Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

Intelligent Detection of Mechanical, Electrical, and Plumbing (MEP) Metrics Based on 2D Floor Plans

TARANDEEP SINGH MANDHIRATTA[1], ANK ZAMAN[2], (Member, IEEE) and ABDUL-RAHMAN MAWLOOD-YUNIS[3], (Member, IEEE)

[1]Dept. of Physics & Computer Science, Wilfrid Laurier University (WLU), Waterloo, ON, Canada (mand4710@mylaurier.ca)
[2]Dept. of Physics & Computer Science, Wilfrid Laurier University (WLU), Waterloo, ON, Canada (e-mail: azaman@wlu.ca)
[3]Dept. of Physics & Computer Science, Wilfrid Laurier University (WLU), Waterloo, ON, Canada (e-mail: amawloodyunis@wlu.ca)

Corresponding author: ANK Zaman (e-mail: azaman@wlu.ca).

ABSTRACT This research developed a neural network-based model to extract various information from 2D floor plans. We detect lighting symbols, identify the appropriate type of light, and extract the associated texts with lights. The study aims to enable efficient floor designing and determining the number and type of lights needed per floor, i.e., allow efficient design and estimate the power requirement of the floor plan. The model was developed using Mask RCNN as the base. The images were annotated and converted into a Coco data format for training the model. The model achieved bbox_mAP and segm_mAP values of 0.7596 and 0.7111, respectively. It also performed well at different IoU thresholds, i.e., with bbox_mAP 50 and segm_mAP 75 values of 0.9850 and 0.9219, respectively. The developed model will help various industries, such as architecture and construction, to improve design time and create efficient workflows by automatically detecting Mechanical, Electrical, and Plumbing (MEP) objects from floor plans, and it is the first step towards building tools that will help energy-efficient building design.

INDEX TERMS Intelligent detection, MEP metrices, 2D floor plans, Mask RCNN, Coco data format, Workflow efficiency, Energy efficiency, Engineering design process.

I. INTRODUCTION

TODAY'S technology has made collecting and processing information in indoor environments possible. This has positively impacted facilities management, indoor localization, and indoor data models, but it has also increased the demand for creating actual databases of indoor information.

One area still underdeveloped [1], [2] is the analysis of the Mechanical, Electrical, and Plumbing (MEP) matrices of buildings, which is crucial for efficient engineering design processes. Various approaches have been used to analyze MEP matrices of buildings, including those that use 3D laser scanning, CAD plans, architectural sketches, and mobile applications. This study focuses on using 2D floor plans, as they provide the most fundamental source of information for existing buildings.

Recent efforts have been put into using machine learning models for extracting and labelling data from floor plans. However, this is a challenging problem, as the notation of different floor plans varies significantly according to the institutions and the design offices. Furthermore, floor plans archived as complex, fuzzy architectural drawings usually characterize raster images.

This research work aims to intelligently collect MEP-related metrics from floor plans to aid in improving the efficiency of the engineering design process. As a first step, we developed a neural network model to collect three types of information from 2D floor plans: a) detect lighting symbols, b) identify the appropriate type of light, and c) extract the associated texts with lights. This work converted PDF files of 2D floor plans into image format to achieve these objectives. We annotated the images manually. This is because we use a private dataset provided by the sponsor, and the data is not annotated. Once the annotations were complete, the data was converted to coco data format. Each image and its coco dataset were sliced into smaller components to reduce storage space. The images were sliced with slice parameters such that each image was sliced into 1000x1000 grids with an overlap ratio of 0.2. After the data preprocessing step, Mask RCNN [3], [4] was used as a base to develop a neural network model. The configuration files were changed to optimize the model for this use case, and hyperparameter tuning was also done. Initialization, forward propagation compute loss, and backpropagation were performed during training. The model was then trained and validated using a dataset to improve its accuracy. The model was used to detect the lighting symbols in the floor plan. Once the algorithm detects the lighting symbols, it finds the coordinates of the bounding boxes and extracts each electrical component detected, saving it in a file for use in the next step. The next step is to loop through the cropped lighting symbols saved in a file and extract the appropriate text that informs what kind of electrical component it is. Once the type of electrical component is identified, the power consumption for lighting components of the whole floor plan can be estimated by combining the results. Finally, TensorBoard [5] was used to visualize the findings, which improved the model even more. This research represents a significant improvement over the traditional method of intelligently detecting MEP symbols from floor plans. The developed model will help various industries, such as architecture and construction, improve design time and create efficient workflows by automatically detecting Mechanical, Electrical, and Plumbing (MEP) objects from floor plans; it is the first step towards building tools that will help energy-efficient building design.

The rest of the paper is organized as follows: section II surveys and presents recent related works, section III describes the technical details of the implementation and dataset used in the study, section IV describes the result and evaluation, and section V describes the concluding remarks of this paper.

II. LITERATURE REVIEW

In recent years, there has been a growing interest in using deep learning techniques to detect MEP (mechanical, electrical, and plumbing) metrics from 2D floor plans. Researchers have actively explored various approaches to extract components from architectural floor plans automatically. These floor plans, which architects create to provide a scaled representation of buildings, serve as a crucial source of information for MEP systems design and installation. These floor plans, created by architects, provide a scaled representation of the building and contain important details such as walls, doors, windows, stairs, fixtures, and labels that serve as critical information for MEP systems design and installation. For example, Figure 1 presents a floor plan for a residential unit's bathroom and kitchen areas [6].

Several studies have attempted to develop automated methods for detecting MEP systems from floor plans, utilizing machine learning algorithms such as Convolutional Neural Networks (CNNs) to classify symbols. The analysis of architectural floor plans has been a crucial area of research in computer vision. For instance, in 1992, Filipski & Flanderena [7] introduced GTX 5000, a commercially available tool that utilized neural networks for character and symbol detection, making it one of the first commercial applications of convolution image detection.

More recently, in 2017, Ruwanthika et al. [8] attempted to construct complete 3D models of houses for AR purposes, while Dodge et al. [9] applied Fully Convolution Networks (FCN) to analyze architectural floor plans with the goal of processing simplified plans frequently seen on real estate websites, which could not be effectively analyzed through conventional methods. The authors also compared traditional and statistical methods, utilizing Google Vision API to remove any text elements that could skew the results. Through their study, they demonstrated the precision with which one could identify the features of a floor plan. This breakthrough could serve as a building block for automated 3D modelling in the future. The renewed interest in neural networks for floor plan analysis can be attributed to significant advancements in deep learning and the availability of annotated data for training.

Automating the entire process of floor plan analysis without manual intervention is becoming more critical, especially with the increasing demand for smart homes and building management systems. As such, the following section provides an overview of the current techniques for extracting walls, detecting rooms, and performing preprocessing in this field, highlighting the difficulties encountered and the proposed solutions.

A. METHODS FOR EXTRACTION OF WALLS

The techniques used for wall extraction in architectural floor plans have advanced as the goals, difficulties, and technological advancements changed. The field has kept pace with the development of computer vision techniques and increased computing power.

1) Early Techniques for Separating Walls in Floor Plans

The first attempts to create systems for understanding architectural floor plans were driven by the need to convert large collections of drawings that were only available in printed or hand-drawn form into digital format. This task was often done manually, which could be time-consuming and expensive. It involved tracing the scanned image into CAD software either on a computer screen or using specialized tracing tables [10].

The beginning stages of solving the problem of digitizing architectural floor plans [10], [11], [12] only dealt with images with a set pattern of drawings, usually walls depicted as thick lines, and often were only able to handle horizontal and vertical lines. These early methods mainly focused on converting the walls into vectors using morphological operations and thinning the walls down to 1-pixel lines to obtain their vector representation. This traditional pipeline of line extraction, Skeletonization, and vectorization became a commonly used technique for analyzing technical drawings like topographic maps [13].

The earliest works aimed to convert floor plan sketches into CAD format efficiently. A study by Aoki et al. (1996) [11] relied on the use of thin and thick lines to distinguish internal and external walls. The approach involved extracting line segments and closed regions from a preprocessed image and then using pattern matching to identify the elements in the drawing.

Ahmed et al. [14] introduced an early automatic method for analyzing architectural floor plans that involved multiple stages, including information separation, structure analysis,

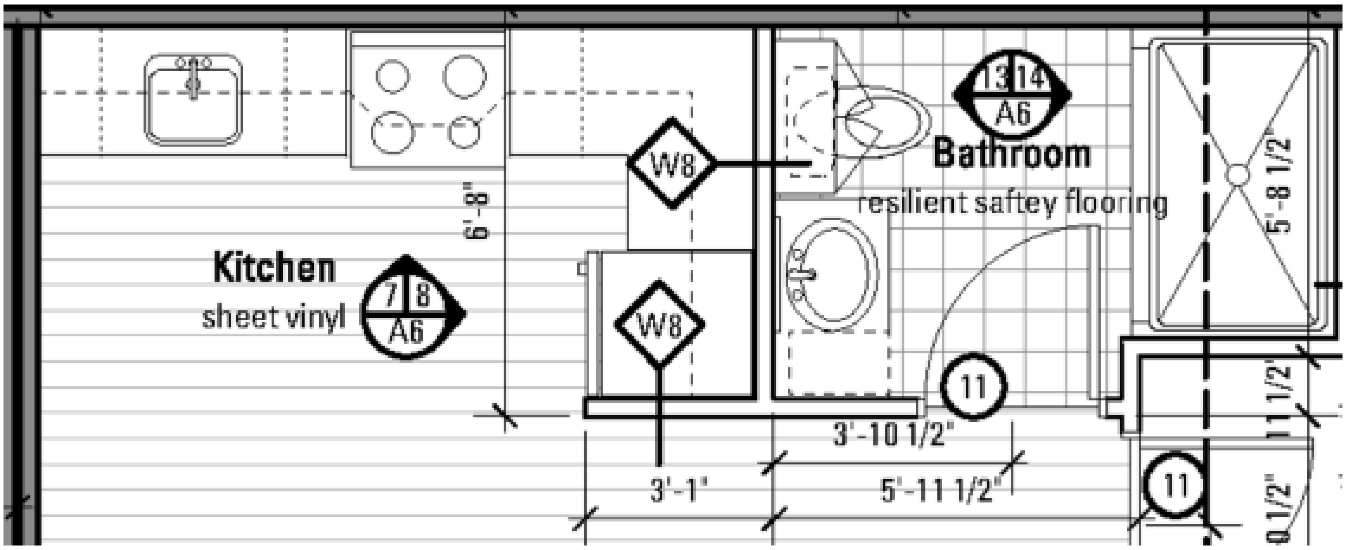


FIGURE 1. An architectural floor plan section, specifically a residential unit's kitchen and bathroom areas [6]

and semantic analysis. However, their wall extraction technique was based on the assumption that walls were drawn as the thickest lines in the image, which restricted its use to that specific drawing style.

2) Conventional Machine Learning Methods

Heras et al. [15] pointed out that architectural drawings have no universal notation. Walls can vary in representation, from a single line of varying widths to parallel lines or even hatching patterns, as shown in Figure 2. This diversity made traditional methods that rely on specific notations insufficient for wall extraction tasks.

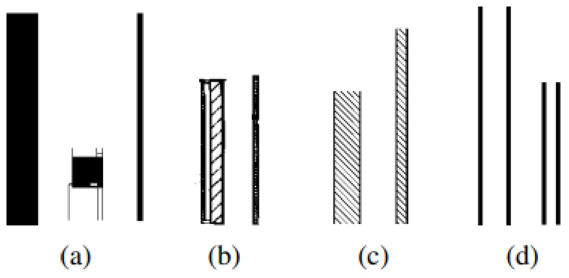


FIGURE 2. Various Wall Notations [16]

They [15] also introduced a groundbreaking machine-learning approach for wall extraction that can manage various notations, modelled on the traditional Bag-Of-Visual-Words model [17].

The authors in [18] note the main limitations of patch-based methods as being restricted to single notation training and requiring ground truth data from a diverse image set for the training process.

3) Methods Utilizing Convolution Neural Networks (CNN)

With the increasing computing power, CNNs have been widely adopted in computer vision and various fields, including architectural floor plan analysis. Liu et al. [19] used a simple CNN approach for heatmap prediction at the pixel level, as described in [20], to detect corners and crosspoints (referred to as "junctions") in floor plans. They trained their method using the LIFULL HOME dataset [21], which manually annotated 1,000-floor plan images. This dataset was large enough for deep learning.

While the CNN-based approaches achieved high quantitative results, qualitatively, they failed to guarantee wall connectivity, sometimes missing entire walls, rendering post-processing recovery impossible.

B. TECHNIQUES FOR IDENTIFYING ROOMS

Methods for detecting rooms focus on identifying and separating the various areas within a floor plan image, using information from the walls as a guide. Accuracy depends on correct and thorough wall extraction, as errors in wall identification can impact results.

1) Pixel-level Approaches

Pixel-based methods of room detection operate by analyzing a raster image of the walls. Koutamanis and Mitossi [12] proposed an early example of this method, which involved using fixed rules to fill gaps in a simplified version of the walls. However, this method was limited in its ability to handle complex cases or wall extraction errors. Another approach, proposed by Ryall et al. [22], involved using a "proximity field" to identify the farthest points from any wall as potential room centers. Its reliance on pixel properties limited this method, resulting in potential under-segmentation or over-segmentation of rooms.

2) Geometry-Driven Approaches

Geometry-based methods for room detection utilize a vector representation of the walls. Mace et al. [23] proposed an example of this method, which used a top-down polygon partitioning approach based on the idea that rooms should be nearly convex regions. This method filled gaps between walls by minimizing concavity in the resulting room polygon.

Other authors [10], [19], [24], [25], [26] have proposed methods for detecting loops in the polygon representation of walls, which are used to close gaps created by doors and windows. These methods often rely on prior detection of wall and symbol information with high precision, which is only feasible on datasets with low complexity or human assistance.

3) Hybrid Approaches

More advanced methods, such as the one proposed in [27], create a pixel image of the extracted walls and fill gaps in the wall structure using information from previous structural analysis steps.

Once the gaps are filled, the rooms are detected as connected components in the image. The hybrid methods utilize semantic and pixel information for room detection, balancing robustness and accuracy.

C. PREPROCESSING TECHNIQUES

Graphics/text separation is a common preprocessing step in many wall extraction methods [10], [15], [17], [18], [24] , [25], [27], [28]. This involves detecting and extracting text from the image to simplify it.

Fletcher and Kasturi [29] introduced a graphics/text separation technique in early wall extraction methods (Section II-A). The method relied on the distinct features of technical drawings to identify text characters. It filtered the image's connected components based on statistical histogram analysis of the component's area, aspect ratio, pixel density, and bounding box dimensions. This method was simple to implement and produced good results, but it could not detect text characters overlapping with other symbols.

Ahmed et al. [30] improved the graphics/text separation method introduced by Fletcher and Kasturi [29] by detecting overlapping text characters. The improved method inferred the location of missing text characters from surrounding detected characters. These methods relied on the assumption that there is sufficient text in the image to make the area of text characters the highest probability area measure among all connected components in the image.

One of the primary challenges in this area of research is the lack of a standardized symbol set for MEP systems. Different construction projects may use different symbols for the same MEP components. Developing a generalized model to detect all possible symbols for MEP systems across different projects is challenging.

Researchers have proposed several solutions to overcome this challenge, such as developing a symbol classification system that can learn from multiple symbol sets and using data augmentation techniques to generate synthetic data. Additionally, some researchers have proposed using an ensemble of models to increase the detection system's accuracy.

Overall, the research in this area shows great promise for developing intelligent detection systems for MEP metrics based on 2D floor plans. With the increasing availability of large-scale annotated data sets and advancements in deep learning techniques, future research is expected to continue to make significant contributions to this field.

III. METHODOLOGY

Engineering design has grown tremendously in recent years, with increasing demand for more efficient and accurate design processes. To address this challenge, researchers have turned to deep learning techniques to automate the collection of metrics related to mechanical, electrical, and plumbing (MEP) systems in building floor plans. This research project aims to enhance the efficiency of the engineering design process by developing an intelligent system capable of accurately identifying lighting symbols in a floor plan, determining the appropriate type of light, and extracting relevant text to estimate the power requirement of the floor plan. The project utilizes PDF files of 2D floor plans, which are converted into image format for analysis.

A neural network model is developed using a well-defined methodology to achieve these objectives, as illustrated in Figure 3. This methodology comprises a series of carefully designed steps, which this section will thoroughly explain.

A. DATA COLLECTION AND PREPROCESSING

Figure 4 presents the steps involve in data collection and preprocessing:

1) Data Collection

Developing a neural network model is a complex process that involves several steps, starting with data collection and preprocessing. The process begins with collecting the training data, as no existing dataset was available for the specific problem. In this case, the industry collaborator provided a set of 2D floor plans in PDF format for buildings with different configurations and layouts, which had a vast amount of floor plans.

2) PDF to Image Conversion

The next step in the data collection and preprocessing phase was to convert the PDF data to image format. The PDF files were too large to be used directly for training the neural network model. Therefore, it was necessary to convert them to image format. PyMuPDF [31] was used for the conversion, and the DPI was set to 300 based on trial and experimentation to balance the image size while still maintaining the clarity of the image.

3) Image Annotation

The third step in the methodology was manual image annotation using VGG Image Annotator [32] (VIA) and converting

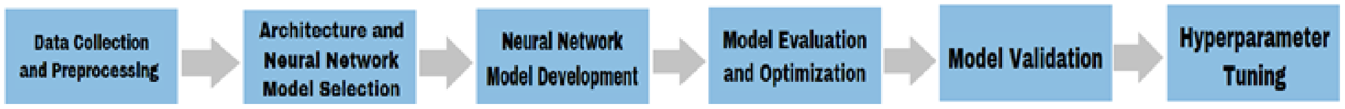

Data Collection and Preprocessing
Architecture and Neural Network Model Selection
Neural Network Model Development
Model Evaluation and Optimization
Model Validation
Hyperparameter Tuning

FIGURE 3. Neural Network Model Development Methodology

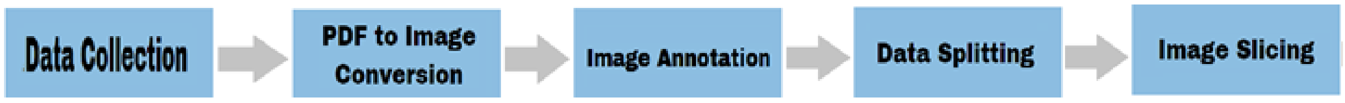


FIGURE 4. Data Collection and Preprocessing

it to coco data format. The images were manually annotated using the VGG Image Annotator, a popular open-source tool for annotating images. The annotations were made for each image in terms of the location of the lighting symbols and the corresponding text. The annotation data was then converted to the coco data format widely used for object detection tasks. Figure 5 represents a sample image that is annotated as part of the preprocessing steps.

4) Data Splitting

The fourth step in the methodology was to split the data into training, validation, and testing datasets. This was done to ensure that the model's performance could be evaluated on unseen data and to avoid over-fitting. This step is crucial in developing a robust and effective neural network model.

5) Image Slicing

Finally, image slicing was performed to slice images into smaller components to reduce training time for the model and improve efficiency. Figures 6 and 7 are showing an original image and it's sliced component respectively. Each image and its coco dataset (see in Figure 8) were sliced into smaller components. The images were sliced with slice parameters such that each image was sliced into 1000X1000 grids with an overlap ratio of 0.2. This allowed the model to be trained on smaller image components, which reduced the training time and improved efficiency.

B. ARCHITECTURE AND NEURAL NETWORK MODEL SELECTION

Selecting an appropriate architecture and neural network model is a critical step in developing an effective system for the intelligent detection of MEP metrics. In this study, the Mask R-CNN [3], [4] algorithm was chosen as the base algorithm for its high accuracy in object detection tasks and ability to detect and segment objects in an image.

Mask R-CNN [3], [4] is a popular deep-learning algorithm for object detection and image segmentation. It is highly accurate and can create a mask that outlines the exact pixels belonging to each object in an image. This ability to accurately segment images differentiates it from popular computer vision models such as Faster R-CNN [3], [4], YOLO [33], and SSD [34].

TABLE 1. Object Detection Model Comparison

Model	Advantages	Disadvantages
Mask R-CNN [3], [4]	Object detection and instance segmentation in a single model, with high accuracy, end-to-end training, open-source implementation	Computationally expensive, slower inference speed compared to other models
Faster R-CNN [33]	Good accuracy, faster inference speed compared to Mask R-CNN	Requires twostage detection, may miss small objects
YOLO [33]	Fast inference speed, real-time object detection	Lower accuracy compared to other models, may miss small objects
SSD [34]	Fast inference speed, real-time object detection, good accuracy for small objects	Lower accuracy compared to other models for larger objects

As shown in Table 1, Mask R-CNN offers several advantages over other models, including its ability to perform object detection and instance segmentation in a single model, achieving higher accuracy, and providing more detailed information about the objects in the image. However, it is important to note that Mask R-CNN is computationally expensive and slower than other models, such as Faster R-CNN and YOLO. It is due to its two-stage architecture, which involves more complex computations and higher memory requirements. Thus, for applications where real-time performance is critical, other models like Faster R-CNN and YOLO may be more suitable.

The choice of the model ultimately depends on the application's specific needs. While Mask R-CNN is slower and computationally expensive, its accuracy and segmentation capabilities make it the right choice for this project. In general, there is always a trade-off between accuracy and computational cost when selecting a model. Hence, it is crucial to carefully evaluate each model's pros and cons to determine which best fits the application's requirements.

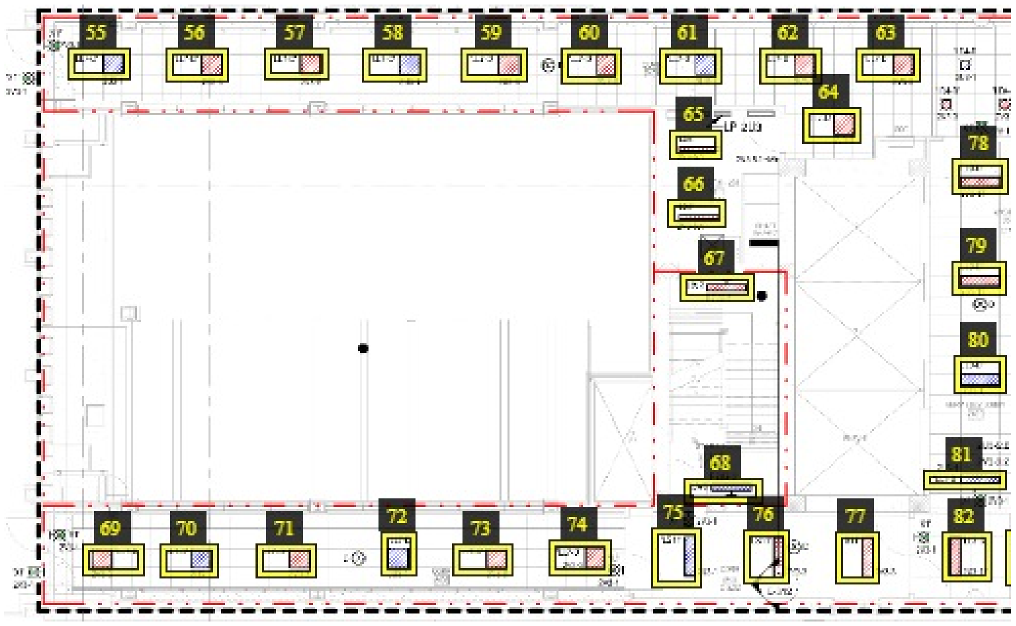


FIGURE 5. Image Annotation

C. NEURAL NETWORK MODEL DEVELOPMENT

The Neural Network Model Development phase aimed to train the Mask R-CNN algorithm to detect and segment MEP metrics. The process involved four stages:

1) Initialization

In this phase, the weights and biases of the neural network are randomly initialized to small values. This step is crucial as it helps avoid getting stuck in local optima.

2) Forward Propagation

During forward propagation, the neural network takes input data in the form of images and performs a forward pass through the network, producing an output. In this case, the output is the predicted bounding boxes, objectness scores, and masks for each object in the input image.

3) Compute Loss

When training a neural network for object detection, tracking how well the model performs is crucial. One way to do this is by using a loss function to measure the difference between the predicted and true outputs. As the network progresses through each training epoch, a loss value is calculated and printed for each batch. These values and other important information are stored as logs in log files.

Most importantly, the log includes information on the different components of the object detection task, such as the loss for the region proposal network classification, the loss for bounding box regression, and the accuracy for the classification task. These losses are used to monitor the training progress and adjust the model parameters.

4) Backpropagation

In the backpropagation phase, the loss is backpropagated through the network, and the gradients of the loss with respect to the weights and biases of each neuron are calculated. The specific backpropagation algorithm used in Mask RCNN is stochastic gradient descent with momentum. This algorithm updates the weights and biases of the network based on the gradients calculated during backpropagation. By iteratively adjusting the weights and biases in the direction that minimizes the loss, the network is able to improve its performance over time.

D. MODEL EVALUATION, OPTIMIZATION, VALIDATION AND HYPERPARAMETER TUNING

This research utilized the validation dataset to evaluate the model's accuracy and optimize its hyperparameters. The primary objective of this process was to improve the performance of the model by adjusting the hyperparameters, including the learning rate, evaluation interval, and number of epochs, among others.

Hyperparameter tuning is a crucial step in machine learning model building, as it enables us to find the best combination of hyperparameters that results in optimal performance. The hyperparameters were tuned using a trial-and-error process until the desired performance, as measured by Average Precision (AP) and Average Recall (AR), was achieved.

The model's performance was evaluated using the AP and AR metrics during optimization. These metrics are widely used in object detection tasks and provide a measure of the model's ability to detect and classify objects accurately. The AP metric measures the average precision of the model across a range of intersections over union (IoU) thresholds, while the

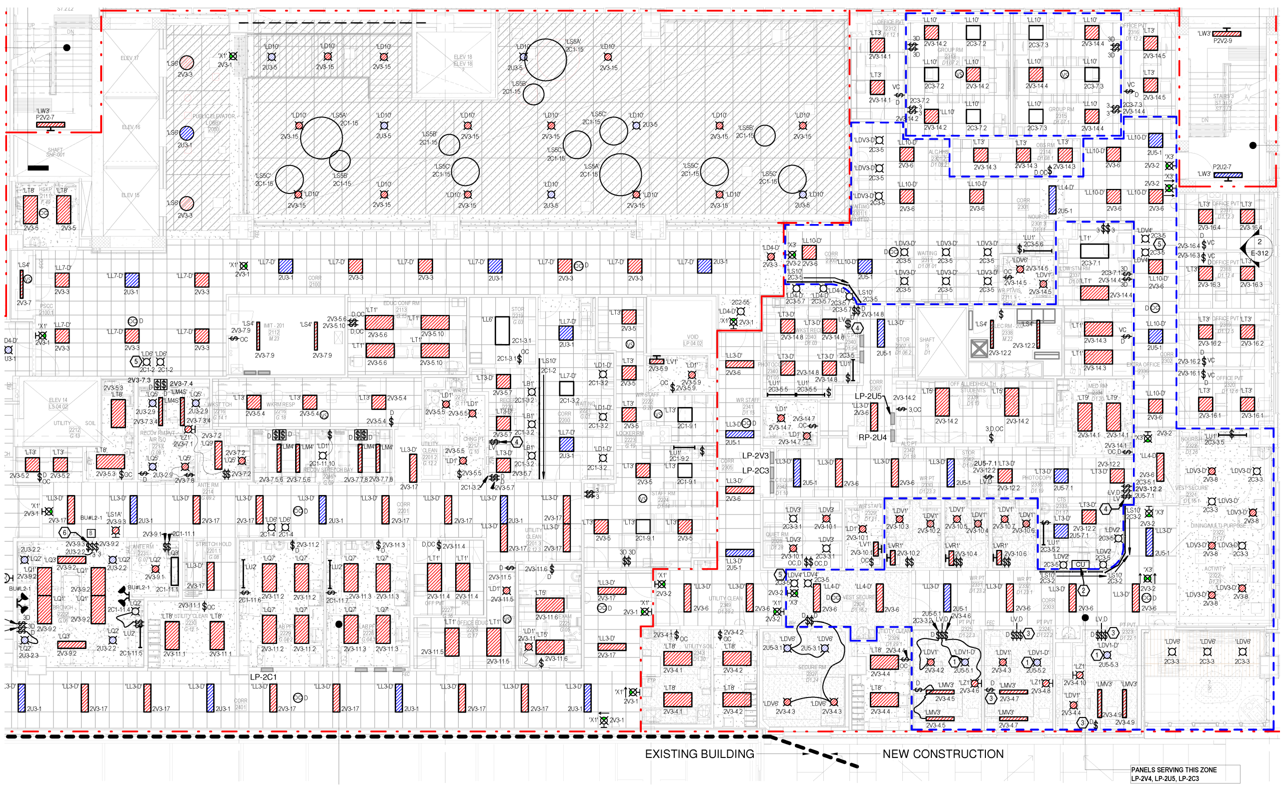


FIGURE 6. Original Unsliced Image

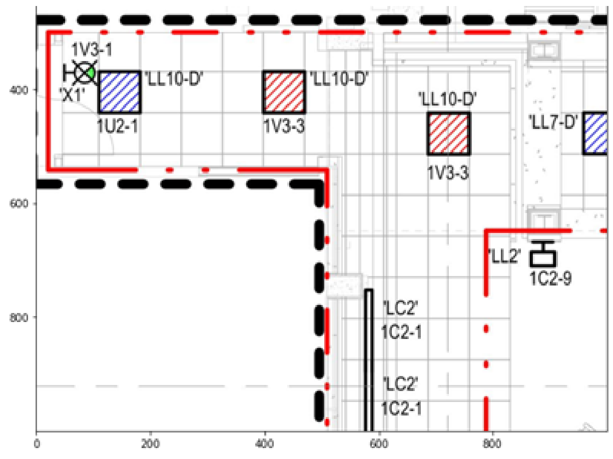


FIGURE 7. Sliced Component of the Original Image

AR metric measures the average recall of the model across a range of IoU thresholds.

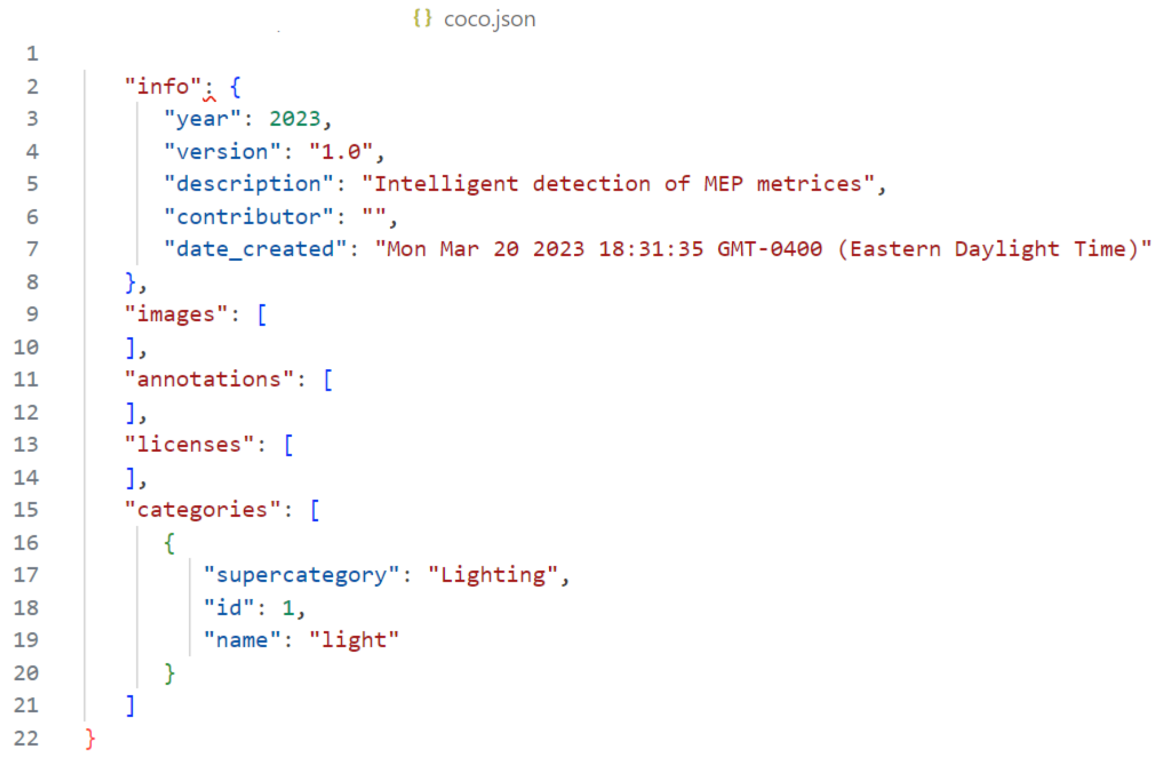
Once the hyperparameters were tuned, the optimized model was tested on a separate test dataset to evaluate its ability to accurately detect lighting symbols. This independent evaluation provided a robust assessment of the model's performance. The rigorous evaluation metrics used during this process ensured that the model's performance was optimal and could generalize to new data. Ultimately, this process led to the development of an intelligent lighting symbol detection model capable of accurately classifying lighting symbols. The final optimized model demonstrated excellent prediction ability, providing high confidence in its ability to be applied in real-world MEP detection scenarios.

IV. RESULTS AND ANALYSIS

This section presents the research findings and interprets them in the context of the research objectives. The collected data is analyzed comprehensively, and the patterns, relationships, and emerging trends are described. The implications of the findings are also discussed, and key insights that contribute to the understanding of the research problem are highlighted. This section aims to provide a clear and concise presentation of the results and their significance, enabling the reader to evaluate the research outcomes.

A. ANALYSIS OF MODEL PERFORMANCE DURING TRAINING

This subsection provides a comprehensive analysis of the model's performance during training. The data presented include the loss values, accuracy, and memory usage for each epoch, covering a span of 20 epochs. The loss values for each model component are provided, and their trends are analyzed to determine if any overfitting or underfitting occurs. Tensorboard was used as a powerful visualization tool to monitor and analyze the model's performance, providing an interactive dashboard that displays various metrics such as the loss function, accuracy, and learning rate over time.



```

    "info": {
        "year": 2023,
        "version": "1.0",
        "description": "Intelligent detection of MEP metrices",
        "contributor": "",
        "date_created": "Mon Mar 20 2023 18:31:35 GMT-0400 (Eastern Daylight Time)"
    },
    "images": [
    ],
    "annotations": [
    ],
    "licenses": [
    ],
    "categories": [
        {
            "supercategory": "Lighting",
            "id": 1,
            "name": "light"
        }
    ]
}
```

FIGURE 8. **COCO Data Format**

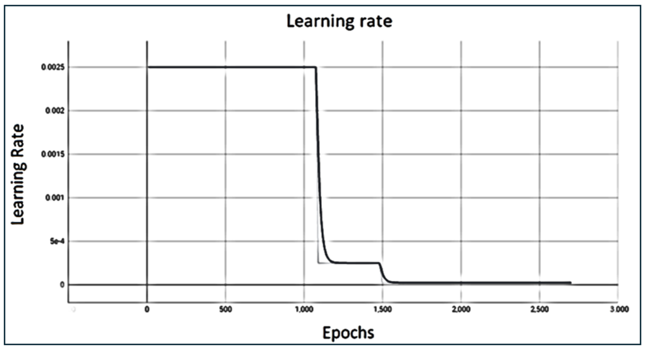


FIGURE 9. **Learning Rate Graph**

Figure 9 displays the learning rate graph generated by Tensorboard, which depicts the model's learning rate changes over time. The learning rate graph helps determine the appropriate learning rate for optimal performance.

In addition to the learning rate graph, the Figure 10(a) presents the accuracy of the model and Figures 10(b) to (g) are showing the training loss graphs generated by Tensorboard. These graphs show how the model's loss changes over time during training. The figures show that the training loss decreases steadily over time, indicating that the model effectively learns from the data and improves its performance.

Table 2 shows that the loss_rpn_cls, loss_rpn_bbox, loss_cls, and loss_bbox decrease significantly with each epoch, while the accuracy improves. The loss_mask, however, shows a slight decrease in the first epoch but stays relatively stable in the following epochs. The results indicate that the algorithm used in this research study is effective in detecting MEP metrics.

B. PERFORMANCE EVALUATION

The performance of the proposed intelligent detection model for MEP metrics was evaluated using the mean Average Precision (mAP) and mean Average Recall (mAR) metrics. The results of the evaluation show that the model achieved a mAP of 0.760 and a mAR of 0.810 for bounding box annotations with an IoU threshold of 0.50:0.95, and a maximum detection of 100.

For segmentation annotations, the model achieved a mAP of 0.711 and a mAR of 0.765 with the same IoU threshold and maximum detection (Figure 11). The model also achieved high accuracy for different object sizes, with a mAP of 0.700 for small objects, 0.662 for medium-sized objects, and 0.780 for large objects. The mAR results for different object sizes were 0.700, 0.725, and 0.830, respectively. The results demonstrate the effectiveness of the proposed model in accurately detecting MEP metrics.

C. EVALUATION RESULTS

The following subsection provides an analysis of the results obtained from the evaluation of the performance of MEP metrics detection. The results are presented in terms of detection accuracy metrics such as bbox_mAP, segm_mAP, and their

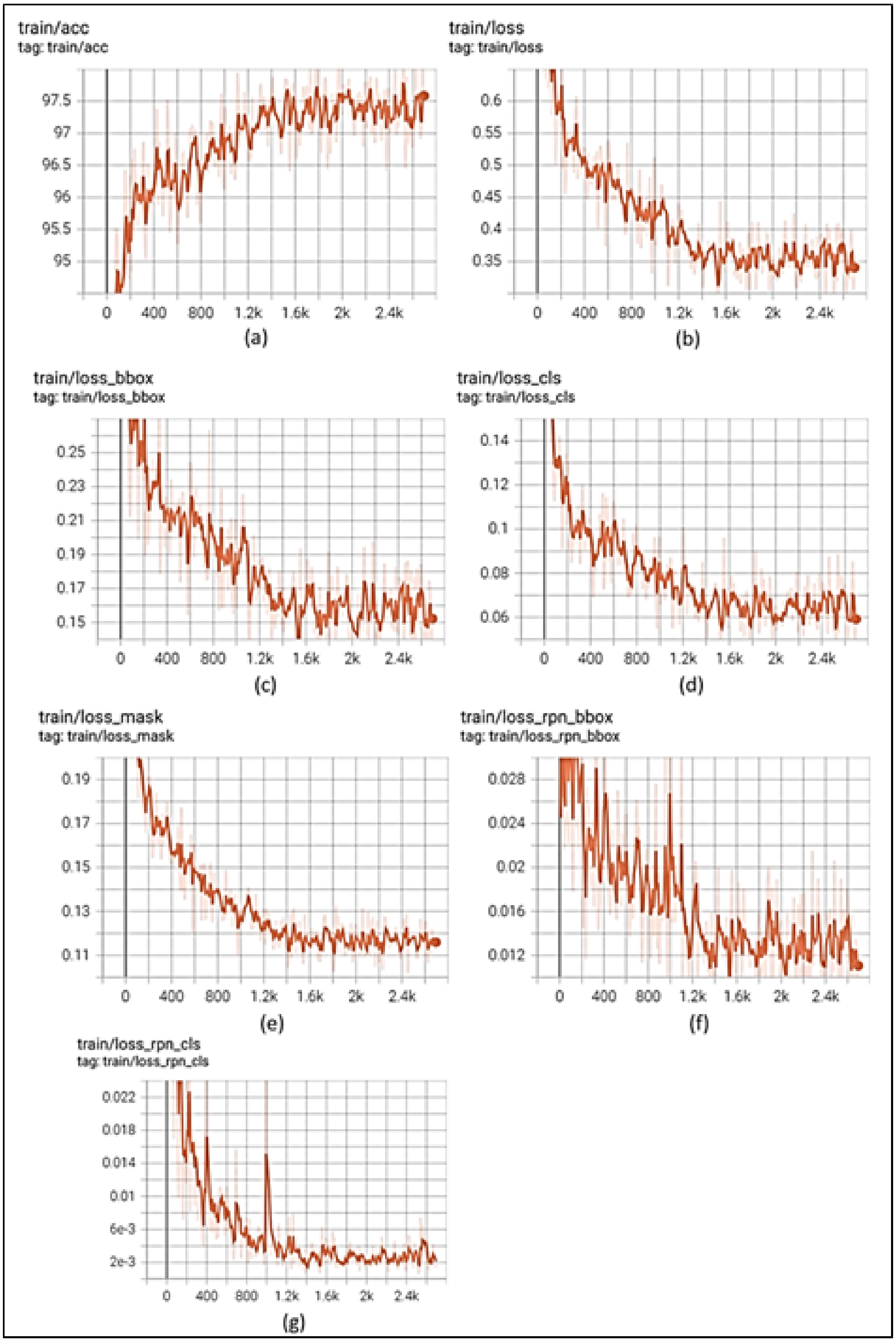
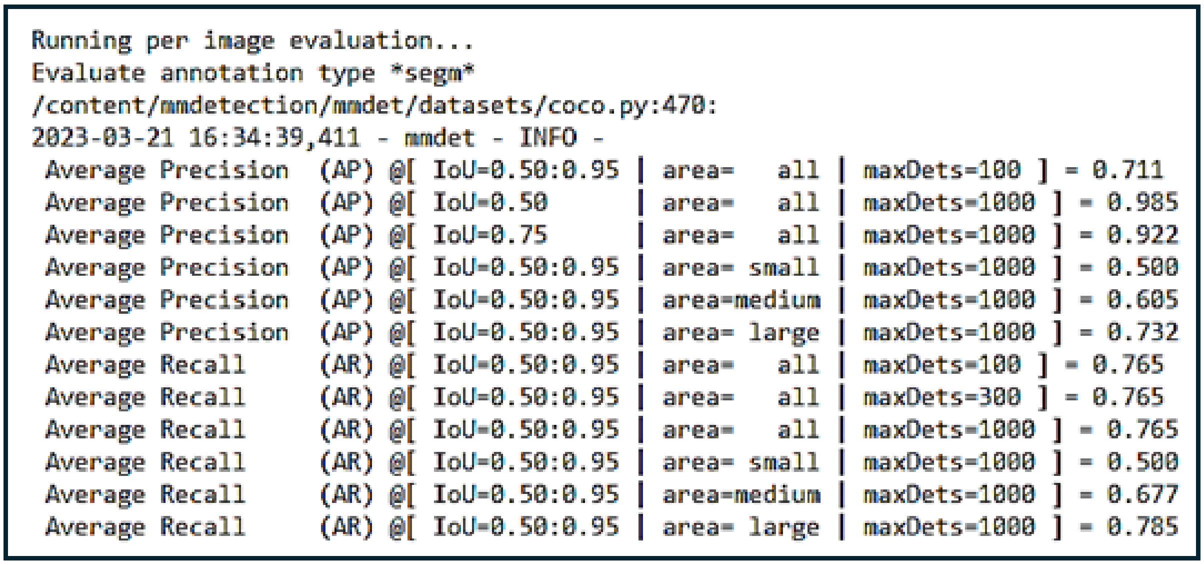


FIGURE 10. Training Accuracy and Loss Graphs

TABLE 2. Training Metrics

Epoch	Learning Rate	Training Time	Memory	Loss_rpn_cls	Loss_rpn_bbox	Loss_cls	Accuracy	Loss_bbox	Loss_mask	Loss
1	2.500e-03	0:40:42	2543	0.1327	0.0245	0.3299	86.1719	0.2994	0.5616	1.3480
10	2.500e-04	0:10:28	2904	0.0019	0.0125	0.0716	97.0508	0.1685	0.1259	0.3804
20	2.500e-05	0:00:02	2904	0.0014	0.0105	0.0570	97.5586	0.1559	0.1161	0.3409



```
Running per image evaluation...
Evaluate annotation type *segm*
/content/mmdetection/mmdet/datasets/coco.py:470:
2023-03-21 16:34:39,411 - mmdet - INFO -
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.711
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.985
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.922
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.500
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.605
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.732
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.765
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.765
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.765
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.677
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.785
```

FIGURE 11. Segmentation Evaluation Results

variations. The evaluation was conducted using the MMDetection framework and the evaluation results are shown in Figure 12.

The evaluation metrics provide insight into the performance of the model in terms of its ability to detect MEP metrics accurately. The bbox_mAP metric measures the mean average precision of bounding box detection, while segm_mAP measures the mean average precision of segmentation. The variations of these metrics, such as bbox_mAP_50 and segm_mAP_75, measure the average precision at different intersection over union (IoU) thresholds.

The results obtained from the evaluation demonstrate that the model has achieved high accuracy in detecting MEP metrics, with bbox_mAP and segm_mAP values of 0.7596 and 0.7111, respectively. The results also show that the model performs well at different IoU thresholds, with bbox_mAP_50 and segm_mAP_75 values of 0.9850 and 0.9219, respectively. The analysis of the results provides insight into the performance of the model and highlights areas for further improvement.

D. DETECTION AND EXTRACTION OF LIGHTING SYMBOLS FOR ENERGY CONSUMPTION PREDICTION

The trained model was applied to an unseen image to evaluate its performance. Figure 13 displays the output of the model, which successfully identified the lighting symbols in the image.

The lighting symbols were then extracted, and using the Pytesseract tool, the text associated with each symbol was extracted shown in Figure 14.

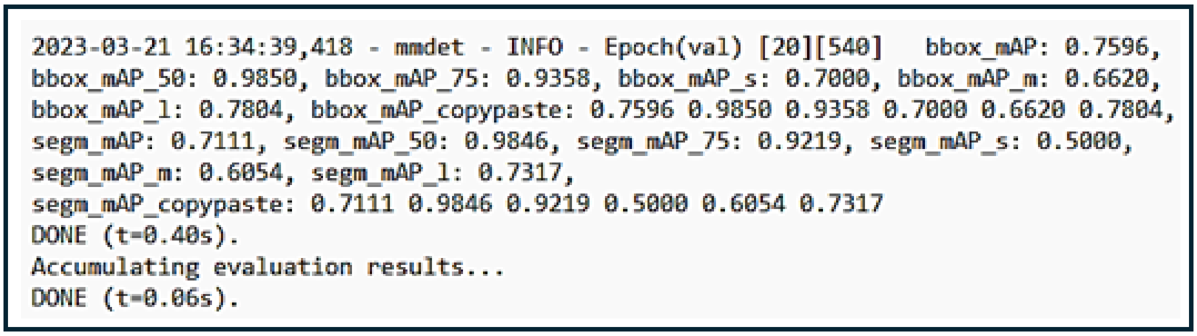
This text provides information about the type of light present in the floor plan, which can be used to predict the overall power consumption. The successful extraction of text from the image demonstrates the effectiveness of the developed model in accurately identifying and extracting MEP metrics from floor plan images.

V. CONCLUSION

In this research work, we have applied the Mask Region-based Convolutional Neural Network (Mask RCNN) to intelligently detect MEP symbols from floor plans to improve the engineering design process's efficiency, accuracy, and cost-effectiveness.

The study achieved bbox_mAP and segm_mAP values of 0.7596 and 0.7111, respectively, and performed well at different IoU thresholds, with bbox_mAP_50 and segm_mAP_75 values of 0.9850 and 0.9219, respectively. These results represent a significant improvement over the traditional methods.

The developed approach can automate the manual work involved in MEP symbol detection, reducing design time and increasing workflow efficiency. It has the potential to significantly impact the fields of architecture, construction, and real estate by improving energy efficiency during the



```
2023-03-21 16:34:39,418 - mmdet - INFO - Epoch(val) [20][540]   bbox_mAP: 0.7596,
bbox_mAP_50: 0.9850, bbox_mAP_75: 0.9358, bbox_mAP_s: 0.7000, bbox_mAP_m: 0.6620,
bbox_mAP_l: 0.7804, bbox_mAP_copypaste: 0.7596 0.9850 0.9358 0.7000 0.6620 0.7804,
segm_mAP: 0.7111, segm_mAP_50: 0.9846, segm_mAP_75: 0.9219, segm_mAP_s: 0.5000,
segm_mAP_m: 0.6054, segm_mAP_l: 0.7317,
segm_mAP_copypaste: 0.7111 0.9846 0.9219 0.5000 0.6054 0.7317
DONE (t=0.40s).
Accumulating evaluation results...
DONE (t=0.06s).
```

FIGURE 12. Evaluation Results for Epoch 20

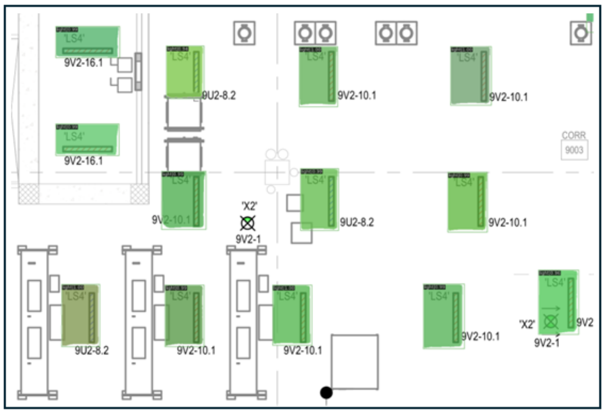


FIGURE 13. Detected Lighting Symbols in Test Image

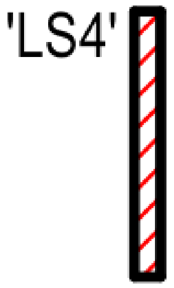


FIGURE 14. Extraction of Lighting Symbols and Associated Text

design process and contributing to the net-zero greenhouse gas emissions goal the Government of Canada set by 2050.

The method employed for intelligently detecting MEP metrics represents a substantial advancement in the MEP engineering sector. It establishes a groundwork for future studies aimed at enhancing its precision and versatility across various image types and qualities.

REFERENCES

[1] Y. Wang, L. Zhang, H. Yu, and R. L. Tiong, “Detecting logical relationships in mechanical, electrical, and plumbing (mep) systems with bim using graph matching,” *Advanced Engineering Informatics*, vol. 54, p. 101770, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034622002282

[2] J. Hu, Q. Bao, T. Zhou, K. Li, L. Shang, J. Zhang, and X. Fu, “Automatic generation construction shop design model of the mep hanger based on bim,” *Buildings*, vol. 13, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2075-5309/13/4/867

[3] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[4] W. Abdulla, “Mask r-cnn for object detection and instance segmentation on keras and tensorflow,” https://github.com/matterport/Mask_RCNN, 2017.

[5] B. Dubois, “Tensorboard,” https://github.com/tensorflow/tensorboard/tree/master, 2019. [Online]. Available: https://github.com/tensorflow/tensorboard/tree/master

[6] A. Rezvanifar, “Analyzing symbols in architectural floor plans via traditional computer vision and deep learning approaches,” pp. 29–31, 2021.

[7] A. Filipski and R. Flandrena, “Automated conversion of engineering drawings to cad form,” *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1195–1209, 1992.

[8] R. Ruwanthika, P. Amarasekera, and R. Chandrasiri, “Dynamic 3d model construction using architectural house plans,” in *6th National Conference on Technology and Management (NCTM)*. IEEE, 2017, pp. 181–184.

[9] S. Dodge, J. Xu, and B. Stenger, “Parsing floor plan images,” in *Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*. IEEE, 2017, pp. 358–361.

[10] C. Ah-Soon and K. Tombre, “Variations on the analysis of architectural drawings,” in *Proceedings of the Fourth*, vol. 1. IEEE, 1997, pp. 347–351.

[11] Y. Aoki, A. Shio, H. Arai, and K. Odaka, “A prototype system for interpreting hand-sketched floor plans,” in *13th International Conference on Pattern Recognition*, vol. 3. IEEE, 1996, pp. 747–751.

[12] A. Koutamanis and V. Mitossi, “Automated recognition of architectural drawings,” in *11th IAPR International Conference on Pattern Recognition*, 1992, pp. 660–663.

[13] D. Greenlee, “Raster and vector processing for scanned linework,” *Photogrammetric Engineering and Remote Sensing*, vol. 53, no. 01, pp. 1383–1387, 1987.

[14] S. Ahmed, M. Liwicki, M. Weber, and A. Dengel, “Automatic room detection and room labeling from architectural floor plans,” in *10th IAPR International Workshop on Document Analysis Systems*. IEEE, 2012, pp. 339–343.

[15] L.-P. d. l. Heras, J. Mas, G. Sánchez, and E. Valveny, “Wall patch-based segmentation in architectural floorplans,” in *International Conference on Document Analysis and Recognition*, 2011, pp. 1270–1274.

[16] L.-P. de las Heras, D. Fernández, E. Valveny, J. Lladós, and G. Sánchez, “Unsupervised wall detector in architectural floor plans,” in *2013 12th International Conference on Document Analysis and Recognition*, 2013, pp. 1245–1249.

[17] L.-P. d. l. Heras, J. Mas, G. Sánchez, and E. Valveny, “Notation-invariant patch-based wall detector in architectural floor plans,” in *Graphics Recognition New Trends and Challenges*. Berlin, Heidelberg: Lecture Notes in Computer Science, Springer, 2013, pp. 79–88.

[18] L.-P. d. l. Heras, D. Fernández, E. Valveny, J. Lladós, and G. Sánchez, “Unupervised wall detector in architectural floor plans,” in *12th International Conference on Document Analysis and Recognition*, 2013, pp. 1245–1249.

[19] C. Liu, J. Wu, P. Kohli, and Y. Furukawa, “Raster-to-vector: Revisiting floorplan transformation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2195–2203.

[20] A. Bulat and G. Tzimiropoulos, "Human pose estimation via convolutional part heatmap regression," in *Computer Vision – ECCV*. Springer International Publishing, 2016, pp. 717–732.

[21] Y. Kiyota, "Promoting open innovations in real estate tech: Provision of the lifull home's data set and collaborative studies," in *International Conference on Multimedia Retrieval*. ICMR, 2018, pp. 6–6.

[22] K. Ryall, S. Shieber, J. Marks, and M. Mazer, "Semi-automatic delineation of regions in floor plans," in *3rd International Conference on Document Analysis and Recognition*, vol. 2, 1995, pp. 964–969 vol.2.

[23] S. Macé, H. Locteau, E. Valveny, and S. Tabbone, "A system to detect rooms in architectural floor plan images," in *9th IAPR International Workshop on Document Analysis Systems, DAS '10*. IEEE, 2010, pp. 167–174.

[24] S.-h. Or, K.-H. Wong, Y.-k. Yu, M. M.-y. Chang, and H. Kong, "Highly automatic approach to architectural floorplan image understanding & model generation," in *Pattern Recognition*, 2005, pp. 25–32.

[25] L.-P. d. l. Heras, S. Ahmed, M. Liwicki, E. Valveny, and G. Sanchez, "Statistical segmentation and structural recognition for floor plan interpretation," *IJDAR*, vol. 17, pp. 221–237, 2014.

[26] R. Wessel, I. Blümel, and R. Klein, "The room connectivity graph: Shape retrieval in the architectural domain," in *The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. UNION Agency-Science Press, 2008.

[27] S. Ahmed, M. Liwicki, M. Weber, and A. Dengel, "Improved automatic analysis of architectural floor plans," in *International Conference on Document Analysis and Recognition*, 2011, pp. 864–869.

[28] P. Dosch, K. Tombre, C. Ah-Soon, and G. Masini, "A complete system for the analysis of architectural drawings," *International Journal on Document Analysis and Recognition*, vol. 3, no. 2, pp. 102–116, 2000.

[29] R. Kasturi, S. Bow, W. El-Masri, J. Shah, J. Gattiker, and U. Mokate, "A system for interpretation of line drawings," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 978–992, 1990.

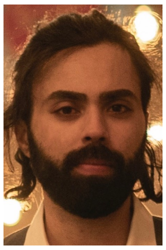
[30] S. Ahmed, M. Weber, M. Liwicki, and A. Dengel, "Text/graphics segmentation in architectural floor plans," in *2011 International Conference on Document Analysis and Recognition*, 2011, pp. 734–738.

[31] J. X. McKie and R. Liu, "Python bindings for the pdf rendering library mupdf," https://pypi.org/project/PyMuPDF/1.16.14/, 2020.

[32] A. Dutta and A. Zisserman, "The VIA annotation software for images, audio and video," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: ACM, 2019. [Online]. Available: https://doi.org/10.1145/3343031.3350535

[33] H. Gu, H. Dong, N. Konz, and M. A. Mazurowski, "A systematic study of the foreground-background imbalance problem in deep learning for object detection," 2023.

[34] A. Chandio, G. Gui, T. Kumar, I. Ullah, R. Ranjbarzadeh, A. M. Roy, A. Hussain, and Y. Shen, "Precise single-stage detector," 2022.



TARANDEEP S. MANDHIRATTA Tarandeep S. Mandhiratta received his Master of Applied Computing degree from Wilfrid Laurier University in 2023 and a Bachelor of Engineering in Computer Engineering from Savitribai Phule Pune University in 2020.

Tarandeep has experience in software engineering, data analysis, artificial intelligence, machine learning, and deep learning. He has applied his expertise in developing innovative solutions using advanced algorithms, neural networks, and data-driven methodologies.

ANK ZAMAN obtained his PhD and MSc in Computer Science from the School of Computer Science, University of Guelph, and the Computer Science Program at the University of Northern British Columbia (UNBC), Prince George, BC, Canada. His research interests encompass Machine Learning, Cybersecurity, Data Science, and Computer Vision. His work aims to develop robust machine learning algorithms for threat detection, data analytics, and computer vision applications, focusing on enhancing digital systems' security and efficiency and harnessing visual data's potential for various domains. He is currently a faculty member at the Dept. of Physics & Computer Science, Wilfrid Laurier University (WLU), Waterloo, Ontario, Canada.

Dr. Zaman has been volunteering for the IEEE Kitchener-Waterloo (KW) (Region 7, Canada) 2013, and he is the Chair of the IEEE Computer Society, IEEE KW, Canada. He is also an active member at Canadian Artificial Intelligence Association (CAIAC).

ABDUL-RAHMAN MAWLOOD-YUNIS Abdul-Rahman Mawlood-Yunis is an assistant professor at Wilfrid Laurier University and the author of the ook "Android for Java Programmers." My expertise lies in Mobile Apps and Smart Devices, Software Engineering, Natural Language Processing (NLP), Distributed Computing, and, Algorithm Design. My recent research projects have centred around Machine Learning on mobile devices and NLP applications for languages with resource constraints.

• • •