MATH70103: Unstructured Data Analysis
Imperial College London
Autumn 2023
CCID: 00951537

Final Project

Plagiarism Statement: *The following assignment is a product of my own work.*

Abstract

In this report I explore the effectiveness of Convolutional Neural Networks in identifying images that contain fires in the wild, with the aim of producing an algorithm that could potentially be incorporated in early wildfire detection systems. I make use of the FLAME (Fire Luminoscity Airbourne-based Machine learning Evaluation) dataset [10] and build upon the *Xception* Deep Convolutional Neural network proposed by Google [4, 14, 13]. Key metrics such as binary accuracy, precision and recall are presented for training, validation, testing, and out of sample testing, the latter of which examines the performance of the model on images outside of the *FLAME* dataset. For this purpose, I use the Fire Classification dataset available on Kaggle [6].

All code and a 100MB subset of the FLAME Dataset have been uploaded to the following Github Repositry: `https://github.com/martinbatek/IC-UDA-Final-Project.git`

Contents

1 Introduction and Problem Statement

Wildfires are a pervasive issue across the globe today, causing millions of dollars in damage every year, loss in natuaral biodiversity in the affected areas, and significant loss and trauma to the affected communities. Recent examples of significant wildfires include the 2023 Canadian wildfire season [9], the 2019-20 Australian bushfires (nicknamed "The Black Summer") [7], and the fires in my home town of Cape Town this Christmas season [8]. It is widely known that natural occuring fires are an integral part of many ecosystems, as they clear out dead organic matter that prevents living orgnisms from accessing vital nutrients [12], alongside numerous other benefits [2]. However multiple sources cite an increasing rate of occurence and degree of

severity of global wildfires, past the point of biological sustainability [<**empty citation**>]. This phenomenan is widely attributed to human intervention such as ... .
The above justifies the need to develop novel methods for detecting wildfires, so that we may anticipate and manage the impact of wildfires in the best way possible. Upon reviewing multiple early-wildfires detection systems, Bouguettaya et al. found that the combined used of unmanned aerial vehicles (UAVs), mounted high resolution imagery and Deep Learning techniques provided the most promising results in terms navigability, versitility, speed and accuracy. Shamsoshoara et al. introduce a high-resulution image dataset composed of drone footage frames that were taken during a perscribed burning slash piles in Northern Arizona. In that same paper, the Google-Keras *Xception* network was applied to create a binary classifier model for fire image detection.
The problem statement for this report is therefore as follows:

Problem Statement:

- **Train a Convolutional Neural Network classifier for fire image classification.**
- **Investigate the effect of the preprocessing, hidden and output layers of the trained CNN model on an example image.**
- **Evaluate the Accuracy, Precision and Recall of the model on Validation, Testing and Out-of-Sample images.**

2 Image Dataset

2.1 Data Selection

Multiple datasets were considered for the purposes of this report, from sources such as Kaggle, and other publicaly available portals. Besides the stated criteria for this project (complexity and originality), I also aimed to use a dataset that is suitable given the findings in "A review on early wildfire detection from unmanned aerial vehicles using deep learning-based computer vision algorithms", wherein Bouguettaya et al. argue that UAV-sensor technology is more suitable for autonomous fire detection systems in terms of cost, accuracy and practicality than alternatives such as satellite imagery or stationery platforms. The ideal dataset would therefore include images taken from the perspective of UAVs.

To this end the FLAME Dataset produced by Shamsoshoara et al. addresses these criteria explicitly. The dataset includes drone imagery taken of the perscribed burning of slash piles in the forests of Arizona during January 2020, during which time the weather conditions were generally cloudy with an average temparature of -6 °C. The following Exploratory Data Anylisis includes visualizations of a few example images in the dataset. The full published dataset includes video footage, images, a variety of specturms used (including the normal RGB, fusion, white-hot and green-hot palettes), as well as images and masks for fire image segmentation (identifying the fire pixels in an image, thereby identifying where the fire is situated from the perspective of the UAV). For the purposes of this report, I use only the **Training, Validation and Test** classifiction images (all in the normal RGB spectrum), namely items 7) and 8) in the following IEEE-Dataport link: https://ieee-dataport.org/open-access/flame-dataset-aerial-imagery-pile-burn-detection-using-drones-uavs.

The classification dataset above is 1.46 GB in size. For the purpose of submission, I have produced a 100MB subset of this dataset using random sampling, and uploaded it to the the aforementioned Github repository, under `data/FLAME_dataset_subset`. However, note that the model and results described in the following report were compiled on the basis of the entire set of images.

2.2 Exploratory Data Analysis

In this section, I aim to visualise the data downloaded from points 7) and 8) on the aforementioned IEEE-Dataport. As a first step, I visualise the directory heirarchy structure in `scripts/UDA_FinalProject_Batek.ipynb`:

```
-FLAME Dataset - Shamsoshoara
    -Test
        -Fire
            -5137 files
        -No_Fire
            -3480 files
    -Training
        -Fire
            -25018 files
        -No_Fire
            -14357 files
```

Figure 1: File structure of the Classification images in the FLAME Dataset

Figure 1 shows that the dataset contains 47992 images, 8617 (18%) of which are in the test set with the remaining 82% in the training/validation set. Examining the file counts clearly shows that there is a significant class imbalance between Fire and Non-Fire images, and that the Fire Class is overrepresented in the training set relative to the Test set. This imabalance will likely influence model specificity and recall later on.

As a next step, I investigate the image dimensions and the pixel values in the training set:

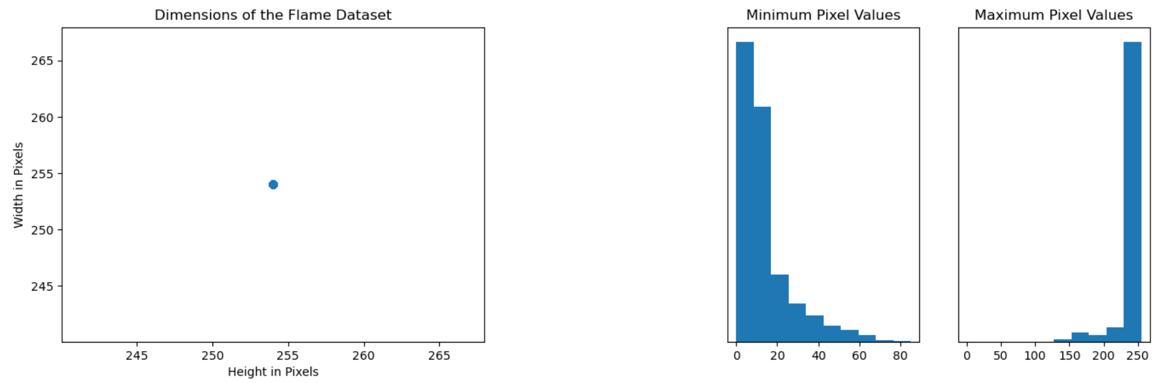


Figure 2: Training set image dimensions in pixels (left), and distribution of minimum and maximum pixel values (right).

It is shown in Figure 2 above that all of the images in the dataset have been rescaled to 254x254 pixels (left), whereas the pixel ranges vary between 0 to 255 (right). This is taken account of in the preprocessing layers of the CNN model later on.

As a final step in the Exploratory Data Analysis, I randomly choose 1 image for each set/class combination and display them:
The images in Figure 3 appear as expected given that they were taken in Arizona in the winter month of January. The weather at the time was generally cloudy, and despite the fact that most of the images do not show any portion of the sky (the drone mounted cameras tend to be pointed at the ground), the affect of the weather on the lighting is apparent in the images. The ground in the images tends to be covered in snow, with the crucial exceptions of the fires and under the

Figure 3: Four images from the FLAME dataset

trees. This all raises the question as to whether a CNN model trained on these images would perform equally well in identifying fires in images taken in different environments and weather conditions (such as during the summer, when the risk of wildfires is the highest). It is for this purpose that I evaluate the CNN model against out of sample images later on in the report.

3 Analysis

In this section, I will introduce and justify the use of the Convolutional Neural Network architecture that I have chosen when training the classifier model for image fire detection. This is a simplified version of the Google-Keras Xception network adopted by Shamsoshoara et al. in "Aerial imagery pile burn detection using deep learning: The FLAME dataset". I will also discuss the model's accuracy, precision, recall and AUC statistics during training, validation and testing against unseen in-sample and out-of-sample images.

3.1 The Google-Keras Xception Network

The Xception Network is a deep Convolutional Neural Network (DCNN) developed by Google researchers Chollet [4] and Szegedy et al. [14, 13]. It builds upon previous the previous Inception architecture by replacing the Inception modules therein with depth-wise separable convolutions.

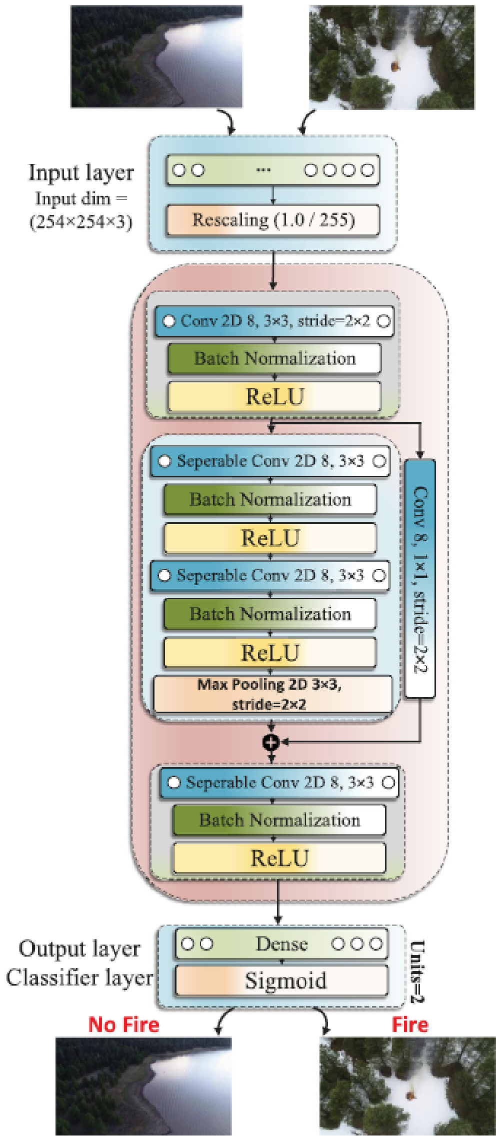


Figure 4: Xception Network graph, as used in [10]

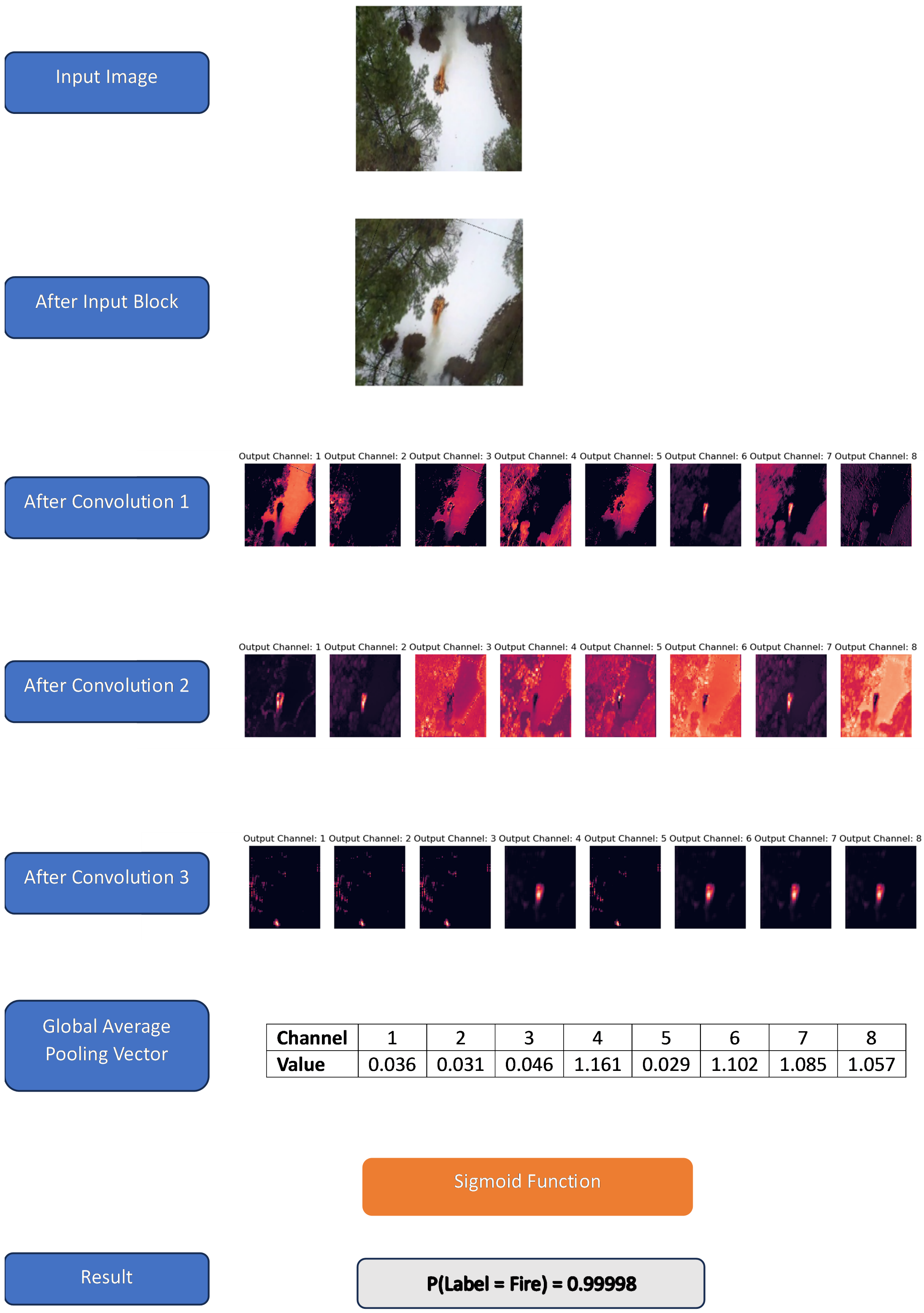
For the purposes of this report, I make use of and build upon a simplified version of the Xception framework used in [10], which is visualized in Figure 4. The network can be subdivided into input (top - blue), hidden (middle - red) and output (bottom - blue) blocks. The input block for the model used in [10] initially only included an input and a Rescaling layer for mappping the integer 0-255 range pixel values to float 0-1 values. I further incorporated a Random Data Augmentation layer (a layer sequentially containing Random Flip and Random rotation layers) in order to introduce some variability to the input data during training, so as to reduce model overfitting and improve general applicability. This was particularly necessary in the case of the FLAME dataset, which is comprised of video footage frames, and therefore has a high degree of internal similarity. (In plan language, two frames near the same point in a video are going to materialy look the same).

The hidden block is comprised of several depthwise separable convolutions, and a shortcut layer between the first and last convolution blocks. The first set of layers in the hidden block contains a 2 dimensional convolutional layer with 8 output channels, a 3x3 kernal and a stride of 2. The remainder if the hidden block contains separable 2D convolutional layers with similar settings. Each convolution block includes a Batch Normalization layer as well as a Rectified Linear Unit activation layer. The prior speeds up the training process by and reduces the likelihood of overfitting by decreasing the importance of the initial model weights, wheres the latter floors the pixel values to zero.

Finally, the output layer summarizes the average pixel values in the 8 channels outputted from the hidden block into a single vector in $\mathbb{R}^8$, $\vec{x}$. For the purpose of binary classification, this is then converted to a probability score by use of the *Sigmoid function*:

$$P(\text{label} = \text{fire}) = \frac{1}{1 + e^{-\vec{\beta}\cdot\vec{x}}}$$

where $\vec{\beta}$ is the weight vector for the last layer in the network.
Figure 5 demonstrates the result of each distinct block in the model. Firstly, we see how the input layer augments the original input image by randomly flipping and rotating it. We then see how each convolution block (2D convolution, batchnormalization and ReLU activation) incrementally segments the image, and places emphases on the pixels that contain the fire. The Global Average Pooling layer then summarizes the average pixel values per channel, wherein the channels that emphasis the fire more directly have a higher score. Finally, the Global Average Pooling vector is passed through the Sigmoid function to produce a final probability score for the image.



Channel	1	2	3	4	5	6	7	8
Value	0.036	0.031	0.046	1.161	0.029	1.102	1.085	1.057

Figure 5: Model layer convolutions applied to an example image

3.2 Training and Validation Results

As in [10], the model was written using the `tensorflow` Python package, and was trained with the objective of minimizing the *binary cross-entropy* loss function, which is suited for binary classification:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(p(\hat{y}_i)) + (1 - y_i) \log(1 - p(\hat{y}_i))$$

The Adam optimizer is used to minimize the loss function and find the optimal model weights. Please see the DCNN Model training section in `scripts/UDA_FinalProject_Batek.ipynb` for the training implementation. The code is compatible with Python version 3.7.12, Tensorflow version 2.3 and Keras version 2.4. For a full list of dependancies, I have exported a requirements.txt, environment.yml and spec-file.txt to the root of the Github Repositry [1]. As I currently do not have access to any computational resources besides my Microsoft Surface Laptop 4, training was computed via CPU only. Due to this limitation, I reduced the number of epochs for training to 20, as opposed to 40 in [10]. The training procedure took approximately 5 hours and 35 minutes to complete using all of the images in the classification Training folder (item 7 in the IEEE Dataport).

Figure 6 shows the training results in terms of Training and Validation loss and accuracy per epoch. For both Binary Cross Entropy Loss and Binary Accuracy, we see a realitively steady improvement for each subsequent epoch, in training as well as the validation hold-out comparisons. This shows little to no evidence of model overfitting, as that would be indicated by a decrease in training loss conciding with an increase in validation loss (and vice versa for accuracy). Figure 7 reveals the same trend in terms of AUC, Precision and recall.

3.3 Testing on Unseen Data

For the next step, I evaluate the model against the FLAME Dataset Classification Test images (item 8 in [11]). Figure 8 shows the result. From this figure we can see that the training set class imbalance that was flagged earlier in the EDA section has indeed resulted in a model that is more prone to assigning a higher likelihood that a given contains a fire. The model correctly identifies 82% of images that truely contain a fire (see recall on the right), but this comes at the cost of a high number False Positives. The 3252 identified dominate the confusion matrix on the left, and the precision score of only 47% is the prime reason for the low Binary Accuracy of 55%, in comparison to the Validation Binary Accuracy of 97% achieved during training.

The most straightforward methods of remediating the issue posed by class imbalance are either to introduce an *initial bias* at model initiation or to introduce *class weights* for model training [5]. The prior effectively teaches the model to initialize the training weights that reflects the overall $P(\text{label} = \text{Fire})$ in the training set, whereas the latter would assign a higher weight to the Loss calculated for "No Fire" samples during training and optimization, effectively forcing the model to "pay more attention" to fitting this class with higher accuracy. The third remediatory adjustment that can be made is to increase the threshold for classifying the Fire images. In Figure 8, it is assumed that any image with a score exceeding 0.5 is classified as Fire by the model, and the number of False Positives can be reduced by increasing this threshold.

3.4 Testing on Out of Sample Data

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem

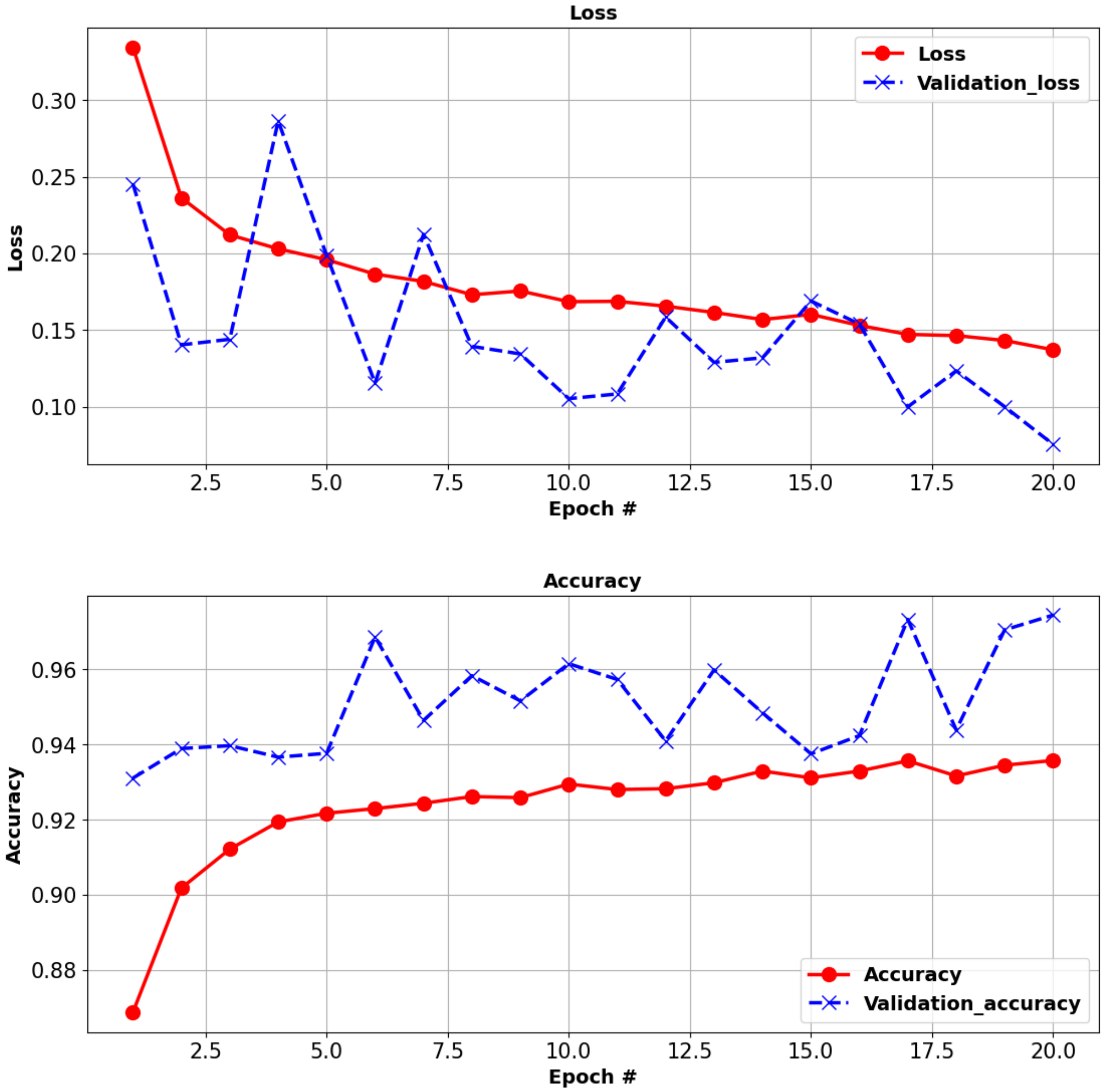


Figure 6: Loss and Accuracy during training and against 15% holdout validation

ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

4 Summary and Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam,

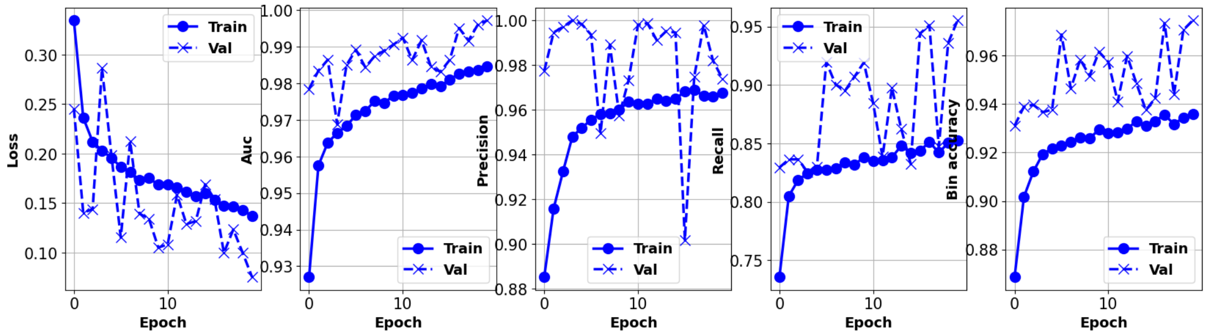
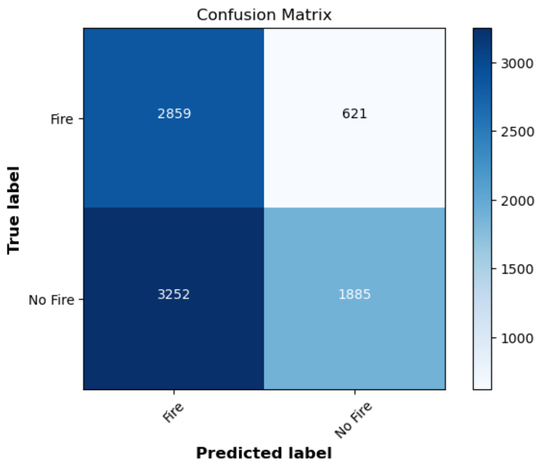


Figure 7: Accuracy, AUC, Precision, Recall, Binary Accuracy for Training and vs Validation




```
Note that Positive indicates "Fire"
loss: 1.4412615299224854
tp: 2859.0
fp: 3252.0
tn: 1885.0
fn: 621.0
accuracy: 0.003829639172181487
bin_accuracy: 0.5505396127700806
precision: 0.4678448736667633
recall: 0.8215517401695251
auc: 0.6649035215377808
```

Figure 8: Left: Test Confusion Matrix. Right: Test Results

luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

References

[1] Martin Batek. *Unstructured Data Analysis Final Project*. URL: https://github.com/martinbatek/IC-UDA-Final-Project.git.

[2] William John Bond and RE Keane. "Fires, ecological effects of". In: *Reference module in life sciences* (2017), pp. 1–11.

[3] Abdelmalek Bouguettaya et al. "A review on early wildfire detection from unmanned aerial vehicles using deep learning-based computer vision algorithms". In: *Signal Processing* 190 (2022), p. 108309. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2021.108309. URL: https://www.sciencedirect.com/science/article/pii/S0165168421003467.

[4] Francois Chollet. "Xception: Deep Learning With Depthwise Separable Convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[5] *Classification on imbalanced data*. Accessed on 2 Jan 2024. Tensorflow, Oct. 2023. URL: https://www.tensorflow.org/tutorials/structured_data/imbalanced_data#calculate_class_weights.

[6] Ahmed Gamaleldin et al. *FIRE Dataset*. https://www.kaggle.com/datasets/phylake1337/fire-dataset. Accessed: 2023-12-18. 2019.

[7] Megan Harwood-Baynes. "Australia bushfires: Devastating flames released more than twice the amount of CO2 than previously thought, says study". In: *Sky News* (Sept. 2021). URL: https://news.sky.com/story/australia-bushfires-devastating-flames-released-more-than-twice-the-amount-of-co2-than-previously-thought-says-study-12408680.

[8] Russel Hope. "Cape Town: Hundreds of firefighters battling blaze in South African city". In: *Sky News* (Dec. 2023). URL: https://news.sky.com/story/cape-town-hundreds-of-firefighters-battling-blaze-in-south-african-city-13034680.

[9] Oliver Milman. "After a record year of wildfires, will Canada ever be the same again?" In: *The Guardian* (Nov. 2023). URL: https://www.theguardian.com/world/2023/nov/09/canada-wildfire-record-climate-crisis.

[10] Alireza Shamsoshoara et al. "Aerial imagery pile burn detection using deep learning: The FLAME dataset". In: *Computer Networks* 193 (2021), p. 108001. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2021.108001. URL: https://www.sciencedirect.com/science/article/pii/S1389128621001201.

[11] Alireza Shamsoshoara et al. *The FLAME dataset: Aerial Imagery Pile burn detection using drones (UAVs)*. 2020. DOI: 10.21227/qad6-r683. URL: https://dx.doi.org/10.21227/qad6-r683.

[12] National Geographic Society. *The Ecological Benefits of Fire*. Accessed on December 20, 2023. 2023. URL: https://education.nationalgeographic.org/resource/ecological-benefits-fire/.

[13] Christian Szegedy et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 31.1 (Feb. 2017). DOI: 10.1609/aaai.v31i1.11231. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11231.

[14] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.