Comparing Simple Deep Learning Models to a Complex Model for Smoke Detection

Jakub Szumny

Math and Computer Science Division, University of Illinois at Urbana-Champaign

jszum3@illinois.edu

ABSTRACT

Forest fires are a major problem, and have detrimental effects on the environment. Current solutions, most commonly being random general republic reports, to detecting forest fires are not efficient enough, as predictions are not always accurate and quickly responsive. This leads to forest fires getting out of control and leading to devastating problems. This study is a continuation of the work done by UCSD, and their SmokeyNet deep learning architecture for smoke detection. In their work, they developed a custom-made deep-learning architecture, for detecting smoke in camera images. I have compared many different deep learning models, to find the best model for this issue and to find a

simple, less complex model, which would take less time for inference, and give fewer false positive smoke detections. This is an important thing to investigate, as it can substantially decrease the time it takes to control fires. The models which I compared are VGG16, UCSD SmokeyNet, Resnet18, Resnet34, and Resnet50. I introduce a new dataset, full of many images from the FigLib dataset, created by the researchers at UCSD, and also images from a dataset of cloud images, from Singapore Whole Sky Imaging datasets. This new dataset includes three different features: smoke, cloud, and other. The feature, "other", is composed of images of ground, horizon, and sky. It is important to include cloud as a feature because it allows the model to try and find more of a difference between smoke and cloud, which tend to look very similar. When trained on this new dataset, the results I received show that a simple model, VGG16, outperforms all other models in instances where wildfire smoke is not easily detectable. When tested on the test dataset which was collected from the same scenes that were used for training and validation, it performs just as well as all the other models, including the SmokeyNet model.

INTRODUCTION

Forest fires around the world have been becoming more consistent and common. Canada, for example, is having one of the worst forest fires in history. In a recent report, "Interagency fire center has recorded 4,241 wildfires since the beginning of 2023" [1]. This shows just how much forest fires happen nowadays. The fires have spread over "27.1 million acres", which is now the worst forest fire in the history of Canada. Forest fires are not only terrible for the environment

but also for humans too. The smoke caused by these fires spread across all of the United States, and significantly decreased the air quality significantly. Smoke contains "carbon monoxide, among many other harmful particles which are extremely unsafe for humans" [2]. Many of these fires tend to get out of control, as they are not detected until it is too late.

Machine learning is a very good way to be able to detect these fires, and could make a serious impact in preventing these fires from getting out of control in the future, by alerting local fire departments or forest preserves. The goal of this research is to see if a simple model can perform just as well as a more complex model in smoke detection, as a simple model has much faster computational speeds. Perfecting this machine-learning solution won't be an easy feat, as smoke in the early stage of fire is very difficult to detect, and once the smoke is detectable it has many similar features to that of clouds, but over time, the models on smoke detection will become better, by fine-tuning these models using additional data, allowing the detection of fires before they get out of control, and lead to other detrimental effects such as poor air quality.

In my work, I propose a new dataset for other environmental scientists, who are interested in wildfires, to use. I also show how a simple model performs just as well as a more complex model, such as the UCSD SmokeyNet model[1] [3], and even in some scenarios, performs even better. Not only does it perform just as well as other models, but it also outperforms other models on datasets with not easily detectable wildfire smoke.

DATA

In this research, I used data from the HPWREN FigLib dataset[2], which contains images of burning events in Southern California. I also used a dataset of clouds from the Singapore

[1] https://www.mdpi.com/2072-4292/14/4/1007
[2] http://hpwren.ucsd.edu/HPWREN-FIgLib/

Whole Sky Imaging dataset [4], which I included in the larger dataset. Each of these images was full of images of the fire and of all the surroundings, which I had to manipulate to create a good amount of data, which has many different features, for training the models. To determine that it was a good dataset, I made sure that it contained many different pictures, the same amount of pictures for each class to avoid bias, and had all the images labeled correctly. I used the data from UCSD's work, as it contained XML files containing locations of bounding boxes of smoke in the larger picture, which made separating smoke from the rest much easier. I also used a completely different dataset from ARM, which allowed me to really test the models on completely different data. The VGG16 model that I was comparing, only accepts images of size 224x224, so I made the data based on that.

METHODS

Smoke separation

Since I was given the images from the FigLib Dataset, along with correlating XML files, which smoke bounding box locations, I had to create a program in order to create a Pandas Dataframe, containing all the locations of the smoke, along with the exact image that those locations correspond to. I created a loop inside of the program, which would grab only the largest and smallest x and y coordinates, as some of the bounding boxes were polygons. This left me with this new data frame, which I could then use for later work. I then created a new program that would loop through each instance in the data frame, and crop each image at the given location of smoke, which left me with 8,437 images of 224 x 224 smoke.

Ground and sky separation

I was able to then create another separate program, which would do almost the opposite as the smoke cropping program. It first uses a condition to check if the smoke is in the first half of the image or the second, and then it would crop the whole area of smoke out of the full image, leaving just the ground and sky, without any smoke. With the smoke, I didn't have to leave out a single image, because in each image the smoke is slightly different, but because the surroundings don't change at all, I limited it to only do this twice per camera location. I then created another program, which split those images into many smaller tiles. I split each larger image into 5 rows by 6 columns, which left me with 30 images per larger image. I separated each of these images into different features, named "Sky", "Ground", and "Horizon". The cloud images were roughly already good for training, I simply just cropped many of them in order to get more images.

Train test split

After having all of the images created it was then time to split up the data in order for training the models. I decided on a 70/20/10 (Train/Validation/Test) split, by randomly separating events, and with some events exclusively in the testing data. This way it wouldn't be able to learn the exact images, and actually give accurate results on the testing data. I created four different instances of this split, to experiment on in order to see which one would perform the best. The different splits were: one with 3 features ("smoke", "cloud", other"), one with two features ("smoke", "other"), one with inverted colors and three features("smoke", "cloud", "other"), and one with the difference of consecutive images and three features("smoke", "cloud", "other").

Training the models

I trained each model on seven epochs, on batches of size six hundred and forty-four. The optimizer I used for training the models was the Pytorch SGD optimizer, with a learning rate of 0.001, and a momentum of 0.9. The learning rate scheduler used a step size of seven and a gamma of 0.1. The loss function I used was the Cross Entropy Loss function from the Pytorch library, in its standard form.

EXPERIMENTS

Two features (smoke, other)

I first made an initial experiment to train each model on only two features, “smoke” and “other”. After looking at the results, I decided not to move forward with this type of split, as the accuracy was not very good. On the initial testing data, each model performed about the same at an accuracy of about 87%, but on the ARM data, the models performed significantly worse, having only a 33% accuracy, while having roughly a 0% true positive rate, with the highest being the Resnet50, with a rate of 5%, as shown in Figure 1.

Three features (smoke, cloud, other)

I did another experiment, where instead of only two features, I separated “other” and “cloud”, which gave me three features, “smoke”, “cloud”, and “other”. I believed that this would work slightly better than with only two features, as the model would have to try and predict “cloud” completely separately, and hopefully, it would catch some pattern to do just that. The models trained on these features performed much better than before on the original testing data, all receiving about 92% accuracy. When tested on the ARM data, only the VGG16 performed

well, having a 96% true positive rate, and a 13% false positive rate, with an accuracy of 90% while all the other models performed very poorly, all receiving a true positive rate of 0%, except for the Resnet34, which got a true positive rate of 4%, as shown in Figure 1.

Three features - image difference (smoke, cloud, other)

This experiment was to see whether or not the difference in images would make it easier for the model to detect smoke. If two consecutive pictures of smoke are taken, they will be significantly different, as smoke moves very sporadically. Clouds, on the other hand, don't move sporadically, but instead move in a rather consistent pattern, usually in one set direction. Two consecutive images of the ground or the sky in theory should stay the exact same, having no differences, as they don't move. In order to do this experiment I took many consecutive images, and subtracted one from the other, leaving me with a gray-scale image of just the differences of the image, and I did this for all the features. When I trained and tested these models, I found that they actually did not perform that well and that the model had a very difficult time differentiating between any of the features. The models again only had a true positive rate of roughly 0%, with the Resnet18 having the highest at only 2%, as shown in Figure 1. The accuracy of all of the models was roughly 49%, which is very poor. I believe that this is because I didn't consider the time of day/the amount of light/shadow on the ground and in the sky, which would cause consecutive images to in fact have a large difference.

Three features - inverted colors (smoke, cloud, other)

I also wanted to see if inverting the colors would help the model become more accurate as well. My mentor recommended I test changing the colors of the different features as he

believed that if the colors had more distinct differences from one another, the model would have an easier time differentiating between them. I decided on inverting the colors as I felt this would be a great way of changing the colors, while still having each feature be consistent with one another. When I trained and tested the models, they also did not have good accuracy. Once again, each model only received a true positive rate of roughly 0%, with the VGG16 performing the best with a 5% true positive rate, as shown in Figure 1. The average accuracy of each model was again about 48%, which is very poor. I believe that this is because I only inverted the colors, and didn't change them individually, so technically the images are still the same. If the colors were manually changed, for example, white to red, and gray to black, there might be a bigger difference in the features which the model catches on to.

RESULTS

Experimental results

From my experiments, the only one which managed to result in high accuracy was by using three features, "smoke", "cloud", and "other", with regular images. On the original testing data, each model performed rather well, all having about 90% accuracy. On the ARM testing data, only the VGG16 performed well, while all the other models performed quite poorly, with very low true positive rates, as shown in Figure 1, and overall very poorly. From these results, I can decide to focus on the experiment which resulted in the highest accuracy, which was with using three features, "smoke", "cloud", and "other", and continue on with the research focusing on only models trained on these specific features.

Performance visualizations

	Model	FPR	TPR	FNR	TNR	Type	NumFeatures
0	VGG16	0.120	0.960	0.04	0.88	Regular	3
1	UCSD	0.000	0.000	1.00	1.00	Regular	3
2	Resnet18	0.000	0.000	1.00	0.99	Regular	3
3	Resnet34	0.040	0.080	0.92	0.96	Regular	3
4	Resnet50	0.000	0.000	1.00	1.00	Regular	3
5	VGG16	0.010	0.000	1.00	0.99	Regular	2
6	UCSD	0.000	0.000	1.00	0.99	Regular	2
7	Resnet18	0.008	0.080	0.92	0.99	Regular	2
8	Resnet34	0.020	0.000	1.00	0.98	Regular	2
9	Resnet50	0.030	0.000	1.00	0.97	Regular	2
10	VGG16	0.010	0.008	0.99	0.99	Image Difference	3
11	UCSD	0.001	0.001	1.00	1.00	Image Difference	3
12	Resnet18	0.020	0.020	0.98	0.99	Image Difference	3
13	Resnet34	0.000	0.070	0.93	0.97	Image Difference	3
14	Resnet50	0.000	0.000	1.00	1.00	Image Difference	3
15	VGG16	0.050	0.000	1.00	0.95	Invert	3
16	UCSD	0.000	0.000	1.00	1.00	Invert	3
17	Resnet18	0.000	0.000	1.00	0.99	Invert	3
18	Resnet34	0.000	0.000	1.00	0.99	Invert	3
19	Resnet50	0.003	0.000	1.00	0.99	Invert	3

FIG. 1. All results of different models and their experiments on ARM data

DISCUSSION

Impact

The dataset of 41,000 images that I have created, includes classes of: “smoke”, “ground”, “sky”, “horizon”, and “cloud”. This dataset is ML-ready and can be used by anyone to improve on this work. Also, I have found that a simple model does in fact compare to a more complex model and even outperforms it in some instances. What this can help with, is that future work can be focused on developing simple models, which will have faster computation speeds. I have created a plugin application that utilizes the models and can be deployed in an edge computing platform, Waggle/Sage, in order to be able to compute and classify live camera feeds from nodes, to detect smoke in real time.

Comparison to past works

UCSD's SmokeyNet architecture is a good model, as they have an accuracy of about 90%, as shown in Figure 2, on the original testing dataset which includes similar instances of burning events to the ones used for training, but the models which I have experimented with in this research, have all performed just as well. These are very simple pre-made models, which don't require much computational time, especially the VGG16 model. The accuracies were very similar, at around 90% on the original testing data, as shown in Figure 2. When the ARM data was introduced, which contained a very small burning event, and captured images from a very different surrounding environment, with much heavier cloud formations, only the VGG16 managed to predict the smoke with a high accuracy of about 90%, while every other model had true positive rates of 0%, or close to that, as shown in Figure 3.

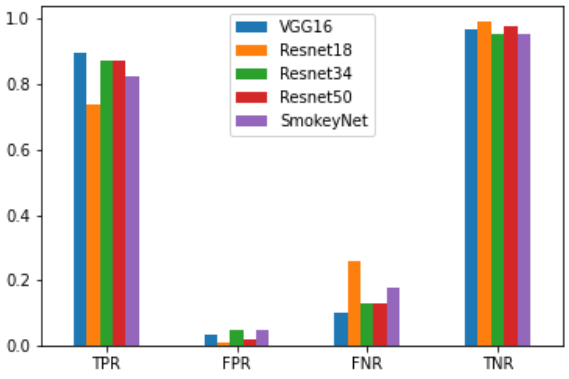

FIG. 2. Results of all models (smoke, cloud, other), on original testing data

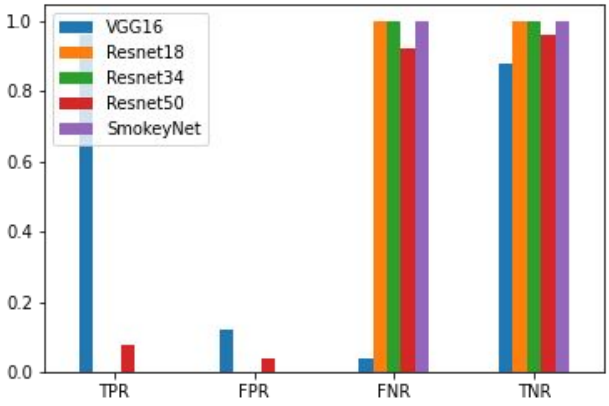

FIG. 3. Results of all models (smoke, cloud, other), on ARM testing data

Future directions

Now that I have found that a simple model can perform just as well as a more complex model, in the case of smoke detection using this dataset, more work can be done on creating a better model for this particular problem. My work is replicable through my GitHub repository, so anyone can try to improve on it. I believe that image augmentation is the key to this problem, and more work should be done to try and find an optimal augmentation, which can help the model better differentiate between smoke and cloud.

CONCLUSION

This research provides a refined, completely labeled, ML-Ready dataset of many different burning events. It also provides a great baseline as to what type of model is best for this problem, as simple models work just as well as complex models. My algorithm also is great for future real-time smoke detection through edge computing platforms/devices, which can be deployed in remote locations for monitoring local areas, as it can already give real-time

classifications from one of these edge computing devices located in Montana, using the VGG16 model. I hope that this work can greatly help the future of smoke detection.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships (SULI) program. I would like to thank my mentors, Seongha Park and Bhupendra Raut, for their consistent support during this project, and for their great advice. I would also like to thank the entire team at UCSD, for providing such a great dataset and research which allowed me to continue their work in smoke detection.

REFERENCES

1. Czachor, Emily Mae, “Canadian Wildfire Maps Show Where Fires Continue to Burn across Quebec, Ontario and Other Provinces,” CBS News, 19 July 2023, [Online]. Available: www.cbsnews.com/news/map-canadian-wildfires-2023-where-are-the-fires-ontario-quebec/. [Accessed: 03 August 2023].

2. Williams, Corey, “Air Quality Plummets in Parts of the U.S. Again Due to Wildfire Smoke from Canada,” PBS, 28 June 2023, [online]. Available: www.pbs.org/newshour/nation/air-quality-plummets-in-parts-of-the-u-s-again-due-to-wildfire-smoke-from-canada. [Accessed: 03 August 2023].\

3. Dewangan, A., Pande, Y., Braun, H.-W., Vernon, F., Perez, I., Altintas, I., Cottrell, G. W., & Nguyen, M. H. (2022). FIgLib & SmokeyNet: Dataset and deep learning model for real-time wildland fire smoke detection. *Remote Sensing*, *14*(4), 1007. https://doi.org/10.3390/rs14041007

4. Dev, Soumyabrata, Yee Hui Lee, and Stefan Winkler, “Color-based segmentation of sky/cloud images from ground-based cameras,” IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 10, no. 1,pp. 231-242, January 2017.