Journal of Image Processing Theory and Applications (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/jipta.2024.070111
ISSN 2560-6239 Vol. 7 Num. 1

Application of YOLOv8 Image Recognition Model for Human Actions Recognition in the Surveillance Filed

Yuhan Lin[1], Jingchu Wang[1], Dong Lin[2,*]

[1]*Jinan University–University of Birmingham Joint Institute at Jinan University, Jinan University, Guangzhou, China*
[2]*Hangzhou Hengsheng Digital Equipment Technology Co., Ltd, Hangzhou, China*
[*]*Corresponding author: lindong@hisome.com*

Keywords: Computer vision, image recognition, YOLOv8, surveillance image processing

Abstract: With the rapid advancement of computer vision technology, the application of image recognition has expanded across various fields, particularly in public safety and intelligent surveillance systems. This paper reviews the evolution of YOLO models from v1 to v8, focusing on the advancements in detection speed, computational efficiency, and accuracy of YOLOv8. We analysed YOLOv8's algorithm and network architecture, detailing its application to human action recognition in surveillance imagery. Through comprehensive testing on diverse surveillance videos, we validate YOLOv8's enhanced performance and efficiency in recognizing human postures and actions. Our findings underscore YOLOv8's significant practical value and its potential for broader application in intelligent surveillance systems.

1. Introduction

Real-time object detection is a crucial branch of image recognition, integral to numerous applications including autonomous vehicle, robotics, video surveillance, and augmented reality. The YOLO framework stands out among object detection algorithms for its swift and precise identification of image targets, enjoying widespread use today. Since 2015, the YOLO series has undergone multiple iterations and developments, culminating in the ongoing advancement of YOLOv9, resulting in a significant improvement in model performance. The YOLOv8 model, introduced in 2023, is the focus of our study and currently the most applicable in the image recognition domain. This paper reviews the evolution of the YOLO series of models, with a detailed analysis of version 8, which will be applied in future surveillance image recognition.

For surveillance image recognition, traditional models merely extract contour information for object identification, resulting in low accuracy and efficiency. However, the ad-vent of deep learning and convolutional neural networks in object detection has been pivotal, with the training and learning capabilities of CNN models greatly enhancing accuracy. YOLOv8 serves as a prime example. Therefore, we have selected it for application in exam room surveillance, aiming to recognize the Behaviours of examinees and teachers and establish a generalizable image recognition model.

The work we have done is mainly shown in the following Figure 1.

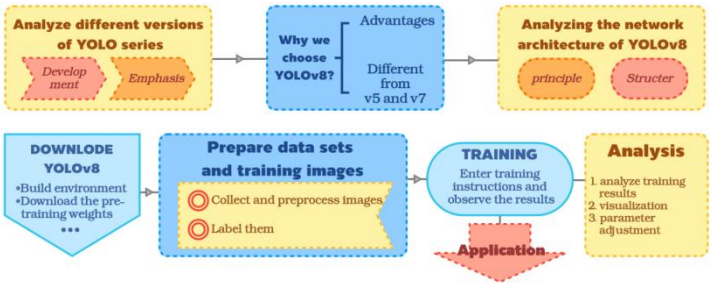

Figure 1: Our work

2. YOLO model analysis

2.1. Development history

The YOLO (You Only Look Once) model has undergone several iterations, each bringing improvements and new features[1].

YOLOv1 (2016): Introduced by Joseph Redmon et al. at CVPR, it proposed a real-time, end-to-end object detection method with 24 convolutional layers and 2 fully connected layers. It used leaky rectified linear unit activations except for the final linear layer. However, it had limitations in detecting multiple objects within a grid cell and struggled with objects lacking aspect ratios.

YOLOv2 (2017): Improved upon YOLOv1 by refining the model on ImageNet and employing a fully convolutional architecture for multi-scale training. This version enhanced network performance, detecting multiple regions for more comprehensive results.

YOLOv3 (2018): Introduced logistic regression for predicting object scores and binary cross entropy for training logistic classifiers. It featured multiple output layers to better detect both large and small targets. However, Joseph Redmon stopped updating YOLO re-search in early 2020 to prevent misuse.

YOLOv4 (2020): Developed by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, this version maintained real-time detection with significant enhancements, including self-adversarial training, a genetic algorithm, and a cosine annealing scheduler for optimal hyperparameters. It featured mosaic enhancement, regularization, and detector innovations.

YOLOv5 (2020): Released by Glen Jocher of Ultralytics, it was developed in PyTorch and included automatic anchoring and other improvements from YOLOv4. YOLOv5 offered five scaled versions (nano, small, medium, large, and extra large) to cater to different ap-plications and hardware needs.

YOLOv6 and YOLOv7 (2022): YOLOv6, by Meituan Vision AI, introduced the EfficientRep backbone and task alignment learning for faster detection. YOLOv7, published on ArXiv by the YOLOv4 authors, improved accuracy without compromising inference speed, using efficient layer aggregation and hierarchical stacking modules.

YOLOv8 (2023): Released by Ultralytics, it featured an anchor-free model with a de-coupled head for separate handling of objectivity, classification, and regression tasks. It used CIoU and DFL loss functions for bounding box accuracy and binary cross entropy for classification, enhancing performance, especially for smaller objects. YOLOv8 supports command line interface and PIP package installation for versatile detection, cutting, and classification tasks[2].

2.2. Analysis of yolov8

The network structure of YOLOv8 primarily consists of three parts:

(1)Backbone: It employs a series of convolutional and deconvolutional layers to ex-tract features, while also using residual connections and bottleneck structures to reduce the network size and improve performance. Notably, the Backbone of YOLOv8 utilizes the C2f module as its basic building unit. Compared to the C3 module in YOLOv5, the C2f module has fewer parameters and superior feature ex-traction capabilities. Thus, the optimization in linkage and computational efficiency in YOLOv8 stems from this component.

(2)Neck: It merges feature maps extracted from different stages of the Backbone using multi-scale feature fusion techniques to enhance feature representation. For in-stance, it employs the Feature Pyramid Networks (FPN).

(3)Head: It is responsible for performing the final object detection and classification tasks of the model. The Head is divided into a detection head and a classification head. The detection head contains numerous convolutional layers to generate detection results, while the classification head employs global average pooling to classify the feature maps.

Figure 2 is a diagram of the module provided in the official YOLOv8 paper.

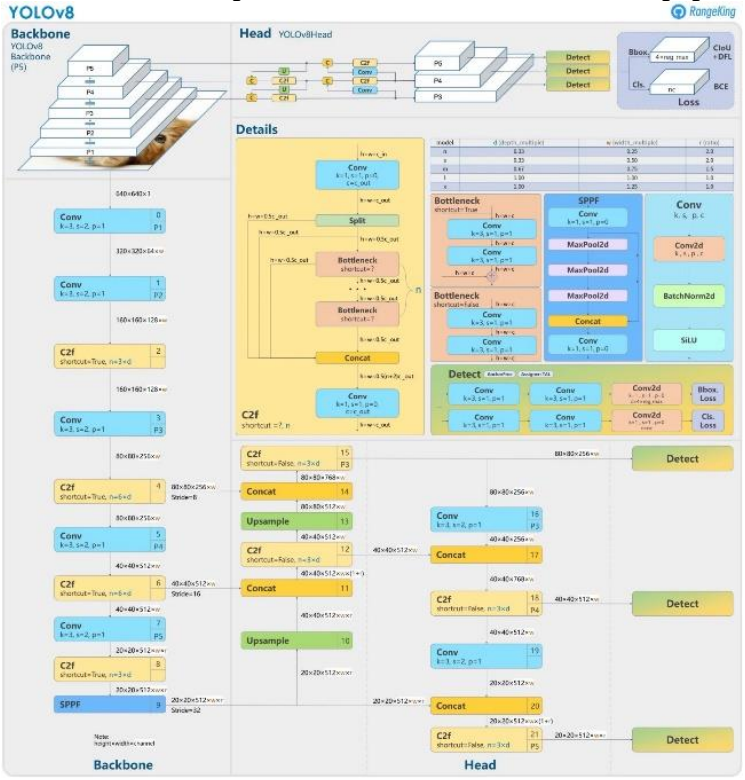

Figure 2: The diagram of YOLOv8 module

On the left is the Backbone part, consisting of five convolutional modules, four C2F modules, and one SPPF module. The main structure of the convolutional modules includes Conv2D, Batch Normalization layers, and the SiLU activation function, which accelerates and stabilizes neural network training, reducing YOLOv8's dependence on weight initialization.

SiLU activation function is $SiLU(x) = x\left(\frac{1}{1+e^{-x}}\right)$.

It is worth mentioning the significant changes in the C2F module compared to other versions in the YOLO series. In the C2F module, the input first passes through a convolutional module with k=1, s=1, p=0, c=c_out, and then through a residual connection composed of Split and Concat. After being processed by n DarknetBottleneck modules, the features are concatenated and output. The residual connections here refer to the transmission of residual information through skip connections, which helps to avoid "gradient vanishing" or "gradient exploding" problems in deep neural networks.

For examples, in a layer by layer traditional neural network, the output of a layer L can be expressed as:

$$Output(L) = f(W_L \cdot Output(L-1) + b_L)$$

But as for the neural network with residual connection, the output can be expressed as:

$$Output(L) = f(W_L \cdot Output(L-1) + b_L) + Output(L-1)$$

In the YOLOv8 model, the Neck component plays a crucial role in feature extraction and fusion. The Figure 3 following diagram illustrates the operational workflow of the Neck component:

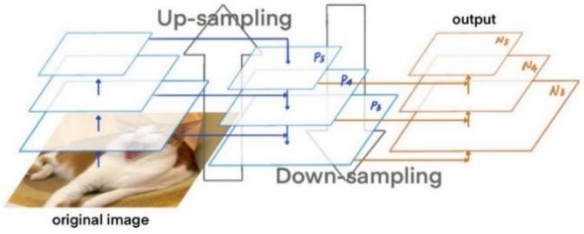

Figure 3: Flow diagram of neck part

The Neck component adopts the concept of PAN-FPN(Path Aggregation Network & Feature Pyramid Networks), merging features obtained from different layers of the Back-bone to better capture information on targets of various sizes. The Neck employs both up-sampling and down-sampling to adjust low and high-resolution images to the same size, enhancing adaptability to target detection in complex scenarios. This is one of the reasons we selected the YOLOv8 version for action recognition in surveillance images.

The Head component is the integration area of the preceding work, producing the model's output. In the YOLOv8 Head region, three decoupled-heads correspond to the three feature maps T1, T2, and T3 output by the Neck region. Each decoupled-head splits the detection head of the previous version into two branches, connecting to the loss function with four 3*3 convolutions and two 1*1 convolutions to analyze the displacement deviation be-tween the predicted and ground truth boxes. Finally, the four vertex coordinates of the predicted boxes are output.

The main network structure of YOLOv8 is as mentioned above. Next, we will share the process of applying yolov8 into surveillance.

3. Application the model

As for the application part, the main process of our work is shown in Figure 4.

Figure 4: Process of our work

3.1. Download model and configure environment

Firstly, to utilize the YOLOv8 model on our computer, we need to create a virtual environment in

Anaconda using Python 3.10.

Subsequently, download the source code and pretrained weights from the YOLOv8 open-source repository. YOLOv8 offers five model variants tailored for different tasks, namely n, s, m, l, and x. For our application in surveillance image recognition, we have chosen the YOLOv8m model, which provides a balanced trade-off between parameter size and training complexity.[3][4]

The following figure 5 shows the model weights we need to download in advance, as well as the differences in CPU response times among various versions of the YOLOv8 model:

▼ Detection (COCO)

See Detection Docs for usage examples with these models trained on COCO, which include 80 pre-trained classes.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

- **mAP^{val}** values are for single-model single-scale on COCO val2017 dataset.
  Reproduce by `yolo val detect data=coco.yaml device=0`
- **Speed** averaged over COCO val images using an Amazon EC2 P4d instance.
  Reproduce by `yolo val detect data=coco.yaml batch=1 device=0|cpu`

Figure 5: Different versions of the YOLOv8 model

After the configuration is completed, we need to verify whether it can run.

3.2. Preparation of training dataset images

In the process of preparing the dataset, we initially collected approximately 500 surveillance images to be the training set. Given that our intended application pertains to act recognition of examinees and invigilators in examination monitoring scenarios, it was imperative to categorize the labels prior to annotating the training set. We used the Co-labeller tool for labelling.

Overall, the identification of individuals needed to be classified into two major categories: upper body (denoted by 'B') and full body (denoted by 'Q'). Additionally, for specific behaviours such as drinking water, we also annotated small items like water bottles.

For the first target, which is examinee behaviour detection, action labels primarily fall into the following categories:

(1) Mobile phone-related actions: holding a phone, looking at a phone, using a phone;

(2) General actions: whispering, raising head, raising hand, turning head, lowering head, standing, bending over to pick up objects, etc.;

(3) Object-related actions: passing objects, picking up objects horizontally;

(4) Miscellaneous.

For the second target, invigilator behaviour detection, action labels mainly include: standing, picking up examination paper bag as a signal, sitting invigilation, holding security scanner, etc.

When annotating the test set images, we established the following rules:

(1) Every identifiable individual should be enclosed within a bounding box as much as possible, including all examinees and invigilators present in the photo, unless identification is difficult;

(2) When actions related to objects are present, they should be outlined with a separate bounding box within the larger bounding box of the individual;

(3) If the image contradicts common sense, for example, if an individual is depicted as standing fully but heavily obscured, they should be bounded within an upper body box, indicating that the labelling is determined primarily by the information provided in the image.

Here is Figure 6 an example when preparing the training data set:

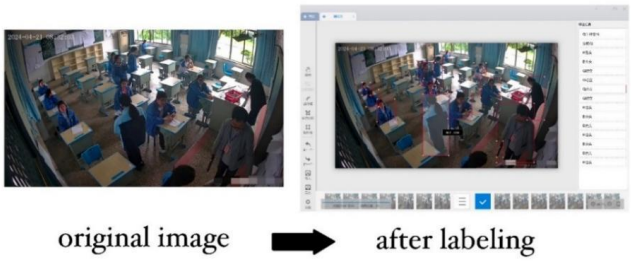

Figure 6: An example for labeling

3.3. Train the model

For the training part, the first step is to establish the path for the dataset to be trained. The processed training set images from section 4.2 will be stored in the “image” folder.

The second step involves determining the number of epochs. An "epoch" refers to the process of feeding the entire training dataset into the model once during training. Within an epoch, the model computes predictions through forward propagation, then updates its weights through backpropagation to minimize errors. Typically, the training process con-sists of multiple epochs to allow the model to gradually learn and improve its performance. To enhance the accuracy of surveillance image recognition, we set the number of epochs to 1000.

Below are the instructions for inputting during the training of our model:

yolo train model=yolov8m.yaml

data=hs_data.yaml imgsz=960

epochs=1000 batch=16 patience=100

device=0 name=kc_v8m

4. Analysis and adjustment of training results

4.1. Result analysis

Below are the visualized curves of various metrics from the training results Figure 7.

As shown in the figure, the upper part represents the parameter results of the training set, while the lower part represents the parameter results of the validation set. Each section contains 5 subplots, each depicting different losses and metrics. Since these metrics indicate the completeness of the training, we will analyze them individually in the following text.

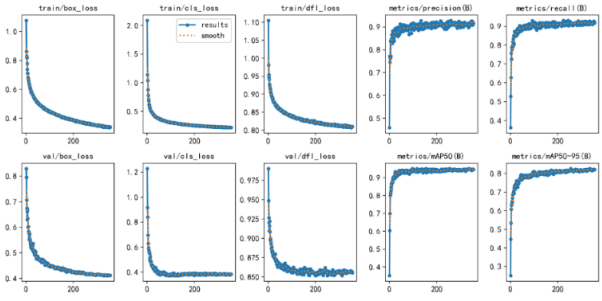

Figure 7: Visualizing training results

4.2. Training set results

Training set results are first five figures of Figure 7[5].

1) train/box_loss: This figure represents the variation of the Bounding Box Loss in the training set over the course of training iterations. YOLOv8 employs Complete Intersection over Union (CIoU) to measure bounding box loss.

$$CloU = IoU - (\frac{\rho^2(b, b^g)}{c^2} + \alpha v)$$

Where $IoU = \frac{A_I}{A_U}$, A_I is the area of the intersection between the predicted box and the ground truth box, and A_U is the area of their union. The term v measures the difference in aspect ratio between the predicted and ground truth boxes (specifically, $v = \frac{4}{\pi^2}(\arctan\frac{w^g t}{h^g t} - \arctan\frac{w}{h})^2$), and $\alpha = \frac{v}{(1-IoU)+v}$ is a scaling factor.

The downward trend of the loss values in the graph indicates that our model is becoming increasingly accurate in object localization through training.

2) train/cls_loss: This figure illustrates the changes about the classification loss in the training set over iterations. YOLOv8 uses Binary Cross Entropy (BCE) loss to measure the difference between predicted class probabilities and true class labels. The BCE is computed as follows:

$$BCE = -[y\log(p) + (1-y)\log(1-p)]$$

Where y is the true label and p is the predicted probability. The decreasing trend in the loss values in the graph suggests that our model's performance in object classification is improving.

3) Train/dfl_loss: This figure depicts the variation of the Distribution Focal Loss (DFL) in the training set. This loss is unique to YOLOv8 and indicates the model's focus in prediction. The decreasing loss values indicate an improvement in the model's performance in this aspect.

4) Metrics/precision(B): This figure shows the Precision in the training set over training iterations. Precision gradually increases, indicating that the accuracy of the model's predictions is improving. Precision is calculated as follows:

$$Precision = \frac{TP}{TP + FP}$$

where TP is the number of True Positives, FP is the number of False Positives.

5) Metrics/recall(B): This figure shows the recall in the training set over training iterations. Recall is also increasing, indicating that the model is identifying more target objects. Recall is calculated as follows:

$$Recall = \frac{TP}{TP + FN}$$

where FN is the number of False Negatives.

4.3. Validation set result

Validation set result are last five figures of Figure 7[5].

1) val/box_loss: This plot represents the variation of the bounding box loss on the validation set over training iterations. Similar to the training set, the loss value gradually decreases, indicating that our model's localization performance on the validation set is improving.

2) val/cls_loss: This plot depicts the change in classification loss on the validation set over training

iterations. The declining loss value suggests that the model's classification performance on the validation set is enhancing.

3) val/dfl_loss: This plot shows the variation in distribution focal loss (DFL) on the validation set. The loss value is also decreasing, indicating that the model's performance in this aspect is improving on the validation set.

4) metrics/mAP50(B): This plot illustrates the change in mean Average Precision (mAP) @ 50% IoU on the validation set over training iterations. mAP is a crucial metric for evaluating the overall performance of the model; higher values indicate better performance. The formula for mAP is as follows:

$$mAP = \frac{1}{Q}\sum_{q=1}^{Q} AP_q$$

where Q represents the number of queries, and AP_q is the precision for the q-th query. The formula for AP is: $AP = \sum(R_n - R_{n-1})P_n$, where P_n is the precision at the n-th threshold, and R_n is the recall at the n-th threshold.

In the plot, mAP is gradually increasing, indicating an improvement in the overall performance of the model on the validation set.

5) metrics/mAP50-95(B): This plot represents the change in mAP @ 50-95% IoU on the validation set over training iterations. This is a more stringent performance metric that encompasses the average precision across different IoU thresholds. The rising value indicates that the model's performance across various IoU thresholds is improving.

Additionally, the training output results also have the loss value corresponding to each epoch.

Overall, these plots indicate that, during the training process, the YOLOv8 model exhibits a consistent decrease in various loss metrics and an improvement in performance metrics on both the training and validation sets. This trend suggests that our trained YOLOv8 model is continually learning and optimizing, demonstrating its capability for ap-plication in surveillance image recognition.

4.4. Application

After training the YOLOv8 model, we obtained a set of weights suitable for recognizing human actions in surveillance images. To apply the model, here is an example of using the YOLOv8 model to process surveillance images according to our specified labelling requirements.

As shown in the figure 8, YOLOv8 accurately identifies targets in a surveillance image without duplication or omission. It uses precise bounding boxes to select all individuals in the image and provides the action category and probability for each target. Compared to manually labeling results, all categories identified by YOLOv8 meet the requirements. This result demonstrates that after 1000 epochs, the model possesses application-level object detection capabilities for standard surveillance images.

It is noteworthy that, as indicated by the green dashed line in the image, manual labelling of images can inevitably result in some omissions. However, YOLOv8, enhanced through deep learning, can identify uncaptured targets. This indicates that deep learning models can compensate for human errors in object detection, which is one of the reasons we aim to apply YOLOv8 in the field of surveillance imagery.

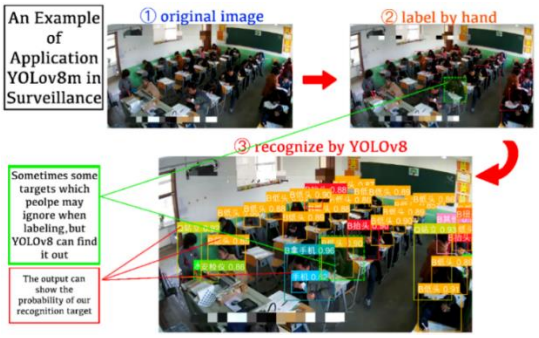

Figure 8: An example of our YOLOv8 model in application

Our primary objective is to recognize the actions of students and invigilators in exam room surveillance images. Therefore, certain behaviors require special attention. For in-stance, in the first row of the image, a student is using a mobile phone, an action that needs to be highlighted during labeling. We set the label color for this action to blue and also an-notate the special item within the main action bounding box. This facilitates subsequent invigilation efforts.

5. Model Evaluation and Further Discussion

5.1. Evaluation and Model Improvement

Our YOLOv8 model has shown excellent speed and accuracy in practical applications, but it can still be improved on the original basis to achieve the best object detection and image recognition results[6].

Here, we consider using an improved loss function to increase the convergence speed and obtain more accurate regression results.

The **MPDIoU** new bounding box similarity metric is an effective loss metric, and the principle is shown in the formula below. This simplifies the similarity comparison between two bounding boxes and can adapt to overlapping or non-overlapping bounding box regression.

A and B are two convex shapes, whose width and height are w and h respectively.

The coordinates of the upper left corner and lower right corner of A are:

$$(x_1^A, y_1^A) \ , \ (x_2^A, y_2^A);$$

Similarly, the coordinates of the upper left corner and lower right corner of B are:

$$(x_1^B, y_1^B) \ , \ (x_2^B, y_2^B);$$

We definite that:

$$d_1 = (x_1^B - x_1^A)^2 + (y_1^B - y_1^A)^2$$
$$d_2 = (x_2^B - x_2^A)^2 + (y_2^B - y_2^A)^2$$

Then our $MPDIoU = \frac{A \cap B}{A \cup B} - \frac{d_1^2}{w^2+h^2} - \frac{d_2^2}{w^2+h^2}$;

Its related loss function is $L_{MPDIoU} = 1 - MPDIoU$;

The optimal loss for a metric is the metric itself. MPDIoU loss can be used as the optimal bounding box regression loss in all applications that require 2D bounding box regression, which can improve detection accuracy, exceed existing loss functions, and greatly improve efficiency. This is worth subsequent code improvements and multiple tests of its effectiveness.

5.2. Strengths and Weaknesses

The strengths of this model are:

1) It can be effectively used in image monitoring and detection, outputting different actions and sensitive objects of people under monitoring with high accuracy.

2) Through training and testing with a large number of training sets, our yoloV8 model can make up for some of the deficiencies in manual labels, check for omissions and perform identification.

3) Our model has high practical application value and can be used in monitoring and identification in life.

The weaknesses of this model are:

1) The preliminary work is cumbersome and requires manual labeling, which may lead to errors and omissions.

2) In order to find the optimal parameters, the training burden is heavy and it takes a long time to get a feasible model.

6. Conclusions

YOLOv8, as an outstanding model in object detection algorithms, has been the focus of extensive research by computer vision and deep learning scholars. Our innovative contribution lies in applying this model to the recognition of human actions in examination room surveillance images.

In this study, we utilized the open-source code of YOLOv8, setting up the environment, analysing the model structure, interpreting the code, and collecting a large number of surveillance photos for training. This process enabled us to develop a human posture recognition model with high accuracy. The training process was complex, involving continuous parameter tuning and analysis of the loss function and recall rate. Ultimately, we achieved optimal parameters and successfully applied them to detect surveillance images.

In practical applications, the trained model not only accurately detects and identifies humans and objects in images but also compensates for the omissions of manual annotations, meeting high precision requirements. For future research, we plan to improve the model starting with the loss function, aiming to minimize feature loss and enhance operational efficiency while ensuring accuracy.

References

[1] Terven, J., & Cordova-Esparza, D. (2024, February 4). A comprehensive review of YOLO architectures in Computer Vision: From YOLOV1 to Yolov8 and Yolo-Nas. arXiv.org. https://arxiv.org/abs/2304.00501v7

[2] Detailed explanation of YOLOv8 [Network Structure + Code + Practical Implementation]. (n.d.). CSDN Blog. http://t.csdnimg.cn/78mzS

[3] Zhao Jida, Zhen Guoyong, Chu Chenqun. UAV image target detection algorithm based on YOLOv8[J]. Computer Engineering, 2024, 50(04), 113-120. https://doi.org/10.19678/j.issn.1000-3428.0068268

[4] Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2024, May 22). Real-time flying object de-tection with yolov8. arXiv.org. https://arxiv.org/abs/2305.09972v2

[5] Ma, S., & Xu, Y. (2023, July 14). MPDIoU: A loss for efficient and accurate bounding box regression. arXiv.org. https://arxiv.org/abs/2307.07662

[6] Yin Beichen, Wang Zijjian., Cheng Zhi, et al. Research on inspection robot target detection method based on improved YOLOv8 model [J]. Medical Equipment, 2024, 45(03), 1-8. https://doi.org/10.19745/j.1003-8868.2024041.