

Dataflow Programming: Theano Example

```
$ python
Python 2.7.6
>>> import theano
Using qpu device 0: GeForce GTX TITAN X (CNMeM is disabled, cuDNN 5005)
>>> a = theano.tensor.scalar() # symbolic variable
>>> b = theano.tensor.scalar() # symbolic variable
>>> c = a+b # define computation graph - this line is executed only once!
>>> f = theano.function([a,b], c) # compile fcn. that maps [a,b] to c
>>> f(1,1000)
array(1001.0, dtype=float32)
>>> f(2,2)
array(4.0, dtype=float32)
>>> f(1,2) # execute f as often as you want
array(3.0, dtype=float32)
>>> type (a)
<class 'theano.tensor.var.TensorVariable'>
>>> type (b)
<class 'theano.tensor.var.TensorVariable'>
>>> type (c)
<class 'theano.tensor.var.TensorVariable'>
```



Neural Networks in Lasagne

```
inputs = theano.tensor.tensor4() # 4-dimensional symbolic array
targets = theano.tensor.tensor4() # 4-dimensional symbolic array
nchannels = 3
network = lasagne.layers.InputLayer(shape=(None, nchannels, None, None), input var=inputs)
network = lasagne.layers.Conv2DLayer(network, num filters=128, filter size=(3,3))
network = lasagne.layers.Pool2DLayer(network, pool size=(2,2))
network = lasagne.layers.Conv2DLayer(network, num filters=256, filter size=(3,3))
network = lasagne.layers.GlobalPoolLayer(network, pool function=theano.tensor.max)
network = lasagne.layers.DenseLayer(network, num units=1024)
network = lasagne.layers.DropoutLayer(network, p=0.5)
network = lasagne.layers.DenseLayer(network, num units=1)
# optional arguments: nonlinearity, weight initialization, stride, pad, ...
predictions = lasagne.layers.get output(network)
                                                                                  # symbolic
trainable params = lasagne.layers.get all params(network, trainable=True)
                                                                                 # symbolic
                 = lasagne.objectives.squared error(predictions, targets).mean() # symbolic
loss
                = lasagne.updates.adam(loss, trainable params, learning rate=0.01)
adam updates
train fn = theano.function([inputs, targets], [predictions, loss], updates=adam updates)
for i in range (10000):
                                                          # mini-batch training loop
   X,Y = my minibatch producer.get()
                                                         # load data
   minibatch predictions, minibatch loss = train fn(X,Y) # train
```

Vladimir Golkov (TUM) 2