Improving Forest Fire Detection using Deep Learning and Multispectral Imaging

Introduction

Forest fires have severe impacts on the ecology, wildlife, and human lives. These fires can occur naturally through lightning strikes or as a result of human activities such as campfires, cigarettes, fireworks, intentional sabotage, or prescribed fire which can escalate. Prescribed fire is a controlled fire intentionally set by trained professionals to manage the vegetation in a specific area (Bond & Keane, 2017). Once a fire starts, it can quickly spread throughout the forest, burning everything in its path, including trees, bushes, grasses, and wildlife habitats.

The impact of forest fires on human life cannot be overemphasized as they can lead to loss of property, displacement, and even loss of life. In 2019, the Australian bushfires caused the death of 33 people (Ogie et al., 2022) and an estimated 3 billion animals (Vernick, 2020), including threatened and endangered species such as Koalas, turtles, and gliders (Morton, 2020). Forest fires can also affect the economy by destroying infrastructure, disrupting tourism, and causing long-term damage to the environment.

Livestock is also severely affected by forest fires. Grazing animals like cattle, sheep, and goats rely on vegetation to survive, and forest fires destroy grazing land and water sources (Tekalign & Kebede, 2016), causing animals to die from starvation and dehydration.

Forestation is a critical component of the ecosystem as forests provide habitats for a variety of wildlife, purify air and water, and regulate the climate. Forest fires cause deforestation, leading to the loss of these vital services. Once a forest is destroyed by fire, it takes decades, if not centuries, for it to regenerate fully. This loss of vegetation also affects the water cycle, leading to soil erosion, flooding, and decreased water quality. In addition to the immediate effects of forest fires, they also contribute to long-term environmental issues responsible for carbon dioxide (CO2) emissions, which contribute to global warming. In 2021, wildfires contributed to 1.76 billion tonnes of CO2 emissions (World Economic Forum, 2021).

To mitigate the negative effects of forest fires, there is a need to improve fire detection and response mechanisms. One promising approach is to use deep learning algorithms, which this

project intends to implement using multispectral imaging. This will enable real-time monitoring of forests and can detect fires at their earliest stages, enabling faster response times. This can help reduce the number of human lives, livestock, and property lost to forest fires.

Background

Over time, the methods for detecting forest fires have evolved from traditional methods based on human observation and basic sensors to more advanced systems that utilize remote sensing, data processing, and artificial intelligence techniques. Early methods of detecting forest fires relied on visual detection by humans, which was often limited by factors such as weather and visibility (Alkhatib, 2014). Basic sensors such as thermal detectors were also used, but they had limitations in terms of accuracy and sensitivity (San-Miguel-Ayanz et al., 2005). However, with advancements in technology, remote sensing techniques have become more widely used for forest fire detection. This includes the use of satellite imagery, aerial surveys, and ground-based sensors (Thapa, 2021).

Data processing techniques in conjunction with machine learning techniques have also been employed to analyze the information gathered by these sensors, allowing for more accurate and timely detection of forest fires. This is even effective in analyzing large amounts of data and identifying patterns that may not be apparent to humans. The evolution of forest fire detection methods highlights the importance of using advanced technologies to combat the increasing threat of forest fires.

Zhang et al. (2016) proposed a novel deep learning-based approach for detecting forest fires. They created a deep convolutional neural network (CNN) that combined a full image classifier with a fine-grained patch fire classifier. The fire detection process was carried out in a pipelined manner, with the global image-level classifier applied to the entire image first, followed by the fine-grained patch classifier to pinpoint the precise location of fire patches. The patch-level detector achieved an impressive 97% and 90% detection accuracy on the training and testing datasets, respectively. The authors created a benchmark for fire detection with patch-level annotations, which they believe is the first of its kind, to aid in the evaluation of different fire detection methods in the research community.

Jiao et al. (2019) highlighted the potential of using unmanned aerial vehicles (UAVs) for forest fire detection and monitoring due to their high mobility and cost-effectiveness. To improve the speed and accuracy of traditional fire detection algorithms based on the RGB color model, the authors proposed a forest fire detection algorithm that utilizes YOLOv3 on UAV-based aerial images. They developed a UAV platform for forest fire detection and implemented a small-scale convolution neural network (CNN) with the help of YOLOv3, based on the available computation power of the onboard hardware. The results show that this algorithm achieved a recognition rate of about 83% and a frame rate of detection exceeding 3.2 frames per second (fps). This method offers significant advantages for real-time forest fire detection applications using UAVs.

Singh et al. (2013) proposed a fusion information process for forest fire detection in wireless sensor networks that takes into account data from multiple sources before making a final decision. To improve detection accuracy, they used two algorithms based on the threshold ratio method and Dempster-Shafer theory, as well as support vector machine (SVM) classification and logistic regression. However, the limited amount of energy required for data processing, the short communication range, the limited computations, the complexity of ML algorithms when executed on sensor nodes, and the difficulty of being distributed on every sensor node limit their implementation of machine learning techniques for forest fire detection systems in wireless sensor networks.

Vipin (2012) proposed an approach for forest fire detection using image processing techniques. The proposed algorithm utilizes a rule-based color model for fire pixel classification based on the RGB and YCbCr color space. The advantage of using YCbCr color space is that it can effectively separate the luminance from the chrominance, making the fire pixel classification more accurate. The algorithm was tested on two sets of images, one containing fire and the other containing fire-like regions, and it demonstrated higher detection rates and lower false alarm rates compared to standard methods. However, the proposed method has limitations in terms of detecting fires in heavily smoke-obscured areas, as well as in detecting small or low-intensity fires.

Various methods for forest fire detection have been explored, including Fire Watch Towers, Wireless Sensor Networks, and Satellite and Aerial Monitoring. Fire Watch Towers involve human observation, but this can be limited by operator fatigue, time of day, time of year, and geographic location. Wireless Sensor Networks have coverage limitations and battery charge issues. Satellite and Aerial Monitoring can cover large areas, but the resolution of satellite imagery is low, and the

systems are expensive. Additionally, weather conditions like clouds can decrease the accuracy of satellite-based forest fire detection.

Objectives

The primary objective of this project is to develop a more effective method for detecting forest fires by utilizing the potential of multispectral imaging and deep learning techniques. To achieve this project's objectives, the model will be trained using:

- A convolutional neural network as base model
- Xception Model via transfer learning
- Experiment with subsampling for imbalance class distribution

The model will be trained using multispectral images that gather data across multiple wavelengths, which provides more comprehensive information about a scene compared to conventional RGB images. By employing this sophisticated methodology, the project aspires to attain better precision and quicker response times in identifying forest fires, thereby curtailing the damage caused by these calamities.
Overall, this project seeks to provide a more reliable, efficient, and accurate system for detecting forest fires, which will contribute towards mitigating the hazardous effects of these incidents.

Methodology

The methodology of this project involved various steps, including data collection, preprocessing, data exploration, model architecture, and evaluation.

Data Collection

The data for this project was collected from Kaggle and consists of two classes: fire and smoke. The initial distribution had a severe class imbalance with 12631 instances of smoke and only 1102 instances of fire, resulting in a total of 13733 instances.
To address this issue, the majority class (smoke) was subsampled to match the number of instances in the minority class (fire) for the sake of the second phase of the experimentation which will involve retraining with a balanced class distribution. While this helped to balance the classes, it is important to note the negative impact of such class disparity on the model's performance

(Zheng & Jin, 2020). With fewer instances of the minority class available for training, the model may struggle to learn the distinguishing features of the fire class, leading to lower accuracy and higher false negatives. The variance of the model may also be affected, as it may become over-reliant on the majority class, leading to poor generalization to unseen data. Therefore, careful consideration must be given to the class distribution during the preprocessing stage to avoid these negative impacts.

Figure 1 visually shows the variance between the classes.

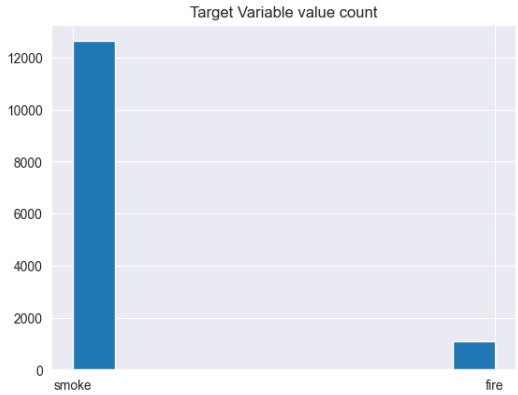

Figure 1: Class distribution

Preprocessing

The preprocessing phase was carried out in two phases; the initial phase involved setting the image height and width which made up the image size and identifying the image channel which when added to the image size, made up the complete image shape.

The images were resized to a height of 200 pixels and a width of 400 pixels with 3 channels. A batch size of 80 was chosen for the training set, and the images were split into a training set, a testing set, and a validation set.

The training set was set to 90% of the data, while the testing set and the validation set were set to 5% each. The number of samples in the testing set was used to determine an appropriate batch size for the testing generator.

To preprocess the images, the ImageDataGenerator class from the Keras library, which was used to perform data augmentation by standardizing the image to scale the pixel values between 0 and 1 and flipping them.

The second phase of the preprocessing involved subsampling the dataset to overcome the imbalance distribution by randomly sampling the majority class to match the number of samples in the minority class. The resulting data had 1102 samples for the fire class and 1102 samples for the smoke class, thus addressing the disparity between both classes.

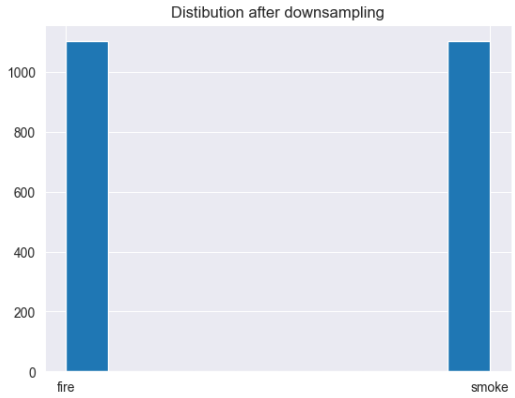

Figure 2: Class distribution after down-sampling

Data Exploration

To gain insight into the dataset and understand the images it contains, exploratory data analysis was conducted by creating a plot to display sample images from both classes. This plot provided a visual representation of the dataset and allowed for an initial assessment of the quality and diversity of the images.

In order to generate the plot, a sample batch of images and labels was obtained from the generator using the next method. The batch was then looped through using a for loop with an index to plot each image and its corresponding class name. The standardization done earlier ensures that the pixel values of the images are in a standardized range, which is necessary for accurate analysis and classification.

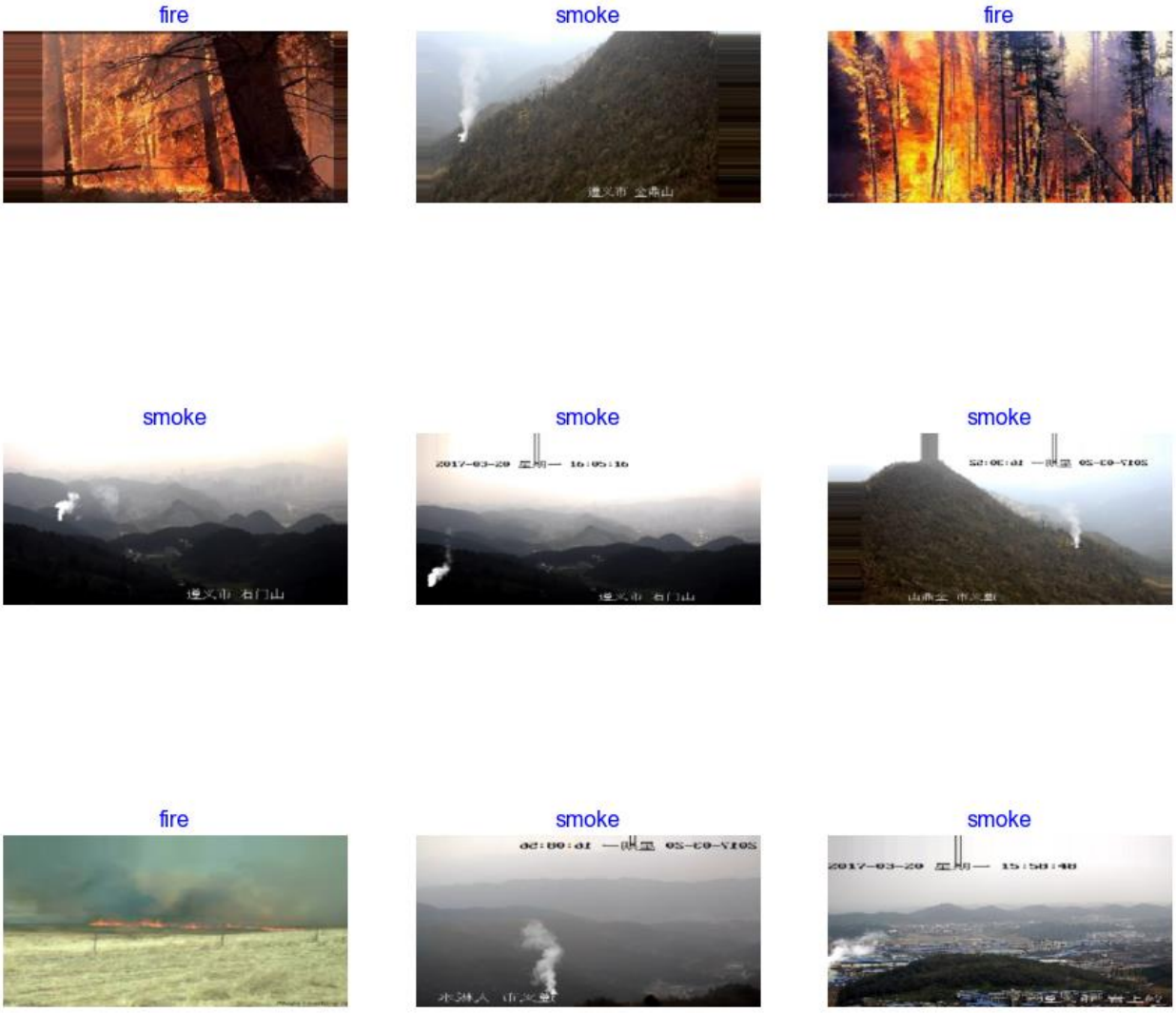

Figure 3: Sample images from image classes

Experiments

The experimentation involved building two models for each phase of preprocessing done (i.e. with the original class distribution and with the subsampled class to fix class imbalance).
The first model was a base model that utilized convolutional neural networks, while the second model was a transfer learning model built using Keras' Xception model. The architecture of the two models differed, with the former utilizing the Adam optimizer while the latter used the Adamax optimizer for the sake of experimentation.

The CNN model was built using a deep architecture consisting of multiple Conv2D layers and MaxPooling2D layers. Each Conv2D layer had a set of learnable filters applied to the input image

to extract features. The architecture started with a Conv2D layer with 32 filters of size 3x3, followed by a MaxPooling2D layer with a 2x2 pool size. This was followed by a Conv2D layer with 64 filters of size 3x3, and another MaxPooling2D layer with a 2x2 pool size. The model then had a Conv2D layer with 128 filters of size 3x3, followed by another MaxPooling2D layer with a 2x2 pool size.

The resulting output was then flattened into a vector and passed through a dropout layer with a rate of 0.5 to prevent overfitting. This was followed by a dense layer with 256 units and ReLU activation, and a BatchNormalization layer to improve training speed and stability. Another dropout layer with a rate of 0.5 was added before the final dense layer with a sigmoid activation function, which had a single unit for binary classification.

The pseudocode is illustrated as

- Define the model
    - Conv2D layer with 32 filters of size 3x3 and ReLU activation, with input shape defined by image shape
    - MaxPooling2D layer with 2x2 pool size
    - Conv2D layer with 64 filters of size 3x3 and ReLU activation
    - MaxPooling2D layer with 2x2 pool size
    - Conv2D layer with 128 filters of size 3x3 and ReLU activation
    - MaxPooling2D layer with 2x2 pool size
    - Flatten the output into a vector
    - Dropout layer with a rate of 0.5 to prevent overfitting
    - Dense layer with 256 units and ReLU activation
    - BatchNormalization layer to improve training speed and stability
    - Dropout layer with a rate of 0.5
    - Output dense layer with sigmoid activation for binary classification

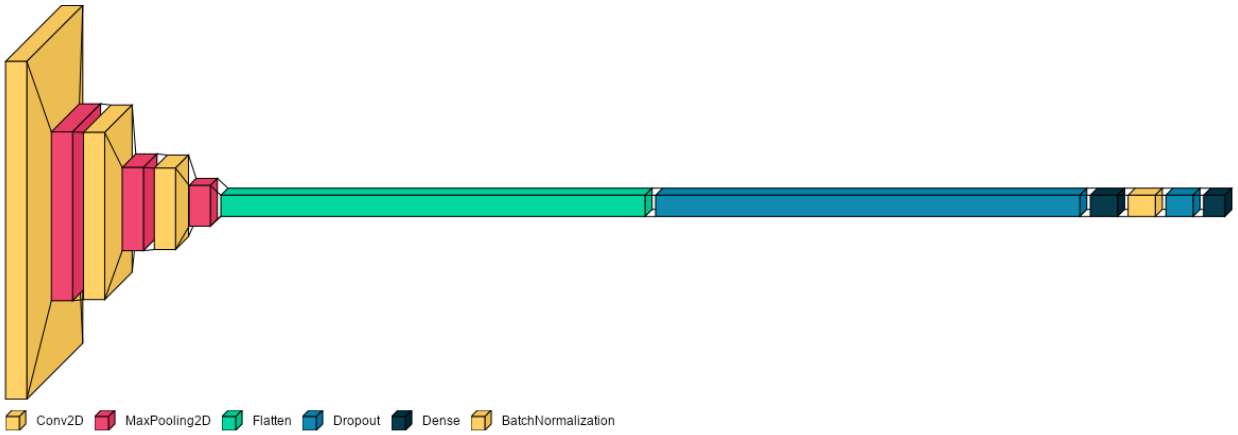

Figure 4: Base model architecture visualization

- Compile the model
  - Use Adam optimizer
  - Use binary cross-entropy loss
  - Monitor accuracy during training

Transfer learning with Xception involves utilizing a pre-trained neural network that was developed by Google and is commonly used for image classification tasks (Tsang, 2018). The Xception model was trained on a massive amount of image data and has learned to identify features from various image classes, including those that are relevant to the detection of fires and smoke.

In the implementation of the Xception model, the pre-trained layers were frozen to prevent them from being updated during training. A new classifier was then layered on top of the previously trained Xception base, allowing for the network to learn new features specific to the task at hand while still retaining the previously learned features.

To obtain the output of the Xception model, the output from the last convolutional layer was first passed through a GlobalAveragePooling2D layer, which averages the output across spatial dimensions (Brownlee, 2019). This was followed by two dense layers with Rectified Linear Unit (ReLU) activation, which helps the network learn non-linear relationships between features. Finally, a sigmoid activation function was applied to the output layer, which provided a binary classification of either fire or smoke for each input image.

The pseudocode is illustrated as

- Load pre-trained Xception model
  - Specify ImageNet weights and input shape
  - Do not include fully connected layer
  - Freeze base model layers

- Add custom classification layers
  - Define input layer with matching shape
  - Pass input through base model
    - Apply global average pooling to convert features to a single vector
  - Add a dense layer with ReLU activation
  - Output dense layer with sigmoid activation for binary classification

- Compile the model
  - Use Adamax optimizer
  - Use binary cross-entropy loss
  - Monitor accuracy during training

Results

To evaluate the models' performance comprehensively, various metrics such as accuracy, f1 score, precision, and recall, along with the confusion matrix were employed. This approach was necessary to capture misclassifications in each class, which could not be identified by accuracy alone.

Remarkably, the base model achieved an accuracy of 99% and the Xception model achieved a remarkable accuracy of 100% both without and with subsampled data. This surprisingly shows there’s no difference between the two experiments, although the confusion matrix shows the difference.

The proximity in performance between the base model and the Xception model could be attributed to the use of multiple regularization techniques such as batch normalization and dropout, which enhance the models' robustness to noise and overfitting.

Table 1: Models performance on each class for data without subsampling

Original Experiment					
Model	**Class**	**Precision**	**Recall**	**F1 Score**	**Accuracy**
Base Model	Fire	1	0.88	0.94	0.99
	Smoke	0.99	1	0.99	
Xception Model	Fire	1	0.98	0.99	1.0
	Smoke	1	1	1	

Table 2: Models performance on each class for subsampled data

Subsampled Experiment					
Model	**Class**	**Precision**	**Recall**	**F1 Score**	**Accuracy**
Base Model	Fire	0.97	0.97	0.97	0.99
	Smoke	1.0	1.0	1.0	
Xception Model	Fire	1.0	1.0	1.0	1.0
	Smoke	1.0	1.0	1.0	

Table 1 and Table 2 shows the model's performance on the test sets for each individual class for both models for each phase of the experiment (without and with subsampled data).

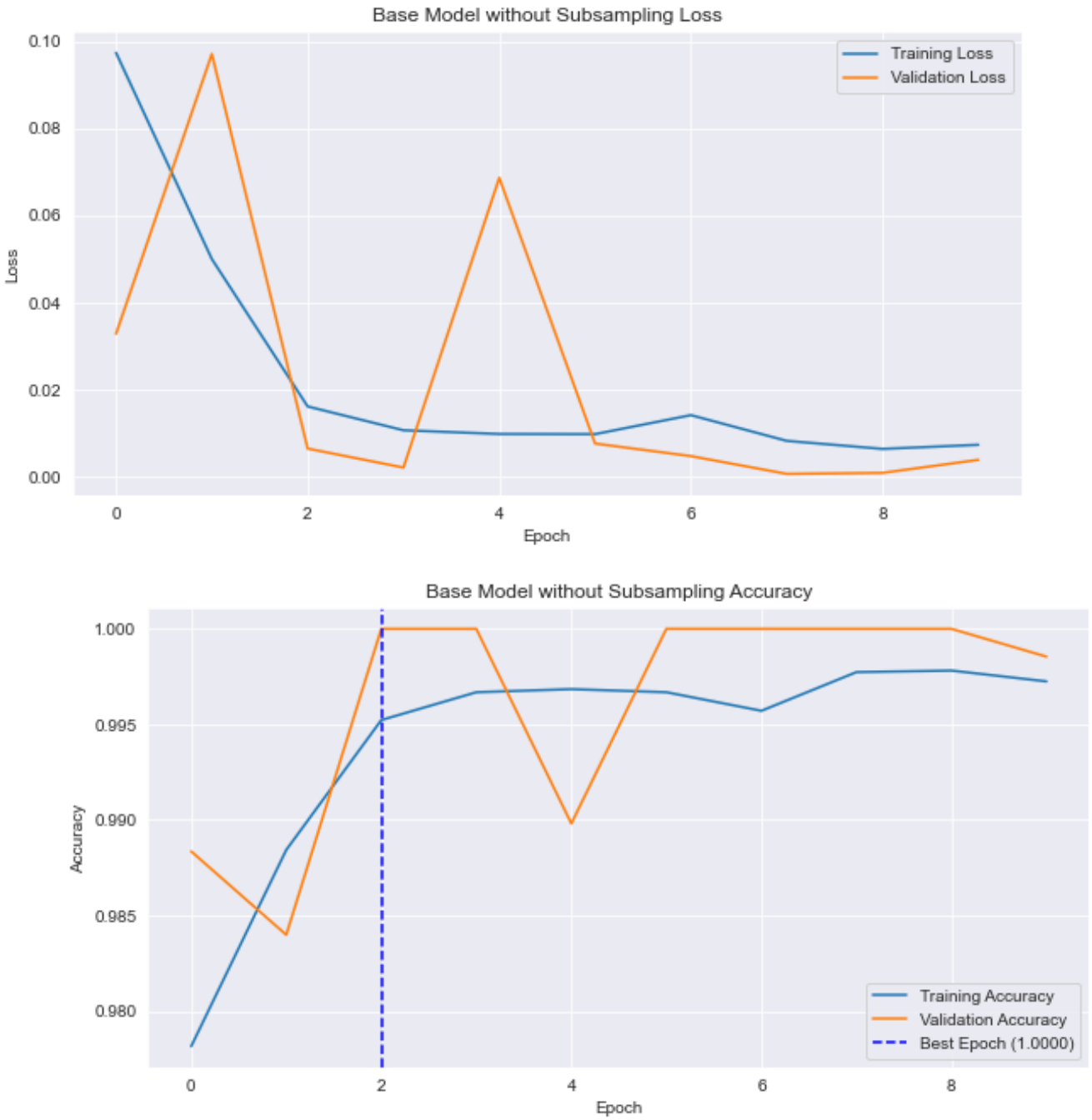

Figure 5: Base model training performance without subsampling

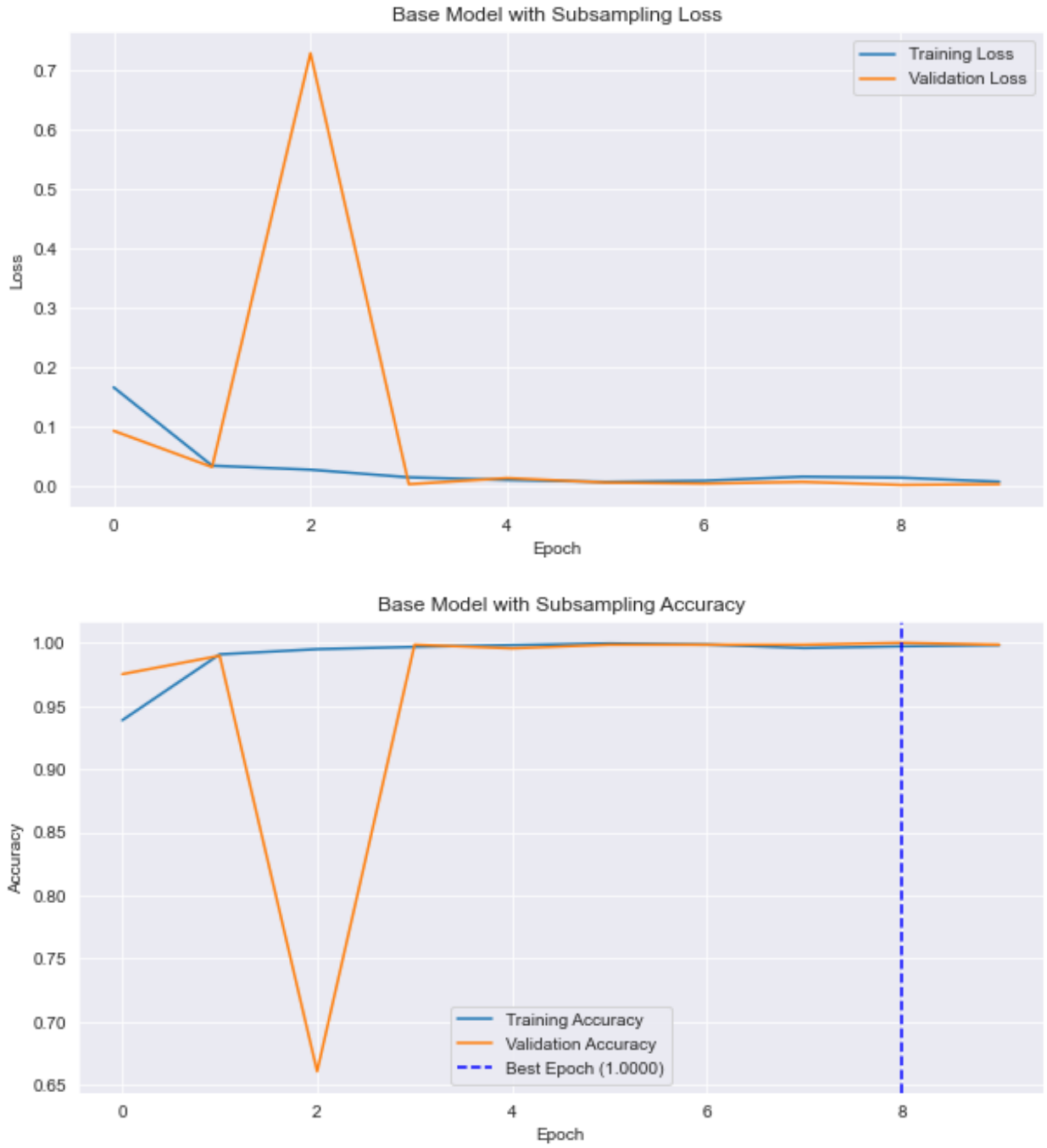

Figure 6: Base model training performance with subsampling

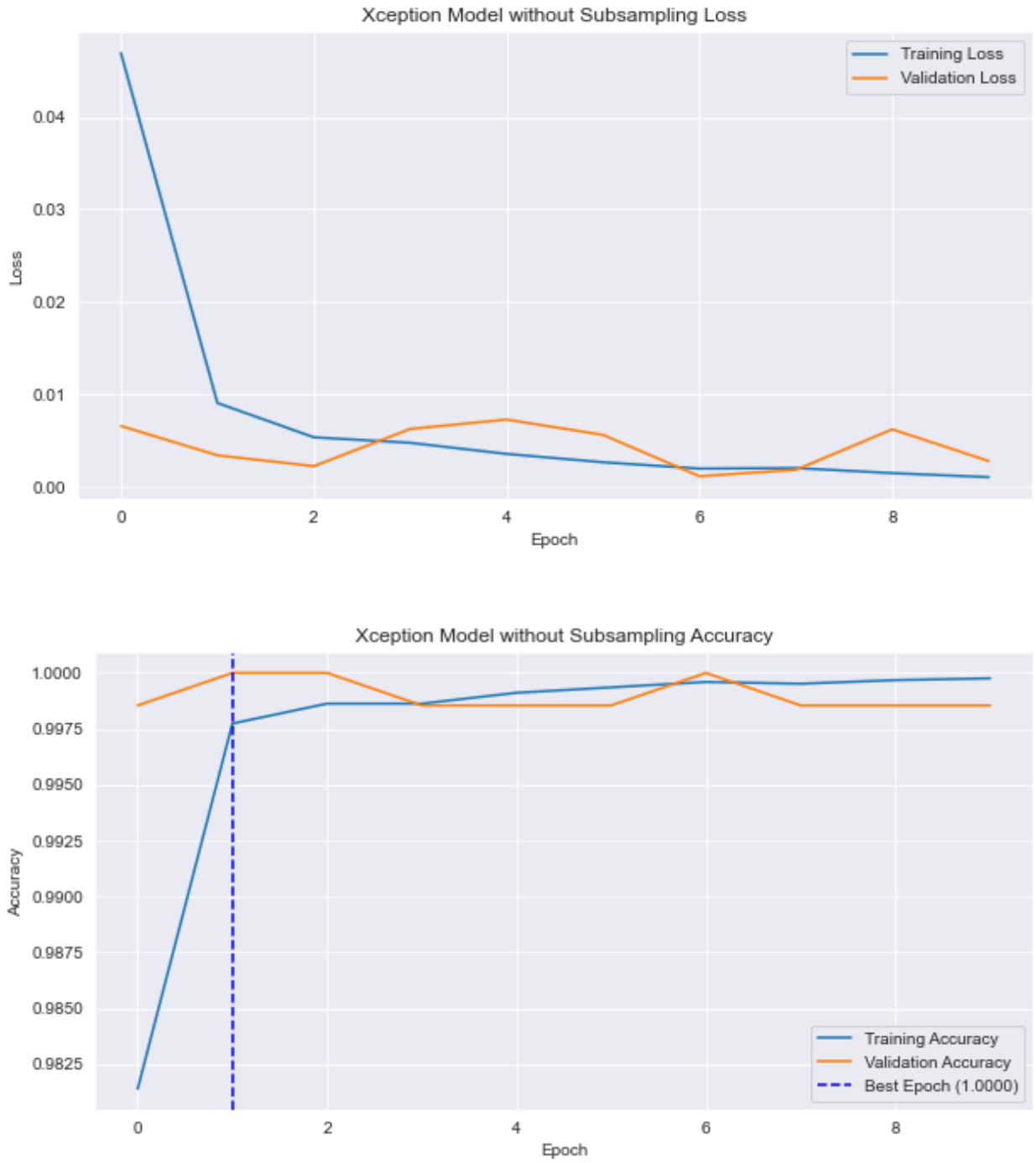

Figure 7: Xception model training performance without subsampling

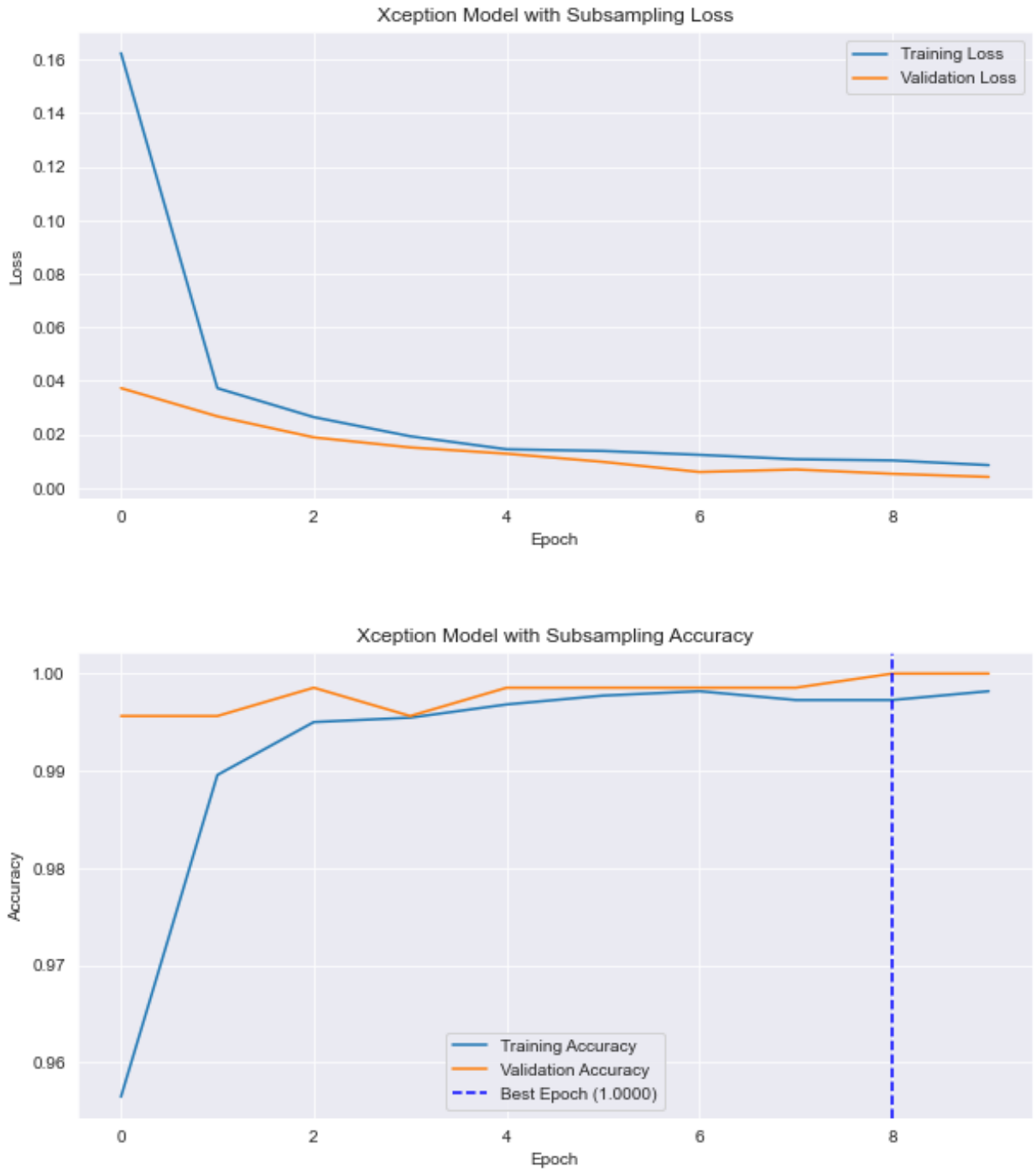

Figure 8: Xception model training performance with subsampling

Although the test set performance of both models was comparable, the difference in their training performance was more evident. Figure 5 shows that the base model trained on the data without subsampling achieved the best performance early on before dropping and regaining the optimal accuracy after the fifth epoch. In contrast, figure 6 shows the subsampled data allowed the model to reach optimal accuracy by the eighth epoch, with less fluctuation than the initial model.

Likewise, the Xception model demonstrated better training performance. The only difference between the two experiments (with and without subsampling) was the speed at which the best accuracy was reached: it occurred at the first epoch with class imbalance (figure 7) and at the eighth epoch with the balanced classes (figure 8).

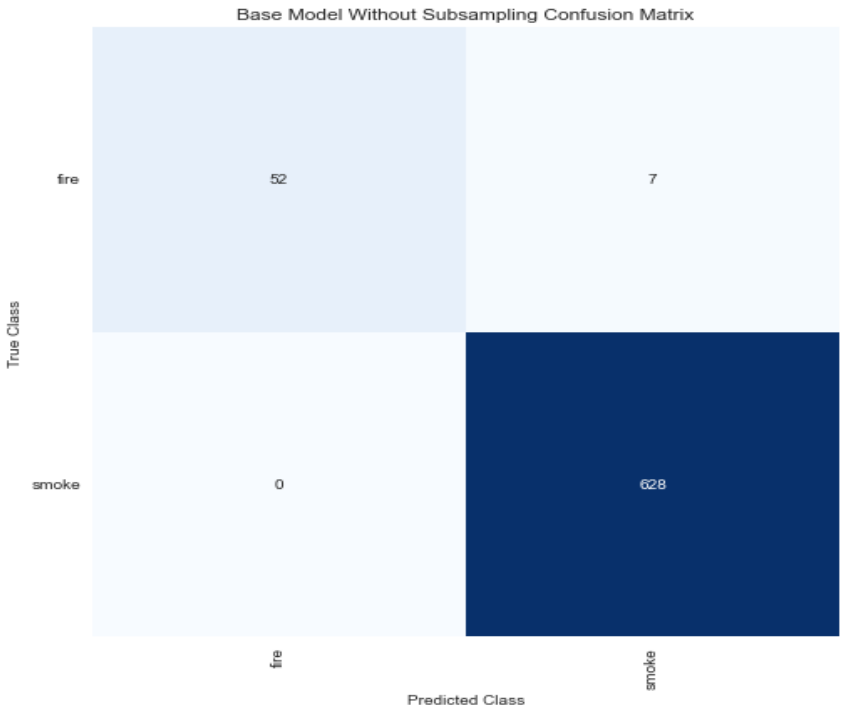

Figure 9: Base model without subsampling confusion matrix

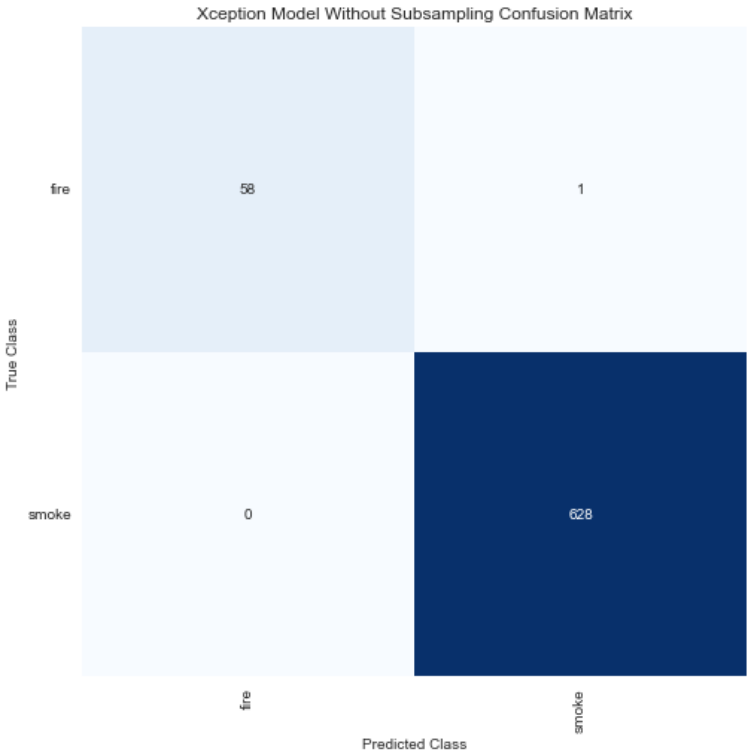

Figure 10: Xception model without subsampling confusion matrix

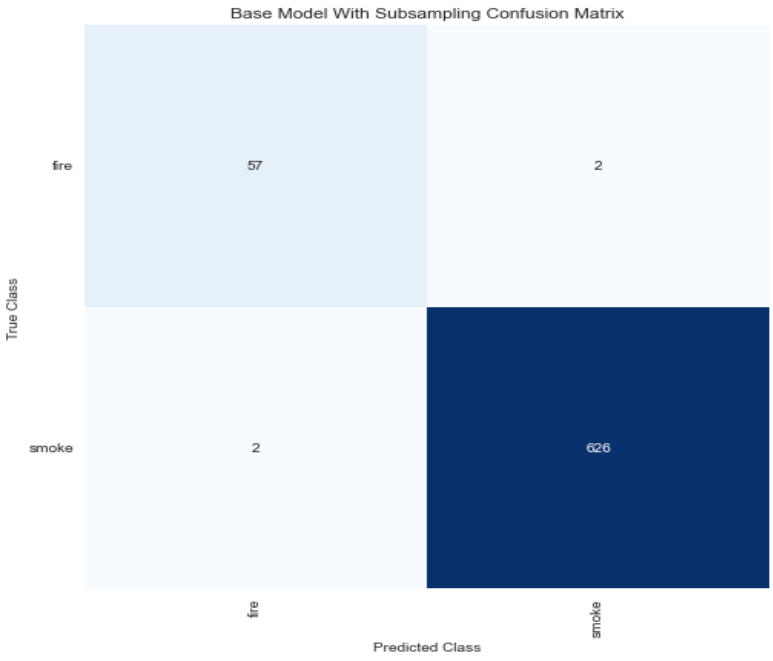

Figure 11: Base model with subsampling confusion matrix

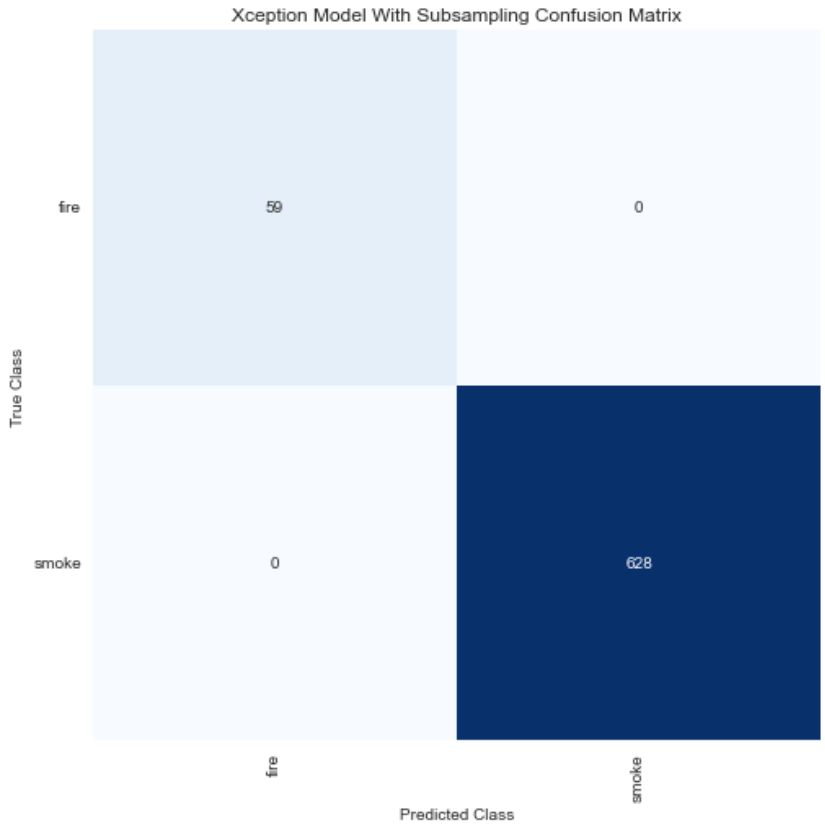

Figure 12: Xception model with subsampling confusion matrix

The classifications of both models for the initial experiment with the imbalance dataset are presented in Figure 9 and 10. In this case, the base model misclassified seven fire images as smoke. However, the Xception model only misclassified one fire image as smoke. Both models correctly classified all smoke images.

Figure 11 shows the matrix for the base model with subsampled data. It misclassified only 2 fire and smoke images each. On the other hand, Figure 12 showcase the outstanding performance of the Xception model when trained on a balanced dataset, classifying all images in their respective classes, with no errors observed.

Conclusion

Based on the results obtained, it can be concluded both models demonstrated exceptional performance in accurately classifying images and distinguishing between the two classes with high precision and recall. The approach taken in this study for early detection of fires in forests was effective, as reflected by the overall accuracy and F1 score of nearly 100%. These results are promising and hold important practical applications for real-world use.

The deployment perspective for early fire detection in forests is particularly noteworthy. By integrating the model into a monitoring system, forest fires can be detected early (in the smoke phase), and relevant authorities can be alerted for immediate action. The model's ability to detect fire and smoke at an early stage could reduce the risk of extensive damage to the forest and surrounding areas, as well as potentially save lives. The deployment of this model in real-world scenarios could revolutionize the way we approach forest fire detection and mitigation.

References

Alkhatib, A. A. (2014). A Review on Forest Fire Detection Techniques. *International Journal of Distributed Sensor Networks*, *10*(3). 10.1155/2014/5973
[Assessed 8/2/2023]

Arpit, D., Zhou, Y., & Govindaraju, V. (2016, June). *Normalization Propagation: A Parametric Technique for Removing Internal Covariate Shift in Deep Networks*. Proceedings of Machine Learning Research.
Available online: https://proceedings.mlr.press/v48/arpitb16.html
[Assessed 17/2/2023]

Atto, A. M., Pastor, D., & Mercier, G. (2019, May 22). *Smooth sigmoid wavelet shrinkage for non-parametric estimation - ICASSP 2008: IEEE international conference on acoustic*. HAL.
Available online: https://hal.science/hal-02136546/file/ICASSP_ATTO_2008.pdf
[Assessed 21/3/2023]

Bond, W. J., & Keane, R. E. (2017). Fires, ecological effects of. *Reference module in life sciences*, 1-11.
[Assessed 14/3/2023]

Brownlee, J. (2019, April 22). *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks - MachineLearningMastery.com*. Machine Learning Mastery.
[Assessed 11/3/2023]

Jiao, Z., Zhang, Y., Xin, J., Mu, L., Yi, Y., Liu, H., & Liu, D. (2019). *A Deep Learning Based Forest Fire Detection Approach Using UAV and YOLOv3*. IEEE. 10.1109/ICIAI.2019.8850815
[Assessed 16/2/2023]

Llugsi, R., Yacoubi, S. E., Fontaine, A., & Lupera, P. (2021). *Comparison between Adam, AdaMax and Adam W optimizers to implement a Weather Forecast based on Neural Networks for the Andean city of Quito*. IEEE Fifth Ecuador Technical Chapters Meeting (ETCM).
[Assessed 16/3/2023]

Morton, A. (2020, January 19). *More than 100 threatened species hit hard by Australian bushfires, pushing many towards extinction*. The Guardian.
[Assessed 11/3/2023]

Ogie, R., Moore, A., Wickramasuriya, R., Amirghasemi, M., James, S., & Dilworth, T. (2022). Twitter data from the 2019–20 Australian bushfires reveals participatory and temporal variations in social media use for disaster recovery. *Scientific Reports*, *12*(1). 10.1038/s41598-022-21265-6
[Assessed 21/3/2023]

San-Miguel-Ayanz, J., Ravail, N., Kelha, V., & Oller, A. (2005). Active Fire Detection for Fire Emergency Management: Potential and Limitations for the Operational Use of Remote Sensing. *Natural Hazards*, *35*, 361–376. 10.1007/s11069-004-1797-2
[Assessed 11/3/2023]

Singh, Y., Saha, S., Chugh, U., & Gupta, C. (2013). *Distributed Event Detection in Wireless Sensor Networks for Forest Fires*. IEEE. 10.1109/UKSim.2013.133
[Assessed 25/3/2023]

Tekalign, W., & Kebede, Y. (2016). Impacts of Wildfire and Prescribed Fire on Wildlife and Habitats: A Review. *Journal of Natural Sciences Research*.
[Assessed 13/4/2023]

Thapa, P. (2021). *Forest Fire Area Detection using Satellite Remote Sensing Technologies*. ResearchGate.
[Assessed 11/3/2023]

Tsang, S.-H. (2018, September 25). Review: Xception — With Depthwise Separable Convolution, Better Than Inception-v3 (Image…. Towards Data Science.
Available online: https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568
[Assessed 2/3/2023]

Vernick, D. (2020, July 28). *3 billion animals harmed by Australia's fires | Stories*. WWF.
Available online: https://www.worldwildlife.org/stories/3-billion-animals-harmed-by-australia-s-fires
[Assessed 23/3/2023]

Vipin, V. (2012). *Image Processing Based Forest Fire Detection* (Vol. 2). International Journal of Emerging Technology and Advanced Engineering.

[Assessed 12/3/2023]

World Economic Forum. (2021, December 10). *How much carbon dioxide do wildfires emit?* The World Economic Forum.
Available online: https://www.weforum.org/agenda/2021/12/siberia-america-wildfires-emissions-records-2021/
[Assessed 21/3/2023]

Wu, X. X., & LIU, J. G. (2009). *A new early stopping algorithm for improving neural network generalization* (Vol. 1). IEEE.
[Assessed 19/4/2023]

Zhang, Q., Xu, J., Xu, L., & Guo, H. (2016). *Deep Convolutional Neural Networks for Forest Fire Detection*. Atlantis Press. 10.2991/ifmeita-16.2016.105
[Assessed 21/3/2023]

Zheng, W., & Jin, M. (2020). The Effects of Class Imbalance and Training Data Size on Classifier Learning: An Empirical Study. *SN COMPUT. SCI*, *1*(71). 10.1007/s42979-020-0074-0
[Assessed 2/3/2023]