DeepOBS

A Deep Learning Optimizer Benchmark Suite

Frank Schneider

30 Sep 2019

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Max Planck Institute for Intelligent Systems

imprs-is

The Status Quo of Benchmarking Deep Learning Optimizers

...far from optimal!

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Everyone creates their own benchmark

- Repeated work (including bugs)
- Benchmarks are not comparable (not even the metric)
- Potential for cherry-picking

Doing proper benchmarks requires time

- Use of rather small test problems (MNIST)
- Use only a few test problems
- Own optimizer gets more attention than the competition

The Goal of DeepOBS

...we are not quite there yet!

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

1. Run the optimizer on a test problem
2. Compare to the state of the art
3. Plot the results

Running your Optimizer

...as simple as possible


```python
import tensorflow as tf
from deepobs import tensorflow as tfobs

optimizer_class = tf.train.RMSPropOptimizer
hyperparams = {"learning_rate": {"type": float},
               "decay": {"type": float, "default": 0.9}}

runner = tfobs.runners.StandardRunner(optimizer_class, hyperparams)
runner.run()
```

Figure: run_rmsprop.py

```
python run_rmsprop.py mnist_mlp --learning_rate=1e-3
```

```
********************************
Evaluating after 0 of 100 epochs...
TRAIN: loss 353.934
VALID: loss 353.813
TEST: loss 353.447
********************************
********************************
Evaluating after 1 of 100 epochs...
TRAIN: loss 338.48
```

The DeepOBS Test Problems

...still room for more

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Data set	Model	Description	Conv	RNN	Drop	BN	WD
2D	Noisy Beale	*Noisy version of the Beale function*					
	Noisy Branin	*Noisy version of the Branin function*					
	Noisy Rosenbrock	*Noisy version of the Rosenbrock function*					
Quadratic	N-Dimensional	*100-dimensional ill-conditioned noisy quadratic*					
MNIST	Log. Regr.	*Logistic regression*					
	MLP	*Four layer fully-connected network*					
	2c2d	*Two conv. and two fully-connected layers*	✓				
	VAE	*Variational Autoencoder*	✓		✓		
Fashion MNIST	Log. Regr.	*Logistic regression*					
	MLP	*Four layer fully-connected network*					
	2c2d	*Two conv. and two fully-connected layers*	✓				
	VAE	*Variational Autoencoder*	✓		✓		
CIFAR-10	3c3d	*Three conv. and three fully-connected layers*	✓				✓
	VGG 16	*Adapted version of VGG 16*	✓		✓		✓
	VGG 19	*Adapted version of VGG 19*	✓		✓		✓
CIFAR-100	3c3d	*Three conv. and three fully-connected layers*	✓				✓
	VGG 16	*Adapted version of VGG 16*	✓		✓		✓
	VGG 19	*Adapted version of VGG 19*	✓		✓		✓
	All-CNN-C	*The all convolutional net*	✓		✓		✓
	Wide ResNet-40-4	*Wide Residual Network*	✓			✓	✓
SVHN	3c3d	*Three conv. and three fully-connected layers*	✓				✓
	Wide ResNet-16-4	*Wide Residual Network*	✓			✓	✓
ImageNet	VGG 16	*VGG 16*	✓		✓		✓
	VGG 19	*VGG 19*	✓		✓		✓
	Inception-v3	*Inception-v3 network*	✓		✓	✓	✓
Tolstoi	CharRNN	*Recurrent Neural Network for character-level language modeling*		✓	✓		

Running your Optimizer

...the results


```
results
└── quadratic_deep
    └── RMSPropOptimizer
        ├── num_epochs__100__batch_size__128__decay__9.e-01__learning_rate__1.e-02
        │   ├── random_seed__42__2019-09-27-15-15-59.json
        │   └── random_seed__43__2019-09-27-15-19-33.json
        └── num_epochs__100__batch_size__128__decay__9.e-01__learning_rate__1.e-03
            └── random_seed__42__2019-09-27-15-01-49.json
```

Tuning your Hyperparameters

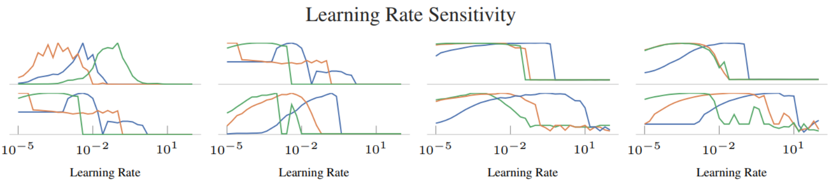
...a semi-automated process

```python
import numpy as np
from torch.optim import SGD
from deepobs.pytorch.runners import StandardRunner
from deepobs.tuner import GridSearch

optimizer_class = SGD
hyperparams = {"lr": {"type": float}}

grid = {"lr": np.logspace(-5, 2, 6)}

tuner = GridSearch(optimizer_class, hyperparams, grid, runner=StandardRunner)

tuner.tune('quadratic_deep', rerun_best_setting=True)
```

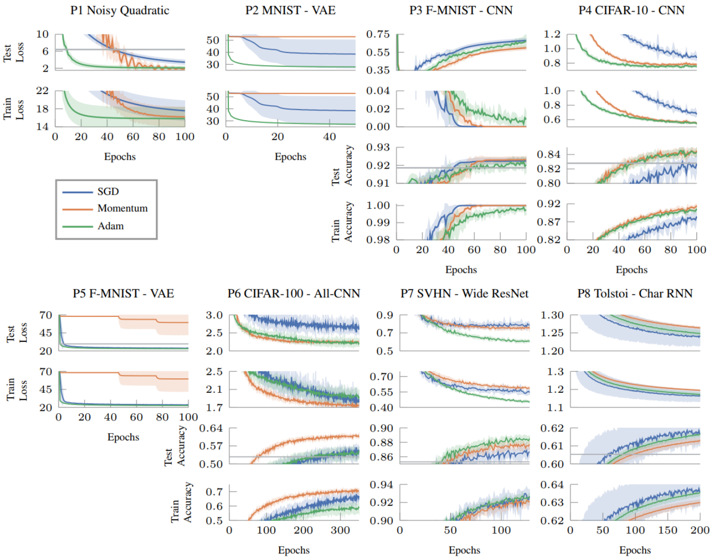
Plotting the Results

...getting page 7 of your paper


```
deepobs_plot_results results/ --full
```



```
deepobs_plot_results results/ --full
```



Plotting the Results

...getting page 7 of your paper

```
deepobs_plot_results results/ --full
```

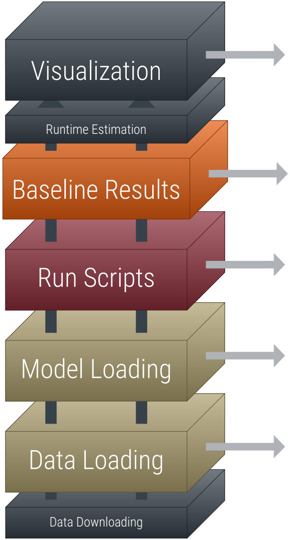
Table 2: DEEPOBS benchmark for the baseline optimizers.

Test Problem		SGD	Momentum	Adam
P1 Noisy Quadratic	Performance	3.42	1.92	2.08
	Speed	45.60	36.90	9.00
	Tuneability	α: 3.98e-03	α:3.98e-04 μ: 0.99	α: 1.00e-01 ϵ: 1e-08 β_1: 0.9 β_2: 0.999
P2 MNIST VAE	Performance	38.54	52.97	27.86
	Speed	1.00	1.00	1.00
	Tuneability	α: 3.98e-03	α: 1.58e-05 μ: 0.99	α: 1.58e-04 ϵ: 1e-08 β_1: 0.9 β_2: 0.999
P3 F-MNIST CNN	Performance	92.25 %	92.32 %	92.03 %
	Speed	38.70	51.10	39.20
	Tuneability	α: 1.58e-01	α: 1.00e-03 μ: 0.99	α: 2.51e-04 ϵ: 1e-08 β_1: 0.9 β_2: 0.999
P4 CIFAR-10 CNN	Performance	82.76 %	84.53 %	84.30 %
	Speed	77.10	41.00	44.60
	Tuneability	α: 1.58e-02	α: 3.98e-04 μ: 0.99	α: 2.51e-04 ϵ: 1e-08 β_1: 0.9 β_2: 0.999

Test Problem		SGD	Momentum	Adam
P5 F-MNIST VAE	Performance	23.94	59.44	23.11
	Speed	3.90	93.30	1.50
	Tuneability	α: 3.98e-03	α: 2.51e-04 μ: 0.99	α: 1.58e-04 ϵ: 1e-08 β_1: 0.9 β_2: 0.999
P6 CIFAR-100 All CNN C	Performance	55.39 %	60.79 %	54.34 %
	Speed	167.40	72.20	194.00
	Tuneability	α: 1.58e-01	α: 3.98e-03 μ: 0.99	α: 1.00e-03 ϵ: 1e-08 β_1: 0.9 β_2: 0.999
P7 SVHN Wide ResNet	Performance	86.69 %	87.74 %	88.53 %
	Speed	40.00	42.10	34.60
	Tuneability	α: 2.51e-01	α: 3.98e-03 μ: 0.99	α: 6.31e-04 ϵ: 1e-08 β_1: 0.9 β_2: 0.999
P8 TOLSTOI Char RNN	Performance	61.71 %	61.29 %	61.68 %
	Speed	57.90	96.30	79.20
	Tuneability	α: 2.51e+00	α: 3.98e-02 μ: 0.99	α: 6.31e-04 ϵ: 1e-08 β_1: 0.9 β_2: 0.999

The DeepOBS Stack

...a modular environment



`.tex` files of learning curves for new optimizer and the baselines.

Performances results of the most popular optimizers.

Optimization performance of an optimizer on a specific test problem.

Losses and accuracy of a deep learning model.

Pre-processed and batched data.

The DeepOBS Leaderboard

...a work in progress

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

deepobs.github.io

Leaderboard

Overview over the current optimizer leaderboard on the DeepOBS test problems. Click on See Full Results to see the plots and tables of the full benchmarking results.

Quadratic Deep

A 100-dimensional noisy quadratic problem with an eigenspectrum similar to the one reported for deep neural networks.

	Optimizer	Test Loss	Speed
#1	Momentum	87.05	70.5
#2	Adam	87.11	39.9
#3	SGD	87.40	51.1

See Full Results

MNIST - VAE

A basic variational autoencoder for the MNIST data set with three convolutional and three deconvolutional layers.

	Optimizer	Test Loss	Speed
#1	Adam	27.83	1.0
#2	SGD	38.46	1.0
#3	Momentum	52.93	1.0

See Full Results

F-MNIST - CNN

A simple convolutional network for the Fashion-MNIST data set, consisting of two conv and two fully-connected layers.

	Optimizer	Test Accuracy	Speed
#1	Adam	92.34 %	40.1
#2	SGD	92.27 %	40.6
#3	Momentum	92.14 %	59.1

See Full Results

CIFAR-10 - CNN

A slightly larger convolutional network for the Cifar-10 data set, with three conv and three fully-connected layers.

	Optimizer	Test Accuracy	Speed
#1	Adam	84.75 %	36.0
#2	Momentum	84.41 %	40.7
#3	SGD	83.71 %	42.5

See Full Results

F-MNIST - VAE

A basic variational autoencoder for the Fashion-MNIST data set with three convolutional and three deconvolutional layers.

	Optimizer	Test Loss	Speed
#1	Adam	23.07	1.0
#2	SGD	23.80	1.0
#3	Momentum	59.23	1.0

See Full Results

CIFAR-100 - All CNN C

Variant C of the All Convolutional Network from **Striving for Simplicity** for the CIFAR-100 data set consisting solely of convolutional layers.

	Optimizer	Test Accuracy	Speed
#1	Momentum	60.33 %	72.8
#2	SGD	57.06 %	128.7
#3	Adam	56.15 %	152.6

See Full Results

SVHN - Wide ResNet 16-4

The **Wide ResNet 16-4** for the Street View House Numbers data set using the variant with 16 conv layers and a widening factor of 4.

	Optimizer	Test Accuracy	Speed
#1	Momentum	95.53 %	10.8
#2	SGD	95.37 %	28.3
#3	Adam	95.25 %	12.1

See Full Results

Tolstoi - Char RNN

A recurrent neural network for character-level language modeling on the novel *War and Peace* by *Leo Tolstoy* using two LSTM layers.

	Optimizer	Test Accuracy	Speed
#1	SGD	62.07 %	47.7
#2	Momentum	61.30 %	88.0
#3	Adam	61.23 %	62.8

See Full Results

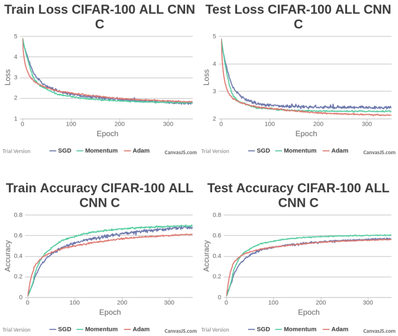
The DeepOBS Leaderboard

...a work in progress

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

deepobs.github.io

Leaderboard CIFAR-100 - All CNN C



Overview Table

Rank	Optimizer	Final Test Accuracy	Best Test Accuracy	Final Train Accuracy	Best Train Accuracy	Final Test Loss	Best Test Loss	Final Train Loss	Best Train Loss	Speed
1	Momentum	60.33%	60.83%	69.25%	70.25%	2.29	2.24	1.79	1.75	72.8
2	SGD	57.06%	57.77%	68.48%	69.31%	2.39	2.36	1.74	1.70	128.7
3	Adam	56.15%	56.41%	60.89%	61.52%	2.13	2.12	1.83	1.81	152.6

The DeepOBS ecosystem

...lots of ways to participate

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

>_ Install DeepOBS

```
pip install deepobs
```

TensorFlow

PyTorch
Coming soon

Visit deepobs.github.io

Leaderboard

Overview over the current optimizer leaderboard on the DeepOBS test problems. Click on See Full Results to see the plots and tables of the full benchmarking results.

Quadratic Deep

A 100-dimensional noisy quadratic problem with an eigenspectrum similar to the one reported for deep neural networks.

	Optimizer	Test Loss	Speed
#1	Momentum	87.05	70.5
#2	Adam	87.11	39.9
#3	SGD	87.40	51.1

See Full Results

MNIST - VAE

A basic variational autoencoder for the MNIST data set with three convolutional and three deconvolutional layers.

	Optimizer	Test Loss	Speed
#1	Adam	27.83	1.0
#2	SGD	38.46	1.0
#3	Momentum	52.93	1.0

See Full Results

F-MNIST - CNN

A simple convolutional network for the Fashion-MNIST data set, consisting of two conv and two fully-connected layers.

	Optimizer	Test Accuracy	Speed
#1	Adam	92.34 %	40.1
#2	SGD	92.27 %	40.6
#3	Momentum	92.14 %	59.1

See Full Results

CIFAR-10 - CNN

A slightly larger convolutional network for the Cifar-10 data set, with three conv and three fully-connected layers.

	Optimizer	Test Accuracy	Speed
#1	Adam	84.75 %	36.0
#2	Momentum	84.41 %	40.7
#3	SGD	83.71 %	42.5

See Full Results

F-MNIST - VAE

A basic variational autoencoder for the Fashion-MNIST data set with three convolutional and three deconvolutional layers.

	Optimizer	Test Loss	Speed
#1	Adam	23.07	1.0
#2	SGD	23.80	1.0
#3	Momentum	59.23	1.0

See Full Results

CIFAR-100 - All CNN C

Variant C of the All Convolutional Network from **Striving for Simplicity** for the CIFAR-100 data set consisting solely of convolutional layers.

	Optimizer	Test Accuracy	Speed
#1	Momentum	60.33 %	72.8
#2	SGD	57.06 %	128.7
#3	Adam	56.15 %	152.6

See Full Results

SVHN - Wide ResNet 16-4

The **Wide ResNet 16-4** for the Street View House Numbers data set using the variant with 16 conv layers and a widening factor of 4.

	Optimizer	Test Accuracy	Speed
#1	Momentum	95.53 %	10.8
#2	SGD	95.37 %	28.3
#3	Adam	95.25 %	12.1

See Full Results

Tolstoi - Char RNN

A recurrent neural network for character-level language modeling on the novel *War and Peace* by *Leo Tolstoy* using two LSTM layers.

	Optimizer	Test Accuracy	Speed
#1	SGD	62.07 %	47.7
#2	Momentum	61.30 %	88.0
#3	Adam	61.23 %	62.8

See Full Results

Check out Github for the Beta of DeepOBS 1.2.0

`fsschneider.github.io/DeepOBS`