Configuration Manual

Gankidi Sai Charan Reddy

Student ID: X23253207

1. Introduction

Disasters are becoming frequent across the globe, and their impacts are more severe owing to climate change. Accurate forecast of these disasters and its efficient response may be able cut down the number of lives lost, the amount of property destroyed and ensure efficient utilization of available resources. The principal goal of this project is to change the currently existing process of disaster prediction and response through the help of AI, ML, and DL.

The main idea of primary focus is to train and build reliable artificial intelligence models that can estimate natural disasters with different numerical fields as well as the image datasets. Structured data are analysed using the ML models and depending on some features such as the temperature, precipitation, and past disasters history. In parallel, DL approaches including CNNs and transfer learning with VGG16 and ResNet50 are used under image classification to recognize disasters that include floods, hurricanes, and wildfires.

One of the important components of this work is the translation of it into application in real-time using the Gradio interface. Not only can such weather parameters be entered into the platform for instant analysis, but visual data related to disasters can also be immediately fed into the system. The system not only predicts the kinds of disasters but also suggests ideas for emergency program declarations. The result is a practical disaster management resource that satisfies both the need for high predictive precision and practical value when managing disasters.

2. System Requirements

2.1 Hardware

- **Processor:** Quad-core 2.5 GHz or better for efficient data processing and model training.
- **RAM:** A minimum of 16 GB is recommended to handle large datasets and memory-intensive computations.
- **GPU:** A CUDA-enabled NVIDIA GPU (e.g., Tesla T4 or higher) is essential for training deep learning models, significantly accelerating computation times.

2.2 Software

- **Operating System:** Compatible with Windows, Linux, or macOS platforms.
- **Development Environment:** Google Colab with GPU enabled is used for development, providing a cloud-based, resource-efficient platform.
- **Programming Language:** Python 3.9+ for its extensive libraries and ease of use in AI development.
- **Python Libraries:** Key libraries include:
  - TensorFlow for deep learning model training and evaluation.
  - Scikit-learn for machine learning algorithms and preprocessing.
  - Gradio for real-time interactive deployment.
  - **Additional libraries:** Pandas, NumPy, Matplotlib, OpenCV, Imbalanced-learn, and XGBoost.

3. Dataset Configuration

This project analyses numerical data and disaster images to construct AI-based prediction and natural disaster categorisation. Great care must be taken to handle as well as preprocess these datasets in order to get the right results.

3.1 Sources

Structured Data: The data extracted is historical in nature for weather and disaster processes is collected from reputable data banks like NOAA and ECMWF. The data is saved in CSV format for instance, DisasterWeatherNoDuplicates.csv and contains variables including precipitation, temperature, disaster types, geography.

Image Data: Disaster images, in formats of .jpg and .png, are organized according to disasters, which include; flood disaster images, hurricanes disaster images, wildfire disaster images. These images are applied in the disaster image classification using deep learning approaches.

3.2 Preprocessing

Handling Anomalies: Replace, for example, -9999 in the temperature and precipitation columns with the median value to minimize data inaccuracy.

Image Normalization: All images should be resized to 224 by 224 pixels and the pixel intensity values should be scaled to the range 0 to 1 for uniformity in the entire dataset.

Class Balancing:

- For structured data, use Synthetic Minority Oversampling Technique (SMOTE) to fix the problem of imbalance with regards to disaster types.
- With images, add the variations such as rotation and flipping for image data and scaling so as to enhance the data variety and improve the ability of the model to generalize.

4. Code Configuration

4.1 Setup

1. Mount Google Drive

Datasets and output models are stored in Google Drive, which is mounted for easy access:

```
from google.colab import drive

drive.mount('/content/drive')
```

2. Install Dependencies

pip install tensorflow scikit-learn pandas matplotlib seaborn gradio joblib opencv-python imblearn xgboost

4.2 Preprocessing

Structured Data:

- For the numerical columns, use the following standards with regard to problematic cells i.e. (-9999’s) Replace these with median averages.

- Take out outliers from the data set adding a leverage in the process the Interquartile Range or IQR method.
- Convert the variable such as state, incidentType, and declarationType to machine learning acceptable format with LabelEncoder.

Image Data:

- Make images small to match the size expected by the model with an ideal size of 224 * 224 pixels.
- Therefore, a scale of pixel values is normalized to come within [0 1] for the stability of computation.
- Augment the data by normalising and standardising the pictures and presenting different rotations, flips and sizes of the images.

4.3 Workflow

The workflow is implemented within a single integrated script, executed in Google Colab.

- load and preprocesses structured and image data.
- Create train machine learning models using Random Forest and train deep learning models using CNN, VGG16 and ResNet50.
- Save trained models on a Google Drive for reuse.
- Gradio has been used to deploy models for real time interaction, enabling users to input structured data, or upload images to predict disaster types.

5. Model Training and Evaluation

The model training and evaluation process is divided into two pipelines: Structured data machine learning (ML) and image-based disaster classification using Deep Learning (DL). In this section, the models, configurations and evaluation metrics underwent are described.

5.1 ML Pipeline

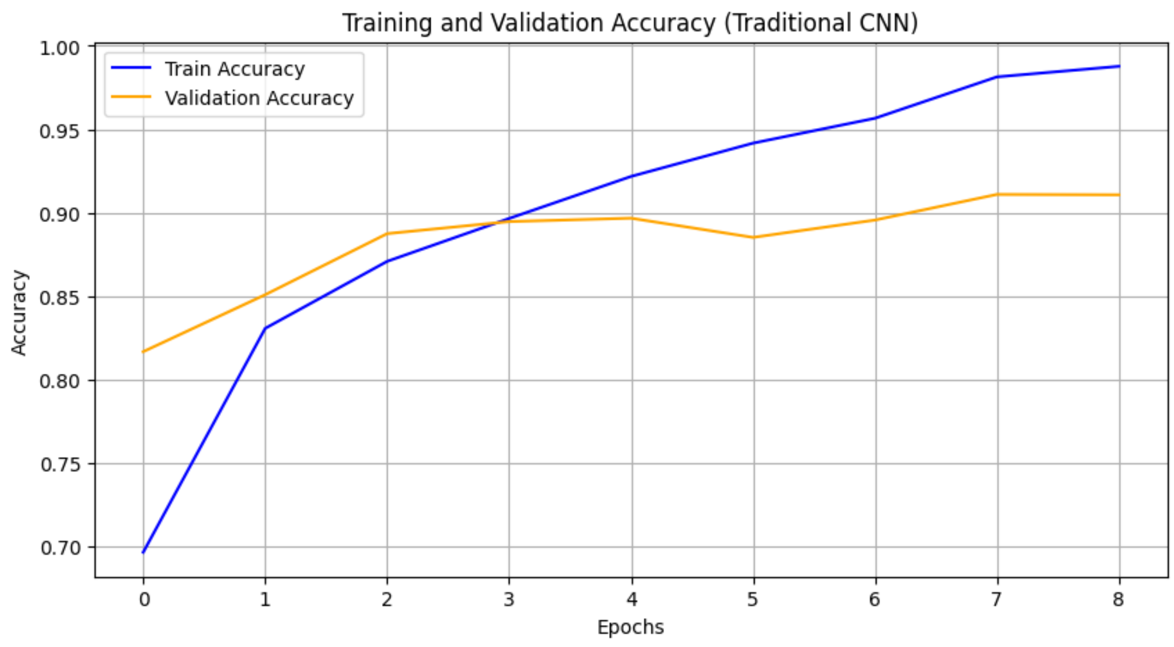
The structured data used to benefit from the ML pipeline uses the Random Forest algorithm to predict disaster categories.

- **Input Features:** Temperature, Precipitation, Cooling Days, Heating Days and Declaration type are the key predictors. SMOTE was first applied on these features to deal with anomalies and to balance the class distribution.
- **Training:** With optimized hyperparameters the Random Forest classifier was trained on structured data and achieved an accuracy over 94%.
- **Output:** The Weather information variables are used to predict disaster categories (i.e. floods, hurricanes) and the model is very successful.

5.2 DL Models

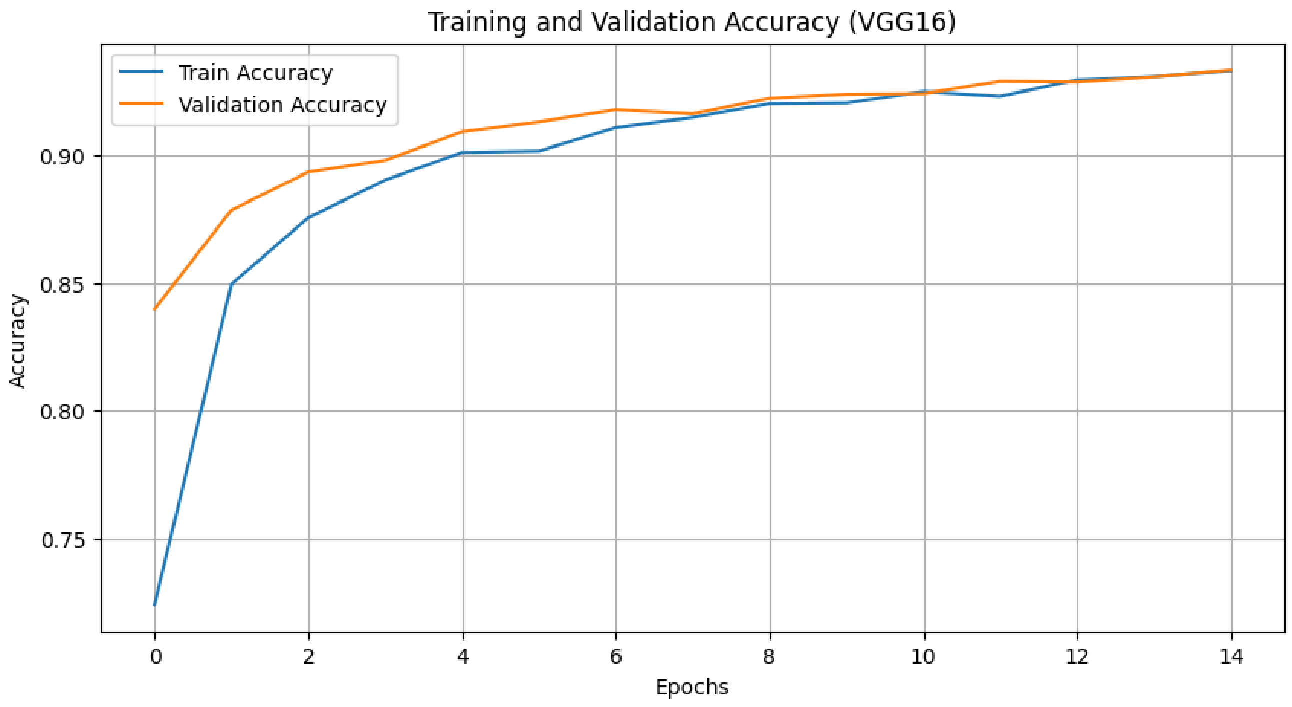
Convolutional Neural Network (CNN):

- It was used for initial disaster image classification through the convolutional layers to extract feature and dense layer to classify.
- It was able to achieve ~91% validation accuracy, distinguishing these disaster categories (like floods or wildfires).



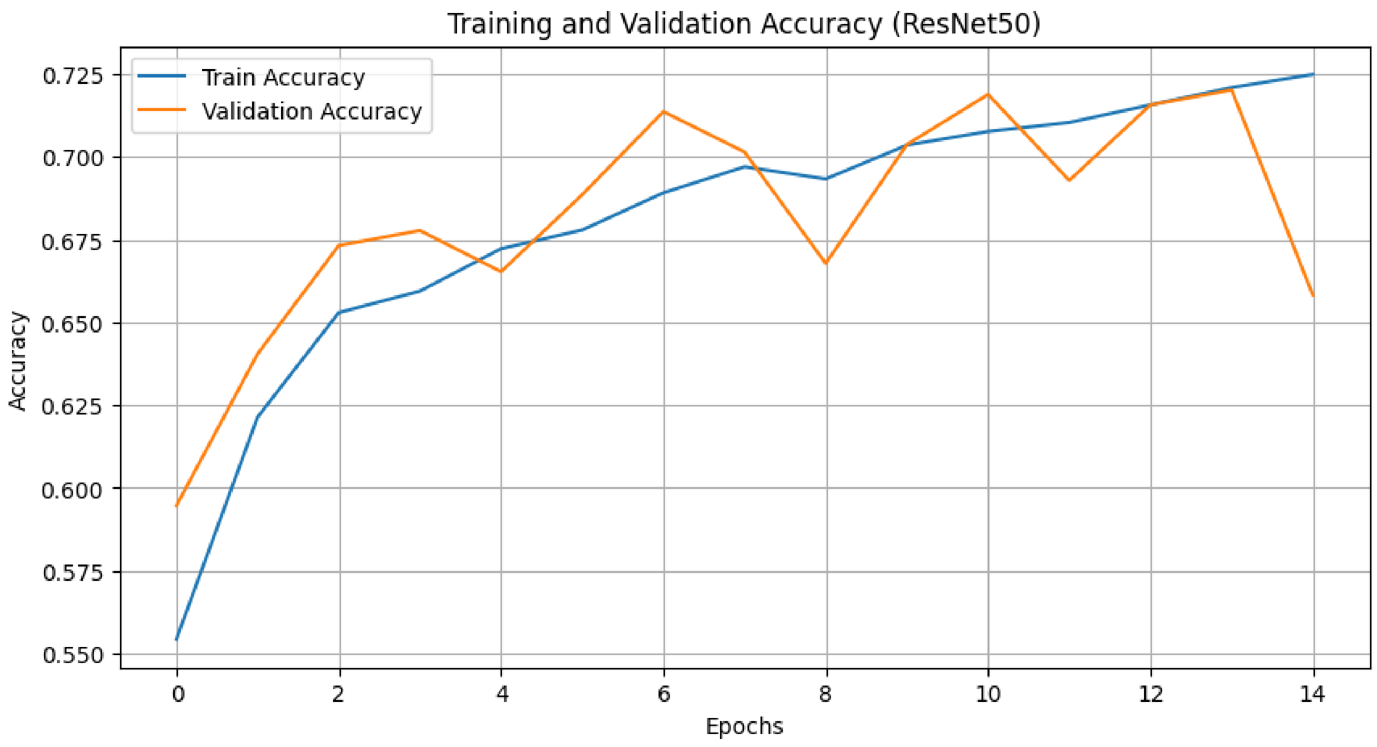
VGG16 (Transfer Learning):

- The model has been pretrained on ImageNet then tuned on disaster image datasets.
- It achieved ~94% validation accuracy, achieving robustness and scalability for image-based classification tasks.



ResNet50 (Transfer Learning):

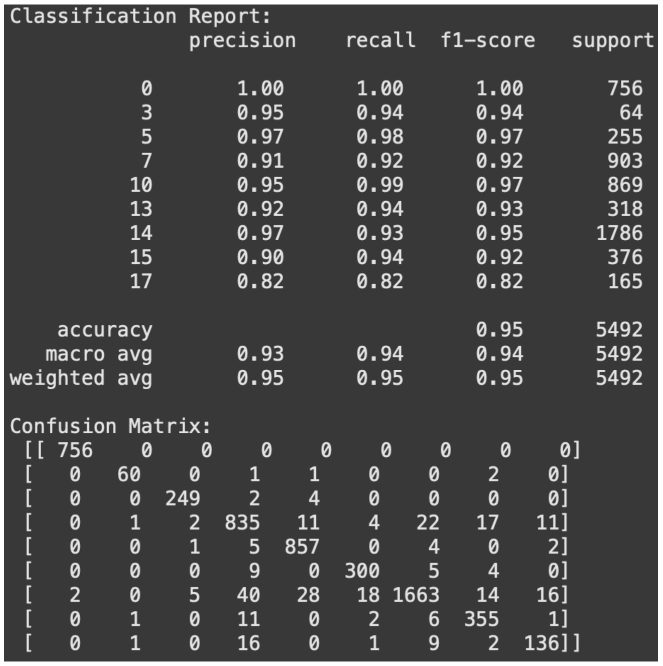
- Another highly optimised, advanced pretrained model tuned for large scale image datasets.
- We achieved ~73% accuracy, which suggests that it could handle high dimensional and complex datasets with high accuracy.



5.3 Evaluation Metrics

The models were evaluated using:

- **Classification Metrics:** For measuring the performance across disaster categories - Accuracy, Precision, Recall and F1-Score.



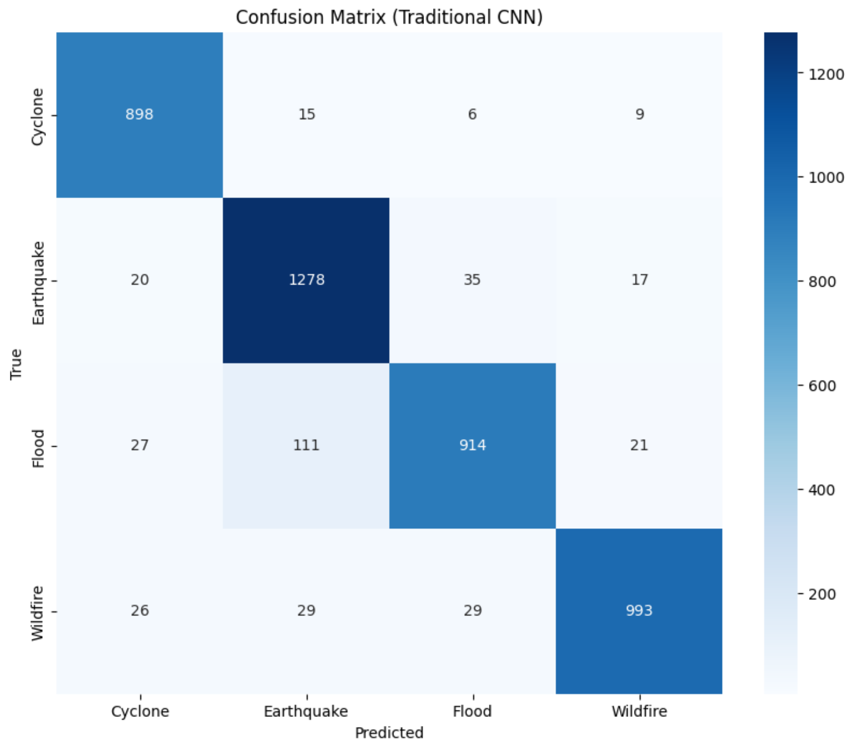
```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       756
           3       0.95      0.94      0.94        64
           5       0.97      0.98      0.97       255
           7       0.91      0.92      0.92       903
          10       0.95      0.99      0.97       869
          13       0.92      0.94      0.93       318
          14       0.97      0.93      0.95      1786
          15       0.90      0.94      0.92       376
          17       0.82      0.82      0.82       165

    accuracy                           0.95      5492
   macro avg       0.93      0.94      0.94      5492
weighted avg       0.95      0.95      0.95      5492

Confusion Matrix:
 [[ 756    0    0    0    0    0    0    0    0]
 [    0   60    0    1    1    0    0    2    0]
 [    0    0  249    2    4    0    0    0    0]
 [    0    1    2  835   11    4   22   17   11]
 [    0    0    1    5  857    0    4    0    2]
 [    0    0    0    9    0  300    5    4    0]
 [    2    0    5   40   28   18 1663   14   16]
 [    0    1    0   11    0    2    6  355    1]
 [    0    1    0   16    0    1    9    2  136]]
```

- **Visual Performance Checks:**
  - **Confusion Matrix:** It highlighted true and false classifications of disaster types.
  - **Loss/Accuracy Plots:** Helped in terms of the trends you have in training and validation so that you're not overfitting or, really, underfitting.

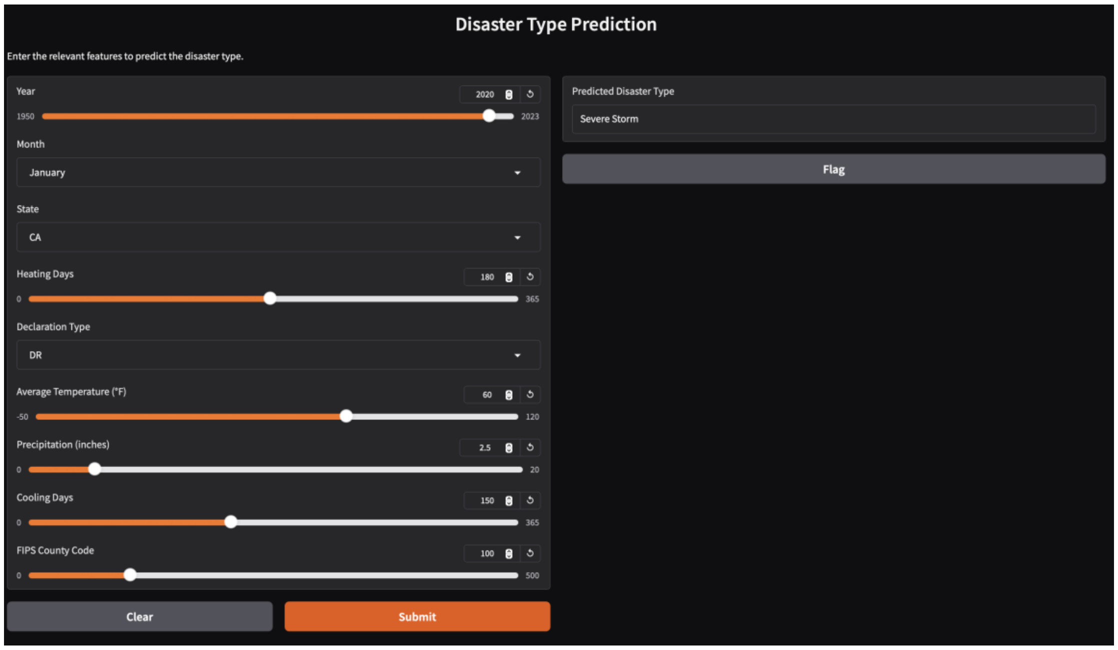


6. Deployment

In the deployment phase, the trained Machine Learning (ML) and Deep Learning (DL) models are deployed into a user-friendly, real-time interface with Gradio. This guarantees both access and usability for disaster prediction applications.

6.1 Features

Structured Data Predictions: Given user supplied weather metrics (temperature, precipitation, heating/cooling days, declarations) the ML model predicts disaster categories. The structured input provides accurate forecasts for real world disasters.


Disaster Type Prediction

Enter the relevant features to predict the disaster type.

Year: 2020 (1950 – 2023)

Month: January

State: CA

Heating Days: 180 (0 – 365)

Declaration Type: DR

Average Temperature (°F): 60 (-50 – 120)

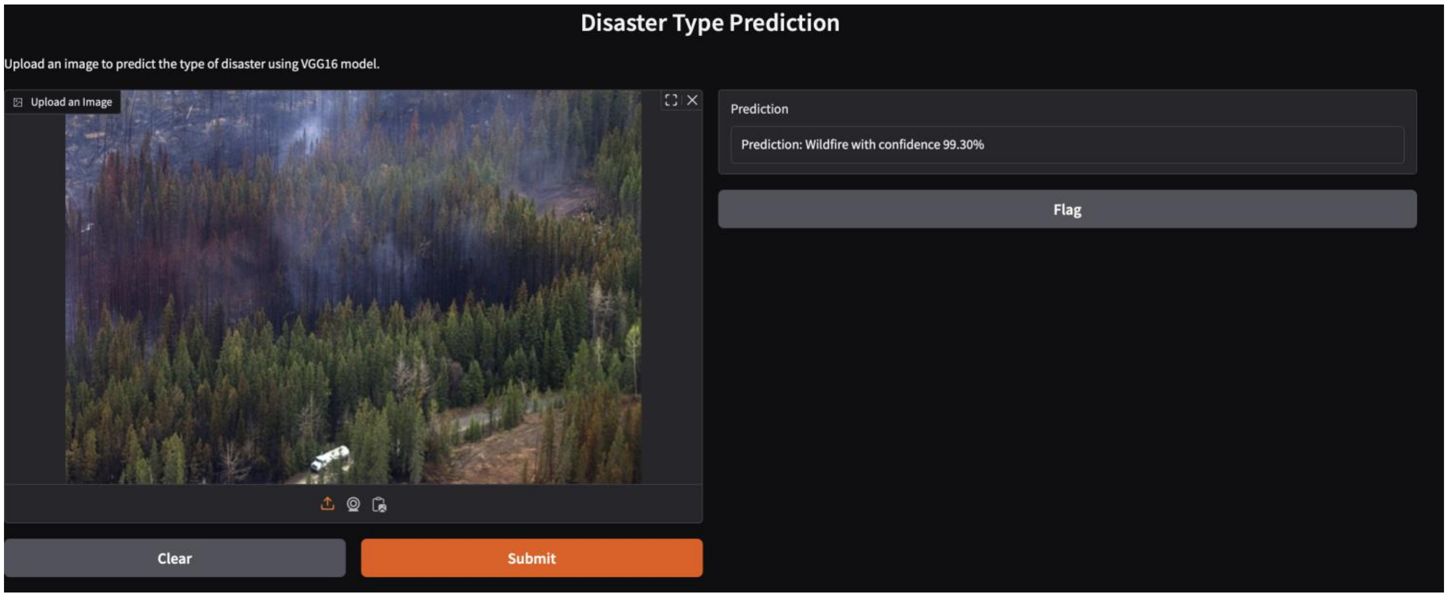
Precipitation (inches): 2.5 (0 – 20)

Cooling Days: 150 (0 – 365)

FIPS County Code: 100 (0 – 500)

Predicted Disaster Type: Severe Storm

Image Predictions: Uploaded images are classified disaster types using the DL models (CNN, VGG16, ResNet50). With confidence scores, users instantly know floods, wildfires, or hurricanes are disasters.


Disaster Type Prediction

Upload an image to predict the type of disaster using VGG16 model.

Prediction: Prediction: Wildfire with confidence 99.30%

6.2 Deployment Steps

Define Prediction Functions: create a function(predict_disaster) which preprocesses inputs, feeds them into the right model (ML for structured data is right; DL for images) , and returns predictions.

Configure Gradio Interface:

Set up Gradio's input and output widgets to match the models' requirements:

- **Inputs:** Structured data and image inputs file upload, Sliders, dropdowns.
- **Outputs:** A text box displaying predicted disaster type and confidence score.

7. Outputs and Artifacts

7.1 Models

- **Random Forest Model (random_forest_model.pkl):** We trained over structured data for predicting disaster categories with high accuracy.
- **VGG16 Model (final_vgg16_model.h5):** Applied to disaster classification using transfer learning, optimized for image-based applications.
- **ResNet50 Model (final_resnet50_model.h5):** Designed for robust disaster image classification in very large databases.

7.2 Deployment Files

Gradio UI Script: It incorporates an interactive interface for real time predictions using structured data and disaster images. It also includes input configurations, model integration and displays of outputs.

7.3 Pre-processed Data

- **Balanced Datasets:** Corrected structured data with anomalies and SMOTE for class balancing, includes.
- **Augmented Image Data:** Images practically set to train, pre-processed and augmented to ensure diversity and improved model generalizability.

8. Troubleshooting and Recommendations

8.1 Common Issues

- **CUDA Errors:** The need to train deep learning models on large dataset requires GPU support. If CUDA-related errors occur, ensure that the GPU is enabled in the Google Colab environment:
  - Go to Runtime > Change Runtime Type and select GPU under hardware accelerator settings.
  - Verify GPU availability with:
  - **import tensorflow as tf print("GPU Available:", tf.config.list_physical_devices('GPU'))**
- **Incorrect File Paths:** Disrupted data loading and model saving is possible from mismatched file paths. Make sure that files path in the script are matching with real Google Drive locations. You should use absolute path, e.g. /content/drive/MyDrive/Project/Data.

8.2 Recommendations

Data Diversity:

This will be improved by incorporating other disaster types as well as other datasets. Suppose adding datasets for earthquakes, tsunamis and landslides is coupled with the model, thereby increasing its applicability to other scenarios.

Model Explainability:

Interpretable models are often what stakeholders prefer. The right techniques for visualizing feature importance and explaining the decision-making process of machine learning models.

Scalability:

Fully expand the Gradio interface to integrate IoT devices or real time satellite feeds. With this, the system could more dynamically give predictions using live weather conditions and disaster imagery.

9. References

Akhloufi, M.A. and Shahbazi, M., 2025. AI for Natural Disasters Detection, Prediction and Modeling. *MDPI*. Available at: https://www.mdpi.com/topics/XJL0V1XRH2.

Beloglazov, A. and Buyya, R., 2015. Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds. *Concurrency and Computation: Practice and Experience*, 27(5), pp.1310–1333.

Feng, G. and Buyya, R., 2016. Maximum revenue-oriented resource allocation in cloud. *IJGUC*, 7(1), pp.12–21.

Huang, L., Li, J., Hao, W. and Zhang, S., 2020. Machine Learning in Disaster Management: Recent Developments in Methods and Applications. *MDPI*. Available at: https://www.mdpi.com/1620898.

Jiang, S., Ma, J., Liu, Z. and Guo, H., 2022. Can we detect trends in natural disaster management with artificial intelligence? A review of modeling practices. *Natural Hazards*, Springer. Available at: https://link.springer.com/article/10.1007/s11069-024-06616-y .

Kune, R., Konugurthi, P., Agarwal, A., Rao, C.R. and Buyya, R., 2016. The anatomy of big data computing. *Software: Practice and Experience*, 46(1), pp.79–105.

Lin, J.T., Melgar, D., Thomas, A.M. and Searcy, J., 2021. Artificial intelligence and machine learning for disaster prediction: a scientometric analysis of highly cited papers. *Natural Hazards*, Springer. Available at: https://link.springer.com/article/10.1007/s11069-020-04429-3.

Shiri, S., Mozhdeh, S. and Correia, A., 2020. Leveraging AI for Natural Disaster Management: Takeaways from the Moroccan Earthquake. *arXiv*. Available at: https://ar5iv.labs.arxiv.org/html/2311.08999.

IEEE Xplore, 2024. Artificial Intelligence for Natural Disaster Management. *IEEE Xplore*. Available at: https://ieeexplore.ieee.org/document/10044583.

IEEE Xplore, 2024. Deep-Learning-Based Aerial Image Classification for Emergency Response Applications Using Unmanned Aerial Vehicles. *IEEE Xplore*. Available at: https://ieeexplore.ieee.org/document/9025436.

IEEE Xplore, 2024. The Use of Artificial Intelligence in Disaster Management - A Systematic Literature Review. *IEEE Xplore*. Available at: https://ieeexplore.ieee.org/document/9032935.

IEEE Xplore, 2024. Multi-task Multimodal Learning for Disaster Situation Assessment. *IEEE Xplore*. Available at: https://ieeexplore.ieee.org/document/9175548.

IEEE Xplore, 2024. Train and Deploy an Image Classifier for Disaster Response. *IEEE Xplore*. Available at: https://ieeexplore.ieee.org/document/9286248 .

Gomes, D.G., Calheiros, R.N. and Tolosana-Calasanz, R., 2015. Introduction to the special issue on cloud computing: Recent developments and challenging issues. *Computers & Electrical Engineering*, 42, pp.31–32.