$\mathcal{L}$ SciCoNet: **Sci**entific **Co**mputing Neural **Net**works

Lu Lu

Mar 27, 2019 @Crunch Seminar

Why SciCoNet?

- A deep learning library designed for scientific computing on top of TensorFlow.

Being able to go from idea to result with the least possible delay is key to doing good research. — Keras

SciCoNet

Use SciCoNet if you need a deep learning library that

- approximates **functions** from a dataset with **constraints**,
- approximates **functions** from **multi-fidelity** data,
- solves partial differential equations (**PDE**s),
- solves integro-differential equations (**IDE**s),
- solves fractional partial differential equations (**fPDE**s),
- ...

Why $\mathcal{L}$?

The art of physics informed neural networks?

$\Rightarrow$ Design loss $\mathcal{L}$

Features

SciCoNet supports

- **uncertainty quantification** using dropout;
- domain **geometries**: interval, disk, hyercube and hypersphere;
- **networks**: fully connected, & ResNet;
- many different losses, metrics, optimizers, learning rate schedules, initializations, regularizations, etc.;
- **callbacks** to monitor the internal states and statistics of the model during training.

Main modules

Highly-configurable

- domain geometry,
- data, i.e., the type of problems and constraints,
- map, i.e., the function space,
- model, which trains the map to match the data and constraints,

Installation

- Dependencies: Matplotlib, NumPy, SALib, scikit-learn, SciPy, TensorFlow
- Download: `https://github.com/lululxvi/sciconet`

Example 1: Elementary school — dataset

- examples/dataset.py

```
import sciconet as scn

fname_train = "examples/dataset.train"
fname_test = "examples/dataset.test"
data = scn.data.DataSet(
    fname_train=fname_train, fname_test=fname_test,
    col_x=(0,), col_y=(1,)
)

x_dim, y_dim = 1, 1
layer_size = [x_dim] + [50] * 3 + [y_dim]
activation = "tanh"
initializer = "Glorot normal"
net = scn.maps.FNN(layer_size, activation, initializer)
```

Example 1: Elementary school — dataset

```
model = scn.Model(data, net)

optimizer = "adam"
lr = 0.001
batch_size = 0
ntest = 0
model.compile(
    optimizer, lr, batch_size, ntest,
    metrics=["l2 relative error"]
)

epochs = 50000
losshistory, train_state = model.train(epochs)

scn.saveplot(
    losshistory, train_state, issave=True, isplot=True
)
```

Example 2: Middle school — Poisson's equation

- $-\Delta y = \pi^2 \sin(\pi x)$
- $x \in [-1, 1]$
- $y(\pm 1) = 0$

$y(x) = \sin(\pi x)$

- examples/pde.py

Example 3: High school — IDE

- $\int_0^x y(t)dt + \frac{dy}{dx} = 2\pi\cos(2\pi x) + \frac{\sin^2(\pi x)}{\pi}$
- $x \in [0, 1]$
- $y(0) = 0$

$y(x) = \sin(2\pi x)$

- examples/ide.py

Example 4: College — turbulence

Variable fractional model

$$\nu(y) D_y^{\alpha(y)} U(y) = 1, \ \forall y \in (0,1]$$

- $\alpha(0) = 1$, $0 < \alpha \le 1$
- $U(y)$: the mean velocity
- eddy viscosity: $\nu(y) = \Gamma(2-\alpha(y)) Re_\tau^{-\alpha(y)}$
- (Caputo) fractional derivative:

$$\begin{aligned} D_y^{\alpha} U(y) &= \frac{1}{\Gamma(1-\alpha)} \int_0^y (y-\tau)^{-\alpha} U'(\tau) d\tau \\ &\approx \frac{1}{\Gamma(2-\alpha(y))} \sum_{j=0}^{n} ((j+1)^{1-\alpha} - j^{1-\alpha}) \frac{U^{n+1-j} - U^{n-j}}{(\Delta y)^{\alpha(y)}} \end{aligned}$$

$$\nu(y) D_y^{\alpha(y)} U(y) \approx \sum_{j=1}^{n+1} j (j Re \Delta y)^{-\alpha(y)} (U^{n+2-j} - 2U^{n+1-j} + U^{n-j})$$

Example 4: College — turbulence

Re $= 2000$