S & M 3838

Integrations of LabelImg, You Only Look Once (YOLO), and Open Source Computer Vision Library (OpenCV) for Chicken Open Mouth Detection

Hongxiang Ke,[1] Huoyou Li,[2*] Beizhan Wang,[3] Qing Tang,[4]
Yang-Han Lee,[5] and Cheng-Fu Yang[6,7**]

[1]Zhangzhou College of Science and Technology, Zhangzhou 363200, China
[2]School of Mathematics and Information Engineering, Longyan University, Fujian 364012, China
[3]School of Informatics, Xiamen University, Xiamen 361005, China
[4]Fujian Health College, Fuzhou 350101, China
[5]Electrical and Computer Engineering, Tamkang University, New Taipei City 251, Taiwan
[6]Department of Chemical and Materials Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan
[7]College of Artificial Intelligence, Yango University, Fuzhou, Fujian 350015, China

(Received April 30, 2024; accepted September 11, 2024)

Keywords: chickens, LabelImg, YOLOv3, YOLOv4, OpenCV

In the poultry industry, preventing chickens from developing illnesses during hot summer days is a significant concern. The early detection of abnormal behavior in chickens is crucial for farmers to promptly monitor their health status, and hence, research in this direction is paramount. In this paper, we proposed an automatic image recognition method, employing various software combinations, to analyze whether chickens exhibit abnormal mouth opening. The study began by obtaining images of chickens on the farm, capturing and observing any instances of mouth opening. The training process involved utilizing an image annotation tool, LabelImg, to mark instances of chickens and their mouth openings. Subsequently, the You Only Look Once (YOLO) algorithms, YOLOv3 and YOLOv4, were employed to train on images of chickens with open mouths. Once the YOLO training was completed, the Open Source Computer Vision Library (OpenCV) was used to read and access videos and images, enabling the observation of output results to determine the presence of mouth opening phenomena. In this approach, we integrated an advanced technology to automate the monitoring process, potentially providing farmers with timely alerts regarding any health issues in their chicken population. By leveraging computer vision and machine learning techniques, this method enhances efficiency and accuracy in detecting abnormal behavior, contributing to improved poultry health management strategies.

1. Introduction

Birds (including chickens) under hot and stressful conditions tend to reduce their food intake, exhibit signs of thirst and panting, and often spend most of their time resting with wings spread.

[*]Corresponding author: e-mail: huoyouli@126.com
[**]Corresponding author: e-mail: cfyang@nuk.edu.tw
https://doi.org/10.18494/SAM5108

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

The simplest way to determine if chickens are experiencing these issues is to observe whether they keep their beaks open. In the past, manual observation was primarily used for this purpose, but now, intelligent imaging can be utilized for automated monitoring.(1,2) The Open Source Computer Vision Library (OpenCV) provides numerous functionalities and tools for processing image and video data, executing visual tasks, and implementing machine vision applications, among others.(3,4) Some key advantages of OpenCV include the following:

1. OpenCV is an open-source project, allowing free usage, and it can run on various operating systems, including Windows, Linux, and macOS.
2. OpenCV offers a rich set of functionalities and tools, including image processing and analysis, object detection, feature extraction, image segmentation, and stereo vision.
3. OpenCV is implemented using C/C++ and provides interfaces for languages such as Python and Java, enabling fast image processing and analysis on various hardware platforms.
4. OpenCV has extensive documentation, tutorials, and an active developer community. This function makes it easy for developers to learn and use OpenCV.
5. OpenCV offers many modules and libraries, supporting custom extensions and plugins. This function allows developers to extend OpenCV's functionality according to their needs.
6. OpenCV finds extensive applications in various fields, including machine vision, image processing, image analysis, machine learning, artificial intelligence, robotics, and security surveillance.

The You Only Look Once (YOLO) is an advanced object detection algorithm that offers several advantages over traditional convolutional neural-network-based object detection methods, as follows:(5,6)

1. The primary advantage of the YOLO algorithm is its extremely high processing speed. Compared with CNN-based object detection methods, YOLO can achieve real-time object detection by combining detection and classification, requiring only a single forward pass.
2. YOLO is an end-to-end training approach that enables the direct prediction of object positions and categories from raw images without requiring additional postprocessing steps.
3. YOLO performs object detection and classification through a single forward pass, allowing it to better capture global contextual information and understand the entire image.
4. The YOLO algorithm utilizes multiscale feature maps for object detection, enabling it to detect objects of different scales and sizes, thereby improving detection accuracy and robustness.
5. Despite its speed, YOLO also exhibits excellent detection accuracy. It can achieve highly accurate object detection while maintaining high processing speeds.

LabelImg is a powerful and user-friendly open-source image annotation tool, boasting several key features, as follows:(7,8)

1. LabelImg can run on multiple major operating systems, including Windows, macOS, and Linux, making it suitable for various development environments.
2. LabelImg features an intuitive user interface, allowing users to easily annotate objects or regions within images and assign labels to them.
3. LabelImg supports various label formats such as Pascal VOC, YOLO, and TensorFlow, among others, enabling users to conveniently integrate annotated datasets with different machine learning frameworks.

4. Through customizable hotkeys, users can annotate more efficiently, swiftly completing the annotation of large numbers of images.
5. Users can effortlessly scale images, enabling the more detailed annotation of small objects or regions.
6. LabelImg incorporates an automatic saving feature, preventing data loss due to sudden power outages or unexpected application closures.

OpenCV is widely utilized across various fields, including machine vision, image processing, image analysis, machine learning, artificial intelligence, robotics, security surveillance, and more. The YOLO algorithm simplifies the object detection task into a single neural network model and fully utilizes global information for prediction, thereby achieving high real-time performance, accuracy, and stability, making it an important advanced algorithm in the field of object detection. LabelImg is a powerful, user-friendly, and free image annotation tool suitable for various image annotation needs, and is one of the commonly used tools for many researchers. By integrating these tools, a complete workflow for object detection and image processing can be established. From image annotation using LabelImg to training YOLO models with annotated data, and finally, utilizing OpenCV for image preprocessing and postprocessing, the entire process can be streamlined and optimized. Additionally, the integration allows for customized processing based on specific application requirements, enhancing efficiency in various tasks related to computer vision and image analysis. Therefore, the main innovation of this study lies in integrating LabelImg, different YOLO algorithms (YOLOv3 and YOLOv4), and OpenCV to create a powerful solution for object detection and image processing tasks. This technology's algorithm is utilized to identify chickens and annotate the mouths of chickens exhibiting mouth gaping behavior, observing whether such behavior is present. The technologies investigated in this study can serve as an early indicator of the chickens' health status.

2. Experimental Processes

LabelImg, YOLO (YOLOv3 and YOLOv4), and OpenCV are commonly used tools and technologies in the fields of object detection and computer vision. Each serves different purposes but can be integrated to achieve more efficient and accurate object detection systems. To begin, LabelImg was used to annotate image data, marking objects of interest within the images. Then, this annotated data was utilized to train the YOLO model, a state-of-the-art object detection algorithm known for its speed and accuracy. Once trained, OpenCV was employed to preprocess the images to be tested, such as resizing, normalization, and other transformations, before feeding them into the YOLO model for object detection. Following detection, OpenCV could be utilized again for postprocessing tasks, such as drawing bounding boxes around detected objects, displaying detection results, and performing other necessary adjustments or analyses. By integrating these tools and technologies, highly efficient and accurate object detection systems can be realized, applicable across various domains and applications within computer vision. In this study, we utilized neural network computations, and to expedite the results, training was conducted on computer hardware and software listed in Table 1, and the software core and related algorithms are presented in Table 2.

Table 1
Computer software and hardware utilized for neural network computations.

Software platform		Hardware platform	
Operating system	Windows	GPU	NVIDIA RTX 2080Ti
Deep learning framework	Darknet	CPU	Intel Core i7-3770
		RAM	16 GB

Table 2
Software core and related algorithms.

Software or algorithm	Version	
YOLO	3 or 4	Image recognition algorithms
OpenCV	3.4.10	Image output analysis software
LabelImg	1.8.0	Labeling tools
CUDA	10.1	NVIDIA's computational platform
cuDNN	7.6.4	GPU-accelerated libraries for deep neural network design

YOLO transforms the problem of object detection into a regression problem, where given an input image, it directly predicts bounding boxes and their associated class labels at multiple locations within the image. YOLO is a convolutional neural network capable of predicting multiple bounding box positions and class labels in a single pass, enabling end-to-end object detection and recognition. Its major advantage lies in its speed while maintaining a high mean average precision (mAP). For the detection process of the YOLO system, refer to Refs. 9 and 10.

3. Workshop Production Planning Simulation Modeling

The annotation process involved the utilization of LabelImg, wherein a total of 2011 images depicting both normal and open-mouthed chickens were meticulously annotated. Notably, out of these, 511 images showcased chickens with open mouths, indicating potential anomalies or specific behaviors. Subsequently, the YOLOv3 model underwent training for 6000 and 10000 batches, whereas the YOLOv4 model underwent training for 6000 batches, ensuring comprehensive learning from the annotated dataset. To facilitate annotation, the videos obtained from egg-laying chicken farms were meticulously processed by segmenting them into individual frames. Each frame was meticulously annotated using an image labeling tool, as demonstrated in Fig. 1. Annotations were made meticulously for different angles and species, ensuring the model's robustness across various scenarios, as illustrated in Figs. 2(a) and 2(b). Furthermore, special attention was given to chickens exhibiting open mouths. Annotations around the mouth region were carefully made to ensure the accurate detection and recognition of this specific characteristic. This meticulous annotation process enhances the model's capability to discern and classify open-mouthed chickens accurately, potentially contributing to improving the monitoring and management practices within poultry farming environments.

The annotation of images using LabelImg results in the creation of an XML file upon saving the annotations. This file contains the coordinates of the annotated regions within the images. Since the generated file is in XML format, it needs to be converted into the format required by YOLO. Next, the locations of the images should be stored in TXT files, and two separate TXT

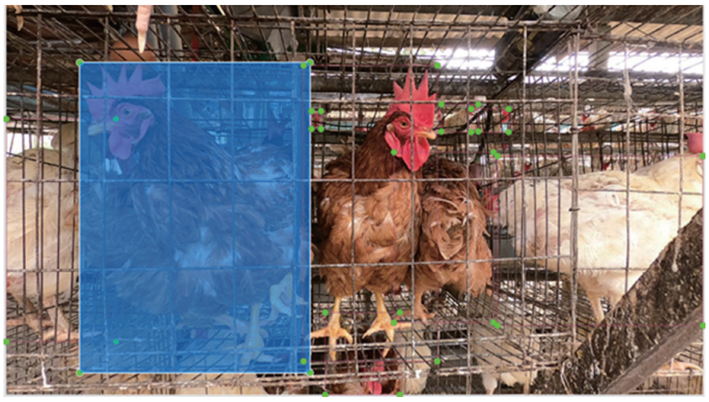
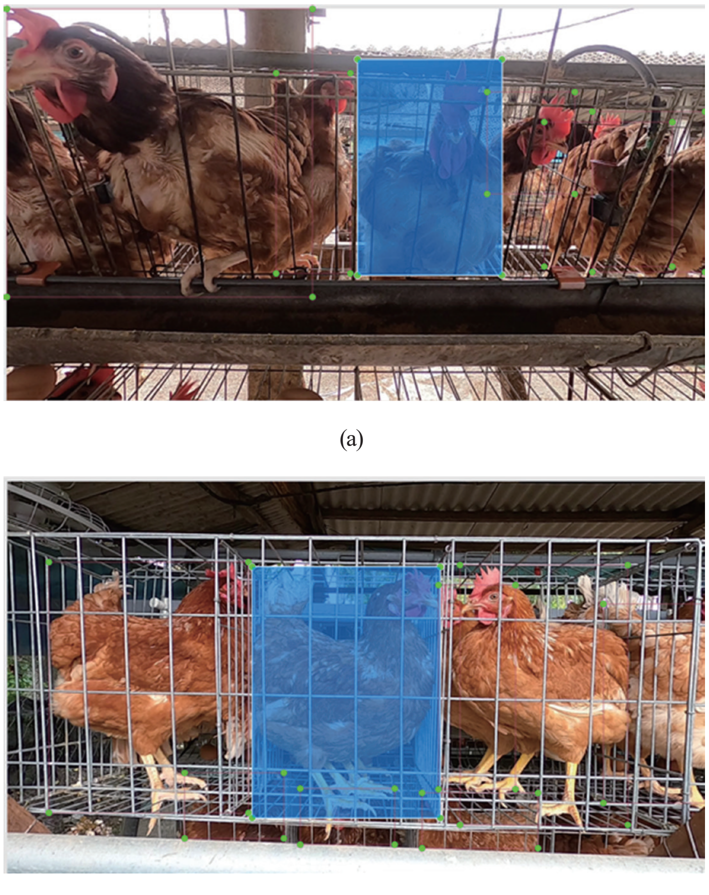


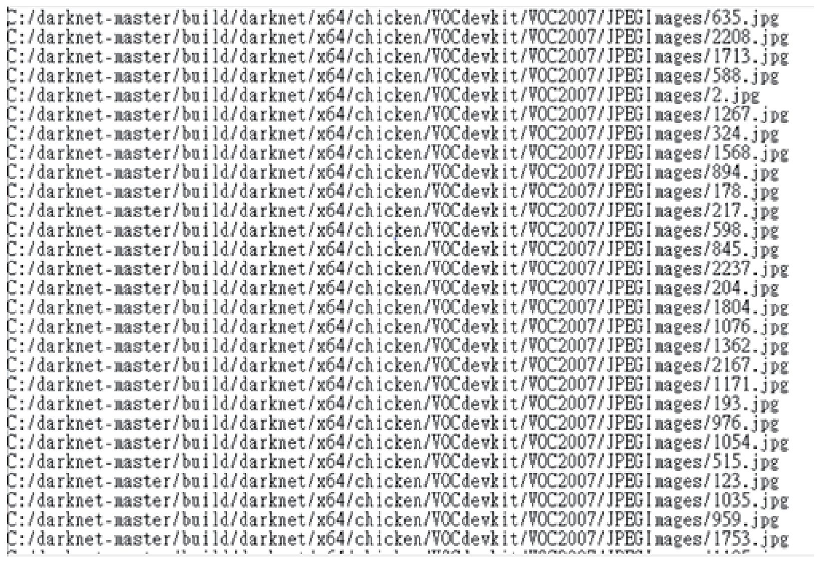
Fig. 1. (Color online) Annotation of chicken image.


(a)

(b)

Fig. 2. (Color online) (a) and (b) Annotations of different chicken species.

files, one for training and one for testing, must be saved. Once this process is completed, as illustrated in Fig. 3, the training of chicken images can commence.



```
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/635.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/2208.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1713.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/588.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/2.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1267.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/324.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1568.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/894.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/178.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/217.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/598.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/845.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/2237.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/204.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1804.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1076.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1362.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/2167.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1171.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/193.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/976.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1054.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/515.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/123.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1035.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/959.jpg
C:/darknet-master/build/darknet/x64/chicken/VOCdevkit/VOC2007/JPEGImages/1753.jpg
```

Fig. 3. Addresses of images' storage for training.

During training, there were three regions, namely, Regions 82, 94, and 106, each responsible for predicting different sizes of boxes. Region 82 represented the largest prediction scale, capable of predicting smaller marked objects. Region 94 corresponded to a moderate prediction scale, capable of predicting medium-sized objects. Finally, Region 106 represented the smallest prediction scale, able to predict larger marked objects. Throughout the training process, the output of each training batch displayed the total loss, average loss, current learning rate, time taken for each batch, and the total number of training images. Using OpenCV to output identified images, there were both training and testing data. The trained model was used to detect images and videos to observe its training results and accuracy. The accuracy indicators used by YOLO were intersect over union (IOU) and mAP. IOU is a standard for measuring the accuracy of detecting corresponding objects in a dataset, and it can be used to determine the accuracy between the predicted bounding box and the ground truth bounding box based on the model's detection results in images. IOU measures accuracy by calculating the IOU of the predicted and ground truth bounding boxes. Typically, a threshold of 0.5 is used, where an IOU greater than 0.5 for the predicted bounding box in Fig. 4 indicates good prediction.

By training with the same data and parameters repeatedly to obtain the optimal model, Tables 3 and 4 show the number of images used in training and the training parameters, respectively. In this paper, we compared the results of training recognition models using YOLOv3 and YOLOv4, followed by comparing their performance. One notable challenge encountered during result analysis is the difficulty in obtaining an adequate number of images featuring chickens with open beaks. The scarcity of such images leads to a situation where the average precision for open beaks approached zero, consequently causing a direct negative impact on the overall mAP, resulting in approximately the halving of the overall mAP. The root cause of this issue can be traced back to the data collection phase.

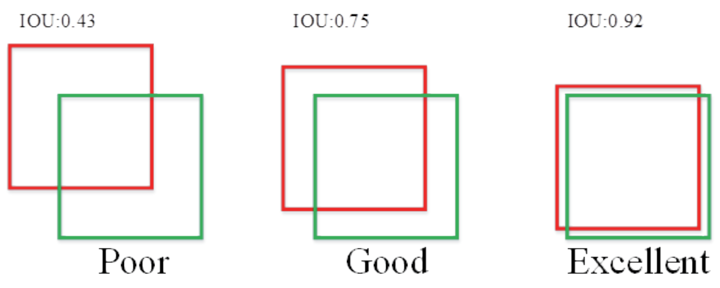


Fig. 4. (Color online) Illustration of IOU prediction standard.

Table 3
Number of training images.

Images taken for egg-laying hens	1500
Images of chickens with their mouths open	511
Images used for training	831
Images used for testing	180
Combined total images	2011

Table 4
Training parameters.

Batch	64
Subdivision	160
Width	608
Height	608
Learning rate	0.001

The insufficient representation of images with open beaks during training hindered the models' ability to effectively learn features associated with this category, resulting in decreased detection accuracy for open beaks. The shortage of images may stem from the rarity of chickens exhibiting open beaks in real-world scenarios or limitations encountered during data collection, such as time constraints, cost, or resource availability, which made gathering such images challenging. However, the situation improved significantly when a sufficient number of images featuring open beaks, totaling 511, were successfully acquired. In this scenario, through the effective labeling and normalization of open beak images, there is a notable enhancement in the overall mAP performance. This enhancement underscores the critical importance of having an adequate dataset for training effective models and emphasizes the significance of proper data collection and annotation in enhancing the performance of target detection models.

Next, we compared the results of training with YOLOv3 for 6000 batches, YOLOv3 for 10000 batches, and YOLOv4 for 6000 batches. Figure 5 shows the variation of loss with the number of batches trained using YOLOv3 for 6000 batches, whereas Fig. 6 shows the relationship between IOU and the number of batches trained with YOLOv3 for 6000 batches. Additionally, Fig. 7 shows the relationship between mAP and the number of batches trained using YOLOv3 for 10000 batches. Other relevant figures are not shown here, but the main results are shown in

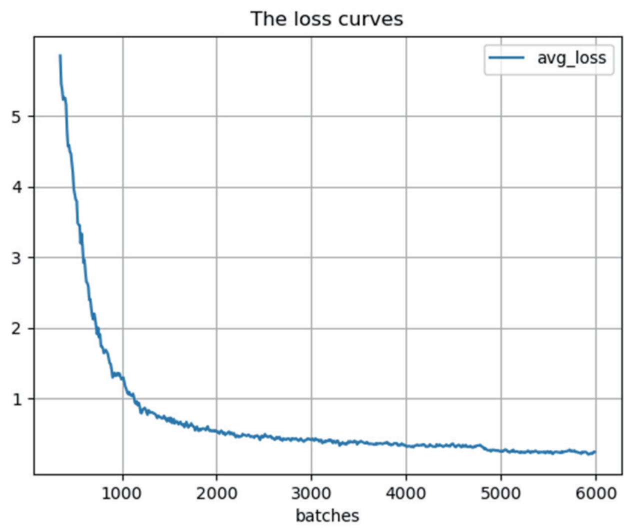


Fig. 5. (Color online) Relationship between loss and the number of batches trained using YOLOv3 for 6000 batches.

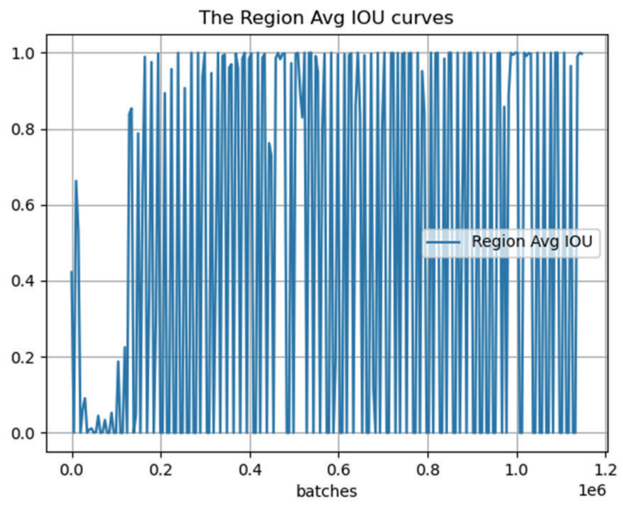


Fig. 6. (Color online) Relationship between IOU and the number of batches trained using YOLOv3 for 6000 batches.

Table 5. Table 5 provides detailed insights into the recall, precision, mAP, IOU, and time results for training with YOLOv3 for 6000 batches, YOLOv3 for 10000 batches, and YOLOv4 for 6000 batches. This comprehensive overview allows for a thorough comparison of the performance metrics across different training scenarios, shedding light on the effectiveness of each model configuration.

The results in Table 5 clearly demonstrate the differences in recall, precision, mAP, and IOU between three different training scenarios: YOLOv3 for 6000 batches (training time: 10 h), YOLOv3 for 10000 batches (training time: 16 h), and YOLOv4 for 6000 batches (training time: 6.5 h). The table indicates that at an IOU threshold of 0.75, the recall, precision, mAP, and IOU values are lower than those at an IOU threshold of 0.5. The potential reasons for these differences may include the following:

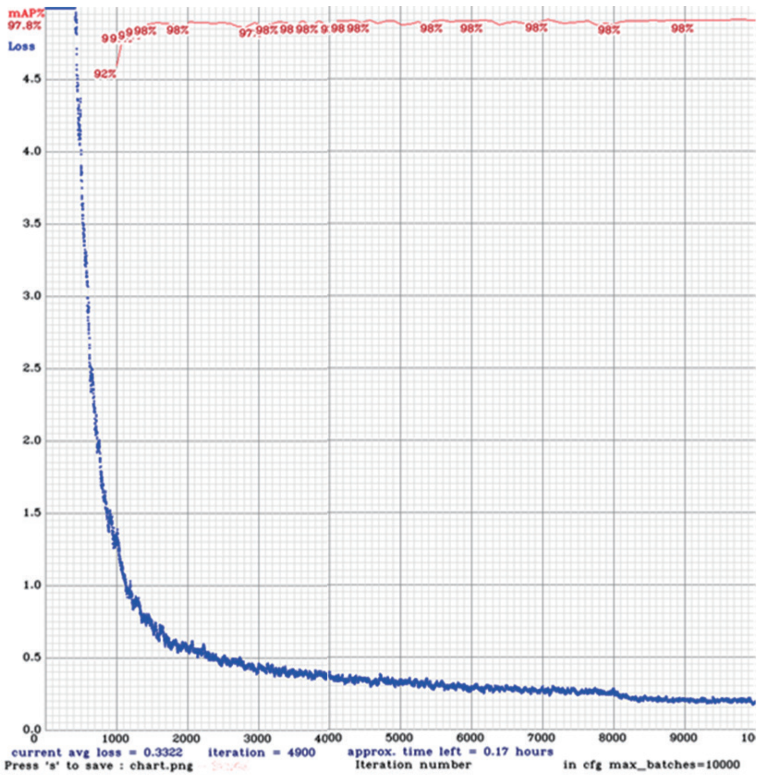


Fig. 7. (Color online) Relationship between mAP and the number of batches trained using YOLOv3 for 10000 batches.

Table. 5
Comparison of results for training with YOLOv3 for 6000 batches, YOLOv3 for 10000 batches, and YOLOv4 for 6000 batches.

	YOLOv3 6000 batches		YOLOv3 10000 batches		YOLOv4 6000 batches	
	IOU = 0.5 (%)	IOU = 0.75 (%)	IOU = 0.5 (%)	IOU = 0.75 (%)	IOU = 0.5 (%)	IOU = 0.75 (%)
Recall	73	44	94	83	93	85
Precision	93	57	95	84	92	85
IOU	71.8	47.7	80.5	72.8	80.1	74.9
mAP	85.0	36.9	97.0	80.8	96.8	86.0
Time	10 h		16 h		6.5 h	

1. Impact of training time and batch size: YOLOv3 requires 10 h for 6000 batches and 16 h for 10000 batches, suggesting varying degrees of convergence. Conversely, YOLOv4 achieves training in only 6.5 h for 6000 batches, indicating either faster training or superior convergence within fewer batches.
2. Differences in model architecture: YOLOv4, being an updated version, likely exhibits better feature extraction capabilities and model intricacies, hence outperforming YOLOv3 under similar training conditions. YOLOv3 might not have had sufficient time to achieve results comparable to YOLOv4.

3. Impact of IOU threshold: Varied IOU thresholds can yield different evaluation outcomes. A higher IOU threshold demands more overlap between detected bounding boxes and ground truth labels, potentially leading to stricter requirements and lower evaluation scores. At an IOU value of 0.75, the detector must provide more precise detections, which might result in decreased recall, precision, and mAP, especially with limited training time.
4. Effects of hyperparameters and dataset: Different hyperparameter settings and dataset characteristics may lead to varied model performance under different training conditions. Parameters such as learning rate, batch size, and data augmentation strategies can affect training speed and overall performance.

These factors collectively contribute to the observed differences in model performance across different training scenarios. Further experimentation and analysis are necessary to comprehensively understand the underlying causes of these variations.

When IOU = 0.5, after 16 h of training with 10000 batches, YOLOv3 achieved a slightly higher mAP value than YOLOv4 trained for 6.5 h with 6000 batches. YOLOv3 and YOLOv4 are both real-time object detection algorithms, but they differ in training methodology and performance. YOLOv3 utilizes Darknet-53 as its backbone network, whereas YOLOv4 employs CSPDarknet53, an enhanced version of Darknet-53 incorporating CSPNet and SPPNet for improved performance. In terms of training, YOLOv3 adopts multi-scale training, whereas YOLOv4 employs single-scale training. Multi-scale training enhances the model's detection accuracy for objects of various sizes but requires more training time. The reason why YOLOv3 achieves a slightly higher mAP value than YOLOv4 at IOU = 0.5 may be attributed to its use of multi-scale training, whereas YOLOv4 uses single-scale training. Multi-scale training enhances the model's detection accuracy for objects of different sizes; hence, at IOU = 0.5, YOLOv3's mAP value might be slightly higher than that of YOLOv4. Furthermore, YOLOv3's longer training duration allows the model to learn more features, thereby improving its mAP value. Additionally, other factors such as dataset quality and hyperparameter settings may affect the mAP values of YOLOv3 and YOLOv4. In the experiment, YOLOv3 was trained for 16 h, whereas YOLOv4 was trained for 6.5 h. Therefore, the slightly higher mAP value of YOLOv3 than that of YOLOv4 may be due to the longer training duration. However, in a scenario where YOLOv3 is trained for 10 h with 6000 batches and YOLOv4 is trained for 6.5 h, the results show that YOLOv4 performs significantly better, as indicated in Table 5.

4. Conclusions

In this study, we employed image recognition technology, utilizing widely adopted AI image classification methods to assess the phenomenon of chickens opening their mouths or not. We utilized the YOLO algorithm, known for its above-average training speed and accuracy, along with OpenCV's image analysis techniques, to evaluate the behavior of chickens. Traditional methods were time-consuming and require substantial manpower. By leveraging this identification technology, not only can labor be saved, but poultry farmers can also remotely monitor the physiological conditions of their chickens without the need to personally visit the poultry farm. The findings revealed the significant improvements when using YOLOv4 for system training compared with YOLOv3, regardless of whether the IOU was set at 0.5 or 0.75, and when using 6000 batches. Recall, precision, IOU, and mAP values were all substantially higher with YOLOv4. However, when YOLOv3 was trained with 10000 batches for system training, the results were comparable to those of YOLOv4 trained with 6000 batches, regardless of the IOU threshold. The results obtained in this study suggest the efficacy of YOLOv4 in accurately identifying chicken behaviors, providing insights that could revolutionize poultry farming practices.

Acknowledgments

This work is in part supported by the National Natural Science Foundation of China General Program (No. 61977055), Fujian Provincial Department of Education Project (No. JAS22390), Longyan University's Qi Mai Science and Technology Innovation Fund Project of Shanghang County (No. 2017SHQM07), and Fujian Provincial Science and Technology Special Commissioner Project (No. 202135060173).

References

1 M. El-Tholoth, H. Bai, M. G. Mauk, E. Anis, and H. H. Bau: Animals **11** (2021) 3203.
2 A. Mao, C. S. E. Giraudet, K. Liu, I. D. Almeida Nolasco, Z. Xie, Z. Xie, Y. Gao, J. Theobald, D. Bhatta, R. Stewart, and A. G. McElligott: J. R. Soc. Inter. **19** (2022) 20210921.
3 OpenCV: https://sourceforge.net/projects/opencvlibrary/ (accessed January 2024),
4 GeeksforGeeks: https://www.geeksforgeeks.org/opencv-overview/ (accessed January 2024),
5 J. Redmon, S. Divvala, R. Girshick, and A. Farhadi: 2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR) Las Vegas, USA (2016) 779–788.
6 Yi. Q. Huang, J. C. Zheng, S. D. Sun, C. F. Yang, and J. Liu: Appl. Sci. **10** (2020) 3079.
7 GitHub: https://github.com/HumanSignal/labelImg/ (accessed February 2024).
8 viso.ai: https://viso.ai/computer-vision/labelimg-for-image-annotation/ (accessed February 2024).
9 J. Redmon and A. Farhadi: YOLOv3: An Incremental Improvement (2018). https://doi.org/10.48550/arXiv.1804.02767
10 A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao: YOLOv4: Optimal Speed and Accuracy of Object Detection (2020). https://doi.org/10.48550/arXiv.2004.10934