Image Recognition Application
CS655 GENI Mini Project Report

Group members: Wei Jiang, Kaihong Wang, Ruikang Wang, Nianyi Zhang

GitHub Link: https://github.com/ZJAllenJiang/Image-Recognition-Application
GENI Link: backend node: proj-wj-node - weijiang@pcvm2-23.mcv.sdn.uky.edu
frontend node: Project-IRA - rkwang@pcvm2-4.instageni.colorado.edu

1. Introduction / Problem Statement

In this project, we need to implement a webpage based image-recognition program deployed on two nodes, including a client node and a server node. The client node should take care of the image uploaded from the user and query the server node which provides an interface that could be used to recognize an image and return the result.
Technically, we can break down the whole process into three parts: image receiving and querying at the client node, communication between the client node and the server node and finally image recognition at the server node. As a consequence, the techniques we expect to learn from this project include:

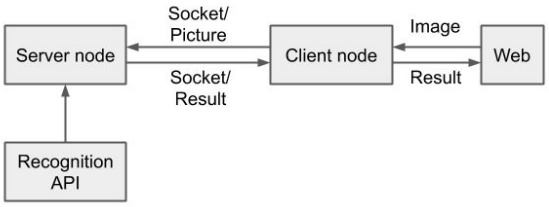
1. The client node should be able to receive an image uploaded through a web page and query the image to the server node;
2. The server node should then receive the image from the client node, call the image recognition API and return the result back to the client node.
3. The image recognition API should be able to recognize the input image and return its class name.

2. Experimental Methodology

Since our system mainly consists of three modules, we introduce the technical details of them separately:

- Client node
  The task of the client node is to accept the image uploaded by a user and transmit it to the server node. We first need to catch the uploaded image using an HTML file at the client node. Then we use CGI to call the function that sends the image to the server node using a socket when the recognition process is triggered on the webpage. Finally, the client node will wait for the result of image recognition from the server node and shows it on the webpage.
- Server node
  The server node keeps listening from the client node to establish communication using socket programming. Once retrieving an image from the client node, we utilize the image recognition API to process and categorize corresponding images. In the end, the recognition result will be sent back to the client node.
- Image recognition API
  The image recognition API will be called when the server node receives the image. After the image is passed in, the API preprocess the image in an appropriate way and input it into a pretrained GoogleNet model implemented using PyTorch. Finally, the recognition result vector will be output at the end of the model, converted into the corresponding class name and passed back to the program that called this API.

- Architecture

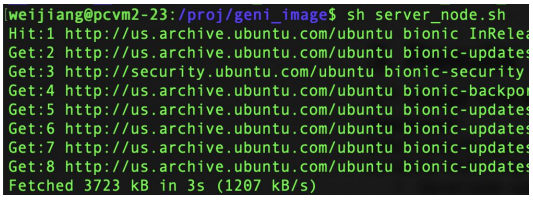


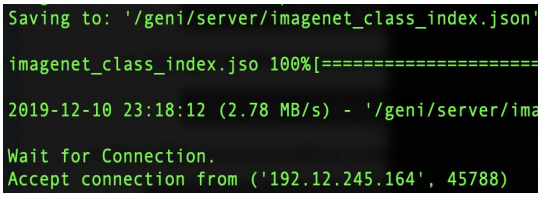
3. Results

3.1 Usage Instructions

Before using the application, you should make sure reserve two nodes whose version should be Ubuntu 18.04 or higher and the configuration file runs correctly on each node:

1. Server node: use server_node.rspec to reserve resources on the virtual machine and use server_node.sh to configure the required environment.
   In this case, we should have python3.6 and PyTorch, torchvision, PIL/pillow installed. Then the script will self-run the image_server.py which runs as a server program.



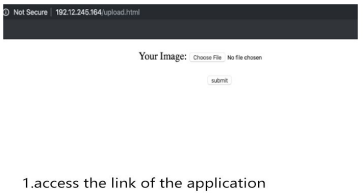


2. Client node: use client_node.rspec to reserve resources on the virtual machine and use client_node.sh to configure the required environment and code.
   In this case, we are using the 192.12.245.164 and have Apache2 deployed and put the upload.html and client.py on the client node. In addition, please update the IP address in upload.html and client.py.

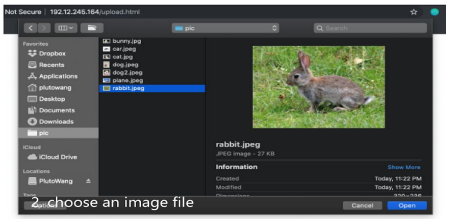
Our Webpage based Image Recognition Application is easy to deploy and very user-friendly. Once the client and server nodes are set up as we indicate above, users can easily make use of our system by accessing the link: http://192.12.245.164/upload.html.

There are 4 steps to use the application and these are screenshots of each step:

1. access the link of the application
2. choose a single image file(jpg,png or jpeg file) that you want to upload and detect
3. submit and upload the image
4. display the result



1.access the link of the application



2. choose an image file

3. upload the image

4.display the result

3.2 Analysis

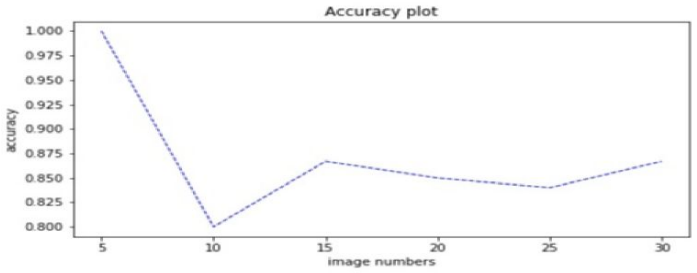
We do these experiments using our webpage-based image recognition application:

1. We want to know what is the accuracy rate of this system. Given a certain amount of images, how many of them could be recognized by this application correctly?
2. We want to know whether there exists a significantly positive correlation between the image file size and the total time to get the result. Is it that the bigger the datafile is, the longer it takes for us to get the result?
3. We want to know how robust our system is. In other words, is our system strong enough to handle some exceptions?

3.2.1 Accuracy rate

We download many random images from the Internet including cars, animals, clothes, food and so on. And we calculate the accuracy rate as the amount of the images goes up. We produce a figure that shows the average accuracy rate as a function of the number of images we test.

Image number	5	10	15	20	25	30
Accuracy	100%	80%	86.7%	85%	84%	86.7%

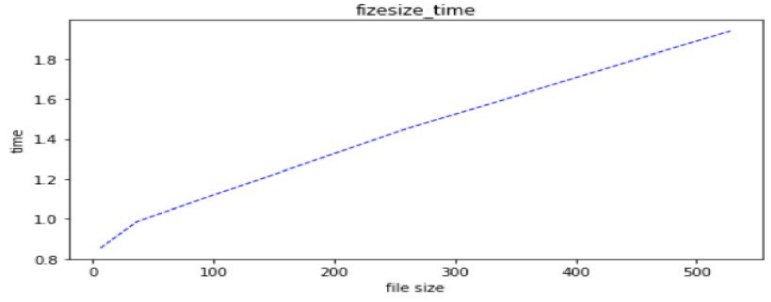


We can see from Figure 1 that as the number of files increases the accuracy rate is stable at around 85%.

3.2.2 Image File Size vs. Total response time

We test our system with different size of image files and record the time it uses. Then we draw the figure that shows the average time as a function of the image file size.

File size(KB)	7	14	37	252	528
Time(s)	0.854	0.885	0.986	1.435	1.941



We can see that the total response time is positively related to the file size. As the file size increases, the total time it takes to recognize the image goes up. This result makes sense because the file data will be transmitted between the client and the server node. It takes a longer time to transmit the bigger file.

3.2.3 Exception Handling

Our system is designed to detect and recognize images. We want to test whether our system can handle exceptions correctly. For example, what if we upload a txt or doc file?

This shows that if we upload a wrong type file (txt file in this case) and then the system will warn us that we should upload an image file. Our system is designed to recognize images so the file type is important, and this shows that our system is robust enough to handle this wrong file type exception.

4. Conclusion

In this project, we successfully developed an image-recognition system deployed on two nodes and we also learned techniques about the interaction between HTML and Python, communication between nodes using socket programming as well as deep learning applied in image recognition area. In the future, we can upgrade our system so that it can take multiple recognition tasks in parallel.

5. Division of Labor

Kaihong Wang is responsible for creating and maintaining the image recognition model API.
Nianyi Zhang is responsible for testing and the analysis part of the whole system.
Wei Jiang is responsible for image transmission, socket programming and nodes environment configuration.
Ruikang Wang is responsible for the frontend, including cgi configuration, web and socket programming.