Senior Design
ENG EC 464

To: Professor Pisano

From: Rishav De (rishavde@bu.edu), Pranet Sharma (pranetsh@bu.edu), Jueun Kang (vkang@bu.edu), Thomas Bowler (tsbowler@bu.edu), Young-Chan Cho (yccho@bu.edu)

Team: Team 1

Date: 03/08/2024

Subject: Team 1 Lab Testing Report #2

1.0 Equipment

FireNet Model

Hardware:

- Computer: A machine with CPU, GPU, and storage for deep learning model training and testing
- Internet Connection: For downloading large datasets and accessing cloud resources

Software:

- Python
  - TensorFlow / Keras: For building and training deep learning models.
  - OpenCV: For image processing.
  - Matplotlib / sklearn: For data visualization and performance evaluation.
- Google Colaboratory
  - For running models in .ipynb files using GPU

Binary Classification Model

Hardware:

- Computer: A machine with CPU, GPU, and storage for deep learning model training and testing
- Internet Connection: For downloading large datasets and accessing cloud resources

Software:

- Python
  - Tensorflow
  - Image Augmentation and Generation
  - Keras
  - Numpy, Scipy
  - Plotly
  - sklearn
  - seaborn
- Google Colaboratory
  - For running models in .ipynb files using GPU

FireDataUS Model

Hardware:

- Computer: A machine with CPU, GPU, and storage for deep learning model training and testing
- Internet Connection: For downloading large datasets and accessing cloud resources

Software:

- Python
  - Tensorflow
  - Image Augmentation and Generation
  - Keras
  - Numpy, Scipy
  - Plotly
  - sklearn
  - seaborn
  - sqlite3
- Google Colaboratory
  - For running models in .ipynb files using GPU

2.0 Setup

2.1 Software Installation and Configuration:

I. Python Installation

II. Library Installations: Install TensorFlow and Keras, the primary libraries for building and training your deep learning model. TensorFlow offers a comprehensive set of tools and libraries for machine learning, while Keras simplifies neural network configurations.

III. OpenCV for Image Processing: Install OpenCV, which allows you to manipulate images and prepare them for modeling, such as resizing and reshaping.

IV. Setting up Google Colab: Google Colab provides powerful computing resources that could be used for processing large datasets and speeding up model training.

V. Data Visualization Tools: Install Matplotlib and sklearn. Matplotlib is necessary for plotting graphs and visualizing data, which is vital for analyzing model performance. Sklearn offers data mining and analysis tools, including performance metrics like confusion matrix.

2.2 Google Drive and Dataset Setup:

I. Google Drive: This is where you'll store your datasets and trained models.

II. Connecting to Google Colab: In your Python scripts, include the code to mount your Google Drive to Google Colab. This step ensures that Colab can access and use the data stored in your Drive.
III. Uploading Dataset: Upload your training and testing datasets to a specific folder in Google Drive. Organize your data in a way that is easily accessible.

3.0 Testing Procedure

FireNet Model

I. Model Loading and Initialization: Start by loading the pre-trained model using Keras.
II. Testing Data Preparation: Use the same function as in training to prepare the test dataset, ensuring it's separate from the training and validation sets. Apply consistent preprocessing, including resizing and normalization.
III. Model Prediction: Run predictions on the test data using the model.predict() function, and store these predictions for analysis.
IV. Performance Evaluation: Construct a confusion matrix from the predictions and actual labels. Calculate key performance metrics such as Accuracy, Precision, Recall, and F-Measure to quantitatively assess the model.
V. Visualization and Analysis: Utilize Matplotlib for visualizing the confusion matrix and other metrics. Conduct an in-depth analysis to identify the model's strengths and areas needing improvement, focusing on biases or misclassifications.
VI. Documentation and Reporting: Document all findings, including performance metrics and insights. Provide clear reporting on the model’s performance and suggest potential improvements or future modifications.

Binary Classification Model

I. Dataset Loading: Start by loading the dataset from Google Drive
II. Dataset Analysis: Observe dataset balance and load into a data frame for further use in model
III. Image Augmentation and Generation: Generate Train, Validation and Test sets for model
IV. Model Creating: Define model and add layers and parameters
V. Model Compile and Fit: Compile and fit model using certain metrics and callbacks
VI. Model Evaluate: Evaluate the model using test data on both dataset sizes
VII. Visualization and Analysis: Use confusion matrices and single image test to show model output and analyze areas of improvement.

FireDataUS Model

I. Model Loading and Initialization: Start by loading the pre-trained model using Keras.

II. Testing Data Preparation: Use the same function as in training to prepare the test dataset, ensuring it's separate from the training and validation sets. Apply consistent preprocessing, including resizing and normalization.

III. Model Prediction: Run predictions on the test data using the model.predict() function, and store these predictions for analysis.

IV. Performance Evaluation: Construct a confusion matrix from the predictions and actual labels. Calculate key performance metrics such as Accuracy, Precision, Recall, and F1-Score to quantitatively assess the model.

V. Visualization and Analysis: Utilize Matplotlib for visualizing the confusion matrix and other metrics. Conduct an in-depth analysis to identify the model's strengths and areas needing improvement, focusing on biases or misclassifications.

VI. Documentation and Reporting: Document all findings, including performance metrics and insights. Provide clear reporting on the model's performance and suggest potential improvements or future modifications.

All the equipment and steps, both the setup and the testing procedure, for this prototype testing were consistent with the test plan.

4.0 Measurements

FireNet Model

i. Precision: The ratio of true positive samples among all samples predicted as positive.
    1. TP / (TP + FP), where TP is True Positives and FP is False Positives.

ii. Recall: The ratio of true positive samples detected among all actual positive samples.
    1. TP / (TP + FN), where FN is False Negatives

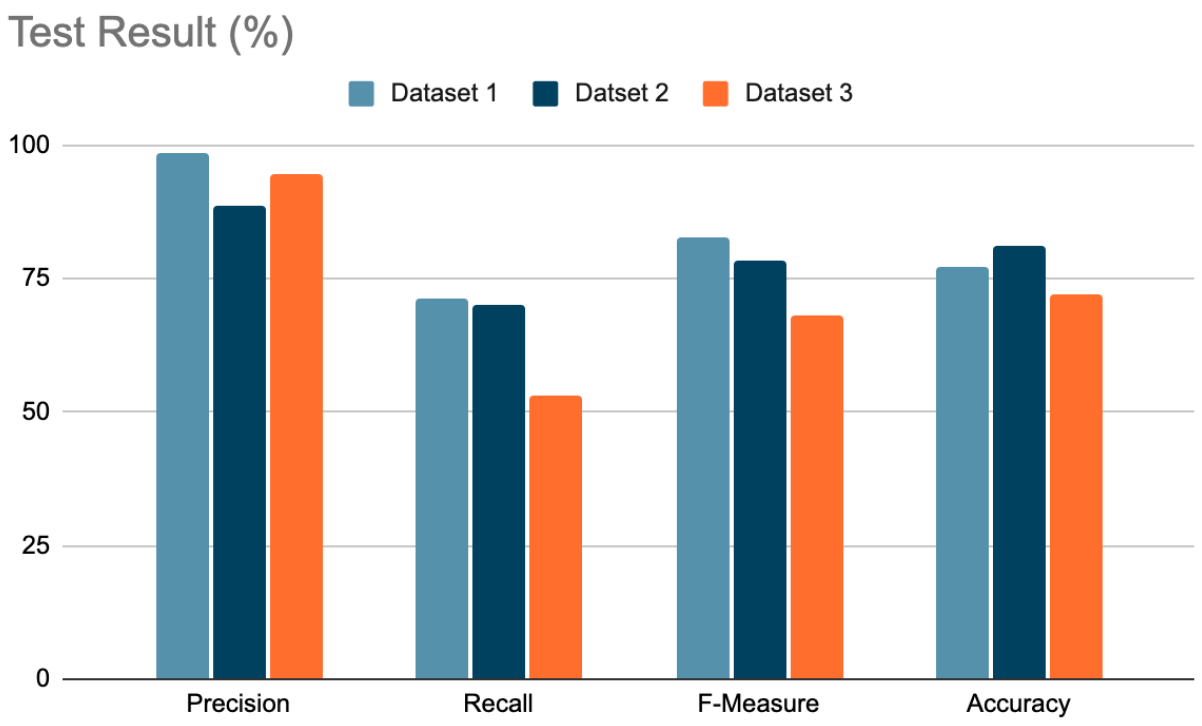
iii. F-Measure: The harmonic mean of Precision and Recall.
    1. 2 * (Precision * Recall) / (Precision + Recall)

iv. Accuracy: The percentage of samples correctly predicted by the model.
    1. (TP + TN) / Total number of cases

	Dataset 1	**Correct**	**Dataset 2**	**Correct**	**Dataset 3**	**Correct**
Precision	98.67%	Yes	88.52%	Yes	94.44%	Yes
Recall	71.37%	Yes	70.08%	Yes	53.13%	No
F-Measure	82.83%	Yes	78.22%	Yes	68.00%	No

Accuracy	77.26%	Yes	81.07%	Yes	71.93%	Yes

* Correct: >70%



- Dataset 1: FIRE Dataset
  - https://www.kaggle.com/datasets/phylake1337/fire-dataset
- Dataset 2: Fire Images Dataset
  - https://www.kaggle.com/datasets/rachadlakis/firedataset-jpg-224
- Dataset 3: Forest Fire Images
  - https://www.kaggle.com/datasets/mohnishsaiprasad/forest-fire-images

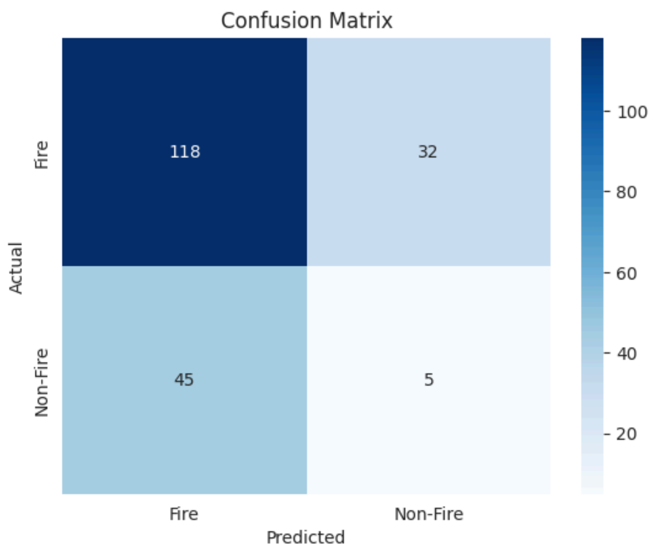
Binary Classification Model

I. Precision: The ratio of true positive samples among all samples predicted as positive.
   A. TP / (TP + FP), where TP is True Positives and FP is False Positives.
II. Recall: The ratio of true positive samples detected among all actual positive samples.
   A. TP / (TP + FN), where FN is False Negatives
III. F1-Score: The harmonic mean of Precision and Recall.
   A. 2 * (Precision * Recall) / (Precision + Recall)
IV. Accuracy: The percentage of samples correctly predicted by the model.

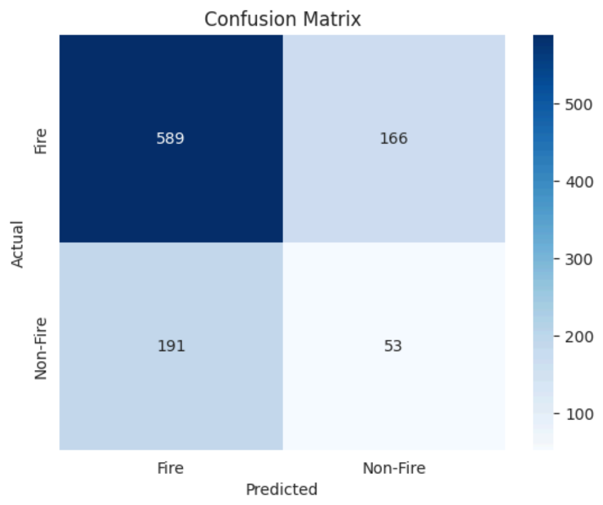
A. (TP + TN) / Total number of cases

These criteria have been selected to better understand how well the model is performing when compared to similar models.

	Prediction DS 1	Prediction DS 2	Correct (Y/N
Precision	Fire >0.70 Non-Fire >0.10	Fire >0.75 Non-Fire >0.20	Y Y
Recall	Fire >0.70 Non-fire >0.10	Fire >0.75 Non-fire >0.20	Y Y
F1-Score	Fire >0.70 Non-fire >0.10	Fire >0.75 Non-fire >0.20	Y Y
Accuracy	>0.60	>0.60	Y



Prediction DS 1



Prediction DS 2

FireDataUS Model

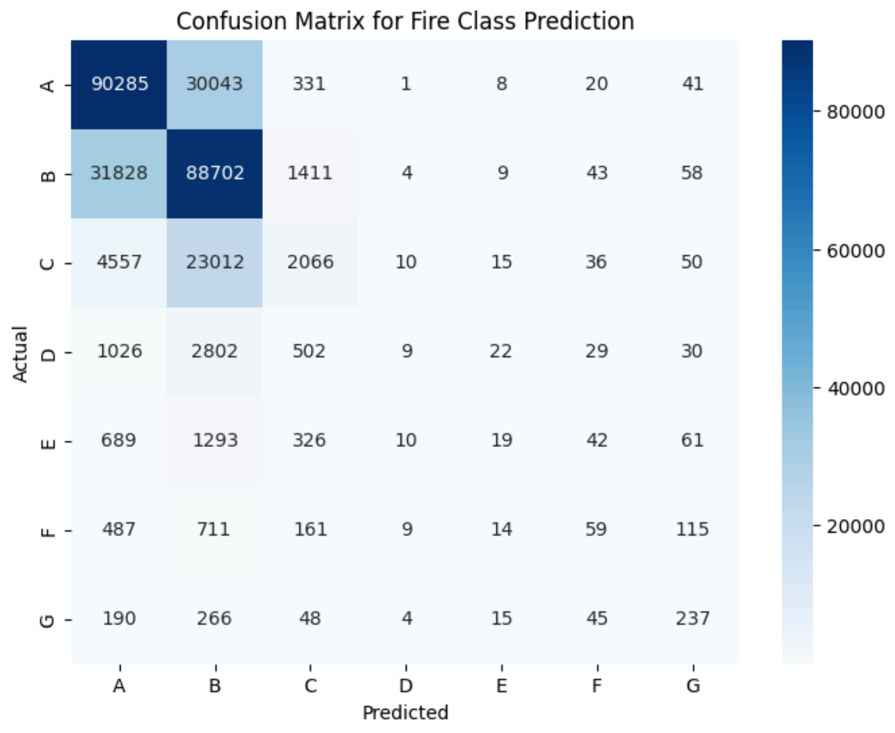
1. Precision: The ratio of true positive samples among all samples predicted as positive.
   a. TP / (TP + FP), where TP is True Positives and FP is False Positives.
2. Recall: The ratio of true positive samples detected among all actual positive samples.
   a. TP / (TP + FN), where FN is False Negatives
3. F1-Score: The harmonic mean of Precision and Recall.
   a. 2 * (Precision * Recall) / (Precision + Recall)
4. Accuracy: The percentage of samples correctly predicted by the model.
   a. (TP + TN) / Total number of cases

These criteria have been selected to better understand how well the model is performing when compared to similar models.

Different Fire classes will have different scores. Since data is heavily imbalanced, looking only at class A and B.

	Prediction	**Correct (Y/N**
Precision	A >0.70 B >0.60	Y Y
Recall	A >0.70 B >0.60	Y Y
F1-Score	A >0.70 B >0.60	Y Y
Accuracy	>0.60	Y

Dataset - https://www.fs.usda.gov/rds/archive/catalog/RDS-2013-0009.6



	A	B	C	D	E	F	G
A	90285	30043	331	1	8	20	41
B	31828	88702	1411	4	9	43	58
C	4557	23012	2066	10	15	36	50
D	1026	2802	502	9	22	29	30
E	689	1293	326	10	19	42	61
F	487	711	161	9	14	59	115
G	190	266	48	4	15	45	237

5.0 Conclusions

FireNet Model

The model demonstrates high precision across each of the 3 datasets. However, Dataset 3 falls below the acceptable recall threshold, indicating potential missed positive cases which is concerning in the context of fire detection. While F-Measure exceeds 70% for Datasets 1 and 2, Dataset 3 fails to reach the threshold which indicates reduced effectiveness in the particular conditions present in Dataset 3.

Targeted improvements are necessary to enhance recall without compromising precision. Strategies like threshold tuning and cost-sensitive learning could address these challenges. Also, investigating Dataset 3's characteristics can inform necessary model adjustments. Despite Dataset 3's issues, the model shows potential for generalization across diverse conditions, emphasizing the importance of addressing weaknesses and continuous testing for real-world reliability.

Future work should focus on refining the model with more sophisticated architectures, data augmentation, and advanced training strategies. Exploring different hyperparameter settings and incorporating feedback loops for continuous learning could further boost performance. Overall, improvements can be realized by adapting to diverse data characteristics and extending the model's applicability to broader scenarios.

Binary Classification Model

Testing data shows that the current model works within expectations. Due to class imbalances, the false positive rate is greater than the false negative rate, which is acceptable since we would rather a model classify a non-fire as a fire than mistake a fire for a non-fire. Thus, there is room for improvement in the precision of the classification. The model does, however, return a good accuracy estimate when tested on test data, and the majority of the fire images are classified correctly. The results are better when using a larger dataset to train and test, thus theoretically the accuracy could increase given much more data.

FireDataUS Model

The testing data shows that there is a significant class imbalance issue. The dataset has many more examples of smaller fires (A and B) than bigger fires. This causes the predictions for bigger classes to be much more inaccurate. However, using historical fire data and looking at the data we have a lot (class A and B fires) show that it is very accurate in predicting fire class. The inaccurate predictions could be fixed by implementing class imbalance handlers, but augmenting the dataset with generated data would not be effective since this is historical data, so other options like focal loss could be looked into. The model is highly precise when it comes to small fires, and future improvements should increase precision for different data. Exploring the impact of different hyperparameter settings and incorporating feedback loops for continuous model learning could further enhance performance, and be used to improve overall model accuracy. In conclusion, it appears historical fire data can be a reliable indicator of when fires may occur.