CS655 Computer Networks
Mini Project - Image Recognition Application

Team members
Ruizhi Jiang (U17637349), Tiancheng Zhu (U24177199),
Anzhe Meng (U50590533), Jiahao Song (U57363411)

Github Link:
https://github.com/ztcric/CS655-MiniProject

1. Introduction / Problem Statement

We intend to develop an image recognition system, and deploy it on GENI node. The users utilize our system to perform an image recognition classification by submitting an image query via a web interface. Users first need to upload their pending image through our web interface, then, the front server will relay the image to our backend node with recognition system. After the classification is done, the result will be sent back to the web interface. Lastly, our web interface is responsible for displaying the classifying result. We plan to implement a simple recognition system classifying multiple classes using a pre-trained model (ResNet50) and this project is scalable for more sophisticated image recognition tasks in the future.

2. Experimental Methodology

2.1 Web interface

This node is responsible for providing an entry for users to submit an image query or display image recognition output. Basically, a web server will be set up on this node. When the user initiates a get request to the web interface, a landing page for uploading the image will be displayed. Once the image is successfully uploaded, the second responsibility of the web interface is to relay the image to the back end node with recognition system. Once the results are sent back from the recognition system, the third job of the web interface is to display the result to the user.

2.2 Recognition System

We successfully transferred the pre-trained deep learning model, ResNet50,to a multi-class classifier. There are 1000 optional outputs and our website prints out its top-1 prediction and its corresponding accuracy percentage. Note that the weights of our neural network are collected from the Internet and no training process was involved throughout our work.

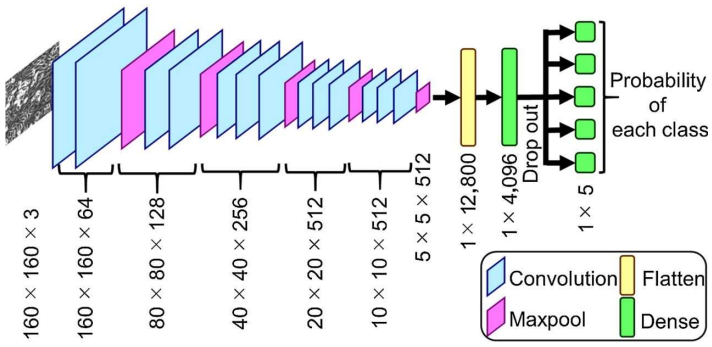

Fig 1: ResNet50 Model

The image fed to the model is sent via Flask route. As long as the type of the received file is legal (i.e. .jpg, .jfif), our model will predict the category of the image and then return the result to the front end interface as a display.

3. Results

GENI Setup:

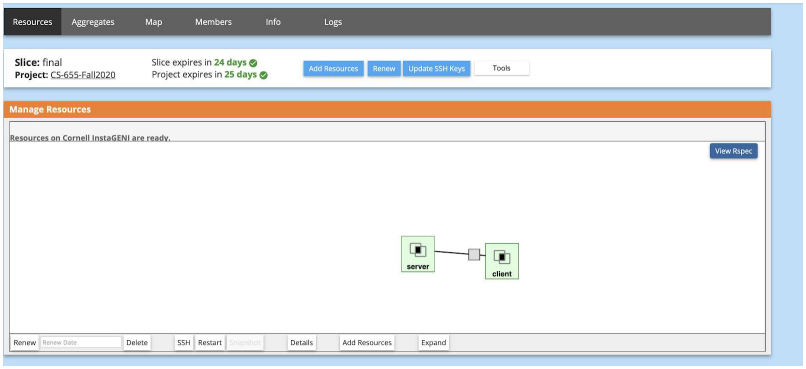

Fig 2: Slide Resources

How to run: (**group members and administrators only**)

Web interface:

- SSH to client node using ssh username@pcvm1-39.geni.it.cornell.edu -p 22
- Activate virtual environment - under /users/rzjiang/cs655geni - type command '. venv/bin/activate'
- Turn on web interface server - under /users/rzjiang/cs655geni/webapp - type command 'python3 app.py'

Backend server:

- SSH to server node using ssh username@pcvm1-40.geni.it.cornell.edu -p 22
- Activate virtual environment - under /users/rzjiang/BackendServer - type command '. venv/bin/activate'
- Turn on backend server - under /users/rzjiang/BackendServer/webapp - type command 'python3 backend.py'

Go to your browser:
Visit web interface at: http://192.122.236.116:5000

3.1 Usage interface

We used a web interface that allows users to upload an image to our system. After the image was uploaded, the image was sent and saved to our back end server which then performs the image recognition process; After the process is finished, the backend server will send the result back to the web interface for users to view.

Fig 3: Input Interface

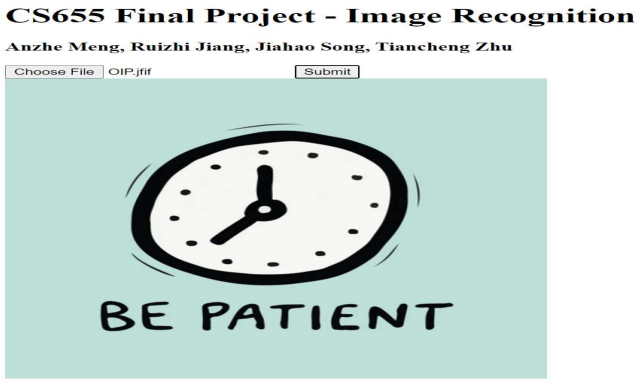

Fig 4: Loading Interface

Fig 5: Output Interface

3.2 Analysis

Different images may have different processing time, as the time of image recognition model cost for various images can vary.

Performance test

Image Size	Image Type	Processing Time	Internet Delay	Result from System
346KB	pen	3.573s	0.046s	ballpoint, (94.1%)
87KB	panda	3.582s	0.032s	giant panda, (99.8%)
26KB	dog	3.747s	0.013s	Labrador retriever (90.5%)

4. Conclusion

First, we utilized two GENI nodes to implement our project; A client node which as a web server hosting web interface that allows users to interact with our system; A backend server node that runs our image recognition model; With this design, our project can be extended to perform different kinds of tasks, as web interface and backend server being different modules, we can easily modify each or all of them to perform any other tasks in this network setup.

Second, from the analysis above, larger images naturally cause higher internet delay; Processing time is independent of image size, it only depends on how complex each image is for the model to classify.

Finally, Http protocol is imperative in our system. We utilize the Get request to get our landing page and utilize the Post request to upload our image. There are also other types of requests in Http protocol such as Delete and Put, but it's not necessary in our case. There are a lot of http frameworks or libraries out there, we chose to use Flask because our image recognition system is written in python. By using Flask, it's easier for us to connect our web server with the image recognition system.

5. Division of Labor

Member	Task
Ruizhi Jiang	Web interface & Rest API & GENI setup & Flask setup
Tiancheng Zhu	Web interface & Rest API & GENI setup & Flask setup
Jiahao Song	Neural network implementation & Training & Classifier script
Anzhe Meng	Neural network implementation & Training & Classifier script