|  |  |
| --- | --- |
|  |  |

Undergraduate Project Report

2018/19

**App design and realization for Photo identification of circuit**

|  |  |
| --- | --- |
| Name: | Wang Lin |
| School: | International School |
| Class: | 2015215120 |
| QM Student No.: | 151007051 |
| BUPT Student No.: | 2015213432 |
| Programme: | Internet of Things Engineering |

**Date: 06-05-2019**

Table of Contents

[Abstract 3](#_Toc8070062)

[Chapter 1: Introduction 5](#_Toc8070063)

[Chapter 2: Background 7](#_Toc8070064)

[2.1 Image recognition—HOG+SVM 7](#_Toc8070065)

[2.1.1 Feature 7](#_Toc8070066)

[2.1.2 Pre-process 7](#_Toc8070067)

[2.2 SVM 9](#_Toc8070068)

[2.3 Faster‑RCNN 10](#_Toc8070069)

[2.3.1 PRN candidate frame extraction module; 11](#_Toc8070070)

[2.3.2 Fast R-CNN 12](#_Toc8070071)

[2.4 Analysis Terminology 12](#_Toc8070072)

[2.5 OCR (Optical Character Recognition) 13](#_Toc8070073)

[2.6 TensorFlow 14](#_Toc8070074)

[2.7 PaddlePaddle 15](#_Toc8070075)

[Chapter 3: Design and Implementation 17](#_Toc8070076)

[3.1 Image Recognition Module 17](#_Toc8070077)

[3.1.1 Traditional Recognition Method 17](#_Toc8070078)

[3.1.2 Deep learning with TensorFlow 20](#_Toc8070079)

[3.1.3 Deep learning with PaddlePaddle (baidu, 2019) 25](#_Toc8070080)

[3.2 Character Recognition Module 28](#_Toc8070081)

[3.2.1 OCR 28](#_Toc8070082)

[3.2.2 Kerbs 28](#_Toc8070083)

[3.2.3 CTC (Connectionist Temporal Classification) 29](#_Toc8070084)

[3.3 App Design 30](#_Toc8070085)

[3.3.1 Overview 30](#_Toc8070086)

[3.3.2 Encapsulation 30](#_Toc8070087)

[Chapter 4: Results and Discussion 33](#_Toc8070088)

[4.1 Function Demonstration 33](#_Toc8070089)

[4.1.1 LogIn/Register Functions 33](#_Toc8070090)

[4.1.2 Photo Search 33](#_Toc8070091)

[4.1.3 Circuit Components recognition 34](#_Toc8070092)

[4.1.4 Add, modified and delete 35](#_Toc8070093)

[4.1.5 Recommendation algorithm 36](#_Toc8070094)

[4.2 Evaluation 36](#_Toc8070095)

[Chapter 5: Conclusion and Further Work 38](#_Toc8070096)

[5.1 Conclusion 38](#_Toc8070097)

[5.2 Further Work 38](#_Toc8070098)

[References 40](#_Toc8070099)

[Acknowledgement 41](#_Toc8070100)

[Appendix 42](#_Toc8070101)

[Risk Assessment 43](#_Toc8070102)

[Environmental Impact Assessment 44](#_Toc8070103)

Abstract

With the development of image recognition technology, more and more fields have integrated image recognition technology with their own products to enhance the competitiveness of their own products. In the field of education, search-based software combined with image recognition came into being. This kind of software makes self-study more efficient. However, compared to the Chinese subject, the search software market for university subjects is still blank. Many college students can only use traditional books to learn when they are exposed to new subjects. Many times they do not know the answers when they do exercises. Process of learning is deadlocked and inefficient.

In order to solve this problem, the project is aimed at one of the more difficult subjects in the university - electronic circuits, designing a software that recognizes circuit components and solve problems, helping beginners get started quickly. First, the project uses the PaddlePaddle framework, combined with existing models and APIs, to train a custom recognition model for electronic circuit subjects. The model can identify the circuit components in the circuit diagram, such as capacitors, resistors, power supplies, etc. In addition, the project also has a search and answer function, through the combination of OCR technology for text recognition of the title, the recognition result will fuzzy match with the background question bank, to get the corresponding result. When the user takes a photo of the question, the app will immediately return the details of corresponding question, and the user can view and proofread with own answers. Finally, the software also has a simple recommendation function, the software will record the frequency of each question in the background, and according to the relationship between the title and frequency, to topping the question that the user is not familiar with, to help novice learn better.

Keywords: Image Recognition, OCR, Android, Deeplearning, PaddlePaddle 摘要

随着图像识别技术的发展，越来越多的领域将图像识别技术与自身的产品相融合，以提高他们自身产品的竞争力。在教育领域，结合图像识别的搜题软件应运而生，这种软件让自学变得更加有效率。但是，相对于中学科目，针对大学科目的搜题软件市场仍然是一片空白。许多大学生在接触新学科时，只能用传统的书本去学习，做练习题时很多时候也不知道答案，学习陷入了僵局，效率低下。

为了解决这个问题，该项目针对大学其中一个较难科目——电子电路，设计一款可以识别电路元件和解题的软件，帮助初学者快速入门。首先，该项目使用PaddlePaddle框架，结合已有模型和API，训练出针对电子电路科目的识别模型，该模型可以通过拍照识别出电路图中的电路元件，如电容，电阻，电源等，帮助初学者打牢基础，快速入门。此外该项目还拥有搜题解答功能，通过结合OCR技术进行题目的文字识别，识别结果和后台题库进行模糊匹配，在有对应题目的前提下，用户给题目拍照，app会立刻返回相应的题目详情，包括题目和相应答案，用户可以查看并且与自己的解答相互校对。最后，该软件还拥有简单的推荐功能，该软件会后台记录每道题目的查看频率，并且根据题与题之间的关联性，置顶用户不熟悉的题目，来帮助初学者更好的学习。

关键词: 图像识别, OCR, Android, Deeplearning, PaddlePaddle

# Introduction

Image recognition technology is a technique based on images, which uses computers to process, analyse and understand images to identify objects of different modes. With the development of image recognition technology, the field of image recognition technology is more and more widely used. They integrate image recognition technology with their own products to improve market competitiveness. This project is required to design an intelligent recognition app of circuit diagram, with deploying the image recognition module to the local Android App, helping the circuit beginners to identify the circuit components through model training. In addition, combined with OCR technology, this app can also answer some simple questions.

There is a huge problem solving software market expectation in university, but the related products are scarce. If there is a software that helps users get started quickly at the beginning of their studies, it will capture the market share rapidly, with the same time helping beginners building a knowledge system that simplifies the learning process.

In this paper, I will first introduce the basics involved in implementing this project, including the network framework and principles, and secondly I will introduce the ideas I have in programming, as well as the errors and solutions. After that, I will show you the products I have made, including the methods and precautions I will use. Finally, I will explain my experience and gains through this project.

This project abandons the traditional object recognition HOG+SVM single object feature extraction classification method, and turns to a deep learning method with higher correct rate.That is, under the framework of TensorFlow, combined with the Faster-R-CNN network, by adding an RPN network, a candidate frame is generated based on the Anchor mechanism (instead of the original selective search), and integrated all items into one network and input to the ROI pooling layer to achieve target recognition and location by target classification and coordinate regression. Compared with the original recognition framework, the Faster-R-CNN network has good positioning efficiency and accuracy.

In addition, in order to be easier to deploy to the local and android mobile, the project also uses the paddlepaddle framework to deploy the trained model to the mobile terminal better by calling the existing api. And the app also adds the ability to understand the problem, through the existing api in the paddlepaddle combined with the CTC model to achieve the OCR function on the text image, then fuzzy matching with the data in the question bank, giving the corresponding answer. The project also has a recommendation function, based on the user's viewing frequency for each question. Record to view “heat” in the background, and recommend unfamiliar knowledge to the user according to the relationship between the questions and the questions, help the users learn the circuit subject and improve the practicality.

# Background

## Image recognition—HOG+SVM

First, pre-process the input image and calculate the gradient value of the pixel, to form the gradient histogram, then normalize the blocks. Finally, the HOG feature is collected and placed in the SVM for supervised learning to achieve target detection.

### Feature

Feature is the extraction of useful information images. Usually the feature converts an image of w\*h\*3 (width\*height\*3, 3 channels) into a vector/matrix of length n.

In the direction gradient histogram (HOG), the direction distribution of the gradient is used as a feature. A gradient along the X and Y axes of a picture is useful，Because the gradient values ​​at the edges and corners are large, and the edges and corners contain shape information for many objects. (Slyne\_D, 2017)

### Pre-process

* Gamma correction and grayscale
* Gamma correction is the effect of reducing luminosity on the experiment.
* Grayscale is to turn a colourful picture into a grayscale image

Reduce the error by the above method

Here the pixels of the Figure 1 are 720\*475, we choose the patch of 100\*200 size to calculate the HOG feature. Take this patch out of the and adjust the size to 64\*128

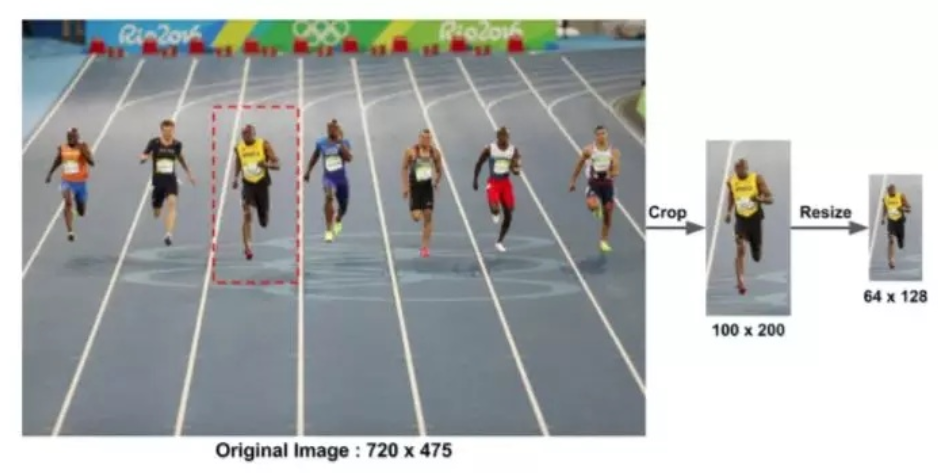


Figure Original Image

Calculate the gradient image by Figure 2

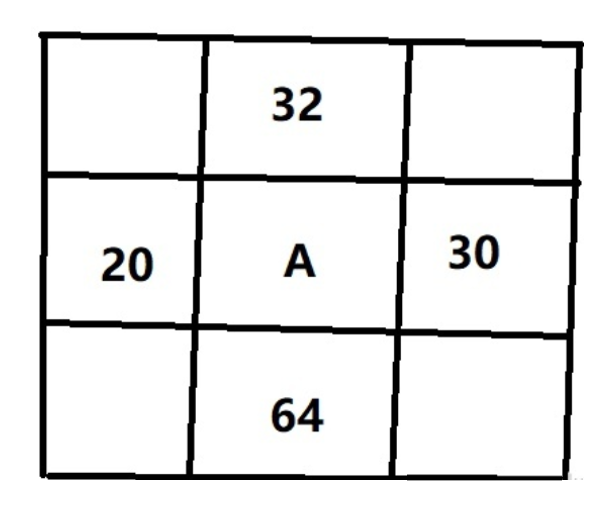


Figure operator

Calculate the gradient value of each pixel to get the gradient map

Next, use the following formula **(1)** to calculate the magnitude g and direction theta of the gradient:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

The result is shown as Figure 3

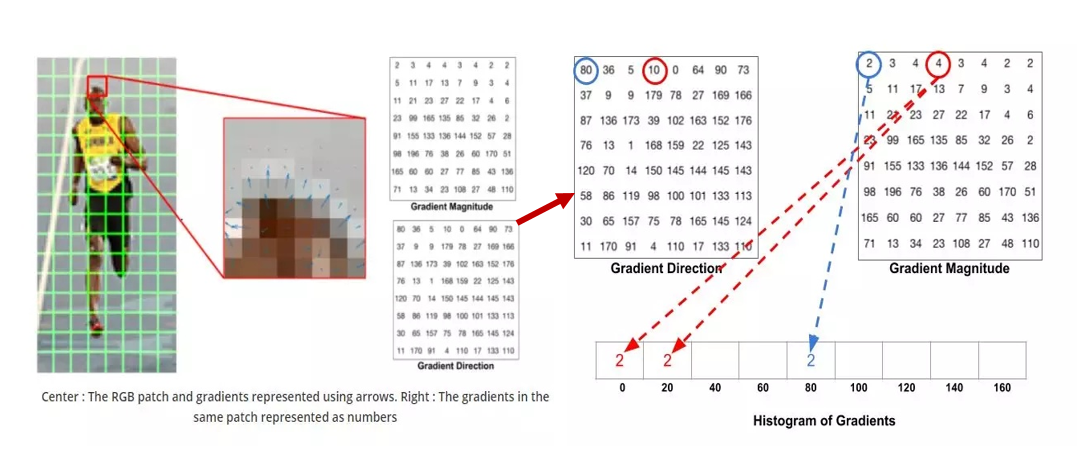


Figure HOG of target Image

We used the gradient magnitude and direction of the grid in the Figure 3

Select one bin to use according to the direction, and determine the size of the bin based on the secondary value.

#### Result

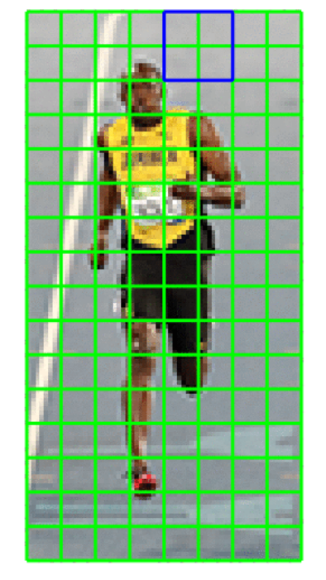


Figure Result

As Figure 4 shown, the green square is an 8\*8 size cell, and the blue square is a block of 4 cells. Each block will be iteratively intercepted in the same way as the blue square above.

The final result is the HOG feature. (Navneet Dalal, 2005)

## SVM

SVM is supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is shown as Figure 5, which is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. (Wikipedia, 2019)

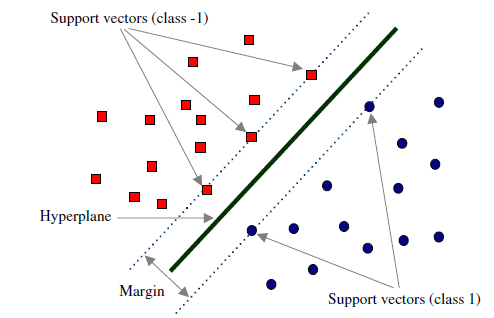


Figure SVM model

## Faster‑RCNN

Faster-RCNN is the first true end-to-end deep learning detection algorithm proposed in 2015. The biggest innovation is that the candidate framework is generated based on the Anchor mechanism by adding the RPN network (instead of selective search).

Finally, feature extraction, candidate frame selection, border regression and classification are integrated into one network, which effectively improves detection accuracy and detection efficiency. The Faster RCNN model is shown as Figure 6**.**

The Faster-R-CNN algorithm consists of two major modules:

* PRN candidate frame extraction module;
* Fast R-CNN detection module.

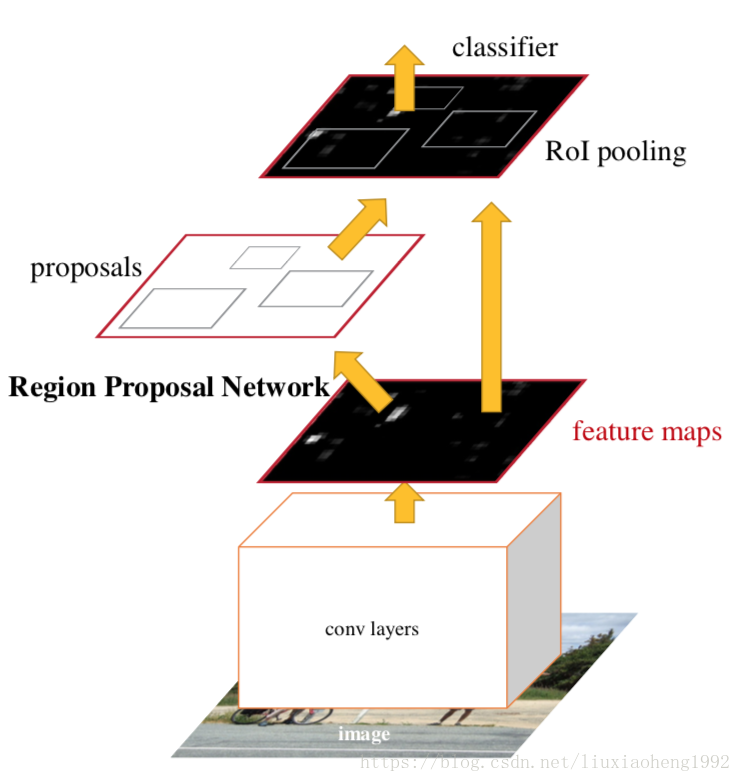
Among them, RPN is a full convolutional neural network for extracting candidate boxes; Fast R-CNN detects and identifies targets in proposal based on the RPN extracted proposal (Michaelliu\_dev, 2018)

Figure Faster RCNN model

### PRN candidate frame extraction module;

The core idea of ​​RPN is to use the CNN convolutional neural network to directly generate Region Proposal. The method used is essentially a sliding window (just slide over the last convolution layer), because the anchor mechanism and the border regression can be multi-scale and long Region Proposal.

The RPN network is also a fully-convolutional network (FCN), which can be trained end-to-end for the task that generates the detection suggestion box, and can simultaneously predict the boundary and score of the object. Just two additional convolutional layers (full convolutional layers cls and reg) have been added to CNN. (Michi, 2018) The processing of RPN mapping is shown as Figure 7.

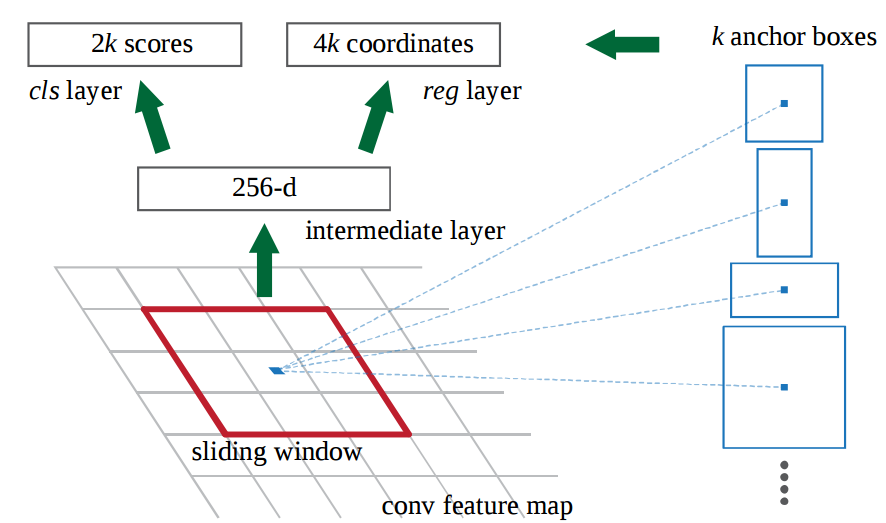


Figure PRN mapping

### Fast R-CNN

After extracting proposals from RPN, the author chose to use Fast-R-CNN to achieve the detection and identification of the final target.

RPN and Fast-R-CNN share 13 convolution layers of VGG. Obviously, it is not a wise choice to completely isolate the two networks. Alternating training is used to share convolutional features.

After the input image is scaled, the convolutional layer is extracted to obtain a feature map, and then the feature map is sent to the RPN network to generate a series of possible candidate frames, and then all the candidate frames of the original feature maps and RPN outputs (ROI matrix (N)

\*5)) Input to the ROI pooling layer, extract and collect the proposal, and calculate the fixed size maps of fixed size 7×7, and send them to the fully connected layer for target classification and coordinate regression.

## Analysis Terminology

1. TP (True Positive): predicting the correct answer
2. FP (False Positive): wrong to predict other classes as this class
3. FN (False Negative): This type of label is predicted to be other types of labels mAP (mean average precision)
4. Precision: refers to the proportion of positive samples in the positive example determined by the classifier **(2**

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

1. Recall rate: refers to the proportion of the total positive case that is predicted to be positive.**(3)**

|  |  |  |
| --- | --- | --- |
|  |  | (3) |
|  |  |  |

1. Accuracy: represents the correct weight of the classifier for the entire sample.**(4)**

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

1. F1-score: F1-score is the harmonic mean of the average accuracy and recall rate for each category.**(5)**

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

1. mAP (mean average precision) :For object detection tasks, each type of object can calculate its precision and recall rate. It can be calculated multiple times under different thresholds. Each class can get a P-R curve, the area under the curve is mAp.

## OCR (Optical Character Recognition)

OCR (Optical Character Recognition) refers to an electronic device (such as a scanner or digital camera) that checks characters printed on paper, determines its shape by detecting dark and bright patterns, and then translates the shape into a computer by character recognition.

The process of text; that is, for printed characters, the text in the paper document is optically converted into a black and white dot matrix image file, and the text in the image is converted into a text format by the recognition software for further processing by the word processing software.

How to debug or use auxiliary information to improve the recognition accuracy is the most important issue of OCR.

In recent years, with the advancement of deep learning technology, it has played a positive role in promoting the correct rate of OCR.

Some mature product-level OCR systems can recognize Chinese characters, letters, numbers, special characters and perform layout restoration at the same time. They can be applied to personal identification identification, business card recognition, printed text recognition and other specific scenes in identity verification.

## TensorFlow

TensorFlow is based on data flow graphs and is an open source framework for large-scale distributed numerical computing.

A node represents an abstract calculation, and an edge represents a tensor of interconnected nodes.

TensorFlow supports a variety of heterogeneous platforms, supports multiple CPU / GPU, servers, mobile devices, with good cross-platform features; TensorFlow architecture is flexible, can support a variety of network models, has good versatility; In addition, TensorFlow architecture has Good scalability, extended support for OP, outstanding performance in Kernel specialization. The structure of TensorFlow is shown as Figure 8.

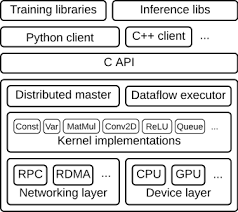


Figure Tensorflow structure

TensorFlow is currently the most popular algorithmic engine for artificial intelligence in terms of usage and activity.

It provides implementations of the basic elements of deep learning, such as conv, pooling, lstm, and other basic operators.

The Tensor library is transparent to CPU/GPU and implements a lot of operations (such as slicing, array or matrix operations).

The transparency here means that how to run on different devices is the framework to help users to achieve, users only need to specify which device to perform which operation.

## PaddlePaddle

The open source of the PaddlePaddle framework began in September 2016. Its biggest feature and positioning is easy to use, so many algorithms are completely encapsulated.

Not only for currently available CV, NLP and other algorithms (such as VGG, ResNet, LSTM, GRU, etc.), it encapsulates the word vector under the model library models (https://github.com/PaddlePaddle/models). Including Hsigmoid accelerated word vector training and noise contrast estimation accelerated word vector training), RNN language model, click rate estimation, text classification, sorting learning (one of the core issues of information retrieval and search engine research), structured semantic model, naming

General solution for artificial intelligence in multiple technical fields such as entity recognition, sequence-to-sequence learning, reading comprehension, automatic question and answer, image classification, target detection, scene text recognition, and speech recognition

Each of the above solutions is designed for a certain technical scenario. Therefore, the developer may only need to understand the source code principle slightly, execute the running command according to the example of the official website, replace it with its own data, and modify some hyper parameters to run. The structure of PaddlePaddle is shown as Figure 9.

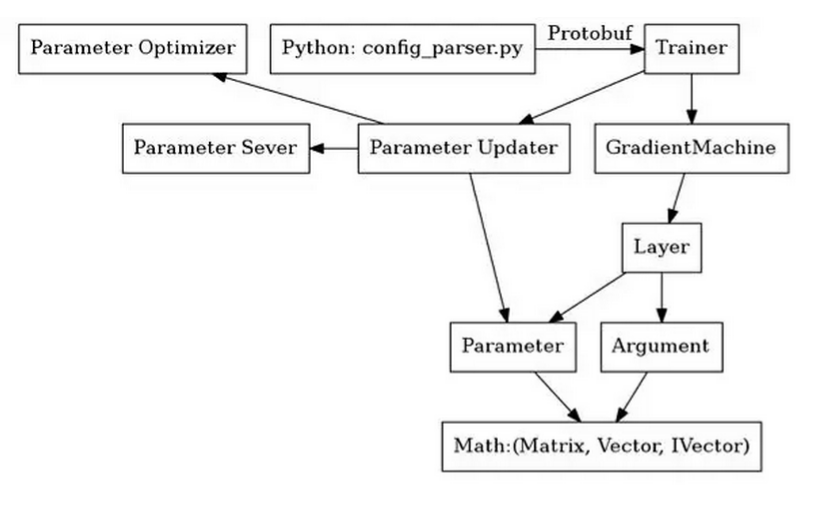


Figure PaddlePaddle Structure

And because there is no exposed python interface to the user, it is easier to understand and use.

However, because the focus is on use, in terms of scientific research or new functions, if you modify the algorithm, you need to start from the C++ bottom layer of the framework.

It has many advantages. First of all, it is ease of use. Compared to the underlying Google TensorFlow, the ease of use of PaddlePaddle is very obvious: it allows developers to focus on building high-level parts of the deep learning model.

Second, it is faster. As mentioned above, PaddlePaddle's code and design are more concise, and using it to develop models obviously saves developers some time.

This makes PaddlePaddle ideal for industrial applications, especially those that require rapid development.

# Design and Implementation

## Image Recognition Module

This part mainly introduces the image recognition module of the app, which is the core part of the project. The main functions include the identification and positioning of circuit components, and also the determination of circuit logic, and will also be related to the OCR module. In this project, I used three ways to implement this module. Each of these three methods has its own advantages and disadvantages. In the screening process, I also have my thoughts and errors encountered. I finally chose the PaddlePaddle framework to implement this project.

### Traditional Recognition Method

First input origin image into the opencv, then image preprocess, i.e convert the image from RGB image to 2-value image, and denoising b, i.e convert the image from RGB image to 2-value image, and denoising by erode method. And do the edge detection and threshold analysis by Scharr, which is python module. After a serious morphological operation, and extract the feature of object. Finally, combined SVM (Support Vector Machine) to realize the recognition of circuit of compounds. The whole process of HOG+SVM is shown as Figure 10.

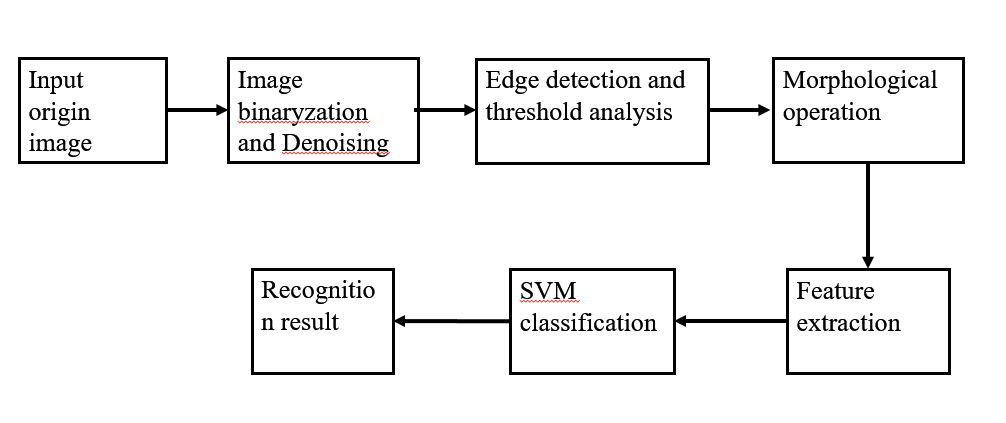


Figure Process of HOG+SVM

##### Image preprocessing

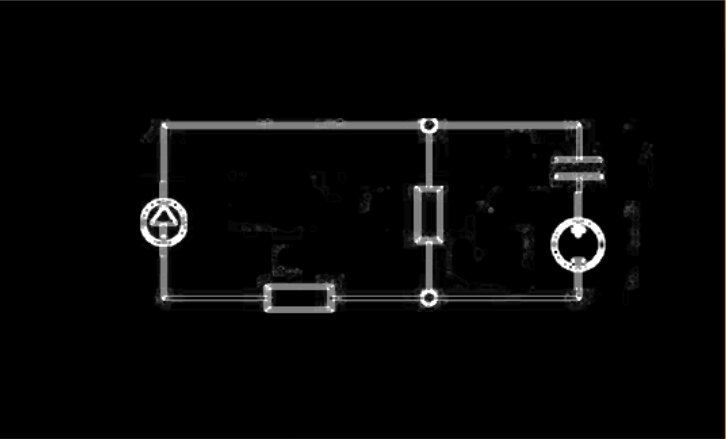
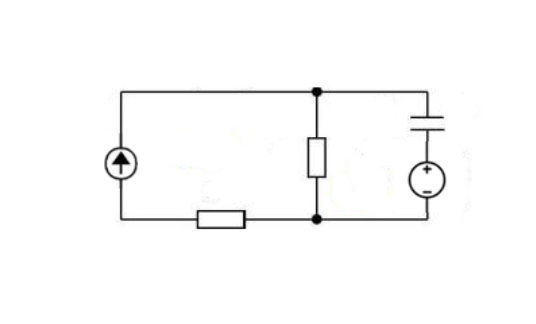
First, the input picture Figure 11 is converted from a three-channel picture to a 2-value picture, then filtered the image, the image intensity is generally composed of derivatives, but the derivative is usually very sensitive to noise. Therefore, filters must be used to improve the performance of the noise-related edge detector. Generally, Gaussian filtering is used to operate.

Figure 11 Original Image

Figure 12 Processed Image

Then, enhance the neighbourhood strength of each point of the image. A point where the image grey point neighbourhood intensity value has a significant change is highlighted.

Then for these enhanced images, we use threshold to detect gradient values ​​for target detection. Finally, use the Scharr library in opencv to extract the horizontal Figure 13 and vertical lines Figure 14 of the target image Figure 12. Then merge the two images, then do the closing operation separately to reduce noise interference. (XIAO Dou, 2016)

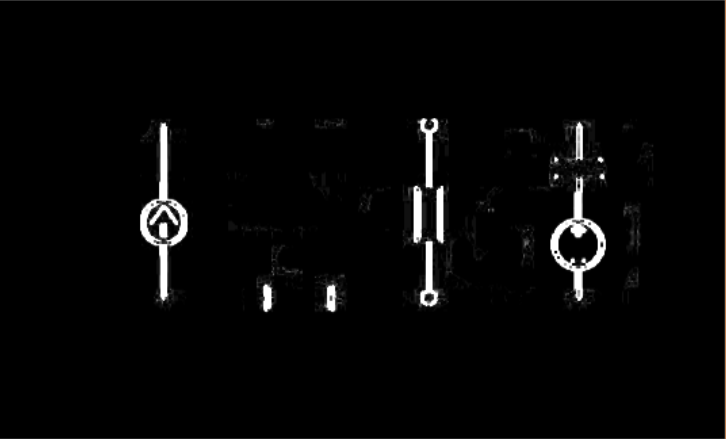
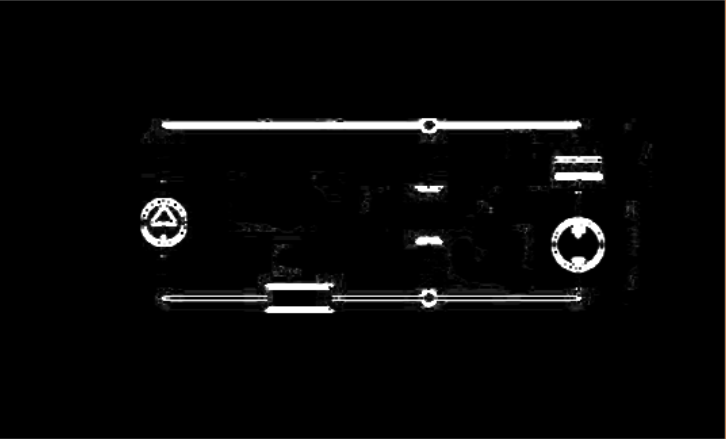


Figure 13 X-axis Processed

Figure Y-axis Processed

##### HOG feature extraction

The Histogram of Oriented Gradient (HOG) feature is a feature descriptor used for object detection in computer vision and image processing.

It consists of calculating and arranging gradient histograms of local regions of the image to form features.

* extracting the edge information of the image by using the Canny operator, and the subsequent HOG features will be extracted at these edges;
* Layer the image, the first layer (denoted as L = 0) is the entire image, and the second layer (denoted as L = 1) is to divide the entire image (the first layer) into four equal parts, the third layer (Remarked as L=2) is to divide each sub-area in the second layer into four, and so on;
* hierarchically calculating the HOG feature, in each layer, statistically analyzing the gradient histogram features of each block region in K directions and concatenating the features of the image under the layer;
* Combine the HOG features of the image under each layer in series to form the final HOG feature.

##### SVM Classification

* Step1: Obtain a positive sample set and use hog to calculate features to obtain a hog feature descriptor.
* Step2: Obtain the negative sample set and use the hog calculation feature to get the hog feature descriptor. The negative sample image can be obtained by randomly cropping the image without the detection target. Usually the number of negative samples is much larger than the number of positive samples.
* Step3: Train the positive and negative samples with SVM to get the model.
* Step4: Use the model for negative sample detection. Perform multi-scale detection on negative samples in the Training set. If the classifier misdetects non-targets, the captured image is added to the negative samples. (Hard-negative mining)
* Step5: Retrain the model in combination with the difficulty.
* Step6: Apply the final classifier model to detect the test set, scan the different scales of each image, extract the descriptor and classify it with the classifier. If the detection is the target, use the bounding box to frame it. Apply non-maximum suppression after image scanning to eliminate overlapping unwanted targets.

##### Evaluation

First of all, when I started the building of this recognition module, I considered the following features of this project.

1. The recognized images are all black and white, easy to convert to 2-value images, avoiding the pre-processing of colour segmentation
2. The identified parts are geometrical figures with relatively regular borders and relatively simple shapes, and the outlines are easily extracted.

Based on the above considerations, I initially considered using the HOG+SVM network framework to achieve this goal, only using opencv to process images, bypassing deep learning to build a network model, and logically processing the target image to achieve the goal of target recognition.

However, in practice, I found that there are two problems:

* One is the problem of positioning the target. Because this project’s requirement is different from most other examples like face recognition models, the goals are different. This project requires much in amount and accurate with similar object, such as the schematic diagram of the capacitor and the power supply. The difference is very small, and the feature acquisition is difficult to get. Especially when the data set is small, which makes it difficult to achieve
* The second problem is the logical judgment about the circuit connection. For the next stage of the project, that is, after the component identification completed, I have to complete the simple connection relationship identification of circuits, such as series connection and parallel connection

Only using the HOG+SVM framework, it is very difficult to achieve this goal. Using logical judgment alone consumes a large amount of memory for each recognition, and time. Especially for non-traditional connections, such as hybrids connection, which are not effective. Therefore, I tried to solve this problem by using the deep learning method, especially, use TensorFlow framework combines with Faster RCNN+Inception-ResNet pre-trained model, which is easy to achieve the Multi-target detection. The detail is following.

### Deep learning with TensorFlow

#### Overview

First, the input image is represented as a tensor (multidimensional array) of Height × Width × Depth. After processing by the pre-trained CNN model, a conv-feature map is obtained. That is, CNN is sent as a feature extractor to the next part. The process of Target Detection is shown as Figure 15

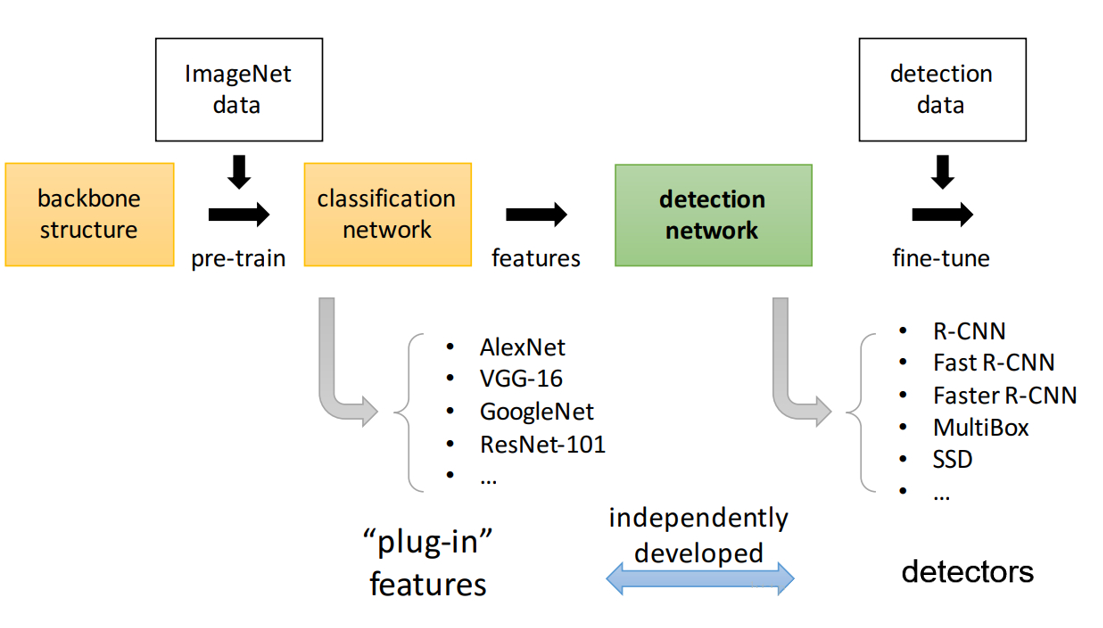


Figure Process of Target Detection

This technique is common in Transfer Learning, especially the use of network weights trained in large-scale data sets to train classifiers on small-scale data sets.

The RPN (Region Propose Network) then processes the extracted convolutional feature map. RPN is used to find a predefined number of regions (regions, bounding boxes) that may contain objects.

The most difficult problem in target detection based on deep learning is to generate a variable-length bounding box list. When constructing a deep neural network, the final network output is generally a fixed-size tensor output.

For example, in image classification, the network output is the tensor of (N, ), N is the number of category labels, and the scalar value of each position of the tensor indicates that the image is the probability value of the category label i

In RPN, the problem of indefinite length of the bounding box list is solved by using anchors, that is, uniformly placing a fixed-size reference bounding box in the original image.

When the possible related objects and their corresponding positions in the original image are obtained, the problem is simpler. (Javier, 2018)

Use CNN extracted features and bounding boxes containing related objects, use RoI Pooling to process and extract the features of related objects to get a new vector.

Finally, based on the R-CNN module, you will:

* + Classify the contents of the bounding box (or discard the bounding box, using background as a label.)
  + Adjust the bounding box coordinates to better use object.

#### Install the TensorFlow Object Detection API

On GitHub, the **TensorFlow** Object Detection API is stored in the **tensorflow/models** project. After downloading the **tensorflow/models** code, you should get a models folder. There is also a research folder in the models folder.

The following installation commands are all executed with the research folder as the root directory. The directories are also based on the research folder.

Compile the **proto** file with **protoc**. Specifically, you should run the Command 1 under the research file:

**protoc object\_detection/protos/\*.proto --python\_out=.**

Command Compile the proto file with protoc

After the operation is complete, you can check the **object\_detect/protos/** folder. If each proto file generates the corresponding python source code with the suffix **py**, the compilation is successful.

The TensorFlow Object Detection API is implemented on the basis of Slim. You need to add the **Slim** directory to **PYTHONPATH** to run correctly.

Specifically, still in the research folder, execute the following command:

**export PYTHONPATH=$PYTHONPATH:'pwd':'pwd'/slim**

Command add the Slim directory

After the execution of the command is complete, you can use the python command to open a python shell. If you run import slim successfully, it means that it has been set correctly.

Then, in the research folder, execute Command 3:

**Python3 object\_detection/builders/model\_builder\_test.py**

Command check if API has been installed

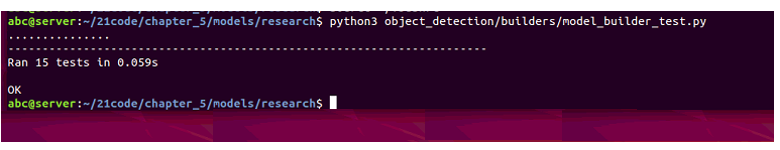


Figure Screen of successing

This command will automatically check if the TensorFlow Object Detection API is installed correctly. If the Figure 16 Screen of successingappears, the installation is successful.

#### Executing a model that has been trained

In this project we use the **Faster RCNN+Inception-ResNet** model that has been trained in the **COCO** dataset. In the Load Model section, you can try different detection models to compare speed and accuracy. Put the image you want to detect into **TEST\_IMAGE\_PATHS** and run it.

We can see Figure 17 from the demo test.

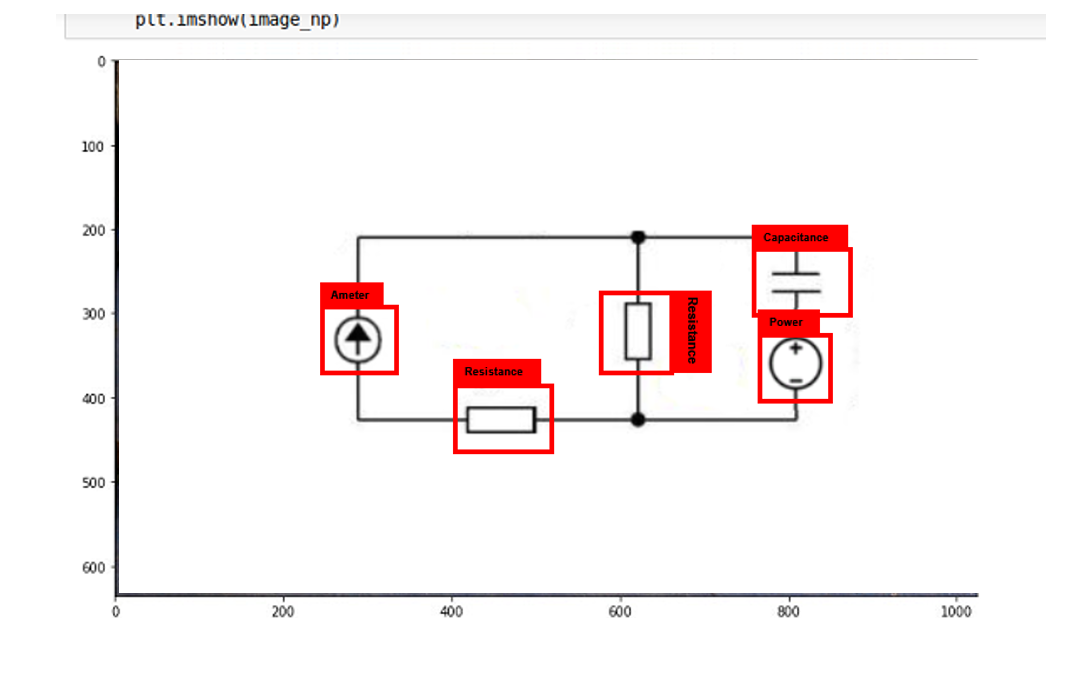


Figure Output screen

The TensorFlow Object Detection API is trained on a special setup file. There are some examples of setup files in the **bject-detection/samples/configs/** folder. Set the conditions created with reference to the **faster-rcnn\_inception\_resnet\_v2\_ atrous\_pets.config** file.

First copy the **fast\_ rcnn-inception\_ resnet\_ v2\_atrous\_pets.config** to the **voc** folder and name **voc.config**.

Then, create a new **tram\_dir** in the **voc** folder as a directory to save the model and logs. Start the training with the following command:

Both the training log and the final model are saved in **train\_dir**, also, **TensorBoard** can also be used to monitor training. Execute the Command 4 Train command

**--python3 object\_detection/train.py**

**--train\_dir object\_detection/voc/train\_dir/**

**--pipeline\_config\_path object\_detection/voc/voc.config**

Command Train command

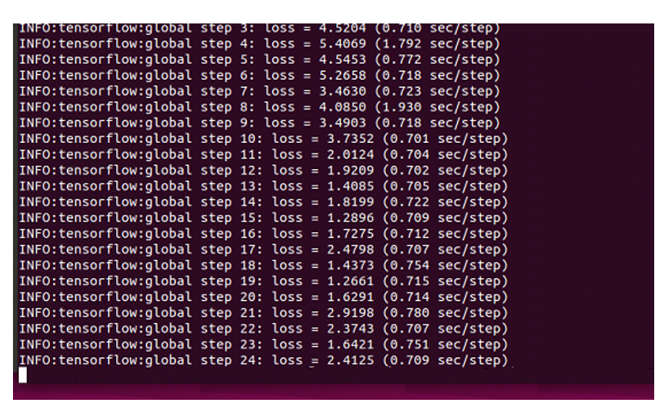


Figure Log output

In the end, the results Figure 18 is shown on the screen and record in log. (Sun, 2018)

#### Evaluation

Although, after google open sourced TensorFlow, with the pre-training model of Faster RCNN+Inception-ResNet, target detection has become very fast and simple.

However, when deploying to the mobile terminal, due to the relatively chaotic ecological environment of the Android app, the compatibility between Android under different versions is very poor, which causes many problems when deployed to the mobile terminal.

Mainly for the following reasons

1. There is no optimization model, which causes the model itself to be too large. It takes too much memory to deploy to the mobile terminal, and it runs at a high cost.
2. Incompatibility between different versions, resulting in software crashes and bugs frequently
3. Different from the pc end test, (the photo is completely horizontal), there is a viewing angle error when the mobile phone is photographed, resulting in a decline in recognition performance, and the accuracy is generally

In summary, we need to solve these problems in other ways. Later when I search for the solution the information, I learned about the PaddlePaddle framework, which can easily help us solve these problems. Detail is following.

### Deep learning with PaddlePaddle (baidu, 2019)

The biggest feature and positioning of the PaddlePaddle framework is easy to use, so many algorithms are completely encapsulated. It has a rich library to call and use, and is designed for specific technical scenarios. Developers may only need to understand the source code principle slightly, follow the example of the official website to execute the running command, replace it with their own data, modify some hyper parameters, call related api to achieve related functions. In the implementation of our project, it is more convenient, faster, and suitable than TensorFlow.

The process of PaddlePaddle is shown as Figure 19.

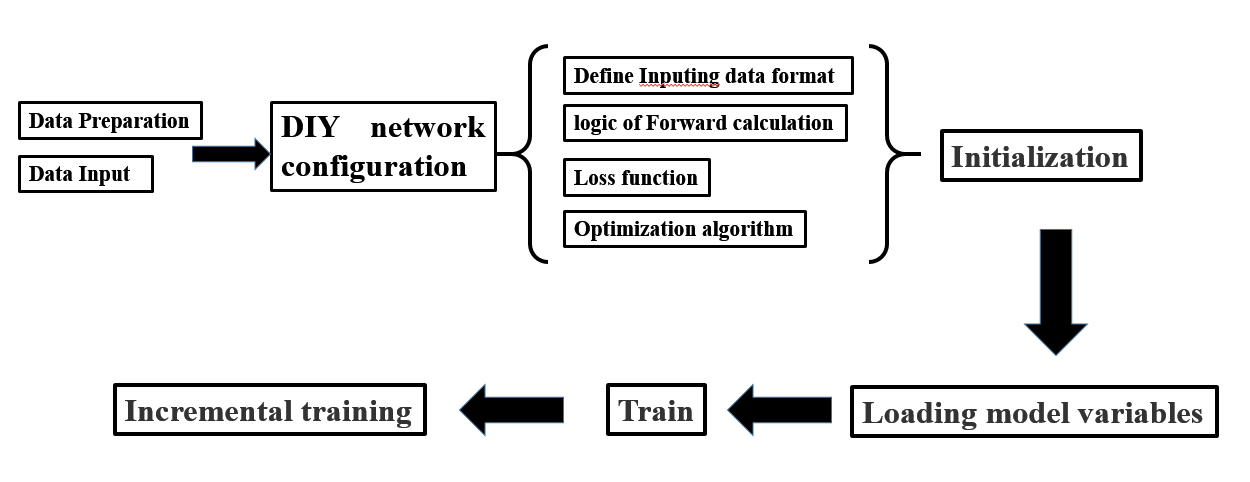


Figure Process of PaddlePaddle

#### Data Preparation

* **Data Preparation**

The generated data type can be **Numpy Array** or **LoDTensor**.

According to the different data forms returned by Reader, it can be divided into Reader and Sample Reader at Batch level.

The Batch-level Reader returns a batch of data each time, and the Sample-level Reader returns the data of a single sample each time.

In this project, we use **Python Reader** to preprocess batch and build batch

* **Data Input**

First configure the data input layer with f**luid.layers.py\_reader** and then configure the data source using **py\_reader's decorate\_paddle\_reader** or **decorate\_tensor\_provider** methods, then read the data via **fluid.layers.read**\_file .

In this framework, data entry is asynchronous with the model training/prediction process, with more efficient.

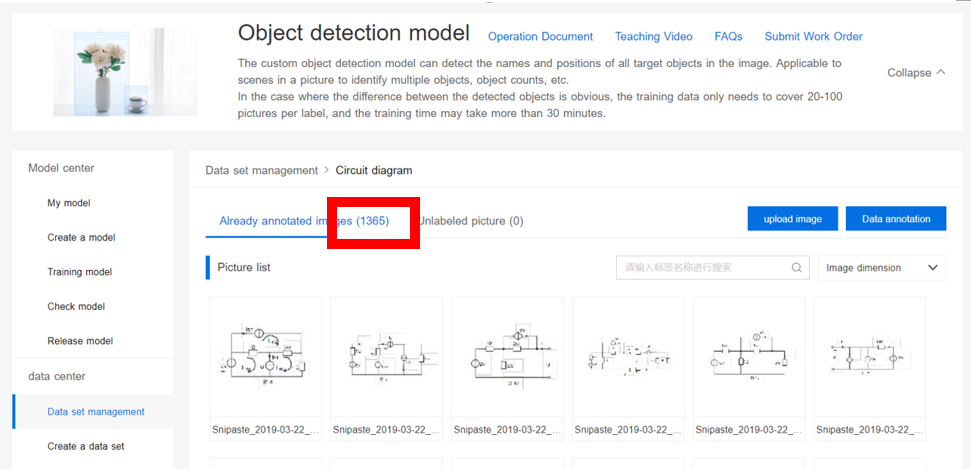


Figure Train sets

As Figure 20 shown, I use the training set framework of paddlepaddle, as shown in the figure. A total of 1365 training sets are used and input into the configured network to train the target model.

#### DIY Network Configuration

When solving practical problems, we must model the problem from the logical layer and clarify the model. There are four main parts when modelling for target recognition. Below we explain with sequence

* **Define Inputting data format**

PaddlePaddle provides the **fluid.layers.data()** operator to describe the format of the input data.

The output of the **fluid.layers.data()** operator is a Variable.

The actual type of this Variable is Tensor.

Tensor has powerful representation capabilities that can represent multidimensional data.

* **Logic of Forward Calculation**

Paddlepaddle uses **LSTM/GRU** and other operators to implement image-related tasks

* **Loss Function**

The loss function corresponds to the solution target, and we can solve the model by minimizing the loss.

The loss function used by most models, the output is a real value.

But the loss operator provided by PaddlePaddle is generally calculated for one sample.

When inputting a batch of data, the output of the loss operator has multiple values, each value corresponding to the loss of one sample, so usually the operator such as mean is used after the loss operator to reduce the loss.

The model will get a loss value after a forward iteration. PaddlePaddle will automatically execute the chain derivation rule to calculate the gradient value corresponding to each parameter and variable in the model.

Command 5 is used to calculate the mean square error loss:

**cost = fluid.layers.square\_error\_cost(input=y\_predict, label=y)**

**avg\_cost = fluid.layers.mean(cost)**

Command mean square error loss

* **Optimization algorithm**

After determining the loss function **(6)**, the loss value can be obtained by forward calculation, and then the gradient value of the parameter is obtained by the chain derivation rule.

After obtaining the gradient value, we need to update the parameters. In this project, we use the random gradient descent method.

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

#### Initialization

When the simple network is configured, you can get two **fluid.Program**, **startup\_program** and **main\_program.**

By default, **fluid.default\_startup\_program()** and **fluid.default\_main\_program()** can be used to get the global **fluid.Program**.

After the user configures the model, the parameter initialization is written to **fluid.default\_startup\_program().**

Run this program with **fluid.Executor()** to randomly initialize parameters in global **fluid.global\_scope()**

#### Loading model variables

By calling the **fluid.io.load\_params** function, PaddlePaddle Fluid will scan all the model variables in the prog, filter out all the model parameters, and try to read and load them from **param\_path.**

#### Train

Run the training **fluid.Program** using the **run()** method in **fluid.Executor()**

#### Incremental training

At the end of the training call **fluid.io.save\_persistables** to save the persistence parameters to the specified location.

After the trained **startup\_program** is executed successfully by the Executor, **fluid.io.load\_persistables** is called to load the previously saved persistence parameters.

Continue training with the actuator Executor or ParallelExecutor.

## Character Recognition Module

### OCR

OCR (Optical Character Recognition) refers to an electronic device (such as a scanner or digital camera) that checks the characters printed on the paper, determines its shape by detecting dark and bright patterns, and then translates the shape into a computer by character recognition. At present, OCR technology is very mature, and many frameworks even have already trained OCR interfaces with high recognition rate.

In this project, I used keras combined with CTC (Connectionist Temporal Classification) to reproduce this function in the existing Github (https://github.com/keras-team/keras/blob/master/examples/image\_ocr.py, 2019), and integrated into the final program to implement the text recognition module.

### Kerbs

Keras is a deep learning modeling environment in Python with CNTK, Tensorflow or Theano as the back end.

Compared to other deep learning software such as Tensorflow, Theano, Caffe, etc., Keras has some significant advantages in practical applications. The main advantage is that Keras is highly modular and supports existing common models (CNN).

, RNN, etc.), more importantly, the modeling process is quite convenient and fast, speeding up the development speed

### CTC (Connectionist Temporal Classification)

Suitable for time series problems where the alignment relationship between the input features and the output tags is uncertain, the CTC can automatically optimize the model parameters and the boundaries of the aligned segments automatically and end-to-end. Figure 21 is the structure of CTC (Graves1, 2016)

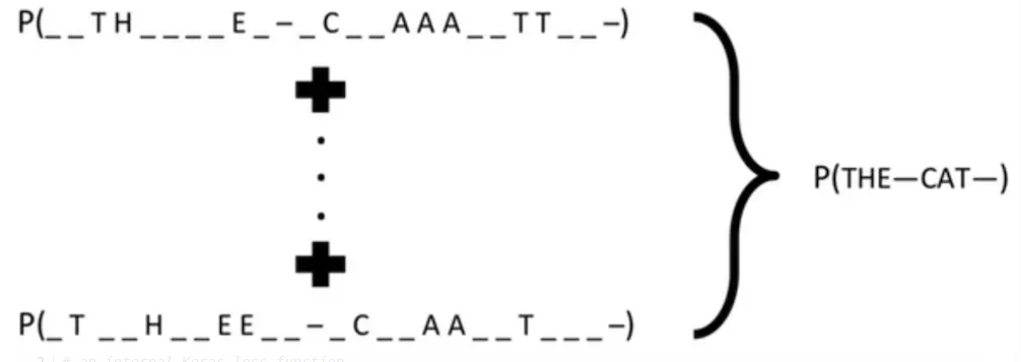


Figure CTC Structure

For example, a 32 x 256 image can be divided into 256 columns, that is, the input feature has a maximum of 256, and the maximum length of the output label is 18, which can be optimized by the CTC model.

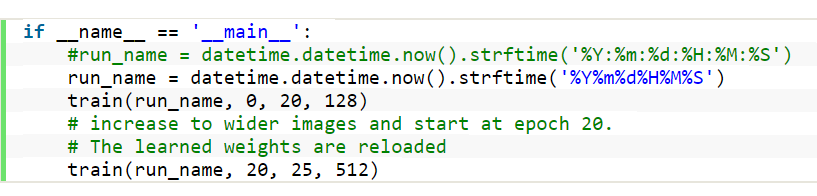


Figure Execute Code

Defines the number of training starts and ends, as well as the width of the image.

The functions Figure 22 mainly include input parameters and neural network parameters, generating training samples, network structure, optimization methods and training.

Among them, the minibatch\_size is adjusted to check whether the whole network can run normally with less training samples, and the whole amount is adjusted to 1

Finally, combined with the previous text recognition module, with fuzzy matching.

Finally realize the function of understanding the problem.

## App Design

### Overview

I use Android Studio to build the most basic UI interface, including registration login, search questions, question bank additions, component identification, and algorithm recommendations.

By calling Camera api from Android to get the target image, then use the OCR module to convert the target image information into a string and store it in local sql, and use sql's fuzzy matching algorithm to match the target answer.

In addition, combined with the image recognition module of paddlepaddle, the circuit components of the circuit diagram can be directly identified by this app.

Finally, according to the user's browsing frequency and the relationship between the questions and the questions, the app can automatically set the topic more suitable for the user.

### Encapsulation

Compared with the PC, the computing power of the mobile device is usually weak, and the CPU of the mobile terminal needs to maintain the power consumption index at a very low level, which brings constraints to the improvement of the performance index.

As a sub-project under PaddlePaddle, Paddle-Mobile is dedicated to deep learning prediction for embedded platforms.

The training task is carried out by PaddlePaddle on the server side, and Paddle-Mobile breaks the barrier of deep learning to land the embedded mobile platform. Figure 23 is the structure of paddle.mobile (https://github.com/PaddlePaddle/paddle-mobile, 2015)

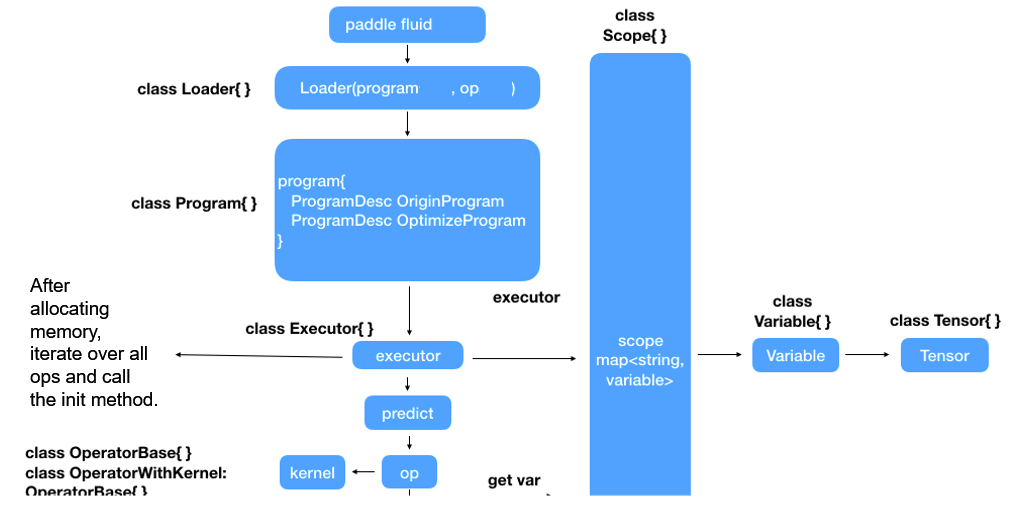


Figure paddle.mobile Structure

#### Loader

The role of the loader module is to load the model structure information Figure 24 into memory, load the protobuf file in the red box into memory, and optimize the model structure (such as merging several fine-grained ops into coarse-grained ops, such as conv, add, batchnorm, relu are merged into conv\_add\_batchnorm\_relu). Easy to optimize the algorithm

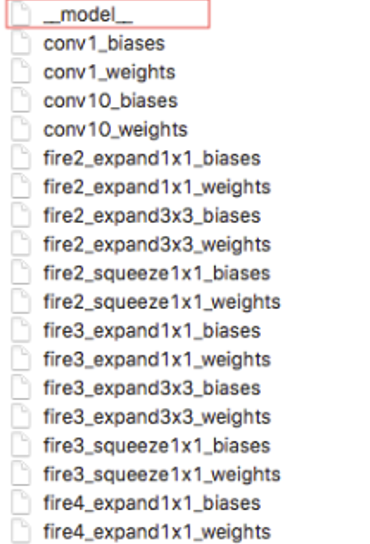


Figure model direction

#### Program

Program is the result of the loader module, including the model structure object before optimization, and the optimized model structure object

#### Executor

The executor is mainly used for the upper-level scheduling operations of op operations. There are two main operations, the executor is instantiated and exposed to the upper-level predict method.

During the executor instantiation process, these operations are mainly performed.

Initializes the operator object based on the program produced by the loader

All the memory that needs to be used is allocated, including the input and output of each op, the weight parameter. The memory format of the current model's weight parameter file is NCHW, and the input and output intermediate matrix parameters of op are also NCHW format.

Calling the init method of each op, the init method is where each op implementor takes care of the parameters, helping to reduce the time-consuming of predict

Predict, mainly used to get external input, sequentially call op's run method to perform operations, and return the final result

#### Op

The operator mainly contains a kernel for operations of storing properties.

The kernel Figure 26 is the underlying operation implementation of op. There are two main functions, Init and Compute, which are used for initialization, pre-processing, and arithmetic operations, respectively

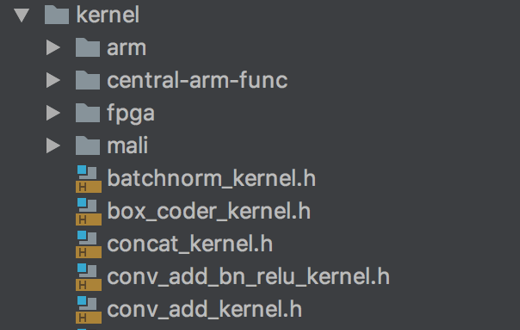
 

Figure 25 App presentation

Figure kernal direction

After deploying the project, we run the program, shown as Figure 25.

# Results and Discussion

## Function Demonstration

### LogIn/Register Functions

The login function is the basis of a software. It can recommend different content for different users. In this app, the user must log in before using the subscribe functions.

In the registration interface, the username must be greater than 4 characters to be registered successfully. The same password must also include a combination of letters + numbers, and a second verification is required. The LogIn/Register Page is shown as Figure 27.

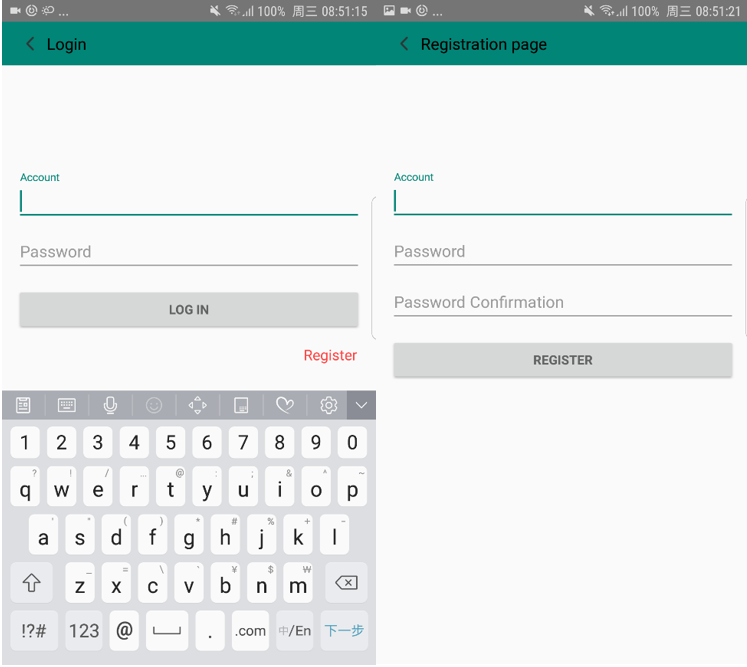


Figure LogIn/Register Page

F

### Photo Search

At the core of this part of the software, the text recognition module and the image recognition module are connected in series, the image is acquired by the camera of the mobile phone, and then the current text is obtained by OCR technology, and then the background blur recognition matches the answer, and finally output to the screen as shown as Figure 28.

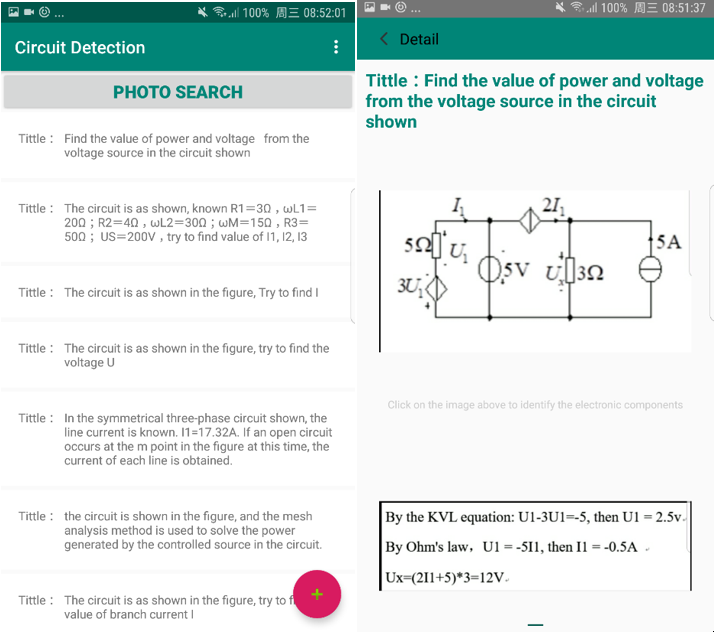


Figure Photo Search and result Page

### Circuit Components recognition

After searching for the answer to the desired question, the user can directly click on the image to identify the circuit component, and the software could adjust the precision ratio, to get different the result of the recognition, as Figure 29 shown.

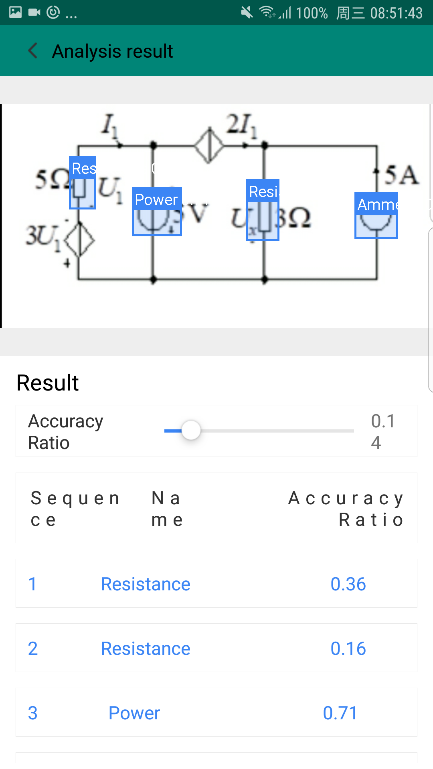


Figure Recognition page

### Add, modified and delete

In addition, the software can add, delete and modify question bank shown as Figure 30 files to enrich problem solving efficiency and success rate.

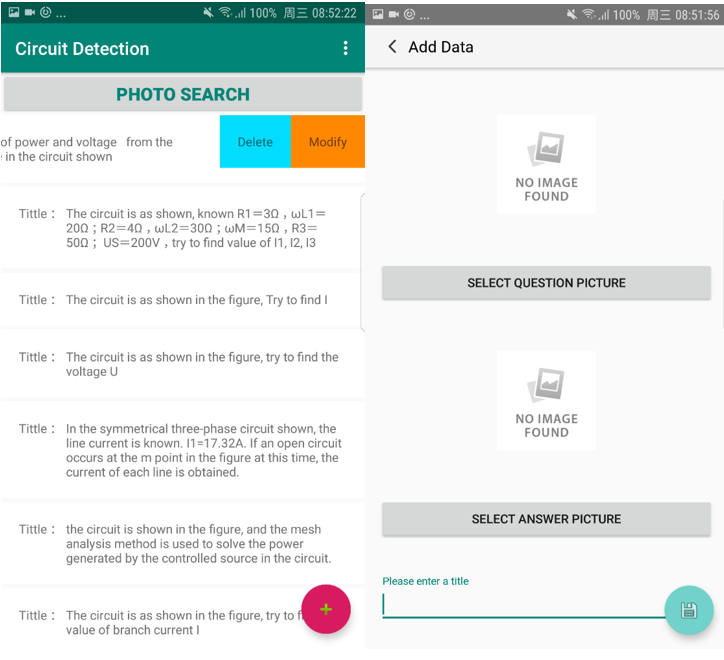


Figure Add, modify, delete page

### Recommendation algorithm

Finally, the software can automatically recommend the content that the user is more interested in according to the relationship between the frequency of the user's browsing topic and the topic. Based on the user's viewing frequency for each question. Record to “view heat” in the background, and recommend unfamiliar knowledge to the user according to the relationship between the questions and the questions, help the users learn the circuit subject and improve the practicality.

## Evaluation

Target detection requires both task targeting and target recognition. The correctness of the target location is determined by comparing the degree of overlap between the predicted border and the ground truth border and the threshold value; the correctness of the target recognition is determined by comparing the confidence score with the threshold.

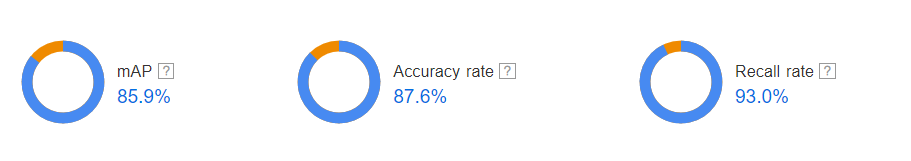


Figure Accuracy Rate

By above figure, we can get the final mAp is 85.9%, and accuracy rate is 87.6%, Recall rate is 93%, which are all shown as Figure 31. According the **(5),** we can easy to calculate the map and F1-Score as Figure 32 show.

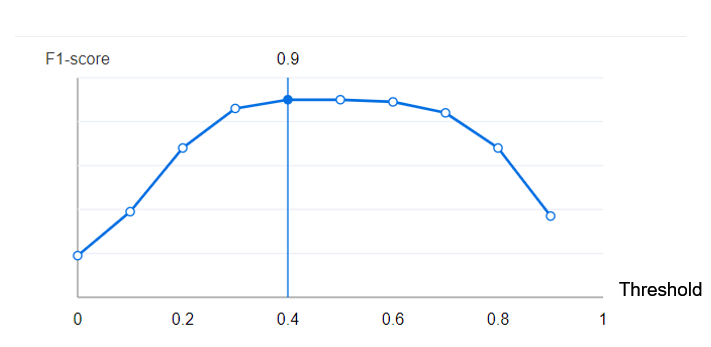


Figure F1-score

In the above figure, we can clearly see that the model can maintain a relatively high accuracy when the threshold is 0.4, but the accuracy is lower as the threshold increases.

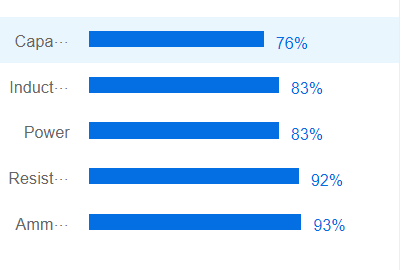


Figure Label accuracy

When different thresholds are set for processing, the F1-score is the highest when the threshold is 0.40, and the identification rate of each label by Figure 33 is

**Resistance: 92%.**

**Ammeter 93%.**

**Power 83%**

**Inductance 83%**

**Capacitance 73%**

**And, the whole mAp is 85.9%**

Above all, although the whole mAp is only 85.9%, as my data set continues to add, I believe that the overall accuracy will continue to improve.

# Conclusion and Further Work

## Conclusion

In this project, I mainly completed the identification of circuit components in the circuit diagram to help beginners quickly learn the basic knowledge. In addition, the project also provides the function of searching for questions, using OCR technology for text recognition, and identifying results and question bank for fuzzy matching. Under the premise of corresponding questions, the user can directly search for the title by using the camera function, and the app will return the corresponding answer to help the beginner learn better.

For the recognition module, I have used the traditional image recognition HOG+SVM single object feature extraction classification method. Then because traditional image recognition methods are difficult to identify logical relationships, I abandoned the traditional method and used deep learning.

Under the TensorFlow framework, combined with the Faster-R-CNN network, target classification and coordinate regression are performed by adding the RPN network to achieve target recognition and location. In the end, in order to solve the terminal use, the TensorFlow framework was abandoned, and the mobile api of the paddlepaddle framework was used to deploy to the terminal.

Then use the existing api to implement the final image recognition module.

For the OCR module, the OCR function on the generated text image is implemented by the existing api in the paddlepaddle combined with the CTC model. After that, it is fuzzy matched with the data in the question bank, and the corresponding answer is given. According to the above steps to achieve the final problem-solving function,

Finally, the project also has a topic recommendation function. According to the user's frequency of viewing each topic, the background records the frequency of viewing, and according to the relationship between the title and the title, the user is recommended to the relevant unfamiliar system to help the beginner learn better, thereby improving efficiency.

## Further Work

Admittedly, the function of the project has been basically completed, and there are still many places to improve.

For example,

* Enrich the question bank and improve the success rate of problem solving.
* Continue to train the recognition module to improve the recognition rate of electronic components
* Improve the recommendation algorithm to improve the recommended success rate
* Introducing search records and tampering records

I will continue to improve and improve the related functions.

References

baidu, 2019. *PaddlePaddle Instruction.* [Online]   
Available at: http://paddlepaddle.org/documentation/docs/zh/1.3/user\_guides/howto/configure\_simple\_model/index.html#user-guide-configure-simple-model  
[Accessed 18 3 2019].

Graves1, A., 2016. Connectionist Temporal Classification: Labelling Unsegmented. *icml*, 8 7, pp. 2-3.

*https://github.com/keras-team/keras/blob/master/examples/image\_ocr.py* (2019).

*https://github.com/PaddlePaddle/paddle-mobile* (2015).

Javier, 2018. *Faster R-CNN: Down the rabbit hole of modern object detection.* [Online]   
Available at: https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/  
[Accessed 23 4 2019].

Michaelliu\_dev, 2018. *Faster R-CNN article detailed interpretation.* [Online]   
Available at: https://blog.csdn.net/liuxiaoheng1992/article/details/81843363  
[Accessed 19 4 2019].

Michi, 2018. *[Target detection basis] Faster Rcnn framework principle analysis.* [Online]   
Available at: https://blog.csdn.net/wangpengfei163/article/details/80961275  
[Accessed 19 4 2019].

Navneet Dalal, B. T., 2005. Histograms of Oriented Gradients for Human Detection. *CVPR*, 12 5, pp. 1-2.

Slyne\_D, 2017. *Image learning - HOG features.* [Online]   
Available at: https://www.jianshu.com/p/395f0582c5f7  
[Accessed 17 04 2019].

Sun, T., 2018. *R-CNN, Fast R-CNN, Faster R-CNN principle and implementation and training examples + achieve your own target detection.* [Online]   
Available at: https://blog.csdn.net/m0\_37407756/article/details/80810364  
[Accessed 12 3 2019].

Wikipedia, 2019. *Support-vector machine.* [Online]   
Available at: https://en.wikipedia.org/wiki/Support-vector\_machine#Definition  
[Accessed 21 4 2019].

XIAO Dou, H. X., 2016. Electrical Symbol Recognition in Circuit Diagram Based on PHOG Feature. *Ship Electronic Engineering*, 12 7, pp. 3-4.

Acknowledgement

This project would not have been possible without help from my classmate and senior. I am especially indebted to supervisor, who have been supportive of my project and who gave me the direction and advice actively to help me to achieve the goal.

I am grateful to all of those who gave me the help during the projects. Each of them has provided me extensive personal and professional guidance and taught me a great deal about the final app design.

Nobody has been more important to me in achieving of this project than the supervisor. I would like to thank him, whose gave lots of advice and direction when I am confused. Thank again for those who gave me the help and advice during process of project, there would be no possible without their help.

Appendix

Risk Assessment

This project is only about software development, which will not lead to any risk or dangerous effects.

Environmental Impact Assessment

This project is just about software development, there will be no impact or damage to the environment.