CNN in C++

Open AI Fab Inc.

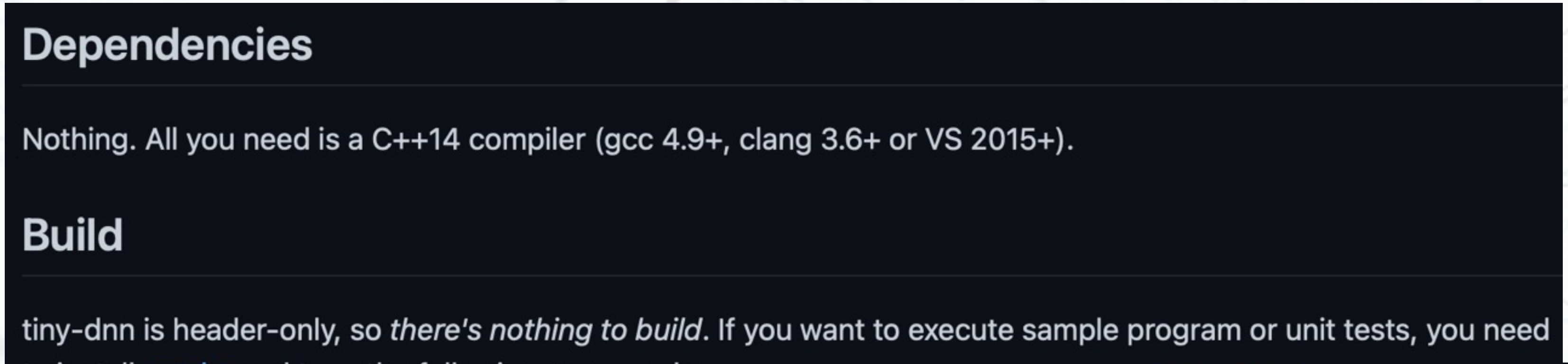
30 November 2021

Jeremy Lai

tiny-dnn/tiny-dnn

tiny-dnn/tiny-dnn

- Good documentation
- io is available (save / load model, reading images)



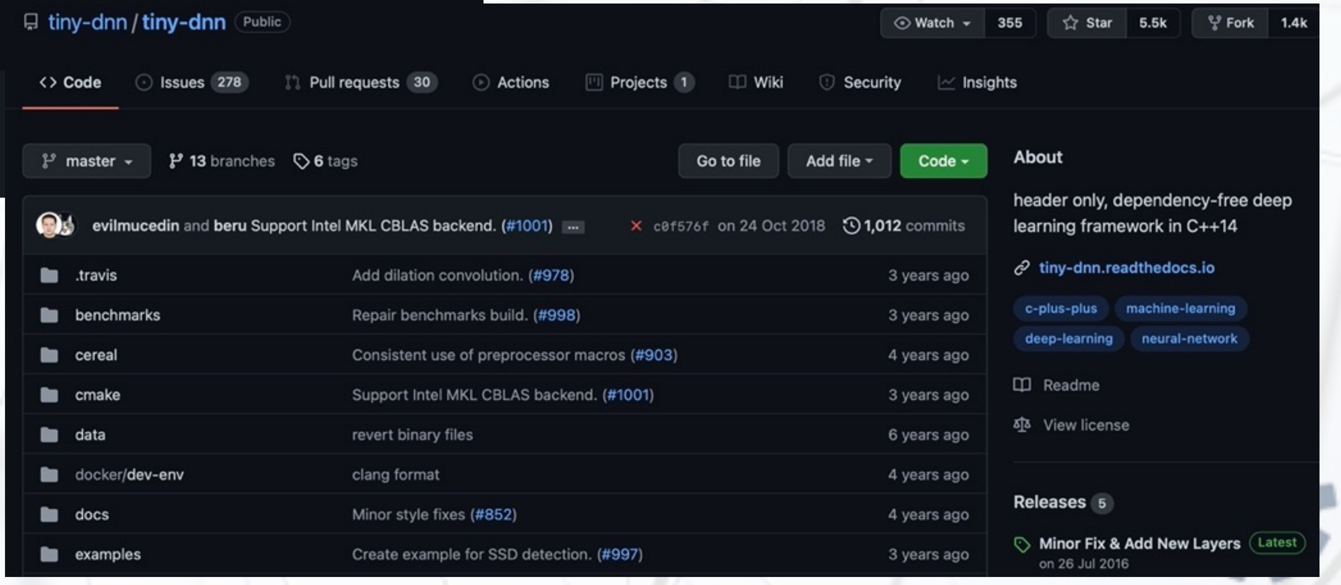
Dependencies

Nothing. All you need is a C++14 compiler (gcc 4.9+, clang 3.6+ or VS 2015+).

Build

tiny-dnn is header-only, so *there's nothing to build*. If you want to execute sample program or unit tests, you need to install cmake and type the following commands:

```
cmake . -DBUILD_EXAMPLES=ON
make
```



evilmucedin and beru Support Intel MKL CBLAS backend. (#1001) ... c0f576f on 24 Oct 2018 1,012 commits

.travis	Add dilation convolution. (#978)	3 years ago
benchmarks	Repair benchmarks build. (#998)	3 years ago
cereal	Consistent use of preprocessor macros (#903)	4 years ago
cmake	Support Intel MKL CBLAS backend. (#1001)	3 years ago
data	revert binary files	6 years ago
docker/dev-env	clang format	4 years ago
docs	Minor style fixes (#852)	4 years ago
examples	Create example for SSD detection. (#997)	3 years ago

About

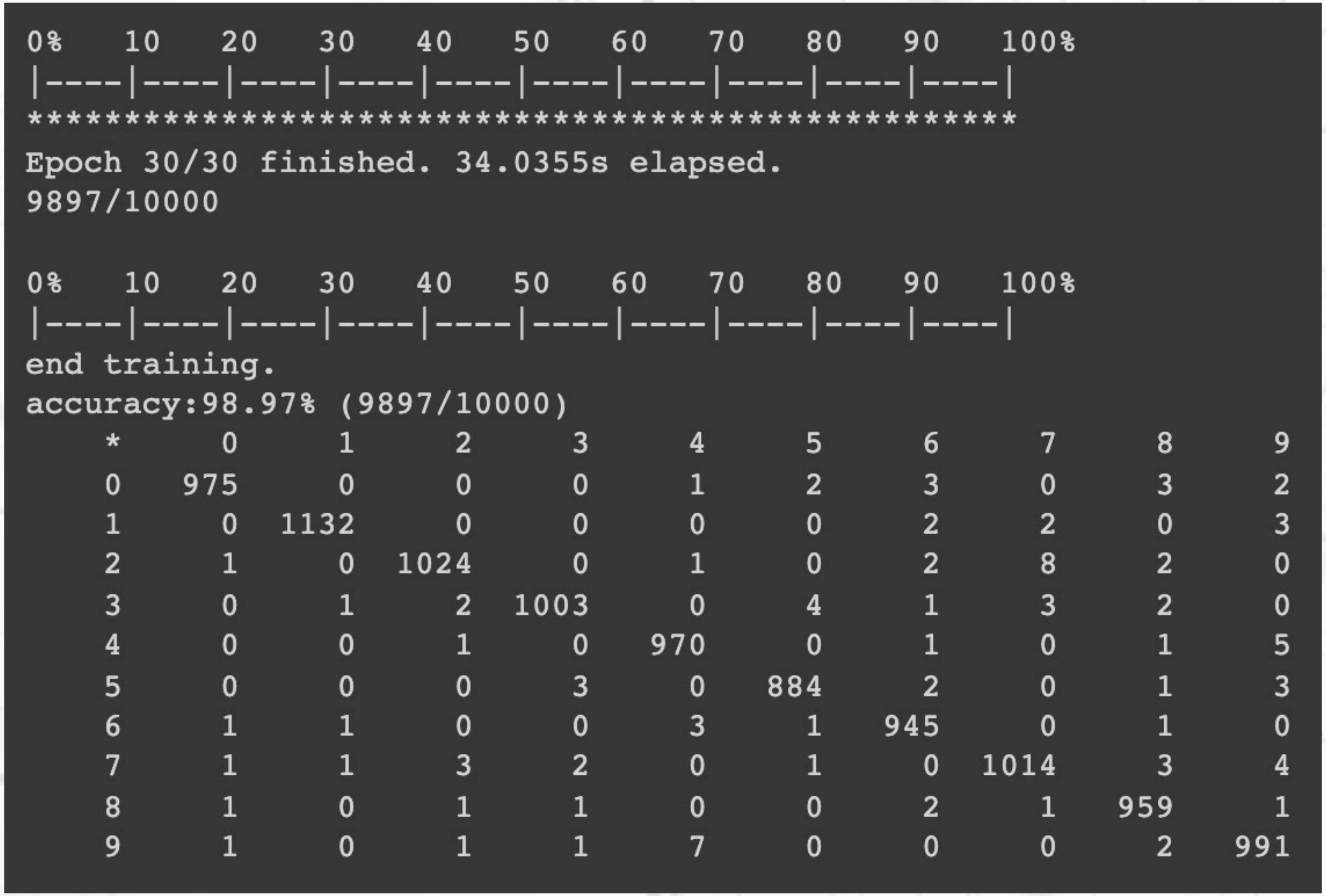
header only, dependency-free deep learning framework in C++14

tiny-dnn.readthedocs.io

Minor Fix & Add New Layers Latest
on 26 Jul 2016

Trial Run (mnist)

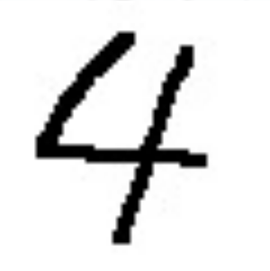
Training

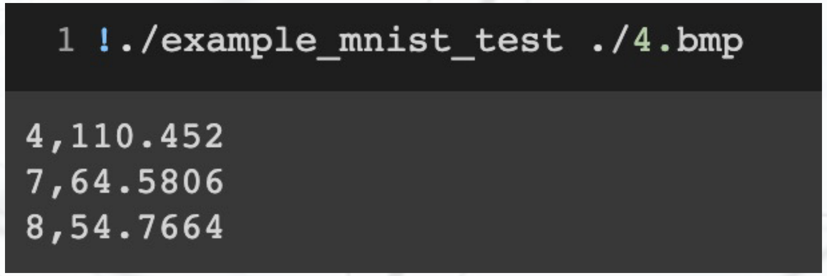


```
0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
***************************************************
Epoch 30/30 finished. 34.0355s elapsed.
9897/10000

0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
end training.
accuracy:98.97% (9897/10000)
    *     0     1     2     3     4     5     6     7     8     9
    0   975     0     0     0     1     2     3     0     3     2
    1     0  1132     0     0     0     0     2     2     0     3
    2     1     0  1024     0     1     0     2     8     2     0
    3     0     1     2  1003     0     4     1     3     2     0
    4     0     0     1     0   970     0     1     0     1     5
    5     0     0     0     3     0   884     2     0     1     3
    6     1     1     0     0     3     1   945     0     1     0
    7     1     1     3     2     0     1     0  1014     3     4
    8     1     0     1     1     0     0     2     1   959     1
    9     1     0     1     1     7     0     0     0     2   991
```

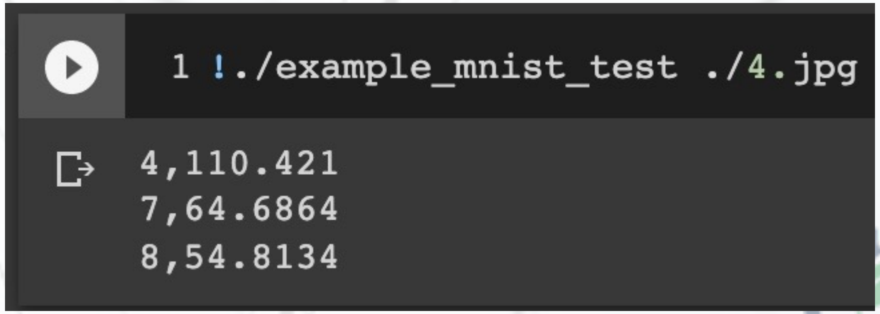
Predict





```
!./example_mnist_test ./4.bmp
```

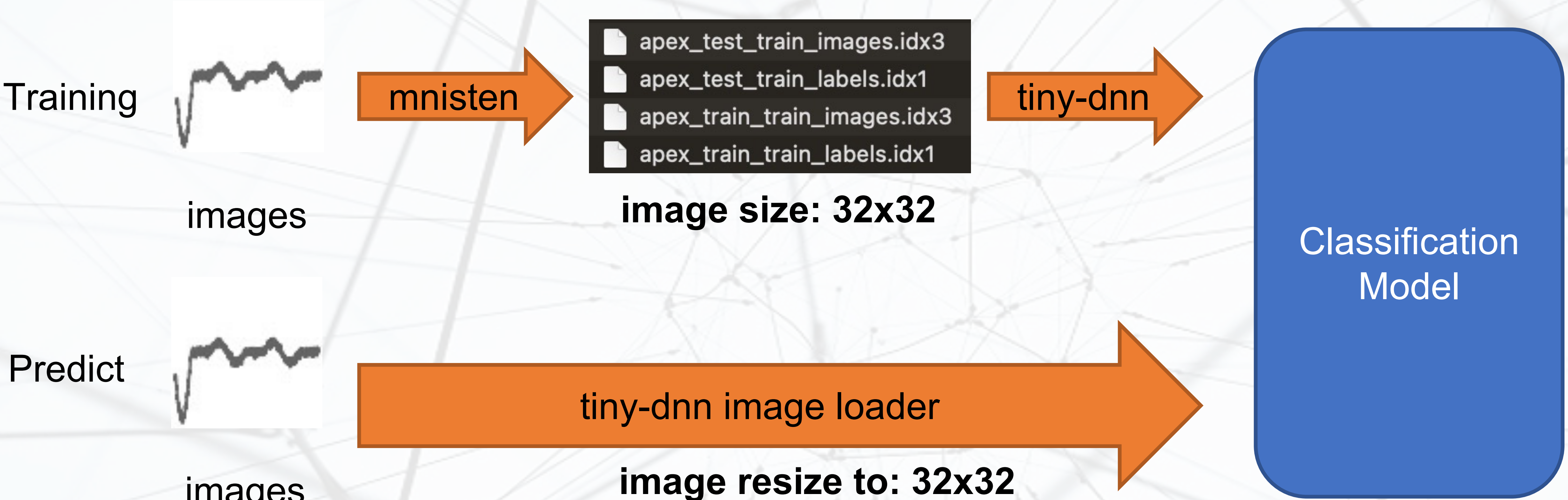
```
4,110.452
7,64.5806
8,54.7664
```



```
!./example_mnist_test ./4.jpg
```

```
4,110.421
7,64.6864
8,54.8134
```

Train and predict with own dataset



Training

images

mnisten

apex_test_train_images.idx3
apex_test_train_labels.idx1
apex_train_train_images.idx3
apex_train_train_labels.idx1

image size: 32x32

tiny-dnn

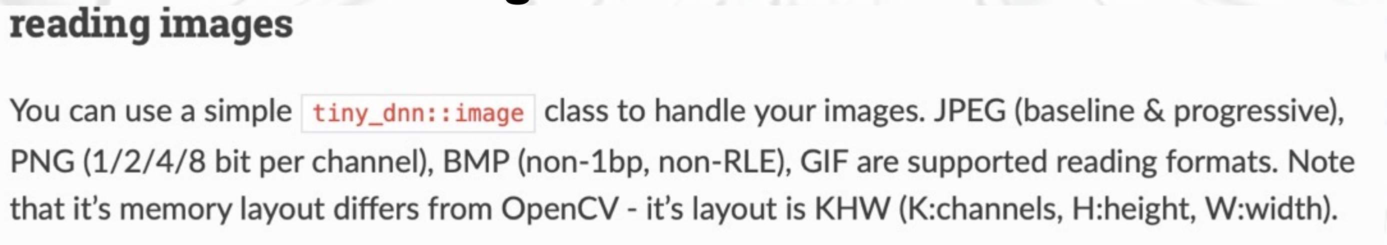
Classification Model

Predict

images

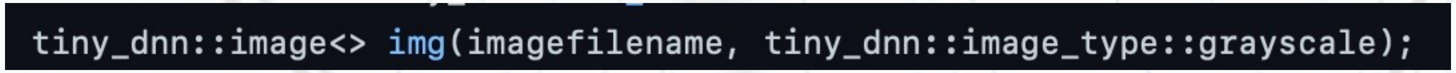
tiny-dnn image loader

image resize to: 32x32



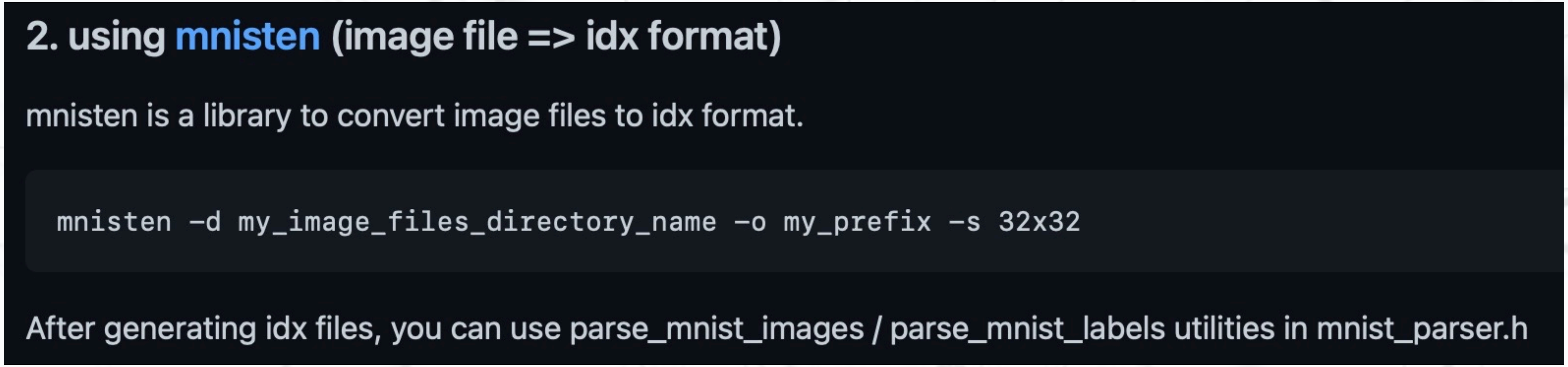
reading images

You can use a simple `tiny_dnn::image` class to handle your images. JPEG (baseline & progressive), PNG (1/2/4/8 bit per channel), BMP (non-1bp, non-RLE), GIF are supported reading formats. Note that it's memory layout differs from OpenCV - it's layout is KHW (K:channels, H:height, W:width).



```
tiny_dnn::image<> img(imagefilename, tiny_dnn::image_type::grayscale);
```

Convert image files to idx format binaries

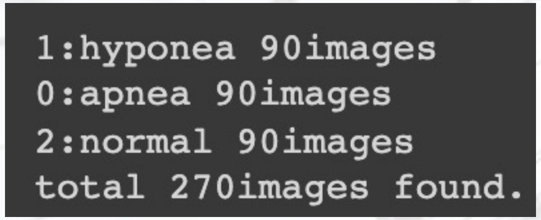


2. using mnisten (image file => idx format)

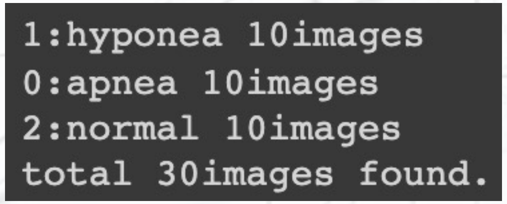
mnisten is a library to convert image files to idx format.

```
mnisten -d my_image_files_directory_name -o my_prefix -s 32x32
```

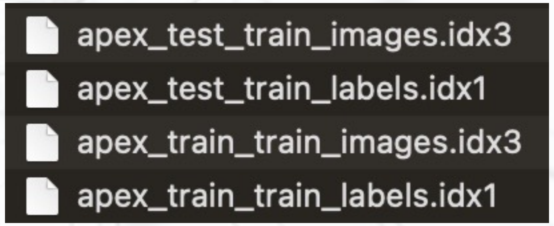
After generating idx files, you can use parse_mnist_images / parse_mnist_labels utilities in mnist_parser.h



```
1:hyponea 90images
0:apnea 90images
2:normal 90images
total 270images found.
```



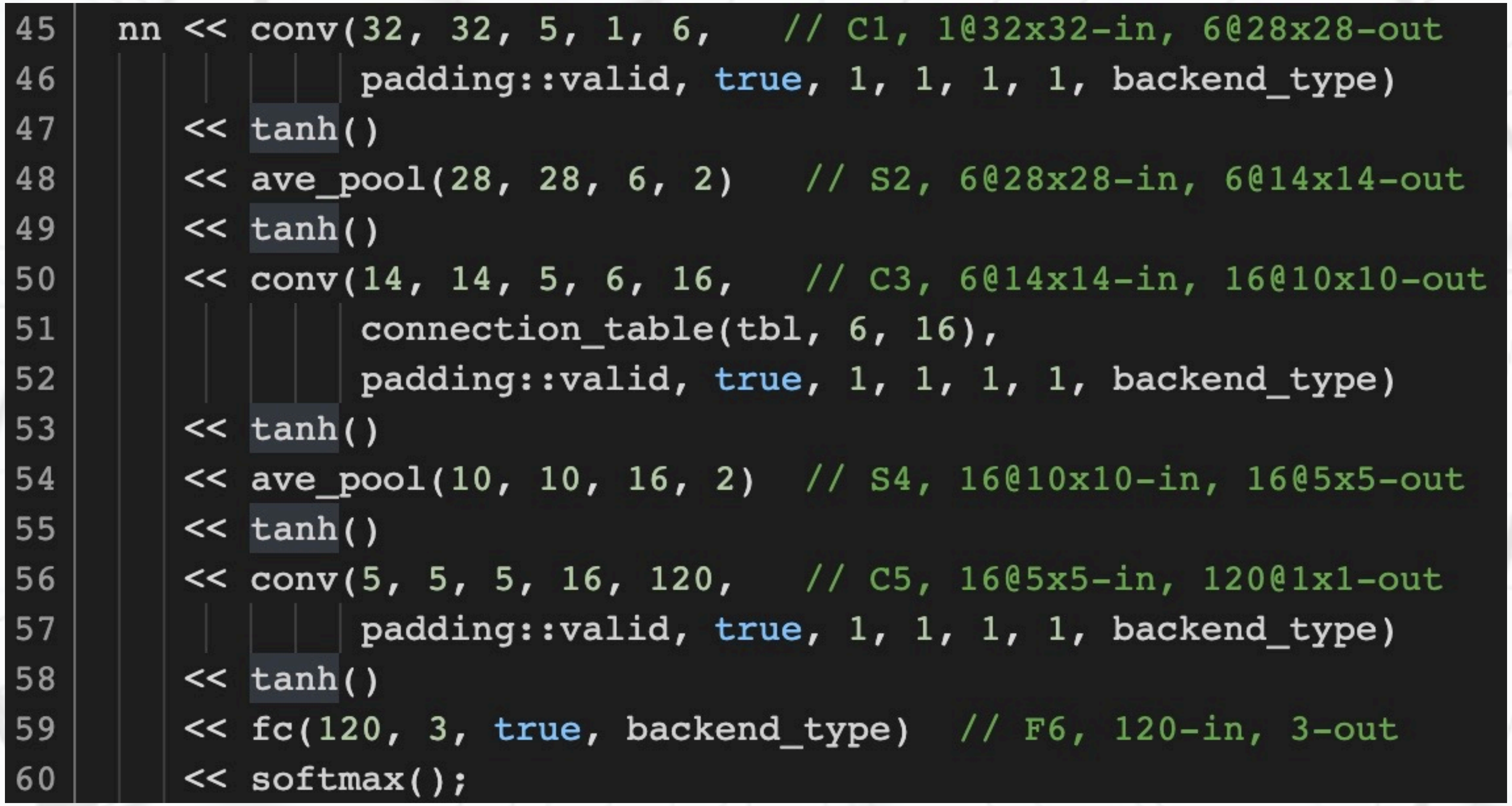
```
1:hyponea 10images
0:apnea 10images
2:normal 10images
total 30images found.
```



```
apex_test_train_images.idx3
apex_test_train_labels.idx1
apex_train_train_images.idx3
apex_train_train_labels.idx1
```

https://github.com/tiny-dnn/tiny-dnn/issues/141
https://github.com/tiny-dnn/tiny-dnn/wiki/Data-Format
https://github.com/nyanp/mnisten

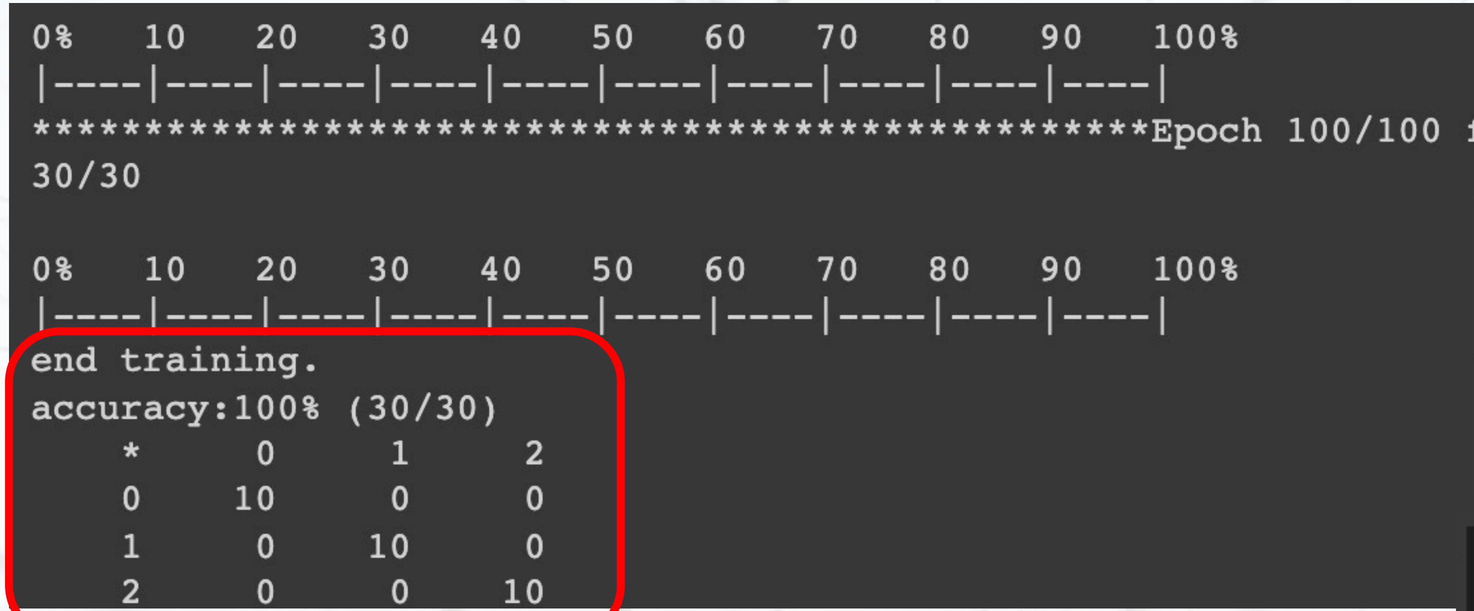
Train and predict with own dataset (LeNet)



```
nn << conv(32, 32, 5, 1, 6,    // C1, 1@32x32-in, 6@28x28-out
           padding::valid, true, 1, 1, 1, 1, backend_type)
   << tanh()
   << ave_pool(28, 28, 6, 2)   // S2, 6@28x28-in, 6@14x14-out
   << tanh()
   << conv(14, 14, 5, 6, 16,   // C3, 6@14x14-in, 16@10x10-out
           connection_table(tbl, 6, 16),
           padding::valid, true, 1, 1, 1, 1, backend_type)
   << tanh()
   << ave_pool(10, 10, 16, 2)  // S4, 16@10x10-in, 16@5x5-out
   << tanh()
   << conv(5, 5, 5, 16, 120,   // C5, 16@5x5-in, 120@1x1-out
           padding::valid, true, 1, 1, 1, 1, backend_type)
   << tanh()
   << fc(120, 3, true, backend_type)  // F6, 120-in, 3-out
   << softmax();
```

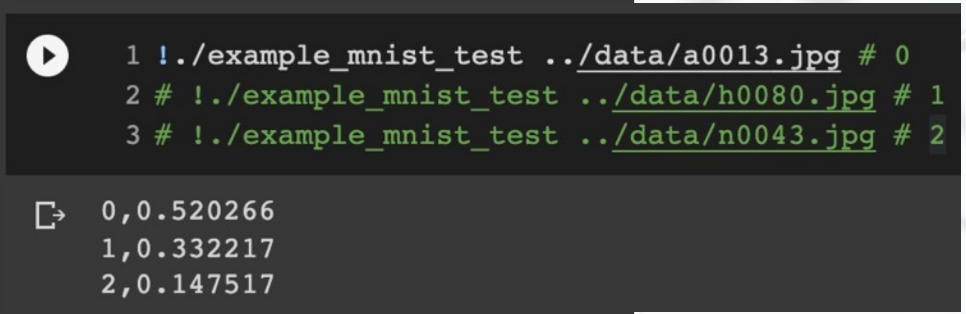
https://github.com/tiny-dnn/tiny-dnn/blob/master/examples/mnist/train.cpp
https://github.com/tiny-dnn/tiny-dnn/blob/master/examples/mnist/test.cpp

Train and predict with own dataset (LeNet)



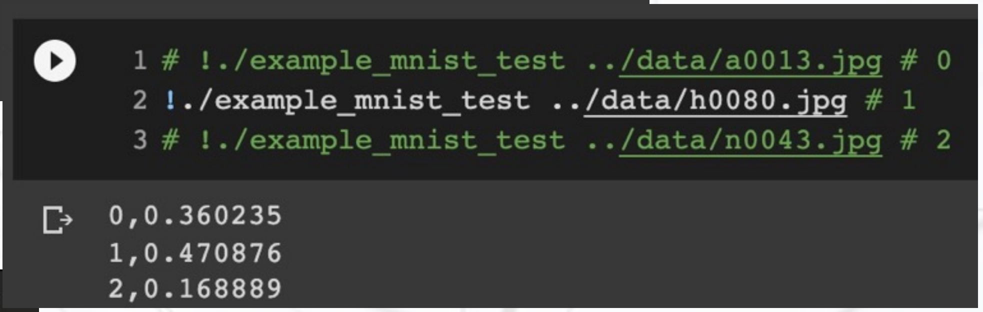
```
0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
***************************************************Epoch 100/100 finished. 0.178834s elapsed.
30/30

0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
end training.
accuracy:100% (30/30)
    *     0     1     2
    0    10     0     0
    1     0    10     0
    2     0     0    10
```



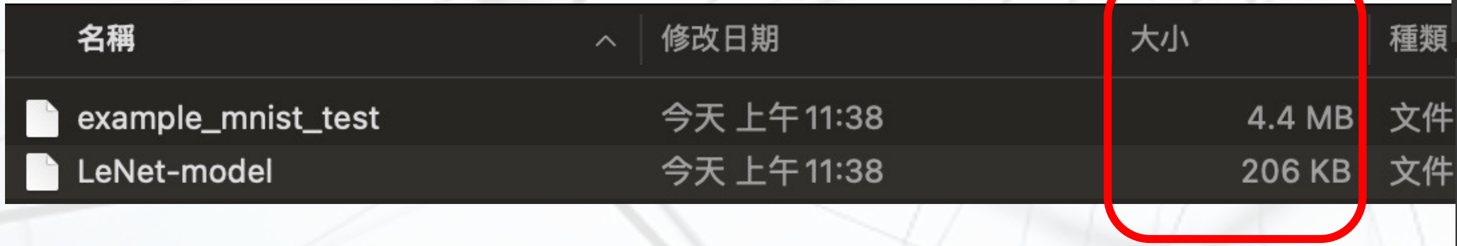
```
!./example_mnist_test ../data/a0013.jpg # 0
# !./example_mnist_test ../data/h0080.jpg # 1
# !./example_mnist_test ../data/n0043.jpg # 2
```

```
0,0.520266
1,0.332217
2,0.147517
```

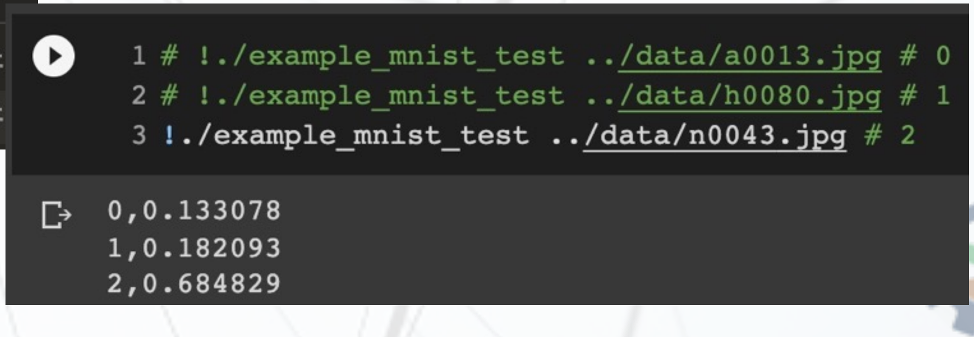


```
# !./example_mnist_test ../data/a0013.jpg # 0
!./example_mnist_test ../data/h0080.jpg # 1
# !./example_mnist_test ../data/n0043.jpg # 2
```

```
0,0.360235
1,0.470876
2,0.168889
```



名稱	修改日期	大小	種類
example_mnist_test	今天 上午11:38	4.4 MB	文件
LeNet-model	今天 上午11:38	206 KB	文件

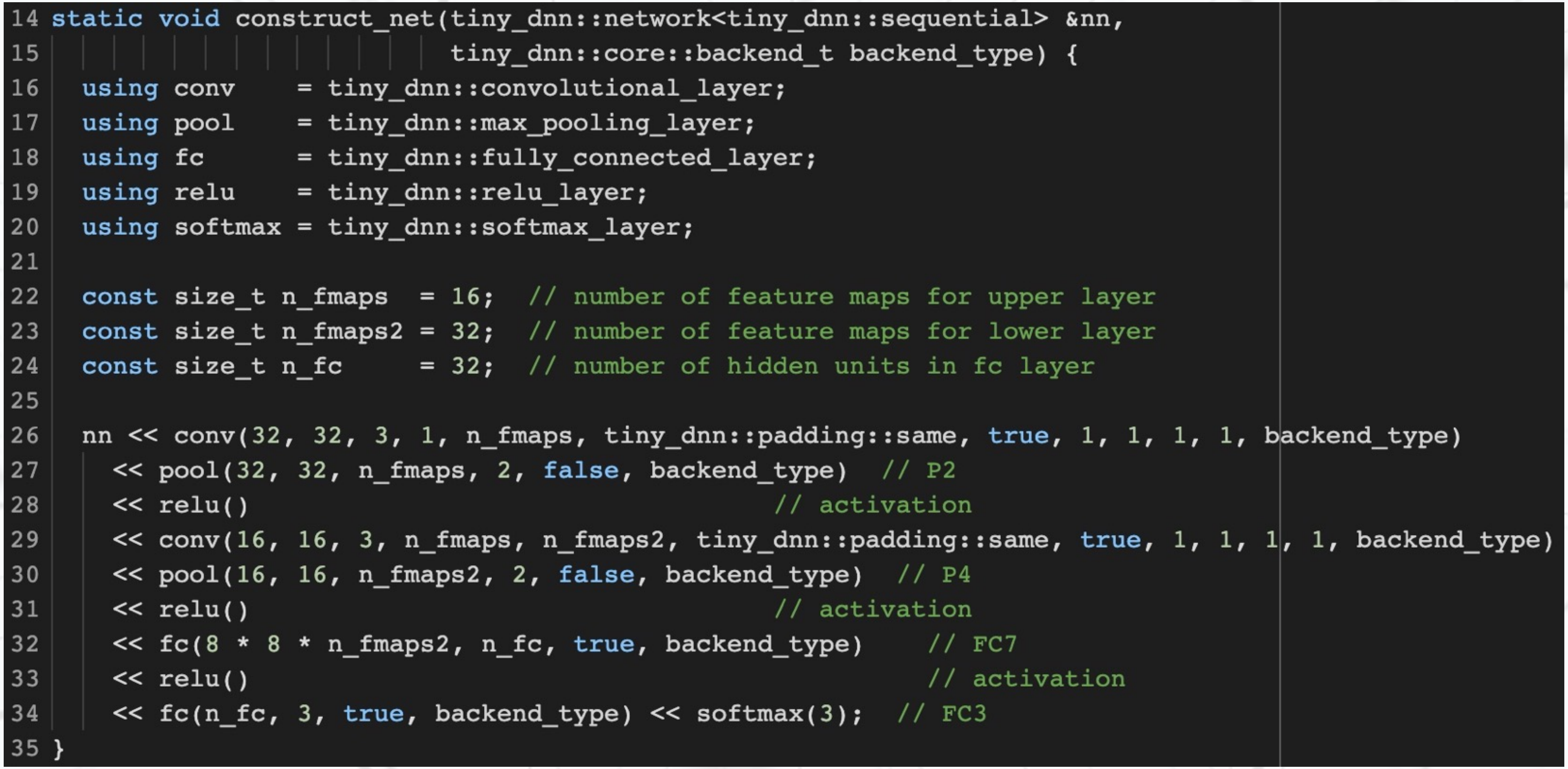


```
# !./example_mnist_test ../data/a0013.jpg # 0
# !./example_mnist_test ../data/h0080.jpg # 1
!./example_mnist_test ../data/n0043.jpg # 2
```

```
0,0.133078
1,0.182093
2,0.684829
```

https://github.com/tiny-dnn/tiny-dnn/blob/master/examples/mnist/train.cpp
https://github.com/tiny-dnn/tiny-dnn/blob/master/examples/mnist/test.cpp

Train and predict with own dataset (CNN)



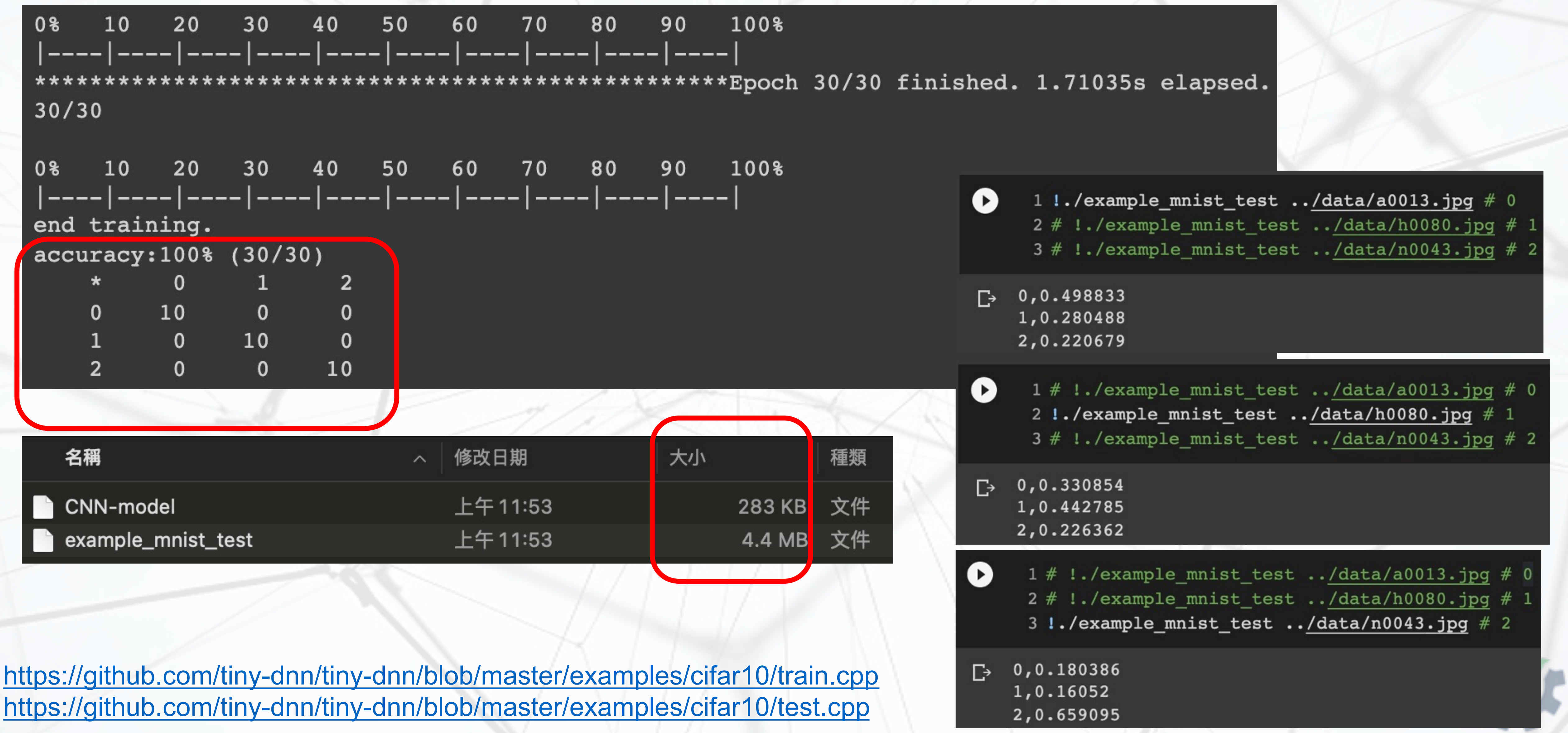
```
static void construct_net(tiny_dnn::network<tiny_dnn::sequential> &nn,
                          tiny_dnn::core::backend_t backend_type) {
  using conv    = tiny_dnn::convolutional_layer;
  using pool    = tiny_dnn::max_pooling_layer;
  using fc      = tiny_dnn::fully_connected_layer;
  using relu    = tiny_dnn::relu_layer;
  using softmax = tiny_dnn::softmax_layer;

  const size_t n_fmaps  = 16;  // number of feature maps for upper layer
  const size_t n_fmaps2 = 32;  // number of feature maps for lower layer
  const size_t n_fc     = 32;  // number of hidden units in fc layer

  nn << conv(32, 32, 3, 1, n_fmaps, tiny_dnn::padding::same, true, 1, 1, 1, 1, backend_type)
    << pool(32, 32, n_fmaps, 2, false, backend_type)  // P2
    << relu()                                  // activation
    << conv(16, 16, 3, n_fmaps, n_fmaps2, tiny_dnn::padding::same, true, 1, 1, 1, 1, backend_type)
    << pool(16, 16, n_fmaps2, 2, false, backend_type)  // P4
    << relu()                                  // activation
    << fc(8 * 8 * n_fmaps2, n_fc, true, backend_type)    // FC7
    << relu()                                            // activation
    << fc(n_fc, 3, true, backend_type) << softmax(3);  // FC3
}
```

https://github.com/tiny-dnn/tiny-dnn/blob/master/examples/cifar10/train.cpp
https://github.com/tiny-dnn/tiny-dnn/blob/master/examples/cifar10/test.cpp

Train and predict with own dataset (CNN)



```
0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
***************************************************Epoch 30/30 finished. 1.71035s elapsed.
30/30

0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
end training.
accuracy:100% (30/30)
    *     0     1     2
    0    10     0     0
    1     0    10     0
    2     0     0    10
```

名稱	修改日期	大小	種類
CNN-model	上午 11:53	283 KB	文件
example_mnist_test	上午 11:53	4.4 MB	文件

```
!./example_mnist_test ../data/a0013.jpg # 0
# !./example_mnist_test ../data/h0080.jpg # 1
# !./example_mnist_test ../data/n0043.jpg # 2
```

```
0,0.498833
1,0.280488
2,0.220679
```

```
# !./example_mnist_test ../data/a0013.jpg # 0
!./example_mnist_test ../data/h0080.jpg # 1
# !./example_mnist_test ../data/n0043.jpg # 2
```

```
0,0.330854
1,0.442785
2,0.226362
```

```
# !./example_mnist_test ../data/a0013.jpg # 0
# !./example_mnist_test ../data/h0080.jpg # 1
!./example_mnist_test ../data/n0043.jpg # 2
```

```
0,0.180386
1,0.16052
2,0.659095
```

https://github.com/tiny-dnn/tiny-dnn/blob/master/examples/cifar10/train.cpp
https://github.com/tiny-dnn/tiny-dnn/blob/master/examples/cifar10/test.cpp