BU CS655 GENI Project Report — Image Recognition

Hanlin Zou, Xiaohan Zou

GitHub repo: https://github.com/Renovamen/BU-CS655-Image-Recognition
Demo video: https://github.com/Renovamen/BU-CS655-Image-Recognition#demo-video
GENI slice name: final-xz
Public routable IP: http://130.127.215.146:80

1. Introduction

Our project provides a web interface which allows users to upload one/several images and response with it tries to recognize the image(s) correctly. It sends image recognition queries to the server which provides image recognition services.

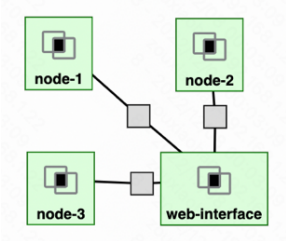
The front-end of our project is implemented using Vue 3, Vite 3 and Ant Design. The back-end is implemented using Flask. We used GoogleNet (PyTorch + torchvision) for image recognition and implemented socket communication between the web interface and workers. The web interface assigns images to available workers and displays recognition results in the browser.

2. Learning Outcomes

- Socket programming
- Deep learning API
- GENI nodes deployment
- Full Stack programming
- Network evaluation

3. Experimental Methodology

The diagram of our architecture is on the right. Resources are reserved from Clemson InstaGENI. The web-interface node receives the images uploaded by users, and forward them to the worker nodes (node-1, node-2, and node-3). The worker nodes recognize the images and return the results to web-interface node.



We implement an emulator to set loss rate, failure rate and delay time for worker nodes. We calculate the throughput after every 4 images uploaded using different parameters.

Assumptions we made:

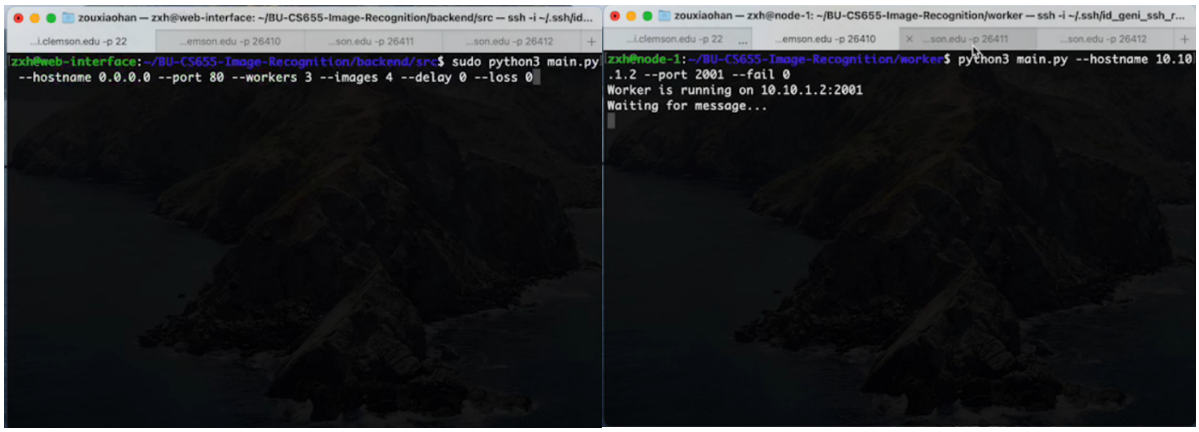
- Web hostname: 0.0.0.0
- Web interface port number: 80
- Maximum number of workers: 3
- Sizes of the uploaded images: 37 KB, 147KB, 94 KB, 168KB

- Delay time and loss rate are same for all workers
- To avoid the situation that all worker nodes fail, we let at most one worker node to fail
- For each group of parameters, we run 5 experiments and reported the 95% confidence interval of throughput (bps)

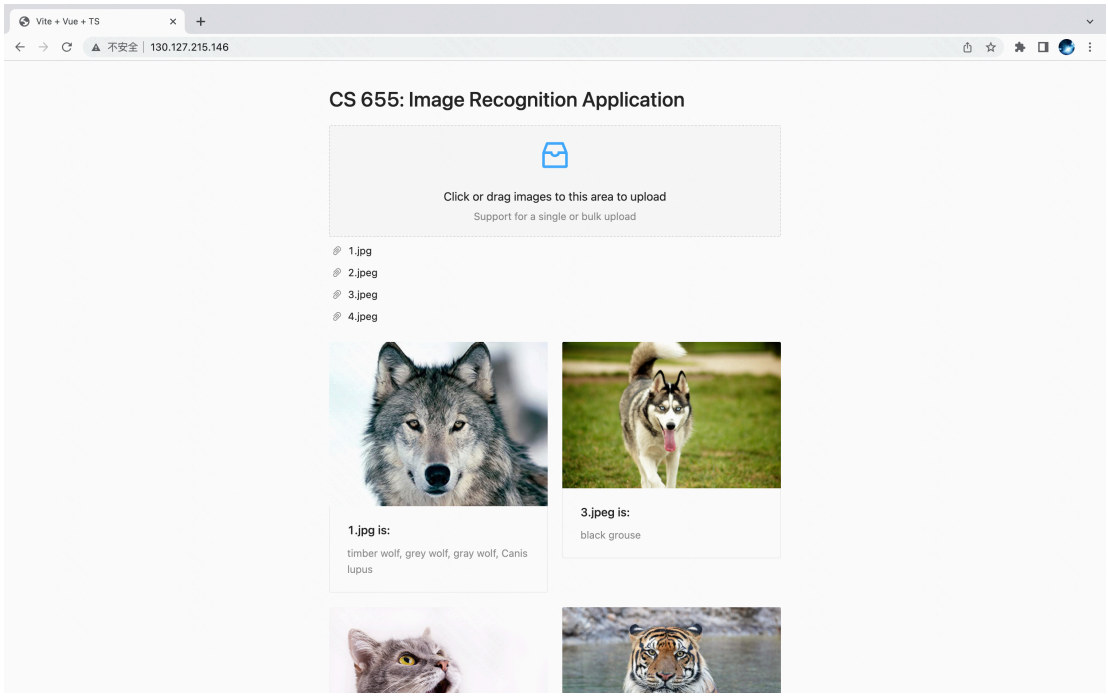
4. Usage Instructions

***NOTE:** The more detailed usage instructions can be found in our GitHub repository

First, follow our README to reserve resources on GENI and install dependencies. Then activate the web-interface node and worker nodes:



After that, paste the public IP you obtained in browser and upload images for recognition!



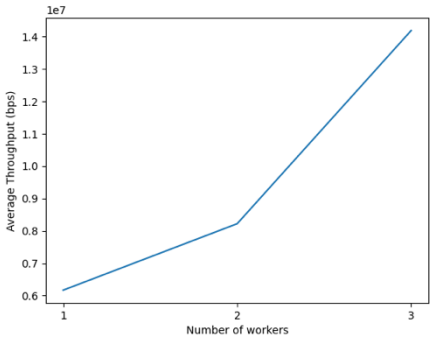
5. Experimental Results & Analysis

***NOTE:** See here for raw experimental results and the images we used during our experiments

5.1. Different numbers of workers

We report the 95% confidence interval of the throughputs when using different numbers of workers in the following table. We also provide a plot for the average throughput as a function of the number of workers. Here the delay time, loss rate and failure rate are set to 0.

Number of workers	95% CI of throughput (bps)
3	14193197.1429 ± 1928313.037
2	8222225.6971 ± 1846675.551
1	6167982.6327 ± 1326339.942

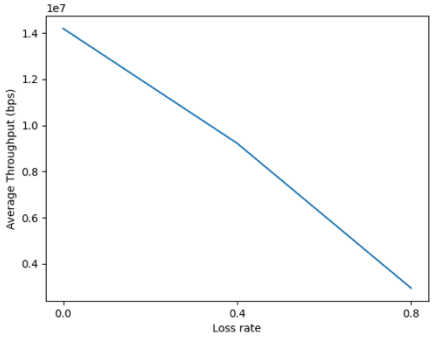


The results show that an increase in the number of workers results in a better performance, since more workers can handle the same workload in a shorter time.

5.2. Different loss rates

We report the 95% confidence interval of the throughputs when using different loss rates in the following table. We also provide a plot for the average throughput as a function of the loss rate. Here the delay time and failure rate are set to 0, while the number of workers is set to 3.

Loss rate	95% CI of throughput (bps)
0	14193197.1429 ± 1928313.037
0.4	9221141.8049 ± 1854527.319
0.8	2943601.3002 ± 513265.35

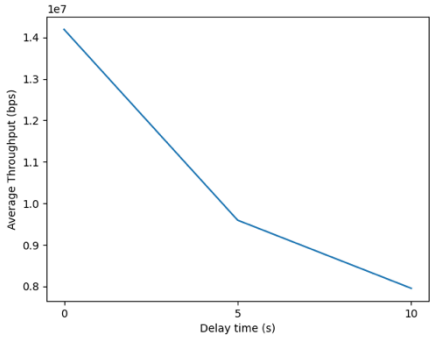


The results show that an increase in the loss rate in a worse performance. Because we need extra time to re-send the images if the results from the workers are lost.

5.3. Different delay times

We report the 95% confidence interval of the throughputs when using different delay times in the following table. We also provide a plot for the average throughput as a function of the delay time. Here the loss and failure rate are set to 0, while the number of workers is set to 3.

Delay time (s)	95% CI of throughput (bps)
0	14193197.1429 ± 1928313.037
5	9594891.2064 ± 3144010.398
10	7953135.7798 ± 2297107.783

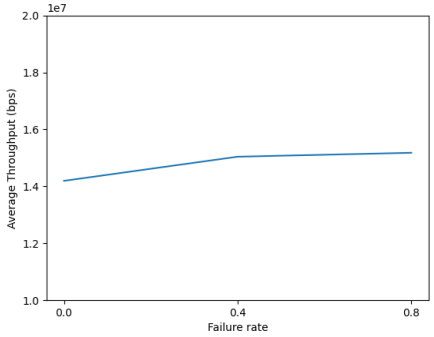


The results show that an increase in the delay time result in a worse performance, since we need more time to obtain responses from the workers.

5.4. Different failure rates

We report the 95% confidence interval of the throughputs when using different failure rates in the following table. We also provide a plot for the average throughput as a function of the failure rate. Here the loss and failure rate are set to 0, while the number of workers is set to 3.

Failure rate	95% CI of throughput (bps)
0	14193197.1429 ± 1928313.037
0.4	15040344.8562 ± 1840763.646
0.8	15177870.0491 ± 2346390.112



The results show that the worker node failure rate make almost no difference to the throughput. Here are some possible reasons: (1) As we mentioned, we assume there is at most one worker node may fail. Our system can handle it effectively by re-assigning the task to another worker. (2) The number and sizes (< 200 KB) of the images we uploaded are small. Increasing the number of images and image sizes may lead to a different result. However, the performance of GENI nodes is too poor to support such experiments.

6. Conclusion

- When loss rate becomes larger or delay time becomes longer, throughput becomes smaller.
- When the number of workers becomes larger, throughput becomes larger.
- Failure rate of workers doesn't have significant impact on throughput.

For possible extension, we could increase workers performance by adding more workers and improving performance of each worker, so that the web wouldn't crash when uploading images larger than 200 KB.

7. Reproducibility

See https://github.com/Renovamen/BU-CS655-Image-Recognition#reproducibility

8. Division of Labor

- Hanlin Zou (U96634471): Front-end, workers, image recognition
- Xiaohan Zou (U18269004): Back-end, GENI deployment, experiments and analysis