Stochastic Optimisation using CIL

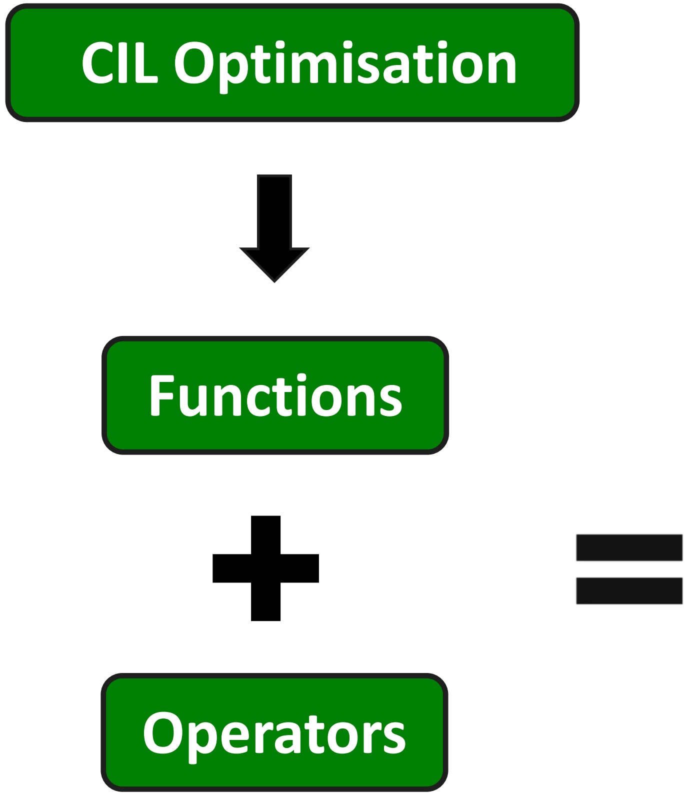
CIL User Meeting

November 2022

Evangelos Papoutsellis – Finden Ltd

epapoutsellis@gmail.com

Overview: Optimisation in CIL



Algorithms

name	description
CGLS	conjugate gradient least squares
SIRT	simultaneous iterative reconstruction technique
GD	gradient descent
FISTA	fast iterative shrinkage-thresholding algorithm
LADMM	linearized alternating direction method of multipliers
PDHG	primal dual hybrid gradient
SPDHG	stochastic primal dual hybrid gradient

Overview: Optimisation in CIL

$$\min_x f(x)\ , \quad f : \text{L-smooth}$$

$$\min_x f(x) + g(x)\ , \quad f : \text{L-smooth}\ ,\ g : \text{proximable}$$

$$\min_x f(Kx) + g(x)\ , \quad f : \text{proximable}\ ,\ g : \text{proximable}\ ,\ K \text{ linear operator}$$

f, g convex

TV Tomography reconstruction with non-negativity constraint

$$\min_u \frac{1}{2}\|Au - d\|^2 + \|\nabla u\|_{2,1} + \mathbb{I}_{\{u>0\}}(u)$$

TGV denoising
Salt and Pepper Noise

$$\min_{u,w} \frac{1}{2}\|u - d\|_1 + \alpha\|\nabla u - w\|_{2,1} + \beta\|\mathbb{E}w\|_{2,1}$$

Stochastic Project

Extend CIL Optimisation (Deterministic) framework to Stochastic Optimisation

- **1st Hackathon:** November 23-26, 2021
- **2nd Hackathon**: April 4-7, 2022
- **Organised**: CCP SyneRBI , CCPi , PET++

Joint work: Kris Thielemans, Gillman Ashley, Tang Junqi, Zeljko Kereta, Imraj Singh, Gemma Fardell, Evgueni Ovtchinnikov, Matthias Ehrhardt, Laura Murgatroyd, Robert Twyman, Edoardo Pasca, Claire Delplancke, Georg Schramm, Jakob Jørgensen, Sam Porter, Margaret Duff, Antony Vamvakeros, Simon Jacques

- **Implement splitting for SIRF DataContainers: PET, SPECT, MRI data**
- **Implement randomized algorithms in CIL, e.g., SGD, SAG, SAGA, SVRG and more**
- **Benchmarking framework for CT and PET applications**

Stochastic Optimisation in CIL

$$\min_x f(x) + g(x)$$

GD	PGA/ISTA	APGA/FISTA
$x_{k+1} = x_k - \gamma_k \nabla f(x_k)$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k))$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(y_k - \gamma_k \nabla f(y_k))$ $\alpha_{k+1} = \dfrac{1 + \sqrt{1 + 4\alpha_k^2}}{2}$ $y_k = x_k + \dfrac{\alpha_k - 1}{a_{k+1}}(x_k - x_{k-1})$

Stochastic Optimisation in CIL

$$\min_{x} \sum_{i=1}^{n} f_i(x) + g(x)$$

- *Avoid computing the full gradient per iteration, i.e., gradient for all n.*

- *Select a random index* $i_k \in \{1, \dots, n\}$ *and compute* ∇f_{i_k} *per iteration.*

GD	PGA/ISTA	APGA/FISTA
$x_{k+1} = x_k - \gamma_k \nabla f_{i_k}(x_k)$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f_{i_k}(x_k))$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(y_k - \gamma_k \nabla f_{i_k}(y_k))$ $\alpha_{k+1} = \frac{1 + \sqrt{1 + 4\alpha_k^2}}{2}$ $y_k = x_k + \frac{\alpha_k - 1}{a_{k+1}}(x_k - x_{k-1})$

Stochastic Optimisation in CIL

GD	PGA/ISTA	APGA/FISTA
$x_{k+1} = x_k - \gamma_k \nabla f_{i_k}(x_k)$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f_{i_k}(x_k))$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(y_k - \gamma_k \nabla f_{i_k}(y_k))$ $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$ $y_k = x_k + \frac{\alpha_k - 1}{a_{k+1}}(x_k - x_{k-1})$
SGD	*Prox-SGD*	*Acc-Prox-SGD*

Stochastic Optimisation Design

- **No direct implementation of Stochastic algorithms, e.g., SGD.**
- **Use a deterministic algorithm, e.g., GD, already available in CIL.**
- **Implement a Stochastic Gradient "Functions" or Variance-Reduced "Functions"**
- **Example SGD := GD (CIL Algorithm) + SGFunction (CIL Function)**

Stochastic Optimisation in CIL

SGD	SPGA/SISTA	SAPGA/SFISTA
$x_{k+1} = x_k - \gamma_k \tilde{\nabla} f_{i_k}(x_k)$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \tilde{\nabla} f_{i_k}(x_k))$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(y_k - \gamma_k \tilde{\nabla} f_{i_k}(y_k))$ $\alpha_{k+1} = \frac{1+\sqrt{1+\alpha_k^2}}{2}$ $y_{k+1} = x_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(x_k - x_{k-1})$

`ApproximateGradientSumFunction`

$$\sum_{i=1}^{n} f_i(x)$$

Stochastic Gradient and Variance-Reduced CIL "Functions"

`SGFunction` `SAGFunction` `SAGAFunction`

`SVRGFunction` `LSVRGFunction`

`and more ...`

Stochastic Optimisation in CIL

Plug and Play Framework - Different Stochastic Gradient Functions

Algorithms / Stochastic Function	GD	ISTA	FISTA
SGFunction	SGD	Prox-SGD	Acc-Prox-SGD
SAGFunction	SAG	Prox-SAG	Acc-Prox-SAG
SAGAFunction	SAGA	Prox-SAGA	Acc-Prox-SAGA
SVRGFunction	SVRG	Prox-SVRG	Acc-Prox-SVRG
LSVRGFunction	LSVRG	Prox-LSVRG	Acc-Prox-LSVRG

```
pgd = ISTA(initial = initial, f = f, g = g,
           update_objective_interval = 1, max_iteration = 10)
pgd.run(1,verbose=1)
```

Stochastic Optimisation in CIL

Plug and Play Framework - Different Stochastic Gradient Functions

Algorithms / Stochastic Function	GD	ISTA	FISTA
SGFunction	SGD	Prox-SGD	Acc-Prox-SGD
SAGFunction	SAG	Prox-SAG	Acc-Prox-SAG
SAGAFunction	SAGA	Prox-SAGA	Acc-Prox-SAGA
SVRGFunction	SVRG	Prox-SVRG	Acc-Prox-SVRG
LSVRGFunction	LSVRG	Prox-LSVRG	Acc-Prox-LSVRG

```
spgd = ISTA(initial = initial, f = SGFunction(fi), g = g,
           update_objective_interval = 1, max_iteration = 10)
spgd.run(1,verbose=1)
```

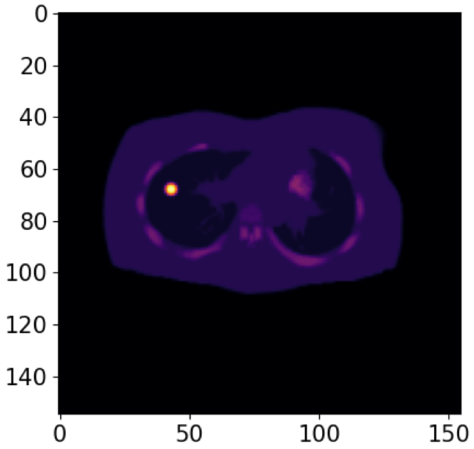
Stochastic Utilities/Improvements

- ✓ **Data splitting methods for AcquisitionData (CIL + SIRF).**
- ✓ **Sampling methods used by Stochastic Functions, i.e., select the next function** f_{i_k}
- ✓ **Callable Classes to improve functionality of CIL Algorithms**
- ✓ **Proximal Gradient Algorithm (PGA) : Base class for Proximal Gradient Algorithms, e.g., GD, ISTA, FISTA**
- ✓ **CIL + SIRF fully compatible --> SIRF Functions can be used with CIL Algorithms**
- ✓ **Benchmark API using Hydra OSS for both CIL and SIRF applications.**

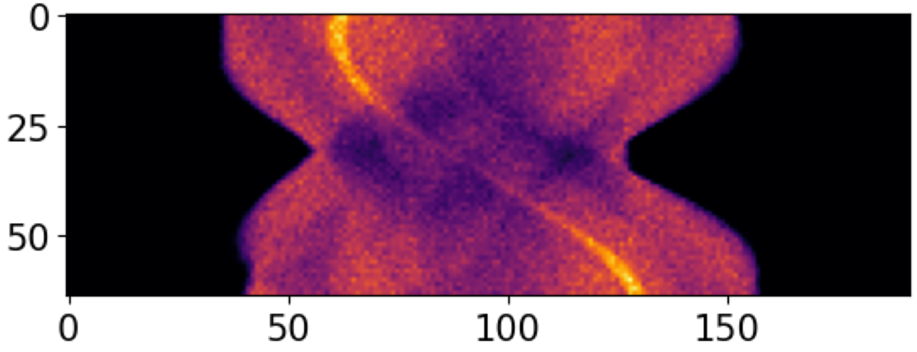
Stochastic Utilities

- ✓ **Example: Data splitting methods for AcquisitionData (CIL + SIRF).**

Simulated
2D Thorax

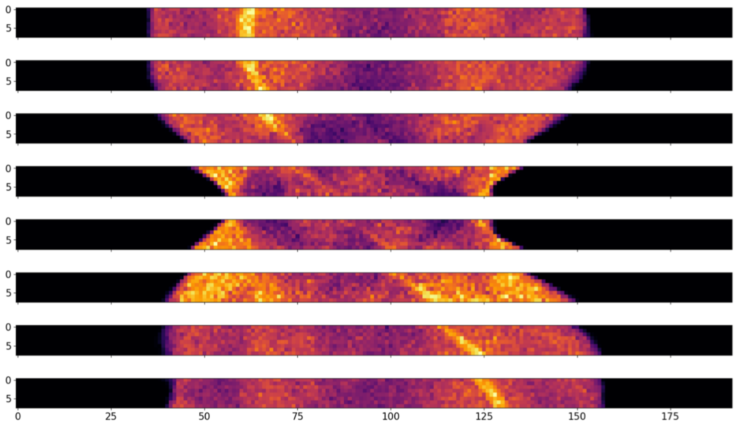


64 projection angles

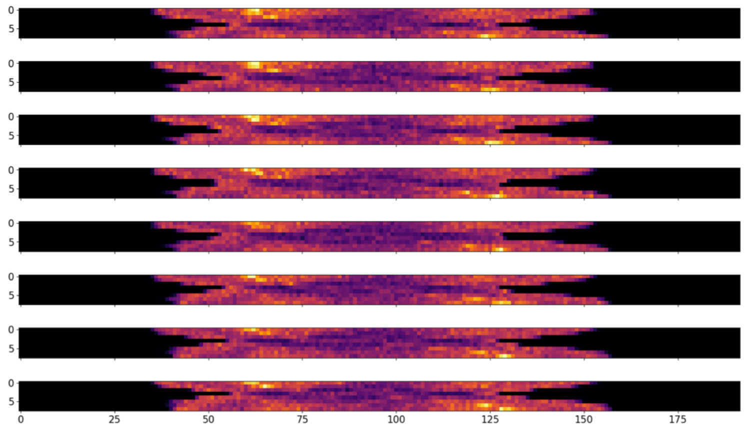


Split to 8 subsets

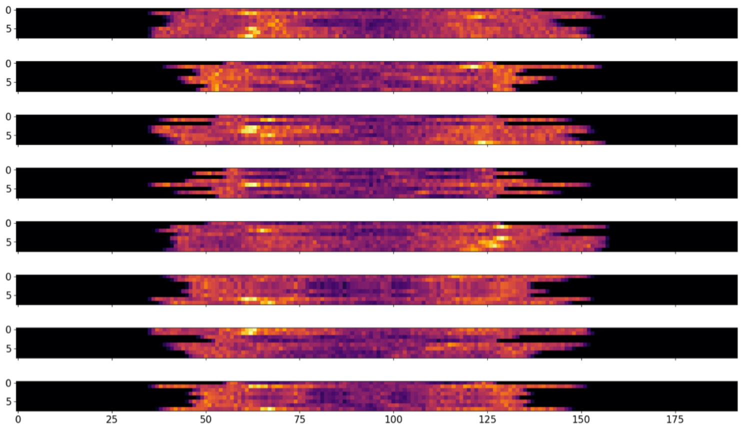
Ordered (Step_size=1)



Ordered (Step_size= NumSubsets)



Random



Stochastic Utilities

- ✓ **Example: Sampling methods used by Stochastic Functions** $\sum_{i=1}^{n} f_i(x)$

Uniform Sampling (With replacement) n=5

Iteration	0	1	2	3	4	5	6	7	8	9
Sample	f_3	f_4	f_0	f_4	f_2	f_2	f_3	f_1	f_4	f_0

RandomShuffle (Without replacement): Random permutation at each epoch

Iteration	0	1	2	3	4	5	6	7	8	9
Sample	f_3	f_1	f_0	f_2	f_4	f_4	f_1	f_2	f_0	f_3
Epoch	0	0	0	0	0	1	1	1	1	1

SingleShuffle (Without replacement): Same permutation at each epoch

Iteration	0	1	2	3	4	5	6	7	8	9
Sample	f_3	f_2	f_4	f_0	f_1	f_3	f_2	f_4	f_0	f_1
Epoch	0	0	0	0	0	1	1	1	1	1

CIL Improvements

✓ **Example: Callable Classes to improve functionality of CIL Algorithms**

```
from cil.optimisation.utilities import MetricsDiagnostics,StatisticsDiagnostics, RSE
from skimage.metrics import structural_similarity as SSIM
from skimage.metrics import normalized_root_mse as NRMSE

cb1 = MetricsDiagnostics(reference_image = fista_1000.solution, verbose=1,
                         metrics_dict={'mae': MAE, 'ssim': SSIM, 'nrmse': NRMSE})
cb2 = StatisticsDiagnostics(verbose=1, statistics_dict={'mean': (lambda x: x.mean())})

fista = FISTA(initial = initial, f = fidelity, g=G, update_objective_interval = 1,
             max_iteration = 5)
fista.run(verbose=1, callback=[cb1, cb2])
```

- *Access to all attributes of the Algorithm*
- *Define custom metrics (images and/or ROI)*
- *Define new stopping criteria*
- *Integrate with Weights and Biases*

Iter	Max Iter	Time(s)/Iter	Objective	mae	ssim	nrmse	mean
0	5	0.000	8.17946e+02	6.51097e-05	5.05983e-01	1.00000e+00	0.00000e+00
1	5	0.542	9.93983e+01	7.07093e-05	2.84027e-01	7.29477e-01	0.00000e+00
2	5	0.391	7.81449e+01	6.34461e-05	3.14256e-01	6.82127e-01	7.00874e-05
3	5	0.329	6.22991e+01	5.61340e-05	3.50318e-01	6.37522e-01	6.89979e-05
4	5	0.297	5.16572e+01	4.95575e-05	3.92073e-01	5.98112e-01	6.79314e-05
5	5	0.330	4.49825e+01	4.41283e-05	4.37438e-01	5.63993e-01	6.69932e-05

Stop criterion has been reached.

CIL Improvements

✓ **Example: Proximal Gradient Algorithm (PGA) Base Class**

$$x_{k+1} = \mathrm{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k))$$

$$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k D(x_k) \nabla f(x_k))$$

```
class Preconditioner(ABC)
class StepSizeMethod(ABC)
```

Nocedal and Wright "Numerical Optimisation"

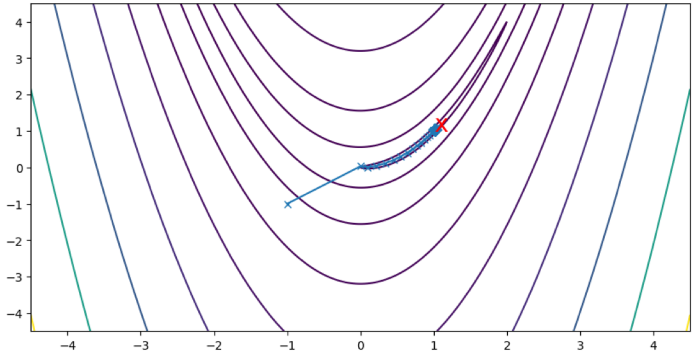
Choose $\bar{\alpha} > 0$, $\rho \in (0, 1)$, $c \in (0, 1)$; Set $\alpha \leftarrow \bar{\alpha}$;
repeat until $f(x_k + \alpha p_k) \leq f(x_k) + c\alpha \nabla f_k^T p_k$
$\alpha \leftarrow \rho\alpha$;
end (repeat)
Terminate with $\alpha_k = \alpha$.

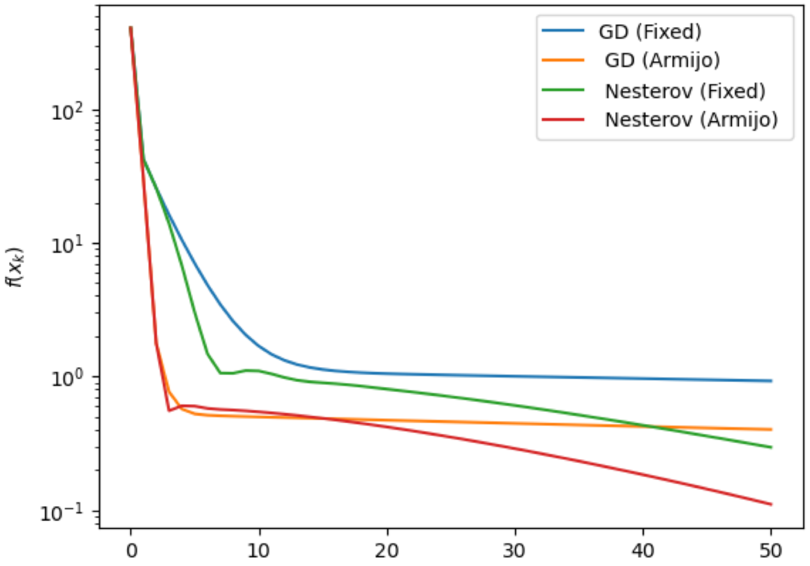
Armijo condition

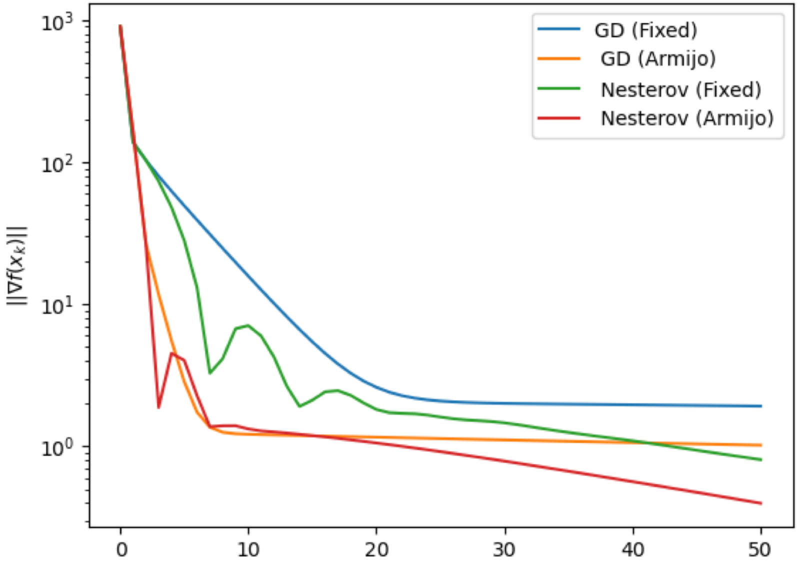
```
armijo = ArmijoStepSize(initial = 1., rho = 0.5, c = 1e-4, iterations=50)
gd = GD(initial = initial, objective_function = f, step_size = armijo,
            max_iteration = 100, update_objective_interval =  1)
gd.run(verbose=1)
```

Rosenbrock Function: minimum at (x,y) = (1,1)

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$







CIL Improvements

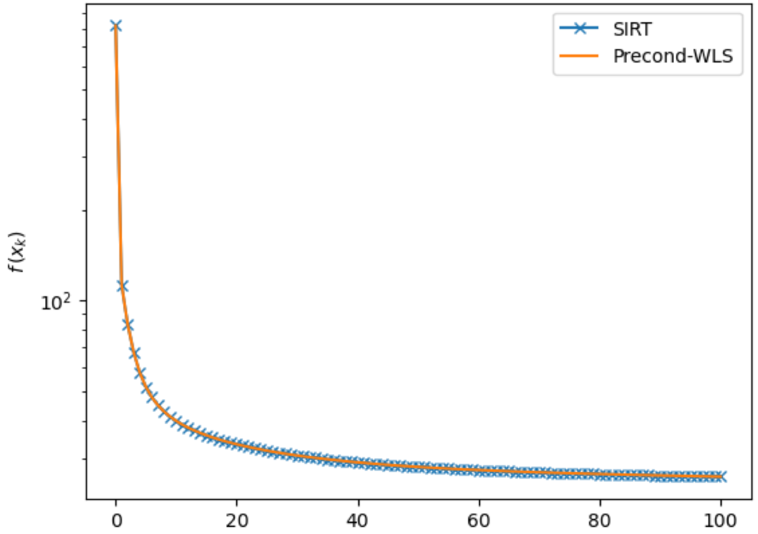
- ✓ **Example: Proximal Gradient Algorithm (PGA) Base Class** $x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k D(x_k)\nabla f(x_k))$

<u>*SIRT Algorithm*</u> $x_{k+1} = P_{>0}(x_k - DA^T W(Ax_k - d)),\quad D = \frac{1}{A^T\mathbf{1}}, W = \frac{1}{A\mathbf{1}}$

```
sirt = SIRT(initial, A, d, lower=0., max_iteration=100, update_objective_interval=1)
```

<u>*SIRT Algorithm*</u> → <u>*Preconditioned Projected Gradient Descent on Weighted Least Squares*</u>

```
W = 1./A.direct(ones)
f = LeastSquares(A=A, b=d, c=0.5, weight=W)
g = IndicatorBox(lower=0.)
D = Sensitivity(A)
precond_wls = ISTA(initial, f=f,  g=g,
                   step_size = 1., preconditioner = D,
                   max_iteration=100, update_objective_interval=1)
```



<u>*Using Stochastic Framework:*</u> → <u>*OS-SIRT, Randomized Kaczmarz*</u>

CIL Improvements

✓ **CIL + SIRF fully compatible --> SIRF Functions can be used with CIL Algorithms**

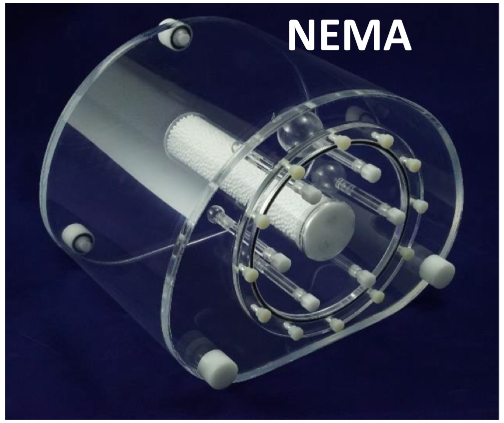
```
class ObjectiveFunction(object)
class Prior(object)
```

$$f(x) = \mathrm{KL}(d, Ax) + \alpha\,\mathrm{RelativeDifference}(x)$$

$$x_{k+1} = P_{>0}(x_k - \gamma_k D(x_k)\nabla f(x_k)) \Rightarrow \quad x_{k+1} = \frac{x_k}{A^T\mathbf{1}} A^T\left(\frac{d}{Ax_k}\right) \quad (MLEM)$$

$$\alpha = 0, \gamma_k = 1$$

$$D(x_k) = \frac{x_k}{A^T\mathbf{1}}$$



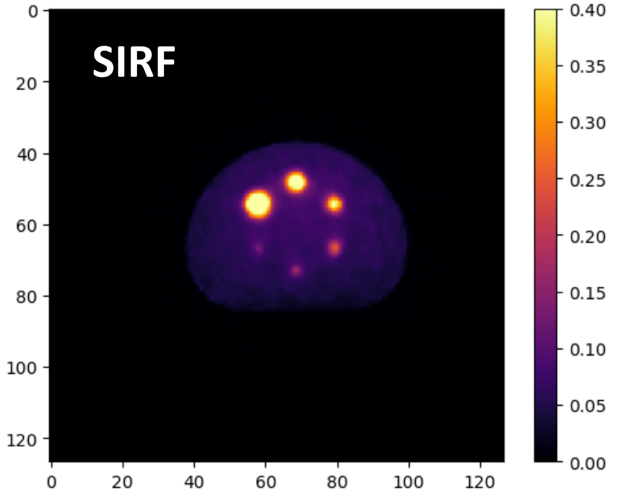
SIRF

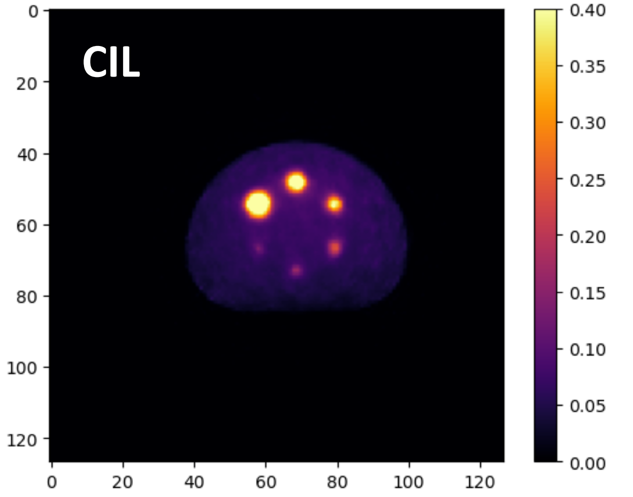
```
objective = make_Poisson_loglikelihood(d)
objective.set_acquisition_model(A)
objective.set_prior(RelativeDifference)

recon = OSMAPOSLReconstructor()
recon.set_objective_function(objective)
recon.set_num_subsets(1)
recon.set_num_subiterations(50)
recon.set_up(initial)
recon.set_current_estimate(initial)
recon.process()
```

CIL

```
D = AdaptiveSensitivity(A)
pgd = ISTA(initial = initial, f = objective, g = IndicatorBox(lower=0.),
           preconditioner = D, step_size = -1.,
           update_objective_interval=1, max_iteration=50)
pgd.run(verbose=1)
```





CIL Improvements

✓ **Benchmark API using Hydra for both CIL and SIRF.**

$$\min_u \sum_{i=1}^{n} f_i(u) + \alpha \|\nabla u\|_{2,1} + \mathbb{I}_{\{u>0\}}(u)$$

$$f_i(u) = \frac{1}{2}\|A_i u - d_i\|^2$$

$$n = 10, 20, 50, 100, 200, 400$$

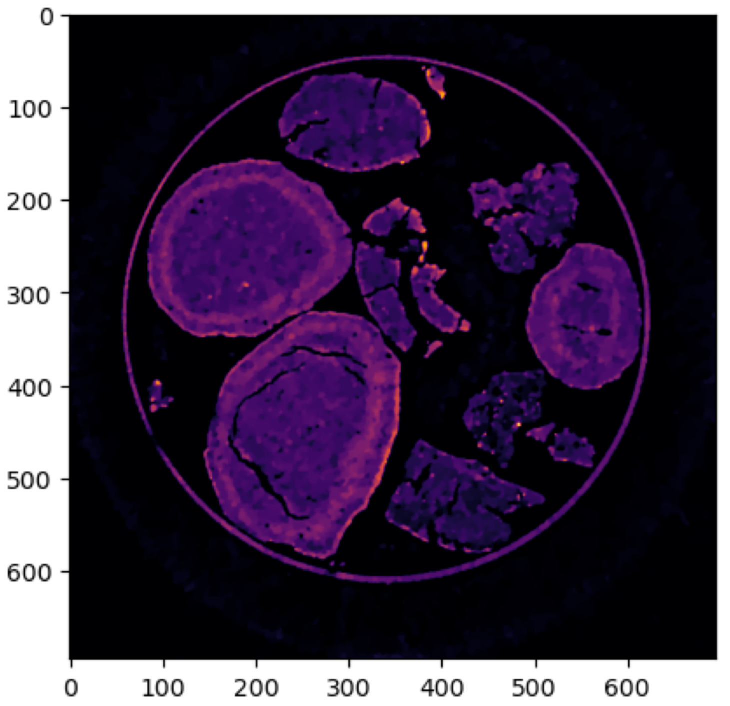
```
python run_stochastic_tv_tomo_recon.py
         --multirun splitting.subsets=10,20,50,100,200,400
       algorithm/sfunction=sgd,saga,svrg,lsvrg  epochs=30
```

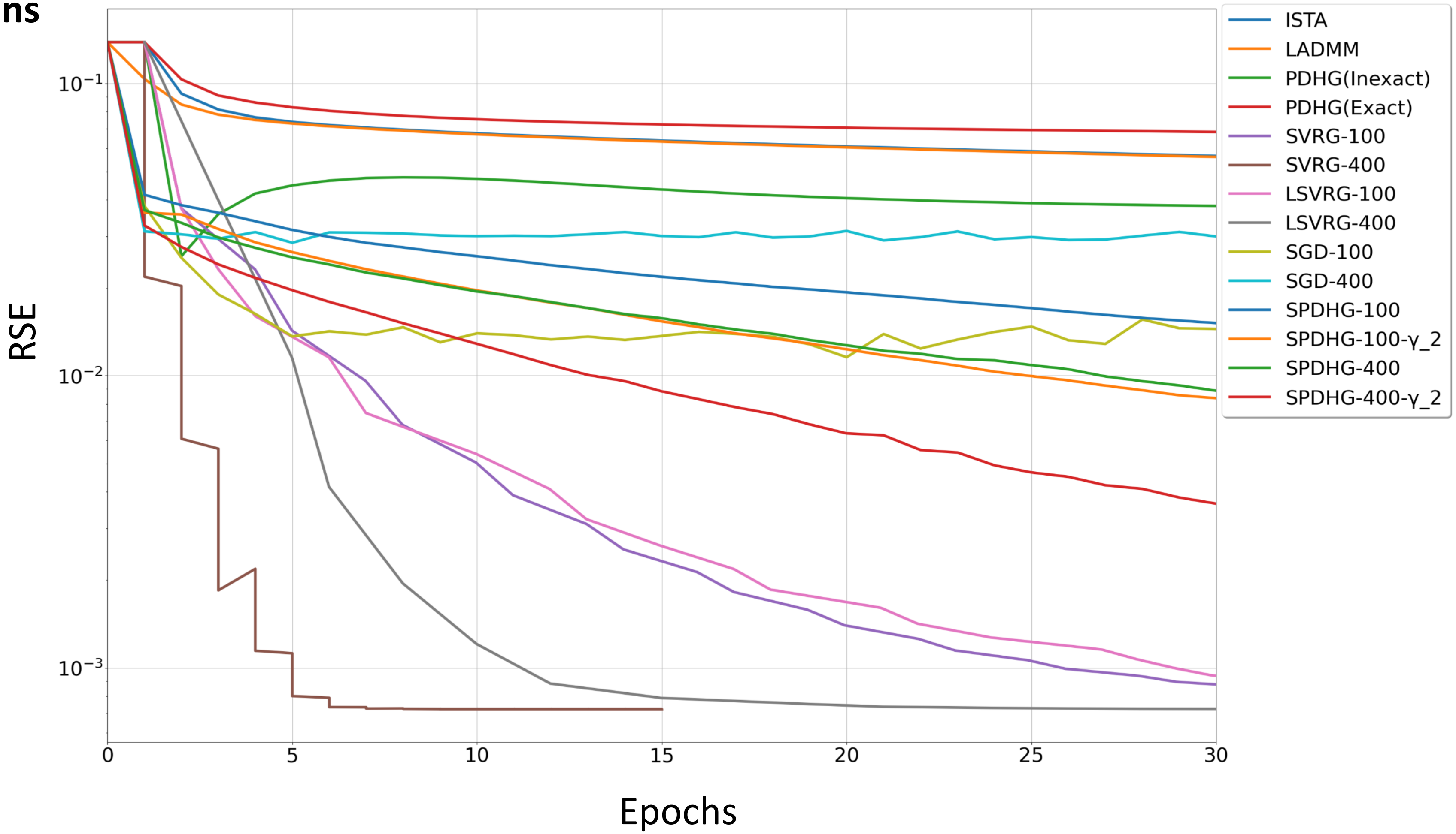
```
[2023-09-27 12:16:14,844][HYDRA] Launching 24 jobs locally
[2023-09-27 12:16:14,844][HYDRA]        #0 : splitting.subsets=10 algorithm/sfunction=sgd epochs=30
[2023-09-27 12:16:15,330][root][INFO] - Load Sinogram and FBP reconstuction
[2023-09-27 12:16:15,460][root][INFO] - Load geometries
[2023-09-27 12:16:15,484][root][INFO] - Number of subsets is 10
[2023-09-27 12:16:15,485][root][INFO] - Splitting method is ordered
[2023-09-27 12:16:15,485][root][WARNING] - Batch size is (constant) self.num_indices//self.num_batches
[2023-09-27 12:16:15,528][root][INFO] - Stochastic function is sgd
[2023-09-27 12:16:15,530][root][WARNING] - Batch size is (constant) self.num_indices//self.num_batches
[2023-09-27 12:16:16,002][root][INFO] - ISTA setting up
[2023-09-27 12:16:16,006][root][INFO] - ISTA configured
```

Applications (CT - CIL)

- **Dataset: 2D (NiPd, MicroCT)**
- **Splitting Method (Data): Ordered**
- **Detector: 634x634, 800 Projections**
- **Sampling Method (Functions): Random with replacement**
- **Optimal Solution: FISTA 5000 iterations**
- **Initial: FBP reconstruction**

$$\min_x \left\{ \frac{1}{2}\|Ax-d\|^2 + \alpha\|\nabla x\|_{2,1} + \mathbb{I}_{\{x>0\}}(x) \right\}$$



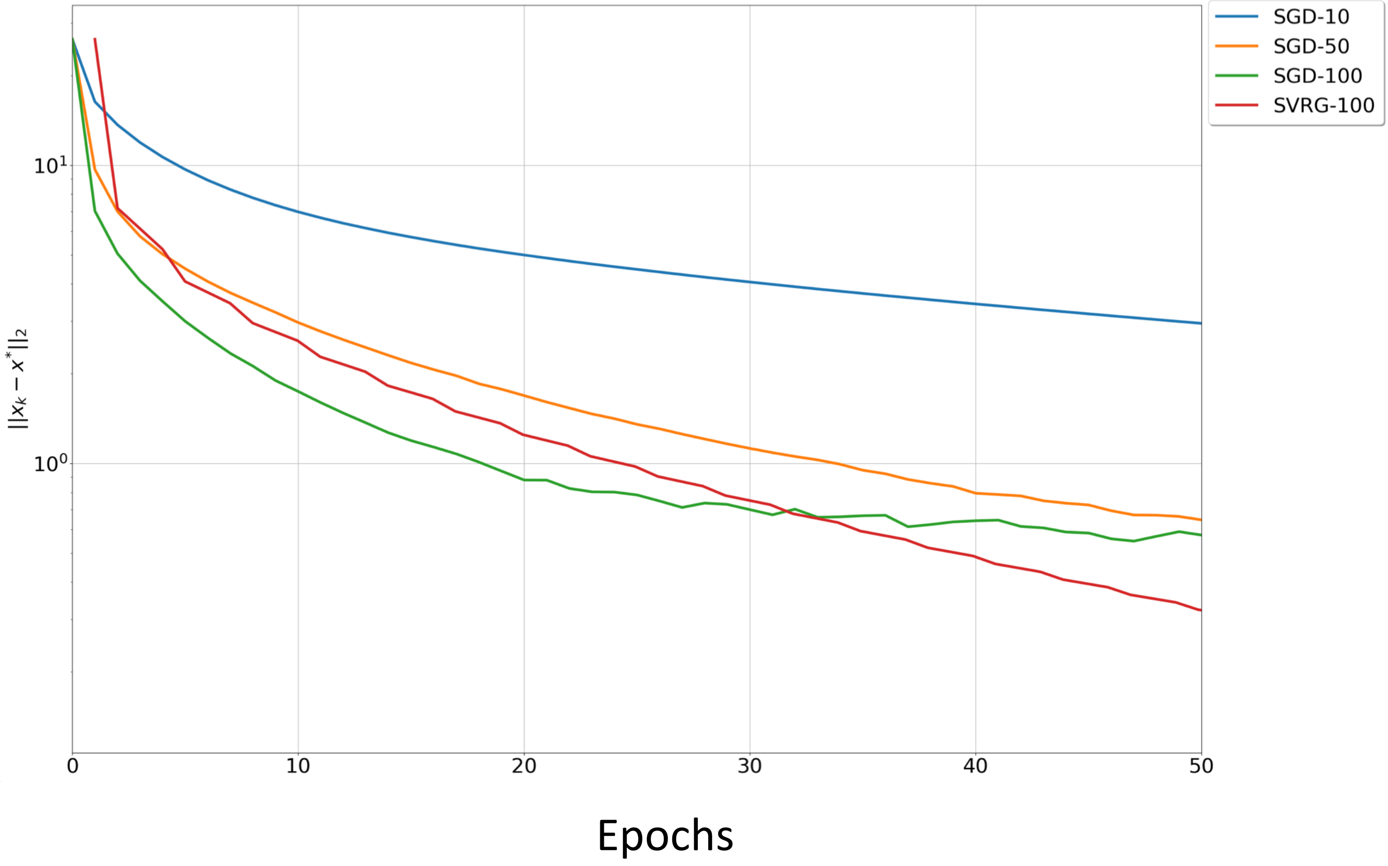


Finden UKRI Innovate UK KTN

Applications (CT - CIL)

- **Dataset: 3D Battery (300 slices, 900 projections)**
- **Detector: 1561x1561**
- **Splitting Method (Data): Ordered**
- **Sampling Method (Functions): Random with replacement**
- **Optimal Solution: FISTA 1000 iterations**

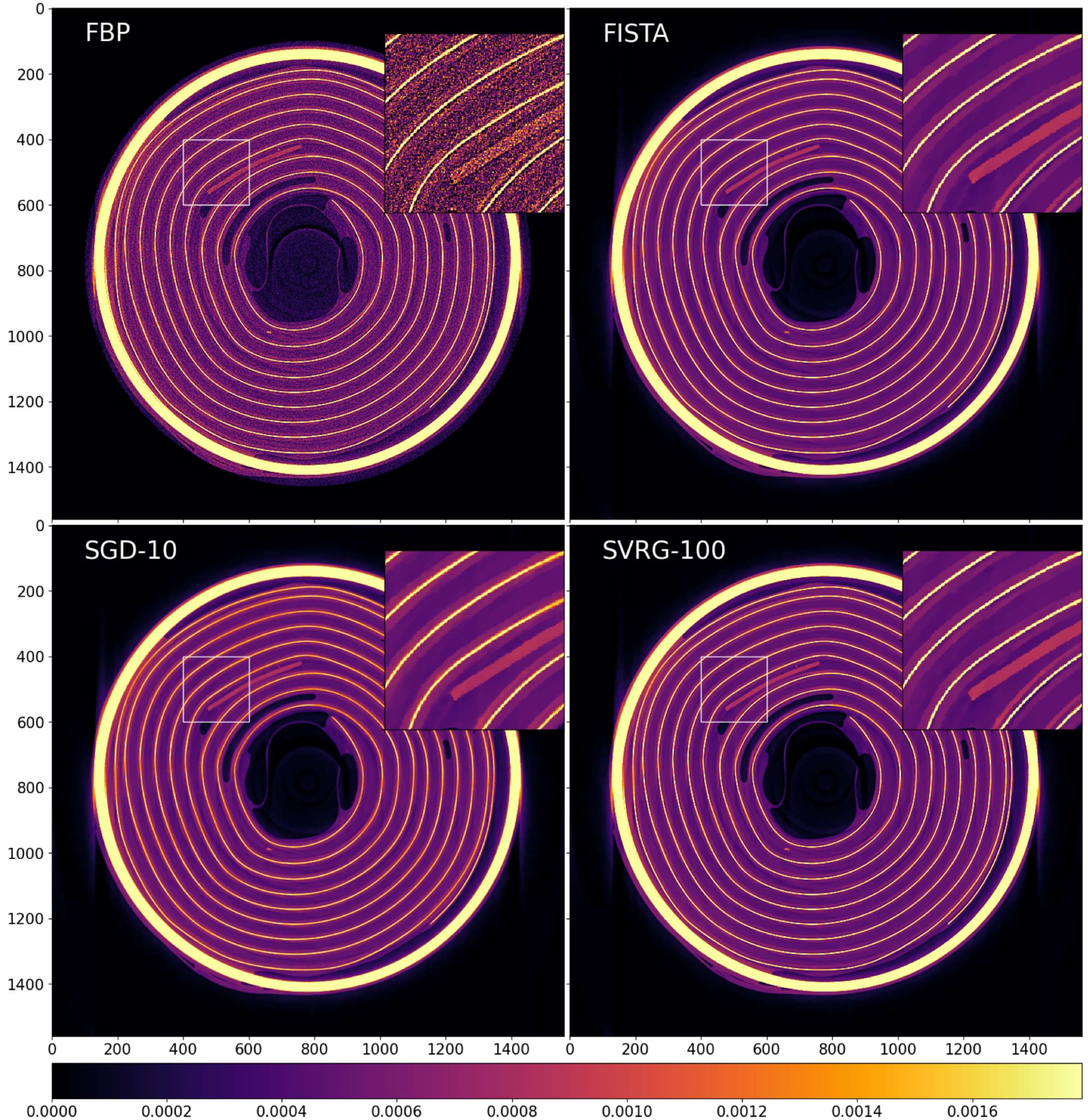
$$\min_{x} \left\{ \frac{1}{2}\|Ax - d\|^2 + \alpha\|\nabla x\|_{2,1} + \mathbb{I}_{\{x>0\}}(x) \right\}$$

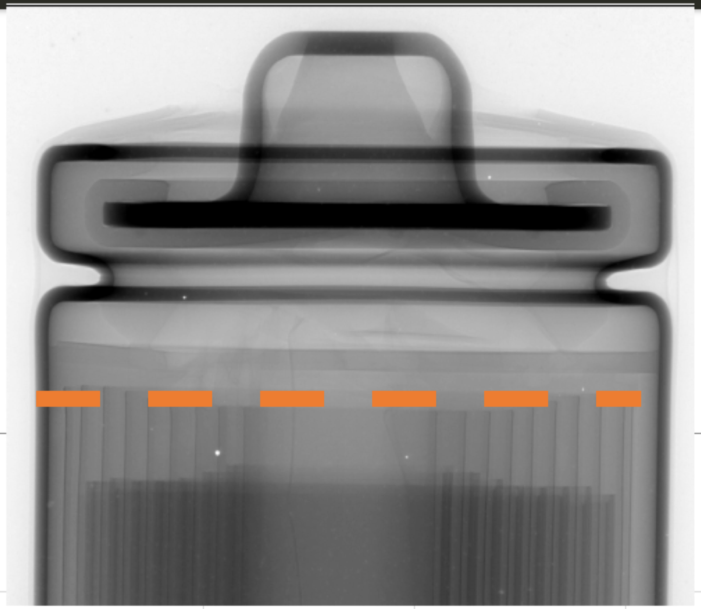


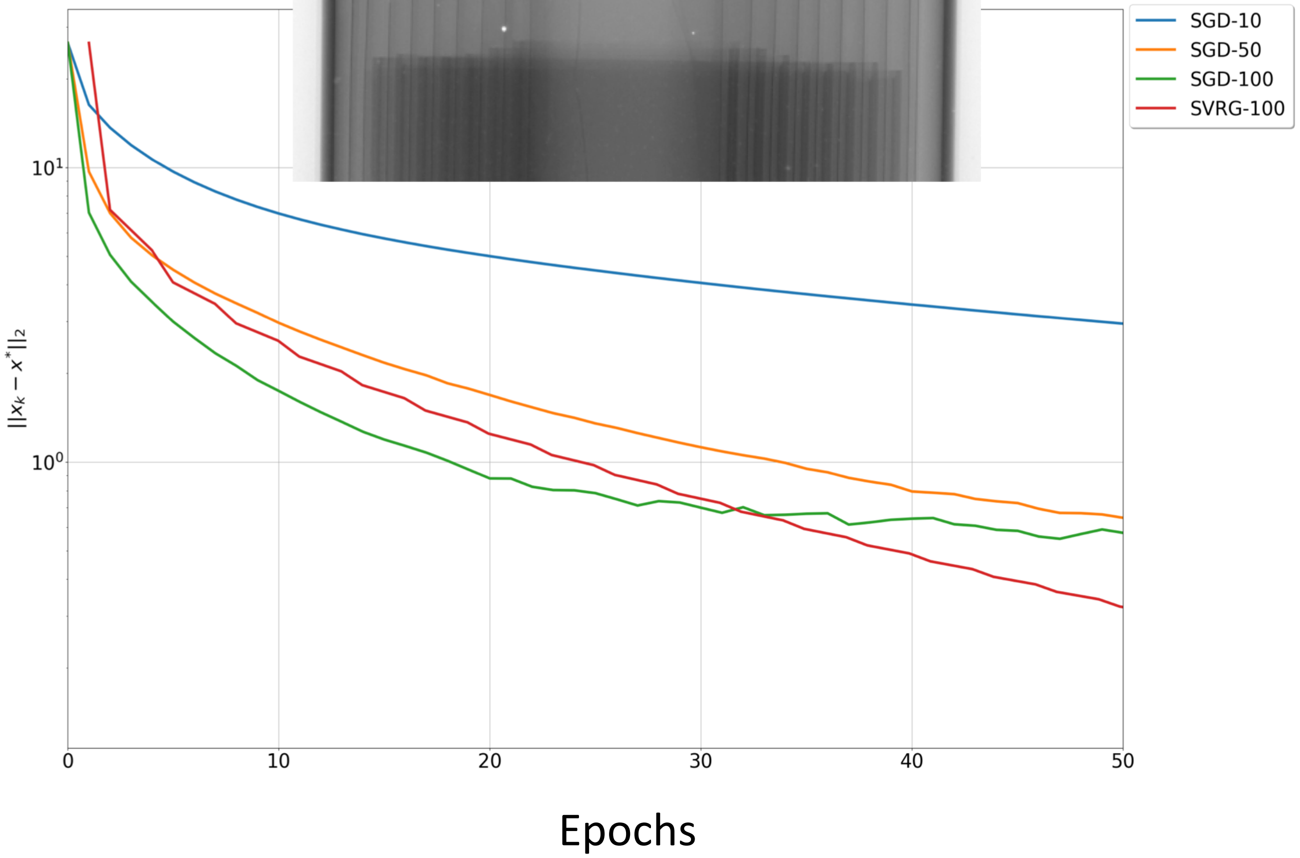
Finden

Innovate UK KTN

Applications (CT - CIL)



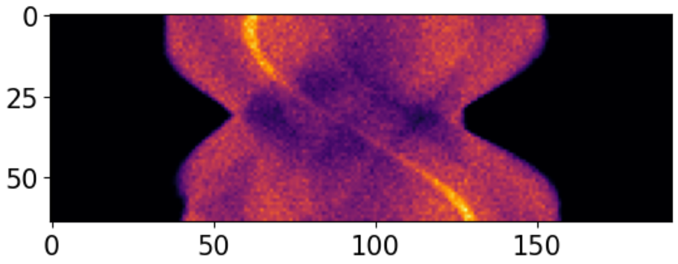


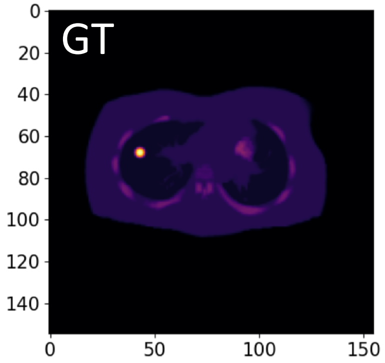


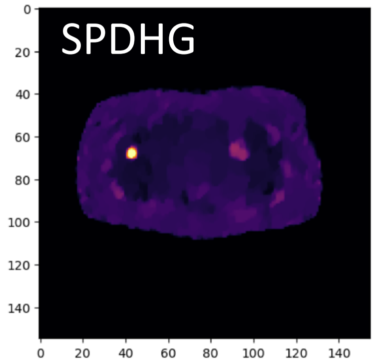
Applications (PET - SIRF)

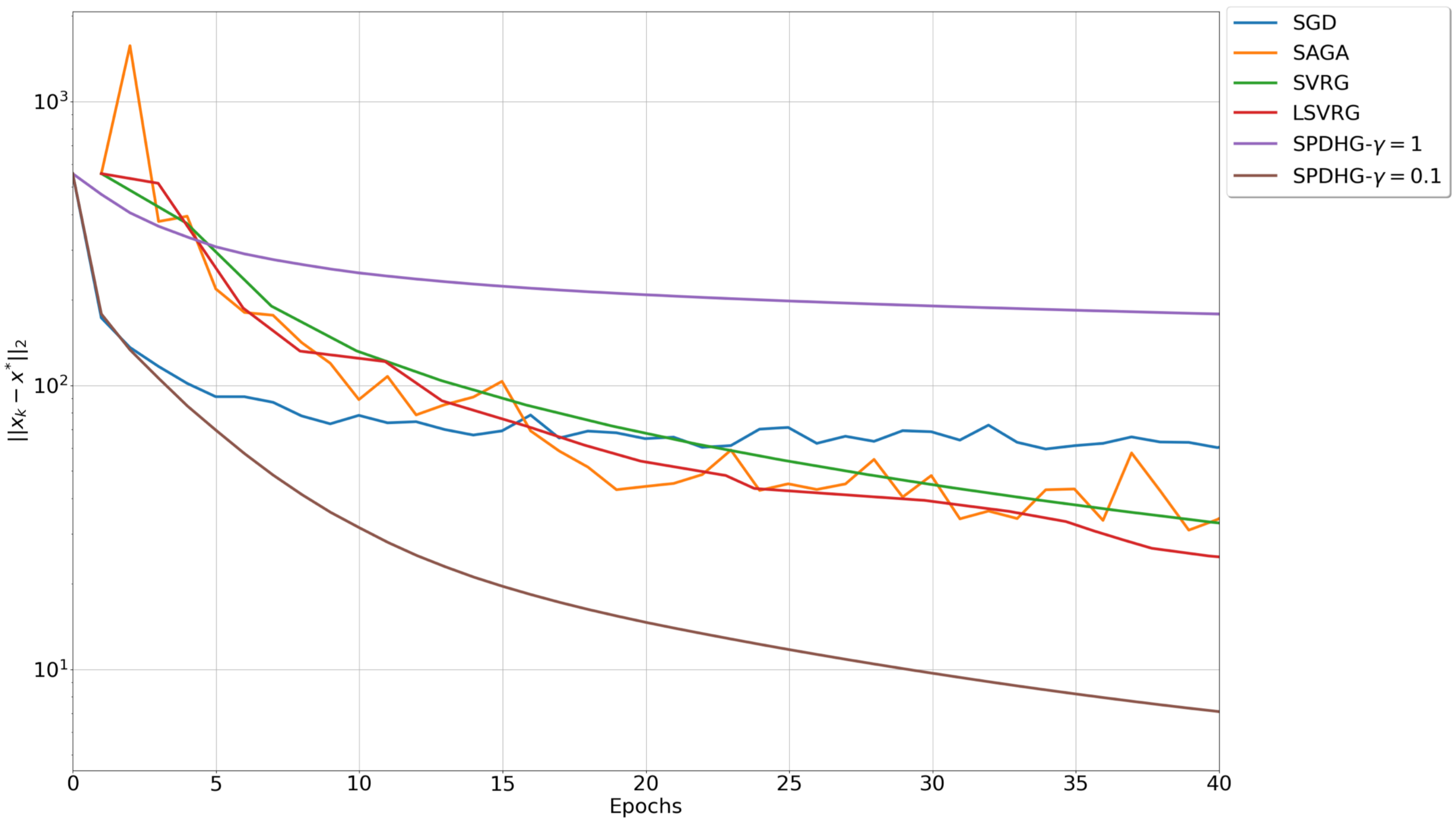
- **Splitting Method (Data): Ordered (64 projections)**
- **Sampling Method (Functions): Random with replacement**
- **Subsets = 64**
- **Optimal Solution: FISTA 1000 iterations**

$$\min_x \int Ax - d\log(Ax+\eta) + \alpha\|\nabla x\|_{2,1} + \mathbb{I}_{\{x>0\}}(x)$$







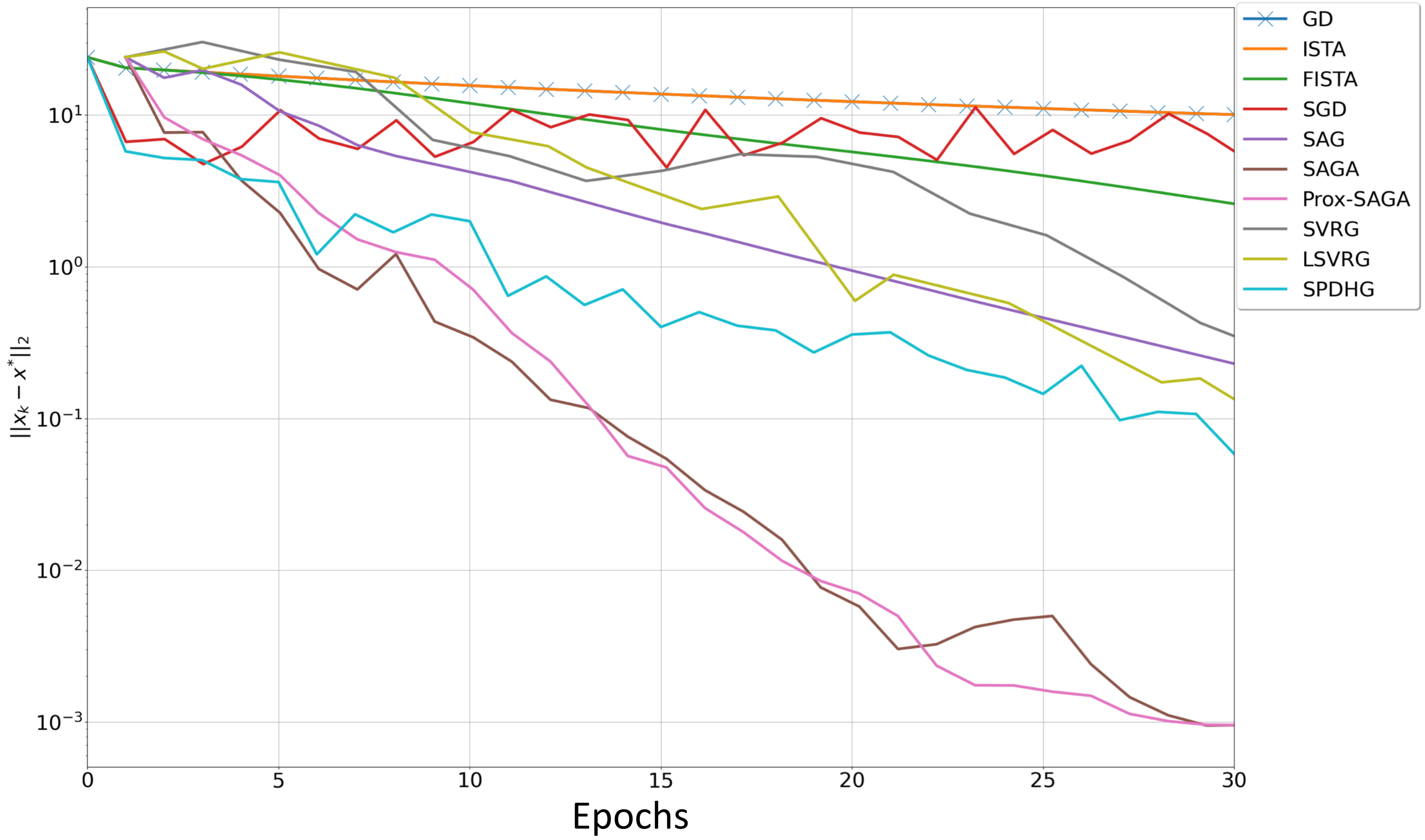


SyneRBI

Applications (ML)

- **Ridge Regression**
- **Housing (Boston) Dataset (LIBSVM): 404 samples, 13 features**
- **Optimal Solution: CVXpy**

$$\min_x \|Ax - d\|^2 + \alpha\|x\|^2$$



Summary

- **Stochastic Optimisation Framework in CIL**
- **Flexible design with Plug and Play Stochastic Estimators**
- **Different modalities CT, PET, SPECT, MRI**
- **Improvements for CIL Optimisation Module**

- Website https://www.ccpi.ac.uk/CIL
- Docs https://tomographicimaging.github.io/CIL
- Discord https://discord.gg/kmBcU2kebB
- Contact evangelos@finden.co.uk, epapoutsellis@gmail.com

Thank you! Questions?