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

ROS is our primary tool 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].

3.1.2 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.

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.