Forest Fire Image Generation using DCGAN and Classification

Krithika Raj Dorai Raj
SUNY University at Buffalo
kdoraira@buffalo.edu

Abstract

Every year, wildfires or forest fire, cause immense damage to ecosystems all across the world. The rates of occurrence of this natural disaster is fueled by climate change and is expected to be on the rise in the coming years. Early detection of these wildfires in remote areas is crucial to contain the fire and minimize the damage it causes. Despite the vast research in this area, there are very limited datasets available to train deep networks on, and majority of them are not publicly available. This project aims to generate new images of forest/wildfires using the limited images available on the web by using a Generative Adversarial Network. These images are used to train a classifier to examine its performance with the generated dataset. In addition to this, the classifier is also trained on a images of synthetically produced forest fire from simulated smoke. A comparison of the two methods shows that the classifier trained with the GAN generated images has better generalizability and performs better on real world forest fires images.

1. Introduction

Wildfires affect all biomes, from forests and savannahs to grasslands and tundra [1]. In recent years, the crisis of wildfires has been on the rise, increasing by 13% in April 2020, a record year for fires. An increasing share of wildfires are due to human activity. If not caused artificially, the alarming climate change has triggered some of the most dangerous fires in the past years and the forest fires will only worsen climate conditions, both reinforcing each other. The Amazon fires in 2019 were unprecedented in their destruction burning more than 7600 square kilometers in their destruction. The California fires in 2019 caused wide scale, irrevocable damage to the environment and also causing numerous home evacuations and loss of property. Wildfires can have immediate and long-term effects on the quality of rivers, lakes, and streams and most importantly air quality. The fires in Australia proved deadly to many endangered species of plants and animals.

The severe long- and short-term effects of these fires begs immediate attention. Early detection of these fire spots could play an instrumental role in our ability to contain the fire. There is an incredible amount of work being done in the early detection of fires by NASA. Their satellite instruments are often the first to detect wildfires burning in remote regions.

Due to dynamic properties of smoke and fire that have no definite shape or form, traditional image processing methods aim to detect them using rule-based image processing and temporal variation [3]. Many of the traditional models for smoke and fire detection fall short of adequately capturing the varied colors of shapes of real time fires. The biggest challenge in this area of research is accurate detection of early smoke and fire, without raising a lot of false positives in the presence of fog and clouds. Deep convolution networks are a promising approach to tackle this issue, if they are exposed to a wide range of such smoke and wildfire during training. In order for a deep model to be trained effectively, it requires a large set of images to learn from.

A number of methods have been proposed to tackle the challenge of early detection of wildfire and smoke with deep learning methods- from early smoke detection using deep convolution networks, to computer vision techniques to model smoke and fire color patterns, to reduce false positive rates of detection, etc. Being a very specific application problem, despite the ongoing research, there is a dearth of a good-sized dataset for training deep learning models. Each publication uses a small set of images, and reports accuracy that is incomparable to other methods, because of the obscure nature of the dataset used in each, and the absence of publicly available datasets.

This project aims to tackle this challenge. To alleviate the dearth of images for forest fires, the goal of this project is to generate new images of wildfires using the limited data available using generative adversarial networks. Due to the nature of the problem, that is the dynamic nature of the fire and the smoke, generated images by nature will produce distorted smoke features and fires interlaid in landscape images that previously did not contain fires, creating a varied dataset.

1.1. Related Work

There is plenty research in the area of forest fire detection using traditional smoke detection methods using sensor-based tools. This is an effective solution when monitoring areas that are known to prone to such wildfires. Constant monitoring of these systems might prove to be effective if fire in already anticipated in the area. Further, it fails to communicate the intensity of the fire, the direction and the rate of spreading. For monitoring on a larger scale, in remote regions, satellites prove to be effective, as shown by NASA's initiatives. On a smaller scale, Unmanned Aerial Vehicles (UAVs) have gained a lot of traction in this area of research. The use of UAVs has led to various image and video-based techniques being implemented to study forest fires. The challenge with this approach is the dynamic nature of the smoke and fire images, and the quality of the images that are capture such smoky, low visibility environments. One concern raised by all the research papers for wildfire detection referenced below and otherwise, is the lack of a standardized dataset to train and evaluate performances.

Abdulaziz Namozov et all [9] have proposed a novel deep convolutional neural network algorithm, that replaces traditional rectified linear units or tangent function with piecewise linear units in the hidden layers of the network. Qingjie Zhang et all [8] proposed a 2 stage deep learning method for forest fire detection where both a full image and fine grained patch fire classifier in a joined deep CNN(AlexNet) are trained. The fire detection is done in a cascaded fashion- the full image is first tested by the global image-level classifier, if fire is detected, the fine-grained patch classifier is followed to detect the precise location of fire patches. Zhentian Jiao et all in their paper [2] have proposed to use a UAV platform equipped with a small scale convolutional neural network implemented with YOLOv3, to fit on the onboard hardware of the UAV, reporting about 83% recognition rate. Unlike traditional region-based approaches, YOLO is a one-stage algorithm, passing the image only once in a fully convolutional neural, network (FCNN), making it very fast for real time applications.

Qi-zing Zhang et all [4] have proposed a novel method to tackle the lack of training data for wildland fire and smoke detection. They created synthetic smoke images by inserting real smoke or simulative smoke into a forest background. They also used a faster CNN model to detect the fire/smoke to evaluate the effectiveness of using real smoke inserted images to simulative smoke inserted images. They compared the performance of these two classifiers to conclude that simulative smoke images perform better. On examination of the dataset provided by the authors, the dataset that contains about 12000 images for each real smoke and simulated smoke, is mostly constant in its landscapes. Meaning, there are approximately 10 difference scenes/landscapes and each is duplicated many times with interlaced real or simulated smoke in various positions and shapes across the image.

In the study presented by Minsoo Park et all [7], wildfire images were generated using a cycle-consistent generative adversarial network (CycleGAN) to eliminate data imbalances. A densely-connected convolutional networks based (DenseNet-based) framework was proposed and its performance was compared with pre-trained models. They reported best performance result with an accuracy of 98.27% on test images concluding that this framework demonstrated high wild fire detection accuracy. Cycle GAN is used in applications where image translation is the task. It is used when there are any two collections of unrelated images that can be and the general characteristics extracted from each collection is used in the image translation process. This works well in this application scenario when landscape without fires are translated to landscapes with fire. However, this is a computationally expensive process while compared to GAN, because it essentially trains 2 generator and discriminator model.

2. Method

In this project, a similar framework is adopted to generate images using a Deep Convolution Generative Adversarial Network (DCGAN) to train a deep classifier to detect the presence of wildfires. A DCGAN is trained with available images of wildfires and images of landscapes. The model generates images with elements of fire and smoke interlaced with the landscapes. These generated images, along with real forest fire images, are used to train a classifier network (VGG16) to classify a presented image as one containing wildfire or not. Another VGG16 model is trained with synthetic smoke dataset as described in the Qi-zing Zhang et all paper [4] and the performance of this model on real forest fire images is compared to that of our model trained with generated images.

2.1. Data collection and preparation:

Data collection and preparation was the most labor intensive, time consuming task of this project, proving the 80/20 rule in data science (spending 20% time collecting data, 60%-time cleaning and organizing data, 20% using the data for task at hand). This project provided a first-hand, invaluable experience in learning to deal with real data and making it suitable to train a deep learning model. Being accustomed to readily available datasets for all projects we have had to complete for coursework, there are few opportunities to truly realize and appreciate this luxury of a clean precompiled, pre-labeled dataset. When it comes to application of deep learning models to real world problems, data plays the most important role in determining the quality of the performance of the model. The images used in all datasets were obtained from two main sources:

2.1.1 Real Fire, Simulated Fire Data (RF-SF data):

From the reference papers, the only suitable dataset that was publicly available was that of the Synthetic smoke from the Qi-zing Zhang et all paper [4]. The dataset consisted of two types of images, 12000 each- real fire/smoke inserted into landscapes and simulated smoke (smoke created in front of a green screen) inserted into the same landscapes. Amongst the 12k images, at least 25% of the images were not in a forest/wildland setting, it included buildings in the landscape and therefore were not ideal for usage in this scenario. There landscapes were largely identical – at most 10 variations of landscapes were present and were duplicated with the smoke location and size being varied with the same landscape.

2.1.2 Web Crawled Images:

Most of the other reference papers did not provide much information about the dataset used, nor were any publicly available (if resources were disclosed). There to build a dataset for the class of fire and for no fire was the biggest challenge. In order to put together a decent sized dataset, bing-image-downloader, a python library to download images from bing, was used to web crawl images of forest fires and of natural landscapes. Image results for the following search tags were gathered (1000 images for each tag):

1. “Rolling hills”, “rocky hills”, “forest”, “mountains green rocky” – for the no fire class
2. “Wildland smoke aerial”, “forest fire smoke aerial” – for the fire class

Again, most of the images for the forest/wildfire class was repeated, or unusable due to the presence of other large objects in the image that would corrupt the dataset.

2.2. Data sets for the various tasks:

2.2.1 Data set for image generation using DCGAN:

Images from the previously described web crawled tags were shortlisted for both classes – presence of fire and no fire landscapes. A total of 1360 images were filtered for usage(drastically reduced number due to repetition of crawled images), which included a moderate ratio of images from the RF-SF dataset. About 25-30% of the images included no fire landscape images as well.

2.2.2 Data set for Classifier trained with Generated Images:

For the fire class, the low-resolution images generated using the dcgan were included as 50% of the dataset. The other 50% consisted of web crawl images and a small percentage of RF_SF images were used as well, for diversity and comparison. A total of 880 images were present for this class. For the No- fire class, 880 shortlisted images from the web crawl results were collected (the dataset for this class is common and used as RF-SF data also).

2.2.3 Dataset for classifier trained with RF-SF images:

The 24k images available in the RF-SF dataset, were downsized to about 500 images to include only those images that provided some diversity, and excluded a vast portion of the dataset to prevent the model from over-fitting very drastically on this repetitive data. A total of 1200 images for the wildland fire class were selected, along with some of the web crawled images of forest fire to add to diversity, especially since the no fire class of images predominantly included similar web crawled images of landscapes without fire(size of data for this class = 880, the same set is used for training both classifiers).
A sample of these datasets is presented in figures 1,2 and 3.

2.3. Data processing and Labeling:

For the classifier tasks, images of each class were uploaded in separate folders on google drive and loaded into colab as separate datasets for each class. Once imported these images were converted to a numpy file and stored in the same directory as the images for quicker access in consequent runs. Before saving to a numpy file, the only preprocessing performed on images were resizing – to 96x96 pixels. For training the model, data augmentation was performed with ImageDataGenerator, where the images were normalized by using the rescale factor as 1./255, along with other augmentation parameters like rotation angle, width and height shift, zoom range and horizontal flip. This was a crucial step since the data set for each class was merely 880 images, and hence insufficient to properly train a deep classifier like VGG16.

For the Data Generation task, the dataset was loaded the same way, from one single folder. Apart from the resizing of the images to 96x96, an additional pre-processing step for images here was applying tanh normalization to the images, as suggested by Alec Radford & Luke Metz et all [5], in their paper for DCGAN, for improved performance (speedy learning) of the GAN in scenarios of limited processing power availability.

2.4. Models

2.4.1 DCGAN for Image Generation

As mentioned before, a DCGAN is chosen over a Cycle GAN in this implementation to explore its effectiveness in generating images with fire and smoke. The cycle gan approach described by Minsoo Park et all [7] is a good approach to the forest fire image generation problem, but dcgan provides a simplier, less computationally intensive model for image generation. The implementation of DCGAN here follows the paper by Alec Radford & Luke Metz et all [5] that sets some architectural guidelines for stable deep convolution GANs. The guidelines are as follows:

1. Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
2. Use batch norm in both the generator and the discriminator.
3. Remove fully connected hidden layers for deeper architectures.
4. Use ReLU activation in generator for all layers except for the output, which uses Tanh.
5. Use LeakyReLU activation in the discriminator for all layers.

The implementation of GAN is as follows:

- The only preprocessing applied to training images is scaling the pixel values of the image to the range of the tanh activation function [-1, 1].
- Adam optimizer with learning rate 1.5e-4 was used.
- Weights are initialized from a zero-centered Normal distribution with standard deviation 0.02.
- In the LeakyReLU, the slope of the leak was set to 0.2, for generator but normal relu was used for discriminator.
- Momentum term was suggested at 0.9, but the implementation used 0.5 as the value for training.
- Following the TensorFlow 2.0 example code for implementation of the dcgan and the TensorFlow Keras implementation example of GAN, the training of the dcgan is implemented with the help of tf.function and GradientTape. Tf.function annotation causes the function to be precompiled, improving performance, while the use of GradientTape enables us to train the model according to the principles of GAN (ie training the discriminator and generator separately), allowing the discriminator and generator to be trained together, but separately.

2.4.1.1 Architecture of Generator

A sequential model is defined with 4 blocks of an up-sampling layer, a conv2d layer, a batch normalization layer followed by relu activation layer. The first layer is Dense layer of size 4096 which takes inputs of the size 100 (length of random sequence fed to the generator to produce an image), this is reshaped to 4*4*256 to provide as input to the first block of layers previously described. At the end of the fourth block, there is a final convolution layer that applies tanh activation and produces the output of size 96x96x3 which is a 96x96 sized, colored image, that is generated. The filters for each block is summarized in the table [table]. In this architecture, batch normalization layers have a momentum of 0.8, all convolution layers are ok kernel size 3 with constant, default stride.

2.4.1.2 Architecture of Discriminator

This sequential model takes the input image of size 96x96x3 and is first passed through a conv2d layer, followed by a leaky relu with decay rate 0.2. Next, four blocks of conv3d layers with dropout, batch normalization and leaky relu activation are present. Finally the output of these four blocks is fed to a dense layer with one neuron to predict how real the image is. The filter sizes for each block can be found in the table 1. In this architecture, the dropout layers have a rate of 0.25, batch normalization has a momentum of 0.8, all convolution layers are ok kernel size 3 with varying strides.

Block no.	Generator	Discriminator
1	256	64
2	256	128
3	128	256
4	128	512

Table 1: Filter sizes in DCGAN

2.4.2 Classifier Models

Three types of models were used to compare performances. First, a basic CNN model was implemented, followed by a pre-trained VGG16 model, and then a fine-tuned version of the same with the last convolutional block re-trained.

2.4.2.1 Basic CNN

A simple convolution model was built with three blocks of convolution layers, along with Batch normalization and drop out layers, to explore the possibility of using a simpler model.

2.4.2.2 Pretrained VGG16 Model

The pretrained vgg16 is used as feature extractor, and a fully connected network with 2 dense layers is trained with these features to classify the images as those with wildfire, and those without. In order to use a pretrained VGG16 model (trained on imagenet), it was first imported from keras applications with the input shape 96x96x3 to fit the images in this project. The vgg16 model was first proposed in the paper [6] and provided an improvement to the alexnet by replacing large kernels of size 11 and 5, with multiple 3x3 kernal filters to improve computation efficiency. Since the vgg16 model pretrained on imagenet is state of the art, the decision to utilize it for this project was due to the small size of the curated dataset. Training a vgg16 model from scratch would require immense computational power and downsizing this architecture would not be a viable solution since images are noisy, and fewer in number. For the purpose of confirming this, a simple convolution model was built with three blocks of convolution layers, along with

Batch normalization and drop out layers, to explore the possibility of using a simpler model.

2.4.2.3 Fine-tuned VGG16 model

To classify the two sets of images, a pretrained VGG16 model is fine tuned by freezing all but the last block of conv2d layers to train them on the specific dataset. The final dense layers are of size 256 and 1, with a dropout layer between them. To properly fine tune the last layers of the model, first the whole vgg16 pretrained model without the top model (dense layers) are used to extract the features of the specific dataset. A separate fully connected model is trained using these extracted features. This model is as such used to train on the data set and reported a good accuracy. Now, the vgg16 model is instantiated again without the top model and the small fully connected model we previously trained with the specific dataset is added on top of the vgg16 model. The layers of this model upto the last convolution block are frozen and the last layers are again trained on the dataset, to fine tune the last convolutional layers. This is done so that the while fine tuning the model, all the layers already have properly trained weights, to start off with. Without this, large gradient updates triggered by random initial weights would mess with the already learned weights in the base convolutional layers.

3. Experiments and Results

3.1. Generating images with DCGAN:

Using the previously described architecture, and implementation guidelines, the dataset described in 2.1.1 was used to train the model. Once trained, this model was used to produce 1000 generated images by feeding it with randomly generated arrays of size 100, to produce images of size 96x96. Some of the hyperparameters- learning rate and momentum were tuned in accordance with the suggestions by the paper [5].

3.2. Classification tasks:

3.2.1 A basic 3 layered convolutional network

The simple convolution model was built with three blocks of convolution layers, along with Batch normalization and drop out layers proved to be inadequate to learn the features of this dataset due its shallow nature. Learning dynamic representations of smoke and fire from noisy images was too big a task to ask of a shallow CNN. After experimenting with regularization methods- using batch normalization and drop out layers and tuning of hyperparameters, the basic model performance peaked at 60% accuracy for the Generated Images dataset and only a 51.4% for the RF-SF dataset. A summary of the evaluation metrics is presented in table 2. The better performance on the generated image data set is largely due to the noisy dataset with low resolution and not clearly defined boundaries and shapes acting as good regularization, enabling the network to perform better with real world images. The RF-SF dataset trained model is unable to generalize well.

3.2.2 Pretrained VGG16 model

This model showed the best performance out of all three classifiers trained. This model was first trained with each dataset and its performance on a common test set of 70 images were reported in terms of accuracy, precision and recall. The fully connected network was trained with RMSprop with learning rate 0.001, with a drop out layer added before the final dense layer with 1 neuron, with decay rate 0.5. The evaluation metrics for this model trained on both datasets are presented in the table 4.

3.2.3 Fine-tuned VGG16 model

This method did not yield better results that using the pretrained network but performed better than the basic CNN model for both datasets. The training of this model was tested with RMSprop, SGD and Adam optimizers, with various learning rates, and the best performance was obtained with SGD at a learning rate 0.0001. The precision peaked at 76.5%, with accuracy and recall following at 55.7% for the Generated image dataset. For the RF-SF dataset, the performance of this model was better than its counterpart for the other dataset- 61.42% accuracy and recall followed by 78.22% precision. Since the final block of the pretrained model was fine-tuned on the particular dataset, and with the strength of the previous convolution blocks' pretrained values, this model was able to perform considerably well. The evaluation metrics for this model trained on both datasets are presented in the table 5.

Overall, is observed that the pretrained classifier trained with generated images perform better on real forest fire images than the one trained with RF-SF data. This result is primarily attributed to the diversity of data that is used in training the first model. The GAN generated images, are low resolution noisy images, owing to limited computing power to produce high quality, well defined images. Rather than this being a disadvantage, these images also introduce a noise factor to the classification model, acting as good regularization. The RF-SF dataset as previously described are very repetitive and hence are not able to capture the diversity of real-world scenarios.

4. Conclusion

This project shows that the major issue in tackling detection of wildfires- the lack of a large dataset, benefits from using image generation methods to reproduce a large variety of images that are impossible to obtain from real world examples. This experiment also shows that adequate performance of the classifier can be obtained even when high resolution images are not available. With greater computation power, better quality images can be generated to further strengthen the performance of these classifiers. However, in comparison to the task of synthesizing simulated smoke and then overlaying these on landscape images, the method proposed in this project achieves comparably better results in terms of generalization.

Basic CNN	GAN Generated	RF-SF
Precision	68.87	75.36
Recall	60	51.4
Accuracy	60	51.4

Table 2: Evaluation Metrics of Basic CNN

Fine-tuned VGG	GAN Generated	RF-SF
Precision	76.51	78.22
Recall	55.7	61.4
Accuracy	55.7	61.42

Table 3: Evaluation Metrics of Fine-tuned VGG16

VGG16	GAN Generated	RF-SF
Precision	98.025	90
Recall	98.011	90
Accuracy	98.011	90

Table 4: Evaluation Metrics of pre-trained VGG16

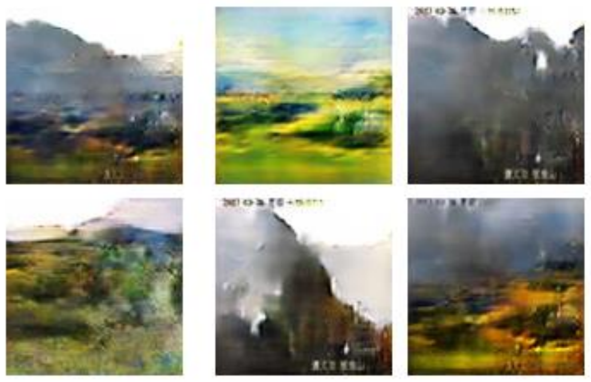

Fig 1: Sample of Generated Images

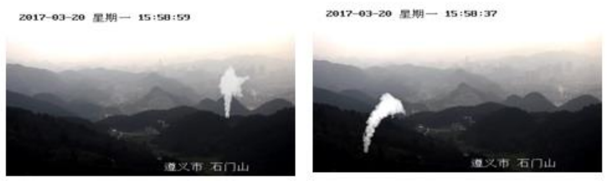

Fig 2: Sample of the RF-SF Images

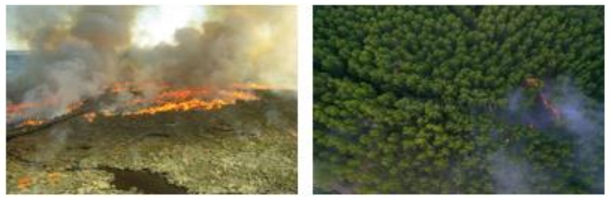

Fig 3: Sample of real wildfire images in the Test dataset

References

[1] https://wwf.panda.org/discover/our_focus/forests_practice/forest_publications_news_and_reports/fires_forests/

[2] Zhentian Jiao et all, A Deep Learning Based Forest Fire Detection Approach Using UAV and YOLOv3

[3] Mubarak A. I. Mahmoud et all, Forest Fire Detection Using a Rule-Based Image Processing Algorithm and Temporal Variation

[4] Qi-zing Zhang et all, Wildland Forest Fire Smoke Detection Based on Faster R-CNN using Synthetic Smoke Images

[5] Alec Radford & Luke Metz, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Netoworks

[6] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition

[7] Minsoo Park et all, Wildfire-Detection Method Using DenseNet and CycleGAN Data Augmentation-Based Remote Camera Imagery

[8] Qingjie Zhang , Deep Convolutional Neural Networks for Forest Fire Detection

[9] Abdulaziz Namozov et all, An Efficient Deep Learning Algorithm for Fire and Smoke Detection with Limited Data

[10] GAN tutorial for implementing Face Generation- https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class_07_2_Keras_gan.ipynb

[11] GAN Tutorial- How to build a basic gan for mnist dataset - https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-an-mnist-handwritten-digits-from-scratch-in-keras/

[12] https://towardsdatascience.com/training-gans-using-google-colaboratory-f91d4e6f61fe

[13] https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html