Sensors and Materials, Vol. 37, No. 8 (2025) 3621–3631
MYU Tokyo

S & M 4141

High-precision Wildfire Detection Algorithm Using an Enhanced You Only Look Once Model in a Jetson Xavier Environment

Tae-Hwan Kim,[1] Eun-Su Seo,[2] and Se-Hyu Choi[3*]

[1]Department of Spatial Information, Kyungpook National University, 41566, 80, University Road, Buk-gu, Daegu, South Korea
[2]School of Science and Technology Acceleration Engineering, Kyungpook National University, 41566, 80, University Road, Buk-gu, Daegu, South Korea
[3]School of Architectural, Civil, Environmental and Energy Engineering, Kyungpook National University, 41566, 80, University Road, Buk-gu, Daegu, South Korea

(Received April 29, 2025; accepted August 8, 2025)

Keywords: deep neural networks, artificial intelligence, YOLO, Xavier, wildfire smoke

The You Only Look Once (YOLO) algorithm is employed to develop a system for detecting wildfire smoke and providing early warnings, with a deep neural network (DNN) as its underlying architecture. To achieve accurate real-time smoke detection, the system was optimized through experiments conducted under diverse conditions. It was then implemented in an embedded computing environment to enhance the efficiency of wildfire detection. The findings demonstrate the effectiveness of this DNN-based smoke detection system in real-world environments.

1. Introduction

We employed the You Only Look Once (YOLO) model to enhance wildfire detection by integrating artificial intelligence into an embedded environment.(1) Specifically, a Jetson Xavier-based system is utilized in conjunction with a Pixhawk flight controller—a widely adopted configuration in UAV applications. The Pixhawk functions as the primary control unit, enabling the stable management of peripheral components and seamless communication with the Xavier module. For real-time wildfire smoke detection, YOLOv5 is selected as the core detection model owing to its lightweight structure, high compatibility with embedded systems, and robust performance in resource-constrained scenarios. Compared with YOLOv8, YOLOv5 offers simpler deployment and more efficient TensorRT optimization, which is essential for real-time inference. Additionally, official support within NVIDIA's DeepStream SDK reinforces its suitability for field-based disaster detection tasks. YOLO addresses object detection as a regression problem, utilizing a single network to ensure rapid processing.(2) The key advantage of YOLO is its ability to significantly reduce false positives in background detection.(3) Additionally, YOLO demonstrates a high capacity for learning common patterns, enabling its

*Corresponding author: e-mail: shchoi@knu.ac.kr
https://doi.org/10.18494/SAM5716

ISSN 0914-4935

application in artistic fields such as painting.[4] Its adaptability across diverse domains further highlights its versatility.[5] In object detection, YOLO tracks multiple objects and their locations by combining classification and localization within a unified framework.[6]

Before the adoption of YOLO, object detection was primarily achieved using deformable part models (DPMs) and region-based convolutional neural networks (R-CNNs).[7] An R-CNN model follows a two-stage detection approach, where region proposal and classification occur sequentially.[8] This process involves image data initialization, execution of the two-stage detector for region proposal and classification, selective search, and region proposal network operations, followed by classification through support vector machines (SVMs) and Softmax.[9] After retraining, results are obtained through multiclass classification and bounding box regression.[10] Despite its complexity, R-CNN achieves high mean average precision (mAP).[11]

In contrast, YOLO's learning method is based on human-like pattern recognition, performing classification and localization simultaneously.[4] Unlike previous two-stage detectors, YOLO formulates object detection as a single regression problem, enabling real-time performance.[5] The model processes entire images at once, effectively minimizing errors related to background misclassification—an issue common in conventional methods.[2] Furthermore, YOLO's ability to learn generalized representations facilitates seamless adaptation to novel domains.[12]

YOLO also offers remarkable efficiency. The base model operates at 45 frames per second (fps), whereas the fast variant reaches 150 fps. For comparison, typical frame rates include 24 fps for films, 30 fps for television dramas, and 60 fps for sports broadcasts.[13,14] Given these advantages, we leveraged YOLO's speed, accuracy, and adaptability to develop an efficient, real-time wildfire detection system.

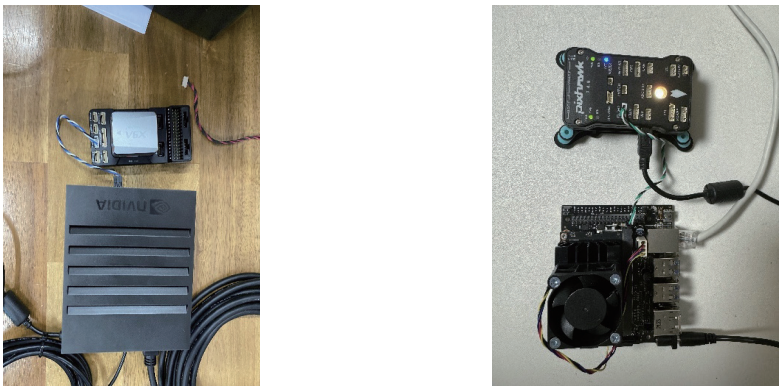
2. Data, Materials, and Methods

The research apparatus consists of a Xavier system connected to a Pixhawk device. The Pixhawk, commonly used in drones, serves as the primary controller for managing various components. This connection between the Xavier and the Pixhawk forms a fundamental part of the broader research setup, facilitating the operation and testing of the system.

To enable real-time wildfire smoke detection on the embedded Xavier platform, YOLOv5 was adopted as the core AI model. Compared with YOLOv8, YOLOv5 offers greater compatibility, simpler structure, and smoother TensorRT optimization for deployment in resource-constrained environments. Moreover, it is officially supported in NVIDIA's DeepStream SDK, which enhances its reliability for real-time inference in embedded disaster detection systems.

Figure 1(a) shows the system architecture for onboard wildfire smoke detection. The Jetson Xavier module serves as the AI inference engine, whereas the Pixhawk unit is used to interface with UAV or sensor systems for control and coordination. These devices were not used for image acquisition, but for deployment testing and validation of the model in embedded scenarios.

The training dataset was obtained from the Wildfire Smoke Dataset provided by Roboflow, which includes real wildfire smoke images annotated for object detection. The dataset contains imagery collected in part by High Performance Wireless Research and Education Network



(a) (b)

Fig. 1. (Color online) (a) Embedded computer environment A (Jetson Xavier AGX & Pixhawk V6X) and (b) embedded computer environment B (Jetson Xavier NX & Pixhawk 4).

under the support of the United States National Science Foundation, and it is suitable for training models to recognize early smoke patterns associated with wildfires.

2.1 Dataset preparation for wildfire detection

The use of catastrophe data for machine learning purposes can be developed and implemented in a variety of ways. We provide a detailed explanation and practical guide to loss function optimization, particularly for those unfamiliar with AI.

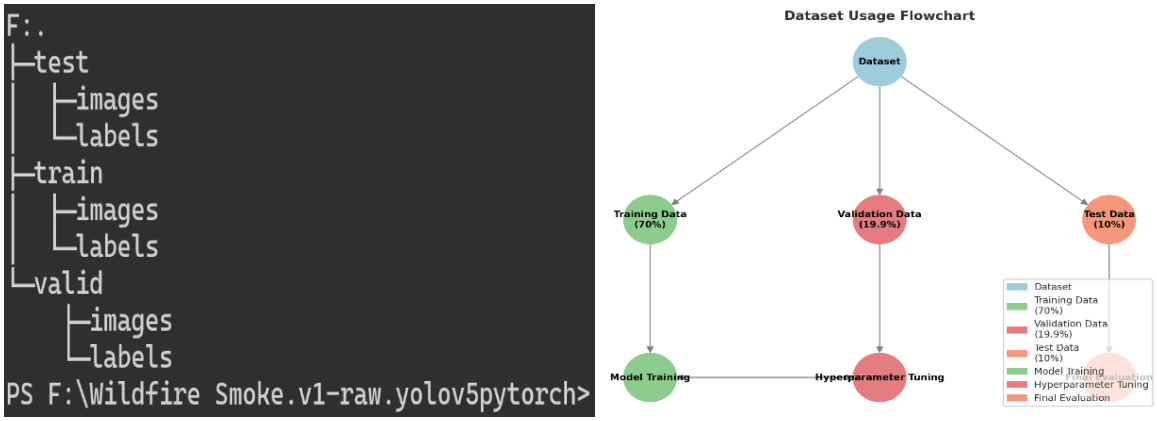
In terms of data acquisition, the dataset is crucial for training AI and deep learning models. Typically, the data is divided into three categories: training, validation, and testing. The purpose and role of each dataset are as follows:

- Training dataset: This is used to train the model, during which the model learns patterns in the data and adjusts its weights.
- Validation dataset: This is employed during training to evaluate the model's performance and prevent overfitting. This dataset helps assess the model's learning process and ensures it does not memorize the data.
- Test dataset: This is used to evaluate the model's final performance. It is not involved in the training or validation phases, and its purpose is to assess the generalizability of the model to new, unseen data.

The dataset distribution is as follows: 519 training samples (70.0%), 147 validation samples (19.9%), and 74 test samples (10%). The file structure is shown in Figs. 2(a) and 2(b).

2.2 YOLO model architecture and training configuration

The theoretical methods we employed are as follows: focal loss was developed to prioritize challenging examples during the model's learning process, particularly when dealing with unbalanced datasets. The formula for focal loss is



(a) (b)

Fig. 2. (Color online) (a) Dataset file structure and (b) dataset usage flowchart.

$$focal\ loss(p_t) = -\alpha_t (1 - p_t)^{\gamma} \cdot \log(p_t), \tag{1}$$

where p_t is the predicted probability value, with $p_t = p$ for positive examples ($\gamma = 1$) and $p_t = 1 - p$ for negative examples ($\gamma = 0$). α_t is the weighting factor for positive/negative examples, increasing the weight of positive examples to improve model performance on unbalanced data. γ is the damping factor, where a larger value reduces the model's loss on easy-to-predict samples while emphasising difficult samples.

The modified focal loss function is applied to the YOLOv5 model as

$$L_{focal} = \sum_{l=1}^{N} \alpha (1 - p_l)^{\gamma} \cdot \log(p_l) \text{ for each prediction } p_l, \tag{2}$$

where α and γ are used to adjust the learning process for difficult examples in the unbalanced dataset.

Next, the weights of the loss functions are adjusted to enhance the performance of the YOLOv5 model. These weights determine the model's focus on specific loss factors during training. The weights for each element of the loss function are defined as

$$L_{total} = \lambda_{box} \cdot L_{box} + \lambda_{obj} \cdot L_{obj} + \lambda_{cls} \cdot L_{cls}. \tag{3}$$

• Bounding box loss weight (λ_{box}): This weight reduces the difference between the predicted bounding box and the actual box. Adjusting this value allows the model to focus on the accuracy of bounding box predictions.

• Object loss weight (λ_{obj}): This assesses how well the model detects the presence of objects. Increasing this weight makes the model focus more on detecting objects, which is crucial for identifying ambiguous objects such as smoke.

• Class loss weight (λ_{cls}): This evaluates how well the model predicts the class of detected objects. By reducing this weight, the model can focus more on object detection and bounding box accuracy, and less on class prediction.

Bounding box loss (L_{box}) is based on the intersection over union (IoU) between the predicted and actual boxes, whereas the object presence loss (L_{obj}) evaluates the probability that an object exists. The class prediction loss (L_{cls}) measures the difference between the predicted and actual class labels.

The mathematical explanation of the modified code is that the settings of α and γ in the adjusted loss function are based on the focal loss formula, which prioritizes hard samples during training. The weights in the loss function (λ_{box}, λ_{obj}, and λ_{cls}) determine which aspects the model focuses on during learning. In this implementation, γ = 1.2 increases the weight of difficult samples, but the value is kept moderate to maintain training stability. Here, α = 0.5 boosts the contribution of positive examples, ensuring the model learns more effectively from them, while γ = 1.2 assigns more weight to hard examples without compromising training stability. The modified weight settings are as follows: λ_{box} is increased to 1.15, λ_{obj} is increased to 1.30, and λ_{cls} is reduced to 0.85. These adjustments ensure that the model places greater emphasis on detecting bounding boxes and objects, ultimately leading to more accurate detection outcomes. This process guides the YOLOv5 model to focus on challenging objects, such as smoke and smog, while allowing for the flexible optimization of the model's performance through the loss function.

3. Results

3.1 Adjusting the weights of the YOLO loss function

The data reveals a precision of 0.91 and a mAP50 of 0.929 for the YOLO model with 50% overlap, which is notable. This model represents the unoptimized YOLO model. After training, the results can be checked using *val.py*, which often shows near-identical results to those from the final stage of *train.py*. Table 1 shows the results after optimizing the model, based on the guide outlined in Sect. 2.

The optimization results show a significant improvement in precision, which increased from 0.910 to 0.937 after adjusting the loss function and applying focal loss. This led to a reduction in the rate of false positives and an improvement in prediction accuracy.

Recall initially decreased from 0.912 to 0.876 in the model with the adjusted loss function, but the focal loss model improved its Recall to 0.939. mAP50 was highest in the focal loss model,

Table 1
AI model performance scorecard.

	Precision	Recall	mAP50	mAP50-95
YOLOv5	0.910	0.912	0.928	0.548
Loss function (weighted)	0.928	0.876	0.915	0.517
Loss function (weighted) + Focal loss model	0.937	0.939	0.949	0.540

reaching 0.949, indicating excellent results with a narrow IoU. However, mAP50-95 remained similar to that of the original YOLOv5 model, at 0.548 and 0.540, respectively.

From the analysis, the focal loss model stands out as optimal, exhibiting the best performances in terms of Precision, Recall, and mAP50. However, the improvement in mAP50-95 is marginal, suggesting that there is still room for enhancing performance across different IoU criteria, potentially through data augmentation, IoU-based loss function adjustments, and similar techniques. Figure 3 presents a performance evaluation graph of the deep neural network (DNN), demonstrating its training stability and detection accuracy over time.

For practical applications, the recommended model is the focal loss model, as it offers a good balance between Precision and Recall with high overall performance. Future improvements can focus on data augmentation to increase IoU diversity, which can improve mAP50-95. Additionally, incorporating learning methods that consider different IoU criteria during training can further optimize the model's performance.

3.2 Application and effects of focal loss on YOLO model

The model that combined weighted loss and focal loss delivered the best performance. We further trained this model with the results shown in Table 2.

When analyzing the impact of epoch numbers on optimization, Precision peaked at 0.942 at Epoch 200, showing the best reduction in the rate of false positives. However, at Epoch 100, Precision dropped to 0.913, suggesting potential overfitting or instability in the early stages of learning.

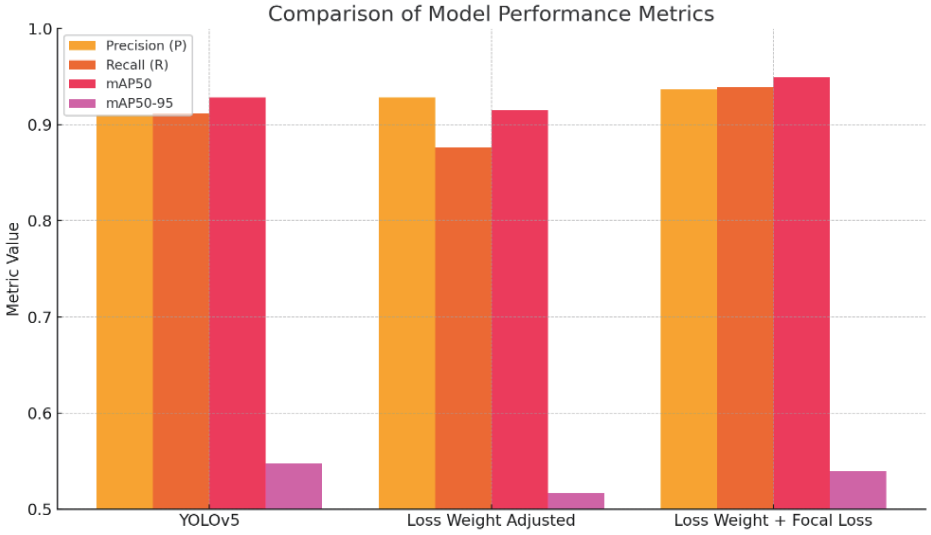


Fig. 3. (Color online) Deep neural network performance evaluation graph (Precision, Recall, and F1 score).

Table 2
Model performance scorecard by number of epochs.

	Precision	Recall	mAP50	mAP50-95
Epoch 50	0.937	0.939	0.949	0.540
Epoch 100	0.913	0.924	0.942	0.561
Epoch 200	0.942	0.905	0.952	0.545

Recall peaked at 0.939 at Epoch 50, detecting the truest positives, but decreased as the number of epochs increased, dropping to 0.905 by Epoch 200. This decline likely reflects overfitting, limiting the model's ability to generalize.

The highest mAP50 score of 0.952 was achieved at Epoch 200, with Epoch 50 also delivering a high score of 0.949, indicating that additional epochs do not always guarantee better results.

The mAP50-95 score peaked at 0.561 at Epoch 100, which was the best across the various IoU criteria. Epochs 50 and 200 showed slightly lower mAP50-95 scores of 0.540 and 0.545, respectively.

On the basis of these results, Epoch 50 provides a balanced and stable performance, offering high Precision, Recall, and mAP50 with relatively low mAP50-95. Epoch 100 excels in mAP50-95, providing strong performance across multiple IoU criteria, although with reductions in Precision and Recall. Epoch 200 achieves the highest Precision and mAP50 but suffers from reduced Recall and mAP50-95.

In conclusion, as shown in Fig. 4, the DNN demonstrates stable performance across various training conditions. Epoch 50 offers the best overall balance of Precision and Recall, making it suitable for general use. Epoch 100 is recommended when performance across multiple IoU criteria is a priority. Epoch 200 is most appropriate when maximizing Precision and mAP50 is essential, although the risks of reduced Recall and potential overfitting should be considered.

4. Discussion

In particular, the model trained with class-weighted loss and focal loss showed a noticeable improvement in mAP50-95 compared with the baseline YOLOv5. Since mAP50-95 evaluates detection performance across a wide range of IoU thresholds, the improvement suggests that the model is better at detecting small or faint smoke regions, which are critical in the early stages of wildfire development.

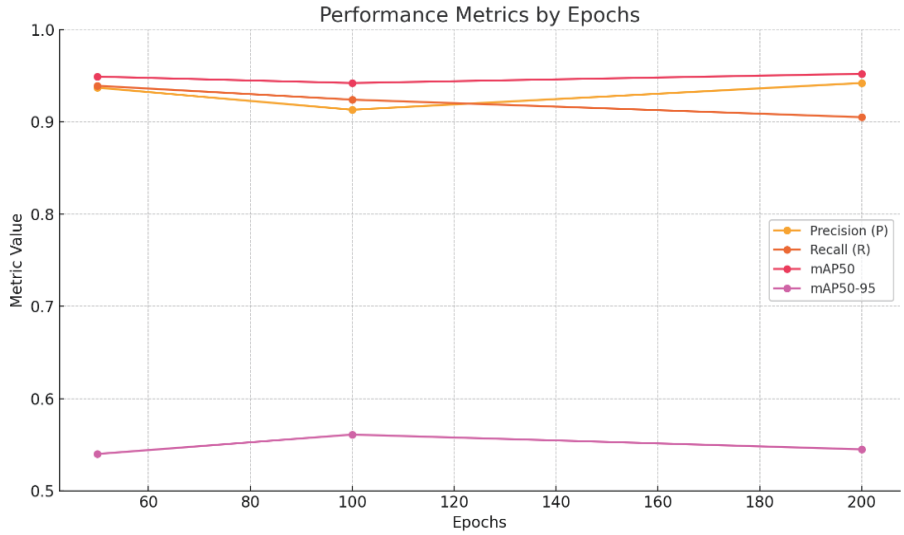


Fig. 4. (Color online) Deep neural network analysis graph.

Therefore, the numerical gains are not just statistical but reflect a meaningful improvement in the practical early detection capability of the model in real-world wildfire scenarios. A more detailed examination of the YOLO model is presented below. Initially, YOLO processes the input image by dividing it into an S-by-S grid. It then performs two key operations. First, each grid cell is assigned two bounding boxes. The model learns the center, height, width, and confidence value of each box, akin to the bounding box regression in R-CNN. Second, a classification task is performed for each grid cell. Classes are defined on the basis of the dataset, and the model calculates the probability that each grid cell belongs to a particular class. The class with the highest probability is then assigned to the grid. After these two steps, irrelevant data is discarded, and the results of training the YOLO algorithm are generated.

Figure 5 shows the result of applying the proposed wildfire smoke detection model to a real wildfire video recorded in a mountainous area in South Korea. The model successfully detects the smoke region with a bounding box and confidence score, demonstrating its inference capability on real-world data. This result visually supports the model's applicability to practical wildfire monitoring scenarios.

Focal loss is an extension of the standard cross-entropy loss. It calculates the difference between the predicted class probability distribution (as per the cross-entropy loss model) and the true labels. Cross-entropy loss yields a smaller value when the prediction is closer to the true label, and a larger value when the prediction is further from the truth. Focal loss, however, is specifically designed to give more weight to difficult examples—those the model struggles to predict. It is especially effective in addressing the issue of class imbalance, which is common in many datasets.

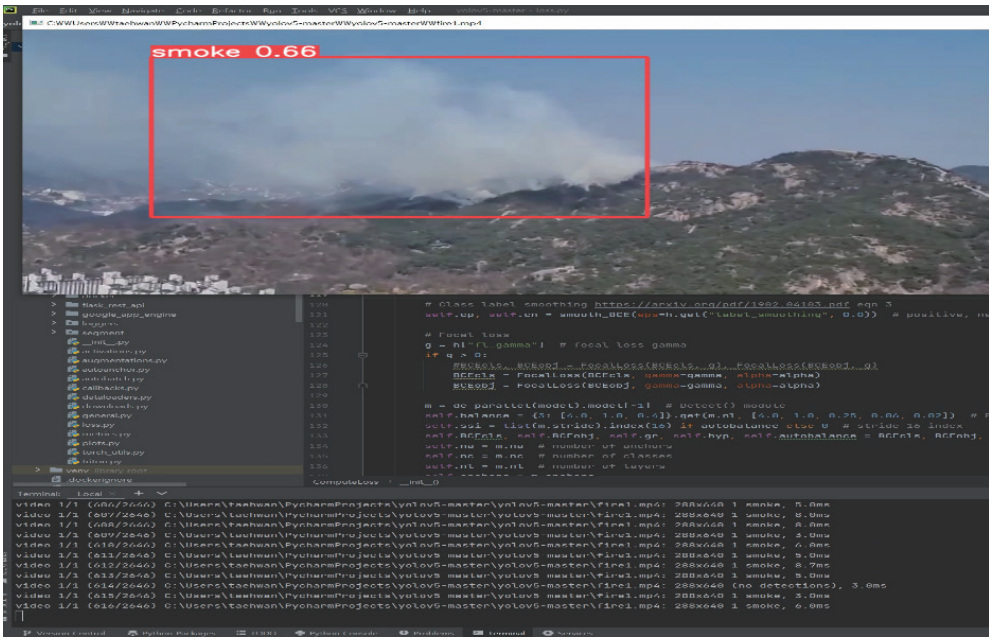


Fig. 5. (Color online) Smoke detection (DNN).

In summary, while cross-entropy loss calculates the loss equally for all examples, focal loss shifts focus toward the more challenging examples by increasing their weight. This ensures that the model places greater emphasis on those predictions that are difficult to make, improving performance on hard-to-detect objects.

The use of focal loss in YOLOv5 helps mitigate the problem of data imbalance. In typical object recognition tasks, there is often an overwhelming number of background pixels and fewer objects in each image. This imbalance can lead the network to focus excessively on the background in easy examples, resulting in poor performance when detecting more difficult objects.

Various strategies are employed to optimize performance, such as tuning the loss function, implementing data augmentation, optimizing the hyperparameters, enhancing the model structure, applying regularization techniques, and planning learning strategies. Focal loss is one such strategy, built upon cross-entropy loss but with greater emphasis on difficult examples and reduced emphasis on easy ones. This shift allows the network to focus on the challenging examples that it previously failed to predict accurately, ultimately improving the model's performance.

Focal loss is particularly useful for small-object detection and unbalanced datasets, offering more reliable and accurate detection results. The advantages of incorporating focal loss in YOLOv5 include a reduction in the rate of false negatives, improved model stability, and enhanced mAP. Fewer false negatives mean that the model better detects objects by prioritizing challenging ones. Greater stability in model learning helps reduce issues with unstable training, especially on unbalanced datasets, while the improvement in mAP boosts model performance in terms of both Precision and Recall.

These performance gains were most prominent at epoch 50 in our experiments; however, this result is dependent on the characteristics of the specific dataset used. For datasets with different properties, the optimal number of training epochs may vary and should be determined accordingly.

5. Conclusions

We explored AI-based wildfire smoke detection using the YOLOv5 object detection framework, enhanced with class-weighted and focal loss techniques. Through the comparative analysis of training statistics, we identified that epoch 50 provided the most stable and balanced performance across Precision, Recall, and mAP50, making it the optimal choice for general applications. Epoch 100 is preferable when the mAP50-95 performance is prioritized, while epoch 200, although achieving higher precision, increases the risk of overfitting.

These findings demonstrate the practical applicability of customized loss functions and training strategies within the YOLOv5 architecture for geospatial AI in disaster detection. Furthermore, the results provide a foundation for improving dataset quality and selecting appropriate training configurations for real-time wildfire monitoring systems. Future research will focus on multi-disaster datasets and deploying lightweight models in embedded environments to ensure faster and more reliable inference.

Acknowledgments

This research was supported by the Project for Practical Use of Regional Science and Technology Performance of the Commercialization Promotion Agency for R&D Outcomes (COMPA) funded by the Ministry of Science & ICT (grant number: RS-2022-CP000143, Kyungpook National University).

References

1 G. Jocher, A. Stoken, J. Borovec, C. Changyu, and A. Laughing: Front. Environ. Sci. **12** (2023) 1486212. https://doi.org/10.3389/fenvs.2024.1486212
2 J. Redmon, S. Divvala, R. Girshick, and A. Farhadi: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2016) 779–788. https://doi.org/10.1109/CVPR.2016.91
3 J. Redmon and A. Farhadi: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2017) 7263–7271. https://doi.org/10.1109/CVPR.2017.690
4 J. Redmon and A. Farhadi: arXiv preprint (2018). https://arxiv.org/abs/1804.02767
5 A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao: arXiv preprint (2020). https://arxiv.org/abs/2004.10934
6 W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg: Proc. Eur. Conf. Comput. Vis. (2016) 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
7 R. Girshick, J. Donahue, T. Darrell, and J. Malik: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2014) 580–587. https://doi.org/10.1109/CVPR.2014.81
8 S. Ren, K. He, R. Girshick, and J. Sun: Adv. Neural Inf. Process. Syst. **28** (2015) 91.
9 R. Girshick: Proc. IEEE Int. Conf. Comput. Vis. (2015) 1440–1448. https://doi.org/10.1109/ICCV.2015.169
10 K. He, G. Gkioxari, P. Dollár, and R. Girshick: Proc. IEEE Int. Conf. Comput. Vis. (2017) 2961–2969. https://doi.org/10.1109/ICCV.2017.322
11 T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár: Proc. IEEE Int. Conf. Comput. Vis. (2017) 2980–2988. https://doi.org/10.1109/ICCV.2017.324
12 C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2021) 13029–13038. https://doi.org/10.1109/CVPR46437.2021.00294
13 G. Jocher, A. Stoken, J. Borovec, C. Changyu, and A. Laughing: Zenodo (2020). https://doi.org/10.5281/zenodo.4154370
14 C. Y. Wang, H. Y. M. Liao, Y. H. Wu, P. Y. Chen, J. W. Hsieh, and I. H. Yeh: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW) (2020) 390–391. https://doi.org/10.1109/CVPRW50498.2020.00203

About the Authors

Tae-Hwan Kim received his bachelor's degree from Daegu University in 2018 and his master's degree from Kyungpook National University in 2021. Since 2021, he has been pursuing his doctorate degree at Kyungpook National University. His research interests include AI, MEMS, and spatial information (taehwan.417@knu.ac.kr)

Eun-Su Seo received his B.S., M.S., and Ph.D. degrees in 2010, 2013, and 2016, respectively, from Kyungpook National University, Korea, where he has been a research professor since 2022. His research interests are in digital twins, disaster prevention, and acceleration. (esseo@knu.ac.kr)

Se-Hyu Choi received his B.S., M.S., and Ph.D. degrees in 1990, 1995, and 2000, respectively, from Kyungpook National University, Korea, where he has been a professor since 2004. His research interests are in AI, disaster prevention, and sensors. (shchoi@knu.ac.kr)