Object Detection of Fire Safety Equipment in Images and Videos Using Yolov5 Neural Network

H. Bayer[1] and A. Aziz[1]

[1]Chair of Computing in Engineering, Ruhr University Bochum, Universitaetsstraße 150, 44801 Bochum, Germany

E-Mails: Hakan.Bayer@ruhr-uni-bochum.de

Abstract: The whole lifecycle of building information modeling (BIM) has great potential and intersections with other emerging technologies such as Artificial Intelligence (AI). Depending on the building's life cycle phase, there are many specific AI applications to use and possibilities to investigate. In particular, facility managers are recognizing the value of AI for different maintenance tasks, especially for the field of fire safety management. Building fire safety documentation is not just for newly completed buildings but also for existing structures that have not been digitized yet. Furthermore, this documentation is usually required to be updated due to recurring maintenance work, relocations, or system changes of fire safety equipment (FSE). However, performing specific FSE inspections in the traditional way and documentation is time-consuming and error prone. The computer vision abilities in the analysis of images and videos can provide important information on the current condition of the FSE. This study investigates machine learning and computer vision methods to provide an overview of state-of-the-art detection algorithms as a first step to improving the automation level of fire safety inspections. For performing the detection of FSE objects, You Only Look Once (YOLO) v5 was considered and utilized to train a custom neural model. Additionally, transfer learning is utilized from the Microsoft COCO dataset due to limitations of the amount of available data. To address this issue, an open-source dataset was combined with self-created images. The images were classified into four object classes: fire extinguisher, emergency call point, smoke detector and fire safety blanket. Moreover, image preprocessing and augmentation techniques were applied to increase the variability of the dataset. Based on selection criteria, the pre-trained Yolov5 model was utilized and trained on different datasets. The results show a significant detection accuracy in images and live videos. In future work, the concept of this research will be extended to transfer the FSE object directly to a BIM model after detection.

Keywords: Machine Learning, Object Detection, Computer Vision, Fire Safety Equipment

1 Introduction

1.1 Problem Statement

Building Information Modeling (BIM) is enjoying increasing popularity and acceptance in the construction sector due to its numerous advantages. It serves as a shared knowledge repository for information about a facility, providing a solid foundation for decisions made throughout the building's life cycle. BIM is also an interactive computerized database that all building stakeholders involved in the design, construction, and operation phase of a facility can access. The owners, architects, engineers, contractors, and facility managers are among the stakeholders. Technical experts such as fire safety managers (FSM) are also usually involved in any construction project by ensuring and documenting fire safety aspects. By using the BIM method, an FSM can locate and share important information regarding fire protection equipment in a BIM model. Despite the use of the BIM method, inspections are still carried out manually with checklists and transferred to a BIM model afterwards. However, this documentation of fire safety equipment (FSE) is still very error-prone, time-consuming, and expensive. Therefore, this paper investigates possible methods to automatically extract the necessary information, such as the presence of FSE in images by using computer vision.

1.2 Motivation

The integration of artificial intelligence (AI) with BIM has only recently begun, and the combination of these two powerful technologies is sure to grow in the future. It will help increase productivity in construction projects. Computer vision, especially object recognition using deep learning, has had limited research in this area. Nevertheless, a few researchers have addressed object detection scenarios in the construction field using neural networks. For instance, in [1] the authors adapt a deep convolutional neural network approach using Mask R-CNN to automatically recognize and segment building objects with arbitrary shapes from images. The segmented objects are further geometrically processed and fitted to construct surface geometries and to be defined in the Industry Foundation Classes (IFC) data format. Also, in [2] one part of the study focuses on automatic semantic segmentation of building interiors from images using deep learning methods. However, also in the field of fire safety management, researchers investigated the potential of computer vision. Corneli et al. [3] investigated the detection of fire safety assets such as fire extinguishers and emergency signs using a YOLOv2 [4] algorithm. Their results showed that fire extinguishers and emergency signs can reasonably detected. Unlike the forementioned study, this research paper concentrates on implementing the latest version of the YOLO algorithm family for FSE detection – YOLOv5. Also, additional FSE object classes were included. All training models were tested on a test dataset and evaluated based on performance metrics.

2 Technical Background

The You Only Look Once v5 - YOLOv5 [5] is a natural extension of the YOLOv3 [6] and the latest version of YOLO architecture series. The detection accuracy of this network model is high, and the inference speed is fast compared to the previous versions. Also, the size of the weight file of YOLOv5 target detection network model is small, indicating that YOLOv5 model is suitable for the deployment to embedded devices to implement real-time detection. The YOLOv5 architecture contains multiple varieties of pre-trained models, specifically named YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x, respectively. The main difference between them is that the number of feature extraction modules and convolution kernels is different at specific locations in the network. The size of models and the amount of model parameters in the four architectures also increase in turn but as the models get larger the accuracy gets better. On COCO dataset [7] evaluations, YOLOv5x model performs best compared to smaller versions. Despite YOLOv5x performs slightly better than YOLOv5l on COCO dataset evaluations, it is almost two times the size of YOLOv5l and with slower inference. Considering this detail, we selected YOLOv5l for this study. The three primary pieces of the YOLOv5 architecture are the backbone for feature extraction, the head for feature fusion, and the output for object detection. The backbone network uses Darknet which is a Convolutional Neural Network (CNN) that collects and forms image features at various granularities, also includes cross stage partial network (CSPNet) [8] into its architecture, resulting in the CSPDarknet architecture. CSPDarknet handles repeating gradient information in long backbones and integrates gradient change into feature map, which speeds up inference, improves accuracy, and decreases model size by lowering parameters. The head is made up of layers that aggregate image characteristics before sending them to detection algorithm. To improve information flow, YOLOv5 uses a path aggregation network (PANet) [9] as the head. PANet uses a new feature pyramid network (FPN) topology with an improved bottom-up approach to improve low-level feature propagation. Simultaneously, adaptive feature pooling, which connects the feature grid to all feature levels, is employed to ensure that meaningful information from each feature level reaches the next subnetwork. PANet improves the use of precise localization signals in lower layers, which can significantly improve the object's localization accuracy. Finally, the output generates three distinct sizes of feature maps allowing the model to detect small, medium, and large objects.

3 Methodology

3.1 Data Acquisition and Annotation

To ensure that our models can detect different types of fire safety equipment detection tasks, we have collected self-made images of different types of buildings (university buildings, student dormitories, etc.) in Germany. Additionally, the open source FireNet dataset from University College

London [10] as used to enrich the self-made dataset. For the self-made images a mobile camera was used to capture the images with resolution of 12MP. After that we combined the self-made images with the FireNet dataset and manually separated the images into different class folders. This dataset folder contains a total of 841 images showing FSE objects such as fire extinguisher, emergency call point, smoke detector and fire safety blanket. Since the resolution of the FireNet dataset images were lower, after combining the images, the average resolution became ~9.1MP with a mean size of 2988x4032. From the main fire safety equipment dataset containing 841 images, we separated 83 images for testing purpose and the remained was used to creat two dataset: Dataset_L containing 758 images (606 train + 152 val.) and Dataset_S containing 590 images (465 train + 125 val.). Each dataset contains 80% training and 20% validation set. To label the fire safety equipment dataset we used online open-source annotation tool called Roboflow. After each annotation Roboflow automatically generates a YAML config file and one text file per jpeg image file. The text file contains the location information of the bounding box which is the center location of the rectangle (x, y), the width-height of the rectangle, and the object class in numbers.

3.2 Data Preprocessing and Augmention

After annotating all the datasets, we applied several preprocessing and augmentation techniques to reduce the training time and increase the model performance. To reduce the training time, we downsized high-resolution images to 640x640 pixels which is the default input size of YOLOv5. Two different resizing techniques were applied using the Roboflow annotation tool. In the first method, we downsized the image to 640x640 while preserving the aspect ratio and filling the padding with black pixels. In the second method, we downsized the image to 640x640 while preserving the aspect ratio and filling the padding with the reflection of the image content. To compare the effect of this preprocessing method, we also kept the images with their original size which varies based on the image. To avoid overfitting and improve our model performance, it is important to use data augmentation techniques. With each training batch, YOLOv5 passes training data through a data loader, which augments the data based on the selected hyperparameters. The data loader applies transformations to the image, such as geometric transformations like rotation, scaling, image translation and flip translation, photometric transformations like hue saturation value (hsv) and shear, image occlusion techniques like Mixup and Mosaic data augmentation. Mixup data augmentation generates weighted combinations of random image pairs from the training data. Whereas Mosaic data augmentation is used to create a new image by combining 4 training images in specific ratios into one image. As for our problem, we chose the data augmentation techniques considering different inner building conditions such as the lightening condition and the camera perspective. After preprocessing and augmentation of the images, we generated 8 different datasets using Dataset_S and Dataset_L as shown in the Table 1.

Table 1: Applied image preprocessing and augmentation parameters on different models

Model	Dataset	Preprocessing	Augmentation
Model_S_1	Dataset_S_1	Resize to 640x640 (Fit (black edges))	hsv, translate, scale, mosaic
Model_S_2	Dataset_S_2	Resize to 640x640 (Fit (reflect edges))	hsv, translate, scale, mosaic
Model_S_3	Dataset_S_3	Original size	hsv, translate, scale, mosaic
Model_L_1	Dataset_L_1	Resize to 640x640 (Fit (black edges))	hsv, translate, scale, mosaic
Model_L_2	Dataset_L_2	Resize to 640x640 (Fit (reflect edges))	hsv, translate, scale, mosaic
Model_L_3	Dataset_L_3	Original size	hsv, translate, scale, mosaic
Model_L_4	Dataset_L_4	Original size	hsv, translate, scale
Model_L_5	Dataset_L_5	Original size	None

4 Training and Evaluation of The Neural Network

After creating the datasets as shown in Table 1, we trained our models using Yolov5l coco pretrained weight with fixed input image size as 640x640, batch size of 24, with 300 epoch and learning rate of 0.01. For training the models, Google Colab Pro was utilized. Google Colab Pro runs on Ubuntu 18.04.3 and is embedded with Intel(R) Xeon(R) CPU 2.00GHz, GPU of NVIDIA Tesla T4 (16GB) and RAM of 24GB. Furthermore, we used PyTorch 1.11.0, CUDA 11.3 and Python 3.7.13. While training the model, YOLOv5 saves the best and the last trained weight after each epoch. The best weight is selected based on a fitness function. This function is a weighted combination of mAP@0.5 and mAP@0.5:0.95 which gives more weights on the mAP@0.5:0.95 side. The mAP@0.5:0.95 is also the primary metric in the COCO object detection challenge. Therefore, we preferred to use mAP@0.5:0.95 for model evaluation and selection.

5 Object Detection Results and Discussion

Table 2 shows the validation and test results of each model on all of the classes. Here it can be seen that the validation mAP@0.5 values for all models are above 91% and validation mAP@0.5:0.95 values for all models are above 80%. The top three models based on validation mAP@0.5:0.95 are respectively, Model_S_3 with 89.5%, Model_L_3 with 89.3% and Model_S_1 with 87.4%. When we look at the test metrics, we see that for all the models the test mAP@0.5 values are above 89% and the test mAP@0.5:0.95 values are above 74%. Top three models based on Test mAP@0.5:0.95 are respectively Model_L_3 with 80.1%, Model_S_3 with 79.7%, Model_L_4 with 77.1% accuracy. Model_L_3 has 0.4% better mAP@0.5:0.95 than Model_S_3 on the test dataset. Model_L_3 has higher precision than Model_L_4 in both validation and test mAP@0.5:0.95 values. This difference is due to the additional mosaic data augmentation applied on Dataset_L_3. Based on validation and test mAP@0.5:0.95, Model_L_5 performs the worst with 80.6% validation and 74.3% test precision. Model_L_3 performs 8.7% better on validation and 5.8% better at mAP@0.5:0.95 compared to Model_L_5. These results indicate that augmentation techniques are important for a proper model performance.

Table 2: Validation and test mAP results of the models

Model	Class	Validation mAP@0.5	Validation mAP@0.5:0.95	Test mAP@0.5	Test mAP@0.5:0.95
Model_S_1	all	97.3	87.4	91.2	75.9
Model_S_2	all	94.9	84	89.5	76.6
Model_S_3	all	98.4	89.5	91.7	79.7
Model_L_1	all	96.2	85.6	90.3	74.4
Model_L_2	all	95.8	85.9	91.9	76
Model_L_3	all	96.6	89.3	92.5	80.1
Model_L_4	all	94.5	86.8	89.4	77.1
Model_L_5	all	91.5	80.6	89.8	74.3

When we look at the averaged validation mAP@0.5:0.95 results of individual classes for the models we see that the fire safety blankets can be detected with 91%, fire extinguisher with 84%, call point with 73%, and detector with 73% precision. The averaged test mAP@0.5:0.95 results of individual classes for the models are for fire safety blanket with 86%, fire extinguisher with 81%, call point with

80%, and detector with 61% precision. Since we used the large version of YOLOv5 for all the datasets, we didn't observe considerable difference in object detection speeds. For the test images, the average detection speed is 43.2 FPS and for the live test video, the average detection speed is 51.5 FPS. Since the camera had a very high resolution, even the objects far from the camera could be successfully detected in a live test video. The training time of a model depends on the hardware and the preprocessing steps as well. We observed that the training duration is affected by the preprocessing step. As an example, Dataset_S_1 was downsized to 640x640, and Dataset_S_3 was kept as its original size but only downsized to 640x640 after running the algorithm. For Model_S_1 the training took 1.94 hours and for Model_S_3 8.18 hours. For Model_L_3 the training took 5 hours longer than Model_S_3. This means that the training duration is affected by preprocessing steps such as downsizing the images beforehand and the number of trained images. Even though Model_L_3 has a slightly higher test mAP@0.5:0.95 value compared to Model_S_3, the number of data it requires to train, and the training duration is much longer. To see how the models perform, object detection test results of Model_S_3 and Model_L_5 is shown in Figure 1. It is noticeable that Model_S_3 provides more precise bounding boxes with higher confidence compared to Model_L_5.

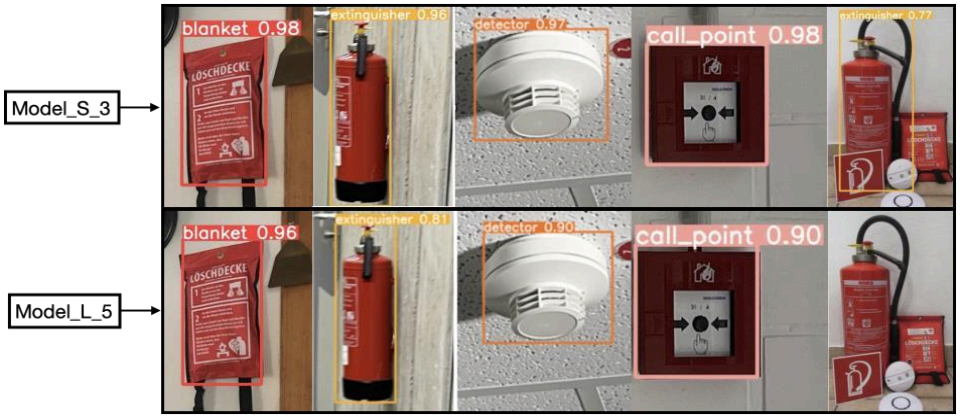

Figure 1: The detection results of model_S_3 and model_L_5 on the test dataset. The images are zoomed in for a clearer view of the bounding box and the precision.

6 Conclusion and Future Work

In this research, we utilized pretrained YOLOv5 large neural network model for fire safety equipment detection. We collected self-made images from different building types in Germany combined with an open-source dataset. We used preprocessing methods to resize the images and applied augmentation techniques to increase the performance of the models. The results showed that preprocessing techniques reduced the training time and applying augmentation techniques increased the model performances. Based on the validation and test results we concluded that YOLOv5 can successfully detect fire safety equipment with up to 80.1% of test mAP@0.5:0.95 and

up to 89.5% of validation mAP@0.5:0.95. The live video detection speed of 51.5 FPS and the performance of YOLOv5 proved that this algorithm can contribute for automated inspections in the future.

References

[1] H. Ying and S. Lee, “A mask R-CNN based approach to automatically construct as-is IFC BIM objects from digital images,” Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC), 2019.

[2] E. Gülch and L. Obrock, “Automated semantic modelling of building interiors from images and derived point clouds based on Deep Learning Methods,” The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XLIII-B2-2020, pp. 421–426, 2020.

[3] A. Corneli, B. Naticchia, M. Vaccarini, F. Bosché, and A. Carbonari, “Training of Yolo Neural Network for the detection of fire emergency assets,” Proceedings of the 37th International Symposium on Automation and Robotics in Construction (ISARC), 2020.

[4] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[5] Ultralytics, YOLOv5 [Online]. Available: https://github.com/ultralytics/yolov5. [Accessed: 26-May2022].

[6] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” arXiv.org, 08-Apr-2018. [Online]. Available: https://arxiv.org/abs/1804.02767. [Accessed: 27-May-2022].

[7] “Common objects in context,” COCO. [Online]. Available: https://cocodataset.org/#home. [Accessed: 31-May-2022].

[8] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, “CSPNet: A new backbone that can enhance learning capability of CNN,” 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020.

[9] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, “Panet: Few-shot image semantic segmentation with prototype alignment,” 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[10] J. Boehm, F. Panella, and V. Melatti, “FireNet,” figshare, 31-Jul-2019. [Online]. Available: https://rdr.ucl.ac.uk/articles/dataset/FireNet/9137798#:~:text=Fire-Net%20is%20an%20open%20ML,c lassification%20scheme%20to%20name%20objects.