Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)
Vol. 7, No. 2, September 2024, pp. 404 – 418
p-ISSN: 2614-3372 | e-ISSN: 2614-6150

Crowd Density Level Classification for Service Waiting Room Based on Head Detection to Enhance Visitor Experience

[1]Atika Istiqomah, [2]Fatih Seida, [3]Nadhira Virliany Daradjat,
[4]Rahman Indra Kesuma, [5*]Nugraha Priya Utama
[1,2,3,4,5]School of Electrical and Informatics Engineering, Institut Teknologi Bandung, Indonesia
Email: [1]23523024@std.stei.itb.ac.id, [2]23523039@std.stei.itb.ac.id, [3]23523041@std.stei.itb.ac.id,
[4]33221026@std.stei.itb.ac.id, [5]utama@staff.stei.itb.ac.id

Article Info

Article history:
Received Apr 24th, 2024
Revised Jun 25th, 2024
Accepted Jul 20th, 2024

Keyword:
Camera footage
Crowd Detection
Density Classification
Energy Saving

ABSTRACT

The crowd within a confined space can potentially lead to air stagnation in waiting areas. Constantly running air conditioning throughout the day to balance air circulation may result in excessive energy consumption by the building. To address this issue, Heating, Ventilating, and Air-Conditioning (HVAC) systems are employed to manage and regulate indoor energy usage. However, sensor-based detection often fails to capture human variables promptly, resulting in less accurate density readings. Camera footage proves to be more reliable than sensors in accurately detecting crowds. This research utilizes You Only Look Once version 8 (YOLOv8), a robust algorithm for object detection, particularly effective in crowd detection for images, along with Convolutional Vision Transformer (CvT) for crowd density level classification into "Normal" and "Crowded" levels. CvT enhances classification accuracy by incorporating function from Convolutional Neural Network (CNN) in model training, including receptive field, shared weights, etc. By integrating YOLOv8 and CvT, this method focuses on accurately classifying crowd density levels after identifying human presence in the waiting area (indoor). Evaluation metrics include mean Average Precision (mAP) for YOLOv8, and accuracy, precision, recall, and f1-score for CvT. This approach directly influences the management of HVAC systems.

Copyright © 2024 Puzzle Research Data Technology

Corresponding Author:
Nugraha Priya Utama,
School of Electrival and Informatics Engineering,
Institut Teknologi Bandung,
Jl. Ganesa No.10, Lb. Siliwangi, Kecamatan Coblong, Kota Bandung, Jawa Barat 40132.
Email: utama@staff.stei.itb.ac.id

DOI: http://dx.doi.org/10.24014/ijaidm.v7i2.29965

1. INTRODUCTION

Efficiency in energy consumption within residential spaces has become a critical consideration for enhancing energy savings indoors. Energy conservation is closely linked to electricity consumption and human density within enclosed areas. Humans play a pivotal role in the energy conservation of a building. The comfort of a residence equates to the comfort experienced within its spaces. Waiting rooms are among the areas necessitating meticulous management and comfort systems. Heating, Ventilating, and Air-Conditioning (HVAC) systems are imperative for regulating room temperature and air circulation in waiting areas [1]. Studies indicate that insights into building occupancy can lead to energy savings ranging from 10% to 40%, encompassing factors such as the number of occupants within a shared space [2]. HVAC systems aim to minimize air-conditioner usage while maintaining indoor air circulation.

Waiting rooms serve as designated spaces for queueing and receiving services. Each waiting room typically accommodates varying capacities based on room size and service importance. Overcapacity tends to disrupt air circulation within waiting rooms [1]. Increased occupancy in waiting areas can lead to congestion and stuffy conditions. Regulating room temperature can mitigate electricity usage indoors.

Manual methods for counting individuals in waiting areas are not only time-consuming and resource-intensive but also prone to errors and inaccuracies. Alternative methods such as Passive Infrared (PIR) sensors and CO2 sensors each have their limitations for indoor crowd detection. PIR sensors only detect moving objects within a room, thus failing to detect stationary occupants. CO2 sensor methods still rely on scheduled air circulation additions, such as opening windows to increase circulation according to a predetermined schedule, and are constrained by their unscheduled system nature [2]. Hence, a more sophisticated and automated approach is needed to address waiting room density issues.

Recent advancements propose numerous Convolutional Neural Network (CNN)-based methods with promising results, where CNN can map images into denser density maps than hand-crafted features [3]. This technique swiftly and accurately detects and classifies human density levels in waiting rooms, subsequently determining room temperatures. YOLOv8 is recognized as the most advanced head detection technology to date. By delving deeper into CNN layers, YOLOv8 can extract more detailed feature representations. YOLOv8 incorporates this concept into its structure by employing repeated modules and multiple head detections for predictions [4]. Vision Transformer (ViT) was the first to demonstrate that pure Transformer architectures can achieve optimal image classification performance when provided with sufficient data [5]. The design of the Convolutional Vision Transformer (CvT) introduces convolution to the two core parts of the ViT architecture. CvT achieves optimal performance when evaluated during the pre-training stage on a larger scale.

Prior research has laid the groundwork for crowd detection, yet there remains a gap in effectively addressing crowd detection and suggest for future development to optimize feature extraction networks and refining the selection method of bounding boxes to streamline the process [14]. Considering YOLOv8, which implements more advanced algorithms on predicted bounding boxes and objectiveness scores compared to its previous versions, this research will adopt YOLO v8 for waiting room crowd detection and CvT for classifying crowd level based on images processed using YOLO. This study focuses on measuring crowd levels indoors rather than simply counting individuals. The innovation of this research lies in developing a model that integrates YOLOv8 with CvT, a method not extensively explored in current literature.

2. RELATED WORKS

2.1. Crowd Detection

Crowd measurements have been conducted using sensor-based approaches. Sensor-based approaches have proven to be accurate for measuring crowds if there are minimal changes within a single capture. Crowd measurement via Wireless Fidelity (WiFi) can leverage existing devices such as smartphones. However, detection constraints or errors may occur when someone does not carry a smartphone or if someone reads their smartphone device more than once. Infrared technology sensors and CO2 detection sensors within a room are also commonly used technologies to estimate indoor occupancy crowd [1]. However, both encounter their own constraints that render them inaccurate for crowd estimation. According to Zou et al. [1], infrared sensors have limitations in capturing the presence of individuals and are unreliable in detecting stationary individuals. Sensors that examine CO2 dispersion within a room will require time before real-time crowd estimation within the dwelling/room can be achieved [2].

According to Sun et al. [2], camera captures can provide precise results regarding occupancy both in real-time and statically. The presence of stationary individuals or individuals obstructed by other objects such as tables, chairs, partitions, etc., can be effectively addressed when utilizing cameras as occupancy detectors. The models used with camera captures are also increasingly diverse and evolving. The developments yielded can address issues such as unstable images due to human movements, similarity between detected heads and other objects, and/or objects obstructing the human form [2].

2.2. YOLO

You Only Look Once (YOLO) utilizes a deep CNN to process images end-to-end and generate object predictions in a single step (one-shot). YOLO v8 divides the input image into a grid of cells with a specific size. Each cell in the grid is responsible for predicting multiple bounding boxes and the classes of objects that may be contained within them then utilizes a series of algorithms applied to the predicted bounding boxes and objective scores generated by the neural network [9]. This version introduces numerous improvements compared to the earlier versions of YOLO such as a new neural network architecture that utilizes both Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) and a new labeling tool that simplifies the annotation process [4]. The FPN works by gradually reducing the spatial resolution of the input image while increasing the number of feature channels. This results in the creation of feature maps that are capable of detecting objects at different scales and resolutions. The PAN architecture, on the other hand, aggregates features from different levels of the network through skip connections. This combination is better than YOLO v5 which uses a modified version of CSPDarknet architecture. Architecture of YOLO v8 is shown in Figure 1.

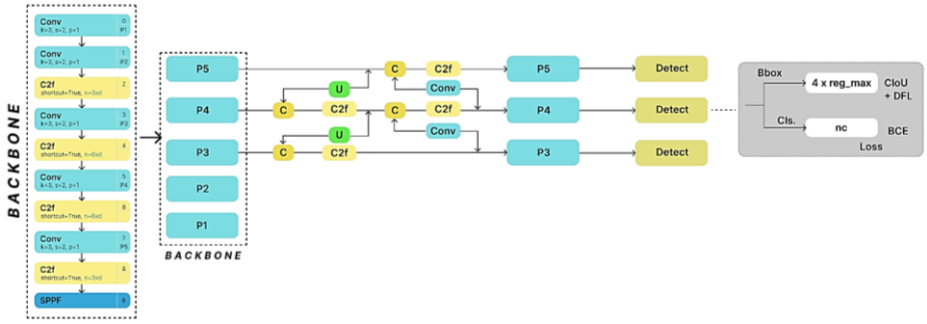

Figure 1. YOLO v8 Architecture

YOLO v8 has a drawback: it is slightly slower than YOLO v5 in terms of object detection speed. However, YOLO v8 is still capable of processing images in real-time on modern GPUs. Based on research [19] for human detection using aerial images, it was found that YOLO v8 outperformed with a precision of 0.84, recall of 0.75, and an F1-score of 0.79. In comparison, YOLO v5 achieved a precision of 0.81, recall of 0.75, and an F1-score of 0.78. The F1-score is considered the accuracy measure of the model in detecting humans, thus YOLO v8 is superior. In research [21] for headcount and detecting suspicious activity, YOLO v8 performed better compared to YOLO v4 and YOLO v3. YOLO v8 produced an F1-score of 42.40% and an mAP of 51.60%, while YOLO v4 produced an F1-score of 39.70% and an mAP of 44.90%, and YOLO v3 produced an F1-score of 29.30% and an mAP of 34.40%.

2.3. CvT

Transformers have dominated empirical machine learning models of natural language processing [7]. The Transformer is an architecture in machine learning that processes sequential data using attention mechanisms and feed-forward neural networks to capture relationships between sequence elements and solve long-distance dependency problems. Its advantage lies in its ability to process data in parallel.

The Vision Transformer (ViT) is the first computer vision model to rely exclusively on the Transformer architecture to obtain competitive image classification performance at large scale. In ViT, initially, images are divided into discrete non-overlapping patches (e.g. 16x16). Each patch is combined with a unique positional encoding to capture basic spatial details, and then fed into a series of standard Transformer layers iteratively, consisting of Multi-Head Self-Attention module (MHSA) and Position-wise Feed-forward module (FFN). This process aims to capture global relationships within the image, facilitating classification tasks.

In Convolutional Vision Transformer (CvT), convolutions are introduced to two primary parts of the vision Transformer: first, to replace the existing Position-wise Linear Projection for the attention operation with Convolutional Projection, and second, to use hierarchical multi-stage structure to enable varied resolution of 2D reshaped token maps, similar to CNNs [5].

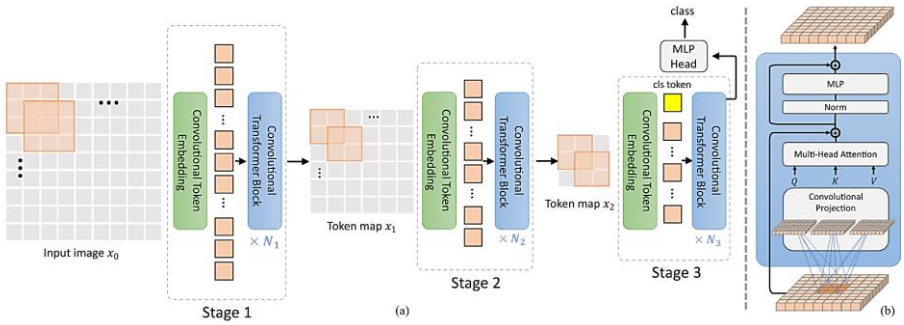

Figure 2. CvT Architecture

Figure 2 shows the architecture of CvT. In the first stage, the image will be divided into token patches. Subsequently, convolutional operations will be embedded using stacked Convolutional Transformers. In the second stage, tokens are once again embedded with convolutional operations at a larger scale, providing a broader context. In the third stage, tokens are re-embedded to refine the process while gathering deeper spatial and contextual information. In this final stage, the tokens map undergoes processing with additional blocks to produce the final representation for classification.

In research [22] on galaxy morphology classification, it was found that CvT provided the best results with an accuracy of 0.98 compared to ViT, which achieved an accuracy of 0.94. This research was conducted using a galaxy image dataset. The study conducted by [5] provided the best results for CvT on the CIFAR 100 dataset, with an accuracy of 94.09 compared to ViT, which achieved an accuracy of 91.67.

3. RESEARCH METHOD

3.1. Design of Experiment

This research employs an experimental approach by modifying the hyperparameters used in each model, YOLOv8 and CvT. Hyperparameters are one of the components or supporting variables of the model that can be adjusted or customized. The combination of hyperparameters can determine the success and reliability of the model, besides being determined by the completeness and variety of the data, and also the model used. The Design of Experiment is clearly outlined in Table 1, Table 2, and Table 3.

Table 1. Experiment setup model YOLO

Hyperparameter	Used Parameter
Epoch	50, 100, 150
Learning Rate	0.01
Batch Size	16
Optimizer	Adam

Table 2. Experiment setup model CvT

Hyperparameter	Used Parameter
Epoch	100, 500
Learning Rate	0.000125 ; 0.0001
Batch Size	32, 64
Batch Normalization	0.0 ; 1.0
Optimizer	Adam

Table 3. Experiment setup model Pre-Trained CvT

Hyperparameter	Used Parameter
Epoch	5

Epochs assist the model in iterating through the dataset into the YOLOv8 and CvT networks. Epochs are used to optimize the model's learning process. For every batch or sample data that completes the reading process until the modeling process returns to the initial training stage, it signifies the completion of one epoch. Iteration refers to repeating the process of reading one batch of data, thus the number of iterations will be equal to the number of batches.

The determination of hyperparameter combinations is done manually, a process often referred to as 'babysitting' or 'trial and error' [18]. Using the babysitting method involves implementing hyperparameter values after building the model, which also considers the short project timeline, thus only a few hyperparameter values are tested. The hyperparameter values are determined based on research related to hyperparameter tuning [15] and the development of the YOLOv8 model, which uses a learning rate of 0.01 [16].

In [15], an epoch of 50 and a batch size of 32 were considered optimal when using grid search, while an epoch of 100 and a batch size of 32 were found to be optimal when using random search. Since this study employs the babysitting method, every combination resulting from the previous grid search and random search studies was tested. The goal is to achieve optimal modeling results with the right combination of hyperparameters, applied to indoor density data. The determination of the learning rate also follows the studies from [16] and [17].

The decision to use the YOLOv8 model for crowd detection is based on YOLOv8's ability to detect objects, especially overlapping objects, making it suitable for detecting human density based on detected heads in a crowd. The CvT to be tested will use both an untrained CvT and a pre-trained CvT for density classification. These two models will be integrated or work in a pipeline, affecting their performance after modeling compared to their performance when each model (YOLOv8 and CvT) works separately. Previous research has not yet explored the pipeline integration of YOLOv8 and CvT for head detection and spatial density classification.

3.2. Metric Evaluation

Evaluation serves as a measurement of the model's reliability when used to process new data, particularly in detecting crowd density and determining its level [6]. Evaluation metrics provide an objective assessment of the model. The evaluation metrics for the first stage modeling (density detection) by YOLOv8 will employ precision, recall, and mean Average Precision (mAP). The metrics for the second stage modeling

(classification) by CvT will utilize precision, recall, and f1-score. True Positive (TP) indicates the number of targets correctly predicted as positive categories. False Negative (FN) represents the number of positive targets incorrectly predicted as negative categories. False Positive (FP) indicates categories incorrectly predicted as positive categories.

Precision measures the model's success in correctly predicting targets as positive categories among all data predicted as positive categories [11]. It is used as an evaluation criterion if the focus is on examining targets predicted as positive categories.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

Recall indicates the ratio of the model's success in correctly predicting positive categories among all true values of positive categories.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

F1-Score considers the values of Recall and Precision. It takes two values into consideration for a more balanced evaluation, as it includes FP and FN, thus accounting for prediction errors [8]. It is typically used when the number of instances in each class is unbalanced, hence F1-Score is considered as the evaluation metric.

$$\text{F1-Score} = \frac{2 \times precision \times recall}{precision + recall} \tag{3}$$

Mean Average Precision (mAP) is a commonly used metric in object detection. The calculation is based on the AP (Average Precision) values, taking into account the Intersection over Union (IoU) threshold. AP calculates the average area under the precision-recall curve. IoU indicates the overlap between predicted and actual bounding boxes in the image. In mAP, a threshold of 0.5 or 50% will be used, meaning that if the IoU value reaches 0.5 or higher, it will be included in the mAP calculation. Mean Average Precision as an evaluation metric will calculate for all object classes.

$$\text{AP (Average Precision)} = \int_0^1 P(R)dR \tag{4}$$

$$\text{mAP (mean Average Precision)} = \frac{1}{N}\sum_{i=1}^{n} AP_i \tag{5}$$

3.3. Dataset

This research utilizes publicly accessible datasets from Roboflow (head detection) and Room Human Counting (RHC) + Bank. The Roboflow dataset consists of RGB images with bounding boxes that detect heads within crowds. The data are captured directly by cameras in crowded indoor locations. This dataset is employed to support modeling for cases of crowd density in waiting areas, thereby assisting in modeling human head detection/crowd detection within indoor spaces.

The RHC dataset exhibits similar conditions, containing data of crowds in meeting rooms captured by CCTV cameras. Images from the RHC dataset are combined with the Bank dataset, obtained manually through browser searches for bank waiting room images. The Bank dataset needs to be merged with the RHC dataset due to their similar indoor enclosed conditions and the lack of detected head information, as well as to address the imbalance in the dataset conditions.

Density detection modeling with YOLOv8 utilizes the Roboflow dataset as input images and is divided into three data partitions: training data, validation data, and test data. The training data is used during model training to assist the model in learning patterns or training its knowledge to detect human heads. The learned results are validated or confirmed for prediction accuracy using validation data. Finally, the reliability of the head detection model is tested using test data. The breakdown can be seen in Table 4.

Table 4. Roboflow Dataset (YOLOv8)

Data Split	Number of Images	Percentage
Train	1630	69,98%
Val	466	20,00%
Test	233	10,00%

Figure 3. Roboflow Crowd Sample

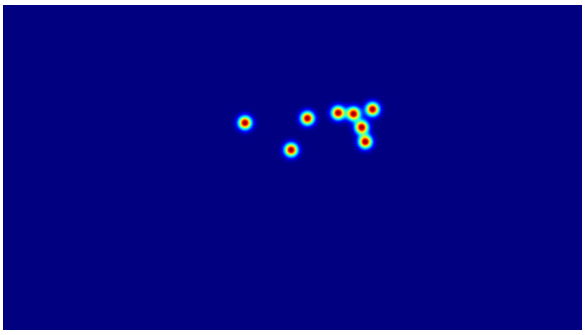

Figure 4. Density Map Sample

Figure 3 shows an example image depicting indoor human density. Detected heads are marked with green boxes or bounding boxes labeled "head." The images undergo pre-processing, including resizing to 224x224, to standardize the input for modeling. This data aids in developing a head detection model using YOLOv8. Subsequently, the process continues to the image class annotation stage for each data point detected by the YOLOv8 model.

The classification process will utilize both the RHC + Bank dataset and the Roboflow dataset. For the RHC + Bank dataset, further preprocessing is performed after detecting the head using YOLO v8, which involves converting it into a density map with a Gaussian kernel specification of 20x20 pixels and a sigma size of 5, it can be seen from Figure 4. This kernel size was chosen considering the estimated size of the head in the images [20][23].

The division into "Normal" and "Crowded" classes of each image is determined by some calculations and also determined by estimating or predicting the number of people in it. For the RHC dataset, due to its static camera characteristics, head detection is counted. If more than 5 heads are detected, it is categorized as dense, and if fewer than 5 heads are detected, it is categorized as normal. The annotation is performed manually based on the knowledge of 3 persons of the authors by separating images that meet the density criteria into their respective class folders. To ensure annotation consistency, an Inter-Rater Reliability (IRR) check is conducted using Cohen's Kappa (K) as an agreement coefficient, with a value greater than 0.8, determined by the following formula,

$$K = \frac{p_o - p_e}{1 - p_e} \tag{6}$$

where p_o represents the number of actual observed agreements and p_e represents the number of chance agreements. The results of the class division can be observed in Table 5 and Table 6.

Table 5. Roboflow Dataset (CvT)

Data Split	Class	Number of Images	Percentage
Train	Normal	200	66,22%
	Crowded	200	
Val	Normal	62	20,52%
	Crowded	62	
Test	Normal	40	13,24%
	Crowded	40	

Table 6. RHC + Bank Dataset (CvT)

Data Split	Class	Number of Images	Percentage
Train	Normal	429	84,28%
	Crowded	429	
Val	Normal	50	9,82%
	Crowded	50	
Test	Normal	30	5,89%
	Crowded	30	

3.4. Device

This research utilizes both local and online devices. Locally, the model is run using a laptop device, while online, it is executed through Google Colab. The specifications for both can be seen in the following Table 7.

Table 7. Device & Environment Specification

Device	Specification
Laptop (local)	Up to Intel® Core™ i7-13620H processor Windows 11 Home / Windows 11 Pro (MSI recommends Windows 11 Pro for business.) Up to GeForce RTX™ 4060 Laptop GPU 8GB GDDR6 Powered by NVIDIA DLSS 3, ultra-efficient Ada Lovelace arch, and Max-Q Technologies. RAM 16GB GDDR5
Google Colab (Online - Pro)	RAM 84GB RAM GPU 40GB HardDisk Space 200GB GPU Nvidia A100 Tensor Core GPU

3.5. Proposed Method

In this study, YOLOv8 will be specifically utilized for crowd detection, where its detection results will be fed into the subsequent model for classification purposes. With its ability to directly generate object predictions and enhance accuracy in object detection, YOLO v8 is expected to provide an effective solution for addressing crowd detection tasks. This signifies a progressive stride in image analysis, where the initial detection outcomes from YOLOv8 serve as a cornerstone for more contextual and profound decision-making by the CvT model in assessing crowd levels classification.

This section will explain the architectural design of the pipeline model of YOLOv8 for crowd detection and CvT for image classification.

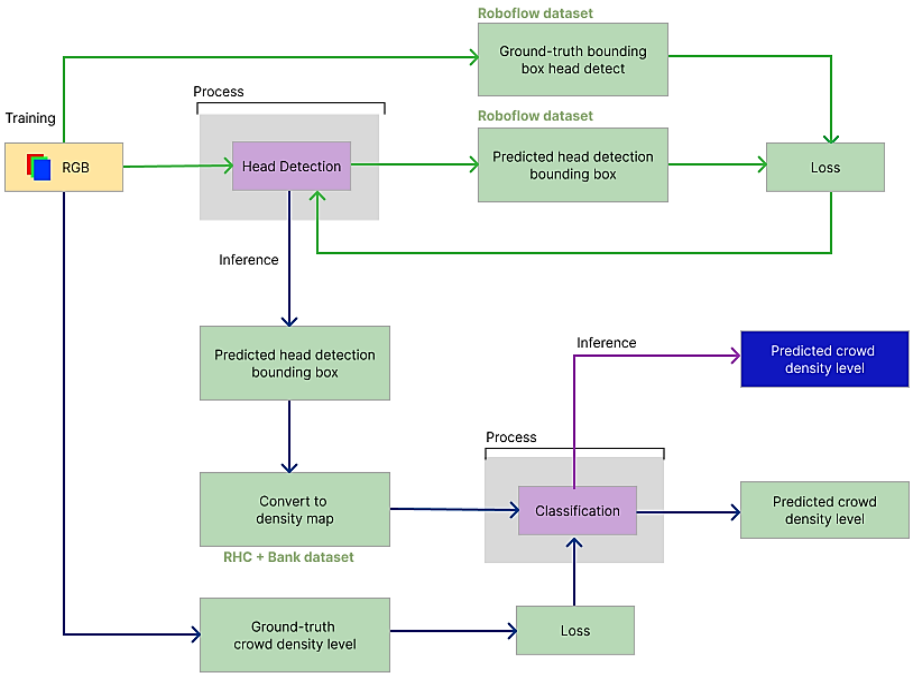

Figure 5. Architecture of Crowd Density Level Classification

From this research architecture in Figure 5, in the first stage of the pipeline, the RGB image will be trained to detect crowds based on each detected head within the image. This training is conducted by the head detection model using YOLOv8, which will yield a loss value and a trained model capable of forming head bounding boxes on the image, with the aim of approximating the ground-truth of the crowd in the image. The loss formula used in YOLO v8 is as follows.

$$
\begin{aligned}
L \;=\; & \frac{\lambda_{box}}{N_{pos}} \sum_{x,y} \quad \Uparrow c^{*}_{x,y} \left[1 - q_{x,y} + \frac{\left\| b_{x,y} - \hat{b}_{x,y} \right\|_2^2}{\rho^2} + \alpha_{x,y} v_{x,y} \right] \\
+ \; & \frac{\lambda_{cls}}{N_{pos}} \sum_{x,y} \quad \sum_{c \in classes} \quad y_c log(\hat{y}_c) + (1 - y_c) log(1 - \hat{y}_c) \\
+ \; & \frac{\lambda_{dfl}}{N_{pos}} \sum_{x,y} \quad \Uparrow c^{*}_{x,y} [-(q_{(x,y)} + 1 - q_{x,y}) log(\hat{q}_{x,y})
\end{aligned}
\tag{7}
$$

$$+ \left(q_{x,y} - q_{(x,y)-1}\right) log\left(\hat{q}_{(x,y)+1}\right)]$$

where:

$$q_{x,y} = IoU_{x,y} = \frac{\hat{\beta}_{x,y} \cap \beta_{x,y}}{\hat{\beta}_{x,y} \cup \beta_{x,y}}$$

$$v_{x,y} = \frac{4}{\Pi^2}\left(arctan\left(\frac{w_{x,y}}{h_{x,y}}\right) - arctan\left(\frac{\widehat{w}_{x,y}}{\widehat{h}_{x,y}}\right)\right)^2$$

$$\alpha_{x,y} = \frac{v}{1 - q_{x,y}}$$

$$\hat{y}_c = \sigma(\cdot)$$

$$\hat{q}_{x,y} = softmax(\cdot)$$

and:

- N_{pos} is the total number of cells containing an object.
- $\Uparrow c^*_{x,y}$ is an indication function for the cells containing an object.
- $\beta_{x,y}$ is a tuple that represents the ground truth bounding box consisting of $(x_{coord}, y_{coord}, width, height)$.
- $\hat{\beta}_{x,y}$ is the respective cell's predicted box.
- $b_{x,y}$ is a tuple represents the central point of the ground truth bounding box.
- y_c is the ground truth label for class c (not grid cell c) for each individual grid cell (x,y) in the input, regardless if an object is present.
- $q_{(x,y)+/-1}$ are the nearest predicted boxes IoUs (left and right) $\in c^*_{x,y}$.
- $w_{x,y}$ and $h_{x,y}$ are the respective boxes width and height.
- ρ is the diagonal length of the smallest enclosing box covering the predicted and ground truth boxes.

This works by first dividing the input image into an S x S grid where each grid cell. Each cell is responsible for predicting the bounding box of an object within the image which is facilitated by a loss function composed of three components. Firstly, the Complete IoU (CIoU) loss serves as the box loss, quantifying the spatial agreement between the predicted and ground-truth bounding boxes. Secondly, the standard binary cross-entropy is employed as the classification loss, allowing each cell to predict multiple classes by treating each class prediction as a binary classification problem. Lastly, the distribution focal loss, acting as the third term, helps address class imbalance by prioritizing the accurate classification of difficult classes during model training. These three components can enhance the overall performance of the object detection model.

After the best-trained head detection model is generated, the image will then be processed. The identification results will be marked by a bounding box surrounding each detected head. Next, in the second stage of the pipeline, the images containing bounding-boxes generated from the previous stage and with the ground-truth labels from annotation results will be fed as inputs to the classification model using CvT. During the training process, a loss function will be utilized to iteratively optimize the model parameters to minimize the difference between the predicted and ground-truth crowd level labels, and aims to enhance the model's ability to accurately classify images into "Normal" and "Crowded". The loss function of this process is Cross-Entropy and Label Smoothing Losses.

The use of Cross-Entropy and Label Smoothing Losses in the Convolutional Vision Transformer (CvT) aims for faster convergence and a higher level of neural collapse [10], as shown in the formula (7), where P is the target distribution and Q is the approximation of the target distribution.

$$H(P, Q) = -\sum_{x \in X} P(x) log(Q(x)) \quad (8)$$

Label Smoothing is designed to shift 10% of the weights from the target label to other labels evenly, aiding in reducing overfitting. This Label Smoothing Cross-Entropy provides mild regularization to the model, decreases overconfidence, and smoothens gradients during backpropagation, while still maintaining sharpness.

Beside the optimization of the classification model through loss functions like Cross-Entropy and Label Smoothing Losses, CvT integrates a crucial convolutional process. This convolution operation within CvT serves as a pivotal mechanism to model local spatial contexts. Below is the formula of the convolution.

$$H_i = \left[\frac{H_{i-1} + 2p - s}{s - o} + 1\right], W_i = \left[\frac{W_{i-1} + 2p - s}{s - o} + 1\right] \quad (9)$$

The convolution operation in CvT aims to model local spatial contexts, ranging from low-level edges to high-level semantic primitives. H_{i-1} and W_{i-1} represent the height and width of the input to the layer, respectively. The padding (p) refers to the amount of padding added to each side of the input. Stride (s) is the number of steps the convolutional filter moves across the input, and dilation (o) affects the spacing between the points sampled by the kernel in the input.

This convolution operation transforms x_{i-1} into a new token map f(x_{i-1}) with channel size C_i and new dimensions for height and width. The formula takes into account stride, padding, and input size to determine the output dimensions. Subsequently, the process of normalization and tokenization, derived from the previous token map, involves flattening into the size $H_i W_i$ x C_i and normalization using layer normalization, aimed at stabilizing learning and model convergence.

The Convolutional Token Embedding process then allows for adjustments in the dimensions of token features and the number of tokens at each stage. At every stage, the length of the token sequence is progressively reduced while the dimension of the token features is increased (enabling tokens to represent increasingly complex visual patterns over larger spatial footprints).

The goal of the convolutional projection part is to achieve additional local spatial context modeling and to offer efficiency benefits by allowing subsampling from the K and V matrices. Fundamentally, the proposed Transformer block with Convolutional Projection is a generalization of the original Transformer block, wherein the Convolutional Projection utilizes Multi-Head Self-Attention (MHSA) with depth-wise separate convolutions as shown in Figure 6.

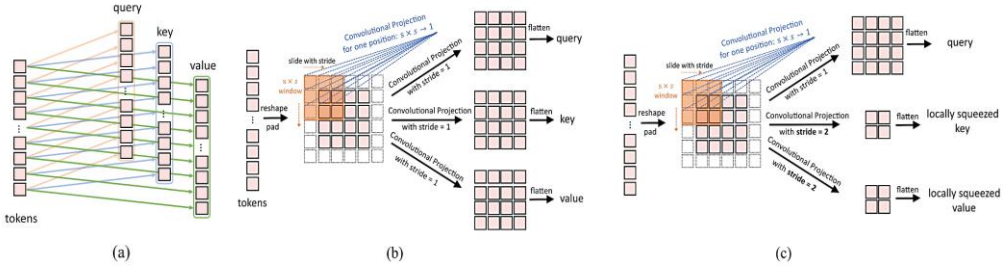

Figure 6. Convolutional Projection Flow of Process

In implementation, there are several stages, beginning with the original position-based linear projection used in ViT. This is followed by the previously discussed s x s convolutional projection. Initially, tokens are transformed into 2D token maps. The convolutional projection is implemented using a depth-wise separable convolution layer with a kernel size of s. Projected tokens are then flattened into 1D for subsequent processing as follows.

$$x_i^{q/k/v} = Flatten\ (Conv2d(Reshape2D(x_i), s)) \quad (10)$$

$x_i^{q/k/v}$ represents the input token for the q/k/v matrices at layer i, xi is the undisturbed token prior to convolutional projection, Conv2d is the depth-wise separable convolution implemented by Depth-wise Conv2d > BatchNorm2d > Point-wise Convolution, and s (stride) refers to the kernel size of the convolution. Depthwise Convolution is a process where convolution is performed separately on each channel. BatchNorm2d normalizes the data to accelerate convergence and enhance training effectiveness. Point-wise Convolution is the aggregation of information from the output of depthwise convolution using a 1×1 filter. This transformation ensures that during training, layers can continue learning from an input distribution that shows less internal covariate shift, thereby speeding up the training process.

Convolutional projection allows for a reduction in the number of tokens required for MHSA operations by using a stride greater than 1 on k (key) and v (value), reducing the amount of data that must be processed by MHSA by four times. This design employs depth-wise separable convolution, which significantly reduces the number of parameters and the required floating-point operations (FLOPs).

And lastly, to enhance performance in both models and optimize the training process, the Adam optimizer is employed. Adam is a frequently chosen option in neural network model training due to its ability to adaptively adjust the learning rate for each model parameter [12]. Figure 7 shows the algorithm of Adam.

Input: Network N with trainable parameters Θ; subset of activations $\{x^{(k)}\}_{k=1}^{K}$
Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$
1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:
$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$
11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$
12: **end for**

Figure 7. Adam Optimization Algorithm

This model utilizes predicted weights to perform both forward pass and backward propagation, and then uses gradients with respect to these predicted weights to update the model parameters. This approach is intended to more effectively direct weight updates in the "correct" direction, as opposed to using only the current weights. The use of weight prediction is expected to enhance the convergence of model training, allowing the model to more rapidly adapt and refine weights in reducing prediction errors. Below is the formula of Adam optimizer.

$$\theta_t = (1 - \gamma\lambda)\theta_{t-1} - \frac{\gamma\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon'} \tag{11}$$

$$s.t. \begin{cases} g_t = \nabla_t f(\theta_{t-1}), \\ m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t, \\ v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2, \\ \hat{m}_t = \frac{m_t}{1-\beta_1^t}, \\ \hat{v}_t = \frac{v_t}{1-\beta_1^t} \end{cases} \tag{12}$$

4. RESULTS AND DISCUSSION

4.1. Result

The research results are obtained after undergoing the training and testing processes of the model. The evaluation metrics obtained are the results of testing on the test data, which are data that have never been presented during model training. The Roboflow dataset is used to train and test YOLOv8. Classification is tested using both the RHC + Bank dataset and the Roboflow dataset.

The YOLOv8 modeling yields a model capable of detecting density based on human heads, with human heads being marked with bounding boxes. The model accurately predicts and marks the presence of human heads in images, despite the varied sizes of the heads and diverse human positions in the images. The classification model is obtained from modeling with CvT. The testing results prove to be reliable for both datasets, Roboflow and RHC + Bank, and the best results are selected from all experiments.

The modeling results are influenced by hyperparameter tuning that supports the training speed, data grouping, and the number of model iterations in learning data patterns. Hyperparameters play a crucial role in training the model so that it can detect density quickly and accurately and can determine its density level.

Table 8. YOLOv8 Crowd Detection Model Result

Hyperparameter	mAP50	Precision	Recall
Epoch : 50 Learning Rate : 0.01 Batch-Size : 16	0.4992	0.914	0.839
Epoch : 100	0.92683	0.90623	0.86974

Hyperparameter	mAP50	Precision	Recall
Learning Rate : 0.01 Batch-Size : 16			
Epoch : 150 Learning Rate : 0.01 Batch-Size : 16	**0.92947**	**0.9171**	**0.87007**

The testing results of the YOLOv8 model show optimal performance in the third configuration based on Table 8, with 150 epochs, a learning rate of 0.01, and a batch size of 16. With 150 epochs, it means that the optimal model is obtained after the data has been read 150 times. The mAP score obtained is 0.92, with precision and recall values of 0.91 and 0.87, respectively. Compared to previous research and also with the implementation of the model on datasets that have never been tested before, the crowd detection model has demonstrated improved performance and represents the best development of the YOLOv8 model.

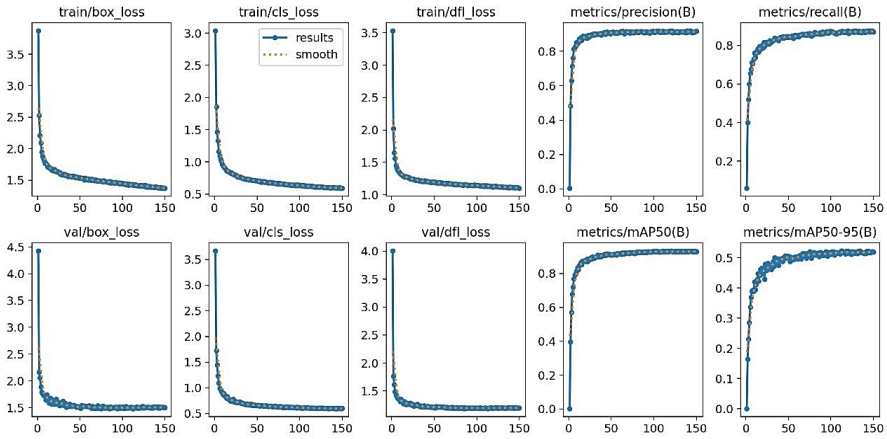

Figure 8. Loss graph and metrics when training YOLOv8

From Figure 8, YOLOv8 training graph shows that the loss continues to decrease over time as the epoch value increases. This indicates that the model continues to evolve, thus showing no signs of overfitting. Overfitting occurs when the model performs better during the training phase compared to testing.

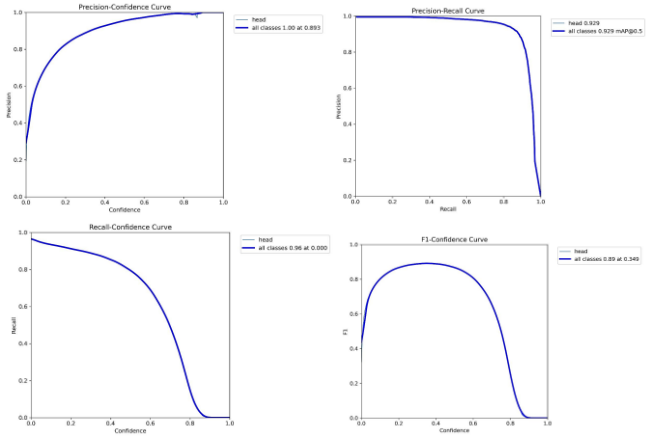

Figure 9. Metric graph when testing YOLOv8

Figure 9 shows the YOLOv8 testing graph, the mAP value for a threshold of 0.5 is observed to be 0.929. The precision-recall graph shows an inverse relationship, where high precision values result in low recall values, and vice versa.

Table 9. CvT Classification Model Result for Roboflow Dataset

Hyperparameter	Accuracy	F1-Score	Precision	Recall	Average Loss
Epoch : 50 Learning Rate : 0.0001	0.84	0.84	0.86	0.84	0.685

Hyperparameter	Accuracy	F1-Score	Precision	Recall	Average Loss
Batch-Size : 32					
Epoch : 150 Learning Rate : 0.0001 Batch-Size : 32	**0.85**	**0.85**	**0.87**	**0.85**	**0.625**
Epoch : 500 Learning Rate : 0.0001 Batch-Size : 32	0.84	0.84	0.86	0.84	0.713

Table 10. CvT Classification Model Result for RHC + Bank Dataset

Hyperparameter	Accuracy	F1-Score	Precision	Recall	Loss
Epoch : 50 Learning Rate : 0.0001 Batch-Size : 32	0.47	0.36	0.46	0.47	0.52
Epoch : 150 Learning Rate : 0.0001 Batch-Size : 32	0.51	0.49	0.51	0.51	0.67
Epoch : 500 Learning Rate : 0.0001 Batch-Size : 32	0.66	0.58	0.60	0.66	0.34

Table 11. Pre-Trained CVT Classification Model Results for RHC + Bank Dataset

Hyperparameter	Accuracy	F1-Score	Precision	Recall	Loss
Epoch : 5	**0.91**	**0.90**	**0.91**	**0.91**	**0.31**

CvT result can be seen from Table 9, Table 10, and Table 11. Classification using CvT yielded different results on the Roboflow dataset and the RHC + Bank dataset. The best results were obtained with the Roboflow dataset, achieving an accuracy of 0.85. In contrast, the RHC + Bank dataset produced significantly lower results during the CvT training process, ranging only between 0.4 and 0.6. The Pre-Trained CvT was tested to enhance the CvT model's performance on the RHC + Bank dataset. Higher performance was achieved when testing with the RHC + Bank dataset using the Pre-Trained CvT and 5 epochs, resulting in an accuracy of 0.91, an F1-score of 0.90, and both precision and recall of 0.91.

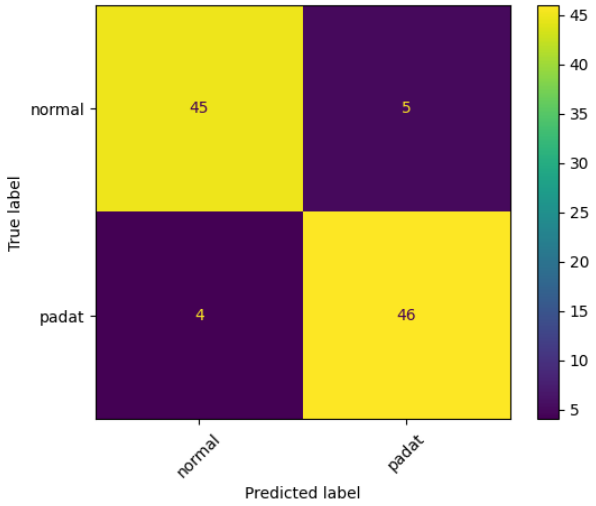

Figure 10. Testing Result of RHC Dataset using Confusion Matrix

The confusion matrix shown evaluates the performance of a classification model used to distinguish between two categories: "normal" and "padat" (crowded). The matrix displays the number of correct and incorrect predictions made by the model in a tabular format. Overall, the confusion matrix demonstrates that the model has a high accuracy, with most predictions being correct. The minor misclassifications indicate areas where the model could be further refined for improved performance.

4.2. Discussion

The testing results obtained through the YOLOv8 and CvT modeling demonstrate their reliability, consistent with previous research. The YOLOv8 testing results show little difference between recall and precision. This indicates that 0.87 parts of the head data were successfully predicted as heads, while 0.91 parts

predicted as heads still include some incorrect predictions as heads. Precision focuses on quantity, while recall focuses on quality.

The combination of hyperparameters can influence the indication of overfitting. The results of the YOLOv8 modeling do not indicate overfitting. This can be seen from the stable downward movement of the testing graph for loss values and the upward trend for precision, recall, and mAP. This suggests that with epoch value 150, learning rate 0.01, and batch size 16, it has become a balanced combination to enhance the performance of YOLOv8 when applied to this research's study case (crowd detection).

The pre-trained CvT model yielded the best results for density classification. The differing results between the Roboflow dataset and the RHC + Bank dataset could be due to the possibility of overfitting in one of the dataset tests or could be related to data variation. The best classification results were obtained by CvT with the RHC + Bank dataset because this dataset tends to be more diverse and easier to distinguish between classes.

The use of YOLOv8 and CvT for detection and classification of density levels is appropriate. Given that previous research has shown YOLOv8 to provide the best results as an object detector, it is also suitable for detecting density. Additionally, the use of CvT in classification tasks has proven to give better results compared to ViT, as the convolutional layers in CvT enhance the precision of transformers for density classification. Both models work in a pipeline and have achieved the research objective of addressing the issues of density detection and classification, and conducting new experiments by combining both models for the tasks of density detection and classification. The results of previous experiments and those in this research can differ, showing either lower or higher performance. This is because in this research, the model was retrained using different data from previous research, resulting in potentially different outcomes, but not significantly different. Particularly in classification results, the accuracy obtained is quite different from previous research, which could be due to the pipeline process that begins with the YOLOv8 model and then is used as input for CvT.

The model developed in this research can be implemented in further platform development. Implementation involves integrating it with camera sensors, particularly through CCTV cameras, and connecting it to devices such as air conditioners. Configurations related to device integration pose additional challenges for further development. An air conditioner integrated with a camera and capable of accurately reading density through the developed model can support the implementation of HVAC systems. HVAC helps achieve energy savings indoors and ensures smooth air circulation. Efficient air circulation, especially in waiting areas, can enhance the comfort experience for visitors awaiting services.

5. CONCLUSION

The new method combining YOLOv8 and CvT involves YOLOv8 functioning as crowd detection, while CvT is utilized for classifying waiting room density levels. The model operates in a pipelining manner, where the output from YOLOv8 modeling is used as input for CvT. The optimal results for YOLOv8 were obtained with hyperparameter tuning, specifically with an epoch of 150, learning rate of 0.01, and batch size of 16. This resulted in an mAP value of 0.92, precision of 0.91, and recall of 0.87 for the YOLOv8 model. The classification using CvT has proven reliable in distinguishing each density class, "Normal" and "Crowded." Modeling with CvT yielded the best results with the pre-trained CvT and 5 epoch. The best results were achieved when using the RHC + Bank dataset, as it was considered more diverse and easier for the model to learn to distinguish each class. The evaluation results obtained were a score of 0.91 for accuracy, 0.90 for f1-score, and 0.91 for recall and precision.

The research has successfully achieved its objective of combining two models that can detect heads and classify density levels. The issue of integrating the two models has been successfully addressed, although the results are not as optimal as when the two models work separately. This research has some weaknesses, such as the determination or division of dataset classes for "Normal" and "Crowded" density levels. The authors still manually divided images into each class based on simple knowledge and considerations. Further assessment and validation from spatial planning experts or room efficiency experts are needed, especially regarding the accuracy of images with their labels.

REFERENCES

[1] H. Zou, Y. Zhou, J. Yang, and C. J. Spanos, "Device-free occupancy detection and crowd counting in smart buildings with WiFi-enabled IoT," Elsevier, 2018

[2] K. Sun, Q. Zhao, Z. Zhang, and X. Hu, "Indoor occupancy measurement by the fusion of motion detection and static estimation," Elsevier, 2021

[3] J. Chen, S. Wen, and Z. Wang, "Crowd counting with crowd attention convolutional neural network," arXiv:2204.07347v1, Apr. 2022

[4] D. Reis, J. Kupec, J. Hong, dan A. Daoudi, "Real-Time Flying Object Detection with YOLOv8," Georgia Institute of Technology, 17 May 2023. [Online]. Available: arXiv:2305.09972v1

[5] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, dan L. Zhang, "CvT: Introducing Convolutions to Vision Transformers," 29 March 2021. [Online]. Available: arXiv:2103.15808v1
[6] H. Chen, G. Zhou, dan H. Jiang, "Student Behavior Detection in the Classroom Based on Improved YOLOv8," Sensors, 2023
[7] T. Xiao, J. Zhu, "Introduction to Transformers: an NLP Perspective", 29 November 2023. [Online]. Available: arXiv:2311.17633v1
[8] Talaat, F. M., & Zain Eldin, H. (2023). "An improved fire detection approach based on YOLO-v8 for smart cities." Neural Computing and Applications
[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," University of Washington, Allen Institute for AI, Facebook AI Research, 2016
[10] L. Guo, K. Ross, Z. Zhao, G. Andriopoulos, S. Ling, Y. Xu, and Z. Dong, "Cross Entropy versus Label Smoothing: A Neural Collapse Perspective," New York University Shanghai, New York University Abu Dhabi, 2024, arXiv:2402.03979v2
[11] C. Goutte and E. Gaussier, "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation," in Lecture Notes in Computer Science, Springer, April 2005, DOI: 10.1007/978-3-540-31865-1_25
[12] L. Guan, "Weight Prediction Boosts the Convergence of AdamW," Department of Mathematics, National University of Defense Technology, August 2023, arXiv:2302.00195v2
[13] McHugh ML. Interrater reliability: the kappa statistic. Biochem Med (Zagreb). 2012;22(3):276-82. PMID: 23092060; PMCID: PMC3900052
[14] T. Zhou, L. Zheng, Y. Peng and R. Jiang, "A Survey of Research on Crowd Abnormal Behavior Detection Algorithm Based on YOLO Network," 2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE), Guangzhou, China, 2022, pp. 783-786, doi: 10.1109/ICCECE54139.2022.9712684. keywords: {Computer vision;Object detection;Machine learning;Real-time systems;Detection algorithms;Consumer electronics;abnormal crowd behavior;deep learning;YOLO network;target detection}
[15] Firdaus, F. F., Nugroho, H. A., & Soesanti, I. (2021). Deep Neural Network with Hyperparameter Tuning for Detection of Heart Disease. The 2021 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob) (pp. 59-65). IEEE
[16] Khow, Z. J., Tan, Y.-F., Karim, H. A., & Rashid, H. A. (2024). Improved YOLOv8 Model for a Comprehensive Approach to Object Detection and Distance Estimation. IEEE Research Article, 63754-63767
[17] Li, H., Chaudhari, P., Yang, H., Lam, M., Ravichandran, A., Bhotika, R., & Soatto, S. (2020). RETHINKING THE HYPERPARAMETERS FOR FINE-TUNING. Conference Paper at ICLR. arXiv
[18] Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 295-316
[19] Sary, I. P., Armin, E. U., & Andromeda, S. (2023). Performance Comparison of YOLOv5 and YOLOv8 Architectures in Human Detection Using Aerial Images. Ultima Computing : Jurnal Sistem Komputer, 8-13
[20] Quan, Y., Wu, Z., & Ji, H. (2021). Gaussian Kernel Mixture Network for Single Image Defocus Deblurring. 35th Conference on Neural Information Processing Systems (NeurIPS).
[21] Dorai, S., Jayakumar, S., Suresh, A., Krishnan, A., Prasad, S., & Karthik, P. (2023). Proactive Headcount and Suspicious Activity Detection using YOLOv8. 3rd International Conference on Evolutionary Computing and Mobile Sustainable Networks (ICECMSN 2023) (pp. 61-69). Elsevier.
[22] Cao, J., Xu, T., Deng, Y., Deng, L., Yang, M., Liu, Z., & Zhou, W. (2024). Galaxy morphology classification based on Convolutional vision Transformer (CvT). *Astronomy & Astrophysics (A&A)*, 1-11.
[23] A. P. Prakoso, W. Piseno, D. Bisono, D. B. Saefudin, A. Fudholi, and F. N. Rohman, "Comparative Study on NACA-9405, NACA-9503 and NACA-9506 Airfoil Profiled Blade Open-Channel Flow Cross-Flow Turbine," Public Research Journal of Engineering, Data Technology and Computer Science, vol. 1, no. 1, pp. 10–19, Jul. 2023, doi: 10.57152/predatecs.v1i1.833.

BIBLIOGRAPHY OF AUTHORS

Atika Istiqomah is a student at School of Electrical Engineering and Informatics, Institut Teknologi Bandung (ITB) in the Informatics Master Program. Competence in data science, data visualization, machine learning, and information systems. She had a bachelor's degree in computer science from Lampung University, Indonesia. Currently, she is focusing on thesis research in Natural Language Processing (NLP) more specifically in text style transfer at the School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Indonesia.

Fatih Seida is currently pursuing a Master's degree in the School of Electrical Engineering and Informatics (STEI) at the Bandung Institute of Technology (ITB). He holds a Bachelor's degree in Computer Science from UIN Sunan Gunung Djati Bandung. He is competent in Information Systems with a keen interest in organizational governance leveraging technology and artificial intelligence. Fatih is focused on managing and utilizing current technology for companies, providing sharp analytical insights, and presenting them effectively. His thesis focuses on object detection and object segmentation

Nadhira Virliany Daradjat is currently pursuing a Master's degree in the School of Electrical Engineering and Informatics (STEI) at the Bandung Institute of Technology (ITB). She holds a Bachelor's degree in Computer Science from Binus University. She is competent in data science, data analysis, and systems analysis. Her thesis focuses on graph analytics and graph neural networks.

Rahman Indra Kesuma is a lecturer at Institut Teknologi Sumatera (ITERA) in the Informatics Engineering Undergraduate Program. Competence and mastery in artificial intelligence, expert systems, decision support systems, machine learning, and deep learning. He had a bachelor's degree in computer science from Lampung University, Indonesia, and a master's degree in computer science from Gadjah Mada University in the same country. Currently, he is focusing on dissertation research in video anomaly detection and prediction with self-supervised learning at the School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Indonesia.

Nugraha P. Utama is a lecturer at the Informatics Department, School of Electrical Engineering and Informatics, Institute Technology Bandung. He has a background in computational Intelligence and Neuroscience from Tokyo Institute of Technology, and National Center of Neurology and Psychiatry, Japan. His current work focuses on artificial intelligence and computer vision in all ways of applications.