

GPU Mode

```
caffe.set_mode_gpu()
```

Net

The main class that the pycaffe interface exposes is the Net. It has two constructors:

Create a Net (in this case using the Data Layer specified for training)

```
net = caffe.Net('/path/prototxt/descriptor/file', caffe.TRAIN)
```

Creates a Net and automatically loads the weights as saved in the provided caffemodel file - in this case using the Data Layer specified for testing.

```
net = caffe.Net('/path/prototxt/descriptor/file', '/path/caffemodel/weights/file', caffe.TEST)
```

Parameters

```
nice_edge_detectors = net.params['conv'].data
```

```
higher_level_filter = net.params['fc'].data
```

backward()

Computing gradients

```
net.backward(start='conv1', end='fc')
```

```
softmax_probabilities = net.blobs['prob'].data
```

Transformer

```
transformer = caffe.io.Transformer({'data': (1, image.shape[2], image.shape[0], image.shape[1])})
```

PoolMethod

```
caffe.params.Pooling
```

Monitoring

```
tools.solvers.MonitoringSolver
```

LMDB I/O

```
import tools.lmdb_io
```

Prediction

```
pred = net.predict([input])
```

CPU Mode

```
caffe.set_mode_cpu()
```

Net.blobs

```
data = net.blobs['data'].data
```

```
net.blobs['data'].data[...] = my_image
```

```
fc_activations = net.blobs['fc'].data
```

Solver iteration

A forward/backward pass with weight update

```
solver.step(1)
```

Run the solver until the last iteration

```
solver.solve()
```

forward()

Add Data to the net

```
net.forward(start='conv', end='fc')
```

```
softmax_probabilities = net.blobs['prob'].data
```

Solver

Solver needed in order to train a caffe mode

```
solver = caffe.SGDSolver('/path/to/solver/prototxt/file')
```

The networks are accessible with

The networks are accessible with

```
training_net = solver.net test_net = solver.test_nets[0] # more than one test net is supported
```

Data augmentation

```
tools.data_augmentation
```

Transformation

```
tools.prototxt.train2deploy
```

Pre-processing

```
import tools.pre_processing
```

Image Input

```
input_image = caffe.io.load_image(IMAGE_FILE)
```

