CS655 GENI Project Report

Handwritten Digit Recognition Application

Ci Chu, Xiaotong Niu, Yuwan Xiao, Hexuan Zhang

GitHub Link:https://github.com/yizheshexin/CS655GeniProject
In the GitHub repo, server_final.py, index.html, back_cgi.py is the code file to build the application; client.sh is the shell script to run on client node; server.sh is the shell script to run on server node; instruction_on _how _to_reproduce application.pdf and working_demon_video are as their name; server_extension.py, back_cgi_extension.py are some extension works which to show our effort but consist of some trivial bugs. And several image are also provided for you to have a test.
GENI:https://portal.geni.net/secure/slice.php?project_id=d9040240-2587-4a8d-840e-9e2864b18b5d&slice_id=6364caa3-dd2f-44be-812c-59642beefb45&member_id=
Name of GENI slice: geni6

1. Introduction

In this project, we build an Online Handwritten Digit Recognition Application by implementing an image-recognition service which has a web interface, where a user would be able to submit an image of a hand-written digit and the service would use some of the image recognition techniques to classify the image and return the designated result. More specifically, we use the svm model from sklearn to deal with the image recognition part. The web interface and recognition systems are set on a separate GENI node and we connect them using python socket programming. For the web interface part, we use html and python cgi to finish their building ups.

2. Experimental Methodology

2.1. Web interface: we use html and python cgi to build our web interface. Once the image is uploaded to the index.html file, the back_cgi.py will store the image to a local path and then deal with the socket part. The result is then displayed in the web interface.
2.2. Client socket : The client use the socket programming technique to make sure the connection to the server is reliable. To deal with the distance between the client and the server, we can set the timeout parameter and see if the average RTT is large and tell if the client and the server are far apart from each other. In case of data loss or timeout, the client will keep retrying (for at most 5 retryings in case something happen with the network, the client will stop and don't waste network resources), retransmit the data, print out the result. If failed after 5 retryings, the client will see the message "fail", and can choose to retry if they want to.
2.3. Server socket: The server is designed to keep listening to requests from the client. It will accept data and then send it to the backend image recognition model for further steps. Once the backend model has concluded with a digit result, it will send the result to the

client. Moreover, the server is designed for multi-threaded clients. Each client will assign a unique ID and receive the result.

2.4: Digit Recognition: The digit recognition part makes use of svm model from sklearn, as stated above. The svm model training procedure starts with loading up training data provided by the sklearn library in python itself, containing 1797 samples of pre-stored hand-written digit images from 0 to 9. These images are then passed to the svm model which gamma amount is set to 0.001, regularization parameter is set to 10 and kernel function as rbf for better accuracy. After the svm model gets successfully trained, pictures uploaded by users are then transferred to the function of modifying dimensions so that the uploaded pictures have the same dimensions as the pictures trained in the model. After reducing dimensions, uploaded pictures are then loaded into this the svm model for recognition and returns the designated recognition result.

3. Results

3.1. Usage Instructions

Our web interface can be reached by typing http://204.102.244.76/ into the browser. In this page, the user can upload an image he/she wants into our system to recognize. If the user doesn't upload anything and click upload, our system will warn him/her that no picture has been uploaded and the user can click 'try again ' to go back to the original page.

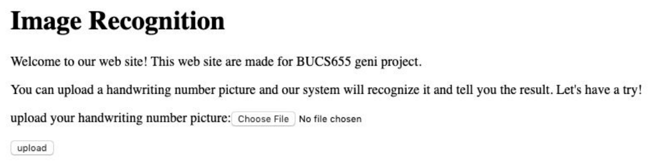
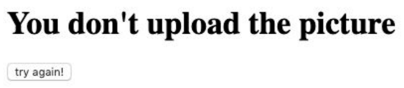


Image Recognition

Welcome to our web site! This web site are made for BUCS655 geni project.

You can upload a handwriting number picture and our system will recognize it and tell you the result. Let's have a try!

upload your handwriting number picture:

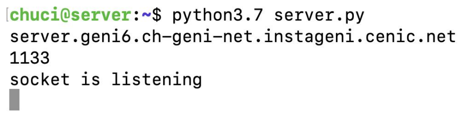


You don't upload the picture

After the user uploaded an image and clicked upload, the page will present the image that the user uploaded plus the result of image recognition. Here, the user can click try again to go back to the original page to upload another image. More recognition results will be shown in the below analysis part.

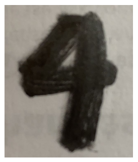
3.2. Analysis

On server side, once the program is executed, it keeps listening to the incoming connection:

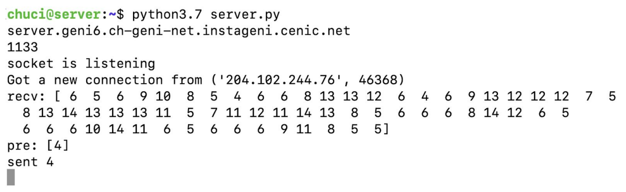


```
chuci@server:~$ python3.7 server.py
server.geni6.ch-geni-net.instageni.cenic.net
1133
socket is listening
```

On client side, a picture containing 4 is uploaded first:

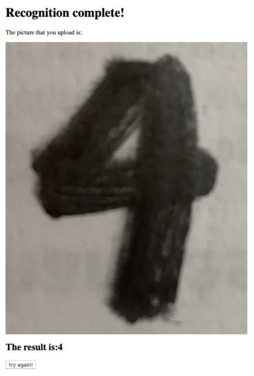


After the picture has been uploaded, we can see from the server side that data has been successfully received and recognition system has also produced the result and sent it back.



```
chuci@server:~$ python3.7 server.py
server.geni6.ch-geni-net.instageni.cenic.net
1133
socket is listening
Got a new connection from ('204.102.244.76', 46368)
recv: [ 6  5  6  9 10  8  5  4  6  6  8 13 13 12  6  4  6  9 13 12 12 12  7  5
  8 13 14 13 13 13 11  5  7 11 12 11 14 13  8  5  6  6  6  8 14 12  6  5
  6  6  6 10 14 11  6  5  6  6  6  9 11  8  5  5]
pre: [4]
sent 4
```

Finally, on our web interface, the result is displayed as following:

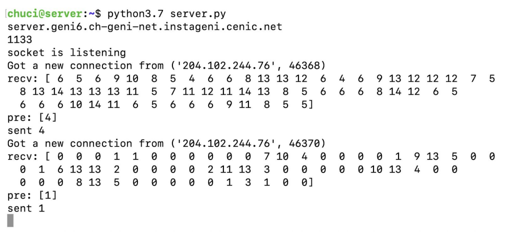


Recognition complete!

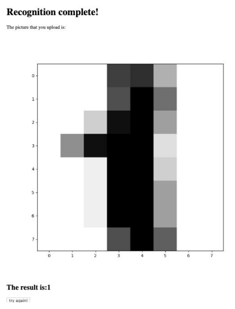
The picture that you upload is:

The result is:4

Then, another image containing digit 1 is uploaded:



```
chuci@server:~$ python3.7 server.py
server.geni6.ch-geni-net.instageni.cenic.net
1133
socket is listening
Got a new connection from ('204.102.244.76', 46368)
recv: [ 6  5  6  9 10  8  5  4  6  6  8 13 13 12  6  4  6  9 13 12 12 12  7  5
  8 13 14 13 13 13 11  5  7 11 12 11 14 13  8  5  6  6  6  8 14 12  6  5
  6  6  6 10 14 11  6  5  6  6  6  9 11  8  5  5]
pre: [4]
sent 4
Got a new connection from ('204.102.244.76', 46370)
recv: [ 0  0  0  1  1  0  0  0  0  0  0  7 10  4  0  0  0  0  1  9 13  5  0  0
  0  1  6 13 13  2  0  0  0  0  2 11 13  3  0  0  0  0  0 10 13  4  0  0
  0  0  0  8 13  5  0  0  0  0  0  1  3  1  0  0]
pre: [1]
sent 1
```

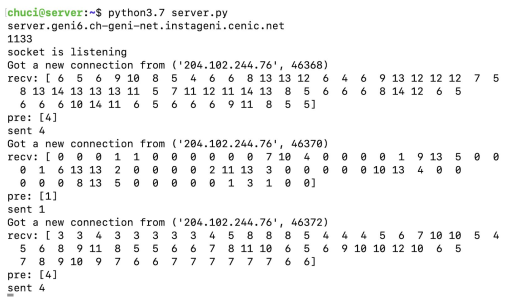


Recognition complete!

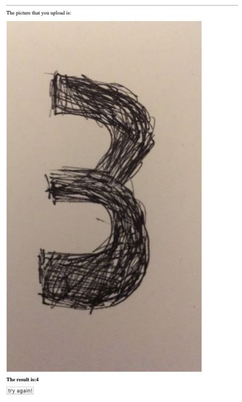
The picture that you upload is:

The result is:1

However, in this case our recognition system cannot guarantee 100% accuracy, here is an example of incorrect recognition:



```
chuci@server:~$ python3.7 server.py
server.geni6.ch-geni-net.instageni.cenic.net
1133
socket is listening
Got a new connection from ('204.102.244.76', 46368)
recv: [ 6  5  6  9 10  8  5  4  6  6  8 13 13 12  6  4  6  9 13 12 12 12  7  5
  8 13 14 13 13 13 11  5  7 11 12 11 14 13  8  5  6  6  6  8 14 12  6  5
  6  6  6 10 14 11  6  5  6  6  6  9 11  8  5  5]
pre: [4]
sent 4
Got a new connection from ('204.102.244.76', 46370)
recv: [ 0  0  0  1  1  0  0  0  0  0  0  7 10  4  0  0  0  0  1  9 13  5  0  0
  0  1  6 13 13  2  0  0  0  0  2 11 13  3  0  0  0  0  0 10 13  4  0  0
  0  0  0  8 13  5  0  0  0  0  0  1  3  1  0  0]
pre: [1]
sent 1
Got a new connection from ('204.102.244.76', 46372)
recv: [ 3  3  4  3  3  3  3  3  4  5  8  8  8  5  4  4  4  5  6  7 10 10  5  4
  5  6  8  9 11  8  5  5  6  6  7  8 11 10  6  5  6  9 10 10 12 10  6  5
  7  8  9 10  9  7  6  6  7  7  7  7  7  7  6  6]
pre: [4]
sent 4
```



The picture that you upload is:

The result is:4

The main goal of this project is to make us code with internet part(GENI, socket and apache) but not code with machine learning, so we think this is totally fine.
Besides these test cases, we have also provided several example images in the GitHub repo for further tries.

4. Conclusion

We have successfully built the application on the GENI nodes. On the client side, we enable apache server and python cgi, write a simple html page for user to upload the image. After the image is uploaded, it will be transferred to feature bytes and then be transferred to server node. Server node keeps listening to the incoming connection, and receive feature bytes, reorganize the bytes back to feature number, and use the svm model to do the prediction. Finally we send back the prediction result and present it on the client side's web page.

Some possible extensions are that we may try to use some more complicated machine learning model so that any image can be recognized with high precise, rather than just handwritten digits. To accomplish that, the socket part also needs to be modified since the whole image will need to be transferred rather than only the feature part. In fact, we do write an extension bonus part for using the complicated model and updated socket to deal with socket buffer in server_extension.py and client_extension.py which can be checked in the github repo.

5. Divisor of Labor

Ci Chu: Set up environment on geni node, install and enable apache and python cgi, write web interface and python cgi part, finally organize all codes together and make it run.
Xiaotong Niu: Set up the svm training model from sklearn, collect the machine learning training dataset, modify dimensions of uploaded pictures to be recognized, predict the image recognition result and improve accuracy after that.
Yuwan Xiao: Build up the socket of the client-side, and keep trying (at most 5 retryings) until the result is successfully received from the server. Prepare and send the data to featured bytes for the server side.
Hexuan Zhang: Build up the socket of the server-side, which will keep listening and accept multi-clients' requests. Prepare the data for the image recognition model and send out the result.

References

1. https://www.youtube.com/watch?v=ELFdP7eEZ5w&t=2s
2. https://www.youtube.com/watch?v=mBKU56uMQX4&t=4s
3. https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html#sphx-glr-auto-examples-classification-plot-digits-classification-py