Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

Ultra-Early Wildfire Detection for Dryad Systems

Submitted to

Dryad Networks

by

Team #1

Dryad

Team Members

Rishav De rishavde@bu.edu

Pranet Sharma pranetsh@bu.edu

Victoria Kang vkang@bu.edu

Thomas Bowler tsbowler@bu.edu

Young-Chan Cho yccho@bu.edu

Submitted: 4/15/2024

Dryad ML Models User Guide

Table of Contents

Executive Summary (Rishav De)

Dryad Networks' project aims to improve its AI and IoT-based forest fire prevention and detection system. With wildfires contributing to global CO2 emissions and escalating climate threats, the need for more accurate and swift detection is paramount. The project aims to refine the machine learning model, deploy it at the edge and in the Cloud, and collaborate with data scientists to enhance evaluation mechanisms. The focus is on early wildfire detection, aligning with engineering requirements for performance, reliability, scalability, integration, and security. Dryad Networks' system prioritizes early detection using innovative technology to aid global wildfire prevention.

1 Introduction (Victoria Kang)

Worldwide wildfires release about 2 billion tons of CO2 into the atmosphere, cause about 300 deaths yearly, and contribute to a staggering 4 billion US dollars of damages almost every year. Our client, Dryad, has developed the Silvanet Suite, a collection of sensors and supporting technologies that use AI/ML to detect the early onset of wildfires to combat this issue. Our task has been to contribute towards the research and development of improved AI/ML models that can more accurately predict forest fires based on various measurements.

The research we have done has been put into practice through 3 different ML models built in Google Colaboratory, using Python toolkits. The project has aimed to compare different data sources and see which suits fire detection the best and to generate models that are fast, accurate, and small in size.

Due to time constraints and other issues, the models have not been able to be deployed on the sensors themselves. Thus, this manual will focus on the operations and outcomes of the models. The models have been measured against parameters commonly used in the field, such as F1-score, Accuracy, etc, and have been compiled here. The three different models are the *Binary Classification Model, FireNet Model,* and *FireData US Model*. We hope that these models will serve as a foundation for others to decide which methods are best for forest fire detection and will guide their decision when implementing such models in edge devices and Cloud servers. Additionally, information about these models will be available on a website (currently being developed) which will take a hands-on approach to show a user the models and their results and operations.

In short, the resulting product shown below is more of a guide to ML in the forest fire detection space, but we hope that it will enable you to gain a better understanding of the issues and solutions that may be used in this field.

2 System Overview and Installation (Rishav De, Young-Chan Cho)

2.1 Overview block diagram

Binary Classification Model:

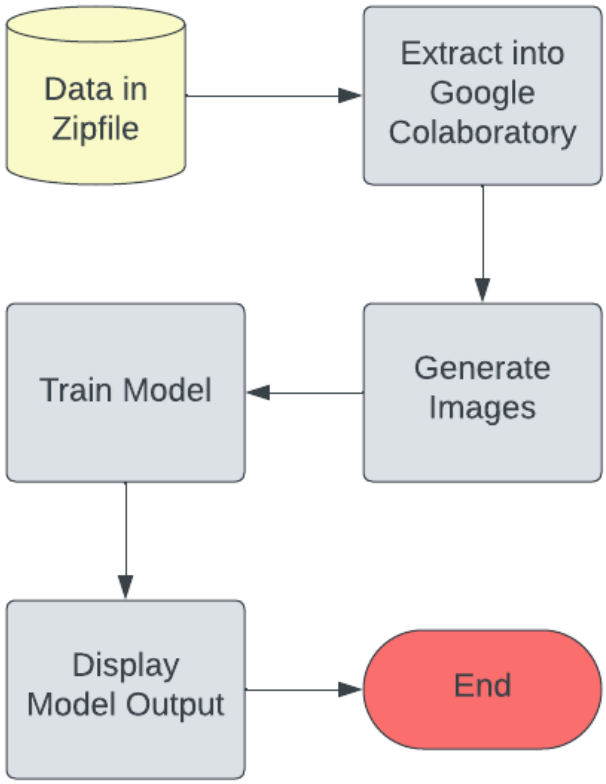

Fig 1: Overview of the flow of the Binary Classification model

FireNet Model:

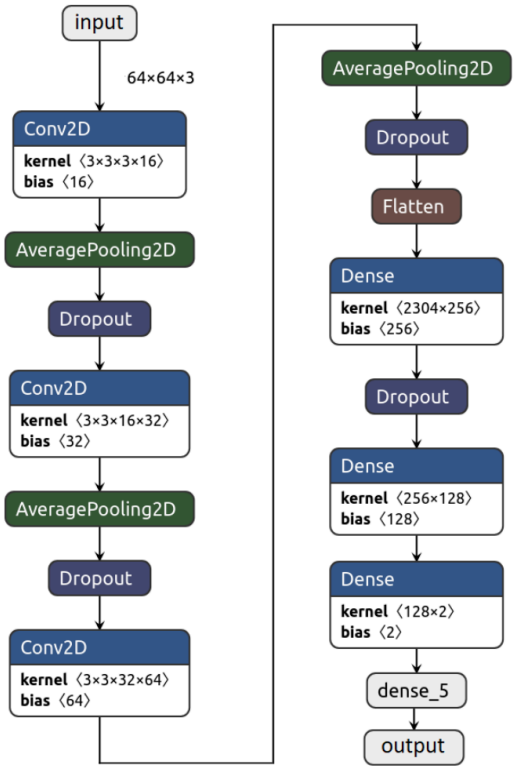

Fig 2: Architecture of the neural network of FireNet

FireData US Model:

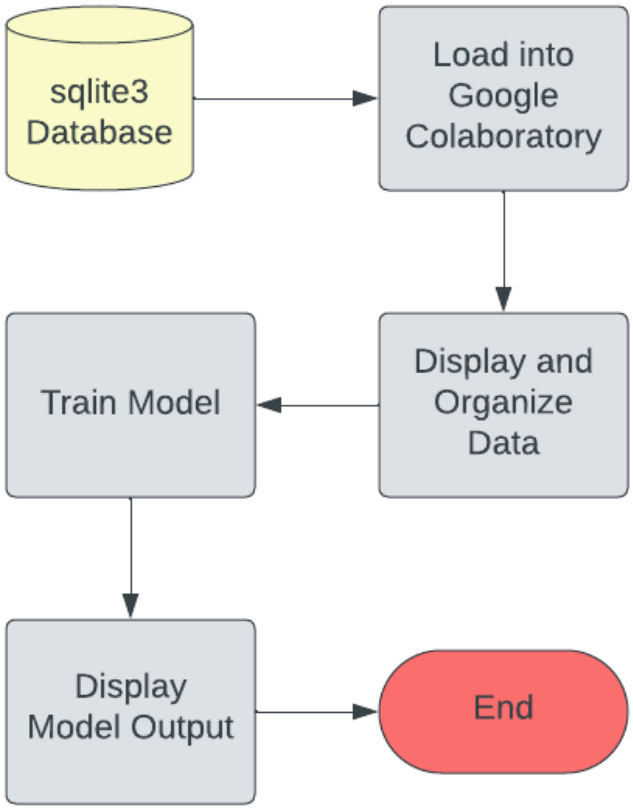

Fig 3: Overview of the flow of the FireData US model. The SQLite database originates from the Department of Agriculture

2.2 User interface

For each model, the user interface will be the Google Colaboratory, which will look similar to the diagram below:

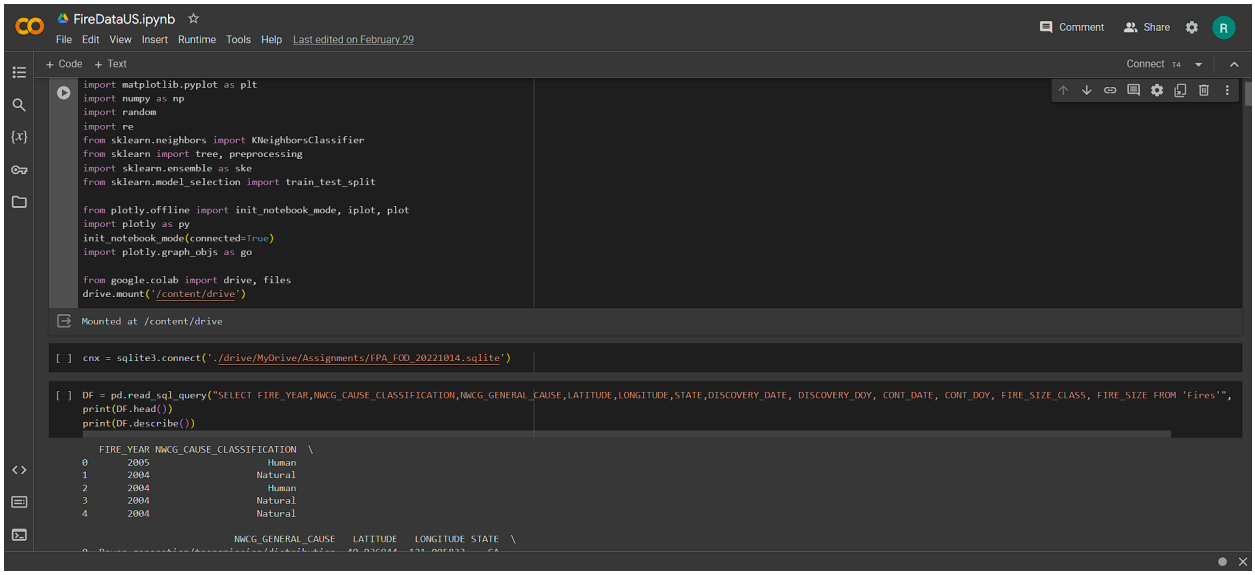

Fig 4: Google Colaboratory User-facing GUI for FireData US Model

A website has been built, which provides detailed information regarding the models and a link to view the source code for each model. The user interface looks like the following:

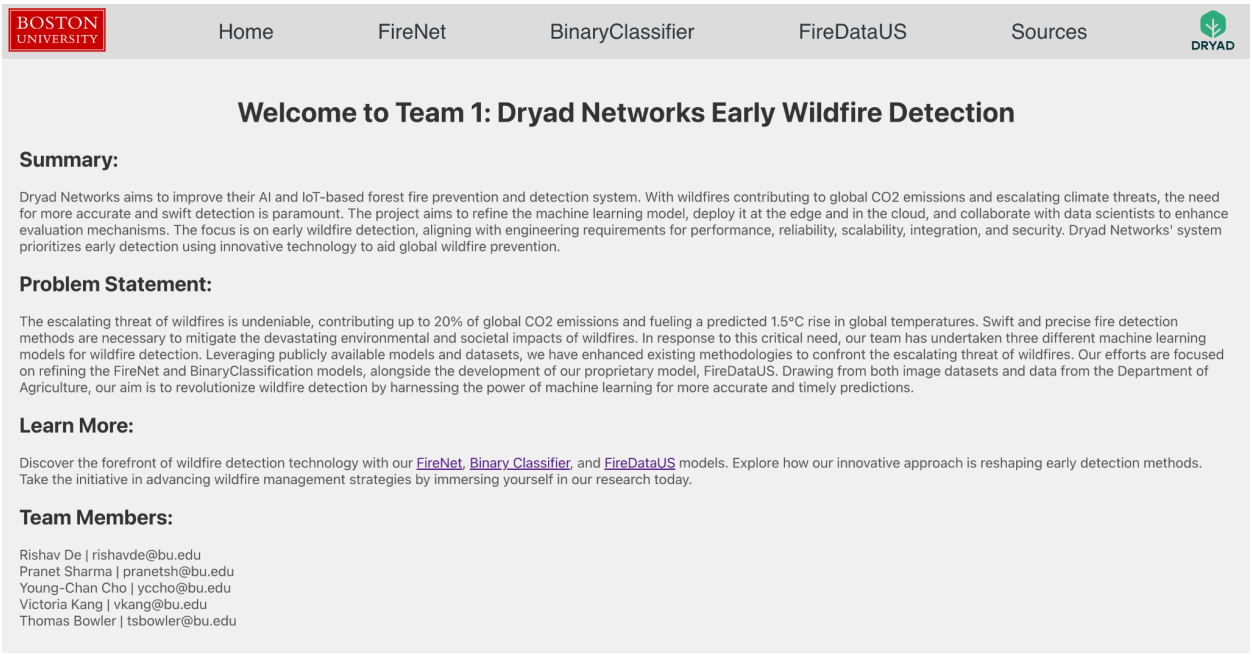

Fig 5: Website UI

Website Navigation and Description

This website will serve as the central hub for our project, providing detailed insights, results, and methodologies behind the development and implementation of our fire detection and analysis models. Below is a comprehensive guide to navigating our website and understanding its content.

1. Frontpage:
    a. Summary of the Project: An overview of our project, highlighting the objectives, scope, and significance
    b. Problem Statement: A detailed description of the problem we aim to solve, outlining the challenges and necessities driving our project
    c. Why We Did It: Our motivation behind undertaking this project, emphasizing the expected impact and the benefits it aims to deliver
    d. Team Members
2. Model Pages: Each of our models is dedicated to a specific page, providing an in-depth look at their functionalities, data sources, theoretical frameworks, and outcomes.
    a. Firenet Model
    b. Binary Classification Model
    c. US Forest Model
3. References: Data Related to Fires. A comprehensive list of references, datasets, and resources we utilized or referenced throughout our project. This section aims to provide

transparency and allow for further exploration or verification of our data and methodologies.

2.3 Installation, setup, and support

Prerequisites: Access to Google Colaboratory (https://colab.research.google.com/), access to Google Drive, stable Internet connection, permission for Google Colaboratory to access Google Drive (user will be prompted to grant permission when the model runs)

The following steps should be followed in order to be able to run the models:

1. From the website, download the Python source code for the particular model you wish to run along with the associated dataset.
2. The Python source code must be uploaded to Google Colaboratory(https://colab.research.google.com/). This can be done using the upload feature.

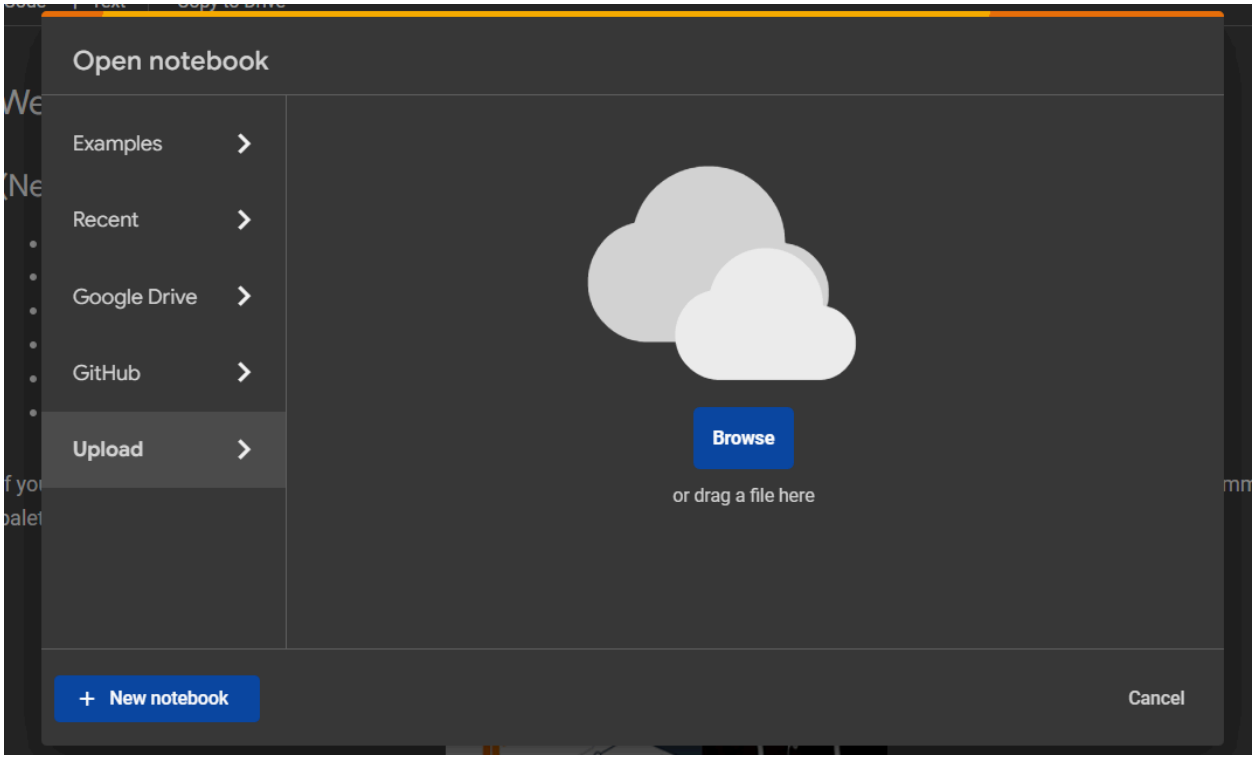

Fig 6: Colab Upload Feature

3. The dataset must be uploaded to the linked Google Drive in the particular folder that follows the naming convention as described in the model.
    a. Binary Classification Model: /content/drive/MyDrive/FireImageData.zip
    b. FireNet Model:
        i. Test1: /content/drive/My Drive/fire_dataset_kaggle

1. https://www.kaggle.com/datasets/phylake1337/fire-dataset

ii. Test2: /content/drive/My Drive/fire_images_dataset/test

1. https://www.kaggle.com/datasets/rachadlakis/firedataset-jpg-224

iii. Test3: /content/drive/My Drive/forest_fire_images/Test_Data

1. https://www.kaggle.com/datasets/mohnishsaiprasad/forest-fire-images

c. FireData US Model: /drive/MyDrive/Assignments/FPA_FOD_20221014.sqlite

Once the customer follows the above steps, the models are ready to operate.

3 Operation of the Project (Pranet Sharma, Young-Chan Cho)

3.1 Operating Mode 1: Normal Operation

For all Models:

Under normal operating conditions:

1. Model Initialization: Launch your Python environment through Google Colaboratory. Load the FireNet model. Ensure all necessary libraries (TensorFlow, Keras, OpenCV, Matplotlib, sklearn) are installed.
2. User Interface: The interface is primarily command-line inputs within Python scripts in Google Colab. For Google Colab, use the option to mount Google Drive and access your data sets directly. The “Run All” option under “Runtime” should start everything up properly.
3. Exiting the Operation: To exit, simply stop the Python script. For Google Colab, disconnect the session to free up resources.

3.2 Operating Mode 2: Abnormal Operations

For all Models:

Under abnormal operating conditions (like upload failures, compilation errors, etc):

1. Diagnostic Mode: Implement a diagnostic script to check the health of the model and dataset integrity (e.g., correct formatting, no corrupted files). This script should automatically run before the main prediction process starts.
2. User Intervention: If the diagnostic mode identifies issues, it will prompt the user with specific errors and suggested corrective actions. For model compatibility issues, ensure the TensorFlow and Keras versions match the project requirements.
3. Recovery from Abnormal States: The system should guide the user through correcting common issues, such as reinstalling dependencies or reformatting datasets. For more complex problems, direct to a troubleshooting guide or community forum for additional support.

4. Restart the Runtime: By disconnecting and reconnecting the runtime under “Runtime”, many issues regarding loading and compilation will be solved.

If all fails, close the model and reload as described in Normal Operations.

4 Technical Background (Rishav De)

Since this is a strongly software-focused project, no hardware has been used.

Google Colaboratory:

The models have been built, tested, and deployed on Google Colaboratory due to the platform's easy-to-use nature and support for running models on GPU’s. A Notebook is used for the models and the availability of different code cells allows for better debugging and presentation possibilities.

ML Libraries:

Various libraries like *scikit*, *Tensorflow*, and *sklearn* have been used to implement the models. These libraries allow for various model types like XGBoost and Random Forest and allow for easy hyperparameter changes. These libraries also support displaying model metrics, allowing us to understand the performance of the models.

Plotly:

This has been used for Data visualization, and to make easy-to-read and colorful interactive graphs. This has been used mostly to visualize data parameters and see how the data directly affects the models and their outputs.

React:

React has been used to develop the web app along with HTML with CSS, while hosting is performed by GitHub Pages. We have been using React’s vast libraries and routing methods to develop a user-friendly interface to display the results of the models and take the users through the process of building the models and understanding the outputs. HTML and CSS were used for styling purposes to keep the application appealing to users. Also, GitHub Pages is a convenient method to host the website as deployment is extremely simple.

5 Relevant Engineering Standards (Jueun Kang)

Coding Standards:

The project adheres to the coding standards to ensure the quality, readability, and consistency of our code. For instance, we have implemented style guidelines, such as ensuring proper spacing, indentation, consistent naming conventions, and comprehensive commenting so that the code remains easy to read and maintain.

Testing Standards:

Software testing standards are a set of guidelines and principles for software testing procedures. It offers a framework to test the software in order to ensure its quality and functionality. We followed the five key elements of the testing standard: test planning, test case development, test execution, test analysis, and test reporting.

6 Cost Breakdown (Thomas Bowler)

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost
ML Libraries	Nil	ML libraries to support model development and deployment	$0	$0
Google Colaboratory	Nil	Development environment for model	$0	$0
React	Nil	Web design framework	$0	$0
Server	1	To Host Website and other ML needs	$300	$300
			Beta Version-Total Cost	$300

Table 6.1. Production Cost for Dryad

The cost of the server is listed as a yearly cost (A business website on GCP costs that much per year). The other costs are minimal, as the software itself to develop the product is free. There are paid versions that may improve performance but they are not necessary.

7 Appendices (Thomas Bowler)

7.1 Appendix A - Specifications

1. **Precision**: The ratio of true positive samples among all samples predicted as positive.

   TP / (TP + FP), where TP is True Positives and FP is False Positives.

2. **Recall**: The ratio of true positive samples detected among all actual positive samples.

   TP / (TP + FN), where FN is False Negatives

3. **F1-Score**: The harmonic mean of Precision and Recall.

   2 * (Precision * Recall) / (Precision + Recall)

4. **Accuracy**: The percentage of samples correctly predicted by the model.

   (TP + TN) / Total number of cases

BinaryClassifier

Requirements	**Results**
Precision	A >0.70 B >0.60
Recall	A >0.70 B >0.60
F1-Score	A >0.70 B >0.60
Accuracy	>0.60

FireNet

Requirements	**Results (Mean Value)**
Precision	93.88%
Recall	64.86%
F-Measure	76.35%
Accuracy	76.75%

FireDataUS

Requirements	**Results**
Precision	A >0.70 B >0.60

Recall	A >0.70 B >0.60
F-Measure	A >0.70 B >0.60
Accuracy	>0.60

7.2 Appendix B – Team Information

Rishav De
Worked on FireData US model creation and improvement as well as website development. Will be studying ECE at the Carnegie Mellon University Graduate Program.

Pranet Sharma
Worked on Binary Classification model improvement as well as website development. Will be working as a Software Engineer at Viasat with a focus on embedded software development.

Victoria Kang
Worked on website content and dataset collection.

Thomas Bowler
Worked on website design, interested in medical devices industry

Young-Chan Cho
Worked on FireNet model improvement and data source sorting, as well as website development. Planning to start a career as a software engineer in the United States.