WILDFIRE SPREAD FINAL REPORT

Yizhou(Linda) Cai
Student# 1009035321
yz.cai@mail.utoronto.ca

Vanessa Lu
Student# 1008979116
van.lu@mail.utoronto.ca

Jiaxin(Audrey) Tian
Student# 1009089567
audjiaxin.tian@mail.utoronto.ca

Sicheng(Harry) Zhou
Student# 1008828299
harrysc.zhou@mail.utoronto.ca

ABSTRACT

This report summarizes our team's deep learning wildfire spread prediction model. It covers background and related works, data processing, architecture, baseline, results (quantitative and qualitative), and model evaluation.

—-Total Pages: 9

1 LINK TO COLAB

The following link is the addressed to our project in Google Colab. https://colab.research.google.com/drive/1-5T6MIAU7YZwujaJg-4TWDosirF_YlbV?usp=sharing

2 INTRODUCTION

With the annual summer wildfires increasing in frequency and severity, wildfires and their spread have become prevalent concern in the world today. To tackle this issue, our team created a deep learning project to forecast wildfire spread. This model takes in sequential image data from a dataset of past and present fire masks and environmental factors like wind patterns, rainfall, and temperature.

To handle large number of samples and diverse variables of the image data, deep learning is the ideal model for analyzing large and complex datasets with many features, providing more accurate predictions than statistical methods. The sequential nature of the datasets make it challenging to capture wildfire spread, therefore we employed transfer learning and a recurrent neural network (RNN). RNNs have demonstrated proficiency in sequential data predictions. Our model outputs a single map image indicating the spread of fire based on the input images of environmental factors and previous day's fire map.

Through precise fire forecasts, our project aims to offer early warnings, facilitating timely evacuations for life and property protection. Additionally, this aids in directing firefighting resources to high-risk zones, reducing risks for firefighters. This proactive strategy enhances wildfire containment, safeguarding forests and wildlife.

3 BACKGROUND AND RELATED WORK

1. Predictive Modeling of Wildfires: A New Dataset and Machine Learning Approach [1]
   This paper combines Big Data, Remote Sensing, and Data Mining to predict wildfires using MODIS satellite images, achieving 98.32% accuracy, surpassing existing systems.
2. A Multi-modal Wildfire Prediction and Early Warning System Based on A Novel Machine Learning Framework [2]

This study devises a spatio-temporal machine learning for wildfire prediction, reaching 97% accuracy in 2018, anticipating large fires and proving life/environmental protection potential.

3. Wildfire Risk Prediction and Detection using Machine Learning California [3]

   This study creates a machine-learning-based fire risk model with geographic factors, achieving 100% risk prediction accuracy via ensemble analysis; faster R-CNN also proves effective with 93% fire detection accuracy, enhancing prevention and real-time identification.

4. Machine learning predicts how big wildfires will get [4]

   Researchers built a machine learning algorithm predicting wildfire size post-ignition; analyzed climate, atmosphere, vegetation, achieving 50% success; vital factors: vapor pressure deficit, nearby black spruce percentage.

5. A newly developed AI-based method can accurately predict wildfires [5]

   This study combines deep learning, weather forecasts for advanced wildfire danger prediction; hybrid AI-weather technique enhances accuracy, cost-effective for effective fire management.

4 Data Processing

For our data, we used the Next Day Wildfire Spread dataset, which contained sets of 13 remote-sensing images of past wildfires in the US [6]. 11 of the 13 images represent environmental factors such as wind speed, minimum and maximum temperatures, and precipitation, and the remaining two are the previous and current days' fire masks [Figure 1]. For the purposes of our project, the current fire masks serve as the labels for each set of data as we are trying to predict the spread of wildfire. Due to the long training time and computation power our model requires, only half of the available data files, which translates to 7 out of the 15 available training data files and 1 of the 2 available sets for both our validation and testing data.

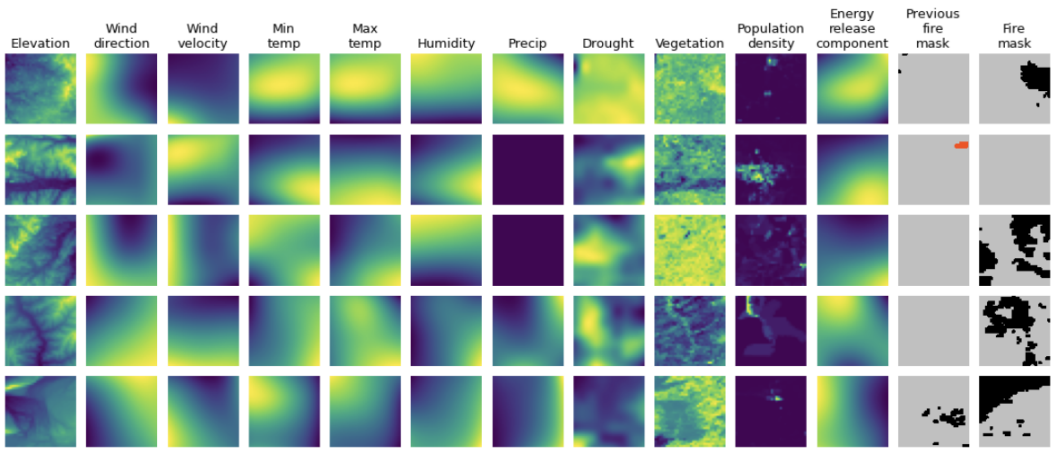

Figure 1: Sample of 5 sets of training images

Our main goals for processing this data were is to convert the TensorFlow Record format of the dataset to Pytorch numpy arrays such that it can be passed into our AlexNet transfer learning network. Additionally, data processing removes any uncertain data due to environmental factors like cloud coverage, which were denoted by gray pixels in the images [Figure 2].

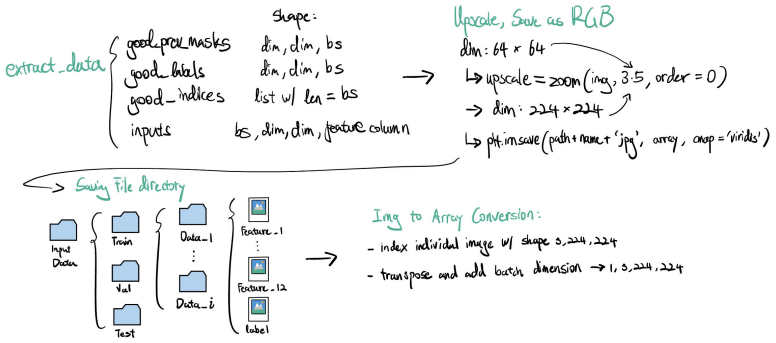

Figure 2: Flow of data through loading and cleaning processes

To accomplish our first step, we wrote a function, extract_data, that iterates through the inputted dataset and extracts the TFRecord data as 12 inputs and 1 label, returning them in list object with each list consisting of all features per data of the dataset, a list of all the previous fire masks, a list of their corresponding labels, and lastly the input itself. We then concatenated the data into numpy arrays with shape of # of data x width x height x 13. Since there were 7 training files, we had to concatenate the training arrays again into one array after the previous step. We cleaned the data such that there are 483 sets of data (483x13 = 6279 images) in the training array, 73 sets (73x13 = 949 images) in the validation array, and 67 sets (67x13 = 871 images) in the test array. This translates to a 77%:12%:11% ratio for the train/val/test split.

Our next step was upscaling the images from 64x64 to 224x224. Each image was then expanded to have 3 RGB channels, which we accomplished by using plt.imsave and color mapping them using either the 'viridis' colormap for the 11 environmental factor images or a custom colormap for the 2 fire masks. This also saved the transformed input images to a folder in our Google Drive, with each saved image having a final shape of 224x224x3; with a simple transform and numpy to tensor conversion, these images could now be used in transfer learning.

Lastly, we cleaned the data that have gray pixels by iterating through and locating the fire masks that had gray RGB values of [192,192,192], then removing them from the Google Drive folder. This cleaning cut down our available data to a final train/val/test split ratio of 75%:13%:12%. An example of as set of cleaned data is shown below in Figure 3.

Figure 3: Example of 1 set of cleaned and processed data in Google Drive

5 ARCHITECTURE

The final fire prediction model consists of a transfer learning portion with pre-trained AlexNet, gated recurrent unit (Gru) layer, and a combination of transposed convolution and upsampling layers. The transfer learning part serves as an encoder, transforming each image into embeddings. The pre-trained AlexNet with frozen parameters is used to reduce the computational cost of the model. It would be impossible for each image feature of a data to be passed through its own separate convolutional layer. Next, a single layer Gru model is used to learn the 12 image sequential data associated with each label. The Gru model is able to account how individual features affect the next day's fire outcome. Lastly, a six layer decoder with three transposed convolution and upsampling layers in alternating sequence is employed to construct an image of the fire's predicted growth.

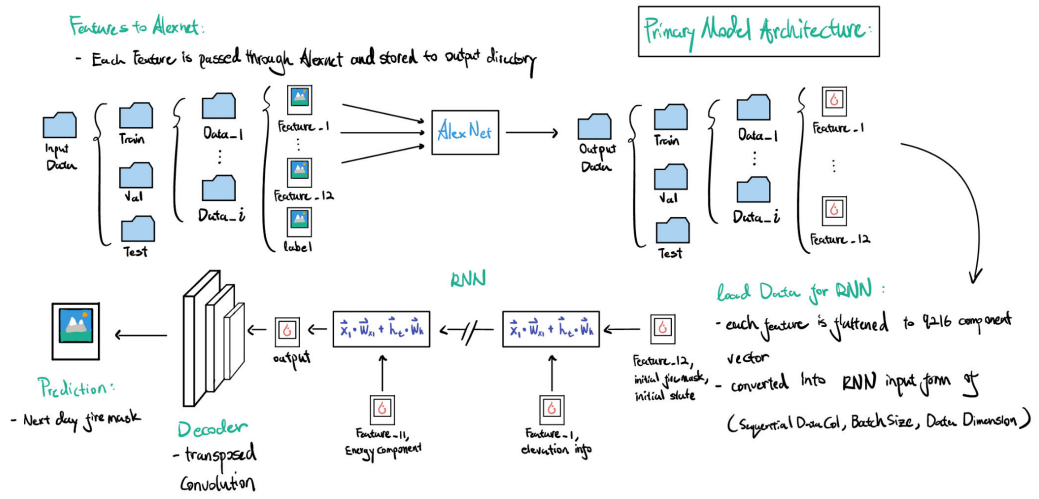

Figure 4: Neural network architecture illustration

6 Baseline Model

Our baseline model is a self-coded mathematical approach using the average fire size from previous fire masks as a reference. This model is chosen due to its ability to predict fire spread like the primary model, analyzing growth statistically for reliable comparison with complex deep learning approaches; its simple mechanism aids easy setup, thus deemed suitable for our project. The function "avg_red()" calculates this average. Another function, "pred_correct()", obtains fire size from previous and current masks. If prior fire is smaller, it might shrink and the same goes the opposite. We compare growth prediction to labeled fire size change. If they match, it's a correct prediction. The model forms a list of predicted outcomes (True/False) and calculates accuracy—correct prediction percentage.

```
153] def avg_red(dataset):
       red = 0
       count = 0
       for data, label in dataset:
         red += np.count_nonzero(data == 255)
         count += 1

       return red/count

     def pred_correct(data, label, average):
       red_data = np.count_nonzero(data == 255)
       red_label = np.count_nonzero(label == 255)
       if (red_data < average) and (red_label < red_data):
           return True
       elif (red_data > average) and (red_label > red_data):
           return True
       elif (red_data == average) and (red_label == red_data):
           return True
       else:
           return False
```

```
prediction = []
average_red = avg_red(training_baseline)
for data, label in training_baseline:
    pred = pred_correct(data, label, average_red)
    prediction.append(pred)

print("Number of correct predictions:", sum(prediction))
print("Number of incorrect predictions:", len(prediction)-sum(prediction))

print("The training accuracy of our baseline model is", sum(prediction)/len(prediction))
```

```
Number of correct predictions: 115
Number of incorrect predictions: 155
The training accuracy of our baseline model is 0.42592592592592593
```

Figure 5: Baseline model code

7 Quantitative Result

When evaluating training performance, a custom accuracy function shown below is coded for the model.

$$Accuracy(y, \hat{y}) = \frac{1}{\frac{1}{n}\sum_{k=1}^{n} |y_i - \hat{y}_i|}$$

The function's output is not bounded between [0, 1] but $\forall x \in \mathbb{R}, x \geq 0$

The function takes the average absolute difference between each pixel of label y and prediction y hat then passes the average through a rational function. The reason being that when outputting a

prediction, the majority of the pixel on the canvas will be black, and it is easy for the model to learn to reconstruct images with black pixels. This will lead to a high accuracy of the model but it does not actually reflect the model's actual performance because the pixel representing fire is only a small part of the image. The rational function penalizes the model so that it pays more attention to the fire pixels. It will reward the model for achieving a smaller averaged difference between the label and prediction by returning a larger accuracy because Accuracy = $\lim_{difference \to 0^+} \frac{1}{difference} \to \infty$.

At epoch 300, we have a training loss of around 118.03, and a validation loss of around 137.43. Although both these numbers are high compared to the losses seen in past course works, the training loss started above 1500 and the validation loss started around 3000, hence showing our training has been successful. As shown in figure 6, both losses dropped to a plateau at around 100 epochs.

```
Epoch 300: Train acc: 413.8586388075217, Train loss: 118.03093719482422 |Validation acc: 473.1545104072314, Validation loss: 13
7.42868041992188
Finished Training
Total time elapsed: 23510.36 seconds
```

Figure 6: Final training and validation results

Our model has a final training accuracy of around 413.86, and a final validation accuracy of around 473.15. The accuracies are not in traditional percentages bound between 0 and 1, but rather numbers that can increase infinitely. The Train vs. Validation Accuracy graph could indicate that our model does not overfit, as the validation accuracy is higher than training accuracy.

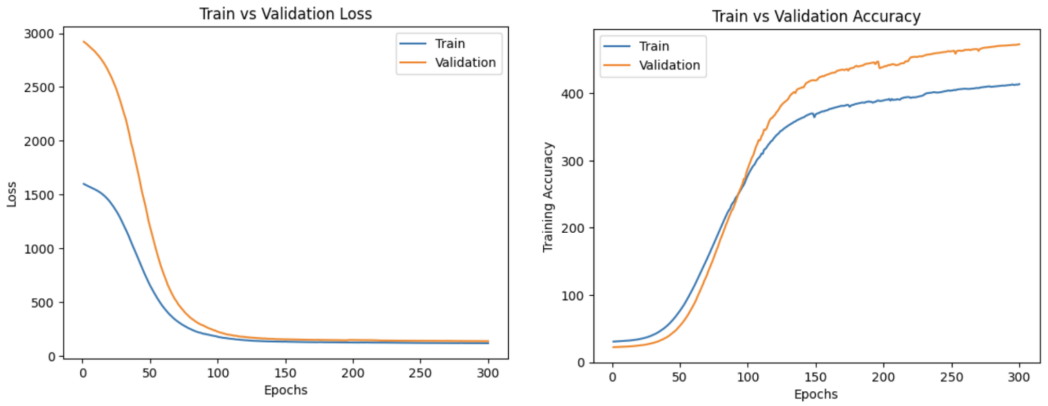

Figure 7: Training and validation loss and accuracy against the number of epochs

8 Qualitative Result

Our model performed the best in two scenarios, with the first being when the label is an all black image, which represents the wild fire has died. The comparison between the label and the generated output for this instance is shown on the left. The second scenario is when the fire has spread over the entire region, and the label is an all red image; the comparison between this label and our generated output is shown on the right. Due to the large amount of noises from the 12 feature inputs, the non-fire prediction contains scattered blue pixels, while the all-fire is yellow instead of red. This difference in colour is reflected in how our model's loss plateaus at 100 after roughly 200 epochs. This discrepancy in colour does not hinder our model's predicting capability.

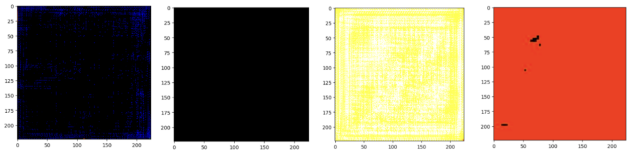

Figure 8: Examples of good performance

On the other hand, our model does not do well when faced with the problem of predicting very small amounts of fire scattered around the region, often over-predicting the spread of the wildfire in those cases. Some instances of this are shown in the figure below. However, this result may actually be beneficial since an overreaction to a fire will still allow for emergency personnel to arrive on scene more urgently and control its spread faster, and it is always better for a fire to be located than for it to be missed.

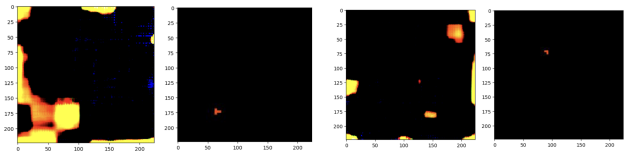

Figure 9: Examples of bad performance

The figure below shows the best results from training our model. As depicted, our model was able to learn and generate predictions that represented the location of the fire spread, with the yellow sections and blue gradients showing fire masks that closely match that of the label. Hence, our model successfully achieves our project's goal of forecasting wildfire spread.

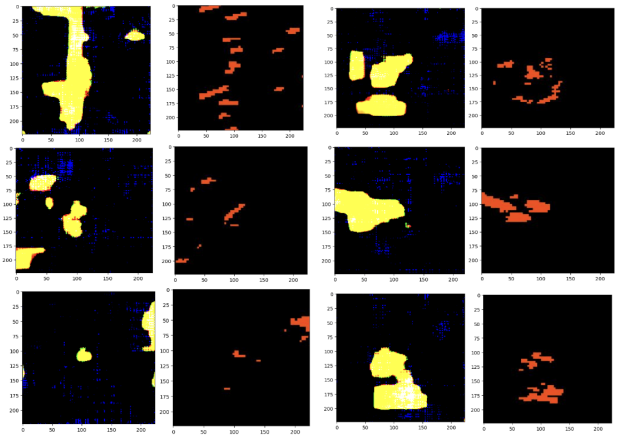

Figure 10: Best 6 sets of fire prediction results from training

9 Model Evaluation on New Data

To ensure the quality of evaluation of our model, A set of testing data that is completely unknown to our model from the training and validation process is needed. Due to the uniqueness of our data (consisting of 11 environmental factor images and 2 fire masks), we were unable to collect or create brand new data to use in final evaluation. Therefore, we had to use the provided data from Kaggle for testing.

The first problem was that the two test sets provided (Figure 11) contained the exact same data and so are the two evaluation data sets in the dataset. Our solution was to use the last training set amongst the 15 provided as our new test set and renamed it as test_02.

However, more issues emerged when we first checked our qualitative results. The data in test_02 had several overlaps with the 7 training sets we used. To solve this, we wrote a function eliminate_duplicate (Figure 12) that compared the existing data in our training set with our test set, and deleted any duplicates found within the test set. We also expanded the initial test set size prior to cleaning to 295 by combining it with the last three training sets 'train_11', 'train_12', and 'train_13'(Figure #). After applying this function, we ended up with our final test set for evaluation, which consisted of 38 sets of data.

Figure 11: All Kaggle files prior to cleaning

```
def eliminate_duplicate(train_loader, test_loader, train_name, test_name):
  count = 0
  num = 0
  folder_path = '/content/drive/MyDrive/Colab Notebooks/APS360 Project/Input'
  for data, label in test_loader:
    test_data = data
    for data1, label1 in train_loader:
      train_data = data1
      if np.array_equal(test_data, train_data):
        dup_direct = folder_path + '/' + test_name + '/data_' + str(count)
        if os.path.exists(dup_direct):
          for root, dirs, files in os.walk(dup_direct):
          # For each file in the directory
            for file in files:
              # Construct the full path to the file
              file_path = os.path.join(root, file)
              # Delete the file
              os.remove(file_path)
              # For each subdirectory in the directory
            for dir in dirs:
              # Construct the full path to the subdirectory
              dir_path = os.path.join(root, dir)
              # Delete the subdirectory
              os.rmdir(dir_path)
          # Delete the top-level directory
          os.rmdir(dup_direct)

          num += 1
    count += 1
  print("Number of files that are duplicates:", num)
```

Figure 12: eliminate_duplicate function code

By applying our trained model on the newly created test_unseen_data set, we were able to obtain both qualitative and quantitative results. The test predictions successfully depict the general shapes of the fire masks in yellow, showing high quality performance with small fire spread located in the middle of the images. The test accuracy is 465.92, which is between our training and validation accuracies.

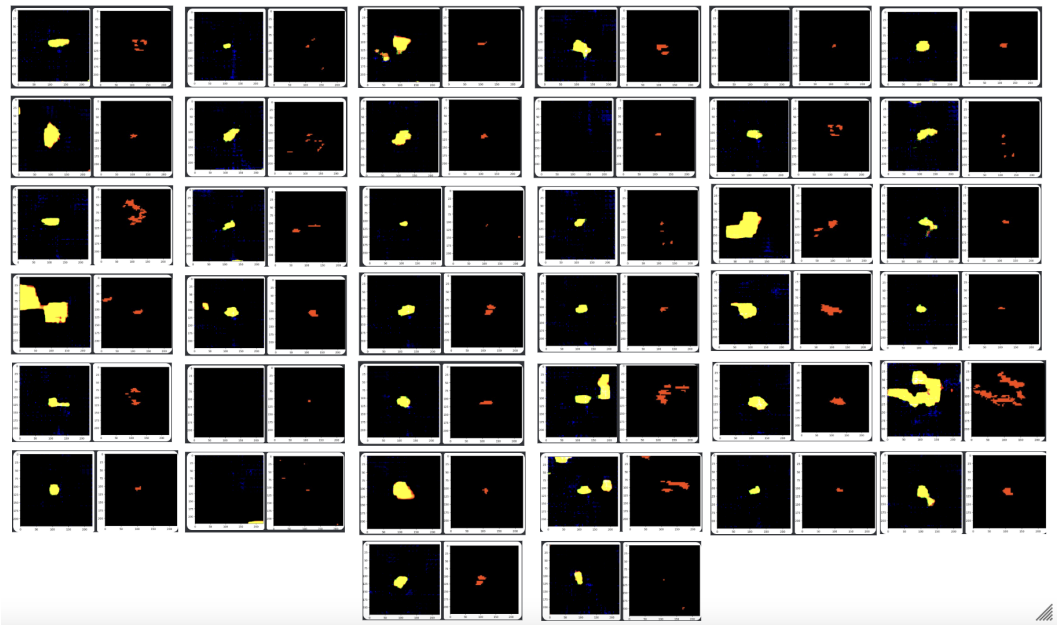

Figure 13: 38 sets of fire predictions in test_unseen

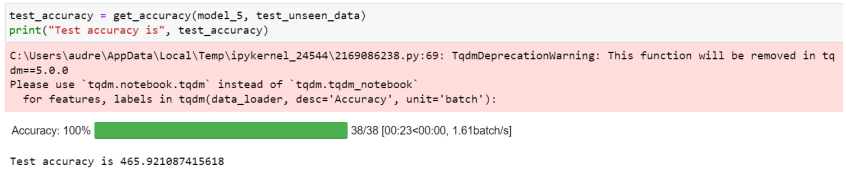

Figure 14: The test accuracy of the model

10 Discussion

This project is extremely challenging. First, the model being a supervised generative one is already inherently complex. On top of that, recurrent layer is added to the model to take multiple inputs. In additionally, these sequential data are images that requires separate embedding. There are nearly no precedent models in the machine learning community that have similar architectural concepts and the project was built without any references.

From the qualitative and testing results, we conclude that our model is performing adequately in predicting wildfire spread. Both our loss and accuracy have been optimized by our meticulous hyperparameter tuning and lengthy training, which included changing the number of epochs, the learning rate, the type of recurrent layer (RNN vs Gru), neural network depth, and decoder architecture. This resulted in loss decreasing and accuracy increasing by significant amounts over 300 epochs, reflecting the successful learning done by the model. The model's success in generating likely fire masks based on input data is also evident in the high similarity between the model's prediction and the corresponding label. Our model is performing better than we expected considering its complex architecture, which we believe is due to our combination of Upsample and ConvTranspose in our decoder. This was especially evident since training time decreased from 9 hours to 6 hours after adding Upsample layers. The upgraded model also brought notable changes to the color accuracy of our predictions, as there was more red instead of yellow pixels that denoted a fire.

Throughout this project, we gained insights from both advancements and mistakes. In terms of applications, we learned to self-code multiple functions for data cleaning and loading, model training, and evaluation. Additionally, we learned how color maps could be used to modify and normalize colours for an image. Reflecting on our errors, and we also discovered that using transposed convolution in the decoder introduces more noise during training.

From our test set, we conclude that our true positive and false negative rate is high. This implies that our model is able to predict if the fire grow to a larger size, but is inapt in predicting if the fire will become smaller or extinguished. Hence, the model still requires more training and modification on its architecture. If used in a real life setting, the model will incorrectly predict the fire being extinguished when it is still a blaze in smaller sizes. This may lead to an incorrect allocation of resources and firefighters.

11 Ethical Consideration

Since we are dealing with wildfires, we must keep in mind the implications associated with the damage of wildfires. First, our project would contribute to the protection of lives, property, and the environment, so our model must be accurate in order to ensure the validity of our predictions. Second, predicting the spread of wildfires would allow for more efficient allocation of resources and rebuilding of infrastructure, which can lead to more awareness and prevention of wildfires. On the other hand, it must be noted that our model can also be used maliciously by those who wish to harm the environment (such as using it to start a fire that would spread very quickly). Overall, these ethical concerns should form the basis of our team values when approaching this project, and will thus inform the decisions and justifications we make.

As for the limitations on our model, a recurrent neural network is prone to the exploding and vanishing gradient problems, and since it is recurrent, computation can be slow. As such, we were only able to use half of the dataset due to the large amount of time it took to train our model, which meant that our loss and accuracy may not have been as optimized as they could have been if it was trained

on the entire dataset. Lastly, since our training data is sequential, it is limited in the fact that random searching is not possible. This means that we cannot select the best of random combinations of hyperparameters when training the model.

REFERENCES

[1] Y. O. Sayad, H. Mousannif, and H. Al Moatassime, "Predictive modeling of wildfires: A new dataset and machine learning approach," Fire Safety Journal, vol. 104, pp. 130–146, Mar. 2019, https://www.sciencedirect.com/science/article/pii/S0379711218303941.

[2] R. T. Bhowmik, Y. S. Jung, J. A. Aguilera, M. Prunicki, and K. Nadeau, "A multi-modal wildfire prediction and early-warning system based on a novel machine learning framework," Journal of Environmental Management, vol. 341, p. 117908, Sep. 2023, https://www.sciencedirect.com/science/article/pii/S0301479723006965.

[3] A. Malik, N. Jalin, S. Rani, P. Singhal, S. Jain, and J. Gao, "Wildfire Risk Prediction and Detection using Machine Learning in San Diego, California," 2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI), Oct. 2021, https://ieeexplore.ieee.org/document/9604370.

[4] B. B.-U. Irvine, "Machine learning predicts how big wildfires will get," Futurity, Sep. 18, 2019. [Online]. https://www.futurity.org/wildfires-machine-learning-prediction-2163272/.

[5] B. Grimes, "A newly developed AI-based method can accurately predict wildfires," interestingengineering.com, Oct. 24, 2022. [Online]. https://interestingengineering.com/science/a-newly-developed-ai-based-method-can-accurately-predict-wildfires.

[6] "Next Day Wildfire Spread: A Data Set to Predict Wildfire Spreading from Remote-Sensing Data" www.kaggle.com [Online]. https://www.kaggle.com/datasets/fantineh/next-day-wildfire-spread.

[7] "Data Reader and Visualization: Python" www.kaggle.com [Online]. https://www.kaggle.com/code/fantineh/data-reader-and-visualization.