Comparison Platform Design for Neural Network Models Evaluation in Driver Monitoring Systems

Alexey Kashevnik[1,2], Ammar Ali[2]
[1]SPC RAS, St. Petersburg, Russia
[2]ITMO University, St. Petersburg, Russia

Abstract—**Neural networks have become more and more popular in the last years. They are used for different classification tasks. There are a lot of different models that can be generated which will have similar functionality but different accuracy and execution time. Herewith model evaluation is one of the main parts of the model development process to find the best model that meets the requirements for a particular project or task. Neural network evaluation main methods represented by the hold-out approach that is aimed at dividing the data-set to training, validation, and testing as well as cross-validation. More further, special platforms that are provided by different companies (like Google, Microsoft, Neptune, etc.) aimed to facilitate the model evaluation for inferencing in different environments. In the paper, we proposed a new platform designed to evaluate the neural network models developed for object detection and human behavior monitoring. We evaluated the platform for the task of driver monitoring in the vehicle cabin. The proposed platform allows to identify several cases and show the accuracy for each of the cases in the considered area. We propose the classification of such cases that allows us to compare the different models accurately.**

I. Introduction

Unit testing is an essential part of developing software applications. Also called component testing, it is all about isolating one unit of code to verify that it's working as it should be. Neural Networks are designed to solve classification tasks in software applications. which make the unit testing is really important to be applied over it especially because models are represented as a black box on the system and couldn't be changed from the inside unless we changed the data, architecture, or loss functions. One important thing to be considered that the machine learning model is usually used on the backbone of the application or project. It means that skipping the testing phase or not considering it as the main factor could cause enormous time consumption in the development process. Currently, existing platforms will not support such functionality directly due to the nonexistence of public data-sets that cover this field of study and the difference between metrics for object detection, classification, and event detection which are included in one model in our case. As well as the restrictions related to the devices used in the development process. We introduce a new platform designed using python (CherryPy), CSS, HTML, and JavaScript to evaluate the models which are built to analyze the driver behavior inside the car cabin (detect such classes). As well as the ability to convert the models to support the devices we use. And to generate reports for comparing different models according to the points of interest for the development. The platform will provide a new service to developers not just to test object detection and classification algorithms over a testing dataset but also a full evaluation process taking into account the architecture of the model and how it will perform in a given hardware device. The number of Flobs and parameters memory, CPU and GPU Usage, and running time. Conversion service for AI models to multiple inference environments and track how that will affect the accuracy. The platform is developed especially to evaluate algorithms for driver behavior analysis so it will support Face Detection, Head pose estimation, Face recognition, and other activities for the driver inside the car cabin. Driver behavior analysis system is proposed in details in [1], [2], [3] is aimed to detect dangerous situations in the vehicle cabin. We use the YoloV3 neural network to detect belt unfastened, mobile phone usage, eating/drinking, and smoking. Relearning the neural network model for more accurate phone detection can cause accuracy to decrease for other states. In this case, after every relearning process, we have to compare the new model with the previous one.

II. Related Work

The Authors of the paper [4] developed a web platform to help researchers to prepare the dataset training for the TensorFlow library. They proposed Using WebQual V4.0 [5] to evaluate integrated deep learning platform. They focused on the measurement of the platform evaluation according to the WebQual metrics for three main categories (usability, information, and service interaction). However, the functionality of the platform is tied to data preparation for TensorFlow while we generate data that will be usable by all known libraries (Keras, Tensorflow, Darknet, Pytorch). Authors of the paper [6] proposed unification of different computation as a part of a single batch or stream. Neptune is now a famous library and platform multiple developers are using it not just to evaluate the machine learning models but also for data analytic applications the problem was the general indicators and metrics used and provided by Neptune may not fit all requirements for the evaluation of a particular model. Our platform is connected to Neptune making our platform support all their functionalities. A popular library for machine learning [7] introduced a new service called TensorBoard [8] that is a tool for providing the measurements and visualizations needed during the machine learning workflow. It

TABLE I. MSE, AND M-ESTIMATORS RM COMPARISON [12]

performance function	RM
MSE	0.0104
LMLS	0.0106
L1	0.0156
FAIR	0.0132
CAUCHY	0.0107
GM	0.0117
HUBER	0.6121

TABLE II. COMPARISON OF THE DIFFERENT METRICS BY DATA MANIPULATION. [15]

	Baseline	Low_conf	All	Tail Boost
	0	3076	3616	71781
mAP	0.64	0.63	0.63	0.70
NAB	0.29	0.29	0.17	0.31
CaTDet	13.6	13.6	15.1	12.5
AD	9.0	11.5	13.8	8.9

enables tracking experiment metrics like loss and accuracy, visualizing the model graph, projecting embedding to a lower-dimensional space, etc. But it is tied to TensorFlow models more valuable to track through the training process and not fit the requirements for a high level of unit testing. TensorBoard gives an overview of the model architecture and how it will perform in different environments and it supports its models so testing other model needs conversion for the model and logs to their environment and the evaluation is tied to the way the model behave not including special metrics for our particular task. Authors of the paper [9] mentioned the importance of machine learning growth and the necessity of the evaluation for these models and developed a platform to automate tasks like ”data pre-processing, feature engineering, model selection, hyper-parameter optimization, and prediction result analysis”. However, the developed platform is not open-source for developers. More further, it is necessary to mention the metrics that should be taken into account when we are considering object detection and classification tasks. Authors of the paper [11] explained the intersection over union (IoU) loss for 2D and 3D object detection. IoU is a well-known metric used widely as an evaluation metric to evaluate the performance of different detectors in the testing stage. We used it for evaluation of our model in the loss function to minimize the discrepancy between the predicted and ground truth Bounding Box [3]. Mean Square error expansion (MSE) is derived for Euler–Maruyama numerical solutions of stochastic differential equations (SDE). It has the main role in minimizing loss in machine learning and optimization techniques [12]. The authors of this paper show the differences between multiple M estimators that are used in the field of machine learning in training or testing according to the needs and the strong relationship between these metrics and the Bayesian setting with quadratic cost function should be mentioned when we are talking about machine learning. They also showed a useful comparison between MSE and M estimators with the RM see Table I. The mean root was taken for each performance function and it was shown that MSE has the minimum Error root as a response for that particular training.

This could be taken into account since these numbers are obtained for training a classification model. Authors of the paper [13] show the importance of AUC - ROC curve for classification problem. It is a performance measurement for classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. So, the higher the AUC better the model at predicting Class A as Class A, not B. mAP metric is a well-known metric used to measure the accuracy of object detection models [14]. Voc Pascal is the one who defined a map metric as a metric for object detection. Lately, authors of the paper [15] defined a new metric called Average Delay Metric (AD) for object detection to measure and compare detection delay. They showed that the mAP is not sensitive enough to reflect the temporal characteristics of a video object detector see Table II. The baseline is an r-fcn detector with ResNet-101. Retardation makes detection slower by suppressing the first 5 detections of a ground truth instance. In the case of low-conf, we only suppress the detections with low confidence, while in the case all, all detections are suppressed regardless of their confidence scores. Tail boost improves the detections that are 20 frames later than the first occurrence of ground truth. Note that for CaTDet and AD, lower numbers indicate better results.

The proposed platform support all mentioned metrics (recall, precision, mAP, AD, and M-estimators. it also supports finding the best NMS and confidence thresholds for object detection to get the best performance for a particular model.

III. PROPOSED PLATFORM

The Implemented platform has been built using Python, HTML, CSS, and JavaScript. CherryPy library has been used to connect the python scripts to the User interface. All were containerized using docker and build on the server see Fig(1).

A. *Evaluation Functionalities*

The platform has been designed to test models that can detect different driver in-cabin events like belt fastness, phone using, eating/drinking, and smoking. On the other hand, it could evaluate models that detect sub-classes from the main five. For example, we can evaluate the model just for belt detection. It provides different evaluation methods. Firstly, the user can test the models on predefined testing data, collected by the driver monitoring system. More than 1500 images in different conditions are provided for testing and still increasing data simultaneously with the development process. Over this data, the user can evaluate the models using the general testing options. Secondly, an option for testing over a particular image uploaded by the user as a unit testing is also provided. If the user is an administrator then he/she has the permission to add this image to our testing data to enhance after choosing to which class and conditions it should be added. Lastly, the ability to test the models over a new data-set uploaded by the

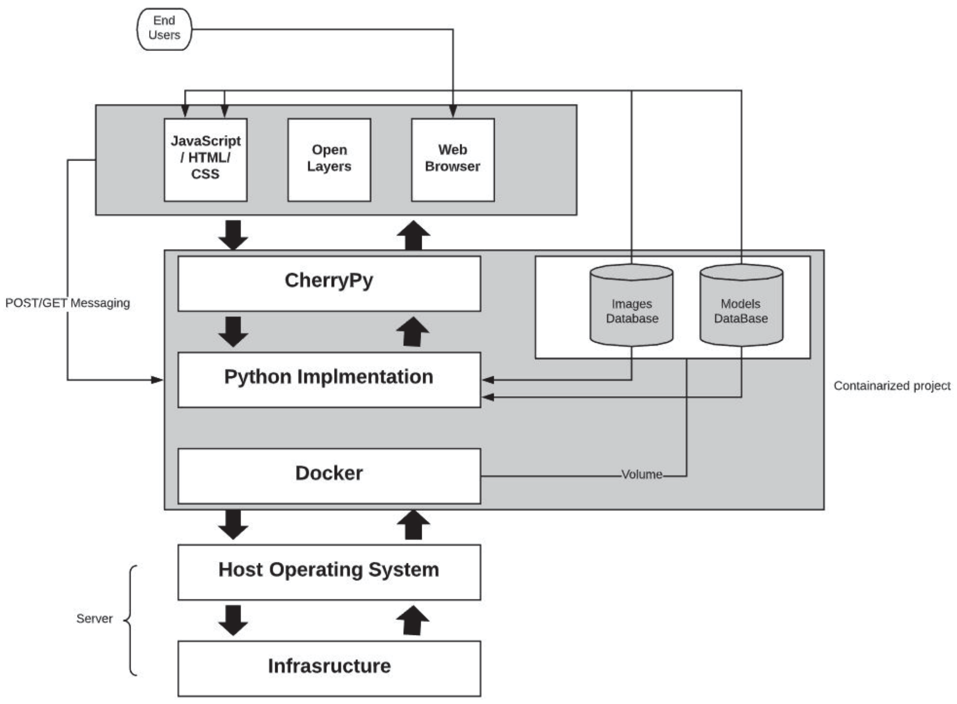


Fig. 1. Platform architecture

user taking into account that the images should be added in a fixed template folder structure provided as a template from the platform.

B. Evaluation Metrics

Before diving into the evaluation metrics for object detection and classification let us review the main concepts. **Confidence Score**: is the probability that an anchor box contains an object. **Intersection over Union**: is defined as the area of the intersection of a predicted Bounding box (B_p) and a ground-truth box (B_r) over the area of the union of these boxes.

$$IoU = \frac{area(B_p \cap B_r)}{area(B_p \cup B_r)}.$$

Both Confidence score and IoU are used to determine whether detection is truly positive or false positive. So we need both of them when we want to calculate the precision and recall. A detection is considered a true positive only if it satisfied three conditions:

1) Confidence score bigger than a threshold.
2) The predicted class matches the real class (ground truth).
3) The predicted bounding box has an IoU greater than a threshold which is not important in our case because we are interested to detect the events even if the box didn't exactly bound the objects.

False positives on the other hand, occur when violating one of the latest two conditions in general but we can ignore the third one since it is not in the point of interest for our functionality. When the confidence score of detection is lower than the threshold then the detection counts as a false negative. But if the confidence score is not supposed to detect anything less than the threshold then it counts as a true negative but that usually not included in the object detection and not important in the case of study. **Precision** is defined as the number of true positives divided by the summation of true positives and false positives. This means how many times the model said that a predicted class doesn't match the real one.

$$precision = \frac{TP}{TP + FP}.$$

Recall defined as the number of true positives divided by the summation of true positives and false negatives. It is clear that the summation holds for the ground truth and it is the most valuable measurement for our models.

$$recall = \frac{TP}{TP + FN} = \frac{TP}{GT}.$$

Now if we want to discuss the dangerous states we are observing. Then we can classify them into two main classes:

1) Belt Detection: we consider this as a dangerous state if the belt is not detected on the frame. It means if the driver has not fastened the belt and the system said that it does. A few false positives will not be a problem because the absence of the detection will directly message a dangerous state. In this case, we can say that the recall is more important or valuable from precision here but of course, both should be considered.
2) Drinking, Smoking, and phone usage: we consider a dangerous state if any event is detected. The problem here that we cannot make a trade between precision

and recall because low precision means a lot of false dangerous states detection. On the other hand, low recall means a lot of dangerous states will be ignored.

One good indicator could give us a trading factor for the second case that we are working with videos. That means we may get a hundred frames to include the phone as an example and even with low recall, we will get the dangerous state message. But that is not applied for the belt because it does mean that we have a hundred frames for all the belt detector should work perfectly or we will get a false dangerous state message. What we can conduct from this, that we need high recall for the belt even with low precision. And high recall but more important high precision for other classes.

AUC-ROC curve is a performance measurement for classification problem at various thresholds settings. Higher AUC means that the model is predicting belt as belt and phone as a phone, etc while the ROC curve is plotted with TPR against FPR. IT means that ROC is a probability curve.

TPR is the trues positive rate or sensitivity and it equals to recall.

FPR is the false positive rate.

$$FPR = 1 - Specificity,$$

where

$$specificity = \frac{TN}{TN + FP}.$$

Average Precision. The precision-recall curve can be used to evaluate the performance of a detector, it is not an easy comparison when we want to compare multiple detectors if the curves intersect with each other. Here a numerical metric could be used directly for comparison which is the average precision (AP). For multiple recall levels we interpolate the precision and the interpolated precision defined as follows:

$$precision_i(r) = maxp(r^{`}) for r^{`} \le r,$$

where r is a certain level for recall and $r^{`}$ is any recall level bigger than r. Recall levels could are 11 equally spaces for .0 to 1.0 for 0.1 steps. Then AP will be defined by the relation:

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) precision_i(r_{i+1}).$$

Some other methods to define the recall levels related to the IoU but since it does not affect the performance of our system we don't need to calculate it or AR (average recall).

Mean Average Precision mAP is defined as the mean of AP across all k classes and defined:

$$mAP = \frac{\sum_{i=1}^{k} AP_i}{k}.$$

Average Delay AD: incorporate fairness and comprehensiveness.

Fairness: AD considers the trade-off between false positives and false negatives to avoid the case of reducing delay by detecting many false positives.

Comprehensiveness. AD covers a wide range of operating conditions, analogous to AP [15].

$$AD = \frac{1}{\bar{p}} - 1 = \frac{1}{\frac{1}{R}\sum_r \frac{1}{D_r^* + 1}} - 1,$$

where R is the total number of frame and D_r^* defined by the relation:

$$\overline{D^*} = \frac{1}{N}\sum_{i=1}^{N} min(D_i, W),$$

where W is the detection window we are using in the video and D is the delay and it was shown in the paper that it follows the discrete exponential distribution.

$$D \approx exp(p).$$

C. Testing Report Generation

The platform shows the results for evaluation in the form of the auto-generated report. This report includes different information some could be manipulated by the user. Starting with visualization images for the curves related to the average accuracy, RoC curve then histograms show the number of classes in the data-set and for the predicted classes as well the running time over the data-set and CPU usage. Charts show the response of the system "recall" for different confidence thresholds. The numerical results will be shown in a table about the average running time, the type of model used, recall, and precision. It could be contained that results for more than one model if the report was generated to compare two models. Also, may it includes multiple tables for different conditions. This report is displayed by the Platform and the ability to download it as an excel project for extra experiments if needed.

D. Confidence Threshold Computing

As it has been mentioned earlier that the confidence threshold is responsible for the trading between recall and precision. And as we said that the recall is the most important factor for our considerations and precision comes in the second place. Then we need to find the maximum threshold that gives the best recall defining that in mathematical expressions will be: find CT_b where:

$$recall_{CT_b} = max(recall_{\forall CTi}).$$

Since the equation could have a set of solutions we want to find the one which maximizes the precision. Suppose that the answer to the previous relationship is a set of confidence values S then we choose CT_b by:

$$precision_{CT_b} = max(precision_{\forall CTi \in S}).$$

To solve the previous problem, two methods have been considered (linear solution and binary search algorithm).

1) Linear Solution: This algorithm is a basic solution. By iterating linearly for all possible values of confidences with a step equal to 0.01 and calculating the recall and precision than solving the previous equations will be done easily (see Fig. 2). This solution has a bad time complexity because for each step we are calculating the precision and recall for all testing data. And for each image in the data, we are passing it to a neural network. On the other hand, it will give a clear vision about the relationship between confidence and precision, recall and how it influences these values. that may help the analyzer "User" to find a better solution to enhance the models. But also, the step for confidence is 0.01 which means iteration over the testing data 80 times in the worst-case scenario, and if more accurate confidence needed more time complexity. Here come the second solution benefits.

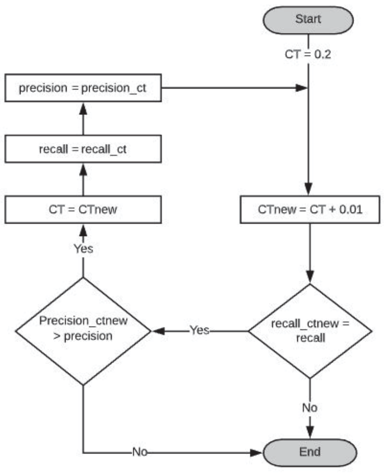


Fig. 2. Linear Solution

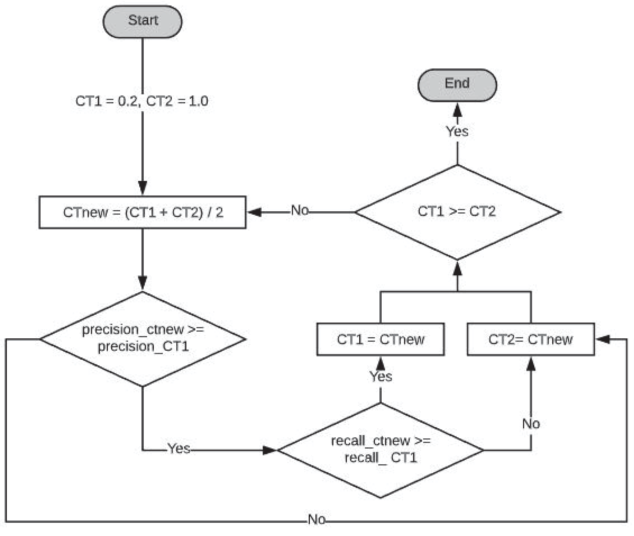


Fig. 3. Binary Search Solution

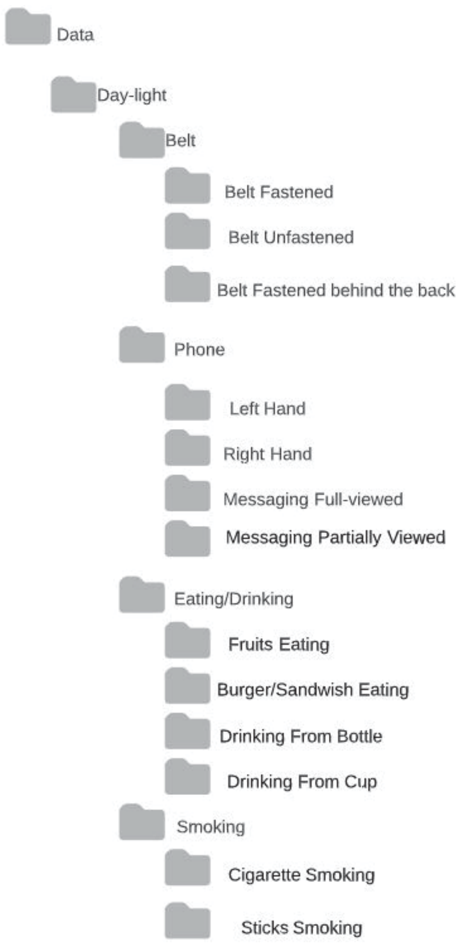


Fig. 4. Data structure of the testing dataset

2) Binary Search Solution: Binary search is a well known searching algorithm. In the definition of this algorithm that it has the best value that maximizes or minimizes a function with the constraints that the function is increasing or decreasing. Here is the idea that the precision function could be approximated to an increasing function according to the confidence value while recall is a decreasing one. the algorithm is shown in Fig. 3.

This algorithm has a better time complexity because it needs log_2 time to find the solution. Since we know that the total number of operations needed in the linear search equal to 80 then we can find that the total number of operations for the Binary search algorithm is equal to $log_2(80) \approx 7$. It means about 7 times instead of 80 in the binary search algorithm we need to find the optimal confidence (For each iteration we need to go through all the testing data to find the accuracy and precision for a particular confidence value). So, assuming that we have 1500 images for detecting the time needed to find the optimal confidence value in the linear algorithm will be $1500.80.T$ where T is the time needed to pass one image to the neural network. While using Binary search we need $7.1500.T$ which is 11 times less. Anyway, it will not provide the same detailed figure for the relationship between confidence, recall, and precision that is why both algorithms are included in the platform.

E. Supporting Neural Network Models

The models have been build using the Darknet. So the main supported models are those that were built by Darknet. especially Yolo V2/3/4. Other network architectures are supported as well but it may need to implement a script to handle the detections by the User. The inference was done using OpenCV 4.4. So any architecture or layers not supported by the DNN library will not behave fine. OpenCV was chosen because it supports Tensorflow frozen models, ONNX models, Caffe Models, Darknet models, and more.

F. Conversion Functionalities

The Platform also supports model conversion. As already mentioned that the base models were built using Darknet but the conversion service gives the ability to obtain Keras models and TFLite Models For Android to be optimized and have better performance on Android devices.

G. Testing Data Structuring

The testing dataset provided by the platform is gathered locally and covered wide different states for each class (see Fig. 4).

It includes images from different cars and men women drivers, with and without glasses as well as different clothing colors and types as well for the hairstyle. The images were taken also in different lighting conditions ”day time, night and dusk”. With the sun from different directions and in the rain as well. From the classes we can as well guarantee that there are images of belt fastened, unfastened, or fastened behind the back. For phone usage by the left/right hand or messaging even if the phone is partly viewed. For drinking from a cup or a bottle and different types of both. Eating burgers, sandwiches, and fruits. Smoking cigarettes sticks using pipe (IQOS and others). The video data has been marked by the dispatchers that analyze the video from the vehicle cabin and highlighted the dangerous states. The user of the platform could also specify the categories to test the models over a particular class or subclass. Nighttime and soft light conditions have the same structure.

IV. Results

We had successfully launched the platform on the production server for the Drive Safely mobile application [1]. It can evaluate about 700 images per second using RTX 2080 Super GPU for our models. Public access is just for testing part of our models on user data. For the model detection, the user can choose the method of testing (on built-in testing data set, uploaded data set, or on a single image). For each, the platform gives the user the ability to change confidence parameters for detected classes. First, the user should choose the models he/she want to test. We propose two main possibilities: (1) test one model and (2) compare between two different models. As well, an option to change the image size forwarded to the neural network which affects the running time directly. The confidence threshold could be chosen manually by the user or loaded from predefined values or finding the best confidence threshold according to the algorithms mentioned before. All options are selected by default. The user should uncheck the options according to the needs. Pressing Test Button will take a while then generate a report that includes the evaluation results specified by the user (see Fig. 5).

V. Conclusion

In this paper, we proposed a platform to evaluate the machine learning models in the object detection and classification in the topic of the driver monitoring system. This platform evaluates the models taking into account different metrics. And could convert these models into different types for inference with support to multiple ways of testing. For future work, we are going to enhance run-time complexity. the platform could test about 25 images per second on CPU Core i7 for input image size (416, 416). Also, different functionalities will be provided to support different types of models and for the other tasks like (face detection, facial landmarks, face reconstruction, pose estimation, face recognition, and hand detection).

VI. Acknowledgment

The research is funded by the Russian Science Foundation (project 18-71-10065). Platform architecture development has been supported by the Russian State Research 0073-2019-0005. Testing report generation has been supported by the grant of ITMO University (project № 618268).

References

[1] A. Kashevnik, I. Lashkov, A. Gurtov, ”Methodology and Mobile Application for Driver Behavior Analysis and Accident Prevention”, *IEEE Transactions on Intelligent Transportation Systems*, 2019, Vol. 21(6), pp. 2427–2436,

[2] A. Kashevnik, I. Lashkov, A. Ponomarev, N. Teslya, A. Gurtov, ”Cloud-Based Driver Monitoring System Using a Smartphone”, *IEEE Sensors*, IEEE. 2020. Vol. 20(12). pp.6701–6715.

[3] A. Kashevnik, A. Ali, I. Lashkov and N. Shilov, ”Seat Belt Fastness Detection Based on Image Analysis from Vehicle In-abin Camera, *” 2020 26th Conference of Open Innovations Association (FRUCT), Yaroslavl, Russia,* 2020, pp. 143-150,

[4] S. Sukaridhoto, D. K. Basuki, H. Yulianus, and R. P. N. Budiarti, “Performance Evaluation of Integrated Deep Learning Web Platform for Dataset Training”, *ATCSJ, vol. 2, no. 2,*Mar. 2020.pp. 117-128,

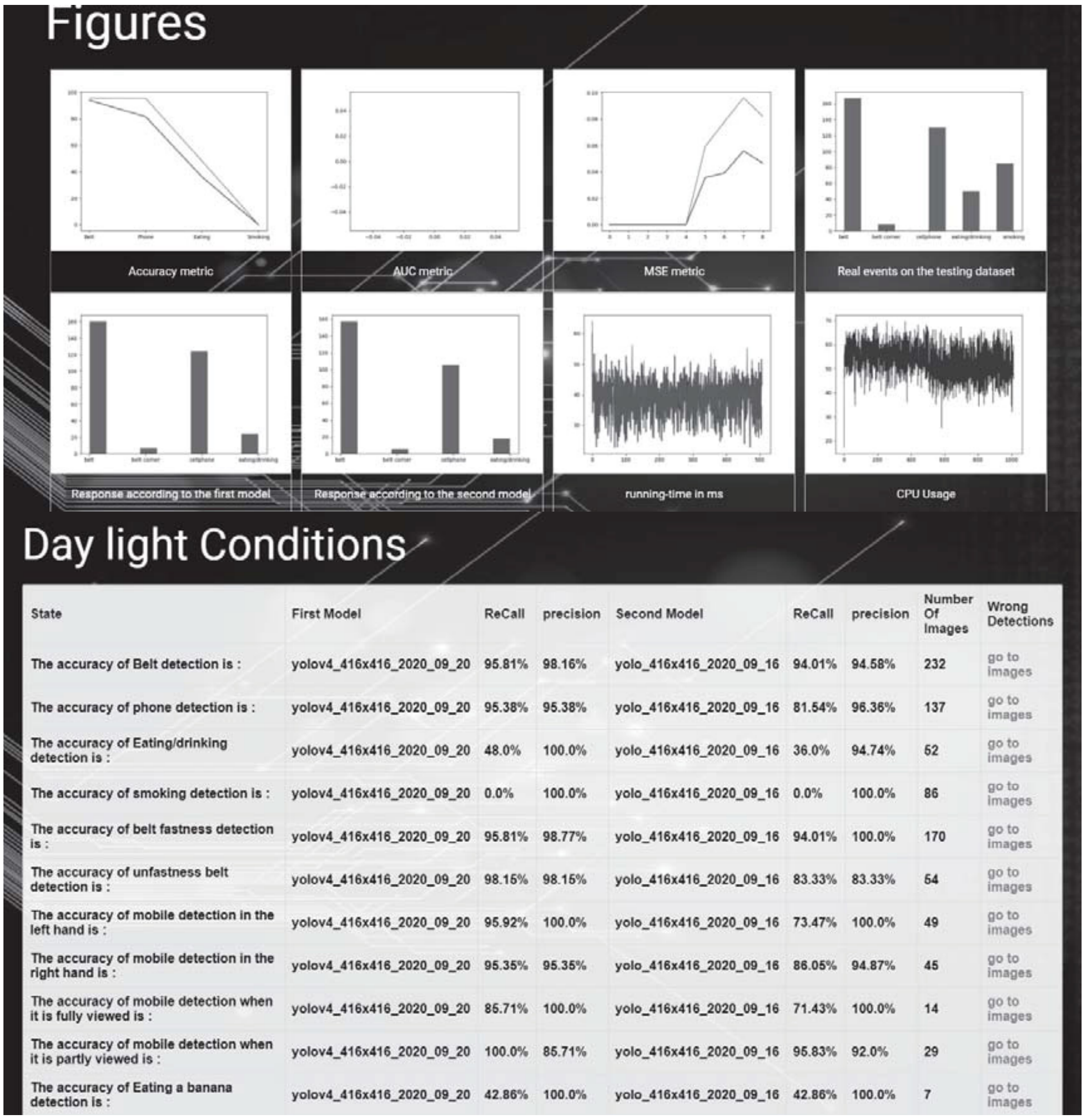
[5] B. Stuart and V. Richard. WebQual: An Exploration of Web-Site Quality. *Conference: Proceedings of the 8th European Conference on Information Systems, Trends in Information and Communication Systems for the 21st Century, ECIS 2000, Vienna, Austria*(2000). pp.298-305.

[6] G. Panagiotis and K. Konstantinos and P. Peter. (2019). Neptune: Scheduling Suspendable Tasks for Unified Stream/Batch Applications. *Conference: the ACM Symposium* 2019 pp.233-245. 10.1145/3357223.3362724.

[7] A. Martín , B. Paul , C. Jianmin , C.Zhifeng , D. Andy , D. Jeffrey , D. Matthieu , G. Sanjay , I. Geoffrey , I. Michael , K. Manjunath , L. Josh , M. Rajat , M. Sherry , M. Derek , S. Benoit , T. Paul , V. Vijay , W. Pete and Z. Xiaoqiang. TensorFlow: A system for large-scale machine learning. (2016).

[8] https://www.tensorflow.org/tensorboard/get_started

[9] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss and R. Farivar, ”Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools, *” 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Portland, OR, USA,* 2019, pp. 1471-1479, doi: 10.1109/ICTAI.2019.00209.


Figures
Accuracy metric
AUC metric
MSE metric
Real events on the testing dataset
Response according to the first model
Response according to the second model
running-time in ms
CPU Usage
Day light Conditions

State	First Model	ReCall	precision	Second Model	ReCall	precision	Number Of Images	Wrong Detections
The accuracy of Belt detection is :	yolov4_416x416_2020_09_20	95.81%	98.16%	yolo_416x416_2020_09_16	94.01%	94.58%	232	go to images
The accuracy of phone detection is :	yolov4_416x416_2020_09_20	95.38%	95.38%	yolo_416x416_2020_09_16	81.54%	96.36%	137	go to images
The accuracy of Eating/drinking detection is :	yolov4_416x416_2020_09_20	48.0%	100.0%	yolo_416x416_2020_09_16	36.0%	94.74%	52	go to images
The accuracy of smoking detection is :	yolov4_416x416_2020_09_20	0.0%	100.0%	yolo_416x416_2020_09_16	0.0%	100.0%	86	go to images
The accuracy of belt fastness detection is :	yolov4_416x416_2020_09_20	95.81%	98.77%	yolo_416x416_2020_09_16	94.01%	100.0%	170	go to images
The accuracy of unfastness belt detection is :	yolov4_416x416_2020_09_20	98.15%	98.15%	yolo_416x416_2020_09_16	83.33%	83.33%	54	go to images
The accuracy of mobile detection in the left hand is :	yolov4_416x416_2020_09_20	95.92%	100.0%	yolo_416x416_2020_09_16	73.47%	100.0%	49	go to images
The accuracy of mobile detection in the right hand is :	yolov4_416x416_2020_09_20	95.35%	95.35%	yolo_416x416_2020_09_16	86.05%	94.87%	45	go to images
The accuracy of mobile detection when it is fully viewed is :	yolov4_416x416_2020_09_20	85.71%	100.0%	yolo_416x416_2020_09_16	71.43%	100.0%	14	go to images
The accuracy of mobile detection when it is partly viewed is :	yolov4_416x416_2020_09_20	100.0%	85.71%	yolo_416x416_2020_09_16	95.83%	92.0%	29	go to images
The accuracy of Eating a banana detection is :	yolov4_416x416_2020_09_20	42.86%	100.0%	yolo_416x416_2020_09_16	42.86%	100.0%	7	go to images

Fig. 5. Results example: comparison of the model built in based on Yolo v3 and Yolo v4 on the same dataset

[10] D. Zhou et al., ”IoU Loss for 2D/3D Object Detection,” 2019 International Conference on 3D Vision (3DV), *Québec City, QC, Canada, 2019,* pp. 85-94, doi: 10.1109/3DV.2019.00019.

[11] D. Zhou et al., ”IoU Loss for 2D/3D Object Detection,” *2019 International Conference on 3D Vision (3DV), Québec City, QC, Canada,* 2019, pp. 85-94, doi: 10.1109/3DV.2019.00019.

[12] M. Zahra and M. H. Essai and Ali R. Abd Ellah Performance Functions Alternatives of Mse for Neural Networks Learning, *International journal of engineering research and technology*, 2014,

[13] D. Jesse and G. Mark The Relationship between Precision-Recall and ROC Curves *Proceedings of the 23rd International Conference on Machine Learning. Pittsburgh, Pennsylvania, USA* 2006 pp. 233 - 240

[14] M. Everingham , L. Gool , C. Williams , J. Winn and A. Zisserman The PASCAL Visual Object Classes (VOC) Challenge *International Journal of Computer Vision manuscript*2007.

[15] M. Huizi , Y. Xiaodong and D. William J. A Delay Metric for Video Object Detection: What Average Precision Fails to Tell *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* 2019. pp. 573-582