Deep Learning Second Project Increment Report

Project Team 1
Billy Capps, Class ID 6
Adam Carter, Class ID 7
CS 5542 – Spring 2017

Introduction

This is the summary report of the second iteration of work on our Virtual Field Trip project. The goal for this iteration was to continue the implementation process and improve the existing work done in the last iteration. Specifically, for this iteration, our goal was to improve the Spark accuracy, implement the Tensorflow workflow, and improve the functionality of the Conversation API in relation to our project.

Project Goals and Objectives

Our project is to create a virtual environment where students can observe an area and attempt to locate items of educational value. An example would be to tour a cave and locate a stalagmite. Teachers would be able to add new images to the collection. Through deep learning, objects in the images could be identified reducing the time teachers need to spend preparing the images. It will also allow the system to give immediate feedback. The software can either confirm the student selected the correct object or told what object they selected instead.

Our motivation is to provide students with the possibility of having first-hand experiences in places that they cannot physically visit. Caves, pyramids, the moon, and the ocean floor could be visited by students regardless of geography or income level. Scavenger hunts ensure that students are actively participating and focused on areas of educational importance.

System Features and Use Case

As stated in our Project Proposal, the Google Cardboard interface will allow students to select which image they want to explore from a library. Once the image opens, they will be given a brief introduction and then told which items to locate. It is tempting to include additional educational information while the student is exploring; however, the purpose of this project is supplement teacher instruction and not replace it. The interface needs to remain minimalist so that the student can focus on exploring the new environment. When the student believes that he or she has located the requested item and focused the cursor on it, the button on the top of the Google Cardboard device will be pressed. The system will then either congratulate the student and offer the next item to seek or tell the student what item the name of the object they selected and ask them to keep trying.

Approach

Our application is currently leveraging the following technologies:

- Android using Google Cardboard
- Spark MLib (though this is not currently linked to the Android Application
- Clarifai REST API
- API.AI

- Tensorflow using Python3

For this second iteration, one of the goals was to improve the accuracy of the system and give a framework for comparing the relative performance between Spark MLib and Tensorflow. In order to do this, we are continuing to use the Game Room again, but realized we had to take a different approach to the problem. There are nine classifiers used in our data set, and that amount of data proved to be unmanageable in our last iteration. The accuracy from the model completed in the first iteration was just above 53%. Technical problems with the OpenCV libraries prevented the use of a larger training data set for the Spark MLib Api.

One realization from the VR implementation in the first iteration is that the image content is known and unchangeable, so rather than use indirect images or data from a different source, we decided to sample the core VR image from which the snapshots will be generated. To do this, we defined a viewable box around each of our core visual elements that ensured that at least 50% of the desired element was included in the image, then drew a number of 400 x 400 pixel subimages that represent the item to be classified. This resulted in between 10k – 15k images per classifier. From these images, we then sampled a collection of 150 images per classifier to form a new dataset drawn from the VR image. We then further sampled another 10 images per classified to form our validation data set.

We tried generating a dataset with 1000 training images per classifier and a validation data set of 50 images, but this proved to be too much data for Tensorflow and Spark to process without generating critical failures.

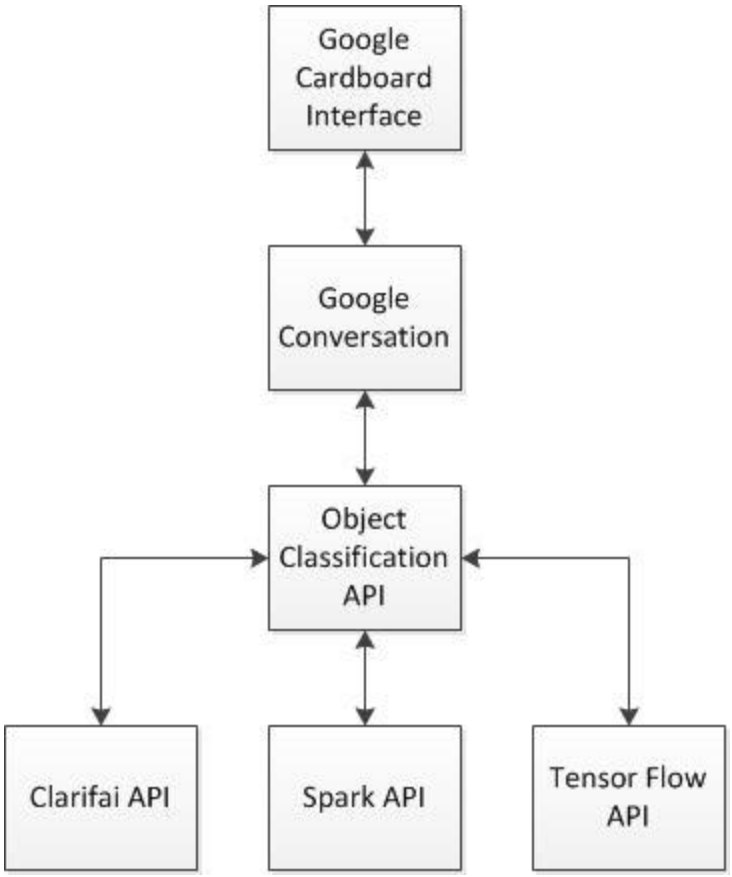
Using the API.AI chatbot application, we have created an interface that can respond to the user's voice. This allows the user to access items that would otherwise be buried in menus. It also allows the user ask questions about the virtual environment. At the moment, the agent is fairly generalized, but as it becomes more integrated into the system, we will begin incorporating data from the images.

Related Work

No new research was done into related work during this iteration, though a local news story seemed relevant. The V Form Alliance is a new startup in Kansas City whose mission is to "allow elementary and middle school students to take a virtual reality 'field trip' exploring landmarks in Kansas and Missouri that are relevant to black history." [1] Clay Middle School has a page of virtual field trips. [2] This is a collection of web pages that offer virtual tours of places with educational value. Google Cardboard Camera allows users to create their own VR photos. [3] NYT VR takes users on a journey to places of significance and includes information about the associated New York Times stories. [4] Titans of Space lets users explore our solar system using virtual reality. [5]

Application Specification

At this point in the project, most of the pieces are in the early stages of development and are largely unconnected. As future iterations proceed, we expect this connectivity to improve. We will touch on the various systems and their function.



Implementation

At this point in the project, most of the pieces are in the early stages of development and are largely unconnected. As future iterations proceed, we expect this connectivity to improve. We will touch on the various systems and their function.

Android Application

This application leverages the VrPanoramaView implemented by Google VR. This is separate from the new GvrView full VR implementation using OpenGL, which means it has limited functionality, but is sufficient for this projects' needs. This means that the native onCardboardTrigger() method cannot be used, so we instead trap any screen touch event (which is essentially what the trigger does) to take a screen shot using the Pitch and Yaw recorded by the headset to determine the current center of the Panoramic Image.

One difficulty of this task was handling images that would wrap around the 'seam' of the image, where the flat image boundaries would need to be crossed to complete the screenshot. Because the VrPanoramaView is more limited in its feature set, the native Android capability of capturing a screenshot is not available. A custom algorithm had to be written to find the current view center and grab the screen shot manually. For the case where we would wrap the scene, a separate image file is referenced that has been rotated 180° so we can cleanly capture the seam.

Spark Image Recognition

The code from Lab4 and Lab5 had to be uplifted because OS-specific errors were occurring that stem from an older version of OpenCV. To overcome this, the JavaCP and JavaCPP libraries had to be uplifted from version 0.11 to 1.3.1. This also meant that features like SIFT Feature Extraction were no longer available (as they are not freely available for patent issues). The code was converted to use the AKAZE feature extraction model, and the results were faster performance wise, but lead to smaller feature vectors and less accurate prediction. Results are descripted below in the documentation setting.

While the performance of the system is significantly improved, there are still several current limits to this approach. First are the OS issues that prevent using a larger dataset (OpenCV is not entirely stable, and frequently causes fatal OS errors that interrupt parsing). We were able too generate a more consistent dataset that was also larger in scope than the first iteration, but that increase caps not much further beyond where we drew the line. Second, I believe converting to Greyscale significantly reduces accuracy. Both SIFT and AKAZE models tend to strongly favor one image (The Star Wars posters, which are chosen significantly more frequently than any other classifier) which would appear different if processing RGB vectors separately. That change was determined to be outside of a realistic scope for this iteration. Finally, the low quality of our VR screenshots in particular makes accurate analysis more challenging. We hope to find ways to address this shortcoming in the next iteration.

Tensorflow Image Recognition

Using the same dataset as the Spark MLib approach, the Tensorflow approach had to actually compress the image in order to process successfully. Specifically, we had to compress the image size from 400 x 400 to 200 x 200. By doing that, we were able to successfully build a multilayer CNN similar to the one used in Lab 9 that was able to process our training data, then evaluate itself against our test data. This process took a tremendous amount of time to compete, however. It took approximately six hours to complete 200 iterations through our CNN, with a random data sampling of 10%, or 135 images, each iteration. Further discussion of the results of this approach are shown below.

Documentation

Presented are some of the screen shots taken from our application. We'll give a quick overview of some of the visual artifacts used to build our dataset, then some of the artifacts created by our application.

Basic VR Image of Game Room used as the basic for early analysis

Screenshot of Google Cardboard Viewer while running

Six images captured from our Android Application of different areas in the VR Viewer.

With training images taken in the same basic format that screenshots would be taken, our model accuracy was able to be significantly improved. This is the Confusion Matrix generated by AKAZE against a dataset with 150 images per classifier and 10 validation images per classifier:

```
|================== Confusion matrix ==========================
10.0  0.0   0.0   0.0   0.0  0.0   0.0   0.0  0.0
0.0   10.0  0.0   0.0   0.0  0.0   0.0   0.0  0.0
0.0   0.0   10.0  0.0   0.0  0.0   0.0   0.0  0.0
0.0   0.0   0.0   10.0  0.0  0.0   0.0   0.0  0.0
0.0   1.0   0.0   0.0   9.0  0.0   0.0   0.0  0.0
0.0   0.0   0.0   0.0   0.0  10.0  0.0   0.0  0.0
0.0   0.0   0.0   0.0   0.0  0.0   10.0  0.0  0.0
0.0   0.0   0.0   0.0   0.0  0.0   0.0   6.0  4.0
0.0   0.0   0.0   0.0   0.0  0.0   0.0   0.0  10.0
0.9444444444444444
```

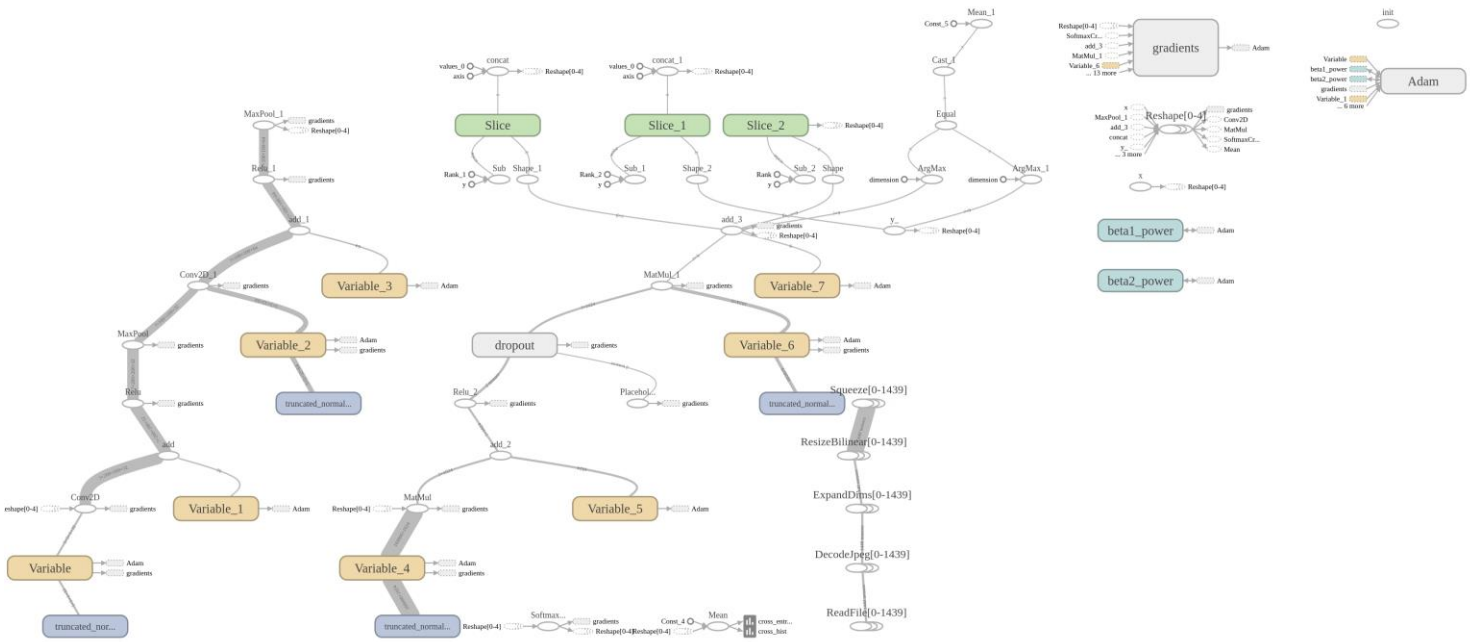
This is a significant improvement on our 52% overall model (though only 19% on screenshot data). With our training data more closely aligned with the expected prediction data, our accuracy has been significantly improved. It does still appear that the table classifier (classifier 8, which was only 60% accurate, still struggles the most, particularly with sections of empty table that are edge-sparse.

Tensorflow Results

The Tensorflow execution took a long time to complete, and, as it would seem, ran twice as long as it needed to. We ran 200 executions with a 10% sampling (135 images per iteration). We hit a perfect 100% accuracy by the time we hit iteration 100 and never dropped from there. Here is the logging from the execution of the Tensorflow job. Full logging can be found in the documentation folder of the project.

```
Number of Classes:  9
Number of Images:  1350
...
Running Iteration  0
step 0, training accuracy 0.185484
...
Running Iteration  50
step 50, training accuracy 0.962121
...
Running Iteration  100
step 100, training accuracy 1
...
Running Iteration  150
step 150, training accuracy 1
...
Running Iteration  199
test accuracy 1
```

For whatever reason, the logging generated by the run failed to generate anything other than a system graph. There were errors trying to launch tensorboard that I couldn't find an explanation to, but we will hope to be able to generate better logging through Tensorboard for the final iteration. As it stands, here is the image generated for the system graph:



Project Management

We have also chosen not to use ZenHub for tracking the work being done on this project. Both of us have working experience, and consciously chose not to work through the overhead of a tracking tool for a two-man project. Because of this, we have also not tracked hours spent on this project.

Adam was responsible for the initial data set generation, writing the initial Android application using Cardboard viewer, and working on the Spark and Tensorflow implementations. Bill was responsible for the creating the conversation agent.

One of the largest issues encountered was the generation of a screen shot from the Cardboard Viewer that could be trigger using the cardboard trigger. The documentation around both the VrPanoramaView and the GvrView and associated classes (Google's newest VR approach that replaced the previous CardboardView API) is very poor, and new enough that there have been few questions asked or tutorials generated. It took a reasonable amount of experimentation to determine that the OpenGL libraries were both overkill and not wholly suited to our needs, but the VrPanoramaView exposes virtually none of the underlying functions to developers.

There has also been mention made about the difficulty in training a dataset. The OpenCV libraries used for image processing are not fully stable, and routinely crash above a certain volume of image data in the Feature Extraction process. Part of the next iteration, which will focus on integrating the Spark API with our application, will need to be resolved to improve the accuracy of our models.

We both contributed to the documentation effort for our system.

The next major iteration requirements are as yet undocumented, but as it is the final iteration, we expect it to be one mostly about assembling the system pieces together into a cohesive ecosystem of REST services.

References

[1] http://www.startlandnews.com/2017/02/virtual-reality-field-trips-offer-black-history-experiences-kc-students/
[2] http://www1.ccs.k12.in.us/clm/media-center/fieldtrips
[3] https://play.google.com/store/apps/details?id=com.google.vr.cyclops&hl=en
[4] https://play.google.com/store/apps/details?id=com.im360nytvr&hl=en
[5] https://play.google.com/store/apps/details?id=com.im360nytvr&hl=en