Wildfires Detection Using UAV Images

Ricardo Gonzalez, 2003297

Abstract—This my abstract And this is how it ends

Index Terms—CNN; Deep Learning; Wildfires

1 Introduction

Convolutional neural networks (CNN's) currently are one of the must explored topics due to their high efficiency in several areas of ML, in particular Computer Vision and Image Recognition. This led to the creation of popular datasets such as MNIST and ImageNet, which are used nowadays to introduce the topic or to pre-train the CNN's to be able to perform Transfer Learning for more specific tasks. This had led to the researchers to not only improve the performance of the CNN's but also found a use for them, one of these cases have been for the detection of Wildfires.

Wildfires currently are on of the must dangerous natural disasters that are threating the world. According to an investigation done by the NOAA (National Oceanic and Atmospheric Administration), the emissions produced by the wildfires often lead to harmful pollution not only in the area near the wildfire but also can extent even farther from the area, harming humans and other specials alike [1]. In addition to that, there are ecosystems which fauna and flora are being threatened due to the fires and to human intervention. Being that the case of the Amazon Rainforest, where the mentioned causes alongside droughts, could led to a possible "tipping point" where the Reinforces would be unsustainable in the case that there's no effective intervention of the matter [2].

Due to the formerly mentioned there have been attempts by the scientific community to develop models that can detect based on the image if there is fire or not. Leading to the creation of datasets such as FLAME, a dataset created and provided by the IEEE [3], or less academic but still important FIRE dataset hosted on Kaggle which is also home of several images showing environment with and without fire [4].

Nevertheless, the models are not very accurate, as an example the model trained with the FLAME dataset had only a 76% accuracy [3], showing that there are still are of improvement in the regard. And with the knowledge that the FLAME model was trained only with the FLAME dataset, it's possible that using both the FLAME and Kaggle dataset in addition to some data augmentation and a new model it will be possible to improve the score previously obtained solely by the FLAME dataset.

2 Background / Literature Review

2.1 Convolutional Neural Network

Are a type of Neural Networks commonly used for the areas of Image Recognition and Computer Vision, this type of networks have the main feature of being divided in several deeply connected convolutional and pooling layers, to finally end with a classifier which is a fully connected layer comparable to a layer of traditional Multi-Layer Perceptron (MLP).

2.1.1 CNN Architectures

As CNN's are one of the state of the art topics in research nowadays, there have been many popular architectures that either have very high performance, or have very little parameters with the objective of having a lightweight model with good performance. For developing actual applications for a problem researchers actually use this predefined architectures to solve the problem they're facing as this provides the advantages of being a proven useful architecture.

In addition that due to their popularity people train these models with popular datasets so researchers can use these pre-trained weights enabling to do transfer learning, which is when using an already trained model, retrained the top layer to be able to have a good performance in the new problem.

Actually in the case of the FLAME research done, it's mentioned in the paper that the model that achieved a 76% of accuracy was using an architecture called Xception [5].

Another example of an architecture is EfficientNet published in September 2019, that wanted to tackle the problem of scaling up an architecture by having a smaller amount of parameters, but also by keeping/increasing the accuracy of the model, this was able by the use of compound coefficient [6].

Also it exists the architecture of ReXNet which was proposed in July 2020, aiming to follow the trend of creating accurate and lightweight architectures, distinguishing themselves by the use of representational bottlenecks. The result was quite good improving the accuracy when doing transfer learning from trained models using the COCO dataset, while keeping a small amount of parameters to train [7].

2.2 Data augmentation

Is a method that allows to increase the amount of data that we have in a dataset, allowing to be persistent being that you generate new data based on the previous one and save it on the same dataset. Or add randomness prior to training a batch, making that every epoch and batch should not consist of the same data in the same order [8].

For images datasets, normally the transformation/augmentations done to increase the dataset is by performing modifications in the same image being by simple image transformations such as shearing, rotation and translation. But it also may be the modification of the brightness, flipping the image vertically or horizontally, or by cropping it [9].

3 Methods

Following it will be described the methods used for each of the sections that we used for the research. If you're actually interested in the code this can be found in the github repo [1]

3.1 Dataset

As mentioned in the section 1, the objective of this research is to use both the FLAME and Kaggle datasets, thus prior to training both datasets will be merged. As the distribution of the classes end up being skewed, it will be performed as part of the preprocessing data augmentation in the images of the lower class (No_Fire), to perfectly balance both classes

3.1.1 Data Augmentation Transformations

All random transformations have a probability of 50% of happening

- A reduction or increase of brightness and contrast within a range of 0.75-1.25. The final number is defined by a uniform probability
- A random rotation of 5º
- A random horizontal flip and a random vertical flip

3.2 Model

As mentioned in the section 2.1.1, the architecture that the FLAME researchers used was the Xception architecture, which improves over its previous version called Inception, which improves primarily the performance while keeping the same amount of parameters. Nevertheless, the architecture could be consider old, being published in 2017 and many other architectures that have improve accuracy, or just other models which already had better performance than this case.

Due to the former, if we want to improve the performance of the classifier it's recommended that we attempt to use other architectures. For this case the models consider to test will be EfficientNet and ReXNet. With these models instead of training randomized weights we will do transfer learning on a model trained using the ImageNet dataset [10], training only the top layer which is a densely connected layer in both cases.

3.3 Training

For the training must things will be very straightforward, the validation will be checked splitting the training dataset into 80% training and 20% validation, just like in the FLAME paper. In addition this training dataset will also have a set of transformations both to do data augmentation and to normalize the data, this with the intention to not always have the same data in a batch every epoch and give diversity and randomness to the training.

3.3.1 Optimizer and Criterion/Loss Function

The optimizer chosen for the training is the Adam optimizer. Meanwhile, the criterion used to calculate the loss of the model will be due to being a binary classification problem, the binary cross-entropy loss function with logits, meaning that a Sigmoid function will be used as a final activation function and based on this the loss function will calculate the current loss.

1. https://github.com/ricglz/CE888_activities/tree/main/assignment

3.3.2 Hyperparameters

- Batch-size: 32
- Learning Rate: 5e-5
- Max epochs: 15

3.3.3 Callbacks

- LRScheduler, this callback modifies the current learning rate based on the current epoch, leading to a function that has the following structure, using as max_lr: 5e-3, min_lr: 1e-5, num_warmup_steps: 6 and num_training_steps: 9

```
if epoch <= num_warmup_steps:
    return log(max_lr / lr) /
           log(num_warmup_steps)
return log(min_lr / max_lr) /
       log(num_training_steps)
```

3.3.4 Transformations

All random transformations have a probability of 50% of happening. And the random transformations are only for the training dataset

- Resize the image to a size of (254, 254)
- A reduction or increase of brightness and contrast within a range of 0.75-1.25. The final number is defined by a uniform probability
- A random rotation of 5º
- A random horizontal flip and a random vertical flip
- Normalization using the mean and std of each channel of the ImageNet dataset

4 Results

Model	Accuracy	Loss
ReXNet	99.62	0.0293
EfficientNet	98.28	0.0582
FLAME	96.79	0.0857

TABLE 1: Model's accuracies and losses in training dataset

Model	Accuracy	Loss
ReXNet	99.62	0.0136
EfficientNet	98.32	0.053
FLAME	94.31	0.1506

TABLE 2: Model's accuracies and losses in validation dataset

Model	Accuracy
ReXNet	71.46
EfficientNet	61.46
FLAME	76

TABLE 3: Model's accuracies in test dataset

To determined which of the models trained is the best, it will be used the information displayed in table 3 showing the results of the models in the testing dataset, showing that the ReXNet is definitely the best model of the ones which were trained. Nevertheless the results doesn't seem as favorable,

having an accuracy of 71%, which means a 7% decrease in the accuracy compared to the model done by the FLAME team. All of this seems to be a case of over-fitting as the current model outperforms FLAME's, with a significantly decrease in both the validation and training loss, showing also an increase in the accuracy in both datasets as shown in table 1 and 2.

Some other things it should be consider is that due to hardware constraints and to be 100% to transfer learning, it was only trained the top layer of the CNN. This brought improvements, such that it was necessary to train the model in less epochs and probably in less time in the same hardware not only due to the use of transfer learning but also due to the nature of the architecture itself, nevertheless it could also be the reason that it's difficult to tune for a better generalization more specific to our current situation. This shows the advantages and disadvantages that can bring transfer learning.

5 Conclusion

At the end we were not able to actually outperform in the testing dataset, even though we augment the dataset and trained a model using transfer learning of the ImageNet dataset. Nevertheless, the model and training overall seems to have potential, as it was able to have a decent performance with fewer epochs and with an architecture that requires less resources. Being an advantage specially if the model ends up being used in a IoT environment where efficient models are required due to the hardware that is being used.

Also even though the performance on the test dataset is not the best, it doesn't mean that there's not hope for actually outperforming what we have as the best model yet, as there are some areas which could help to improve the overall performance of the models. Such as training the inner layer of the CNN, not only the top layer, this is known as Fine-tuning and it seems like it will be the next step to take to train and improve the performance of the models.

In addition to that, it's possible that new architectures can bring even more potential results, being the one that caught my attention was the recently published RepVGG. A new approach to the traditional VGG architecture which aims to simplify the architecture improve the speed of preprocessing and inference, while also keeping a good accuracy, a model that attempts all of this while avoiding the know complexity that the current state-of-the-art models have [11].

References

[1] NOAA, "The impact of wildfires on climate and air quality," Online, 2020. [Online]. Available: https://csl.noaa.gov/factsheets/csdWildfiresFIREX.pdf

[2] Y. Malhi, L. E. Aragão, D. Galbraith, C. Huntingford, R. Fisher, P. Zelazowski, S. Sitch, C. McSweeney, and P. Meir, "Exploring the likelihood and mechanism of a climate-change-induced dieback of the amazon rainforest," Proceedings of the National Academy of Sciences, vol. 106, no. 49, pp. 20 610–20 615, 2009.

[3] A. S. F. A. A. R. L. Z. P. F. E. Blasch, "The flame dataset: Aerial imagery pile burn detection using drones (uavs)," 2020. [Online]. Available: https://dx.doi.org/10.21227/qad6-r683

[4] Ahmed Saied, "Fire dataset," Online, 2020. [Online]. Available: https://www.kaggle.com/phylake1337/fire-dataset

[5] A. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P. Z. Fulé, and E. Blasch, "Aerial imagery pile burn detection using deep learning: the flame dataset," 2020.

[6] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[7] D. Han, S. Yun, B. Heo, and Y. Yoo, "Rexnet: Diminishing representational bottleneck on convolutional neural network," 2020.

[8] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," Journal of Computational and Graphical Statistics, vol. 10, no. 1, pp. 1–50, 2001.

[9] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017.

[10] R. Wightman, "Pytorch image models," https://github.com/rwightman/pytorch-image-models, 2019.

[11] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," 2021.

6 Plan

- Week 7 (March 1st-7th). Implement fine-tuning of both models to attempt to improve the accuracy
- Week 8 (8th-14th). Implement initially implementation of RepVGG
- Week 9 (15th-21st). Fine-tune RepVGG
- Week 10 (22nd-28th). Investigate new ways to improve the score (probably using new datasets)
- Week 11 and 12 (March 29th - April 12th). Implement techniques researched prior week
- Week 13 (13th - 25th). Write research paper