The Motions of Forest Fires Final Paper

1. Abstract

Can I make an AI model that can predict how forest fires will move? I am trying to take data about the environment and a current fire and predict where the fire will go / if it will grow or shrink. Forest fires are burning more and more each year, as evidenced by the record breaking fires in Australia in 2020. The top ten years with the largest annual acreage burnt have all happened since 2004, and forest fires are now causing 7.4 million more acres of tree cover loss per year since 2001.
This project would hopefully let firefighters predict where the fire is going to prevent it. The data I used was a graph of where the fire was and other features such as the surrounding wind speed, terrain, plant content, etc, and then the label comes from another graph showing where the fire moved to. I tried several different models, namely a Multi Layer Perceptron (or Neural Net), Logistic Regression and Linear Regression before settling on a CNN that predicts if each square kilometer will have a fire or not. I then was unable to get this to work so I settled on making a model that predicts how dangerous the fire will be. I was eventually able to get it to predict this with 60% accuracy. The major conclusion I had with this problem is that it is very tough but it seems like there is a way to predict where a forest fire will go. I would like to keep researching to see if I can find out how.

2. Introduction

My research project is to try to design an AI model that can predict where a forest fire would move to. This could help people know to evacuate their homes, firefighters know where to get prepared and remove brush, and identify possible hot spots where fires would go. I wanted to include all of the relevant environmental data. This problem differs from other AI problems because for a normal problem I would input an image or number and get out one number, for example, the probability the input image was a cat, or how popular a youtube video would be. It would be a regression problem if the model was predicting any possible number, and a classification problem if the model was limited to predicting one of only a few numbers, representing different labels. For example, 1 = cat, 2 = dog, and so forth. So this problem could be either depending how the question is framed and what model

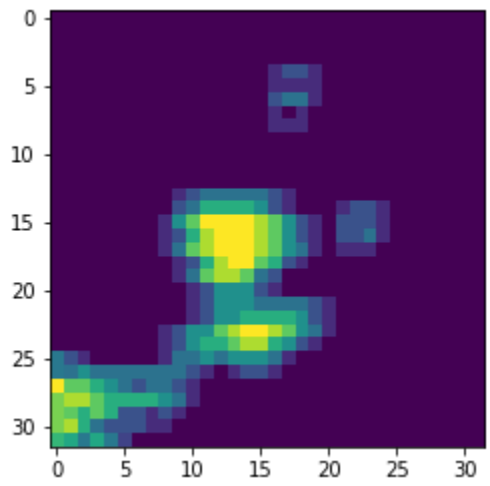
is used, and the data is manipulated from there. It leans more on the classification side, but the data can also be used for regression (arguably more difficult to get higher success from because of the sheer number of possibilities to predict). The data doesn't fit into a definition well either, in the sense that the data isn't quite images, and it isn't just numerical. It is a grid of numbers, where each number is either the humidity, or the elevation, and this grid kind of forms an image.

3. Background

Forest fires form a vicious cycle with climate change. Rising temperatures cause forest fires to be more common and more severe, and then the smoke pouring into the atmosphere creates more greenhouse gasses, furthering climate change. Droughts and heat waves cause acres and acres of land to be covered with dry shrubs that will burst into flame without notice and will quickly become full fledged wildfires. Some approaches to preventing forest fires have been clearing shrubs and monitoring dry spots. Firefighters known as hotshots are on the ground during a forest fire, and they create what is known as a firebreak, where they remove all possible fuel for the fire in a large area in the hopes that the fire won't be able to spread in that direction. Smokejumpers parachute into a fire to put out small fires that coil become large ones. In addition, some ground crews also start backfires, which are fires that advance towards the wildfire, burning up any fuel before the main wildfire reaches it. Other approaches being used to predict forest fires all try to predict where the forest fire will happen. They take in data about the environment and whether or not the fire happened and used it to predict where fires will spring up. I want to take this one step farther and predict where a fire will go.

4. Dataset

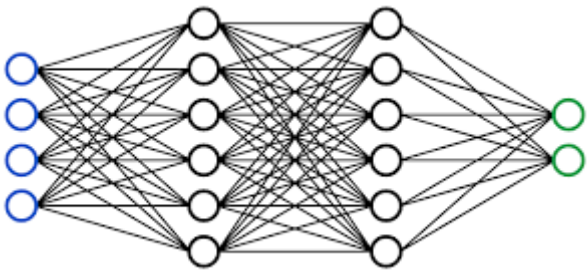
The dataset I am using is a large collection of 64 by 64 images, where each pixel shows if there was a fire there or not, the fire masks, and the elevation in meters, the Palmer Drought severity index, the Vegetation Index, the Precipitation in mm, the Humidity from 0% to 100%, the wind direction in degrees, the minimum and maximum temperatures, the wind speed in m/s, the NFDRS fire danger index energy release component, and the population density. Each pixel corresponds to a 1 km by 1 km square in real life. The data was stored in tfrecords, which were very difficult to work with, and resulted in having to try several different package and function installations to end up converting the data into a pandas dataframe and then a numpy dataframe, but it eventually worked. I made a function that averages each pixel with its neighbors so, if there is a little 1 km by 1 km fire with no fire around it, it gets a very low score, but if there is a large fire, the center gets a high score that gets smaller closer to the edge, see image. I decided to cut the 64 by 64 images into 32 by 32 ones, for ease in training the model, and took 100 of these images, out of the 18000+ that were in the dataset. One problem was that there were some fuzzy spots in the data where there was cloud coverage, so I needed to remove these. I was able to remove them from the fire masks showing the day before, but not from the ones showing where the fire went. I instead created a one-hot encoding for these, where I created three buckets for each pixel: (1, 0, 0) meant fire, (0, 1, 0) meant no fire, and (0, 0, 1) meant it was unknown if there was a fire in that pixel. This way the model could predict on a per-pixel basis for the fire movement. I also decided to apply the averaging function to all of the input data. I also chose a subset of only 10 of the variables to start training with for ease. I then split the data into training and testing with a 80% train size. Unfortunately, I was not able to get the project to work in this way, so I decided to make the model predict how dangerous the fire will get. I did this by adding up how many pixels were on fire, counting the unknown pixels as half.I then split the data into 5 groups, depending on how much fire there was in the image. I made a one-hot encoding of the data,



so that way the model would predict which group it would fall in and how dangerous the fire would be.

5. Methodology/Models

For this project I had a little bit of a challenge using a model to predict an image because of the hybrid nature of this question, so I had to try many different models. For this project I wanted to make sure the model would know what was happening around a pixel, not just right there, also to uncomplicate the training. I accomplished this with the averaging function that takes in surrounding pixel information for each pixel. The first route I thought to take was to run this averaging function on each image and then split each pixel apart from its source image and make predictions for each pixel independently based on the 10 variables. I would use a model like Logistic Regression for this, that would be a binary classification problem to predict if there was a fire there at that pixel. The problem with this method is that just doing the averaging is not enough for this and there are simply too many possible varying values. In other models that deal with images, like a CNN, the model is using every single part of an image together as opposed to pixels independently to make a prediction. The accuracy I would get from a Logistic Regression model used in this way would probably be pretty terrible. Another idea I had was to use a MLP (multi-layer perceptron) also called a Neural Network. This takes in the data in a flattened list of numbers, and runs like a string of logistic regression models with what's called neurons that multiply each piece of incoming data and then send it to other neurons, that multiply and sum values as well as pass through a non-linearity function of choice and then send it to other neurons, and on and on until the one final neuron that took in all of the data that makes the prediction; see image. I wanted to convert all of my data into a list of all the different values for each image, independent of where in the image it was (this is only possible in a CNN model), and put it into this model with a 1000 output neurons that would each represent a pixel in the image and give the probability of there being a fire in that pixel location. The problem with this solution is that the model was not able to understand what all of the data meant, and there were too many possible labels and values for each categorical label to predict, as well as no spatial awareness of the original image since it was flattened. Also a MLP is not able to produce multiple predictions that are also binary. They really aren't made to predict



lots of different labels with infinite value (regression, not classification in buckets), and these are probabilities that map to what each variable stands for (probability of each class/label.).

After all this I decided to try a CNN. The benefit of a CNN is I could keep all my data in the right order, so the model would know where each pixel was from and what each piece of data meant, but also they have excellent accuracy and can keep the data in an image form for longer (until the last Dense layer). They work by taking the image and running a series of filters on it to look for vertical lines, or dark spots, or a type of shape, and then extracting that information and looking for bigger shapes in that data. Eventually it takes all the data it found and gives it to a layer of neural network that runs on the data and makes the predictions for each class. They are extremely useful and powerful and are used all the time. Lots of the time with these models the data is compressed so only the important information is left, making it run faster and more accurately, but I didn't want this because I wanted my image to stay a large image for the final prediction. When I started I was using a sigmoid activation, which takes the score the neurons give and makes it a probability, in this case of there being a fire at a pixel. The problem I had was that this was giving probabilities, not actual ones and zeroes, so my model never knew what the label categories were. After this I tried one-hot encoding my labels, which categorizes each pixel into one of three numbers, like (0,1,0). I then made it so that the model used softmax to predict which of the three classes it would be. The problem with this is that CNN's are not able to do multi-class prediction on all of the pixels at once, rendering this solution impossible. With no other solution to try, I decided to alter the question and simplify the model to predict how dangerous the fire will be overall, not pixel-wise predictions. I added up how many pixels contained a fire within each image, counting unknown squares as .5; if there were less than 10 out of the 1024 pixels with a fire in that spot, it was a 'little' fire, if there were under 20 total pixels it was a 'medium' fire, under 40 it was a 'large' fire, under 50 it was a 'very large' fire and over that it was a 'huge' fire. The largest label in my dataset had a total of 165 pixels containing fire, but most were around 20 or less. I then did a one-hot encoding and predicted with softmax. I also added in some Maxpooling layers, which compress the data, which greatly increased training speed and accuracy and prevents over-fitting.

6. Results and Discussion

My final model predicted how much fire there would be the day after the photo was taken. It was a CNN with 8 layers that used categorical cross-entropy to calculate loss, and it calculated its accuracy with accuracy and precision metrics. The first layer was a convolutional layer that applied 128 filters with a kernel size of 3 and had padding and ReLU activation. The kernel size is how many pixels it looks at, in this case, 3x3, the filter is how many different ways it looks for patterns, and the padding adds a border of zeros, which lets the kernels look for patterns at the very edge of the image. ReLu activation means the model makes all negative values equal to zero, and makes the model have a more complex way of looking at the problem.
The second layer was also a convolutional layer this time with 64 filters and no padding, also with ReLU and a kernel size of 3. The third layer was the same as the one before, this time with padding, and the fourth was a Max Pooling layer, which takes the largest value from each area and gets rid of the rest, which lets the model look for the important information. The Max Pooling layer had a kernel size of 2. Then there was one final convolutional layer with 32 filters, and no padding.
After this there was a flatten layer, which turns the image into a string of numbers, and a Dense layer with 408 neurons and ReLU activation. The final layer was a Dense layer with 5 neurons and softmax activation. Softmax makes each neuron output a probability with all of the probabilities adding up to 100%. The entire model was compiled with a RMSprop optimizer with a learning rate of .0001 and a decay of 1e-6. The learning rate is how fast the model changes its variables, like the filters and neurons. Categorical cross-entropy is the sum of how far the model was from what it was supposed to say. It is used for classification commonly. Accuracy is just how much of the time the model was correct, like it was correct 50% of the time, and Precision is how many times it said true that it was right. I trained the model for 15 epochs, (how many times it adjusted its variables) with a validation accuracy of around 60% every time I ran it and a similar precision. If the model was making random guesses it would have an accuracy of 20% and a precision of 20%.

7. Conclusion

In this project, I set out to make a forest fire-predicting model. I tried out several different models with several different approaches, namely, Logistic Regression, a Neural Network and a CNN to try to find a way to predict where a forest fire would go and if it grew or shrink depending on several input variables. Eventually though, I had to change the question slightly to predict how big the forest fire would get, not exactly where it spread or shra(The Associated Press 2022)nk to, and this is useful information to show how dangerous a fire will become which is a prevalent worldwide problem. I think I underestimated the difficulty of this question and dataset, especially in the time constraints, but I learned a lot. As a next step, I could make a way for the model to predict the ratio of the size of the fire from one day to the next. Eventually I would like to try a U-net which is used to take in an image and provide a different version of the same image, which might be able to do something for this project. I also want to keep tuning the hyperparameters to try to get higher and higher accuracy and prevent overfitting which is happening slightly in my CNN. Overall, this project was a great learning opportunity to build from scratch and explore unknowns, and is a socially impactful domain.

8. Acknowledgements

I would like to thank Inspirit AI for making this all possible, especially my teacher Sophia for guiding me every step of the way.

9. References

(The Associated Press 2022; Kimbrough 2022; Rosebrock 2021; InspiritAI n.d.; Boose 2022; Gupta 2021))

Boose, Yvonne. 2022. “Are Wildfires Getting Worse Due to Climate Change?” August 18, 2022. https://blog.breezometer.com/are-wildfires-becoming-more-common/.

Gupta, Ayush. 2021. “A Comprehensive Guide on Deep Learning Optimizers.” *Analytics Vidhya* (blog). October 7, 2021. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/.

InspiritAI. n.d. “Convolutional Neural Networks.” Google Colab Pro. Accessed August 28, 2022. https://colab.research.google.com/drive/1DKuvKvTGL4HBhfENnzMIrXwnqr7Hq6rw#scrollTo=LFVHyPKn-V4N.

Kimbrough, Liz. 2022. “Forest Fires Are Getting Worse, 20 Years of Data Confirm.” Mongabay Environmental News. August 17, 2022. https://news.mongabay.com/2022/08/forest-fires-are-getting-worse-according-to-new-20-year-analysis/.

Rosebrock, Adrian. 2021. “Convolutional Neural Networks (CNNs) and Layer Types.” *PyImageSearch* (blog). May 14, 2021. https://www.pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/.

The Associated Press. 2022. “Four People Have Died in Northern California’s Huge Wildfire.” *NPR*, August 2, 2022, sec. National. https://www.npr.org/2022/08/02/1115214709/california-wildfire-deaths-mckinney-fire.