Group 67 - Final Spring Report

Deep Sequential Learning for Object Recognition on a Mobile Robot

Julian Weisbord, Michael Rodriguez, Miles McCall

Oregon State University

CS 463 Spring 2018

Abstract

The Spring Final Report describes all the work our team has done on our completed project, explaining key features of the software and demoing how each component functions.

CONTENTS

1 Introduction to Project

For our senior design capstone project, we will build an image classifier on top of an autonomous robot. By leveraging ROS (Robot Operating System) and the existing mobile robot platform, we can devote all of our resources to sequentially training a Convolutional Neural Network (CNN) with online learning. In this project, we propose a plan of action and several potential solutions to the issues brought about from sequential learning. Additionally, there are multiple environmental variables that must be addressed during training in order to classify everyday objects in a wide variety of settings. To build a robust classifier, we will pursue three different methods to improve on the current system at the Personal Robotics Lab of OSU. These are: designing new data capture methods, overfitting to data in different environmental contexts using multiple classifiers, and testing different online learning models to achieve the best classification rate.

One of the biggest issues Machine Learning as a whole faces is that deep neural networks aren't very amenable to online learning techniques. When new data is passed through the network, weights change via backpropagation and previous "knowledge" is lost. This process is known as Catastrophic Interference and without the ability to sequentially receive and classify new training data, intelligent agents will have to be retrained on the entire data set whenever it is presented with a new input sample to predict.

Dr. Smart of the Personal Robotics Lab is our client, and he created the project as an extension of the ongoing studies being done at the lab. He proposed the project to look deeper into the image classification techniques being used in the lab, and improve upon the methods in place. The work is important as it will lay the foundation for future projects to expand upon our findings in the same manner. We are developing a pipeline designed to be used by other members of the lab, meaning the project has significance to the involved lab members. Our team consists of three members, Michael Rodriguez, Julian Weisbord, and Miles McCall. Each group member handled certain main functions of the pipeline and was responsible for the deliverable portion associated with the functions. Michael focused mainly of data gathering and the scripts associated with it, Julian worked on creating the image classification model and managed the project, and Miles focused on the transitional scripts and auxiliary functions to assist the online learning portion of the classifier. As a client, Dr. Smart attended regular meetings with our group members to keep in touch with our progress and offer insight on issues we ran into.

Group 67 - Requirements Document

Deep Learning for Object Recognition on a Mobile Robot

Julian Weisbord, Michael Rodriguez, Miles McCall

Oregon State University

CS 461 Fall 2017

Abstract

This paper goes in-depth into the requirements that we as a team must accomplish. We will be working to build an image classifier to work with our autonomous robot to help recognize basic objects in different environmental settings.

CONTENTS

1 INTRODUCTION

1.1 Purpose & Scope

This capstone project is in partnership with the Personal Robotics Lab (PRL) at Oregon State University (OSU). The overall goal of our research is to take a bare-bones robot and train it 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 into an Intelligent Agent. Currently their autonomous robots lack the ability to efficiently understand, learn, and classify objects in their environment with a high degree of certainty. By leveraging the ROS (Robot Operating System) libraries and the existing mobile robot platform, we aim to sequentially train a Convolutional Neural Network (CNN) with different online learning techniques. This CNN will then be capable of using robots as a data-collecting network to make itself smarter over time.

This is a research oriented project, which means the specific requirements are based on how we will both find and implement these machine learning algorithms. As a starting point, we will research algorithms that have worked well for similar projects. Once we have a substantial knowledge of common algorithms, we will apply them to the robots in the PRL of OSU. Moreover, this document proposes a plan of action for how robots like the PR2 and Fetch can be used to classify everyday objects in a wide variety of settings.

1.2 Definitions, Acronyms, & Abbreviations

Machine Learning: This evolved from the study of pattern recognition and computational learning theory in artificial intelligence,[4] machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Overfitting: Trained model only makes predictions for data set that you have trained on and not new data.

Back-propagation: Used to calculate the error contribution of each neuron after a batch of data is processed. Calculates gradient of loss function (gradient descent/ synthetic gradients). Back-propagation is a generalization of the Delta Rule.

Forward Propagation: We apply a set of weights to the input data and calculate an output. For the first forward propagation, the set of weights is selected randomly.

Neural Network Training: Forward Prop (start with random weights) sum products of the inputs with their corresponding set of weights to arrive at first values of hidden layer; apply activation function to hidden layers; sum product of hidden layer results with the second set of weights to determine output sum; take activation function at that output sum to get the final output result; Back Propagation [1]

- Output sum margin of error = target - calculated

Online Machine Learning (sequential learning): A method of machine learning in which data becomes available sequentially. The opposite of batch learning, Online learning updates its predictor for future data continuously with each new piece of data. Ex: If you were to show a child a ripe banana for the first time, they would learn what a banana looks like. But maybe next time they see a banana, it is actually green and not ripe. The child would still know it is a banana but now they would know that bananas can be different colors. This is Online learning. It is more common for neural networks to use batch learning. With batch learning, if we apply the same analogy to the robot, if it were to see a green banana, its definition would completely forget that bananas can also be yellow. So we need Online Learning so that our intelligent robot can keep learning new information while remembering old information.

Intelligent Agent: In artificial intelligence, an intelligent agent (IA) is an autonomous entity which observes through sensors and acts upon an environment using actuators (i.e. it is an agent) and directs its activity towards achieving goals.

Intelligent agents may also learn or use knowledge to achieve their goals. They may be very simple or very complex: a reflex machine such as a thermostat is considered an example of an intelligent agent. [2]

Catastrophic Interference: Catastrophic Interference becomes present when learning is sequential. Sequential training involves the network learning an input-output pattern until the error is reduced below a specific criterion, then training the network on another set of input-output patterns. A backpropagation network will forget information A if it learns input A and then input B. Catastrophic forgetting occurs because when many of the weights, where knowledge is stored, are changed, it is impossible for prior knowledge about past data to be kept intact. During sequential learning, the inputs become mixed with the new input being superimposed over top of the old input. To recognize multiple sets of patterns, the network must find a place in weight space that can represent both the new and the old output.One way to do this is by connecting a hidden unit to only a subset of the input units. [3] [4]

Elastic Weight Consolidation (EWC) Elastic Weight Consolidation is a technique used to prevent a neural network from forgetting the past data its learned while training on new results. To do this, the algorithm locks the weights generated by previous runs in a way where the values converge on low error rates for both old and new data. EWC uses a different loss function than a standard neural network architecture to accomplish this.

1.3 References

REFERENCES

[1] S. Miller, "Neural nets," https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/, 08 2015, (Accessed on 10/25/2017).

[2] S. J. R. Peter Norvig, "Artificial intelligence a modern approach," Prentice Hall.

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

[4] F. P. Ron Kohavi, "Glossary of terms," http://robotics.stanford.edu/~ronnyk/glossary.html, 1998, glossary.

1.4 Overview

To build a robust classifier for the robots, we will pursue three different methods to improve on the current system at the Personal Robotics Lab of OSU. These are: improving the data capture process, overfitting to data in different environmental contexts using multiple classifiers, and testing different online learning models to achieve the best classification rate. Project Goals:

- We will analyze the benefits of at least three different lifelong learning techniques added to the pipeline such as: Dual Memory Architecture, Functional Approximation, etc. This will allow the intelligent agent to continue to learn more about its environment, and thus be able to classify objects at higher and higher rates as time goes on.
- We will research different ways to improve the current image capture system in place at the OSU Personal Robotics Lab. A possible bottleneck of image recognition can occur when an image dataset contains too many different objects in one image. The team can improve this with more accurate image data capturing.
- The intelligent agent will be able to classify at least 3 different object classes(i.e: Dog, Mug, table) with a minimum of 80% accuracy. The intelligent agent should also be able to identify the differences between objects within the same class(i.e his mug vs. my mug).

2 OVERALL DESCRIPTION

2.1 Product perspective

The product we aim to deliver consists of a code base to be ran on a PR2 and/or Fetch robot with the ROS operating system. As a component of this larger existing system, our code base will append to and in some cases, replace current models running on the robots.

2.2 Product functions

Our software package encapsulates a few key functionalities. It will augment the robot's data capturing system with modern image identification algorithms, image tagging, and image recognition software. Next, the robot will take the image from its sensors and query the neural network classifiers that we have built in order to accurately classify the image. Finally, once the image has been successfully classified, the robot will update its classifiers with this image so that it can use it as an example when trying to classify other images. This is known as Sequential Learning or Online Learning. All of these functionalities currently exist within the running system, but we will improve upon these functions with algorithms that can classify at a higher rate.

2.3 User characteristics

The intended user of our software package primarily consists of individuals already involved in the robotics lab. These users have a high degree of knowledge on the major topics covered within our project, and already have training on how to interact with the robot and integrated systems. Beyond the lab, however, users expand into a much broader range of individuals. Its entirely possible that or models could be deployed on a robot in house settings, labs, or workplace environments. The end goal user will have a large spectrum of knowledge regarding the robot and encompassing software and use the robot in a supporting role to increase their productivity.

2.4 Constraints

Our project is largely framed by the technologies provided to us. We are given, the Fetch and the PR2 mobile robots, servers with NVDIA GPU's, and lab access. We are very much constrained by the quality of the camera/sensors and the overall movement capabilities. In regards to training, testing, and designing our machine learning algorithms, we will be utilizing the robotics lab's server environments. Our local/personal machines will likely lack the performance needed to effectively train our models.

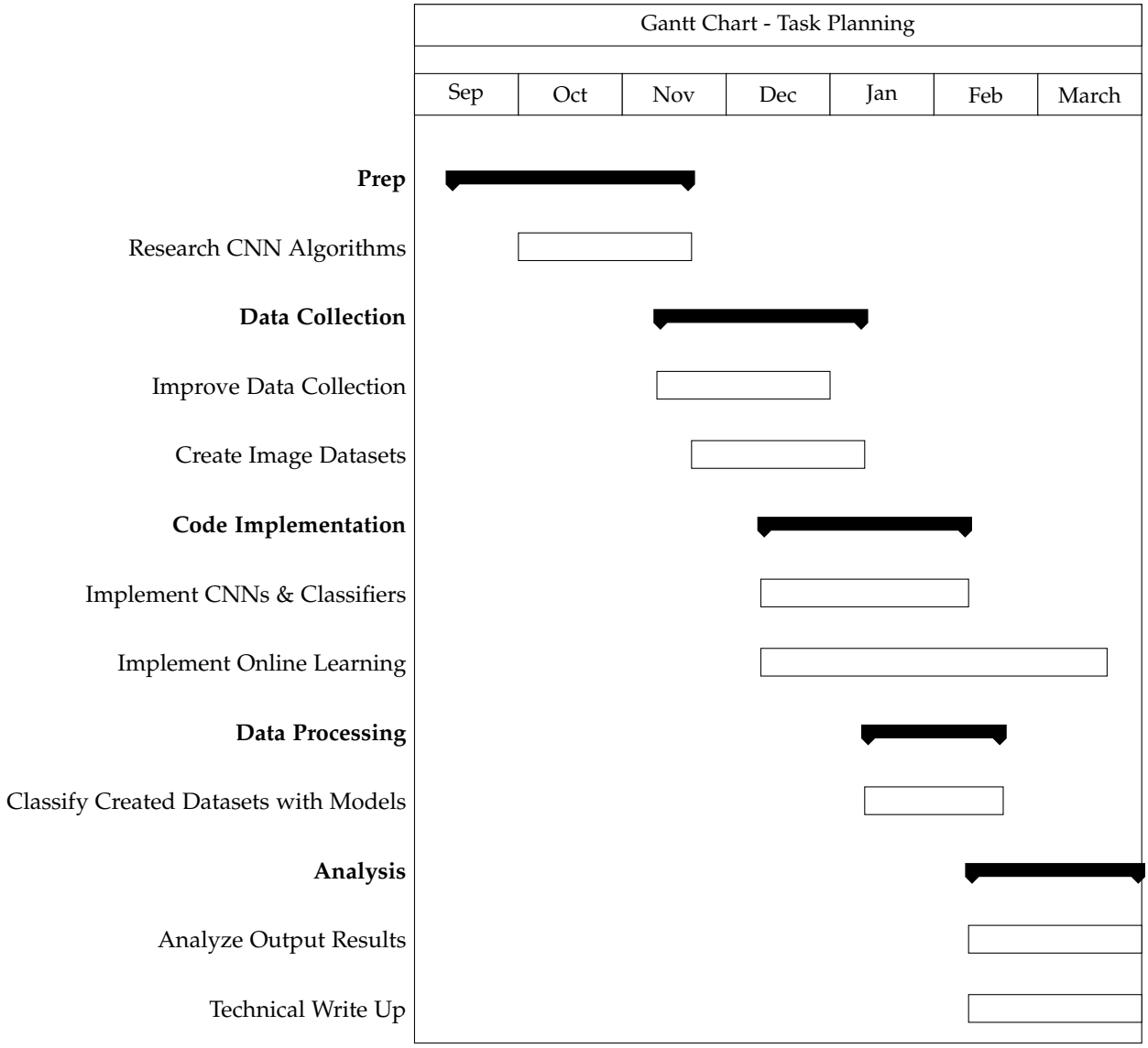
2.5 Assumptions and dependencies

Our overarching project depends entirely on the described and agreed upon robot hardware and operating system. The robots have different functionality and features which will affect the resulting software we create. The operating system, however, should remain more consistent across different platforms. Luckily we can assume the ROS platform handles all communications with the underlying hardware and robot accessories, so API's will make our code more portable and modular.

3 SPECIFIC REQUIREMENTS

- Researchers will research three different ways to improve the current image capture system in place at the OSU Personal Robotics Lab.
- Researchers will analyze the benefits of at least three different lifelong/online learning techniques (defined above) such as: Dual Memory Architecture, Functional Approximation, etc. They will use the most accurate Online Learning technique (one that remembers the most information) for the final classifiers.
- Researchers will implement two Convolutional Neural Networks to showcase the benefits of certain models for image classification.
- Researchers will evaluate the success of the different algorithms by how much the average rate of classification increases compared to the current classifier.
- The intelligent agent will classify at least 3 different object classes with a minimum average of 80% accuracy.
- Researchers will compile a comprehensive write up detailing the results of their findings.

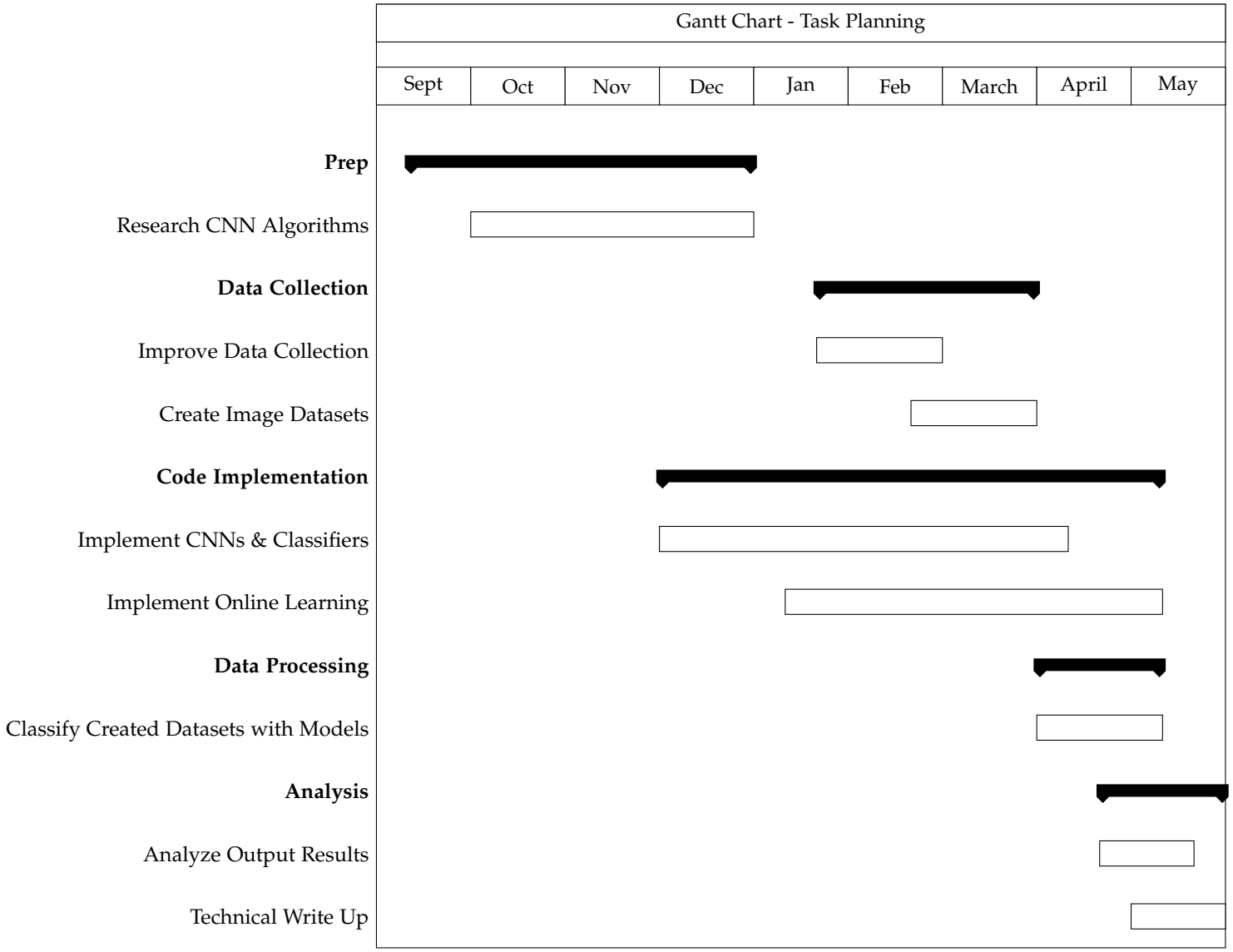
4 GANTT CHART



5 REVISION CHANGE LOG

- Definitions, Acronyms, &Abbreviations Section
  - Added: Elastic Weight Consolidation (EWC)
- Specific Requirements Section
  - Original
    - Researchers will research different ways to improve the current image capture system in place at the OSU Personal Robotics Lab.
      - They use a data capture system that utilizes AR tagging to show the robot where the image that it wants to capture is located. Researchers will research and implement several common methods to see which yields the most accurate classification rate in the end.
      - Researchers will document and report the benefits of each methodology in a technical writeup.
    - Researchers will analyze the benefits of at least 3 different lifelong/online learning techniques(defined above) added to the pipeline such as: Dual Memory Architecture, Functional Approximation, etc. Use the most accurate (remembers the most information) Online Learning technique in the final classifiers.
    - Researchers will implement at least 3 Convolutional Neural Networks that will be responsible for different aspects of image classification.
    - Researchers will evaluate the success of the different algorithms by how much the average rate of classification increases compared to the current working classifier.
    - The intelligent agent will classify at least 3 different object classes with a minimum average of 80% accuracy.
  - Revised
    - Researchers will research three different ways to improve the current image capture system in place at the OSU Personal Robotics Lab.
    - Researchers will analyze the benefits of at least three different lifelong/online learning techniques (defined above) such as: Dual Memory Architecture, Functional Approximation, etc. They will use the most accurate Online Learning technique (one that remembers the most information) for the final classifiers.
    - Researchers will implement two Convolutional Neural Networks to showcase the benefits of certain models for image classification.
    - Researchers will evaluate the success of the different algorithms by how much the average rate of classification increases compared to the current classifier.
    - The intelligent agent will classify at least 3 different object classes with a minimum average of 80% accuracy.
    - Researchers will compile a comprehensive write up detailing the results of their findings.

2.1 Final Gantt Chart



Group 67 - Preliminary Design Document

Deep Learning for Object Recognition on a Mobile Robot

Julian Weisbord, Miles McCall, Michael Rodriguez

Oregon State University

CS 461 Fall 2017

Abstract

The design document provides a step by step explanation of specific algorithms, concepts, libraries, and testing frameworks that will be necessary to build an intelligent agent capable of continually building upon its knowledge base. This document lays out specific design choices we will follow through the rest of the project.

CONTENTS

1 INTRODUCTION

1.1 Scope

This is a research-oriented project, which means the specific requirements are based on how we will both find and implement these machine learning algorithms. Moreover, this document proposes a plan of action for how robots like the PR2 and Fetch can be used to classify everyday objects in a wide variety of settings.

1.2 Purpose

The overall goal of our research is to take a bare-bones robot and train it 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 into an Intelligent Agent. Currently their autonomous robots lack the ability to efficiently understand, learn, and classify objects in their environment with a high degree of certainty. By leveraging ROS (Robot Operating System) and the existing mobile robot platform, we aim to sequentially train a Convolutional Neural Network (CNN) with different online learning techniques.

This software design document describes the architecture and system design of our image classification pipeline, laying out major decisions and why they were made.

2 DEFINITIONS

Machine Learning: This evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data [1]

Overfitting: Trained model is biased towards the training data and does not learn the overall trend, leading to reduced accuracy on new data sets.

Backpropagation: Used to calculate the error contribution of each neuron after a batch of data is processed. Calculates gradient of loss function (gradient descent/ synthetic gradients). Backpropagation is a generalization of the Delta Rule.

Forward Propagation: We apply a set of weights to the input data and calculate an output. For the first forward propagation, the set of weights is selected randomly.

Neural Network Training: Steps we used to train the network:

1) Forward Prop (start with random weights)
2) Sum products of the inputs with their corresponding set of weights to arrive at first values of hidden layer
3) Apply activation function to hidden layers
4) Sum product of hidden layer results with the second set of weights to determine output sum
5) Take activation function at that output sum to get the final output result
6) Back Propagation [1]
   -Output sum margin of error = target - calculated

Online Machine Learning (sequential learning): A method of machine learning in which data becomes available sequentially. The opposite of batch learning, Online learning updates its predictor for future data continuously with each new piece of data. Ex: If you were to show a child a ripe banana for the first time, they would learn what a banana looks like. But maybe next time they see a banana, it is actually green and not ripe. The child would still know it is a

banana but now they would know that bananas can be different colors. This is Online learning. It is more common for neural networks to use batch learning. With batch learning, if we apply the same analogy to the robot, if it were to see a green banana, its definition would completely forget that bananas can also be yellow. So we need Online Learning so that our intelligent robot can keep learning new information while remembering old information.

Intelligent Agent: In artificial intelligence, an intelligent agent (IA) is an autonomous entity which observes through sensors and acts upon an environment using actuators (i.e. it is an agent) and directs its activity towards achieving goals. Intelligent agents may also learn or use knowledge to achieve their goals. They may be very simple or very complex: a reflex machine such as a thermostat is considered an example of an intelligent agent. [2]

Catastrophic Interference: Catastrophic Interference becomes present when learning is sequential. Sequential training involves the network learning an input-output pattern until the error is reduced below a specific criterion, then training the network on another set of input-output patterns. A backpropagation network will forget information A if it learns input A and then input B. Catastrophic forgetting occurs because when many of the weights, where knowledge is stored, are changed, it is impossible for prior knowledge about past data to be kept intact. During sequential learning, the inputs become mixed with the new input being superimposed over top of the old input. To recognize multiple sets of patterns, the network must find a place in weight space that can represent both the new and the old output. One way to do this is by connecting a hidden unit to only a subset of the input units. [3] [4]

3 Technology Overview

3.1 Operating System

We are utilizing Ubuntu as the main operating system (OS) on our project to control both the intelligent agent and all computers that communicate with it. The intelligent agent's on-board OS functions like any other computer, but runs ROS (Robot Operating System) which is an Open Source set of libraries that provide a series of high-level functions to help users write software for mobile robots. The computer used to communicate with the intelligent robot must also run Ubuntu and ROS to create an effective channel for robot communication and data transfer. Programs such as Rviz, Gmapping, and Mapsaver are all used as auxilary programs to aid us in gathering our test data. Two key concepts within ROS we heavily rely on are packages and the stack. Packages are directories containing external libraries, API data, configuration files, nodes and an XML configuration file. The stack is a collection of packages that offers functionalities like navigation, positioning, mapping, and much more [5].

3.1.1 Method

We utilize ROS as our method to manipulate the fetch robots sensors, positioning, mapping, and cameras. ROS libraries provide the system with the necessary tools to map an entire room so the fetch robot can familiarize itself with its test environment. Additionally, ROS will help us move the robot 360-degrees around the target object so that we are able to gather our training data set. ROS also provides us with methods to implement an image tagging technique, involving radio waves and RFID tags to locate a tagged object in its environment.

Ubuntu is the operating system used on our project to control both the intelligent agent and all computers that communicate with it. The intelligent agent's on-board OS functions like any other computer, but runs a specialized program called ROS (Robot Operating System) to control all ”robotic” functions. The computer used to communicate with the intelligent agent must also run Ubuntu and ROS, as the software used alongside ROS must operate through it. Programs such as Rviz, Gmapping, and Mapsaver are all used as auxilary programs to aid us in gathering our test data.

While ROS is not a true operating system in the same sense Linux is, the program runs constantly in the background of the intelligent agent managing everything from sensors, to motors, to inter-process communication. Compontents of the intelligent agent are registered as devices under ROS and controlled similarly to typical device drivers.

ROS is also the most common operating system for robotic systems. It is open source and has BSD licensing which are two big reasons we chose it as part of our design. Being open source is one of the main reasons that RuOS has evolved so quickly and become so prominent in the field of robotics research and development. Two key concepts exist that stand out more than the others and that is the package and the stack. The package is a directory that has external libraries, API data, configuration files, nodes and an XML configuration file. The stack is a collection of packages that offer functionalities like navigation, positioning, mapping, and others [5].

3.2 Data Gathering Methods

Data gathering is one of the most pivotal parts of the project. Without data, the intelligent agent has nothing to run through its neural network to classify objects. Gathering a training set of images large enough for the robot to train on is typically expensive on computational resources and time.

3.2.1 Five Image Classes

We have decided as a team to focus on five main image classes. These image classes will consist of mugs, staplers, chairs, books, and screwdrivers. We picked objects that we thought would be natural to find in the lab environment thus making testing a variety of objects easier. We also believe that picking five distinct looking objects would be best because it allows us to give our fetch robot a wide range of knowledge on how different object shapes can look in the real world.

3.2.2 Method

Our method for gathering data on the five image classes involves having the fetch robot take roughly an hour of time going around an object taking 50 pictures at varying angles to create a training data image set. The fetch robot is then going to repeat this process for five objects per image class, so a total of 1,250 pictures will be taken. This will be the bare minimum of raw image data that the fetch robot requires to build its full training data set.

3.2.3 Results

Once image data has been gathered on all 5 object, the images will be used to create a validation and data training data set for the overarching classifier (in section 3.4). The validation data training set consists of a small number of images that is going be used by the fetch robot as a reference key for when it is trying to identify an object after having already been trained on it. The training set consists of the full raw image data set that gets sent through the overarching classifier to help it classify images successfully.

3.3 Image Tagging Techniques

Image tagging involves using a marker system that involves a set of patterns that our robot can detect and familiarize itself with. The purpose of having a marker system is that it helps the robot orient itself in the environment that it presides in. For our testing purposes, we want the robot to pin the marker on its coordinate map of the room generated from its sensors. This specific spot on the map will serve as the testing center where we will be placing objects for the robot to train on.

3.3.1 Method

Having the robot take more accurate images of the objects in its environment is the primary goal of our data gathering program. In order to do so, our intelligent agent uses ROS libraries to make a radar scan of its environment, identifying all objects that would obstruct its path. After the mobile robot scans the room, we are able to add points of interest in 3D space to the intelligent agent's map. With these tags, the robot can identify objects to classify and take a series of pctures capturing all angles of the object. This tagging system provides us with the most accurate data set for our research with limited oversight of the overall process.

3.3.2 Results and Data Analysis

Once the fetch robot has successfully been able to locate the object it can begin to create a training data set on it. After the training data set is finished being created, the fetch robot is either going to add on to its pre-existing knowledge of that object or create a new instance of that object under the appropriate image class classifier. In the event that it does not fit either scenario the fetch robot should simply disregard the object.

3.4 Overarching Classifiers

3.4.1 Method

We plan to implement multiple layers of classifiers where each layer feeds its results to the next, increasing our algorithms accuracy and specificity. The first classification layer is the most general. Its goal as the overarching classifier is to try to match the object to one of our predefined classes. If the classifier cant meet the initial threshold required for us to consider the object a certain class, the classifier must file the object in a miscellaneous category, as it is not capable of recognizing unknown object classes.

3.4.2 Training and Validation

The overarching classifier is trained on our predetermined classes: mugs, staplers, chairs, books, and screwdrivers. Each class has a folder containing fifty images of each object, with five unique objects per class. This gives the training set enough variety between general class shape and individual detail, while still a manageable amount of raw data to process.

Our classifier pipeline design creates an increasing level of validation accuracy as the results are passed onto the next layer of classifiers. This allows us to start with a more open-ended object and slowly narrow down the possible results. Following this design, our overarching classifier will be the least accurate of the three layers. Its goal is to simply choose class categories based on the main distinguishing features of each class.

3.4.3 Results

The first classifier can select multiple classes it considers valid as the more specialized classifiers will effectively double check the results. After determining one or more classes the object could potentially belong to, the first classifier then initiates a set of classifiers for each potential class that focuses on the specific features of said class.

3.5 Sub-Classifiers

3.5.1 Pipeline Architecture

The sub-classifiers can be split into the second and third layer of the waterfalling classifier architecture. The overarching classifier passes its initial results to the relevant class specific classifiers in the second layer. These classifiers verify

whether the initial prediction chose the right class. Only one group of sub-classifiers should return a confirmation, which narrows the individual object down to one class. The second layer then passes this result to the third layer, an even more specialized group of classifiers that attempt to distinguish unique features on the object, narrowing the object down to one object it has been trained on.

3.5.2 Training and Validation

While the overarching classifier has examples of all classes in its training library, the sub-classifiers only refer to images of objects in one class. The second layer sub-classifiers determine if the object does or does not match the associated class based on its narrower image library. If a group of second layer classifiers returns a match, we can proceed with relative certainty that the object has been accurately validated.

3.5.3 Results

The second layer sub-classifiers activate the third layer classifiers if a match is confirmed, and terminate otherwise. The third layer is activated in this way, and refers to the same image data set as the two above layers. This group of classifiers compare the unknown object to an image library with detailed pictures of distinguishing features of each object. It will then return either an object match, or finally classify the object as unknown or miscellaneous. This can mean two things, however, either the object was a false positive for that class and was mistakenly validated, or the object does belong to the class but has yet to be added to the training library. If the latter is the case, sequential learning is utilized to append the newly discovered object to the object class.

3.6 Overfitting Algorithms

3.6.1 Method - Overfitting Throughout the Pipeline

The concept of overfitting in machine learning applications refers to an algorithms classifiers being too specialized to the specific training set. Classifiers can be overfitted to the data, underfitted, or an acceptable balance can be found between the two. A classifier that is overfitted will have a stricter threshold of validation required to recognize an object as a certain class, while an underfitted classifier will return more false positive matches as it is less picky with its validation.

In our pipeline, we attempt to utilize overfitting to focus on specific objects in an environment with little changes being made to it. By having an amount of overfitting present in each level of the classifier pipeline, we can build an increasing certainty that the algorithm is generating an accurate validation. The initial, overarching classifier is the least overfitted to the data as it is classifying an object into multiple different classes. The sub classifiers apply a higher level of scrutiny to the same object and wont pass on a positive match to the next layer without the object meeting an increasingly higher threshold. This is how we are applying overfitting to the pipeline to generate more confident results at the end.

3.7 Backpropagation

In Deep Neural Networks, Backpropagation is the main technique used by a neural network to update its weights or learn how to correctly predict the output value of your input data. Without backpropagation, neural networks would cease to be useful because they wouldnt be accurate. Backpropagation gets its name from the process that it implements. Backpropagation propagates neural network accuracy updates by going backwards, starting at the output layer of the network and ending at the first layer of the network.

3.7.1 Synthetic Gradients Algorithm

Synthetic Gradients is an algorithm that was recently developed by Deep Mind and has already shown greater prediction accuracy than Stochastic Gradient Descent and Batch Gradient Descent [6]. The only down side is that it requires very powerful GPUs and is the most computationally expensive of the 3 algorithms. One of the issues with Stochastic and Batch Gradient Descent is that the neural network can only update after a full forward and backwards pass of itself. This can be thought of as a serial process, however the Synthetic Gradients Algorithm accomplishes backpropagation with a more parallel process [7].

The benefit of Synthetic Gradients is that when neural networks become more complex with a lot of layers, the time it takes to update the weights in the network becomes a massive bottleneck. Synthetic Gradients is an algorithm that decouples the layers of the neural network and updates these layers all independently from one another so that each layer no longer has to wait for all of the other layers to update.

3.7.2 Results and Data Analysis

Batch Gradient Descent is the most common method of backpropagation and is very easy to implement so it is a perfect way to test how effective Synthetic Gradients is in comparison. To formally convey the results, we will conduct a quantitative 2-sample data analysis.

3.8 Types of Online Learning

Online Learning or Sequential Learning or Lifelong Learning is the concept that intelligent agents should be able to learn more about their environment and previous knowledge throughout its lifetime. For example, a child will likely know what a laptop is but when they learn something knew about laptops such as the differences between Microsoft and Mac, they will add this information to their memory. To put it bluntly, not only Neural Networks, but AI in general is very bad at this seemingly basic process. This process is Online Learning. The following technique is how online learning can teach an intelligent agent new ways to recognize data that would be helpful in our project.

3.8.1 Same Objects, New Orientation (SONO)

This process involves learning old data better. An example of this would be if the robot/intelligent agent learned what a mug looked like in normal usage but then the robot is presented with an upside-down mug, and expected to classify it as a mug. The benefit of SONO is that it isnt too difficult to implement and there are research papers written that use similar algorithms for their research purpose. The down side is that it isnt that useful, the robot will just learn its current environment better. For example: Just because we trained the agent on a bunch of mugs doesnt mean it will suddenly know what a wine glass looks like.

3.8.2 Results and Data Analysis

In the end, SONO will allow the intelligent agent to train itself on an endless number of images of an object. If the agent is continuously trained, this will give it knowledge of the object from seemingly every angle.

3.9 Mitigating Catastrophic Interference

Catastrophic Interference or Catastrophic Forgetting is an issue that comes about from Online Learning. A backpropagation network will forget information if it learns input A and then input B. Catastrophic forgetting occurs because

when many of the weights in a neural network, where knowledge is stored, are changed, it is impossible for prior knowledge about past data to be kept intact. During sequential learning, the inputs become mixed with the new input being superimposed over top of the old input. To recognize multiple sets of patterns, the network must find a place in weight space that can represent both the new and the old output.

3.9.1 Elastic Weight Consolidation (EWC)

The technique of elastic weight consolidation has proven effective at training a network on new data without forgetting previous results. This is accomplished by locking the weights generated by the previous runs in a way where the values converge on low error rates for both old and new data. EWC uses a different loss function to achieve these results. This is an integral part in implementing a sequential learning system as the network must handle new input effectively.

3.9.2 Results and Data Analysis

This project uses EWC to ideally improve the agent's rate of classification over time. Without using EWC, the agent will only remember the information it learned in its last training epoch which most likely will not be as high of classification rate. This gives us something to compare EWC against.The results of using a method to mitigate catastrophic interference should be huge in comparison to an online learning technique that does not. We will conduct a quantitative 2-sample data analysis to show a significant increase in the rate of successful classification when using Elastic Weight Consolidation.

4 Conclusion

There are two big steps that must be implemented to create an intelligent agent that can correctly recognize objects in its environment, these are the Data Collection phase and the Image Recognition phase. For an agent to be able to recognize and predict things about it's environment, this agent must first be trained. For this project, the agent is trained on images from varying angles of common objects in a lab environment. The agent will be provided with pictures from 5 object classes: mugs, staplers, chairs, books, and screwdrivers. Once several images are gathered and labeled, the agent will train itself on this dataset. This is the first step in actual image recognition [3]. The main overarching classifier is trained on images from the 5 object classes. This classifier will output a percentage likelihood for each object class and then will pass the highest class prediction percentage to sub-classifiers that can more accurately classify the object because they specialize in the specific features to that object class. Finally, there is an additional step to image recognition, that when implemented, allows the agent to continually learn new information about these objects over time, this is called Sequential Learning [8]. EWC is implemented in order to maximize the amount of new and old information that the intelligent agent can encode, store, and retrieve [9].

There are two large systems that must be implemented to create an intelligent agent that can correctly recognize objects in its environment, these are the Data Collection phase and the Image Recognition phase. For an agent to be able to recognize and predict things about it's environment, this agent must first be trained. For this project, the agent is trained on images from varying angles of common objects in a lab environment. The agent will be provided with pictures from 5 object classes: mugs, staplers, chairs, books, and screwdrivers. Once several images are gathered and labeled, the agent will train itself on this dataset. This is the first step in actual image recognition [3]. The main overarching classifier is trained on images from the 5 object classes. This classifier will output a percentage likelihood for each object class and then will pass the highest class prediction percentage to sub-classifiers that can more accurately classify the

object because they specialize in the specific features to that object class. Finally, there is an additional step to image recognition, that when implemented, allows the agent to continually learn new information about these objects over time, this is called Sequential Learning [8]. A DMA is implemented in order to maximize the amount of new and old information that the intelligent agent can encode, store, and retrieve [9].

REFERENCES

[1] S. Miller, "Neural nets," https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/, 08 2015, (Accessed on 10/25/2017).

[2] P. Norvig and S. J. Russel, "Artificial intelligence a modern approach," Prentice Hall.

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

[4] R. Kohavi and F. Provost, "Glossary of terms," http://robotics.stanford.edu/~ronnyk/glossary.html, 1998, glossary.

[5] V. Mazzari, "Ros robot operating system," https://www.generationrobots.com/blog/en/2016/03/ros-robot-operating-system-2/, 03 2016, (Accessed on Nov 12).

[6] S. Raval, "Synthetic gradients explained," https://www.youtube.com/watch?v=qirjknNY1zo, 10 2017.

[7] Czarnecki, "Understanding synthetic gradients and decoupled neural interfaces," https://deepmind.com/research/publications/understanding-synthetic-gradients-and-decoupled-neural-interfaces/, 03 2017.

[8] Kirkpatrick and R. Pascanua, "Overcoming catastrophic forgetting in neural networks," Deep Mind.

[9] McDermott and Roediger, "Memory," http://nobaproject.com/modules/memory-encoding-storage-retrieval, washington University in St. Louis.

3.1 Changes to Design Doc

Our design document required only minor modifications that we had approved by Dr. Smart and McGrath and Kirsten. We removed a term definition that did not end up being included in the final project design.

Technology Review

Michael Rodriguez

Oregon State University

Group 67 - CS 444 - Fall 2017

Abstract

For this senior design project we will be partnering with the OSU Robotics department to help train an autonomous robot recognize regular objects. We will try to accomplish this by using ROS (Robot Operating System), trying different image tagging techniques, and improving data gathering.

1 INTRODUCTION

This project is formally called, ”Deep Learning for Object Recognition on a Mobile Robot” and my role involves improving data gathering methods and using robust image tagging techniques. The project as a whole however will use neural networks to help train the robot to recognize regular objects and classify them into the proper class. In addition to the neural networks, the robot will also utilize online learning techniques which will help the robot build knowledge on pre-existing knowledge that it may already contain from looking at an item in the past. This is a place where we will be majorly invested in because the robotics department has yet to implement this type of learning on the robot. My role in this project is more tilted towards improving the techniques to gather data for the robot so that the learning can be more efficient and effective than it currently is. Therefore, in this technology review I will be looking at different technologies that are available to help me choose the best operating system for the robot, test out different image tagging techniques and research different data gathering methods.

2 OPERATING SYSTEM FOR ROBOT ACTUATION (OVERVIEW)

An operating system for a robot basically serves as the brain of the robot. The operating system provides the tools necessary to make the robot do an unimaginable number of different things. One of those things that it can help us out with is object recognition and classification. In this section, I will be discussing the potential operating systems that our PR2 robot could be running.

2.1 ROS

ROS (Robot Operating System) is the most commonly known and used operating system for robots. The main idea behind the creation of ROS was mainly to offer standardized functionalities that perform hardware abstraction and helping those in robotics development from having to continue rebuilding the wheel on their own. ROS is maintained by a company called Willow Garage which develops software and hardware for their robots which conveniently happens to include the PR2. Everything that is produced by Willow Garage is open source and has BSD licensing. Being open source is one of the big reasons that ROS has evolved so quickly and become so prominent in the field of robotics research and development. ROS provides resources that are organized into a hierarchical structure on disk. Two key concepts exist that are stand out more than the others and that is the package and the stack. The package is a directory that has external libraries, APIs data, configuration files, nodes and an xml configuration file. The stack is a collection of packages that offer functionalities like navigation, positioning, mapping, and others [1].

2.2 Microsoft Robotics Developer Studio

The Microsoft Robotics Developer Studio is a Windows-based environment used to help create robotic applications. Some of the perks that RDS boasts is a lightweight REST-style, service-oriented runtime, a set of visual authoring and simulation tools, tutorials and sample code to help developers get started. RDS makes programming simple by allowing asynchronous input from the multiple sensors on the robot and output to actuators and motors. This would help our robot improve the speed on how fast it trains itself on an object by expediting the process. RDS also has a DSS Service Oriented Architecture that helps make it simple to interact with the robot through a web browser or windows-based application [2]. By being able to interact and respond to our robot through such an interface, it would benefit the feedback compartment of our project where the robot periodically checks in with us to confirm that it is targeting the right object.

2.3 NAOqi

NAOqi is the software that runs and controls the robot. The NAOqi framework which is used to program NAO solves basic robotic needs such as parallelism, resources, events, and synchronization. This framework is versatile in that it allows development in Windows, Linux, and Mac OS. NAOqi is similar to ROS in that its API functions for both C++ and Python and its modules are transparent between both languages as well [3]. Although this operating system seems to have the same major components that ROS has, it doesnt have nearly as much open source content and documentation. Additionally, NAOqi doesnt have a visual interface that we can interact with to verify that our robot is focusing on the correct object.

2.4 Conclusion

The operating system we decided on to use on the robot is ROS. This choice was a no brainer in that the PR2 robot we will be working with was actually developed by Willow Creek, the same guys that started and maintain ROS. In addition to being related to the same company, Oregon State University is the primary hosting site for ROS and every roboticist in the lab uses it. This will make it easier for us to reach out to faculty in the robotics department for help since we will be speaking the same robot language as them. Our client Professor Smart is also fond of ROS, so we ultimately did not have a say in what operating system we wanted to use.

3 Image Tagging - Object Selection with Interface

Image tagging involves using a marker system that involves a set of patterns that our robot can detect and familiarize itself with. The purpose of having a marker system is that it helps the robot orient itself in the environment that it presides in. For our testing purposes, we want the robot to pin the marker on its coordinate map of the room generated from its sensors. This specific spot on the map will serve as the testing center where we will be placing objects for the robot to train on.

3.1 Vision Markers

Vision Markers in computer vision are typically either squared or circular. The images contained within these two shapes just has to be something that the robots camera is able to recognize. The size of the marker is not a super important, what is important is that you are able to maximize the distance from which the robot can successfully detect the marker. The shapes have individual advantages that make it better or worse depending on the goal at hand. Squares can be accessed uniformly using homography and unique vertices to orient the camera in a unique way. This allows for square tags to carry a larger symbolic data payload when used. Circular tags on the other hand can be accessed from any camera angle by computing from the whole contour surrounding the marker. This makes circular tags have better location and pose accuracy [4]. For our project, we want to be able to do a full 360 rotation around the object from varying angles so having circular tags and their ability to be recognized from all angles is a huge upside.

3.2 RFID

Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags that are near or on objects. RFID work great with robots because they are a low-priced sensor that eliminate the problem of robots having difficulty visually identifying tags. It doesnt matter if it is light or dark out, whether the object is right side

up, sideways, or upside down, or even if its in a different room. The robot can be equipped with long range UHF RFID antennas that allow it to roam a room and see where the strongest signal from the RFID tags is coming from /citeRFIDtag. The robot is essentially playing the game hot or cold with the tag until it is able to locate it. The method of using RFID tags would be the most accurate and have the best long-range detection capability if implemented correctly.

3.3 Chemical Markings

Chemical markings can be used as a technique for storing temporary information in an environment. These markings eventually fade away once their purpose has been fulfilled for a certain amount of time. Similar to how dogs and cats mark their territories with pheromone markings, robots can use the same technique to send/receive a message. These short-term markings can be referred to as short-lived navigational markers (SLNMs). These SLNMs can be detected with sensors that our robot possesses and interpret its meaning by us telling it what exactly it should be associating with that specific chemical [5]. This technology isnt super practical for our testing purposes and is a little outdated. In most cases we want our marker to be permanent for the testing environment so that it can always use it as a frame of reference.

3.4 Conclusion

For the image tagging technique we decided on using the RFID tagging method. This method not only saves us the valuable computational resources needed to identify your average visual marker, but also allows us to utilize our robots sensors to their full potential. RFID tagging is capable of increasing our maximum distance from the robot to the tag while being more accurate and efficient in finding said tag.

4 Data Gathering

Data gathering is one of the most pivotal parts of this project. Without data, the robot has nothing to run through its neural network to classify objects. Gathering a training set of images large enough for the robot to train on is typically expensive on computational resources and time. In this section, I am going to explore different data gathering techniques to see which one suits our needs best.

4.1 Traditional Approach

The traditional data gathering approach involves first taking pictures of the object in its natural environment. Next, the robot either searches the environment for the object or it uses object instances gathered prior to the test to place the object in the environment in a natural way. The next step involves hand-tagging the pictures marking bounding boxes or tight outlines in each picture [6]. This last step repeats as the robot goes 360 degrees around the object gathering raw picture data for the object. This method is not the most accurate because the quality of the dataset relies heavily on the human input required to get the pictures. A similar data gathering approach is currently being used in the OSU robotics lab, which means there is plenty of room for improvement.

4.2 Synthetic Approach

A synthetic approach implemented by a group of students at Stanford University involves taking pictures of the object in a green screen setting. The next step is to build a new larger training dataset by perturbing foreground, background,

and shadow of the images taken using a probabilistic model. This allows for the true distribution of an object class to be modeled just as good as if it were to have been real data [6]. This approach saves a large amount of man hours that would have otherwise been needed for hand-labeling the pictures for the dataset. Our current lab resources do not offer the green screen technology needed to implement such a solution but it is something that we can bring to the attention of our client. The robot currently takes roughly about over an hour to gather its dataset to train on for one object so decreasing the amount of time needed to create a dataset would be fantastic.

4.3 Flashlight Approach

A data gathering approach that I believe will help us gather better quality pictures in our training set is called the flashlight approach. One of the biggest obstacles that we face when trying to classify an object is the amount of light exposure that it receives and having that affect the quality of the images we gather. To make the testing more consistent across all objects, I suggest that the robot hold a flashlight up to the object that it is capturing images on. This will allow the robot to get a much clearer image of the object and keep it consistent across all tests. This approach might produce a bit of glare depending on the material of the object but since the robot currently relies on human interaction with the hand-labeling, we can selectively choose to not include sections with glare in it. This last approach is a bit of a stretch but is something worth trying on at least one test to determine if it is a significant enough increase on the quality of the training set.

4.4 Conclusion

The data gathering method that we decided to go with is the synthetic approach. The main advantage of this approach is that the results you get from it are nearly the same as if you were to have gathered real life pictures of the object in optimal conditions. Only difference of course being the process is being done synthetically and saves our team a huge amount of time in the lab for when we want to gather new training sets on new or old objects.

REFERENCES

[1] V. Mazzari, "Ros robot operating system," https://www.generationrobots.com/blog/en/2016/03/ros-robot-operating-system-2/, 03 2016, (Accessed on Nov 12).

[2] Microsoft, "Welcome to robotics developer studio," https://msdn.microsoft.com/en-us/library/bb648760.aspx, (Accessed on Nov 12).

[3] "What is naoqi framework," http://doc.aldebaran.com/1-14/dev/naoqi/index.html, (Accessed on Nov 12).

[4] J. K. A. Pagani and D. Stricker, "Detection and identification techniques for markers used in computer vision," http://vesta.informatik.rwth-aachen.de/opus/volltexte/2011/3095/pdf/6.pdf, 01 2010, (Accessed on Nov 13).

[5] R. A. Russell, "Odour detection by mobile robots," https://books.google.com/books?id=O2FTjO9X2xQC&pg=PA117&lpg=PA117&dq=Marking+techniques+for+robots&source=bl&ots=uODndBxYdz&sig=jbNGH4OUaFGM2tnXnsL6dVSCTTM&hl=en&sa=X&ved=0ahUKEwjNwPHs-r3XAhVMwmMKHTlPBN0Q6AEIRTAH#v=onepage&q=Marking%20techniques%20for%20robots&f=false, 07 1999, (Accessed on Nov 13).

[6] B. Sapp, A. Saxena, and A. Y. Ng, "A fast data collection and augmentation procedure for object recognition," http://ai.stanford.edu/~asaxena/robotdatacollection/stairdatacollection.pdf, 2008, (Accessed on Nov 13).

Group 67 - Technology Review

Deep Learning for Object Recognition on a Mobile Robot

Miles McCall

CS 461 Fall 2017

Abstract

The Technology Review is used to break the project down into its core components with three main points describing each section. For each point, three technology options are compared and contrasted, and the one most likely to be implemented for the project is explained.

CONTENTS

1 ROLE

Our group has three members, and we all have equal responsibility across the project. At this point we are all working on every piece of the project to establish a consistent knowledge base. Further along into developing our pipeline we may divide tasks more and gain more expertise in a specific portion of the project, but for now we are all working together equally.

I am particularly interested in the image classification and neural network involved in the project. While the whole project is technically about deep learning and object recognition, the portion that covers the details, implementation, and design philosophy behind these sections is what I would prefer to focus on.

2 WHAT YOU ARE TRYING TO ACCOMPLISH

Our group is working to rewrite and improve upon the current software on the Fetch robot at the Personal Robotics Lab on campus. Through this redesign we aim to recreate the data collection and image recognition systems currently in place.

While the data gathering software currently being used on the robot, or intelligent agent, is likely adequate in its current state, our group would like an improved version to match the new classification design as well. By adding tag sensing to the current object location procedure, we hope to make image localization more accurate. The intelligent agent will be able to scan its environment searching for the tags placed on each test object. With a better measurement of where the object exists in 3D space, the intelligent agent will better center itself around the object and take more on-target images. This will improve the quality of our input and training data sets, and allow the classification pipeline to learn and process on more accurate representations of the objects.

Part of the way we plan to improve the image classifier is by implementing a Sequential Online Learning neural network model, as opposed to the currently running Batch Learning model. The goal is for the robot to be capable of training itself with an initial dataset, then gathering further data as needed to make the CNN more accurate. By using a more complicated model we should be able to append to the created datasets the robot already recognizes.

I plan to improve and build upon my own skills and knowledge base throughout the course of the project. While we will all be contributing to every part of the design process, I wish to become especially competent in the implementation

of our full classification pipeline. I'd like to accomplish a full hierarchy of neural networks with sequential learning implemented over the top of the classifiers.

3 TECH REVIEW

In this technology review I will be focusing on the first half of our image classification program, specifically types of image classifiers and sub classifiers, and how overfitting applies to our project.

3.1 Project Overview

- Data Collection
  - ROS for robot actuation
  - Image Tagging - Object Selection with Interface
  - Data Gathering
- **Image Recognition Part 1**
  - **Overarching Classifiers**
  - **Sub Classifiers**
  - **Overfitting in the algorithms**
- Image Recognition Part 2
  - Backpropagation Methods
  - Types of Online Learning
  - Minimizing Catastrophic Interference

3.2 Image Recognition Part 1

3.2.1 Overarching Classifiers

The most critical program in our pipeline is the classification model, as it performs the most complicated task and actually predicts the object's class. Data prediction is an increasingly large field and collaboration is common practice in the industry to help grow the public knowledge base. Scientific literature on the subject of classification is readily available, and I found an article detailing a broad overview of modern techniques. Image classification is particularly complicated and can be accomplished numerous ways.

In *"A survey of image classification methods and techniques for improving classification performance"*, authors D. Lu and Q. Weng "examine current practices, problems, and prospects of image classification". The review notes that choosing an appropriate algorithm and designing a suitable processing procedure is essential to a successful image classifier, making this one of the most critical decisions for the entire pipeline. I am looking for modern approaches to our problem that take full advantage of the processing resources we have available to us, while still having the backing of the community and proper support resources available to developers. The article emphasizes nonparametric classifiers that allow for a dynamic number of model parameters, specifically neural networks, decision tree classifiers, and knowledgebased classification. Based on the analysis in the review, our group has agreed upon a convolutional neural network for image classification. In recent years the algorithm has been widely adopted for the task due to advantages such as "arbitrary decision boundary capability, easy adaptation to different types of data and input structures, ... , and generalization

for use with multiple images". Flexibility and generalization will make training the neural network possible with the number of image classes we want to include in the data set, and may come into play again when we investigate data overfitting. [1]

3.2.2 Sub Classifiers

While my research into literature regarding classifiers compared different methods and algorithms in a concise manner, our project is designed to be more specific than just using pre-existing libraries predict image classes. We are aiming to implement a classification model designed by our group, incorporating two distinct levels of classification to improve prediction accuracy. First, an overarching classifier is designed to be generalized and only recognize different object classes. Processing image data with this layer will result in a prediction as to which class the object belongs, such as stapler or book. This network is trained on all of our desired object classes, but passes its prediction to a more specialized classifier. The second layer is designed to be overfit to the data, or too accustomed to the images it has already seen. We are hoping this allows the classifier to identify individual differences and features in the objects it sees. While there is only one overarching classifier, there will be a specialized second layer network overfitted to each object class to achieve both broad and specific predictions.

As I reviewed more articles on the subject I came across *"Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis"* written by Patrice Y. Simard, Dave Steinkraus, and John C. Platt. The article specifically analyzes convolutional neural networks which are the preferred method to classify images. It presents concise advice on methodologies to adhere to for best results when implementing such a network.

According to their research, the most critical practice is to use the largest training set possible. Increasing the data set improves model accuracy, and our pipeline includes an online learning model to sequentially add new images to the data set over time. The article also notes they expanded their data set with a new form of distorted data which we have incorporated in our design as well. After our initial training set is collected we will generate an auxiliary image set with additional image noise and object variance to diversify the data set.

The second most important practice is to utilize convolutional neural networks instead of fully connected networks for image recognition. Due to the huge amount of data analysis time saved by a CNN not connecting every node in the network, "a simple do-it-yourself implementation of convolution with a flexible architecture is suitable for many visual problems". [2]

3.2.3 Overfitting in the Algorithms

Overfitting has become a recurring problem as the machine learning industry has evolved. If your neural network is over-trained on a data set it will struggle to predict features not present in the values it has already viewed. This is generally a negative result of training, but can be prevented by combining the predictions of multiple models. This isn't a perfect solution, however, as it can be difficult to implement in your algorithm without slowing down as data sets scale.

In *"Dropout: A Simple Way to Prevent Neural Networks from Overfitting"*, dropout is the recommended best practice to address overfitting. By randomly dropping units and their connection from the neural network model while training, units are prevented from co-adapting over time. The dropout method "samples from an exponential number of different thinned networks." Then, these subsets' predictions are averaged with one un-thinned using smaller weights. The article presents the method as a straightforward solution to the overfitting problem, and our group will experiment with implementing the technique to differentiate our overarching classifier and specific classifiers. [3]

REFERENCES

[1] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *International Journal of Remote Sensing*, vol. 28, no. 5, pp. 823–870, 2007. [Online]. Available: https://doi.org/10.1080/01431160600746456

[2] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." vol. 3, pp. 958–962, 2003.

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

Technology Review

Julian Weisbord

Written by Team: ASAP BOT

Oregon State University

CS 461 Fall 2017

TABLE OF CONTENTS

1 BACKPROPAGATION METHODS

In Deep Neural Networks, Backpropagation is the main technique used by a neural network to update its weights or learn how to correctly predict the output value of your input data. Without backpropagation, neural networks would cease to be useful because they wouldnt be accurate. Backpropagation gets its name from the process that it implements. Backpropagation propagates neural network accuracy updates by going backwards, starting at the output layer of the network and ending at the first layer of the network.

1.1 Backpropagation with Batch Gradient Descent Algorithm

By far the most common method for implementing backpropagation, is the batch gradient descent method. To understand gradient descent, readers need a basic understanding of differential calculus. Gradient descent iteratively applies a gradient or multidimensional-partial derivative with respect to the weights of each layer. By doing this, we are trying to approximate a global minimum value of the neural networks loss function. The loss function describes how inaccurate our neural network is at different points during neural network training. The global minimum of the loss function will be the most accurate iteration of the trained neural network. This is when we will want to stop training on the data set because our neural network will not likely achieve any better results. The reason we need to calculate gradients is because the less negative the gradient/slope(ideally zero), the closer we are to the desired global minimum.

The benefits of gradient descent are numerous, because it is the most common method used in training Convolutional Neural Networks, there are decades of: research papers, implementations, and tutorials online. There are many examples of the Gradient Descent Algorithm on the internet that our team would be able to base our solution off of. The major down sides of using Gradient Descent are that it is slow and it doesnt always produce the most accurate results. Batch gradients descent is slow because it is computationally expensive to do that much calculus on a computer and it is also compiling the gradient using the whole dataset. This algorithm is also slowly losing public favor because of how accurate the Synthetic Gradients algorithm is becoming

1.2 Backpropagation with Stochastic Gradient Descent Algorithm (SGD)

Stochastic Gradient Descent is very similar to Batch Gradient Descent. The only difference is that it performs backpropagation on small chunks of the data set. This is beneficial because it proves to take much less time and be much less taxing on your graphics processing units(GPUs) however it is known to be slightly less accurate than Batch Gradient Descent. We are trying to achieve the best classification rate that we can on certain objects so we will not be using Stochastic Gradient Descent

1.3 Backpropagation with Synthetic Gradients Algorithm

Synthetic Gradients is an algorithm that was recently developed by Deep Mind and has already shown greater prediction accuracy than Stochastic Gradient Descent and Batch Gradient Descent [1]. The only down side is that it requires very powerful GPUs and is the most computationally expensive of the 3 algorithms. One of the issues with Stochastic and Batch Gradient Descent is that the neural network can only be update after a full forward and backwards pass of itself. This can be thought of as a serial process, however the Synthetic Gradients Algorithm accomplishes back propagation with a more parallel process [2].

1.4 Conclusion

The benefit of Synthetic Gradients is that when neural networks become more complex with a lot of layers, the time it takes to update the weights in the network becomes a massive bottleneck. Synthetic Gradients is an algorithm that decouples the layers of the neural network and updates these layers all independently from one another so that each layer no longer has to wait for all of the other layers to update. Because of this, our neural networks will be using the Synthetic Gradients method.

2 Types of Online Learning

Online Learning or Sequential Learning or Lifelong Learning is the concept that intelligent agents should be able to learn more about their environment and previous knowledge throughout its lifetime . For example, a child will likely know what a laptop is but when they learn something knew about laptops such as the differences between Microsoft and Mac, they will add this information to their memory. To put it bluntly, not only Neural Networks, but AI in general is very bad at this seemingly basic process. This process is Online Learning. The following concepts, are ways that online learning can teach an intelligent agent new ways to recognize data that would be helpful in our project. Because these techniques are designed specifically for our project goals, I have named them.

2.1 Same Objects, New Orientation (SONO)

This process involves learning old data better. An example of this would be if the robot/intelligent agent learned what a mug looked like in normal usage but then the robot is presented with an upside down mug, and expected to classify it as a mug. The benefit of SONO is that it isnt too difficult to implement and there are research papers written that use similar algorithms for their research purpose. The down side is that it isnt that useful, the robot will just learn its current environment better. For example: Just because we trained the agent on a bunch of mugs doesnt mean it will suddenly know what a wine glass looks like.

2.2 Learn New Data from Old Data (LNDOD)

If our robot knows what a ripe banana looks like but not what an unripe banana looks like, the following will be our algorithms thought process: I know what bananas look like but this thing is green, I guess bananas can be both green and yellow. The benefit of LNDOD is that it will use its previous knowledge to associate characteristics of new objects from old object data set training. For example, The robot may deduce from a data set of mugs that because mugs are for drinking out of, maybe wine glasses are too. They share some of the same characteristics and shapes as mugs. The down side of this is that there isnt a lot of information available on successful examples of this process. If there were tried and true ways to do this, AIs would be smarter than humans

2.3 You Liked X, You May Like Y

This method takes knowledge of past environmental object preferences and predicts new environment objects are process that could be useful. I doubt Netflixs AI uses this but if sequential learning continues to grow, this method will be useful. An example would be, I see that you like these comedies, I bet that you would like these other ones based on the type of humor/ kind of jokes that you like (British humor, dry humor, dumb humor,etc). This would be the toughest algorithm to implement is not likely something we can accomplish with this project.

2.4 Conclusion

We will use the SONO type of Online Learning for our Capstone Project. Online learning can be incredibly difficult, SONO will allow us to implement an online learning algorithm that is successful in increasing the rate at which our intelligent agent classifies objects in its environment without nearly as much of the training and overhead that the other algorithms require.

3 Mitigating Catastrophic Interference

Catastrophic Interference or Catastrophic Forgetting is an issue that comes about from Online Learning. A backpropagation network will forget information if it learns input A and then input B. Catastrophic forgetting occurs because when many of the weights in a neural network, where knowledge is stored, are changed, it is impossible for prior knowledge about past data to be kept intact. During sequential learning, the inputs become mixed with the new input being superimposed over top of the old input. To recognize multiple sets of patterns, the network must find a place in weight space that can represent both the new and the old output.

3.1 Dual Memory Architecture (DMA)

The dual memory architecture for online learning is a semi-common algorithm in regards to implementing Sequential Online Learning. The Dual Memory Architecture (DMA) is an algorithm that creates multiple deep networks to handle online learning. Each of these networks is constructed when a specific amount of data from an unseen probability distribution is accumulated, and thus creates a deep representation of the data in a specific time [3]. Essentially, when new information comes in about past data, this new information, another neural network is created as a branch off of the previous network and new information is stored in the weights of the new neural network.

3.2 Elastic Weight Consolidation (EWC)

In brains, synaptic consolidation enables continual learning by reducing the plasticity of synapses that are vital to previously learned tasks [4]. EWC performs a similar process with neural networks. For two different neural network training periods A and B, while learning task B, EWC protects the performance in the previous task A by constraining the parameters to stay in a region of low error for task A. EWC must be highly customized to your application because it needs to remember only the past weights that are important to your system. This may require a lot of time and effort to see significant classification improvements. Additionally, in an experiment carried out by DeepMind [asdf] where an intelligent agent attempted to learn several Atari games using EWC, there wasnt a significant difference between Online Learning with EWC and regular Batch Learning. This result from EWC could recur in our project because both apply online learning to a mainly visual context.

3.3 Local Winner Take All (LWTA)

In LWTA, new information in an Artificial Neural Network(ANN) overlaps with old information. To decide which is useful, each layer of the ANN is separated into blocks of size n where n is the number of potential neurons that will compete with each other to pass on their information or activation. This decision is made through local competition of these neurons, the neuron with the quantitatively highest decided activation value is passed on to the network [5]. This can be thought of as: The neural network will remember the knowledge passed to the neuron with the highest

activation value. One problem with this algorithm is that activation functions in the neural network must be linear, which has shown to have detrimental effects on ANNs that operate based on classification.

3.4 Conclusion

The preceding algorithms are online learning algorithms that minimize the amount of catastrophic forgetting that occurs in the neural network. Our team will tentatively implement a Dual Memory Architecture for this project as it is known to provide consistently positive classification results. In a research paper published by Seoul National University [2], The Dual Memory Architectures outperformed several other common online learning techniques such as: Online Fine-Tuning, Last-Layer Fine-Tuning, Nave Incremental Bagging, and Incremental Bagging with transfer.

3.5 References

REFERENCES

[1] S. Raval, "Synthetic gradients explained," https://www.youtube.com/watch?v=qirjknNY1zo, 10 2017.

[2] Czarnecki, "Understanding synthetic gradients and decoupled neural interfaces," https://deepmind.com/research/publications/understanding-synthetic-gradients-and-decoupled-neural-interfaces/, 03 2017.

[3] K. K. Lee, Lee, "Dual-memory deep learning architectures for lifelong learning of everyday human behaviors," https://bi.snu.ac.kr/Publications/Conferences/International/IJCAI2016_SLee.pdf.

[4] R. P. Kirkpatrick, "Overcoming catastrophic forgetting in neural networks," Deep Mind.

[5] s. t. j. Srivastava, jonathan, "Compete to compute," https://papers.nips.cc/paper/5059-compete-to-compute.pdf.

5 Weekly Blog Posts

5.1 Michael's Blog Posts

5.1.1 Fall Term

Week 1

Met with professor Smart to switch from being on the Ebola project to the robotics object recognition project with Julian Weisbord and Miles McCall. Professor Smart said he would email McGrath before the Tuesday night deadline.

Week 2

We started discussing how we are planning on building on top of the already existing architecture and one of the main ideas we want to implement is online learning. We will discuss this with Bill Smart at our next meeting with him.

Week 3

This week we were able to schedule meetings every other week with one of Smarts grad student assistants named Chris. This meetings will help us get questions answered about ROS and any robot related questions that we may have. Additionally, Chris is going to help us narrow down the scope of our project so we know exactly what is expected of us.

Week 4

The meeting we held with Chris on Monday was super helpful in terms of guiding us in the right direction and making sure we are all on the same page. I now have a better understanding of the topics I'm going to need to look into in order to improve the already made object recognition classifier. Chris gave us three main points to focus on for this project which were 1) Overfitting 2) Improvements to data capturing. 3) Testing Online Learning Models (Different benefits and trade-offs)

Week 5

This week we were able to get in contact with Chris and get him to send us the code that he has been working on so far for the life long learning of the robot. We're also planning on trying to build our first basic neural network as a team to get a feel for what the code looks like and fully understand how they work. We're going to use the Youtuber named Siraj Raval as a source of guidance to help build our network.

Week 6

At our group meeting this week we discussed neural networks in depth as a group and started looking into the several different kinds that exist. We decided to look into synthetic gradients by watching a video from Siraj on YouTube. Now that we have a more solidified understanding of how neural networks work, we're going to try to implement our own next week using TensorFlow.

Week 7

This week we held our meeting with Chris and Dr. Smart and were able to catch them up on what has been going on in our project in terms of writing assignments like our technology review that's due next week. We were also able to build a simple dog/cat classifier during one of our group meetings to show to behn at our weekly TA meeting. We also

researched Convolutional Neural Networks and started looking into MNIST datasets.

Week 8

We created our technology review with the help of Chris and Dr. Smart. We also had our weekly TA meeting with behn. We mainly focused on our writing assignments this week during our group meetings. We currently dont have anything set up specifically to be able to test our network once it's developed, so as an alternative we can use Chris' machine to run our tests. Could check in with EECS for resources to get access to Nvidia servers.

Week 9

This week we submitted our final draft for the technology review. We also talked with behn about cancelling our TA meeting this week since most of our group members went home before our scheduled Wednesday meeting. Lastly, we held our group meeting remotely this week and just touched up on some main points that we should address before the end of the term.

Week 10

This week we turned in the final version of our design document. We also held our weekly TA meeting with behn over the phone and he advised us to try and get access to Pelican servers by emailing McGrath and CC'ing Dr. Smart in the email. Dr. Smart said he approved our message and Kevin laid out some options for us basically saying it can be done. Lastly, we created a rough draft of what the hierarchy of our neural network is going to look like which we will go over with Dr. Smart and Chris at our meeting on Monday.

5.1.2 Winter Term

Week 1

This week I looked into getting Linux set up on my computer so I can conduct the data capturing process from my computer. From what Chris has told us it seems like I need to install Ubuntu 14.04 since that is what's required to be able to work with the fetch robot. I have downloaded the .ISO image onto a flash drive and will try to install it this weekend when I get a chance to back up my computer just in case anything goes wrong.

Week 2

This week I tried to install Ubuntu 14.04 and after a tedious amount of work, it didn't end up working. My Mac OS still worked but my computer was acting different and I believe I need to completely reinstall my OS from scratch. My plan for now is to get my Mac OS working normally again and then trying to use a virtual machine to simulate the Linux environment like how I had it at the end of fall term when I was messing around with installing ROS packages and dependencies.

Week 3

This week I made sure that my Linux virtual machine had the necessary ROS catkin_ws set up. The next step for me now is to go into the robotics lab and make sure that my computer can connect to bandit without problems. However, we have ran into a minor problem with the fetch. Chris told us that he is having battery issues and is temporally down so we won't be able to use him until he is fixed. We were also able to get weekly meetings on Mondays for about 30 min

with Chris and Professor Smart to discuss out progress and issues every week.

Week 4

This week I was able to do some ROS tutorials on my virtual machine to familiarize myself with how ROS works and better understand how it works with Bandit. Chris told us at our weekly meeting that the fetch is still out of service but they had contacted technical support and are supposed to be fixing it sometime this week when their appointment is. At our meeting we updated Professor Smart on our progress.

Week 5

This week I accomplished one of goals which was to send Smart our new design doc so that he could approve it. I also went into the robotics lab to do small demo to make sure that everything was working fine on my computer with the fetch. I ran into a problem when running the visualization software as it was too heavy to run on the 4gb of RAM I had allocated for my virtual machine. This means I'm going to have to try to dual boot my computer again with Linux so I have access to my full 8gb of RAM that my computer has.

Week 6

This week I was successfully able to install Linux 14.04LTS on my Mac by following a more thorough guide than the first time. It was tedious but ended up working in the end! I have installed several ROS packages to mimic the labs computer set up but ended up needing to install more fetch packages as the code for data capturing wasn't working at first. Once I did get it working, I was able to map the room and conduct our first data capturing session!

Week 7

This week when I went into the HRI room to collect more data, I noticed that my set up had been moved around by a different group. The room wasn't too different but it was annoying to have to replicate my set up all over again by moving their stuff our of the way. This week I was only able to collect 3 data sets because of midterms so next week I'll have to do a lot more to reach my goal of 12.

Week 8

This week I practically spent my entire time in the HRI room collecting data. I was in there Wednesday, Thursday, and Friday for about 5 hours each day. Some of the days I went in the room was also used by another group that rearranged the room entirely which made the room unrecognizable to Bandit to some extent and made him keep getting lost. To fix this I simply just mapped the room the again so he would have an updated view on what the room looks like. This helped tremendously and made the data capturing process go much more smoothly.

Week 9

This week I plan on finishing gathering 23/25 total objects and leaving the last two mugs for Julian to do the data collection process on. In my final 8 sets I plan on adding image noise by moving the objects around in between pictures to make our classifier more robust. Other than that, this week was normal like the rest in that we had our weekly meeting with Chris and our weekly TA meeting with Behn.

Week 10

This week I made a github PR for the code needed to run the data capture. I also worked on the image cropping code to prepare our datasets for training this coming week. I made a PR for this as well and was able to get all my data sets cropped. We held our final meeting with Behn this term. We also had our meeting with Chris and decided that we dont need to do weekly meetings anymore since it's the end of the term.

5.1.3 Spring Term

Week 1:

This week I slowly got back into the grind of things. Did a little bit of work on looking over the data collection file and thought of ways to improve it. We also tried attending our weekly TA meeting with Behn but he wasn't there. He eventually responded and said our meetings start next week.

Week 2:

Mainly worked on the data collection rewrite this week with Miles. We also held our first TA meeting of the term with Behn and looks like our meetings will be much shorter and more answering questions that we have about upcoming assignments or the class in general. By the end of the week we pushed the data_collection.py rewrite to our GitHub.

Week 3:

This week I mainly worked on doing the majority of the PEP8'ing of the code we had on GitHub. We also again held our weekly meeting with Behn where we discussed release forms, the Wired article and he informed us about the change log that we can add to our requirements doc since we are planning on making some modifications to it.

Week 4:

This week we submitted our final poster that will be printed and used at the engineering expo. Additionally, I worked on making an HTML web-page from a Wired HTML code skeleton that I got from their website. I finished the Wired article and submitted my work though Canvas.

Week 5:

This week I primarily focused on working on the midterm progress report and presentation that are due on Sunday May 6th. At this point in our project I feel pretty good about the work we've accomplished and feel like we're ready for expo. In the presentation I included a live video demo of running the data capture live to show off where our project is at this point.

Week 6:

This week we had our code freeze. The primary goal of this week was to merge any upstanding PR's and do one last run through to make sure the pep8 standard was being followed throughout. In addition to that, we are planning to meet up with the other teams we will be sharing our expo space with to get a better idea of who is going to have what area of the room and such.

Week 7:

The big day is finally here, the engineering expo that we've been awaiting since the beginning of the year! This week our main goal was to get a video of bandit conducting the data capturing in a live setting so we communicated with Chris and he helped us learn how to properly and safely transport Bandit around campus so we could take him to Kelley. This served as a trial run and good practice for expo. We recorded a video of bandit doing the data capturing in the Kelley Atrium and made a video to loop through at expo.

Week 8:

At this point we have just presented our project at the engineering expo. This week I didn't do much and kind of just thought/planned the final things that are going to need to be done in this course to finish up the year.

Week 9:

Our primary goal this week was to present Smart the current state of our project. We are trying to figure the best possible time to do so sometime in the next coming week. Did minor touches to our GitHub repo.

Week 10:

On the final week of the term Miles and I went into the robotics one last time to get footage of our final code running on bandit. This footage was used in our final presentation and showed everything that we were able to accomplish this year. In addition to that, we briefly went over the final report that is due on Tuesday.

5.2 Julian's Blog Posts

5.2.1 Fall Term

Week 1:

Met with Dr. Smart (client) Emailed professor McGrath and Dr. Smart with selected project name. Week 2:

Created machine learning Wiki for our group to collaborate on different machine learning tutorials and resources that we find. Notes from meeting with Dr. Smart:

-Question: How do you want us to specify an object in a 3d image and how is it different than the way that Chris has done it? Answer: Look into Grabcut algorithm, Convex Hull in 3-space, robot will look at object from different angles, robot and object must be in same coordinate frame. How do we keep the object in frame? Could add an object to the plane of view that we know the coordinates for so therefor we will know where both of them are with respect to eachother in camera view. Right now they use AR tags(very thin hard to see with different angles)

-train a deep network to recognize the object. Can explore other times of machine learning, look into different deep network arhitectures. Specialization on one particular object may be easier than recognizing all objects in a class (cup example). Setup is that this robot will be in your environment for a long time and learn your stuff over time.

Note: Focus on deep learning, have Chris show us how to walk around with the robot

-Question: What are the desired outcomes, how will we know when we have solved your problem? Answer: Pick a set of objects and say that we want to recognize those with a high accuracy, make sure classifier is competitive. Analyze does tight cropping matter, does good localization matter.

How robust is the classification, can it recognize the object if it is moved if there is something in the way? How clean does the training data have to be (tight cropping is important)? Fitting depth images

Lifelong learning: The object will look different depending on the time. Create a dataset at one time of day, then add another training set at another time and try to combine them.

Summary: We started discussing how we are planning on building on top of the already existing architecture and one of the main ideas we want to implement is online learning. We will discuss this with Bill Smart at our next meeting with him.

Week 3:

For the first half of the week, I emailed the client to clarify some project details. I also worked to revise the project

problem statement. During the second half of the week, I spoke with a grad student named Chris about the project and sent the problem statement to both him and Dr. Smart for review. On Wednesday we met with our TA who was very informative. Today I will be preparing the final draft of our problem statement and adding it to github.

Week 4:

Meeting with Chris Eriksen. Overfitting to data in different environmental contexts (use different classifiers) Improvements to data capturing. Testing different online learning models and seeing benefits of each other Decided on objects (same n objects throughout whole project). This week we focused on narrowing down specific project goals. Our rough draft was pretty vague but we met with a doctoral student in the College of Robotics and decided to divide the project into 2 or 3 categories. They are: 1. Overfitting to data in different environmental contexts, 2. Testing different online learning models and seeing benefits of each, and 3. Improvements to data capturing.

Week 5:

Wednesday meeting with TA:

- Send emails to TA when I update meeting notes - Talking about requirements document - Need to meet with Kirsten to speak about problem statement

Goal: Write and submit requirements document rough draft

Week 6:

Requirements meeting with Dr. Smart and Chris Eriksen: Work Plan: Overfitting to data in different environmental contexts (use different classifiers) -Testing different online learning models and seeing benefits of each other. Questions: a. Who is the ideal user (maybe a research lab?) i. Yes research lab that learns its environment really well and can use that knowledge. TODO: 1. Decide on 3-5 object classes for classification 2. Continue Researching Convolutional Neural Networks 3. Look at different sequential learning techniques

Going to have a more technical meeting November 7th. This week we met with a grad student named Chris who works under Dr. Smart and updated him on what has been going on in our project. We also spent a lot of time working on our requirements document and finished it on Friday. This weekend, we established several rules for how our group meets and we all have a better understanding of each others individual goals. We are meeting Mondays, Wednesdays, and some Fridays.

Week 7:

Researching convolutional neural networks and working on building a simple program to show our TA. Looking at common datasets for ML such as the MNIST dataset. Met with Chris: - Talked about ROS - Chris showed us how the robot will navigate the environment by itself to find the goal image - Look into rviz Research different marker ideas for data capturing Requirements: -Need to make our 3 requirements more detailed, talk about what we will implement and research - Talk about what Chris mentioned for robot movement and image capturing Having some trouble understanding what a requirement should be, maybe we should speak to Kirsten a bit

Week 8:

TA Meeting: - Add picture to biography and resend link to Ben - Talked about weekly summaries - Asked Ben about Technology Review - Asked questions about Requirements Document (Fix requirements, and high level documentation) Smart Goals: - Finish Requirements Document - Continue to work on Technology Review and finish rough draft.

Summary: This week, we created our technology review document rough draft and had a meeting with Chris and Dr. Smart. At this meeting, we received help on our Technology Review.

Week 9:

No Class. We created and turned in the final version of our Technology Review

Week 10:

Smart Goals: Two assignments due by the end of the term: 1. Progress Report 2. Preliminary Design Document 3. According to our Gant Chart from the Requirements Document, we as a group should be well versed in Convolutional Neural Network Design. 4. We need to design the hierarchy of our Neural Networks

Meeting with Ben: -Technology Review -What are we doing over break: Lots of research, possibly data capturing -Due dates for Design Document, Final Progress Report(powerpoint, video, written doc) This week we had a phone meeting with Ben and talked about what we were going to do in this class over break. We also wrote and turned in our design document. This was a difficult assignment because it required a very large amount of research on my part. We also have a rough idea of our neural network hierarchy but need to go over the plan with Dr. Smart

5.2.2 Winter Term

Week 1:

Spent a lot of time researching Convolutional Neural Networks and image recognition topics. This took at least 25 hours. I implemented several practice convolutional neural nets in both Tensorflow and Keras.

Week 2:

This week I continued to learn about convolutional networks. I also looked into other resources such as the YOLOv2 Object Recognition algorithm. Also made a template for our github file and python module structure. At this time, I researched very specific details about image classification such as the challenges with an image dataset. Week 3:

Made our Week 3 and 4 Plan 1/24/18 to 2/3/18

Michael and Miles Goals: a. Data collection: Make sure you know how to operate/control the PR2, schedule some time Monday (1/29) with Chris ASAP to Learn how to interact with the PR2 using ROS multiple machines http://wiki.ros.org/ROS/Tutorials/MultipleMachines Michael and Miles together should be able to reproduce the process of logging on to the robot, mapping the room, making object point cloud, creating visualization and sending it to a file. b. Michael must be very knowledgeable about ROS and the data collection process by the end of this time frame. c. data_collect.py: This will be used for data collection shortly, completely understand Chris's data collection module and play around with it. d. Miles: get the Movidius sticks functionally working and do some research about them

Julian Goals: a. Continue working on inference, loss, and training of model b. Take Chris's data and start using it on model c. Oversee and help with Michael and Miles' tasks d. Look into final CNN Architecture e. Take image data and start making it into a Tensorflow or Keras dataset

Everybody Goals: a. Are we going to use software cropping, rfid's, circle dots, etc. b. Gather objects that we will collect data on (chair, pen, books, mugs, stapler)

Week 4:

Client Meeting: Told Chris and Dr. Smart about week 3/4/5/6 goals. Updated him and Dr. Smart about our progress. Dr. Smart added me to the HRI Room Calendar and the Fetch Calendar. He also added me to the Fetch robot calendar which eliminated a lot of our scheduling bottlenecks. Week 5:

Created Week 5/6 Team Goals: Miles: -Mentor Julian and help him learn the data collection process (Due Week 5) (Done 02/06/18). -run a CNN on the Movidius sticks (Due Week 5) LATE. -Research CNN's, determine the opportunity

cost of using Movidius vs. asking McGrath for GPU's (Ask Ben for card/SSH into Chris' desktop) (Partially Complete 02/18/18). -Remove unusable data from image datasets that were scraped by Julian (Done 02/19/18). -Week 3/4 Overflow: Install Movidius drivers (Scrapped Movidius). -Week 3/4 Overflow: Save rviz visualiztion to a file. (Done 02/06/18).

Michael: -Send Dr. Smart our design doc to be approved and follow up (Done). -Collect data on Fetch (Have at least 3 good sets of images by end of week 6 (Done 02/18/18). -Week 3/4 Overflow: Recreate and understand the data localization (Done). Week 3/4 Overflow: Michael Understand data collection python module, play around with, and run on the the Fetch (Done). Julian: -Make CNN Model code pretty (Done 02/13/18) -Scrape data for our objects from internet and send to Miles (Done 02/10/18). -Finish prepare_data module (Due Week 5) (Done 02/11/18). -Experiment with different parameters of CNN, it should be training on scraped and practice data, create PR for everyone to review (Partially Complete 02/18/18). -Continue researching Sequential Learning Architectures (Done 02/18/18). -Research Synthetic Gradients Backpropagation LATE. -Research Resnet Inception and Tensorflow alternatives (Done 02/18/18). -Try to install Movidius drivers if Miles doesn't have it working by Sunday (Reassess whether Movidius is worth it) (Done 02/13/18).

Everybody: -Gather any missing objects (Close Enough 02/14/18). -Review and approve/deny any and all PR's (Done). -Make Poster (Due 02/14/18) (Done 02/14/18). -Make video (Due 02/16/18) (Done 02/16/18). -Do report (Due 02/16/18) (Done 02/16/18).

Week 6:

Worked on Residual Neural Network implementation and other things that carried over from week 5. Week 7:

Week 7/8 Goals: Miles: -Learning about CNN Architectures LATE -Add caching to prepare_data.py, create PR and have it approved by end of Week 8 LATE. -Create Novel Image python file (This file takes a new set of images and prepares them), create PR and have it approved by end of Week 8 LATE. -Collaborate with Julian on Classify.py (Inference, runs new data on our model) LATE. -Make the changes that Dr. Smart wants to our design doc (Done 03/01/18). -Ask McGrath for GPU's and/or ask Ben for card/SSH into Chris' desktop (Done 03/01/18).

Michael: -Collect more data (8 datasets) on 5 different objects, add image noise, introduce varying object placement (Partially Complete). -Look into ways to improve data collection (Done) -Figure out how to do image cropping, is there a better way? How can we ensure that the whole object is within the cropped image? LATE. -Minor modifications to data_collection.py for modularity (Done). -Create a Github PR that includes all files related to data collection and have it approved LATE.

Julian: -Create Github PR for CNN Model (Done 02/26/18). -Continue researching Sequential Learning Architectures and determine the best algorithm or make up your own (Done 03/03/18). -Implement ResNet Inception and compare to the CNN model that I already have (Partially Complete). -Continue researching Synthetic Gradients Backpropagation to apply to our pipeline LATE. -Decide to scrap or use Movidius (Done, will scrap). -Collect data on fetch (Michael assists) (2 datasets) LATE. -Create Classify.py (File runs new data on our model) (Done 02/26/18). -Talk to Dr. Smart about having the robot at the Expo LATE.

Week 8:

No Client Meeting this week. Continue working on tasks from week 7. Week 9:

Created week 9/10 Goals:

Julian:

-Talk to Dr. Smart about having the robot at the Expo (Done 3/13/18). -Compare ResNet Inception to the CNN model

that I already have (Done 3/13/18). -Have PR approved with ResNet Inception (Done 3/18/18) -Implement Sequential Learning Algorithm into our pipeline and make a PR. (Getting There). -Add Tensorboard to our models (Done 3/18/18).

Miles: -Add caching to prepare_data.py, create PR and have it approved by end of Week 8 LATE. -Create Novel Image python file (This file takes a new set of images and prepares them), create PR and have it approved by end of Week 8 LATE. -Collect data on fetch (Michael assists) (2 datasets) (Did 4). -Start implementing Synthetic Gradients LATE. -Ask Chris for GPU and set up ssh (Done). -Design Document.

Michael: -Figure out how to do image cropping, is there a better way? How can we ensure that the whole object is within the cropped image? (Done 3/18/18). -Create a Github PR that includes all files related to data collection and have it approved (Done). -Add image noise to data, introduce varying object placement (Done). -Collect 8 sets of image data (Done).

Week 10:

On 03/07/18, I attended a tech talk at Oregon State University given by Eric Eaton, a faculty member at Penn State. Eric specializes in Sequential Lifelong learning which is also one of the key aspects of our project. https://www.seas.upenn.edu/ eeato

Worked on Elastic Weight Consolidation Implementation, there are more details on my One Note under Week 10.2

5.2.3 Spring Term

Week 1: Created Week 1 Goals-

Julian: -Continue Implementing EWC Learning Algorithm into our pipeline (Done) -Document issues in repository and make progress towards fixing them -Switch object classes to correct ones (Done) -PEP8 Files that you have added to our repository and make a PR (Done)

Miles: -Add caching to prepare_data.py, create PR and have it approved by end of Week 8 LATE -Create Novel Image python file (This file takes a new set of images and prepares them), create PR and have it approved by end of Week 8 LATE -Start implementing Synthetic Gradients LATE -Have Design Document Approved and send to McGrath et al LATE -PEP8 Files that you have added to our repository and make a PR LATE

Michael: -PEP8 Files that you have added to our repository, merge existing PRs, and make a PR LATE -Major rewrite of data_collection.py, make sure it runs correctly w/ the Fetch (Miles to assist) LATE

Created Week 2 Goals:

Julian: -Make a PR for EWC Learning Algorithm LATE -Repository should be pretty clean, modular, etc. -Merge any PEP8 PRs (Done) -Try to remove layers from ResNet Inception LATE -Continue work on system that compares the results of Sequential Learning vs no sequential learning. (Done)

Miles: -Determine if Synthetic Gradients is a good idea and make a PR if necessary LATE -Have Design Document Approved -PEP8 Files that you have added to our repository and make a PR LATE -Merge any PEP8 PRs LATE

Michael: -Finish rewrite of data_collection.py and have it merged into master LATE -Merge any PEP8 PRs LATE

Week 2:

Week 3/4 Goals: Julian: -Make a PR for EWC Learning Algorithm -Try to remove layers from ResNet Inception -Continue work on system that compares the results of Sequential Learning vs no sequential learning.

Worked on Elastic Weight Consolidation implementation into pipeline. This has been a difficult process, I have looked at other implementations on Github but they seem to be computing the Fischer Diagonal differently than the paper on Elastic Weight Consolidation

Week 3:

TA Meeting: -Get down logistics for Expo -We need to write a research paper for Dr. Smart -behn said that we can pretty much work on our project code up until Expo.

More work on Elastic Weight Consolidation, I may just have to implement the novel_image.py and classify.py by myself because they are necessary for testing sequential learning functionality.

Week 4:

Created better visualizations of the training process using Tensorboard

Week 5:

What I have done this term so far: Work on adding sequential learning Added and cropped new set of images to our training and testing sets Made a better visualization of training process using Tensorboard Adding novel image preparation to our existing prepare_data.py Working on Classify.py, which is the inference file. Team management Updated both neural networks with small changes, tuned some parameters Starting documenting how parameters effect the training process

Week 6

Meet w/ other groups 4pm on Tuesday in room 1005: -Need outlet

KEC Data Visualization -Check out the robot -table -An object -Put robot in case and walk it or drive it in a car -scope out KEC tomorrow at 5pm to see how busy it is -Email Chris to show us how to move the robot and ask him to help on Tuesday or Wed at 4:30ish

-text Nick, Tyler, Natalie, and others to come see our robot -Email the robotics department about coming to see bandit

Week 7

Preparing for expo, Brought Fetch robot to KEC to film a video that will be played on the monitor during expo. Talked to Dr. Smart and Chris Eriksen about how to transport the Fetch robot. Expo went well.

Week 8

Continued work on Elastic Weight Consolidation part of project.

Week 9

Presented current state of project to Dr. Smart

Our team worked on the Spring Progress Report and presentation

Week 10

Work on finalizing project, research paper, and final report

5.3 Miles' Blog Posts

5.3.1 Fall Term

Week 1:

In these first few weeks we are getting accustomed to the class lay out and investigating all the projects we would be interested in. My goal is to decide on a topic or issue I want to pursue for the year and look for projects that match that.

Week 2:

After looking into the different available projects last week we started voting on which options we would like to participate in. I am hoping to get a project involving machine learning in some fashion, and there are a few projects like Dr. Smarts mobile robot that stick out that I chose. I submitted my top choices all with machine learning and will be settled by next week.

Week 3:

This week our teams were finalized and we had our first meeting to get connected and in contact with each other. We discussed the project as a whole and how we might each wish to contribute, and went through some initial set up steps like making a GitHub repository. We've set up meetings with Dr. Smart and his grad student Chris Eriksen to keep in touch on a regular basis.

Week 4:

This week we started narrowing down the scope of our project and defining the issues we'd like to address in the project. After meeting with Chris we discussed overfitting, data capture, and online learning, and how Dr. Smart wishes to have these aspects worked into the project.

Week 5:

Our team meetings have been helpful establishing how we will distribute the responsibilities and tasks for the project. As a group we've been researching machine learning and neural networks, and have found intuitive video tutorials and scholarly articles. Chris also introduced us to the data gathering files he already has in place for the Fetch bot and shared all relevant code he has with us.

Week 6:

This week I worked on adding sources to our annotated bibliography regarding neural networks and image classification. Our group has been unclear on how to make our project research oriented and the special lecture on research projects was very helpful for my understanding of how our project differs from those with more of a deliverable product. Kirsten laid out what we should be looking for in scholarly articles and how to connect them to our project.

Week 7:

This week our meeting with Dr. Smart and Chris was helpful for me, sharing our current research status with them and getting useful feedback and guidance. The tutorials and articles I've been looking into are good sources for the Technology Review coming up and with these new insights I should be able to complete my part easily. My tech review is focusing on the main components of the classification model, so I'm writing on neural networks and how we might use different technologies to our advantage.

Week 8:

I finalized the tech review and we got the ideas approved by Dr. Smart in our meeting this week. Most of the week was spent individually and as a group putting these documents together, but we also continued to try to establish a good processing platform to test our neural network on. My desktop has an Nvidia 560ti for a GPU and Julian has a newer card, but we may end up SSH'ing into Chris' workstation in the lab to utilize an even stronger card. So far the remote access to shared resources hasn't been successful, but we will continue to investigate while we use local machines.

Week 9:

This week covered a lot of organization and preparation for our group. We all submitted our tech reviews and made plans for a work schedule over the break. While we do not need to work too much over the break, we want to be set up to get a fast start winter term. Behn gave us good insight on how to finish the term strong and prepare for the break like we were wanting.

Week 10:

This week finalizes our term and we submitted two key documents, the design document and final progress report.

We started the week with a decent draft of the design document which lays out our plans to complete our project, and the progress report summarizes the second half of fall term. I had issues with video capture and screen recording, but luckily fixed them before we had to submit the report. We also reached out to McGrath and Dr. Smart to inquire about GPU server access, as we want to have a processing machine in place over the break.

5.3.2 Winter Term

Week 1:

This week I looked into the data gathering code and begin the environment set up process on my laptop. I am collaborating with Michael to complete data gathering, and while it is mainly his responsibility I want to be able to replicate his results on my machine. My laptop is old and underpowered, however, so certain tools designed for ROS like Rviz don't run very well or at all. Luckily my laptop was already running Ubuntu, so I simply reset the version of the OS to 14.04 (required for the Fetch's set up of ROS) and installed all requirements in a virtual environment.

Week 2:

While waiting for Michael to have time to finish his set up and compare to my own, I spent this week looking into all GPU resources available to us and how we may be able to run our project with them. There are many ways for us to process our data, each with advantages and disadvantages, and we are trying to decide as a group the best solutions. Ideally we will use a local machine with a strong enough GPU to avoid scheduling time on a shared server resource.

Week 3:

This week we got more set up in the robotics lab and experiment room we will be using for the rest of the data gathering steps. We had a meeting with Chris where he demonstrated the full data gathering process and all programs that must be ran to do so. Our goal is to reproduce this process without help or supervision so we can quickly collect images. Unfortunately the Fetch robot needs a new battery and is currently out of service while we are trying to get all this set up. I also picked up Intel Movidius Computing sticks from McGrath, which may be an interesting resource for running the model on.

Week 4:

I faced some speed bumps and delays this week, but did my best to push through these issues the best I can. The fetch bot is still out of commission while the battery is replaced and the Movidius sticks McGrath gave us have been difficult to get running properly on my machine. I was hoping the installation of their software would be straightforward, but I may need more time than I was anticipating to get them stable on my laptop. I am also helping Michael with his ROS tutorials and Linux troubleshooting.

Week 5:

This week Michael and I were able to work more with ROS and the Fetch bot. Our goals were to save the floor mapping generated by the radar scanner to a file and finish putting together our collection of objects we will use to classify. I spent time catching Julian up to speed on the data gathering process, but we will need more time to master ROS and the visualization tools.

Week 6:

This week I was able to focus on evaluating the value the Movidius sticks have bring to our project. Because of the difficulty I've had implementing them successfully on my machine we want to research the cost-benefit analysis of the technology before pursuing them further. We want the full pipeline to be easily replicated on many environments, and specific hardware like the sticks need to be worth including in the project for this reason. I spent time cleaning

Julian's scraped data sets as well for training the classifier. Images were automatically pulled from the internet and I went through the results to remove any images that didn't align with the object classes we wanted in the training data set.

Week 7:

This week I revised the design document to match our updated understanding of the different technologies in our project and any design decisions we've changed since the document was created. I collaborated with Julian to get caught up on his progress with the convolutional neural network implementation, and I need to spend more time learning about the software he is currently using. In our weekly meeting with Behn we discussed speeding up the data gathering process and ways we might improve the image quality of the Fetch bot.

Week 8:

This week I helped Michael finish our initial data capturing efforts. This was a significant goal of ours, as we now have a "rough draft" of all of our results and can train the classifier on a data set we are happy with. I began work on adding specialized programs to Julian's classification model, a script to handle new input data and a way to cache our data set state for future reference. I struggled with Tensorflow compatibility on my laptop, but these issues shouldnt be present on the machines we are actually using to test the classification model on.

Week 9:

I spent the week back in the robotics lab, revising the data capture code and setting up the environment again on my laptop. I wasn't paying attention and updated the OS past 14.04 and had to revert my installation to communicate properly with the Fetch again. After completing the initial data set with Michael, I am working on our auxiliary data set that will be used more so for image prediction instead of model training. I also met with McGrath and picked up an Nvidia 980 which we can use in any of our machines for the duration of the class.

Week 10:

This week I completed my auxiliary data set collection, which was another goal I was really hoping to have done by the end of winter term. I cleared out the experiment room of all our set up material, and unless we decide we need more data in spring term we don't need to revisit data collection at this point. As a group we met with Chris and set up a calendar to share access to his lab workstation as an extra GPU resource. We also completed our final progress report for the term and met to discuss our break plans to start spring term on the right track.

5.3.3 Spring Term

Week 1:

We met with Behn this week to get back in touch and plan our path to the expo. We double checked our registration status for the expo with McGrath and looked into getting our poster printed. I spent the week reviewing our design document a final time for discrepancies between our understanding of the project with Dr. Smart and what our document was representing and created a change log for the document outlining the changes we made.

Week 2:

Our group still has some confusion about upcoming requirements, but our meeting with Behn clarified most of these issues. Our expo poster, release forms, and WIRED articles are due soon, and we discussed what our user manual and research deliverable are going to look like. I spent the week finalizing our design document and requirements document, and submitted pull requests for both.

Week 3:

This week I laid out the full skeleton of my WIRED article and met with Omeed and his group to trade presentations. We presented our project to their group and answered any questions they had, then we asked them about their "I Heart Corvallis" app. I struggled to decide how I wanted to format my article, but I am happy with how my layout compares to actual WIRED article reviews.

Week 4:

This week our meeting with Behn helped explain more questions we've had about finishing our project. We won't have a meeting next week, but got good information on how to make our code commenting more readable and translate our notes into a user manual. We submitted our WIRED articles, and our midterm progress report is due next week which we ran through in a group meeting.

Week 5:

This is the week before the expo, and we've been ensuring our presentation is ready. We've been running the model to gather additional results, and have added these new values to the poster. Michael and I met with the other groups that will be staged in the same room as us for the expo, and decided on how we all wanted to organize the room.

Week 6:

This week was Expo week! The rest of the week was spent preparing for Friday. We filmed a video to play during our presentation, and our group met to go over key discussion points that might arise during the expo. The event itself went over smoothly and we answered all questions brought up by a variety of audience members.

Week 7:

This week was our code freeze, meaning we had to have all functions in our pipeline implemented completely with bugs resolved. Our group met to address the important outstanding branches open in our repository. After the freeze we'll still be able to update our documents and documentation, but all functional programs must be left alone. I spent most of my attention finishing the additional online learning functions, and made sure the code was compatible with Julian's final versions.

Week 8:

This week I focused on creating our research paper. I looked into different IEEE formats for scientific journals, and used the research lecture from earlier in the year as a reference. I created a skeleton for the document, but need more time to finish writing the body of the essay.

Week 9:

We are moving into the final stages of our project, and capstone as a whole, and this week and the next are dedicated wrapping up any loose ends for our project. I am working on our research paper to include in our final report, and we filmed and submitted our final presentation. Our group is also working on compiling the final version of our user manual from our combined notes. I stopped by McGrath's office to return all remaining equipment we borrowed this term.

Week 10:

This week our group has been merging our remaining Github branches to finish our repository. We compiled our final report early this week, and handed our project over to Dr. Smart for a final evaluation.

COLLEGE OF ENGINEERING

Electrical Engineering and Computer Science

CS 67

ONLINE DEEP LEARNING FOR OBJECT RECOGNITION ON A MOBILE ROBOT

Watch as our robot, Bandit, recognizes objects in the room completely on his own! Train him, show him a new object, and Bandit will correctly identify it!

INTRODUCTION

Our project investigates the struggle to train machines to retain old task knowledge while learning new data. This issue is known as catastrophic forgetting and by using Sequential Deep Neural Networks we are able to eliminate a majority of the problem, allowing our robot to recognize objects with high certainty.

We are using Bandit, a Fetch mobile robot, to gather images of objects to classify. Our goal is to train Bandit to predict the class of new objects with an accuracy of 80% or higher.

BACKGROUND

We hypothesized that by adjusting the network model to value knowledge compatible with old and new data, our network's efficiency will increase dramatically. This allows our intelligent robot to learn to recognize new objects much quicker than with common techniques like batch learning.

This technology is based on Deep Mind's Elastic Weight Consolidation (EWC), a state of the art online learning algorithm.

Our intelligent agent is dynamically learning about its environment, making our model applicable in a range of environment where humans need assistance. This system will shine in settings such as: labs, factories, offices, kitchens, and more!

Oregon State University

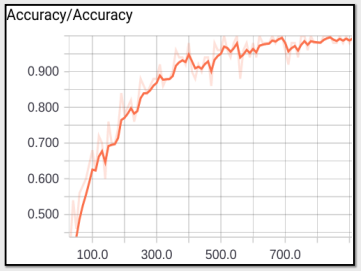
PROGRAM DESCRIPTION

Our team implemented a software pipeline to manage interacting programs in our project. After Bandit gathers images of objects in a room, a collection of scripts perform data preparation, sequential convolutional neural network (CNN) training, and object class prediction. Online/Sequential Learning makes it easy to repeat this process over and over again, ensuring that Bandit learns the most from his environment

Our classification system is initialized with general object classes to keep predictions relatively broad. With online learning, as time goes on, the network overfits itself to the objects it is exposed to and learns their individual features, such as logos, shapes, colors, and orientations.

With this in mind, our data set contains a variety of commonplace objects with different key features and dimensions. Individual objects have distinguishing traits Bandit can recognize, and consist of staplers, mugs, chairs, screwdrivers, and books.

RESULTS



97% Accuracy

The average classification success rate of our model trained on the 5 object classes using Inception ResNet for transfer learning, 21% higher than our starting goal!

2 hours

The average training time of our model with 500 images on an Nvidia GeForce GTX 970 GPU.

CONCLUSION

As we predicted, our model is able to correctly classify objects a vast majority of the time. It is capable of identifying an object's overarching class and is able to differentiate between old and new objects.

Through online learning, the model is also able to learn new traits about previously seen objects over time. This technology was the fundamental concept that allowed our model to efficiently and dynamically add image data to the network's current knowledgebase.

We demonstrated a template use case in which placeholder objects were introduced and learned over time, and are now interested in expanding these concepts to larger, more fast-paced environments.

Team From Left to Right

Miles McCall mccallm@oregonstate.edu
Julian Weisbord weisborj@oregonstate.edu
Michael Rodriguez rodrigmi@oregonstate.edu

Client

Dr. Bill Smart smartw@oregonstate.edu

Organization

Oregon State University Personal Robotics Lab

Acknowledgements

Special thanks to Chris Eriksen for helping to guide the project and assist throughout.

Group 67 - Research Paper

Deep Learning for Object Recognition on a Mobile Robot

Miles McCall, Michael Rodriguez, Julian Weisbord
Oregon State University
CS 463 Spring 2018

***Abstract*—The research paper is the write up of our research findings. It includes the methods and technologies we tested in our project and the results from those experiments. This document acts as the conclusion to the research aspect of our project.**

***Index Terms*—Deep Learning, Online Learning, Object Recognition, Mobile Robot**

Contents

I. Introduction

Our project is research oriented, meaning we dedicated a portion of our design process to experimenting and investigating different technologies and techniques. Our software pipeline has several major programs that interact with each other to collect and process data, and for each of these steps we looked into alternative solutions to the problem. Our team analyzed details such as cost benefit analyses and the potential to scale technologies as the classification model grows. We used these comparisons to decide which aspects to incorporate into our project.

II. Background

The Personal Robotics Lab has already been working with the Fetch robot and has code in place to perform some of the basic functions in our pipeline, such as data gathering. Our aim was to take the ideas and methods Dr. Smart and the other members of the lab have already investigated and expand upon them, bringing certain features they had discussed to life in our pipeline. The majority of fall term itself was spent designing our pipeline and researching related works to inform our decisions.

III. Literature review

We looked into literature and scholarly articles relevant to each function in our pipeline to get a range of approaches and opinions about the technologies. Our group doesn't have the time or resources to implement and test every method currently used in the industry, so we relied heavily on the insights in the research we read. Each method was judged on how well it would fit into our pipeline architecture and scale based on the resources available.

A. Project Overview

- Data Collection
  - ROS for robot actuation
  - Image Tagging - Object Selection with Interface
  - Data Gathering
- Image Recognition Part 1
  - Overarching Classifiers
  - Sub Classifiers
  - Overfitting in the algorithms
- Image Recognition Part 2
  - Backpropagation Methods
  - Types of Online Learning
  - Minimizing Catastrophic Interference

B. Data Collection

Operating System for robot actuation (Overview): An operating system for a robot basically serves as the brain of the robot. The operating system provides the tools necessary to make the robot do an unimaginable number of different things. One of those things that it can help us out with is object recognition and classification. In this section, I will be discussing the potential operating systems that our PR2 robot could be running.

ROS: ROS (Robot Operating System) is the most commonly known and used operating system for robots. The main idea behind the creation of ROS was mainly to offer standardized functionalities that perform hardware abstraction and helping those in robotics development from having to continue rebuilding the wheel on their own. ROS is maintained by a company called Willow Garage which develops software and hardware for their robots which conveniently happens to include the PR2. Everything that is produced by Willow Garage is open source and has BSD licensing. Being open source is one of the big reasons that

ROS has evolved so quickly and become so prominent in the field of robotics research and development. ROS provides resources that are organized into a hierarchical structure on disk. Two key concepts exist that are stand out more than the others and that is the package and the stack. The package is a directory that has external libraries, APIs data, configuration files, nodes and an xml configuration file. The stack is a collection of packages that offer functionalities like navigation, positioning, mapping, and others [1].

Microsoft Robotics Developer Studio: The Microsoft Robotics Developer Studio is a Windows-based environment used to help create robotic applications. Some of the perks that RDS boasts is a lightweight REST-style, service-oriented runtime, a set of visual authoring and simulation tools, tutorials and sample code to help developers get started. RDS makes programming simple by allowing asynchronous input from the multiple sensors on the robot and output to actuators and motors. This would help our robot improve the speed on how fast it trains itself on an object by expediting the process. RDS also has a DSS Service Oriented Architecture that helps make it simple to interact with the robot through a web browser or windows-based application [2]. By being able to interact and respond to our robot through such an interface, it would benefit the feedback compartment of our project where the robot periodically checks in with us to confirm that it is targeting the right object.

NAOqi: NAOqi is the software that runs and controls the robot. The NAOqi framework which is used to program NAO solves basic robotic needs such as parallelism, resources, events, and synchronization. This framework is versatile in that it allows development in Windows, Linux, and Mac OS. NAOqi is similar to ROS in that its API functions for both C++ and Python and its modules are transparent between both languages as well [3]. Although this operating system seems to have the same major components that ROS has, it doesnt have nearly as much open source content and documentation. Additionally, NAOqi doesnt have a visual interface that we can interact with to verify that our robot is focusing on the correct object.

Findings: The operating system we decided on to use on the robot is ROS. This choice was a no brainer in that the PR2 robot we will be working with was actually developed by Willow Creek, the same guys that started and maintain ROS. In addition to being related to the same company, Oregon State University is the primary hosting site for ROS and every roboticist in the lab uses it. This will make it easier for us to reach out to faculty in the robotics department for help since we will be speaking the same robot language as them. Our client Professor Smart is also fond of ROS, so we ultimately did not have a say in what operating system we wanted to use.

Image Tagging - Object Selection with Interface: Image tagging involves using a marker system that involves a set of patterns that our robot can detect and familiarize itself with. The purpose of having a marker system is that it helps the robot orient itself in the environment that it presides in. For our testing purposes, we want the robot to pin the marker on its coordinate map of the room generated from its sensors. This specific spot on the map will serve as the testing center where we will be placing objects for the robot to train on.

Vision Markers: Vision Markers in computer vision are typically either squared or circular. The images contained within these two shapes just has to be something that the robots camera is able to recognize. The size of the marker is not a super important, what is important is that you are able to maximize the distance from which the robot can successfully detect the marker. The shapes have individual advantages that make it better or worse depending on the goal at hand. Squares can be accessed uniformly using homography and unique vertices to orient the camera in a unique way. This allows for square tags to carry a larger symbolic data payload when used. Circular tags on the other hand can be accessed from any camera angle by computing from the whole contour surrounding the marker. This makes circular tags have better location and pose accuracy [4]. For our project, we want to be able to do a full 360 rotation around the object from varying angles so having circular tags and their ability to be recognized from all angles is a huge upside.

RFID: Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags that are near or on objects. RFID work great with robots because they are a low-priced sensor that eliminate the problem of robots having difficulty visually identifying tags. It doesnt matter if it is light or dark out, whether the object is right side up, sideways, or upside down, or even if its in a different room. The robot can be equipped with long range UHF RFID antennas that allow it to roam a room and see where the strongest signal from the RFID tags is coming from [5]. The robot is essentially playing the game hot or cold with the tag until it is able to locate it. The method of using RFID tags would be the most accurate and have the best long-range detection capability if implemented correctly.

Chemical Markings: Chemical markings can be used as a technique for storing temporary information in an environment. These markings eventually fade away once their purpose has been fulfilled for a certain amount of time. Similar to how dogs and cats mark their territories with pheromone markings, robots can use the same technique to send/receive a message. These short-term markings can be referred to as short-lived navigational markers (SLNMs). These SLNMs can be detected with sensors that our robot possesses and interpret its meaning by us telling it what exactly it should be associating with that specific chemical [6]. This technology isnt super practical for our testing purposes and is a little outdated. In most cases we want our marker to be permanent for the testing environment so that it can always use it as a frame of reference.

Findings: We found that vision markers using PointCloud2 within RVIZ seems to be the most straight forward way

for us to plot our points in 3D space so that bandit can know where it is at precisely. Additionally, this helps with image capturing and image cropping because I can save the metadata from these points and use them later on as a point of reference. For example, I can calculate the center of the 4 points in 3D space and that should ideally be right on target with the object, which allows bandit to line himself up with this center point and take pictures.

Data Gathering: Data gathering is one of the most pivotal parts of this project. Without data, the robot has nothing to run through its neural network to classify objects. Gathering a training set of images large enough for the robot to train on is typically expensive on computational resources and time. In this section, I am going to explore different data gathering techniques to see which one suits our needs best.

Traditional Approach: The traditional data gathering approach involves first taking pictures of the object in its natural environment. Next, the robot either searches the environment for the object or it uses object instances gathered prior to the test to place the object in the environment in a natural way. The next step involves hand-tagging the pictures marking bounding boxes or tight outlines in each picture [7]. This last step repeats as the robot goes 360 degrees around the object gathering raw picture data for the object. This method is not the most accurate because the quality of the dataset relies heavily on the human input required to get the pictures. A similar data gathering approach is currently being used in the OSU robotics lab, which means there is plenty of room for improvement.

Synthetic Approach: A synthetic approach implemented by a group of students at Stanford University involves taking pictures of the object in a green screen setting. The next step is to build a new larger training dataset by perturbing foreground, background, and shadow of the images taken using a probabilistic model. This allows for the true distribution of an object class to be modeled just as good as if it were to have been real data [7]. This approach saves a large amount of man hours that would have otherwise been needed for hand-labeling the pictures for the dataset. Our current lab resources do not offer the green screen technology needed to implement such a solution but it is something that we can bring to the attention of our client. The robot currently takes roughly about over an hour to gather its dataset to train on for one object so decreasing the amount of time needed to create a dataset would be fantastic.

Flashlight Approach: A data gathering approach that I believe will help us gather better quality pictures in our training set is called the flashlight approach. One of the biggest obstacles that we face when trying to classify an object is the amount of light exposure that it receives and having that affect the quality of the images we gather. To make the testing more consistent across all objects, I suggest that the robot hold a flashlight up to the object that it is capturing images on. This will allow the robot to get a much clearer image of the object and keep it consistent across all tests. This approach might produce a bit of glare depending on the material of the object but since the robot currently relies on human interaction with the hand-labeling, we can selectively choose to not include sections with glare in it. This last approach is a bit of a stretch but is something worth trying on at least one test to determine if it is a significant enough increase on the quality of the training set.

Findings: The traditional approach works best with the method of image tagging that we decided to pursue. Having to plot four points in 3D space allows for us to run the data gather with ease as we can just keep generating random goals around these four points for Bandit to make his way to and take different pictures from.

C. Image Recognition Part 1

Overarching Classifiers: The most critical program in our pipeline is the classification model, as it performs the most complicated task and actually predicts the object's class. Data prediction is an increasingly large field and collaboration is common practice in the industry to help grow the public knowledge base. Scientific literature on the subject of classification is readily available, and I found an article detailing a broad overview of modern techniques. Image classification is particularly complicated and can be accomplished numerous ways. In *"A survey of image classification methods and techniques for improving classification performance"*, authors D. Lu and Q. Weng "examine current practices, problems, and prospects of image classification". The review notes that choosing an appropriate algorithm and designing a suitable processing procedure is essential to a successful image classifier, making this one of the most critical decisions for the entire pipeline. I am looking for modern approaches to our problem that take full advantage of the processing resources we have available to us, while still having the backing of the community and proper support resources available to developers. The article emphasizes nonparametric classifiers that allow for a dynamic number of model parameters, specifically neural networks, decision tree classifiers, and knowledgebased classification. Based on the analysis in the review, our group has agreed upon a convolutional neural network for image classification. In recent years the algorithm has been widely adopted for the task due to advantages such as "arbitrary decision boundary capability, easy adaptation to different types of data and input structures, ... , and generalization for use with multiple images". Flexibility and generalization will make training the neural network possible with the number of image classes we want to include in the data set, and may come into play again when we investigate data overfitting. [8]

Sub Classifiers: While my research into literature regarding classifiers compared different methods and algorithms in a concise manner, our project is designed to be more specific than just using pre-existing libraries predict image classes. We

are aiming to implement a classification model designed by our group, incorporating two distinct levels of classification to improve prediction accuracy. First, an overarching classifier is designed to be generalized and only recognize different object classes. Processing image data with this layer will result in a prediction as to which class the object belongs, such as stapler or book. This network is trained on all of our desired object classes, but passes its prediction to a more specialized classifier. The second layer is designed to be overfit to the data, or too accustomed to the images it has already seen. We are hoping this allows the classifier to identify individual differences and features in the objects it sees. While there is only one overarching classifier, there will be a specialized second layer network overfitted to each object class to achieve both broad and specific predictions. As I reviewed more articles on the subject I came across *"Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis"* written by Patrice Y. Simard, Dave Steinkraus, and John C. Platt. The article specifically analyzes convolutional neural networks which are the preferred method to classify images. It presents concise advice on methodologies to adhere to for best results when implementing such a network. According to their research, the most critical practice is to use the largest training set possible. Increasing the data set improves model accuracy, and our pipeline includes an online learning model to sequentially add new images to the data set over time. The article also notes they expanded their data set with a new form of distorted data which we have incorporated in our design as well. After our initial training set is collected we will generate an auxiliary image set with additional image noise and object variance to diversify the data set. The second most important practice is to utilize convolutional neural networks instead of fully connected networks for image recognition. Due to the huge amount of data analysis time saved by a CNN not connecting every node in the network, "a simple do-it-yourself implementation of convolution with a flexible architecture is suitable for many visual problems". [9]

Overfitting in the Algorithms: Overfitting has become a recurring problem as the machine learning industry has evolved. If your neural network is over-trained on a data set it will struggle to predict features not present in the values it has already viewed. This is generally a negative result of training, but can be prevented by combining the predictions of multiple models. This isn't a perfect solution, however, as it can be difficult to implement in your algorithm without slowing down as data sets scale. In *"Dropout: A Simple Way to Prevent Neural Networks from Overfitting"*, dropout is the recommended best practice to address overfitting. By randomly dropping units and their connection from the neural network model while training, units are prevented from co-adapting over time. The dropout method "samples from an exponential number of different thinned networks." Then, these subsets' predictions are averaged with one un-thinned using smaller weights. The article presents the method as a straightforward solution to the overfitting problem, and our group will experiment with implementing the technique to differentiate our overarching classifier and specific classifiers. [10]

D. Image Recognition Part 2

Backpropagation Methods: In Deep Neural Networks, Backpropagation is the main technique used by a neural network to update its weights or learn how to correctly predict the output value of your input data. Without backpropagation, neural networks would cease to be useful because they wouldn't be accurate. Backpropagation gets its name from the process that it implements. Backpropagation propagates neural network accuracy updates by going backwards, starting at the output layer of the network and ending at the first layer of the network.

Backpropagation with Batch Gradient Descent Algorithm: By far the most common method for implementing backpropagation, is the batch gradient descent method. To under- stand gradient descent, readers need a basic understanding of differential calculus. Gradient descent iteratively applies a gradient or multidimensional-partial derivative with respect to the weights of each layer. By doing this, we are trying to approximate a global minimum value of the neural networks loss function. The loss function describes how inaccurate our neural network is at different points during neural network training. The global minimum of the loss function will be the most accurate iteration of the trained neural network. This is when we will want to stop training on the data set because our neural network will not likely achieve any better results. The reason we need to calculate gradients is because the less negative the gradient/slope(ideally zero), the closer we are to the desired global minimum. The benefits of gradient descent are numerous, because it is the most common method used in training Convolutional Neural Networks, there are decades of: research papers, implementations, and tutorials online. There are many examples of the Gradient Descent Algorithm on the internet that our team would be able to base our solution off of. The major down sides of using Gradient Descent are that it is slow and it doesnt always produce the most accurate results. Batch gradients descent is slow because it is computationally expensive to do that much calculus on a computer and it is also compiling the gradient using the whole dataset. This algorithm is also slowly losing public favor because of how accurate the Synthetic Gradients algorithm is becoming.

Backpropagation with Stochastic Gradient Descent Algorithm (SGD): Stochastic Gradient Descent is very similar to Batch Gradient Descent. The only difference is that it performs backprop- agation on small chunks of the data set. This is beneficial because it proves to take much less time and be much less taxing on your graphics processing units(GPUs) however it is known to be slightly less accurate than Batch Gradient Descent. We are trying to achieve the best classification rate that we can on certain objects so we will not be using Stochastic Gradient Descent

Backpropagation with Synthetic Gradients Algorithm: Synthetic Gradients is an algorithm that was recently developed by Deep Mind and has already shown greater prediction accuracy than Stochastic Gradient Descent and Batch Gradient Descent. [11] The only down side is that it requires very powerful GPUs and is the most computationally expensive of the 3 algorithms. One of the issues with Stochastic and Batch Gradient Descent is that the neural network can only be update after a full forward and backwards pass of itself. This can be thought of as a serial process, however the Synthetic Gradients Algorithm accomplishes back propagation with a more parallel process. [12]

Conclusion: The benefit of Synthetic Gradients is that when neural networks become more complex with a lot of layers, the time it takes to update the weights in the network becomes a massive bottleneck. Synthetic Gradients is an algorithm that decouples the layers of the neural network and updates these layers all independently from one another so that each layer no longer has to wait for all of the other layers to update. Because of this, our neural networks will be using the Synthetic Gradients method.

Types of Online Learning: Online Learning or Sequential Learning or Lifelong Learning is the concept that intelligent agents should be able to learn more about their environment and previous knowledge throughout its lifetime . For example, a child will likely know what a laptop is but when they learn something knew about laptops such as the differences between Microsoft and Mac, they will add this information to their memory. To put it bluntly, not only Neural Networks, but AI in general is very bad at this seemingly basic process. This process is Online Learning.

Mitigating Catastrophic Interference: Catastrophic Interference or Catastrophic Forgetting is an issue that comes about from Online Learning. A backprop- agation network will forget information if it learns input A and then input B. Catastrophic forgetting occurs because when many of the weights in a neural network, where knowledge is stored, are changed, it is impossible for prior knowledge about past data to be kept intact. During sequential learning, the inputs become mixed with the new input being superimposed over top of the old input. To recognize multiple sets of patterns, the network must find a place in weight space that can represent both the new and the old output.

Dual Memory Architecture (DMA): The dual memory architecture for online learning is a semi-common algorithm in regards to implementing Sequential Online Learning. The Dual Memory Architecture (DMA) is an algorithm that creates multiple deep networks to handle online learning. Each of these networks is constructed when a specific amount of data from an unseen probability distribution is accumulated, and thus creates a deep representation of the data in a specific time. [13] Essentially, when new information comes in about past data, this new information, another neural network is created as a branch off of the previous network and new information is stored in the weights of the new neural network.

Elastic Weight Consolidation (EWC): In brains, synaptic consolidation enables continual learning by reducing the plasticity of synapses that are vital to previously learned tasks. [14] EWC performs a similar process with neural networks. For two different neural network training periods A and B, while learning task B, EWC protects the performance in the previous task A by constraining the parameters to stay in a region of low error for task A. EWC must be highly customized to your application because it needs to remember only the past weights that are important to your system. This may require a lot of time and effort to see significant classification improvements. Additionally, in an experiment carried out by DeepMind [asdf] where an intelligent agent attempted to learn several Atari games using EWC, there wasnt a significant difference between Online Learning with EWC and regular Batch Learning. This result from EWC could recur in our project because both apply online learning to a mainly visual context.

Local Winner Take All (LWTA): In LWTA, new information in an Artificial Neural Network(ANN) overlaps with old information. To decide which is useful, each layer of the ANN is separated into blocks of size n where n is the number of potential neurons that will compete with each other to pass on their information or activation. This decision is made through local competition of these neurons, the neuron with the quantitatively highest decided activation value is passed on to the network. [15] This can be thought of as: The neural network will remember the knowledge passed to the neuron with the highest activation value. One problem with this algorithm is that activation functions in the neural network must be linear, which has shown to have detrimental effects on ANNs that operate based on classification.

Conclusion: The preceding algorithms are online learning algorithms that minimize the amount of catastrophic forgetting that occurs in the neural network. We used the EWC type of Online Learning for our Capstone Project. Online learning can be incredibly difficult, EWC will allow us to implement an online learning algorithm that is successful in increasing the rate at which our intelligent agent classifies objects in its environment without nearly as much of the training and overhead that the other algorithms require.

IV. Findings

A. Results

In this implementation of common object recognition, using a Fetch robot and Transfer Learning with DeepMind's Inception ResNet, our team was able to achieve an average accuracy of 96.9%

References

[1] V. Mazzari, "Ros robot operating system," https://www.generationrobots.com/blog/en/2016/03/ros-robot-operating-system-2/, 03 2016, (Accessed on Nov 12).

[2] Microsoft, "Welcome to robotics developer studio," https://msdn.microsoft.com/en-us/library/bb648760.aspx, (Accessed on Nov 12).

[3] "What is naoqi framework," http://doc.aldebaran.com/1-14/dev/naoqi/index.html, (Accessed on Nov 12).

[4] J. K. A. Pagani and D. Stricker, "Detection and identification techniques for markers used in computer vision," http://vesta.informatik.rwth-aachen.de/opus/volltexte/2011/3095/pdf/6.pdf, 01 2010, (Accessed on Nov 13).

[5] E. Ackerman, "Robots use rfid to find and navigate to household objects," https://spectrum.ieee.org/automaton/robotics/robotics-hardware/robots-rfid-find-and-navigate-objects, 09 14, (Accessed on Nov 13).

[6] R. A. Russell, "Odour detection by mobile robots," https://books.google.com/books?id=O2FTjO9X2xQC&pg=PA117&lpg=PA117&dq=Marking+techniques+for+robots&source=bl&ots=uODndBxYdz&sig=jbNGH4OUaFGM2tnXnsL6dVSCTTM&hl=en&sa=X&ved=0ahUKEwjNwPHs-r3XAhVMwmMKHTlPBN0Q6AEIRTAH#v=onepage&q=Marking%20techniques%20for%20robots&f=false, 07 1999, (Accessed on Nov 13).

[7] B. Sapp, A. Saxena, and A. Y. Ng, "A fast data collection and augmentation procedure for object recognition," http://ai.stanford.edu/~asaxena/robotdatacollection/stairdatacollection.pdf, 2008, (Accessed on Nov 13).

[8] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *International Journal of Remote Sensing*, vol. 28, no. 5, pp. 823–870, 2007. [Online]. Available: https://doi.org/10.1080/01431160600746456

[9] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." vol. 3, pp. 958–962, 2003.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[11] S. Raval, "Synthetic gradients explained," https://www.youtube.com/watch?v=qirjknNY1zo, 10 2017.

[12] Czarnecki, "Understanding synthetic gradients and decoupled neural interfaces," https://deepmind.com/research/publications/understanding-synthetic-gradients-and-decoupled-neural-interfaces/, 03 2017.

[13] K. Lee, Kwak, "Dual-memory deep learning architectures for lifelong learning of everyday human behaviors," https://bi.snu.ac.kr/Publications/Conferences/International/IJCAI2016_SLee.pdf.

[14] R. P. Kirkpatrick, "Overcoming catastrophic forgetting in neural networks," Deep Mind.

[15] s. t. j. Srivastava, jonathan, "Compete to compute," https://papers.nips.cc/paper/5059-compete-to-compute.pdf.

8 PROJECT DOCUMENTATION

8.1 Data Capturing

So the very first thing that I had to do was install Linux using Ubuntu 14.04 LTS. The reason for this is that the intelligent agent (also known as Bandit) is essentially a Linux computer and also all the ROS code that is required needs Linux to function properly. Once Linux is installed, the first thing you want to do is install terminator which is another form of terminal that makes it easy to work with multiple terminal windows as ROS requires you to use several at one time just to get the data capturing to work. Once terminator is installed, you should immediately changed your local machines .bashrc file to includes the ROS Master URI and Port of Bandit. Once this is done, you need to set up your catkin workspace with a simple install command from the ROS tutorial website. Next, you need to add the necessary dependencies for Bandit by searching for packages that are missing. Once you have these packages you can start mapping the room that you will be collecting data in and once you have that map, update the launch file to use that map with RVIZ and youre all set to run the data capture. Now all you have to do is launch the launch file and then then once RVIZ is loaded up, you should plot 4 points in 3D space around the object so that the data capturing code knows where the object is to collect the data from. Now simply just run the data capture and bandit will take care of the rest.

So now that we know all the steps necessary to get to the data capturing portion of the project, lets talk about the method that bandit uses to collect data. He first starts out by generating a random goal in the room that fits under the criteria of the minimum and maximum radius from the object. Once this goal is determined, bandit determines a random route to take to get to this goal by maneuvering around the room without running into any objects. If he isnt able to make it to the goal on his first attempt, he will stop and generate a new random route in an attempt to reach that goal again. This process will repeat until the reaches that goal. Once he makes it to the goal, he will adjust his spine by a random amount to get different height angles of the object. After adjusting his spine, he will zero in on the object and center himself with it. At this point he will quickly sleep for 3 seconds and then take a picture. The reason for sleeping for 3 seconds is so that the pictures dont come out blurry and it gives me a small time frame in which I can add image noise. This process will repeat for x amount of pictures that are going to be gathered during that data capturing session.

User Guide

For our senior design capstone project, we will build an image classifier on top of an autonomous robot. By leveraging ROS (Robot Operating System) and the existing mobile robot platform, we will provide a Convolutional Neural Network (CNN) model that utilizes online learning so that the robot can continuously learn to recognize objects in its environment.

Installation

1. git clone [this repo]

2. pip install -r dependencies.txt

or chmod +x install.sh

./install.sh

3. Set PYTHONPATH in your .bashrc

```
export PYTHONPATH=${PYTHONPATH}:~/Deep-Learning-For-Object-Recognition-on-a-Mobile-Robot/src
```

To quickly run the learning models

1. Download the image files from
   https://drive.google.com/drive/folders/1BR0TPG5I_UDa_JNA9NSm8fQuZPBc4DZT?usp=sharing
   and execute: mv ~/Downloads/image_data src/

2. cd into src/learning and execute: python2 model.py [path to image directory]

3. cd into src/learning and execute: python2 in_resnet.py [path to image directory]
   If the script has trouble downloading the weights file, manually install it from:
   https://github.com/fchollet/deep-learning-models/releases/download/v0.7/inception_resnet_v2_weights_tf_dim_ordering_tf_kernels.
   then execute: mv ~/Downloads/inception_resnet_v2_weights_tf_dim_ordering_tf_kernels.h5 Deep-Learning-For-Object-Recognition-on-a-Mobile-Robot/src/learning/

ROS Dependencies

For now, this repository only supports Python 2

If you are working on the fetch robot in the OSU Robotics department, make sure you have access and can connect to the hidden robotics wifi network.

Install ROS Indigo for Ubuntu 14.04LTS

Install Catkin with:

```
sudo apt-get install ros-indigo-catkin
```

Create catkin_ws by running the following commands:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

Add /src/image_capture/lifelong_object_learning package to the robots catkin_ws/src directory

Add the following two lines to your .bash_rc

```
export ROS_MASTER_URI=http://bandit.engr.oregonstate.edu:11311
export ROS_IP=00.000.000.000  //Add your computers IP adress
```

Install the necessary ROS fetch packages to your local computer

```
sudo apt-cache search fetch
sudo apt-get install ros-indigo-fetch-description
```

To Run Data Capture

Make new map

ON ROBOT SIDE

1. ssh into robot and source workspace

```
    ssh username@bandit.engr.oregonstate.edu
    source catkin_ws/devel/setup.bash
```

2. Start creating map of room by running the command below and then driving the robot around room to scan

```
    rosrun gmapping slam_gmapping scan:=base_scan _odom_frame:=odom
```

ON LOCAL COMPUTER SIDE

3. Once the entire room is scanned, save map with:

```
    rosrun map_server map_saver -f <map_name>
```

4. Save another copy to make the keepout file

```
    rosrun map_server map_saver -f <map_name_keepout>
```

5. Edit the map_name_keepout.pgm with an image editor to add black lines to indicate which sections of the map the robot should be staying out of. You can look at ours in results/setup/graf_HRI_keepout.pgm to get an idea of what it can look like.

6. Add .yaml and .pgm of both maps (4 files total) to the robot under lifelong_object_learning/mapping

```
    scp map_name.ext
username@bandit.engr.oregonstate.edu:/home/user/catkin_ws/src/lifelong_object_l
earning/mapping/
```

SWITCH BACK TO ROBOT SIDE WINDOW

8. Edit the launch file to tell it to read from these newly added map files

```
vim catkin_ws/src/lifelong_object_learning/launch/startup.launch
```

Run capture_data.py

ON ROBOT SIDE

1. Open new terminal window to ssh into robot and source workspace

```
ssh username@bandit.engr.oregonstate.edu
source catkin_ws/devel/setup.bash
```

2. Launch the startup launch file

```
roslaunch lifelong_object_learning startup.launch
```

ON LOCAL COMPUTER SIDE

3. Open new terminal window and run the command below to make sure you are talking to Fetch on robotics network

```
rostopic echo base_scan
```

4. Copy the fetch.rviz file from results/setup/fetch.rviz to your current working directory and then run the following command:

```
rosrun rviz rviz -d fetch.rviz
```

ON ROBOT SIDE

5. Repeat Step 1

6. Register Points in point cloud by clicking "register point" in top right of RVIZ GUI and put 4 points around the object.

7. Add marker to visualization by clicking add in bottom left of RVIZ GUI and look for /object_point_marker. Once added, you should see a blue dot appear where you clicked in RVIZ.

8. Run capture_data.py to start data capturing process with the --class and --number parameters set accordingly.

```
rosrun lifelong_object_learning capture_data.py --class mug --number 1
```

or

```
python path/to/capture_data.py --class mug --number 1
```

9 Recommended Technical Resources for Learning More

Youtube is a great online resource for visual explanations of many of the technologies we researched throughout the project. Videos posted to the site offer a wide range of expertise levels, and can often demonstrate a complex idea easier than reading a related scientific journal. It is important to verify this information, however, as it isn't peer reviewed the same manner journals are. Chris Eriksen is Dr. Smart's graduate student currently working the Personal Robotics Lab, and acted as another mentor for our group throughout the year. He attended our weekly client meetings and offered advice on each step of the project. He shared code, research, and hardware resources with our team and helped provide access to necessary connections.

10 Conclusions and Reflections

10.1 Michael's Reflection

Over the course of the entire year, I've learned so much over robotics with ROS and machine learning in general. Prior to engaging in this project, I had very minimal (basically nonexistent) knowledge of both of these topics. This project on its own is so complex with the AI deep learning algorithms, data capturing algorithm, and complexity of the neural networks that we researched. To me it is still surreal to think that the neural networks we were working with are mathematical models inspired by the biological brain that function in a similar matter in that it allows a network to learn within its environment over time with a web of interconnected neurons. And seeing it come to life essentially and start making its own decisions as to whether the items we were feeding through the net were staplers, chairs, screwdrivers, mugs or books.

One of the biggest non-technical takeaways that I got from working on this year long project is the experience of working with colleagues of mine on the same project for such an extended period of time. We had our ups and downs as teams but ultimately came out on top and we couldn't have done it without each of us being committed to our Senior Capstone. In addition to this, I believe the pitch we practiced for expo was a good experience as we got to try that out in person with people of all levels of competence. It was great being able to share with others our project and explain to them this new technology that is paving the future of machine learning.

This project has gave me valuable experience with project work. This was a whole new level of commitment, work, planning, and research that seems to more closely simulate what our jobs will be like as professionals. Dealing with the GitHub repository and making sure that our repository was kept organized and well structured was slightly more challenging than I imagined since I'm used to smaller projects where I can just throw everything into a "src" folder with a makefile and call it good.

This project has also taught me that project management is an essential component to team success. You could have the best programmers in the world working together but without adequate project management, that team will likely fail or take a lot longer than necessary to finish a project. I am grateful for Julian stepping into that role to guide us on this machine learning adventure and help keep us on track by ensuring we knew what things we were being held responsible for in order to have a successful end product.

Working in teams for this project has definitely given me the best feel for what it will be like in industry to work with other coworkers on projects together. We were regularly collaborating and reaching out to one another when we needed something such as a particular piece of code or explaining how something worked. I've also learned that working with your friends on the same team can be challenging because sometimes its hard to draw that line between being casual

with one another and having to be professional for certain parts. This likely wouldn't have been an issue if I didn't know my team members since I would've been more to the point about getting the project done and not lolly-gagging around as much since we were comfortable with each other.

If I could do this project over again, I would try to not be as ambitious as to the amount of work we could accomplish on a project that involved complex methodologies that over half the team wasn't familiar with or had any experience with. Instead, we should have had more stretch goals that could've totally been doable but would've also allowed us to get a fully functioning software pipeline working first before trying to add on fancy new methods for machine learning. I mean I spent over half my time learning ROS and figuring out robotics things that I wasn't familiar with which was essential to us being able to collect data to feed into our network. I also would've enjoyed this project a second time around if we didn't have to do all the documentation that is required for this class as that would've freed up more time to spend on the project.

10.2 Julian's Reflection

I have learned so many new technical skills from this project which crossed several disciplines. I learned about robotics and how to develop on top of the ROS platform, I also walked away with tons of new skills in data preparation, machine learning, and data visualization. I spent most of my time on the machine learning side of the project and it is partly because of those responsibilities that I was able to acquire multiple job offers in the field of machine learning. I learned how to implement neural networks in Tensorflow and Keras and used several machine learning topics such as: Transfer Learning, Hyperparameter Tuning, Online Learning, and Data Preparation and Sampling. In addition to many technical skills I also learned many non-technical skills. By communicating my ideas to peers and the client, my public speaking and presenting skills improved tremendously. I also learned about LaTex, making video presentations, and writing papers with IEEE formatting. From this project, I have learned that group work can be challenging, you get many new perspectives and great ideas that you wouldn't have thought of through group discussion. However there is an added time sink that occurs when you have to convince your teammates that things should be done a certain way or have to explain a process instead of just implementing it right away. I have definitely learned a lot about project management. Most importantly, the way I like to be managed and the things that incentivize me to do my best work, may not be applicable to everyone else. As a manager you need to find out how your employees and team members work best and help make that a reality. Additionally, I learned to delegate work and determine who would be best for what portion of the project. Working in a team means cooperating with other team members, I have learned to be sensitive to other team members schedules and current workload when delegating work and when asking questions. It is also important to note that you can't just assign all the work that no one wants to do to yourself because this will likely be too much work. Finally, if I could do the whole project over,

10.3 Miles' Reflection

We covered so much material over the past three terms, it's important to pause and record your findings so you can look back on them. While I did keep my weekly blog updates, these only looked at that small scale, and it can be nice to reflect on the entire year as a whole as a conclusion to the document and the project. Our project was quite technical, and involved complex software algorithms and technologies. Over the course of the year I've learned much about AI and robotics from a broad point of view, and the experience with the Fetch and ROS programs is valuable going forward as more tools in my arsenal. I also realized how little I knew about the machine learning algorithms I'm familiar with, and

working on our classification program forced me to research these technologies. These areas had largely been a ”black box” to me, but learning about each layer made the concepts as a whole easier to grasp, and allowed me to provide more valuable feedback to my teammates. Besides studying the technologies we included in our project, capstone has been great practice for my more meta career skills as well. From software version control and project management, to team communication, client interactions, and mock interviews, these non technical skills will help me in my computer science career. While I have worked on long term projects before, I have never played such a large role in an endeavor this big before. Term long projects do not require the same level of organization and coordination between group members that capstone does. And not only is our final product a functional software pipeline, the coding standards are much stricter to maintain a cohesive repository and one that will be references by others in the future. It is extremely important to plan out the implementation of features for the overarching project, and dividing tasks intelligently and fairly is a skill on its own. In a team with only three members, each of us had to work on completing tasks and managing the work flow. I'm grateful for the tools and software we utilized outside of coding; OneNote made organizing content between each other and across each term. Websites like GitHub make assigning technical tasks and following up with each other as painless as possible too. As the year progressed Julian took on more of the role as the project coordinator, but we all worked to hold each other accountable for our decided responsibilities. Working as a team for a time span as long as this project is almost another part of the homework requirements. Knowing each team members strengths and weaknesses helps with assigning tasks, and communicating regularly keeps thinks moving smoothly. Often, if one team member was struggling with something the others would be able to view the problem with their perspective and help find a solution, however this is only possible with good communication between all members. If I could redo the entire capstone class, I would've slowed down, stepped back, and looked at the grand scheme of the course more frequently. When you're racing to meet deadlines it can be hard to analyze your progress in meaningful ways, but if I had taken the time to access my situation more often I feel I would've saved a lot of time and effort in the end. While I did try to document my process as much as possible, I also highly value methodical documentation. I would have spent more time creating guides, summaries, and interfaces to assist myself for the portions of the project I worked on. In the end if I can't remember how to run the code I wrote the other week and didn't take the time to properly notate my work it wasn't very useful to begin with!

11 Appendix 1: Essential Code Listings

GitHub URL: https://github.com/julianweisbord/Deep-Learning-For-Object-Recognition-on-a-Mobile-Robot

Learning Code Essential File Paths:

- src/learning/classify.py
- src/learning/in_resnet.py
- src/learning/model.py
- src/learning/sequential_learning.py

Data Capturing Code Essential File Paths:

- src/image_capture/lifelong_object_learning/src/data_capture/capture_data.py
- src/image_capture/lifelong_object_learning/src/launch/startup.launch