Design of a Universal Image Classification and Recognition System Based on Tensorflow

Ding Lilei

College of Electronic and Information Engineering, Ankang University, Ankang, Shaanxi, China

Abstract: With the continuous development of computer technology and big data applications, research on image classification and recognition has become an important content in the fields of science and engineering, with wide applications in multiple industries. On the basis of studying the process of image classification and recognition, the deep learning framework TensorFlow and Convolutional Neural Network (CNN), this system divides the entire recognition and classification process into multiple universal module units, and designs a visual and universal image classification and recognition system that can automatically adapt to the accurate classification and recognition of various types of images. The experimental results show that the system has a high recognition accuracy (over 95%) and good universality.

Keywords: Tensorflow; Image Classification; CNN

1. Introduction

Image classification and recognition is one of the important tasks in the field of computer vision, widely used in fields such as facial recognition, object detection, and autonomous driving. Traditional image classification methods often rely on manually designed feature extractors, and their performance is limited by the experience and skills of feature designers. In recent years, deep learning techniques, especially CNN, have achieved significant results in image classification and recognition tasks. TensorFlow as a powerful deep learning framework, provides strong support for the implementation of image classification and recognition systems. This article aims to design a universal image classification and recognition system based on TensorFlow, enabling engineers and researchers to focus on applications, model frameworks, and related parameter research without having to write a lot of code.

2. Image Classification and Recognition Process

Image classification and recognition, for users, refers to the process of inputting the image to be recognized into the system, recognizing it through the system, and ultimately storing the image classification in a storage medium. For researchers, it is necessary to continuously modify the network structure and parameters to study their impact on classification and recognition results, in order to continuously improve the accuracy of the system. the system needs to be easy to operate and highly visualized to meet the needs of different personnel.

Image classification and recognition based on deep learning first require model training, and then use the trained model to perform image classification operations. the model training stage first preprocesses the original dataset, unifies the image size and labels it as input for the deep learning network. Next, it is necessary to build the network and select parameters according to actual needs, train the network model, and save the trained model (type:H5). During image recognition operations, the trained model needs to be loaded for image classification and recognition. Based on the above analysis, the model training workflow is shown in **Figure 1.**

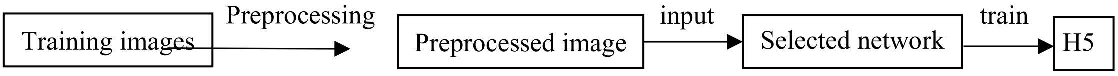


Figure 1. Model Training Workflow

The image classification operation first loads the trained H5 model for recognition, and automatically saves the recognition results in the set folder to complete the image classification. the workflow of image classification operation is shown in **Figure 2**.

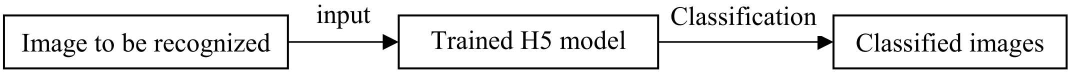


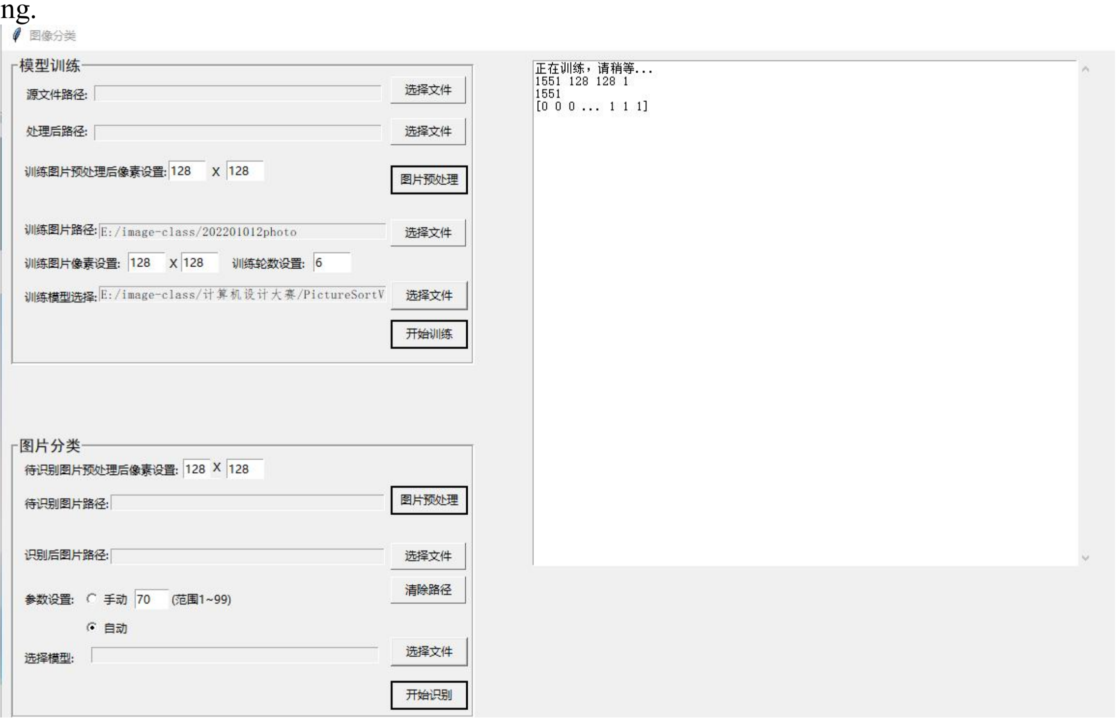
Figure 2. Workflow diagram of image classification operation

3. System Design

3.1 Overall System Design

Due to the different usage scenarios of image classification and recognition, such as inconsistent image categories and sizes, as well as different network structures and parameters required for different usage scenarios, through analyzing the entire operational workflow, the entire recognition and classification process was divided into multiple universal module units to enhance the universality of the system. the model training stage is divided into units such as dataset preprocessing, model building, model training and optimization, and model saving; the image classification operation stage is divided into modules such as image preprocessing, image recognition, and result saving.

The entire system design uses Python language, which is a cross platform language that is convenient to run on different computers. There are mainly two libraries used, the Tkinter library is used for interface design, and the TensorFlow library is used for deep learning [2]· Tkinter is the standard graphical user interface (GUI) library for Python, which supports cross platform GUI program development and is suitable for small GUI program writing. When using it, simply import the library directly. According to the actual usage scenario, the overall interface design of the system is shown in **Figure 3**. the left side of the interface is divided into two parts: model training and image classification. the right side is the information output area, which includes the system's working status, progress, and parameters.



源文件路径:
处理后路径:
训练图片预处理后像素设置: 128 X 128
训练图片路径: E:/image-class/202201012photo
训练图片像素设置: 128 X 128 训练轮数设置: 6
训练模型选择: E:/image-class/计算机设计大赛/PictureSortV

待识别图片预处理后像素设置: 128 X 128
待识别图片路径:
识别后图片路径:
参数设置: ○ 手动 70 (范围1~99)
● 自动
选择模型:

```
正在训练，请稍等...
1551 128 128 1
1551
[0 0 0 ... 1 1 1]
```

Figure 3. System Interface

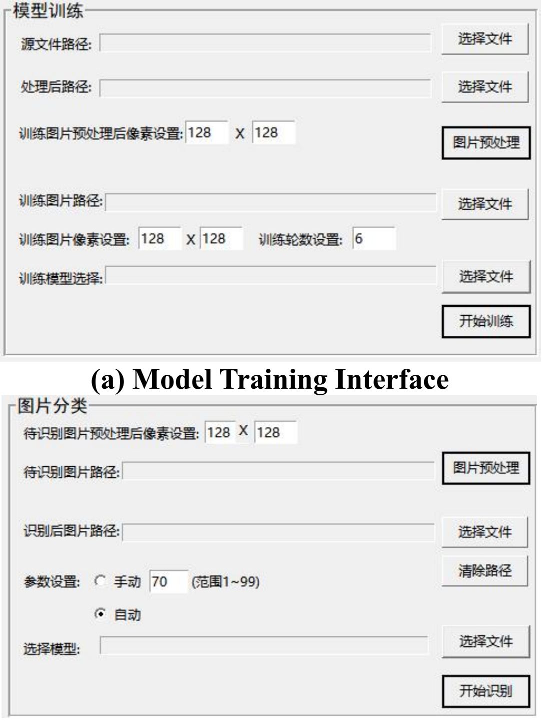
3.2 Model Training Module

This classification and recognition system is based on the deep learning framework of TensorFlow, an open-source artificial intelligence system launched by Google. It has the characteristics of flexibility, efficiency, good scalability, portability, and can be applied to various computing environments from smartphones to large computing clusters [3]·

The system uses TensorFlow's Keras package to build a convolutional neural network model. A general convolutional neural network model includes convolutional layers, max pooling layers, fully connected layers, loss layers, output layers, etc. Each layer needs to set relevant parameters. As for commonly used convolutional layers, it is generally necessary to set the image size, size of convolutional kernels, number of convolutional kernels, activation function, step size, and fill mode of the input convolutional layer. the pooling layer sets the size and step size of the pooling window [4-5]. tf. keras. models. Sequential() function is an advanced API in TensorFlow based on the Keras API, allowing users to build neural networks in a sequential manner without manually configuring the connections of the neural network graph. In the input parameters of Sequential(), the network structure from the input layer to the output layer is described, and the network structure can be added layer by layer [6]. This system completes the construction of the network layer and parameter settings by filling out tables for the hierarchical structure, that is, it transforms the construction of the network structure through programming into filling out Excel tables to complete the construction of the deep learning network structure. During system execution, the network structure and related parameters set in the Excel spreadsheet will be automatically read to complete the automatic construction of the network structure. the model training interface is shown in **Figure 4(a).**

3.3 Image Classification Module

In the image classification module, it is necessary to preprocess the image size uniformly and save it to improve the effectiveness and speed of image recognition. Set the path for saving the image after recognition, and then select different trained models for image classification and recognition, the classification results will be automatically saved in the preset path. In order to meet different needs and facilitate research, there are two modes for the confidence level of the recognized image: manual and automatic. In automatic mode, the highest confidence level in the output classification is used as the recognition result. In manual mode, if the highest confidence level does not reach the set value, the image does not belong to any category and will not be saved in any classification file. Instead, a new file will be created for saving, which means that the category of the image belongs to is unknown. In the process of binary classification of images, if the system recognizes a certain image with a confidence level of 50% for the first category and 60% for the second category, it will be saved in the second category folder in automatic mode. However, if it does not reach the preset 70% confidence level in manual mode, it will not be saved in the first and second category folders, and a new folder will be created for saving. the image classification interface is shown in **Figure 4(b)**.



源文件路径:

处理后路径:

训练图片预处理后像素设置: 128 X 128

训练图片路径:

训练图片像素设置: 128 X 128 训练轮数设置: 6

训练模型选择:

(a) Model Training Interface

待识别图片预处理后像素设置: 128 X 128

待识别图片路径:

识别后图片路径:

参数设置: ○ 手动 70 (范围1~99)

● 自动

选择模型:

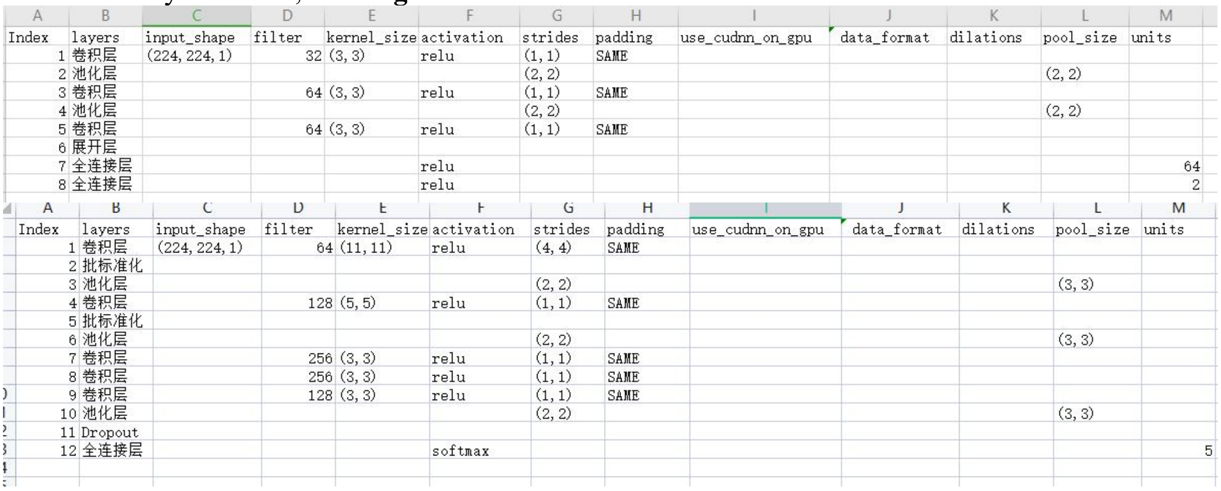
(b)Image Classification Interface
Figure 4. Model Training and Image Classification Interface

4. SystemTesting

System testing is mainly divided into functional testing and universality testing. Functional testing mainly tests whether the system can run stably and complete basic functions, such as data preprocessing, loading different models through different Excel spreadsheets, training and saving models, loading network models, and automatic classification and saving after recognition. Universal testing checks whether the system can classify images of different sizes and types.

4.1 Model Training and Testing

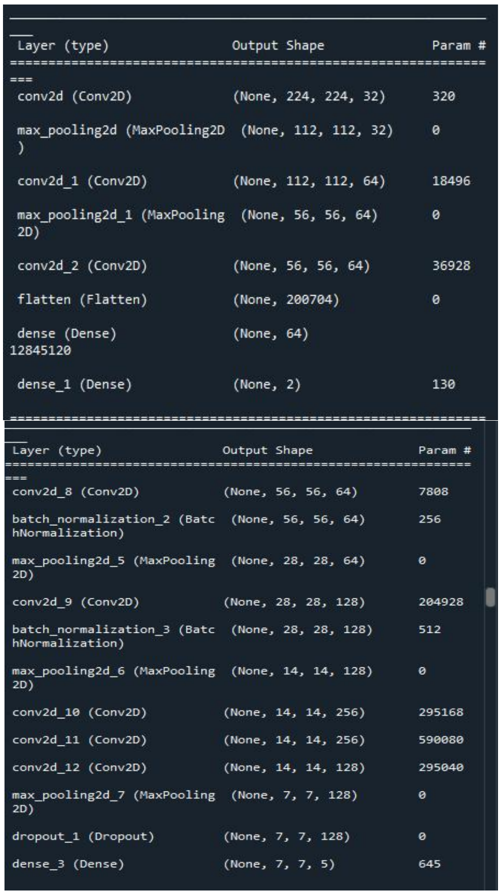
Create two new table files locally in advance, such as the base model BaseModel. xlsx (8 layers) and the AlexNet model AlexNetModel. xlsx (12 layers), and select different tables to load network structures for model training. **Figure 5** shows the network structure constructed by the table, and **Figure 6** shows the network structure information output by the system after loading the table, reading relevant parameters, and successfully building the model.



Index	layers	input_shape	filter	kernel_size	activation	strides	padding	use_cudnn_on_gpu	data_format	dilations	pool_size	units
1	卷积层	(224, 224, 1)	32	(3, 3)	relu	(1, 1)	SAME					
2	池化层					(2, 2)					(2, 2)	
3	卷积层		64	(3, 3)	relu	(1, 1)	SAME					
4	池化层					(2, 2)					(2, 2)	
5	卷积层		64	(3, 3)	relu	(1, 1)	SAME					
6	展开层											
7	全连接层				relu							64
8	全连接层				relu							2

Index	layers	input_shape	filter	kernel_size	activation	strides	padding	use_cudnn_on_gpu	data_format	dilations	pool_size	units
1	卷积层	(224, 224, 1)	64	(11, 11)	relu	(4, 4)	SAME					
2	批标准化											
3	池化层					(2, 2)					(3, 3)	
4	卷积层		128	(5, 5)	relu	(1, 1)	SAME					
5	批标准化											
6	池化层					(2, 2)					(3, 3)	
7	卷积层		256	(3, 3)	relu	(1, 1)	SAME					
8	卷积层		256	(3, 3)	relu	(1, 1)	SAME					
9	卷积层		128	(3, 3)	relu	(1, 1)	SAME					
10	池化层					(2, 2)					(3, 3)	
11	Dropout											
12	全连接层				softmax							5

Figure 5. Network Model Constructed by the Table



```
____________________________________________________________
___
 Layer (type)                Output Shape              Param #
=============================================================
===
 conv2d (Conv2D)             (None, 224, 224, 32)      320

 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 64)        36928

 flatten (Flatten)           (None, 200704)            0

 dense (Dense)               (None, 64)
12845120

 dense_1 (Dense)             (None, 2)                 130

=============================================================
```

```
____________________________________________________________
___
 Layer (type)                Output Shape              Param #
=============================================================
===
 conv2d_8 (Conv2D)           (None, 56, 56, 64)        7808

 batch_normalization_2 (Batc  (None, 56, 56, 64)       256
 hNormalization)

 max_pooling2d_5 (MaxPooling  (None, 28, 28, 64)       0
 2D)

 conv2d_9 (Conv2D)           (None, 28, 28, 128)       204928

 batch_normalization_3 (Batc  (None, 28, 28, 128)      512
 hNormalization)

 max_pooling2d_6 (MaxPooling  (None, 14, 14, 128)      0
 2D)

 conv2d_10 (Conv2D)          (None, 14, 14, 256)       295168

 conv2d_11 (Conv2D)          (None, 14, 14, 256)       590080

 conv2d_12 (Conv2D)          (None, 14, 14, 128)       295040

 max_pooling2d_7 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 7, 7, 128)         0

 dense_3 (Dense)             (None, 7, 7, 5)           645
```

Figure 6. Network Structure Information Output after Successfully Building the Model

4.2 General testing

In order to test the universality of the system, two datasets were selected. the first dataset is a self built dataset, with images collected from a certain experiment. the regular images have a hysteresis curve shape, while the others belong to irregular images. In the early stage, the training set was divided into two categories through manual labeling: regular images and irregular images. the size of the images was 1920×1440, with 983 regular images and 483 irregular images. First, the model is trained systematically, and then the images to be classified are classified. During the testing, a trained model was used to test 2408 photos. the test result showed that the system was able to automatically classify and save the images to be recognized in two folders: regular (T) and irregular (F). Among them, 1074 were recognized as regular images (11 with classification errors) and 1334 were irregular photos (35 with classification errors), with an accuracy rate of 98%. the system classification results are shown in **Figure 7**.

The second dataset is a test set of flower data downloaded from the internet, consisting of 5 categories with 650 images per category. After training, use the trained model to recognize 162 images to be classified. Test results: the system can automatically classify and save the recognized results in 5 folders, daisy/dandelion/rose/sunflower/tulip. Among them, there is 1 recognition error in the tulip group, 1 recognition error in the rose group, 3 recognition errors in the dandelion group, and all others are correct, with an accuracy rate of 97%.

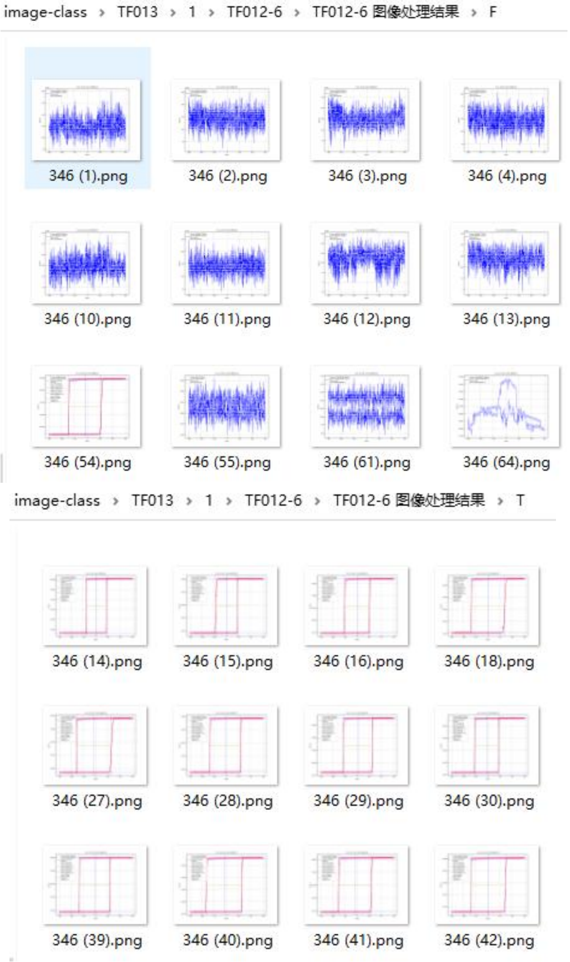


Figure 7. Test Results of Self Built Dataset

The above test results indicate that the system can effectively complete model training and image classification, and the accuracy of the test results for both datasets is also high.

5. Conclusion

This article designs and implements a universal image classification and recognition system based on TensorFlow. the recognition and classification process of images is divided into several universal modules. Excel tables are used to select the network layer structure and related parameters, and Tkinter is used to create a GUI application, which is conducive to observing the training process, analyzing and adjusting image size and various parameters in a timely manner. the test results show that the system can automatically adapt to different types and quantities of images to be recognized, making it convenient for users to operate, making it more suitable for a wider range of scenarios. the system has also been used for a long time, with stable performance and high accuracy in image recognition and classification.

Acknowledgments

Shaanxi Provincial Department of Education Science Research Program Project (20JK0475).

Reference

[1] Zhao Liping, Yuan Xiao, Zhu Cheng, et al. Research on the Progress of Residual Networks for Image Classification [J]. Computer Engineering and Applications, 2020, 56(20):9-19.

[2] Feng Guier. GUI (Tkinter) instance development based on Python [J]. Information and Computer (Theoretical Edition), 2023, 35(01):175-178.

[3] Xing Yanfang, Duan Hongxiu, He Guangwei the Application of TensorFlow in Image Recognition Systems [J]. Computer Technology and Development, 2019, 29(05):192-196.

[4] Yuan Wenchui, Kong Xue. Research on Convolutional Neural Networks Based on TensorFlow Deep Learning Framework [J]. Microcomputer Applications, 2018, Vol. 34, Issue 2.

[5] Zhao Wenchao, Wang Yanxin, Yong Junfei, et al. Underwater pipeline inspection robot based on TensorFlow image classification [J]. Modern Computer, 2022, 28(11):92-95.

[6] Liu Mengya, Mao Jianlin. Research on the Impact of an Improved Pooling Model on the Performance of Convolutional Neural Networks [J]. Electronic Measurement Technology, 2019, Volume 42, Issue 5.