CS230

Wildfire Prediction Using Deep Learning Models for Remote Sensing Data

Project Category: Computer Vision

Raghav Sharma*
Department of Civil and Environmental Engineering
Stanford University
raghavsh@stanford.edu
SUNet ID: raghavsh

Rishabh Aggarwal†
Graduate School of Business
Stanford University
arishabh@stanford.edu
SUNet ID: arishabh

Abstract

Predicting extreme weather events accurately is essential to implement consequential disaster management strategies. In this project, we use remote sensing data of historical wildfires aggregated using various publicly available data sources available in Google Earth Engine. Given the features influencing wildfire, we predict where wildfire will spread the next day. We frame wildfire prediction problem as an image segmentation task and employ a network similar to U-Net convolutional neural network to exploit the spatial features of the remote sensing data. We find that the dice coefficient of the trained model is 21.5% and the IoU is around 12.2%. In additional analysis, we find that this input feature rich model is less prone to over-fitting and excluding highly correlated features do not improve test set metrics.

1 Introduction

Climate change is making extreme weather events more probable and it is becoming increasingly important to estimate extreme event risk to build informed resilience. Wildfires are one such event of extreme importance, especially in California where wildfire risk is high. For authorities to implement consequential wildfire management, accurate estimation of wildfire likelihood is essential.

We propose a deep learning model for predicting wildfire occurrence based on remote sensing data of factors influencing wildfires such as topography, max/min temperature, precipitation, drought index, wind speed, Normalized Difference Vegetation Index (NDVI), and humidity. Given the features influencing wildfire and the location of the fire, we predict the spread of wildfire the *next* day.

2 Related Work

The main source of remote-sensing data is Google Earth Engine which is also used by Huot et al. [2020, 2021]. As a result, Huot et al. [2021] is the closest paper and forms the basis of our analysis. In addition, Prapas et al. [2021] selects learning algorithms chosen carefully based on the type of dataset used (spatial vs temporal) and serves as a guide to model architecture. Similarly, Sayad et al. [2019] evaluate their models via various metrics whereas Ghorbanzadeh et al. [2019], Pham et al.

*We sincerely thank Fantine Hout for her guidance for the project.
†The code for the project is posted on Github.

[2020] provide validation and performance evaluation techniques for different models. Further, Jain et al. [2020] provides an exhaustive review of the machine learning applications in wildfire science and Barmpoutis et al. [2020] reviews the literature on early wildfire detection using remote sensing.

3 Dataset and Features

We use the 'Next Day Wildfire Spread' data set compiled by Huot et al. [2021] from various data sources available through Google Earth Engine (GEE).[3] Here, we provide a brief description of the dataset and the features.[4]

The dataset combines historical wildfire events with remote sensing images from GEE across United States from 2012 to 2020. The data is extracted as images of 64km $\times$ 64km regions at 1 km resolution and includes input features that influence wildfire. There are 11 input features: elevation, wind direction and wind speed, minimum and maximum temperatures, humidity, precipitation, drought index, normalized difference vegetation index (NDVI), population density and energy release component (ERC). Although the data sources, that the input features are extracted from, have different spatial resolutions, all the data is aligned to 1 km resolution, which corresponds to the spatial resolution of the fire masks.

The historical wildfire dataset is processed to represent the fire information as a fire mask over each 64km $\times$ 64km region, showing the locations of 'fire' versus 'no fire', with an extra classification for uncertain labels and includes both the fire mask at time t, denoted as 'previous fire mask' and at time $t+1$ day, denoted as 'fire mask' to provide two snapshots of the fire spreading pattern. Fires separated by more than 10 km are considered to be belonging to a different fire. For the purpose of machine learning algorithm, the fire mask at time t is considered as an input feature and the 'fire mask' at time $t+1$ day as labels. For predicting fire spreading, only the samples for which the 'previous fire mask' contains any fire are kept while we drop the samples containing 'uncertain' labels.

The objective is to predict where the fire will spread on date $t+1$ given input features at date t.

3.1 Data Preprocessing

There are three main pre-processing elements added to the dataset. First, the input features, except the fire masks, are clipped at 0.1% and 99.9% percentiles for each feature. Second, each feature is then normalized by subtracting the mean and dividing by the standard deviation. These statistics are computed over the training set after clipping. Third, the dataset is augmented by randomly cropping 32 km $\times$ 32 km from the original 64 km $\times$ 64 km regions. The final dataset consists of 13,602 examples. The dataset is split between training, development and test sets according to 8:1:1 ratio.

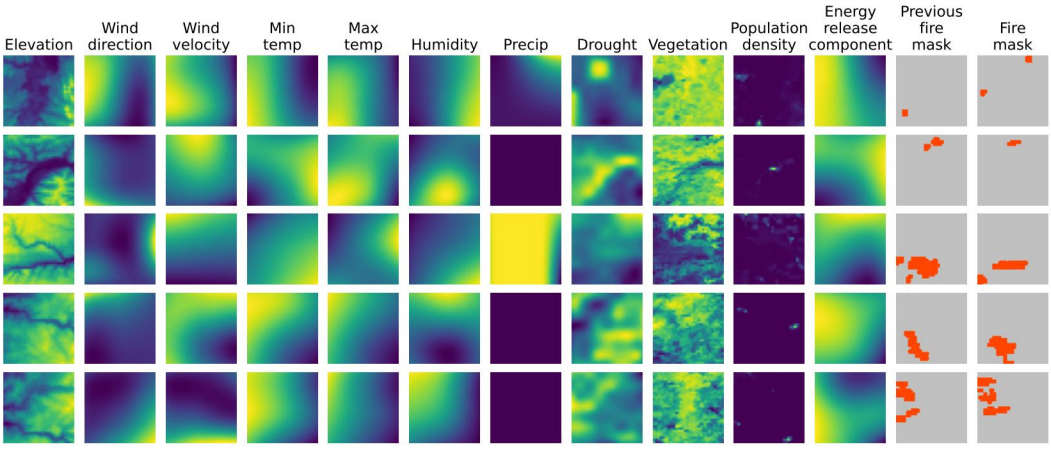


Figure 1: Examples from the dataset

[3]The dataset is made public on Kaggle by the authors.

[4]Further details regarding data sources and aggregation can be found in Huot et al. [2020, 2021].

Figure 1 displays examples from the dataset where each row represent one example of 32 km $\times$ 32 km at 1 km resolution. Each row corresponds to the 11 input features, previous fire mask at time t at a particular location and the fire mask at time $t+1$. As mentioned, the fire mask at date t is considered as an input feature and labeled as the 'previous fire mask' whereas 'fire mask' corresponds to the $t+1$. In the fire masks, red denotes fire, while grey implies no fire.

4 Model Architecture

Since the dataset consists of spatial features, we employ a U-Net like convolutional neural net architecture. A U-Net CNN frames the prediction task as an image segmentation problem where we classify each area as either containing fire or no fire given the location of the fire on the previous day and other input features.

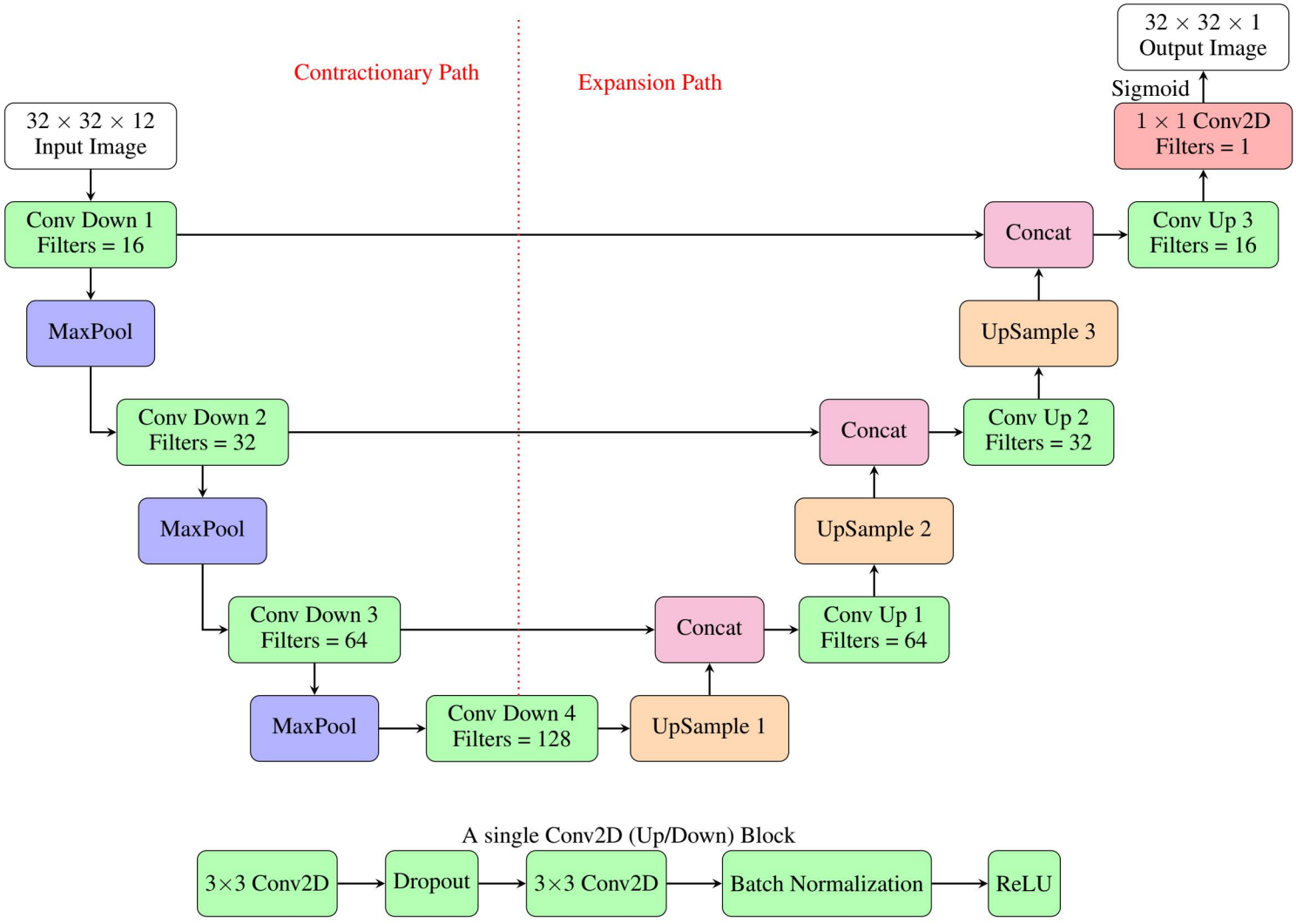


Figure 2: Deep learning model architecture: The top figure shows the U-Net model architecture for image segmentation. The bottom figure shows the layers of a single convolutional up/down block

Figure 2 shows the architecture for the image segmentation problem. The input to the network is a 32 $\times$ 32 $\times$ 12 image and the output is a 32 $\times$ 32 $\times$ 1 image of fire masks on day $t+1$. All convolutionals are 3 $\times$ 3 with a stride of 1 and same padding, max pooling is 2 $\times$ 2 and we use ReLU activation function for the hidden layers and a sigmoid activation for the output layer. A U-Net CNN

model architecture shrinks the image size in the contractionary paths and expands it back to the output size during the expansion path. We use 4 down Conv blocks with 16, 32, 64 and 128 filters where each Conv block is followed by a max pooling. Each Conv block consists of a 3×3 Conv2D block, dropout, another 3×3 Conv2D block followed by batch normalization and ReLU activation (shown at bottom of Figure 2). During the expansion path, we use 2×2 upsample and same Conv (up) block. The last layer has 1 filter with 1×1 Conv2D block and sigmoid activation function for the predicted fire mask image.

4.1 Training Details and Hyperparameters tuning

Since each area is labelled as either 'fire' or 'no fire', we use a binary cross-entropy loss function for training the model parameters. For evaluation, we use two metrics - dice coefficient and the Intersection Over Union (IoU) metrics. We perform the hyperparameter selection for learning rate, dropout rate and batch normalization size by grid search.

Table 1: Tuning of Hyperparameters

	(1) Loss	(2) Dice Coefficient	(3) IoU
	Panel A: Learning Rate (dropout rate = 0.1, batch size = 32)		
$\alpha = 0.0001$	0.0851	0.2155	0.1220
$\alpha = 0.001$	0.0844	0.2104	0.1187
$\alpha = 0.01$	0.0837	0.2018	0.1134
	Panel B: Batch Size (learning rate = 0.001, dropout rate = 0.1)		
Batch Size = 16	0.0853	0.2164	0.1225
Batch Size = 32	0.0844	0.2104	0.1187
Batch Size = 64	0.0858	0.2177	0.1234
	Panel C: Dropout Rate (learning rate = 0.001, batch size = 64)		
Dropout Rate = 0.1	0.0858	0.2177	0.1234
Dropout Rate = 0.2	0.1002	0.2276	0.1302

Table 1 reports the value of loss function and evaluation metrics on the validation set for various learning rate, batch size and dropout rate. Panel A reports the results where we fix batch size to be 32 and dropout rate equal to 0.1 and vary the learning rate. We can observe that as the learning rate decreases, both the dice coefficient and IoU improves. However, a very small learning rate increases the computational time significantly and only a marginal improvement in both the evaluation metrics. Therefore, we select learning rate of 0.001. Next in Panel B, we explore the batch sizes of 16, 32 and 64 keeping learning rate of 0.001 and dropout rate of 0.1. Here, we find that a batch size of 64 gives the best model evaluation results. Lastly, in Panel C, we explore the effect of dropout rate on model performance holding learning rate of 0.001 and batch size of 64 fixed. Here, the dropout rate of 0.1 performs the best as it gives a lower loss function and performs only marginally worse in terms of the evaluation metrics relative to other rates explored.

Hence, hyperparameters chosen are learning rate of 0.001, dropout rate of 0.1 and batch size of 64.

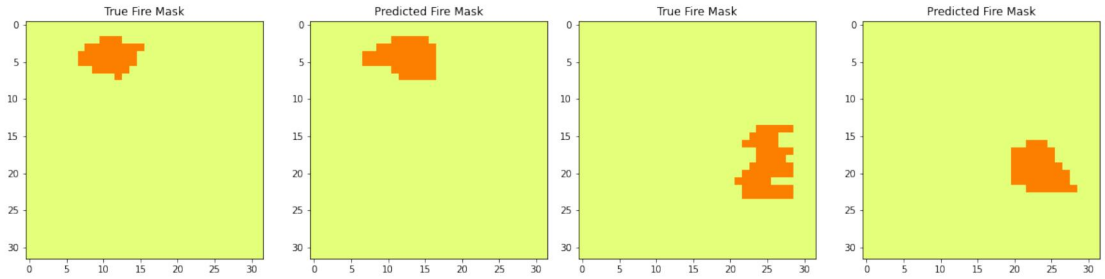
5 Results

With the tuned hyperparameters, we then run the model for 250 epochs and evaluate the model both on the evaluation and the test sets. Table 2 shows that the trained model has a dice coefficient of 21.5% and IoU of 12.2% on the test set.

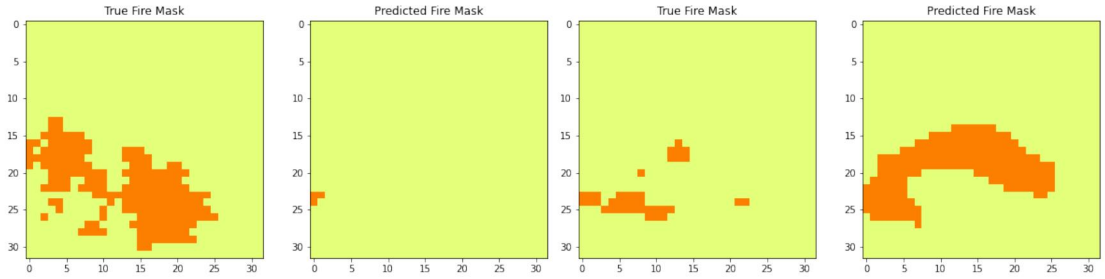
Table 2: Prediction Results: Main Model

	(1) Loss	(2) Dice Coefficient	(3) IoU
Validation Set	0.1443	0.2577	0.1652
Test Set	0.1849	0.2154	0.1216

To visualize the predicted fire masks, Figure 3 shows the predicted fire masks given the previous day's fire masks and compare with the actual fire masks. Top Panel shows the cases where the model predicts the true fire mask relatively well whereas the bottom panel shows the cases where the model fails to do so. On inspection, we found that the model performs relatively poor in cases where there is either a large increase or decrease in the fire mask on day $t+1$ relative to day t. Hence, the model is unable to predict drastic changes over consecutive days.



(a) Good Predictions



(b) Worse Predictions

Figure 3: Examples of predicted vs the true fire-masks

5.1 Feature Analysis

To analyze if the model is over-fitting due to the presence of 12 input features, we re-train the model by eliminating features that are highly correlated. For instance, the main model consists of both minimum and maximum temperature as features. However, these are highly correlated in the data. So we exclude minimum temperature. We train two additional models: one including 9 features where we drop minimum temperature, precipitation and population density, and another including 7 features additionally dropping humidity and ERC. The choice of which features to exclude is motivated by Huot et al. [2021] where they show that these features are the least important to model evalation.

Table 3: Prediction Results: Main Model

	(1) Loss	(2) Dice Coefficient	(3) IoU
All features	0.1443	0.2577	0.1652
9 features	0.1593	0.2526	0.1460
7 features	0.1582	0.2397	0.1372

Table 3 reports the results. Row labelled “All Features" is out main model of Section 5. We can observe that both the dice coefficient and IoU monotonically decrease as we exclude the features. This suggests that the model evaluation is less sensitive to the presence of correlated variables. Since all the models take similar time to train, we conclude that the model with all the features performs the best while avoiding the over-fitting problem.

6 Conclusion

We use the 'Next Day Wildfire Spread' dataset compiled using remote sensing images from Google Earth Engine to predict where the wildfire will spread the next day given features on the previous day. The dataset includes information regarding historical wildfires across US from 2012-2020 and factors influencing wildfire such as temperature, wind speed, precipitation etc. We implement the wildfire spread prediction task as an image segmentation problem and employ a U-Net like convolutional neural network architecture. We find that the model has a dice coefficient of 21.5% and an IoU of 12.2%.

In further analyses, we find that even though the model has 12 input features, it is less sensitive to the overfitting problem. Excluding features that are highly correlated with other does not necessarily improve the model performance suggesting that all the 12 input features are important in predicting next day wildfire spread.

References

Panagiotis Barmpoutis, Periklis Papaioannou, Kosmas Dimitropoulos, and Nikos Grammalidis. A review on early forest fire detection systems using optical remote sensing. *Sensors*, 20(22):6442, 2020.

Omid Ghorbanzadeh, Khalil Valizadeh Kamran, Thomas Blaschke, Jagannath Aryal, Amin Naboureh, Jamshid Einali, and Jinhu Bian. Spatial prediction of wildfire susceptibility using field survey gps data and machine learning approaches. *Fire*, 2(3):43, 2019.

Fantine Huot, R Lily Hu, Matthias Ihme, Qing Wang, John Burge, Tianjian Lu, Jason Hickey, Yi-Fan Chen, and John Anderson. Deep learning models for predicting wildfires from historical remote-sensing data. *arXiv preprint arXiv:2010.07445*, 2020.

Fantine Huot, R Lily Hu, Nita Goyal, Tharun Sankar, Matthias Ihme, and Yi-Fan Chen. Next day wildfire spread: A machine learning data set to predict wildfire spreading from remote-sensing data. *arXiv preprint arXiv:2112.02447*, 2021.

Piyush Jain, Sean CP Coogan, Sriram Ganapathi Subramanian, Mark Crowley, Steve Taylor, and Mike D Flannigan. A review of machine learning applications in wildfire science and management. *Environmental Reviews*, 28(4):478–505, 2020.

Binh Thai Pham, Abolfazl Jaafari, Mohammadtaghi Avand, Nadhir Al-Ansari, Tran Dinh Du, Hoang Phan Hai Yen, Tran Van Phong, Duy Huu Nguyen, Hiep Van Le, Davood Mafi-Gholami, et al. Performance evaluation of machine learning methods for forest fire modeling and prediction. *Symmetry*, 12(6):1022, 2020.

Ioannis Prapas, Spyros Kondylatos, Ioannis Papoutsis, Gustau Camps-Valls, Michele Ronco, Miguel-Ángel Fernández-Torres, Maria Piles Guillem, and Nuno Carvalhais. Deep learning methods for daily wildfire danger forecasting. *arXiv preprint arXiv:2111.02736*, 2021.

Younes Oulad Sayad, Hajar Mousannif, and Hassan Al Moatassime. Predictive modeling of wildfires: A new dataset and machine learning approach. *Fire Safety Journal*, 104:130–146, 2019.