applied sciences

MDPI

Article

Early Fire Detection System by Using Automatic Synthetic Dataset Generation Model Based on Digital Twins

Hyeon-Cheol Kim †, Hoang-Khanh Lam †, Suk-Hwan Lee * and Soo-Yol Ok *

Department of Computer Engineering, Dong-A University, Busan 49315, Republic of Korea; iii4625@donga.ac.kr (H.-C.K.); 2371049@donga.ac.kr (H.-K.L.)

* Correspondence: skylee@dau.ac.kr (S.-H.L.); sooyol@dau.ac.kr (S.-Y.O.)

† These authors contributed equally to this work.

Abstract: Fire is amorphous and occurs differently depending on the space, environment, and material of the fire. In particular, the early detection of fires is a very important task in preventing large-scale accidents; however, there are currently almost no learnable early fire datasets for machine learning. This paper proposes an early fire detection system optimized for certain spaces using a digital-twin-based automatic fire learning data generation model for each space. The proposed method first automatically generates realistic particle-simulation-based synthetic fire data on an RGB-D image matched to the view angle of a monitoring camera to build a digital twin environment of the real space. In other words, our method generates synthetic fire data according to various fire situations in each specific space and then performs transfer learning using a state-of-the-art detection model with these datasets and distributes them to AIoT devices in the real space. Synthetic fire data generation optimized for a space can increase the accuracy and reduce the false detection rate of existing fire detection models that are not adaptive to space.

Keywords: digital twin smart city; particle system; synthetic learning data; early fire detection; object detection

check for updates

Citation: Kim, H.-C.; Lam, H.-K.; Lee, S.-H.; Ok, S.-Y. Early Fire Detection System by Using Automatic Synthetic Dataset Generation Model Based on Digital Twins. *Appl. Sci.* **2024**, *14*, 1801. https://doi.org/10.3390/app14051801

Academic Editor: Nikos D. Lagaros

Received: 8 December 2023
Revised: 15 January 2024
Accepted: 22 January 2024
Published: 22 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Among the disasters that might occur in a city, fire accidents can gradually spread their harm, causing property loss as well as damage to human life. A National Fire Agency study [1] states that there were 38,659 fires in Korea in 2020. Buildings and structures accounted for 64.5 % of all fire events, of which 82.2, 82.6, and 88.7% resulted in fatalities, injuries, and property damage, respectively.

Therefore, fire prevention requires considerable effort, and it is very important to realize this as soon as possible if a fire has not been prevented. Many studies on smart cities are being conducted to address these fire risks by fusing technologies such as artificial intelligence and the Internet of Things (IoT). The term "smart city" refers to an urban plan developed by the Ministry of Land, Infrastructure and Transport that uses modern technologies such as ICT big data to address urban issues and enhance quality of life [2]. According to urban population density, this urban plan is supposed to address a variety of social issues such as disasters, traffic jams, energy, and environmental pollution.

One of the technologies used in smart cities that is gaining interest is digital twin technology, which reproduces the world using twins that reflect the real world in a digital virtual space by sending data on the physical space gathered via IoT [3]. A digital twin is not the name for a specific piece of IT hardware. This concept spans a wide range of technologies, including sensors that gather information from actual physical environments; augmented reality (AR), which enhances the real world; and virtual reality (VR), which creates a virtual world [4]. Qiu et al. [5] created a distributed feedback carbon monoxide (CO) sensor based on lasers utilized for early fire detection and tested its reliability with

various trials. Li et al. [6] investigated and proposed an early fire detection approach based on a gas turbulent diffusion (GTD) model and particle swarm optimization (PSO). The test results showed that the sensor system performed well in terms of fire detection and location. Chen et al. [7] suggested a quick and low-cost indoor fire alarm system that could detect fires, carbon monoxide, smoke, temperature, and humidity in real time and effectively perform data processing and classification. These sensor-based detection networks have limitations in the coverage of the sensors involved, making them difficult to apply in early fire detection.

Meanwhile, with the recent development of data-driven artificial intelligence technologies such as deep learning, image-based initial fire detection methods have the advantage of being able to quickly recognize various types of fire-situation-related information, such as fire scale, compared to previously introduced fire monitoring sensors. Furthermore, by leveraging existing CCTVs, installation costs can be reduced, and the rate of misleading dispatch can be minimized by detecting the fire condition of the site in advance before deploying the alarm. However, using image-based deep learning models like CNNs requires a significant amount of data to train the model, but relatively few data exist, and they cost a lot of money to develop. Furthermore, it is safe to presume that no fire data exist for the early detection of fires—the goal pursued in this work. The development of synthetic learning data via digital twins is one way to overcome the problem of this learning data shortage and create high-quality learning data that can be deployed in the field. In this paper, we propose an early fire detection system using a digital-twin-based autonomous learning data generation model that can detect fires via image recognition sensors such as CCTVs.

In 2020, Fuller et al. [8] defined the concept of a digital twin (DT) as a two-way mutually influential relationship between a physical object and a digital object and presented DT technologies and tasks in fields such as manufacturing, healthcare, and smart cities. In the field of disaster safety systems for the prevention and prediction of large disasters such as natural disasters, fires, collapses, and environmental pollution accidents, simulation in the actual environment is often difficult. Therefore, a digital twin system that can simulate various scenarios by utilizing twins in the real and virtual worlds is necessary. A DT-based disaster safety management integrated platform study was co-planned at the Electronics and Telecommunications Research Institute (ETRI) in 2019, and research on the DT-based underground common area fire disaster support integrated platform commenced in 2020 [9]. This research used an integrated fire detection multi-sensor in the cavity and a fixed high-resolution LiDAR on the platform to detect irregularities and disaster locations in the cavity pre-space information, and then performed an intelligent escape induction method based on these details. Zohdi [10,11] suggested a simulated aerial firefighting activity DT framework after modeling the aircraft dropping of fire retardants in a fire scenario. A machine learning algorithm (MLA) model was optimized using DT simulation in the proposed method to quickly identify the optimal aircraft dynamics for maximizing the fire retardant release effectiveness while tracking the trajectory of airborne substances emitted from controlled aircraft. DT system studies on fire and catastrophe safety management are still in the early stages. It is especially important to investigate DT systems in early detection and reaction situations for massive fire prevention. Using a convolutional neural network (CNN), Pincott et al. [12] created a flame and smoke detection model intended for indoor use. The majority of early vision-based smoke detectors employed inference algorithms based on simple feature representations [13,14].

It is important to note that the presented technique has various shortcomings, including the misclassification of clouds as smoke from flames. Researchers raised the issue of insufficient high-quality data for detecting fires and smoke in study archives in [14]. Owing to the unique and challenging nature of collecting learning data in multiple situations for fires, it is cited as one of the difficult tasks in increasing accuracy using deep learning models. There are two types of fire detection model studies. The initial step is to transform the fire status of the entire image using the classification model, followed by finding fire

objects in the image using the object detection model and rebinarizing and reclassifying objects using the Backbone classification model.

Kim et al. [15] proposed a model that combines GoogleNet's Inception [16] block and ResNet's Skip connection [17] to detect fires in underground communities in dark environments. They enhanced the biased learning effect by categorizing the learning data as the target and supporting data, allowing the model to recognize the light source and geometric properties of the fire. Furthermore, both low-light scenarios, such as underground settlements and typical situations, were collected, and the model was learned using 10,200 target data points and 14,850 supporting data points. Liau et al. [18] developed a FireSSD model based on a single shot detector (SSD) appropriate for edge devices. They added Residual connection and group convolution to the SqueezeNet-based SSD model and used the Wide Fire module, Dynamic Mbox detection layer, normalization, and Dropout modules to improve the accuracy while maintaining real-time performance in the CPU. In these tests, the FireSSD model achieved an approximately 70.6 mAP performance on the Pascal VOC 2007 dataset, identifying processing speeds that were almost six times faster than the SSD300 model. Thomson et al. [19] presented two Convolutional Neural Network (CNN) designs, NasNet-A-OnFire and ShuffleNetV2-OnFire, with low complexity for detecting the boundaries of fire pixel areas in image space. They employed Dunnings et al.'s dataset of 14,266 fire photos and 12,073 non-fire images as well as a superpixel learning set of 54,856 fire and 167,400 non-fire superpixels on a test set of 1178 fire and 881 non-fire samples. The experiment yielded a 95% accuracy and 40 fps frame rate in fire classification for the entire frame and a 97% accuracy and 18 fps frame rate in super pixel localization. Additionally, testing on low-power devices in Nvidia Xavier-NX showed that this model is acceptable for real-world deployment applications, with 40 fps over full-frame classification with ShuffleNetV2-OnFire.

In general, datasets utilized in fire detection studies consist of data collected and analyzed from actual fire sites, and these data are frequently inappropriate for early fire detection. Furthermore, fire data from various contexts are restricted for use in closed fire-vulnerable spaces such as factories, offices, and specified locations. As a result, a dataset that synthesizes fires in digital twin simulations of fire-vulnerable locations is required, and this dataset should be developed to reflect the illumination, location, and diffusion processes in varied contexts from early to mid-fire.

The fire early detection system proposed in this study uses RGB-D cameras to obtain RGB-D images that are nearly identical to those of actual site CCTV cameras, generates virtual fires that are difficult to distinguish from real fires based on particle simulation, and creates various fire situations for highly accurate detection and low false alarm rates. Consequently, a vast amount of optimal customized learning data are generated to train an artificial intelligence model for early fire detection.

2. Materials and Methods

The proposed fire early detection system using digital-twin-based learning data generation is shown in Figure 1 and consists of steps such as real image data collection, virtual fire occurrence composite image generation, automatic generation of learning datasets, fire early model transfer learning, AI device inference, and post-processing.

2.1. Real Environment Data

Intel's RealSense D455 model was chosen to collect the RGB and Depth data required to create virtual data. RealSense was placed in the same composition as the RGB camera to detect fires at the actual site to secure RGB data and depth data. As the RealSense D455 features an IMU sensor as well, it can capture the camera angle while it is being recorded. This allows you to use Unity3D for the simulation to replicate the same composition in a virtual space. Recordings are possible using the Intel RealSense Viewer tool provided by Intel. The recorded content is stored in the ". bag" format, and the desired frame data can be obtained using the RealSense SDK. Figure 2 shows an example of the obtained

RGB and Depth frames. To secure actual image data at various sites, the recording was conducted considering possible environmental changes at the site, such as when the lights were turned off.

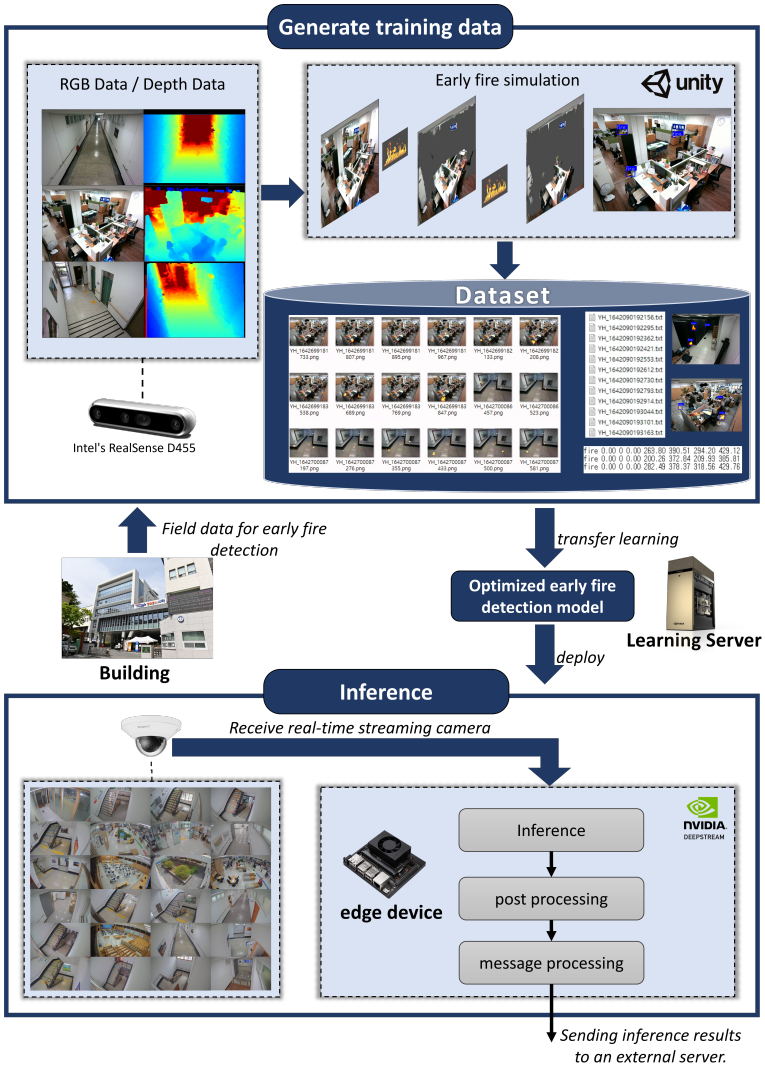
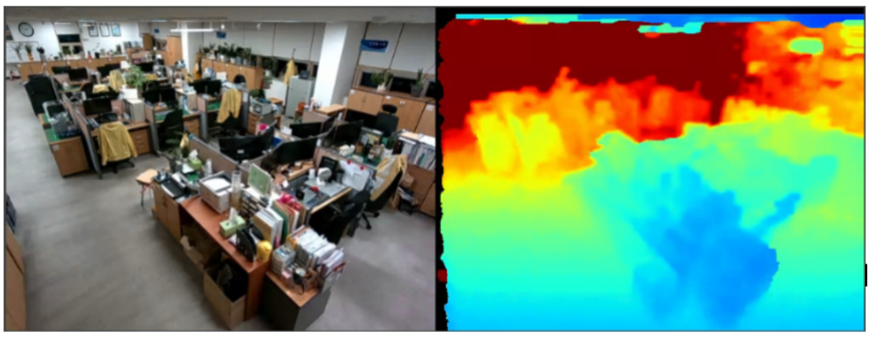


Figure 1. Structural diagram of the proposed digital-twin-based early fire detection system. Note: As depth increases, the depth data are scaled from blue to dark red.



(a) (b)

Figure 2. (**a**) A real office and (**b**) a depth image gathered from RealSense in the same viewpoint as CCTV in an office environment. Note: As depth increases, the depth data are scaled from blue to dark red.

2.2. Virtual Fire Data

The proposed system uses a Particle System to implement virtual fire. The Particle System is widely used to implement various phenomena that can be expressed by various particles such as fire, smoke, and water. Two-dimensional images are attached to the particles, and realistic simulations are possible with animation and proper particle movement. Figure 3 shows an example of a virtual fire that gradually increases using the Unity3D Particle System.



Figure 3. An example of a virtual fire implemented with Unity3D's Particle System.

Using the RealSense SDK, the recorded images from the actual environment were synthesized using the fire particle system on Unity 3D. The Background Segmentation Shader, which uses RealSense SDK RGB and Depth data, allows as many pixels as a distance in each section and transparently processes the rest. The backdrop of the actual space can be determined using this method by measuring the distance between the fire in each virtual space and the virtual camera and modifying the backdrop segmentation parameter as if the fire had reached the real space. If the data collected by the virtual fire and RealSense can be fused accurately, the data obtained by the virtual fire and RealSense can be synthesized. Figure 4 illustrates the depth-information-based synthesis process.

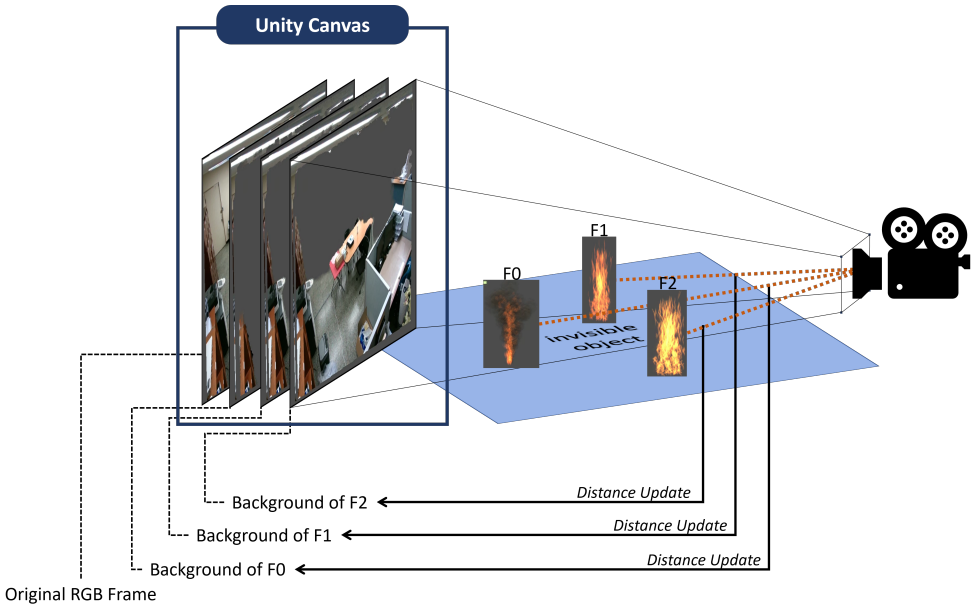


Figure 4. Synthesis process of virtual fire and real environment.

First, we create unprocessed RGB frames on Canvas in Unity3D, and then we begin rendering from the fire that is furthest away from the camera. We draw each fire, then calculate the distance between it and the virtual camera. Pixels closest to the fire are covered

by rendering the result of transparently executing a Background Segmentation of all pixels having a longer distance value. If all fires are drawn in the same manner by taking depth information into account, the natural occurrence of the item in front of the fire covering the fire from the camera's point of view can be recreated, as seen in Figure 5.

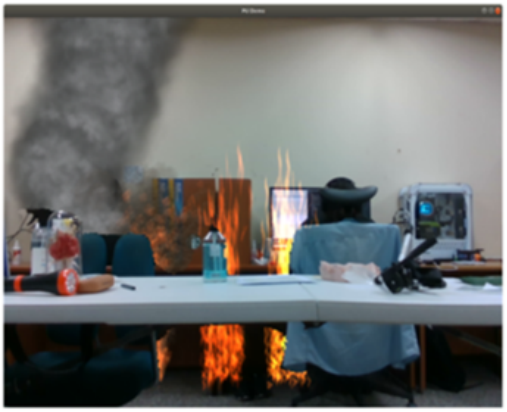


Figure 5. Results of a virtual fire simulation considering depth information.

Various fires were prepared by simulating different shapes based on the types of combustible materials, as shown in Figure 6. The scale of the fires was categorized into small, medium, and large, resulting in a total of 35 virtual fire objects used for generating fire data. The combustible materials included alcohol, animals, electricity, fibers, gasoline, kitchen fires, lamp oil, paint, plastic, rubber, vegetable oil, and wiring. The figures below illustrate some examples of these simulated fires.

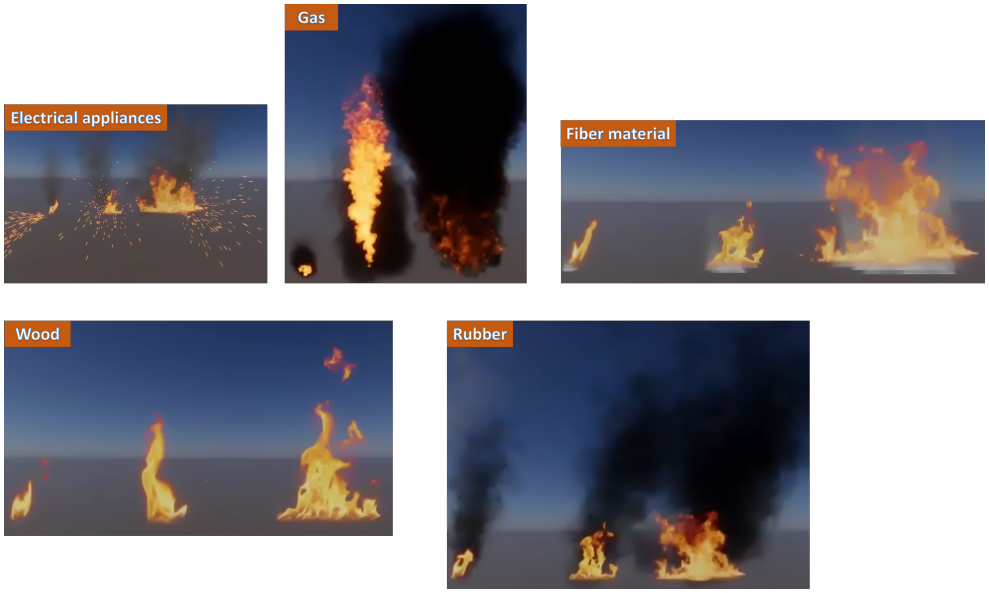


Figure 6. Example of different fire created when different burning materials.

2.3. Automatic Dataset Generation

A virtual fire must be maintained at an acceptable distance as it approaches the virtual camera viewing angle to avoid generating incorrect information. The automatic generating method requires technology to properly position the fire when a system is built that synthesizes with the actual space recorded when a fire is deployed, as described in Section 3.2. The challenge of automatic fire placement was addressed in this study in a straightforward manner. First, a virtual collider is positioned in front of the virtual camera. As shown in Figure 7a, we created a ray in the virtual space with the corresponding screen coordinates and ran Raycast to temporarily place the fire at the point where it hits the

virtual collider. The distance value in the actual space was read from the depth data at the point where each fire was rendered, and the distance between the virtual fire and the camera was adjusted so that the distance was less than a certain value. Figure 7b illustrates this process.

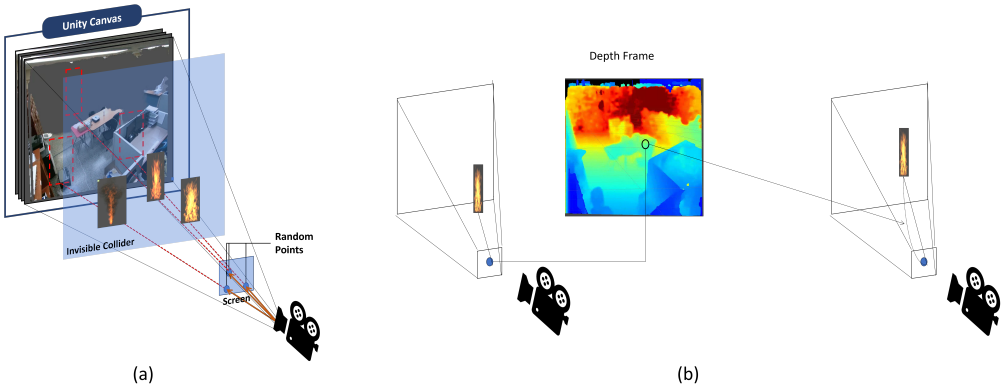


Figure 7. (**a**) Temporary positioning of fires using random on-screen coordinates and (**b**) automatic distancing for virtual fires. Note: As depth increases, the depth data are scaled from blue to dark red.

We extract natural fire-simulated images from automatically placed fires. Annotation information is required for this image, which uses the BoxCollider drawn by $(x_{min}, y_{min}, z_{min};\ x_{max}, y_{max}, z_{max})$ to specify the range of fire or smoke. When creating a virtual fire and calculating and sending out the axis-linked bounding box (AABB), which is a rectangular parallelepiped with a basis vector perpendicular to each of its faces, that the BoxCollider makes on the screen when extracting the learning data. Figure 8a is an example of setting up BoxCollider for Annotation, and Figure 8b shows AABB, which contains all the vertices when BoxCollider is drawn on the screen, in blue.

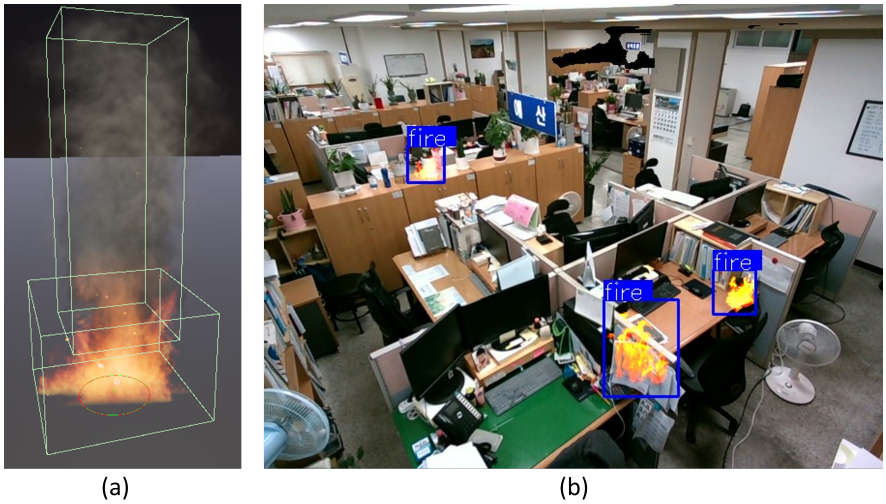


Figure 8. (**a**) Example of using BoxCollider for annotation and (**b**) automatically generated annotation information for fires on images.

Subsequently, the dataset to be utilized in the learning stage was built by merging the automatically generated virtual and actual data. The dataset [20], made available by FireNET, was used for the actual fire data. Figure 9 depicts the actual fire data, and Figure 10 depicts the virtual fire data generated automatically by the proposed technology.



Figure 9. Real-world fire training data released by FireNET [20].



Figure 10. Automatically generated virtual fire data optimized for your environment.

2.4. Fire Detection Model

Model learning was performed using the NVIDIA TAO Toolkit [21]. NVIDIA TAO is a toolkit that simplifies the establishment of deep learning models, such as the selection, optimization, and fine adjustment of pre-training models for transfer learning, including the Transfer Learning Toolkit (TLT). In addition, it provides a workflow from transfer learning to distribution on AI devices at the edge for actual field inference. There is a limitation in selecting a model for development because of the limited range of support models in NvidiaTAO. In this study, resnet18 was selected as YOLOv4 and as the backbone architecture among the Object Detection models provided by NVIDIA. Figure 11 shows a simplified model learning process.

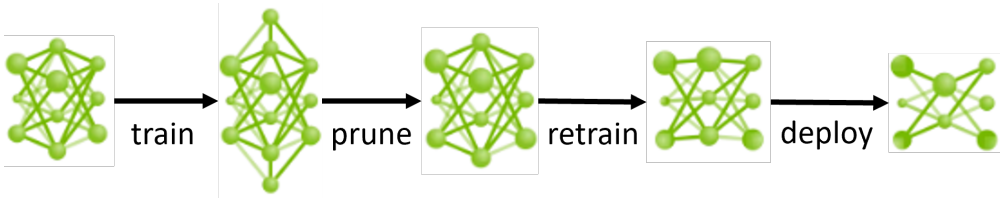


Figure 11. Transfer learning and model deployment course at NVIDIA TAO.

Basically, transfer learning is performed with a pre-learned model; however, once you learn, you reduce the number of parameters and learn once more through the pruning process. It is then converted into an encrypted model for distribution to edge devices. The model conversion process uses Deepstream in the inference step, which is described later. Figure 12 shows the result of inferring the test data by setting the bounding box display threshold to 0.6.



Figure 12. Inference from testing data.

2.5. AI Inference and Post-Processing

The field-optimal model that learns virtual data is inferred using an AI device at the edge end placed at each site. Devices that can be linked to the NVIDIA TAO include the NVIDIA Jetson's Xavier series and Orin series. In this study, the Jetson Xavier NX was used.

DeepStream is a GStreamer plug-in and library that can accelerate the inference process using deep learning. The models learned through the Transfer Learning Toolkit (TLT) can be distributed in Deepstream. DeepStream internally uses the TensorRT inference engine. Therefore, the encoded model file distributed by TAO is converted into an engine file for TensorRT using a Tao converter. By forming a GStreamer pipeline in a deepstream application for inference, the result of the final fire detection can be derived from input video decoding, such as CCTV, and post-processing to solve potential problems in the experimental field to be described later, and then sent to the outside by file or RTSP through an on-screen display (OSD). The inference result on the video, sink, or inference result can be sent to an external data server using a message broker. In this study, an external Kafka server was built and fire detection information was transmitted to the server as a Kafka message. Figure 13 shows the process from the model learned in TAO to the inference and result data of the Kafka message transmission.

Specifically, the message containing the fire detection information is composed as shown in Figure 14. For the detected object (fire), classID of fire or smoke, detection rectangle, and how confident the model was about this detection result were included, and the ID value of the edge device was attached as metadata.

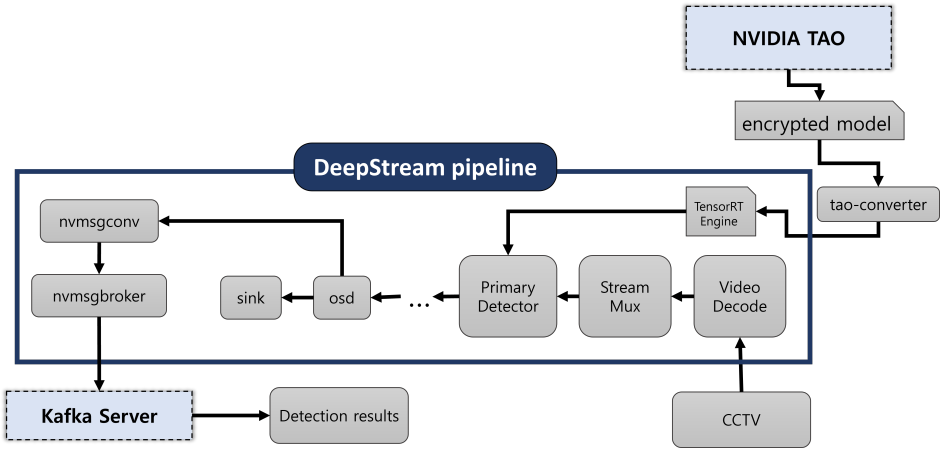


Figure 13. The process of utilizing models deployed in TAO in deepstream for inference and post-processing.

```
{
  "messageid" : "6360e7bb-8d57-4fc9-beb0-fcf3350809c1",
  "@timestamp" : "2023-08-23T07:36:29.034Z",
  "source_width" : 854,
  "source_height" : 480,
  "fireDetectionModule" : {
    "id" : "DONGA_TEST_A6000_DS6.2_CAM0",
    "detectionInfo" : [
      {
        "classID" : 0,
        "objectID" : 0,
        "left" : 377.0,
        "top" : 277.0,
        "width" : 61.0,
        "height" : 73.0,
        "confidence" : 0.96150881052017212
      }
    ]
  }
}
```


Figure 14. A message about the detection results sent by deepstream.

For both future research and the purpose of counting the number of fires, fire tracking is crucial. Consequently, post-processing was developed to accomplish this. We stored the difference between the diagonal length of the first bounding box of the tracked fire and the diagonal length of the subsequent bounding boxes and computed a $d_{cummulative}$ from the 15 most recent differences to verify the fire movement against a specific threshold.

$$d_t = |p_{0_{RB}} - p_{0_{LT}}| - |p_{t_{RB}} - p_{t_{LT}}|; d_{cummulative} = \sqrt{\sum_{t=1}^{n} (d_0 - d_t)^2} \quad (1)$$

As shown in Figure 15, from the center of the first bounding box of the tracked fire, we find the cosine of the angle between the vector $v_{0_{LT}}$ to the top left of the first bounding box and the vector $v_{t_{LT}}$ to the top left of the t_{th} bounding box $\theta_{t_{LT}}$ and check whether it falls within a certain threshold to verify whether it is a fire. Based on our experiment, we found that 0.9–0.98 was the best threshold to verify whether it was a fire.

$$cos\theta_{t_{LT}} = \frac{\vec{v_{0_{LT}}}.\vec{v_{t_{LT}}}}{|\vec{v_{0_{LT}}}||\vec{v_{t_{LT}}}|}; cos\theta_{t_{RB}} = \frac{\vec{v_{0_{RB}}}.\vec{v_{t_{RB}}}}{|\vec{v_{0_{RB}}}||\vec{v_{t_{RB}}}|} \quad (2)$$

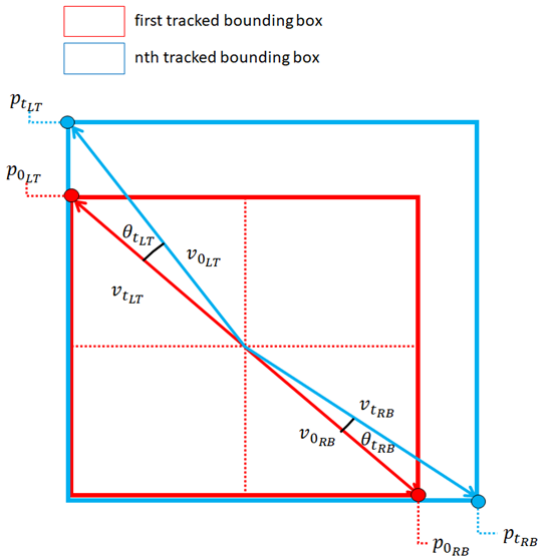


Figure 15. Analysis between two bounding boxes for tracking fire.

3. Experimental Results

3.1. IoT Installation

In this study, a unit comprising a CCTV for real-time video source retrieval at the edge, a router for Internet connectivity, and NVIDIA Xavier NX for running the fire detection application was assembled to facilitate inference at the edge. This unit was attached to the top of the wall where the CCTV was positioned. Figure 16 illustrates the internal components of the unit, its installation, and CCTV mounted on the ceiling.

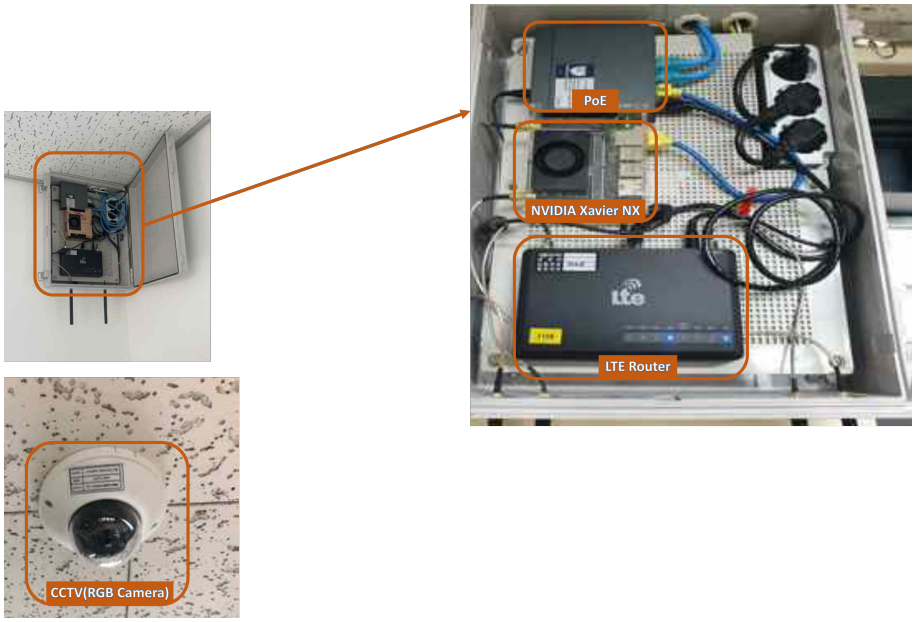


Figure 16. IoT is integrated into the proposed system.

3.2. Virtual Fire Data

As illustrated in Figure 17, it is composed of a virtual fire dataset and a real fire dataset generated using a digital-twin-based virtual fire simulation.

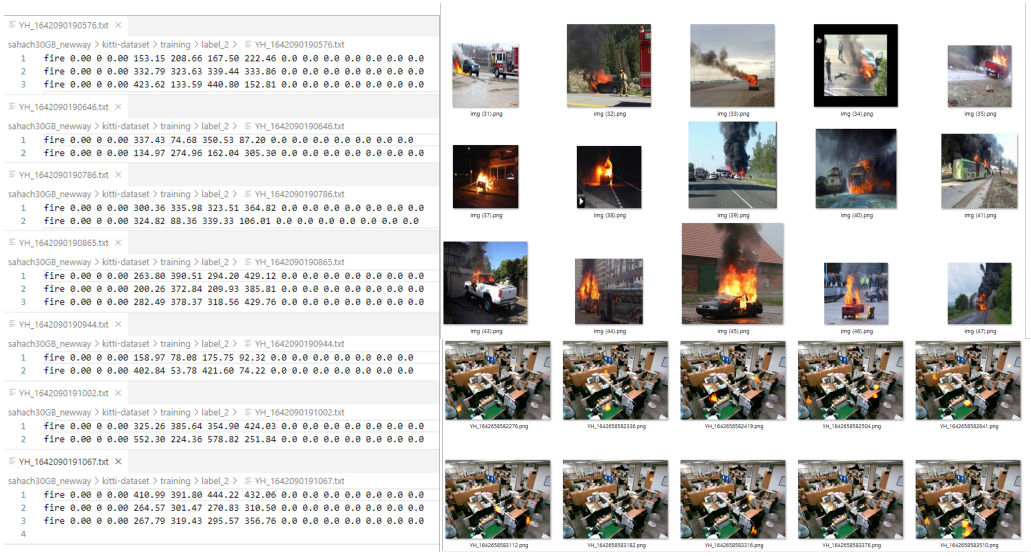


Figure 17. Training dataset consisting of virtual and real images of early fire and different fire types.

As a real environment for generating virtual fire data, we recorded RGB and Depth data for 43 indoor locations (classrooms, administrative offices, electrical rooms, server rooms, and corridors) when the indoor lights were turned on and off, and generated 7000 virtual flame particle data points and 7000 virtual smoke particle data points for each

location, with 14,000 images per location, for a total of approximately 600,000 virtual fire data points. Figure 18 shows an example of generated virtual fire images.

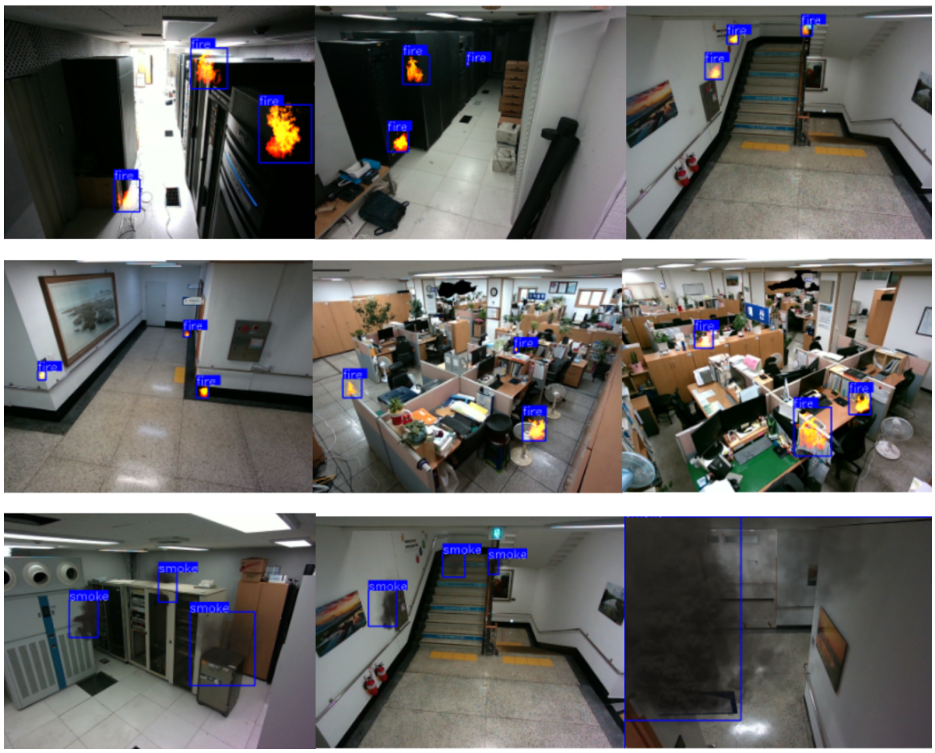


Figure 18. Virtual fire data generated from multiple real-world environments.

The main purpose of this study was to enable rapid detection of small initial fires at specific real sites. Therefore, the virtual data for each site and real-world data unrelated to the site were combined to obtain an optimized model for that site. The dataset for any site was constructed as shown in Table 1.

Table 1. Number of training images for a single specific environment.

Data	Virtual Data	Real-World Data	Total
Training data	4375	412	4787
Testing data	625	90	715
Total	5000	502	5502

3.3. Results

In the learning stage, NVIDIA's DGX A100 was used as a learning environment, as shown in Figure 19, and the Object Detection models provided by NVIDIA TAO, DetectNet_v2, FastRCNN, YOLOv4, EfficientDet, and DINO were developed and trained. The results of the evaluation using the test set of all learning models are shown in Table 2 below.

Table 2. The final evaluation results of the trained model.

Model	Unpruned Model Parameters	AP	Pruned Model Parameters	AP	Retrain/Model
DetectNetV2	11,200,458	0.93515	9,561,530	0.96316	0.85367
FasterRCNN	12,751,352	0.9528	10,434,616	0.9506	0.81831
YOLOv4	34,829,183	0.90909	3,659,191	0.9091	0.10506
EfficientDet	3,876,308	0.426	2,130,676	0.426	0.54966
DINO	-	0.83	-	-	-
D-DERT	-	0.71433	-	-	-



Figure 19. The NVIDIA DGX A100, which was used as a training environment.

In terms of optimizing the site, it is necessary to be very careful not to cause scientific practice, as it has learned the fire data that has occurred virtually at the site. To verify this, we used fire particles to generate virtual data to infer new data not used for learning, and fire images created using fire particles with different shapes were not used when generating virtual data. First, in the case of the former, regardless of the model, confidence was detected as 0.95 or higher without any problems, as shown in Figure 20, and there was no false detection.

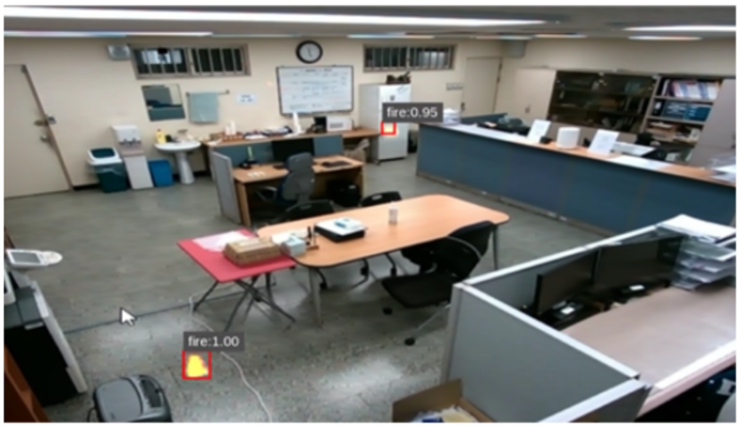


Figure 20. Inferring new fire footage from fires used to generate data for training.

In addition, in the latter case, we inferred virtual fires and other types of fires to generate learning data, and the results are shown in Figure 21.

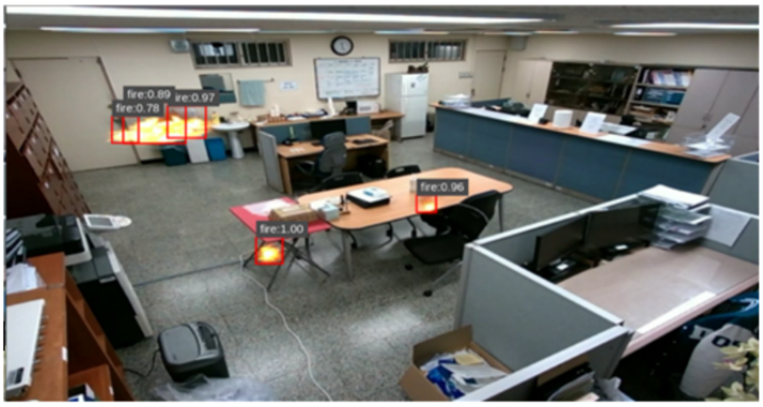


Figure 21. Detection of fire shapes that have never been used in training.

Even though the shape of the flame changed significantly, it exhibited a high detection rate. Similarly, there were a few misdetection results. However, this study learned data for field-optimized models, and even test sets for verification are simply composed of virtual fire data and actual fire data; therefore, it can be interpreted that the model performance indicators of all models are high. In contrast to the high-performance evaluation score, some models did not infer general fire images or were false positives. Figures 22 and 23 show the differences in inference performance between the models.



Figure 22. YOLOv4 model correctly inferred small fires (**a**) and the DetectNetV2 model did not (**b**).



Figure 23. YOLOv4 model does not make false detections for dark environments other than fire (**a**) and DetectNetV2 model detects dark areas as smoke (**b**).

We observed that the YOLOv4 model spotted the initial fire with the fewest false detections, and we learned to utilize resnet50, resnet101, and cspdarknet19 instead of the YOLOv4 model's backbone, resnet18. Table 3 lists the performance evaluation variations when the backbones are different.

Table 3. Evaluation of the detection model according to the backbone of the YOLOv4 model.

Backbone	Model Parameters	mAP	Retrain Model Parameters	mAP	Retrain/Model
resnet18	11,200,458	0.90798	3,659,191	0.9088	0.10506
resnet50	85,346,559	0.90798	22,902,807	0.90792	0.26835
resnet101	122,286,335	0.90673	3,000,711	0.90365	0.02453
cspdarknet19	53,444,895	0.9062	38,253,879	0.90847	0.71576

For ResNet, we can see that the size of the pre-trained model varies greatly as the number of layers increases from 18 to 101. However, after pruning before relearning, Resnet101, which had the largest number of layers, showed fewer parameters than the other backbones. The pruning process of the model optimizes the model to suit reasoning in edge devices such as embedded devices by reducing unnecessary parameters while maintaining the model's performance. As before, when we deduced small initial fire or general fire images apart from the mAP evaluated from the dataset, the sensitivity of the fire was detected depending on the backbone. Figure 24 shows the results.

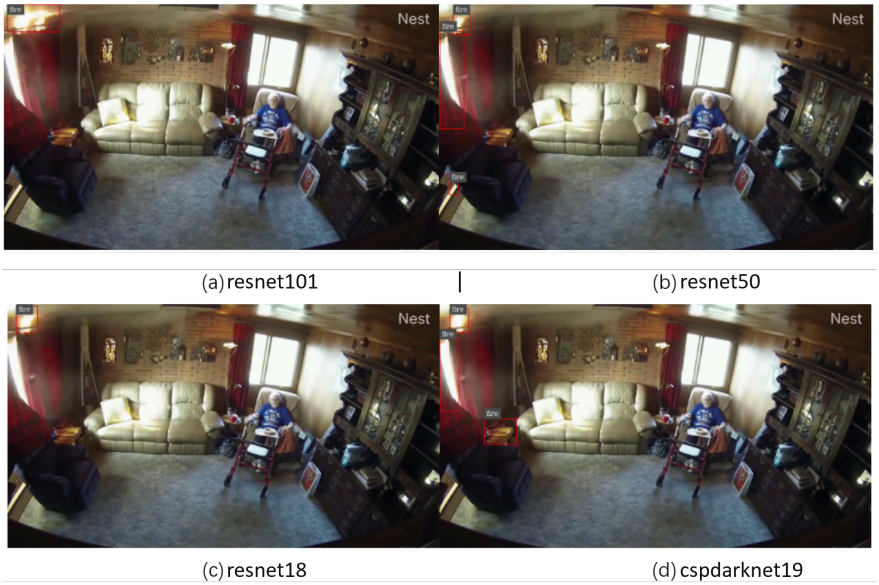


Figure 24. Real-world fire inference depends on the backbone of the YOLOv4 model.

Here, we can see that it is not necessarily good to sensitively detect fires. Resnet50 and cspdarknet19 had good detection from the beginning of the fire, but false detection occurred frequently. On the other hand, in the case of resnet101, there was little error detection, but it did not detect small fires in their initial state as quickly as in other models. In the case of resnet18, it was confirmed that the initial state of fire was detected well, and the error detection was very low. We tested a very small fire. Figure 25 shows the result of inferring the image of the flame, which appears to be approximately 20 × 20 pixels, by emitting a small ember at a point approximately 35m away from the camera using resnet18 as a backbone [22].



Figure 25. Inference for very small fires with the YOLOv4+resnet18 model.

3.4. Post-Processing

Even if the YOLOv4+resnet18 model was used, it was not possible to completely remove false detections in scientific practice because of the nature of the approach.

As can be seen in Figure 26, false detection by simple colors sometimes occurred, and to solve this problem, a post-processing process was added to inspect the physical properties of the fire's time series. In the inferring element of the deepstream pipeline, the nvTracker element tracks each fire object for the inference of the model, calculates the variation in the size or shape of the detection bounding boxes of previous frames for tracked

fires, and determines that it has the characteristics of a flame when it exceeds a certain value, eliminating the case of finally detecting it as a fire for non-fire objects. Figure 27 shows a different perception of the actual fire and the part that has a similar color to the fire. A blue object is an object that was finally judged to be a fire during the post-processing process, and the red bounding box is suspected to be a flame during the post-processing stage; however, it is not a characteristic of the flame in its movement or shape.

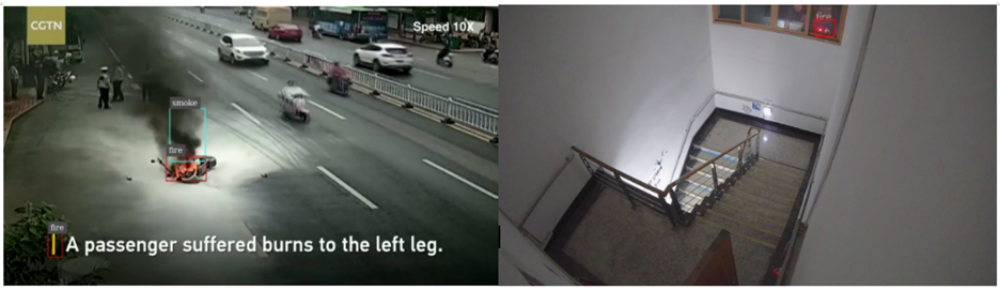


Figure 26. Examples of false positives in YOLOv4+resnet18 model inference.



Figure 27. Inference results after post-processing.

Despite the addition of this post-processing step, the detection performance for the original fire was unaffected, as shown in Figure 28.



Figure 28. Correctly inferring a very small fire in the initial state, even after post-processing is added.

4. Conclusions

In this study, we propose a model that automatically generates digital-twin-based field-optimal learning data for the early detection of fires. It was confirmed that the detection model learned from a dataset simulating possible fire conditions at the site created by combining field recording using RGB-D cameras and virtual fires was sufficient to detect small initial fires at the site with very high confidence. This experiment confirmed that a fire detection model based on virtual fire synthesis data has great potential for the rapid detection of initial fires. In addition, we developed a system that can detect fires at the site early with little error by overcoming scientific practices that inevitably occur in the process of learning models optimized for the site through post-processing.

In addition, there are some limitations, and future work can be developed based on the following:

- Virtual fires were simulated with the actual field as the backdrop, but detailed descriptions such as reflections of fire or background elements burning and turning into ash were not addressed. However, these aspects may not significantly contribute to fire data for early detection.
- Additionally, since the background of the training images is constructed from recorded video data, it is crucial to record a diverse range of scenarios that could occur in the actual field to achieve significantly improved performance.
- Besides that, a moving camera is also in our plan for future work; this kind of camera will change the angle as well as the view based on the detected fire. This implementation requires more research and experimentation; once performed, it will improve dramatically in real time.
- This proposed method is trained and tested on NvidiaTAO, which has a lack of supporting models. Therefore, other state-of-the-art computer vision models should be considered for testing, such as YOLOv8.

Ultimately, this undertaking not only adds value to the realm of research, but also plays a pivotal role in advancing the creation of a sophisticated digital twin dedicated to fire evacuation, wherein the central focus lies on the incorporation of early fire detection as the fundamental technique.

Author Contributions: Conceptualization, H.-K.L.; Methodology, H.-C.K.; Software, H.-C.K.; Validation, H.-K.L.; Formal analysis, H.-C.K.; Investigation, S.-H.L. and S.-Y.O.; Resources, S.-H.L. and S.-Y.O.; Writing—review and editing, H.-K.L.; Visualization, H.-C.K.; Supervision, S.-H.L. and S.-Y.O.; Project administration, S.-H.L. and S.-Y.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data relevant to the study are made available to the reviewers here: http://bit.ly/3wauwQo (accessed on 22 January 2024).

Acknowledgments: This work was supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) (No. 2022-0-00622, Digital Twin Testbed Establishment), and also supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2020R1F1A1069124 and No-2021R1A2C2013933).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fire Statistics Yearbook 2020. Sejong (Korea): National Fire Agency 119. Available online: https://www.nfds.go.kr/stat/general.do (accessed on 31 December 2020).
2. Smart City Korea. Available online: https://smartcity.go.kr/en/%ec%86%8c%ea%b0%9c/ (accessed on 17 October 2023).

3. Kim, H.; Lee, S.H.; Ok, S.Y. Early Fire Detection System by Synthetic Dataset Automatic Generation Model Based on Digital Twin. *J. Korea Multimed. Soc.* **2023**, *26*, 887–897. [CrossRef]
4. Sepasgozar, S.M.E. Differentiating Digital Twin from Digital Shadow: Elucidating a Paradigm Shift to Expedite a Smart, Sustainable Built Environment. *Buildings* **2021**, *11*, 151. [CrossRef]
5. Qiu, X.; Wei, Y.; Li, N.; Guo, A.; Zhang, E.; Li, C.; Peng, Y.; Wei, J.; Zang, Z. Development of an early warning fire detection system based on a laser spectroscopic carbon monoxide sensor using a 32-bit system-on-chip. *Infrared Phys. Technol.* **2019**, *96*, 44–51. [CrossRef]
6. Li, Y.; Yu, L.; Zheng, C.; Ma, Z.; Yang, S.; Song, F.; Zheng, K.; Ye, W.; Zhang, Y.; Wang, Y.; et al. Development and field deployment of a mid-infrared CO and CO_2 dual-gas sensor system for early fire detection and location. *Spectrochim. Acta Part Mol. Biomol. Spectrosc.* **2022**, *270*, 120834. [CrossRef] [PubMed]
7. Chen, S.; Ren, J.; Yan, Y.; Sun, M.; Hu, F.; Zhao, H. Multi-sourced sensing and support vector machine classification for effective detection of fire hazard in early stage. *Comput. Electr. Eng.* **2022**, *101*, 108046. [CrossRef]
8. Fuller, A.; Fan, Z.; Day, C.; Barlow, C. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access* **2020**, *8*, 108952–108971. [CrossRef]
9. Misuk, L.; Kim, E. A Study on the Disaster Safety Management Method of Underground Lifelines based on Digital Twin Technology. *Commun. Korean Inst. Inf. Sci. Eng.* **2021**, *39*, 16–24.
10. Zohdi, T. A machine-learning framework for rapid adaptive digital-twin based fire-propagation simulation in complex environments. *Comput. Methods Appl. Mech. Eng.* **2020**, *363*, 112907. [CrossRef]
11. Zohdi, T. A digital twin framework for machine learning optimization of aerial fire fighting and pilot safety. *Comput. Methods Appl. Mech. Eng.* **2021**, *373*, 113446. [CrossRef]
12. Pincott, J.; Tien, P.W.; Wei, S.; Calautit, J.K. Development and evaluation of a vision-based transfer learning approach for indoor fire and smoke detection. *Build. Serv. Eng. Res. Technol.* **2022**, *43*, 319–332. [CrossRef]
13. Abdusalomov, A.; Baratov, N.; Kutlimuratov, A.; Whangbo, T.K. An improvement of the fire detection and classification method using YOLOv3 for surveillance systems. *Sensors* **2021**, *21*, 6519. [CrossRef] [PubMed]
14. Yazdi, A.; Qin, H.; Jordan, C.B.; Yang, L.; Yan, F. Nemo: an open-source transformer-supercharged benchmark for fine-grained wildfire smoke detection. *Remote Sens.* **2022**, *14*, 3979. [CrossRef]
15. Kim, J.; Lee, C.; Park, S.; Lee, J.; Hong, C. Development of Fire Detection Model for Underground Utility Facilities Using Deep Learning: Training Data Supplement and Bias Optimization. *J. Korean Soc. Ind. Acad. Technol.* **2020**, *21*, 320–330. [CrossRef]
16. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. *arXiv* **2014**, arXiv:1409.4842.
17. Wu, D.; Wang, Y.; Xia, S.T.; Bailey, J.; Ma, X. Skip Connections Matter: On the Transferability of Adversarial Examples Generated with ResNets. *arXiv* **2020**, arXiv:2002.05990.
18. Liau, H.; Yamini, N.; Wong, Y. Fire SSD: Wide Fire Modules based Single Shot Detector on Edge Device. *arXiv* **2018**, arXiv:1806.05363.
19. Thomson, W.; Bhowmik, N.; Breckon, T.P. Efficient and Compact Convolutional Neural Network Architectures for Non-temporal Real-time Fire Detection. *arXiv* **2020**, arXiv:2010.08833.
20. GitHub-OlafenwaMoses/FireNET: A Deep Learning Model for Detecting Fire in Video and Camera Streams—github.com. Available online: https://github.com/OlafenwaMoses/FireNET (accessed on 17 October 2023).
21. Open Images Pre-trained Object Detection. 2023. Available online: https://docs.nvidia.com/tao/tao-toolkit/text/model_zoo/cv_models/open_images/open_images_pt_object_detection.html (accessed on 22 January 2024).
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.