A Union of Scikit-learn and PyTorch

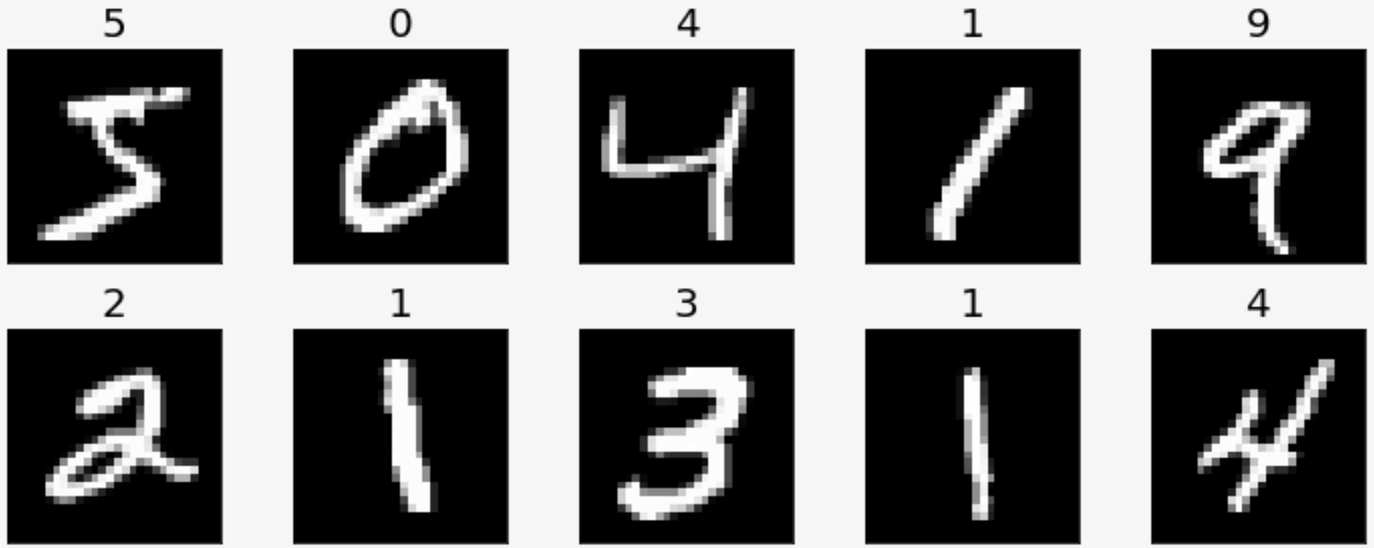
Thomas J Fan
Scikit-learn Developer @ DSI Columbia University


PyTorch
scikit
learn

SKORCH

1. Scikit-Learn compatible neural network library that wraps PyTorch.
2. Abstracts away the training loop.
3. Reduces the amount of boilerplate code with callbacks.

MNIST - Data



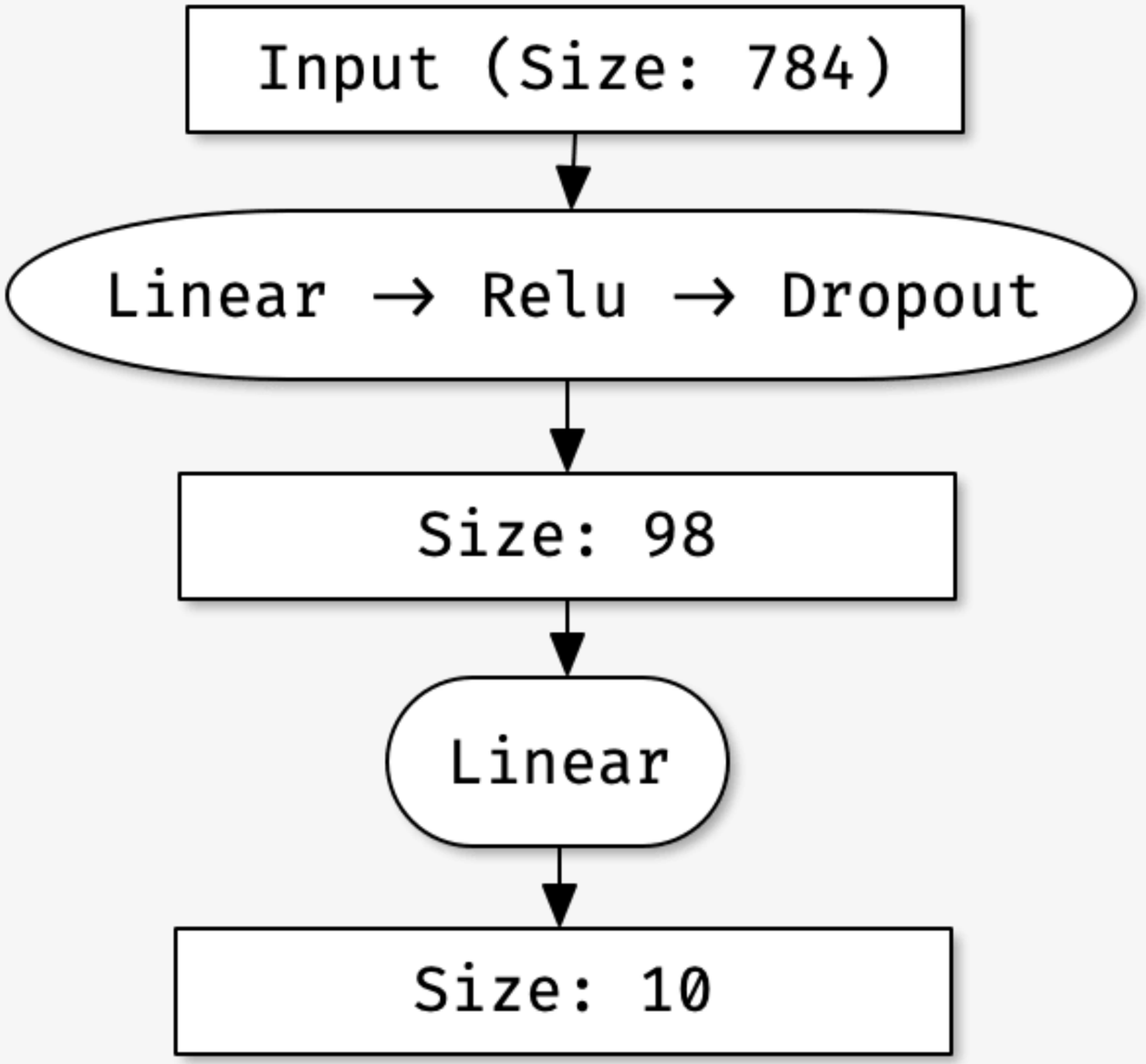

```
print(X.shape, y.shape)
# (70000, 784) (70000,)
```

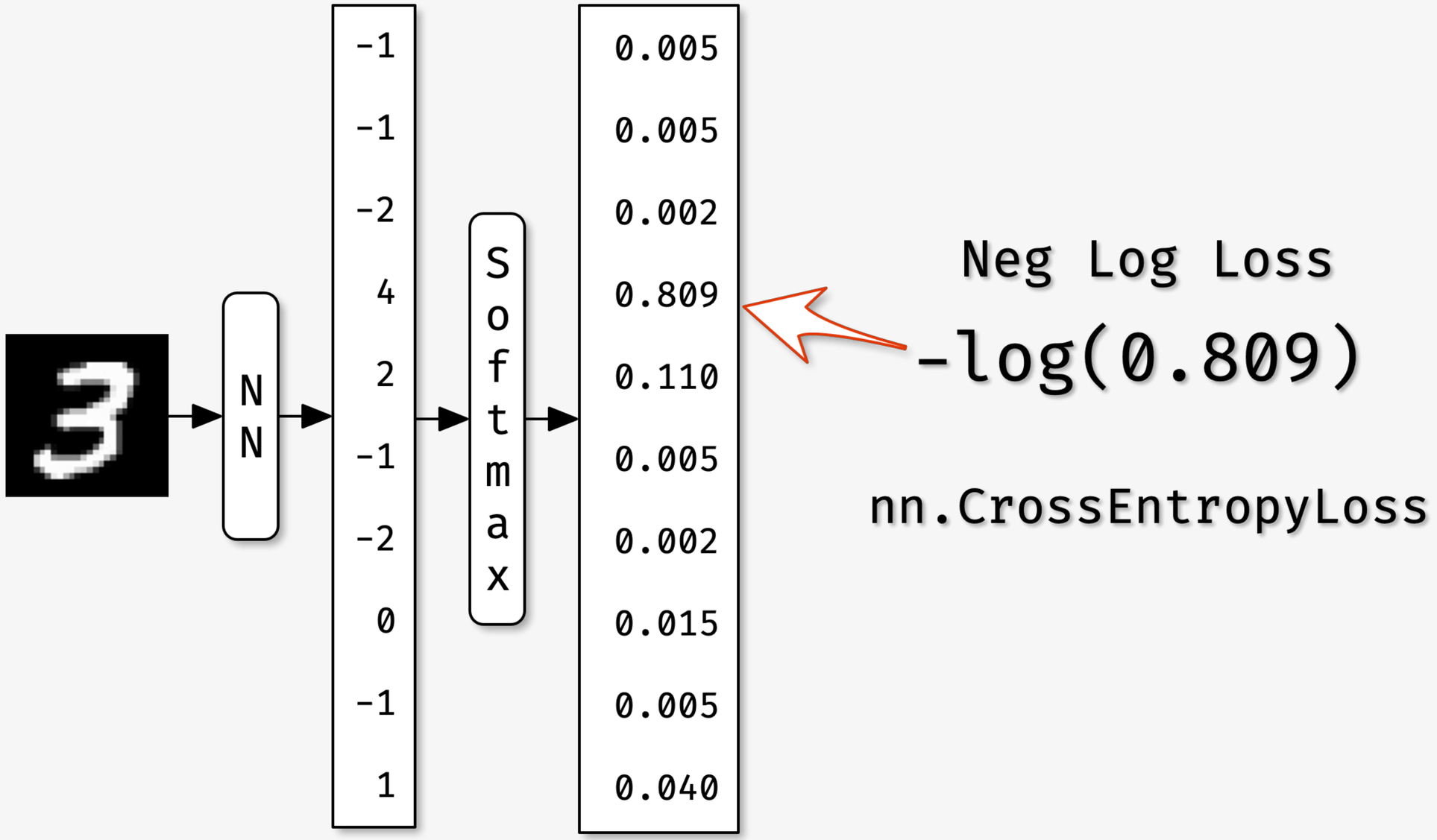
MNIST - Data Code

```
from sklearn.model_selection import train_test_split

X_scaled = X / X.max()

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.25, random_state=42)
```

MNIST - Neural Network Module

```
from torch.nn as nn

class SimpleFeedforward(nn.Module):
    def __init__(self):
        super().__init__()
        self.module =  ...

    def forward(self, X):
        return self.module(X)
```




NN
-1
-1
-2
4
2
-1
-2
0
-1
1
Softmax
0.005
0.005
0.002
0.809
0.110
0.005
0.002
0.015
0.005
0.040
Neg Log Loss
-log(0.809)
nn.CrossEntropyLoss

MNIST - Loss function skorch

```
from skorch import NeuralNet

net = NeuralNet(
    SimpleFeedforward,
    criterion=nn.CrossEntropyLoss,
    max_epochs=10,
    lr=0.3,
    # device='cuda', # uncomment out to run on gpu
)
```

MNIST - Fitting

```
_ = net.fit(X_train, y_train)
```

epoch	train_loss	valid_loss	dur
1	0.5775	0.2973	1.1797
2	0.3208	0.2521	1.4316
3	0.2710	0.1968	1.2767
4	0.2397	0.1940	1.1888
5	0.2192	0.1624	1.2892
6	0.2069	0.1385	1.1394
7	0.1936	0.1300	1.1347
8	0.1856	0.1268	1.1136
9	0.1784	0.1310	1.1285
10	0.1715	0.1239	1.0846

MNIST - Continue Training

```
net.set_params(max_epochs=5)
_ = net.partial_fit(X_train, y_train)
```

11	0.1609	0.1173	1.0857
12	0.1588	0.1164	1.1218
13	0.1551	0.1202	1.0993
14	0.1520	0.1080	1.3218
15	0.1540	0.1319	1.3325

MNIST - History

```
len(net.history)
# 15

net.history[-1, 'valid_loss']
# 0.10163110941932314

net.history[-2:, 'train_loss']
# [0.15514005536124 82,
#  0.15202351114863 44,
#  0.15403895963941303]
```

MNIST - Accuracy Score

```
from sklearn.metrics import make_scorer

def accuracy_argmax(y_true, y_pred):
    return np.mean(y_true == np.argmax(y_pred, axis=-1))

accuracy_argmax_scorer = make_scorer(accuracy_argmax)
```

MNIST - EpochScoring and Checkpointing

```
from skorch.callbacks import EpochScoring

epoch_acc = EpochScoring(
    accuracy_argmax_scorer,
    name='valid_acc',
    lower_is_better=False)

cp = Checkpoint(monitor='valid_acc_best',
                dirname='exp_01')
```

MNIST - Model Checkpointing

```
net = NeuralNet(
    SimpleFeedforward,
    criterion=nn.CrossEntropyLoss,
    max_epochs=10,
    lr=0.8,
    # device='cuda', # uncomment out to run on gpu
    callbacks=[epoch_acc, cp]
)
_ = net.fit(X, y)
```

MNIST - Fitting With Callbacks

epoch	train_loss	valid_acc	valid_loss	cp	dur
1	0.5531	0.8747	0.4470	+	0.7692
2	0.3142	0.9126	0.2912	+	0.7537
3	0.2641	0.9539	0.1579	+	0.7501
4	0.2427	0.9488	0.1874		0.7449
5	0.2268	0.9554	0.1484	+	0.7306
6	0.2141	0.9543	0.1516		0.7583
7	0.2058	0.9529	0.1734		0.7170
8	0.1987	0.9533	0.1547		0.7559
9	0.1918	0.9591	0.1440	+	0.7329
10	0.1818	0.9586	0.1391		0.7389
11	0.1789	0.9656	0.1190	+	0.7413
12	0.1687	0.9527	0.1630		0.7221
13	0.1697	0.9623	0.1210		0.7429
14	0.1646	0.9642	0.1196		0.7989
15	0.1626	0.9522	0.1848		0.7028

MNIST - Prediction

```
net.load_params(checkpoint=cp)

y_pred = net.predict(X_test)
print('test accuracy:', accuracy_argmax(y_test, y_pred))
# test accuracy: 0.9645142857142858
```

MNIST - Scikit-Learn Integration

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler

pipe = Pipeline([
    ("min_max", MinMaxScaler()),
    ("net", net)])

_ = pipe.fit(X_train, y_train)
```

Skorch - Closing

1. Scikit-Learn compatible neural network library that wraps PyTorch.
2. Abstracts away the training loop.
3. Reduces the amount of boilerplate code with callbacks.
    - `EpochScoring`
    - `Freezer`
    - `Checkpoint`
    - `LRScheduler`

Skorch - Whats next

- skorch.readthedocs.io
- skorch Tutorials
- github.com/dnouri/skorch
- github.com/thomasjpfan/python_meetup_feb_19