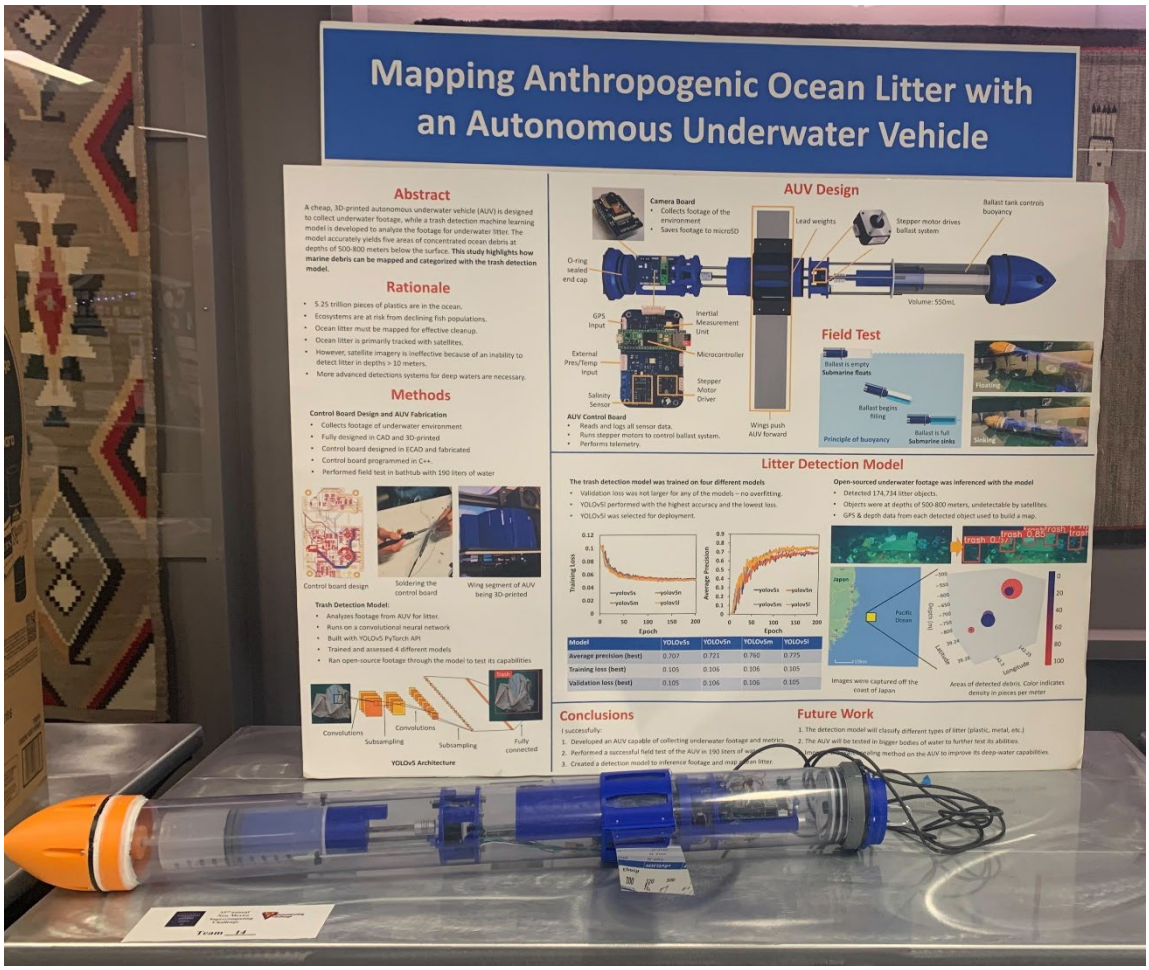
Finalist Reports
2022-2023



https://supercomputingchallenge.org

Printed by PNM
Compiled by the
Supercomputing Challenge

Cover: **Mapping Anthropogenic Ocean Litter with an Autonomous Underwater Vehicle**

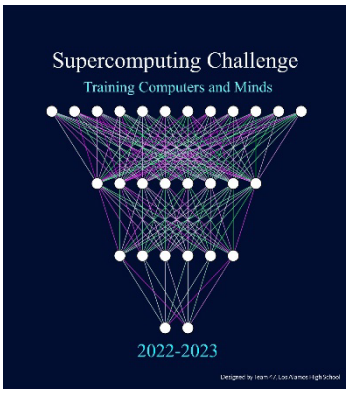
Team 14, Los Alamos High School
Team Member: Daniel Kim
Sponsor: Michela Ombelli

Winner in the Technical Poster Competition

Notification: These final reports are presented in an un-abridged form, though printing might be done in black and white. Complete copies of the final reports, including color graphics, along with the code, are available from the archives of Supercomputing Challenge web site: https://supercomputingchallenge.org .

New Mexico Supercomputing Challenge
2022 – 2023 Finalist Reports

Table of Contents



Supercomputing Challenge Vision

The Vision of the Supercomputing Challenge is to be a nationally recognized program that promotes computational thinking in science and engineering so that the next generation of high school graduates is better prepared to compete in an information-based economy.

Supercomputing Challenge Mission

The Mission of the Supercomputing Challenge is to teach teams of middle and high school students how to use powerful computers to analyze, model and solve real world problems.

About the Supercomputing Challenge

The Supercomputing Challenge (the Challenge) is an exciting program that offers a truly unique experience to students in our state. The opportunity to work on the most powerful computers in the world is currently available to only a very few students in the entire United States, but in New Mexico, it is just one of the benefits of living in the "Land of Enchantment."

The Challenge is a program encompassing the school year in which teams of students complete science projects using high-performance computers. Each team of up to five students and a sponsoring teacher defines and works on a single computational project of its own choosing. Throughout the program, help and support are given to the teams by their project advisors and the Challenge organizers and sponsors.

The Challenge is open to all interested students in grades 5 through 12 on a nonselective basis. The program has no grade point, class enrollment or computer experience prerequisites. Participants come from public, private, parochial and home-based schools in all areas of New Mexico. The important requirement for participating is a real desire to learn about science and computing.

Challenge teams tackle a range of interesting problems to solve. The most successful projects address a topic that holds great interest for the team. In recent years, ideas for projects have come from Astronomy, Geology, Physics, Ecology, Mathematics, Economics, Sociology, and Computer Science. It is very important that the problem a team chooses is what we call "real world" and not imaginary. A "real world" problem has measurable components. We use the term Computational Science to refer to science problems that we wish to solve and explain using computer models.

Those teams who make significant progress on their projects can enter them in the competition for awards of cash and scholarships. Team plaques are also awarded for: Teamwork, Written Report, Professional Presentation, Research, Creativity and Innovation, Environmental Modeling, High Performance, Science is Fun and the Judges' Special Award, just to name a few.

The Challenge is offered at minimal cost to the participants or the school district. It is sponsored by a partnership of federal laboratories, universities, and businesses. They provide food and lodging for events such as the kickoff conference during which students and teachers are shown how to use computers, learn programming languages, how to analyze data, write reports and much more.

These sponsors also supply time on the supercomputers and lend equipment to schools that need it. Employees of the sponsoring groups conduct training sessions at workshops and advise teams throughout the year. The Challenge usually culminates with an Expo and Awards Ceremony in the spring at the Los Alamos National Laboratory or in Albuquerque. During the Covid pandemic, three Expo and Awards Ceremonies, as well as two Kickoffs, were held virtually.

History

The New Mexico High School Supercomputing Challenge was conceived in 1990 by former Los Alamos Director Sig Hecker and Tom Thornhill, president of New Mexico Technet Inc., a nonprofit company that in 1985 set up a computer network to link the state's national laboratories, universities, state government and some private companies. Sen. Pete Domenici, and John Rollwagen, then chairman and chief executive officer of Cray Research Inc., added their support.

In 2001, the Adventures in Supercomputing program formerly housed at Sandia National Laboratories and then at the Albuquerque High Performance Computing Center at the University of New Mexico merged with the former New Mexico High School Supercomputing Challenge to become the New Mexico High School Adventures in Supercomputing Challenge.

In 2002, the words "High School" were dropped from the name as middle school teams had been invited to participate in 2000 and had done well.

In the summer of 2005, the name was simplified to the Supercomputing Challenge.

In 2007, the Challenge began collaborating with the middle school Project GUTS, (Growing Up Thinking Scientifically), an NSF grant housed at the Santa Fe Institute.

In 2013, the Challenge began collaborating with New Mexico Computer Science for All, an NSF funded program based at the Santa Fe Institute that offers a comprehensive teacher professional development program in Computer Science including a University of New Mexico Computer Science course for teachers.

2022—2023 Challenge Awards

33rd Annual New Mexico Supercomputing Challenge winners

First Place

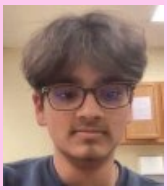
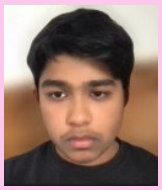
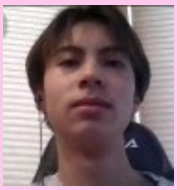
Team 4 — **La Cueva High School**

Machine Learning based Accessible Mobile App for Activity Recognition and Freezing of Gait Monitoring in Parkinson's Patients

Team Members: Abitpal Gyawali, Aditya Koushik, Aiden Shoppel, Amandeep Prasankumar, Venkata Menta

Sponsor: Jeremy Jensen

WONDERFUL ACHIEVEMENT!


Supercomputing Challenge 89

Second Place

Team 14 — **Los Alamos High School**

Mapping Anthropogenic Ocean Litter with an Autonomous Underwater Vehicle

Team Member: Daniel Kim

Sponsor: Michela Ombelli

GOOD WORK!!

Supercomputing Challenge 87

Third Place

Team 6 Sandia High School
SINGS: A Simple Interactive N-body Gravitation Simulator
Team Member: Tristan Eggenberger
Sponsor: Bradley Knockel

CONGRATULATIONS!

Supercomputing Challenge 85

Supercomputing Challenge students across the state have completed their computational projects on topics such as Parkinson's disease, astrophysics, ocean science, fire science, and environmental issues. This year the Supercomputing Challenge celebrated their 33rd annual Expo and Awards Ceremony on April 24 and 25, 2023 which featured student final presentations, judging and tours held at the Los Alamos National Laboratory and an award ceremony held in Los Alamos.

In this program, middle school and high school students are mentored by a community of volunteer scientists, computer programmers and professors. Several alumni also serve as volunteers. The Supercomputing Challenge partners include the New Mexico Consortium, Los Alamos National Laboratory, Sandia National Laboratory, Public Service Company of New Mexico, Air Force Research Laboratory, Westwind, and most New Mexico colleges and universities. A complete list of sponsors and supporters of the Challenge is on the website at https://supercomputingchallenge.org/22-23/sponsors.php.

"I am always impressed with the students in our state. We are so proud to be able to showcase their abilities," said David Kratzer, the executive director of the Supercomputing Challenge.

By participating in the Challenge, students learn to be prepared and successful in any career. They learn to be persistent by practicing their computer programming skills, completing research, and meeting deadlines to cross the finish line. The Challenge likes to refer to itself as an academic marathon, and these students should be recognized as critical thinkers, communicators, collaborators, and computer scientists.

Scholarships worth $15,000 were awarded to participating students. Many other awards were distributed ranging from $50 per team member for finishing the academic marathon to team

prizes of up to $1000 for 1st and additional prizes for other categories such as teamwork, technical writing, programming ability, and community impact. Random drawings were held for $50 door prizes.

The Supercomputing Challenge is open to New Mexico middle and high-school students, including home-schooled students. Students work in teams and follow their own interests to choose a topic to computationally model. New students interested in becoming involved can make plans to start the next academic year, by contacting consult@supercomputingchallenge.org

Scholarships and Other Awards

A complete list of all winning student teams.
https://supercomputingchallenge.org/22-23/expo/AllWinnersList.pdf

Scholarship winners

Scholarships worth $15,000 were awarded at the Supercomputing Challenge Awards Ceremony to eight seniors: Isel Arragon, Zachariah Burch, Gloria Engle, Violet Kelly, Daniel Leon, Kiara Onemoto, Isaac Rankin, and Ayvree Urrea.

All final reports are online.
https://supercomputingchallenge.org/22-23/finalreports/submitted.php

More about the Supercomputing Challenge

"The goal of the yearlong event is to teach student teams how to use powerful computers to analyze, model and solve real-world problems," said David Kratzer, Executive Director. "Participating students improve their understanding of technology by developing skills in scientific inquiry, modeling, computing, communications, and teamwork."

The New Mexico Supercomputing Challenge teaches written and oral communication, collaboration with peers and professionals, critical thinking including research and coding including computer modeling to middle and high school students throughout the state. Any New Mexico middle-school or high-school student, including home-schooled students are eligible to participate in the Supercomputing Challenge. Students follow their own interests to choose a topic to model.

Teams Finishing the Challenge and submitting final reports:

Team 1, La Cueva High School, *An Integrated Approach for Immediate and Long-Term Air Quality Regulation and Monitoring in Mexico*
Team Member: Eliana Kai Juárez
Sponsor: Ashli Knoell

Team 2, Cottonwood Classical and Del Norte High School, *For Crying "Drought" Loud*
Team Members: Ayvree Urrea, Kiara Onomoto, Violet Kelly
Sponsor: Karen Glennon
Mentor: Flora Coleman

Team 3, McKinley Middle School,
Best Conditions For Mushrooms And How They Dissolve Trash
Team Member: Lennon Martinez
Sponsor: Sharee Lunsford
Mentors: Karen Glennon, Patty Meyer

Team 4, La Cueva High School, *Machine Learning based Accessible Mobile App for Activity Recognition and Freezing of Gait Monitoring in Parkinson's Patients*
Team Members: Abitpal Gyawali, Aditya Koushik, Aiden Shoppel, Amandeep Prasankumar, Venkata Menta
Sponsor: Jeremy Jensen

Team 6, Sandia High School, *SINGS: A Simple Interactive N-body Gravitation Simulator*
Team Member: Tristan Eggenberger
Sponsor: Bradley Knockel

Team 7, Sandia Preparatory School,
Simulating the Creation of Kerosene Using Sunlight Water Vapor and Air
Team Members: Humdan Qureshi, Patrick Blewett
Sponsor: Nicholas Aase

Team 8, Capital High School, *Analyzing the Impact of COVID and its Correlation with Vaccines*
Team Members: Julia Almeida, Leslie Gutierrez, Guadalupe Rojo, LaTrisha Padilla
Sponsor: Irina Cislaru
Mentor: Jon Brown

Team 9, Capital High School, *A Robotic Fire and Fire Weather Measuring System, Smokey*
Team Members: Zachariah Burch, Isel Aragon, Britny Marquez, Daniel Leon
Sponsor: Barbara Teterycz
Mentor: David Ritter

Team 10, Academy For Tech & Classics,
Predicting Heuristics Related to the Controversiality of a Social Media Post
Team Members: Henry Tischler, Gene Huntley
Sponsor: Jenifer Hooten

Team 13, La Cueva High School, *Tracking Cislunar Orbits*
Team Members: Hadwyn Link, Ximena Serna, Kylen Reyner
Sponsor: Jeremy Jensen
Mentor: Mario Serna

Team 14, Los Alamos High School,
Mapping Anthropogenic Ocean Litter with an Autonomous Underwater Vehicle
Team Member: Daniel Kim
Sponsor: Michela Ombelli

Team 15, Cleveland High School, *Artificially made or not?*
Team Member: Layla Barela
Sponsor: Ashli Knoell

Team 17, New Mexico School for the Arts, *Locating Smoke Plumes & Fires Accurately*
Team Members: Deisy Jaramillo, Tania Pizano, Elisea Jackson, Margot Esparza Torres
Sponsor: Sarah Rowe
Mentor: Stephen Guerin

Team 18, Media Arts Collaborative Charter School,
Mental Health Apps And The Need For A Crisis Button
Team Members: Eduardo Dorado, Ana Sofia Rodriguez, Wyatt Wade
Sponsor: Tanya Mueller

Team 19, Santa Fe Preparatory School, *Genetics of ADHD*
Team Members: Camila Carreon, Greta Swanson, Luke Rand, Nandita Ganesan
Sponsor: Jocelyn Comstock
Mentors: Juergen Eckhert, Quinton Flores

Team 20, Truman Middle School, *Relationship Between Deforestation and Climate Change*
Team Members: Sebastina Puentes, Yahir Prieto
Sponsor: Natali Barreto Baca

Team 21, Truman Middle School, *Amazon Deforestation*
Team Members: Briana Anaya, Kamilah Chavez, Yaritzel Castro
Sponsor: Natali Barreto Baca

Team 22, Tucumcari High School,
Using Sensors to Detect and Maintain Distances in the Real World
Team Members: Aaron Chand, MiKayla Klinger
Sponsors: Thomas Evans, Morisa Evans
Mentor: Harry Henderson

Team 24, Tucumcari High School,
Solid Oxide Fuel Cell Visual Recognition of Images to Show 3d Printed Part Effectiveness
Team Members: Nolan Ryen, Marcus Lopez
Sponsor: Thomas Evans
Mentor: Creighton Edington

Team 25, Justice Code/International/Harrison,
Sparks vs. Bolts: An Exploration of the Environmental Efficacy of Electric and Gasoline Vehicles
Team Members: Isaac Rankin, Aileen Ukwuoma, Alexandrina Ukwuoma
Sponsors: Caia Brown, Becky Campbell, Ryan Palmer
Mentor: Daniel Fuka

Team 26, Justice Code/International/Harrison, *M.E.D.I.C Bot: AI in Healthcare*
Team Members: Chinyere Offor, Mekhi Bradford, Kingsley Walker, Noel Emma-Asonye
Sponsor: Becky Campbell
Mentors: William Fong, Justin Schmetterer

Team 27, Monte del Sol Charter School, *Stress Anxiety Monitor (SAM)*
Team Members: Gabriella Armijo, Emlee Taylor-Bowlin
Sponsor: Rhonda Crespo
Mentor: Mark Galassi

Team 28, Los Alamos High School,
Developing an Autonomous Aerial Package Transport System
Team Member: Andres Iturregui
Sponsor: Allan Didier
Mentor: Clint Hubbard

Team 29, Tucumcari High School, *Wind Turbine Blade Fencing*
Team Members: Sariah Mardo, Rachel Mardo
Sponsor: Thomas Evans
Mentor: Creighton Edington

Team 35, Justice Code, *The Water Robot*
Team Members: Aaliyha Maestas, Jenifer Batista Ortiz, Sara Santiago, Alonda Jimenez Ramirez
Sponsor: Becky Campbell
Mentor: Dr. Anthony Lupinetti

Team 36, Justice Code, *Deflect Lightning*
Team Members: Alfonso Salas, Estevan Manriquez, Daniel Mason, Leland Martinez
Sponsor: Becky Campbell
Mentor: Dr. Anthony Lupinetti

Team 37, Justice Code, *TESTING THE EFFICACY OF OUR SUPERSPRAY ON THE ORAL MICROBIOME*
Team Members: Valerie Ozigbo, Gloria C. Enga
Sponsor: Becky Campbell
Mentor: Dr. Anthony Lupinetti

Team 39, Justice Code, *AI Basketball*
Team Members: Frances Emma-Asonye, Praise Ejimkonye
Sponsor: Becky Campbell
Mentor: Patty Meyer

Team 40, Justice Code, *How Visuals can affect the way people experience sound?*
Team Member: Chibuike Offor
Sponsor: Becky Campbell

Team 41, Justice Code/International/Harrison, *Tsunami Disaster Detection*
Team Members: Kolton Walker, Delight Emma-Asonye, Leilani Baty
Sponsors: Caia Brown, Becky Campbell, Ryan Palmer

Team 43, Taos High School, *Should we be farming Hemp?*
Team Members: Christopher Muniz, Pedro Escobar, Juan Romo, Orlando Cruz, Jaden Alford
Sponsor: Tracy Galligan

Team 44, La Cueva High School, *Tech Generated Art*
Team Member: Victoria Outkin
Sponsor: Jeremy Jensen

Team 47, Los Alamos High School, *Fire And Water*
Team Member: Andrew Morgan
Sponsor: Nathanial Morgan

Team 50, St. Thomas Aquinas School, *Galaxies Far, Far Away*
Team Member: Catherine Sedillo
Sponsor: Eric Vigil
Mentor: James Sedillo

Team 51, St. Thomas Aquinas School, *Modeling Earthquakes*
Team Member: Ethan Ong
Sponsor: Eric Vigil

Judges

Ed Angel, Professor Emeritus UNM
Daniel Appel, BlueHalo, Supercomputing Challenge Alumni
Joan Appel, Syncronys, Supercomputing Challenge Alumni
Hussein Al Azzawi, University of New Mexico
Gennie Barrett, Talking Talons Youth Leadership
Richard Barrett, Sandia National Laboratory/Retired
Nick Bennett, TLG Learning
Hope Cahill, Santa Fe Public Schools
Yie Sheng Chen, University of New Mexico
Jesse Crawford, GitLab
Sharon Deland, Sandia National Laboratories/Retired
Mohit Dubey, Los Alamos National Laboratory, Supercomputing Challenge alumni
Creighton Edington, Rural Education Advancement Program
Daniel Fuka, Virginia Tech University
Susan Gibbs, Project GUTS
Stephen Guerin, Simtable
Christopher Hoppe, Hoppe's Art Studio, Supercomputing Challenge alumni
David Janecky, Los Alamos National Laboratory/Retired
Elizabeth Jimenez, University of New Mexico
Philip Jones, Los Alamos National Laboratory
Krisha Kalyanam
Amy Knowles, New Mexico Institute of Mining and Technology
Nicholas Kutac, Puget Sound Naval Shipyard, Supercomputing Challenge alumni
Peter Lamborn, Red Violet
Maximo Lazo, University of New Mexico
Scott Levey, Sandia National Laboratories
Joan Lucas, University of New Mexico-Los Alamos
Monique Morin, Sandia National Laboratories

Ziyi Niu, Eastern New Mexico University
Joseph Olonia, Central New Mexico Community College
James Overfelt, Sandia National Laboratories
Veena Parboteeah, New Mexico Highlands University
Chris Peterson
Maureen Psaila-Dombrowski, Los Alamos National Laboratory
Abdalaziz Raad, University of New Mexico
Lonnie Rednour, San Juan College
Dana Roberson, Central New Mexico College
Thomas Robey, Gaia Environmental Sciences
Mary Sagartz
Reffat Sharmeen, Sandia National Laboratory
Tim Thomas, Sandia National Laboratories
Michael Trahan, Sandia National Laboratories
Geoff Valdez, Los Alamos National Laboratory
Peter Watson, Los Alamos National Laboratory
James Wernicke, Los Alamos National Laboratory, Supercomputing Challenge alumni
Uri Wilensky, Northwestern University
Kaley Woelfel, BlueHalo
Louise Yakey, Tutors to Teachers

Do you want to become a supporter of the Supercomputing Challenge?
Please email us at consult@supercomputingchallenge.org for details.

Machine Learning based Accessible Mobile App for Activity Recognition and Freezing of Gait Monitoring in Parkinson's Patients

Team 4, La Cueva High School, April 6, 2022

Team Members : Abitpal Gyawali, Aditya Koushik, Aiden Shoppel, Amandeep Prasankumar, Venkata Menta

Mentor : Jeremy Jensen

Table of Contents

Executive Summary

Parkinson's disease is a progressive neurological disorder that affects movement. It is caused by the death of cells in the brain that produce a chemical called dopamine, which helps regulate movement. An extremely evident symptom of Parkinson's disease is the "Freezing of gait (FOG) ." This is when an individual's feet gets "stuck" to the ground, making it difficult or impossible to take a step. Freezing of gait can be very distressing and significantly impact a person's ability to move around and participate in activities. The current diagnosis treatment for detecting Parkinsons can be time consuming and involves the patient being given a series of questionnaires that provides insight to doctors regarding whether the patient has the disease. So in most cases the patient struggles with the disease before necessary medication and attention is given to the patient, and the problems remain to be persistent as medical professionals are unable to detect the extent of the disease as the diagnosis has no definitive test and is extremely vulnerable to misdiagnosis. In consideration of these disadvantages, this project sets an attempt to solve this problem by using a machine learning model built on a mobile app for activity recognition and freezing of gait detection enabled by the use of triaxial motors in present mobile phones. The problem solution is finding a more accessible, accurate and practical way of identifying Parkinson's disease by detecting freezing of gait via a smartphone app and machine learning. This app will also allow for an easy, hands-free way to keep track of how many FOG events occur per day which is a very useful diagnostic marker to identify progression or regression of disease. With support from the National Science Foundation and in collaboration with Rexa.Info, UC Irvine Machine Learning Repository released a dataset called the "Daphnet Freezing of Gait" in 2013. Using this published dataset, we trained an AdaBoost machine learning model on the dataset, where over 200 FOG episodes were recorded over the course of the experiment. The model learns based on a sliding window of FOG vs Non-FOG events, with over 500 statistical features (including fourier transform) being calculated over the windows. After successful training, an Activity Recognition model built with an XGBoost framework will use a similar sliding window approach to learn activities based on accelerometer data. The AdaBoost model with a Random Forest classifier baseline was trained and tested using a 5 fold cross validation and showed a 87% accuracy in distinguishing FOG windows vs Non - FOG Windows. The original Daphnet study achieved 73% sensitivity in predicting FOG events, while our model achieved 77%. Our XGBoost activity Recognition model scored >90% F1 scores in predicting activities such as walking, jogging, sitting, and standing.The next step taken was to integrate this study on the app, for the front end React Native(JS) CLI + Expo CLI was used to provide compatibility for iOS. In the backend Node JS, Express JS and Tensorflow JS were used. Finally, to obtain the accelerometer data from the phone, expo sensors were used. User testing of the app and further tuning of the models is still in progress.

Introduction

Background: Parkinson's Disease, Freezing of Gait (FOG), Machine Learning Models, Fourier Transform

- ***Parkinson's Disease***

1. ***What is Parkinson's Disease?***

Parkinson's disease is a progressive neurological disorder that affects movement and is characterized by symptoms such as tremors, stiffness, and difficulty with movement and balance. It is caused by a loss of dopamine-producing cells in the brain. While there is no cure for Parkinson's disease, medication and other treatments such as deep brain stimulation can help manage symptoms and improve quality of life. Identifying early signs is important to allow for earlier intervention and support.

2. ***What causes Parkinson's Disease?***

Parkinson's disease is caused by the death of dopamine-producing cells in a region of the brain called the substantia nigra. The exact cause of this cell death is not yet fully understood, but it is believed to be a combination of genetic and environmental factors. Research has identified several genetic mutations that may contribute to the development of Parkinson's disease, but these mutations are relatively rare and are thought to account for only a small percentage of cases. Environmental factors such as exposure to certain toxins, head injuries, and infections may also play a role in the development of Parkinson's disease, although more research is needed to fully understand these connections.There is also evidence to suggest that inflammation and oxidative stress may contribute to the development of Parkinson's disease. Inflammation can damage neurons and disrupt the normal functioning of the brain, while oxidative stress can damage cells and contribute to the death of dopamine-producing cells in the substantia nigra. Overall, Parkinson's disease is a complex condition with multiple factors contributing to its

development. While the exact cause of Parkinson's disease remains unknown, ongoing research is working towards a better understanding of the condition and potential treatments.

3. ***What are some common symptoms of Parkinson's Disease?***

Parkinson's disease is a neurological disorder that affects movement, and the symptoms can vary from person to person. However, some of the most common symptoms of Parkinson's disease include:

- Tremors: Uncontrollable shaking or tremors in one or more limbs, typically starting in one hand or arm.
- Bradykinesia: Slowness of movement, making everyday activities such as getting dressed or brushing teeth more difficult.
- Rigidity: Stiffness and resistance to movement in the limbs or trunk.
- Postural instability: Difficulty with balance and coordination, leading to falls.
- Freezing of gait: Brief episodes where the person's feet seem to get "stuck" to the ground, making it difficult to take a step.
- Changes in speech: Difficulty with speaking clearly or softly, slurring of words, or a monotone voice.
- Micrographia: Small, cramped handwriting.

Other symptoms may include fatigue, depression, anxiety, sleep disturbances, constipation, and loss of sense of smell. It's important to note that not all individuals with Parkinson's disease will experience all of these symptoms, and the severity and progression of symptoms can vary widely.

Freezing of Gait

1. ***What is Freezing of Gait?***

Freezing of gait (FOG) is a common symptom of Parkinson's disease that affects a person's ability to initiate and continue walking. FOG is characterized by a sudden and brief inability to

move the feet, making it feel like the person's feet are glued to the ground. It can occur when starting to walk, turning, or approaching an obstacle.

FOG is a complex symptom that can be caused by a combination of physical and cognitive factors. Physical factors include muscle weakness, poor balance, and rigidity, while cognitive factors can include anxiety, distraction, or hesitation. FOG can be particularly distressing for people with Parkinson's disease, as it can lead to falls and reduced mobility.

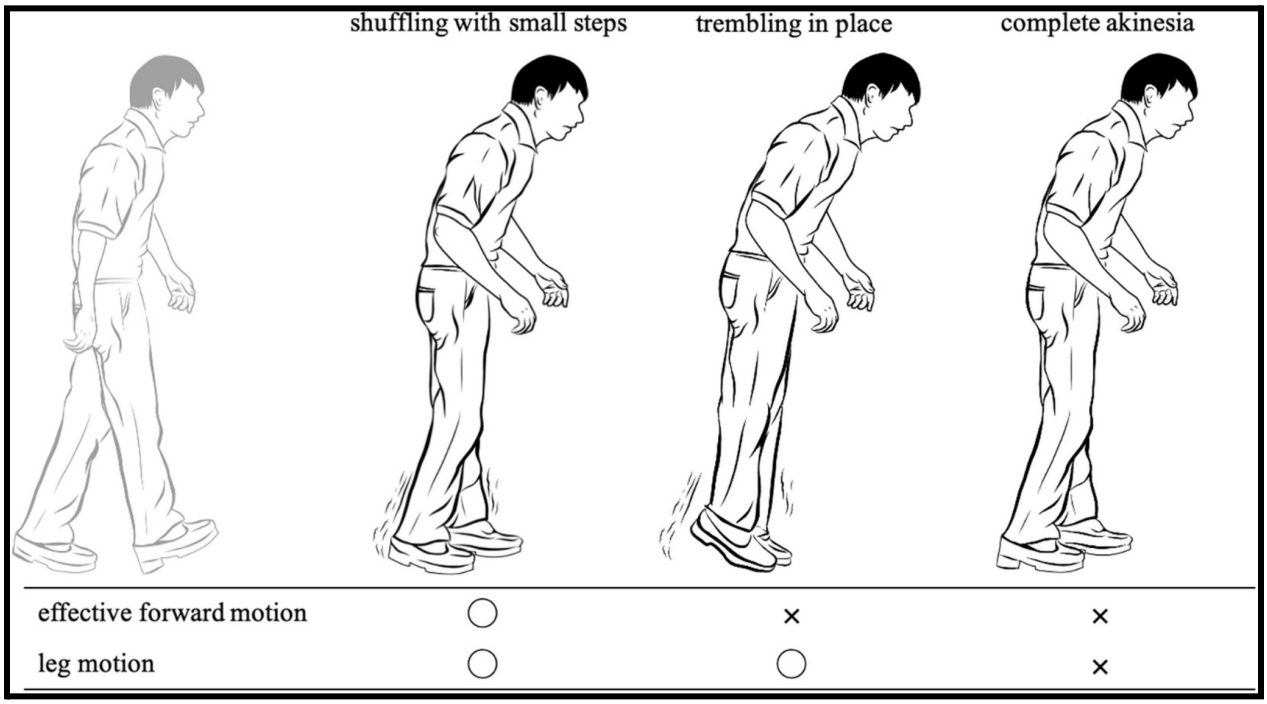


Figure 1

2. How can Freezing of Gait be detected?

Accelerometers are devices that can detect changes in movement and are commonly found on most smartphones. They can be used to detect freezing of gait (FOG) by measuring changes in movement patterns associated with FOG episodes. One way accelerometers can be used to detect FOG is by analyzing the gait pattern during normal walking and comparing it to the pattern during FOG episodes, which is performed by our machine learning model. During FOG, there may be a sudden decrease in the amplitude and frequency of lower limb movements. This reduction in movement can be detected by an accelerometer placed on the foot or leg.

- ***Ensemble Methods***

1. ***What are Ensemble methods?***

Ensemble methods are a set of machine learning techniques that combine multiple models to improve the accuracy and robustness of the predictions. Ensemble methods work by combining the predictions of several models, either through a voting system, weighted average or stacking, in order to produce a single more accurate prediction. Ensemble methods are especially useful in situations where the data is noisy or uncertain, as they can help to reduce the impact of errors or biases in individual models.Some of the most popular ensemble methods include bagging, boosting, and random forests. Bagging involves building multiple models on different subsets of the training data, and then averaging their predictions. Boosting, on the other hand, trains multiple models sequentially, with each model attempting to correct the errors of the previous model. Random forests combine the ideas of bagging and decision trees, by building an ensemble of decision trees on random subsets of the data. Ensemble methods have become increasingly popular in recent years, and are widely used in a variety of applications, including image and speech recognition, natural language processing, and recommendation systems.

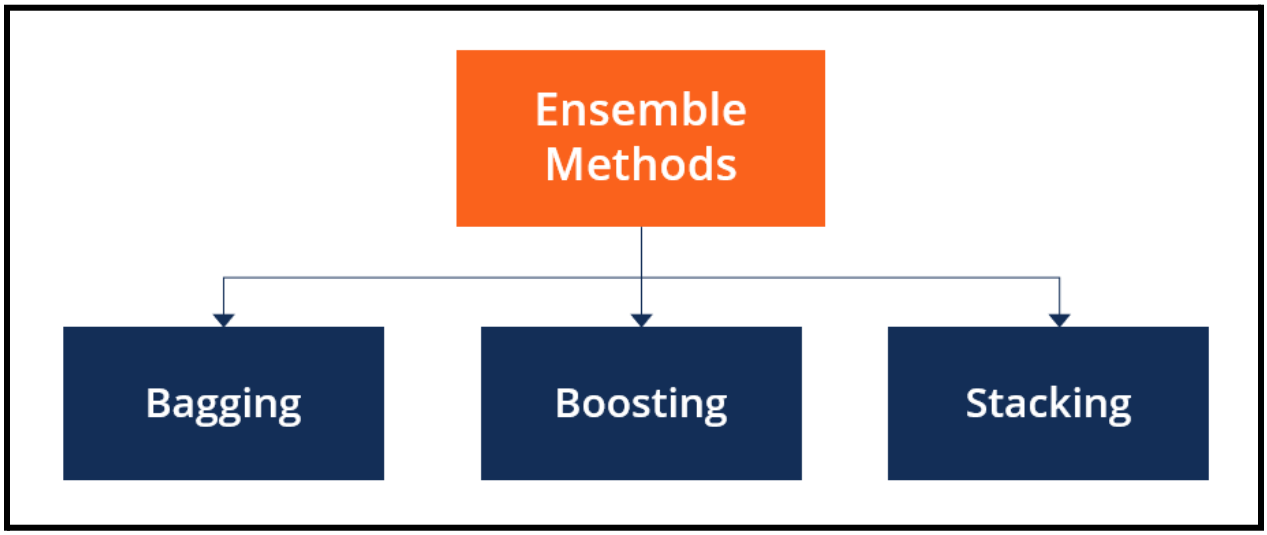


Figure 2

2. *What is ADABOOST?*

AdaBoost, short for Adaptive Boosting, is a machine learning ensemble algorithm that combines weak learners to create a strong learner. In AdaBoost, weak learners are simple models that perform slightly better than random guessing, such as decision trees with a small number of levels or simple regression models. AdaBoost iteratively trains these weak learners on the same dataset, with more emphasis on the samples that were misclassified by the previous model. This allows the subsequent models to focus more on the difficult samples that were misclassified, thereby improving the overall accuracy of the ensemble. During the training process, AdaBoost assigns weights to each sample in the dataset, with more weight given to the misclassified samples. In each iteration, the weak learner is trained on the weighted data, and a new weight is assigned to the model based on its accuracy. The final prediction of the ensemble is a weighted sum of the predictions of each individual model. AdaBoost is known for its high accuracy, robustness, and ability to handle complex datasets with high-dimensional feature spaces. It has been successfully applied to a wide range of applications, such as computer vision, natural language processing, and bioinformatics.

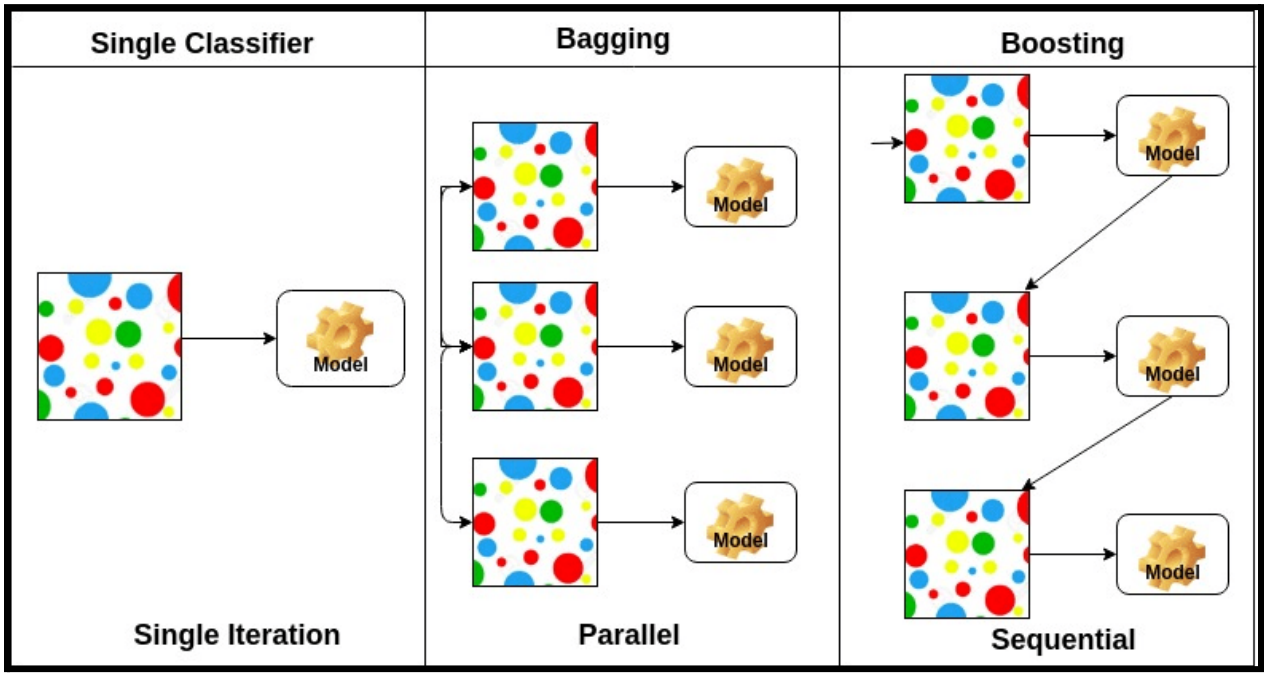


Figure 3

3. *What is XGBOOST?*

XGBoost stands for "Extreme Gradient Boosting," and it is a popular open-source machine learning algorithm for solving classification and regression problems. It is based on the gradient boosting framework and has gained popularity due to its speed, performance, and flexibility. The main idea behind XGBoost is to create a series of decision trees that can be used for making predictions on new data. Each decision tree is trained on a subset of the training data and the features are selected based on their importance in improving the performance of the model. The output of the decision trees is then combined in a way that maximizes the overall performance of the model.XGBoost has several key features that make it popular among data scientists and machine learning practitioners. These include:

- Speed: XGBoost is designed to be fast and efficient, making it suitable for large-scale datasets.
- Regularization: XGBoost has built-in regularization techniques such as L1 and L2 regularization to prevent overfitting and improve the generalization performance of the model.
- Handling missing values: XGBoost can automatically handle missing values in the data without the need for imputation techniques.
- Interpretability: XGBoost provides information on feature importance, which can help in understanding the underlying data patterns.

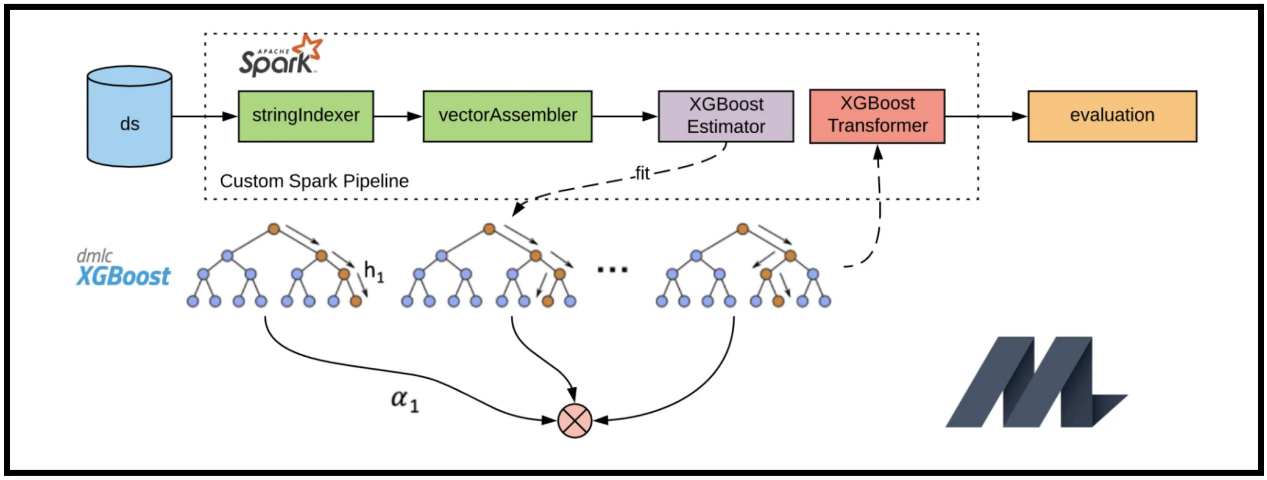


Figure 4

1. What are Random Forests?

Random Forests is a machine learning algorithm used for classification, regression, and other tasks. It is an ensemble learning method that combines multiple decision trees to create a more accurate and robust model. Random Forests works by building multiple decision trees on randomly selected subsets of the data and features. Each decision tree is trained on a random subset of the training data and a random subset of the features. The final prediction is then made by taking the majority vote (in classification) or the average (in regression) of the predictions of all the individual trees.

The main advantage of Random Forests is that it reduces overfitting by averaging the predictions of multiple trees. This makes it less sensitive to noise and outliers in the data, and less prone to overfitting compared to a single decision tree. Random Forests can also handle missing data and perform well with high-dimensional datasets. It can be used for both supervised and unsupervised learning tasks, such as classification, regression, clustering, and anomaly detection.

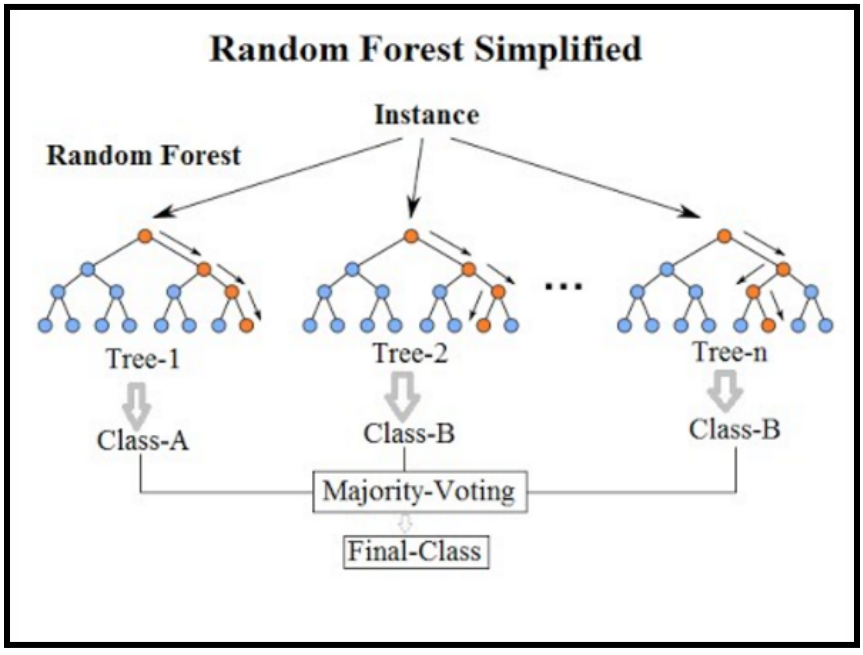


Figure 5

4. What is Fourier Transform

The Fourier transform is a mathematical technique that allows us to convert a signal or a function from its time or spatial domain to its frequency domain. In other words, it decomposes a complex signal or function into its constituent sinusoidal components. This transformation is named after Joseph Fourier, a French mathematician who discovered it in the early 19th century.

The Fourier transform is an important tool in time series forecasting because it can help identify the underlying periodicities and trends in a time series. By analyzing the frequency components of a time series using Fourier transform, we can identify patterns such as seasonality or cyclical behavior that are not immediately apparent from the raw data.Once the frequency components of a time series have been identified, various techniques can be used to model and forecast the time series based on those components. For example, one approach is to decompose the time series into its seasonal and trend components using techniques such as seasonal decomposition of time series (STL) or the Holt-Winters method. The Fourier transform can be used to extract the seasonal component of the time series, which can then be modeled and forecasted separately.

Another approach is to use Fourier analysis to identify the dominant frequencies in the time series and then use that information to build a forecasting model. This approach is often used in signal processing and can be applied to time series data as well. In summary, the Fourier transform is important in time series forecasting because it helps identify the underlying patterns and periodicities in the data, which can be used to build more accurate forecasting models. By using the Fourier transform as a tool for feature extraction and analysis, we can gain deeper insights into the structure of the time series and develop better forecasting models.

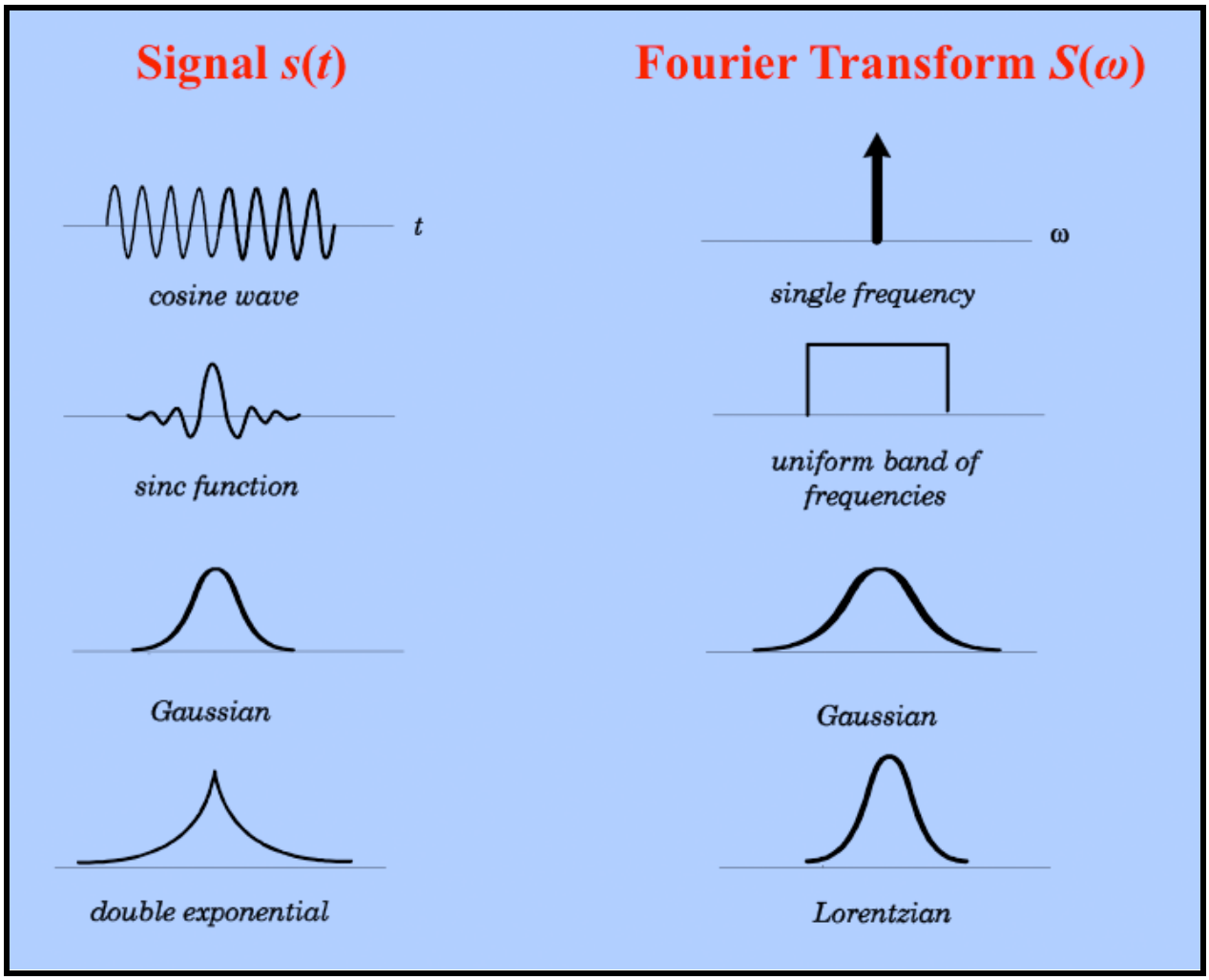


Figure 6 (Shows the following fourier transform or base sinusoid of the signal produced)

5. What is the Fast Fourier Transform?

The Fast Fourier Transform (FFT) is an algorithm that computes the Discrete Fourier Transform (DFT) of a sequence, which is a mathematical transformation that converts a signal from the time domain into the frequency domain. In simpler terms, the FFT is a way of analyzing a signal to determine the frequency components that make it up. For example, if you have an audio signal of a person speaking, the FFT can be used to determine the different frequencies of sound waves that make up that speech signal.

The FFT can be used in time series forecasting to identify and extract useful patterns from the

data in the frequency domain. This can be particularly useful for identifying cyclic patterns or seasonality in the data.

Here are the steps to use the FFT in time series forecasting:

2. Preprocessing: The first step is to preprocess the time series data by removing any trends or seasonal components that can interfere with the frequency analysis. This can be done using techniques such as differencing or seasonal decomposition.
3. Compute the FFT: The next step is to compute the FFT of the preprocessed time series data. This will transform the data from the time domain to the frequency domain.
4. Identify the dominant frequencies: The FFT will produce a frequency spectrum that shows the various frequencies present in the data and their respective amplitudes. The analyst can identify the dominant frequencies by examining the frequency spectrum and selecting the frequencies with the highest amplitudes.
5. Forecasting: Once the dominant frequencies have been identified, they can be used to create a forecast by extrapolating the pattern into the future. This can be done by creating a sinusoidal function with the identified frequency and amplitude and projecting it into the future.
6. Validation: Finally, it is important to validate the forecast by comparing it to actual data. This can be done by comparing the forecasted values to the actual values and calculating the forecast error.

Overall, the FFT can be a powerful tool for time series forecasting as it allows analysts to identify and extract useful patterns from the data in the frequency domain.

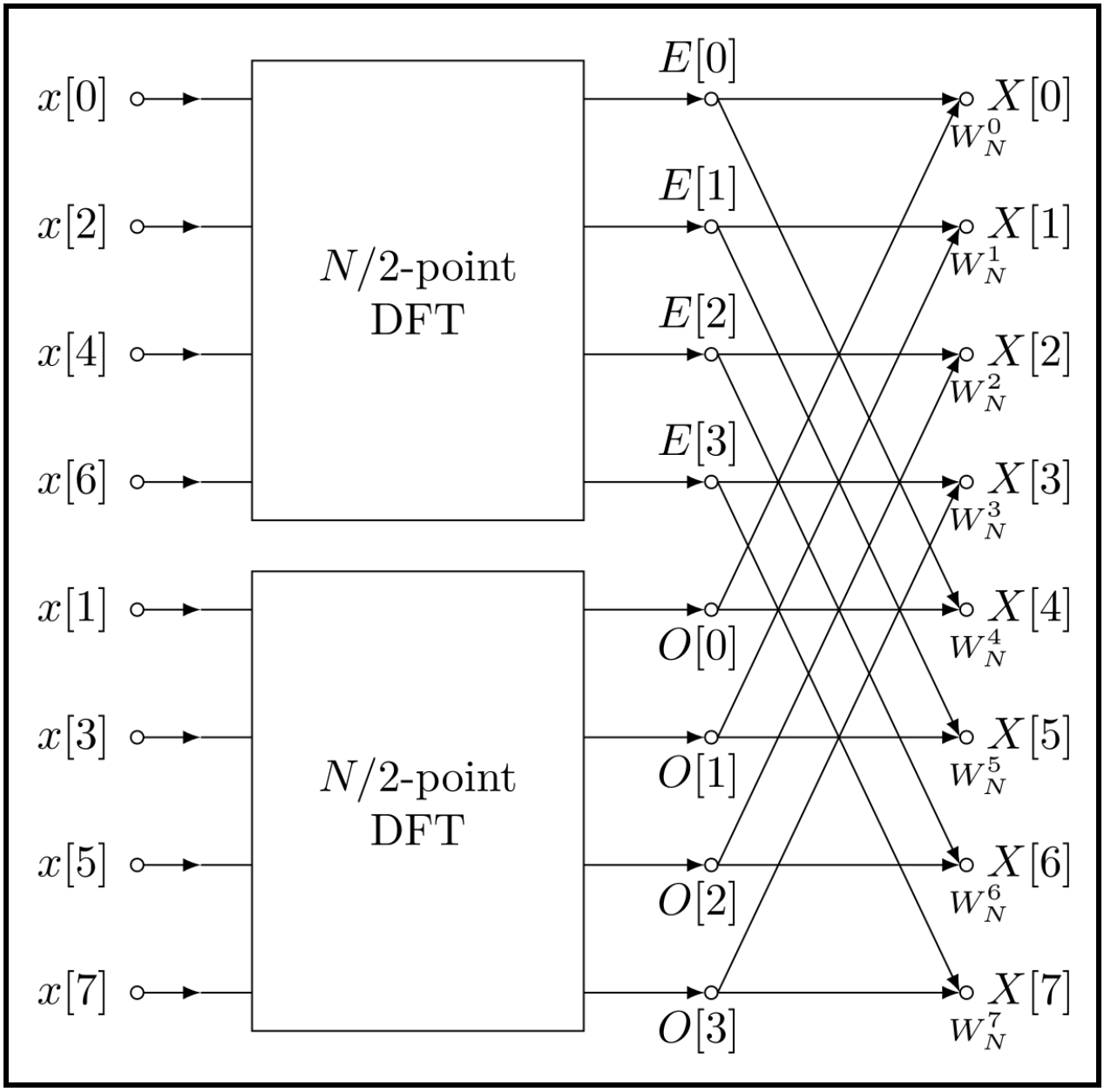


Figure 7 (The following illustration shows 8 data points being split into two groups of 4 points each. Each set goes through an algorithm that analyzes the data points in terms of frequency domains. which represent the signal in the frequency domain. The magnitude and phase of each data point is representative of the amplitude and phase of a corresponding frequency component in the signal. The DFT can then be effectively calculated using the FFT algorithm. The algorithm enables the use of the predictable nature of sinusoids and how they overlap to make the necessary computation)

Purpose

Finding a more accessible and practicable way of monitoring Parkinson's disease by detecting freezing of gait and user activities via a smartphone app and machine learning.

Materials

- Daphnet FOG dataset (https://archive.ics.uci.edu/ml/datasets/Daphnet+Freezing+of+Gait)
- WISDM Activity recognition dataset (https://www.cis.fordham.edu/wisdm/dataset.php)
- React Native (JS) CLI + Expo CLI
- Node JS
- Express JS
- Tensorflow JS
- Expo Sensors
- Python Flask
- Google Cloud Engine

Methods

- **Creating the FOG model**

Step 1: Loading the Dataset

The first step of the project was to train two models, the AdaBoost and XGBoost models, in being able to make accurate predictions from the Activity Recognition and Daphnet Freezing of Gait Dataset. From both datasets, upper thigh acceleration was recorded from three axes. The dataset used to train the model consisted of accelerometer data from subject 5 of the study sampled from three axes at the ankle, thigh, and trunk. Only the UHF (Upper leg thigh acceleration - horizontal forward acceleration [mg]), UV (Upper leg thigh acceleration - vertical [mg]) , and UHL (Upper leg thigh acceleration - horizontal lateral [mg]) acceleration data could be used to train the model, however, since the WISDM activity recognition dataset consisted of values only measured from the thighs.

	Time	AHF	AV	AHL	UHF	UV	UHL	THF	TV	THL	Annotation	ι
26879	420000	50	1039	188	-209	962	90	9	1019	77	0	
26880	420015	50	1039	168	-209	962	101	19	1009	67	0	
26881	420031	50	1029	168	-209	972	80	29	1000	77	0	
26882	420046	50	1029	168	-209	972	90	19	1009	58	0	
26883	420062	50	1019	178	-209	972	111	19	1019	106	0	
...	...	...	...	...	...	...	...	...	...	...	...	
163195	2549937	202	1019	188	-890	111	515	-155	971	291	0	
163196	2549953	202	1009	178	-863	157	434	-155	980	281	0	
163197	2549968	212	1000	158	-881	203	373	-155	971	281	0	
163198	2549984	202	1019	148	-890	240	393	-155	961	252	0	
163199	2550000	202	1019	178	-927	259	434	-155	971	262	0	

111365 rows × 12 columns

Figure 8: Daphnet dataset

The dataset also consisted of a Time column which was measured in milliseconds (ms) where acceleration values were recorded every 15 ms over the course of 36 minutes (duration of the experiment). The final and most important column is the "Annotation" column, where a 0 indicates that a given acceleration reading did not occur during a FOG event (No-FOG) and a 1 indicates that the acceleration reading occurred during a FOG event.

Step 2: Training the model

After the FOG dataset was loaded in, feature calculation was performed through the tsfresh python library, which automatically uses a sliding window through the dataset to compute over 1000 features. In order to prevent high dimensional input data, only a certain number of the original features could be selected for training the model. Most Calculated features showed a negligible correlation with the occurrence of FOG events, so $r > 0.1$ was used as the filter to acquire slightly correlated features with FOG activity. In total, 593 features were calculated and would be used to train the model.

Because the data was a time series dataset in nature, classical machine learning techniques could not be used to predict groups of collective anomalies or FOG events. Because of this, a sliding window (Figure 9) with a size of 50 data points (50 acceleration values) would scan through the dataset and the tsfresh library would compute features over each window of data. Examples of a few of the features calculated are sum of acceleration values, mean acceleration in the window, median acceleration, standard deviation of acceleration, kurtosis, etc. These features would be computed for UHF, UV, and UHL accelerations (x,y, and z axis). A window size of 50 was decided since it resulted in highest model accuracy when compared to other window sizes below 66. A window size of 66 would be a window of approximately 1 second of acceleration data (66*15), which would also mean that there would be a 1 second latency period between model prediction and live data collection. 1 second or 66 rows was decided as the max window size (a larger number would result in a greater latency), and 50 rows was the smallest number that seemed to show the best model performance. Another variable that had to be decided was if a window would be labeled as a FOG or Non-FOG event. Based on extensive testing and experimenting, it was found that a threshold of 0.4 seemed to show the best model performance (If 40% of annotations in the window are a 1, then the entire window would

be classified as a FOG window). This variable has the greatest influence on model prediction, so further tuning of this parameter upon testing of the app is a future goal.

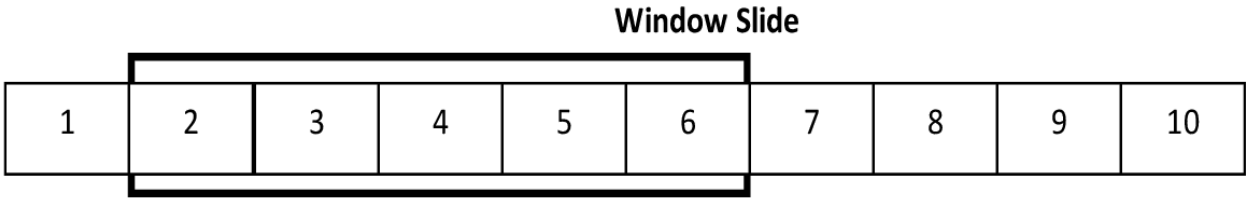


Figure 9: Example of sliding window

After all sliding windows were compressed into a single row or data value, the final dataset ultimately had 1356 rows where each row represents a window of 50 rows of the original data. After the scikit learn min-max scaler was applied to normalize each column of the data between values of 0 and 1, an AdaBoost model with a random forest baseline classifier was trained on the data. The model would be validated through a 5-fold cross validation and a LOOCV (Leave one out cross validation) as described in the results.

- **Creating the Activity Recognition Model**

Step 1: Loading the Dataset

The WISDM activity recognition dataset was loaded into the jupyter notebook file with the pandas module. The dataset consisted of ~1,000,000 rows of triaxial thigh accelerometer data from 36 users as shown by Figure 10.

	user	activity	timestamp	x-axis	y-axis	z-axis
0	1	Walking	4991922345000	0.69	10.80	-2.030000
1	1	Walking	4991972333000	6.85	7.44	-0.500000
2	1	Walking	4992022351000	0.93	5.63	-0.500000
3	1	Walking	4992072339000	-2.11	5.01	-0.690000
4	1	Walking	4992122358000	-4.59	4.29	-1.950000
...	...	...	...	...	...	...
1085355	36	Standing	15049012250000	-0.91	9.43	2.533385
1085356	36	Standing	15049062268000	-1.18	9.51	2.492524
1085357	36	Standing	15049112287000	-1.50	9.53	2.533385
1085358	36	Standing	15049162275000	-2.07	8.77	2.179256
1085359	36	Standing	15049212262000	-2.14	9.89	3.255263

1085360 rows × 6 columns

Figure 10

The “activity” column consisted of annotation for 6 possible activities the user was engaging in: Walking, Jogging, Sitting, Standing, going upstairs and going downstairs (Figure 11 and 12). Because the goal of creating the activity recognition model was to determine when the FOG model should be activated, the model needed to be able to accurately predict when a user is walking/jogging vs sitting/standing.

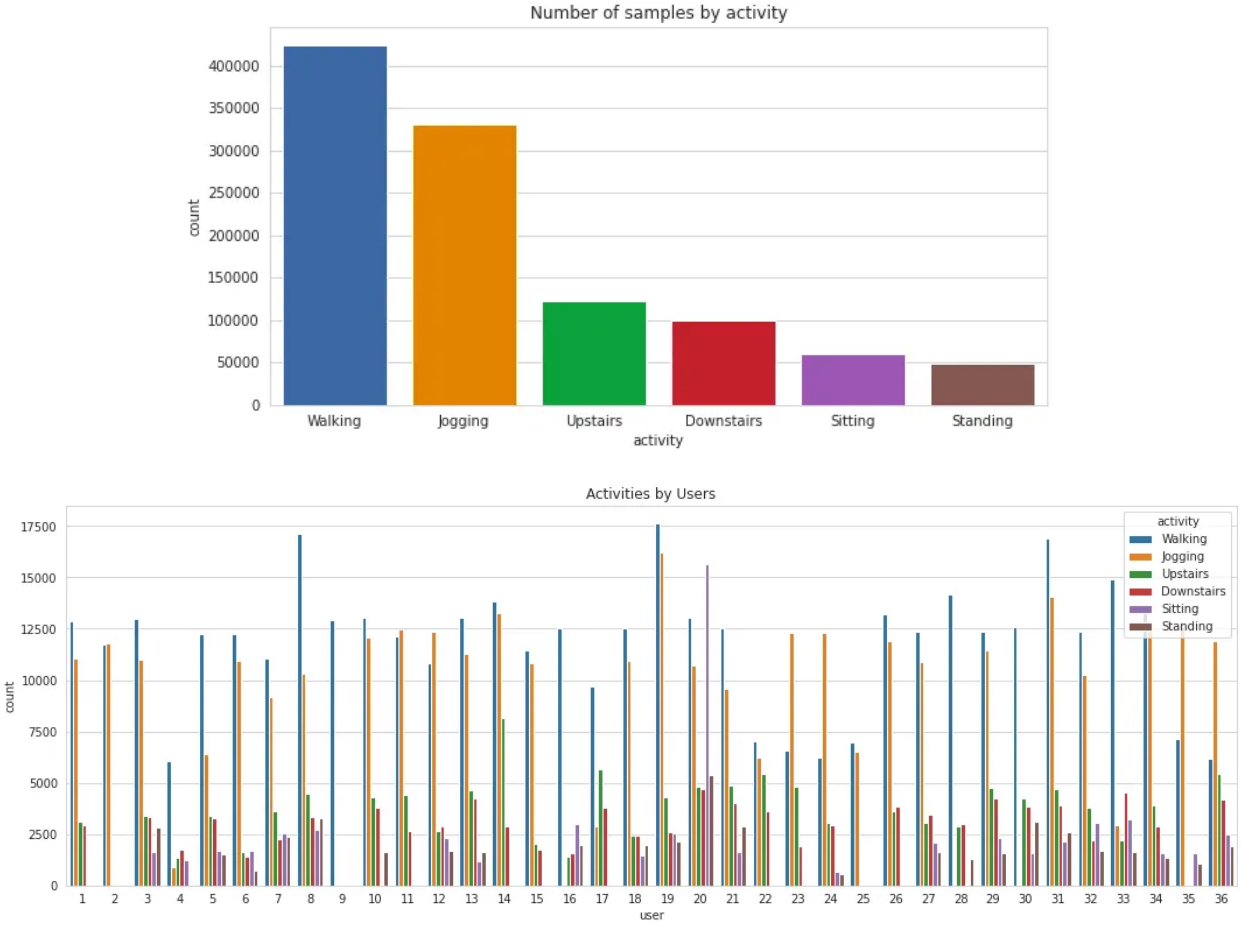


Figure 11 and 12

Step 2: Training the model

The process for training the model was similar to the FOG model. Because this dataset was sampled at 50 ms, a 15 row window size had to be created in order to produce the same prediction time (0.75 seconds) for both FOG model and activity recognition model (FOG: 50

rows * 15 ms vs Activity Recognition: 15 rows * 50 ms). The process of feature engineering was slightly different from the FOG model.

First, 18 simple statistical features were calculated for each axis on the 15-row sliding window as the entire dataset was scanned (18*3 = 54 features). These statistical features condensed accelerometer data from 15 rows into 1 value for each of the three axes of acceleration. The calculated features were mean acceleration, standard deviation, average absolute deviation, minimum acceleration, maximum, difference of maximum and minimum values, median, median absolute deviation, interquartile range, negative values count, positive values count, number of values above mean, number of peaks, skewness, kurtosis, energy, average resultant acceleration, and signal magnitude area. Finally, the fast fourier transform and power spectral density functions were applied to each window and all above features were recalculated from the new arrays. After the feature engineering, Figure 13 shows the final dataset before training the model, where each row in the below dataset represents a window of 15 rows in the original dataset. In total 156 features (excluding label column) were engineered from the original triaxial acceleration values.

	x_mean	y_mean	z_mean	x_std	y_std	z_std	x_aad	y_aad	z_aad	x_min	...	z_skewness_psd	x_kurtosis_psd	y_kurtosis_psd
0	3.162000	8.838000	-0.355333	5.561302	2.895201	3.089077	4.192800	2.372533	1.929422	-4.59	...	0.782335	-0.811544	-0.982768
1	4.117333	10.388667	-0.112667	6.115939	4.715891	2.744759	5.055822	4.066756	2.238133	-5.13	...	1.069621	-1.378746	-0.863224
2	3.739333	10.346667	-1.092667	3.188873	3.142998	2.441615	2.363200	2.504889	1.852444	-1.99	...	0.643871	-1.656969	-1.060669
3	3.904667	9.336667	0.306667	6.060706	3.206100	4.152841	4.743644	2.655556	3.227111	-6.66	...	0.494942	-0.678446	-0.667652
4	3.063333	10.270000	-0.693333	6.346753	4.537879	4.434250	5.123556	3.716000	3.317333	-7.35	...	1.072850	-0.676067	-1.307711
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69043	-1.345333	9.430667	2.468007	0.050049	0.045821	0.038908	0.035200	0.031556	0.030510	-1.50	...	0.491357	-0.821850	-1.162280
69044	-1.410667	9.428000	2.451663	0.073072	0.071852	0.053796	0.057778	0.059733	0.043585	-1.53	...	-1.118993	-1.347098	-1.065117
69045	-1.340000	9.457333	2.406261	0.063770	0.052848	0.037439	0.053333	0.043556	0.027846	-1.46	...	0.406866	-1.759316	-0.678545
69046	-1.386000	9.443333	2.434410	0.064992	0.046857	0.057241	0.059200	0.040889	0.046975	-1.50	...	0.023344	-1.876104	-0.677919
69047	-1.335333	9.442667	2.516132	0.103271	0.070943	0.109807	0.070489	0.054133	0.078211	-1.61	...	1.023570	-0.749592	-0.694386

69048 rows × 157 columns

Figure 13

The above dataset was normalized with the min max scaler and was used to train an XGBoost model to make multiclass predictions of a user's activity based on the 156 provided features. The model's accuracy was evaluated through a 5-fold cross validation test.

App Creation

Terminology

Front-end: Frontend is the portion of the app that is shown to the user. It consists of the User Interface (UI) and must be able to handle user inputs; front-end handles all the interactions with the user.

Back-end: Backend of an app is the portion of the app that handles data computation, data storage, and any other logical processes that allow the app to operate. In other words, the back-end handles the logic of the application, and does not interact with the user.

API (Application Programming Interface): APIs are essentially the connection between the front-end and the back-end of the application. It allows these two components to communicate with one-another through a series of requests and responses.

Objective of the Application

The table below demonstrates what each component of the app must be able to do:

Front-end	Back-end	API
- Be able to read and collect iPhone accelerometer data - Be able to send the collected accelerometer data to the back-end - Have a consistent UI that allows the user to navigate the app smoothly - Be able to interact with the back-end	- Be able to receive accelerometer data from the front-end through an API - Be able to feed in the received accelerometer data into a machine learning model and obtain a prediction - Be able to send the model prediction to the front-end through	- Create a stable connection between the front-end and the back-end - Be able to send accelerometer values from the front-end to the back-end - Be able to send predictions from the back-end to the front-end

through an API	an API	

Step 1: Front-end Prototype - Choosing the Software

The first step of creating a front-end for the application is to decide which software to use to make the application. There are several main questions that go into deciding the software:

- What platform will the application run on? Android? iOS?
- Will the application be available on all major platforms?
- What kind of tools are present in the application?
- What kind of tools does the application need access to?

We wanted the application to be available on all major platforms (most notably iOS and Android). In addition to that, the application must be able to read and store accelerometer values of the cell-phone. React Native, a software built by facebook for app development, fulfilled both of these major requirements. In addition to fulfilling these two requirements, our team also had prior experience with React JS (the Javascript library that React Native is built on), allowing us to build the application without having to learn a new language.

Collecting accelerometer values of the cell phone

In order to collect the accelerometer values of the user's phone, we used a React Native library called Expo Sensors. However, Expo Sensors could not be used without implementing the app using Expo CLI. Therefore, our initial prototype of the app was built using Expo CLI. However, in order for our application to fully function, we must be able to send this accelerometer data to our backend through an API. Since we used a local computer to run the backend server (through local host) during development, our server was hosted under HTTP protocol, rather than a more secure HTTPS protocol. However, due to Apple's security protocols, requests to insecure sites (such as those not protected by HTTPS) are blocked. This can be avoided through changing the application's metadata. However, this change of metadata could not be done through Expo CLI due to its high level nature. This problem was fixed through creating a project through the command line command "create-react-native-app", which allowed

projects to incorporate both React Native CLI and Expo CLI. This essentially allowed us to maintain access to the phone's accelerometer data while enabling control over the application's metadata.

The code below creates an exception to Apple's security protocol, allowing the application to connect to my computer locally.

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
    <key>NSExceptionDomains</key>
    <dict>
        <key>{Domain}</key>
        <dict>
            <key>NSIncludesSubdomains</key>
            <true/>
            <key>NSExceptionAllowsInsecureHTTPLoads</key>
            <true/>
            <key>NSExceptionRequiresForwardSecrecy</key>
            <true/>
        </dict>
    </dict>
</dict>
```

Accelerometer values are collected as follows:

- Accelerometer values are collected every 15 milliseconds
- At each instance of collection, acceleration in the x-axis, y-axis, and z-axis are stored in their respective list

- After a certain amount of accelerometer data is collected, the arrays of x-axis acceleration, y-axis acceleration, and z-axis acceleration are sent to the backend through a POST request, and the array is cleared

This overall process is shown in the pseudo code below:

```
x_list = []
y_list = []
z_list = []

data_length = {model's ideal length of data}

//app is running
//DeviceMotion handles the accelerometer value events

DeviceMotion.addListener((accelerometer_data)=>{
     x_list.push(accelerometer_data.x)
     y_list.push(accelerometer_data.y)
     z_list.push(accelerometer_data.z)

     if x_list.length == data_length
          POST Request -> body: [x_list, y_list, z_list]

})
```

This allows for the prototype of the front-end with essentially all the necessities of the application.

Step 2: Back-end Prototype - Choosing the Software

As aforementioned, the backend of the application will handle the application's interactions with the machine learning model. This includes preparing the accelerometer data for model input, which mainly consists of computing the features of the accelerometer data (these features are the inputs to the models). Both the activity recognition model and the FOG detection models utilize complex statistical methods to compute the features. Therefore, it is in our best interest to use a language such as Python, that has the tools necessary to compute these features. In addition to that, the original model was trained on features computed using Python libraries, so in order to minimize the chances of errors in the code, we decided to use Flask (a Python Back-end library) to create our backend. This would allow us to mimic the original method of feature engineering.

Overall Process (Backend) :

The overall process for the backend is as follows:

1) Input arrays (x-list, y-list, z-list) are converted from JSON to a python array
2) The features are computed for the array (for both the activity recognition model and the FOG detection model)
3) Predictions are made by the model
4) These predictions are sent to the frontend as a response to the original POST request

Step 3: Testing

We tested the application on two main components: prediction when the user is still and when the user is walking. The inefficiencies of setting up a controlled experiment to test the application's detection of upstairs (walking upstairs) and downstairs (walking downstairs) discouraged us from testing these two types of predictions. In order to test still and walking, we recorded the accuracy of the application's predictions when the user is still (sitting and standing still) for three minutes, and accuracy of the predictions when the user is walking continuously for three minutes. We perform several trials of these tests (2~3). Through these testing periods, the

iPhone (our testing device) was put inside a pocket, located on the upper thigh. These tests were conducted in a house to make sure the conditions of testing were realistic.

Window Size	15	50
Accuracy - Still	98%	100%
Accuracy - Walking	79%	Data Collection in Progress

Points of Errors → Errors mostly occurred during these conditions:

- Fast paced walking – errors seldom occurred when the user was walking faster, in which the model would predict the user was walking. This most likely occurred from the fact that the training data was taken from older patients.
- Turning – although errors during moderate turns did not occur often, sharp turns would often cause the model to predict the user was jogging

Although the data collection is in progress for measuring the accuracy of walk predictions for the window size of 50, from what we have gathered so far, the accuracy has been between mostly around 85%, showing a significant increase from the 79% with the window size of 15.

Google Cloud App Engine

After enough tests have been performed and the model has been tuned, the app will be run on the google cloud app engine. This will allow for the application to be easily scalable, and will also allow the application to bypass Apple's security guidelines regarding secure domain connections. Figure 14 shows the predicted logic of the application after development.

Progress

So far, the application can do the following:

Front-end	Back-end	API

- Be able to read and collect iPhone accelerometer data - Be able to send the collected accelerometer data to the back-end - Be able to interact with the back-end through an API	- Be able to receive accelerometer data from the front-end through an API - Be able to feed in the received accelerometer data into a machine learning model and obtain a prediction - Be able to send the model prediction to the front-end through an API	- Create a stable connection between the front-end and the back-end - Be able to send accelerometer values from the front-end to the back-end - Be able to send predictions from the back-end to the front-end

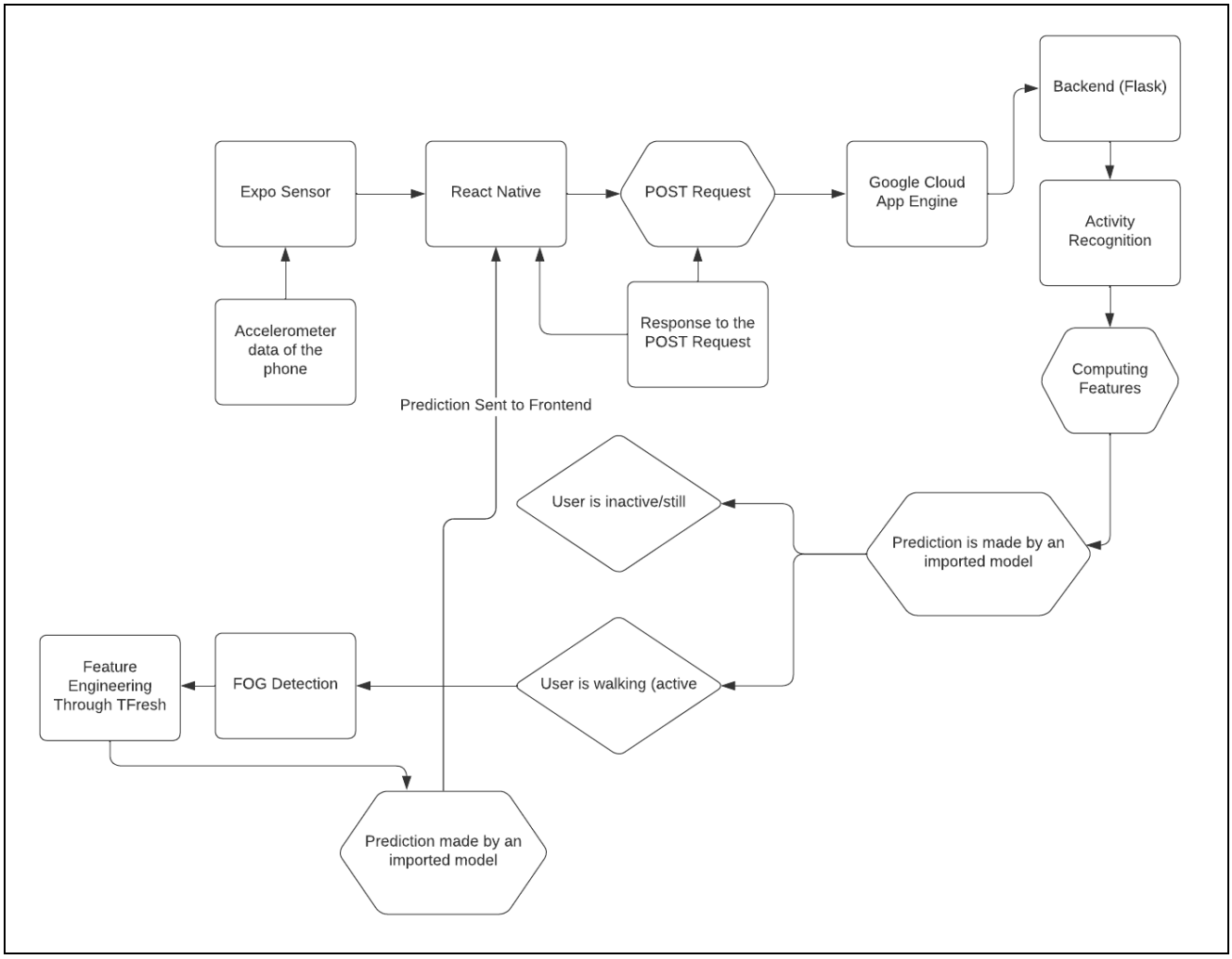
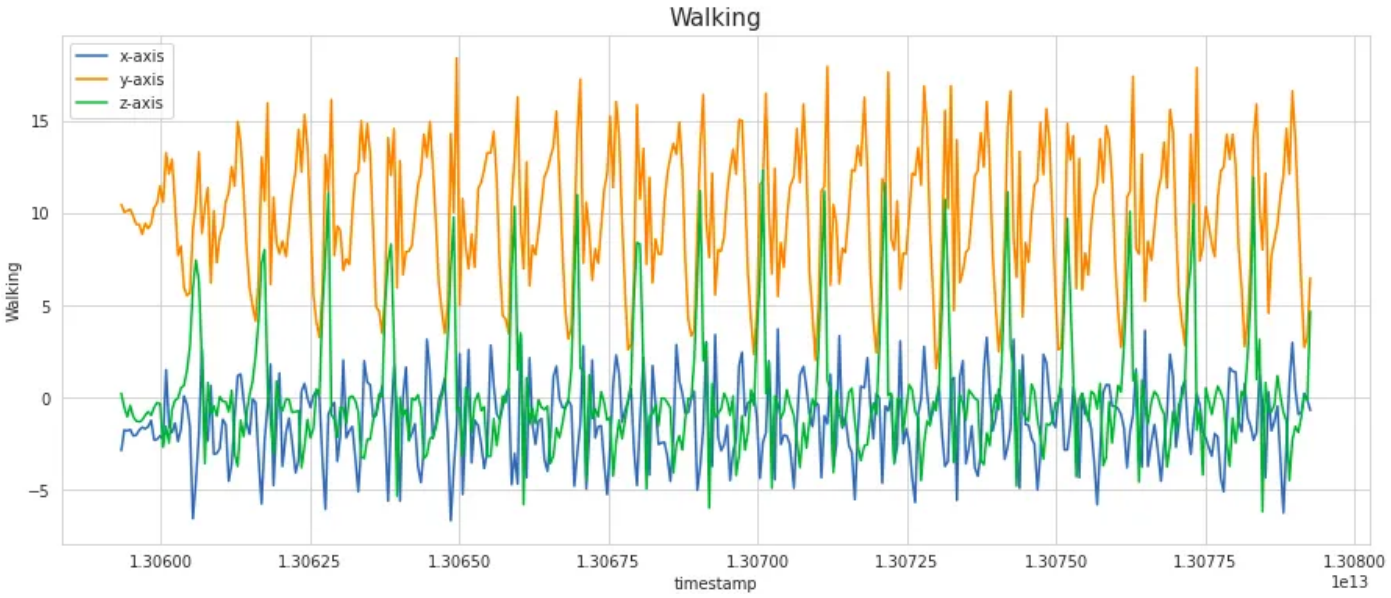
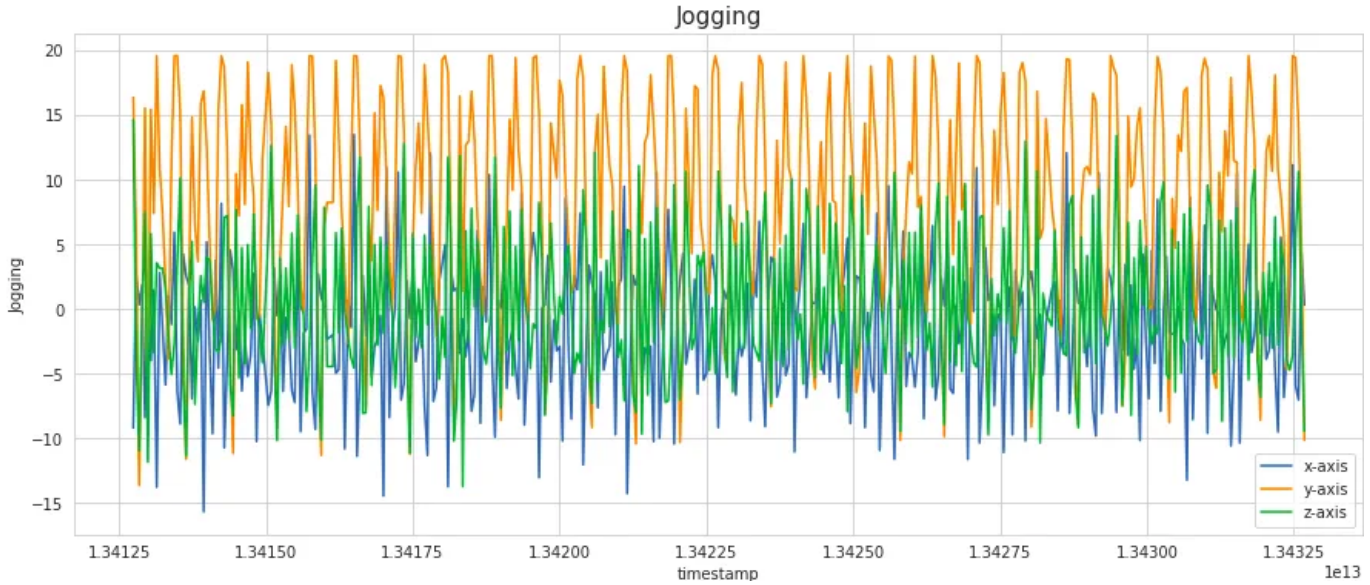


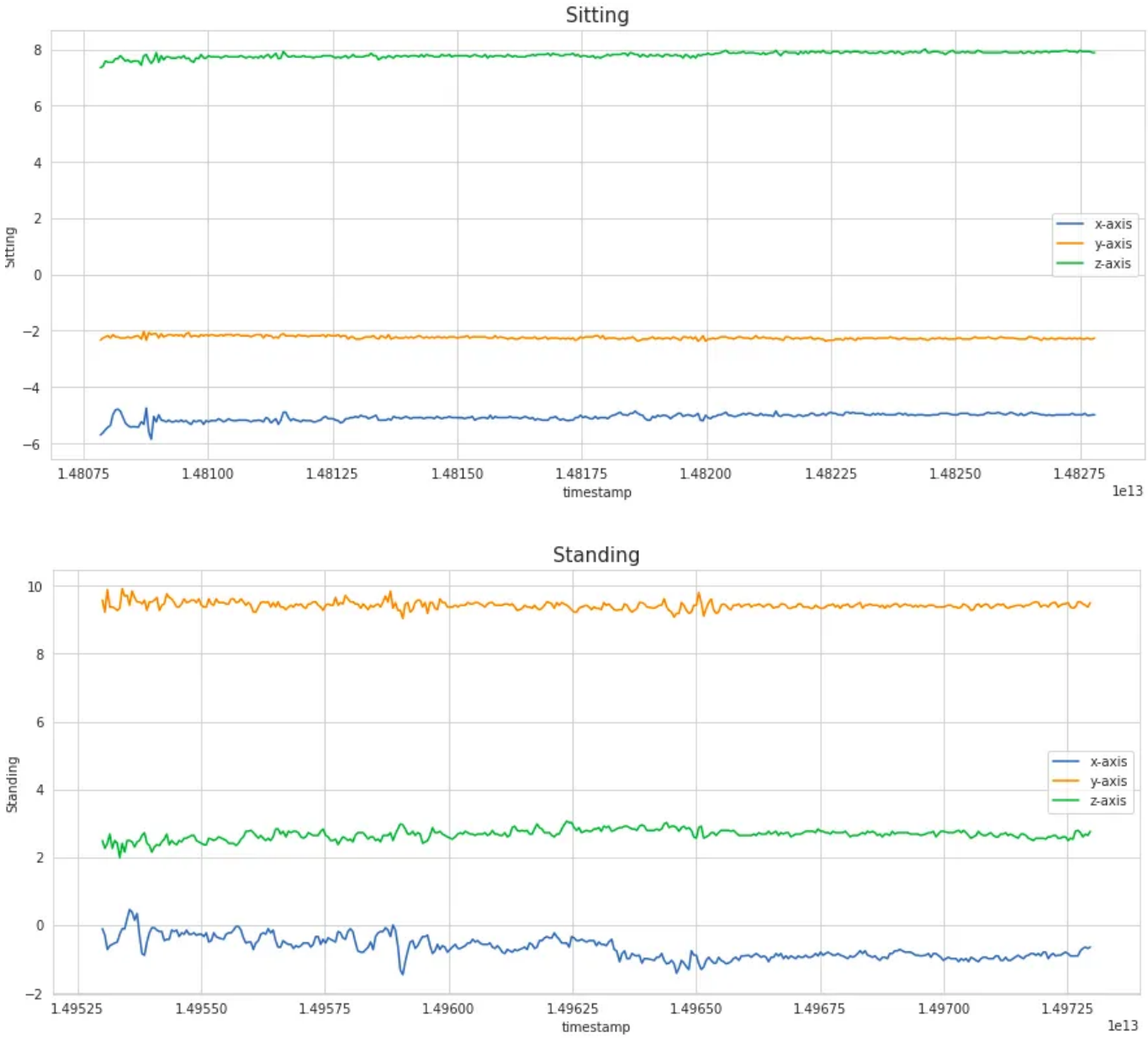
Figure 14: Overview of App - Application Flowchart

Results

Exploratory Data Analysis (EDA)

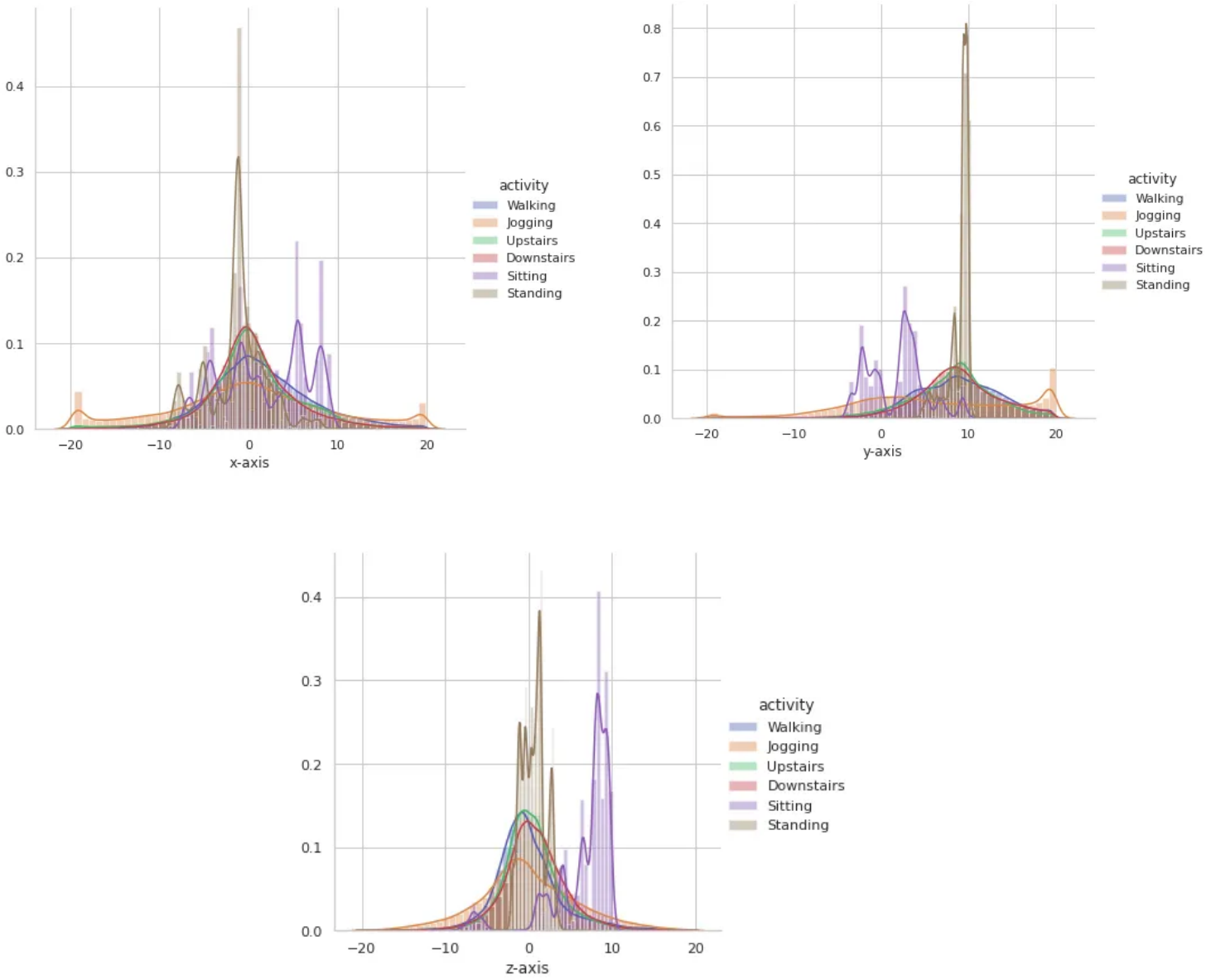






Figures 15-18

Figures 15-18 show matplotlib generated graphs of the triaxial accelerometer data for the first 400 samples of the signal. Visually, accelerometer data while the user is walking/standing have higher amplitudes and frequency than data from sitting/standing.



Figures 19-21

Figures 19-21 show the distribution of signal data for 3 axes with each activity hue, to see if there was an obvious pattern in accelerometer data between different activities. It is observed that there is very high overlap in the data among activities like Upstairs, Downstairs, Walking, Jogging and Standing on all the axes. Sitting appears to have distinctive values along the y-axis and z-axis.

Model Training with Fourier Transform

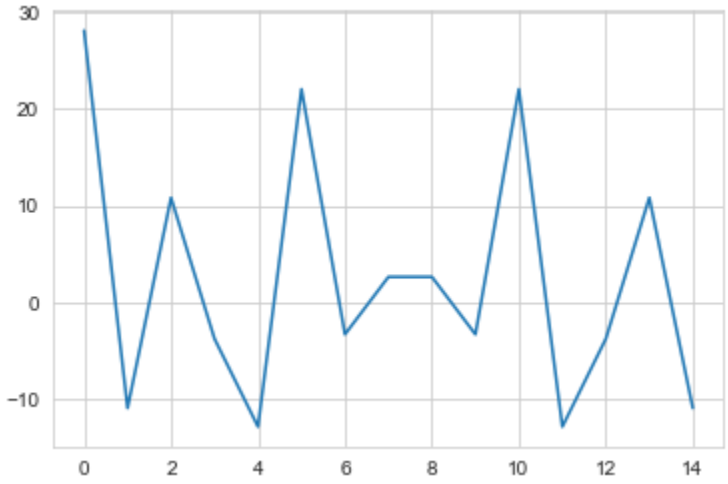


Figure 22

Figure 22 shows a fourier transform applied to the time series dataset. The DC component is the first value and is usually high, but the frequency signal is symmetrical about the center (around the 7th accelerometer value). As mentioned before, 18 key features were extracted from the second half of the fourier transform (mean, median, mode, etc) since the DC component is high and should be discarded.

Model Testing - FOG Model

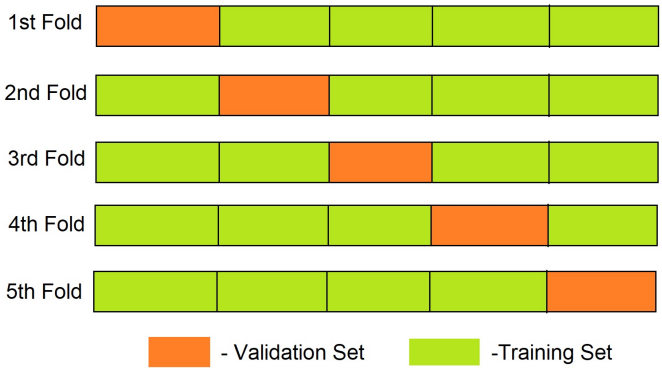


Figure 23

The FOG model was tested with a 5 fold Cross validation performed in Python. Data is typically split using the train test split (70% of the data used to train models, and the other 30% used to test the model accuracy) in classical problems, but splitting the data with a train-test split has more bias and can lead to overfitting, so k-fold splits the data in a different manner (Figure 23). K-fold cross validation involves splitting data into k equal folds (Figure 23). The first k-1 folds are used for training, and the remaining are used for testing. This is repeated for all k-folds, and the mean of the accuracies of each k-fold is returned. The 5-fold CV calculates sensitivity, specificity, F1 score, MCC, and accuracy. Sensitivity measures the ability of the model to predict FOG windows. Specificity measures the ability of the model to predict Non-FOG windows or be able to distinguish FOG from Non-FOG. The equations are shown below.

$$Sensitivity = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives(FN)}}$$

$$Specificity = \frac{\text{True Negatives (TN)}}{\text{True Negatives (TN)} + \text{False Positives(FP)}}$$

Equation 1**:** Sensitivity and Specificity measurements

F1 score is the harmonic mean of precision and recall, which measures the balance between the two metrics. It provides a single number to summarize the overall performance of a model. MCC (Matthews correlation coefficient) measures the correlation between predicted and actual binary classifications, taking into account true positives, true negatives, false positives, and false negatives. It ranges from -1 (completely incorrect) to 1 (perfectly correct), with 0 indicating no correlation. The equations for each are shown below.

$$F1 - score = \frac{2 \text{ x (precision x recall)}}{(\text{precision} + \text{recall})}$$

$$\text{MCC} = \frac{\text{TP x TN} - \text{FP x FN}}{\sqrt{(\text{TP+FP})(\text{TP+FN})(\text{TN+FP})(\text{TN+FN})}}$$

Equation 2**:** F1 and MCC measurements

Accuracy	0.87
F1 Score	0.82
Sensitivity	0.77
Specificity	0.91

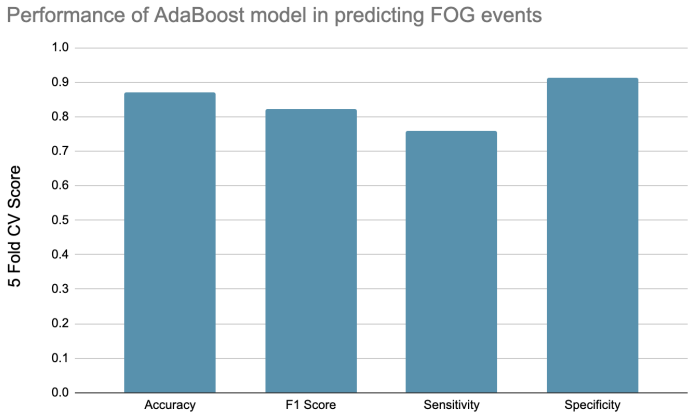


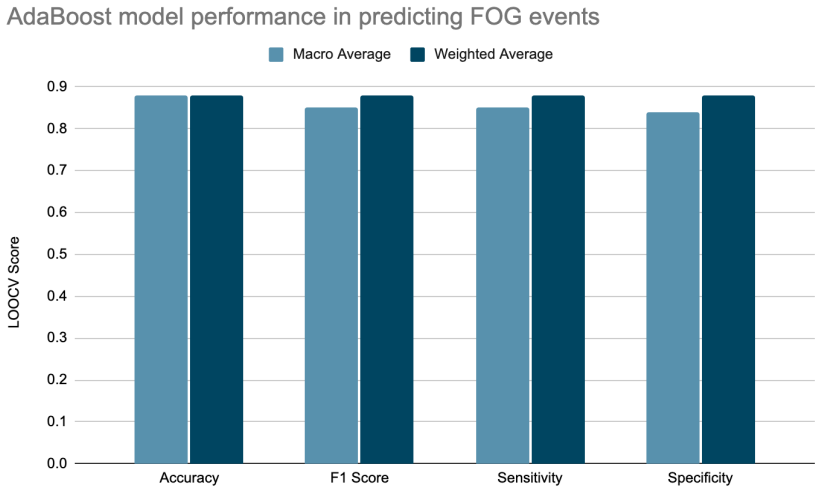
Table 1 and Figure 24: Model Performance

Table 1 and Figure 24 show the performance of the AdaBoost model in predicting FOG windows vs Non-FOG windows (windows of accelerometer data annotated during FOG events). The scores are the average of the model's performance for each fold during the 5-fold CV. The model seems to have slightly outperformed the Daphnet study in predicting FOG events, as the sensitivity of 77% is greater than the Daphnet reported sensitivity of 73%. The model also shows a fairly high accuracy of 87% in predicting FOG events. In addition to 5-fold CV, another method of cross validation was used to test the model called LOOCV (Leave-One-Out-Cross-Validation). LOOCV is a technique used for assessing the performance of an MLmodel, particularly in cases where the data sample size is relatively small. In LOOCV, the model is trained on all but one data point in the dataset, and then the performance of the model is evaluated on the one data point that was held out. This process is repeated for each data point in the dataset, and the performance of the model is averaged across all the held-out data

points.This method of cross validation more accurately represents the model's performance in the real world, where it is trained on the entire dataset and only makes predictions on a single data point. However, the method is also computationally expensive (since the model has to be tested on every datapoint) so was only used to evaluate the model.

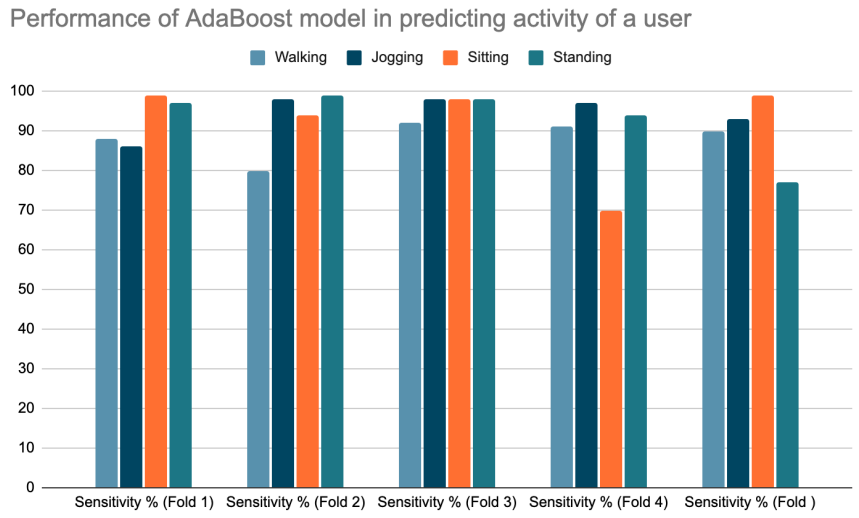
	Macro Average	Weighted Average
Accuracy	0.88	0.88
F1 Score	0.85	0.88
Sensitivity	0.85	0.88
Specificity	0.84	0.88

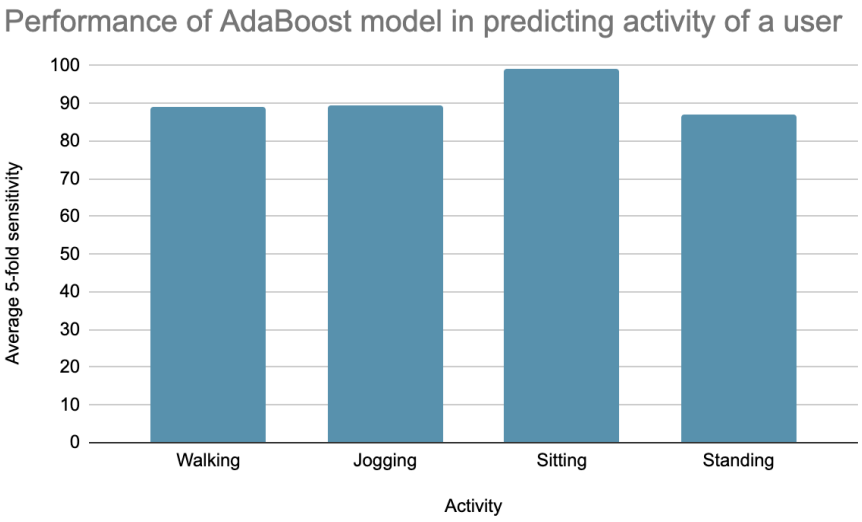
Table 2 and Figure 25: Model Performance



Upon LOOCV, the model performed well with a large improvement in sensitivity (88%) and an improvement in accuracy of 88%.

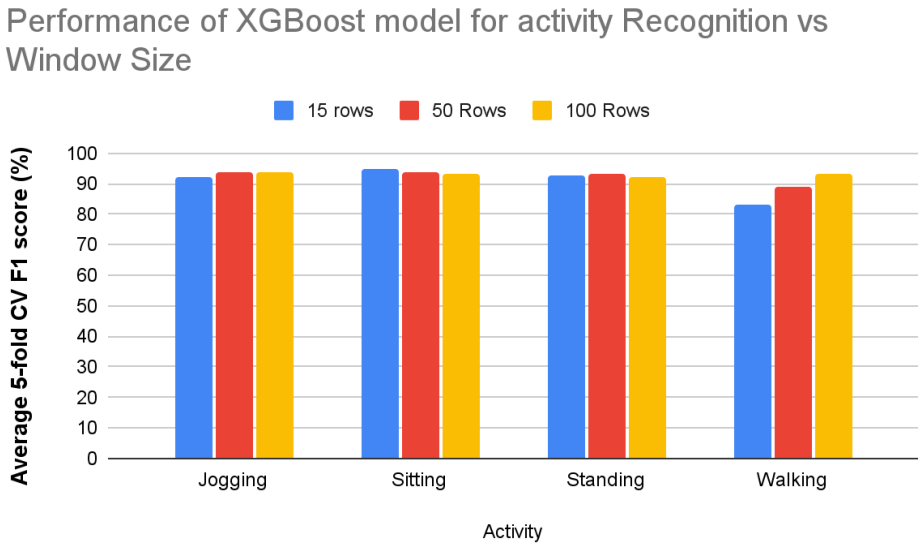
Activity Recognition Model





Figures 26 - 27

Figures 26 and 27 show the 5-fold cross validation results for the sensitivity of the XGBoost Activity Recognition model in distinguishing walking, jogging, sitting, and standing. The model showed an average 5-fold CV sensitivity of 88, 86, 99, 97 % for predicting walking, jogging, sitting, and standing respectively. This model was also tested at 3 different window lengths (15, 50, and 100) to identify the best window length. The average F1-score across 5-folds for the model in predicting each activity is shown in Figure 28.



	15 rows	50 Rows	100 Rows
Jogging	92.2	93.6	93.6
Sitting	95	94	93.2
Standing	92.6	93.4	92.2
Walking	83.4	89	93

Figure 28 and Table 3

Based on Figure 28, a higher window size clearly shows a higher F1 score when predicting walking. However, for other activities, window size seems to have a negligible effect on model performance. Table 3 shows that the model shows a high F1 score (>90%) for predicting all activities when trained on windows of 50 and 100 rows and a high F1 score in all activities except walking when trained on windows of size 15.

Conclusions and Discussions

This project successfully created a robust machine learning pipeline with an integrated and scalable iOS prototype application for activity recognition and Freezing of Gait (FOG) monitoring in Parkinson's Patients. Upon LOOCV and 5-fold cross validation, the AdaBoost machine learning model achieved an 87 and 88% accuracy respectively in classifying FOG events in the training dataset. The Activity Recognition Model with an XGBoost model framework achieved F1 scores of 93, 94, 93.4, and 89 in classifying jogging, sitting, standing, and walking respectively.

Further development of this app will benefit Parkinson's Patients by providing an accurate and reliable method of diagnosis via monitoring of Freezing of Gait, and will allow a user to keep track of daily FOG events to determine the severity of their disease symptoms. There are a few limitations with the project: 1) The use of a sliding window without overlapping windows may decrease model accuracy, since overlapping windows ensures that every subsequent row in the transformed dataset has some information from the data in the previous window as well. 2) The FOG model's accuracy can still be improved, and an active learning approach may be required for more accurate FOG detection for a specific user. 3) Only data from one subject from the original Daphnet dataset was used to train the FOG model, since the model performed poorly (< 70% accuracy) with other data or with combinations of subject data. 4) The use of smaller sliding windows (n <50) may improve app latency but may not always be able to capture the full duration of FOG events, which are usually much longer. 5) App development is still in a preliminary stage, and a UI is still yet to be created.

In the future, we plan on implementing an active learning approach where a user can manually input data on recorded FOG events to improve the model's predictive accuracy and to have the model learn the patterns of FOG in a specific user. In this study, only a baseline model to predict FOG was created and active learning may be required for more accurate detection of FOG. In addition, user testing of the app is in progress as of now, and real-life app testing data will be collected by our presentation day. Finally, the creation of an app user interface is still in progress.

Acknowledgements

We would like to thank Jeremey Jensen, our teacher, for supporting and guiding us through the project and our parents for their encouragement and support. We would also like to thank the Supercomputing Challenge Judges for giving us feedback during the midterm interview.

Bibliography

1. *Dataset*. WISDM Lab: Dataset. (n.d.). Retrieved April 4, 2023, from https://www.cis.fordham.edu/wisdm/dataset.php
2. Ensemble methods. Corporate Finance Institute. (2022, December 14). Retrieved April 4, 2023, from https://corporatefinanceinstitute.com/resources/data-science/ensemble-methods/
3. Fourier transform (ft). Questions and Answers in MRI. (n.d.). Retrieved April 4, 2023, from https://mriquestions.com/fourier-transform-ft.html
4. Kondo, Y., Mizuno, K., Bando, K., Suzuki, I., Nakamura, T., Hashide, S., Kadone, H., & Suzuki, K. (2022, April 29). Measurement accuracy of freezing of gait scoring based on videos. Frontiers. Retrieved April 4, 2023, from https://www.frontiersin.org/articles/10.3389/fnhum.2022.828355/full
5. Lutins, E. (2017, August 2). *Ensemble Methods in Machine Learning: What are they and why use them?* Medium. Retrieved April 4, 2023, from https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f
6. Mayo Foundation for Medical Education and Research. (2023, February 17). Parkinson's disease. Mayo Clinic. Retrieved April 4, 2023, from https://www.mayoclinic.org/diseases-conditions/parkinsons-disease/symptoms-causes/syc-20376055
7. Nabriya, P. (2021, July 12). *Feature engineering on time-series data*. Medium. Retrieved April 4, 2023, from

https://towardsdatascience.com/feature-engineering-on-time-series-data-transforming-signal-data-of-a-smartphone-accelerometer-for-72cbe34b8a60

8. Navlani, A. (2018, November 20). AdaBoost classifier algorithms using python Sklearn tutorial. DataCamp. Retrieved April 4, 2023, from https://www.datacamp.com/tutorial/adaboost-classifier-python
9. NHS. (n.d.). NHS choices. Retrieved April 4, 2023, from https://www.nhs.uk/conditions/parkinsons-disease/treatment/#:~:text=You%20may%20not%20need%20any,and%20your%20family%20or%20carers.
10. Productionizing distributed XGBoost to train deep tree models with large data sets at uber. Uber Blog. (2019, December 10). Retrieved April 4, 2023, from https://www.uber.com/blog/productionizing-distributed-xgboost/
11. Saini, A. (2023, March 3). *Master the adaboost algorithm: Guide to implementing & understanding adaboost*. Analytics Vidhya. Retrieved April 4, 2023, from https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/
12. UCI Machine Learning Repository: Daphnet freezing of Gait Data Set. (n.d.). Retrieved April 4, 2023, from https://archive.ics.uci.edu/ml/datasets/Daphnet+Freezing+of+Gait
13. Wikimedia Foundation. (2023, January 18). *Random Forest*. Wikipedia. Retrieved April 4, 2023, from https://en.wikipedia.org/wiki/Random_forest
14. Wikimedia Foundation. (2023, March 24). Fast fourier transform. Wikipedia. Retrieved April 4, 2023, from https://en.wikipedia.org/wiki/Fast_Fourier_transform

Mapping Anthropogenic Ocean Litter with an Autonomous Underwater Vehicle

New Mexico Supercomputing Challenge

April 3, 2023

Team 14

Los Alamos High School

Team Members:

Daniel Kim

Teacher:

Michela Ombelli

Project Mentor:

None

Mapping Anthropogenic Ocean Litter with an Autonomous Underwater Vehicle

Daniel Kim [a*]

daniel.kim@studentlaschools.net

[a)] Los Alamos High School, Los Alamos NM

EXECUTIVE SUMMARY

This project explores the possibilities of pairing an autonomous underwater vehicle (AUV) with a deep-learning computer vision model for marine debris mapping. A cost effective, 3D-printed AUV with a motorized ballast system was designed to collect underwater footage continuously at various depths for several weeks. A simulated underwater environment using hardware-in-the-loop (HIL) procedures was used to test and evaluate the AUV. A trash detection machine learning model was developed to analyze the footage for underwater litter. To assess the accuracy and capabilities of the trash detection model, footage from various underwater vehicles was compiled and run through the model, yielding five areas of highly concentrated ocean debris at depths of 500-800 meters below the surface. This study highlights how many pieces of marine debris - undetectable by satellite data – can be mapped and categorized with the proposed AUV and trash detection model.

KEYWORDS: Autonomous underwater vehicle; Hardware-in-the-loop; Tracking and mapping; Marine debris; Real-time object detector; Machine learning.

1. INTRODUCTION

The use of plastics in the past 65 years has significantly outpaced the use of all other materials. Of the 8.3 billion metric tons of plastic ever produced, only 2500 metric tons are currently in use (Geyer et al., 2017). These plastics are the most dominant type of litter found in our oceans and contribute to the 244,000 metric tons of marine debris (Parker, 2022).

This issue has already caused substantial ecological and economic problems. An estimated 845 million people are at risk of iron, zinc, or vitamin A deficiency because of declining fish populations (Golden et al., 2016) and global ecosystem delivery systems are projected to lose $500-$2500 billion in value of benefits (Beaumont et al., 2019) because of global plastic accumulations.

Currently, quantitative assessments of marine litter are primarily done through satellite imagery. For example, the MARLISAT project combines orbital imagery and machine learning algorithms to detect plastic along beaches and oceans (Petersen, 2022). Although this method has given a clear view on concentrations of surface-level plastic, about 40% of plastics in the ocean are less dense than seawater (Andrady, 2011), leaving them undetectable through satellite imagery.

One of the largest garbage patches in the world, the Great Pacific Garbage Patch, was mapped using a fleet of 18 vessels and 642 surface nets (Lebreton et al., 2018). This method created a comprehensive understanding of the largest garbage patch in the world but required vast resources and manpower for a single assessment. The ceaseless movement of ocean debris requires continuous observation, but the use of large resources for one measurement is not sustainable. Therefore, advanced marine debris detection systems with autonomous capabilities are needed for the effective mapping of marine debris.

The purpose of this study is to design and develop an autonomous underwater vehicle (AUV) and a trash detection machine learning (ML) model to assist in marine litter quantification. The AUV is capable of surveying and capturing footage of different marine environments at various depths for long distances, while gathering other seminal metrics such as temperature and salinity data. Unlike satellites, AUVs allow for metrics to be collected at various depths, and contrary to current surface net and trawl quantification methods, AUVs are cheap and require very little manpower. The ML model analyzes the post-deployment footage from the AUV for litter, which is used to build a comprehensive map of marine debris.

Using this method, the performance and capabilities of the engineered AUV are evaluated and open-sourced underwater video datasets are analyzed to create a model of ocean litter off the coast of Kamaishi, Japan. Using the ML model, the possibility of using AUVs to collect data on the scope of marine debris was assessed.

2. METHODOLOGY

2.1 Design of AUV

The AUV is modeled after an underwater glider - a robotic vehicle suited for data collection in remote locations at a low cost. Underwater gliders do not have propellers or engines; they utilize changes in buoyancy to move up and down which creates lift to propel the vehicle forward. This method allows for the glider to operate for days, weeks, or even months autonomously before being recovered for data.

The AUV was first designed in computer-aided design (CAD), and 3D-printed. Each component of the AUV was a separate, modular design, which was then assembled onto a steel rod backbone. The entire assembly then slid into a 102 mm×914 mm clear PVC tube and was sealed with an O-ring endcap. Each of the 3D-printed modules was manufactured in polylactic acid (PLA), with high infill to increase the density of the vehicle. Since the inner printed parts are independent from the outer PVC tube, none of the inner components required water proofing. By assembling the AUV primarily with 3D printed components, the design allowed for modularity and repeatable manufacturing of parts – all at a very low cost.

The AUV was assembled using the 3D printed parts and had a width of 10.16 centimeters, a wingspan of 0.72 meters, and a length of 1.08 meters. Initially, the mass of the vehicle was insufficient to sink, thus 3.63 kg of copper plated lead weights were added to the interior of the AUV to make it neutrally buoyant, giving it a final mass of 6.63 kg.

The AUV control board, the circuit board that operates the vehicle, was first designed, and developed in electronic computer-aided design, then outsourced for fabrication by JLCPCB. Once the bare printed circuit board (PCB) was fabricated, individual integrated circuits were soldered using reflow. The microcontroller used on the control board was the Teensy® 4.1 (Teensy 4.1, 2020), featuring the Arm® Cortex-M7 (Cortex-M7, 2014) at 528 MHz and 1024 KB of memory. The software that operated the control board was developed in C++ and a complementary graphical user interface (GUI) was developed in Typescript.

2.2 Trash Detection Model

The trash detection model was built on YOLOv5, an open-source ML framework. The dataset used to train the neural network was the Trash ICRA-19 dataset (Fulton, 2019), which contained 5700 annotated images of underwater trash. Training was performed on a Google Colab server with a NVIDIA Tesla T4 with 16GB of memory and 2560 CUDA cores. A total of four different neural networks were trained and evaluated to find the best-suited model for trash detection. Table 1 summarizes the four different models. Once the models were trained, validation and deployment of the models was performed on a local server featuring the NVIDIA 1650Ti with 4GB of memory and 896 CUDA cores.

Table 1. The four models trained for trash detection. The model that showed the highest accuracy is then used for trash detection.

Model	Parameters (millions)	FLOPs @ 640 pixels (billions)
YOLOv5n	1.9	4.5
YOLOv5s	7.2	16.5
YOLOv5m	46.5	49.0
YOLOV5l	86.7	109.1

3. RESULTS AND DISCUSSION

3.1 Component Design for AUV

The AUV consists of four key components: a ballast tank to control buoyancy, a microcontroller system to control and operate the vehicle, aluminum wings to push the vehicle forward when rising and sinking, and a camera module to collect video footage of the surroundings (Figure 1).

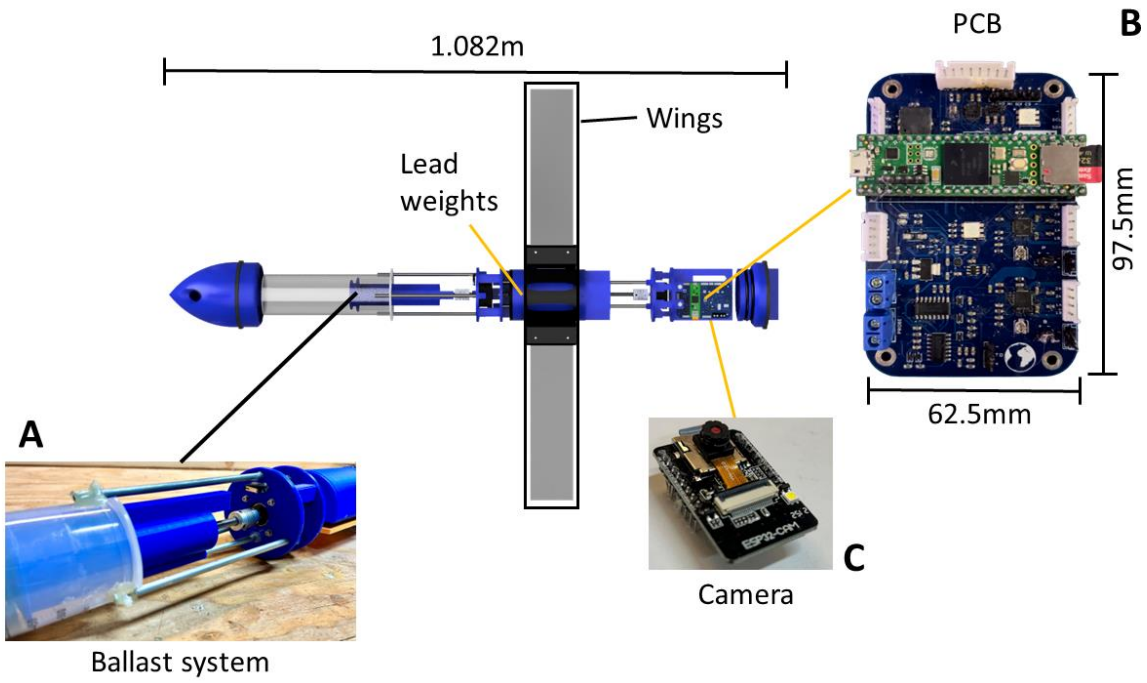


Figure 1. The AUV with its key components.

3.1.1 Ballast System

The buoyant force, F_B, exerted on the vehicle is equivalent to the weight of the volume of water displaced by the vehicle. If the buoyant force is greater than the weight of the vehicle, F_W, the vehicle floats. If the weight of the vehicle is greater than the buoyant force, the vehicle sinks. The AUV controls its weight by using a ballast tank. By pulling in water from its surroundings, the AUV increases in weight and can sink, and by pushing that water out, the AUV can then float.

For the AUV in this study, the ballast system utilizes a 550 mL syringe with a plunger driven by a stepper motor (Figure 2C). A threaded rod is connected to the stepper motor (Figure 2A) which pushes the syringe plunger in and out. The high accuracy of the stepper motor allows for fine tune adjustments to the amount of water in the tank to be controlled with very low power consumption. However, unlike servo motors, stepper motors provide no positional feedback, thus a limit switch is added to the end of the plunger's range of motion. By pulling the plunger until the limit switch is pushed (Figure 2B), the position of the plunger can be determined. The ballast assembly was able to pull in 450 mL of the 550 mL capacity, allowing for an additional 0.450 kg of water to be added to the AUV.

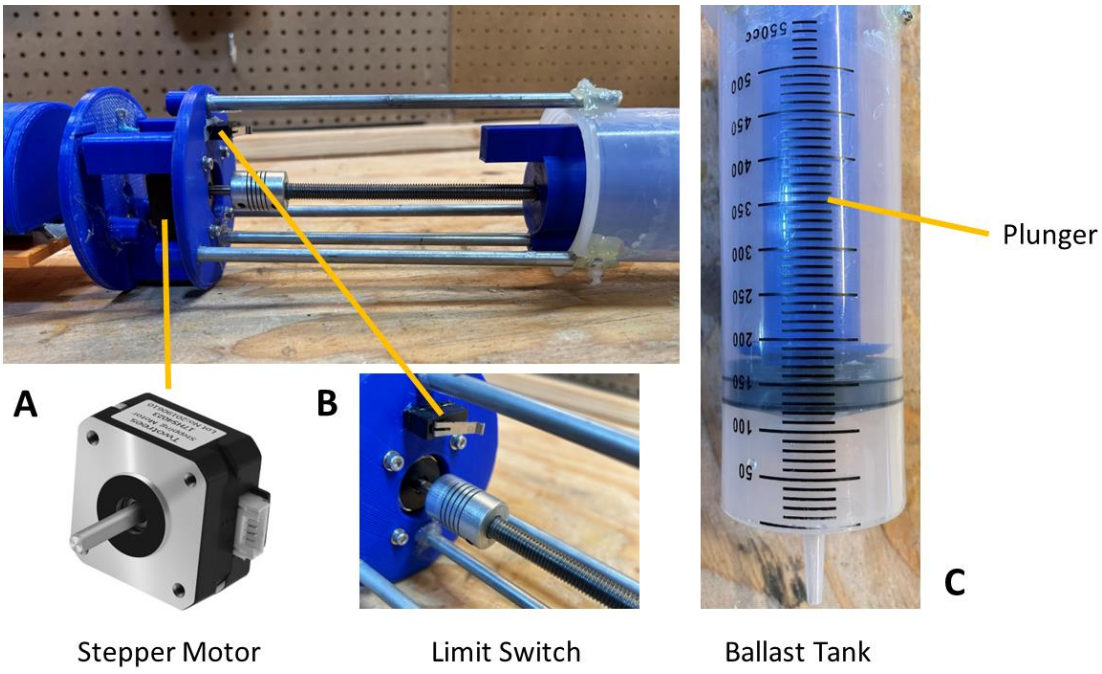


Figure 2. The AUV's ballast system.

To test the ballast system, the AUV was placed in a tub of water and was programmed to continuously fill and empty the ballast tank to sink and float. Initially, the AUV was too buoyant to sink, so an additional 308 g of lead was added to the rear of the AUV, for a final mass of 6.629 kg. Lead was chosen due to its high density and relatively low cost. The lead was copper plated to reduce the environmental and safety hazards associated with it. After the addition of the extra weight, the AUV was able to control its buoyancy with the ballast system and repeatedly float and sink (Figure 3).

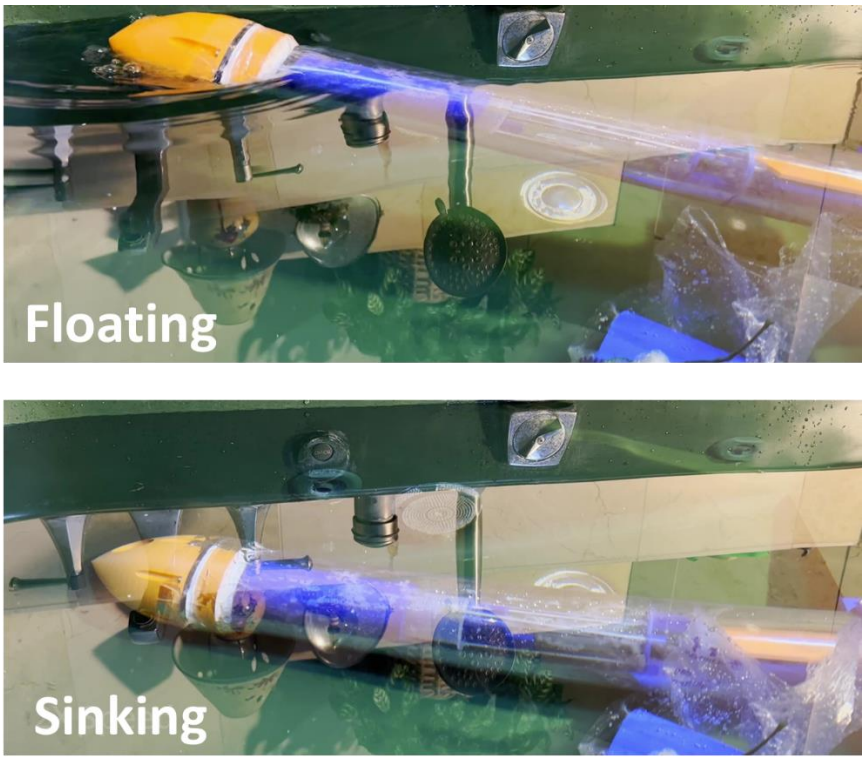


Figure 3. Field test of the AUV's ballast system.

3.1.2 AUV Control System

To read sensor data, control and navigate the AUV, and perform telemetry, a custom printed circuit board was designed and assembled (Figure 1B). At its core, the control system has a 528 MHz microcontroller unit that runs the AUV control software and controls all aspects of the vehicle.

To calculate the orientation and relative speed and position of the vehicle, the control board features an accelerometer, gyroscope, and magnetometer.

The board also includes a total dissolved solids (TDS) sensor to calculate external solute concentrations. The TDS sensor calculates the concentration of solutes within the AUV's environment by measuring the current across two electrodes providing 3 V AC power at 50 Hz. The excitation source is alternating instead of direct current to prevent the sensor from polarization by preventing the buildup of charged particles on the electrodes. Since the conductivity of water increases as temperature increases, a thermistor is installed next to the TDS probe to calculate a more accurate reading.

To collect data needed for marine litter mapping, a GPS and external pressure sensor are all connected to the board. These two sensors allow for the location and depth of any detected litter to be identified. All data collected from these sensors are then logged onto a microSD

card. The ballast tank is controlled through a stepper motor, which the control system operates using a stepper motor driver.

Since most radio frequencies greater than 1 MHz do not work at distances greater than 10 meters underwater (Qureshi, 2016), the control board allows for wired connections through the serial peripheral interface (SPI) or inter-integrated circuit (I2C) protocols. Although not implemented, connections to external modules on the surface (using tethered buoys) would allow for telemetry and GPS data to be transmitted and received. The SPI and I2C interfaces also allow for other additional components to be added to the AUV framework, such as auxiliary sensors or cameras.

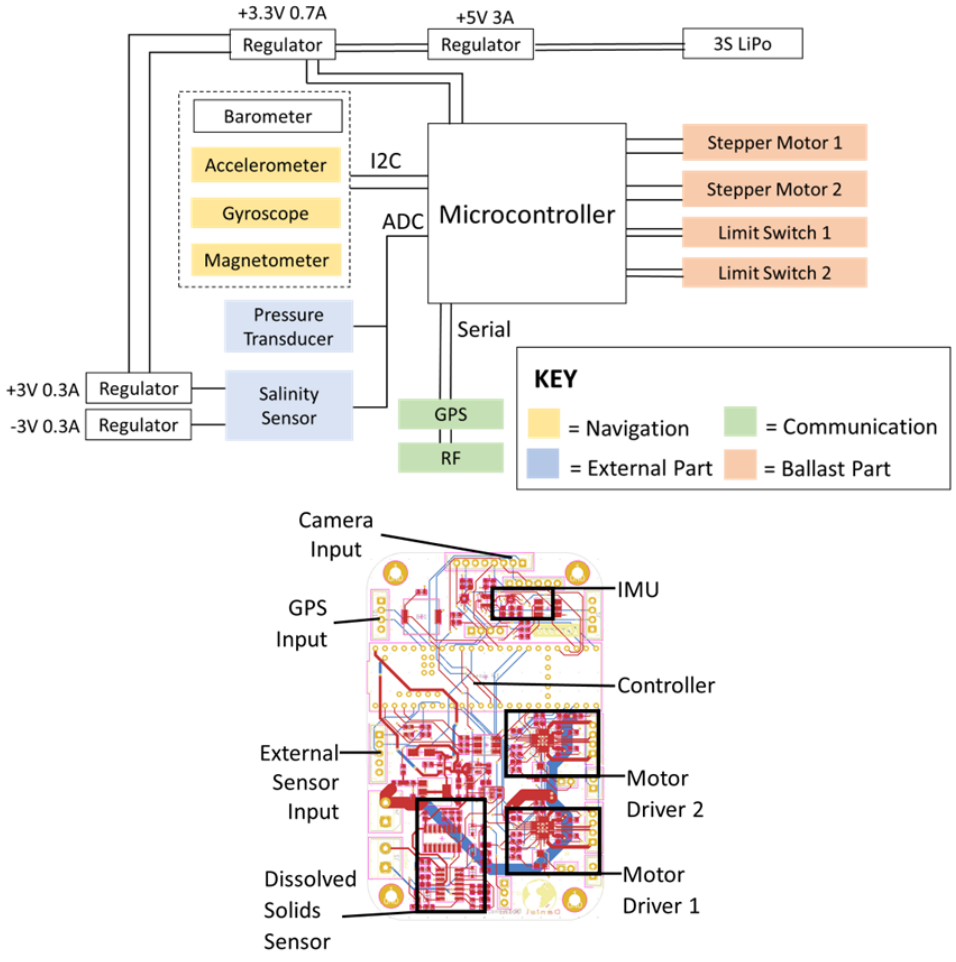


Figure 4. The components of the control board (top) and the labeled ECAD model of the control board (bottom)

3.1.3 Camera System

Although the main control board allows for an external camera module to be connected, image processing and logging are very CPU and memory intensive operations, thus a separate module was added to collect footage of the AUV's surroundings. The ESP32-S microcontroller with the OV2640 2-megapixel camera was chosen to collect footage of the AUV's surroundings (Figure

1C). The 32-bit ESP32 microcontroller runs at 240 MHz and features 2 cores and 520 KB of RAM, which is sufficient to capture video footage continuously and save to a microSD card. The small size of the module (27mm × 49mm) allows for multiple camera units to be setup within the AUV, which can provide multiple camera angles.

3.1.4 AUV Control Software

Although the control system and camera system provided the hardware to operate the AUV, respective software had to be developed to run the control system. The software controls everything from the stepper motors to drive the ballast tank to the data collection and logging. The AUV control software is written in C++ and had two key aspects: sensor data collection and data logging/telemetry.

3.1.4.1 Sensor Data Collection

The AUV had seven key sensors that collected data about the state of the vehicle. The TDS sensor measures the number of solutes in the environment in conjunction with an external temperature and pressure sensor, while an accelerometer, gyroscope, magnetometer, and GPS were used to measure orientation and provide an estimate of the AUV's position.

To calculate the external solute concentration with the TDS sensor based on temperature, the temperature compensated reading, ϕ_T must be first calculated:

$$\phi_T = \left(\frac{A_S * V_{REF}}{1023}\right) 1.0 + 0.02(T - 25) \quad (1)$$

where:

- A is the raw analog reading from the TDS sensor.
- V_{REF} is the reference voltage of the microcontroller.
- T is the temperature in degrees Celsius measured by the thermistor.

This is necessary because the conductivity increases as temperature increases. The solute concentration can then be calculated by

$$S_T = 0.5 * [133.42 * {\phi_T}^3 - 255.86 * {\phi_T}^2 + 857.39 * \phi_T] \quad (2)$$

where:

- S_T is the solute concentration in g/mL at temperature T

Temperature readings from the thermistor not only help provide a more accurate TDS measurement but can also be used to measure rising ocean temperatures throughout multiple AUV deployments. The temperature is first calculated by measuring the resistance of the thermistor by calculating the voltage drop:

$$R_T = (V_{REF} - V_{therm}) * R_k \div V_{therm} \quad (3)$$

where:

- R_T is the resistance of the thermistor in ohms at temperature T.
- V_{REF} is the reference voltage of the microcontroller.
- V_{therm} is the voltage measured across the resistor.
- R_k is the known resistance in the voltage divider.

The temperature can then be calculated with the Steinhart-Hart equation as:

$$T = A_c + B_c \ln(R_T) + C_c[\ln R_T]^3 - 273.15 \quad (4)$$

where:

- A_c, B_c, and C_c are the Steinhart-Hart coefficients varying on the type of thermistor used.
- T is the temperature measured in degrees Celsius.

External pressure readings provide accurate measurements of the depth of the vehicle. The pressure transducer outputs a simple analog output directly proportional to the pressure of the environment. The accelerometer, gyroscope, and magnetometer, (collectively referred to as the inertial measurement unit or IMU here) are each separate devices that provide readings to the microcontroller through the I2C communication protocol. For positioning, the AUV utilizes GPS data for absolute positioning and the IMU for localization. The GPS module provides horizontal positioning up to 2.5 m in accuracy at 10Hz while the IMU provides accurate horizontal positioning through dead reckoning. Using GPS, pressure measurements, and the IMU, the position of the AUV in all three dimensions can be estimated. The IMU is also used to calculate the orientation of the vehicle using the Madgwick algorithm (Madgwick, 2010). The Madgwick algorithm fuses accelerometer, gyroscope, and magnetometer readings to provide a low-drift estimation of the AUV's orientation.

3.1.4.2 Data logging and Telemetry

During deployment, data is serialized onto a microSD card in the JSON format. A converter was also developed in C++ to convert the JSON data into more easily readable formats, such as CSV. Data is transmitted to an external GUI through the universal asynchronous receiver-transmitter (UART) protocol. The GUI is built in Typescript-React and provides live data feeds from the sensors and allowed for control over the AUV's ballast system. The GUI allows for extensive control of the vehicle's ballast system and provided critical information such as vehicle orientation, external pressure, and battery voltage. Telemetry could be sent wirelessly through a tethered buoy but is sent directly through a wire directly connected to the control board during AUV testing.

3.2 Hardware-In-The-Loop

After designing and assembling the AUV and its respective software, the platform required extensive testing to evaluate its effectiveness. Unfortunately, many garbage patches exist in the

middle of the ocean and the means of testing this application is geographically restricted. To overcome this challenge, a method called hardware-in-the-loop (HIL) was adopted to test the AUV. HIL testing simulates reality by feeding real signals through the platform while test and design iteration take place. HIL allows for comprehensive testing of complex systems through many of possible scenarios without spending time and money associated with real-world tests (NI, 2022).

Data from the open-sourced Slocum glider from Teledyne Webb Research was fed through the software while simulating underwater conditions (Figure 5). The data from this dataset included 1.139 megabytes of latitude, longitude, depth, pressure, temperature, and salinity data collected from May 25, 2018, to July 16, 2019. Although the AUV continuously read from the onboard sensors, the readings from the sensors were overridden with data from the dataset. While the data was fed through the system, the AUV performed its normal operations, such as ballast tank control, telemetry, sensor data reading and logging, and underwater footage collection. These tests evaluated the functionality of the sensors and mechanical aspects of the AUV, while also testing the abilities of the software and control board to operate the AUV.

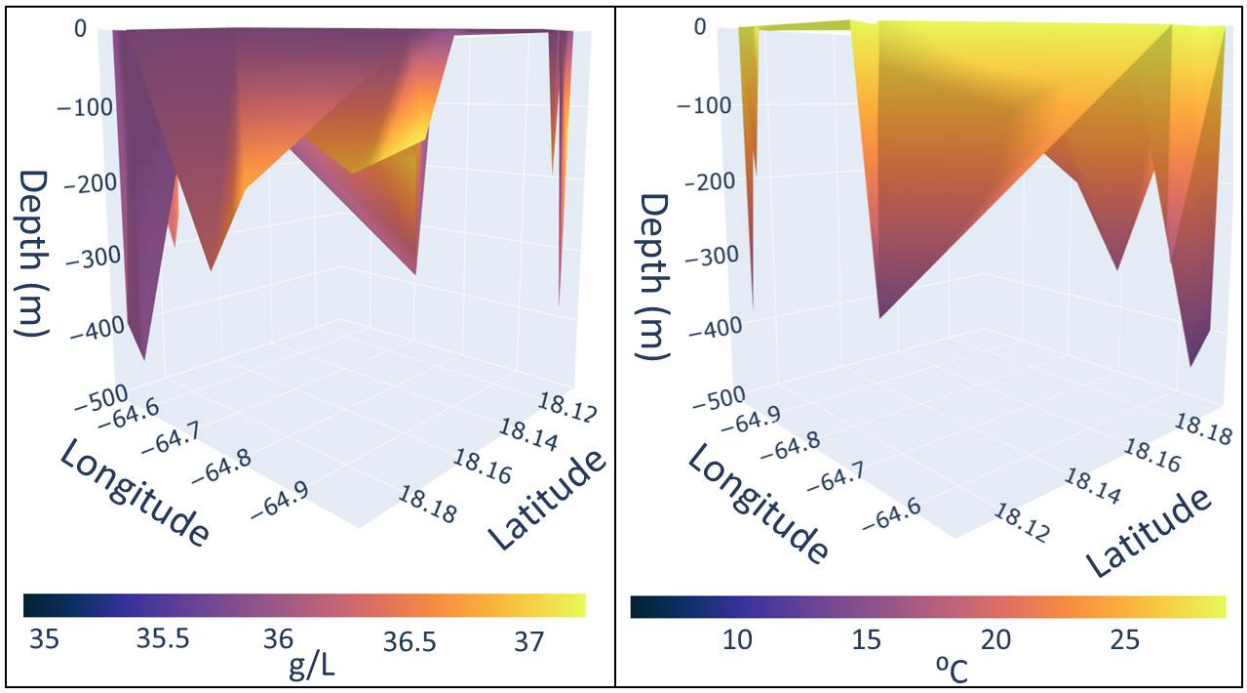


Figure 5. Salinity data from the HIL test in grams per liter (left) and the temperature data from the HIL test in degrees Celsius.

3.3 Trash Detection Model

The trash detection model locates and classifies litter in an image or video using a deep neural network. For each image or frame of a video, the model outputs a predicted bounding box surrounding the piece of litter. Once the AUV is recovered, the footage and sensor data are run through this model. If a piece of litter is detected, the model then extracts the respective depth

and GPS data, which can be used to build a comprehensive map of ocean litter. The trash detection model trained in this study only has one class – meaning that it can detect if there is trash or no trash within a certain image. Training was done on the Trash ICRA-19 Dataset (Fulton, 2019) which contained 5700 images of underwater trash, each of which were annotated with a ground truth bounding box which enclosed the litter within the image.

3.3.1 Training

Before the model could be deployed, it needed to first be trained on a large dataset. For training, four different convolutional neural networks (CNNs) were used: YOLOv5n, YOLOv5s, YOLOv5m, and YOLOv5l (Jocher, 2023). To evaluate the best model to use for trash detection, several metrics were measured throughout the model training. The model that exhibited the best metrics would be chosen as the final detection model.

In machine learning, true positives, TP, are predictions that the model made correctly. False positives, FP, are outcomes where the model incorrectly predicted that trash was detected. False negatives, FN, on the other hand, are instances where trash was present, but the model did not predict litter at that position.

Using these metrics, the precision and recall of the model can be calculated. Precision is defined as:

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

Precision gives the proportion of positive identifications that were correct. By contrast, recall gives the proportion of actual positives that were identified correctly, and is defined as:

$$Recall = \frac{TP}{TP+FN} \tag{6}$$

For each of the images the model inferences, it returns a confidence score which shows the probability of trash being within the bounding box. If the confidence score is greater than the confidence threshold, a value set by the user, the object is classified as trash. Thus, the confidence threshold affects the ratio of true positives, false positives, and false negatives, and therefore causes a tradeoff between precision and recall. Typically, a high-precision model will have a lower recall and vice versa. Therefore, to evaluate the model, a different metric, called mean average precision, mAP, must be used. mAP accounts for both precision, recall, and differing confidence thresholds giving an intuitive measure of the accuracy of a model. To calculate mean average precision, first, the average precision, AP, for each class of the model is calculated. AP is defined as:

$$\int_0^1 p(r)dr \quad (7)$$

where:

- $p(r)$ is the precision and recall function, giving us the precision of the model at a specific recall, r

Thus, a model that can detect 5 different categories (classes) will have 5 AP values. To calculate the mean average precision, the mean of the AP values is calculated.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \quad (8)$$

where:

- AP_k is the average precision of class k.
- n is the number of classes.

Since the trash detection model only has one class, the mean average precision of the model is equivalent to the average precision. Although mean average precision gives an overall quantification of the performance of the model based on precision and recall, another metric, loss, is also needed to measure the accuracy of the bounding boxes produced by the model. Loss consists of three parts: classification loss, localization loss and objectness loss (Liu, 2016). Localization loss measures differences between the predicted and ground truth bounding box coordinates, while confidence loss measures the probability of object presence or absence within a predicted bounding box. Classification loss is a measure of the model's ability to predict the correct class. However, since the trash detection model only has a single class, this metric is negligible. Localization loss, L, is defined as:

$$\gamma_c \sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{I}_{ij}^{obj}[(x_i + \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \gamma_c \sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{I}_{ij}^{obj}[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2] \quad (9)$$

where:

- γ_c is the scaling factor for localization loss.
- S^2 is the number of grid cells in the output feature map (the output of the convolutional layers).
- B is the number of bounding boxes predicted for every grid cell.
- $\mathbb{I}_{ij}^{obj}$ is the indicator variable that equals 0 unless the j-th bounding box in the i-th grid cell is responsible for detecting an object, in which case it equals 1.
- x_i and y_i are the coordinates of the center of the predicted bounding box.
- w_i and h_i represent the width and height of the predicted bounding box, respectively.
- $\hat{x}_i$ and $\hat{y}_i$ are the coordinates of the center of the ground-truth bounding box.

- $\hat{w}_i$ and $\hat{h}_i$ represent the width and height of the ground-truth bounding box, respectively.

If an object is detected within the box, the objectness loss O is:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \, (C_i - \hat{C}_i)^2 \tag{10}$$

If an object is not detected within the box, O is:

$$\gamma_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} \, (C_i - \hat{C}_i)^2 \tag{11}$$

where:

- S^2 is the number of grid cells in the output feature map (the output of the convolutional layers).
- B is the number of bounding boxes predicted for every grid cell.
- $\mathbb{I}_{ij}^{obj}$ is the indicator variable that equals 0 unless the j-th bounding box in the i-th grid cell is responsible for detecting an object, in which case it equals 1.
- $\mathbb{I}_{ij}^{noobj}$ is the complement to $\mathbb{I}_{ij}^{obj}$.
- C_i is the predicted objectness score for the j-th bounding box in the i-th grid cell.
- $\hat{C}_i$ is the ground-truth objectness label for the j-th bounding box in the i-th grid cell.
- γ_{noobj} is a scaling factor that controls the weight of the no object loss when detecting background.

The final loss of the single class model is the sum of the localization and objectness loss defined as:

$$\gamma_c \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} [(x_i + \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \gamma_c \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} [\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2] + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \, (C_i - \hat{C}_i)^2 + \gamma_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} \, (C_i - \hat{C}_i)^2 \tag{12}$$

The loss function was calculated on the training images and on a separate validation dataset that was not used during training. By comparing the training loss with the validation loss, the model can be evaluated for overfitting. In general, if the validation loss is greater than the training loss, the model is overfitting because it has learned to predict only the training data too closely.

In short, mean average precision measures the ability of a ML model to detect objects and loss measures the model's ability to minimize the difference between ground-truth and its predictions. Figure 6 and Table 2 show the different metrics of each of the four CNNs that were trained throughout the 200 epochs for which the model was trained. The YOLOv5l model exhibited the highest mean average precision and the lowest training and validation loss, while not showing significant signs of overfitting.

Table 2. Best and average mean average precision, training loss, and validation loss for each neural network model. The percent difference defines the difference between the best training and validation loss metric.

	YOLOv5s	**YOLOv5n**	**YOLOv5m**	**YOLOv5l**
mAP (best)	0.706	0.720	0.759	0.774
mAP (average)	0.548	0.554	0.612	0.616
Training loss (best)	0.105	0.105	0.105	0.105
Training loss (average)	0.059	0.059	0.056	0.056
Validation loss (best)	0.105	0.105	0.105	0.105
Validation loss (average)	0.059	0.059	0.056	0.056
Percent difference (best training vs. best validation)	0.000	0.000	0.002	0.000

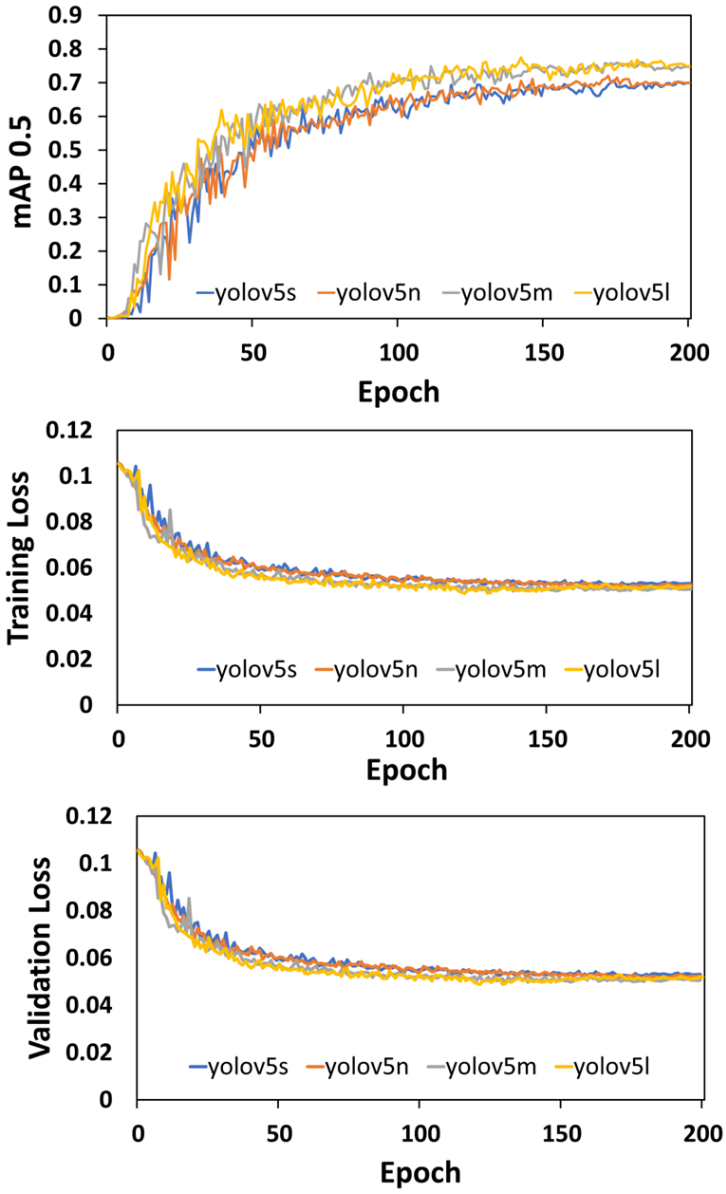


Figure 6. Mean average precision at intersection over union threshold 0.5 throughout the 200 epochs each model was trained on (top). Training loss of each model throughout 200 epochs (middle), and the validation loss of each model throughout 200 epochs (bottom). As the model trains for more epochs, it slowly improves by increasing the mean average precision and decreasing the loss.

3.3.2 Trash Detection Model Deployment

Since the AUV was tested using hardware-in-the-loop, it produced no real footage to run through the trash detection model. To evaluate the abilities of the model, open-source footage was collected from the Japan Agency of Marine-Earth Science and Technology (JAMSTEC)

HYPER-DOLPHIN submersible. The model detected a total of 174,734 litter objects at depths of 500-800 m below the surface. Objects were considered trash if the confidence score was greater than the confidence threshold, set at 0.4. Figure 7 shows the model inferencing a frame from the submersible and the detected objects from the model deployment.

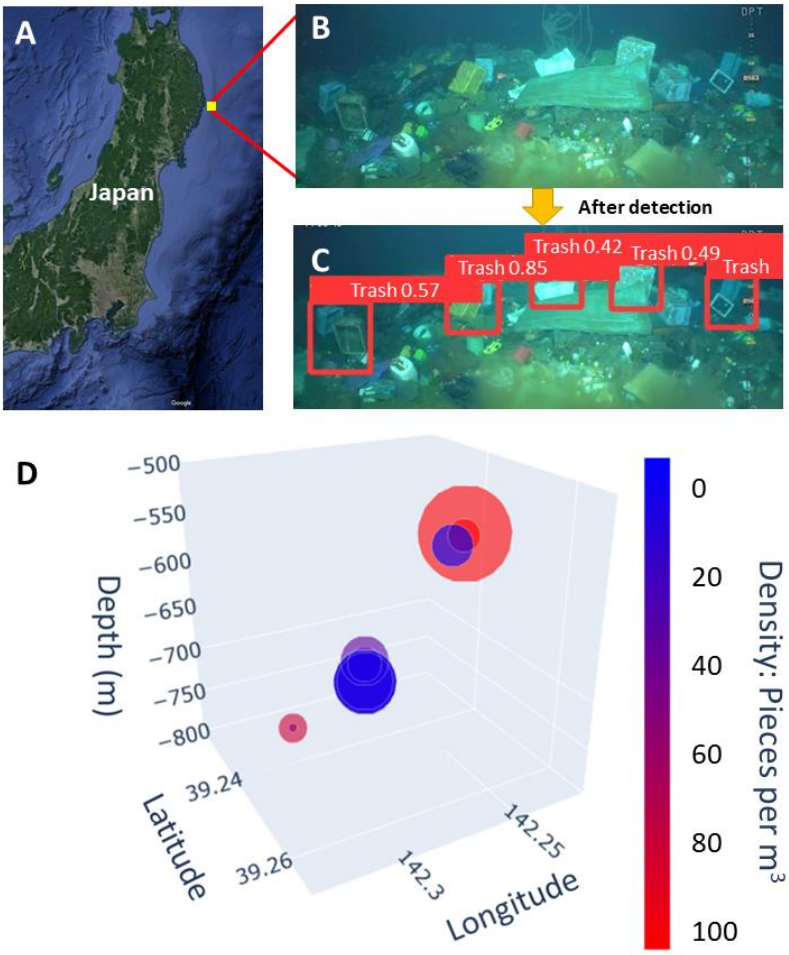


Figure 7. A: Location of the footage collected from the submersible off the coast of Kamaishi. B: Raw footage from the submersible. C: Inferenced image after being processed by the YOLOv5l trash detection model. D: Areas of debris detected by the model and the relative density of litter in each area represented by color.

4. CONCLUSIONS

This study shows an engineered AUV along with the training and deployment of a trash detection model. As shown by the HIL tests, the AUV can accurately read sensor data and process footage while navigating underwater, and the high precision of the computer vision model can seamlessly detect and analyze litter within underwater footage. The development of both a prototype AUV and novel trash detection computer model indicate that this system would be a viable method to quantify and map litter concentrations across our world's seas.

The AUV demonstrated an ability to collect sensor data and operate in underwater conditions, while the trash detection model found 174,000 debris objects underwater at depths undetectable to satellites and surface net analysis.

An exploration into replacements to the external parts of the AUV (the outer shell of the vehicle in contact with the environment) could yield better materials to handle salty ocean conditions and the use of more durable plastics for 3D printed parts could give the vehicle a more robust design. However, the biggest improvement is for more testing and evaluation to be done on the vehicle. Further tests in local bodies of water or even in the ocean could yield important flaws within the vehicle design. Furthermore, I hope to implement usage of the accelerometer onboard the AUV to track the forces of ocean currents to track the movement of debris, rather than just the position.

Currently, the trash detection model has only one classification level – whether trash is in the image or not. By using a more comprehensive dataset during training, the model would be able to classify different types of debris (e.g., plastic, metal, glass). Furthermore, by tuning hyperparameters within the existing model (number of epochs, learning rate, and model architecture), the mean average precision and loss of the model can be improved.

AUTHOR INFORMATION

Corresponding Author
Daniel Kim
Los Alamos High School
1300 Diamond Drive
Los Alamos, New Mexico 87544
Email: daniel.kim@studentlaschools.net

APPENDIX A

Symbol	Description
AUV	Autonomous Underwater Vehicle
CAD	Computer-aided-design
CNN	Convolutional Neural Network
ECAD	Electrical computer-aided-design
GUI	Graphical user interface
HITL	Hardware-in-the-loop
I2C	Inter-integrated circuit
IMU	Inertial measurement unit
ML	Machine Learning
PLA	Polylactic acid
PVC	Polyvinyl chloride
SPI	Serial peripheral interface
TDS	Total dissolved solids
UART	Universal asynchronous receiver-transmitter
A_c	Steinhart-Hart coefficient
AP	Average precision
AP_k	Average precision of class k
A_s	Analog reading
B	Number of bounding boxes predicted for every grid cell
B_c	Steinhart-Hart coefficient
C_c	Steinhart-Hart coefficient
C_i	Predicted objectness score for the j-th bounding box in the i-th grid cell
$\hat{C}_i$	Ground-truth objectness score for the j-th bounding box in the i-th grid cell
F_B	Buoyant force
FN	False negative
FP	False positive
F_W	Weight force
h_i	Height of the predicted bounding box
$\hat{h}_i$	Height of the ground-truth bounding box
$\mathbb{I}_{ij}^{obj}$	Indicator variable that equals 0 unless the j-th bounding box in the i-th grid is responsible for detecting an object, in which case it equals 1
$\mathbb{I}_{ij}^{noobj}$	Complement to $\mathbb{I}_{ij}^{obj}$
mAP	Mean average precision
n	Number of classes
$p(r)$	Precision value at recall level r
R_k	Known resistance in the voltage divider
R_T	Resistance at temperature T
S_T	Solute concentration in g/mL at temperature T
S^2	Number of grid cells in the output feature map

T	Temperature reading in degrees Celsius
TP	True Positive
V_{REF}	Microcontroller reference voltage
V_{therm}	Voltage measured across the thermistor
w_i	Width of the predicted bounding box
$\hat{w}_i$	Width of the ground-truth bounding box
x_i	X coordinate to the center of the predicted bounding box
$\hat{x}_i$	X coordinate to the center of the ground-truth bounding box
y_i	Y coordinate to the center of the predicted bounding box
$\hat{y}_i$	Y coordinate to the center of the ground-truth bounding box
ϕ_T	Temperature compensated TDS reading at temperature T
γ_c	Scaling factor for localization loss
γ_{noobj}	Scaling factor that controls the weight of the no-object loss

ACKNOWLEDGEMENT

I would like to give thanks to Dr. Robert Hermes for his help in acquiring lead weights used in this study. Thank you to my parents for overseeing the many tests and evaluations required in this study.

REFERENCES

Andrady, A. L. Microplastics in the marine environment. *Mar. Pollut. Bull.* [Online] 2011, *62* (8), 1596-1605. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S0025326X11003055?via%3Dihub (accessed Feb 15, 2023).

Beaumont, N. J.; Aanesen, M.; Austen, M. C.; Börger, T. et al. Global ecological, social and economic impacts of marine plastic. *Mar. Pollut. Bull.* [Online] 2019, *142,* 189-195. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S0025326X19302061?via%3Dihub (accessed Feb 15, 2023).

Cortex-M7. https://developer.arm.com/Processors/Cortex-M7 (accessed Feb 21, 2023).

Fulton, M.; Hong, J.; Islam, M.; Sattar, J. Robotic Detection of Marine Litter Using Deep Visual Detection Models. *IEEE* [Online] 2019, 5752-5758. https://ieeexplore.ieee.org/abstract/document/8793975 (accessed Feb 15, 2023).

Geyer, R.; Jambeck, J.; Law, K. Production, use, and fate of all plastics ever made. Science. [Online] 2017, 3 (7), https://www.science.org/doi/10.1126/sciadv.1700782 (accessed Feb 15, 2023).

Golden, C. D.; Allison, E. H.; Cheung, W. W.; Dey, M. M. et al. Nutrition: Fall in fish catch threatens human health. *Nature* [Online] 2016, 317-320. https://doi.org/10.1038/534317a (accessed Feb 15, 2023).

Jocher, G. *YOLOv5*. https://github.com/ultralytics/yolov5 (accessed Jul 20, 2022).

Lebreton, L.; Slat, B.; Ferrari, B.; Sainte-Rose, B, et al. Evidence that the Great Pacific Garbage Patch is rapidly accumulating plastic. *Sci. Rep.* [Online] 2018, *8*, https://doi.org/10.1038/s41598-018-22939-w (accessed Feb 15, 2023).

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C, et al. SSD: Single Shot MultiBox Detector. *LNIP*. [Online] 2016, *9905,* Springer. https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2#chapter-info (accessed Feb 15, 2023).

Madgwick, S. O.; Harrison; A. J.; Vaidyanathan, R. Estimation of IMU and MARG orientation using a gradient descent algorithm. IEEE [Online] 2011, 1-7. https://ieeexplore.ieee.org/document/5975346 (accessed Feb 15, 2023).

National Instruments. What Is Hardware-in-the-Loop?, 2002. https://www.ni.com/en/solutions/transportation/hardware-in-the-loop/what-is-hardware-in-the-loop-.html (accessed Jan 23, 2023).

Parker, L. Ocean Trash: 5.25 Trillion Pieces and Counting, but Big Questions Remain, National Geographic. https://education.nationalgeographic.org/resource/ocean-trash-525-trillion-pieces-and-counting-big-questions-remain (accessed Jan 23, 2023).

Petersen, C. C. Satellites are Tracking Rivers of Garbage Flowing Across the Oceans, 2022. Universe Today. https://www.universetoday.com/157156/satellites-are-tracking-rivers-of-garbage-flowing-across-the-oceans/ (accessed Jan 23, 2023).

Qureshi, U. M.; Shaikh, F. K.; Aziz, Z.; Shah, S, et al. RF Path and Absorption Loss Estimation for Underwater Wireless Sensor Networks in Different Water Environments. Sensors [Online] 2016, 16 (6), 890. National Library of Medicine. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4934316/ (accessed Feb 15, 2023).

Teensy® 4.1 Development Board. https://www.pjrc.com/store/teensy41.html (accessed Feb 21, 2023).

SINGS: A Simple Interactive *N*-body Gravitation Simulator

New Mexico

Supercomputing Challenge

Final Report

April 5, 2023

Team 6

Sandia High School

Team Members

Tristan Eggenberger

Teacher & Project Mentor

Dr. Bradley Knockel

Table of Contents

Abstract & Executive Summary

In this paper we describe an astrophysical simulation code **SINGS**, the **S**imple **I**nteractive ***N***-body **G**ravitation **S**imulator, which has been designed to accurately model a wide range of astrophysical systems, ranging from simulations of planetary motion to cosmological simulations of structure formation. SINGS provides support for both collisionless and collisional particles, the latter of which employ impulse-based collision methods. SINGS provides various parallel force code and integration schemes, allowing for a high degree of customisability. In addition to the integration and force codes provided by SINGS, the code also provides a rudimentary programmatic framework for the creation of simulation scenarios, which can then be compiled to a simulation snapshot. These simulations can also be diagnosed at the moment of a single snapshot. The simulation snapshots may also be independently analysed by the user, employing a simple struct based layout suitable for analysis. In this report, we detail these various components of SINGS and evaluate their performance and accuracy.

SINGS overall is still in its infancy as a tool. The program we detail, while indeed powerful, should not be seen as an alternative or improvement upon existing mainstream N-body codes like AREPO (Springel, 2010) or GADGET-4 (Springel, Pakmor, et al., 2021). Our code uses OpenMP for parallelisation, which is unable to harness the full power of supercomputers running node-based architectures like more advanced codes. To do so would entail the use of OpenMPI and a radical change in the architecture of the program. Another feature missing from SINGS is a parallel Cloud-In-Cell (CIC) Particle-Mesh (PM) code utilising Fast Fourier Transforms (FFTs) to compute the gravitational potential across the whole simulation, an additional order $\mathcal{O}(N \log N)$ model to complement Barnes-Hut. However, implementing the model proved difficult given the existing SINGS architecture, which also presented significant difficulties in parallelising the Cooley-Tukey algorithm used for the FFTs. This meant that by the Supercomputing Challenge deadline we could not finish a complete implementation. As a result, the version of SINGS we present does not include the CIC PM code. There are a myriad of other features, such as the implementation of a more realistic collisional gas modeling code (such as SPH) and the inclusion of a Hubble parameter to support analysis of ΛCDM cosmologies, that we could not implement which we discuss in Section 7. As SINGS is intended to be open source on release under GPLv2, we invite others to collaborate in fleshing out the software. Overall though, SINGS in its current form is a fully functional 3 dimensional N-body code with impressive capabilities on supported hardware, with large simulations with particle counts in excess of 1,000,000 having been successfully demonstrated.

1 Introduction

Simulations of astrophysical simulations have proved to be an invaluable tool in the various realms of astrophysics. Numerical simulations have been used to understand the formation of planetary systems, the interactions between galaxies and galaxy clusters, and the large scale structural evolution of the universe. The rapid improvements in computer processing capabilities and the development of faster, more elaborate calculation schemes has allowed for exceptionally complex astrophysical simulations.

Prior to the advent of astrophysical simulations, study of large stellar objects like globular clusters, galaxies, and galaxy clusters were limited to the static frames viewable in the night sky. While information about the expansion of the Universe (Hubble, 1929), and to some extent, dark matter (Zwicky, 1933), questions regarding the nature of galaxy collisions and structure formation, among others, were impractical to be put to numerical analysis given the $\mathcal{O}(N^2)$ order of the computations involved.

Even some of the earliest and rudimentary astrophysical simulations like those of Holmberg (1941) and White (1976) demonstrated their power as a tool to validate and understand the implications of various astrophysical models. The former simulated the collision between two galaxies of stars by use of a clever setup involving lightbulbs in place for particles, the nature of which allowed the lowering of the calculation order from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. Through manual computation, the dynamics of galaxy collisions was modeled for the first time, the spiral structures of the two galaxies being torn apart in the interaction, as many particles were flung on escape trajectories. The latter was a computational simulation, thus able to overcome the $\mathcal{O}(N^2)$ order of the direct method, and demonstrated the evolution of galaxy clusters approximated with 700 particles evolving under cosmic expansion.

The introduction of more advanced force calculation techniques such as the so-called tree (Appel, 1981) and mesh codes (Eastwood and Roger Williams Hockney, 1974) with order $\mathcal{O}(N \log N)$ allowed for the investigation of even more complex astrophysical phenomena. This progress culminated in the Millenium Run (Springel, White, et al., 2005) wherein 2160^3 particles were simulated in a $(500\ \text{Mpc})^3$ cube, in effect modelling the evolution of the universe from shortly after the big bang to the present. Today, even more ambitious cosmological simulations like AbacusSummit (Maksimova et al., 2021) simulate simulation spaces hundreds of mega-parsecs across, with trillions of particles. These simulations provide key insight into the evolution of the early universe, regions with high redshift $z > 10$ that are not easily accessible to direct observation.

With a clear motivation for developing these codes, we now present our work in creating the code which can run simulations of scientifically useful resolution and magnitude: SINGS

2 General Physics

2.1 Formulation of Gravity

The general formula we'll use for calculating the forces between two particles i and j we'll derive from Newton's law of Universal Gravitation

$$F = G\frac{m_i m_j}{r_{ji}^2} \tag{1}$$

where F is the magnitude of the force between two particles i and j, G is the gravitational constant, m_i and m_j are the masses of the respective particles, and r_{ji} is their common distance. This formula can itself be derived as the derivative with respect to position of the gravitational potential energy U

$$U = -G\frac{m_i m_j}{r_{ji}} \tag{2}$$

Our first point of modification will come from the introduction of a softening parameter ε. Close range interactions will be prone to gross violation of the conservation of energy, as $r_{ij} \to 0$, the computed forces will diverge. While ε will reduce the accuracy of the simulation, for appropriate values this error is both negligble on large scales and increases the likelihood of energy preserving interactions. We can introduce ε into U as such,

$$U = -G\frac{m_i m_j}{\sqrt{r_{ji}^2 + \varepsilon^2}} \tag{3}$$

and taking $\dfrac{d}{dr}[U]$ to get F,

$$F = G r_{ji}\frac{m_i m_j}{(r_{ji}^2 + \varepsilon^2)^{\frac{3}{2}}} \tag{4}$$

What we've just described is an implementation of Plummer softening (Dyer and Ip, 1993), and this will serve as our generic force calculation formula when calculating the direct forces between two particles. For the general force calculation methods we'll discuss later we'll still use this Plummer softened force model, unless we have $\varepsilon = 0$, in which case we'll use (1). This is because both are essentially computationally equivalent in featuring a single square root.

We also can analyse the relative error δ between the two predicted F at a constant r_{ji}. Defining $\delta(\varepsilon)$ by

$$\delta(\varepsilon) = \left|\frac{F_{\text{plummer}} - F_{\text{actual}}}{F_{\text{actual}}}\right| \tag{5}$$

and considering that $F_{\text{plummer}} < F_{\text{actual}}$ for all $\varepsilon \neq 0$ we can remove the absolute value by negating the expression, expanding and simplifying to produce

$$\delta(\varepsilon) = 1 - \frac{r_{ji}^3}{(r_{ji}^2 + \varepsilon^2)^{\frac{3}{2}}} \tag{6}$$

The relationship between the error, force computed by (4), and the actual force from (1) is shown in Figure 1.

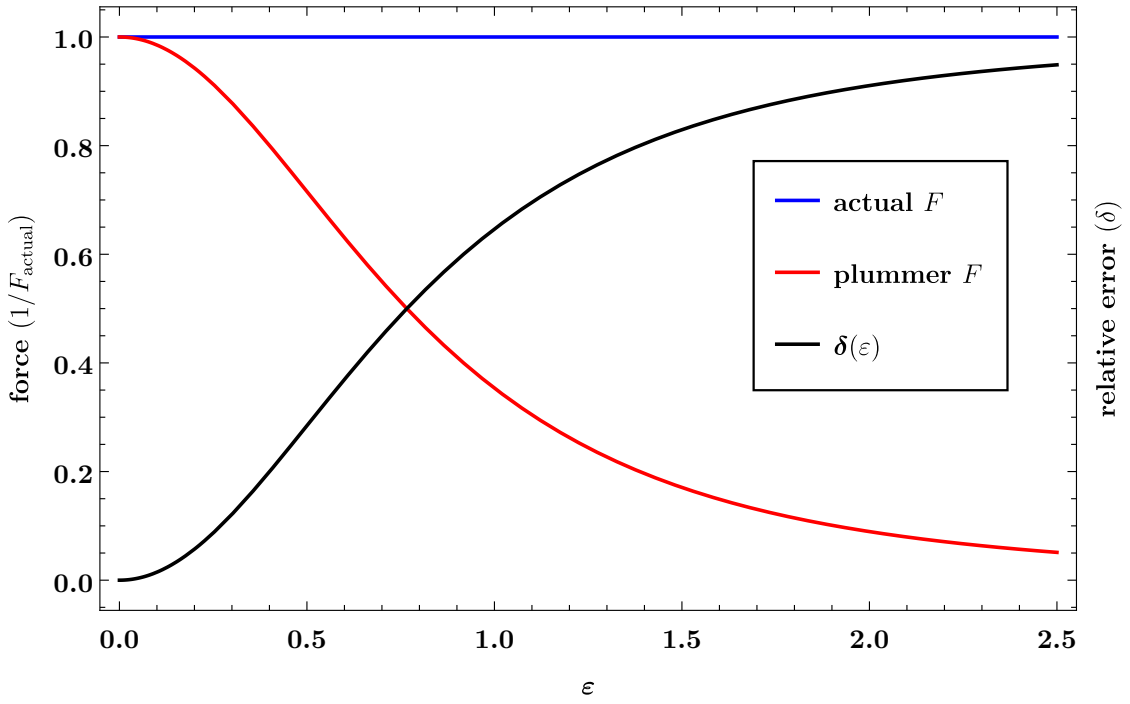


Figure 1: A plot illustrating how the force computed by the Plummer model and Newton's law, along with the the relative error ε, using absolute units where $r_{ji} = m_i = m_j = G = F_{\text{actual}} = 1$

2.2 Modelling Collisions

In modelling the collisions between two objects, we make use of an impulse based model. That is, when two collisional particles i and j have collided, we compute an impulse on each of the particles J along the collision normal vector $\hat{\mathbf{n}}$, which we'll consider to be the collisional normal vector on i from j. If we consider the change in the momentum for the

two particles, we can set up a simple system

$$\Delta\vec{\mathbf{P}}_i = m_i\vec{\mathbf{v}}'_i - m_i\vec{\mathbf{v}}_i \tag{7a}$$

$$\Delta\vec{\mathbf{P}}_j = m_j\vec{\mathbf{v}}'_j - m_j\vec{\mathbf{v}}_j \tag{7b}$$

Given that J is going to conserve the momentum of the particles, we can modify (7) to yield

$$J\hat{\mathbf{n}} = m_i\vec{\mathbf{v}}'_i - m_i\vec{\mathbf{v}}_i \tag{8a}$$

$$-J\hat{\mathbf{n}} = m_j\vec{\mathbf{v}}'_j - m_j\vec{\mathbf{v}}_j \tag{8b}$$

or, rearranging and solving for $\vec{\mathbf{v}}'_i$ and $\vec{\mathbf{v}}'_j$

$$\vec{\mathbf{v}}'_i = \vec{\mathbf{v}}_i + \frac{J\hat{\mathbf{n}}}{m_i} \tag{9a}$$

$$\vec{\mathbf{v}}'_j = \vec{\mathbf{v}}_j - \frac{J\hat{\mathbf{n}}}{m_j} \tag{9b}$$

Assuming a perfectly elastic collision (that is, one with a coefficient of restitution $\epsilon = 1$), we can produce the equation

$$\vec{\mathbf{v}}'_{\text{rel}} \cdot \hat{\mathbf{n}} = -\vec{\mathbf{v}}_{\text{rel}} \cdot \hat{\mathbf{n}} \tag{10}$$

where $\vec{\mathbf{v}}'_{\text{rel}}$ and $\vec{\mathbf{v}}_{\text{rel}}$ is the relative velocity vector between i and j from i after and before the collision, respectively. The expression $\vec{\mathbf{v}}'_{\text{rel}} \cdot \hat{\mathbf{n}}$ may also be expressed by

$$\vec{\mathbf{v}}'_{\text{rel}} \cdot \hat{\mathbf{n}} = (\vec{\mathbf{v}}'_i - \vec{\mathbf{v}}'_j) \cdot \hat{\mathbf{n}} \tag{11}$$

Substituting in the values for $\vec{\mathbf{v}}'_i$ and $\vec{\mathbf{v}}'_j$ from (9) and expressing $\vec{\mathbf{v}}'_{\text{rel}}$ in terms of $\vec{\mathbf{v}}_{\text{rel}}$ from (10) and simplifying produces

$$-\vec{\mathbf{v}}_{\text{rel}} \cdot \hat{\mathbf{n}} = \vec{\mathbf{v}}_{\text{rel}} \cdot \hat{\mathbf{n}} + J\hat{\mathbf{n}}\left(\frac{1}{m_i} + \frac{1}{m_j}\right) \cdot \hat{\mathbf{n}} \tag{12}$$

along with multiple regions and solving for our impulse

$$J = -2\frac{\vec{\mathbf{v}}_{\text{rel}} \cdot \hat{\mathbf{n}}}{\frac{1}{m_i} + \frac{1}{m_j}} \tag{13}$$

This equation for our impulse magnitude we can now finally use to derive new velocity

vectors for i and j using the formulas we set up in (8)

$$\vec{\mathbf{v}}_i' = -2\frac{(\vec{\mathbf{v}}_i - \vec{\mathbf{v}}_j)}{1 + \frac{m_i}{m_j}} + \vec{\mathbf{v}}_i \tag{14a}$$

$$\vec{\mathbf{v}}_j' = 2\frac{(\vec{\mathbf{v}}_i - \vec{\mathbf{v}}_j)}{1 + \frac{m_j}{m_i}} + \vec{\mathbf{v}}_j \tag{14b}$$

Notice that the collision normal vector $\hat{\mathbf{n}}$ is not present in our final equation. The final order of business is to tackle when collisions themselves happen. For that we use the same octree that we describe in 3.2. We search up the tree starting from a given particle and search up the tree a number of layers n, and if the criterion $r < \varepsilon$ is satisfied—where r is the distance between two particles in the neighboring nodes and ε is the plummer softening parameter from 2.1—we compute the new collision velocities and run a single timestep to update their position until they are no longer colliding with any particles.

3 Force Calculation

3.1 Direct Method

With the general form for the magnitude of forces between two particle pairs, we can apply a force vector $\vec{\mathbf{F}}_{ji}$ to i, that is the force vector on i from j, by multiplying our force magnitude by the normal vector which points from i to j

$$\vec{\mathbf{F}}_{ji} = Gr_{ji}\frac{m_i m_j}{(r_{ji}^2 + \varepsilon^2)^{\frac{3}{2}}}\frac{\vec{\mathbf{r}}_j - \vec{\mathbf{r}}_i}{|\vec{\mathbf{r}}_j - \vec{\mathbf{r}}_i|} \tag{15}$$

where $\vec{\mathbf{r}}_j$ and $\vec{\mathbf{r}}_i$ are the position vectors of j and i respectively. Applying this same step to (1) and noting that $r_{ji} = |\vec{\mathbf{r}}_j - \vec{\mathbf{r}}_i|$, we can produce the following softened and unsoftened force equations:

$$\vec{\mathbf{F}}_{ji} = G\frac{m_i m_j}{(r_{ji}^2 + \varepsilon^2)^{\frac{3}{2}}}(\vec{\mathbf{r}}_j - \vec{\mathbf{r}}_i) \tag{16a}$$

$$\vec{\mathbf{F}}_{ji} = G\frac{m_i m_j}{r_{ji}^3}(\vec{\mathbf{r}}_j - \vec{\mathbf{r}}_i) \tag{16b}$$

Now that we have formulae for computing the force vector applied to i from j, we can

use Newton's second law to derive the acceleration vector $\vec{\mathbf{a}}_i$

$$\vec{\mathbf{a}}_i = \frac{\sum_{j \neq i}^{N} \vec{\mathbf{F}}_{ji}}{m_i} \tag{17}$$

Inserting in (16a) and (16b) yields the following set of equations

$$\vec{\mathbf{a}}_i = \sum_{j \neq i}^{N} G \frac{m_j}{(r_{ji}^2 + \varepsilon^2)^{\frac{3}{2}}} (\vec{\mathbf{r}}_j - \vec{\mathbf{r}}_i) \tag{18a}$$

$$\vec{\mathbf{a}}_i = \sum_{j \neq i}^{N} G \frac{m_j}{r_{ji}^3} (\vec{\mathbf{r}}_j - \vec{\mathbf{r}}_i) \tag{18b}$$

where (18b) provides an exact $\vec{\mathbf{a}}_i$ and (18a) a softened one. These two equations are the backbone of the direct method implementation in SINGS, which we also use as a reference to compare the force calculations provided by other methods, particularly Equation (18a) as it exactly gives particle accelerations. The chief drawback of the direct method is that computations scale in order $\mathcal{O}(N^2)$, which for simulations with large N necessitates the use of other force codes.

3.2 Barnes-Hut

The algorithm described by Barnes and Hut (1986) (hereafter Barnes-Hut) is a hierarchical tree algorithm that scales with order $\mathcal{O}(N \log N)$. The Barnes-Hut tree is constructed by recursively partitioning the simulation space into a sequence of cubes, with each cube containing 8 child cubes representing an octant of the parent cube. These cubes form an octree structure, where each cube is a node in the octree. We construct the octree starting with one node representing the whole simulation space, and then partition it in such a way that each node either 0 or 1 particle, or is itself a parent to more nodes. We call nodes of the first variety external nodes, and the second internal nodes. The internal nodes store the center of mass and total mass information of all their internal particles, such that they can be used for approximate force calculation. A diagram illustrating the construction of a Barnes-Hut tree is shown in Figure 2.

We then calculate the force on a particle by walking the tree and summing all the approximate forces. In traversing the tree, we approximate a node as a particle if the relation

$$\frac{l}{r} < \theta \tag{19}$$

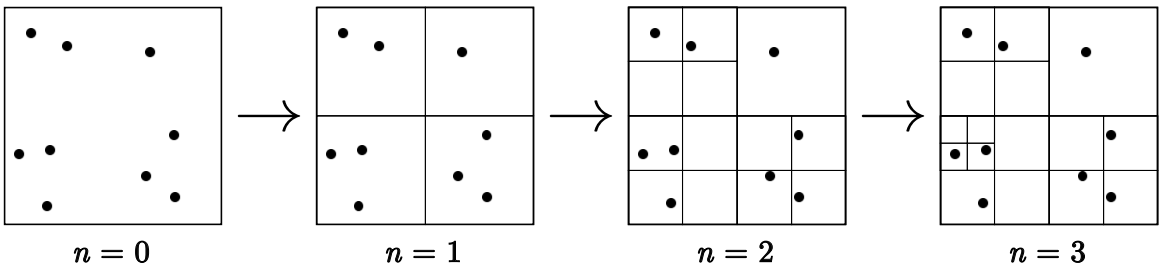


Figure 2: An illustration of a 2 dimensional Barnes-Hut tree down to various depths n. From the base layer, the space is subdivided until all particles occupy external nodes.

is met, where l is the length of the node, r is the distance from the particle to the node, and θ is a tuneable accuracy parameter. If a node doesn't satisfy (19), we expand the node, testing on its child nodes. If we end up on an external node with a particle we do a direct particle-particle force calculation before traversing back up through the tree. These steps are repeated until all internal nodes satisfying (19) and other external nodes are visited.

Larger values of θ will result in larger approximations and thus larger errors, but will also result in reduced compute times. As $\theta \to 0$, computations tend to grow with order $\mathcal{O}(N^2)$, and the computations become exactly those of the direct method at $\theta = 0$. The force calculation errors of Barnes-Hut for various values of θ is shown in Figure 3. as computed across various SINGS simulations.

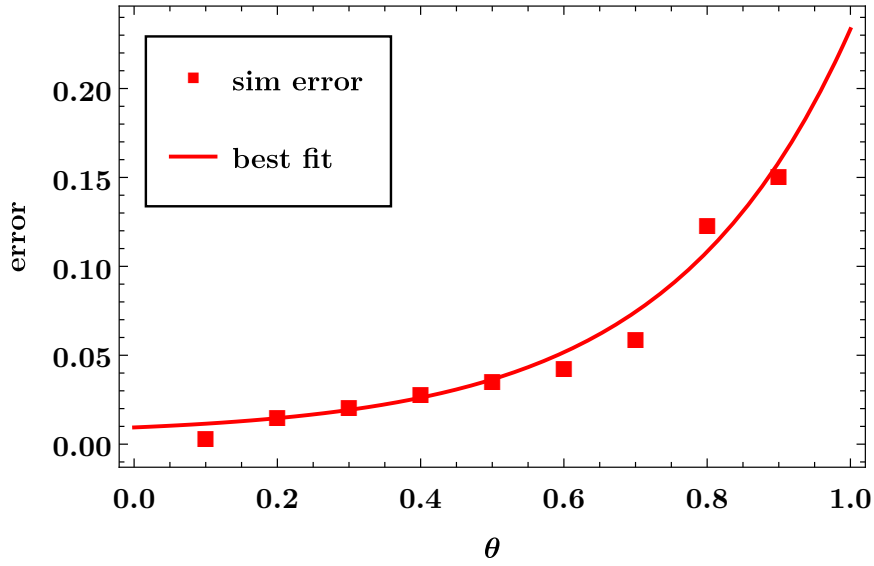


Figure 3: The average error in computed forces for various θ compared against the direct method across 100 sims/θ, given by $\mathbf{error}(\theta) = \dfrac{|\vec{\mathbf{F}}_{\mathrm{BH}(\theta)} - \vec{\mathbf{F}}_{\mathrm{Direct}}|}{|\vec{\mathbf{F}}_{\mathrm{Direct}}|}$.

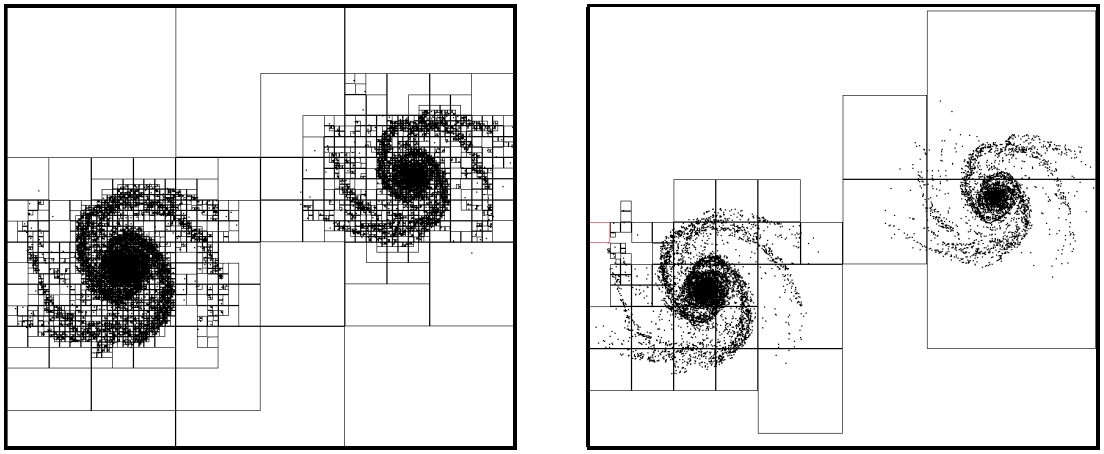


Figure 4: A 2 dimensional SINGS 1 simulation of 12,000 particles, modelling the collision between two galaxies. The frame on the left features the Barnes-Hut tree, the one on the right, the nodes used in calculating the force on a particle on the far left.

These errors though are more than manageable for carefully chosen values of θ, and allow us to simulate far larger and more complex systems. Figure 4 features frames from the 2 dimensional SINGS 1 code illustrating the massive reduction in computations enabled by the Barnes-Hut method.

4 Integration

In this section, we discuss various methods of numerical integration for updating particle positions and velocities given the acceleration vectors generated by our force codes we describe in Section 3. Two of the methods we describe are energy-conserving under Hamiltonian mechanics, that is, they are symplectic integrators. The others are general integration schemes which can still maintain high order accuracy for certain choices of timesteps Δt. Regarding timesteps, we use a static timestep as opposed to adaptive, particle-specific timesteps for the purpose of simplicity. Adaptive timesteps allow various particles to take on lower or higher accuracy timesteps appropriately, which generally results in higher accuracy simulations (Aarseth and Hoyle, 1963). However, the complexity they introduce resulted in us sticking to fixed particle time steps.

4.1 Euler Methods

The two Euler methods are the simplest methods we include in SINGS for the purpose of numerical integration. We include a second order version of the standard Euler method,

where new velocities and positions are derived from

$$\vec{\mathbf{r}}_{i+1} = \vec{\mathbf{r}}_i + \vec{\mathbf{v}}_i \Delta t \tag{20a}$$

$$\vec{\mathbf{v}}_{i+1} = \vec{\mathbf{v}}_i + \vec{\mathbf{a}}_i \Delta t \tag{20b}$$

where $\vec{\mathbf{r}}_i$, $\vec{\mathbf{v}}_i$, and $\vec{\mathbf{a}}_i$ are the position, velocity, and acceleration vectors of a particle at a timestep i respectively. This method is non-symplectic and has first order error. Given the low order of the error and its non-symplectic nature, this is generally not recommmended as an integrator except for stable simulation configurations (such as modelling planetary orbits.)

The other Euler method SINGS employs is the symplectic (sometimes, the semi-implicit) Euler method, which computes the updated position vector using the new velocity vector as such

$$\vec{\mathbf{v}}_{i+1} = \vec{\mathbf{v}}_i + \vec{\mathbf{a}}_i \Delta t \tag{21a}$$

$$\vec{\mathbf{r}}_{i+1} = \vec{\mathbf{r}}_i + \vec{\mathbf{v}}_{i+1} \Delta t \tag{21b}$$

Note that the order in which these computations have been made is flipped from (20). While this method is still of the first order, its symplectic nature makes it ideal for simple and even relatively complex simulations.

4.2 Runge-Kutta

The Runge-Kutta method is a non-symplectic method in SINGS. It computes functional derivatives at times between t and $t + \Delta t$, and weights them to produce a prediction for the function at the next time step. The method we use, the fourth-order Runge-Kutta method (henceforth RK4) has fourth order error, which makes it quite appealing despite it's non-symplectic nature.

If we consider $\vec{\mathbf{y}}_i$ to be a vector containing the dependent variables of our system $\langle \vec{\mathbf{r}}_i, \vec{\mathbf{v}}_i \rangle$ and $\vec{\mathbf{f}}$ to be a vector containing the derivatives of $\vec{\mathbf{y}}_i$, we can compute $\vec{\mathbf{y}}_{i+1}$ for our second

order system by

$$k_1 = \Delta t\, \vec{\mathbf{f}}(t, \vec{\mathbf{y}}_i) \tag{22a}$$

$$k_2 = \Delta t\, \vec{\mathbf{f}}(t + \frac{1}{2}\Delta t, \vec{\mathbf{y}}_i + \frac{1}{2}k_1) \tag{22b}$$

$$k_3 = \Delta t\, \vec{\mathbf{f}}(t + \frac{1}{2}\Delta t, \vec{\mathbf{y}}_i + \frac{1}{2}k_2) \tag{22c}$$

$$k_4 = \Delta t\, \vec{\mathbf{f}}(t + \Delta t, \vec{\mathbf{y}}_i + k_3) \tag{22d}$$

$$\vec{\mathbf{y}}_{i+1} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{22e}$$

where k_1, k_2, k_3, and k_4 are the RK4 weights. In order to compute the various values of $\vec{\mathbf{f}}$, we propagate each individual particle forward at their old velocity $\vec{\mathbf{v}}_i$ at the timesteps which are specified by the RK4 coefficeints. So while the integrator itself remains $\mathcal{O}(N)$ like all the other integrators, we are having to calculate the force on each particle at those new locations 4 separate times, thus compared to other integrators we'd expect to see compute times at least 4 times greater.

4.3 Leapfrog Integration

Leapfrog integration is the final integration method provided by SINGS, and is the primary one used in major astrophysical simulation tools like GADGET (Springel, White, et al. (2005), Springel, Pakmor, et al. (2021)). The Leapfrog integrator is a 3rd order symplectic integrator, where with our fixed timesteps we calculate our position and velocity vectors by

$$\vec{\mathbf{r}}_{i+1} = \vec{\mathbf{r}}_i + \vec{\mathbf{v}}_i \Delta t + \vec{\mathbf{a}}_i \Delta t^2 \tag{23a}$$

$$\vec{\mathbf{v}}_{i+1} = \vec{\mathbf{v}}_i + \frac{1}{2}(\vec{\mathbf{a}}_{i+1} + \vec{\mathbf{a}}_i)\Delta t \tag{23b}$$

This simple scheme allows us to quickly integrate our simulation in a similar time to the Euler methods, yet with significantly lower computational error provided by the Leapfrog integrator's 3rd order accuracy.

5 Computation

Here we detail the specific programming aspects of SINGS, discussing the architecture of the program, the libraries we used, and specific challenges during development.

5.1 Why C?

SINGS is programmed entirely in vanilla C using the C99 standard for the Linux kernel. No external libraries were used with the exception of OpenMP, which we discuss further in 5.3. The reason we chose C to write SINGS was its simplicity and performance. The object oriented features of other languages like C++ wouldn't be particularly useful and we felt would only result in code ambiguity. C provided a simple framework where major program elements, like the simulation state, particles, and the functions which act on them, boiled down to passing structs between functions. That simplicity, coupled with the far higher performance from C being a low-overhead, compiled language made it ideal for the high computational demands of SINGS.

5.2 Modularity

The basic architecture of SINGS is highly modular, with SINGS execution split into 5 primary steps, those being:

1. Simulation Initialisation
2. Force Calculation
3. Particle Integration
4. Collision Resolution
5. Simulation Serialisation

In the first step, SINGS reads in a snapshot and parameterfile. The latter of which is used to build the internal simulation structure, and the former sets the internal simulation variables that'll be used. During the force calculation stage, the specified force code is applied to all the particles in the simulation, and when completed, pass the particles off to the desired integrator to have their positions updated. Before we pass off the simulation to be serialised, we resolve any particle collisions via the methods outlined in 2.2. In the last step, the simulation may be serialised and saved into a snapshot file. This doesn't necessarily mean the simulation has concluded, but rather that by user specification in the parameterfile—specifically snapshot frequency—that the current timestep is one that we output. Large simulations have snapshots that may take up many gigabytes, and so allowing the user to choose the rate at which we write those snapshots to the disk paramount, lest we run the risk of losing a run because of a crash, or filling up the disk if we were to serialise every frame. Finally. Assuming the simulation still isn't complete, we revert back to step 2 and propagate

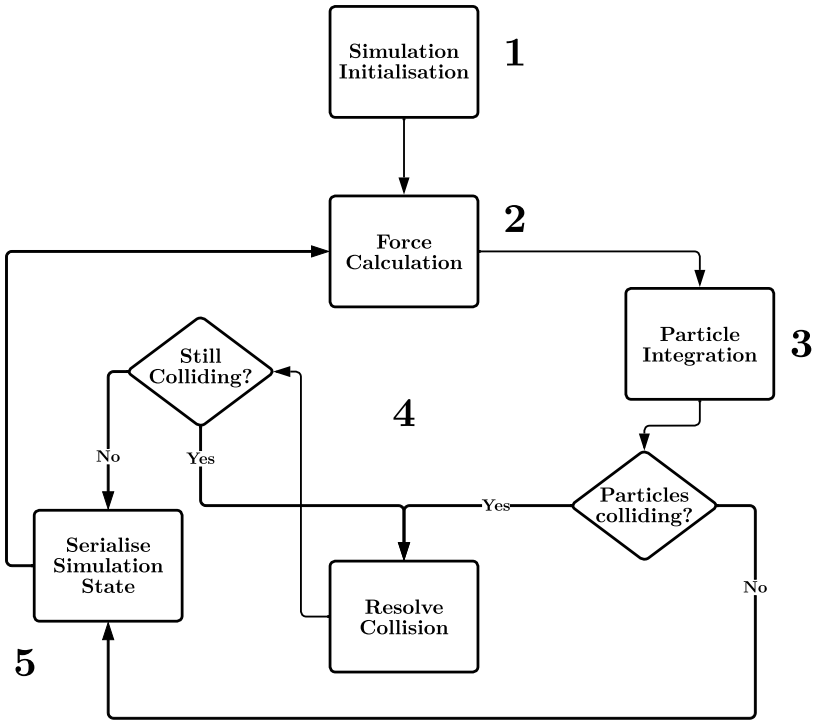


Figure 5: A flowchart illustrating the large scale architecture of SINGS, in particular the standard runtime loop between the 5 main execution stages.

the simulation forwards. A graphical representation of this entire process is shown in Figure 5.

A significant advantage in segmenting SINGS execution into these 5 stages is that they each remain completely compartmentalised. No modification to any one of the discrete parts of SINGS requires modification of any of the others.

To illustrate this point, we'll show the modularity of the code in terms of integrators and force codes (stages 2 and 3) for a single frame of a hypothetical simulation using a leapfrog integration scheme and the direct method.

Here is the main simulation loop, as we are concerned with the integration step, we only include it here. SINGS picks the configured integrator from a list of function pointers called `integrators[]`, and then passes in the simulation and force code from the array of function pointers `force_codes[]` to run. Given we are using leapfrog integration and the direct method, the indices for `integrators[]` and `force_codes[]` and are 3 and 0, respectively.

```
//main.c
...
for(int i = 0; i < sim.sim_cfg.timesteps; i++ {
        ...
        integrators[sim.sim_cfg.integrator](&sim, force_codes[sim.sim_cfg.force_method]);
        ...
}
```

For the leapfrog integrator, we must hold on to all of our current particle acceleration vectors so we can complete the integration step (23). We then call our force code. Notice, nothing in the leapfrog integrator touches the force codes, they simply pass the simulation state back and forth. We then run a parallel loop to complete the integration step on each of the particles.

```
//integration.c
...
void leapfrog(state *sim, void (*update)(state*)) {
    vec3 *ai = calloc(sim->num_particles, sizeof(vec3));
    for(int i = 0; i < sim->num_particles; i++) {
        ai[i] = sim->particles[i].acc;
    }
    update(sim);

    #pragma omp parallel for
    for(int j = 0; j < sim->num_particles; j++) {
        sim->particles[j].pos = vec_add(sim->particles[j].pos,
                                vec_add(vec_scale(sim->particles[j].vel,sim->sim_cfg.dt),
                                vec_scale(ai[j], .5f * sim->sim_cfg.dt*sim->sim_cfg.dt)));
        sim->particles[j].vel = vec_add(sim->particles[j].vel,
                                vec_scale(vec_add(ai[j], sim->particles[j].acc),
                                 .5f * sim->sim_cfg.dt));
    }

    free(ai);
}
```

Finally the force calculation step step, just the implementation of the direct method from (18a). We run a parallel for loop across all the particles. The main aspect of these functions to note is that the implementation of these functions do not inherently matter, as long as they all use the same simulation `state` structure. All we have done in this 3 step process is pass `sim` around between these various independent functions

```
//physics.c
...
void direct(state *sim) {
    #pragma omp parallel for
    for(int i = 0; i < sim->num_particles; i++) {
        sim->particles[i].acc = (vec3){0,0,0};
        direct_acc(&sim->particles[i], *sim);
    }
}

void direct_acc(particle *p, state sim) {
    for(int j = 0; j < sim.num_particles; j++) {
        if(p == sim.particles+j)
            continue;
        p->acc = vec_add(p->acc, vec_scale(fg_calc(*p,
                        sim.particles[j], sim.sim_cfg), 1/p->mass));
    }
}
```

5.3 Parallelisation

The parallelisation of SINGS is achieved via the use of OpenMP, the only external library used in this project. We chose OpenMP for its simplicity, as it allowed us to focus on developing an initially single core framework that could later be easily adapted for multple cores. At its heart, SINGS is essentially a bunch of nested loops, loops on particles, whether that for calculating forces on those particles or integrating those forces to find their positions and velocities. Because of the modularity of SINGS and the manner in which our simulation data is stored, the `state` structure, we can easily parallelise the high level loops which update particle parameters. This can be seen first hand in 5.2, where a simple `#pragma omp parallel for` is all that's needed to parallelise the algorithms.

To demonstrate the effectiveness of this parallelisation, we ran several simulations using the direct method and our two symplectic integrators with various thread counts up to 12, then normalised the y-axis by taking 1/runtime (we take runtime to be roughly $\propto 1/\text{threads}$). The results of this are shown in Figure 6.

5.4 Model Validation

We can validate SINGS models by checking for the consistency of certain well known simulation parameters, and seeing how they evolve over the course of a simulation. Examples we've discussed earlier related to Plummer softening (Figure 1) and the Barnes-Hut method (Figure 3). Another method besides computing the force directly is measuring the simulation

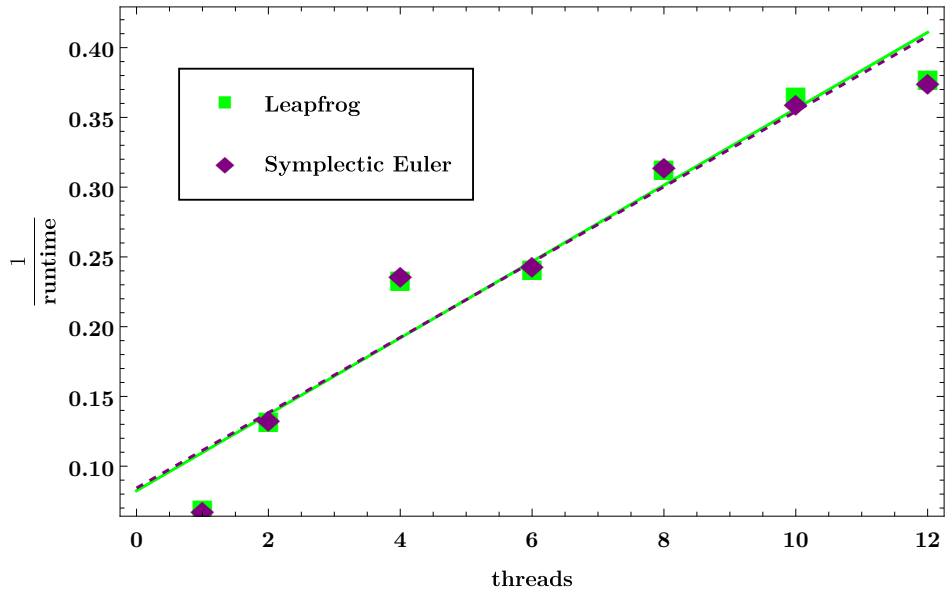


Figure 6: A figure showing 1/runtime for various thread counts between the leapfrog and symplectic Euler integration schemes.

energy. Recalling 2.1,

$$U = -G\frac{m_i m_j}{r_{ji}} \tag{2}$$

where U is the gravitational potential energy between two bodies. The only other source of energy in the simulation is going to be the individual particle kinetic energies, given by

$$K = \frac{1}{2}m_i v_i^2 \tag{24}$$

thus, if we want to compute the total mechanical energy E of the simulation at a time t, we can do a direct summation over all particle-pairs of their respective gravitational potential energies and their kinetic energies. Doing this yields

$$E = \sum_{i \neq j}^{N} \frac{1}{2}m_i v_i^2 - G\frac{m_i m_j}{r_{ji}} \tag{25}$$

This is one of the chief tools SINGS employs for diagnostics, as the reduction or increase in energy of the system over time, while expected (especially with low particle counts with many close-range interactions) might result in unrealistic outputs. A plot featuring the relative energy E_{rel} (given by $E_{\text{rel}} = \left|\frac{E_f}{E_i}\right|$) across a simulation with various integrators is shown in Figure 7.

Another measure of error that we haven't yet discussed is the error resulting from high

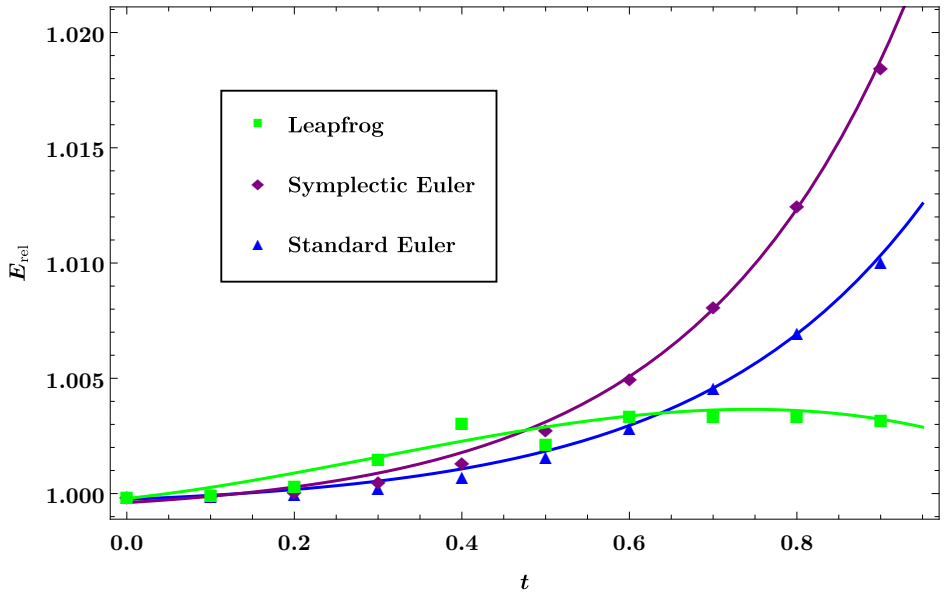


Figure 7: A plot of the relative energy E_{rel} over the course of a full simulation for various integrator choices.

Δt values. If we run the same analysis over the course of a simulation, comparing the change in E_{rel} for various Δt we can produce the graph in Figure 8.

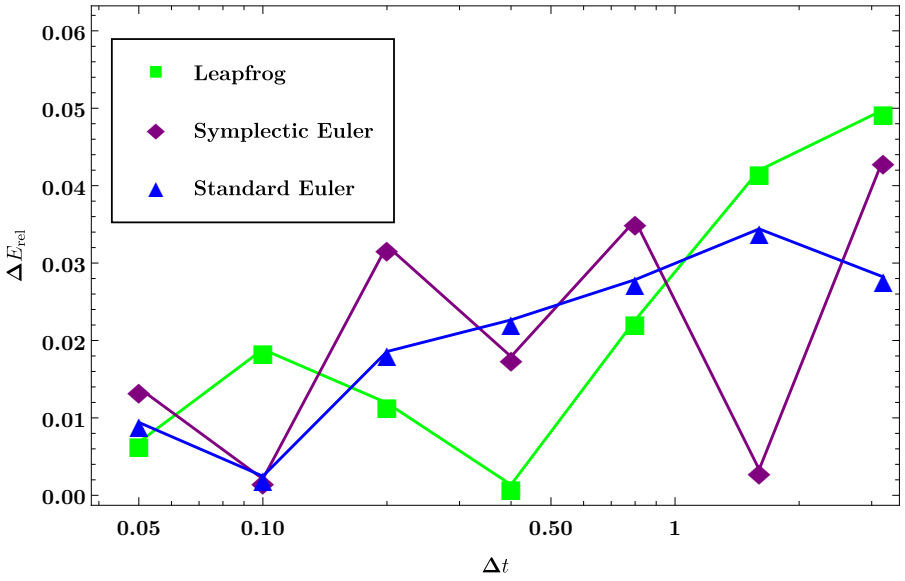


Figure 8: The relative change in energy over the course of a simulation using the direct method and different numerical integration methods, for various Δt in the range [0, 3.2]

We see quite a clean relationship in the change in energy of the system with respect to Δt across all integrators, though the randomness of a particular simulation state will mean that at certain Δt's deviations from this fit will occur.

6 Next Steps

As powerful a tool as SINGS is today, there are many areas within the codebase that could use improvement along with a plethora of new techniques and approaches to N-body computation which SINGS would benefit immensely from.

A specific example as discussed in the Abstract & Executive Summary was the Cloud-in-Cell (CIC) Particle Mesh (PM) method. A high level description is that the Particle Mesh method uses the Poisson equation and a Fast Fourier Transform (FFT) to compute the gravitational potential in cubic cells across the simulation space. Using this potential we can then apply a force to a particle inside the cell. This method like Barnes-Hut has an order $\mathcal{O}(N \log N)$, and is particularly well suited for computing the long-range forces on particles extremely quickly (depending on the FFT algorithm), but is less well suited for close interactions. This gives rise to derivations of the method like the Particle-Particle-Particle Mesh (P^3M) method (Roger W Hockney and Eastwood, 1988) which does a direct force calculation between nearby particles. This latter method was initially planned to be in SINGS at release but the present architecture did not adapt well to the method and our FFT was not easily parallelised. However, we feel strongly that the future of SINGS will likely involve some P^3M implementation.

A reworking of the architecture to support P^3M would also make way for the implementation of a more robust and realistic collisional particle model via Smoothed-Particle Hydrodynamics (SPH) (Gingold and Monaghan, 1977). Our current collisional gas model may be accurate if we take the particles to be like individual molecules colliding with one another but in terms of astrophysical simulations, it simply doesn't cut it. In short, SPH is a mesh-free approach to model fluid flows. It approaches the problem by dividing a fluid into discrete moving elements governed by a so-called "kernel function", which mediates particle interactions. The power of SPH as a computationally efficient way to accurately model the fluid-like flows of the interstellar and intergalactic mediums makes for its would-be implementation in SINGS highly desireable.

Before tackling any of these problems, however, there exists a major short term goal, and that is the implementation of periodic boundary conditions. These are necessary for high resolution structure formation simulations, and presently SINGS does not support them. In essence, periodic boundary conditions result in particles that leave the simulation space being acted on by mirrors of the simulation space. This is relatively cheap to compute as the force from the mirrors of the simulation can be acquired by a coordinate transformation on the particle. This is a high priority goal, and is likely the first major upgrade to SINGS after the conclusion of the New Mexico Supercomputing Challenge.

A smaller goal is to support a Hubble constant, essentially for modelling cosmic expansion. This would allow SINGS to analyse real ΛCDM cosmologies, scenarios like those close to the big bang, given the modification to the architecture to handle the numerical errors associated with such extreme conditions. And on the larger scales, it would allow SINGS to analyse structure formation in the context of an expanding universe. This isn't a high priority goal, but compared to some of the other ideas addressed here, it is the one most in reach.

And finally, a distant stretch goal is the implementation of Open MPI. Modifying the architecture to support the segmented memory of supercomputers would allow SINGS to contribute to meaningful research, given the changes we've outlined in this section. The current architecture around OpenMP is exceptionally simple in comparison to what an MPI implementation would take, however, it would truly prove SINGS' worth as a tool for modelling astrophysical systems.

7 Conclusion

What we have demonstrated through analysis of the data generated by SINGS is that it is an extremely capable tool for running complex astrophysical simulations. While the architecture cannot handle the high particle counts and the collisional dynamics of more mainstream astrophysical simulation codes, it more than holds its own. SINGS thus far has been validated in handling simulations in particle counts in excess of 1,000,000, and with the performance improvements facilitated by the multithreading of OpenMP will surely enable the creation of larger, more complex simulations. OpenMP support brings many opportunities for furthering the development of the software, and its implementation has probably been the single greatest achievement in developing SINGS. The fleshed out Barnes-Hut implementation already provided a strong foundation for further advancements, in addition to SINGS' general architectural modularity. The error diagnostic tools and the simulation i/o, however rudimentary, make SINGS flexible and easy to use for those hoping to make and run astrophysical simulations of their own. With the implementation of certain features discussed in Section 6, SINGS has the potential to become far more than just a humble N-body gravitation simulator.

The repositories for SINGS and SINGS I can be found on Github with the following links. Also included is a repository containing all the materials used in the creation of this document, including the raw data files generated from SINGS outputs. SINGS will slowly be released onto the Github under the GPLv2 license, as the source code is polished and documented. However, SINGS I as featured in this paper is already fully available for use,

however rudimentary it is in comparison.

SINGS: https://github.com/CodingKraken/SINGS
SINGS I: https://github.com/CodingKraken/SINGS-I
This Paper and More: https://github.com/CodingKraken/SCC-SINGS-Paper

Acknowledgements

This project started over 1 year ago as a spring break project. We had just discussed gravitation in AP Physics I, and I was eager to see what I could pull off, while also furthering my familiarity with C. I never could have forseen what this break time project would become.

First off I would also like to thank those involved with the Supercomputing Challenge itself, for putting on such an event to inspire and push people to not only develop their software skills, but also to inspire a love of the art of programming, of STEM subjects in general. I would like to thank Dr. Sharon Deland who provided pointed feedback on my interrim report, and additionally even though I only got to know them for half an hour, I would like to thank my February evaluators, Mr. Scott Levy and Mr. Abdalaziz Raad, for taking their time to review my project and my presentation. Their feedback was truly invaluable. I would also like to sincerely thank Karen Glennon, who is providing me the opportunity to go to the expo in person.

And finally, I never would have made it this far were it not for my Project Mentor and Physics teacher of the last two years Dr. Bradley Knockel. He was a constant source of inspiration and insight throughout the entire process, providing ideas, criticism, and motivation, raising concerns of problems I had never even thought to consider, and providing support down avenues of research I could not have approached otherwise. I would not be participating in this competition were it not for his endless support, and for that I am immensely grateful. Additionally, I want to thank my friends who pushed me to continue working on the project, even when I felt there was no moving forward. To all of them I am immensely indebted.

References

Aarseth, Sverre J and F Hoyle (1963). "Dynamical evolution of clusters of galaxies, I". In: *Monthly Notices of the Royal Astronomical Society* 126.3, pp. 223–255.

Appel, Andrew W (1981). "Investigation of Galaxy Clustering Using an Asymptotically Fast N-Body Algorithm". PhD thesis. Princeton Department of Physics.

Barnes, Josh and Piet Hut (1986). "A hierarchical O (N log N) force-calculation algorithm". In: *nature* 324.6096, pp. 446–449.

Dyer, Charles C and Peter SS Ip (1993). "Softening in N-body simulations of collisionless systems". In: *Astrophysical Journal, Part 1 (ISSN 0004-637X), vol. 409, no. 1, p. 60-67.* 409, pp. 60–67.

Eastwood, James W and Roger Williams Hockney (1974). "Shaping the force law in two-dimensional particle-mesh models". In: *Journal of Computational Physics* 16.4, pp. 342–359.

Gingold, Robert A and Joseph J Monaghan (1977). "Smoothed particle hydrodynamics: theory and application to non-spherical stars". In: *Monthly notices of the royal astronomical society* 181.3, pp. 375–389.

Hockney, Roger W and James W Eastwood (1988). "Particle-particle-particle-mesh (P3M) algorithms". In: *Computer simulation using particles*, pp. 267–304.

Holmberg, Erik (1941). "On the Clustering Tendencies among the Nebulae. II. a Study of Encounters Between Laboratory Models of Stellar Systems by a New Integration Procedure." In: *Astrophysical Journal, vol. 94, p. 385* 94, p. 385.

Hubble, Edwin (1929). "A relation between distance and radial velocity among extra-galactic nebulae". In: *Proceedings of the National Academy of Sciences* 15.3, pp. 168–173.

Maksimova, Nina A et al. (2021). "ABACUSSUMMIT: a massive set of high-accuracy, high-resolution N-body simulations". In: *Monthly Notices of the Royal Astronomical Society* 508.3, pp. 4017–4037.

Springel, Volker (2010). "E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh". In: *Monthly Notices of the Royal Astronomical Society* 401.2, pp. 791–851.

Springel, Volker, Rüdiger Pakmor, et al. (2021). "Simulating cosmic structure formation with the GADGET-4 code". In: *Monthly Notices of the Royal Astronomical Society* 506.2, pp. 2871–2949.

Springel, Volker, Simon DM White, et al. (2005). "Simulating the joint evolution of quasars, galaxies and their large-scale distribution". In: *arXiv preprint astro-ph/0504097*.

White, Simon DM (1976). "The dynamics of rich clusters of galaxies". In: *Monthly Notices of the Royal Astronomical Society* 177.3, pp. 717–733.

Zwicky, Fritz (1933). "Die rotverschiebung von extragalaktischen nebeln". In: *Helvetica Physica Acta, Vol. 6, p. 110-127* 6, pp. 110–127.

For Crying “Drought” Loud

New Mexico Supercomputing Challenge Final Report

Team: 2

Cottonwood Classical and Del Norte High School

Team Members:

Ayvree Urrea ayvreeurrea@gmail.com

Kiara Onomoto kiaraonomoto@gmail.com

Violet Kelly kellyviolet1111@gmail.com

Sponsor Teacher:

Karen Glennon

Mentor:

Flora Coleman

Table of Contents:

Executive Summary:

Drought in the Rio Grande River has and will continue to worsen as the years go on. According to research completed by Williams et al. as a result of climate change "...2000-2021 was the driest 22-yr period since at least 800 [CE]" [1]. This type of drought is dangerous due to the widespread effect on wildlife, crops, and general livelihood. Water is a resource that is important to the prosperity of not only human species, but also animals and crops as well. The rain that we get during monsoon season is not enough to relieve drought. We have also seen that drought is occurring more often, for longer periods, and sooner in the year. Through our project we researched the effects of drought in New Mexico specifically. We have found that methods such as optimizing federal reservoirs to more than one entity, redefining dams in a way that is more infrastructurally beneficial to its environment, and redistributing water sources, can lead to a healthier river that can withstand dryness and be drought-resistant. As the Rio Grande river has changed in the last 100 years, its systems of irrigation and water distribution should change as well. We believe that these solutions can save the amount of water supply in the Rio Grande, as well as use it in more impactful ways that can save the drying environment around the river. Our project focused on visualizing drought as it relates to the Palmer Drought Severity Index in different divisions of New Mexico monthly from the years 1895-2022. This visualization has allowed us to draw conclusions based on patterns of past drought conditions as it relates to seasonal and monthly data. We created a map of New Mexico using the eight divisions of New Mexico with a gradient that shows where drought is more severe and what months of the year this happens.

Problem:

The Rio Grande River, which flows through New Mexico, has been facing severe drought due to climate change and needs a better distribution of water plan. In addition, New Mexico needs to have additional options for water resources. In Las Cruces specifically, the Rio Grande didn't flow until March and was dry by September last year, completely disrupting the irrigation season which normally lasts from February to October. Since many cities in New Mexico reside along the Rio Grande River, this drought has a widespread effect on wildlife, crops, and livelihood. In Las Cruces, water is diverted and drained according to "water rights" which accounts for three-quarters of the state's surface and groundwater even though this only makes up 2.4% of New Mexico's Gross Domestic Product. Due to this distribution of water, New Mexican residents suffer not only by not being able to enjoy the pleasures of living by the water, but wildlife and nature suffer as well. Last year, Albuquerque also experienced a dry river on a 5-mile stretch of the Rio Grande for the first time in 40 years which means that drying events are taking place earlier and farther north than normal. Although rain alleviates drought, the Rio Grande is a snow-melt-driven system, and this is not enough to reverse long-term drought conditions. Water resources need to be reconsidered and redistributed to prepare for the nearing endemic state of drought in the Rio Grande River.

Objectives:

Our objectives were to create a code utilizing pycharm which could not only visualize drought as it already occurs in the Rio Grande and New Mexico, but also make predictions as to what patterns this drought will continue to follow. Specifically, we wanted to investigate how climate change may affect the future drought that we may face and the problems this could cause for our community. Finally, we wanted to discuss possible mitigation practices to the effect of drought or relief possibilities.

To start our project, we wanted to investigate possible measurements of drought. These include the Palmer Drought Severity Index which is a measurement of the duration and intensity of long term drought patterns and the Standardized Precipitation Evapotranspiration index which takes into account precipitation and potential evapotranspiration. We decided to use the Palmer Drought Severity Index because it could be best used to look at the historical aspect of drought without variables related to human impacts like irrigation or snowfall. This will be used to make predictions regarding drought in the future as it relates to historical patterns. To garner accurate results, we gathered New Mexico precipitation data from the past few years from NASA Worldview. In order to best accurately display drought in different areas of NM, we used the eight divisions of New Mexico's weather divisions which was obtained through the National Centers for Environmental Information. This has monthly PDSI historical division data from 1895-2022 and helps us to understand/visualize drought patterns for specific months of the year and areas of New Mexico. We also discussed possible strategies to mitigate drought such as water redistribution and reprioritization.

Research:

Drought Indices:

To investigate the relationship between drought and the future of the Rio Grande, we have investigated drought indexes that have recorded historical data. We primarily researched the Palmer Drought Severity Index (PDSI) and the Standardized Precipitation Evapotranspiration Index (SPEI). The Standardized Precipitation Evapotranspiration Index utilizes both precipitation and evapotranspiration data [7]. A limitation to this method is that it needs a long base period of around 30-50 years to collect data and process accurate results. Therefore, we looked into PDSI.

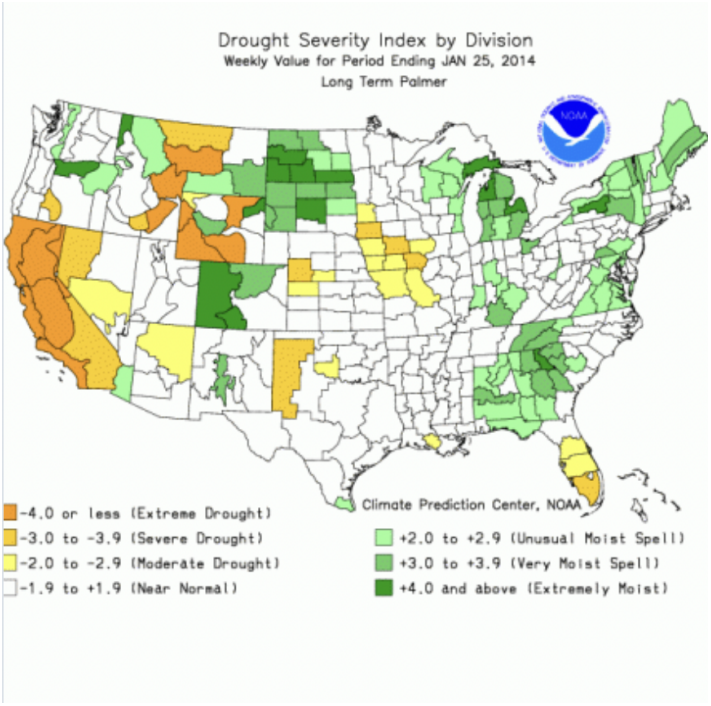


Figure 1

(https://climatedataguide.ucar.edu/climate-data/palmer-drought-severity-index-pdsi)

The Palmer Drought Severity Index is a historically-based drought index that contrives temperature and precipitation data from a given location. This data is then calculated into a scaled index (reference Figure 1) that generally spans from -4 (indicating severe drought) to +4 (indicating extreme moisture). This index uses and determines the water balance equation, including evapotranspiration, soil recharge, runoff, and moisture loss from the surface layer. It

"...uses precipitation, temperature, and available water capacity (AWC) as input data, thus the parameters of water balance can be calculated, such as runoff, recharge, evapotranspiration, and coefficient of soil moisture loss" [7]. Limitations within this index are its nonconsideration of human impacts such as irrigation, the variation of water balance components that also vary with elevation and land cover, and lastly, snowfall. In our project, we utilized this index in a way that put current conditions into a historical perspective. We have obtained monthly index data ranging from the years 1895-2022 in order to compare and validate these values with possible predictions.

Nasa Worldview:

NASA Worldview was a tool to strengthen our understanding of drought data. NASA Worldview gave a comprehensive list of data sets from varying factors that contribute to drought. Some of these factors include land surface reflectance, land surface temperature, population density, precipitation rate, radiance, and soil moisture (as seen in Figure 2). NASA Worldview allowed us to recognize how drought has its impacts on both humanity and the environment. A higher population density will more easily experience drought than a low population density. We looked into using NASA Worldview for our data but it only allowed data to be downloaded one day at a time. Considering we wanted to look at data at least a few decades in the past, this wasn't an option for us.

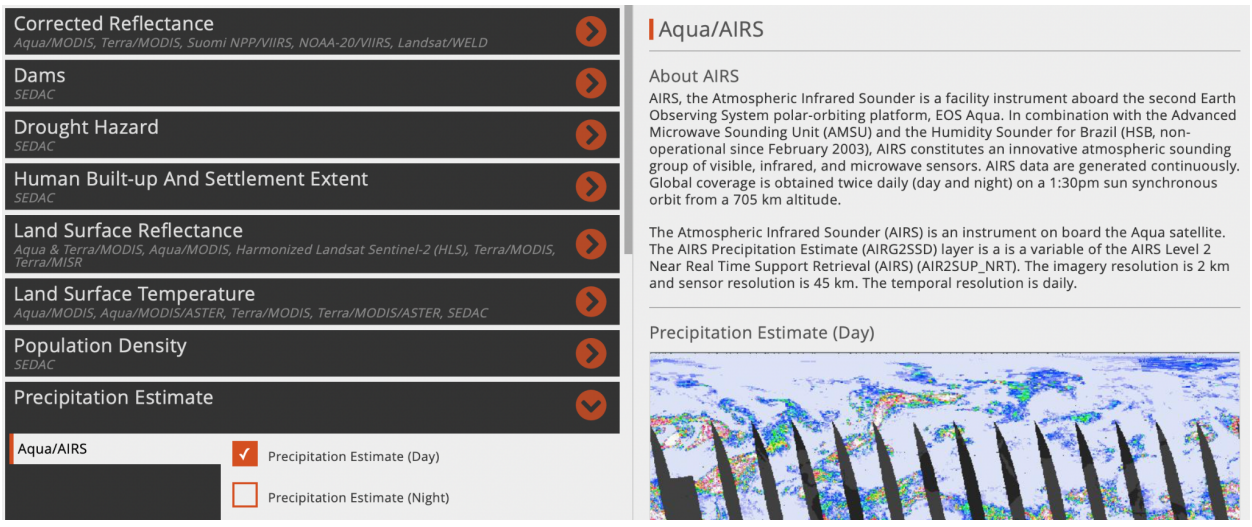


Figure 2

(https://worldview.earthdata.nasa.gov/?v=-122.84927811493296,27.101252117697136,-90.03298740584484,43.6461320168624&l=Reference_Labels_15m(hidden),Reference_Features_15m(hidden),Coastlines_15m,AIRS_Precipitation_Day,VIIRS_NOAA20_CorrectedReflectance_TrueColor(hidden),VIIRS_SNPP_CorrectedReflectance_TrueColor(hidden),MODIS_Aqua_CorrectedReflectance_TrueColor(hidden),MODIS_Terra_CorrectedReflectance_TrueColor&lg=true&s=-106.1084,34.4213&t=2023-01-21-T06%3A24%3A22Z)

Climate Divisions:

To accurately display drought in different areas of NM, we are using the eight divisions of New Mexico's weather divisions. These divisions include as shown in Figure 3: Central Valley (5), Central Highlands (6), Northeastern Plains (3), Southeastern Plains (7), Northwestern Plateau (1), Northern Mountains (2), Southern Desert (8), Southwestern Mountains (4). We obtained this division index from the National Centers for Environmental Information (NCEI). It presents historical division data of monthly PDSI from 1895-2022 [6]. We used this in our project to understand where drought may be the most severe in New Mexico as well as which months of the year this continually occurs in. These patterns have helped us understand drought and also enable us to make predictions about what will happen in the future if these patterns do continue.

Figure 3

Water Prioritization:

As aforementioned, humans can affect drought just as much as nature. Both in population (how many people need water) but also in management. What is the most effective way to allocate our water resources? The three largest sectors that this gets broken into are industrial, domestic, and agricultural. The industrial sector will use water for production of goods. A caveat of supplying the industrial sector with water is that it will most likely become contaminated after use. Water purification is a process that requires a lot of time and energy. The domestic sector

covers an individual's use of water. For example, showering, laundering clothing, running the dishwasher, and drinking. This sector's allocation will depend on the population of the state or county. Finally, the agricultural sector, where water is used to water crops or give to animals. This sector is especially important in New Mexico considering our relatively large production of pecans, peppers, and corn. Our climate is naturally dry so most plants require artificial watering in order to thrive.

During droughts, water has to be allocated precisely in order to fulfill the needs of each main sector. In addition to our drought predictions, we thought it would be interesting to explore the relationship between humans, economy, and nature. In a future project, it would be interesting to create an algorithm which would calculate the most beneficial use of a certain amount of water. Figure 4 shows the average separation of each sector.

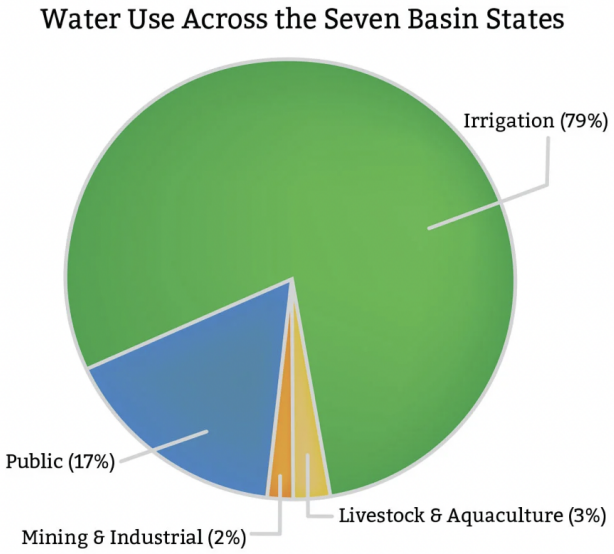


Figure 4

(https://learn.genetics.utah.edu/content/earth/waterallocation)

Predictions and Probability

One of the objectives of this project was to be able to predict future drought, thus identifying how severely climate change will affect our weather patterns. Our hypothesis is that as the Earth experiences this positive feedback loop of growing warming and warming. Drought will also become more and more severe. The increased temperature will directly cause an increase in evaporation (decrease in soil moisture). This rapid evaporation will severely dry out soil, so when it does rain, the soil won't be able to soak up the water. We will instead probably experience an increase in flash flooding. We predict that drought will continue to intensify and therefore we must begin to find solutions now.

Probability goes hand in hand with predictions. Most predictions have a likelihood, a probability. We researched how to incorporate probability formulas and concepts into this project. This would also eliminate any misleading or false information by stating "this is our predicted rate of precipitation with a likelihood of …" We researched independent events, mutually exclusive events, conditional probability, and how to calculated expected value. This general knowledge of probability allows us to accurately integrate the topic into our code.

Python and Pycharm

Objective: Creating a model that computationally predicts future droughts, in order to identify possible solutions in water distribution.

Overview:

In our program, we wanted to create a model that visually represented New Mexico and its historical data pertaining to droughts. This would then allow us to then transition to the idea of predicting droughts and the impacts it has on surrounding environments. To start this, we analyzed historical data of PDSI in different divisions of New Mexico to find patterns and the relationships that are present with this large array of data.

Obtaining and converting data:

In our research, we were able to find PDSI drought data from NCEI databases that provided files of monthly drought conditions in the contiguous United States. We then converted this data into our program and found a way to simplify and understand this abundant data in order to only focus on New Mexico drought data. Then we had to dissect and understand the format of the series of ten digits.

Finally, we used shapefiles also acquired from NCEI databases to create a visual of New Mexico in our program. This allowed us to create a structure and shape of the state, as well as display the eight climate divisions.

```
0503051926   1.10  -0.20  -0.40  -0.91  -1.07  -1.28  -0.72  -1.05  -1.05  -1.12   0.28   0.40
0503051927   0.25   0.48   1.18   1.66  -1.30   1.34   2.02   2.95  -0.14  -0.59  -0.72  -0.70
0503051928  -0.92  -0.93  -0.97  -1.34   0.71   2.95   4.26   4.09   3.35   3.81   4.02   3.30
0503051929   2.68   2.61   1.86   2.48   2.16   1.51   1.67   1.91   2.84   3.55   3.86   3.22
0503051930   2.82   2.16   1.49   1.09   2.33   2.11   2.38   3.61   3.02   3.82   4.45   3.84
0503051931   3.17   3.37   3.52  -0.15  -0.58  -1.05  -1.80  -1.77  -2.47  -2.74  -2.57  -2.39
0503051932  -2.24  -2.15  -2.11  -1.39  -2.28  -1.85  -1.49  -1.66  -2.12  -2.15  -2.46  -2.24
0503051933  -2.43  -2.52  -2.66  -1.57  -1.19  -2.38  -2.71   2.36   2.44  -0.72  -1.21  -0.90
0503051934  -1.25   0.70  -0.58  -1.50  -3.18  -3.67  -4.94  -5.21  -5.06  -5.35  -5.13  -4.93
0503051935  -4.85  -4.50  -4.67  -4.01  -2.20  -1.30  -2.26  -2.93  -2.22  -2.48  -2.26  -2.29
0503051936  -2.21  -2.22  -2.50  -2.59  -2.47  -3.41  -4.14  -4.31  -3.64  -3.30  -3.47  -3.13
0503051937  -2.93  -2.78  -2.48  -2.86  -3.80  -3.50  -4.43  -4.93  -4.44  -4.30  -4.07  -3.44
0503051938  -3.29  -3.15  -2.91   0.58   1.17  -0.47  -0.27  -0.62  -0.12  -0.87  -0.86  -0.79
```

Figure 5

Nested Dictionary:

In our program, we implemented a nested dictionary to separate the vast amount of data by state, climate division, month, and year. We chose to use a nested dictionary in our program because of its ability to comprehend and reiterate data in a limited amount of time. This would then help us have the ability to find certain points of drought data in a given year, month, and climate division. In figure 6 and 7, it shows the structure of our nested dictionary and our ability to print certain divisions and months, in order to analyze specific points.

```
def create_dictionary(input_data):
    """
    Create a dictionary using the divisions in New Mexico
    """

    division_names = ['NORTHWESTERN PLATEAU', 'NORTHERN MOUNTAINS',
                      'NORTHEASTERN PLAINS', 'SOUTHWESTERN MOUNTAINS',
                      'CENTRAL VALLEY', 'CENTRAL HIGHLANDS',
                      'SOUTHEASTERN PLAINS', 'SOUTHERN DESERT']

    month_names = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
                   'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

    data_dictionary = {}
```

Figure 6

```
print(len(nm_dictionary['CENTRAL VALLEY']['Jan']))
print(nm_dictionary['CENTRAL VALLEY']['Jan'])
```

Figure 7

Visual Gradient:

One of the vital parts of our program was the representation of the drought data in a visual setting using shapefiles, PDSI drought data, and the nested dictionary. One of our main goals for

this program was to be able to represent the change of drought over time in these eight climate divisions. The graph displays the shape of New Mexico and the borders of the climate divisions. It then uses a color gradient to show the difference in drought in varying shades of oranges. The darkest orange represents a drier climate condition compared to lighter shades of orange which show wetter and more sustaining conditions. With the implementation of the nested dictionary, we are able to show different time frames throughout a given year. This has allowed us to compare the varying months in years spanning from 1895 to 2022 (figure 8 and figure 9)

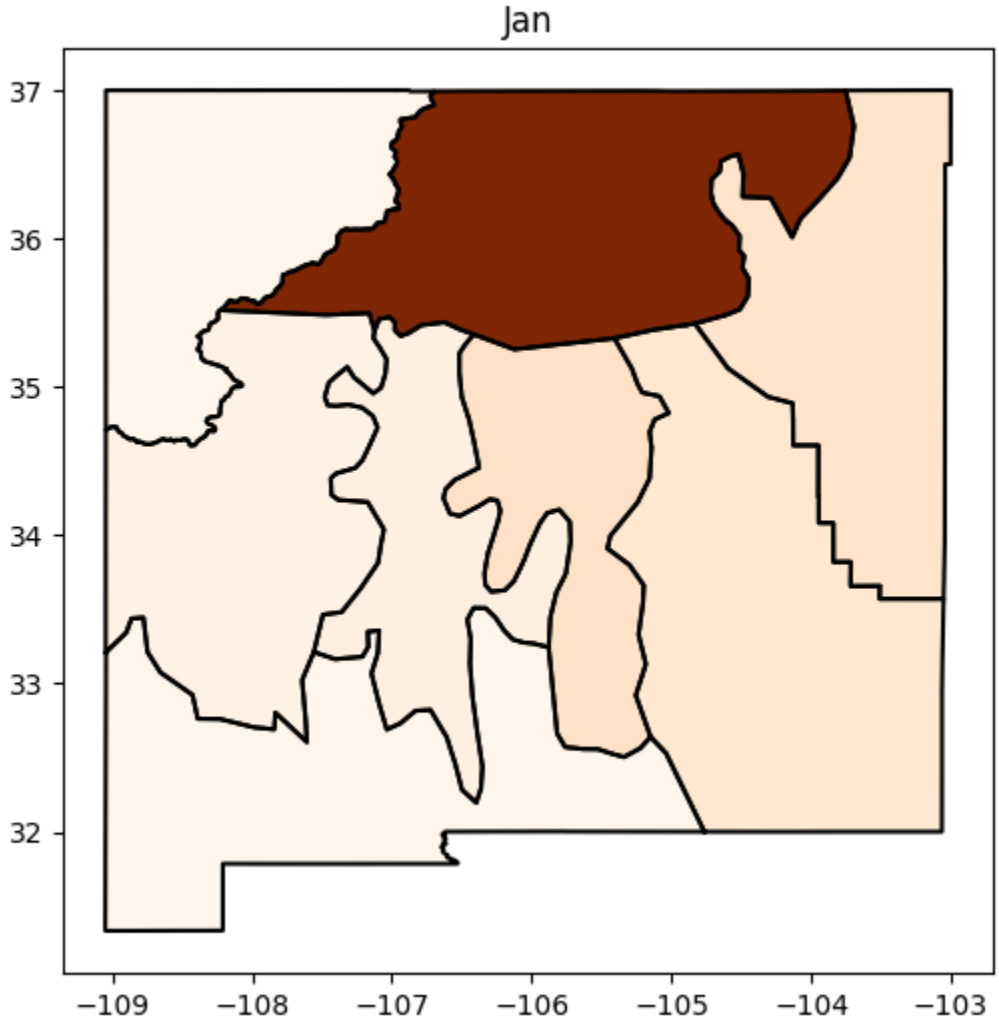


Figure 8

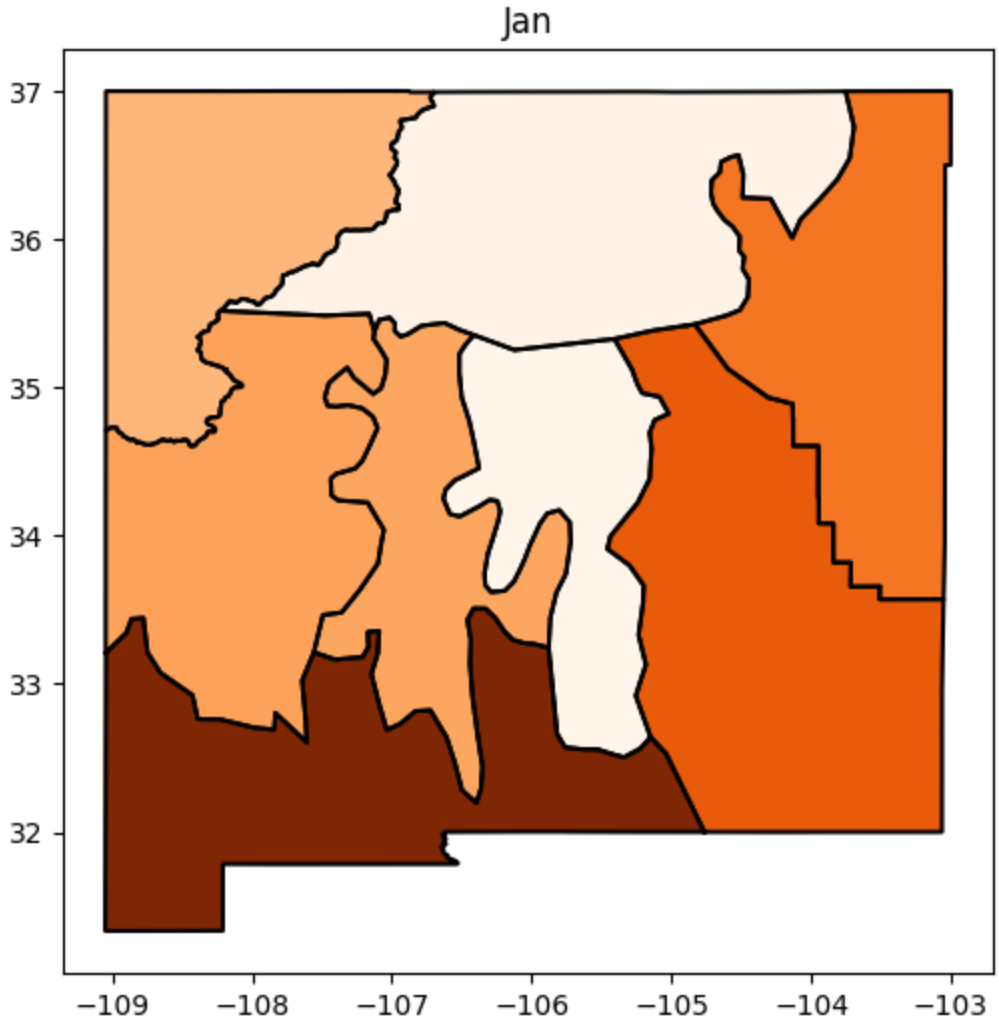


Figure 9

Tables and Graphs:

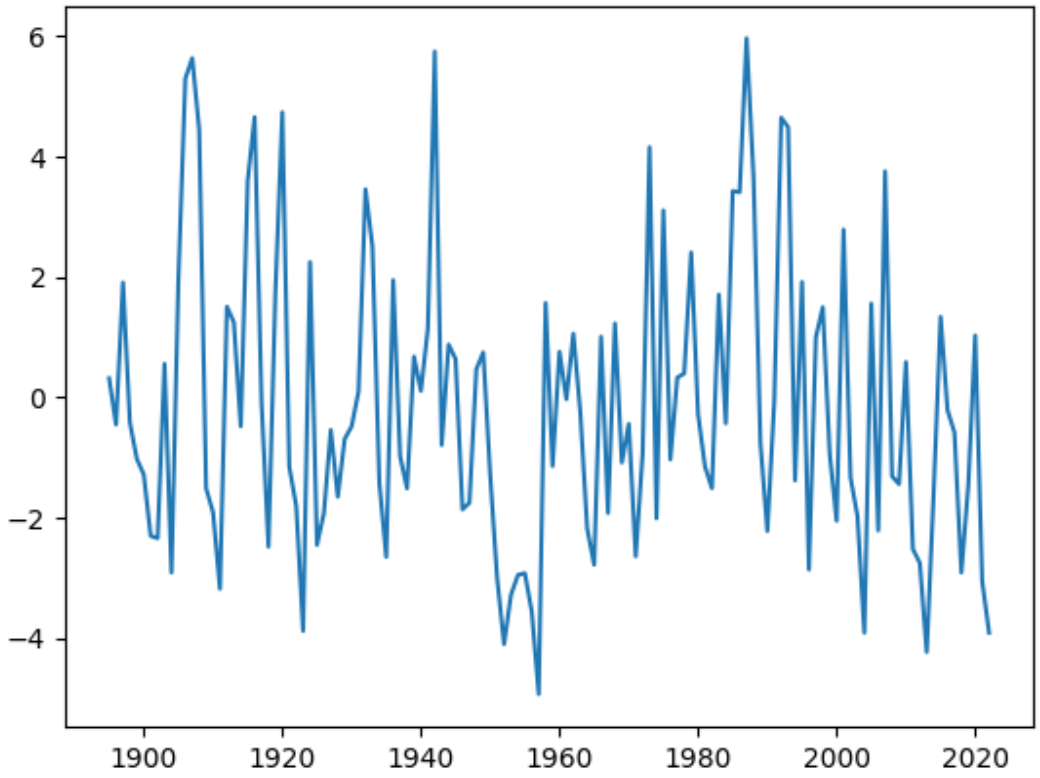


Figure 10

In Figure 10, the x-axis depicts year and the y-axis depicts the precipitation rate. The data represents each plot of PDSI monthly data between the years 1895 to 2022.

Results:

This year our results were mostly visual and pattern related. We used our code to use historical PDSI data to model drought in the eight divisions of New Mexico. A gradient is used in order to see where drought is more severe and the code runs through all 12 months of a specific year. Through running our code we found some patterns for specific years. An observation was that in 1895 drought was present in central NM (division 5). As compared to 2022 drought was more stagnant in the SW of NM in 2022, and drought was shown to vary in 1895 with there being no drought in most of NM from December 1895-March 1896. In 1950 drought was harsh on the east of New Mexico, and in 1990 there was consistent drought in divisions 2 and 3. In 2000 drought conditions were severe in October, November, and December. In 2010 drought started in the west and gradually moved up the state while Division 5 stayed impacted by drought. In 2015 drought stayed present in the northeast side and in 2018 the summer was not impacted by drought while the winter was. In 2020 Drought was harsh in the Southwest and there was consistent drought in division 5 of 2021 while there was no drought in division 1. Finally, in 2022 there was consistent drought in the Southwest. As we can see from these observations, drought has remained an issue in various areas of New Mexico throughout the years. Some of the most damaged areas are central NM and the northwestern plateau.

Conclusion:

Through this project we were able to gain a better understanding of drought in New Mexico and possible mitigation strategies. Our code helped us to view PDSI through many years of the past utilizing the eight divisions of New Mexico. Through this information, we were able to find patterns with drought and discover areas of New Mexico where drought is consistently worsened. We have found that methods such as optimizing federal reservoirs to more than one entity, redefining dams in a way that is more infrastructurally beneficial to its environment, and redistributing water sources, can lead to a healthier river that can withstand dryness and be drought-resistant. As we continue, we would like to make predictions of drought in the future with the assumption that drought patterns will continue as seen. Our hypothesis is that as the Earth experiences this positive feedback loop of growing warming and warming. Drought will also become more and more severe. Since drought is an issue that is consistent at the least it is exceptionally important that we start developing strategies to mitigate the problems caused to the community, wildlife, and crops.

Significant Achievements:

The most significant achievement that we had as a team this year was time management and organization. Since all three members of our team are seniors we had to learn how to better manage our time and prioritize our tasks for the challenge while also handling the workload in school and at our jobs. It was an incredibly busy and stressful year so we had to make sure that we had all the deadlines for the challenge set in our calendar. In addition, we had to make sure that our communication was always kept up with one another as we go to different schools. We set times to meet during the week that would work with all of our schedules and we made a commitment to work on our tasks on our own time separate from those meetings as well. We effectively set up a plan to research our project, do the code for our project, and also complete all of our big tasks such as the proposal, interim, evaluation, and final report. As a team of three, we did a great job staying organized and following through with the tasks we embarked on.

Ayvree Urrea:

The most significant achievement that I had this year was staying organized and appreciative amidst an incredibly stressful season of life. I have felt very overwhelmed this year with extracurriculars, sports, school, and work. However, I also wanted to get the most out of every experience with it being my last year in most of my activities. This year I have taken the time to appreciate all that I have learned through my years in the challenge. Some of those things being presenting skills, organizational skills, researching skills, and coding skills. I am very proud of the comfortability I have grown to have with presenting as I have gone through with evaluation and expo every year. I have gained many coding skills in python especially understanding matrices, functions, loops, and if/and statements. I have also become a more knowledgeable researcher in order to find the information needed and utilize it. Most importantly, I have gained confidence in my capacity to take a problem I see in the world and actually make changes- computationally or otherwise- that will help mitigate the problem. This is a skill I will forever take with me in college and beyond.

Kiara Onomoto:

The most significant achievement that I had this year was expanding my skills in communication with my team members and mentor, as well as being able to be the sole programmer on my team. Throughout my time participating in the Supercomputing Challenge, I have felt that my ability to communicate has grown significantly to what it is today. Being from a team that is divided into two schools, and having to communicate with a mentor who also has a busy schedule, this year has taught me the importance of communication. As I have participated in this program, I found it to be important to be a dependable team member. This meant that I had greatly accepted the role of communicator between my team and mentor. This year, I felt I have achieved this by organizing the availability of meetings and keeping track of deadlines. Furthermore, my skills in programming in Python have greatly benefited this year with my dedication to stepping out of my comfort zone to being the sole programmer that fully understood and could comprehend new skills that were taught to us by our mentor. These skills will greatly impact my future endeavors.

Violet Kelly:

My most significant achievement this year was integrating myself into the community of Supercomputing and learning how to present code to an in-person audience. This only being my second year in the challenge, this was the first time I got to go out to Socorro and immerse myself in the Supercomputing culture. I learned a lot through both experience and the lessons we were taught. It was also really inspiring to see everyone who was also competing in the challenge. I also got the opportunity to present our code to professional scientists, which was terrifying at first, but after the shock of nerves I became more comfortable presenting to incredibly intelligent strangers. I feel that one presentation had really prepared me or at least gotten me comfortable with the skill of public speaking that I will no doubt have to utlize in college. The question portion also helped me understand how I can better communicate my ideas to others in a way that is clear and concise. Alongside this, I've learned how to communicate my passion to others. I am passionate about mitigating the effects of climate change and educating

others on the reality we are facing. This coding project was a great way for me to investigate an issue that I was naturally motivated to study and share.

Acknowledgments:

Karen Glennon, Retired Teacher

Patty Meyer, Retired teacher

Flora Coleman, Sandia National Laboratories

Bibliography:

[1] Williams, A.P., Cook, B.I. & Smerdon, J.E. Rapid intensification of the emerging southwestern North American megadrought in 2020–2021. *Nat. Clim. Chang.* 12, 232–234 (2022). https://doi.org/10.1038/s41558-022-01290-z

[2] Vicente-Serrano , Sergio, and Santiago Begueria. “About the Spei.” *Information SPEI, The Standardized Precipitation-Evapotranspiration Index*, https://spei.csic.es/home.html.

[3] Boller, Ryan. “Worldview: Explore Your Dynamic Planet.” *NASA*, NASA, 19 Dec. 2022, https://worldview.earthdata.nasa.gov/?v=-112.30263760851366%2C27.308952305805594%2C-99.0387769912644%2C41.102582503921326&l=Reference_Features_15m%2CCoastlines_15m%2CIMERG_Precipitation_Rate%2CReference_Labels_15m%28hidden%29%2CNDH_Drought_Hazard_Frequency_Distribution_1980-2000%2CVIIRS_SNPP_CorrectedReflectance_TrueColor%28hidden%29%2CMODIS_Aqua_CorrectedReflectance_TrueColor%28hidden%29%2CMODIS_Terra_CorrectedReflectance_TrueColor%28hidden%29&lg=false&t=2022-07-13-T20%3A00%3A00Z.

[4] Earth Science Data Systems, NASA. “Agriculture and Water Resources Data Pathfinder.” *NASA*, NASA, 17 May 2019,

https://www.earthdata.nasa.gov/learn/pathfinders/agricultural-and-water-resources-data-pathfinder.

[5] Jim Robbins / Photography by Ted Wood • June 2, et al. “The Vanishing Rio Grande: Warming Takes a Toll on a Legendary River.” *Yale E360*, https://e360.yale.edu/features/warming-and-drought-take-a-toll-on-the-once-mighty-rio-grande.

[6] NCEI.Monitoring.Info@noaa.gov. “Historical Palmer Drought Indices.” *Historical Palmer Drought Indices | National Centers for Environmental Information (NCEI)*, https://www.ncei.noaa.gov/access/monitoring/historical-palmers/.

[7]F Balbo et al 2019 IOP Conf. Ser.: Earth Environ. Sci. 303 012012

https://iopscience.iop.org/article/10.1088/1755-1315/303/1/012012/pdf

SUPERCOMPUTING CHALLENGE

ROBOTIC WILDFIRE DETECTION

SYSTEM:

SMOKEY

Capital High School

Team 09 members:

Zachariah Burch

Isel I Aragon M

Britny Marquez

Daniel Leon Leon

Teacher Sponsor:

Barbara Teterycz

Mentor:

David Ritter

April, 3 2023

Table of Contents

Abstract/Executive Summary

The purpose of this project is to address and help with the frequent wildfires that are happening in the state of New Mexico every year. Regardless of whether wildfires are controlled or uncontrolled, they should always be treated very seriously and with extreme caution because they are the source of pollution, and are life threatening. By releasing CO2 into the atmosphere, wildfires also contribute to global warming. Furthermore, forests purify the air that we breathe, offer a home to much of the world's wildlife and flora, and provide natural resources (e.g. timber and medical plants).

The result of the project is a robotic fire weather conditions detection system called Smokey, which alerts about either very possible or already existing fires. It includes the following parts: Arduino MEGA 2564, CO2 and TVOC sensor, Dust and Pollen sensor, Temperature and Humidity sensor, RTC Clock, BlueTooth and SD drives, infrared light sensor, parabolic dish, LCD display, and several LEDs. Smokey was programmed using C++ language and has been tested during different experiments and through the data collection and analysis. The experiments included burning incense, candles, and matches, turning humidifiers on/off, and exposing Smokey to an infrared heater.

Smokey correctly detects increased/decreased levels of CO2, TVOC, smoke, dust, temperature, and humidity in the air, as well as it correctly senses IR (infrared) light reflected in a parabolic dish to the IR detector caused by the heat of the flames, which were burnt in the chemical fume hood. This fire weather conditions detector could be used by the Fire Services to determine when to avoid prescribed fires. In addition, by installing a network of such robots in the forest that would communicate with each other via Bluetooth and be powered by the PV solar energy, the Forest Services would be quickly alerted about the wildfires that have already started on their own.

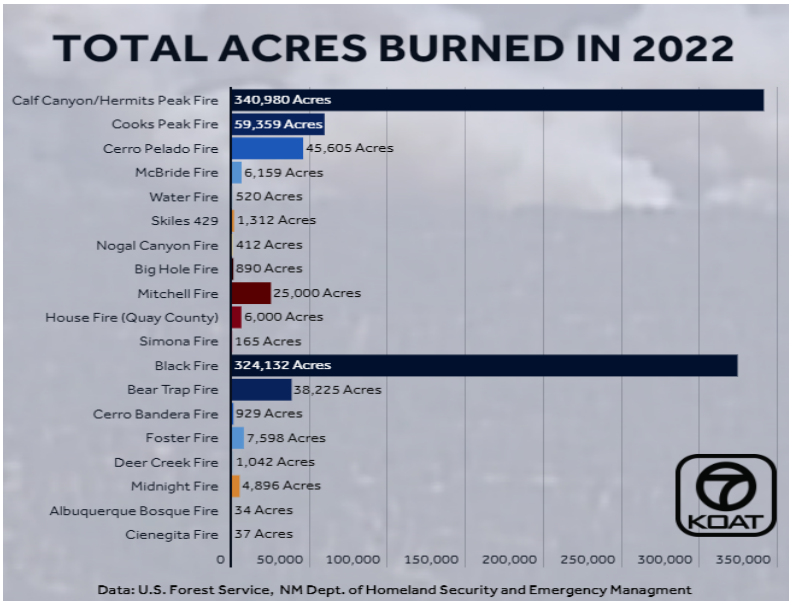
Hypothesis

High levels of dust, smoke, CO2, and temperature, and low level of humidity in the air as well as the presence of infrared light of a flame detected from far away may indicate possible wildfires and wildfire weather conditions. Developing a robotic system with sensors that would measure all these levels and detect such an infrared light could inform about already existing wildfires or predict the potential ones.

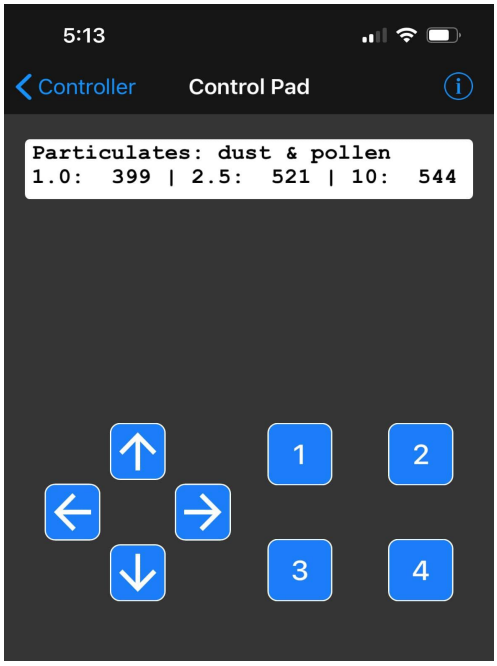
Identify the Problem

Problem Background and Research

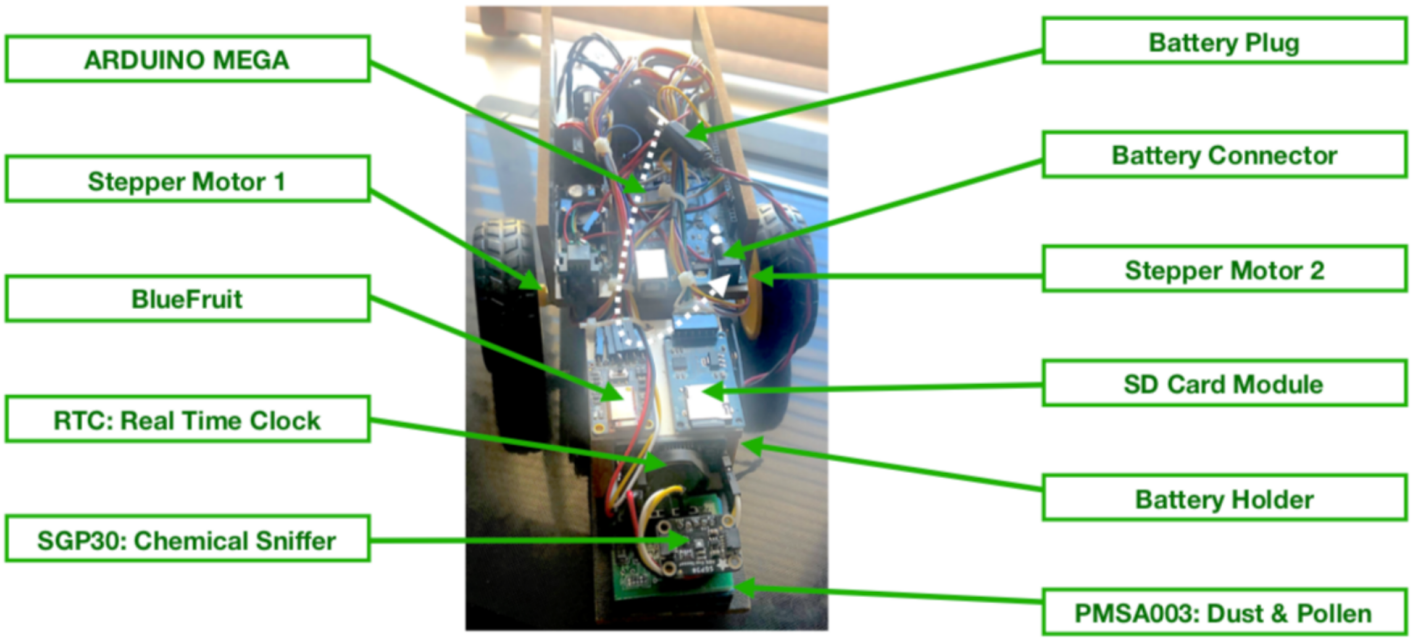
According to InciWeb, “the Hermit's Peak fire began April 6, 2022 as a result of the Las Dispensas prescribed fire on the Pecos/Las Vegas Ranger District of the Santa Fe National Forest” (InciWeb). While “the Calf Canyon fire was caused by a pile burn holdover from January that remained dormant under the surface through three winter snow events before reemerging in April” (InciWeb). As the fires reemerged and grew, they then managed to merge and become one, known as The Hermit's Peak and Calf Canyon fire. All together more than 800,000 acres burned in the state of New Mexico in 2022.



The fires that emerged became immense to the point that the smoke was visible in Santa Fe. Since the team had already developed an air quality monitoring system called Snoopy (that was submitted to the Supercomputing Challenge last year), they were able to measure the level of dust and smoke in the air during that time. The images below show those levels as well as the smoke captured here, in Santa Fe.



Previous Project: Snoopy



Idea Generation / Goal

As mentioned before, Snoopy was able to detect the level of dust, smoke, pollen, TVOC (Total Volatile Organic Compounds), and CO2 in the air. Based on the collected data during the time of wildfire, the team was inspired to upgrade Snoopy to a fire and fire weather measuring system. This is where the idea of Smokey came from.

Work Done By Others

In order to implement this idea and accomplish this goal, the team was regularly meeting with the mentor from industry and their teacher. During the meetings the team learned about each part of hardware, including laying out circuit boards and adding sensors to it, and C++ coding.

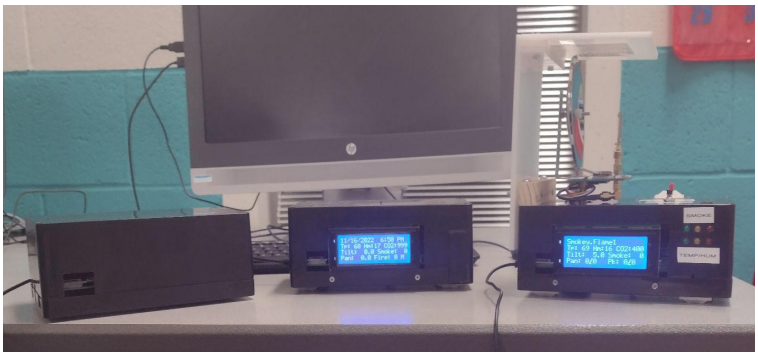
Constraints

Since some of the team members were unable to meet regularly, after meeting with the mentor, the team leader was updating the rest of the team after school. It was a very challenging and time consuming role to do both work with the professional mentor and teach the rest of the team.

Model/Prototype

Description of Model

Smokey is a robotic monitoring system that detects the levels of dust, smoke, CO2, TVOC, humidity, temperature, and infrared light.



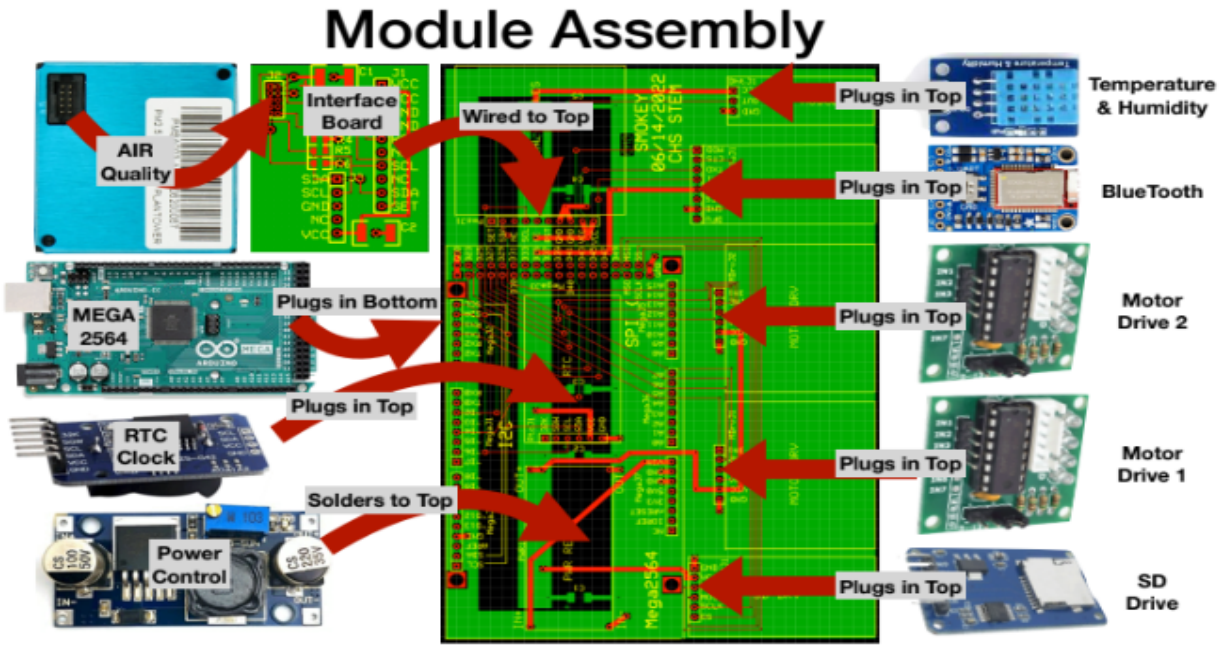
The development of Smokey included the following three phases displayed on the image above:

1. Protype #1 (on the left) only included the basic sensors;
2. Prototype #2 (in the middle) in addition to basic sensors, also included warning LEDs and LCD (Liquid Crystal Display);
3. Prototype #3 (on the right) in addition to components in prototype #2, also included infrared sensor and parabolic dish.

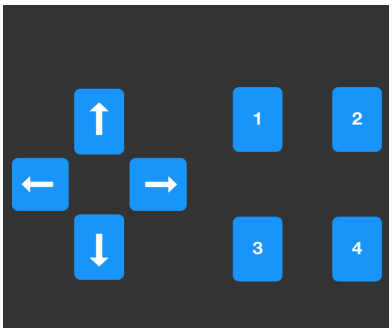
Smokey reused sensors and their software used by Snoopy that detected smoke, dust / pollen particles, as well as TVOC and CO2. The following parts were also added to Smokey: circuit board, power module, infrared, humidity and temperature sensors. The table below lists similarities and differences between Snoopy and Smokey.

Snoopy	Both	Smokey
• Uses wires to connect the sensors • Uses stepper motors to drive around • Detects air quality	• Have CO2, TVOC, pollen sensors, RTC, and SD card modules. • Use C++ coding	• Uses a circuit board with a modular interface • Has an IR sensor, a humidity and temperature sensor, and a power module. • Uses stepper motors to pan and tilt IR sensor and dish • Detects wildfires

As shown on the image below, in order to build Smokey, the following hardware was required: Arduino Mega, circuit board, power control and bluetooth modules, SD and motor drivers, RTC clock, air quality, and temperature and humidity sensors.



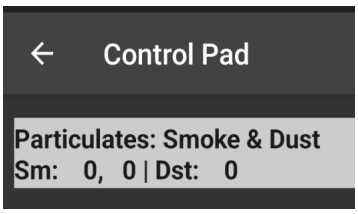
The team learned how to connect Smokey to the Bluefruit app installed on their smartphones through Bluetooth, and how to control the infrared sensor and parabolic dish movement when each of the following arrow buttons was pressed:

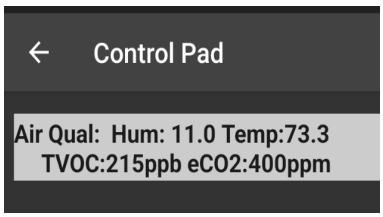


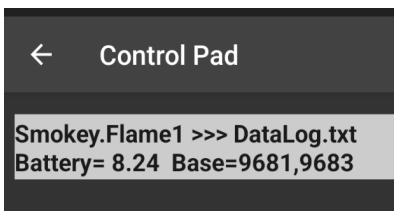
In order to accomplish this task, the team had to add the following instructions to the code:

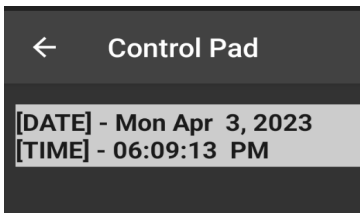
```
14    newCmd = GetCommand();        // Always read from the App input on bluetooth
15    if(newCmd < 0 || newCmd >8) { // Illegal command - cancel it
16      currentCmd = 0;             // Zero out the command variables
17      newCmd =0;
18      MotorsOff();                // Make sure motors are off
19      }
20    ActiveCmd = (currentCmd<=8) && (currentCmd>=5); // 5<=Cmd<=8 is active command
21    if(ActiveCmd == 0)  currentCmd = newCmd; // If its not active cmd, update it
22// Active commands implemented
23    if((currentCmd == 8) && (Auto == 0))  Pan(1);      // Pan right
24    else if((currentCmd == 7) && (Auto == 0)) Pan(-1);   // Pan Left
25    else if((currentCmd == 6) && (Auto == 0)) TiltDown(1);  // Tilt Down - not used at present
26    else if((currentCmd == 5) && (Auto == 0)) TiltUp(1);    // Tilt Up - not used at present
```

In addition, the team also learned how to program each of the four buttons to display data on the screen of the Bluefruit's app. Button 1, when pressed, displays the level of smoke and dust in the air; button 2 displays the level of humidity, temperature, and chemicals, such as CO2 and TVOC, button 3 displays the level of battery, and button 4, when pressed together with button 1, displays the date and time, as shown on the screenshots below:







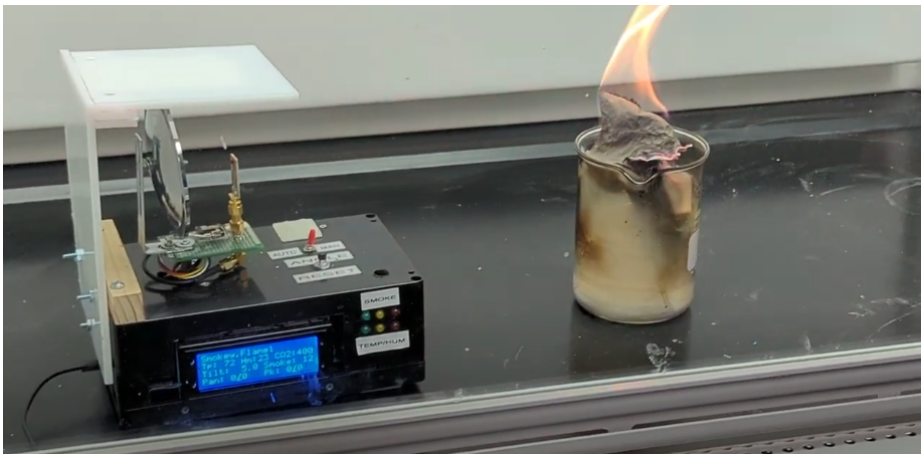


The screenshot below shows the part of the code responsible for accomplishing the above tasks.

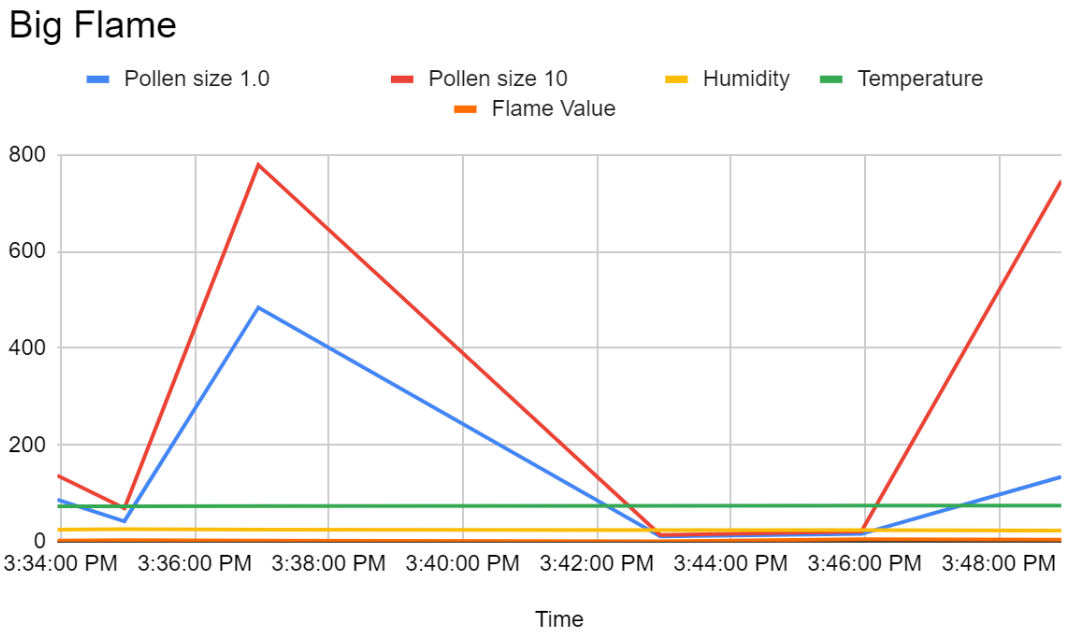
```
27    else if(currentCmd == 4){         // Pressed button 4 ... Date and Time
28      AppShift = TRUE;
29    } else if(currentCmd == 3){   // Pressed button 3 ... Battery & baselines
30      if(AppShift != FALSE) {     // For App Display #n6
31        currentDisplay = 6;
32        updateFlmlDsply();
33        updateDisplay();
34      } else {               // For App Display #3
35      currentDisplay = 3; // Each button goes through same sequence
36      updateNameDsply();
37      delay(100);
38      updateDisplay();
39      currentCmd = 0;      // Commands 4,3,2,1 are button presses and get cancelled
40      newCmd =0;           //   once implemented.
41      delay(100);
42      }
43    } else if(currentCmd == 2){   // Pressed button 2 ... Chemical data CO2 and TVOC
44      if(AppShift != FALSE){      // For App Display #4 & # 1
45        currentDisplay = 5;
46        updateFlm0Dsply();
47        updateDisplay();
48      } else {
49        currentDisplay = 2;
50        updateAirQualDsply();
51        delay(100);
52        updateDisplay();
53        currentCmd = 0;      // Commands 4,3,2,1 are button presses and get cancelled
54        newCmd =0;           //   once implemented.
55        delay(100);
56      }
57    } else if(currentCmd == 1){   // Pressed button 1 ... Particulate data
58      if(AppShift != FALSE){      // For App Display #4
59        currentDisplay = 4;       // Button number is same as display number - stopped here
60        updateTimeDateDsply();    // Format Time/Date for display
61        delay(100);               // Give it some time to settle
62        updateDisplay();          // Write out to display
63        currentCmd = 0;     // Commands 4,3,2,1 are button presses and get cancelled
64        newCmd =0;          //   once implemented.
65        delay(100);         // Wait for settling again
66      } else {                    // For App Display #1
67        currentDisplay = 1;
68        updatePartDsply();
69        delay(100);
70        updateDisplay();
71        currentCmd = 0;      // Commands 4,3,2,1 are button presses and get cancelled
72        newCmd =0;           //   once implemented.
73        delay(100);
```

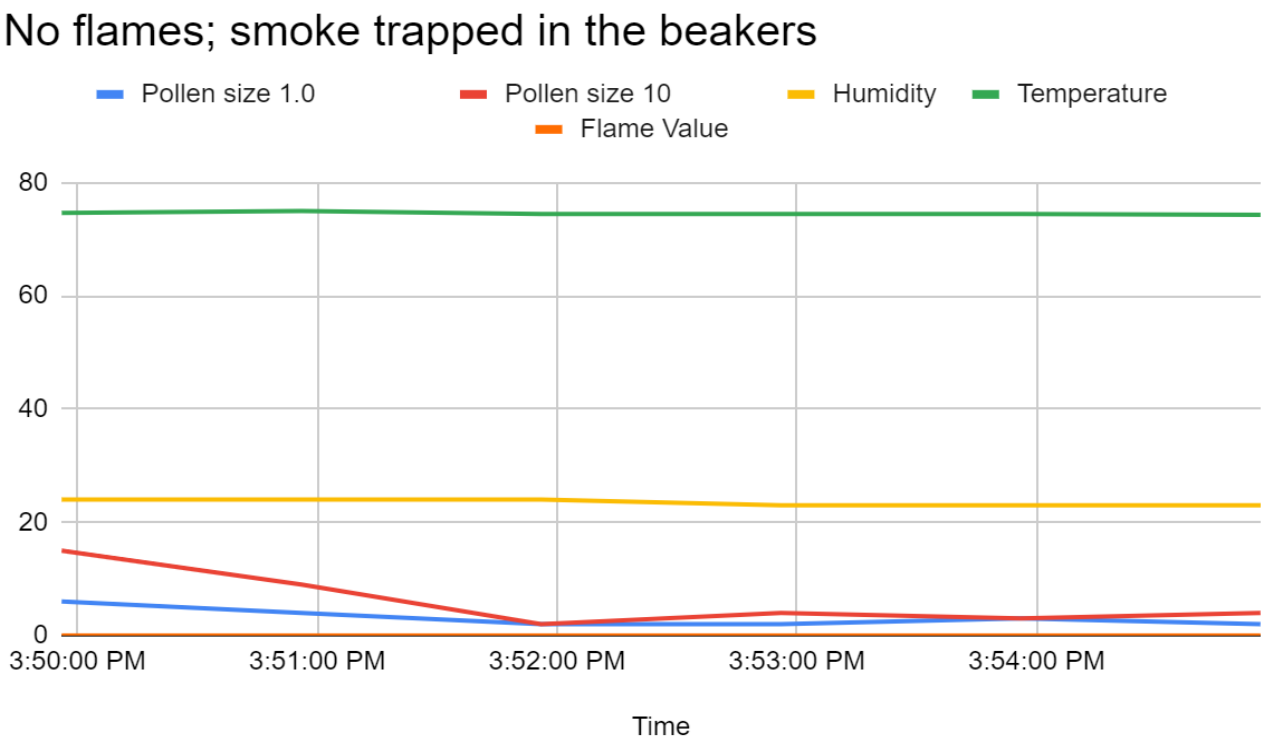
Test model and evaluate

Testing



Many tests were performed with Smokey, including exposing it to the burning flames in the chemical fume hood. The performance of Smokey was recorded and saved onto the SD card and then analyzed and presented graphically in the charts below. Each line within the charts represents a value that was recorded by Smokey, the blue line represents the pollen size 1.0 values, the red line represents the pollen size 10 values, the yellow line represents the humidity values, the green line represents the temperature values and the orange line represents the flame values of this experiment.





In addition to the actual flames, Smokey was also used to detect light from an infrared heater. The chart below displays the efficiency of the range of the infrared sensor from different distances. It was able to easily get a reading all the way up to 90 feet. The experiments were performed in the classroom as well as school corridors. This is proportional to the inverse square law. Every time the distance was doubled, the efficiency of the infrared sensor dropped to ¼. Due to the limitations of the number of characters on LCD display and the Bluefruit app used on cell phones, the maximum value of detected infrared light displayed by Smokey is 99.

Troubleshooting

Before and after the many tests, bug fixes were added to the C++ code and libraries of Smokey. The occasional bugs were observed in the values displayed on the LCD panel. One initial bug was due to an error caused by the fluctuation of voltage in the faulty wiring within the battery pack. Other bug fixes included sorting and separating the code into three different sections that made it more readable and easier to troubleshoot. Another bug that occurred within the LEDs was caused by the pin values being flipped for the smoke, humidity, and temperature LEDs. A recurring issue was with the CO2 module as it would not update to display the correct value. As of this time, most of these bugs have been resolved.

Redesigning

As previously mentioned, Smokey has been redesigned several times as improvements were made to it, leading to three different prototypes. Each next prototype was an improved version of the previous one with some extra components added to it. The idea for adding an infrared sensor to the last prototype of Smokey came from the satellites that use a parabolic dish. As for Smokey, it was built around the size of its circuit board and components. This was done in order to have enough space to fit everything within it and to have the ability to mount it in various places, including the actual forest, buildings, and homes.

Computational/Mathematical Model

The code consists of more than a thousand lines split into 3 different libraries. The first library calls upon the setup process for Smokey's modules and Smokey's DO functions and starts a watchdog timer. The watchdog timer keeps Smokey's software from freezing for long periods of time as it counts out each second. After a series of 10 seconds, a flag is set and the watchdog timer counts to a minute and sets another flag. If the software is still frozen after 5 minutes, the software resets. This unfreezes Smokey's software, and the data still stays saved onto the SD card.

```
ISR(TIMER5_OVF_vect){   // interrupt service routine every half second
  TCNT5 = timer5_counter;   // preload timer (from arduino app notes)
  Flag05sec = TRUE;
  if(timer5_repeat == 0) {  // need to skip every other call because
      timer5_repeat =1;     //  this timer runs every half second.
    } else {                // Only come here once per second.
      timer5_repeat = 0;    // Clear "repeat" flag.
      Flag1sec = TRUE;      // Set the 1sec flag.
      Count10sec = (Count10sec + 1)%10;       // Update 10 sec counter
      if (Count10sec == 0) Flag10sec = TRUE;  // Set 10sec flag on overflow
      Count1min = (Count1min +1)%60;          // Update 1min counter
      if (Count1min ==0) Flag1min = TRUE;     // Set 1min flag on overflow
      CountDisplay = (CountDisplay +1)%DisplayTime; // Update Dispaly counter
      if(CountDisplay == 0) FlagDisplay = TRUE;     // Set display on overflow
      WatchDog += 1;                          // Update watchdog counter
      if(WatchDog >= WatchDogDly) {           // If watchdog hits delay, reset processor
                                              // delay set to 10 at start, up to 300 at setup
        Serial.println("WDog Rst in 1 sec");  // Print watchdog warning
        delay(1000);                          // Pause for printint
        resetFunc(); // if WatchDog not reset in loop for 5min then reboot machine.
      }
    }
```

The second library is the main library of Smokey's coded software that stores each of the main mathematical parts of Smokey. It includes the main functions for the stepper motors that move the parabolic dish and an infrared sensor, such as Pan_ang, Pan_ang_wh, Pan_ang_fr, Tilt_ang, Tilt_ang_wh, and Tilt_ang_fr. The "wh" stands for "whole" and the "fr" stands for fraction. Fraction is used to display the remainder of the tilt or pan angle as a character. Custom bits are stored in order for fractions to be displayed on the LCD and Bluefruit app.

```
584 void computeTilt(){ // calculate tilt angle from steps, make whole and fractional parts
585   Tilt_ang = ((float) Tilt_steps)*0.125; // Need to measure degrees per step
586   Tilt_ang_wh = int(Tilt_ang);
587   Tilt_ang_fr = abs(int(10.0*(Tilt_ang - float(Tilt_ang_wh))));
588 }
589
590 void computePan(){ // calculate pan angle from steps, make whole and fractional parts
591   Pan_ang = (((float)Pan_steps)/Steps180)*180.0;
592   Pan_ang_wh = int(Pan_ang);
593   Pan_ang_fr = abs(int(10.0*(Pan_ang - float(Pan_ang_wh))));
594
595 }
```

The next part in the second library includes values and functions for the LEDs. If the average value of smoke is above the warning value, the corresponding smoke LED turns on. If the average temperature is greater than the warning value and at the same time the humidity level is lower than the set value, then the corresponding humidity and temperature LED lights up. The infrared sensor's whole number displayed on the LCD screen and Bluefruit app represents the value of the strength of the infrared light, while the fraction displays the angle that the infrared light was spotted at. The third library includes all of the DO functions.

```
1230 void computeFlm(){
1231   Flm_read = analogRead(Flm_pin);               // Get current output from flame detector
1232 //+++++ changed numbers here to make it go to zero when no flame
1233   Flm_det = limit( ((800 - Flm_read) / 7),0,99);// Scale it to 0 to 99
1234   if(Flm_det > Flm_max) {     // If its the biggest
1235     Flm_max = Flm_det;        //   save the value
1236     Flm_ang_max = Pan_ang_wh; //   and the angle.
1237   }
1238 }
```

Conclusion

As a result of this work, the team has developed a prototype of a working robotic fire and fire weather conditions measuring system that has been tested in a variety of conditions and circumstances. If this prototype were mass produced and networked in the way that all of them would communicate with each other via Bluetooth, they could be then installed in the forest to serve as the wildfire protection system. Each such Smokey could use a battery that would charge during the day by a small PV solar panel and keep powering Smokey also during the night.

While working on this project the team valued the experience gained. Some team members learned how to design a circuit board and C++ programming while others learned more about video editing and presenting in front of an audience. This project helped each team member discover his/her natural interests, skills, and abilities, as well as to make their mind about the major in the college.

Collaboration

Roles / Responsibilities

Although the team shared the responsibilities and roles working on the project, their contribution depended on the certain natural skills that each team member already had. One team member was involved in research while another was in charge of tracking deadlines and putting presentations together. The team leader was mostly in charge of computer programming and working directly with the mentor, and another team member was in charge of hardware. All the team members were meeting after school to work together using their skills. The main responsibility of the team, as a whole, was to be present for meetings, presentations, and to be

responsive to any communication involving the project.

Contributions

The team leader, Zachariah Burch, was regularly meeting with the mentor, and worked on designing circuit board, the C++ programming, and on the parts of the report that included the components of Smokey, code, graphs, and images.

The team leader's assistant, Isel Aragon, has been putting all the presentations together, regularly stayed in after school meetings, and worked on the report together with the team leader and their teacher.

The team member, Britny Marquez, has been participating in after school meetings and greatly contributing to the presentations of the project.

Another team member, Daniel Leon Leon, together with the team leader, was meeting with the mentor and worked on designing the circuit board.

Works Cited

"Critical mistakes and miscalculations by federal employees caused devastating New Mexico wildfire, report finds." *CBS News*, 21 June 2022, https://www.cbsnews.com/news/new-mexico-wildfire-federal-employees-critical-mistakes-miscalculations/. Accessed 10 March 2023.

Lopez, Tommy. "Las Vegas has about 30 days left of clean water." *KOB 4*, 18 August 2022, http://www.kob.com/new-mexico/las-vegas-has-about-30-days-left-of-clean-water/. Accessed 9 October 2022.

"Nmsnf Calf Canyon Information | InciWeb." *InciWeb*, 19 April 2022, https://inciweb.nwcg.gov/incident-information/nmsnf-calf-canyon. Accessed 4 April 2023.

Popovich, Nadja. "New Mexico Wildfires: Mapping an Early, Record-Breaking Season." *The New York Times*, 1 June 2022, http://www.nytimes.com/interactive/2022/06/01/climate/new-mexico-wildfires.html. Accessed 10 November 2022.

Rodriguez, Vince. "New Mexico wildfires in 2022." *KOAT*, 11 June 2022, http://www.koat.com/article/new-mexico-wildfires-calf-canyon-2022/40257066. Accessed 10 September 2022.

Romero, Simon. "The Government Set a Colossal Wildfire. What Are Victims Owed?" *The New York Times*, 24 June 2022, https://www.nytimes.com/2022/06/21/us/new-mexico-wildfire-forest-service.html. Accessed 16 November 2022.

Weeden, Meaghan. "How Trees Clean the Air." *One Tree Planted*, 5 November 2021, https://onetreeplanted.org/blogs/stories/how-trees-clean-air. Accessed 10 November2022.

Predicting Heursitscs Related to the Controversiality of a Social Media Post

Henry Tischler, Gene Huntley - Team Members

Jenifer Hooten - Sponsoring Teacher

Team #10

Academy for Technology and the Classics

April 2023

Contents

1 Executive Summary

Social media platforms are currently confronted with a large amount of controversial content, which carries significant consequences for the emotional health of their users [1]. Without an effective way to classify this content, social media platforms face difficulty when trying to limit such content.

In this project, we attempt to use deep learning to predict the metrics (heuristics) that a social media platform would use to categorize a post as controversial before the post is even reacted to - simply from the natural language content of the post. By doing this, a post can be classified as controversial, before it's already had a negative impact on a social media platform.

To achieve this, we made use of a BERT model, with our custom layer trained to identify content as controversial. With this, we were able to make use of a pre-trained understanding of natural language, while still creating a model relevant to our task, achieving great success on what would otherwise be a limited dataset.

Through our model, we were able to achieve a high degree of accuracy, successfully demonstrating the potential of our approach in detecting controversial content, and allowing this detection to happen much quicker than otherwise possible.

2 Project Problem and Solution

As nearly any frequent user of a social media platform can attest to, most social media platforms contain a large amount of controversial content. The constant exposure to such content, for many, can harm their emotional health [1].

To reduce the negative impact of controversial content, it may be useful to reduce one's exposure to this content. However, to reduce the exposure of controversial content, it is of course necessary to have the capability to classify content as controversial or non-controversial.

Controversial content is relatively easy to recognize after it's been published, by analyzing how the content is being reacted to. However, this of course has its limitations, as the content has to have a negative effect before it's controversiality can be recognized.

In this project, we use deep learning techniques to predict heuristics related to the controversiality of a social media post, simply from its natural language content. We do this by learning the features of the natural language which correspond with a post's controversiality.

3 Model Structure

To effectively work with natural language, we opted to use a deep learning model. This model is able to take the textual content from a variety of social media posts and learn what within this text content correlates with a controversial post.

3.1 Use of Heuristics

A heuristic is, essentially, a metric that itself may not correlate exactly with what needs to be predicted, but can be used as a part of an informed judgment.

In this project, we make use of heuristics related to post controversiality. Specifically, we use the like-to-retweet ratio of a Twitter post, though other similar metrics exist. These metrics, in many cases, can give us an idea of the controversiality of a post. For example, if a post has many retweets, though not many likes, there's a possibility that users felt the need to give their thoughts on a post, instead of simply expressing their agreement. Thus, a low like-to-retweet ratio is possibly an indicator of controversial content.

Unfortunately, without a large annotated dataset, it's impossible to validate if these heuristics truly correlate with the underlying controversiality of a post. However, while we cannot, in this project, predict controversially itself, by predicting these metrics using nothing but the content of a Tweet, we can demonstrate the potential of using these heuristics to predict the controversiality of a social media post ahead of time, avoiding the need to inflict the negative impact of such content before it can be categorized.

Additionally, while some work has been done in the past into the training of models which use annotated data, these studies often suffer from the relatively low sample size afforded by manual annotation [4].

3.2 BERT Modeling

The simple way to view BERT modeling is as a way to leverage a deep understanding of natural language, from an extremely large, pre-trained model, and then narrow the capabilities of the model to exactly your task, with relatively minimal training.

Essentially, the BERT model makes up the first layers of the neural network. We can then add our layers onto the end of that neural network, and train those layers to our specific task. This way, our model doesn't have to learn about the English language itself and can focus entirely on our specific task.

To provide some more technical detail, "BERT" stands for "Bidirectional Encoder Representations from Transformers". The model can successfully consider the bidirectional context of a word. That is - instead of just providing an idea of what a word means in its most general context, such as what's done with a word embedding, or providing context for a word based on the words preceding it. [2] Instead, BERT can effectively consider the context of a word within the sentence it's in, providing a very rich and useful understanding of natural language [3], which can be applied to a narrower context.

3.3 Our Own Layer

The BERT model we are using, of course, doesn't have any understanding of how to associate the natural language content of a social media post with controversiality. To do this, we needed to add our custom layer, which takes input from the output neurons of the BERT model and outputs its predictions of the metrics which we use to approximate controversiality.

For the primary metric used, like retweet ratio, we created ten different classes for the metric, each of which represents a percentile range of the metric across our training dataset.

So, for example, the first class would represent all like/retweet ratios in the bottom [0-10%), the second class would represent all like/retweet ratios in the [10%-20%) percentile range, and so on.

By using these percentiles, we can make our classes very close to equal in presence, certainly within our dataset, and gain a relatively detailed idea of the predicted metric.

4 Dataset Collection

To train our model, we, of course, need a dataset to use. For this project, we were unable to find a pre-existing, publicly available dataset that would work well for this project. For our dataset, we had three main constraints.

1. The dataset needs to be as large as possible, to allow for the most effective training of our network.

2. The dataset needs to have as much information as possible relating to the public reactions to a tweet.

3. The dataset needs to consist of tweets that have been reacted to as widely as possible, to provide the richest possible reaction metrics.

To fill these constraints, we collected a dataset of Tweets, using a Python program written ourselves. This program aggregated the tweets from the top 50 Twitter accounts, by using the Twitter API to scrape these Tweets over a relatively long period, and compiled all of these Tweets into a consolidated dataset.

In this dataset, we included every possible metric which we could relate to the public reaction to a Tweet. In the end, we used three metrics to approximate the controversially of a post, the first metric is the primary one utilized in this experiment.

In total, we were able to collect 156,677 tweets, each of which was used to train the neural network.

5 Solution Validation Methodology

To validate that our solution performs effectively, it's necessary to perform a test of our model. The testing of our model, following extremely standard Machine Learning methodology, relies on testing the accuracy of our model on a validation dataset. This dataset would not have been shown to the model during training. As such, testing the accuracy of our model on a validation

dataset provides a very close approximation of the performance of the model on data it will encounter when deployed.

In our case, the validation dataset used was taken from our total dataset, representing 25% of our entire dataset.

To gauge the accuracy of our model, we used two different metrics. The first, categorical accuracy, is a fairly standard accuracy metric in the field of machine learning. For this model, this simply represents the accuracy with which the exact category is predicted.

However, unlike most classification models, with our particular model, not all misclassifications are equally incorrect. A misclassification in a nearby percentile should be represented as being more accurate than one in a further percentile. As a result, we also used the "average separation" metric, which represents the average distance between the selected percentile and the correct one.

6 Solution Validation Results

Our model, after three epochs of training, was able to achieve a validation accuracy of *29.94%*, meaning that 29.94% of the posts were categorized into the correct percentile. While not perfect, this level of accuracy still gives a strong indication of the potential in the model, performing much better than the *10%* accuracy that would be archived without any learning. Additionally, the trend from the three epochs of training, seen in Figure 1, seemed to be fairly linear, showing the potential for this accuracy to improve should more compute

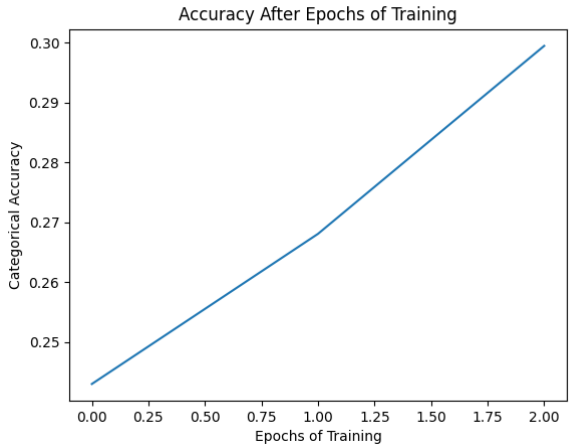


Figure 1: A graph showing the categorical validation accuracy of our model as training progressed.

resources be invested into training the model.

Additionally, we were able to achieve an average separation of 20.7%. This means that, on average, the percentile assigned to each post deviated from the ideal percentile by 20.7%. This, while of course not precisely accurate, shows that this model can be used to give a relatively accurate general idea of how controversial a post is, even when it fails to identify a post's controversiality exactly.

7 Conclusions

In this project, we were able to successfully teach a machine learning model to recognize the signs of controversial content and achieve a level of accuracy that, while not perfect, still allows for the model to be useful in the detection

of controversial content. Through the use of natural language processing, we have shown that it's possible to predict the metrics which can indicate controversiality. Through further work and exploration into different natural language processing techniques, it's possible that the accuracy could be improved even further.

8 Future Work

To expand on this project, given more time and resources, there are several improvements and expansions which could be made. The largest improvement to this project itself would be to train the BERT model for more epochs, to ascertain a more complete idea of the peak performance that can be archived. Additionally, collecting more data from Twitter, over a larger period, would allow for more performance to be achieved. And, of course, the dataset could be expanded to other forms of context, such as spoken language or pictures with text. Finally, given the resources to annotate enough content, the correlation between the statistics

9 Acknowledgements

We would like to provide the following acknowledgments to the individuals and organizations who assisted us with this project:

1. Our Teacher and sponsor, Ms. Hooten, for her support and facilitation of the challenge of the year.

2. Nicholas Kutac for Reviewing our Interm Report and Providing Feedback on the Project

3. The Supercomputing Challenge for facilitating the challenge and providing the associated support with the project.

References

[1] William J. Brady et al. "How social learning amplifies moral outrage expression in online social networks". In: *Science Advances* 7.33 (2021), eabe5641. DOI: 10.1126/sciadv.abe5641. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.abe5641. URL: https://www.science.org/doi/abs/10.1126/sciadv.abe5641.

[2] Jacob Devlin and Ming-Wei Chang. "Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing". In: (2018). URL: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html.

[3] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[4] Marzieh Mozafari, Reza Farahbakhsh, and Noël Crespi. "Hate speech detection and racial bias mitigation in social media based on BERT model". In: *PLoS ONE* 15 (2020).

Tracking Cislunar Orbits

New Mexico
Supercomputing Challenge
Final Report
April 5, 2023

Team 13
La Cueva High School

Team Members:

- Hadwyn Link
- Ximena Serna
- Kylen Reyner

Teachers

- Jeremy Jenson

Project Mentor

- Mario Serna

A Brief Summary

Space is becoming more and more accessible and with that, more and more satellites and resulting debris are put into orbit. Debris can be catastrophic to equipment in space and not only to the equipment, but also the companies making the equipment as it can cost upwards of $600 million dollars. Our goal is to create a “cheap” and effective system of satellites to track space debris (also known as space junk) in order to help bring awareness and allow critical equipment to avoid devastating collisions.

First, the cost program takes inputs including satellite weight, run time, quantum efficiency of the camera, camera lens radius, fov (field of view), shutter speed, and finally the number of satellites. The program can then calculate the cost by taking the total estimated cost of the satellite parts and adding it to the cost of the run time and cost to orbit. A total cost is outputted and the camera specs of the satellite are sent to the luminosity program.

The luminosity program checks if there is any debris in the field of view of the satellites. If there is, the program uses the known distances from celestial bodies to calculate the angle of the debris to the satellite which is then used to find the phase of the debris according to the perspective of the satellite. Then, the program calculates the amount of light going from the sun to the debris by using the inverse square law. That number is then multiplied by the surface area of the debris, which is assumed to be a perfect sphere of a diameter equal to 1 kilometer, and multiplied again by the debris’ material reflectivity, which we have assumed to be approximately 0.14 percent. The outcome of that problem is then multiplied by the phase of the debris calculated earlier. The inverse square law is brought to light once again when it is used to find the amount of light from the debris that enters the lens of the satellite. Here, factors including quantum efficiency, lens surface area, and shutter speed are taken into account, turning the luminosity that reaches the satellite into joules.

The information from the program is then sent to the simulation software which is used to simulate the debris path saving its position and an appropriate time stamp to a file. Based on the orbit and position we can find the debris’ velocity making it possible to predict its future path. This data can then be used to alert satellites and other equipment in the predicted path of the debris so that measures can be used to avoid the debris, therefore avoiding catastrophic collisions.

With all three sections of the code working in unison, we are able to track the path of the debris’ orbit allowing for the avoidance of $600 million dollar collisions. By stopping these collisions we can also keep more debris from being created by resulting collisions therefore making space safer than ever.

Terminology

The following are important terms that will be used throughout the report:

- **Cislunar space** - the space within the Earth's and the Moon's gravitational field, excluding Earth itself.
- **Satellite vs debris** - the satellite in this case is referring to the man-made cube satellite. Debris is the potentially hazardous object that the satellite is observing.
- **Cube Satellite** - a cheap, small satellite shaped like a cube. For our purposes, this is the satellite we would most likely be working with in real life.
- **Law of Cosines** - a mathematical concept which states that, given all three side lengths of a triangle, we can find all unknown angles of any triangle.
- **Inverse Square Law** - a law of physics which states that the amount of light decreases exponentially the further you get from the source of the light (see this article for more information).
- **Luminosity** - a measure of radiated light from a light source, written in joules per meter squared per second (see this article for more information).
- **Quantum Efficiency -** the percentage of photons that are actually registered by a camera (see this article for more information).
- **Uniform circular motion** - celestial body programmed to uniform orbit in space without the interference of non-central celestial bodies.
- **Elliptical motion** - celestial body programmed to move with its orbit dictated by 2 or more celestial bodies.
- **Geosynchronous orbit** - an orbit path at an altitude of 37,000km and stays above the same spot on Earth.

The Problem

As space becomes more accessible and more satellites are put into orbit, the odds of a collision between a satellite and a piece of debris increase significantly. A collision may cause a significant amount of damage - a 'cheap' satellite costs several hundred thousand dollars to get into orbit, and the cost increases exponentially from there. According to the GAO(Government Accountability Office), there are almost 5,500 satellites in orbit as of spring 2022, and many

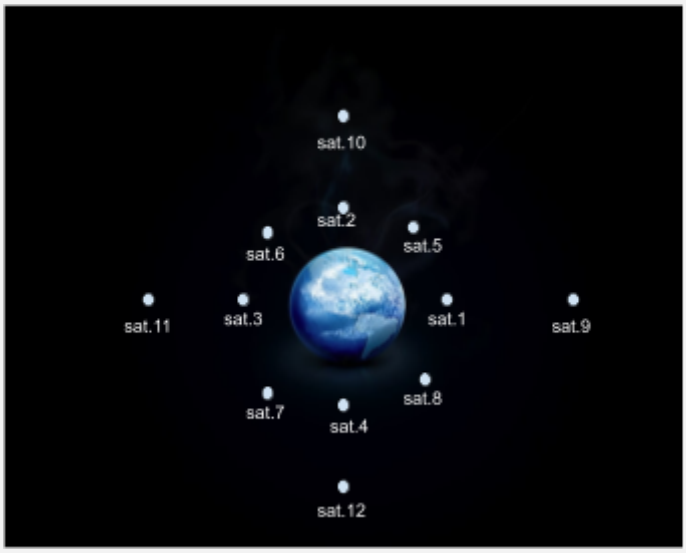
more will be launched over time: it is estimated that an additional 58,000 satellites will be launched by 2030. Immediate development of a tracking system is necessary to avoid the severe property damage and destruction of satellites that may be crucial to things such as national security or scientific research.

The Solution

Our goal is to find the optimal satellite configuration to track objects in cislunar orbit taking into account cost production, maintenance, and view of debris. Finding a cost-effective tracking system for objects in geosynchronous orbit to cislunar orbit can help prevent collisions between debris and satellites. With that in mind, here are the steps that our program goes through to achieve this:

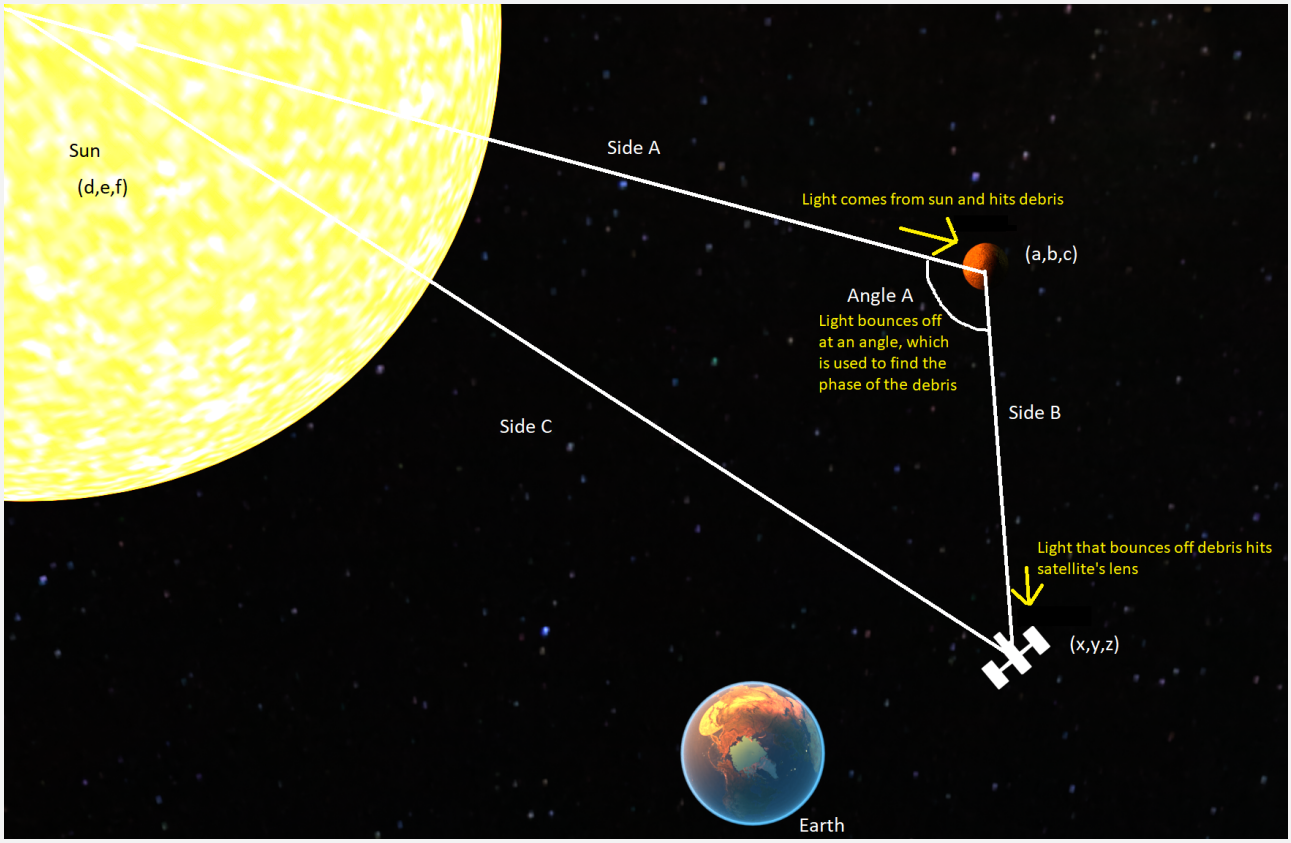
- **The simulation:**

The simulation is composed of two programs that produce the positions for all the satellites and the debris used within our cislunar space. The first program produces the chaotic orbits for the debris one at a time and saves the positions for each time step to a file. The debris orbits generated vary in both position and velocity. The position ranges from 50,000 km to 384,000 km away from the earth. The velocity runs through an assortment of different magnitudes and directions. The second simulation runs through the satellite positions and changes the velocity to stay in circular orbit accordingly. The location for the satellites runs through eight positions at an altitude of 2,000km (low earth orbit) above earth, and four orbiting at 5,000km. The velocity changes accordingly so the satellite orbits in near circular motion. The accuracy of these programs depends heavily on the dt, or 'time step' being used. The greater the time step, the less accurate the data will be.



https://www.youtube.com/watch?v=2h8doa_Indg*(video of the physics simulation running for 6 months, at a timestep of 10 seconds)*

https://www.youtube.com/watch?v=e2lvFIXgcWE(video of the physics simulation running for 6 months, at a timestep of 100 seconds)



- **<u>The Luminosity:</u>**

The luminosity section of the code runs after the simulation data has been compiled. Given the positions of the satellite and debris at various points in time, as well as the specifications that are generated for the satellite's camera, the code iterates through each satellite orbit and each debris orbit and works through over 1,000 different combinations of debris and satellite to find the total light gathered by each satellite in each situation.

The program starts by using the coordinates of each celestial body(shown as [x,y,z], [a,b,c], and [d,e,f] in the diagram) and finds the side lengths of the triangle between the sun, the satellite, and the debris(shown as side A, B, and C in the diagram). Given the side lengths, the Law of Cosines is used to find the angle from the debris to the satellite(shown as angle A in the diagram) which is used to find the phase of the debris from the perspective of the satellite.

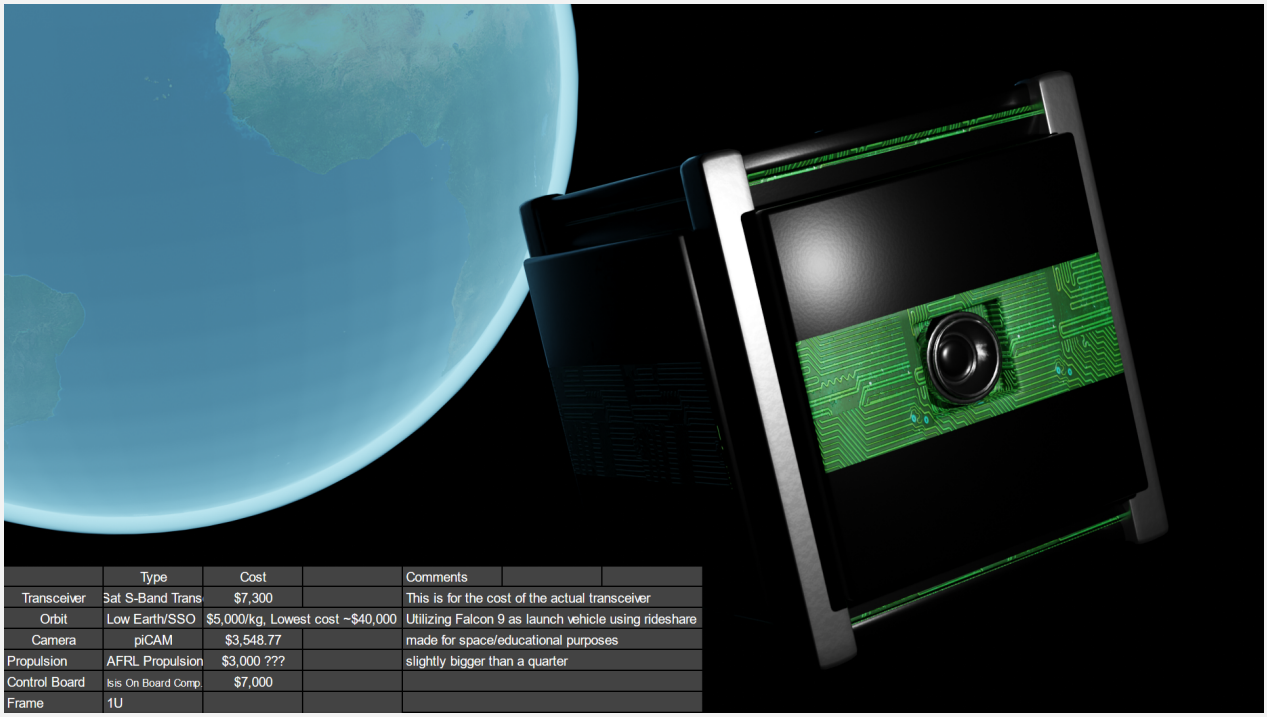
Before calculating how much light from the debris the satellite sees, we need to find whether or not the satellite can see the debris in the first place. To do this, the satellite's field of view is taken into account, assuming it is facing directly away from the Earth. If the angle of the

debris falls within the angle range between the two edges of the satellite's view, it can be seen. If not, the rest of the loop is skipped. If the debris can be seen, however, the bulk of the calculations begin.

First, the amount of light from the sun to reach the debris is calculated with the Inverse Square Law. This number is multiplied by the debris surface area(assumed to be a perfect sphere 1 kilometer across for ease of calculation), multiplied by the reflectivity of the debris' material(around 14%) and the phase of the debris that was calculated in the early stages of the loop. The inverse square law is once again used to find the amount of light from the debris that reflects into the lens of the satellite, where the quantum efficiency, lens surface area, and shutter speed are taken into account, turning the luminosity that reaches the satellite into joules. After this is done, the program writes the results to a file with its timestamp and moves on to the next loop.

- **<u>The Cost</u>**

Our goal is to have a cheap way to track debris in a cislunar orbit. In order to track debris in space, putting satellites into orbit, and close to the same vicinity as the debris, is the best way of accurately tracking it without the interference that a tracking system on the ground would have to deal with. While satellites are becoming more common as accessibility to space increases, they remain incredibly expensive to build and maintain.



	Type	Cost		Comments
Transceiver	Sat S-Band Trans	$7,300		This is for the cost of the actual transceiver
Orbit	Low Earth/SSO	$5,000/kg, Lowest cost ~$40,000		Utilizing Falcon 9 as launch vehicle using rideshare
Camera	piCAM	$3,548.77		made for space/educational purposes
Propulsion	AFRL Propulsion	$3,000 ???		slightly bigger than a quarter
Control Board	Isis On Board Comp.	$7,000		
Frame	1U			

For example, the James Webb Telescope, although serving a different purpose than our satellites would, cost around $10 billion dollars. This is an excessively large number considering our budget and purpose, so with that in mind we turned to the cubesat.

The cubesat is an innovative way to fit the necessary components of a satellite into a small frame, about 10cm^3, while still being able to effectively accomplish our goals. According to our cost calculations it would only cost about $700,000 per satellite. This cost evaluation takes into consideration things like cost to launch (using a rocket like the Falcon 9), cost to run (mostly sending and receiving data from the satellite(s)), and the actual parts being used in the cubesat. With an average lifespan of around one year and a low cost per satellite, compared to the larger current satellites the cubesat is the perfect fit for our needs.

Verification of our data

To verify that our data from the 3-body simulation was accurate, we designed a visualizer in the Unity game engine that would read from the various data sets we generated and show the positions of the satellite and debris in motion, giving us an idea of whether or not our calculations were accurate.

To verify the luminosity calculations, we used the visualizer to see whether or not it was feasible for the satellite to see the debris given its field of view, as well as references from previous physicists who attempted similar things, examples being finding the lux, which can be translated into luminosity, of the moon (see this article for more information) and then plugging in variables that matched the situation they worked with to see if we could replicate their results.

The costs for the satellite were taken from various vendors of satellite parts, as well as NASA's own projection of the average costs for a satellite to be built and launched. Since a great deal of the work for this part of the code was in researching typical costs per part,

Conclusions

link to the generated data

While comparing the results of the luminosity data, some interesting patterns emerged. For instance, the first 3 satellite orbits were able to cover each other's blind spots relatively well. They also occasionally overlapped viewing times, which allows for better tracking of the debris. However, with such a small setup, there were still large blind spots spanning hours or sometimes days. A fourth and fifth satellite at complementary orbits would help keep more than one satellite tracking the debris at a time, and also reduce the time where the debris was continuously out of sight. 8 satellites spaced evenly around the globe would have been optimal,

and rotated at slightly different axes to catch debris coming from any unexpected angles. In terms of the amount of light received, it ended up being rather small due to the large distance between the two objects. More expensive cameras with higher quantum efficiency, longer shutter speed, and other types of satellite cameras that don't rely on visible light to locate things would all be helpful in better locating the debris.

Reflection

Our most significant achievement was undoubtedly the 3-body simulation of cislunar space. The amount of fine-tuning and error fixing that it had to go through made its verification a team effort, and also the most intricate part of our project.

We could also have done quite a few things better during the process. Overall, we lacked organization and throughout the project had trouble communicating what needed to be done by whom. These flaws in our strategy left us constantly scrambling to meet deadlines, which resulted in last-minute crunch time on more than one occasion. The fact that everything in our project relied on the progress of a central, sometimes finicky piece of code only one person could work on at a time further hindered our efforts since often two people would need to wait in order for one person to fix the issues before anyone could progress. During the start of the project we were also unable to come to a solid decision of what to do, effectively wasting two months of the time we had. In our future attempts at the challenge, it would be advantageous to have a formal repository for code so we can better share in the creation of code, rather than having most of the programming be offline. The final mistake we made was our last minute realization that an increase in delta time(dt) caused the satellite positions that we had generated ended up building so much percent error that for the past 3 months, none of the data was valid. We were able to generate a couple more accurate simulations near the end with a lower dt, but by that point we didn't have the weeks it would take to compute the billions of lines of data we needed.

Sources Used

"AFRL Propulsion Unit for CubeSats | VACCO Industries." *CubeSat Propulsion Systems | VACCO Industries*, https://cubesat-propulsion.com/propulsion-unit/.

"Camera - piCAM - Digital CubeSat Camera." *satsearch*, https://satsearch.co/products/skyfox-labs-picam-digital-cubesat-camera.

Chenciner, Alain. "Three body problem." *Scholarpedia*, 3 October 2007, http://www.scholarpedia.org/article/Three_body_problem. Accessed 4 April 2023.

"CubeSat Launch Costs." *SatCatalog*, 10 December 2022, https://www.satcatalog.com/insights/cubesat-launch-costs/. Accessed 4 April 2023.

"CubeSat S-Band Transceiver." *NanoAvionics*, https://nanoavionics.com/cubesat-components/cubesat-s-band-transceiver/. Accessed 4 April 2023.

"Distance to the Moon." *NASA*, https://www.nasa.gov/sites/default/files/files/Distance_to_the_Moon.pdf.

Fizell, Zack. "Orbital Mechanics: The Three-Body Problem | by Zack Fizell | ILLUMINATION." *Medium*, 20 March 2022, https://medium.com/illumination/orbital-mechanics-the-three-body-problem-90be80e4711 3.

"Home." *YouTube*, https://academic.oup.com/astrogeo/article/58/1/1.31/2938119?login=false.

"Home." *YouTube*, https://www.cubesatshop.com/product/isis-on-board-computer/.

"Inverse-square law." *Wikipedia*, https://en.wikipedia.org/wiki/Inverse-square_law.

"The Law of Cosines." *Math is Fun*, https://www.mathsisfun.com/algebra/trig-cosine-law.html.

"Quantum efficiency." *Wikipedia*, https://en.wikipedia.org/wiki/Quantum_efficiency.

Stress Anxiety Monitor (SAM)

New Mexico
Supercomputing Challenge
Final Report
April 4, 2023

Team 27
Monte del Sol Charter School

Team Members:
Gabriella Armijo
Emlee Taylor-Bowlin
Teacher Sponsor:
Rhonda Crespo
Project Mentor:
Mark Galassi

Contents

1 Executive Summary

We have all lost someone prematurely due to a tragic suicide. It often leaves us with a feeling of dread and many "what if"s. What if I saw the signs, what if I had stopped them, what if I had texted them that night? This isn't a burden you should have to bear. Imagine a simple program that could read some text snippets and alert you if you or a friend needed to reach out to a professional? In this model we used machine learning to create an AI that will do just that.

SAM (Stress Anxiety Monitor) is an AI built off of the Random Forest Classifier model. It is trained with data from the "dreaddit" data set (Turcan & McKeown, 2019).

SAM uses two different training and processing approaches: the Bag of Words and Intensity Value methods. The Bag of Words method takes a count of every word in the text snippet. It then feeds this data into a sparse vector, and any word in the English dictionary that it does not find becomes a zero. The words are then processed by frequency, so redundant words like the, as, and a are not counted as heavily. The Intensity Score method makes a weighted score of the "worrisome" key words that show up in the text.

This word data is then fed into a machine learning Random Forest Classifier that processes the data and makes accurate predictions about a person's risk. We found SAM to be accurate 99% of the time on the in-sample set. SAM is 93% accurate on out of sample data. Hyperparameter tuning is still needed for SAM to be as accurate as possible.

As of now SAM can be used to help friends and family recommend people to professional help. We would like to turn SAM into a simple webpage that could process screenshots and take text input. SAM can help reduce the total number of teenage suicides in New Mexico, and keep our communities safer and happier.

2 Motivation and Plan of the Work

2.1 Suicide

Suicide is the #2 leading cause of death among teenagers and young adults worldwide (D'Hotman et al., 2020; "Suicide statistics and facts. SAVE.", 2021). New Mexico is currently ranked #4 in the nation for states with the highest suicide rates. ("America's Health Rankings." 2020) New Mexico's current teen suicide rate is 24 deaths per 100,000 adolescents ages 15-19 a year ("Suicide rates by state 2023." 2023). This results in about 513 teen suicides a year.

Suicide is preventable with intervention; however, it is hard to intervene when people don't openly talk about their feelings and it is easy to miss the warning signs. In this day and age many teens and young adults spend a majority of their time chatting on social media or via text messaging. Cries for help may be easily masked in this dull form of communication. Being dependent on technology means that many teens are afraid of reaching out and talking to someone in person for help, meaning the only way they can convey their emotions is through texts and many hope that someone will pick up on their cry for help.

With the help of machine learning, however, suicide prevention may be easier. We created an AI named SAM that is able to go through social media posts and/or text messages looking for trigger words that may indicate that a person may be showing signs of high stress that may lead to suicide. We hope SAM will lead to early detection and prevention of suicide.

2.2 Project Goal

The goal of this project is to accurately determine if a person may be at high risk of suicide or self-harm based on messages that people share online. Our hope is that SAM can be applied to help notice warning signs early on in things as simple as a text message.

3 Model and Methods

3.1 Obtaining and Preparing Data

Our data set is publicly available on the Kaggle web site:
https://www.kaggle.com/datasets/adtysregita/stress?select=stress.csv

This data set initially contained 2,343 rows and 116 columns in a csv file. All 2,343 rows were usable and didn't contain any missing values. We only use 4 of the columns: the row number, the text, the label, and the confidence.

text	id	label	confidence
He said he had not felt that way before, suggeste	33181	1	0.8
Hey there r/assistance, Not sure if this is the ri	2606	0	1
My mom then hit me with the newspaper and it	38816	1	0.8
until i met my new boyfriend, he is amazing, he	239	1	0.6
October is Domestic Violence Awareness Mont	1421	1	0.8

Figure 1: Snippet of data from worrisome messages.

The label is a binary label that codes for stress or no stress. A label of "1" indicates stress, whereas "0" does not. These numbers were provided by professionals based on the text for that row. Confidence is another professional-provided value that indicates the confidence the expert holds in that label. At this time SAM does not use the confidence metric.

This data is then finally ready to feed into the Random Forest Classifier that we wrote for this project, using the *scikit learn* framework. (Amadebai, 2022; Kessler et al., 2016)

3.2 An Introduction to Random Forest Models

Random Forest Classifiers and Random Forest Regression are both methods of machine learning. Random Forest models take numeric values and feed them into decision trees. The trees then decide where this data continues and then break into two or more trees. They inevitably end up as decision leaves. These leaves are the final output of the decision and are also numeric values. In the case of regressors, they can be a range, why classifiers give a binary value.

Figure 2 shows an example of a Random Forest Regressor. It allows for many different outputs like apple, cherry, or strawberry. It makes it's decisions based off of size.

Figure 2: Decision based off a Random Forest Regressor model. Image borrowed from:
https://developer.nvidia.com/blog/accelerating-random-forests-up-to-45x-using-cuml/

The following figure is an example of a Random Forest Classifier. It can only provide a binary output, in this case yes or no. Our model uses a classifier to give the output of stress or no stress.

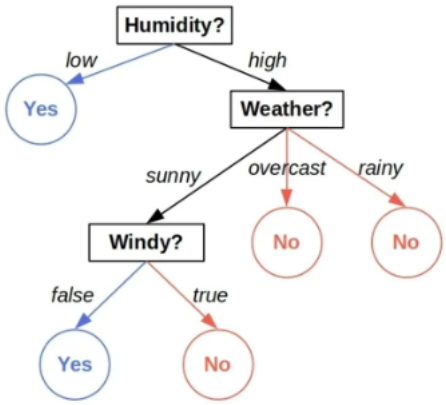


Figure 3: Example of random forest classifier.

3.3 Implementation of a Random Forest Classifier

We used a Random forest Classifier as our model type. It produces a 1 to indicate stress or a 0 to indicate no stress. Many programmers try different model types: we deduced that Random Forest Classifier by far is the most accepted model for natural language learning. We manipulated two different ways of processing the data, and before it was fed into the model.

3.4 Intensity Score Method

The first method of data processing was the Intensity Score method. We wrote a dictionary of "worrisome" keywords, which are words the literature suggests can indicate stress. Each word was given a weight, and the weighed word occurrences are added.

The result was a training set consisting of a single numeric score for each text snippet, and the "expert label" of stress or no stress for that snippet.

This method makes for a very simple initial implementation, but it lacks nuance: the specific use of word pairs is lost in that single score.

3.5 Bag of Words Method

The Bag of Words (Brownlee, 2019) is far more inclusive and more complex. The Bag of Words method reads through the text snippet and accounts for the total number of times any word from the English dictionary appears. It then scales the numbers by importance. Importance is decided by the number of times each word occurs, so words like the, as, and, then will carry far less importance, and not affect the interpretation.

This method is far more reliable, because it accounts for all possible words. However it does come with the downside of producing a sparse vector. A sparse vector is a vector that contains many zeros, showing no data. It can be useful if we see zeros in spots that would indicate words like suicide, depression, or lonesomeness. It does become a nuisance when zeros are always seen to indicate words like Pneumonoultramicroscopicsilicovolcanoconiosis (considered the longest word in English and is used for a certain lung disease).

4 Software Demonstration

Our code can be found at the following repo:
https://codeberg.org/EmleeTaylorBowlin/SAMonitor

Here is an example of how you could clone it and then run the training and analysis;

we include the output, from which you can see the error estimate and the score on both in-sample and out-of-sample data.

`SAMonitor.py` also produces plots and saves them to disk. You can see in Figures 4, 5, and 6.

Note that the program has a `-help` option if you want to see how to test it in more subtle ways.

```
git clone \url{https://codeberg.org/EmleeTaylorBowlin/SAMonitor.git}}
# Make sure you have the following libraries also installed on your computer
# pandas, numpy, matplotlib, and scikitlearn.
# Begin by running the programs in this order:
cd code/SAM
$ ./SAMonitor.py TnPData/dreaddit-train.csv TnPData/dreaddit-test.csv
Hey! SAMonitor here! We will now train on data and analyze snippets.
# loading training data from: TnPData/dreaddit-train.csv
# in-sample error metric (mae): 0.0014094432699083862
# in-sample classifier score: 0.9985905567300916
# saved confusion matrix to file confusion_insample_bag-of-words.png
# ========= out of sample =========
# out-of-sample error metric (mae): 0.3062937062937063
# out-of-sample classifier score: 0.6937062937062937

APPLY_TO_TEXT: worrisome score: """Things had started so well, but
now, two years later, I sit listening to Janacek's quartets - the
Intimate Letters and the Kreutzer Sonata - and I find myself feeling
desperate, and wondering if my life will ever be what I had dreamed it
would be.  Well-meaning people offer me words of wisdom, like 'it gets
better' and 'just navigate the terrain you are in' and 'do not feel
that you are below the standard everyone else is at'.  I know they are
```

```
right and I hope that some day I can take their advice, but I cannot
yet promise that I will."""  =====> [1] i.e. DO_WORRY

APPLY_TO_TEXT: worrisome score: """I was walking down the street,
singing do wa diddy diddy dum diddy do, and when I turned the corner I
saw someone walking with a book by Murakami, my favorite author of
Japanese magical realism.  We started talking and shared a lot of
interests.  This was a year and a half ago, and now we're together
almost every single day, singing do wah diddy diddy dum diddy do."""
=====> [0] i.e. NO_WORRY
```

5 Results

Our final model accurately predicts if someone is at high levels of stress based off of snippets of texts. We were able to test the applicability of our model by feeding it data sets of different sizes, and having our Random Forest Classifier print various performance metrics, namely the "mean absolute error" (MAE), the "accuracy score", and for visualization purposes the "confusion matrix".

Our discussion of metrics is based on the "dreaddit" training (2838 samples) and testing (714 samples) data sets.

A confusion matrix is one of the ways to express if the classifier's predictions were correct or incorrect. The confusion matrix generates both actual and predicted values. All the diagonal elements denote correctly classified outcomes. The misclassified outcomes are represented in the off-diagonal portions of the confusion matrix. They are commonly denoted as true positive, true negative, false positive, and false negative.

In our application the main concern is the lower left entry in the confusion matrix: when we predict "OK" but we should really have been worried. These are called "false negatives", and would lead SAM to ignore a call for help.

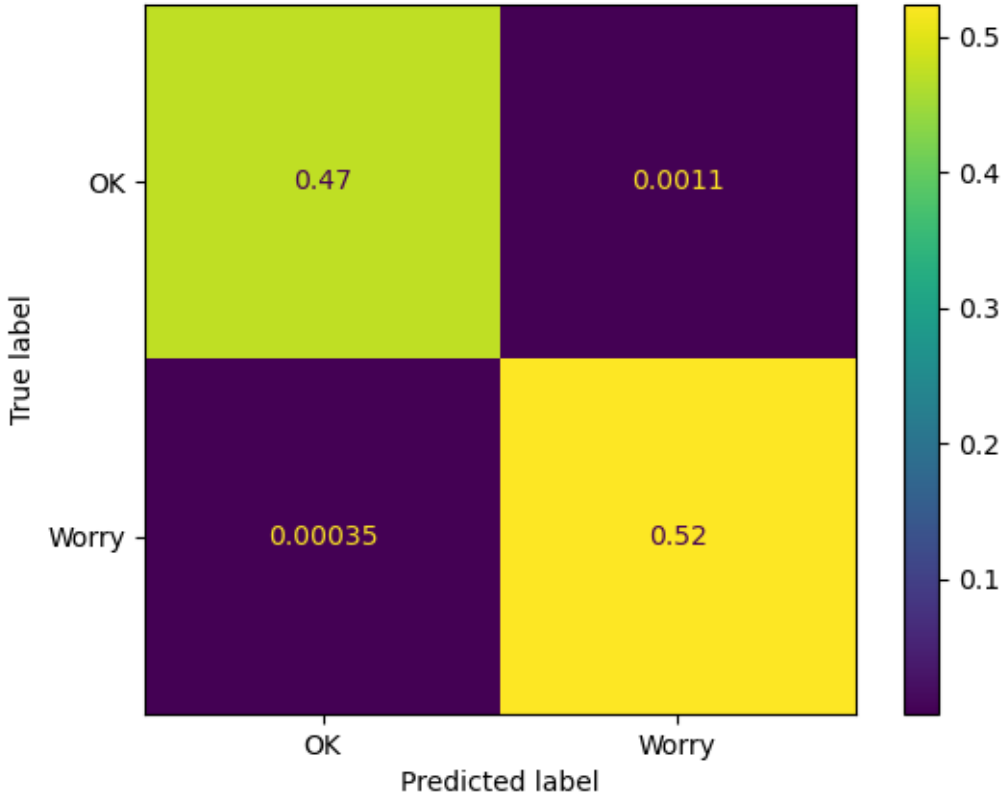


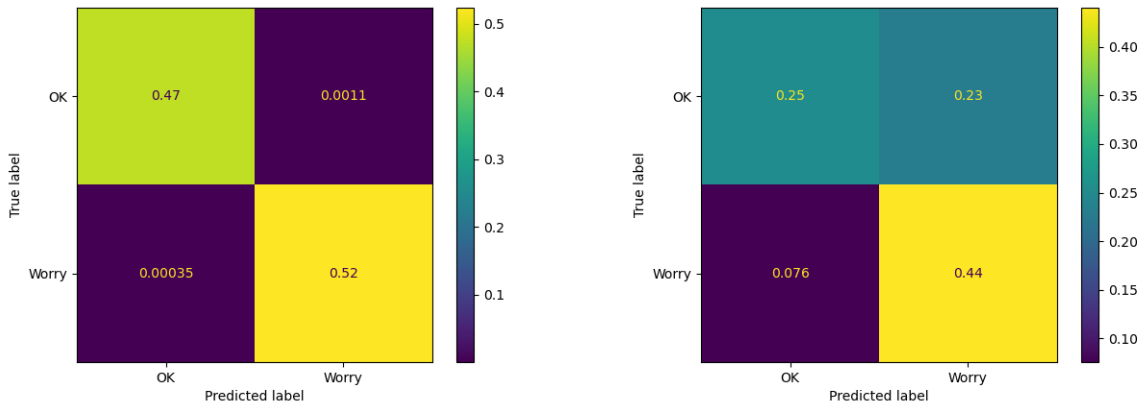
Figure 4: Simple example of confusion matrix for in-sample data on our training set.

in-sample MAE	in-sample accuracy	out-of-sample MAE	out-of-sample accuracy
0.0014	0.99859	0.30629	0.6937

In the confusion matrix in Figure 4 the top left value represents the **True Negative** where the model correctly predicted that the person is stressed. The top right value is a **False Positive** that says that the AI incorrectly predicted that the person was stressed (when they were actually OK). The bottom left is the **False Negative**, the AI incorrectly predicted the person to be ok (when they are actually stressed – this value is the key to whether our method works). The bottom right is a **True Positive** where the model correctly predicted that the person is not stressed.

We can run our model with either the bag-of-words method or the intensity score

method: the user can switch with a command-line option. We use the Bag of Words method by default. Our model compares its predictions to the actual results and produces the confusion matrices. For either method, our model trained itself on a user specified data set sample before it tested itself on data set sample that it has never seen before. Using both methods we were able to deduce that the Bag of Words method was more effective in correctly classifying if someone was stressed or not.



(a) Training Predictions

(b) Testing Predictions

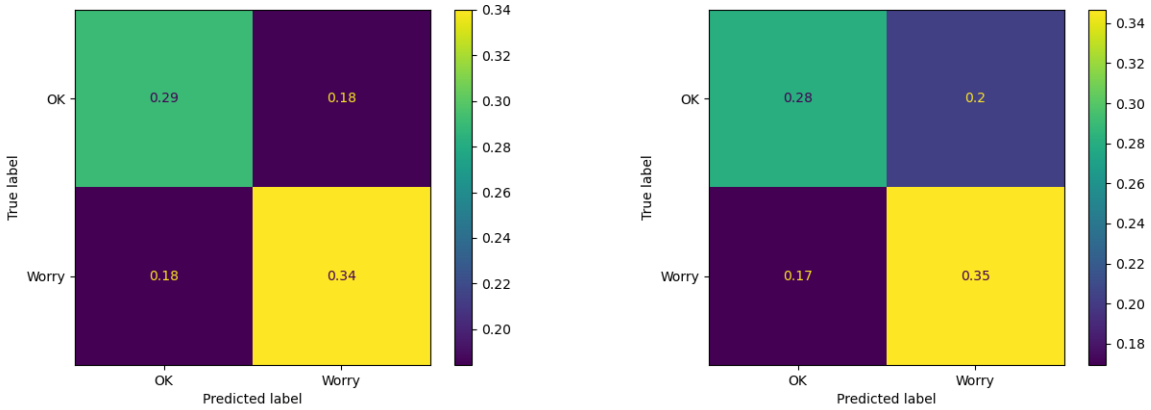
Figure 5: Bag of words predictions: here we look at the confusion matrix for both in-sample and out-of-sample data with the bag-of-words approach.

Figure 5 shows the confusion matrix for the Bag of Words method. In this instance the "in-sample" comes from the dreaddit training data, and the "out of sample" comes from the dreaddit test data.

In Figure 6 we see what we get with the "intensity score" approach, used on the same data sets.

The main take-home from these figures comes from the lower-left square in the confusion matrix. The in-sample "false negative" rate with the bag-of-words approach is extremely low (0.00035, or 0.035%). With the intensity score approach it is much bigger (0.18, or 18%).

The method performs worse with out-of-sample data: 0.076 (or 7.6%) false negatives for bag-of-words, versus 0.17 (17%) false negatives for the intensity score approach.



(a) Training Predictions

(b) Testing Predictions

Figure 6: Intensity score approach: confusion matrix for in-sample and out-of-sample data.

Clearly we can now focus on using the bag-of-words approach.

6 Conclusion

6.1 Takeaways

In conclusion, we were able to make a functional AI stress monitor that was able to detect if someone was considered to be at a high stress risk. This could potentially be used in the future to aid health care officials in screening someone as high risk for suicide based on text messages. It could also help friends and family monitor loved ones and guide them to professionals. Through two ways of extracting data we where able to identify that a Bag of Words approach was the best option when it came to accurately preparing the data predict if someone was considered highly stressed or not. The largest takeaway of SAM is its ability to help friends and professionals intervene before suicide occurs.

6.2 Limitations and Errors

Our current out-of-sample performance is that SAM will incorrectly think that someone "is fine" about 7% of the time. How can this be improved?

One enhancement would be to include other emotions conveyed through the text, like sorrow, aggression, shock, etc. Many of these scores are available in the data set. By including more parameters in determining if someone is more at risk for committing suicide will make our model more accurate and therefore more useful to health professionals.

Hyperparameter tuning is also needed; this would entail changing the hyperparemeters. Hyperparameters are values that are used to control the learning process of the AI.

6.3 Future Work

Given time we would like to expand this project to include other emotions conveyed through text. We would also like to have SAM keep a running tab on how often a certain user shows high risk signs through multiple posts and the number of times they post. We would also like to add a web-based application that can take images and screen shots, as well as text. It could also have a user type answers to questions about how they are feeling. The text would then be fed to SAM for classifcation of stress level.

7 Acknowledgments

We would like to thank our teacher sponsor, Rhonda Crespo, for providing us with support needed to work on and finish this project on time, our mentor, Mark Galassi, for his expertise and guidance with the code involved in our project, and the Supercomputing Challenge, for their resources and support.

8 Further reading

1. Accelerating random forests up to 45x using cuml. NVIDIA Technical Blog. (2022, August 21). Retrieved April 4, 2023, from https://developer.nvidia.com/blog/accelerating-random-forest

2. Ahr. America's Health Rankings. (n.d.). Retrieved from https://www.americashealthrankings.org/explore/annual/measure/Suicide/population/suicide_15-24/state/NM

3. Amadebai, E. (2022, August 12). Decision trees vs. random forests – what's the difference? Analytics for Decisions. Retrieved from https://www.analyticsfordecisions.com/decision-trees-vs-random-forests/

4. D'Hotman, D., Loh, E., & Savulescu, J. (2021, June 1). Ai-enabled suicide prediction tools: Ethical considerations for medical leaders. BMJ Leader. Retrieved from https://bmjleader.bmj.com/content/5/2/102

5. Haque, M. U., Dharmadasa, I., Sworna, Z. T., Rajapakse, R. N., & Ahmad, H. (2022, December 12). "I think this is the most disruptive technology": Exploring Sentiments of ChatGPT Early Adopters using Twitter Data. Retrieved from http://export.arxiv.org/abs/2212.05856v1

6. Kessler, R. C., van Loo, H. M., Wardenaar, K. J., Bossarte, R. M., Brenner, L. A., Cai, T., Ebert, D. D., Hwang, I., Li, J., de Jonge, P., Nierenberg, A. A., Petukhova, M. V., Rosellini, A. J., Sampson, N. A., Schoevers, R. A., Wilcox, M. A., & Zaslavsky, A. M. (2016, January 5). Testing a machine-learning algorithm to predict the persistence and severity of major depressive disorder from baseline self-reports. Nature News. Retrieved from https://www.nature.com/articles/mp2015198

7. Nichol, A. (2022, July 1). Dall·E 2 pre-training mitigations. OpenAI. Retrieved January 6, 2023, from https://openai.com/blog/dall-e-2-pre-training-mitigations/

8. Reduce the suicide rate - MHMD-01. Reduce the suicide rate - MHMD-01 - Healthy People 2030. (n.d.). Retrieved from https://health.gov/healthypeople/objectives-and-data/browse-objectives/mental-health-and-mental-disorders/reduce-suicide-rate-mhmd-01

9. Walsh, C. G., Ribeiro, J. D., & Franklin, J. C. (n.d.). Predicting risk of suicide attempts over time through ... - sage journals. Predicting Risk of Suicide Attempts Over Time Through Machine Learning. Retrieved from https://journals.sagepub.com/doi/10.1177/2167702617691560

10. Suicide rates by state 2023. (n.d.). Retrieved from https://worldpopulationreview.com/state-rankings/suicide-rates-by-state

11. Suicide statistics and facts. SAVE. (n.d.). Retrieved from https://save.org/about-suicide/suicide-statistics/

References

Amadebai, E. (2022). *Decision trees vs. random forests – what's the difference?* https://www.analyticsfordecisions.com/decision-trees-vs-random-forests/

America's health rankings. (2020). https://www.americashealthrankings.org/explore/annual/measure/Suicide/population/suicide%5C_15-24/state/NM

Brownlee, J. (2019). *A gentle introduction to the bag-of-words model.* https://machinelearningmastery.com/gentle-introduction-bag-words-model/

D'Hotman, D., Loh, E., & Savulescu, J. (2020). Ai enabled suicide prediction tools–ethical considerations for medical leaders. *BMJ Leader*, *5*(2).

Kessler, R. C., van Loo, H. M., Wardenaar, K. J., Bossarte, R. M., Brenner, L. A., Cai, T., Ebert, D. D., Hwang, I., Li, J., de Jonge, P., Et al. (2016). Testing a machine-learning algorithm to predict the persistence and severity of major depressive disorder from baseline self-reports. *Molecular psychiatry*, *21*(10), 1366–1371.

Suicide rates by state 2023. (2023). https://worldpopulationreview.com/state-rankings/suicide-rates-by-state

Suicide statistics and facts. save. (2021). https://save.org/about-suicide/suicide-statistics/

Turcan, E., & McKeown, K. (2019). Dreaddit: A reddit dataset for stress analysis in social media. *arXiv preprint arXiv:1911.00133*.

Galaxies Far, Far, Away

New Mexico

Supercomputing Challenge

Final Report

April 5, 2023

Team Number 50

St. Thomas Aquinas School

Team Members

Catherine Sedillo

Teacher

Eric Vigil

Project Mentor

James Sedillo

Contents

Executive Summary

Modern astronomical instruments are revealing an endless number of galaxies everywhere observations are made. This creates an overwhelming volume of information beyond the capability of human effort. With recent improvements in computing power and machine learning, the convolutional neural network shows promise as an aid to the astronomical surveyor especially for galaxy classification.

The capabilities of machine learning for galaxy classification are demonstrated here using Python, the Tensorflow 2 module, the Galaxy Zoo 2 dataset, and a small convolutional neural network model. A custom-method of image preprocessing is detailed and applied to the imagery to aid the convolutional neural network with achieving a high training accuracy.

Final training and validation of the convolutional neural network developed here resulted in a training accuracy of 98% and a validation accuracy near 80%. A simple graphical user interface (GUI) was made to allow the user to train, save, load, and interact with the CNN model and observe its class predictions in real-time.

Introduction

Modern astronomical telescopes are looking deeper into the universe. The images that they produce reveal the existence of countless stars and galaxies. The seemingly infinite quantity of these astronomical objects in these images has become too much for humans to identify and characterize. Machine learning can provide the ability to automatically classify these astronomical objects, allowing astronomers to focus on more abstract concepts.

This project uses a Convolutional Neural Network (CNN) architecture combined with a preprocessing technique to train a neural network to classify galaxies in the Galaxy Zoo 2 dataset. The goal is to achieve high-accuracy predictions of the galaxy classification.

Methodology

Neural networks are popular for recognizing and classifying objects in images. Common uses include video surveillance, self-driving vehicles, and cancer screening. The goal of this project is to apply this capability for the purpose of identifying galaxy types in telescope imagery using a neural network topology called the Convolutional Neural Network, or CNN.

Convolutional Neural Networks

Convolutional Neural Networks make up the first layers of a neural network intended for image recognition. This is because, like the animal visual cortex, CNNs provide a feature-extraction capability to the neural network (NN)[4, p. 518]. The output of a CNN layer is typically connected to a "max pooling" layer creating what is referred to as a "feature map." The layering of CNNs and max-pooling layers provides the broader neural network with the capability to recognize more complex image features. A set of fully-connected neural network layers make-up the final layers of the CNN with the last layer composed of a number neurons corresponding to the number of object classes the CNN needs to identify.

Image Dataset and Image Classes

Training a CNN must be done with the aid of a dataset that has an abundance of images which show the objects of interest. For this project, the dataset was downloaded from the Galaxy Zoo 2 dataset available in [1]. This dataset contains 243,437 images obtained by the Sloan Digital Sky Survey (SDSS) telescope. This dataset features a multitude of galaxy types in color, 424x424 resolution, JPEG images along with

two spreadsheets for referencing each image's classification. Image classes include the following examples [5, p. 2860]:

- Er = smooth galaxy, completely round.
- SBc2m = barred disk galaxy with a just noticeable bulge and two medium-wound spiral arm(s).
- Seb = edge-on disk galaxy with a boxy bulge.
- Sc(I) = disk galaxy with a just noticeable bulge, no spiral structure, and irregular morphology.
- A = star.

For this project, the classes were reduced to 3: elliptical galaxies, spiral/disk galaxies, and edge-on spiral/disk galaxies. Stars and galaxies with irregular features were not included in the training and validation datasets to improve training accuracy.

Software and Hardware

The software was developed using the Python programming language running within an Anaconda environment. Python modules used included the following:

- OpenCV: for loading images, grayscale-conversion, and saving images.
- Scikit-Image: for generating intensity profile lines in images.
- tkInter: for application GUI rendering.
- Numpy: for list and matrix math.
- Pandas: for reading *.csv files
- Matplotlib: for plotting/visualizing data.
- Tensorflow 2: for building, training, and testing the CNN.
- os: for image directory queries.
- Random: for image file name shuffling.

Hardware specifications of the PC were:

- Intel i9-9900KF processor
- 64-GB system RAM.
- nVidia RTX-2080 Super graphics card with 8GB VRAM.
- Windows 10 OS.

Data Preprocessing

Data preprocessing can be any method that helps remove unnecessary information and/or enhances relevant information within an image. According to [2, p.13], "In an image-understanding system, the preprocessing stage often performs functions such as the gray scale manipulation, edge detection, developing descriptions of objects or shapes in the image, image restoration, and geometric correction." The use of preprocessing allows the Neural Network to train on relevant information and comes highly recommended from the research literature.

As suggested by [2, p.13], the galaxy images were first converted to grayscale to reduce the image dimensionality from 3D-RGB (Red, Green, Blue) to 2D-grayscale. This resulted in 424x424-pixel images with intensity values ranging from 0 to 255. Since the images contain an abundance of extra objects such as stars and other galaxies, an approach was devised to remove these objects by interpreting the image as a topographical map. Topographical maps feature contour lines representing locations of equal altitude. For the galaxy images, contour lines represent pixels of equal intensity. Using the scikit-image module available to Python, the find_contours() function was employed to generate contours for pixels of intensity 60 (out of 255). Of the many contours generated by this function, the contour with the largest length was chosen as it was most likely to correlate with the largest object in the image, this being the galaxy of interest. A simple computation of the hypotenuse on each of this galaxy contour's set of coordinate pairs resulted in a list of radius values of the galaxy contour. The largest of these radii was then used as the radius for creating a circular image filter where pixels within the circle had value 1 and all pixels outside the filter had a values of 0. The image filter was made to have the same dimensions as the original image (424x424) with the circular filter overlapping the galaxy. Multiplication of the corresponding pixels of the image and the image filter generally removed most of the objects within the image while keeping the galaxy visible. Examples of this are shown in the figures 1 and 2. The Python application developed for this purpose (countour_prep.py) was used to preprocess a large batch of 8,443 image files for use by the CNN.

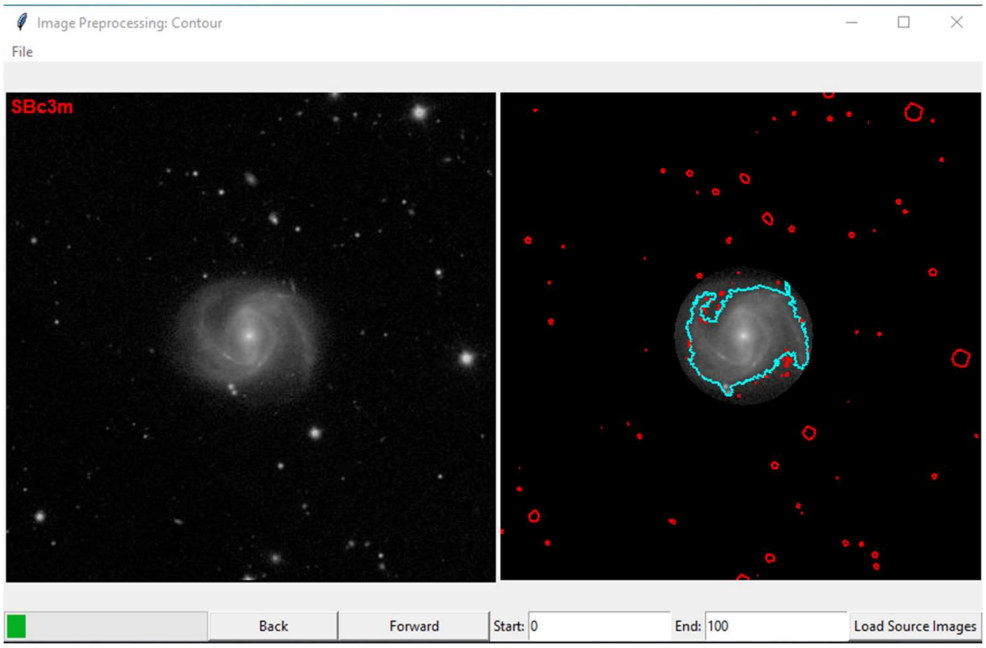


Figure 1 Preprocessing applied to a spiral galaxy image.

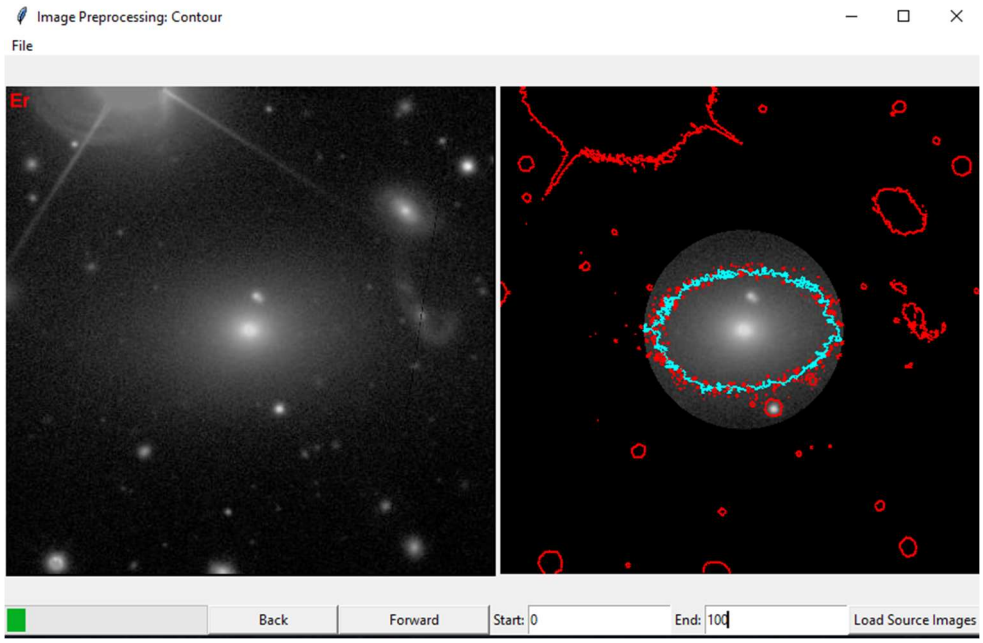


Figure 2 Preprocessing applied to an elliptical galaxy image.

Model and Verification Methods

Convolutional Neural Network Architecture

Various CNN architectures were tested in this study in order to obtain one with the highest validation accuracy. The final architecture tested took the form of figure 3.

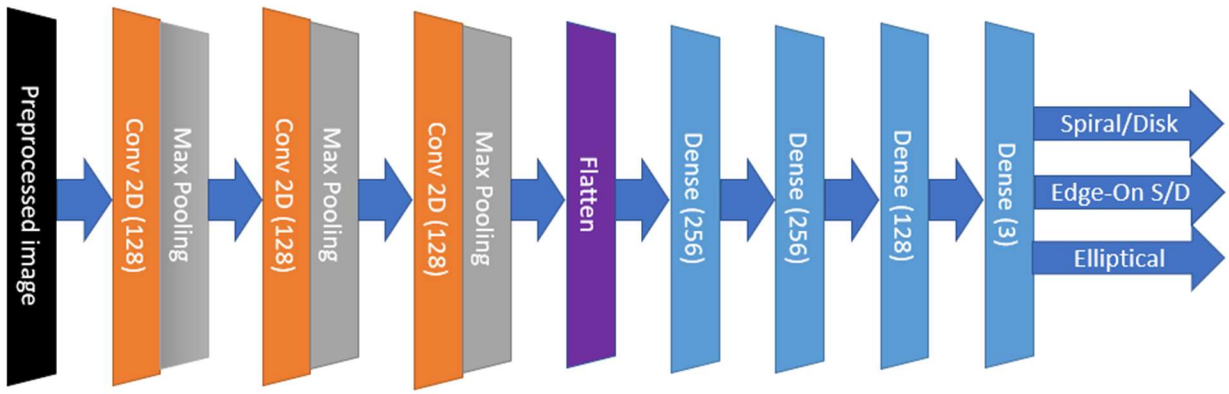


Figure 3 Final CNN architecture

As can be seen from the figure, the CNN takes a preprocessed image at its input. This image is then presented to a convolutional layer for the first feature-extraction phase. Since convolutional layers increase the dimensionality of the data, max-pooling layers are added behind the convolutional layers to reduce or "downsample" the data [3, p.385]. Two additional convolutional layers are added to provide the CNN with the ability to capture higher-level patterns from previous convolutional layers. Again, max-pooling layers are added to reduce the growing dimensionality of the data output from the convolutional layers. Finally, a flattening layer is used as an interface between the convolutional layers and the subsequent dense layers. Dense layers represent the conventional type of neural network. Layer configurations followed from what was commonly used in the research [4, pp.545-547] and [3, pp.392-393] where convolutional layers were set to use 3x3 filters, max-pooling layers used 2x2 filters, and all activation functions were of the "ReLU" type [3, p.258]. The final layer was different from the others in that it used a "Softmax" activation function to provide a confidence value for multiclass classification [3, p.268]. Experimentation was then performed by varying the sizes of the layers and noting the performance of the CNN's training accuracy and validation accuracy.

Tensorflow 2 and Keras

The CNN was created using the Tensorflow 2 module and its Keras wrapper available to Python. Tensorflow allows the neural network developer to conveniently enclose training samples and their class descriptions into objects called "datasets." Furthermore, testing data is also encapsulated into Tensorflow dataset objects. With training and test data ready, the CNN "model" is created layer-by-layer using Keras. Next, the layers are compiled along with the choice of optimizer, loss function, and the metric to optimize. Finally, a fitting function is called against the model object with the dataset objects

and number of epochs as arguments. Tensorflow then proceeds to train the CNN, reporting the training accuracy and validation accuracy after every epoch. Tensorflow also provides training and validation process configurations called “pipelines” that can be used for data shuffling, caching, batching, prefetching, and most important of all: preprocessing. After configuration and fitting, a trained Tensorflow model is available to save to disk in hdf5 format.

Tensorflow 2 Model for Galaxy Classification

The following figure represents Tensorflow 2’s summary of the final CNN model developed for this project.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 200, 200, 1)       0

conv2d (Conv2D)              (None, 198, 198, 128)     1280

max_pooling2d (MaxPooling2D  (None, 99, 99, 128)       0
)

conv2d_1 (Conv2D)            (None, 97, 97, 128)       147584

max_pooling2d_1 (MaxPooling  (None, 48, 48, 128)       0
2D)

conv2d_2 (Conv2D)            (None, 46, 46, 128)       147584

max_pooling2d_2 (MaxPooling  (None, 23, 23, 128)       0
2D)

flatten (Flatten)            (None, 67712)             0

dense (Dense)                (None, 256)               17334528

dense_1 (Dense)              (None, 256)               65792

dense_2 (Dense)              (None, 128)               32896

dense_3 (Dense)              (None, 3)                 387

=================================================================
Total params: 17,730,051
Trainable params: 17,730,051
Non-trainable params: 0
_________________________________________________________________
```

Figure 4 Tensorflow 2 model summary for the Galaxy Classifier App

Following the general consensus of the research literature [4, p.547] and [3, p.393], the following Tensorflow 2 arguments were passed to the compile() function of the model object:

- Optimizer = ‘adam’
- Loss = ‘sparse_categorical_crossentropy’

- Metrics = [‘accuracy’]

Epoch count was limited to an initial run of 30. If the training accuracy suggested that the CNN’s accuracy had more potential for improvement, an additional 30 epochs were run.

Additional minor preprocessing was performed by the Python application. This included cropping a 200x200 pixel image from the original 424x424 image and a normalization of the pixel values from the original 0-to-255 range (8-bit unsigned integer) to a 0-to-1 range (32-bit, floating-point).

The preprocessed images were used to the fullest extent (8,440 out of 8,443 available) with an 80% allocation to the training dataset (6,752 images) and a 20% allocation to the validation dataset (1,688 images). Once all training epochs completed, the Python application (main_final.py) plotted the training accuracy and the validation accuracy as a function of the epoch number.

Results

The final model was trained and validated over a total of 60 epochs in two, 30-epoch runs. The first run resulted in the data shown in figure 5.

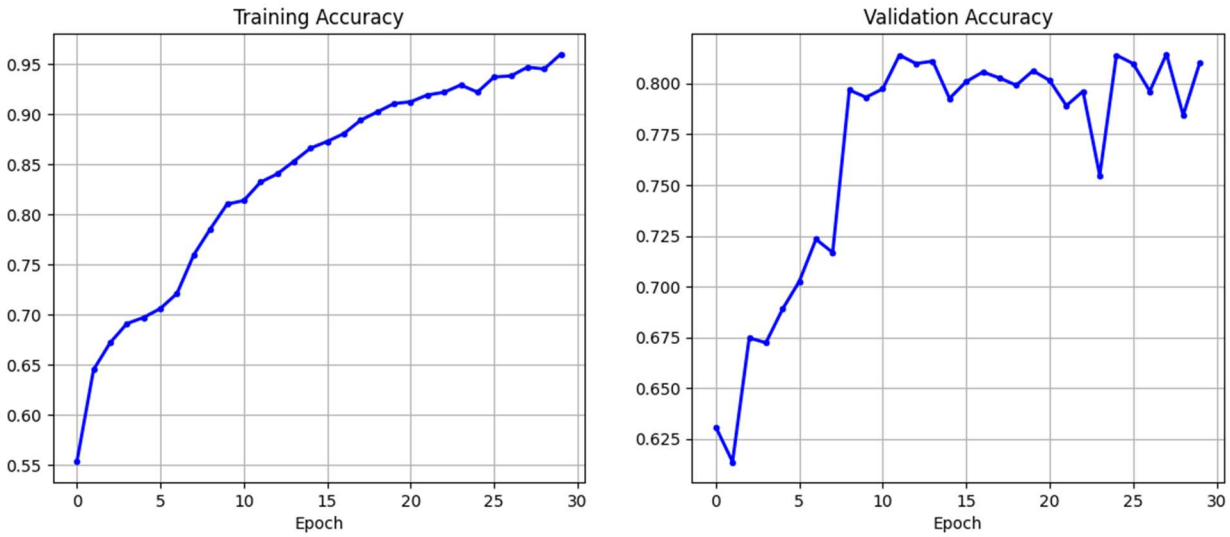


Figure 5 Training and validation accuracy of the final model, first 30-epochs

Training accuracy yielded 96% with validation accuracy yielding 81%. In an attempt to improve the validation accuracy, a second, 30-epoch training run was done yielding the following result in figure 6.

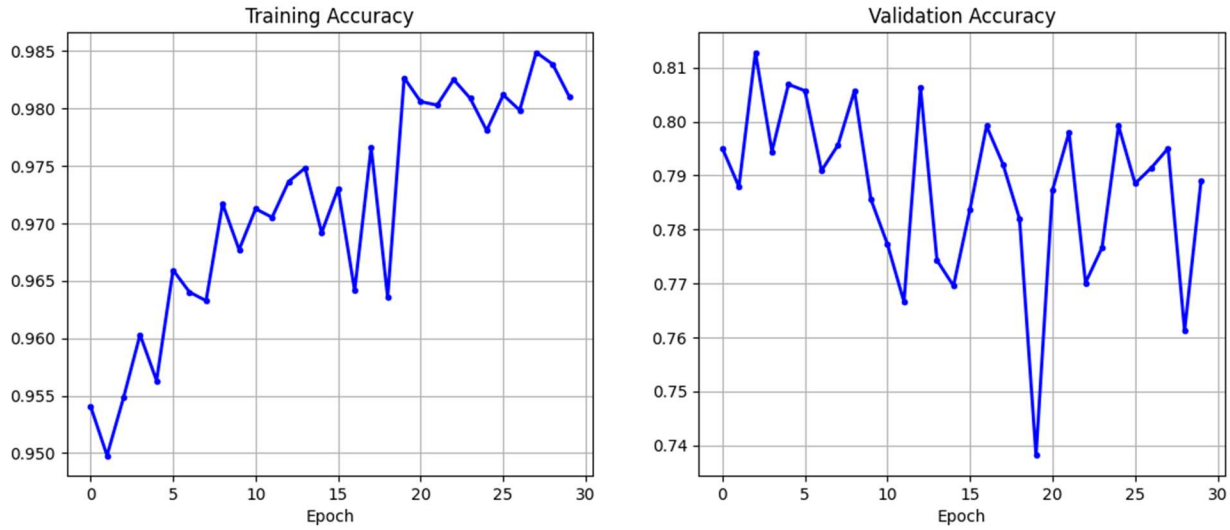


Figure 6 Training and validation accuracy of the final model's second, 30-epoch run.

Here, training accuracy improved to 98%, but validation accuracy declined to 79%, an indication that the additional training was tending toward overfitting the data.

The final realization of the Python program (main_final.py) is an interactive graphical user interface (GUI) shown in figure 7.

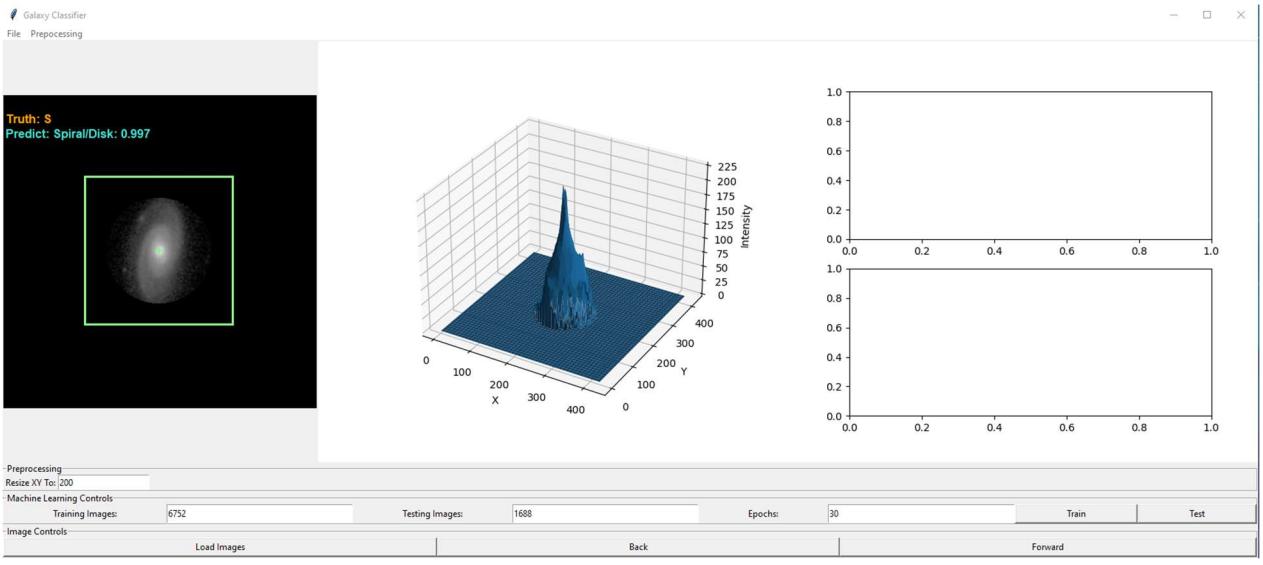


Figure 7 The Galaxy Classifier application GUI

This GUI allows the user to observe the preprocessed galaxy image, along with its true and predicted classification. The 3D plot in the center provides a topographical perspective of the image pixel intensity. Controls are provided at the bottom to load the image set, cycle through the images, and train the CNN.

Conclusions

Machine learning with convolutional neural networks most-certainly demonstrates a promising capability for the classification of galaxies in astronomical surveys. High training accuracies are demonstrably obtained with the convolutional neural network employed, but high-validation accuracies are a bit more challenging to achieve. The use of custom-developed preprocessing methods, such as the technique demonstrated in this project, contribute significantly to the CNN's ability to obtain high-accuracy classification predictions. With the freely-available dataset provided by Galaxy Zoo, the free tools available to Python, any motivated machine-learning developer who wishes to test their own preprocessing algorithm and neural network tuning skills can undertake this galaxy classification challenge.

References

[1] https://zenodo.org/record/3565489#.ZCt2rPbML-h

[2] Kulkarni, Arun D. "Artificial Neural Networks for Image Understanding." Van Nostrand Reinhold, 1993.

[3] Liu, Yuxi H. "Python Machine Learning by Example." Packt Publishing, 2020.

[4] Raschka, Sebastian and Mirjalili, Vahid. "Python Machine Learning." Packt Publishing, 2019.

[5] Willet, Kyle W., et al. "Galaxy Zoo 2: Detailed Morphological Classifications for 304,122 Galaxies from the Sloan Digital Sky Survey." Monthly Notices of the Royal Astronomical Society, 2013.

Acknowledgements

Special thanks to Christopher Hoppe for his generous interim review of this project.