{\partial \ell}{\partial z_{1:T}^*}(J_{g_\theta}^{-1}|_{z_{1:T}^*})$ solve a linear system with quasi-Newton method

$$(J_{g_\theta}^{\mathrm{T}}|_{z_{1:T}^*})y^{\mathrm{T}} + \left(\frac{\partial \ell}{\partial z_{1:T}^*}\right)^{\mathrm{T}} = 0.$$

properties

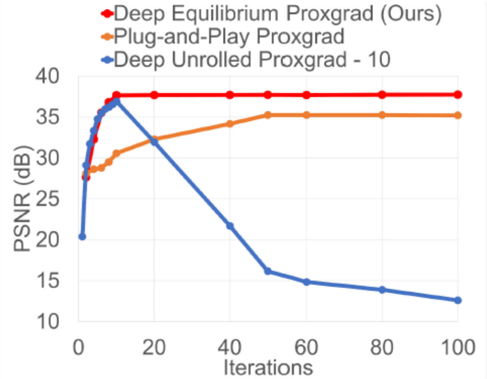
- memory cost: $z_{1:T}$, $x_{1:T}$, f_θ (and matrix-vector product)
- the analysis is independent of θ, but should be stable and constrained
- universality of ”single-layer” DEQs

performance on image recovery



D. Gilton, G. Ongie, R. Willett. IEEE Trans. Comput. Imag. 2021

performance on image recovery



(b) Deblurring (2)

D. Gilton, G. Ongie, R. Willett. IEEE Trans. Comput. Imag. 2021

There are many exciting opportunities in developing algorithms and theories

- the choice of architecture is crucial
- the choice of training data is crucial
- the choice of training algorithm is crucial
- the specific physics / mathematical properties are important
- the theory is scarce
- ...