Group 67 - Winter Midterm Progress Report

Deep Sequential Learning for Object Recognition on a Mobile Robot

Julian Weisbord, Michael Rodriguez, Miles McCall

Oregon State University

CS 462 Winter 2018

Abstract

The Winter Progress Report describes the work that has been accomplished by our team from Fall term through the first half of Winter term. This document catalogs the first 16 weeks of research and design to give readers a feel for the many challenges that must be overcome to build an intelligent agent, capable OF classifying objects with sequential image recognition. It also contains details about particular obstacles our group has overcome, interesting notes from the recapped weeks, analysis, and reflection on the work done so far. lastly, it covers what needs to be done for the project to be successfully completed before the Engineering Expo.

Contents

1 Purpose

The purpose of our research is to take a bare-bones mobile robot platform and design a software package to train robots to recognize the objects in any given environment. We will do this by utilizing several different machine learning algorithms in an effort to make the robot a more intelligent agent. Currently, most robots at the Personal Robotics Lab (PRL) of Oregon State University (OSU) lack custom software, granting the ability to efficiently understand and classify objects in their environment with a high degree of certainty. By leveraging the Robot Operating System (ROS) and the existing mobile robot platform, we aim to create a software pipeline that sequentially trains Convolutional Neural Networks (CNN's) and applies different sequential learning techniques to teach the intelligent agent to classify objects in its environment.

2 Goals

Our goal is to train the Fetch robot on 5 different common object classes and have it predict the class of new objects with an 80% success rate or better. Additionally, the intelligent agent must be able to retain old information about previous objects and learn more about their specific features over time such as: logos, shape, color, and owner. Because our intelligent agent is learning about its environment, this architecture could be used to train robots to assist humans in numerous applications.

Aside from strictly developing the most accurate classifier, we have also examined each part of our pipeline and researched alternative solutions to satisfy different applications, budgets, and environments. Cutting edge technologies and algorithms can not always be trusted to perform as reliably as universally accepted solutions, but a main goal of ours was to test these alternative hardware and software solutions in our pipeline. We tested software such as Inception (GoogLeNet) and hardware such as Intel Movidius neural processing USB sticks to assess their feasibility in our pipeline.

3 Miles McCall's Contributions and Progress

3.1 Description of Current Status

At this point in the timeline of our project, our group has done a good job communicating regularly and discussing how to split the project tasks into logical divisions between the three of us. We have all assumed the responsibility of one major part of the project at a time, and act as aids offering support to the other two members as well. This has allowed us to create more focused tasks and develop a small amount of expertise in our respective subject over our other team members. We all benefit from the other members specializing in their topics and teaching us what they've learned. We have multiple team meetings weekly, as well as weekly TA and client meetings.

Looking at our current position on the project, we have been progressing each major piece of the pipeline in some degree of parallel development. We've worked on perfecting our data collection software while developing the classification algorithm. We were also given some initial data to start our project, and we've been able to utilize these images to test the performance of our CNN as we write it.

Our team has also focused on the research aspect of our project this term, which was my primary focus and role this term. I have been testing a unique hardware solution for CNN predicting, the Intel Movidius Neural Compute Stick. These devices are external USB devices with custom hardware designed to efficiently process classification predictions. The main advantage these USB sticks offered us was their price, starting at seventy dollars for the unit. Instead of investing hundreds of dollars into modern graphics cards, this device could potentially use its custom architecture to perform at similar levels. Despite their appeal, we eventually evaluated the cost of implementing the USB sticks to be too prohibitive a setback to make them worth our time and effort in this project. Aside from the Movidius cards, though, was the option of external PCI slots to plug into a laptop or smaller device. This would allow our team to utilize Nvidia 1080 level graphics cards or higher without a full scale workstation to attach it to.

Besides looking into the usefulness of different hardware technologies, I also assisted Michael with the Fetch set up and data collection steps. It took a significant amount of effort to set up the Fetch robot with the proper custom environment and implement our software to it. After gaining access the to grad student calendars we were able to schedule time to use the robot and experiment room, which increased time we could practice with our software. I have worked with Linux and specifically Ubuntu large amounts in the past and was able to help Michael with initial environment configuration and other operating system specific problems. I assisted Julian's data processing steps as well but mainly only scraped data for him, or manually removed bad capture data from image sets. This process saved him time to work on the data formatting for input into the classification models.

3.2 Whats left to Do

Due to an extended amount of time and effort spent on my research of the Movidius sticks, I didnt progress as far as I would have liked to on the external GPU card slots. If the resources are available this solution could prove to be very effective in our pipeline, as it would assist the training and prediction steps. This research will be complete and implemented in the pipeline if decided to by the end of term.

I need to complete the implementation of the data gathering software as well, which exists as a few Python scripts. This should also be complete by the end of the term, and having this step functional will aid in our team in collecting additional image data for the intelligent agent to sequentially learn with.

With the data gathering portion of the software package complete, essentially the first half of the pipeline is done. I will now focus on assisting Julian develop the second half of the pipeline, the image classification and prediction while Michael continues to utilize the Fetch robot to generate a large collection of object images. Julian has been the main team member responsible for writing the classification code, so instead of attempting to repeat the work he has done I am aiming to assist him in problem solving, potential alternative solutions, and testing and debugging.

3.3 Problems Impeding Progress/Solutions

It took me longer than I was expecting to finish testing the installation process for the Movidius sticks. I ran into many struggles while attempting to set up and configure the interfacing software, such as strict and obscure library version requirements for the custom software to run, having the proper hardware available on my machine. The Movidius software relies on multiple external packages, such as Tensorflow, Nvidia GPU drivers, CUDA, and CUDNN. The software required specific versions of each package to be installed in the environment. I also consistently ran into hardware restrictions where I needed an Nvidia card to properly install the Nvidia software, but the goal of the Movidius sticks was to avoid needing one in the first place. Lastly, the Movidius processing sticks are designed to efficiently use CNNs to predict image classes, but lack the power to train the network in the first place, as we feared from the beginning. This isnt spoken about much by Intel, and took us some time to discover from online resources. After all these setbacks we decided the potential benefits to prediction time were not worth the trouble the sticks presented, and we ditched their application in our software.

3.4 Other relevant/interesting information about project

One interesting element of the project that has arisen so far would be the potential advancements to the data capturing algorithm itself if our team had more time and resources. There are multiple types of item markers that can assist the Fetch through use of sensors to detect the location of objects in 3D space. Markers such as AR tags, RFID tags, and others present ways besides pure object recognition software and manual assistance for the robot to localize on objects around it. The RFID tags for example broadcast a radio signal that can be listened to by the Fetch robot, it can then triangulate the location of the tagged object without needing to visually recognize it as such. This technology could potentially be applied in warehouse or factory environments, where the Fetch robot could locate objects in the entire building without having previous knowledge of their locations.

4 Michael Rodriguez's Contributions and Progress

4.1 Description of Current Status

Currently, I am in the data capturing process with the fetch robot. I am responsible for getting our mass data gathered so we can train our convolutional neural network on our 5 image classes. We have gained access to the fetch robot schedule and the HRI(Human Robot Interaction) room as well so we can carry out our data gathering sessions by booking times on both of these google calendar schedules. This past week I have been in the lab four times to make a new map of our test area, get the fetch robot synchronized with its new environment, gathering objects for our image classes, and clearing space for the fetch to conduct the data gathering. Today, I was able to create the keepout part of the map which is essentially a dead zone for the robot so it doesnt accidentally bump into other objects and expensive equipment that is in the room while it is moving around doing the data capturing. In addition to this, I was able to get a small test run of the robot gathering data on its own by running the modified capture.py file that Ive been working on since the start of winter break. The file basically tells the robot what sensors, cameras, reference points, maps, and positions it needs to use in order to gather the data. Also, the file tells it where to save to pictures which we currently have going into three separate folders that are for the original images, cropped images, and circular images. The images going into these folders are what make up our image classes and will be used for training our convolutional neural network.

4.2 Whats left to Do

The biggest part of my contribution to this project is getting the data with the fetch. As of right now, I still need to conduct about twenty four more data capturing sessions on the remaining twenty four objects. This will take a huge chunk of my time as the data capturing process per object takes about one hour assuming that nothing goes wrong during the process (such as the robot getting lost) and the picture quality meeting our standards. I currently have about one-half of the objects collected so I will also need to hunt down the remaining one-half at some point. I am also still working on improving the speed of the data capturing process by doing more modifications to the capture.py file. Overall as team, we still need to train our network and see how close we are to achieving our 80% success recognition rate.

4.3 Problems Impeding Progress/Solutions

About 4 weeks ago I ran into a roadblock while I was getting close to capturing data with my own laptop. All the training that we did with the robot was using lab computers which made it fairly easy to learn since everything was setup on them for the fetch and PR2 to run functionally. I had the ROS environment set up on my laptop during fall term with a visual machine that only had 4GB of memory which was fine at first for setting up the ROS environment, Fetch packages and all the necessary configurations the robot needed. However, the lack of access to my full 8GB of memory proved to be a big pitfall when I actually tried running all the visualization software from the fetch using RVIZ towards the beginning of winter term. Therefore, I had to start over and get rid of the virtual machine and instead dual boot my computer with Ubuntu 14.04 to gain access to my full 8GB of RAM. Which meant that I had to install the ROS environment, catkin workspace, fetch packages, and modify bashrc and configuration files all over again. This process took longer than I liked as I had done this so long ago that I was forgetting a bunch of little details and steps that made getting a 100% reliable functionality with the fetch difficult to achieve. Thanks to our grad student mentor Chris and his guidance, I was finally ready to capture data at the start of week 6.

Another problem that we encountered that was unexpected was the fetch going out of service for roughly two weeks. Its battery unfortunately stopped working out of nowhere and wasnt charging. As if having the robot down wasnt enough of an issue, this caused an issue for me as I was still in the process of setting up my laptop to work with the fetch so not having access to it delayed my progress in getting fully set up even more. Thankfully, the grad students in the lab did some maintenance work on it by setting up an appointment with Willow Garage, the makers of the robot, to get the fetch back up and running.

One other minor problem that we were running into early on was being able to get time on the fetch as other grad students/faculty members also use it to carry out studies, conduct tests, and use it for research purposes. At first it seemed like it was a first come, first served kind of system which was hit or miss depending on the time of day that we went into the lab. Also, we had trouble gaining access to the HRI room because other research groups regularly use the room to conduct studies and tests. Today for example, the room was booked from 8am to 6:30pm so the only lab time that I was able to get today was from 6:30pm to 9pm. Our client professor Smart realized that this was an issue and solved it towards the beginning of winter term by creating two time schedules, one for the fetch robot and the other for the HRI room. So now when we want to use the fetch and the HRI room, we book both schedules concurrently.

4.4 Other relevant/interesting information about project

One interesting aspect part of my contribution to the project is to see the different ways that the fetch can go about collecting its data. It can collect data with its Infrared camera, raw image camera, or RGB camara. It can localize objects by using visual markers in RVIZ, it can use AR Tags that are set up around the object, or it use RFID tags to find them using radio signals.

5 Julian Weisbord's Contributions and Progress

5.1 Data Preparation

5.1.1 *Purpose and Progress*

In order to take images and pass them through a Convolutional Neural Network (CNN) for classification, we must first label and convert the images into a matrix of binary pixel values. Tensorflow, the machine learning tool that we are using, expects an array of binary values representing the position of each pixel and their respective RGB color values. To accomplish this task, I wrote a Python module (prepare_data.py) to create and ”vectorize” a dataset. The code is fairly dense but the design is as follows: The CNN Model imports prepare_data.py and creates a PrepareData() object. This object contains: the data's location in memory, it's class label (Book, Coffee Mug, Chair, Stapler, or Screwdriver), and a matrix of each images pixel values. Finally, this object holds a Dataset() object which contains an iterator function called, ”next_batch()”, that allows the image classifier to easily iterate over all of the image matrices.

5.1.2 Future Goals and Complications

All of the prepare_data.py functionality has been somewhat tested and is currently in our github repository. The only issue impeding full progress is that there is a bug with the CNN Model and until that is fixed, we cannot pass a full dataset through the Neural Network, even though the prepare_data module has successfully created a full dataset. After testing this module more thoroughly, I would like to add more features. If we can store or ”cache” previous image pixel data and labels in a file, the process of creating that data on the fly multiple times can be instantaneous. As image datasets scale, the time to load the pixel data could become substantial. If we have a file with this information already, we can save a lot of computational resources and time.

5.2 Design and Implementation of Convolutional Neural Network

5.2.1 Purpose and Progress

An Artificial Neural Network (ANN) is a graph of partially or fully connected nodes. Each layer of these nodes has a weight coefficient that is applied to them via mathematical convolution. The weight coefficient is initially random but over time, the weights are updated to more and more accurate values via backpropagation. Convolutional Neural Networks (CNN's) are similar (ANN's) except they make the explicit assumption that the inputs are images. [1] The CNN Architecture that I have created is one that is designed to be all inclusive. The bulk of the image classification takes place inside model.py. model.py sets up all of the layers of the CNN, grabs the dataset from prepare_data.py, determines the loss function, calls gradient descent algorithm, and finally trains the network. Once the CNN is fully functional, it should be able to classify any number of images as to whether or not they are a: coffee mug, stapler, screwdriver, chair, book, or none of the above!

5.2.2 Future Goals and Complications

The CNN Model is not functional as it stands and this is definitely impeding project progress, my goal is to have it fixed within a couple of days. The problem is that the size of the arguments to the loss function are supposed to be the same size, however currently they are not. Once this is fixed, I would also like to implement our project functionality with the ResNet Inception Classifier. The Inception Image Classifier is already trained for large scale image classification on 1000+ classes. [2] The purpose of using Google's Inception model would be to get a more accurate classification rate. Google spends a lot of money and time determining the perfect parameters for CNN's, it is likely that if we take Inception but retrain it on our data, we will receive better results just because we are using highly researched parameters. Finally, I will run both architectures (my original and Inception) and determine which is better at classifying our image data.

5.3 Sequential Learning Research

5.3.1 Purpose and Progress

Sequential or Online Learning allows Neural Networks to be trained on both past, current, and future data. Sequential Learning is important because it can be exploited to improve classifier prediction accuracy. [3] The more common and easily implemented, industry-standard known as, batch Learning, forgets previous knowledge whenever it needs to retrain to learn anything new. We currently have batch learning implemented in our model but plan to add a Sequential Learning Architecture soon. The only progress that I have made here is by reading research papers and learning about Sequential Learning Architectures such as Elastic Weight Consolidation and Dual Memory Architecture.

5.3.2 Future Goals and Complications

I am not aware of any future complications with Sequential Learning but I anticipate that it will require a lot of computational resources and time to make our model more accurate than with Batch Learning. Once Sequential learning is implemented, we hope that our classifier will classify objects at a rate greater than 80% and also be able to learn specialized knowledge about the data.

5.4 Team Management

5.4.1 Purpose and Progress

I was elected the group team leader at the beginning of the year but it is only this term that this task has become a significant amount of work (about 8 hours per week), which is why it should be included in this progress report. Every team should have a manager or leader to take an abstracted look at the group project and make sure that not just individuals are on-track but that the team is on-track. As the manager, I act as the team planner, technical adviser, and presenter. One of the most important tasks that I undertake is divvying out work to group members based on: time constraints, personal skill sets, past performance, and interest. I plan out these tasks weeks in advance based on a detailed timeline of our goals. After everyone has their tasks, I answer questions and advise members, if needed. Every week our group has two meetings, one with our client and one with our TA. For these meetings, I prepare and present our group's successes, failures, future schedule, and ask detailed questions.

5.4.2 Future Goals and Complications

Everything has gone pretty well so far, our group isn't fighting and everyone seems to be happy with their work. Complications have risen when a group member does not accomplish their task by the due date and as a consequence, the task rolls over into the next week. Preventing this in the future can be very difficult, you must get to the route of the issue to determine why the task wasn't accomplished. The problem may be that you assigned too much work to a team member, given them a task that doesn't align with their interests or skill set, or maybe they didn't receive enough guidance. Failure points can be brought on by a number of different issues and it is important to diagnose them early. My future goals for team management are to make tasks that hold individuals more accountable and to also offer guidance but not so much guidance that I feel as if I am doing the task myself.

6 Conclusion

References

[1] A. Karpathy, "Image recognition lectures," http://cs231n.github.io/, (Accessed on 10/26/2017).

[2] TensorFlow, "Image recognition," https://www.tensorflow.org/tutorials/image_recognition.

[3] T. G. Dietterich, "Machine learning for sequential data: A review," http://web.engr.oregonstate.edu/~tgd/publications/mlsd-ssspr.pdf.