Jiayu Xie

2016/11/27

G53IDS Interim Report

Deep learning for playing card detection

1. Introduction

1.1 Project brief

Deep learning is a very powerful and popular approach in machine learning, In recent year, deep learning get dramatic output in many fields like image recognition, image segmentation, image synthesis, voice recognition, voice synthesis, language translate, memory, automatic reasoning. For computer vision filed, convolutional neural network is widely used in deep learning. The aim of this project is to build a app to recognize the playing cards, using convolutional neural network as the main tool. The project outcome can be used in playing cards game, and can find a good network to extract features of playing cards.

1.2 Interpretation of the problem

The intent of this project is to develop an mobile application to recognize the playing cards, but the calculate ability of mobile device is not enough to compute, so need a server to receive the image from mobile application and process the image and send the result back to mobile application, then the mobile application show the result. Because this project use machine learning as the main tool, so it need to collect data, preprocessing data, store data and retrieve data. For more advanced performance, the server need to segment the cards from image.

2. Related work

Image recognition is an important subject of computer vision, and it been dramatic improved in the recently few years and can work in the industry because of the improvement of deep learning.

The difficult part of traditional computer vision is how to extract feature from images because it depends on human, but it changed in 2012, Alex using convolutional neural network to win the champion of ImageNet competition 2012[2], which means a new framework of computer vision is better than traditional methods, and make deep learning be the super tool for Image recognition. After 2012, people keep working on convolutional neural network and obtain dramatic outcome. Microsoft win champion of ImageNet competition 2015, it still uses an improved convolutional neural network and the accuracy higher than human level.

In industry, convolutional neural network is using widely, like the Photos application in Mac can find different face and classify each photo contain the face into the person's class, or helping doctor to diagnose illness like cancer[1]. DeepMind using deep learning reducing Google data center power consumption by 40 percent.

Neural network is a model to match the mapping from input to output, it can learning from data, and learning the implied rule of the data automatically.

Convolutional neural network in fact is to use convolution to extract features from images, and then using traditional neural network(full connect) to classify.

1). LeNet5[3]: This may be the first time using convolutional neural network in image classification and many successful architectures are base on this design. This architecture has 7 layers, first 4 layers are convolutional layer(discrete convolution layer + pooling layer) and the rest layers are full connect layers. Show in figure 2.1.1.

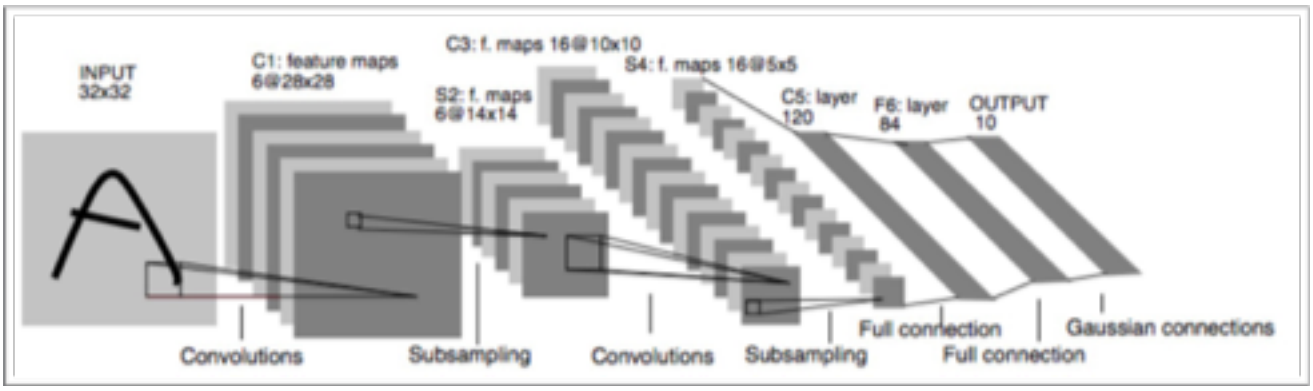


figure 2.1.1

Discrete convolutional layer: a vector is received as input and is multiplied with a matrix to produce an output, for example, we have a matrix kernel like figure 2.1.2, then calculate the convolution like figure 2.1.3.[4]

0	1	2
2	2	0
0	1	2

figure 2.1.2

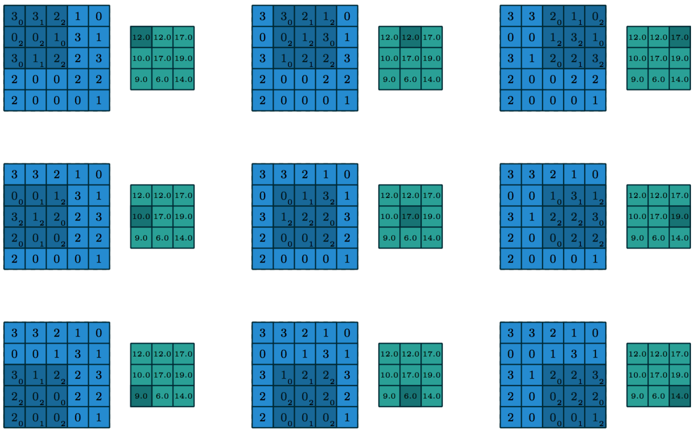


figure 2.1.3

pooling layer: pooling operations make up another important building block in convolutional layer. Pooling operations reduce the size of feature maps by using some function to summarize subregions, such as taking the average or the maximum value. The processing of pooling show in figure 2.4[4].

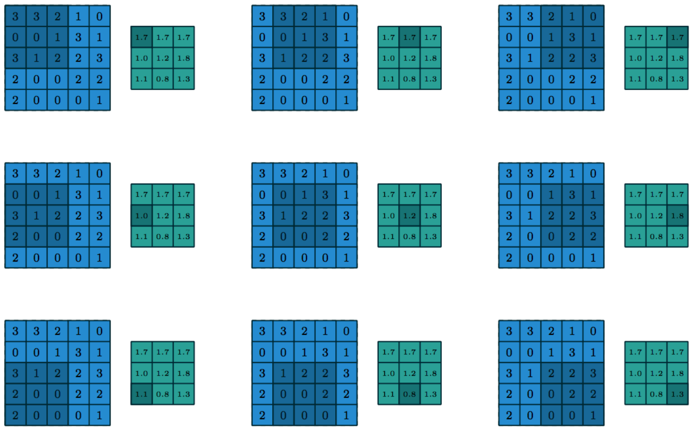


figure 2.1.4

2).AlexNet[5]: Time change to 2012, Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton create a new architecture won the 2012 ImageNet competition and the accuracy rate considerably better than the previous state-of-the-art. Architecture show in the figure 2.2.1. This architecture expand LeNet5 and use new activity function which make the deep neural network training become possible(eliminate gradient vanish), and it's using five convolutional layer at first then combine with 3 full connect layer. The dramatic success of AlexNet let many people focus on the convolution neural network.

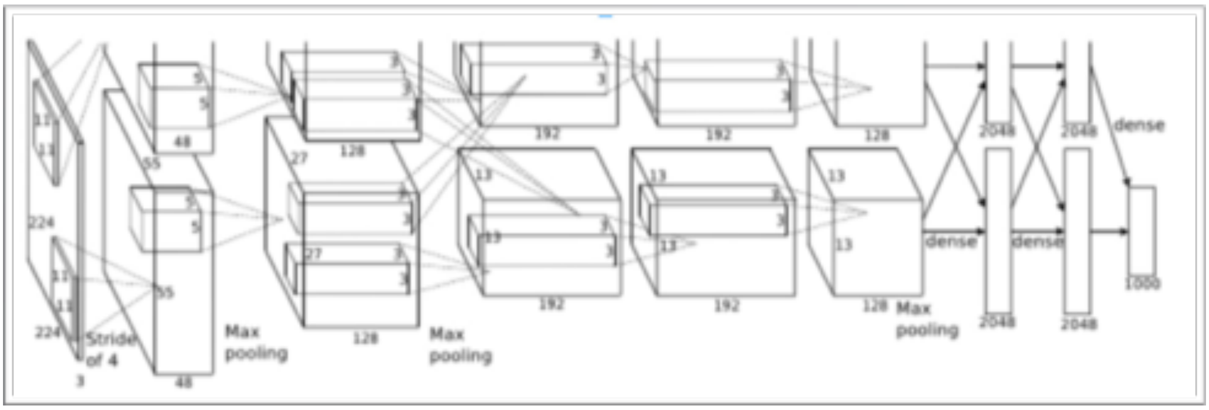
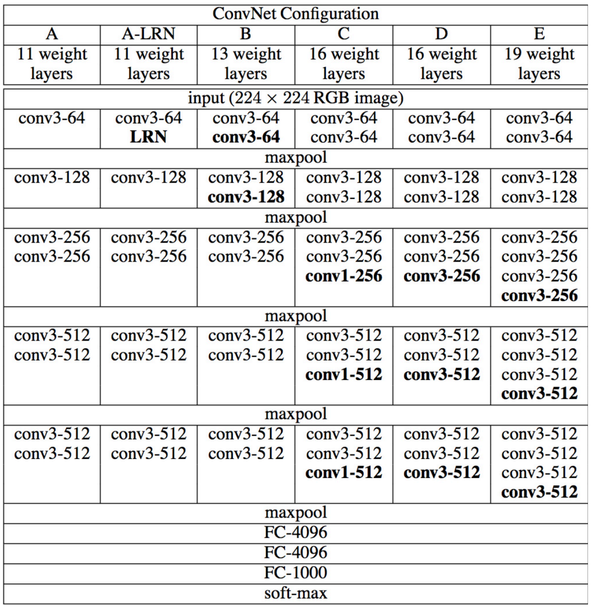


figure 2.2.1

3). VGG[6] and GoogleNet[7]: Both of these two architectures get higher accuracy rate(figure 2.3.3) than other architecture in 2014 ImageNet competition. Both of the them going deeper compare to AlexNet. VGG show in figure 2.3.1 and GoogleNet show in figure 2.3.2.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

figure 2.3.1

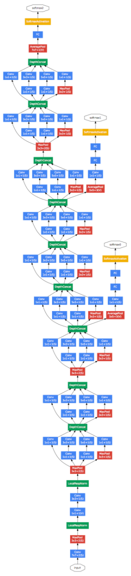
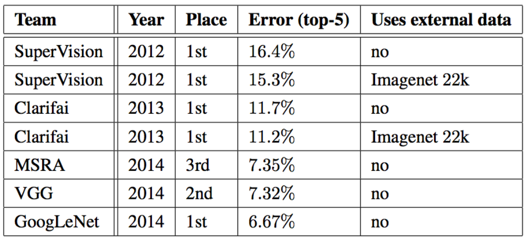


figure 2.3.2



Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

figure 2.3.3

4). Other than classification, object detection and segmentation is another important field in computer vision and also changed by convolutional neural network. Before R-CNN[8] object detection performance on PASCAL VOC dataset has stack into a bottleneck, but R-CNN use convolutional neural network to get a breakthrough, improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012. DeepMask[9] get higher performance than R-CNN on VOC 2007 and it's work on pixel level.

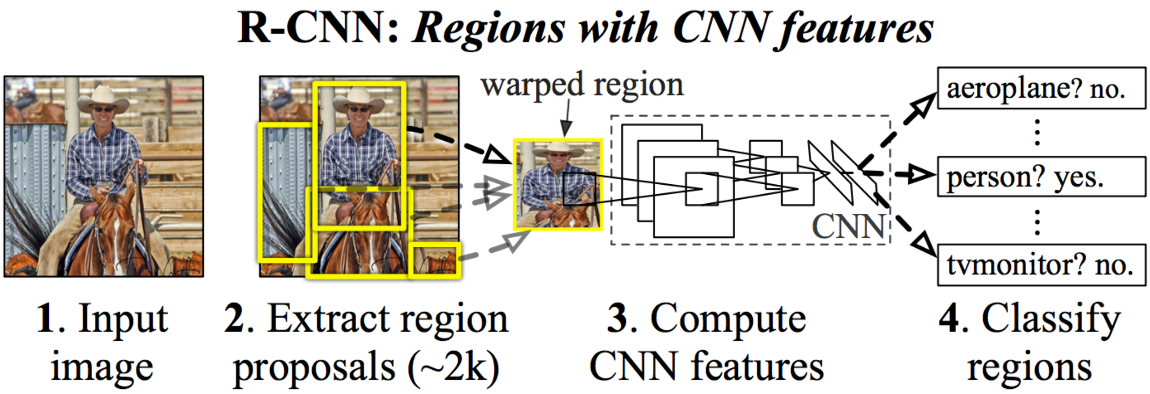


figure 2.4.1 R-CNN

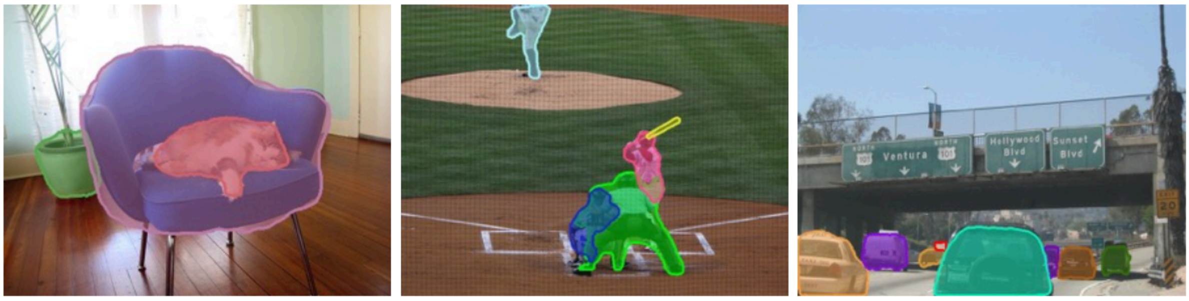


figure 2.4.2 DeepMask

3. Description of the work

The project is using deep learning to recognize the playing cards, so at first need to choose which architecture will be used. At first I chosen VGG-A model, it is easy to understand and easy to change to satisfy the need of classify cards. All machine learning algorithms need data, and deep learning is no exception, even need more. An easy way to collect image data is to slice videos, using this method can collect image data quickly. The idea is intercepting a frame every few frames and changing the size of the image and saving the image to the corresponding folder. Then going to train and save the models, after that an online server is needed to classify photo realtime. Finally is to develop the mobile application.

4. Design

Basic framework of the project has 3 part, Client, Online Server and Offline Server. Client work for users with using it to take photo and show messages. Online Server classifying and segments photos, and sends back results. Offline Server used to train models of deep neural network, and preprocess the collected data.

1. Client part: Application on the smart phone.

requirements:

Take a photo and send photo to server.

record video and send video to server.

receive message(text or image)from server.

2. Online Server part:

requirements:

Create a host on the computer that can be accessed external.

Control route.

Load deep neural network's model.

Receive photo from Client.

Predict the result of photo.

Send the result back.

3. Offline Server part:

requirements:

Create a host on the computer that can be accessed external.

Control route.

Receive videos and store the videos.

Preprocess the videos into satisfied train/validation/test dataset.

Train and save the model

5. Implementation

1. Preprocessing video data:

   1.1. Matlab be chosen to slice the video because it's easy to process video, it do not need to consider the problem of decode the video, and get frames easily.

   1.2. Function definition: slice(file_path,class,store_path,video_use,width,height), The only function in Matlab to slice the video, it store the image into ***store_path/class/*** folder. Different ***video_use*** split images into different datasets, for example, 'train_validation_test' split the video into three datasets, train is first 70% frames of video, next 10% store as validation, the rest 20% are test.

      1.2.1. ***file_path***: where is the video.

      1.2.2. ***class***: what is the class of card in video.

      1.2.3. ***store_path***: where to store the video.

      1.2.4. ***video_use***: using the video as 'train_validation_test' or 'train' or 'validation' or 'test' or 'validation_test'.

      1.2.5. ***width***: output image width.

      1.2.6. ***height***: output image height.

   1.3. Using php and apache to build a server, and Laravel be used as the web framework. The server can handle two post request.

      1.3.1. /slice: need post video, class, video use, width and height, this page can receive data and call Matlab to slice the single video.

      1.3.2. /slice_batch: need to post a path which stores videos, and the name of video is the class of card in that video, this page can call Matlab several times to process a lot of videos at a time.

2. Store and restore dataset:

   2.1. Store dataset: data store as ***store_path/class/train*** , ***store_path/class/validation*** and ***store_path/class/test***. For example, 3_diamond store in ***store_path/3_diamond/train***, ***store_path/3_diamond/validation***, ***store_path/3_diamond/test,*** and each folder contain many images.

   2.2. Restore dataset: because using python and tensorflow to train models, so need to use python to restore data from file. First scan and store every image file name into a list, then using the list as the input feed in the class ImageReader, which will read the image and do some preprocess at the train time. For stochastic gradient descent, ImageReader will read train image as batch randomly, and convert them into matrix with N*(W*H), N represent the number of images in the batch, W represent width of image and H represent height of the image.

   2.3. Store like this is easy to management the dataset, and restore like this will only use little computer memory compare to load all the images into memory at the a time.

3. Model:

3.1. Model configuration: Because VGG-A architecture is easy to understand and easy to change, I build the model base on that. VGG-A model has 5 convolutional layers and 3 full connect layers, and in the third, fourth, fifth convolutional layer contain 2 discrete convolutional layer, and every convolutional layers followed by a max pooling layer. Because the limitation of train data and number of cards, the whole VGG-A is not necessary, a simpler VGG-A model is chosen, show in table 5.3.1. Here only using 4 convolutional layer and two full connect layer.

conv3 3-64
max pooling
conv3 64-128
max pooling
conv3 128-256
conv3 256-256
max pooling
conv3 256-512
conv3 512-512
max pooling
FC 1024
FC 26
softmax

table 5.3.1 The configuration of model.

3.2. Learning:

3.2.1. Softmax: Z is a vector as the output of full connect layer, K represents the

$$p_j = \sigma(\mathrm{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

length of vector Z, the output of softmax shows the probability of each entry in

$$\frac{dp_j}{dp_i} = p_i * (1 - p_i), \mathrm{i} = \mathrm{j}$$

$$\frac{dp_j}{dp_i} = -p_i * \ p_j, \mathrm{i} \ \neq \mathrm{j}$$

the vector. Assume softmax value is p , then the derivatives of softmax is show above, the property of derivatives make it's easy to calculate the derivatives of softmax function.

3.2.2. Cross entropy: the cost function using cross entropy, y is the one hot represent

$$C = \sum_{i=1}^{N} y_i * \log\left(\sigma(z)_i\right)$$

of the class, i.e. [1 0 0] means target is class 1, [0 1 0] means target is class 2. N represents the the length of vector y, $\sigma(z)$is the softmax function. The derivatives of cross entropy show below, where pi is $\sigma(z)$i and yi is corresponding entry in

$$p_i - y_i$$

vector y. The derivatives of cross entropy is simple and easy to calculate.

3.2.3. Learning function using Adam SGD[10], Adam is a adaptive approach to gradient descent, it can automatic change learning rate at the training time, and faster than traditional gradient descent. The size of mini-batch is 50. The input image size is 80*80*3.

3.3. Implement: all neural network build on library tensorflow[11], tensorflow work like a calculate pipeline graph, it using python to build the calculate graph, and run the whole calculate process after complete the graph. Tensorflow can automatic compute derivative of component in the graph, and has build-in gradient descent function.

3.3.1. There are two different components in graph. First one is Variable, which is the parameter that will be change during the training, Variable need to be initialization before run the graph. Second one is Placeholder, which is the component that receive data from outside of the graph.

3.3.2. Tensorflow can record every variable state at any time and show them in tensor board.

4. Online Server:

4.1. Using python to build a server because the model trained in python, and need using python to predict, so using python to build server is easy to implement. The library called web.py used to build server, which is a lightweight library and only need several line codes to build server and process get, post requests.

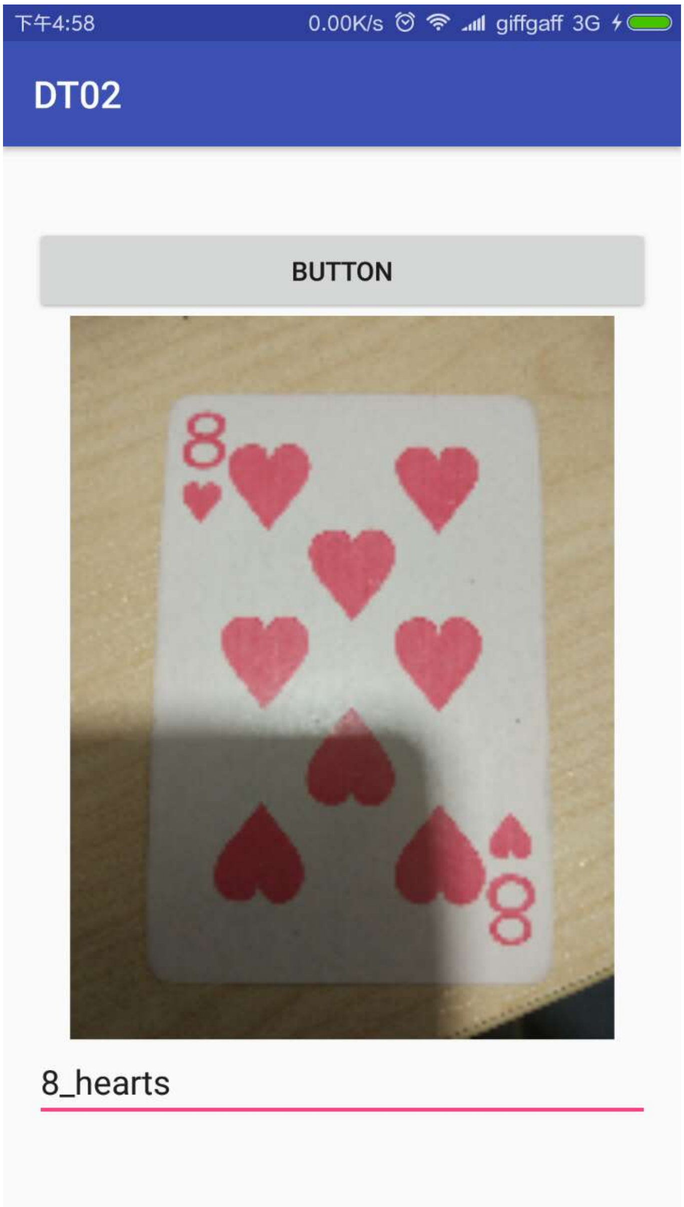
4.2. The code first load the model into memory, then use web.py to build server and wait for request.

4.3. When receive request, first, the transmitted image is stored, and resize the image to satisfy the neural network needed. Then convert resized image into matrix and send to neural network to predict. Finally send the result back to client.

5. Client

5.1. Client use android to build because it’s easy to learn and implemented.

5.2. The application has a button to click to take photo and send image to server, a image view to show the image and a text view to show the result of the classification. Show in figure 5.5.1.



8_hearts

figure 5.5.1

5.3. The application first call system camera to take photo, and then store the photo into disk, after that create a new thread to post data to server, finally using the callback function of thread to change the text view UI to show result.