Automatic detection of dress-code surveillance in a university using YOLO algorithm

Benjamin Jason Tantra, Moeljono Widjaja
Department of Informatics, Faculty of Engineering & Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia

Article Info

Article history:

Received May 9, 2024
Revised Nov 11, 2024
Accepted Nov 24, 2024

Keywords:

Computer vision
Dress-code surveillance
Machine learning
Object detection
YOLO algorithm

ABSTRACT

Dress-code surveillance is a field that utilizes an object detection model to ensure that people wear the proper attire in workplaces and educational institutions. The case is the same within universities, where students and staff must adhere to campus clothing guidelines. However, campus security still enforces university student clothing manually. Thus, this experiment creates an object detection model that can be used in the campus environment to detect if students are wearing appropriate clothing. The model developed for this research has reached an f1-score of 45% with an overall 51.8% mean accuracy precision. With this, the model has reached a satisfactory state with room for further improvements.

Corresponding Author:

Moeljono Widjaja
Department of Informatics, Faculty of Engineering & Informatics, Universitas Multimedia Nusantara
Scientia Boulevard St., Gading Serpong, Tangerang, Banten 15810, Indonesia
Email: moeljono.widjaja@umn.ac.id

1. INTRODUCTION

Object detection is a field of machine learning, a branch of computing science designed to mimic human intelligence to automate processes [1], such as email filtering [2], cancer detection [3], and speech transcription [4]. Adoption of machine learning can be done in many fields, and thanks to its versatility, object detection has become a field of its own. Object detection itself is used to detect objects within a captured frame that aims to detect specific objects contained in digital images [5] in order to track objects in the video, as used by surveillance systems [6], self-driving cars [7], or systems to count the number of objects in a video [8]. Fields such as security [9] and quality control in manufacturing [10] use these methods to quickly automate processes that often take humans a long time to complete. All detection systems use image training data to produce a desired prediction or decision through a model that can be improved over time [11].

The issue of dress code surveillance is often overlooked in many fields. Adherence to a company's dress code helps to ensure that everyone working on the site is comfortable and safe, especially in places such as construction sites and offices [12]. However, dress code surveillance can also be used in other institutions to help ensure that employees and workers adhere to the dress code set by their company or workplace [13], [14]. In educational institutions such as Universitas Multimedia Nusantara (UMN), its dress code is written in the student handbook. Some institutions have used technology to more quickly identify the clothes worn by people within the area using webcams and machine learning models to detect different types of clothing [15]. This research aims to create a YOLO model to detect clothing articles using ultralytics' YOLOv8 architecture, using DeepFashion2 as the training, testing, and validation data set.

2. METHOD

To suit the needs of UMN's dress code, the classes within the training data set must be formatted to fit the clothing allowed and disallowed within the campus area. The preprocessing process ensures that the data requirements are met by removing data that are considered irrelevant to the original requirements of the model [16]. Within the campus grounds, the types of clothing allowed within it are the following: T-shirts, button-down shirts, long pants, shoes, flat shoes, long culottes, long skirts, blouses, and dresses. The clothing items disallowed within the campus area include sleeveless clothing, crop tops, transparent clothing, mini skirts, shorts, ripped jeans, mules, sandals, and rubber shoes. The data set used for this research is the DeepFashion2 data set, which contains 13 clothing classes that include: short sleeve tops, long sleeve tops, short sleeve outerwear, long sleeve outerwear, vest, sling, shorts, trousers, skirts, short sleeve dresses, long sleeve dresses, vest dress, and sling dress [17].

For the filtering process, the annotations for each class of clothing items are first checked through DeepFashion2's documentation for clothing annotations, which shows the different types of annotation for the clothing types within the data set. After this, the types of clothing are matched to the ones allowed and disallowed within the campus area. The classes are categorized as shown in Table 1, which follows the rules as written in the student handbook:

Table 1. Classes allowed and not allowed according to campus guidelines

Allowed	Not allowed
1: 'short sleeve top', 2: 'long sleeve top'	5: 'vest', 6: 'sling'
3: 'short sleeve outwear', 4: 'long sleeve outwear'	7: 'shorts', 12: 'vest dress'
8: 'trousers', 9: 'skirt'	13: 'sling dress'
10: 'short sleeve dress', 11: 'long sleeve dress'	

After preprocessing is completed, the JSON files used as labels need to be converted to the YOLO format, which uses the XY center of the models along with the width and height of the object; the conversion is done using four equations to calculate the `x_center`, `y_center`, `width`, and `height` from the JSON file's `bounding_box` key.

$$x\_center = \frac{(\frac{(x\_max + x\_min)}{2})}{image\_width} \tag{1}$$

In (1) is used to obtain the center X value of the object using the key `bounding_box`. The maximum and minimum x points are added and then divided by two before being subdivided from the full pixel file width of the image.

$$y\_center = \frac{(\frac{(y\_max + y\_min)}{2})}{image\_height} \tag{2}$$

In (2) is used to obtain the center y value of the object using the `bounding_box` key. The maximum and minimum y points are added and then divided by two before subdividing from the pixel height of the whole image file.

$$width = \frac{x\_max + x\_min}{image\_width} \tag{3}$$

In (3) is used to obtain the width of the bounding box by adding the minimum and maximum x values and then dividing by the pixel width of the image.

$$height = \frac{y\_max + y\_min}{image\_height} \tag{4}$$

Finally, the height of the bounding box is also obtained through in (4) that adds the minimum and maximum y values and then divides it by the pixel height value of the image.

After the calculations are performed and looped through each file of the filtered version of the DeepFashion2 data set, the folder structure is prepared for the training, testing and validation process. The file paths are then included in the file `deepfashion2.yaml`, which is used to set the above file paths and the class IDs for each type of clothing item. A class 0 named none is added because the data set uses numbers 1-13, while YOLO requires the ID index to start from 0. After choosing the file paths, the type of architecture is selected to match the hardware capabilities of the device being used. The architecture chosen to train the data is ultralytics' YOLOv8n. The architecture of YOLOv8n itself remains unchanged from the architecture obtained from ultralytics.

YOLO models have a speed advantage over object detection models created using other algorithms. YOLO is a type of CNN algorithm that uses an architecture of 24 convolutional layers and two fully connected layers [18], [19]. YOLO focuses on the speed and accuracy of the model to detect objects in digital images or videos as a regression problem [20] and gives the probability of detecting objects in the image or video as a result. The YOLO algorithm is relatively fast as the convolutional neural network used by YOLO uses single forward propagation [21] to maximize its speed, often used in public systems for physical distancing [22]. The training process is then run using the following hardware specifications: 16 GB of RAM, Processor Intel(R) Core(TM) i7-8700k CPU @ 3.70 GHz (128 GB Memory) and NVIDIA GeForce GTX1080 Ti.

After ultralytics has been set up, the `default.yaml` located within the `AppData` folder is modified to allow for further enhancements during the training process. It is applied during the training process for every batch of images. As shown in Figure 1, the augmentation displays various manipulations such as transparency, rotation, horizontal flips, vertical flips, angle, and zoom to improve the model's detection capabilities [23], created by modifying the `default.yaml` file for ultralytics by modifying a few different parameters in order to increase the model's performance, done by increasing the hue, saturation, and value (HSV)-color augmentations along with the scale and shearing of the image to make sure that the model is able to detect clothing in any situation. The augmentation is done to the existing data rather than creating new data from the augmentations; because of this, the amount of training data remains unchanged, with each clothing instance equal to the data after it was filtered during the preprocessing step.

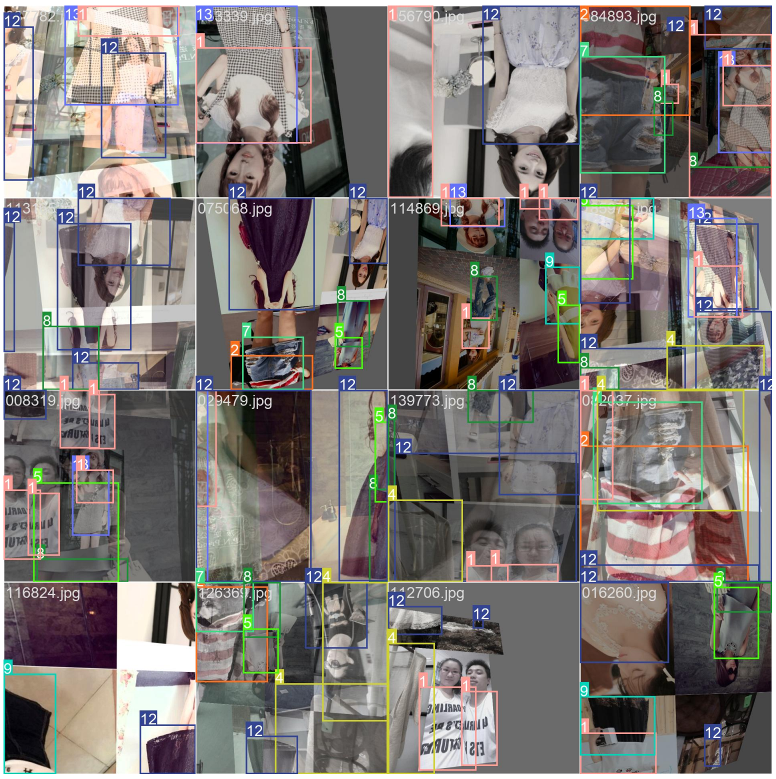


Figure 1. Samples of augmented training data

Hyperparameter tuning is also done to ensure the result of the training process can be used within any environment. The class loss is also changed to 0.4, and the distribution focal loss is changed to 1.0. The YOLO algorithm uses these loss functions to perform regression for the bounding box obtained from the training data to increase the probability that the item is correctly labeled during detection [24]. After the training process, the validation process is done using the `model.val` function from ultralytics, utilizing the validation image and JSON annotation files, written in `deepfashion.yaml`. The validation process then gives out the macro metrics based on the results from the model created.

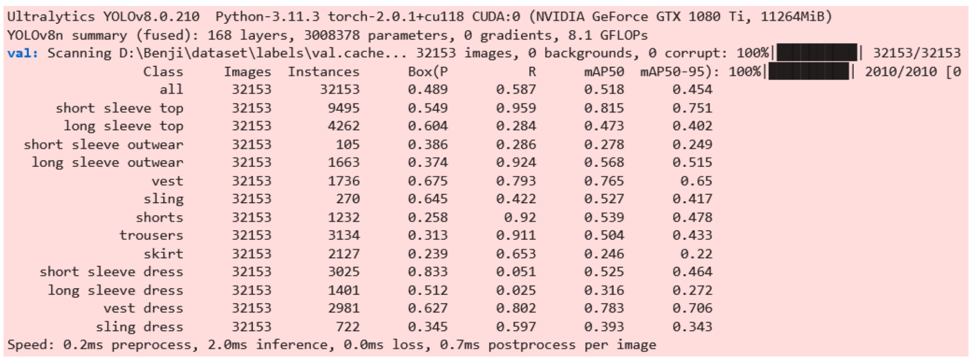
After the validation process is completed, the model then goes through a testing phase to make sure that the model can run and detect the clothing within the captured frame. This testing is then split into two main categories: the first testing on a static image and then the second through a live camera feed. The model's performance is then calculated through the precision, recall, and f1-score.

3. RESULTS AND DISCUSSION

The model created from the training process is in the form of a `.pt` file. The `.pt` file is then used for the validation and testing process, which is done to ensure that the model has good accuracy.

3.1. Validation

The validation process checks the model's quality by utilizing ultralytics' `model.val` to automatically validate the model using the validation directory included within the `deepfashion.yaml` file. The validation process also generates graphs such as the multiclass confusion matrix and the label statistics, which help during the evaluation process for the models created by the training process. Figure 2 shows the validation process results, which show the macro metrics for all classes and the metrics for each of the 13 classes. The metrics shown in the figure are the precision (P), recall (R), mean average precision (mAP) 50, and mAP50-95 (or f1-score). The results of the validation for all of the classes are in the form of macro metrics. Because the YOLO model used to detect objects has multiple different classes, the micro average metrics are chosen as the metrics used for the final evaluation. Besides the micro and macro metrics, a confusion matrix is generated to see if the model can correctly detect the items within the ground truth labels used for the validation set.



```
Ultralytics YOLOv8.0.210  Python-3.11.3 torch-2.0.1+cu118 CUDA:0 (NVIDIA GeForce GTX 1080 Ti, 11264MiB)
YOLOv8n summary (fused): 168 layers, 3008378 parameters, 0 gradients, 8.1 GFLOPs
val: Scanning D:\Benji\dataset\labels\val.cache... 32153 images, 0 backgrounds, 0 corrupt: 100%|██████████| 32153/32153
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2010/2010 [0
                   all      32153      32153      0.489      0.587      0.518      0.454
       short sleeve top      32153       9495      0.549      0.959      0.815      0.751
        long sleeve top      32153       4262      0.604      0.284      0.473      0.402
   short sleeve outwear      32153        105      0.386      0.286      0.278      0.249
    long sleeve outwear      32153       1663      0.374      0.924      0.568      0.515
                  vest      32153       1736      0.675      0.793      0.765       0.65
                 sling      32153        270      0.645      0.422      0.527      0.417
                shorts      32153       1232      0.258       0.92      0.539      0.478
              trousers      32153       3134      0.313      0.911      0.504      0.433
                 skirt      32153       2127      0.239      0.653      0.246       0.22
     short sleeve dress      32153       3025      0.833      0.051      0.525      0.464
      long sleeve dress      32153       1401      0.512      0.025      0.316      0.272
            vest dress      32153       2981      0.627      0.802      0.783      0.706
           sling dress      32153        722      0.345      0.597      0.393      0.343
Speed: 0.2ms preprocess, 2.0ms inference, 0.0ms loss, 0.7ms postprocess per image
```

Figure 2. The validation results

The results from the validation set in Figure 3 show that the model can detect objects not labeled initially. However, further evaluation of the model's results also shows how some of the model's weaknesses, for example, mistakenly detecting tops as outerwear, may affect the detection results during the testing phase. This weakness may be due to the YOLO algorithm used for the model, which focuses on speed rather than accuracy. One of the methods used to evaluate the model's results is utilizing the confusion matrix to see the accuracy of the model in detecting objects and whether or not they are correctly detecting the objects or as other classes. The results can also be normalized to get a more accurate reading on the model results [25].

Figure 4 shows that the model can detect most of the classes with relatively high accuracy, except the short-sleeve dress and long-sleeve dress classes, which is caused by the low amount of training data used for the training after the preprocessing is completed. The confusion matrix includes 15 classes, including the 'none' and 'background' classes. The background class is added to ensure that the model correctly detects background objects, not clothing items.

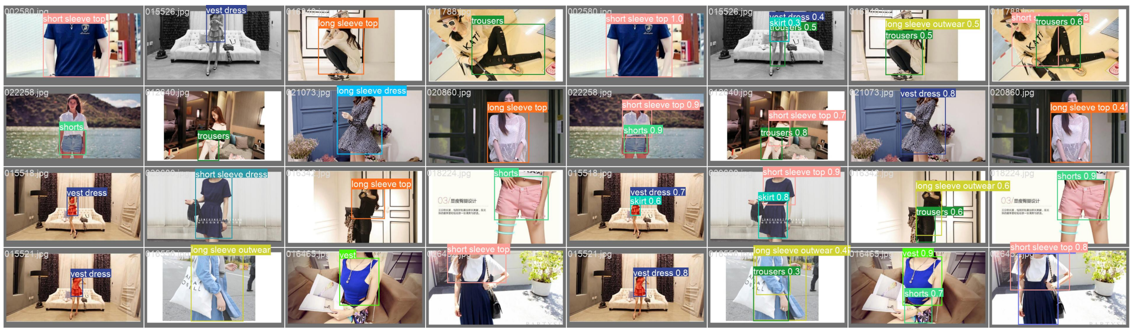


Figure 3. Ground truth labels and prediction results on the validation set

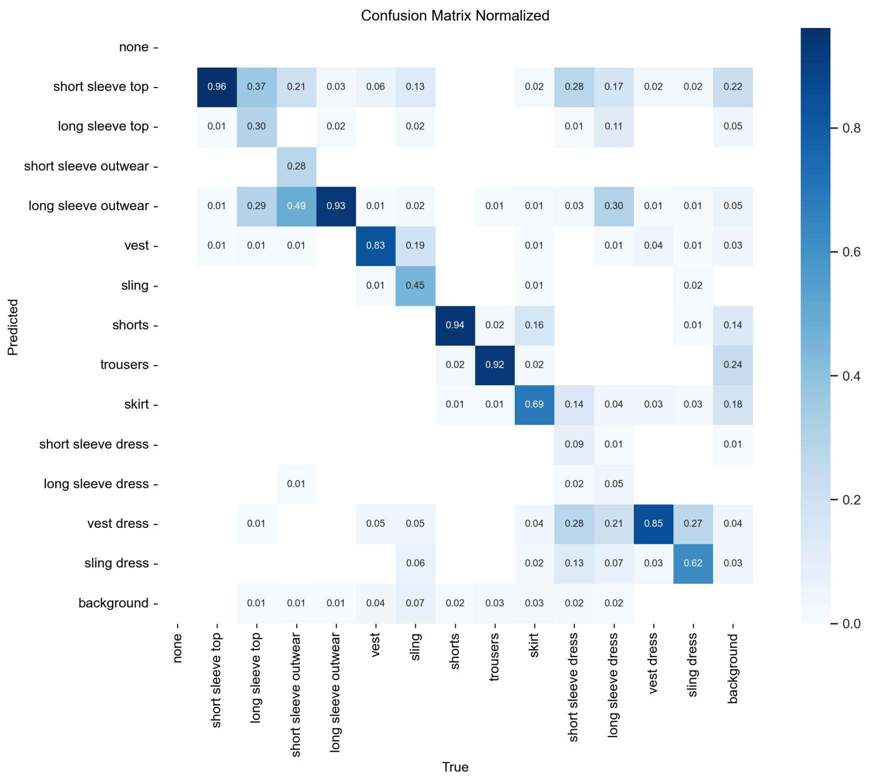
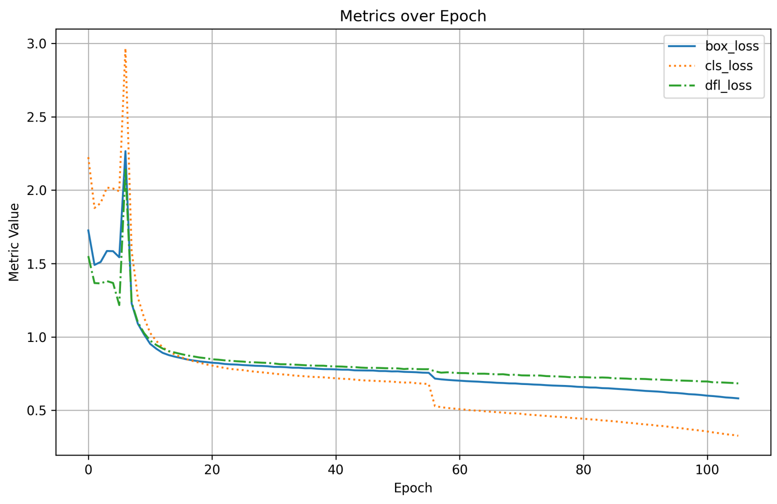


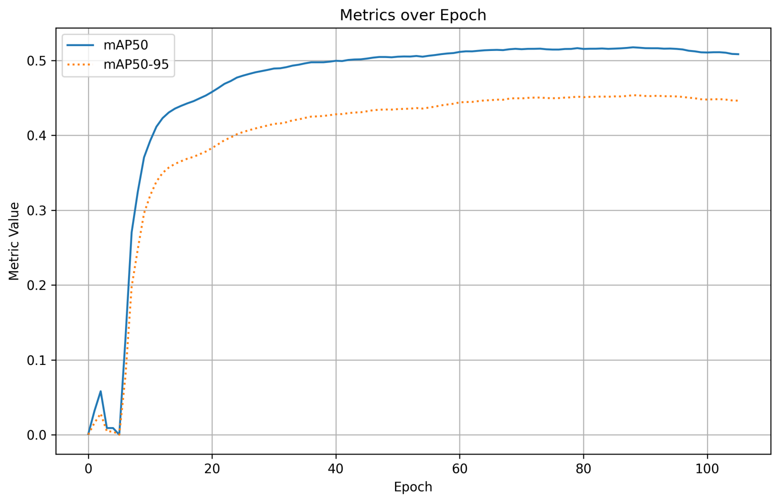
Figure 4. Normalized confusion matrix from validation process

Along with the normalized confusion matrix, a graph is also generated, as seen in Figure 5(a), with curves to show the model's metrics after every epoch and the resulting losses during the training process. The `box_loss` is a metric which displays the error between the predicted bounding box and the true bounding box. The `cls_loss` displays the difference between the predicted class and the true class of the object, and the `dfl_loss` shows the error if the model can differentiate very similar objects apart.

The results labeled with "`train`" show that the box, class, and distribution focal loss has a significant drop at epoch 50, which is done by tuning the `close_mosaic` parameter to stop 50 epochs before the training is finished; this `close_mosaic` adds a mosaic augmentation to the image batches used while training. The precision, recall and f1 score metrics can also show improvements over time, as seen in Figure 5(b). The mAP50 and the f1 score (mAP50-95) show a jump in improvement after the 50th epoch due to the `close_mosaic` being stopped. However, the final model utilizes the metrics with the best metrics rather than the one that uses the model's final epoch. Overall, the validation process shows that the model shows promise with a decent capability to detect the different object classes with about 50% f1-score.



(a)



(b)

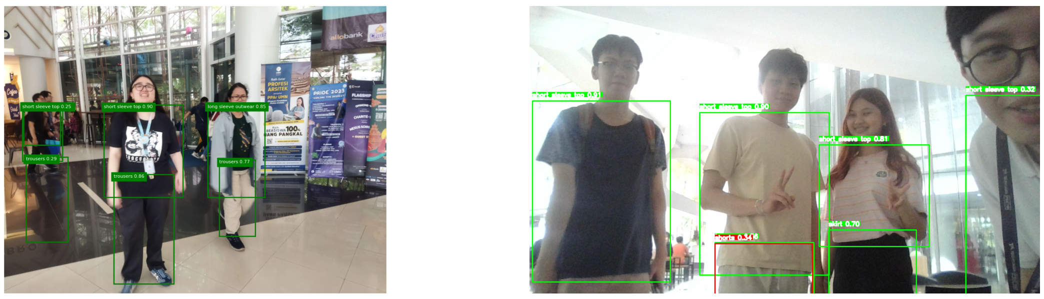
Figure 5. Results for loss and accuracy metrics after every epoch: (a) box_loss, cls_loss, and dfl_loss score metrics; and (b) the precision, recall, and f1 score metrics

3.2. Testing

The model testing is conducted in two main parts to thoroughly check if the model can comprehensively detect different object classes. The two tests are run in the four different buildings to ensure that they can run in every environment within the campus area. The proficiency of the model is then evaluated by checking if it can detect the clothing item correctly in the correct positions and the confidence of the detection done by the model; this allows for a more thorough evaluation of the model, which checks if it can detect the correct clothing in the different environments of the campus.

Figure 6(a) is a static image test done using a picture taken in the UMN building B area; once again, the model can detect the different clothing items with high enough confidence. However, an error can be seen with the model detecting the reflection of a person on the floor instead of the person itself. Despite the low confidence, it shows that there may still be some improvements to be made to make sure that the model is detecting the proper items instead of reflections.

Next, the live-feed camera test is conducted in UMN's building C, as shown in Figure 6(b), utilizing the camera script and the model created through the previous processes. The live-feed camera tests require the use of the OpenCV library. The camera script creates a for loop that checks for objects within the captured frame and draws a bounding box using cv2's rectangle function if an object within the model is detected. The model has shown that it can detect clothing items correctly except for a few errors, which may be due to the inherent weakness of the YOLO algorithm due to its prioritization of speed over accuracy, which may lead to some mistakes in detecting objects.



(a) (b)

Figure 6. Static and dynamic image testing using a picture taken in (a) UMN's building B area and (b) UMN's building C area

4. CONCLUSION

The model created through this process has shown good functionality with a few errors due to the algorithm's limitations. Data augmentation is performed by editing the `default.yaml` file by modifying the `scale`, `shear`, and `flip` parameters within the file. In addition to this, hyperparameter tuning is performed during the training step by changing the `cls` and `dfl` properties. Changing it allows further modifications to the training process' loss and the augmentation of the training data. The final metrics result in the model with a precision of 53.5%, a recall of 44.6%, and an f1-score of 48.6%. Although the model is functional, some features still need to be added, such as identifying and detecting footwear and accessories such as the student ID necklace. Further improvements to the model can also be done using architectures with more parameters, such as YOLOv8l or YOLOv8x.

ACKNOWLEDGEMENTS

This research is fully funded by Universitas Multimedia Nusantara as part of undergraduate thesis work.

REFERENCES

[1] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14413–14423, 2020, doi: 10.1109/TVT.2020.3034800.

[2] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, 2019, doi: 10.1016/j.heliyon.2019.e01802.

[3] S. Johan, F. Natalia, F. V. Ferdinand, and S. Sudirman, "Image-based skin cancer early detection using CNN algorithm," in *2021 6th International Conference on New Media Studies (CONMEDIA)*, 2021, pp. 78–83, doi: 10.1109/CONMEDIA53104.2021.9616993.

[4] J. Padmanabhan and M. J. J. Premkumar, "Machine learning in automatic speech recognition: A survey," *IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India)*, vol. 32, no. 4, pp. 240–251, 2015, doi: 10.1080/02564602.2015.1010611.

[5] Y. Amit, P. Felzenszwalb, and R. Girshick, "Object detection," in *Computer Vision*, Cham: Springer International Publishing, 2021, pp. 875–883, doi: 10.1007/978-3-030-63416-2_660.

[6] S. Jha, C. Seo, E. Yang, and G. P. Joshi, "Real time object detection and trackingsystem for video surveillance system," *Multimedia Tools and Applications*, vol. 80, no. 3, pp. 3981–3996, 2021, doi: 10.1007/s11042-020-09749-x.

[7] P. N. Huu, Q. P. Thi, and P. T. T. Quynh, "Proposing lane and obstacle detection algorithm using YOLO to control self-driving cars on advanced networks," *Advances in Multimedia*, vol. 2022, 2022, doi: 10.1155/2022/3425295.

[8] A. A. Alsanabani, M. A. Ahmed, and A. M. Al Smadi, "Vehicle counting using detecting-tracking combinations: A comparative analysis," in *2020 The 4th International Conference on Video and Image Processing*, 2020, pp. 48–54, doi: 10.1145/3447450.3447458.

[9] S. Narejo, B. Pandey, D. E. Vargas, C. Rodriguez, and M. R. Anjum, "Weapon detection using YOLO v3 for smart surveillance system," *Mathematical Problems in Engineering*, vol. 2021, 2021, doi: 10.1155/2021/9975700.

[10] H. M. Ahmad and A. Rahimi, "Deep learning methods for object detection in smart manufacturing: A survey," *Journal of Manufacturing Systems*, vol. 64, pp. 181–196, 2022, doi: 10.1016/j.jmsy.2022.06.011.

[11] J. Bell, "What is machine learning?," in *Machine Learning and the City*, Wiley, 2022, pp. 207–216, doi: 10.1002/9781119815075.ch18.

[12] C. Wei and X. Yang, "Dress code surveillance at power grid construction site via object detection," in *2021 3rd International Conference on Electrical Engineering and Control Technologies (CEECT)*, 2021, pp. 203–207, doi: 10.1109/CEECT53198.2021.9672656.

[13] L. S. Santos and P. Marasigan, "Dress code policy adherence and self-discipline of university students," *International Review of Social Sciences Research*, vol. 1, no. 4, pp. 47–71, 2022, doi: 10.53378/352595.

[14] D. Agarwal, P. Gupta, and N. G. Eapen, "A framework for dress code monitoring system using transfer learning from pre-trained YOLOv4 model," in *2023 11th International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP)*, 2023, pp. 1–5, doi: 10.1109/ICETET-SIP58143.2023.10151460.

[15] J. Rebekah, D. C. J. W. Wise, D. Bhavani, P. A. Regina, and N. Muthukumaran, "Dress code surveillance using deep learning," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2020, pp. 394–397, doi: 10.1109/ICESC48915.2020.9155668.

[16] Z. Pei, L. Liu, C. Wang, and J. Wang, "Requirements engineering for machine learning: A review and reflection," in *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, 2022, pp. 166–175, doi: 10.1109/REW56159.2022.00039.

[17] Y. Ge, R. Zhang, X. Wang, X. Tang, and P. Luo, "Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5332–5340, doi: 10.1109/CVPR.2019.00548.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[19] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of YOLO algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2021, doi: 10.1016/j.procs.2022.01.135.

[20] H. Rodríguez-Rangel, L. A. Morales-Rosales, R. Imperial-Rojo, M. A. Roman-Garay, G. E. Peralta-Peñuñuri, and M. Lobato-Báez, "Analysis of statistical and artificial intelligence algorithms for real-time speed estimation based on vehicle detection with YOLO," *Applied Sciences*, vol. 12, no. 6, 2022, doi: 10.3390/app12062907.

[21] R. Moezzi, D. Krcmarik, J. Cýrus, H. Bahri, and J. Koci, "Object detection using microsoft hololens by a single forward propagation CNN," in *Proceedings of the International Conference on Intelligent Vision and Computing (ICIVC 2021)*, 2022, pp. 507–517, doi: 10.1007/978-3-030-97196-0_42.

[22] F. R. R. Irvine, F. Natalia, S. Sudirman, and C. S. Ko, "Automated physical distancing monitoring using YOLOv3," *ICIC Express Letters, Part B: Applications*, vol. 14, no. 8, pp. 869–876, 2023, doi: 10.24507/icicelb.14.08.869.

[23] P. Kaur, B. S. Khehra, and E. B. S. Mavi, "Data augmentation for object detection: A review," in *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2021, pp. 537–543, doi: 10.1109/MWSCAS47672.2021.9531849.

[24] X. Li *et al.*, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," in *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020, pp. 1–11.

[25] D. Krstinić, M. Braović, L. Šerić, and D. Božić-Štulić, "Multi-label classifier performance evaluation with confusion matrix," in *Computer Science & Information Technology*, 2020, pp. 1–14, doi: 10.5121/csit.2020.100801.

BIOGRAPHIES OF AUTHORS

Benjamin Jason Tantra received a bachelor's degree in computer science from the Department of Informatics, Universitas Multimedia Nusantara. He currently works in the banking industry as a data scientist. His main interests are in statistics, data analytics and artificial intelligence with a focus on object recognition. He can be contacted at email: benjamin.jason@student.umn.ac.id.

Moeljono Widjaja received a doctoral degree in electrical engineering from Monash University (Australia). He obtained a bachelor degree in electrical engineering from Wayne State University (USA) in 1992 and a master degree in electrical engineering from the Ohio State University (USA) in 1994. He is an Assistant Professor at the Department of Informatics, Universitas Multimedia Nusantara. His main research interests are artificial intelligence, simulation/modeling, and optimization. He developed a fuzzy controller for an inverted pendulum system and a fuzzy-based bidding strategy for generators in an electricity market. He has been working on intelligent energy management systems. He is a professional member of both ACM and IEEE. He can be contacted at email: moeljono.widjaja@umn.ac.id.