Realtime Wildfire Prediction Using Spark

Angelo Botticelli
145 Baugher Avenue
Elizabethtown 570-862-3182
botticellia@etown.edu

Yusuke Satani
1749 Baugher Avenue
Elizabethtown 223-216-1352
sataniy@etown.edu

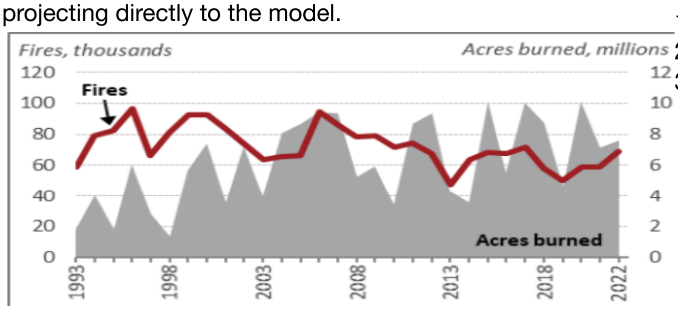
ABSTRACT

The US government has been tracking tens of thousands of burnt acres due to wildfires since the mid 80s. Although there has been a decline in the past few years from the near 100 thousand acres lost yearly, recent statistics indicate a reversal in this trend, with acres lost from fires in the past few years being higher than in 2019, and each subsequent year having more individual wildfires than the last. Furthermore, climate change is disrupting humidity and precipitation, causing an increase in temperatures worldwide, and leading to an inevitable rise in the number and intensity of wildfires across the country.

Nowadays, thanks to advanced image recognition technology, people can be better equipped to combat natural disasters by detecting wildfires from satellite images before they become catastrophic. This project aims to predict the occurrence of a wildfire from satellite images, ideally in real-time, using sophisticated machine learning models such as Convolutional Neural Networks and distributed big data processing frameworks such as Apache Spark.

1.Introduction

As the amount of acres burnt annually continues to peak around 100 million, the need to address wildfires with developing image recognition grows only further.
Image recognition technologies have come a long way in a short time which has led to a breadth of problems made solvable. The appeal of using AI to detect wildfires has not gone unnoticed as multiple sources including the world economic forum claim to be developing neural networks to seek and detect wildfires. The purpose of this project is not only to create a model to detect wildfires, but to design the means for it to predict images in real time as if a satellite were projecting directly to the model.



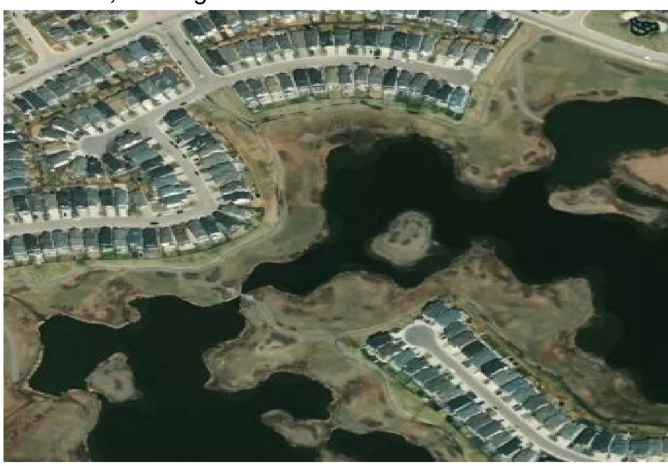
See Above, a chart of wildfires and acres burnt by year

To accomplish this task, we will be taking a two step approach; an offline component of tuning and creating a model using tensorflow and an online component using Apache spark and Spark Streaming to transfer this online and ideally into a real time application where images can be fed and predicted on in a short time.
For our data model, we will be using a dataset on Kaggle pulled from the Canadian government's official forest fire portal.
The dataset is split into two categorical folders labeled Wildfire and No Wildfire based on whether a wildfire occurred at that image.
Each image is a few dozen kilobyte large jpg files and contained in the dataset as a whole is over 40 thousand of them split nearly 50/50 by Wildfire/No Wildfire.
We plan to train and test a model, multiple and choosing the best performing, on the dataset. so it may run in real time using a small sample of images pulled from the testing slice of the dataset.
There are a few other models already done on this dataset that have achieved a high level of accuracy. Following these researches done by before, we apply two deep learning models, CNN and AlexNet model, pursuing high accuracy.
In addition to them, we want to solve the problem that they take a long time to make predictions by applying Apache Spark. Using apache spark is also helpful to improve scalability and tolerance. After applying Apache Spark in the online-training section, we connect the model to Spark Streaming so that we can predict low-latency. So we seek to not only achieve high accuracy but also to feed our model a live pipeline so our model may offer a prediction on wildfires not just from pictures stored on a database, but from images parsed in real-time.

Our contribution for wildfire prediction
1.high-accuracy wildfire prediction using Deep Learning(CNN)
2.Real-time wildfire prediction using spark streaming
3.Improving scrability and fail-tolerance using apache spark.

See below, an image from the dataset labeled 'No Wildfire'

See below, an image from the dataset labeled 'Wildfire'



2.Background

Convolutional Neural Network(CNN) is one of the most impressive forms of Artificial Neural Networks (ANN) and is mainly used to solve image-driven pattern recognition tasks. The main structure of CNN and traditional ANN is almost the same. The learning process is divided into three-layer, the input layer, the hidden layer, and the output layer. The hidden layer makes decisions from the input layer and weighs up a stochastic change within itself, and improves the final output. The notable difference is that CNN is more suitable for computing inputted image data.

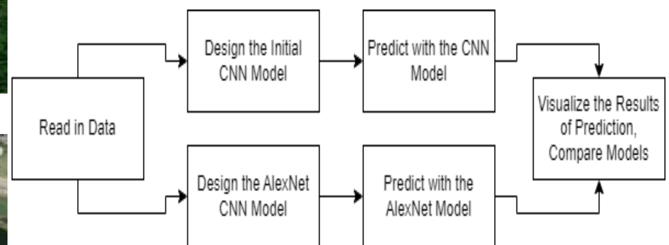
AlexNet is a specific design architecture of a Convolutional Neural Network designed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. It was designed to compete in the ILSVRC competitions which challenge contestants to create a model to best classify the images on the ImageNet dataset- a collection of over a million high resolution images divided into a thousand different classes. As of the 2010 and 2012 competitions, AlexNet outperformed all other contestant models so its ability to decipher objects from images may make it a valuable asset in object and pattern recognition for wildfire detection.

In the scope of wildfire detection using CNN, this paper is not alone; however, compared to another research project, by Karl Kaiser from Cornell University, with the same goal, we implemented Spark streaming and AlexNet architecture which they did not. Furthermore, the final accuracy of our CNN was comparable to their best models.

3.Design

Offline:

For creating this model, we will start by testing a Convolutional Neural Network (CNN) offline which will then be integrated later onto HDFS. We will test both a base, simple architecture for our CNN as well as an AlexNet Network.



See Above: The offline Pipeline

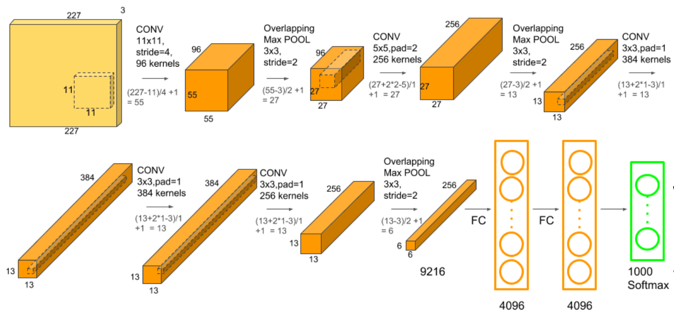
Our simple model uses an architecture based on prior work which was evaluated to a high accuracy. The structure for that may be seen below.

Model: "Wildfire-CNN"

Layer (type)	Output Shape	Param #
conv2d_28 (Conv2D)	(None, 254, 254, 8)	224
conv2d_29 (Conv2D)	(None, 252, 252, 16)	1168
max_pooling2d_14 (MaxPooling2D)	(None, 126, 126, 16)	0
dropout_28 (Dropout)	(None, 126, 126, 16)	0
flatten_14 (Flatten)	(None, 254016)	0
dense_28 (Dense)	(None, 32)	8128544
dropout_29 (Dropout)	(None, 32)	0
dense_29 (Dense)	(None, 2)	66

Total params: 8,130,002
Trainable params: 8,130,002
Non-trainable params: 0

AlexNet Network is a 8 layer deep neural network designed for classifying images. Created for a highly contested competition, AlexNet is one of the leading CNN architectures for image classification.

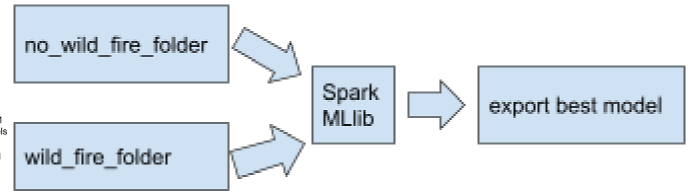


See Above: the design architecture for AlexNet

See Below: Our AlexNet Model

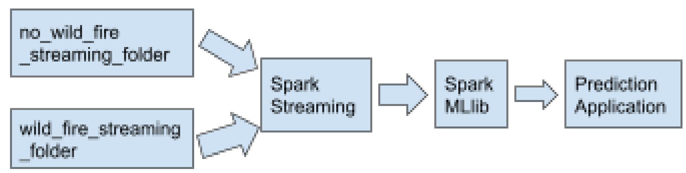
```
Model: "Wildfire-AlexNet"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 55, 55, 96)        34944

max_pooling2d_6 (MaxPooling  (None, 27, 27, 96)        0
2D)

conv2d_14 (Conv2D)           (None, 27, 27, 256)       614656

max_pooling2d_7 (MaxPooling  (None, 13, 13, 256)       0
2D)

conv2d_15 (Conv2D)           (None, 13, 13, 384)       885120

conv2d_16 (Conv2D)           (None, 13, 13, 384)       1327488

conv2d_17 (Conv2D)           (None, 13, 13, 256)       884992

max_pooling2d_8 (MaxPooling  (None, 6, 6, 256)         0
2D)

flatten_2 (Flatten)          (None, 9216)              0

dense_6 (Dense)              (None, 4096)              37752832

dropout_4 (Dropout)          (None, 4096)              0

dense_7 (Dense)              (None, 4096)              16781312

dropout_5 (Dropout)          (None, 4096)              0

dense_8 (Dense)              (None, 2)                 8194

=================================================================
Total params: 58,289,538
Trainable params: 58,289,538
Non-trainable params: 0
_________________________________________________________________
```

Online:



See above: the data pipeline of Spark Mllib

Firstly, We combine two folders, no_wild_fire and wild_fire_folder into one dataframe by labeling 0 for no_wild_fire, 1 for wild_fire. We convert image data into vectors so that machine learning modes can read. We apply some machine learning models and pick up best model and export it.



See above: the data pipeline of Spark streaming

After our model is created, we intend to integrate spark real time streaming to parse images as they are scanned (to emulate this, we will reserve some images aside to be fed in as if data were being collected and distributed in real time).

4.Experiments

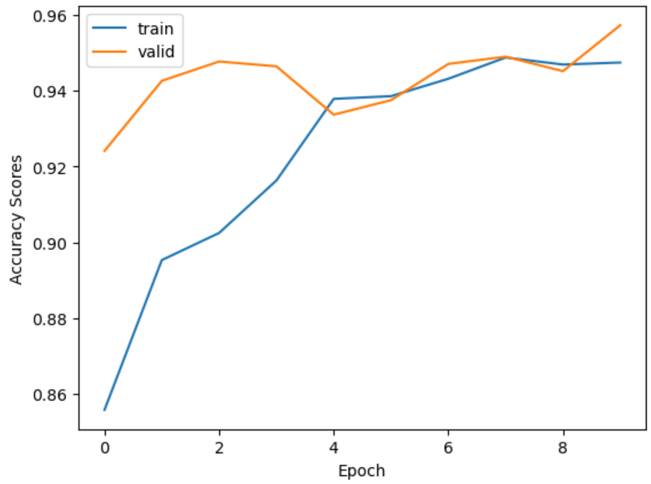
Offline:

Firstly, split the picture data into three groups(70%: training model,15%: testing model accuracy, and 15%: validation).To choose the best model, we will train two deep learning models, normal CNN, and AlexNet. After choosing the best mode, the model will be integrated to spark streaming, and our experiment will move to the real-time analysis part. In this part, we will make pipelines to predict wildfires.

Currently, the standard CNN has been made and evaluated to a high degree of accuracy (about 95% accuracy). The AlexNet based CNN has been tested and achieved an even higher degree of accuracy compared to the base CNN at 96% validation accuracy.

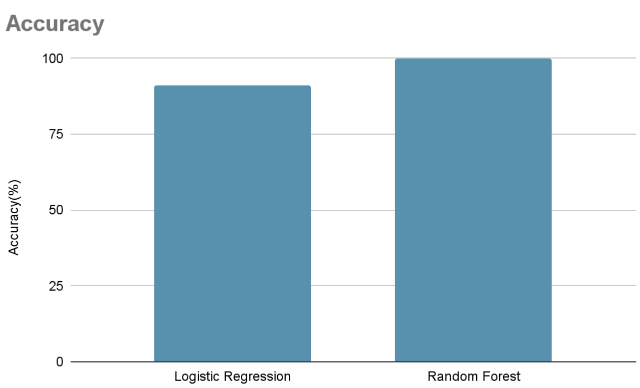
This is higher than our online models; however, the accuracy of those models is also quite high.

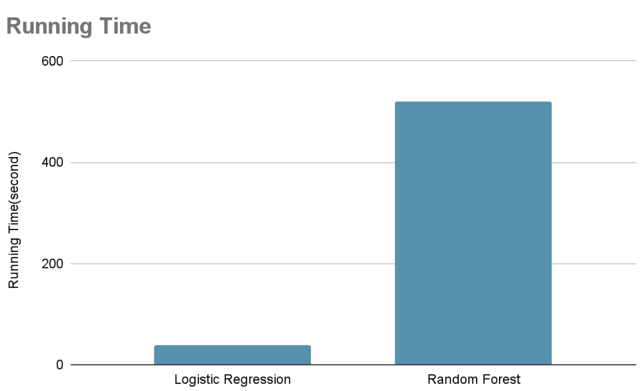
See below: a graph of our simple CNN's accuracy scores across training epochs



Online:

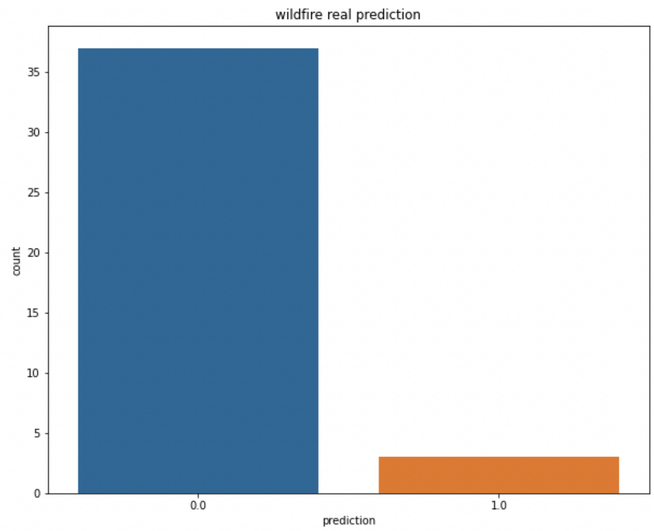
Firstly, We combine two folders, no_wild_fire and wild_fire_folder into one dataframe by labeling 0 for no_wild_fire, 1 for wild_fire. We Convert imagedata into vectors so that machine learning modes can read. We apply machine learning models, Logistic Regression and Random Forest.





See above: the outcomes of mllib model(time,accuracy)

After picking up the best model, make pipelines so that we can predict wild_fire realtime. To predict accepting images from videos or streaming, we make folders named no_wild_fire_streaming, wild_fire_streaming. From each folder, read 4 image files at one action and do image processing so that trained machine learning can read image data. When image data can be taken to machine learning model, we predict wildfire from every new 4 images which will add new pictures every 0.5 seconds. And the outcomes will be visualized as bar graph.



See the above: the application of wild_fire prediction

5. Contributions

- Offline: Data Preprocessing- Angelo, Yusuke
- Offline: Model Building- Angelo
- Offline: Data Visualization- Angelo, Yusuke
- Offline: Hypertuning- Angelo
- Online: Model Building- Yusuke
- Online: Spark Streaming- Angelo, Yusuke
- Online: Application Design- Yusuke

6. Timeline

For the first major milestone, we would like to have completed a functional model and begun steps towards testing HDFS and Spark integration

For the final milestone, we would like to have a model capable of receiving images in real time to make predictions.

7. References

[1] 1983-2023. Wildfires and Acres. Retrieved February 25th, 2023 from https://www.nifc.gov/fire-information/statistics/wildfires

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton 2017. ImageNet Classification with Deep Convolutional Neural Networks. Retrieved February 26th, 2023 from https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[3] Spark 3.3.2 Documentation. Retrieved February 26th from https://spark.apache.org/docs/latest/streaming-programming-guide.html

[4] Keiron O'Shea, Ryan Nash2015. An introduction to Convolutional Neural Networks. Retrieved February 27th, 2023 from Cornell University https://arxiv.org/abs/1511.0845