Wildfire Prediciton using Machine Learning

MSc Research Project
Data Analytics

JAI MAHESH CHAUHAN
Student ID: x20208375

School of Computing
National College of Ireland

Supervisor: Rejwanul Haque

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	JAI MAHESH CHAUHAN
Student ID:	X20208375
Programme:	MSc. DATA ANALYTICS **Year:** 2022-2023
Module:	MSc. Research Project
Lecturer:	
Submission Due Date:	14/08/2023
Project Title:	Wild Fire Prediction using Machine Learning
Word Count:	**471 Page Count:** 6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: JAI MAHESH CHAUHAN

Date: 14/08/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual.

JAI MAHESH CHAUHAN
Student ID: X20208375

1 Introduction

The research gives an in-depth look at wildfire prediction modeling, with a focus on evaluating and understanding the wildfire prediction dataset. Since flames are getting bigger and happening more often around the world, it is important to be able to predict them accurately. This study tries to explain how Convolutional Neural Networks (CNN) were used to look at the dataset, prepare the data, analyze it, model it, and measure its accuracy.

2 Specification of System

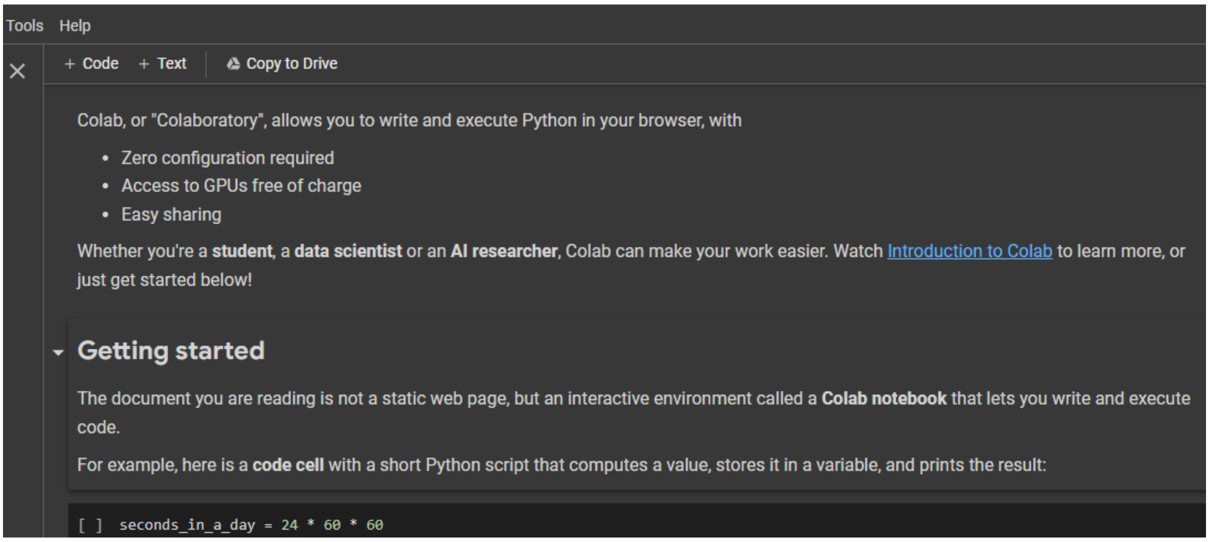
The machine that performed this research configuration is: 16gb RAM, INTEL i7 11th Generation @2.30 GHz processor, 64-bit OS, windows 11.

3 Software Requirements

The software requirements are needed to run the code. Google Colaboratory is used as the environment for the research code. Python language was used for the project. Google drive account is used to link to notebook. Microsoft Excel is needed as to store the data in csv file.

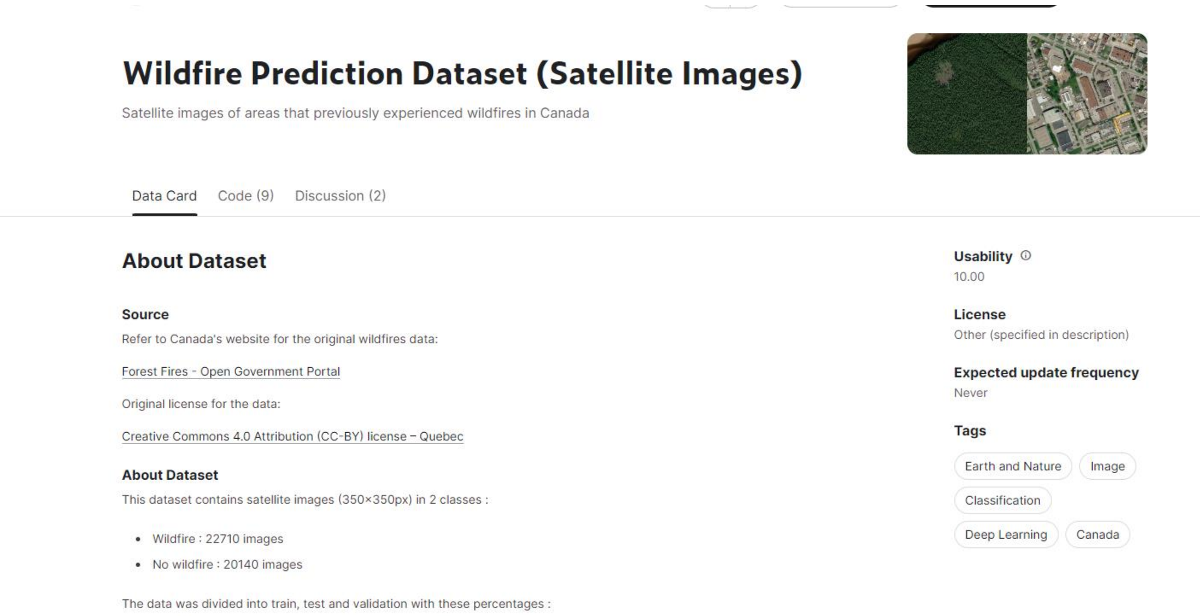
4 Environment Set-Up

The setting up of the colab environment is done following the steps below will allow the code to run for research project. The steps are shown using images for better understanding.



5 Data selection process

The dataset used in the research is form the open source dataset website i.e. Kaggle.com. Dataset is called Wildfire Prediction Dataset (Satellite Images). Given figure below shows the overview of the dataset page on Kaggle.



6 Libraries used

Following are the libraries which are used and will be needed in order to run the code or else result may differ or code will give error.

1. Pandas
2. Tensorflow
3. Numpy
4. Matplotlib

5. CV2
6. OS

```python
import tensorflow as tf
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.utils import load_img, img_to_array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout, BatchNormalization, Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
import pandas as pd
```

7 Implementation

Once the data is downloaded off Kaggle and uploaded to google drive as well as all the files. Mounting the google drive must then be done Please accept access google drive files.

```python
[2] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

To run the file directly on google colab. Set the base path to your drive as shown below in figure for better understanding. Or if you have placed files on some other folder or location on drive set the path accordingly.

Gathering and loading data

Gathering The Data

```python
[3] #Import the libraries
    import zipfile
    import os
    zip_ref = zipfile.ZipFile('/content/drive/MyDrive/dataset/train.zip', 'r') #Opens the zip file in read mode
    zip_ref.extractall('/tmpp') #Extracts the files into the /tmp folder
    zip_ref.close()

[4] zip_ref = zipfile.ZipFile('/content/drive/MyDrive/dataset/test.zip', 'r') #Opens the zip file in read mode
    zip_ref.extractall('/tmpp') #Extracts the files into the /tmp folder
    zip_ref.close()

[5] zip_ref = zipfile.ZipFile('/content/drive/MyDrive/dataset/valid.zip', 'r') #Opens the zip file in read mode
    zip_ref.extractall('/tmpp') #Extracts the files into the /tmp folder
    zip_ref.close()
```

Loading The Data

```
[6] image_shape = (350,350,3)
    N_CLASSES = 2
    BATCH_SIZE = 256

    # loading training data and rescaling it using ImageDataGenerator
    train_datagen = ImageDataGenerator(dtype='float32', rescale= 1./255.)
    train_generator = train_datagen.flow_from_directory('/tmpp/train',
                                                       batch_size = BATCH_SIZE,
                                                       target_size = (350,350),
                                                       class_mode = 'categorical')

    # loading validation data and rescaling it using ImageDataGenerator
    valid_datagen = ImageDataGenerator(dtype='float32', rescale= 1./255.)
    valid_generator = valid_datagen.flow_from_directory('/tmpp/valid',
                                                       batch_size = BATCH_SIZE,
                                                       target_size = (350,350),
                                                       class_mode = 'categorical')

    # loading test data and rescaling it using ImageDataGenerator
    test_datagen = ImageDataGenerator(dtype='float32', rescale = 1.0/255.0)
    test_generator = test_datagen.flow_from_directory('/tmpp/test',
                                                     batch_size = BATCH_SIZE,
                                                     target_size = (350,350),
                                                     class_mode = 'categorical')

Found 7142 images belonging to 2 classes.
Found 1698 images belonging to 2 classes.
Found 1658 images belonging to 2 classes.
```

BuildingModel

Building The Model

```
# defining the coefficient that our regularizer will use
weight_decay = 1e-3

# building a sequential CNN model and adding layers to it
# dropout and the regularizer are used in general to prevent overfitting
first_model = Sequential([
    Conv2D(filters = 8 , kernel_size = 2, activation = 'relu',
    input_shape = image_shape), MaxPooling2D(pool_size = 2),

    Conv2D(filters = 16 , kernel_size = 2, activation = 'relu',
    input_shape = image_shape), MaxPooling2D(pool_size = 2),

    Conv2D(filters = 32 , kernel_size = 2, activation = 'relu',
           kernel_regularizer = regularizers.l2(weight_decay)),
    MaxPooling2D(pool_size = 2),

    Dropout(0.4),
    Flatten(),
    Dense(300,activation='relu'),
    Dropout(0.5),
    Dense(2,activation='softmax')
])
# showing the summary of our model (layers and number of parameters)
first_model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 349, 349, 8)       104

 max_pooling2d (MaxPooling2D  (None, 174, 174, 8)      0
 )
```

Training the Model

Train The Model

```
# don't stop everything if an image didn't load correctly
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

# checkpointer to save the model only if it improved
checkpointer = ModelCheckpoint('first_model.hdf5',verbose=1, save_best_only= True)
# early stopping to stop the training if our validation loss didn't decrease for (10) consecutive epochs
early_stopping = EarlyStopping(monitor= 'val_loss', patience= 10)
# Adam, best optimiser for deep learning models to help with the training
optimizer = optimizers.Adam(learning_rate= 0.00001)
# setting our loss function and which metric to evaluate
first_model.compile(loss= 'categorical_crossentropy', optimizer= optimizer,
                    metrics=['AUC','acc'])

# TRAIN
history = first_model.fit(train_generator,
                    epochs = 15,
                    verbose = 1,
                    validation_data = valid_generator,
                    callbacks = [checkpointer, early_stopping])
```

```
Epoch 1: val_loss improved from inf to 0.69310, saving model to first_model.hdf5
28/28 [==============================] - 60s 2s/step - loss: 0.6541 - auc: 0.6843 - acc: 0.6005 - val_loss: 0.6931 - val_auc: 0.5718 - val_acc: 0.5395
Epoch 2/15
28/28 [==============================] - ETA: 0s - loss: 0.5140 - auc: 0.9084 - acc: 0.8199
Epoch 2: val_loss improved from 0.69310 to 0.47205, saving model to first_model.hdf5
28/28 [==============================] - 39s 1s/step - loss: 0.5140 - auc: 0.9084 - acc: 0.8199 - val_loss: 0.4721 - val_auc: 0.9591 - val_acc: 0.9140
Epoch 3/15
```

Accuracy Check

```
# add history of accuracy and validation accuracy to the plot
plt.plot(history.history['acc'], label = 'train',)
plt.plot(history.history['val_acc'], label = 'valid')

plt.legend(loc = 'lower right')
plt.xlabel('epochs')
plt.ylabel('accuracy')

# show plot
plt.show()
```

References

Ager, A.A., Day, M.A., Alcasena, F.J., Evers, C.R., Short, K.C. and Grenfell, I. (2021)‘Predicting Paradise: Modeling future wildfire disasters in the western US’, *Science of the total environment*, 784, p.147057.

Aguilera, R., Luo, N., Basu, R., Wu, J., Clemesha, R., Gershunov, A. and Benmarhnia, T. (2023) ‘A novel ensemble-based statistical approach to estimate daily wildfire-specific PM2. 5 in California (2006–2020)’, *Environment International*, 171, p.107719.

Al-Kahlout, M.M., Ghaly, A.M.A., Mudawah, D.Z. and Abu-Naser, S.S. (2020) ‘A neural network approach to predict forest fires using meteorological data’, *International Journal of Academic Engineering Research (IJAER)*, 4(9).

Challa, S.K., Kumar, A. and Semwal, V.B. (2022) ‘A multibranch CNN-BiLSTM model for human activity recognition using wearable sensor data’, *The Visual Computer*, 38(12), pp.4095-4109.