Deep Learning Frameworks

使用不同的框架來實作訓練

- **Caffe**
- **Theano**
- **Torch**

Caffe

- **CIFAR-10**
- **60000 32X32 color images**
- **split into 10 classes**
  - airplane, automobile, bird, cat, deer
  - dog, frog, horse, ship, truck
- **$caffe_path/data/get_cifar10.sh**
- **$caffe_path/examples/cifar10/create_cifar10.sh**
- **Blob ，Layer ，Net ，Solve**

Caffe - Blob -- 補充

The type of the data come from net is Blob:

```
{'prob': array([[ 0.,  0.,  0.,  0.,  0.,
1.,  0.,  0.,  0.,  0.]], dtype=float32)}
```

We need to convert it to matrix that we can use:

```
[[ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]]
```

This is second method that can extract any data from any layer:

```
[[ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]]
```

Caffe - Layer -- 補充

```
layer {
  name: "conv1"
  type: "Convolution"
// type: Data, Convolution, Pooling, InnerProduct, ReLU, LRN,
//       DropOut, SoftmaxWithLoss, Accuracy
// type = Data: data (lmdb, leveldb), MemoryData
//       HDF5Data, ImagesData, WindowsData, DummyData

  bottom: "data" # 輸入層：數據層
 top: "conv1" # 輸出層：卷積層1

  # 濾波器（filters）的學習速率因子和衰減因子
 param { lr_mult: 1 decay_mult: 1 }

  # 偏置項（biases）的學習速率因子和衰減因子
 param { lr_mult: 2 decay_mult: 0 }

  convolution_param {
    num_output: 96 # 96個濾波器（filters）
  kernel_size: 11 # 每個濾波器（filters）大小為11*11
    stride: 4 # 每次濾波間隔為4個像素
  weight_filler {
      type: "gaussian" # 初始化高斯濾波器（Gaussian）
   std: 0.01 # 標準差為0.01，均值默認為0
    }
    bias_filler {
      type: "constant" # 初始化偏置項（bias）為零
   value: 0
    }
  }
//source code: src/layers/convolution_layer.cpp | cu
}
```

Caffe - net

```
...
layer {
  name: "cifar"
  type: "Data"
  top: "data"
  top: "label"
  transform_param {
    mean_file: "examples/cifar10/mean.binaryproto"
  }
  data_param {
    source: "examples/cifar10/cifar10_train_lmdb"
    batch_size: 100
    backend: LMDB
  }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  ...
  }
}
...
```

Caffe - solver

```
# reduce the learning rate after 8 epochs (4000 iters) by a factor of 10

# The train/test net protocol buffer definition
net: "examples/cifar10/cifar10_quick_train_test.prototxt"
# test_iter specifies how many forward passes the test should carry out.
# In the case of MNIST, we have test batch size 100 and 100 test
iterations,
# covering the full 10,000 testing images.
test_iter: 100
# Carry out testing every 500 training iterations.
test_interval: 500
# The base learning rate, momentum and the weight decay of the network.
base_lr: 0.001
momentum: 0.9
weight_decay: 0.004
# The learning rate policy
lr_policy: "fixed"
# Display every 100 iterations
display: 100
# The maximum number of iterations
max_iter: 4000
# snapshot intermediate results
snapshot: 4000
snapshot_prefix: "examples/cifar10/cifar10_quick"
# solver mode: CPU or GPU
solver_mode: GPU
```

Caffe - train

```
cd $caffe_path
./build/tools/caffe train
--solver=./examples/cifar10/cifar10_quick_solver.prototxt

#resume training from the half-way
point snapshot
caffe train -solver
examples/mnist/lenet_solver.prototxt
-snapshot
examples/mnist/lenet_iter_5000.solverstate
```

Caffe - time -- 補充

```
caffe time -model examples/mnist/lenet_train_test.prototxt
-iterations 10
```

```
I1005 05:58:47.061524  1233 caffe.cpp:374] *** Benchmark begins ***
I1005 05:58:47.061544  1233 caffe.cpp:375] Testing for 10 iterations.
I1005 05:58:47.115362  1233 caffe.cpp:403] Iteration: 1 forward-backward time: 53 ms.
I1005 05:58:47.169019  1233 caffe.cpp:403] Iteration: 2 forward-backward time: 53 ms.
I1005 05:58:47.222576  1233 caffe.cpp:403] Iteration: 3 forward-backward time: 53 ms.
I1005 05:58:47.275920  1233 caffe.cpp:403] Iteration: 4 forward-backward time: 53 ms.
I1005 05:58:47.329260  1233 caffe.cpp:403] Iteration: 5 forward-backward time: 53 ms.
I1005 05:58:47.383637  1233 caffe.cpp:403] Iteration: 6 forward-backward time: 54 ms.
I1005 05:58:47.438005  1233 caffe.cpp:403] Iteration: 7 forward-backward time: 54 ms.
I1005 05:58:47.491442  1233 caffe.cpp:403] Iteration: 8 forward-backward time: 53 ms.
I1005 05:58:47.544661  1233 caffe.cpp:403] Iteration: 9 forward-backward time: 53 ms.
I1005 05:58:47.598136  1233 caffe.cpp:403] Iteration: 10 forward-backward time: 53 ms.
I1005 05:58:47.598157  1233 caffe.cpp:406] Average time per layer:
I1005 05:58:47.598170  1233 caffe.cpp:409]      mnist    forward: 0.0107 ms.
I1005 05:58:47.598176  1233 caffe.cpp:412]      mnist    backward: 0.0006 ms.
I1005 05:58:47.598179  1233 caffe.cpp:409]      conv1    forward: 5.9642 ms.
I1005 05:58:47.598182  1233 caffe.cpp:412]      conv1    backward: 5.9309 ms.
I1005 05:58:47.598186  1233 caffe.cpp:409]      pool1    forward: 2.7406 ms.
I1005 05:58:47.598188  1233 caffe.cpp:412]      pool1    backward: 0.5376 ms.
I1005 05:58:47.598191  1233 caffe.cpp:409]      conv2    forward: 10.0786 ms.
I1005 05:58:47.598196  1233 caffe.cpp:412]      conv2    backward: 19.9588 ms.
I1005 05:58:47.598208  1233 caffe.cpp:409]      pool2    forward: 1.6229 ms.
I1005 05:58:47.598212  1233 caffe.cpp:412]      pool2    backward: 0.686 ms.
I1005 05:58:47.598214  1233 caffe.cpp:409]        ip1    forward: 1.9727 ms.
I1005 05:58:47.598217  1233 caffe.cpp:412]        ip1    backward: 3.7615 ms.
I1005 05:58:47.598220  1233 caffe.cpp:409]      relu1    forward: 0.0241 ms.
I1005 05:58:47.598224  1233 caffe.cpp:412]      relu1    backward: 0.0282 ms.
I1005 05:58:47.598227  1233 caffe.cpp:409]        ip2    forward: 0.1135 ms.
I1005 05:58:47.598230  1233 caffe.cpp:412]        ip2    backward: 0.1435 ms.
I1005 05:58:47.598239  1233 caffe.cpp:409]       loss    forward: 0.0371 ms.
I1005 05:58:47.598243  1233 caffe.cpp:412]       loss    backward: 0.0018 ms.
I1005 05:58:47.598250  1233 caffe.cpp:417] Average Forward pass: 22.5693 ms.
I1005 05:58:47.598254  1233 caffe.cpp:419] Average Backward pass: 31.0536 ms.
I1005 05:58:47.598258  1233 caffe.cpp:421] Average Forward-Backward: 53.6 ms.
I1005 05:58:47.598263  1233 caffe.cpp:423] Total Time: 536 ms.
I1005 05:58:47.598264  1233 caffe.cpp:424] *** Benchmark ends ***
```

Caffe - Python - API

```python
# Import required Python libraries
%matplotlib inline
import os
import numpy as np
import matplotlib.pyplot as plt
import caffe
import random

# Choose network definition file and pretrained network binary
MODEL_FILE = '/home/ubuntu/caffe/examples/cifar10/cifar10_quick.prototxt'
PRETRAINED = '/home/ubuntu/caffe/examples/cifar10/cifar10_quick_iter_4000.caffemodel'

# Load a random image
x = caffe.io.load_image('/home/ubuntu/caffe/examples/images/' + str(random.randint(1,18)) + '.png')

# Display the chosen image
plt.imshow(x)
plt.axis('off')
plt.show()

# Load the pretrained model and select to use the GPU for computation
caffe.set_mode_gpu()
net = caffe.Classifier(MODEL_FILE, PRETRAINED,

mean=np.load('/home/ubuntu/caffe/caffe/examples/cifar10/cifar10_mean.npy').mean(1).mean(1),
                       raw_scale=255,
                       image_dims=(32, 32))

# Run the image through the pretrained network
prediction = net.predict([x])

# List of class labels
classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# Display the predicted probability for each class
plt.plot(prediction[0])
plt.xticks(range(0,10), classes, rotation=45)
# Display the most probable class
print classes[prediction[0].argmax()]
```

Theano

- **MNIST**
- **28X28 black and white images**
- **handwritten digits**
- **http://yann.lecun.com/exdb/mnist/**

Theano - Python - 補充

- http://deeplearning.net/tutorial/lenet.html#lenet
- https://github.com/lisa-lab/DeepLearningTutorials/blob/master/code/convolutional_mlp.py
- nvidia-docker
- nvidia-docker run -it nakosung/dockerfiles-1
- nvidia-docker run -it nvidia/cuda /bin/bash

```
wget https://repo.continuum.io/archive/Anaconda2-5.0.0.1-Linux-x86_64.sh
bash Anaconda2-5.0.0.1-Linux-x86_64.sh
conda install theano
git clone https://github.com/lisa-lab/DeepLearningTutorials.git
./DeepLearningTutorials/data/download.sh
cp DeepLearningTutorials/data/mnist-* DeepLearningTutorials/code/
cd DeepLearningTutorials/code/
python mlp.py
Using gpu device 0: GeForce GTX 1060 6GB (CNMeM is enabled with initial size: 10.0% of
memory, CuDNN 5105)
/usr/local/lib/python2.7/dist-packages/theano/sandbox/cuda/__init__.py:600:
UserWarning: Your CuDNN version is more recent then Theano. If you see problems, try
updating Theano or downgrading CuDNN to version 4.
warnings.warn(warn)
... loading data
... building the model
... training
epoch 1, minibatch 2500/2500, validation error 9.620000 %
   epoch 1, minibatch 2500/2500, test error of best model 10.090000 %
epoch 2, minibatch 2500/2500, validation error 8.610000 %
   epoch 2, minibatch 2500/2500, test error of best model 8.740000 %
epoch 3, minibatch 2500/2500, validation error 8.000000 %
   epoch 3, minibatch 2500/2500, test error of best model 8.160000 %
```

Torch

- **char-rnn** 程式與實作 **https://github.com/karpathy/char-rnn**
- 資料集 **http://cs.stanford.edu/people/karpathy/char-rnn/**
- 研究分析 **http://karpathy.github.io/2015/05/21/rnn-effectiveness/**

Torch 實作

- nvidia-docker
- nvidia-docker run -it kaixhin/cuda-torch:8.0

```
cd
git clone https://github.com/karpathy/char-rnn.git
th train.lua
th sample.lua cv/lm_lstm_epoch9.46_1.4349.t7
th convert_gpu_cpu_checkpoint.lua cv/lm_lstm_epoch30.00_1.3950.t7
```

- PyTorch https://github.com/nearai/pytorch-tools
- torch lenet https://github.com/NVIDIA/DIGITS/blob/master/digits/standard-networks/torch/lenet.lua