**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**

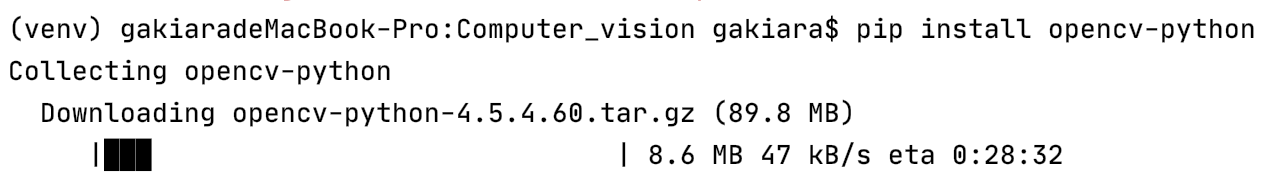
Lenet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper [Gradient-Based Learning Applied to Document Recognition](http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf" \t "/Users/gakiara/Documents\\x/_blank). They used this architecture 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.

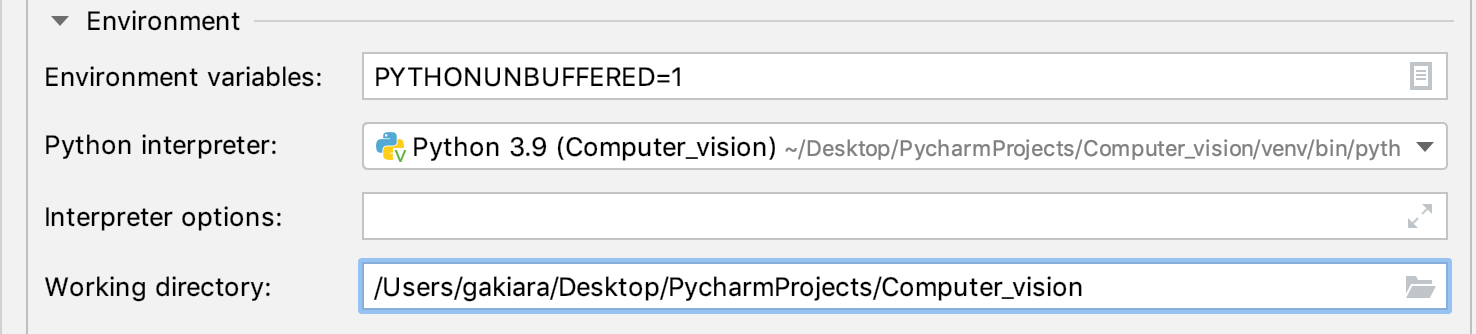
### Chapter 3: Experimental procedure and analysis

## **Install opencv-python**

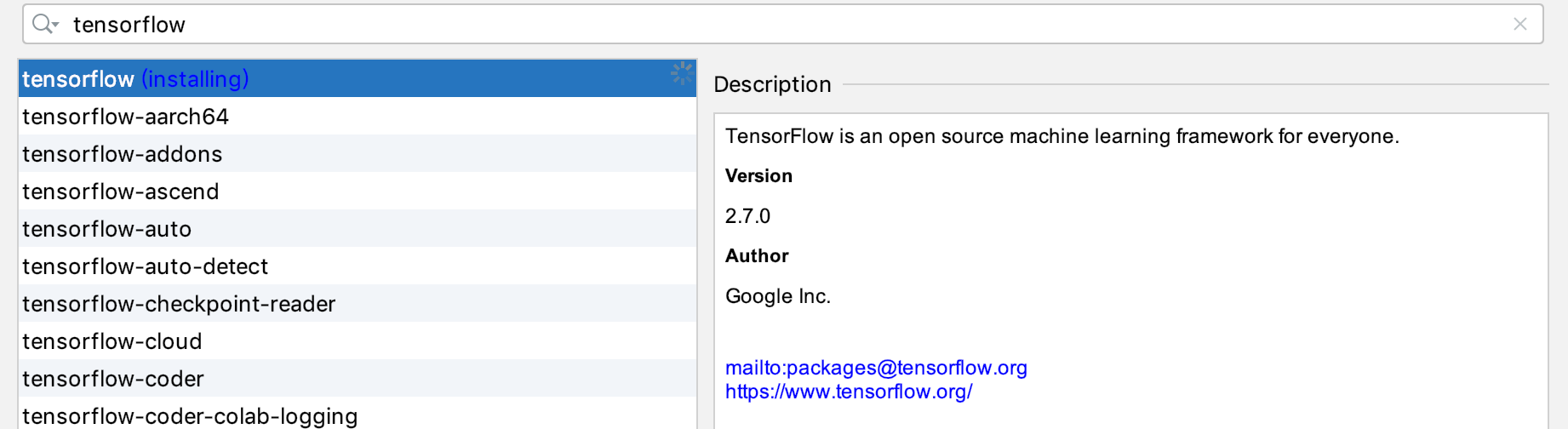
Install opencv-python for the next process

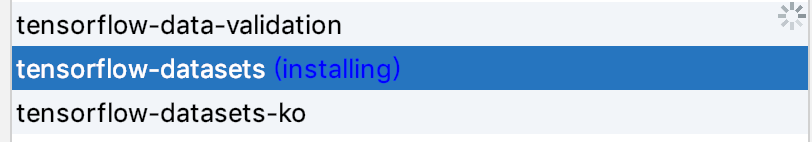


**Set the working directory：**



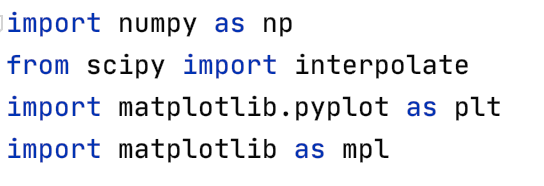
## **Install tensorflow in pycharm**

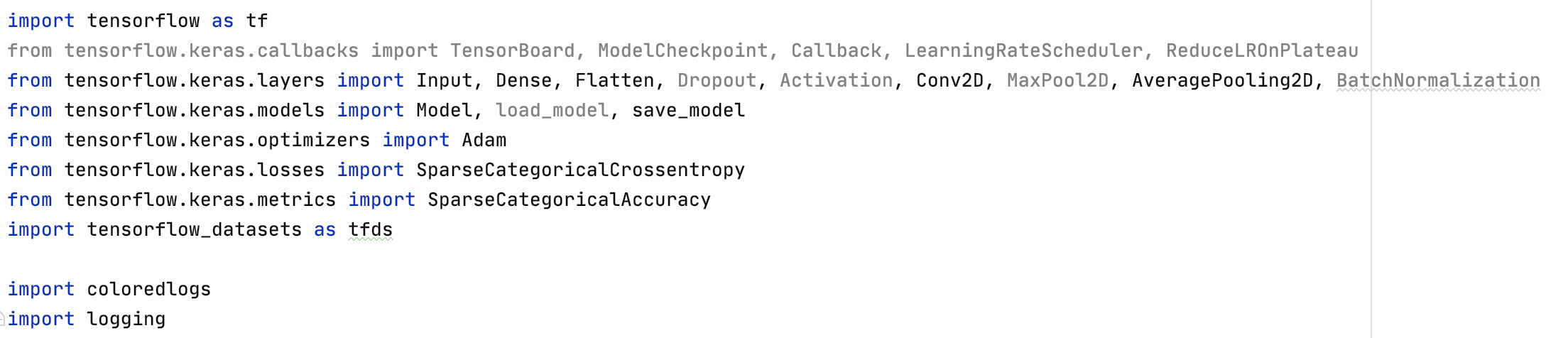




## **Import and configure**

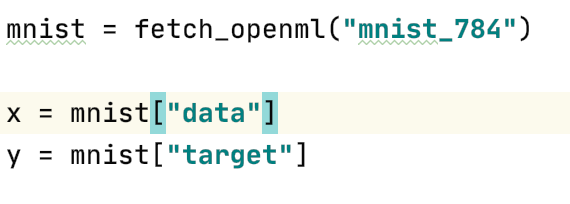
**Import packages:**





## **Get the minist data set**



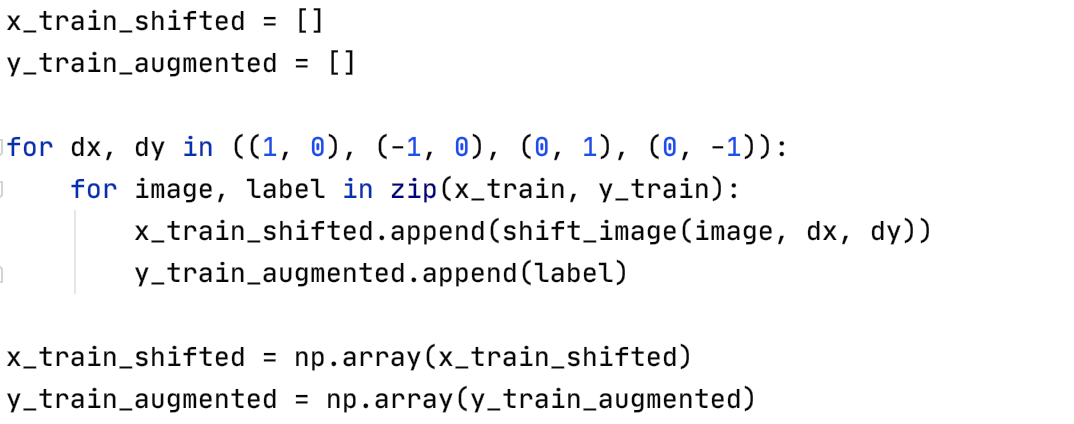


In x, there are 60000 handwritten digital images (image size is 28\*28), and in y are the corresponding correct results marked.

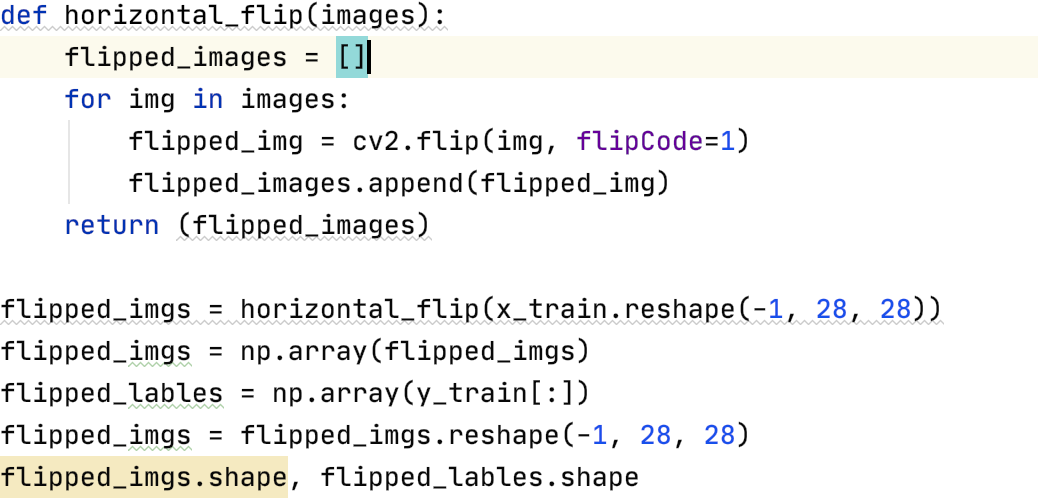
## **Data augmentation**

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:**

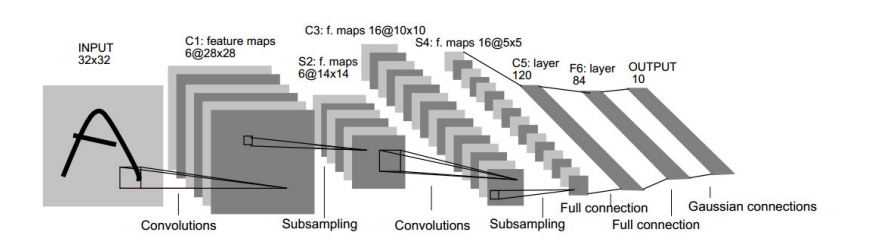


**get the flipped img to enlarge the dataset:**

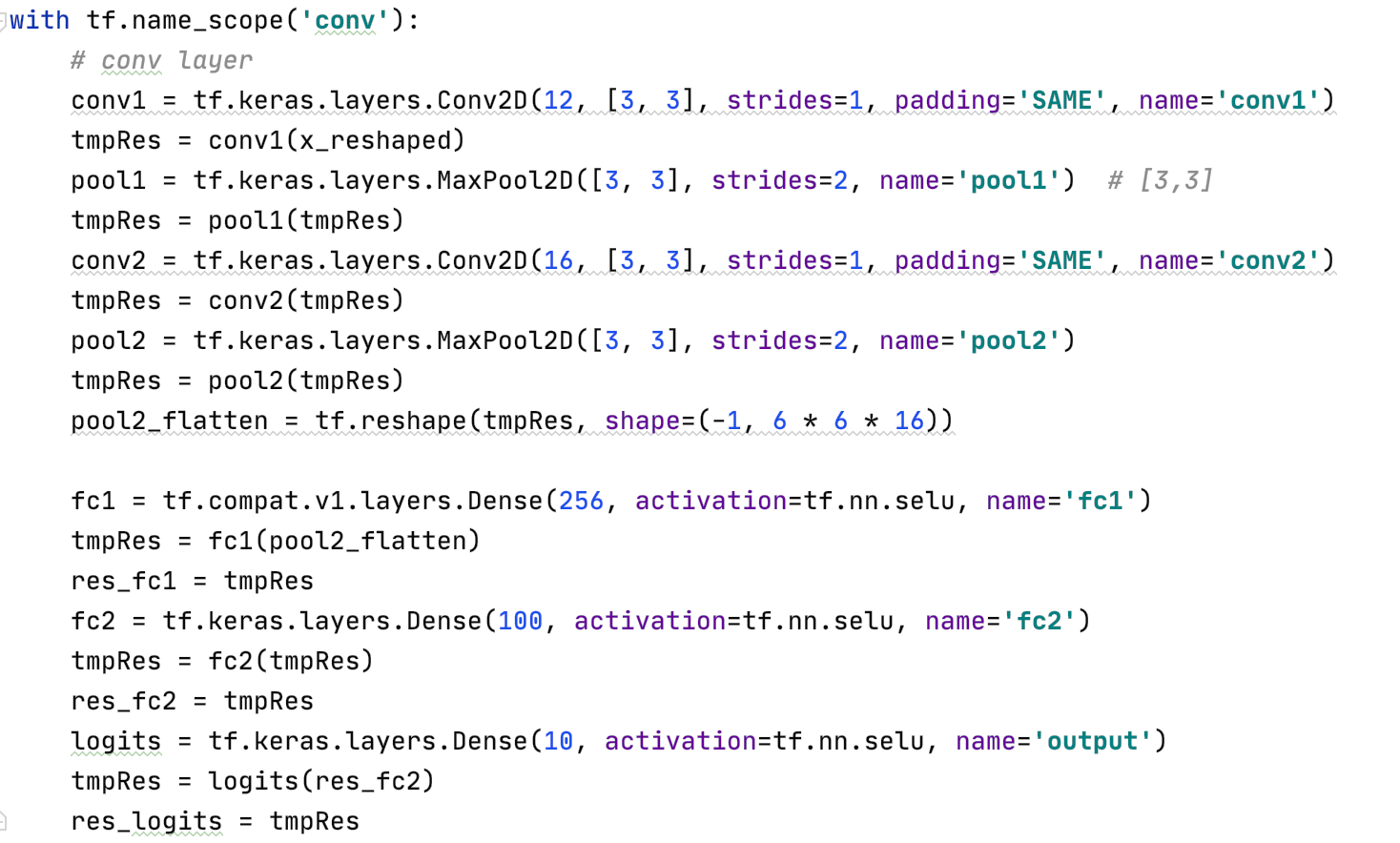


## **Build a simple LeNet5 structure network**

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.

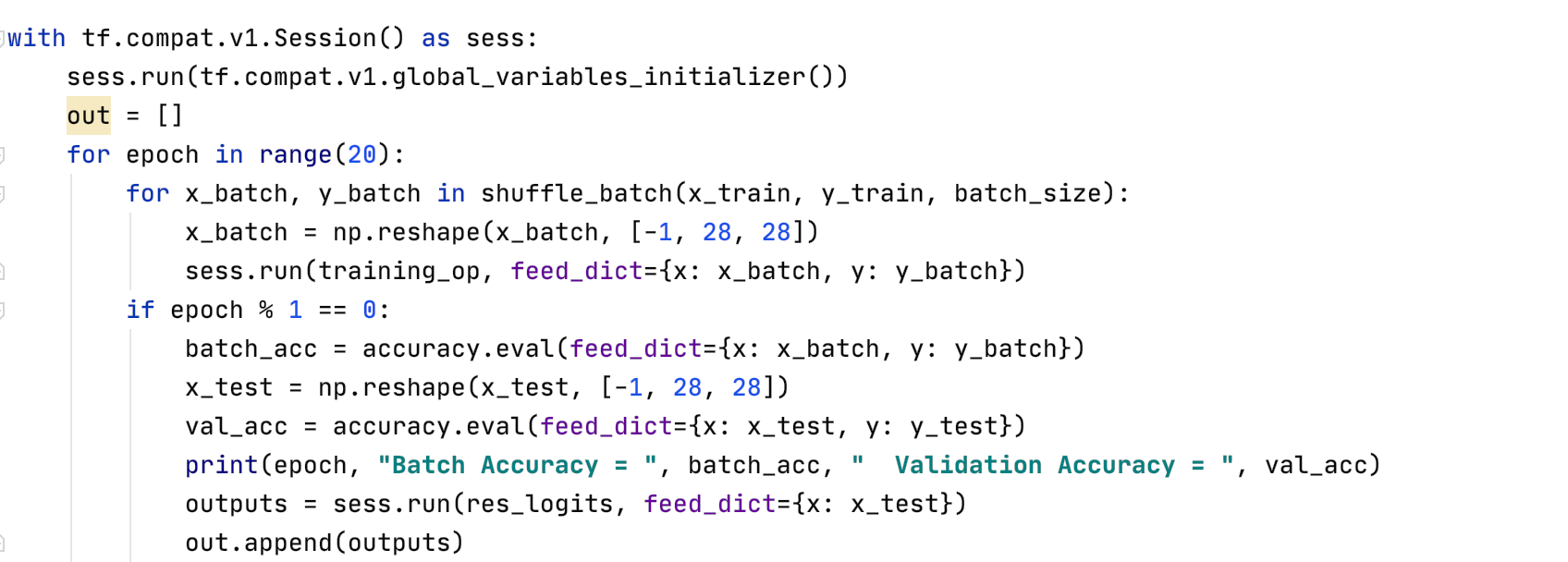


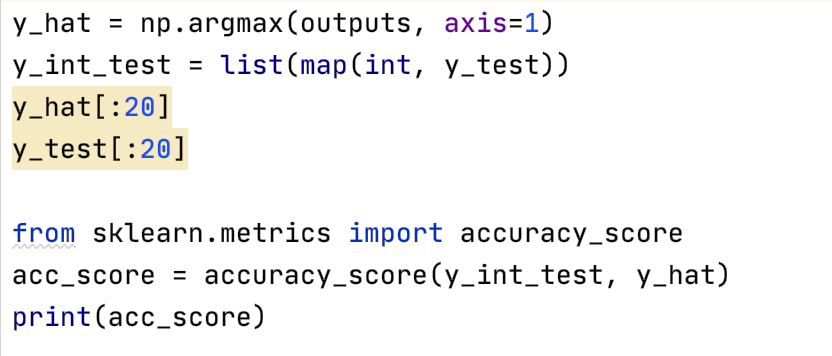
LeNet-5 has 7 layers in total, without input, each layer contains trainable parameters; each layer has multiple Feature Maps, and each FeatureMap uses a convolution filter to extract a feature of the input:



## **Train and check**

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

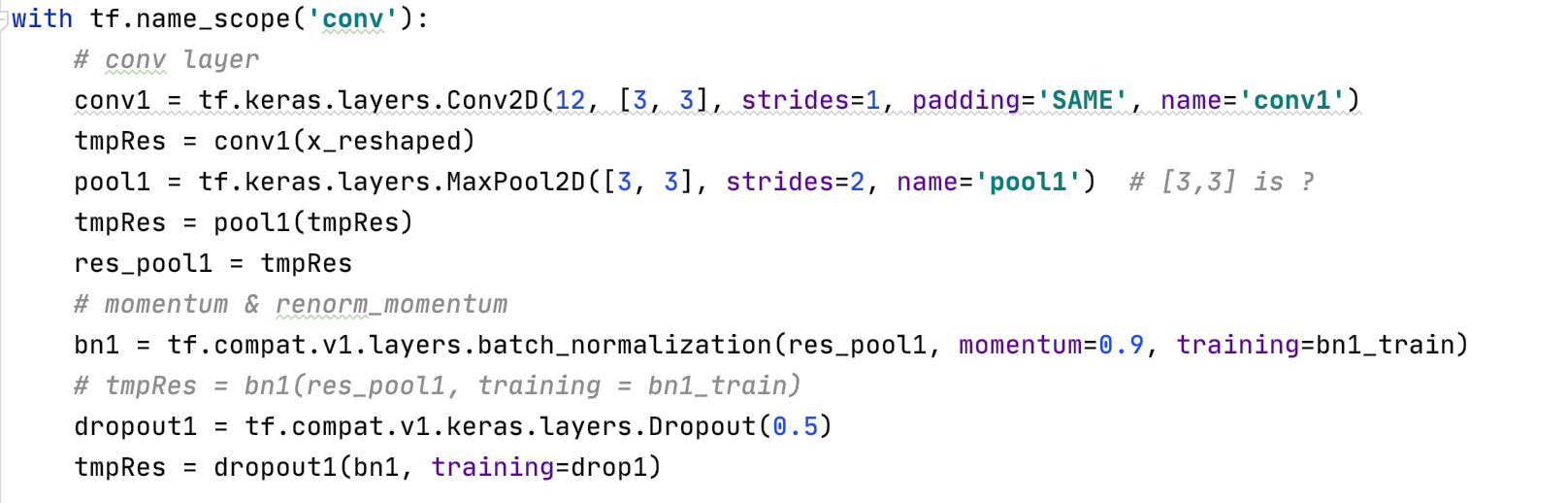


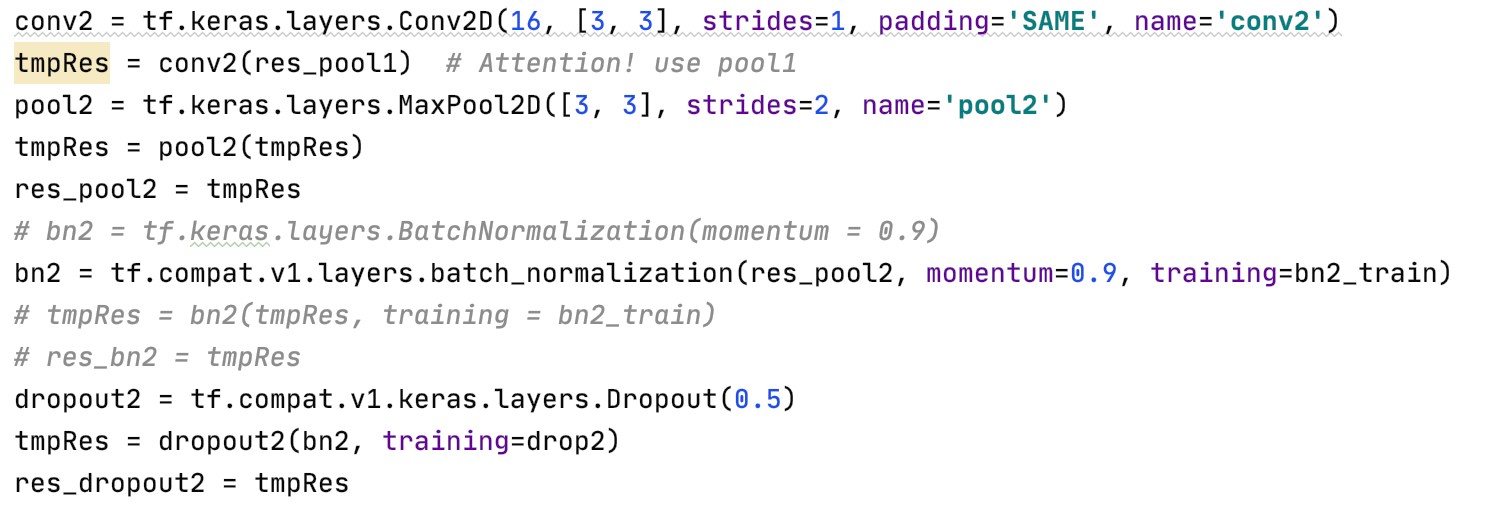


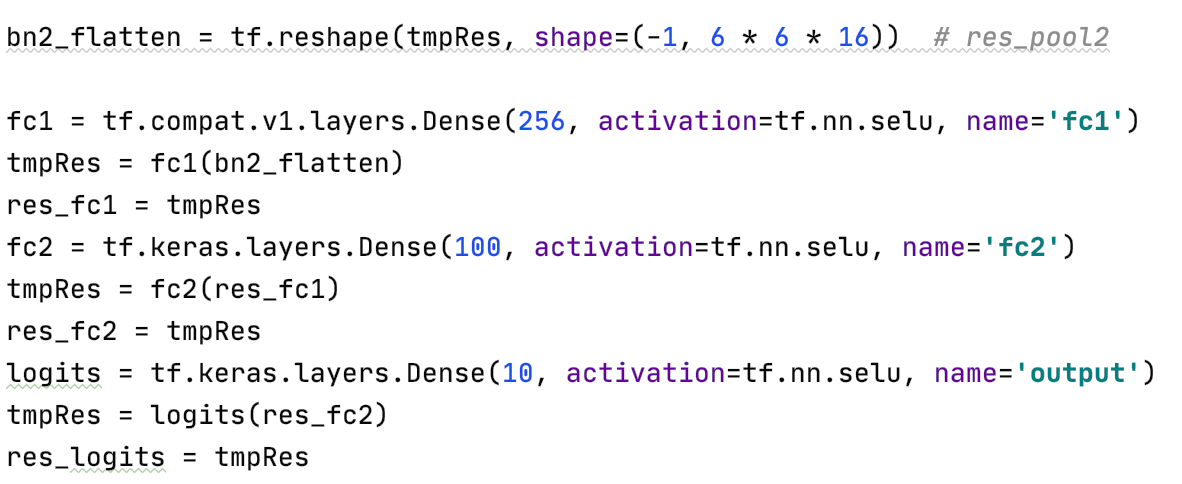
## **Add dropout and batch normalization retraining**

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

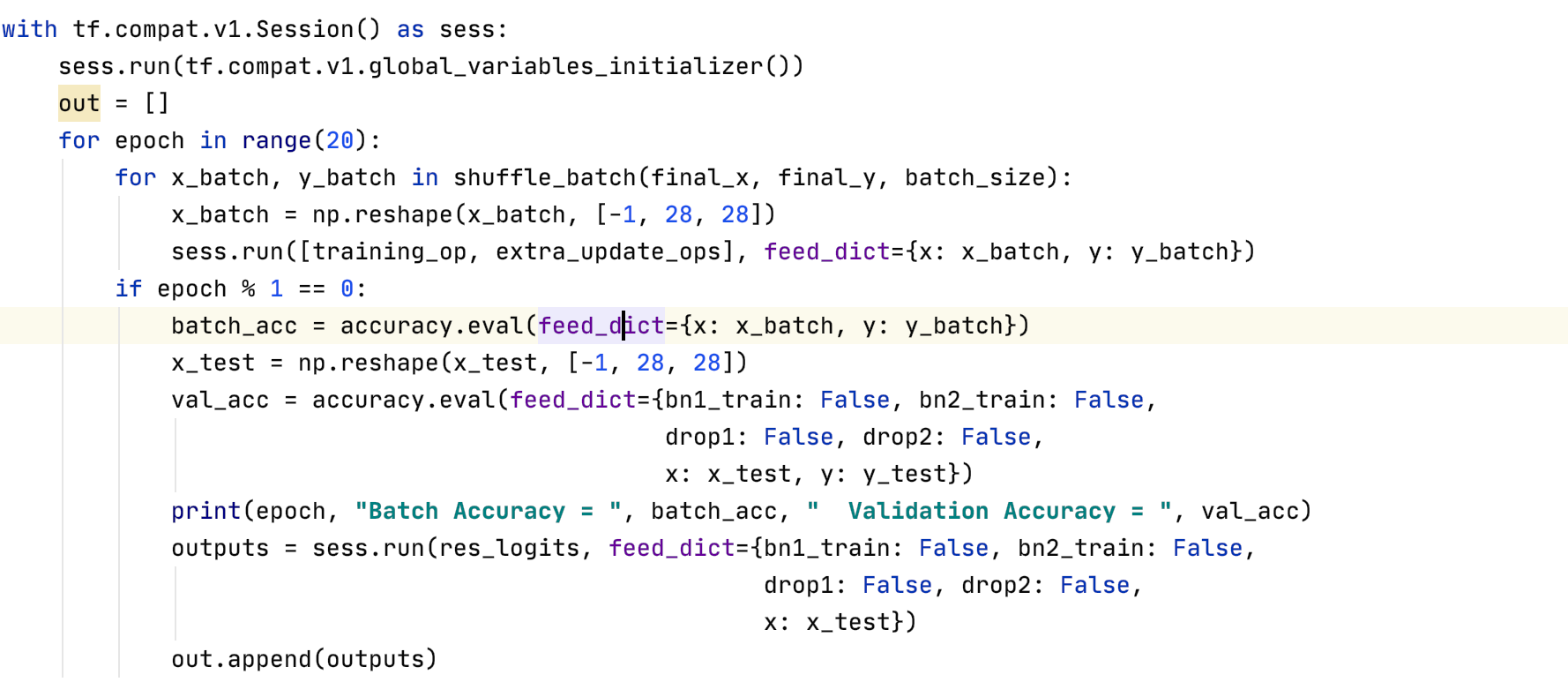
**new with shifted & flipped:**







Train & Test:



### Chapter 4: Experimental environment and operation method

## **Development environment**

- macOS 10.14.6

- python 3.9

- opencv-python 4.5.1.48

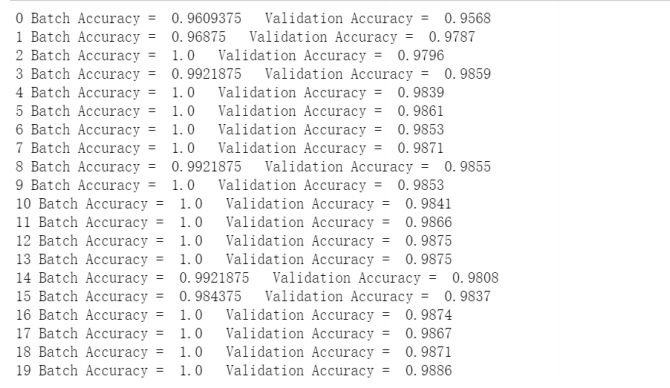
-tensorflow 2.7.0

## **Operation mode**

- python hw5.py

### Chapter 5: Experimental results

## **Results1**

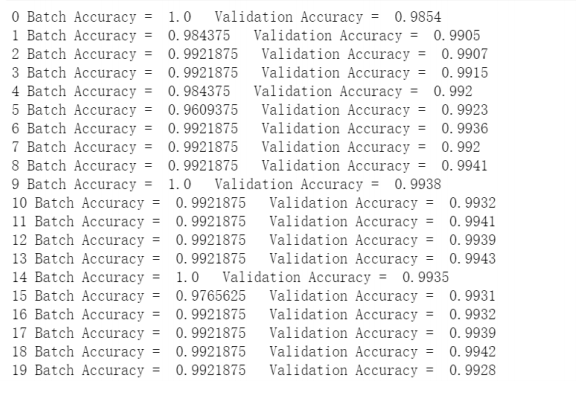


accuracy\_score:



## **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.



accuracy\_score:



### 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**

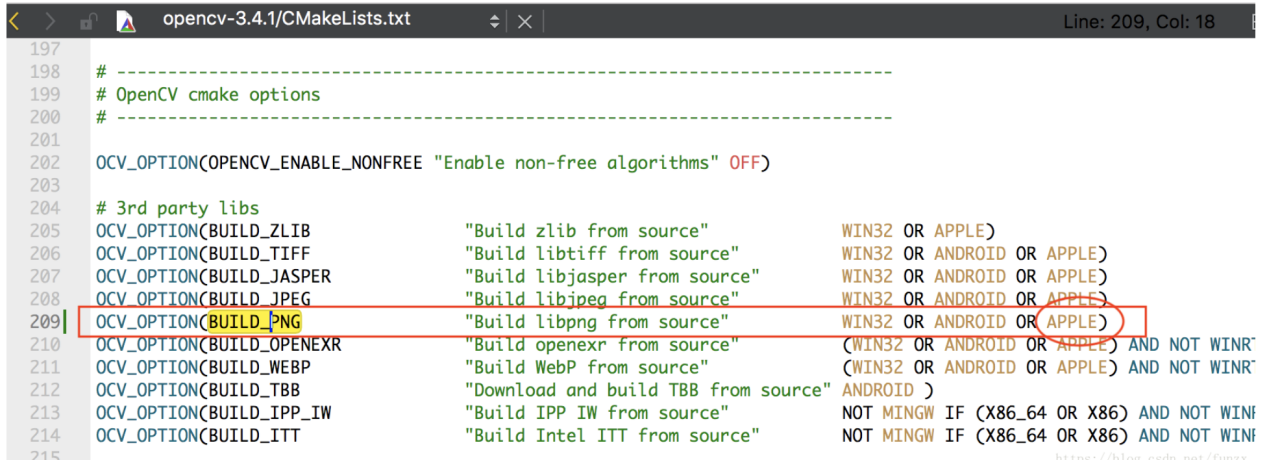
**“libpng warning: Application built with libpng-1.4.12 but running with 1.6.37”**

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



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):



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.