Developing Object Detection Models for Camera Applications in Smart Poultry Farms

Stevan Čakić
DigitalSmart
Podgorica, Montenegro
stevan.cakic@digitalsmart.me

Tomo Popović
DigitalSmart
Podgorica, Montenegro
tomo.popovic@digitalsmart.me

Srdjan Krčo
DunavNET
Novi Sad, Serbia
srdjan.krco@dunavnet.eu

Daliborka Nedić
DunavNET
Novi Sad, Serbia
daliborka.nedic@dunavnet.eu

Dejan Babić
University of Donja Gorica
Podgorica, Montenegro
dejan.babic@udg.edu.me

Abstract—**This paper proposes the use of high-performance computing and deep learning to create prediction models that can be deployed as a part of smart agriculture solutions in the poultry sector. The idea is to create object detection models that can be ported onto edge devices equipped with camera sensors for the use in Internet of Things systems for poultry farms. The object detection prediction models could be used to create smart camera sensors that could evolve into sensors for counting chickens or detecting dead ones. Such camera sensor kits could become a part of digital poultry farm management systems in shortly. The paper discusses the approach to the development and selection of machine learning and computational tools needed for this process. Initial results, based on the use of Faster R-CNN network and high-performance computing are presented together with the metrics used in the evaluation process. The achieved accuracy is satisfactory and allows for easy counting of chickens. More experimentation is needed with network model selection and training configurations to increase the accuracy and make the prediction useful for developing a dead chicken detector.**

Keywords—convolutional neural networks, deep learning, digital farm management, high-performance computing, object detection, poultry farms

I. Introduction

We are witnessing dramatic growth in food demand around the globe, which is reflected in the increase in demand for animal protein. This contributes to the growth in the poultry feed market resulting in the increased global annual production that even surpassed pork production in 2020 (Foreign Agriculture Service/USDA, Livestock and Poultry: World Markets and Trade report) [1]. With these large and increasing numbers in the poultry sector, there is a requirement to continuously optimize and streamline the production process, while simultaneously limiting effects on the environment and improving the well-being of the animals during their short lifespan. Poultry farms, similarly to other livestock farms, are facing consistent and unavoidable challenges such as disease outbreaks, disposal of deceased animals, controlling all the basics of life (food, water, light, air), sanitation, cleaning, etc. We are witnessing the introduction of various digital or smart solutions to address these challenges. For example, using Internet of Things (IoT) sensors and data collection to measure air temperature, air humidity, CO_2, and ammonia levels are required as a basis for a successful production. With the latest developments in IoT and artificial intelligence (AI), it is possible to create even more advanced sensors that could help to reduce the losses, cut down on manual labor, and many other benefits. For example, using AI based prediction models can be used to count animals, detect dead animals, assess growth and homogeneity of animals during the growing cycle. Timely detection of dead animals and/or uneven growth could help early detection of disease and prevent outbreaks and losses.

This paper describes a research study in which high-performance computing (HPC) is used to experiment with deep neural networks (DNN) to create prediction models for object/chicken detection in digital images collected using camera IoT nodes installed in poultry farms. Fig. 1 illustrates the conceptual approach of this study. A dataset containing images extracted from video materials recorded in poultry farms was used. These images were annotated for object detection.

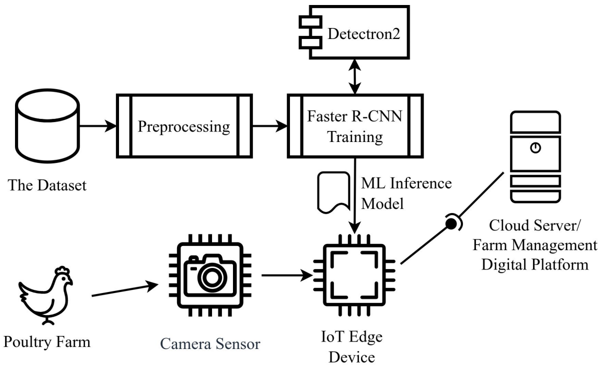


Fig. 1. The context: developing object detection models for chicken farms

The dataset is further enhanced using the pre-processing and augmentation and then utilized to run deep learning algorithms, in this case the Faster R-CNN network from Detectron2 library, to create inference models for detecting chickens in images. There are various ways to utilize these trained prediction models. We envisioned the use of the prediction model in an edge AI setting, where the model is ported onto an inexpensive IoT edge device equipped with the camera sensor. Every time a digital image is taken on the poultry farm, the pre-processing and prediction model is run to detect all the chickens in the image. The resulting number can then be utilized to implement functions such as counting the chickens, detecting dead chickens, etc. The use of inexpensive edge AI devices allows a dramatic reduction of

This research was funded by the FF4EuroHPC project under the Horizon 2020 Framework of the European Union. FF4EuroHPC project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951745.

data transfer, i.e. we are sending only a number dead chickens instead of sending a whole digital image, turning the IoT camera into a sensor for counting chickens, detecting dead ones, etc. Such a solution could be scaled by using several camera nodes running these models and providing the extracted knowledge, the number of dead chickens, to a digital farming platform in the cloud. An ongoing research in this domain as discussed in [2]. An example of the use of camera in poultry farm is given in [3]. In this study, we focus on development of prediction models that can be deployed in edge AI device equipped with an IoT camera sensor.

The rest of the paper is organized as follows. Section II discusses details of the data set collection, annotation, image pre-processing and augmentation, software and hardware tools selection, and setup. Section III provides results and discussion related to the performance metrics, the training process considerations, and an evaluation of the obtained prediction models. Section IV summarizes the findings of this study and discusses the next steps.

II. Materials and Methods

A. *Dataset Collection and Annotation*

The significance that represents data in machine learning is very high. Data collection, labeling, and processing are some of the key operations performed in the preparation phase for processing by DNN algorithms. For this process, the images are used as data. All images are separated from recorded videos collected from the chicken farms. The main functionality of the system that is described in this paper is object detection, which is in this case recognition of chickens on the obtained images. This is a task that initially first refers to computer vision with automatic independent recognition, thus it is clear that some algorithm of AI could be used for this process. To achieve a good performance, that algorithm needs a substantial amount of data. For this purpose, we collected and annotated 4000 images with chickens. Computer Vision Annotation Tool (CVAT) was used for the annotation of the objects in these images [4]. The CVAT provides tools for management and manual image labeling. A team of four people participated in the process of image annotation and labeling. The annotation process assumed drawing rectangles around the chickens manually. With the CVAT, the images in the dataset were organized into tasks, each containing a set of 20 images that need to be annotated. After the annotation, each set of the annotated images was reviewed for quality of annotations. Fig. 2 illustrates an example of an image during the annotation process in CVAT. The quality of the model depends on the quality of the dataset, and the annotation process was a very important part of this effort.

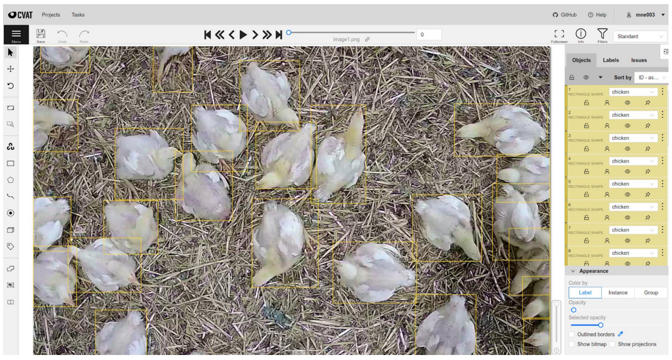
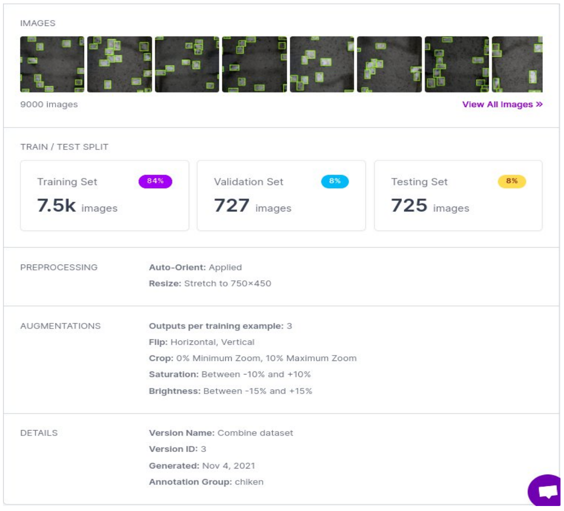


Fig. 2. Using the CVAT tool for annotation

All the annotations that were performed needed to be saved in an appropriate file format for later visualization and the training process to create the prediction models. The CVAT platform supports multiple formats for storing annotations. In this case, we opted for the Common Objects in Context (COCO) format [5]. This format is suitable for the selected software tools used in the development of models and experimentation with DNN. In the COCO format, for each set of annotated images, there is a corresponding JSON file that has a structure for storing the object category and position of each annotation from the image. The COCO standard also provides the key metrics needed for model precision assessment as discussed later in the results section.

B. *Dataset Preprocessing and Augmentation*

In this section, the main focus is on the steps needed to be implemented to ensure that the data is adequate for model training. Pre-processing, organization, and augmentation of the data were achieved using the RoboFlow tool [6]. Fig. 3 shows a screenshot from RoboFlow with a summary of the dataset and preprocessing that took place in this study.



IMAGES

9000 images   View All Images »

TRAIN / TEST SPLIT

Training Set 84%	Validation Set 8%	Testing Set 8%
7.5k images	727 images	725 images

PREPROCESSING
Auto-Orient: Applied
Resize: Stretch to 750×450

AUGMENTATIONS
Outputs per training example: 3
Flip: Horizontal, Vertical
Crop: 0% Minimum Zoom, 10% Maximum Zoom
Saturation: Between -10% and +10%
Brightness: Between -15% and +15%

DETAILS
Version Name: Combine dataset
Version ID: 3
Generated: Nov 4, 2021
Annotation Group: chiken

Fig. 3. The dataset preprocessing and augmentation with RoboFlow

The RoboFlow platform provides a large number of data operations: pre-processing, augmentation, annotation, organization, model training, and model deploying. In this study, the only transformation of the original dataset images for the pre-processing step was fixing the dimensions of the source images to the same size. As for the augmentation process, we used image rotation, lightness change from -15% to +15%, saturation from -10% to 10%, and crop maximum of 10%. After the augmentation process, we ended up with the data set that contains 9000 images, created from the initial 4000. Please note that the size of the dataset after augmentation is limited in the free version of RoboFlow, but 9000 images ended up being a decent number of images in the data set. It is worth mentioning here, that RoboFlow is suitable for the organization of annotated images and it has good interoperability with the CVAT tool that we used for annotation. RoboFlow was used to manage images from the CVAT platform where they were organized into smaller groups and create a data set containing a single large group of annotated and augmented images that were easy to manage. RoboFlow was also used to split the dataset into training, validation, and test data sets as required for the machine learning process. Fig. 3 shows a screenshot from the RoboFlow with a summary of the dataset and pre-processing that took place in this study. Once collecting, annotating, pre-processing, and augmentation are completed, the data set is ready to be used for machine learning experiments and the creation of prediction models for object detection.

C. Selection of Machine Learning Tools

This section provides considerations related to the tools used to create adequate prediction models. The Python programming language was used. Python is recognized as one of the most popular programming languages for machine learning. The key reasons for this are Python simplicity, availability of a large number of packages for effective work with different problems, and a large community of developers [7]. Choosing Python for development opens access to a wide range of packages that can be used for deep neural network development. In this study, the main focus is on the object detection problem, namely the detection of chickens. There are a few good Python libraries that could be used for this purpose. We selected Detectron2 which mostly relies on the PyTorch library. The PyTorch library is used for a wider range of problems for machine learning and it is very popular in the research community [8]. Some of its advantages are simplicity and flexibility. PyTorch is often compared to other packages that have a similar role, mostly with TensorFlow. There is no way to determine which of the two is better, hence both are widely used for implementing AI solutions [9]. The Detectron2 library is composed of state-of-the-art algorithms for object detection and segmentation [10]. Depending on the environment, the installation and configuration of the Detectron2 package are not straightforward. In our experience, the Google Colab environment is a good choice for initial experiments and research [11]. Google Colab provides means to accelerate and facilitate the package installation and configuration processes. There are free and paid/pro versions of Google Colab and a large number of researchers in AI are using it. This platform is based on allowing the users to use virtual machines for a certain time. In the free version, this time is varying and developers are not informed in advance about the duration that their virtual machine will remain active before disconnection. In the pro version, this time is extended. It is possible that the session gets terminated during model training and that the results are not preserved. Using some additional code, it is possible to solve this, by storing model checkpoints during different training phases. Another disadvantage is that the data gets deleted when a developer is disconnected from Google Colab virtual machine. One approach to solve this issue is to store data on Google Drive. The drawback of this approach is that whenever you connect with the virtual machine, you need to restore the data from your Google Drive, which can be extremely time-consuming. To conclude, the free version of Google Colab is a very useful platform as long as we are aware of potential disadvantages.

D. High-Performance Computing Support

HPC systems are usually defined as a cluster of computers with higher performances than typical personal computers. Such computers come with much more processing power, which is defined by the number of installed processing units (CPU cores), optionally equipped with graphical processing units (GPUs). HPC systems often come with plenty of operating (RAM) memory and more storage capacity when compared to an ordinary PC.

In this study, HPC system resources were managed using a Linux utility for resource management (SLURM) architecture. SLURM is an open-source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters [12]. The system used for experiments contained computing nodes organized into two groups, also called partitions: a) *compute* nodes, and b) *gpu* nodes. The *compute* partition is made up of 14 nodes, with the names node01 to node14. This is the default partition and the computing jobs will run on this partition unless specified otherwise. Each of these nodes is equipped with 2 Intel Xeon E5-2690v4 processors having 28 cores. Every node also has 512 gigabytes (GB) of fast DDR4 RAM. The *gpu* partition consists of 8 nodes (gpu01 to gpu08). The only difference with compute nodes is that they each have 4 NVIDIA Tesla M60 GPUs. A Tesla M60 card is made out of two physical NVIDIA Maxwell GPUs with a combined 16 GB of memory. Many applications perceive the card as two separate GPUs, appearing as 8 GPUs per node [13].

III. Results and Discussion

The experimentation in this study was focused on object recognition using DNN to solve computer vision problems [14]. Initial experiments were performed on the Google Colab platform, but the final experimentation was done using a HPC system. The training and evaluation of all cases were done using the GPU nodes.

A. Network Selection and Training Parameters

The considerations provided here are relevant only for experiments conducted on HPC systems. Detectron2 supports different types of deep neural networks for object detection, object segmentation, detecting key points of an object, etc. [15]. The main type of neural network used by Detectron2 to solve object detection problems is Faster R-CNN [16]. For all deep neural networks, it is known how much the choice of specific parameters affects the accuracy of the model. The parameters that are tested with these experiments are *gamma* parameter, *num_iters* parameter, parameter *steps*, and *ims_per_batch*. In addition to these parameters, each of the experiments can be executed with a different number of GPUs: one, two, four, or eight. All experiments are tested on several types of neural networks supported by the Detectron2 package [15]. The idea of these experiments is to detect the appropriate neural network with the appropriate parameters. In addition, by testing model training on different numbers of GPUs, we tried to get some sense of how these changes affect the model accuracy, training time, and parameters selection. As an important parameter for experiments, model prediction time is analyzed with caution. The prediction speed is very important because the final models need to be integrated with IoT devices whose performance is quite limited [17].

Before discussing the results, the meaning of individual parameters needs to be explained. Parameters *gamma*, *num_iters,* and *steps* can be considered as a group of parameters strongly connected with parameters *base_lr*, *lr_policy*, *max_iter*, and *warmup_iters*. For the experiments in this study, the parameters *base_lr*, *lr_policy,* and *warmup_iters* have fixed values:

- *base_lr* = 0.001,
- *lr_policy* = 'steps_with_decay',
- *warmup_iters* = 20.

The *base_lr* parameter indicates the initial value for the learning rate, *warmup_iters* is the number of iterations for updating the learning rate from zero up to the value of *base_lr*, while *lr_policy* represents the update strategy for the learning rate [17]. Parameter *max_iter* represents the maximum number of epochs for model training. For the selected strategy, *lr_policy* = 'steps_with_decay', the learning rate is updated according to (1):

$$learning\_rate = base\_lr * gamma^{step\_index} \quad (1)$$

For example, if parameters values are *base_lr*=0.001, *warmup_iters*=20, *max_iter*=500, *gamma*=0.1, *num_iters*=500 and *steps*=(20, 350, 450), the learning rate change values as follows:

- from 0 to 20th iteration learning rate is changed from 0 to 0.001 (warming up);
- from the 20th to 350th iteration the new value for the learning rate is learning rate = 0.001 * 0.1^0 = 0.001;
- from the 350th to 450th iteration the new value for the learning rate is learning rate = 0.001 * 0.1^1 = 0.0001;
- from 450th until the end (*max_iter* = 500) value for learning rate = 0.001 * 0.2^2 = 0.00001.

This update strategy allows us to reduce the learning rate value during the training process. The learning rate parameter in neural networks is very important when it comes to training efficiency. If the learning rate is too high, the function minimum (when the loss is minimal) can be missed. If the learning rate is too low, it can take a long time to reach the function minimum, thus, the trade-off is important. More information regarding the learning rate and strategies for updating is provided in [18].

Another very important parameter is *ims_per_batch*. This parameter defines the batch size. In this case, it is the number of images per batch for the DNN that will be loaded into the GPU on the HPC system. Depending on this parameter, the number of graphic cards used for model training was updated. Table 1. shows the analysis of the training duration of the mentioned neural network according to the defined parameters. The table shows the achieved accuracy of the model, as well as the acceleration of the training when a different number of GPUs was selected. To understand the quality of the neural network models, the accuracy parameters need to be explained. These parameters are used for assessing the quality of the accuracy of the neural network. When focusing on object detection, specifically when machine learning is based on the COCO format, the main parameters provided by Detectron2 during the evaluation of the model are *BBox_ap*, *BBox_ap50*, *BBox_ap75*, also known as AP, AP50, and AP75, respectively [19]. The AP metric represents an average precision for intersection over union (IoU) between objects from value 0.5 to 0.95 with a 0.05 step. The IoU itself is calculated in (2) [20]:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \quad (2)$$

This metric is best explained with an example. If DNN uses 100 annotated images so that the IoU for 98 of them is at least 0.5 that means that AP50 is 0.98. For the area of overlapping, when a rectangle is used for annotation, the IoU value represents the intersection of the rectangle around the manually annotated object and the rectangle that is generated by the prediction model. With every increase in IoU value, there is a decrease in accuracy. For example, AP75≤AP50 because AP75 IoU must be at least 0.75. *BBox_ap* or short AP represents the average precision or the sum of all values for APs from AP50 to AP95 with step 5 (3):

$$BBox\_ap = \frac{\sum_{i=10}^{19} BBox\_ap(5*i)}{10} \quad (3)$$

To obtain optimal values of output metrics for the selected input parameters, dozens of experiments with different combinations of parameters were executed. In addition, we observed how model training on different numbers of GPUs affected the training acceleration. One such combination of parameters is given as an example:

- *gamma* = 0.5;
- *steps* = 20, 450, 850;
- *max_iters* = 1000;
- *warmup_iters* = 20;
- *ims_per_batch* = 256.

The neural network architecture used for this specific experiment is *faster_rcnn_R_101_C4_3x* [15].

TABLE I. ACCURACY AND TRAINING TIMES

Number of GPUs	*AP*	*AP50*	*AP75*	*Training time [min]*
1	~0.83	~0.97	~0.94	50:55
2				32:32
4				30:00
8				28.53

One of the conclusions that can be singled out is that every increase in the number of GPUs used for training accelerates the training time of the model. This is also the key advantage of increasing the batch size parameter. For a larger batch size, the advantage of parallelization is more obvious. However, for larger batch sizes, the training takes longer and the accuracy is lower. This phenomenon is also noted and described in [21]. During experimentation with model training, another problem can occur: by increasing the value of batch size, the experiment takes more memory, and the training process may be interrupted due to the load on the GPU memory. This happens when there is not enough memory to load the data and send it for processing. Therefore, increasing the batch size needs to be done with caution. On the other hand, by increasing the batch size, it may not be necessary to update the learning rate during the training as discussed in [22]. Our initial experiments without updating the learning rate showed a similar trend with the batch size vs. acceleration, but more testing with the batch sizes will need to be done soon.

B. Prediction Model Evaluation

As mentioned before, the functional requirements for the camera sensor equipped with an object detection prediction model include counting the chickens captured in the image and possibly detecting dead ones. It is necessary to count how many annotation rectangles are drawn for a given image, while the actual location of the rectangle is not as critical. In other words, achieving high numbers for AP75 or AP50 may be sufficient. The challenge here is that sometimes the chickens are very close to one another, which is why AP75 and the overall AP should be as high as possible. The same goes for detection of the dead animals, where the detection depends on running the prediction model on several successive images over time to identify the rectangles that did not move for a longer period. If we have an inaccurate detection with a low AP and AP75, it may appear that the animal is moving while it should have been detected in the same position every time. To obtain better accuracy, additional experiments with a variety of network model selection and configuration parameters will be done shortly. Please note that we discuss only counting the number of chickens or detecting the dead ones as seen in the image.

An illustration of the utilization of the prediction model in real life is illustrated in Fig. 4. This image shows chickens marked by the prediction model. This functionality was successfully integrated into the digital farming platform already collecting the environmental measurements from IoT sensors. The prediction model was ported from the HPC system onto the edge IoT/AI device called NVIDIA Jetson, with Linux, Python and PyTorch/Detectron2 installed. The execution time of the prediction model was around 10 seconds, which makes the whole edge setup acceptable as a camera-based IoT sensor node for counting chickens. It is important to note the prices of camera sensors and edge AI devices are becoming lower and more affordable.

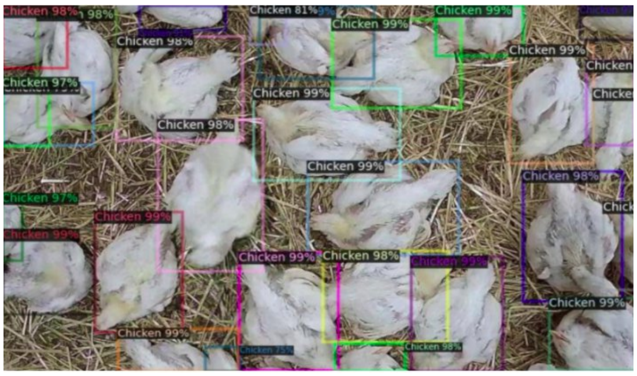


Fig. 4. Evaluating the prediction model with the test dataset

IV. Conclusion

This paper discusses the development of object detection prediction models for use in smart solutions for poultry farms. The inference model of interest is to be used for the detection of chickens in digital images obtained from the camera IoT devices deployed on the farm. This model can be utilized to implement camera sensors for counting chickens and/or detecting dead ones. The developed model can be installed in an edge AI device to reduce the need for uploading large image files, but it could also be integrated into the digital farming platform as a cloud service.

The paper provides considerations for the development approach, data set preparation, software, and hardware tools selection. The results section provides details on the metrics used for evaluation of the training process and selection of configuration parameters, such as the batch size and number of GPUs used. It is shown that when using a Faster R-CNN network, a fairly accurate prediction model for chicken detection can be developed, AP=0.83, AP75=0.94, and AP50=0.97, which would allow for practical use of such a model for counting chickens using camera sensors in an actual poultry farm setting. Further research steps include additional experimentation with different network models and parameter settings, but also additional software development aimed at the integration of developed prediction models into the edge IoT camera devices and/or digital farming solutions for the poultry sector. Further research could include the validation of the solution in actual farms in order to collect metrics needed to estimate benefits of such technology.

References

[1] USDA, “Livestock and Poultry: World Markets and Trade”, United States Department of Agriculture, Foreign Agriculture Service, April 8, 2022, Available online: https://www.fas.usda.gov/data/livestock-and-poultry-world-markets-and-trade, last accessed: 7 June 2022.

[2] S. Neethirajan, “Automated Tracking Systems for the Assessment of Farmed Poultry”, Animals, January 2022.

[3] L. Cao, Z. Xiao, X. Liao, Y. Yao, K. Wu, J. Mu, J. Li, H. Pu, “Automated Chicken Counting in Surveillance Camera Environments Based on the Point Supervision Algorithm: LC-DenseFCN”, Agriculture, 2021

[4] C. Sager, C. Janiesch, P. Zschech, “A survey of image labelling for computer vision applications”, Journal of Business Analytics, 4:2, 91-110, 2021

[5] T. Y. Li, et al, “Microsoft COCO: Common Objects in Context”, arXiv, 2014

[6] Roboflow platform, Available online: https://universe.roboflow.com/, last accessed: 19 April 2022.

[7] S. Raschka, J. Patterson, C. Nolet, “Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence”, Information, vol. 11, pp. 193, 2020

[8] A. Paszke, et al, “PyTorch: an imperative style, high-performance deep learning library”, 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, A721, 8026–8037, 2019

[9] F. Florencio, T. Silva, E. Ordonez, M. Júnior, “Performance Analysis of Deep Learning Libraries: TensorFlow and PyTorch”, Journal of Computer Science, vol. 15, 2019

[10] H. Wen, C. Huang, S. Guo, “The Application of Convolutional Neural Networks (CNNs) to Recognize Defects in 3D-Printed Parts”, Materials, vol. 14, pp. 2575, 2021

[11] T. Pessoa, et al, “Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications”, IEEE Access, 2018

[12] A. B. Yoo, M.A Jette, M. Grondona, “SLURM: Simple Linux Utility for Resource Management”,Job Scheduling Strategies for Parallel Processing. JSSPP, vol 2862. Springer, Berlin, Heidelberg, 2003

[13] Yotta advanced computing, Available online: https://www.yac.hr/, last accessed: 19. April 2022.

[14] Z, Zhao, P. Zhengg, S. Xu, X. Wu, “Object Detection with Deep Learning: A Review”, IEEE Transactions on Neural Networks and Learning Systems, 2019

[15] Detectron2 Facebook AI library for object detection and segmentation, Available online: https://github.com/facebookresearch/detectron2, last access: 19. April 2022.

[16] S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015

[17] I. Jovovic, et al, “Face Mask Detection Based on Machine Learning and Edge Computing”, 21th International Symposium INFOTEH-JAHORINA, 2022

[18] Y. Wu, et al, “Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks”, IEEE International Conference on Big Data, 2019

[19] Y. He, C. Zhu, J. Wang, M. Savvides, X. Zhang, "Bounding Box Regression with Uncertainty for Accurate Object Detection," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 2883-2892, 2019

[20] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, S. Savarese, “Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression”, arXiv, 2019

[21] N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, M. Smelyanskiy, “On large-batch training for deep learning: Generalization gap and sharp minima”, 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 2017

[22] S. Smith, P. Kindermans, C. Ying, Q. Le, “Don't Decay the Learning Rate, Increase the Batch Size”, arXiv, 2017