CS 655 GENI Mini Project -
Image Recognition Application

Lei Huang U67278956

GitHub link: Geni Mini Project Demo video: Demo
Web Interface Address: **192.122.236.110:5000**
Slice name: **final-project-leih**
Server Routable IP: **192.122.236.111**
Client Routable IP: **192.122.236.110**

Introduction / Problem Statement

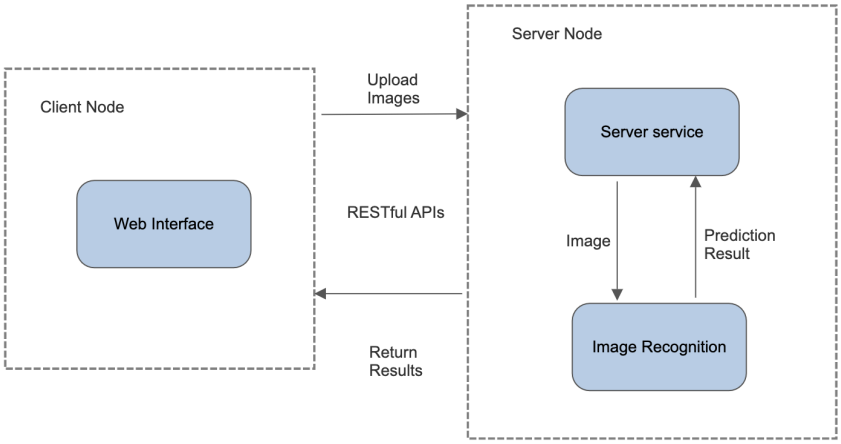
Project Description:

This project is to implement an image recognition service with a web interface. In my design, I will implement a server node which will deal with image recognition and a client node providing the interface and will interact with the server node. Users will use the web interface to submit images to the service node and get results.

Outcomes:

- I learned how to build an easy to use image recognition system with Restful APIs.
- I got more familiar with Python and HTML by using them to implement the application.
- I learned how to use Flask to build a http service, handle different types of requests (GET, POST(with binary file) etc.).
- I learned how to deploy a pre-trained deep learning model.

Experimental Methodology



Architecture Diagram

1. **Client**
   A web interface will be set up on the client node.It will allow users to interact with our backend service. It will let the users select the image and then send it to the image recognition model on the server node. After the model finishes recognition, the web interface will also display the result.

2. **Server**
   a. **Image Recognition Model**
      I used the GoogLENet model and I will select top 5 predictions to the image and send them back to the server.
   b. **Backend Service**
      It will get the image from the client node through the Flask route. Once it gets the image, it will call the image recognition model and pack the returned result. Then the backend server will send the result to the client side and display the result on the web interface.

Assumptions

I assume the maximum QPS is 100 and users can only access my service from the web interface.

Results

Usage Instructions:

1. Run on my nodes:

Server Node:
SSH to server node:
```
ssh [your-username]@pcvm1-17.geni.it.cornell.edu -p 22
```
Activate the virtual environment for Flask:
```
. CS655/bin/activate
```
Run the client service:
```
cd Geni_Mini_Project/Server
python3 server.py
```

Client Node:
SSH to client node:
```
ssh [your-username]@pcvm1-16.geni.it.cornell.edu -p 22
```
Activate the virtual environment for Flask:
```
. CS655/bin/activate
```
Run the client service:
```
cd Geni_Mini_Project/Client
python3 client.py
```

Web Interface:
After the services on the two nodes are turned on.

On your browser, visit: http://192.122.236.110:5000/

2. Or if you want to run it on your own nodes:

Reserve resources on GENI by using this RSpec.
Make sure you have selected the "Publicly Routable IP" option!
Log the routable IP address of both the server node and client node.

Server Node:
SSH to server node:
```
ssh [your-username]@[your-server-address] -p [your-port-number]
```
Download setup ssh file and run:
```
wget https://raw.githubusercontent.com/leih1219/Geni_Mini_Project/main/Server/server_setup.sh
bash server_setup.sh
```
Modify the code to run on your nodes:
```
cd Geni_Mini_Project/Server/templates
vi result.html
```
Press “A” on the keyboard to activate input, go to line 22 and change the address to the routable Ip address of your client node.
```
<form action="http://[your-client-routable-IP]:5000/">
```
Then press “Esc” on the keyboard to exit editing mode, then input “:wq” and press “Enter” to save your change.
Run the server service:
```
cd ..
python3 server.py
```

Client Node:
SSH to client node:
```
ssh [your-username]@[your-client-address] -p [your-port-number]
```
Download setup ssh file and run:
```
wget https://raw.githubusercontent.com/leih1219/Geni_Mini_Project/main/Client/client_setup.sh
bash client_setup.sh
```
Modify the code to run on your nodes:
```
cd Geni_Mini_Project/Client
vi client.py
```
Press “A” on the keyboard to activate input, go to line 28 and change the address to the routable Ip address of your client node.
```
remote_IP = '[your-client-routable-IP]'
```
Then press “Esc” on the keyboard to exit editing mode, then input “:wq” and press “Enter” to save your change.
Run the client service:
```
python3 client.py
```

Web Interface:
After the services on the two nodes are turned on.
On your browser, visit the routable IP address of the client node:
`http://[your-client-routable-IP]:5000/`

Analysis

Processing time may vary depending on the size and complexity of the image.
I tested each image **20** times and calculated the average time.

Performance Test

Image Size (KB)	Object in Image	Process Time (s)	Transfer Time (s)	Accuracy
55 KB	Cat	0.727	0.012	0.607
96 KB	Penguin	1.133	0.015	0.996
287 KB	Squirrel	0.958	0.021	0.821
661 KB	Dog	0.830	0.028	0.938
974 KB	Fox	1.527	0.031	0.661

Conclusion

The client node allows users to upload images and read results from the application. And, the server node is running the image recognition model. It is a classic client-server architecture, so we can modify the web interface and the backend server to adjust to different types of tasks. In the future, I also want to add more nodes to make the application more scalable.

I chose Flask instead of Tomcat, because I think Flask is easier to use and is much more convenient for me to interact with the image recognition model. I used GET and POST requests in this application. The GET request is used to get access to the homepage of the project and I used a POST request to upload the image to the server node.

From the analysis above, we can see that the transfer time will rise with the image size increase. However, the processing time the recognition model takes to classify an image is usually irrelevant with the size of the image.

Division of Labor

All work done by Lei Huang.

Helpful links:
https://github.com/leih1219/Geni_Mini_Project
https://www.youtube.com/watch?v=s_RJ1Gwe8R4