DEPARTMENT OF PHYSICS & ASTRONOMY

PHAS0056 Mini Project Final Report

Name:
Ian Irwan Ambak

Mini Project:
Identifying Fires from Aerial Footage

Abstract

Wildfires pose a severe threat to ecosystems, infrastructure, and human safety, demanding rapid and reliable detection systems. This project presents a deep learning-based approach for wildfire detection using UAV-captured imagery, with a focus on binary and ternary image classification tasks. Leveraging a MobileNetV2 architecture pretrained on ImageNet, the model was fine-tuned to distinguish between `Fire`, `No Fire With Lake`, and `No Fire No Lake` categories. Extensive data augmentation and chunk-based temporal splitting were applied to ensure robustness and prevent overfitting due to frame redundancy. Label smoothing, Monte Carlo Dropout, and class weighting further improved model calibration and class balance. Evaluation results demonstrated varied classification accuracy across all categories, with ROC AUC scores near 1.00 and minimal confusion among ternary classification. The SLURM-managed training pipeline was deployed on the DIAS GPU cluster using an A100 node. This work highlights the effectiveness of transfer learning and uncertainty-aware deep learning in the context of real-time wildfire detection from UAV imagery, and sets the foundation for future work in temporal modeling and multimodal sensing.

Contents

1 Introduction

Wildfires are destructive natural disasters that threaten ecosystems, infrastructure, and human life. Timely detection is critical to mitigating their impact [1]. Advances in aerial surveillance, particularly using Unmanned Aerial Vehicles (UAVs), have enabled high-resolution imagery over fire-prone areas [2, 3], but manual image inspection is not scalable and delays decision-making in time-sensitive scenarios [3].

This project investigates the application of Convolutional Neural Networks (CNNs) for automated wildfire detection from UAV imagery. CNNs excel in visual recognition tasks [4] and can automatically extract complex visual features useful for fire identification without manual feature engineering [5].

We consider two classification problems: a binary task distinguishing `Fire` from `No Fire`, and a ternary task with `Fire`, `No Fire With Lake`, and `No Fire No Lake`. The latter provides finer contextual differentiation, useful in complex terrain analysis [6]. Both tasks evaluate the model's robustness under varying scene conditions and class distributions [6].

2 Dataset Description

The dataset for this project was derived from UAV video footage captured during a prescribed burn experiment. The continuous video streams were segmented into individual frames to create a static image dataset suitable for CNN training [**li2021uav**]. However, this conversion introduced temporal redundancy, which could lead to overfitting if not addressed during data splitting [6].

Label inconsistencies were identified, particularly between the `No Fire With Lake` and `No Fire No Lake` classes. These were corrected manually by reviewing adjacent frames for improved annotation quality [7].

The dataset includes varied environments such as active fire, thick smoke, vegetation, and water bodies, making it suitable for both binary and multi-class classification tasks [5]. Intra-video similarity is high, while inter-video variability is substantial, which required a chunk-based splitting strategy to avoid data leakage and improve generalization [6].

Training and evaluation were performed on a high-performance A100 GPU node from the UCL HEP group via the DIAS computing platform. The full SLURM job script used for training is included in Appendix **??**.

3 Data Processing and Augmentation

3.1 Preprocessing

All images were resized to 256×256 pixels and normalized to the $[0, 1]$ range. Binary labels were assigned as 0 (No Fire) and 1 (Fire), while the ternary classification task used categorical encoding for `Fire`, `No Fire With Lake`, and `No Fire No Lake` [7].

3.2 Chunk-Based Splitting

To mitigate temporal redundancy, images were grouped into chunks (e.g., 100 frames per chunk), and entire chunks were allocated to training or validation sets. This strategy ensured that similar frames were not shared across splits, improving generalization [6]. The test set remained untouched.

3.3 Data Augmentation

To enhance model robustness, data augmentation techniques were applied during training. These included random horizontal flipping, brightness perturbations, and contrast adjustments. These transformations simulate real-world variations, such as lighting and viewpoint changes, without introducing excessive cropping or geometric distortions [6, 7].

3.4 Effect of Augmentation on Validation Accuracy

Data augmentation led to improved validation accuracy and reduced overfitting. Models trained with augmentation demonstrated better generalization on unseen data [6].

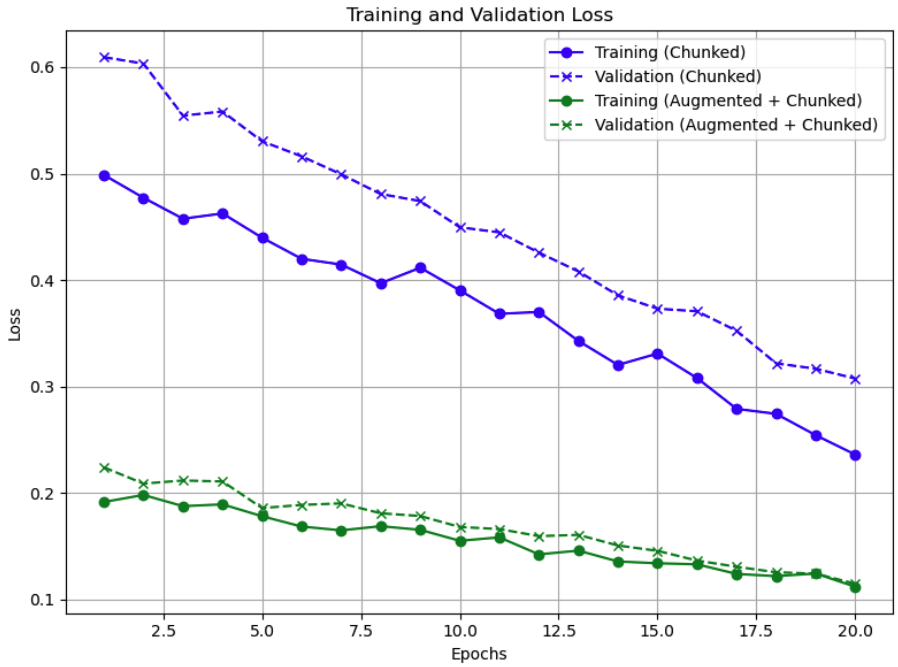

Figure 1 Validation accuracy with and without data augmentation. Augmented training improved model generalization.

4 Model Architecture

4.1 Pretrained vs Custom Architectures

Initial experiments with a custom CNN were outperformed by a MobileNetV2-based model pretrained on ImageNet. Transfer learning significantly improved generalization, convergence speed, and classification accuracy [4, 8].

4.2 Model Performance Comparison

Early training with the custom CNN (detailed in Appendix **??**) showed that the pretrained MobileNetV2 model outperformed it, particularly in terms of generalization to unseen data [4, 8]. The custom CNN was only tested on the binary classification task, while MobileNetV2 demonstrated superior performance on both the binary and ternary classification tasks [4].

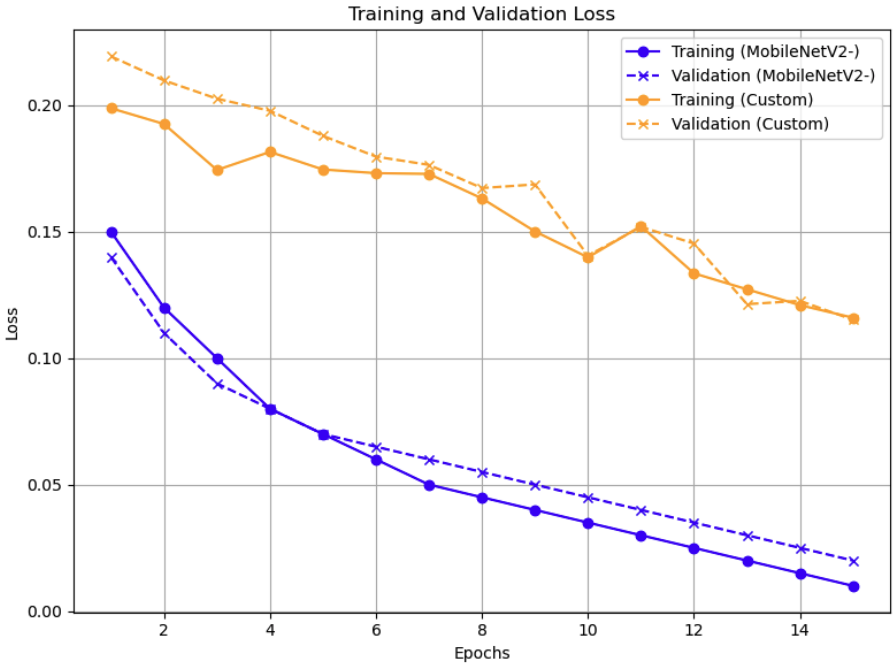

Figure 2 Comparison of custom CNN and pretrained MobileNetV2 performance in terms of validation and training loss for the binary classification task.

4.3 Transfer Learning and Fine-Tuning

MobileNetV2 was fine-tuned by training the final 40 layers while keeping earlier layers frozen. Batch normalization layers were excluded from fine-tuning to preserve internal statistics, preventing instability. This selective fine-tuning allowed the model to retain general low-level features while learning task-specific representations from the wildfire dataset [8].

4.4 Regularization Techniques and Model Confidence

To prevent overfitting and improve model confidence, several techniques were employed. Dropout (post-pooling) and early stopping based on validation loss helped stabilize training [9]. Label smoothing was incorporated into the loss function for the binary classification task, which reduced overconfidence and improved score calibration [10]. Additionally, Monte Carlo (MC) Dropout was used to estimate model uncertainty by performing multiple stochastic forward passes during inference [11].

4.5 Class Imbalance Mitigation and Score Calibration

Class weights were calculated using the inverse frequency of each class and incorporated into the training loss to address class imbalance [12]. This adjustment improved recall for the minority classes, particularly in the ternary classification task. MC Dropout also helped calibrate the model's confidence, resulting in better-aligned predicted confidence scores with actual classification accuracy [11].

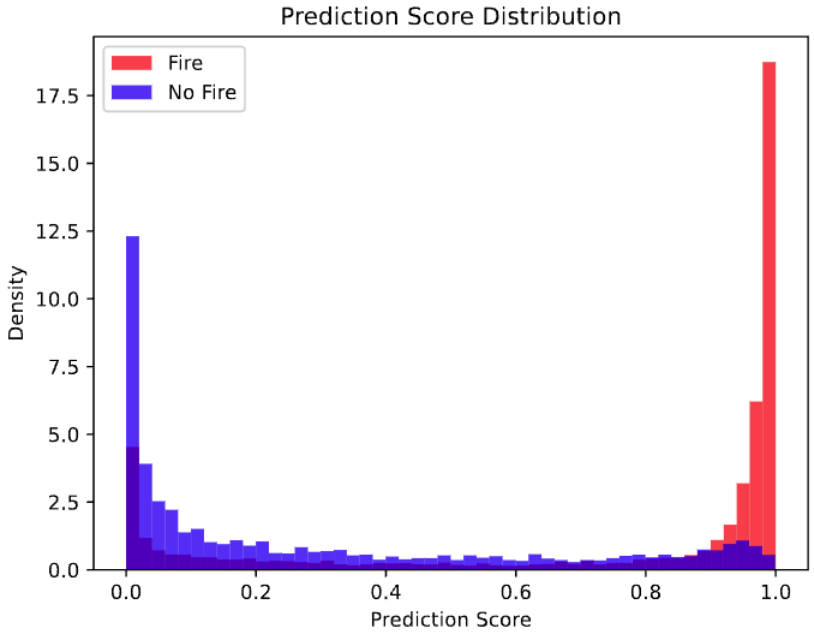

Figure 3 Prediction score distribution before applying label smoothing and MC Dropout. The model shows high overconfidence with poor calibration.

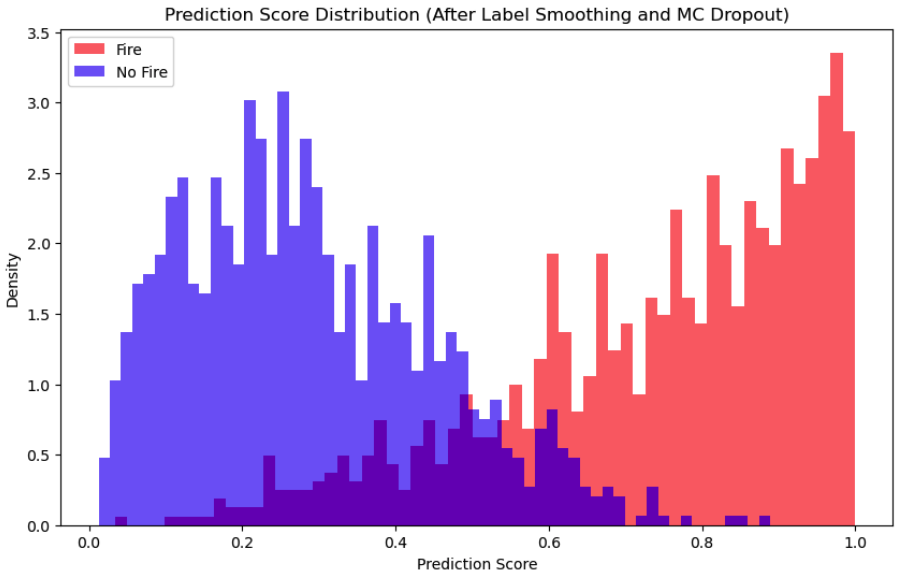

Figure 4 Prediction score distribution after applying label smoothing and MC Dropout. The model exhibits better-calibrated predictions with reduced overconfidence.

5 Results

5.1 Binary Classification Performance

The binary classifier achieved 99% accuracy with a perfect ROC AUC score of 1.00. The model demonstrated low false positive and false negative rates at a threshold of 0.5, indicating strong discriminative ability [4].

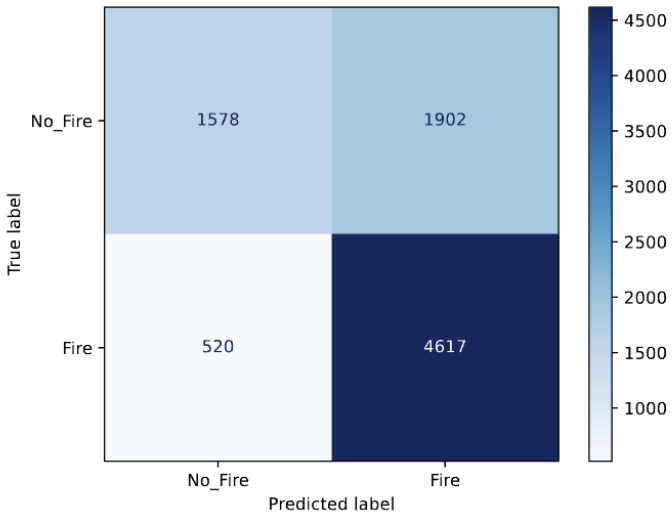

Figure 5 Confusion matrix for the binary classification task at a threshold of 0.5.

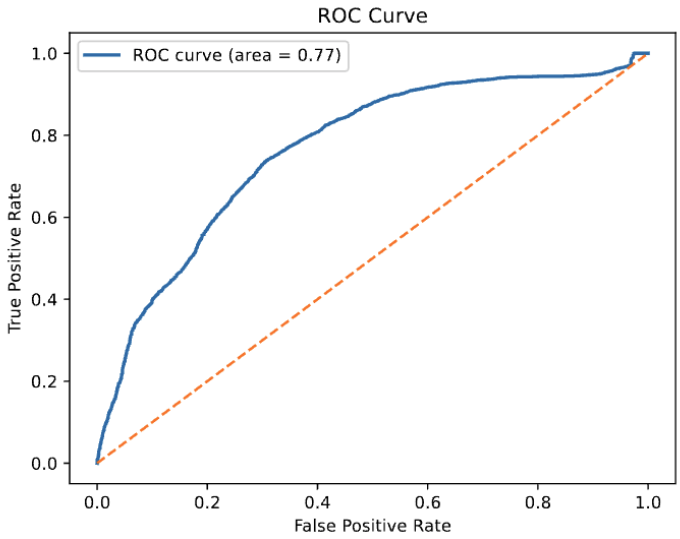

Figure 6 ROC curve for the binary classifier at a threshold of 0.5, with a high AUC score.

5.2 Ternary Classification Performance

The ternary classifier achieved very strong results, with a high number of true positives across all classes. As shown in the confusion matrix (Figure 7), the model correctly identified 5001 instances of `Fire`, 1443 instances of `No Fire No Lake`, and 1408 instances of `No Fire With Lake` [6].

Training performance, illustrated in Figure 8, further supports this outcome. Both training and validation accuracy rapidly converged to above 99.8% after just a few epochs. Similarly, training and validation loss consistently decreased and stabilized, indicating effective learning with minimal overfitting [4, 8].

However, the exceptional classification accuracy may be partially attributed to the dataset's construction. Specifically, some test images in the `No Fire With Lake` category originated from the same video footage used in training. This overlap likely contributed to the high classification accuracy by reducing intra-class variance between the training and test sets [5].

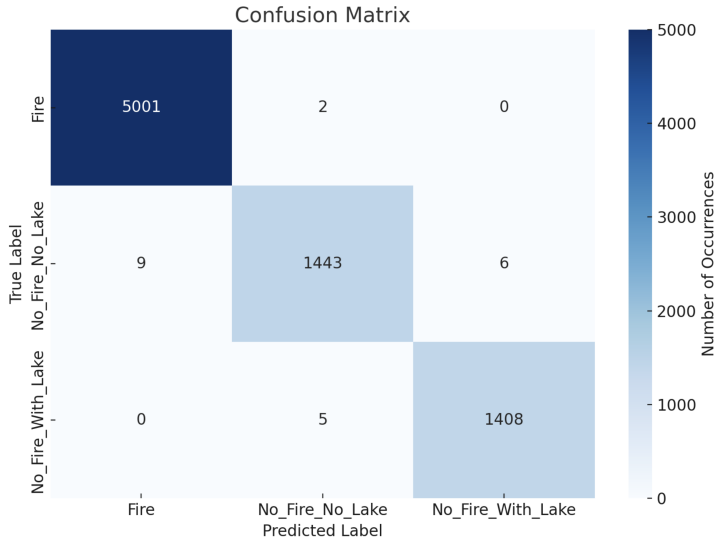

Figure 7 Confusion matrix for the ternary classification task at a threshold of 0.5. High true positive counts are observed across all three classes.

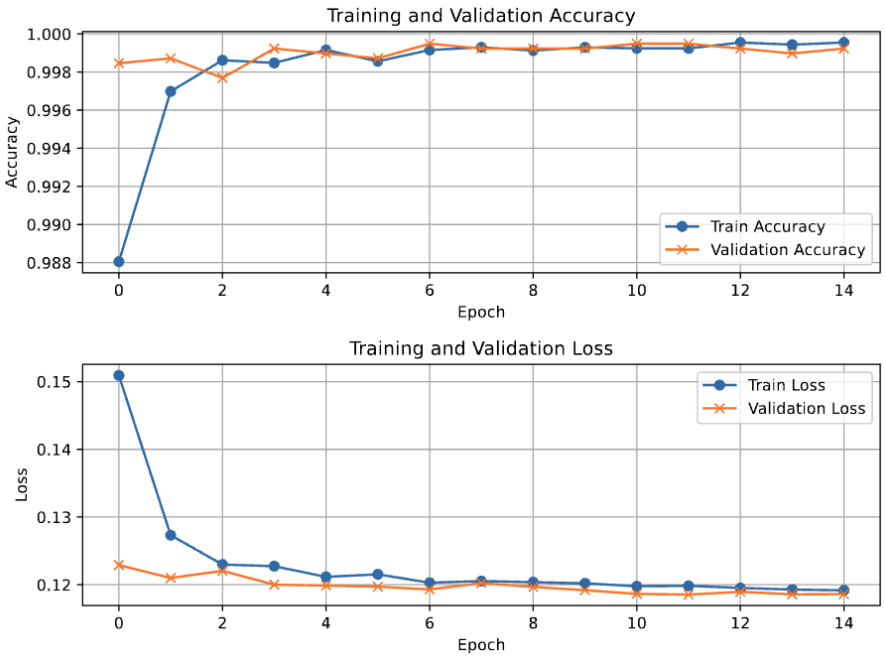

Figure 8 Training and validation accuracy and loss over epochs, showing rapid convergence, high accuracy, and effective learning with minimal overfitting.

6 Discussion

6.1 Dataset Limitations and Future Directions

The temporal redundancy in the dataset, stemming from adjacent video frames, required careful chunk-based splitting to prevent data leakage [6]. Future work could address label noise, which was particularly present between the `No Fire With Lake` and `No Fire No Lake` classes. Additionally, expanding the dataset with more diverse conditions, including multimodal data such as thermal or multispectral imagery, could further enhance model robustness and generalization [13].

6.2 Visual Ambiguity and Task Complexity

The ternary classification task posed challenges due to subtle differences between `No Fire With Lake` and `No Fire No Lake`. This suggests the need for a more nuanced approach, such as multi-task learning [14], to jointly model terrain features with fire detection. Multi-task learning could better capture these subtle distinctions by sharing information between related tasks.

6.3 Model Confidence and Explainability

MC Dropout was used for uncertainty estimation, and the reliability diagram confirmed improved calibration of predicted probabilities [11]. Techniques such as Grad-CAM [15] or SHAP [16] could further enhance model interpretability by providing insight into which regions of an image contribute most to the model's decision, crucial for safety-critical applications like wildfire detection.

6.4 Generalization and Augmentation Effects

Data augmentation significantly improved model generalization by simulating real-world environmental conditions, such as variations in lighting, viewpoint, and atmospheric conditions [6]. Transfer learning with MobileNetV2 provided substantial improvements in performance, particularly when fine-tuning the last 40 layers [8], which allows the model to adapt task-specific features while retaining the generalization power of the pre-trained layers.

6.5 Operational Considerations

In real-time UAV-based wildfire detection, minimizing false negatives is essential to ensure that all fire events are detected, especially in early stages. Future work should focus on optimizing recall, ensuring robust performance under variable terrain and lighting conditions, and considering the real-time limitations of hardware on UAVs [13]. Moreover, real-time decision-making algorithms could be developed to trigger immediate intervention for high-confidence predictions.

7 Conclusion

This project successfully implemented CNN-based models for both binary and ternary wildfire detection using UAV imagery. Key improvements included the application of transfer learning with MobileNetV2, enhanced data augmentation techniques, Monte Carlo (MC) Dropout for uncertainty estimation, and label smoothing for better calibration and reduced overconfidence. Future work will explore temporal modeling, multimodal learning, and real-time deployment in operational settings to further enhance wildfire detection systems.

References

1. Jain, P., Coogan, S. C. & Subramanian, S. G. e. a. A review of wildfire detection using remote sensing. *Remote Sensing* **12,** 924 (2020).

2. Chen, Y., Lin, Z. & Zhao, X. Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **7,** 2094–2107 (2014).

3. Mohammadpoor, M. & Bouffanais, R. UAV-based Wildfire Detection Using Deep Learning for Real-time Monitoring. *arXiv preprint arXiv:2012.14036* (2020).

4. Krizhevsky, A., Sutskever, I. & Hinton, G. E. *ImageNet classification with deep convolutional neural networks* in *Advances in Neural Information Processing Systems* **25** (2012), 1097–1105.

5. Szegedy, C. *et al. Going deeper with convolutions* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), 1–9.

6. Shorten, C. & Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data* **6,** 1–48 (2019).

7. Perez, L. & Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621* (2017).

8. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.-C. *MobileNetV2: Inverted Residuals and Linear Bottlenecks* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), 4510–4520.

9. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of machine learning research* **15,** 1929–1958 (2014).

10. Müller, R., Kornblith, S. & Hinton, G. *Does label smoothing mitigate label noise?* in *Advances in neural information processing systems* **32** (2019).

11. Gal, Y. & Ghahramani, Z. *Dropout as a Bayesian approximation: Representing model uncertainty in deep learning* in *International conference on machine learning* (2016), 1050–1059.

12. Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. *On calibration of modern neural networks* in *International Conference on Machine Learning* (2017), 1321–1330.

13. Tzeng, E., Hoffman, J., Saenko, K. & Darrell, T. *Adversarial discriminative domain adaptation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 7167–7176.

14. Zhang, Y., Liu, M., Yang, X. & Huang, J. *Deep learning for multi-task learning* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 1444–1452.

15. Selvaraju, R. R. *et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization* in *Proceedings of the IEEE International Conference on Computer Vision* (2017), 618–626.

16. Lundberg, S. M. & Lee, S.-I. A unified approach to interpreting model predictions. *Advances in neural information processing systems* **30** (2017).

1 Appendix: Scripts and Training Preparation

1.1 SLURM Job Submission Script

```
#!/bin/bash -l

#SBATCH --job-name=fire-cnn
#SBATCH -p GPU
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --cpus-per-task=4
#SBATCH --gres=gpu:a100:1
#SBATCH --mem=50G
#SBATCH --time=2-00:00:00
#SBATCH --output=/home/xzcapis1/fire/logs/slurm-%x-%j.out
#SBATCH --mail-user=ian.ambak.22@ucl.ac.uk
#SBATCH --mail-type=END,FAIL

cd /home/xzcapis1/fire

module load eb CUDA/12.1.1
source /home/xzcapis1/miniconda3/etc/profile.d/conda.sh
conda activate fire

nvidia-smi
python -c "import␣tensorflow␣as␣tf;␣print(tf.config.list_physical_devices('GPU
    '))"

export TMPDIR=$HOME/tmp_tf
mkdir -p $TMPDIR

python firetestlight.py
```

Listing 1: SLURM batch script used to schedule model training jobs on the Dias GPU cluster, including resource requests and environment setup.

2 Model Architecture

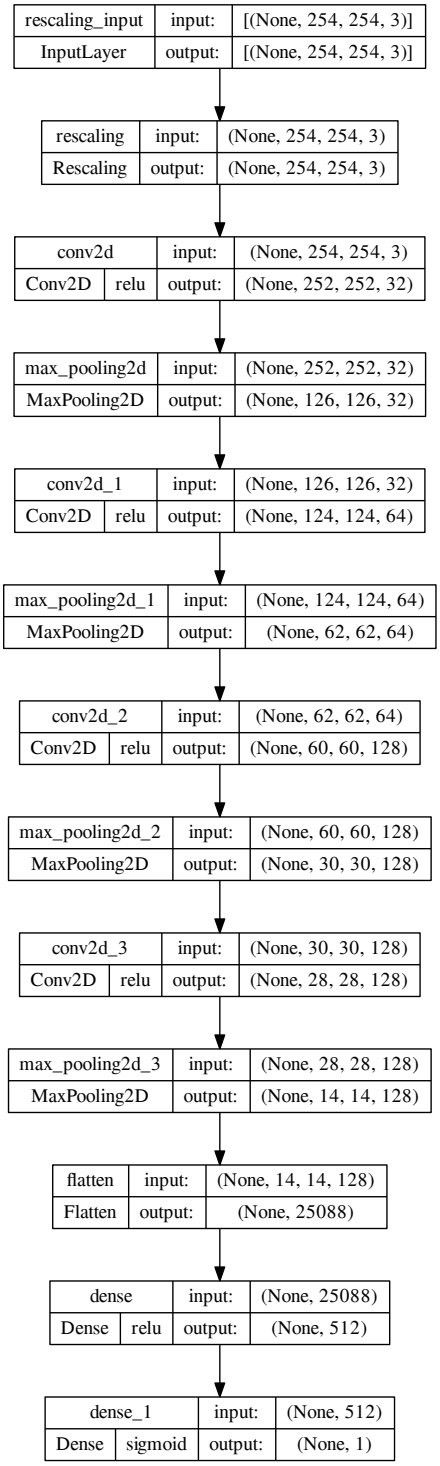

Figure 9 Custom CNN architecture used for binary fire classification.