Learn to use CNN

—Homework5 for computer vision

Name: Wang Jun

Student ID：3170100186

Date: 2021-12-24

Course: Computer Vision

Instructor: Mingli Song

Chapter 1: The purpose and requirements of the experiment

Framework: TensorFlow (includes the following network structure and data set)
Data set: The Mnist Database of handwritten digits
Network structure: LeNet-5

1. Specific task: Use the above data set, network structure and selected TensorFlow framework to realize handwritten digit recognition:
(1) Obtain the minist data set;
(2) Data enhancement;
(3) Construct a simple LeNet5 structure network;
(4) Training and testing the correctness of handwritten digit recognition results;
(5) Add dropout and batch standardization retraining.
2. Submit report (personal realization process + result)

Chapter 2: Experimental content and principle

OpenCV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

Lenet-5

Lenet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition. They used this architecture for recognizing the handwritten and machine-printed characters.

The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification.

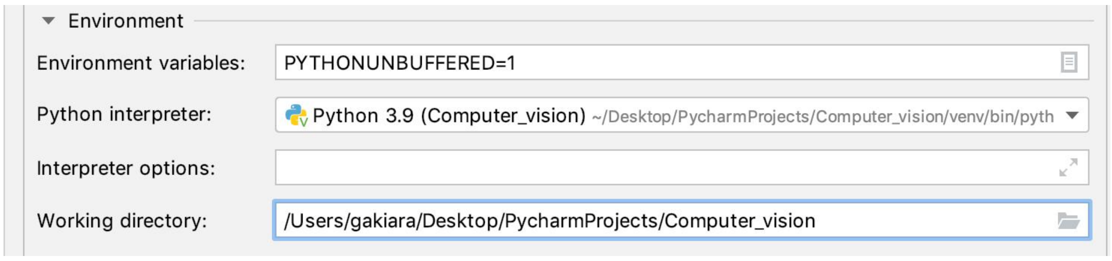
Chapter 3: Experimental procedure and analysis

Install opencv-python

Install opencv-python for the next process

```
(venv) gakiaradeMacBook-Pro:Computer_vision gakiara$ pip install opencv-python
Collecting opencv-python
  Downloading opencv-python-4.5.4.60.tar.gz (89.8 MB)
     |███                              | 8.6 MB 47 kB/s eta 0:28:32
```

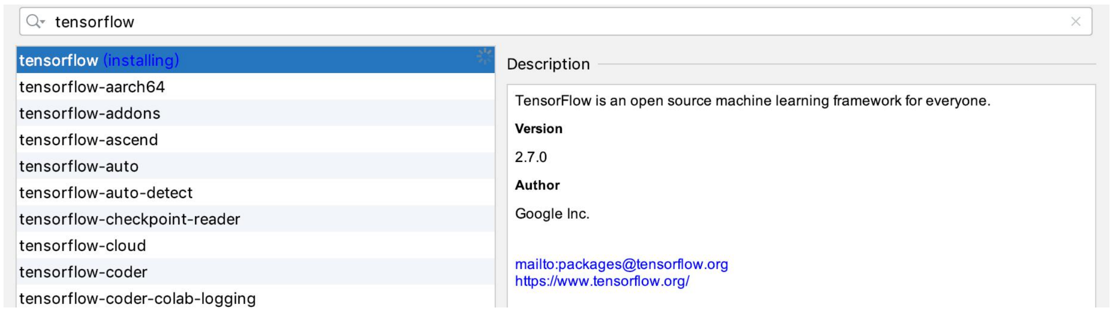
Set the working directory:



▼ Environment

Environment variables:	PYTHONUNBUFFERED=1
Python interpreter:	Python 3.9 (Computer_vision) ~/Desktop/PycharmProjects/Computer_vision/venv/bin/pyth
Interpreter options:	
Working directory:	/Users/gakiara/Desktop/PycharmProjects/Computer_vision

Install tensorflow in pycharm



tensorflow

tensorflow (installing)
tensorflow-aarch64
tensorflow-addons
tensorflow-ascend
tensorflow-auto
tensorflow-auto-detect
tensorflow-checkpoint-reader
tensorflow-cloud
tensorflow-coder
tensorflow-coder-colab-logging

Description

TensorFlow is an open source machine learning framework for everyone.

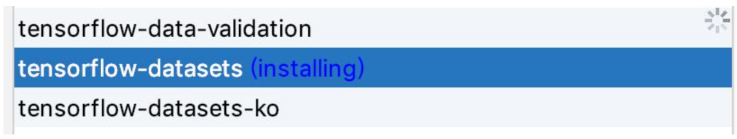
Version

2.7.0

Author

Google Inc.

mailto:packages@tensorflow.org
https://www.tensorflow.org/



tensorflow-data-validation
tensorflow-datasets (installing)
tensorflow-datasets-ko

Import and configure

Import packages:

```python
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
import matplotlib as mpl
import tensorflow as tf
from tensorflow.keras.callbacks import TensorBoard, ModelCheckpoint, Callback, LearningRateScheduler, ReduceLROnPlateau
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout, Activation, Conv2D, MaxPool2D, AveragePooling2D, BatchNormalization
from tensorflow.keras.models import Model, load_model, save_model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.metrics import SparseCategoricalAccuracy
import tensorflow_datasets as tfds

import coloredlogs
import logging
```

Get the minist data set

```python
from sklearn.datasets import fetch_openml

mnist = fetch_openml("mnist_784")

x = mnist["data"]
y = mnist["target"]
```

In x, there are 60000 handwritten digital images (image size is 28*28), and in y are the corresponding correct results marked.

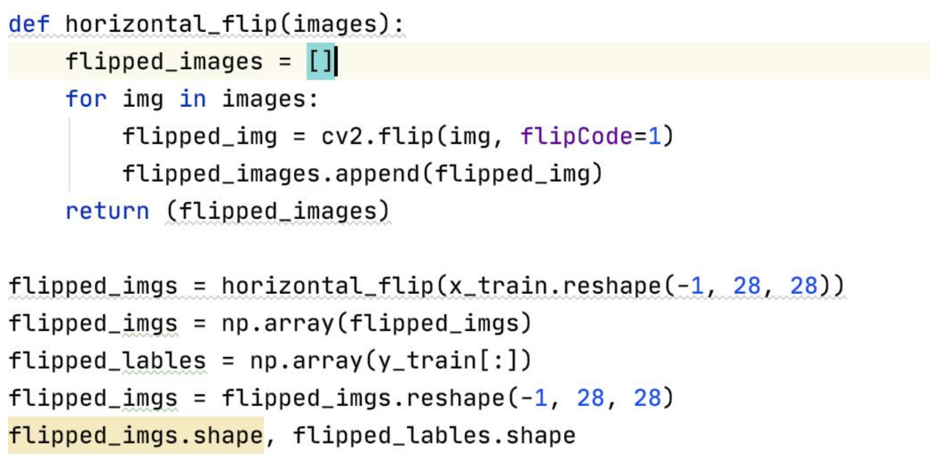
Data augmentation

Here I have used two methods to expand the data set. One is to move the image up, down, left, and right by one pixel, so that the data set is enlarged by 4 times; the other is to flip the image horizontally.

image & label:

```python
x_train_shifted = []
y_train_augmented = []

for dx, dy in ((1, 0), (-1, 0), (0, 1), (0, -1)):
    for image, label in zip(x_train, y_train):
        x_train_shifted.append(shift_image(image, dx, dy))
        y_train_augmented.append(label)

x_train_shifted = np.array(x_train_shifted)
y_train_augmented = np.array(y_train_augmented)
```

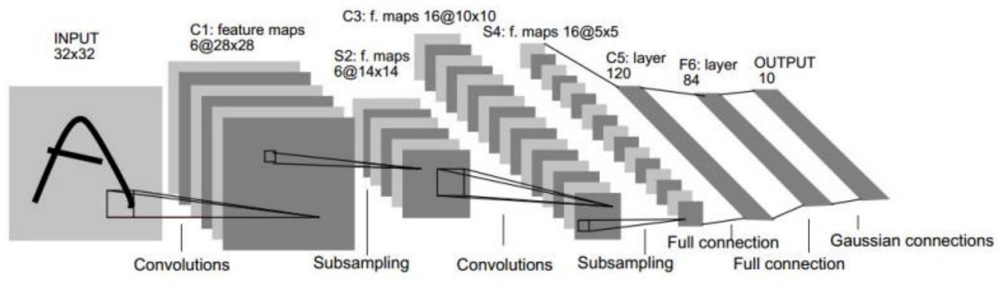
get the flipped img to enlarge the dataset:



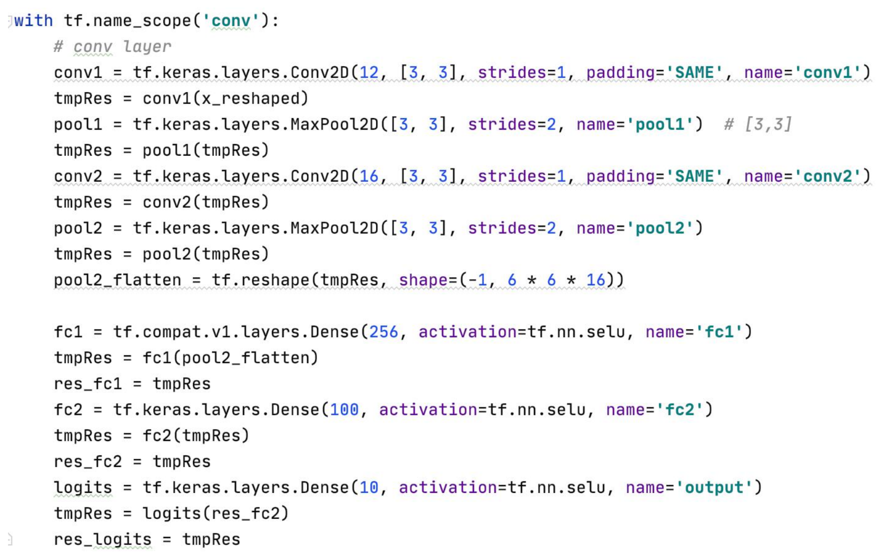
```python
def horizontal_flip(images):
    flipped_images = []
    for img in images:
        flipped_img = cv2.flip(img, flipCode=1)
        flipped_images.append(flipped_img)
    return (flipped_images)

flipped_imgs = horizontal_flip(x_train.reshape(-1, 28, 28))
flipped_imgs = np.array(flipped_imgs)
flipped_lables = np.array(y_train[:])
flipped_imgs = flipped_imgs.reshape(-1, 28, 28)
flipped_imgs.shape, flipped_lables.shape
```

Build a simple LeNet5 structure network

Since the input image size is 28*28, and the input image size in the example structure is 32*32, the parameters are adjusted based on the LeNet-5 structure.



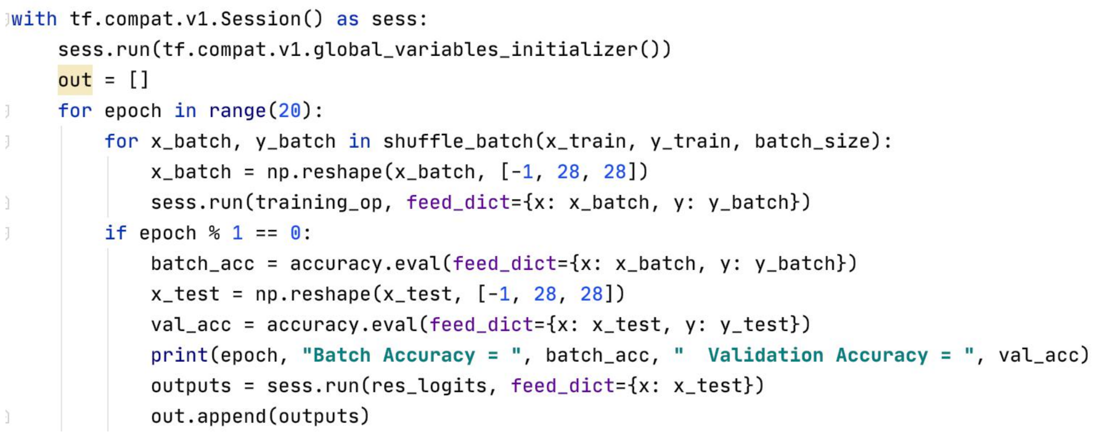
LeNet-5 has 7 layers in total, without input, each layer contains trainable parameters; each layer has multiple Feature Maps, and each FeatureMap uses a convolution filter to extract a feature of the input:



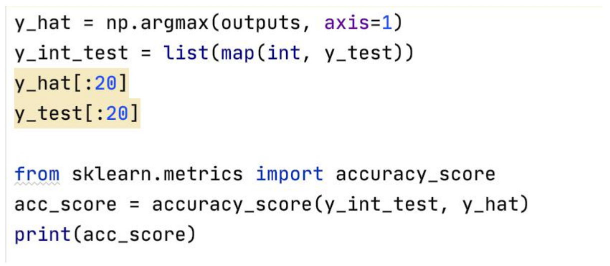
```python
with tf.name_scope('conv'):
    # conv layer
    conv1 = tf.keras.layers.Conv2D(12, [3, 3], strides=1, padding='SAME', name='conv1')
    tmpRes = conv1(x_reshaped)
    pool1 = tf.keras.layers.MaxPool2D([3, 3], strides=2, name='pool1')  # [3,3]
    tmpRes = pool1(tmpRes)
    conv2 = tf.keras.layers.Conv2D(16, [3, 3], strides=1, padding='SAME', name='conv2')
    tmpRes = conv2(tmpRes)
    pool2 = tf.keras.layers.MaxPool2D([3, 3], strides=2, name='pool2')
    tmpRes = pool2(tmpRes)
    pool2_flatten = tf.reshape(tmpRes, shape=(-1, 6 * 6 * 16))

    fc1 = tf.compat.v1.layers.Dense(256, activation=tf.nn.selu, name='fc1')
    tmpRes = fc1(pool2_flatten)
    res_fc1 = tmpRes
    fc2 = tf.keras.layers.Dense(100, activation=tf.nn.selu, name='fc2')
    tmpRes = fc2(tmpRes)
    res_fc2 = tmpRes
    logits = tf.keras.layers.Dense(10, activation=tf.nn.selu, name='output')
    tmpRes = logits(res_fc2)
    res_logits = tmpRes
```

Train and check

Train and check the correctness of handwritten digit recognition results and then start training and testing after creating the loss function.



```
with tf.compat.v1.Session() as sess:
    sess.run(tf.compat.v1.global_variables_initializer())
    out = []
    for epoch in range(20):
        for x_batch, y_batch in shuffle_batch(x_train, y_train, batch_size):
            x_batch = np.reshape(x_batch, [-1, 28, 28])
            sess.run(training_op, feed_dict={x: x_batch, y: y_batch})
        if epoch % 1 == 0:
            batch_acc = accuracy.eval(feed_dict={x: x_batch, y: y_batch})
            x_test = np.reshape(x_test, [-1, 28, 28])
            val_acc = accuracy.eval(feed_dict={x: x_test, y: y_test})
            print(epoch, "Batch Accuracy = ", batch_acc, "  Validation Accuracy = ", val_acc)
            outputs = sess.run(res_logits, feed_dict={x: x_test})
            out.append(outputs)
```

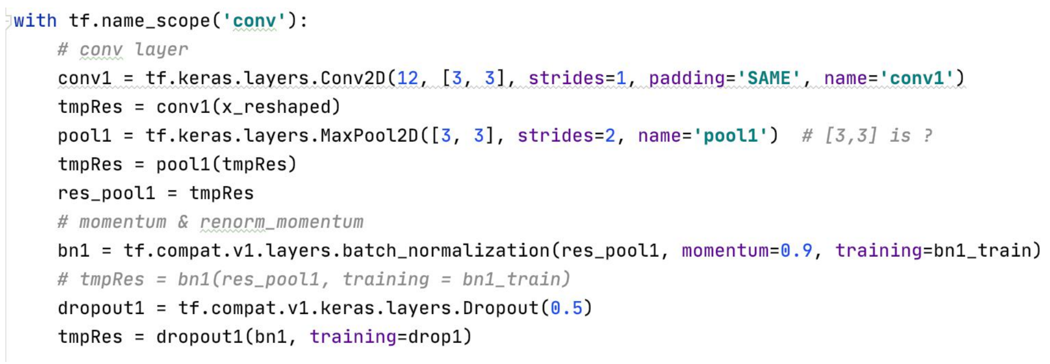


```
y_hat = np.argmax(outputs, axis=1)
y_int_test = list(map(int, y_test))
y_hat[:20]
y_test[:20]

from sklearn.metrics import accuracy_score
acc_score = accuracy_score(y_int_test, y_hat)
print(acc_score)
```

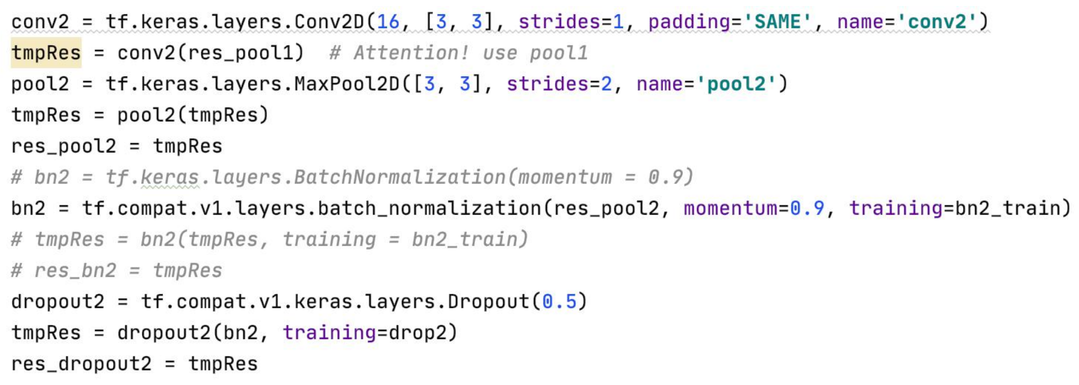
Add dropout and batch normalization retraining

Now add two new elements to the network: dropout and batch normalization, and start training with the enhanced data set:

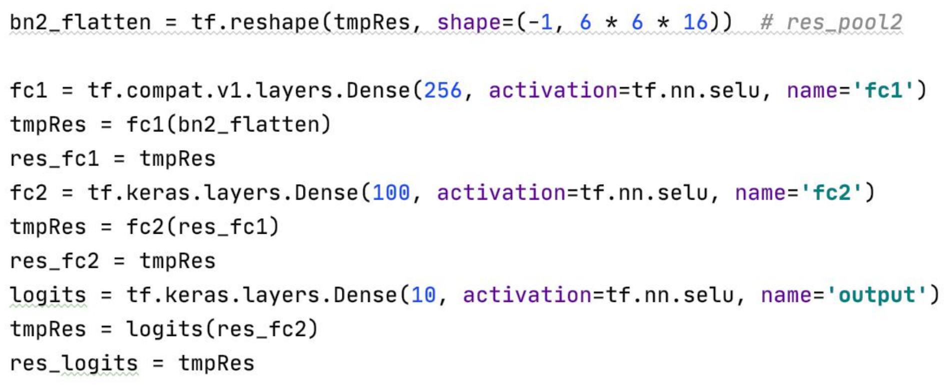
new with shifted & flipped:



```
with tf.name_scope('conv'):
    # conv layer
    conv1 = tf.keras.layers.Conv2D(12, [3, 3], strides=1, padding='SAME', name='conv1')
    tmpRes = conv1(x_reshaped)
    pool1 = tf.keras.layers.MaxPool2D([3, 3], strides=2, name='pool1')  # [3,3] is ?
    tmpRes = pool1(tmpRes)
    res_pool1 = tmpRes
    # momentum & renorm_momentum
    bn1 = tf.compat.v1.layers.batch_normalization(res_pool1, momentum=0.9, training=bn1_train)
    # tmpRes = bn1(res_pool1, training = bn1_train)
    dropout1 = tf.compat.v1.keras.layers.Dropout(0.5)
    tmpRes = dropout1(bn1, training=drop1)
```

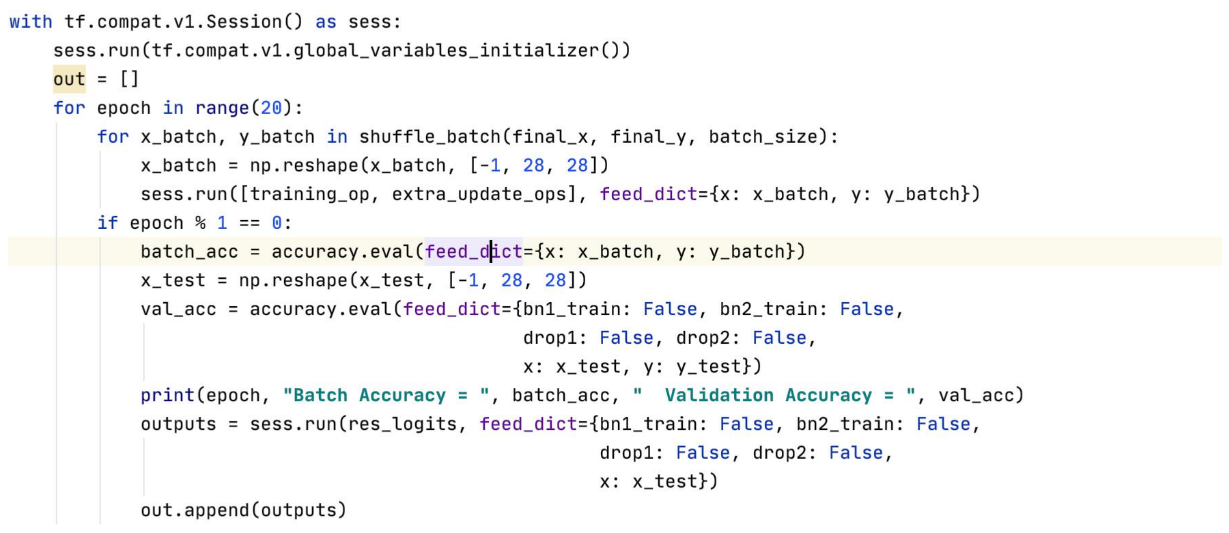


```
conv2 = tf.keras.layers.Conv2D(16, [3, 3], strides=1, padding='SAME', name='conv2')
tmpRes = conv2(res_pool1)  # Attention! use pool1
pool2 = tf.keras.layers.MaxPool2D([3, 3], strides=2, name='pool2')
tmpRes = pool2(tmpRes)
res_pool2 = tmpRes
# bn2 = tf.keras.layers.BatchNormalization(momentum = 0.9)
bn2 = tf.compat.v1.layers.batch_normalization(res_pool2, momentum=0.9, training=bn2_train)
# tmpRes = bn2(tmpRes, training = bn2_train)
# res_bn2 = tmpRes
dropout2 = tf.compat.v1.keras.layers.Dropout(0.5)
tmpRes = dropout2(bn2, training=drop2)
res_dropout2 = tmpRes
```



```
bn2_flatten = tf.reshape(tmpRes, shape=(-1, 6 * 6 * 16))  # res_pool2

fc1 = tf.compat.v1.layers.Dense(256, activation=tf.nn.selu, name='fc1')
tmpRes = fc1(bn2_flatten)
res_fc1 = tmpRes
fc2 = tf.keras.layers.Dense(100, activation=tf.nn.selu, name='fc2')
tmpRes = fc2(res_fc1)
res_fc2 = tmpRes
logits = tf.keras.layers.Dense(10, activation=tf.nn.selu, name='output')
tmpRes = logits(res_fc2)
res_logits = tmpRes
```

Train & Test:



```
with tf.compat.v1.Session() as sess:
    sess.run(tf.compat.v1.global_variables_initializer())
    out = []
    for epoch in range(20):
        for x_batch, y_batch in shuffle_batch(final_x, final_y, batch_size):
            x_batch = np.reshape(x_batch, [-1, 28, 28])
            sess.run([training_op, extra_update_ops], feed_dict={x: x_batch, y: y_batch})
        if epoch % 1 == 0:
            batch_acc = accuracy.eval(feed_dict={x: x_batch, y: y_batch})
            x_test = np.reshape(x_test, [-1, 28, 28])
            val_acc = accuracy.eval(feed_dict={bn1_train: False, bn2_train: False,
                                               drop1: False, drop2: False,
                                               x: x_test, y: y_test})
            print(epoch, "Batch Accuracy = ", batch_acc, "  Validation Accuracy = ", val_acc)
            outputs = sess.run(res_logits, feed_dict={bn1_train: False, bn2_train: False,
                                                      drop1: False, drop2: False,
                                                      x: x_test})
            out.append(outputs)
```

Chapter 4: Experimental environment and operation method

Development environment

- macOS 10.14.6

- python 3.9

- opencv-python 4.5.1.48

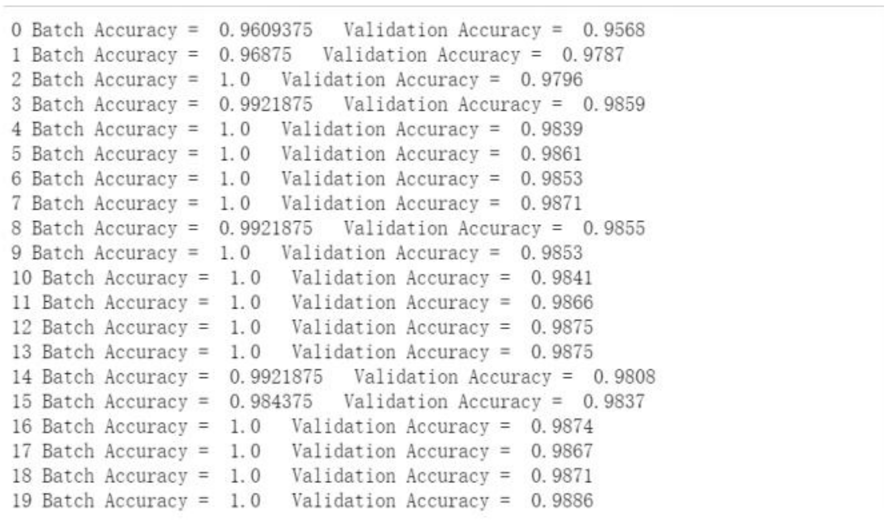
-tensorflow 2.7.0

Operation mode

- python hw5.py

Chapter 5: Experimental results

Results1



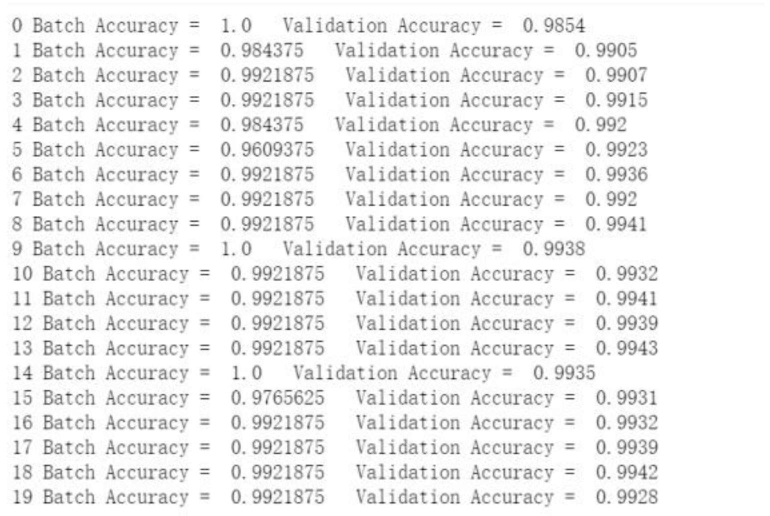
```
0 Batch Accuracy =  0.9609375   Validation Accuracy =  0.9568
1 Batch Accuracy =  0.96875   Validation Accuracy =  0.9787
2 Batch Accuracy =  1.0   Validation Accuracy =  0.9796
3 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9859
4 Batch Accuracy =  1.0   Validation Accuracy =  0.9839
5 Batch Accuracy =  1.0   Validation Accuracy =  0.9861
6 Batch Accuracy =  1.0   Validation Accuracy =  0.9853
7 Batch Accuracy =  1.0   Validation Accuracy =  0.9871
8 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9855
9 Batch Accuracy =  1.0   Validation Accuracy =  0.9853
10 Batch Accuracy =  1.0   Validation Accuracy =  0.9841
11 Batch Accuracy =  1.0   Validation Accuracy =  0.9866
12 Batch Accuracy =  1.0   Validation Accuracy =  0.9875
13 Batch Accuracy =  1.0   Validation Accuracy =  0.9875
14 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9808
15 Batch Accuracy =  0.984375   Validation Accuracy =  0.9837
16 Batch Accuracy =  1.0   Validation Accuracy =  0.9874
17 Batch Accuracy =  1.0   Validation Accuracy =  0.9867
18 Batch Accuracy =  1.0   Validation Accuracy =  0.9871
19 Batch Accuracy =  1.0   Validation Accuracy =  0.9886
```

accuracy_score:

```
0.9886
```

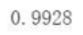
Results2 with dropout and batch standardization

Add dropout and batch standardization, use the enhanced data set, and train the process and results of 20 times. It can be found from the figure that the accuracy rate obtained by the 20th time training has exceeded 99%. The effect is better than the unimproved situation.



```
0 Batch Accuracy =  1.0   Validation Accuracy =  0.9854
1 Batch Accuracy =  0.984375   Validation Accuracy =  0.9905
2 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9907
3 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9915
4 Batch Accuracy =  0.984375   Validation Accuracy =  0.992
5 Batch Accuracy =  0.9609375   Validation Accuracy =  0.9923
6 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9936
7 Batch Accuracy =  0.9921875   Validation Accuracy =  0.992
8 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9941
9 Batch Accuracy =  1.0   Validation Accuracy =  0.9938
10 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9932
11 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9941
12 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9939
13 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9943
14 Batch Accuracy =  1.0   Validation Accuracy =  0.9935
15 Batch Accuracy =  0.9765625   Validation Accuracy =  0.9931
16 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9932
17 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9939
18 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9942
19 Batch Accuracy =  0.9921875   Validation Accuracy =  0.9928
```

accuracy_score:



```
0.9928
```

Chapter 6: Thoughts

CNN

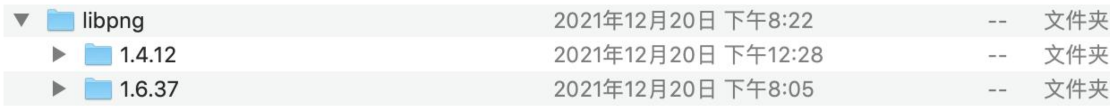
Through the result data of the experiment, we can realize that LeNet-5 is indeed a very efficient convolutional neural network for handwritten character recognition. Without data enhancement and other changes, only slightly changed parameters to adapt to the input After 20 times of training, a recognition success rate of close to 99% can be obtained. After adding dropout and batch standardization, it exceeds 99%.

After the introductory practice of CNN in this experiment, I have a deeper understanding of various concepts in CNN, such as convolutional layer, pooling layer and fully connected layer. It also consolidates the knowledge learned in class, such as input, output, meaning of different parameters, calculation of feature map size, and so on. In general, after this experiment, I have touched many areas of knowledge that I have never studied before, and I have also realized my inadequacy in related mathematics and have benefited a lot. At the same time, this experiment also inspired me to learn more about CNN, hoping to learn more about it in my future study and life.

A little bug

“libpng warning: Application built with libpng-1.4.12 but running with 1.6.37”

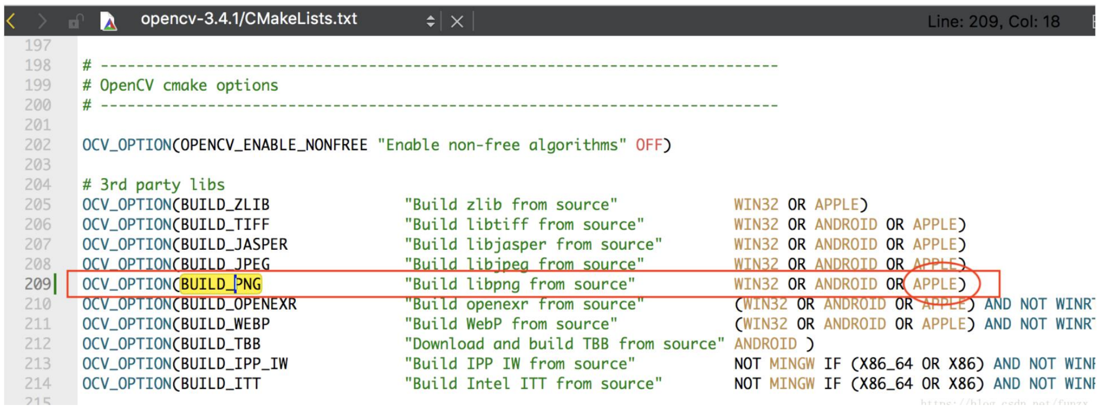
I have checked the system’s version: 1.6.37 and downloaded a matched version.



▼ libpng	2021年12月20日 下午8:22	--	文件夹
▶ 1.4.12	2021年12月20日 下午12:28	--	文件夹
▶ 1.6.37	2021年12月20日 下午8:05	--	文件夹

But at first, I have not found the way to link the old version in mac， and I don’t know why the pycharm would link to the libpng in the mac-os.

Finally I found the answer to the questions: why does OpenCV's own libpng built-in when compiling on mac, but it depends on the system's libpng library on centOS? ? It seems that the answer to all this is to be found in the cmake configuration file. Open the OpenCV/CMakeLists.txt file and immediately found the clue (as shown below):



opencv-3.4.1/CMakeLists.txt   Line: 209, Col: 18

```

# ----------------------------------------------------------------------------
# OpenCV cmake options
# ----------------------------------------------------------------------------

OCV_OPTION(OPENCV_ENABLE_NONFREE "Enable non-free algorithms" OFF)

# 3rd party libs
OCV_OPTION(BUILD_ZLIB               "Build zlib from source"             WIN32 OR APPLE)
OCV_OPTION(BUILD_TIFF               "Build libtiff from source"          WIN32 OR ANDROID OR APPLE)
OCV_OPTION(BUILD_JASPER             "Build libjasper from source"        WIN32 OR ANDROID OR APPLE)
OCV_OPTION(BUILD_JPEG               "Build libjpeg from source"          WIN32 OR ANDROID OR APPLE)
OCV_OPTION(BUILD_PNG                "Build libpng from source"           WIN32 OR ANDROID OR APPLE)
OCV_OPTION(BUILD_OPENEXR            "Build openexr from source"          (WIN32 OR ANDROID OR APPLE) AND NOT WINR
OCV_OPTION(BUILD_WEBP               "Build WebP from source"             (WIN32 OR ANDROID OR APPLE) AND NOT WINR
OCV_OPTION(BUILD_TBB                "Download and build TBB from source" ANDROID )
OCV_OPTION(BUILD_IPP_IW             "Build IPP IW from source"           NOT MINGW IF (X86_64 OR X86) AND NOT WIN
OCV_OPTION(BUILD_ITT                "Build Intel ITT from source"        NOT MINGW IF (X86_64 OR X86) AND NOT WIN
```

Did you see it? It is necessary to "build png" under Apple system! ! But "build png" is not used under centOS, it is naturally linked to the library of the system!

Now that we know that BUILD_PNG is used to control "compile built-in libpng" or "depend on system libpng", then we only need to set BUILD_PNG to ON when compiling OpenCV, then we can compile the libpng that comes with OpenCV forcibly, without being affected by libpng in the system. Impact. In this way, there will be no version mismatch problem, so the solution is:

Solution 1:
When compiling OpenCV, add parameters when executing cmake command: -D BUILD_PNG=ON
which is: cmake -D BUILD_PNG=ON -D CMAKE_BUILD_TYPE=DEBUG -D CMAKE_INSTALL_PREFIX=/usr/local/opencv-3.4.1…

Solution 2:
Replace the existing libpng library in the system with the libpng library of the same version as the libpng that comes with OpenCV. However, this method is not very elegant, and it is troublesome

to operate. You need to download or compile a new version of the libpng library yourself, so it is not recommended.