New Techniques in Optimization and Their Applications to Deep Learning and Related Inverse Problem

Stanley J. Osher

February 8, 2018

Deep Relaxation: PDE's for Optimizing Deep Neural Nets

Joint with: P. Chaudhari, A. Oberman, S. Soatto and G. Carlier

February 8, 2018

Obtain a piecewise convex envelope of a highly nonconvex, high dimensional function via PDE's.

Our first observation: Local Entropy corresponds to regularization by a viscous Hamilton-Jacobi PDE.

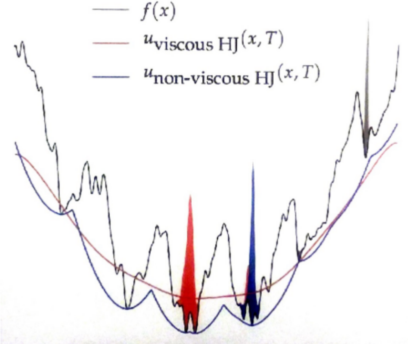


Figure: True solution in one dimension. (Cartoon in high dimensions, because algorithm only works for shorter times.)

Started with:

Entropy SGD: Biasing Gradient Descent Into Wide Valleys, by: P. Chaudhari, P. Choromanska, S. Soatto, Y. LeCun, Y. Baldassi, C. Borgs, J. Chayes, L. Sagun and R. Zecchina

Minimize highly non convex high dimensional $f(x)$, $x \in \mathbb{R}^n$, n large.

Replace $f(x)$ by $f_t(x)$

$$f_t(x) = -\log(G_t * e^{f(x)})$$

$$G_t(x) = Ce^{-\frac{||x||^2}{2t}}$$

$\int G_t(x)dx = 1$, determines C

"local entropy"

(1) for small t, we can evaluate ∇f_t efficiently.

(2) scoping: vary t

first t small,

later t large

"widens minima".

Revaluation!!

$$f_t(x) = u(x,t)$$

where u satisfies a viscous Hamilton-Jacobi Burgers' PDE

$$\begin{cases} u_t + \frac{1}{2}||\nabla_x u||^2 = \frac{1}{2}\Delta_x u & t > 0 \\ u(x,0) = f(x) & , \end{cases}$$

Wow!!

This can be shown via the Cole-Hopf transformation.

Proof:
Let $u(x,t) = -\log v(x,t)$
Can show

$$v_t = \frac{1}{2}\Delta_x v$$

$$v(x,0) = e^{-f(x)}$$

$$v(x,t) = G_t * e^{-f}$$

$$u(x,t) = -\log(G_t * e^{-f})$$

Details

$$u(x,t) = -\frac{1}{\beta}\log v(x,t)$$

$$v(x,t) = e^{-\beta u(x,t)}$$

$$v_t = -\beta u_t v$$

$$\nabla v = -\beta v(\nabla u)$$

$$\Delta v = -\beta v \Delta u + \beta^2 v||\nabla u||^2$$

$$v_t - \Delta\frac{v}{2} = -\beta v[u_t + \beta||\frac{\nabla u}{2}||^2 - \frac{\Delta u}{2}] = 0 \text{ if } \beta = 1.$$

This "widens" local minima regions, "narrows" local maxima regions, (but raises minima a bit)
also

$$-\nabla f_t(x) = \frac{1}{t}\int (x-y)\rho^{\infty}(y,x)dx$$

$$\rho^{\infty}(y,x) = \frac{1}{z(x)} e^{-\left(f(y) - \frac{||x-y||^2}{2t}\right)}$$

Better Procedure:

Viscosity solution to inviscid H-J PDE Burgers'

$$u_t + \frac{||\nabla u||^2}{2} = 0$$

$$u(x,0) = f(x)$$

Lax-Oleinik formula

$$u(x,t) = \min_y \left\{ f(y) + \frac{1}{2t} |||x-y|^2 \right\}$$

This is $\frac{1}{t}$ times Moreau Envelope of tf

$$= \frac{1}{t} \min \left\{ tf(y) + \frac{1}{2} |||x-y|^2 \right\}$$

$$= inf_{convolutionof} \left(f(y), \frac{1}{2t} ||y||^2 \right)$$

$$argmin = y(x,t)$$

Or: $Proximal_{tf}(x) = y(x,t)$.
Proximal Method:

$$x_{k+1} = argmin \left\{ f(y) + \frac{1}{2\Delta t_k} |||y - x_k|^2 \right\}, k = 1, 2, \cdots$$

small Δt_k early, large Δt_k later.

So $$f(x_{k+1}) + \frac{1}{2\Delta t_k} \left\|x_{k+1} - x_k\right\|^2 = u(x_k, \Delta t_k).$$

$f(x_j) \downarrow$, but might go to a local minimum.

The bigger the Δt_k, the more convex $u(x, \Delta t_k)$ is in x.

Also: $$x_{k+1} = x_k - \Delta t_k \nabla f\,(x_{k+1})$$

Backward Euler!!

And

$$\nabla u(x_k, \Delta t_k) = -\frac{x_{k+1} - x_k}{\Delta t_k} = \nabla f(x_{k+1})$$

We have

$$f(x_{k+1}) + \frac{1}{2\Delta t_k} \; \left\|x_{k+1} - x_k\right\|^2 \leq f(x_k)$$

$$\begin{aligned}
\Rightarrow \quad f(x_k) &\leq f(x_0) - \sum_{j=1}^{k} \frac{1}{2\Delta t_j} \; \left\|x_j - x_{j-1}\right\|^2 \\
&= f(x_0) - \sum_{j=1}^{k} \frac{\Delta t_j}{2} \left\|\nabla f(x_j)\right\|^2
\end{aligned}$$

Let, $t(n) = \sum_{j=1}^{n} \Delta t_j \Rightarrow \nabla f(x_n) = 0 \left(\frac{1}{(t(n))^{\frac{1}{2}}}\right)$

Converges to a (perhaps) local minimum if $t(n) \to \infty$.

"Widening minima", "Removing Maxima"

Consider 1 dimension, for simplicity only

$$u_t + \frac{(u_x)^2}{2} = 0, \qquad u(x,0) = f(x)$$

Let $w(x,t) = u_x\,(x,t)$.

Conservation Law: Burgers' equation

Solution: $w(x,t) = f'(x - wt)$

Classical solution until characteristics intersect. Until the first t for which $1 + t^*\, f''(x - wt^*) = 0$

Let $f(x)$ be convex for $x_0 \leq x \leq x_1$ concave elsewhere

$f''(x_0) = f''(x_1) = 0$

let $$p = w = u_x\,(x, t)$$

$$p_x = f''(x - tp)(1 - tp_x)$$

$$p_x = \frac{f''(x-tp)}{1+tf''(x-tp)}$$

$$p_x > 0 \text{ if } f''(x - tp) > 0$$

$$p_x < 0 \text{ if } f''(x - tp) < 0$$

$$p_x = 0 \text{ if } x - tp = x_1$$

or $x = x_1 + tp > x,$ because $p = u_x\,(x, t) > 0$

This means $u_{xx}(x,t) > 0$ for $x_1 < x$,

The convex region has moved to the right, past its original end point, x_1.

Similarly it moves to the left past its original end point x_0.

"Widens" minimal regions

"Narrows" (and shrinks) maxima

Intuition: Rarefaction waves in inviscid Burgers spread out and widen minima.

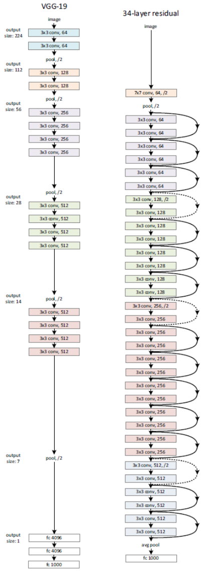
Shock waves collapse to N waves and maxima disappeared!

Deep Learning with Data Dependent Implicit Activation Functions

Stanley J. Osher
Joint with: Bao Wang and professor Zuoqiang Shi

February 8, 2018

Two Important Types of Deep Neural Networks



(a) (b)

Figure: VGG19 v.s. ResNet34

- VGG: K. Simonyan and A. Zisserman, ICLR, 2015.
  $>$ 8900 citations.
- ResNet: He et al, CVPR, 2016.
  $>$ 6000 citations.

Wide applications toward real AI!

- AlphaGo, AlphaGo Zero.
- Autonomous Car.
- Healthcare.
- Many others.

ResNet v.s. Plain Network

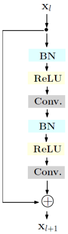


Figure: Residual block.

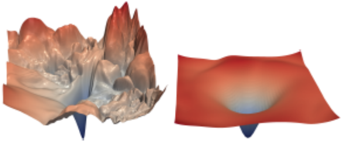


Figure: Energy landscape of plain Network and ResNet.

He et al, CVPR, 2016
Li et al, Arxiv1712.09913, 2018.

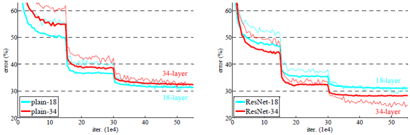


Figure: Performance on ImageNet: ResNets v.s. Plain Networks. Thin line: Training; Thick line: Testing.

Deeper is better if the network is appropriately designed!

ResNet and PDE based Control Problen

Residual Block

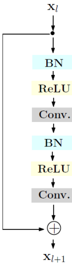


Figure: Residual block.

$$\mathbf{x}_{l+1} = \mathcal{F}(\mathbf{x}_l, \{\mathbf{W}_i\}) + \mathbf{x}_l.$$

Residual block: Discrete dynamical system.

Control Problem of the Transport Equation

$$\begin{cases} \frac{\partial u}{\partial t} + \mathbf{v}(\mathbf{x}, t) \cdot \nabla u(\mathbf{x}, t) = 0 & \mathbf{x} \in \mathbb{R}^d, t \geq 0 \\ u(\mathbf{x}, 1) = f(\mathbf{x}) & \mathbf{x} \in \mathbb{R}^d \\ u(\mathbf{x}_i, 0) = g(\mathbf{x}_i) & \mathbf{x}_i \in \mathbf{T}, \end{cases}$$

where $\mathbf{T}$ denotes the training set, $g(\mathbf{x}_i)$ is the label of instance $\mathbf{x}_i$.

ResNet and PDE based Control Problem

Let $f(\mathbf{x}) = \textbf{softmax}(\mathbf{x})$, with $\textbf{softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$.

And if we choose the velocity field such that

$$\Delta t\, \mathbf{v}(\mathbf{x}, t) = \mathbf{W}^{(2)}(t) \cdot \sigma \left(\mathbf{W}^{(1)}(t) \cdot \sigma(\mathbf{x}) \right),$$

where $\mathbf{W}^{(1)}(t)$ and $\mathbf{W}^{(2)}(t)$ corresponds to the 'weight' layers in the residual block, $\sigma = \mathrm{ReLU} \circ \mathrm{BN}$, Δt is the time step size in discretizing the control problem.

ResNet can be considered as a forward Euler solver to the control problem.

We consider alternative terminal functions!

Manifold Interpolation-Implicit Activation

Let $P = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n\}$ be a set of points on a manifold $\mathcal{M} \subset \mathbf{R}^d$ with the labeled subset $S = \{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_m\}$.

How to extend the labels of S to P?

Harmonic extension by minimizing the Dirichlet energy:

$$\mathcal{E}(u) = \frac{1}{2} \sum_{\mathbf{x},\mathbf{y} \in P} w(\mathbf{x}, \mathbf{y}) \left(u(\mathbf{x}) - u(\mathbf{y})\right)^2 ,$$

with the boundary condition:

$$u(\mathbf{x}) = g(\mathbf{x}),\ \mathbf{x} \in S,$$

The Euler-Lagrange equation for the above energy minimization problem is:

$$\begin{cases} \sum_{\mathbf{y} \in P} \left(w(\mathbf{x}, \mathbf{y}) + w(\mathbf{y}, \mathbf{x})\right) \left(u(\mathbf{x}) - u(\mathbf{y})\right) = 0 & \mathbf{x} \in P/S \\ u(\mathbf{x}) = g(\mathbf{x}) & \mathbf{x} \in S, \end{cases}$$

We infer the label implicitly!

Manifold Interpolation-Implicit Activation

How about only tiny amount of data is labeled?

$$\begin{cases} \sum_{\mathbf{y}\in P}\left(w(\mathbf{x},\mathbf{y})+w(\mathbf{y},\mathbf{x})\right)\left(u(\mathbf{x})-u(\mathbf{y})\right)+ & \\ \left(\frac{|P|}{|S|}-1\right)\sum_{\mathbf{y}\in S} w(\mathbf{y},\mathbf{x})\left(u(\mathbf{x})-u(\mathbf{y})\right)=0 & \mathbf{x}\in P/S \\ u(\mathbf{x})=g(\mathbf{x}) & \mathbf{x}\in S, \end{cases}$$

we use the **weighted nonlocal Laplacian (WNLL)** instead of the graph Laplacian (GL)!

Shi et al, JSC, 2017

How many instances should be labeled at least?

$$N\left(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{N}\right)\approx N\ln N,$$

where N is the number of classes in the dataset.

How to find the weight function w?
Approximate nearest neighbor (ANN) searching!

Muja et al, PAMI, 2014.

Network Structure Design

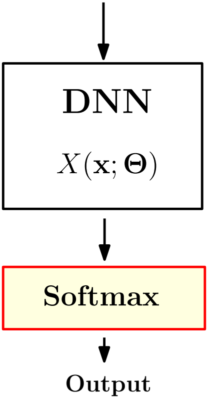


Figure: Vanilla Deep Neural Network.

Network Structure Design

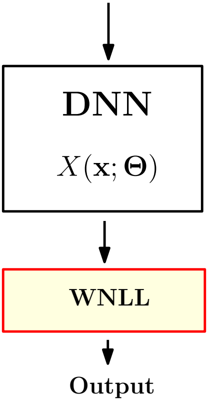


Figure: Deep Neural Network with the WNLL Activation.

Error cannot be back propagated, since the WNLL is an implicit function whose gradient is not explicitly available!

Network Structure Design

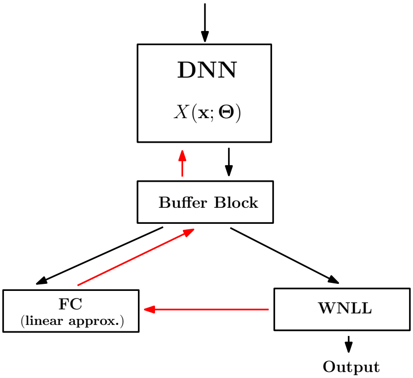


Figure: Deep Neural Network with WNLL Activation.

Training Algorithm

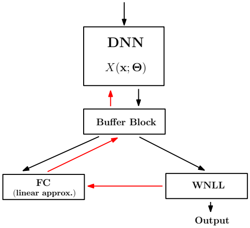


Figure: Deep Neural Network with WNLL Activation.

Alternating between the following three steps:

- **Step 1.** Train the network with only the linear activation functions to steady state. For this purpose, we do not feed the data to the WNLL activation.
- **Step 2.** Run a few training epochs on the network which we freeze the "DNN" and "Linear Activation" blocks, and only fine tune the 'Buffer Block'. In order to back-propagate the error between the ground-truth and the WNLL interpolated results, we feed the data into the pre-trained linear activation function, and use the corresponding computational graph to perform error back-propagation.
- **Step 3.** Unfreeze the entire network, and train the network with data only feeding to the linear activation to the steady state again.

Numerical Results

(a) (b)

Figure: CIFAR image recognition tasks.

Accuracy of Some Simple Classifiers

Table: Accuracy of some simple classifiers over different datasets

Dataset	**KNN**	**SVM (RBF Kernel)**	**Softmax**	**WNLL**
Cifar10	32.77% (k=5)	**57.14%**	39.91%	**40.73%**
MNIST	96.40% (k=1)	**97.79%**	92.65%	**97.74%**
SVHN	41.47% (k=1)	**70.45%**	24.66%	**56.17%**

Accuracy Evolution

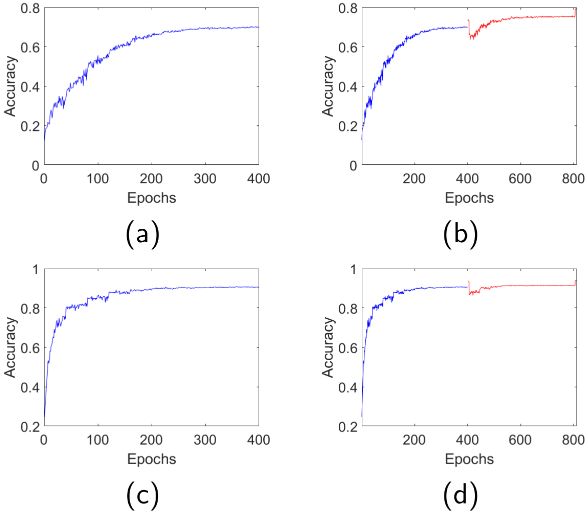


Figure: The evolution of the generation accuracy over the training procedure. Charts (a) and (b) are the accuracy plots for ResNet50 with 1000 number of data for training, where (a) and (b) are plots for the epoch v.s. accuracy of vanilla and WNLL activated DNN. Panels (c) and (d) corresponding to the case of 10000 training data for PreActResNet50. All test are done on Cifar10 dataset.

Degradation of DNN when Lack of Training Data

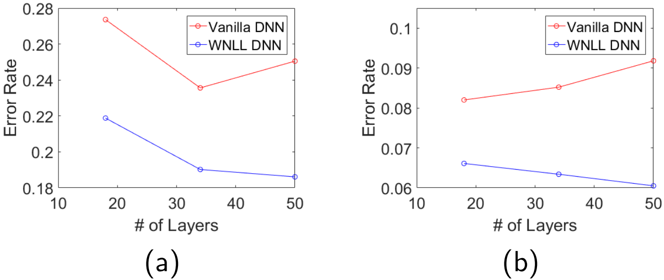


Figure: Taming of the degeneration problem of vanilla DNN by WNLL activated DNN. Panels (a) and (b) plot the generation error for cases when 1000 and 10000 training data is used to train the vanilla and WNLL activated DNN, respectively. In each plot, we test three different networks: PreActResNet18, PreActResNet34, and PreActResNet50. It is easy to see that when the vanilla network becomes deeper, the generation error does not decayed, while WNLL activation resolves this degeneracy. All tests are done on Cifar10 dataset.

Performance on CIFAR10

Table: Generalization error rate over the whole test set of vanilla DNNs and WNLL activated ones trained over the entire and first 10000, 5000, and 1000 instances of the training set of CIFAR10. (Median of 5 independent trials)

Network	Whole		10000		5000		1000	
	Vanilla	**WNLL**	**Vanilla**	**WNLL**	**Vanilla**	**WNLL**	**Vanilla**	**WNLL**
ResNet20	9.06%	**7.09%**	12.83%	**9.96%**	14.30%	**11.24%**	34.90%	**29.91%**
ResNet32	7.99%	**5.95%**	11.18%	**8.15%**	12.75%	**10.63%**	33.41%	**28.87%**
ResNet44	7.31%	**5.70%**	10.66%	**7.96%**	11.84%	**10.14%**	34.58%	**27.94%**
ResNet56	7.24%	**5.61%**	9.83%	**7.61%**	12.39%	**10.17%**	37.83%	**28.18%**
ResNet110	6.41%	**4.98%**	8.91%	**7.13%**	13.45%	**10.05%**	42.94%	**28.29%**
ResNet18	6.16%	**4.65%**	8.26%	**6.29%**	10.38%	**8.53%**	27.02%	**22.48%**
ResNet34	5.93%	**4.26%**	8.31%	**6.11%**	10.75%	**8.65%**	26.47%	**20.27%**
ResNet50	6.24%	**4.17%**	9.64%	**6.49%**	12.96%	**8.76%**	29.69%	**20.19%**
PreActResNet18	6.21%	**4.74%**	8.20%	**6.61%**	10.64%	**8.18%**	27.36%	**21.88%**
PreActResNet34	6.08%	**4.40%**	8.52%	**6.34%**	10.85%	**8.44%**	23.56%	**19.02%**
PreActResNet50	6.05%	**4.27%**	9.18%	**6.05%**	10.64%	**8.35%**	25.05%	**18.61%**

Performance on CIFAR100

Table: Error rate of vanilla DNN v.s. WNLL activated DNN over the whole Cifar100 dataset. (Median of 5 independent trials)

Network	Vanilla DNN	WNLL DNN
ResNet20	35.79%	**31.53%**
ResNet32	32.01%	**28.04%**
ResNet44	31.07%	**26.32%**
ResNet56	30.03%	**25.36%**
ResNet110	28.86%	**23.74%**
ResNet18	27.57%	**22.89%**
ResNet34	25.55%	**20.78%**
ResNet50	25.09%	**20.45%**
PreActResNet18	28.62%	**23.45%**
PreActResNet34	26.84%	**21.97%**
PreActResNet50	25.95%	**21.51%**

DNN with SVM Classifier

Table: Error rate of SVM classifier on the deep learning features from vanilla DNN v.s. vanilla DNN over the whole Cifar10 dataset. (Not end-to-end)

Network	Vanilla DNN	SVM+DNN
VGG11	9.23%	9.70%
VGG13	6.66%	9.66%
VGG16	6.72%	9.70%
VGG19	6.95%	10.11%
ResNet18	6.16%	8.99%
ResNet34	5.93%	8.72%
ResNet50	6.24%	9.17%
PreActResNet18	6.21%	9.16%
PreActResNet34	6.08%	9.00%
PreActResNet50	6.05%	9.02%

Stronger classifier does not improve accuracy of DNN!

Summary

- DNN with data dependent implicit activation.
- Back propagate the gradient of harmonic function by linear function.
- Resolve the degradation problem.
- Relatively 20%-30% accuracy improvement on both CIFAR10 and CIFAR100.
- Reduce the model's size.
- **On going: imageNet challenge: random interpolation.**

Ref: B. Wang, X. Luo, Z. Li, W. Zhu, Z. Shi, and S. Osher, Deep Neural Networks with Data Dependent Implicit Activation Function, Arxiv 1802.00168

BinaryRelax: A Relaxation Approach For Training Deep Neural Networks with Quantized Weights

Joint with: P. Yin, S. Zhang, J. Lyu, Y. Qi and J. Xin

February 8, 2018

Quantized Deep Neural Networks

- Huge number of floating-point weights (parameters) pose challenges to the deployment of DNNs on small mobile devices of limited storage and power.
- Floating-point weights are not essential to achieve good accuracy.
- Benefits from quantized (low-bit) weights:
  - Less storage.
  - Faster inference.
  - Higher energy efficiency.
- There have been new processors for AI applications featuring 8-bit vector operations.

Mathematical Formulation

The training of quantized networks can be abstracted as the constrained optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) := \frac{1}{N} \sum_{i=1}^{N} \ell_i(x) \quad \text{subject to} \quad x \in \mathcal{Q}. \tag{1}$$

- $\ell_i(x)$: the loss associated with the i-th training sample.
- $\mathcal{Q} = \mathbb{R}_+ \times \{\pm q_1, \dots, \pm q_m\}^n$: the set of quantized weights with m quantization levels.
  - 1-bit binarization: $\mathcal{Q} = \mathbb{R}_+ \times \{\pm 1\}^n$
  - 2-bit ternarization: $\mathcal{Q} = \mathbb{R}_+ \times \{0, \pm 1\}^n$
  - b-bit quantization: $\mathcal{Q} = \mathbb{R}_+ \times \{0, \pm 1, \dots, \pm(2^{b-1} - 1)\}^n$

Quantization Step

The quantization of a float weight vector y gives rise to the projection problem

$$\mathrm{proj}_{\mathcal{Q}}(y) := \arg\min_{x \in \mathcal{Q}} \|x - y\|^2,$$

equivalent to the constrained K-means clustering:

$$(s^*,\ Q^*) = \arg\min_{s,Q} \|s \cdot Q - y\|^2$$
$$\text{subject to} \quad s > 0,\ Q \in \{\pm q_1, \ldots, \pm q_m\}^n.$$

Then $\mathrm{proj}_{\mathcal{Q}}(y) = s^* \cdot Q^*$. The standard approach Lloyd's algorithm is impractical here. Analytic solutions exist for binarization and ternarization. Empirical schemes are available for bit-width ≥ 2.

Moreau Envelope

- The Moreau envelope g_t of $g(x)$ is defined by

$$g_t(x) := \inf_{z \in \mathbb{R}^n} g(z) + \frac{1}{2t}\|z - x\|^2.$$

  g_t is locally Lipschitz continuous, and converges pointwise to g as $t \to 0^+$.

- Moreau envelope is closely related to the inviscid Hamilton-Jacobi equation

$$u_t + \frac{1}{2}\,|\,\nabla_x u\,|^2 = 0, \quad u(x,0) = g(x),$$

  where $u(x,t) = g_t(x)$ is the unique viscosity solution via the Hopf-Lax formula.

Relaxation by Moreau Envelope

Training quantized DNNs:

$$\min_{x \in \mathbb{R}^n} f(x) + \chi_{\mathcal{Q}}(x), \tag{2}$$

where $\chi_{\mathcal{Q}}(x)$ is the characteristic function of $\mathcal{Q}$ (**discontinuous**):

$$\chi_{\mathcal{Q}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{Q} \\ \infty & \text{otherwise.} \end{cases}$$

The Moreau envelope of $\chi_{\mathcal{Q}}$ is given by

$$\inf_{z} \chi_{\mathcal{Q}}(z) + \frac{1}{2t}\|z - x\|^2 = \frac{1}{2t}\mathrm{dist}(x, \mathcal{Q})^2.$$

The (squared) distance function $\mathrm{dist}(x, \mathcal{Q})^2$ is continuously differentiable almost everywhere.

BinaryRelax

- Use $\frac{1}{2t}\mathrm{dist}(x,\mathcal{Q})^2$ as the approximant of $\chi_{\mathcal{Q}}(z)$ and minimize

$$\min_{x\in\mathbb{R}^n} f(x) + \frac{\lambda}{2}\mathrm{dist}(x,\mathcal{Q})^2, \tag{3}$$

  where $\lambda = t^{-1} > 0$ is the regularization parameter. When $\lambda \to \infty$, $\frac{\lambda}{2}\mathrm{dist}(x,\mathcal{Q})^2$ converges pointwise to $\chi_{\mathcal{Q}}(x)$.
- Solve (3): hybrid gradient descent + proximal mapping

$$\begin{cases} y^{k+1} & = y^k - \gamma_k \nabla f_k(x^k) \\ x^{k+1} & = \arg\min_{x\in\mathbb{R}^n} \frac{1}{2}\|x - y^{k+1}\|^2 + \frac{\lambda}{2}\mathrm{dist}(x,\mathcal{Q})^2 \\ & = \frac{\lambda\,\mathrm{proj}_{\mathcal{Q}}(y^{k+1}) + y^{k+1}}{\lambda+1}. \end{cases}$$

  $\{y^k\}$: auxiliary float weights; $\{x^k\}$: **nearly** quantized weights.
- Relaxation helps skip bad local minima in $\mathcal{Q}$.

BinaryRelax (cont')

BinaryRelax is a two-phase algorithm

- Phase I with continuation on λ:

$$\begin{cases} y^{k+1} = y^k - \gamma_k \nabla f_k(x^k) \\ x^{k+1} = \frac{\lambda_k \operatorname{proj}_{\mathcal{Q}}(y^{k+1}) + y^{k+1}}{\lambda_k + 1} \\ \lambda_{k+1} = \rho \cdot \lambda_k, \text{ for } \rho \gtrapprox 1 \end{cases}$$

- Phase II with exact quantization (equivalent to BinaryConnect [1]):

$$\begin{cases} y^{k+1} = y^k - \gamma_k \nabla f_k(x^k) \\ x^{k+1} = \operatorname{proj}_{\mathcal{Q}}(y^{k+1}). \end{cases}$$

[1] [Courbariaux, Bengio, and David, 2015]

Remarks

- BinaryRelax resembles the linearized Bregman algorithm [2] for solving the basis pursuit problem

$$\begin{cases} v^{k+1} = v^k - A^\top(Au^k - b) \\ u^{k+1} = \delta \cdot \mathrm{shrink}(v^{k+1}, \mu) \end{cases}$$

- The similar idea of relaxing the discrete sparsity constraint $\|x\|_0 \leq s$ into a continuous and possibly non-convex regularizer such as ℓ_1 norm, has led to great success in the contexts of statistics and compressed sensing.

[2] [Yin, Osher, Goldfarb, and Darbon, 2010]

Experimental Results: CIFAR

We do **layer-wise** quantization. The two baselines are BinaryConnect combined with exact binarization scheme (BWN) [3] and heuristic ternarization scheme (TWN) [4], resp..

CIFAR-10	Float	Binary		Ternary	
		BWN	Ours	TWN	Ours
VGG-11	91.93	88.70	89.28	90.48	91.01
VGG-16	93.59	91.60	91.98	92.75	93.20
ResNet-20	92.68	87.44	87.82	88.65	90.07
ResNet-32	93.40	89.49	90.65	90.94	92.04
ResNet-18 [5]	95.49	92.72	94.19	93.55	94.98
ResNet-34 [5]	95.70	93.25	94.66	94.05	95.07

Table 1: CIFAR-10 validation accuracies.

[3] [Rastegari et al., 2016]
[4] [Li et al., 2016]
[5] Originally for ImageNet classification.

CIFAR-100	Float	Binary		Ternary	
		BWN	Ours	TWN	Ours
VGG-11	70.43	62.35	63.82	64.16	65.87
VGG-16	73.55	69.03	70.14	71.41	72.10
ResNet-56	70.86	66.73	67.65	68.26	69.83
ResNet-110	73.21	68.67	69.85	68.95	72.32
ResNet-18	76.32	72.31	74.04	73.15	75.24
ResNet-34	77.23	72.92	75.62	74.43	76.16

Table 2: CIFAR-100 validation accuracies.

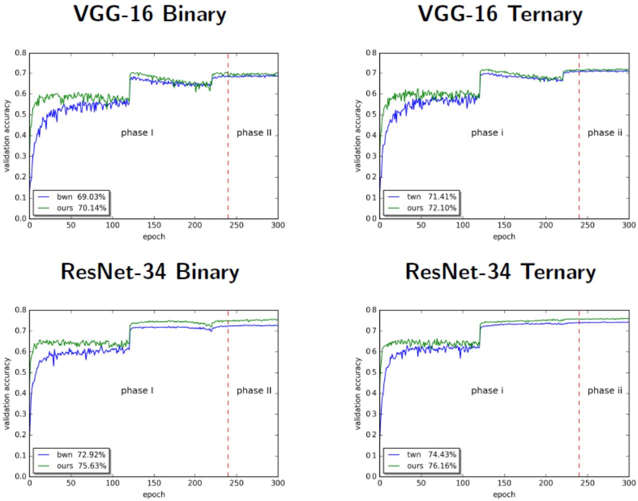


Figure 1: Comparisons of validation accuracy curves for CIFAR-100 using VGG-16 and ResNet-34.

Experimental Results: ImageNet

BinaryRelax recovers the full-precision (32-bit) accuracy using 4-bit weights on ImageNet classification.

ImageNet	Bit-width	Top-1	Top-5
ResNet-18	32	69.6	89.0
	4	**69.7**	**89.4**
ResNet-34	32	73.3	91.4
	4	**73.4**	**91.4**
ResNet-50	32	**76.0**	**92.9**
	4	75.3	92.8

Table 3: ImageNet validation accuracies.

References

- M. Courbariaux, Y. Bengio, and J. David, *Binaryconnect: Training deep neural networks with binary weights during propagations*, NIPS, 2015.
- W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing*, SIAM J. Imaging Sci., 2010.
- M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, *Xnor-net: Imagenet classification using binary convolutional neural networks*, ECCV, 2016.
- F. Li, B. Zhang, and B. Liu, Ternary weight networks, 2016.
- P. Yin, S. Zhang, J. Lyu, S. Osher, Y. Qi, J. Xin, *BinaryRelax: A Relaxation Approach For Training Deep Neural Networks With Quantized Weights*, UCLA CAM report 18-05, 2018.

Phase Retrieval

Primal Dual Hybrid Algorithm with Dual Smoothing

Joint with: M. Pham and P. Yin

February 8, 2018

Problem: given Fourier Coefficient Magnitudes (Different Pattern) $M = |z| = |Fu|$ where F is the Fourier Transform matrix, find the image u
This inverse problem can be formulated as a non-convex optimization with a splitting form

$$\begin{aligned} \min_{u,z} \quad & f(u) + g(z) \\ s.t. \quad & z = Fu, \end{aligned}$$

where f and g are indicator functions

1. $f(u) = \mathcal{I}_{\mathcal{X}}(u)$ is the indicator function of non-negative constraint where: $\mathcal{X} = \{u \geq 0 \text{ in } \Omega, u = 0 \text{ in } \Omega \backslash \mathcal{D}\}$ and Ω and $\mathcal{D}$ are the domain and the support respectively
2. $g(z) = \mathcal{I}_{|z|=M}(z)$ is the indicator of Fourier Magnitude measurements

Note that $\{|z| = M\}$ is non-convex set, hence $g(u)$ is also **non-convex**.
If there is noise in the Fourier Measurement, how do we modify the model?

Phase Retrieval

Example: Vesicle model

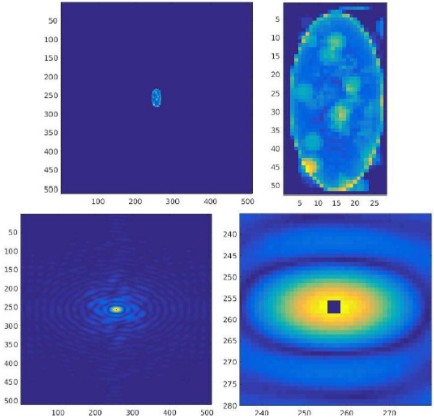


Figure: Top: image. Bottom Fourier coefficients in magnitude. Domain Ω is the big square and support $\mathcal{D}$ is the rectangle around the vesicle

Methods to solve Phase Retrieval

- Alternating Projection Method, also called Error Reduction Algorithm (ER): Alternatively project on the physical and Fourier constraint
- Hybrid Input & Output algorithm HIO: Douglas Rachford with relaxation
- Oversample Smoothness OSS: HIO + smoothing u outside the support

Problem: OSS is not formulated correctly, need fixing
Also, find a denoising model
Since Fourier Measurements contain noise, we replace g by g_σ

$$g_\sigma(z) = \begin{cases} \frac{1}{2\sigma}\||z| - M\|^2 & \text{for Gaussian and small Poisson noise} \\ \frac{1}{\sigma}\sum_i (|z_i| - M_i)\log(|z_i|) & \text{for Poisson noise} \end{cases}$$

Note that, as $\sigma \to 0$, $g_\sigma \to g(z)$

Generalized Infimal Convolution: G-smoothing

Experiments show that the non-negative constraint does not work well. Then, it is replaced by HIO, a relaxed version of Douglas Rachford, which gives better result. However, this is not enough if there is significant noise in physical space

Our approach: replace f by its infimal convolution with a quadratic $q_\gamma(v) = \frac{1}{2\gamma}\|v\|^2$.

Define:

$$f_\gamma(u) = \min_{v+w=u}\left\{f(v) + q_\gamma(w)\right\} = \min_v\left\{f(v) + \frac{1}{2\gamma}\|v-u\|^2\right\}$$

We can use a generalized infimal convolution with a gerneralized quadratic $q_G(v) = \frac{1}{2}v^T G^{-1} v$ where G is a (symmetric) positive definite matrix.

$$f_G(u) = \min_{v+w=u}\left\{f(v) + q_G(w))\right\} = \min_{v+w=u}\left\{f(v) + \frac{1}{2}w^T G^{-1} w)\right\}$$

Define the dual function

$$f^*(y) = \sup_u\left\{u^T y - f(u)\right\}$$

then the infimal convolution gives the separation in dual space.

$$f_G^*(y) = f^*(y) + q_G^*(y) = f^*(y) + \frac{1}{2}y^T G y$$

Primal Dual Algorithm

Now, go back to our problem where f_G and g_σ replace f and g respectively

$$\begin{aligned} \min_{u,z} \quad & f_G(u) + g_\sigma(z) \\ s.t. \quad & z = Fu \end{aligned}$$

Since f is convex, we can reformulate the problem using a primal-dual form

$$\min_z \max_y \quad g_\sigma(z) - f_G^*(y) + z^T F y,$$

Note that $f_G^*(y) = f^*(y) + \frac{1}{2} y^T G y$.

$$\text{Recall:} \quad \operatorname*{prox}_{tf}(x) = \operatorname*{argmin}_y \left\{ f(y) + \frac{1}{2t} \|y - x\|^2 \right\}$$

Solve the problem by Champolle-Pock (or Primal Dual Hybrid Gradient) algorithm

$$\begin{aligned} z^{k+1} &= \operatorname*{prox}_{tg} \left\{ z^k - tFy^k \right\} \\ y^{k+1} &= \operatorname*{prox}_{sf_G^*} \left\{ y^k + sF^{-1}(2z^{k+1} - z^k) \right\} \end{aligned}$$

Since f_G is convex, one can use Moreau decomposition to solve the proximal in either primal or dual space.
When $s = t = 1$, Primal Dual is equivalent to Douglas Rachford (DR)

Primal Dual Algorithm
continue

We have $f(u) = \mathcal{I}(z \geq 0 \text{ in } \Omega, z = 0 \text{ in } \Omega \backslash \mathcal{D})$, the dual function of f is given by

$$f^*(y) = \mathcal{I}(\mathcal{R}e(y) \leq 0 \text{ in } \mathcal{D})$$

which is also indicator function of a convex set, Let $\mathcal{X}^*$ be this set

Let $d^{k+1} = y^k + sF^{-1}(2z^{k+1} - z^k)$, we can rewrite the update of y as followed

$$\begin{aligned} y^{k+1} &= \underset{y}{\operatorname{argmin}} \{f^*(y) + \frac{1}{2} y^T G y + \frac{1}{2s} \|y - d^{k+1}\|^2\} \\ &= \underset{y \in \mathcal{X}^*}{\operatorname{argmin}} \{\frac{1}{2} y^T G y + \frac{1}{2s} \|y - d^{k+1}\|^2\} \end{aligned}$$

If $G = \gamma \mathrm{I}$ or $G = \gamma \operatorname{diag}[r^2(x)]$ where $r(x) = |x|$ is the radius, we have a close form solution.

Primal Dual Algorithm

Choices of G

If $G = \gamma \mathrm{I}$, then $f_G(u) = \frac{1}{2\gamma}\|u - \mathcal{P}_{\mathcal{X}}(u)\|^2$: Least square L^2 regularizer

$$\min_z \quad \max_y \quad g_\sigma(z) - f^*(y) - \frac{\gamma}{2}\|y\|^2 + x^T F y$$

If $G = \gamma\, \mathrm{diag}[r^2(x)]$, we have L^2 (weighted) regularizer (note that $\|DFu\|_{L^2(\Omega)} = \|r \cdot u\|_{L^2(\Omega)}$ by Parseval identity)

$$\min_z \quad \max_y \quad g_\sigma(z) - f^*(y) - \frac{\gamma}{2}\|r \cdot y\|^2 + x^T F y$$

If $G = \gamma D^T D$, we have L^2 gradient regularizer

$$\min_z \quad \max_y \quad g_\sigma(z) - f^*(y) - \frac{\gamma}{2}\|Dy\|^2 + x^T F y$$

In this case, we approximate the solution by

$$y^{k+1} \approx \left(I + s\gamma D^T D\right)^{-1} \operatorname*{Proj}_{\mathcal{X}^*} d^{k+1} \approx \mathcal{G} * \operatorname*{Proj}_{\mathcal{X}^*} d^{k+1}$$

i.e. projection followed by smoothing. $\mathcal{G}$ is Gaussian kernel and $*$ is the convolution
Recall $\mathcal{X}^* = \{y : \mathcal{R}e(y) \leq 0 \text{ in } \mathcal{D}\}$

Meaning of dual variable y

1. With original function f: y is the subgradient of f

$$y \in \partial f(u)$$

When $s = t = 1$, we interpret y as the orthogonal component of projection of u on $\mathcal{X}$

$$y^k = u^k - \operatorname*{Proj}_{\mathcal{X}} u^k$$

i.e. y lies in the orthogonal suspace (or dual space) of $\mathcal{X}$

2. With infimal convolution f_G: y is the smoothing of gradient of f

$$y \in \partial f_G(u)$$

i.e. we smooth the gradient. This technique is very helpful in non-smooth optimization

Comparison HIO v.s. Primal Dual Hybrid

1. OSS algorithm

$$z^{k+1} = M \cdot \exp\left(i \arg(Fu)\right)$$
$$u^{k+\frac{1}{2}} = u^k - \beta F^{-1} z^{k+1} \text{ for HIO}, \left(u^{k+\frac{1}{2}} = \operatorname*{Proj}_{\mathcal{X}}\left(F^{-1} z^{k+1}\right) \text{ for ER}\right)$$
$$u^{k+1} = \mathcal{G} * u^{k+\frac{1}{2}}$$

   Cons: Forget magnitude of Fu, only take its phase to compute Fourier space. The smoothing can be only applied to the outside of support. Mathematically incorrect algorithm

2. Primal Dual Hybrid with smoothing on real space (or Fourier space)

$$z^{k+\frac{1}{2}} = z^k - tFy^k$$
$$z^{k+1} = \frac{M \cdot \exp\left(i \arg z^{k+\frac{1}{2}}\right) + \sigma z^{k+\frac{1}{2}}}{1+\sigma}$$
$$u^{k+1} = F^{-1}(2z^{k+1} - z^k)$$
$$y^{k+\frac{1}{2}} = \operatorname*{Proj}_{\mathcal{X}}\left(y^k - su^{k+1}\right)$$
$$y^{k+1} = \mathcal{G} \cdot y^{k+\frac{1}{2}}, \quad \left(\text{or } y^{k+1} = \mathcal{G} * y^{k+\frac{1}{2}}\right)$$

Experiments

Tuning algorithm variables

- step size t, s: default $t = s = 1$
- quadratic penalty parameter σ: depends on noise level , $\sigma \in (0, \infty)$
- smoothing type: method DR1: $G = \gamma D^T D$, method DR2: $G = \gamma\,\mathrm{diag}[r^2]$, and method DR3: combining both
- smoothing parameter γ: as in OSS: increasing $\{\gamma_k\}_{k=1}^{10}$

Experiments parameters

- models: Vesicle, Nanorice, Yeast Spore
- noise: Gaussian and Poisson noise
- flux: in range $(10^5, 10^9)$ (which causes Poisson noise)

Measurements

- Ratio factor R (relative error of Fourier Magnitude)
- Fourier Shell Correlation FRC

Vesicle Model

Relative Error, flux = 1e7

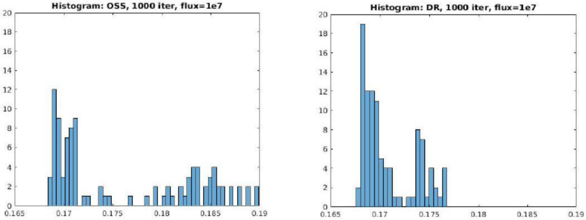


Figure: Relative error of 100 reconstruction using OSS (left), and DR(right)

flux $= 10^7$	OSS	DR $\sigma = 0.1$
min	16.85%	16.80%
max	19%	17.67%
mean	17.62%	17.08%
good minimums ($< 17.1\%$)	52%	71%
$< 16.85\%$	0%	**21**%

i.e. instead of computing 100 OSS reconstructions, we only need to do 5 DR reconstructions. Primal Dual saves computational costs by a factor of 21

Vesicle Model

Relative Error, flux=1e8

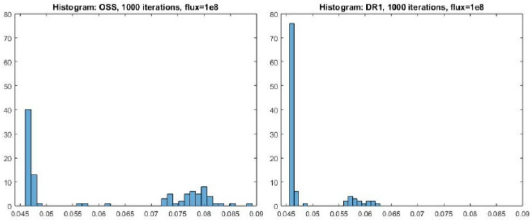


Figure: Relative error of 100 reconstruction using OSS (left), and DR(right)

flux $= 10^8$	OSS	DR $\sigma = 0.1$
min	4.63%	4.62%
max	8.90%	6.26%
mean	6.08%	4.84%
good minimums ($< 5.0\%$)	54%	83%
$< 4.63\%$	0%	**70**%

Primal Dual converges to global minimum with probability 29% higher than OSS. 70% reconstruction of DR gives better result than the best of OSS. Primal Dual save computational costs by a factor of 70.

Vesicle Model

Convergence

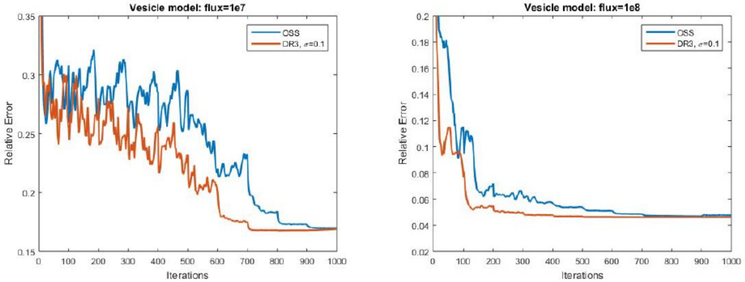
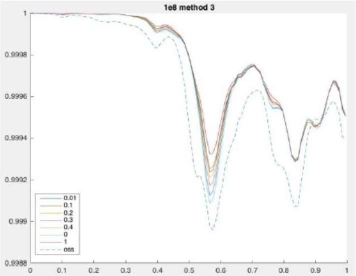


Figure: Convergence of OSS and Primal Dual DR3 (combine 2 smoothing). Left: flux $= 10^7$. Right: flux $= 10^8$. DR converges faster and converges to a deeper minimum than OSS

Vesicle Model
Fourier Shell Correlation



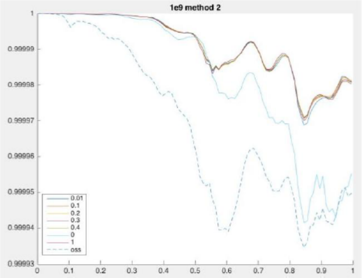


Figure: Fourier Shell Correlation of OSS and DR with different weight σ. Left: flux $= 10^8$. Right: flux $= 10^9$

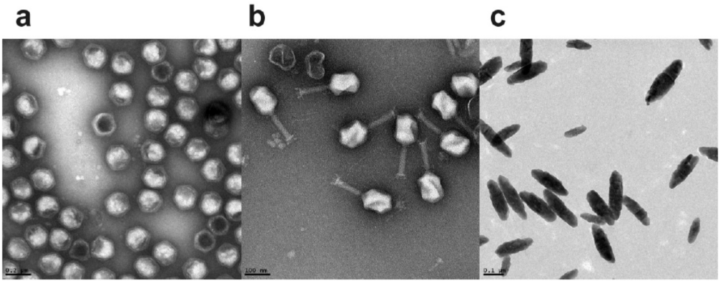


Figure: TEM images of (a) PBCV-1, (b) baceriophase T4, and (c): nanorice (250 x 50 nm)

Nanorice1

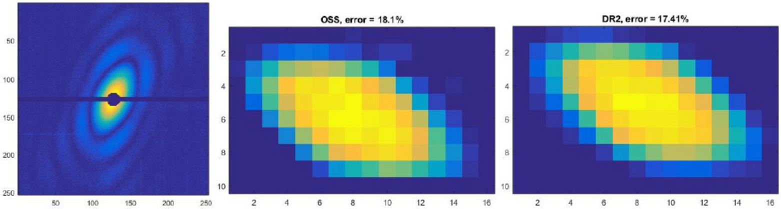
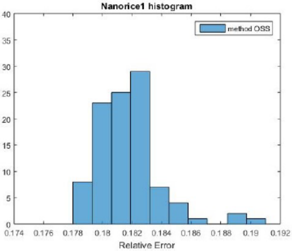
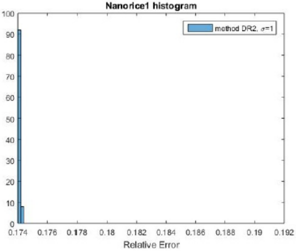


Figure: Nanorice1 different pattern (left), and reconstruction using OSS, error = 18.1%(middle), and Primal Dual, error = 17.41% (right)

Nanorice1

Convergence





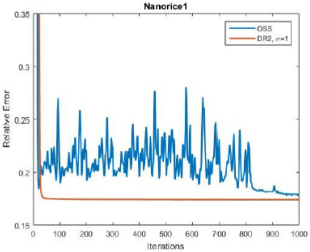


Figure: Top: Relative Error histogram of OSS(left) and DR(right). Bottom: convergence of OSS and DR.

Primal dual is faster, more stable, and converges to a deeper minimum than OSS consistently

Nanorice2

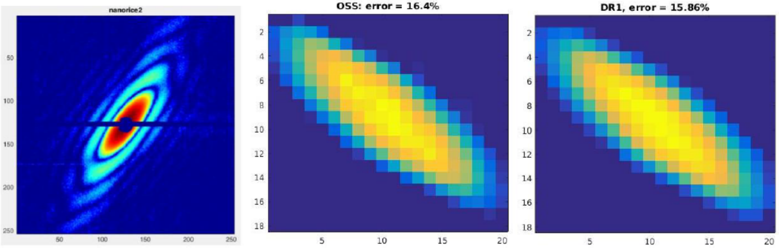
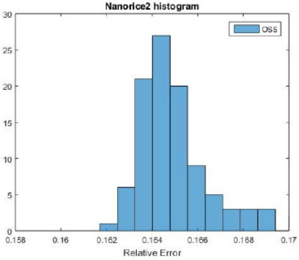
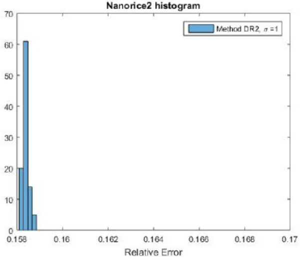


Figure: Nanorice2 different pattern (left), and reconstruction using OSS, error = 16.4%(middle), and Primal Dual, error = 15.86% (right)

Nanorice2

Convergence





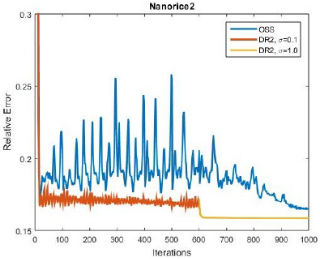
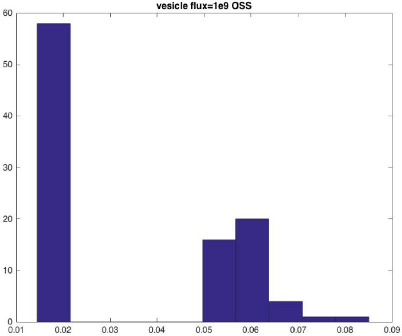
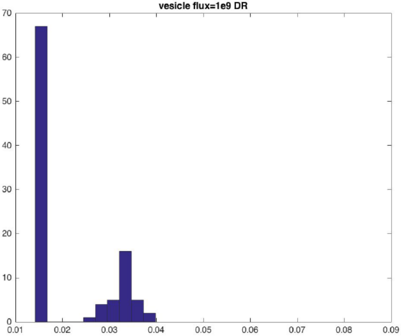
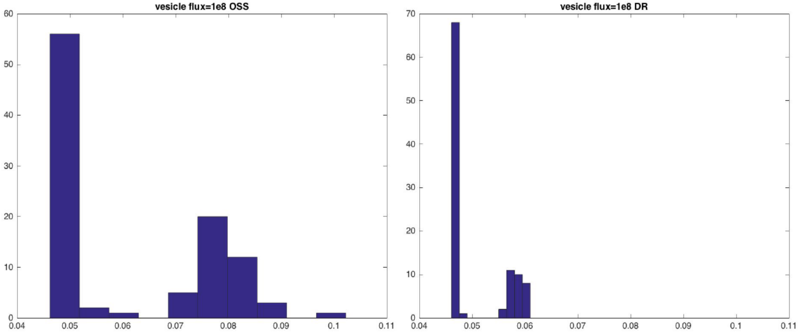


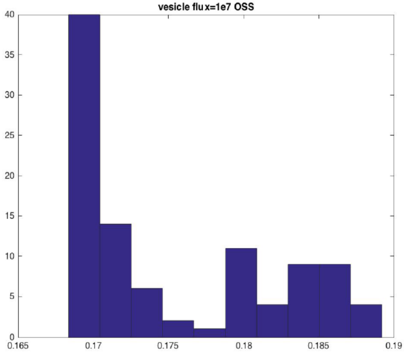
Figure: Top: Relative Error histogram of OSS(left) and DR(right). Bottom: convergence of OSS and DR2. For DR2, $\sigma = 0.1$ where iteration < 600 and $\sigma = 1$ otherwise

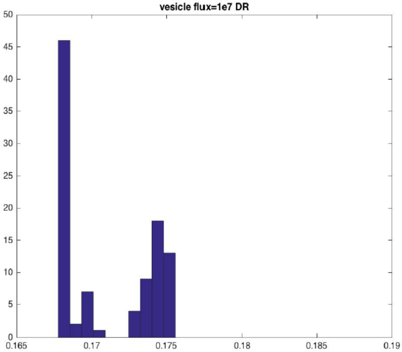
Mean and STD for 100 independent reconstructions each

Vesicle flux level	OSS	DR(best weight and smoothing method)
1e9	0.034 +/- 0.022	0.020 +/- 0.009
1e8	0.061 +/- 0.016	0.050 +/- 0.006
1e7	0.175 +/- 0.007	0.171 +/- 0.003

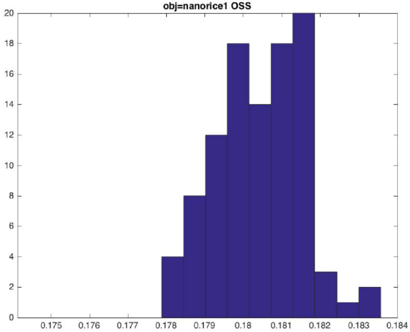

vesicle flux=1e9 OSS
0
10
20
30
40
50
60
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09

vesicle flux=1e9 DR
0
10
20
30
40
50
60
70
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09

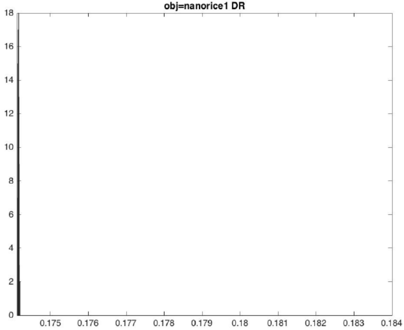

vesicle flux=1e8 OSS
vesicle flux=1e8 DR
0.04
0.05
0.06
0.07
0.08
0.09
0.1
0.11

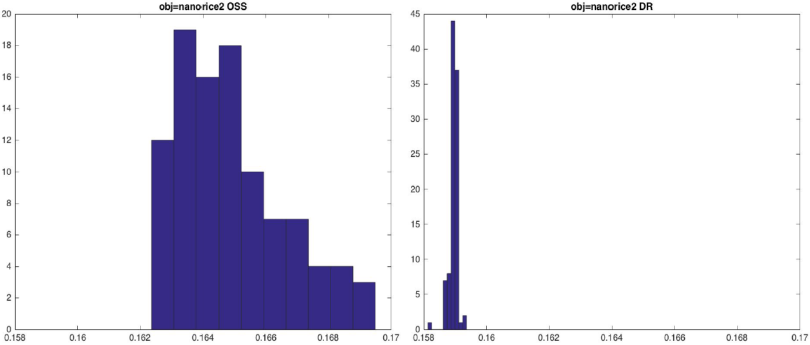

vesicle flux=1e7 OSS
0 5 10 15 20 25 30 35 40
0.165 0.17 0.175 0.18 0.185 0.19


vesicle flux=1e7 DR
0 5 10 15 20 25 30 35 40 45 50
0.165 0.17 0.175 0.18 0.185 0.19

	OSS	DR
nanorice1	0.180 +/- 0.001	0.174 +/- 0.000
nanorice2	0.165 +/- 0.002	0.159 +/- 0.000


obj=nanorice1 OSS


obj=nanorice1 DR


obj=nanorice2 OSS
obj=nanorice2 DR
20
18
16
14
12
10
8
6
4
2
0
0.158
0.16
0.162
0.164
0.166
0.168
0.17
45
40
35
30
25
20
15
10
5
0
0.158
0.16
0.162
0.164
0.166
0.168
0.17

Thank you!