IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 12, 2022, accepted February 11, 2022, date of publication February 24, 2022, date of current version March 8, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3154709

XtremeAugment: Getting More From Your Data Through Combination of Image Collection and Image Augmentation

SERGEY NESTERUK[1], SVETLANA ILLARIONOVA[1], TIMUR AKHTYAMOV[1], DMITRII SHADRIN[1,2], ANDREY SOMOV[1], MARIIA PUKALCHIK[1], AND IVAN OSELEDETS[1]

[1]Skolkovo Institute of Science and Technology, 121205 Skolkovo, Russia
[2]Department of Information Technology and Data Science, Irkutsk National Research Technical University, 664074 Irkutsk, Russia

Corresponding author: Sergey Nesteruk (sergei.nesteruk@skoltech.ru)

This work was supported by the Russian Foundation for Basic Research (RFBR) under Project 19-29-09085 MK. The work of Svetlana Illarionova and Ivan Oseledets was supported by the Analytical Center under the RF Government under Grant 000000D730321P5Q0002 and Grant 70-2021-00145 02.11.2021.

ABSTRACT Deep convolutional neural networks are highly efficient for computer vision tasks using plenty of training data. However, there remains a problem of small training datasets. For addressing this problem the training pipeline which handles rare object types and an overall lack of training data to build well-performing models that provide stable predictions is required. This article reports on the comprehensive framework *XtremeAugment* which provides an easy, reliable, and scalable way to collect image datasets and to efficiently label and augment collected data. The presented framework consists of two augmentation techniques that can be used independently and complement each other when applied together. These are Hardware Dataset Augmentation (HDA) and Object-Based Augmentation (OBA). HDA allows the users to collect more data and spend less time on manual data labeling. OBA significantly increases the training data variability and remains the distribution of the augmented images being close to the original dataset. We assess the proposed approach for the apple spoil segmentation scenario. Our results demonstrate a substantial increase in the model accuracy reaching 0.91 F1-score and outperforming the baseline model for up to 0.62 F1-score for a few-shot learning case in the wild data. The highest benefit of applying *XtremeAugment* is achieved for the cases where we collect images in the controlled indoor environment, but have to use the model in the wild.

INDEX TERMS Image augmentation, computer vision, image segmentation, data collection, Internet of Things, few-shot learning.

I. INTRODUCTION

In recent years deep convolutional neural networks proved to reach the state-of-the-art performance on computer vision (CV) tasks such as classification [1], semantic segmentation [2], object detection [3], domain adaptation [4], pose estimation [5], etc.

In comparison with classic CV algorithms, deep learning (DL) models are trained rather than programmed [6]. It makes them more flexible for different domains and requires less domain-specific knowledge from the developer because the model retrieves patterns directly from data.

The associate editor coordinating the review of this manuscript and approving it for publication was Essam A. Rashed.

The drawback of DL models is that they heavily rely on data and require comprehensive training datasets [7]. The more parameters the model has, the more complex features it can learn [8], but with sufficient training samples only [9].

Indeed, there are many CV related datasets both in the general domain [10] and in specific domains such as autonomous vehicles [11], remote sensing [12], medicine [13], precision agriculture [14], environmental study [15]. However, datasets cannot cover all the existing tasks for every specific problem [16]. Recent research works report that data scientists spend up to 80% of time for data preparation [17]. At the same time, the number of research papers about data in CV is much lower than the percentage of time spent on data in practice (see Figure 1). As expected, we see that this gap

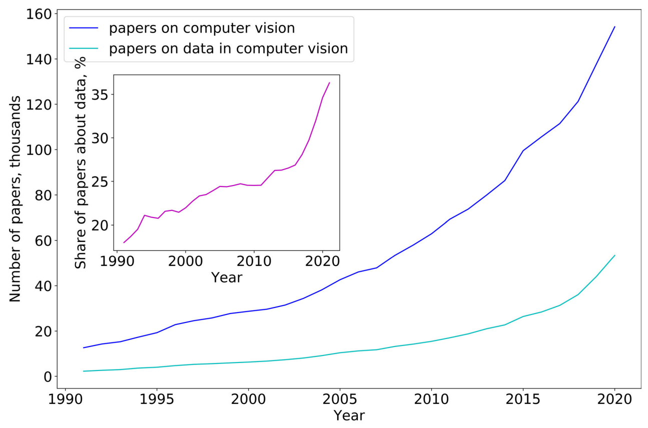


FIGURE 1. Comparison of the total number of papers in CV with the number of papers about data in CV according to WoS. To reproduce the result, one can query the number of papers with at least one of the following keywords: "computer vision", "convolutional network", "pattern recognition", "image", "CNN". To get the number of CV papers that focus on data, we propose to add at least one "data", "dataset", "augmentation", "collection" obligatory keywords to the previous query.

reduces over time, and one can see the exponential increase in the number of research works focused on data in accordance with Web of Science (WoS) [18]. In general, data-centric computer vision methods are aimed to create the data processing pipelines that will increase the overall performance without altering the model. In practical problems, pipelines are essential because they serve as a bridge between the datasets and models.

The primary goal of the papers that focus on dataset preparation is to increase the diversity of training data and improve the model generalization. Some works propose the model pretraining methods [19] or aim to collect new datasets [16]. Other works present the image augmentation techniques [20] or feature-engineering approaches [21]. However, to the best of our knowledge, no studies provide the comprehensive pipelines that cover all the steps from the image collection and labeling to the image augmentation.

In this article, we present the *XtremeAugment* framework that handles and improves both image collection and image augmentation for the cases when a data scientist has to collect images for the experiment. The article shows the efficiency of the framework when applied to the digital agriculture domain. Digital agriculture is the technological paradigm which addresses the global hunger problem by improving the crop cultivation productivity [22] at all stages from seeding to storing. It includes the intelligent plant condition sensing, measuring responses to external impacts, treatment optimization, yield prediction, spoiling detection, etc. The main limitation of this domain is the insufficient amount of well-annotated data [23]. To overcome this issue, we present the *XtremeAugment* framework.

The framework consists of two independent parts: the Image Collection System (ICS) and the Object-Based Augmentation (OBA) library. The ICS part is designed to be flexible, reliable, and easy to use. It is built on a declarative programming paradigm [24] which allows its application without the coding skills. The key feature of the ICS is the support of the Hardware dataset augmentation (HDA). HDA denotes taking images from multiple viewpoints and changing the lighting conditions during imaging to increase the number of natural images from the same scene. It is a powerful technique to expand the dataset size because all the images collected from the same viewpoint and various lighting conditions share the same labeling. It, therefore, requires neither additional time to collect the data nor auxiliary manual labeling.

OBA is the image augmentation approach that exploits the instance segmentation masks to cut objects from the original scene and then paste them to the new background [25]. It is called object-based because it treats each object individually. This approach provides a way to augment objects separately, to explicitly control the number of objects in the scene and to manage the overlapping of objects.

It is important to note that both HDA and OBA can be used separately. Their sequential combination forms *XtremeAugment*. Moreover, OBA does not exclude other generic types of augmentations (such as rotation, flip, Gaussian noise, Mixup [20], Cutout [26], Mobius transformations [27]) and can use them as building blocks. Accumulating the advantages of hardware and software augmentations, *XtremeAugment* provides much more training data and, therefore, ensures better model generalization.

The main contributions of this paper are:

- we revise the definition of dataset augmentation and extend it with the data collection techniques;
- we introduce the reliable and easy-to-use image collection system (ICS) which supports the hardware dataset augmentation (HDA);
- we study the effectiveness of the object-based augmentation (OBA) using the apple rotten segmentation problem as an example;
- we propose the state-of-the-art *XtremeAugment* algorithm which unites HDA with OBA;
- we collect and share the apples rotting image dataset and the auxiliary backgrounds dataset.

The code for HDA and OBA is available online. The dataset is available on the Kaggle platform.[1]

The rest of the paper is organized as follows. In Section II we discuss the state-of-the-art in the area. Section III presents the methodology of hardware and object-based image augmentation. The Experimental setup is described in details in Section IV. Section V provides the results with the *XtremeAugment* framework. Appendix A explains the general hardware related methodology. Appendix B shows the details of the current system implementation.

II. RELATED WORK

In this section, we overview existing image collection systems and image augmentation methods. They are described

[1]Kaggle (https://www.kaggle.com/) is a popular platform in the machine learning community, which allows sharing datasets, hosting competitions, and discussing results.

separately because, before our paper, they were not combined into a single framework.

A. IMAGE COLLECTION SYSTEMS

As mentioned in the previous section, the data collection process plays a crucial role in Data Science. In many domains and cases, dataset collection is not a just time-consuming procedure but also is complicated due to the following reasons:

- historical (sequential) data is required for each object;
- multiple views are required (for images/videos/point clouds), or multiple sources of the data are required, e.g., camera images and sensor data;
- conditions changing is required, e.g., light conditions or plant treatment; item-specific actuation is required, e.g., plant watering.

In this paper, we focus on the precision agriculture domain. The specific difficulty in plant phenomics is the complex plant structure and rapid variability of this structure. Thus, many works are dedicated to datasets or IoT systems for digital agriculture [28]. In [29], authors collected a dataset that contains images of different plant species cultivated under three different growth conditions in a semifield setting. The imaging system mounted on the railing system contains a global shutter camera, Xenon ring flash, Nvidia Jetson TX2 based computer, and solid-state drive (SSD) storage. Some sensors and actuation were embedded in the construction of the setup for automated rain protection.

In [30] authors collected a large dataset of images of species in greenhouses captured in various conditions. Three types of cameras were used in this work: IoT cameras, mobile cameras, and smartphone cameras, which are capable of tracking the growth process of a single type of crop and a variety of crops. The watering process was automated using an autonomous mobile robot, and pinking was also automated using a robot. Images are uploaded from devices to cloud servers.

In [31] authors propose a dataset that contains images of different species at several growth stages acquired with comparable light conditions. A camera mounts on a special rig setup. Along with the specific construction of the rig, an embedded camera flash was used to achieve comparable light conditions, and an external trigger was used to avoid vibrations. To track the species, they use a barcode-based system.

The work [32] is dedicated to the collection of Australian rangelands species dataset for robotic weed control application authors used custom data logging device for recording images and Global Positioning System (GPS) data. This device consists of a Raspberry Pi 3 board, high-resolution Ethernet RGB camera, Arduino board with custom electronics shield, GPS receiver with external antenna, and touchscreen display module. All components are mounted on a tripod.

The work [33] is dedicated to the collection of the dataset of Fuji apple trees point clouds for the fruit detection task. The authors used a special mobile system that consists of a laser imaging, detection, and ranging device (LiDAR), a real-time kinematics global navigation satellite system, and an air-assisted sprayer. The dataset was recorded under different views and air-flow conditions generated by the sprayer.

There is a number of other works that consider application of mobile platforms like human-controlled vehicles [34]–[36] and special field robots [37]–[40] for recording agricultural datasets in the field. Besides RGB cameras, those systems contain RGB-D cameras [34], [39], [40], LiDARs [39], thermal and hyperspectral cameras [40].

Digital agriculture is not the only example of areas where data collection requires sophisticated systems. One can also mention retail industry [41], medicine [42], electrical systems [43], [44], industrial manufacturing [45] and many others.

One can see that data collection can be challenging. In many cases, it is performed manually [46], which requires hundreds of hours and much effort. Many papers automate the data collection process, but they have to use custom software solutions. Thus, implementing a unified and flexible framework that can be reused between different domains and applications can dramatically simplify the process of such datasets collection.

B. IMAGE AUGMENTATION

It is important to have training samples that cover cases that the model should be able to predict [9]. No model can classify a type of object that was not in the training set. Therefore, collecting more training data is always beneficial for model accuracy [8]. However, it is not always possible to collect more data [16]. Moreover, in practical tasks, regardless of the number of collected samples, it is usually possible to generate more samples via image augmentation [47] if there are no domain-specific restrictions [48]. Researches show that such architectures as generative adversarial networks (GANs) and such training approach as metric learning highly rely on image augmentation [49].

Image augmentation is applied from the beginning of computer vision. It mainly included generic image transformations such as rotation, flipping, color shifting, etc. With the rise of deep learning and the dramatic increase of modern models' capacity, augmentation techniques became indispensable in computer vision. Models with many learnable weights require more training data to reach an accuracy plateau. Augmentation is also performed as a regularizer during the training process [50]. More researches started to focus on model training pipelines, but not only model architectures (see Figure 1), so image augmentation approaches become more advanced. One of the first definitions of image augmentation may be stated as is a technique that applies various transformations to original images to increase training data variability [51]. The critical point is that it was assumed that new samples were obtained from the existing data. Now, apart from image augmentations via transformations, we can distinguish new sample synthesis and other approaches [52]. In [53], authors use GANs to vary training samples.

In [54], they used 3D modeling to generate new artificial scenes with sweet peppers. Therefore, augmentation is no longer restricted to the processing of the original data. In [55] authors show an efficient way to augment data of various modalities, including images in latent space. This is another showcase where augmentation is applied not to the original dataset. In [56] authors show that emitting supplementary green light during data collection is beneficial in some plant phenotyping computer vision problems. This technique also can be considered image augmentation. Another case when a model is trained not on the collected data is the dataset distillation. The training dataset consists of distilled image representations rather than original images in this approach [57]. In this paper, we use the term *dataset augmentation* as a general way to describe data manipulations used to pre-process data before or during training. According to our definition, dataset augmentation is a set of techniques that artificially increase the number of data samples used for model training or evaluation. It includes data collection, image augmentation, feature space manipulation, and other techniques. Also, note that usually, augmentation is used for model training, but some methods can be successfully applied for test-time augmentation (TTA).

In recent years many new image augmentation algorithms have been introduced. In [25], authors propose to split them into image-based and object-based augmentations. Image-based approaches target whole images. It can be either basic geometrical and color transformations or combining multiple images like in Mixup [20]. Object-based augmentations target separate objects. They utilize available instance segmentation masks or bounding boxes to cut objects from the original images and paste them to a new scene. For example, Mixup has an object-based adaptation called CutMix that cuts objects by their bounding boxes before pasting them to a new scene [58]. Direct manipulation with objects of interest provides more flexibility and enables substituting the visual domain [59]. It is an up-and-coming concept, and in this work, we focus on it.

First object-based augmentations started by modeling the visual context around the object of interest. [60]. The authors in [61] stated that it is the key to performing augmentation for object detection successfully. However, in [25] and [62], authors show that naive cutting and pasting also improve the model performance sufficiently. While first papers on object-based augmentation focused on single computer tasks like object detection [63] or instance segmentation [62], in [25] authors extend this idea to solve many problems such as classification, semantic segmentation, object detection, and object counting simultaneously.

III. XTREMEAUGMENT METHODOLOGY

Almost all the practical and research computer vision problems require image augmentation. Nevertheless, it is vital to understand that not all transformations improve the quality of the training set. Having several copies of the same image increases the quantity of the available data. However, to be helpful for the model training, new samples must differ from the original image to add variability, and at the same time, the target object must be recognizable. Figure 2 reflects the difference between augmentation approaches. One can see that without any augmentation, the training set entirely belongs to the set of natural images. It means that these are images that can be obtained if one makes some image with a camera. The problem with leaving the training set as it is in its small size for most real-world problems. Applying regular augmentation adds variability and remains the distribution of the extended set close to the distribution of the original one. However, the provided enlargement is not efficient. It is particularly crucial for the few-shot learning cases [64]. In contrast, if the substantial augmentation is applied, it adds much variability, but many samples will differ from the original distribution too much. In [65], authors show that it is beneficial when the augmented set aims to match the original one. In [66], the authors demonstrate that too strong augmentation makes an object not recognizable, and it is better to treat it as a negative training sample. The goal of *XtremeAugment* is to significantly increase the number of training samples and leave augmented set distribution closer to the original training set. To achieve this, *XtremeAugment* combines hardware and object-based image augmentations.

A. HARDWARE DATASET AUGMENTATION

Hardware dataset augmentation denotes altering natural scene setup during image collection. In paper [67], authors show that changing artificial lighting direction is significant to improve model performance in greenhouse conditions. In [68], authors report that controlling lighting conditions can improve outdoor results as well. We do not expect it to be a universal key to boost computer vision, although it looks promising for the cases where it is possible to control lighting conditions manually.

There are many other ways to add variability to original images. We propose to use different types of cameras: web cameras, Single-lens reflex (SLR) cameras, night-vision cameras, multispectral cameras, and other types depending on the problem. Another valuable modification is to place multiple cameras to catch the scene from different viewpoints simultaneously (Figure 4).

B. OBJECT-BASED AUGMENTATION

Object-based augmentations use object masks to crop the object of interest from the original image and paste it to a new background. Among existing OBA mechanics, as a baseline, we choose one reported in [25]. It is a naive solution that does not model the visual context on the scene, but it is easy to use and can be effective for many cases. Some interesting features of this approach are as follows:

- the ability to control objects overlapping;
- the ability to control the scene's orientation;
- support of the scene perspective transform;
- the ability to generate target masks for multiple computer vision problems.

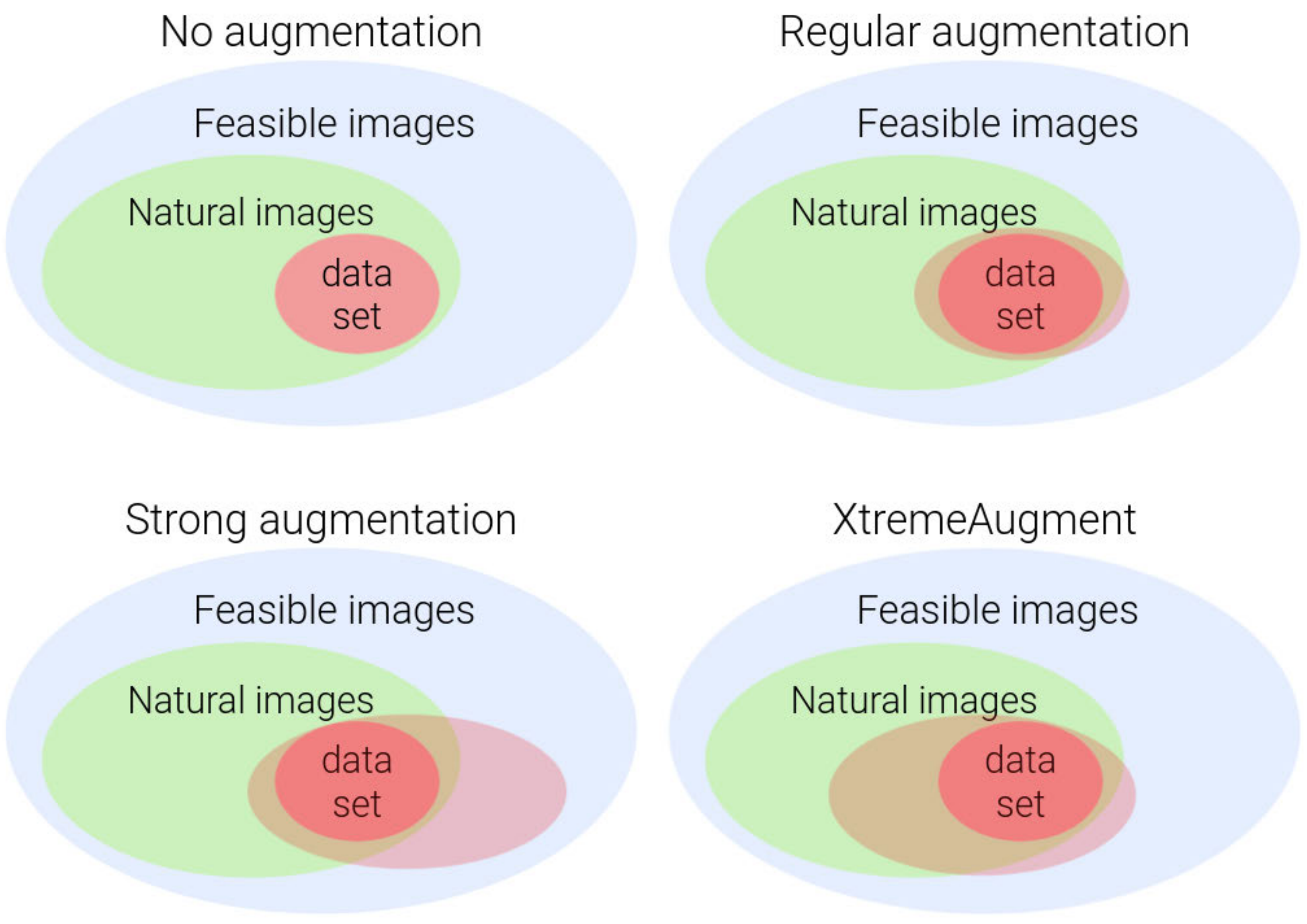


FIGURE 2. ***XtremeAugment* concept. The outer blue oval represents the set of all possible images. The green oval in the middle represents the set of all natural images. A small red oval represents the original training set. The bigger translucent red oval represents the augmented training set obtained with different approaches.**

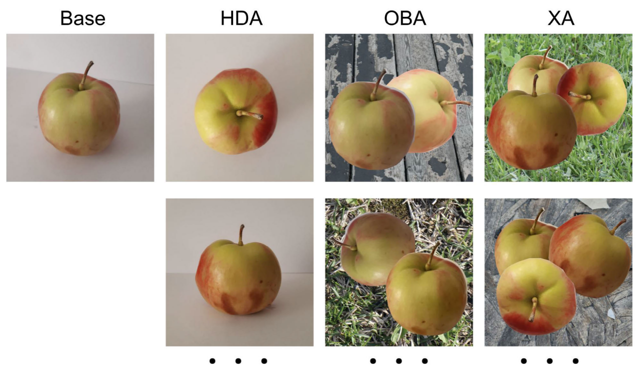


FIGURE 3. Example of a base image and images obtained by different augmentation approaches. The number of augmented samples can vary. Baseline data is a single image. HDA data is images collected from multiple viewpoints and with different light conditions. OBA composes multiple separate objects and adds a new complex background. *XtremeAugment* combines HDA and OBA.

This framework allows performing the task that we call Lab2Wild adaptation. It means collecting images in a laboratory or some other controlled indoor environment and enabling successful model inference in the wild. To achieve this, we can combine several cropped objects in a background from the general domain or the target domain if possible.

The overall idea behind *XtremeAugment* is to use both HDA and OBA (Figure 3). The motivation is as follows. HDA is performed during data collection and captures a real scene with various imaging settings. It guarantees to add extra information to the dataset and keeps all of the samples in the set of natural images. Some viewpoints and light conditions can contribute more than others, but no noise is added at this stage. OBA provides more flexibility. It adds new backgrounds to the data and augments objects separately. As a result, we can get the combinatorial effect by altering backgrounds and replacing objects. The core idea with many possible generated scenes is that every step adds variability, and we do not need very strong augmentation to have many slightly different training samples.

The hypothesis that we test further in this paper is that many modest transformations can work better than a small number of significant transformations.

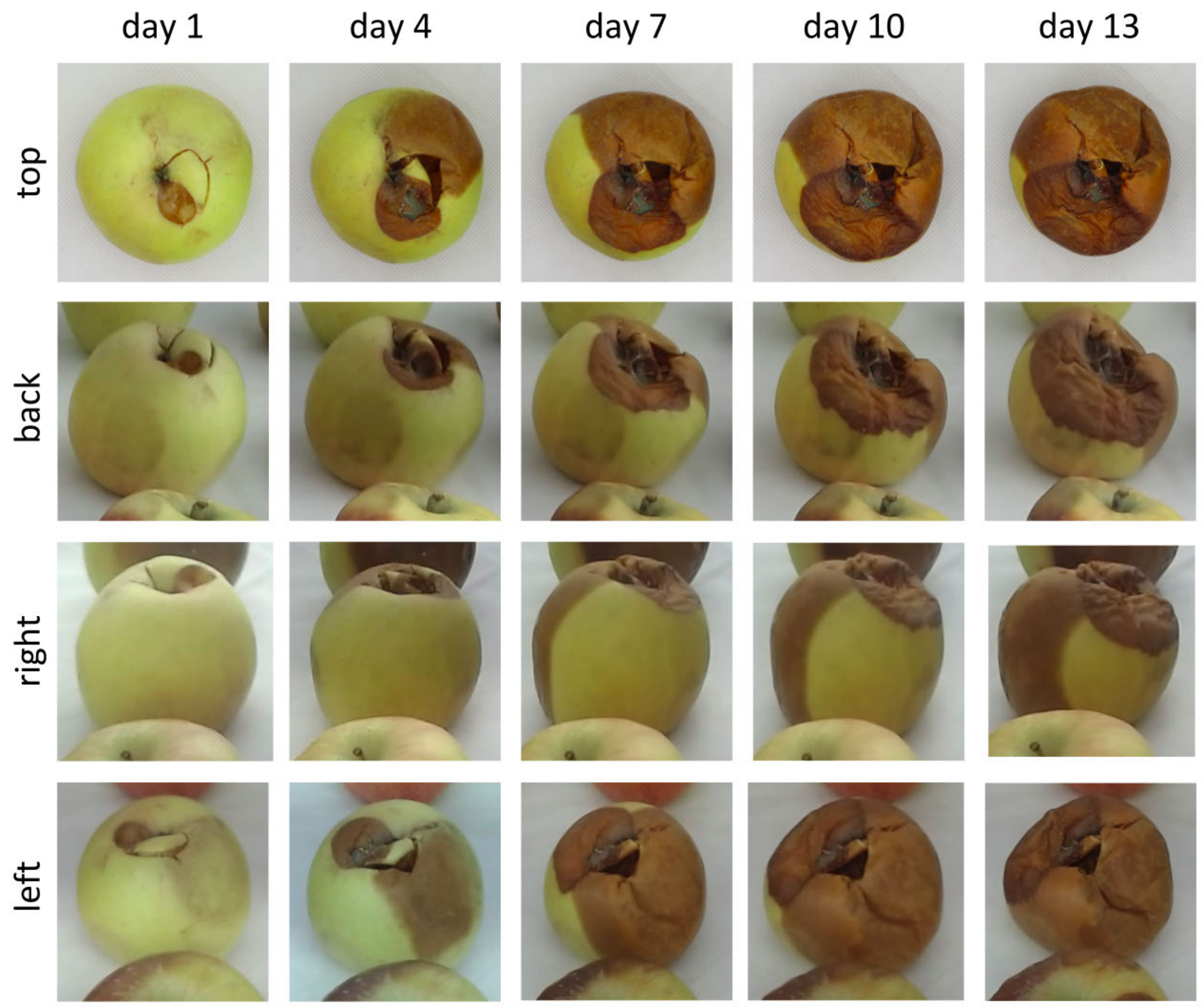


FIGURE 4. Multi-view data collection example.

We set the following ranges for the regular augmentations (modest transformations). For contrast [0.9, 2.1]; for brightness [0, 60]; for saturation [0.5, 2]. Their combinations are also considered. Strong augmentations are outside the aforementioned ranges. Examples of image transformations with different intensities are shown in Figure 5.

IV. EXPERIMENTAL SETUP

A. DATA COLLECTION

To check the performance of *XtremeAugment*, we test it on the apple spoiling segmentation problem. For data collection, we use ICS implemented on Raspberry Pi 4 (see Figure 6).

The system frame comprises the aluminum beams and has a 1 m length, 1 m width, and 1.7 m height. Apples lay on a white tray at the height of approximately 1.3 m above the floor. We use four web cameras Logitech920c and one SLR camera Canon M50. One web camera is placed in the middle top of the frame. Three other web cameras are attached at different sides of the frame at the height of 1.5 m above the floor. An SLR camera is attached at the middle top of the frame. All the cameras are connected to the Raspberry Pi via a USB hub. Raspberry Pi also controls lighting. We use two light modes for this experiment: LED light from the sides, and from the top. The system makes sequential images from all cameras and both lighting conditions.

For the experiment, we choose four apple varieties with four apples each (see Figure 7). The varieties are *Golden Delicious*, *Fuji*, *Gala*, *Reinette Simirenko* (from up to down). Within one variety, every apple has different treatments (from left to right):

- untouched;
- well washed;
- damaged;
- supercooled under $-20\ ^{\circ}C$.

An untouched apple serves as a reference for each variety. Well washed means the removal of the natural protective wax

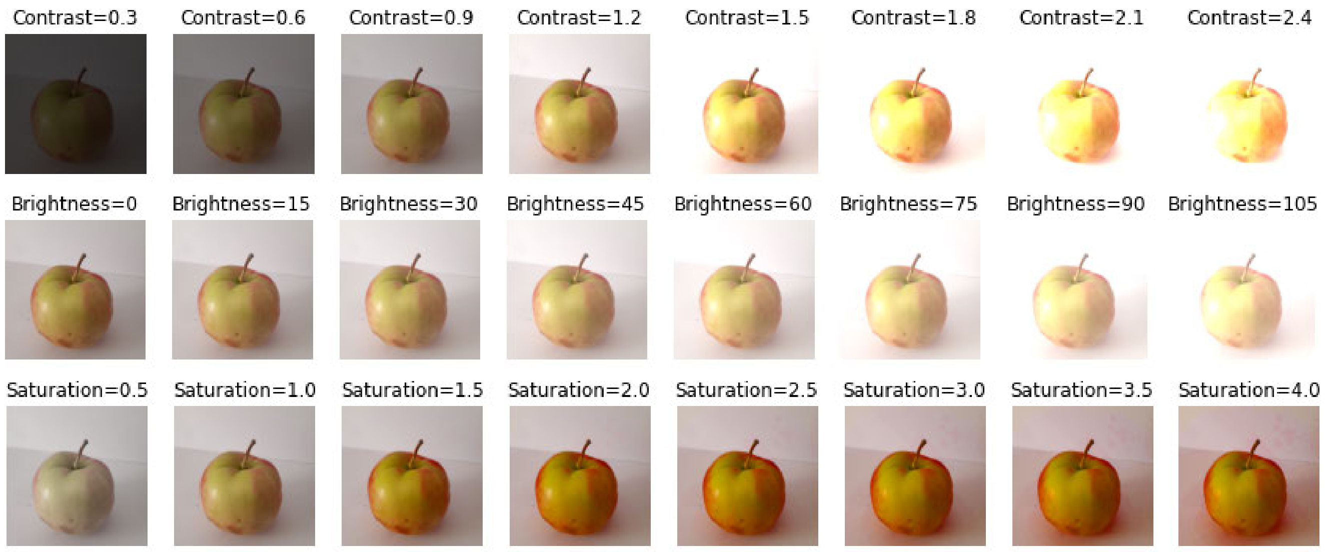


FIGURE 5. **Example of base augmentations with different intensities.**

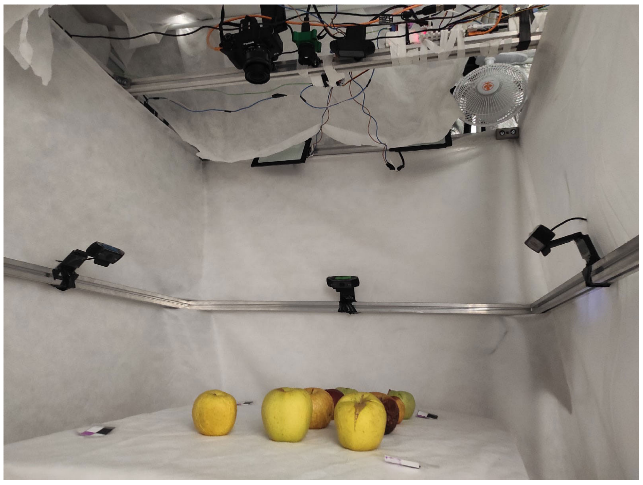


FIGURE 6. **ICS implementation for the apple spoiling segmentation problem.**

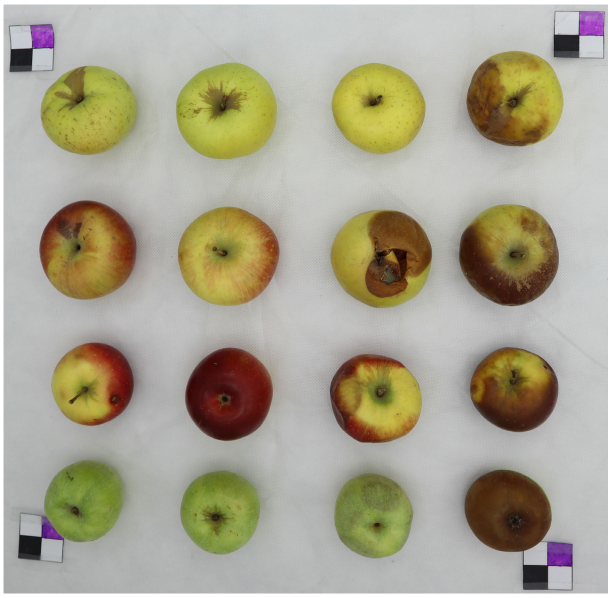


FIGURE 7. **Top view examples.**

layer from an apple. Damaged apple simulates wrong carriage conditions. Supercooled apple simulates wrong storing conditions. The ICS made images every six hours for about three weeks. During the experiment, some apples that were spoiled too much were removed.

To conduct a computer vision experiment, we have to split data into a training set, a validation set, and a holdout set. The training set is used to update the weights of a model via backpropagation. The validation set is used to check the model performance after each training iteration, but it is not used for actual training. The holdout set is used after the training process to report the final result. This type of split implies that only the training set amount of all three sets is used for actual training. Therefore effectively, we have less training data than it was collected. It is especially tangible for the problems characterized by the small datasets. However, it is required to separate the validation set to avoid overfitting and separate the holdout set to check model generalization on the unseen data, increasing the reliability of the result. The presence of several viewpoints in our collected dataset allows us to split the dataset according to that viewpoints. It prevents potential data leakages and also increases the reliability of the result.

To get a better understanding of the data leakage problem, we refer to Figure 8. D denotes the whole dataset. D_i is the i^{th} element of the set D. N is the number of images in the set D. $V^{(1)}$ denotes the subset with viewpoint 1. $V^{(2)}$ denotes the subset with viewpoint 2. $d(\cdot, \cdot)$ is the distance between two images. Here, the distance function measures the semantic dissimilarity between the images. In our showcase, it is an abstract function that has a zero value between an image and itself and raises according to the semantic differences between the compared images. In practice, it can be approximated by the distance between embeddings of a metric learning model [69]–[71]. However, we use it only to reflect the essence of the data leakage problem and qualitatively compare principal approaches to solve it.

Note that the presented case dataset contains the time-series images of the same objects, and images are sorted by time. Moreover, the apple rotting process is one-directional. Considering the irreversibility of apple modifications and the stationarity of camera viewpoints, we can state that:

- $d(V_i^{(k)}, V_i^{(k)}) = 0$;
- $d(V_i^{(k)}, V_j^{(k)}) \neq 0$, where $i \neq j$;
- $d(V_i^{(k)}, V_i^{(m)}) \neq 0$, where $k \neq m$;
- $d(V_i^{(k)}, V_{i+1}^{(k)}) \leq d(V_i^{(k)}, V_{i+2}^{(k)})$.

In our dataset, the time between images is short, and cameras are placed far from each other; therefore: $d(V_i^{(k)}, V_j^{(k)}) < d(V_i^{(k)}, V_i^{(m)})$.

We showcase the dataset splitting procedure on the example of splitting into two sets: a training subset $S^{(train)}$ and a test subset $S^{(test)}$. Note that $S^{(train)} + S^{(test)} = D$. The subsets must follow a number of major requirements:

1) Training and test subsets must not have intersections or very similar cases.
2) Both training and test subsets must cover the cases that the model aims to predict.

If the first requirement is not satisfied, it is impossible to detect the model overfitting. Very similar or precisely the same images are not suitable for checking the model generalization. If the second requirement is not satisfied, the user cannot expect a model to work on uncovered cases. For example, if the training subset has only healthy apples and no rotten apples are included, a model would not predict the spoilage. If the test subset has no rotten apples, it is impossible to evaluate model performance on the spoiled samples.

In Figure 8, we consider three principal approaches for dataset splitting on a simplified example. The first approach splits the samples from a single viewpoint uniformly between training and test sets. To simplify the notation, let the training set contain odd elements from $V^{(1)}$, and the test contains the even elements from $V^{(1)}$. When images are taken often, and the difference between them is negligible, this approach violates the first requirement because the distance between the training and the test sets is very low $d(S_i^{(train)}, S_i^{(test)}) \approx 0$. Taking images rarely is not a good option as well since extra data is beneficial in tasks with complex cases.

The second approach assigns the first part of the images from viewpoint 1 to the training set and the second half to the test set. In general, proportions and the exact split may differ, but the main point is that the subsets will uncover chunks of the dataset that can contain unique cases. In terms of the distance function, if the overall distance covered in a subset is much less than the distance covered by the whole dataset, the split will violate the second requirement: $d(S_{min(i)}^{(train)}, S_{max(i)}^{(train)}) \ll d(D_0, D_{N-1})$ and $d(S_{min(i)}^{(test)}, S_{max(i)}^{(test)}) \ll d(D_0, D_{N-1})$.

The third approach assigns all the data from the first viewpoint to the training subset and the second viewpoint to the test subset. It satisfies both the first and the second requirements and enlarges the amount of available training data. Beyond this example, more than two viewpoints can be used.

Certainly, splitting data by viewpoint is not the only way to obtain the correct splits. Having different available viewpoints simplifies this task. With HDA, a data scientist does not have to manually set a threshold of image similarity between the subsets without the risk of improper splitting results.

As we noted, the proper data splitting is vital to get the correct results, but it reduces the training set. The most expensive part of data collection is usually manual data labeling. It is demanded to do manual work to get the ground truth; in fact, it is labor-intensive. The feature of the HDA augmentation is that we collect more data altering the light conditions, but for various lighting, the position of the target objects remains the same. In practice, it means that we get supplementary images, but we do not have to label them because their masks will match.

The design of the experiment allows testing different setups of HDA and OBA and then unites them into *XtremeAugment*. To have HDA augmentation options, we include or exclude the corresponding parts from the dataset into the training subset. To vary OBA augmentation, we change the probability of using it for every batch. We mix the natural samples with the generated samples with a predefined ratio in OBA experiments. More precisely, we describe the experiments in Table 1.

B. APPLE SEGMENTATION

To assess the proposed augmentation approach, we apply it in a semantic segmentation task. For this task, a convolutional neural network ascribes one of three classes' labels for each pixel: damage, healthy, and background. 'Background' shows the accuracy of the separation of apples from the background. It is important in agricultural problems where the background is usually very complex. 'Healthy' shows the accuracy of finding health regions of apples. 'Damaged' shows the accuracy of finding cracks or rotten regions on apples. The ground truth data is obtained by manual annotation. For each pixel, the corresponding class is assigned. For each image, the target is a tensor with the spatial resolution of the input image and with a channel for each class. Only a single value across all channels for every pixel equals one in a channel which number coincides with the number of the class. The rest values are zeros. Batch for model training consists of

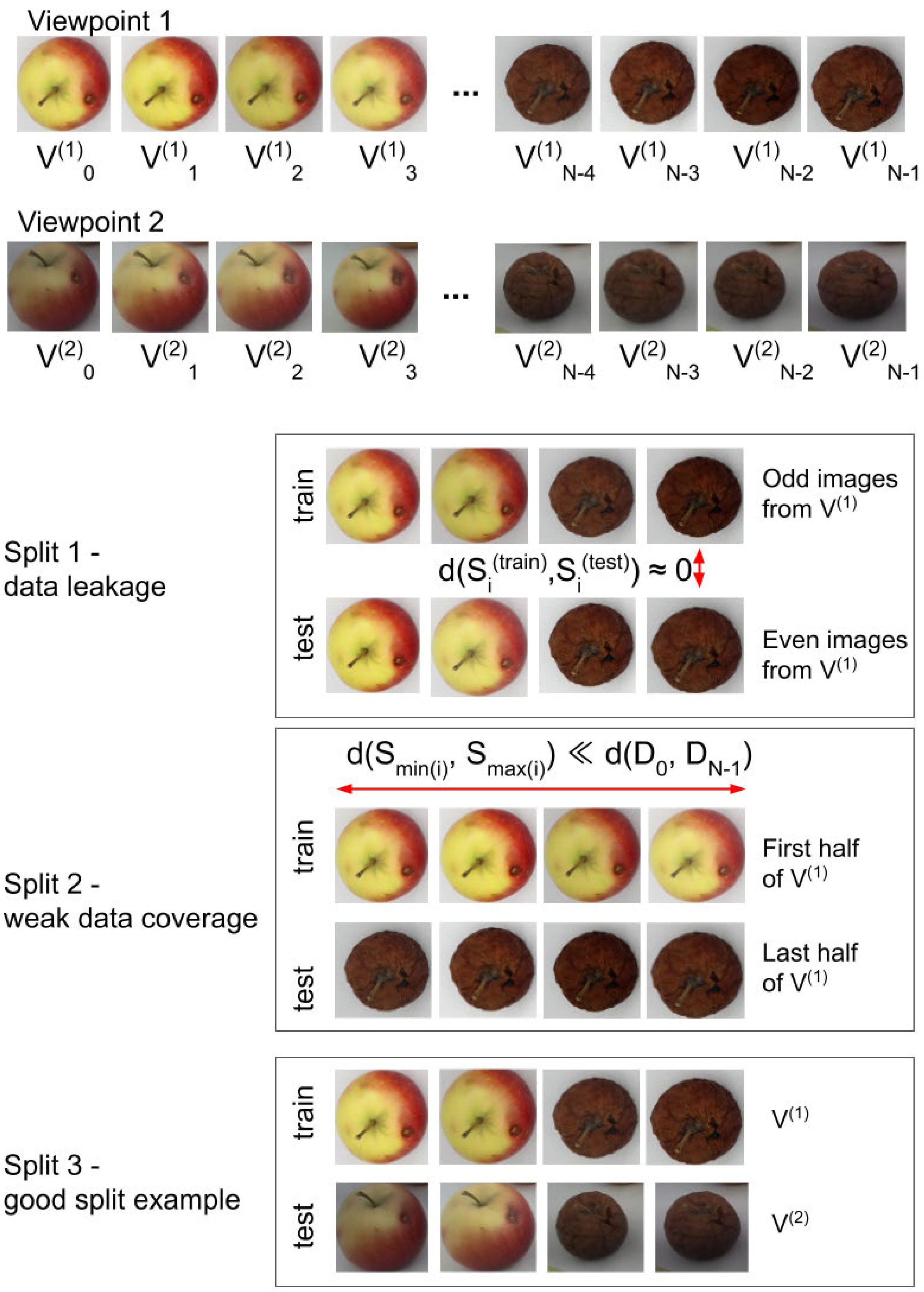


FIGURE 8. The importance of the correct data splitting strategy illustration.

TABLE 1. Experiments description. Rows represent experiments. Columns show which viewpoints or lighting conditions are included in the training set. The validation and the test set are the same for all the experiments.

Experiment	logitech right	logitech top	Canon top	side light	all light	OBA probability
Baseline	+	-	-	+	-	0
HDA_1_1	+	+	-	+	-	0
HDA_1_2	+	-	+	+	-	0
HDA_2_1	+	-	-	+	+	0
OBA_1	+	-	-	+	-	0.25
OBA_2	+	-	-	+	-	0.5
OBA_3	+	-	-	+	-	0.75
OBA_4	+	-	-	+	-	1
XA_1	+	+	+	+	+	0.25
XA_2	+	+	+	+	+	0.5
XA_3	+	+	+	+	+	0.75
XA_4	+	+	+	+	+	1

ten patches (512 per 512 pixels each) randomly sampled from the dataset. As a neural network architecture, we chose Deeplab [72], which shows high performance in semantic segmentation tasks. Encoder is Resnet34 [73]. Optimizer is Adam with the initial learning rate value equal to 0.001. An exponential scheduler decreases the learning rate during training. The activation function for the last layer is softmax. As a loss function, weighted cross-entropy loss is used with weights set proportionally to the classes distribution. Models are trained for 200 epochs with early stopping after ten epochs without loss improving. The model with the lowest loss value is used for further analysis. To evaluate model performance on the test images set, we use F1-score and weighted F1-score [74](Equation 1). F1-score for each class is weighted by the number of samples (pixels) from that class to provide a more balanced final averaged assessment.

$$F1_i = \frac{TP}{TP + \frac{1}{2}(FP + FN)},$$
$$F1_{weighted} = \frac{\sum_i^M F1_i * weight_i}{N} \tag{1}$$

where $F1_i$ – F1-score for the i-th class, M – the number of classes, $weight_i$ – the number of pixels of the i-th class, N – the number of all pixels, TP is True Positive (number of correctly classified pixels of the given class), FP is False Positive (number of pixels classified as the given class while in fact being of other class, and FN is False Negative (number of pixels of the given class, missed by the method).

In our experiment, we apply repeated random subsampling validation with three repetitions. It means that we repeat the data splitting and model training procedure three times with different dataset splits and average the results.

V. XTREMEAUGMENT RESULTS

Results for the semantic segmentation task are shown in Table 2. It includes predictions separately for indoor test images and wild outdoor images. The baseline in the table corresponds to the subset that was used for model training. More precisely, it is data from a single logitech_right viewpoint. Baseline_no_augment means that no augmentation was used during training. Baseline_regular_augment means training on the baseline data with commonly-used augmentation. Baseline_strong_augment means training with a higher number of augmentations and with higher transformations magnitudes. Note that experiments without augmentation and with too strong augmentations are added for demonstration purposes. The proposed *XtremeAugment* approach allows us to improve the baseline F1-score from 0.949 to 0.97. The best probability of using OBA during training is 50%, which means that only half of training samples are augmented using new backgrounds. Original images are vital for robust model performance, and they should not be excluded from the training as it leads to an F1-score decrease to 0.895. We also show the contribution of HDA and OBA approaches separately in the segmentation task. In the Lab test set, HDA allows prediction improvement from an F1-score of 0.949 to 0.963. OBA achieves an F1-score of 0.959.

The results clearly indicate that the baseline model fails to predict wild images (averaged F1-score is 0.29). The proposed approach exhibits significantly higher results for the wild test set (F1-score is 0.91). Models' predictions for wild images are presented in Figure 9. Although the training dataset does not include images with apples taken outside in various conditions, the *XtremeAugment* approach performs more confidently even on wild images.

Beyond the mean values of accuracy across three cross-validation repetitions, the standard deviation also provides essential statistics. In our experiment, it dropped from 0.082 in the baseline to 0.0054 with *XtremeAugment* on images in the wild. The decrease in results standard deviation indicates the increase in model stability.

The reason for the substantial improvement with our approach when the test subset is more complex than the training subset is in narrowing the gap between these domains (see Figure 2). More precisely, in the presented case, a model without OBA is not robust to complex backgrounds because only the uniform background is in the original training subset. HDA contributes to the robustness to change in illumination conditions and view angles.

OBA and HDA separately also improve the baseline as they provide more diverse training data. However, experiments indicate that OBA outperforms HDA in the wild images

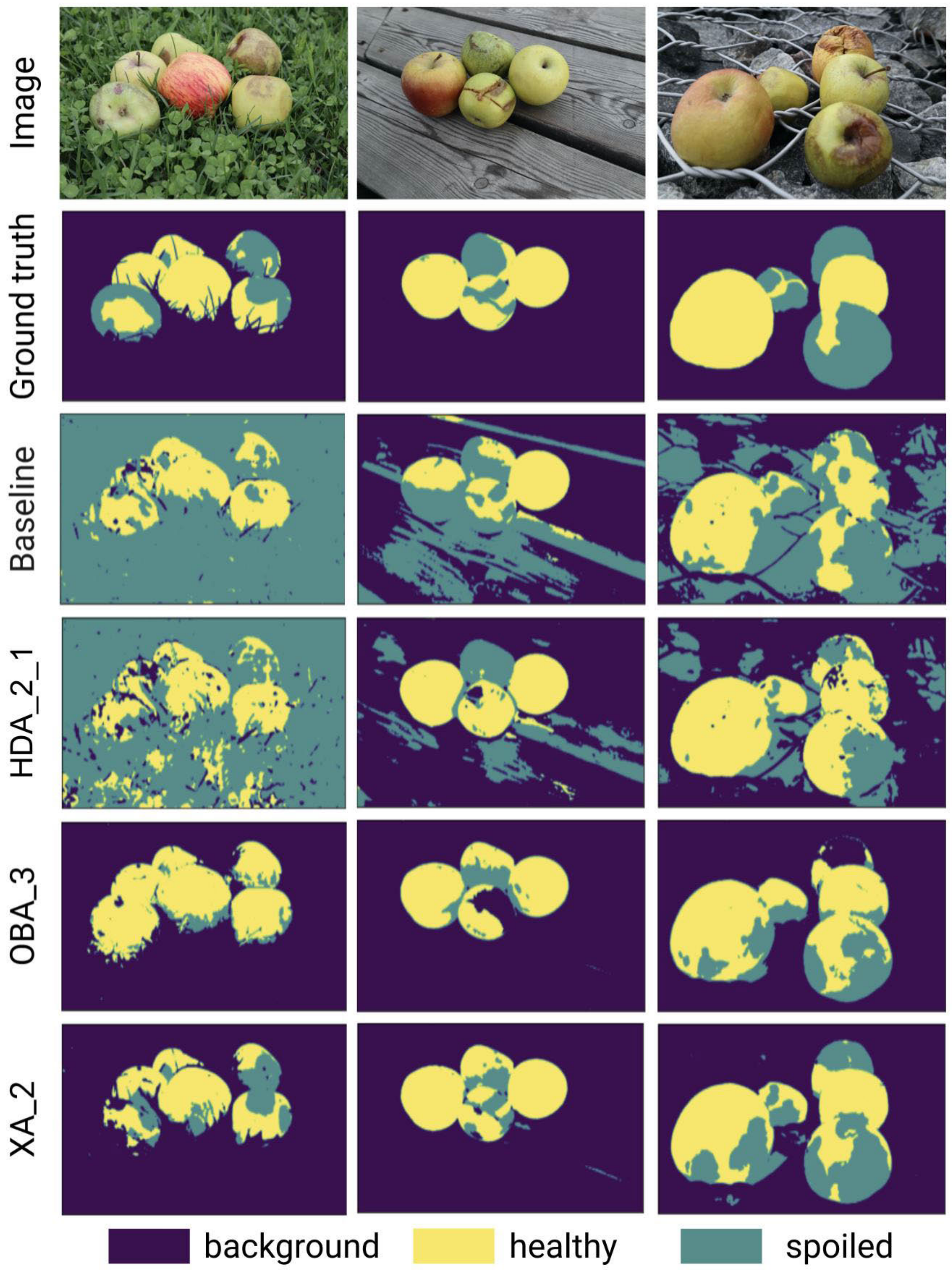


FIGURE 9. Results of the baseline, HDA, OBA, and XtremeAugment.

TABLE 2. Apple segmentation results (F1-score).

	Lab test set				Wild test set			
Experiment	Background	Damage	Healthy apple	Averaged	Background	Damage	Healthy apple	Averaged
Baseline_no_augment	0.988	0.721	0.855	0.923	0.22	0.1	0.55	0.2
Baseline_regular_augment	0.992	0.737	0.863	0.949	0.28	0.12	0.65	0.29
Baseline_strong_augment	0.94	0.653	0.842	0.91	0.2	0.08	0.51	0.18
HDA_1_1	0.992	0.772	0.884	0.954	0.285	0.136	0.68	0.33
HDA_1_2	0.993	0.818	0.908	0.963	0.25	0.125	0.63	0.31
HDA_2_1	0.992	0.761	0.879	0.953	0.3	0.16	0.7	0.35
OBA_1	0.992	0.804	0.898	0.959	0.958	0.29	0.72	0.88
OBA_2	0.991	0.769	0.860	0.947	0.956	0.27	0.69	0.87
OBA_3	0.991	0.774	0.879	0.951	0.958	0.3	0.7	0.88
OBA_4	0.959	0.701	0.736	0.899	0.966	0.34	0.74	0.89
XA_1	0.993	0.843	0.907	0.964	0.9	0.23	0.75	0.84
XA_2	0.992	0.834	0.915	**0.97**	0.97	0.49	0.75	**0.91**
XA_3	0.991	0.806	0.889	0.957	0.95	0.38	0.72	0.89
XA_4	0.944	0.728	0.782	0.895	0.95	0.43	0.78	0.9

segmentation task (F1-score 0.89 and 0.35) compared to the Lab test set. One can also see that baseline without any augmentation or with too strong augmentation performers poorly.

VI. CONCLUSION

In this article, we have proposed the comprehensive framework which provides an easy, reliable, and scalable way to collect image datasets and to efficiently augment the collected data. The proposed approach is called *XtremeAugment* and consists of the hardware dataset augmentation (HDA) which performs during the data collection stage, and the object-based augmentation (OBA) which performs after image labeling.

We prove the efficiency of our method using the apple rotten segmentation problem as an example. *XtremeAugment* is designed for the few-shot computer vision problems. It is especially promising for the type of the domain adaptation tasks where the target domain is more difficult in terms of the surrounding of the objects than a source domain. In our experiment we collect images in the laboratory conditions with good illumination, uniform background, and a large distance between the apples. To test the models we use, on the contrary, images in the wild. *XtremeAugment* outperforms the baseline by a 0.62 F-score providing an accurate model with very limited training data.

The separate components of *XtremeAugment* have their limitations. HDA must be used during the data collection and requires additional control over the imaging process. OBA can be only applied if the dataset is accompanied by the instance segmentation masks. However, we also show that HDA and OBA can be applied independently and still ensure an increase in the model accuracy. Although our method requires labeled images, it allows for significant simplification of the labeling process. First, it can use images with uniform backgrounds and without overlapping that are faster to label. Also, some images obtained with HDA can share the same segmentation mask.

Along with the code for HDA and OBA, we share the collected dataset of multiview apples spoil segmentation and the dataset on various backgrounds in the wild.

In our future work, we will explore the context-based OBA approaches and vary camera and lighting parameters, and combinations of XtremeAugment with other image augmentation techniques. Another promising topic to investigate is the curriculum learning. Curriculum learning changes the 'hardness' of the samples during model training. Typically, it is difficult to mine samples of a certain "hardness"; however, the OBA algorithm allows controlling it explicitly by tuning the objects overlapping and other hyperparameters. It is also worth mentioning that the augmentation hyperparameters can be further optimized, but even with a naive setup, *XtremeAugment* proves its efficiency.

APPENDIX A
IMAGE COLLECTION SYSTEM METHODOLOGY

In this section, we explain the concepts behind the image collection system. We aim to design a system that can be easily used by a wide range of researchers to collect data and automate processes in labs. Therefore our system must meet the following requirements:

- consists of cheap equipment;
- is reliable;
- is easy-to-use.

In practice, there is a trade-off between the device cost and reliability. To have an opportunity to work with non-expensive industrial-level equipment but obtain a stable system as a whole, we adhere to the fault-tolerant design. To simplify the usage and, by implication, reduce the number of potential user-related mistakes, we use a declarative programming project design approach.

A. *FAULT-TOLERANT DESIGN*

To reach a smooth program execution and ensure its sustainability, we propose and apply the following techniques.

1) RELAUNCHING ON FAULT

There are different reasons why a system can turn off. Usually, it is due to power outages. We recommend adding an extra battery to the system circuit to address this issue, but we relaunch a project if accidental halting occurs regardless of its presence.

2) GRACEFUL DEGRADATION

Another vital concept that we use in our design is called graceful degradation: even if some parts of a system fail, the rest will operate smoothly. For example, if any single sensor stops responding, it should not affect the other ones.

3) CRITICAL ISSUES ALERTING

Even though it is preferable to continue system operation when a component does not respond, it is necessary to inform a user that something expects a fix or replacement. For this purpose, we use an alert bot.

4) DATA BACKUPPING

If the project's goal is to collect data, then data is the most valuable part of the whole system. To make sure that collected data is saved, we schedule its backup to another device.

5) CONFLICTS PREVENTION

A common issue that is difficult to debug arises when multiple scripts or pipelines attempt to access the same resource. To exclude this type of problem, we check all the device usages in *Scheduler* before running any pipelines. If any different pipelines share the same devices, a user will see an alert.

6) STATELESS CHILD NODES

A controller device makes it possible to manage the logic of the system. Sensors cannot operate without a controller. The reported system provides an opportunity to have more than one controller. It can be useful either if there are too many sensors to connect to a single controller or when sensors have to be distributed over a wide area. In such a case, one of the nodes will act as a main, and the others will be the child nodes. The child nodes are designed to be stateless, so they do not store any data or past transaction history. This interaction simplification allows the user to easily add, remove, or replace child nodes without main flow interruption.

7) LOGGING

In order to investigate incidents, we log all the actions within the system, including system start, system restart, sensor measurements status, image capturing status, every pipeline execution time, errors during execution, and others.

B. DECLARATIVE PROGRAMMING

Our system is designed according to the declarative programming paradigm and guarantees configuring of projects with different goals easily. Each project is a system in a concrete configuration that will perform the defined actions according to a schedule. Moreover, it is possible to launch multiple projects simultaneously.

The project description in the proposed system is config-based. It allows creating new projects by defining the sequence of sensors to activate and the rule to start these actions. In other words, we declare what to do instead of how to do it. This approach helps both to simplify the modifications and avoid errors and side effects, which is vital when the project complexity starts to grow. For more details, refer to Appendix A.

C. FRAMEWORK ABSTRACTION LAYERS

In this section, we present the abstraction layers of the framework from the user perspective. We aim to show what users can do without diving into implementation details and also what can be modified using every next abstraction layer. In Figure 10, the upper plane corresponds to the highest abstraction layer, and the bottom plane is for the low-level functional.

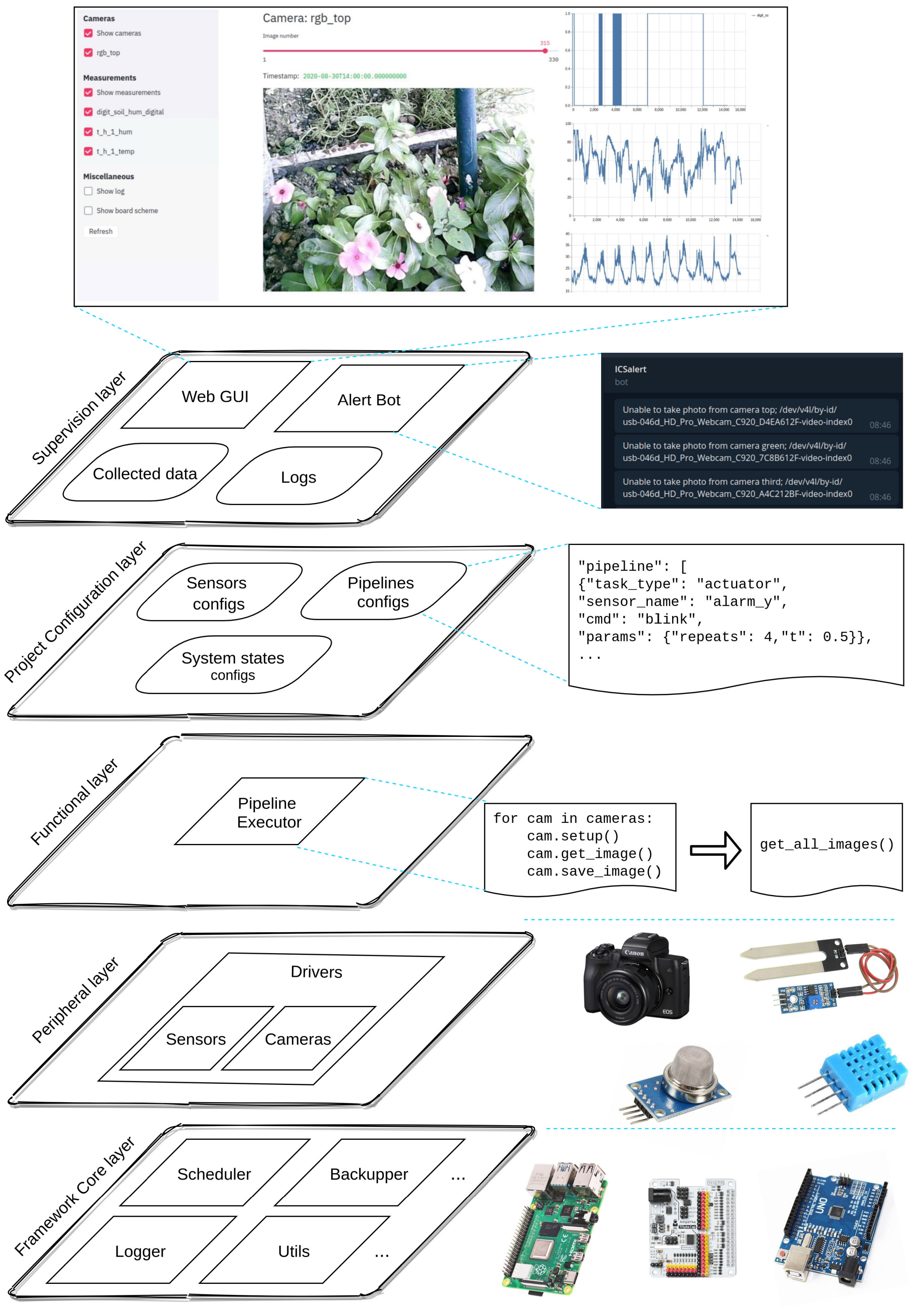
When a project is already configured, all that users have to do is to monitor its performance using the functional of *Supervision layer*. It is easy to do via a Web User Interface (UI). It allows choosing what data to monitor and interactively shows collected images, plots sensors measurements, and describes auxiliary information about project configuration. We can place a monitor directly near the system in indoor projects and view its last collected data and state. While Web GUI is an appropriate solution for the collected data control and the cameras and sensors calibration, we use an alert bot for urgent notifications caused by failures. It will send a message to a user if any check or fix is required.

To start an experiment, a typical user will need to use only the highest two abstraction layers. The *Supervision layer* described above enables continuous project monitoring, and the *Project Configuration layer* provides a straightforward way to set up a project with concrete goals and existing hardware. We propose a config-based project description that does not require any programming skills from the users for a usual project. It means that all that the user has to do is to list his hardware in sensors and camera configuration files and to list the sequence of actions. We describe it in more detail in Appendix A.

Another feature of this layer is the system state configuration. It provides an opportunity to create aggregated states and reuse them. State reusing operates in the same way as the reusing functions, but instead of code, a state can include other low-level states. For instance, a user may need to switch several actuators simultaneously many times. To avoid unnecessary code repetitions, we can aggregate all simultaneous actions in a single state and switch states in a pipeline. An aggregated state will contain instructions for all the involved devices, and all of that instructions will be executed every time when the state is called.

If the *Project Configuration layer* allows to aggregate system states, the *Functional layer* can aggregate and add new functions. Here we can define the actions that will be further used in configuration files. The basic functions needed for most of the projects are already implemented. If one needs some specific operations, it is easy to add them in the *Functional layer* having basic Python programming skills.

The *Peripheral layer* can be used to add new types of hardware. It needs a more advanced programming level; however,



Cameras
- ☑ Show cameras
- ☑ rgb_top

Measurements
- ☑ Show measurements
- ☑ digit_soil_hum_digital
- ☑ t_h_1_hum
- ☑ t_h_1_temp

Miscellaneous
- ☐ Show log
- ☐ Show board scheme

Image number: 315 (1–330)

Timestamp: 2020-08-30T14:00:00.000000000

```
Unable to take photo from camera top; /dev/v4l/by-id/usb-046d_HD_Pro_Webcam_C920_D4EA612F-video-index0 08:46
Unable to take photo from camera green; /dev/v4l/by-id/usb-046d_HD_Pro_Webcam_C920_7C8B612F-video-index0 08:46
Unable to take photo from camera third; /dev/v4l/by-id/usb-046d_HD_Pro_Webcam_C920_A4C212BF-video-index0 08:46
```

```
"pipeline": [
{"task_type": "actuator",
"sensor_name": "alarm_y",
"cmd": "blink",
"params": {"repeats": 4,"t": 0.5}},
...
```

```
for cam in cameras:
    cam.setup()
    cam.get_image()
    cam.save_image()
```

```
get_all_images()
```

FIGURE 10. Image collection system abstraction layers.

most of the hardware has some description or ready-to-use drivers.

The *Framework Core layer* consists of the modules that empower system operation. The system logic and various platform support are implemented within this layer. Usually, the users do not have to worry about this level's details and do not have to make any modifications here.

APPENDIX B
IMAGE COLLECTION SYSTEM IMPLEMENTATION

In this section, we go through the implementation details and describe the relations between system modules. The system is developed in Python 3 programming language and is suitable to use with Linux-driven devices. The working prototype is deployed on Raspberry Pi 4 single-board computer. Some of the system features, such as general-purpose input/output (GPIO) pins access, are now implemented only for the Raspberry Pi boards family and Troyka Cap extension boards.

A child node can be any device that can communicate over Wi-Fi. So, if your main node is not Raspberry and you need GPIO functionality, you can attach, for example, another Arduino or ESP266 micro-controller to the system.

The high-level system architecture is reflected in Figure 11. Rectangles in the figure represent modules; ovals represent data.

If we want to add more than one controller node, a single *main node* will send commands to all the *child nodes*. From the users' point of view, devices on *child nodes* are accessible the same as any device on the *main node*, but they are called remote devices.

D. CONFIGURATION FILES

To create a new project, we have to prepare configuration files (configs). The configs are stored as JavaScript Object Notation (JSON) files.

We distinguish three types of configs: hardware configs, state configs, and pipelines.

1) HARDWARE CONFIGS

We have separate configuration files for sensors and cameras. To add a sensor, we define its name and bind it to some pin on a board. Now the system supports Raspberry Pi and Amperka Troyka Cap boards. Also, we set a device type, and it is ready to use. To add a camera, we separately list web cameras and SLR cameras and define their unique names.

2) STATE CONFIGS

A state is a way to describe multiple simultaneous actions that do not require any feedback. For example, if we have many actuators such as lamps or water pumps, and we need to switch all of them at once, we can define one state with all the actuators switched on and another state with all the actuators switched off. Another opportunity that appears with states is access to the current state from any task. Thus, states enable pipelines to communicate to each other with the mediator instead of direct communication. In Section IV, we outline a case where we switch lighting conditions to alter the resulting images. In this case, the lighting state is known to the task that makes images, and we can add this state name to the image name or record.

To make state configuration flexible, we introduce three types of states. A *permanent state type* is assigned to the system when triggered and becomes available from any pipeline. It both switches the actuators and assigns itself to be the current system state as described above. An *interrupt state type*, in the opposite, only switches actuators without changing the current state name. A *recovery state type* is used to return the system to the latest permanent state after an interruption.

3) PIPELINES

A pipeline in our system is a logically connected sequence of actions that a system must start at some time or start with some intervals. A pipeline is also stored in a JSON file [75]. The user starts defining a pipeline by setting a rule to trigger a pipeline in cron notation [76]. It is a flexible and widespread way to schedule scripts that allow repetitive and one-time tasks to schedule. A pipeline's body consists of blocks with task type, optional device name, and optional parameters.

There are several predefined types of tasks, but users can extend them with new ones. The basic task types are:

- actuator;
- sensor;
- state switching;
- sleeping.

An actuator is any attached device that performs some action. Lamps and various motors are typical examples of actuators.

A sensor is a device that makes some measurements. In our implementation, any call to a sensor is automatically decorated for saving collected data with a timestamp. So, a user does not have to worry about data saving from each sensor separately.

Another feature that sensor abstraction provides is an automatic repetition of multiple measurements in a row with further outlier removal and averaging.

State switching means changing a state, as described in this section above. The user chooses a list of predefined states and picks one of them.

Sleeping is the most generic task type. It is used to stay idle for a certain amount of time.

An important feature is that we can activate multiple pipelines simultaneously. Each pipeline solves its independent problem, and they communicate with other pipelines via the system state.

E. SCHEDULER

The system core unit is the *Scheduler*. It is based on a standard Python APScheduler. For its functioning, *Scheduler* requires configs and pipelines. *Scheduler* sets up a project, creates all the necessary directories, checks pipelines, and launches them. A vital job of the *Scheduler* is to check that different

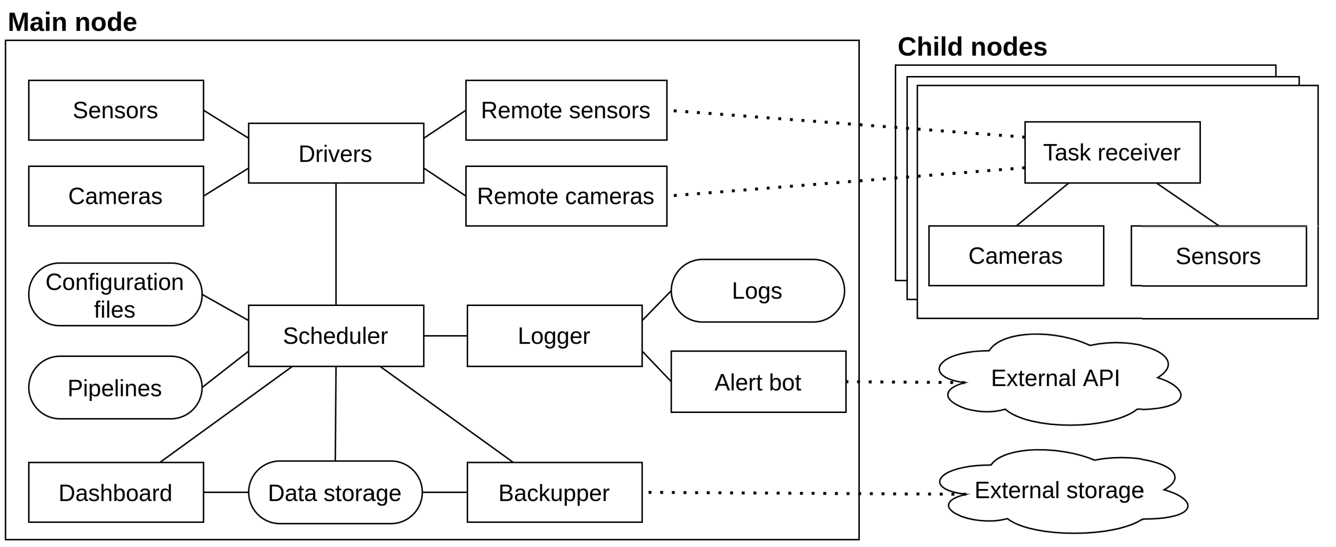


FIGURE 11. **High-level system architecture.**

pipelines do not use the same equipment to ensure that no conflicts will occur. This core module orchestrates the work of all other modules, starts and finishes the project.

An internal part of the *Scheduler* is a *Pipeline Executor*. It parses and completes concrete pipelines received from the *Scheduler*. In Figure 10, *Pipeline Executor* is referred to as a Functional layer because here, all the user functions are implemented.

F. LOGGER

The *Logger* is used to write down all system operations. The *Logger* is based on a standard Python logging module. It records actions for the system in general and each pipeline independently. Actions and errors can have different importance. Depending on the importance, log records can be handled differently. For instance, critical errors are alerted to the users by a bot, and warnings can be shown during the system initialization.

G. DRIVERS

Drivers can be perceived as adapters that enable a uniform interface for all the devices. To reach that, every sensor or camera must be inherited from the abstract base sensor or camera classes accordingly. These base classes oblige to implement necessary functions to return the measured values.

H. BACKUPPER

The goal of the *Backupper* is to send the collected data to some other cloud or local device to increase reliability by adding the redundancy.

I. DASHBOARD

The Dashboard is a Web interface that provides easy access to the collected data. To create an interactive interface, we wrote a server based on the Streamlit framework. A user can choose what data to load. Images are shown for each camera separately in different tiles. A slider near each tile allows scrolling to the requested image capturing time. The sensors' measurements are shown as plots. We can also load the project configuration details and logs to an interface.

ACKNOWLEDGMENT

The authors would like to thank the use of the Skoltech CDISE supercomputer Zhores for obtaining the results presented in this paper and Fablab of Skoltech for discussions about experiment setup. Svetlana Illarionova and Ivan Oseledets were involved in Section III, Section IV, and Section V.

REFERENCES

[1] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 558–567.
[2] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, "Rethinking pre-training and self-training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 3833–3845.
[3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.
[4] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4893–4902.
[5] A. Bulat, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "Toward fast and accurate human pose estimation via soft-gated skip connections," in *Proc. 15th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, Nov. 2020, pp. 101–108.
[6] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Proc. Sci. Inf. Conf.*, 2019, pp. 128–144.
[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[8] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 843–852.

[9] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review," *Int. J. Automat. Comput.*, vol. 14, no. 5, pp. 503–519, 2017.

[10] A. Birhane and V. U. Prabhu, "Large image datasets: A pyrrhic win for computer vision?" in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 1537–1547.

[11] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Found. Trends Comput. Graph. Vis.*, vol. 12, no. 1–3, pp. 1–308, 2020.

[12] P. Liu, "A survey of remote-sensing big data," *Frontiers Environ. Sci.*, vol. 3, p. 45, Jun. 2015.

[13] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017.

[14] Y. Lu and S. Young, "A survey of public datasets for computer vision tasks in precision agriculture," *Comput. Electron. Agricult.*, vol. 178, Nov. 2020, Art. no. 105760. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920312709

[15] S. Illarionova, A. Trekin, V. Ignatiev, and I. Oseledets, "Neural-based hierarchical approach for detailed dominant forest species classification by multispectral satellite imagery," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 1810–1820, 2021.

[16] S. Nesteruk, D. Shadrin, M. Pukalchik, A. Somov, C. Zeidler, P. Zabel, and D. Schubert, "Image compression and plants classification using machine learning in controlled-environment agriculture: Antarctic station use case," *IEEE Sensors J.*, vol. 21, no. 16, pp. 17564–17572, Aug. 2021.

[17] N. Paton, "Automating data preparation: Can we? Should we? Must we?" in *Proc. 21st Int. Workshop Design, Optim., Lang. Anal. Process. Big Data*, 2019, pp. 1–6.

[18] *Web of Science: Clarivate Analytics*. Accessed: Jun. 6, 2021. [Online]. Available: https://app.webofknowledge.com/author/search

[19] P. Goyal, M. Caron, B. Lefaudeux, M. Xu, P. Wang, V. Pai, M. Singh, V. Liptchinsky, I. Misra, A. Joulin, and P. Bojanowski, "Self-supervised pretraining of visual features in the wild," 2021, *arXiv:2103.01988*.

[20] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.

[21] S. Illarionova, D. Shadrin, A. Trekin, V. Ignatiev, and I. Oseledets, "Generation of the NIR spectral band for satellite images with convolutional neural networks," *Sensors*, vol. 21, no. 16, p. 5646, Aug. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/16/5646

[22] A. Khanna and S. Kaur, "Evolution of Internet of Things (IoT) and its significant impact in the field of precision agriculture," *Comput. Electron. Agricult.*, vol. 157, pp. 218–231, Feb. 2019.

[23] S. Nesteruk, D. Shadrin, V. Kovalenko, A. Rodríguez-Sánchez, and A. Somov, "Plant growth prediction through intelligent embedded sensing," in *Proc. IEEE 29th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2020, pp. 411–416.

[24] S. Antoy and M. Hanus, "Declarative programming with function patterns," in *Proc. Int. Symp. Log.-Based Program Synth. Transf.* Berlin, Germany: Springer, Sep. 2005, pp. 6–22. [Online]. Available: https://link.springer.com/chapter/10.1007/11680093_2

[25] S. Nesteruk, D. Shadrin, and M. Pukalchik, "Image augmentation for multitask few-shot learning: Agricultural domain use-case," 2021, *arXiv:2102.12295*.

[26] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.

[27] S. Zhou, J. Zhang, H. Jiang, T. Lundh, and A. Y. Ng, "Data augmentation with Mobius transformations," *Mach. Learn., Sci. Technol.*, vol. 2, no. 2, Jun. 2021, Art. no. 025016.

[28] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour, and E. M. Aggoune, "Internet-of-Things (IoT)-based smart agriculture: Toward making the fields talk," *IEEE Access*, vol. 7, pp. 129551–129583, 2019.

[29] S. Leminen Madsen, S. K. Mathiassen, M. Dyrmann, M. S. Laursen, L.-C. Paz, and R. N. Jørgensen, "Open plant phenotype database of common weeds in Denmark," *Remote Sens.*, vol. 12, no. 8, p. 1246, Apr. 2020.

[30] Y.-Y. Zheng, J.-L. Kong, X.-B. Jin, X.-Y. Wang, and M. Zuo, "CropDeep: The crop vision dataset for deep-learning-based classification and detection in precision agriculture," *Sensors*, vol. 19, no. 5, p. 1058, Mar. 2019.

[31] T. M. Giselsson, R. N. Jørgensen, P. K. Jensen, M. Dyrmann, and H. S. Midtiby, "A public image database for benchmark of plant seedling classification algorithms," 2017, *arXiv:1711.05458*.

[32] A. Olsen, D. A. Konovalov, B. Philippa, P. Ridd, J. C. Wood, J. Johns, W. Banks, B. Girgenti, O. Kenny, J. Whinney, B. Calvert, M. R. Azghadi, and R. D. White, "DeepWeeds: A multiclass weed species image dataset for deep learning," *Sci. Rep.*, vol. 9, no. 1, pp. 1–12, Dec. 2019.

[33] J. Gené-Mola, E. Gregorio, F. A. Cheein, J. Guevara, J. Llorens, R. Sanz-Cortiella, A. Escolà, and J. R. Rosell-Polo, "LFuji-air dataset: Annotated 3D LiDAR point clouds of Fuji apple trees for fruit detection scanned under different forced air flow conditions," *Data Brief*, vol. 29, Apr. 2020, Art. no. 105248.

[34] K. Kusumam, T. Krajník, S. Pearson, G. Cielniak, and T. Duckett, "Can you pick a broccoli? 3D-vision based detection and localisation of broccoli heads in the field," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 646–651.

[35] S. Skovsen, M. Dyrmann, A. K. Mortensen, M. S. Laursen, R. Gislum, J. Eriksen, S. Farkhani, H. Karstoft, and R. N. Jorgensen, "The GrassClover image dataset for semantic and hierarchical species understanding in agriculture," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–9.

[36] P. A. Dias, A. Tabb, and H. Medeiros, "Multispecies fruit flower detection using a refined semantic segmentation network," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3003–3010, Oct. 2018.

[37] S. Haug and J. Ostermann, "A crop/weed field image dataset for the evaluation of computer vision based precision agriculture tasks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 105–116.

[38] P. Lottes, J. Behley, N. Chebrolu, A. Milioto, and C. Stachniss, "Joint stem detection and crop-weed classification for plant-specific treatment in precision farming," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 8233–8238.

[39] N. Chebrolu, P. Lottes, A. Schaefer, W. Winterhalter, W. Burgard, and C. Stachniss, "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1045–1052, Sep. 2017.

[40] A. Bender, B. Whelan, and S. Sukkarieh, "A high-resolution, multimodal data set for agricultural robotics: A Ladybird's-eye view of Brassica," *J. Field Robot.*, vol. 37, no. 1, pp. 73–96, Jan. 2020.

[41] X.-S. Wei, Q. Cui, L. Yang, P. Wang, and L. Liu, "RPC: A large-scale retail product checkout dataset," 2019, *arXiv:1901.07249*.

[42] C.-J. Lin, Y.-Y. Chen, C.-F. Pan, V. Wu, and C.-J. Wu, "Dataset supporting blood pressure prediction for the management of chronic hemodialysis," *Sci. Data*, vol. 6, no. 1, pp. 1–7, Dec. 2019.

[43] D. Murray, L. Stankovic, and V. Stankovic, "An electrical load measurements dataset of united kingdom households from a two-year longitudinal study," *Sci. Data*, vol. 4, no. 1, pp. 1–12, Dec. 2017.

[44] R. Medico, L. De Baets, J. Gao, S. Giri, E. Kara, T. Dhaene, C. Develder, M. Bergés, and D. Deschrijver, "A voltage and current measurement dataset for plug load appliance identification in households," *Sci. Data*, vol. 7, no. 1, pp. 1–10, Dec. 2020.

[45] J. Lee, S. Noh, H.-J. Kim, and Y.-S. Kang, "Implementation of cyber-physical production systems for quality prediction and operation control in metal casting," *Sensors*, vol. 18, no. 5, p. 1428, May 2018.

[46] E. Mwebaze, T. Gebru, A. Frome, S. Nsumba, and J. Tusubira, "ICassava 2019 fine-grained visual categorization challenge," 2019, *arXiv:1908.02900*.

[47] S. Illarionova, S. Nesteruk, D. Shadrin, V. Ignatiev, M. Pukalchik, and I. Oseledets, "MixChannel: Advanced augmentation for multispectral satellite images," *Remote Sens.*, vol. 13, no. 11, p. 2181, Jun. 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/11/2181

[48] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.

[49] V. Khrulkov, A. Babenko, and I. Oseledets, "Functional space analysis of local GAN convergence," 2021, *arXiv:2102.04448*.

[50] S. Chen, E. Dobriban, and J. H. Lee, "A group-theoretic framework for data augmentation," *J. Mach. Learn. Res.*, vol. 21, no. 245, pp. 1–71, 2020.

[51] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," 2018, *arXiv:1809.06839*.

[52] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, Dec. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218310749

[53] S. Liu, J. Zhang, Y. Chen, Y. Liu, Z. Qin, and T. Wan, "Pixel level data augmentation for semantic image segmentation using generative adversarial networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 1902–1906.

[54] R. Barth, J. IJsselmuiden, J. Hemming, and E. J. V. Henten, "Data synthesis methods for semantic segmentation in agriculture: A capsicum annuum dataset," *Comput. Electron. Agricult.*, vol. 144, pp. 284–296, Jan. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169917305689

[55] T.-H. Cheung and D.-Y. Yeung, "MODALS: Modality-agnostic automated data augmentation in the latent space," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–10. [Online]. Available: https://openreview.net/forum?id=XjYgR6gbCEc

[56] A. Zuniga, P. Nurmi, and H. Flores, "Ripe or rotten? Low-cost produce quality estimation using reflective green light sensing," *IEEE Pervasive Comput.*, vol. 20, no. 3, pp. 60–67, May 2021.

[57] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," 2020, pp. 1–11, *arXiv:1811.10959*.

[58] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6022–6031.

[59] S. Illarionova, S. Nesteruk, D. Shadrin, V. Ignatiev, M. Pukalchik, and I. Oseledets, "Object-based augmentation for building semantic segmentation: Ventura and santa Rosa case study," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 1659–1668.

[60] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1310–1319.

[61] N. Dvornik, J. Mairal, and C. Schmid, "Modeling visual context is key to augmenting object detection datasets," in *Computer Vision—ECCV*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 375–391.

[62] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," 2020, *arXiv:2012.07177*.

[63] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le, "Learning data augmentation strategies for object detection," in *Computer Vision—ECCV*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer, 2020, pp. 566–583.

[64] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "Imbalance problems in object detection: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3388–3415, Oct. 2021.

[65] Z. He, L. Xie, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Data augmentation revisited: Rethinking the distribution gap between clean and augmented data," 2019, pp. 1–10, *arXiv:1909.09148*.

[66] A. Sinha, K. Ayush, J. Song, B. Uzkent, H. Jin, and S. Ermon, "Negative data augmentation," 2021, pp. 1–17, *arXiv:2102.05113*.

[67] B. Arad, P. Kurtser, E. Barnea, B. Harel, Y. Edan, and O. Ben-Shahar, "Controlled lighting and illumination-independent target detection for real-time cost-efficient Applications. The case study of sweet pepper robotic harvesting," *Sensors*, vol. 19, no. 6, p. 1390, Mar. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/6/1390

[68] A. Silwal, T. Parhar, F. Yandún, and G. Kantor, "A robust illumination-invariant camera system for agricultural applications," 2021, *arXiv:2101.02190*.

[69] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1–8.

[70] S. Choo, S. J. Ha, and J. Lee, "Semantic-preserving metric learning for video-text retrieval," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 2388–2392.

[71] Y. Cao, M. Long, J. Wang, and S. Liu, "Deep visual-semantic quantization for efficient image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1–8.

[72] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, Atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[74] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *Proc. Eur. Conf. Inf. Retr.* Berlin, Germany: Springer, pp. 345–359, 2005.

[75] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, "Foundations of JSON schema," in *Proc. 25th Int. Conf. World Wide Web*, Apr. 2016, pp. 263–273.

[76] L. Reznick, "Using cron and crontab," *Syst. Admin*, vol. 2, no. 4, pp. 29–32, 1993.

SERGEY NESTERUK received the B.S. and M.S. degrees in information security from the Saint Petersburg University of Aerospace Instrumentation, in 2018 and 2020, respectively, and the M.S. degree in information science and technology from the Skolkovo Institute of Science and Technology (Skoltech), Russia, in 2020, where he is currently pursuing the Ph.D. degree. His research are related to monitoring systems and applying machine learning methods to the collected data. He is involved in the development of the Precision Agriculture Laboratory, Skoltech, and is responsible for the development of greenhouse image collecting systems, the development of image augmentation framework, and computer vision research.

SVETLANA ILLARIONOVA received the bachelor's and master's degrees in computer science from Lomonosov Moscow State University, Moscow, Russia, in 2017 and 2019, respectively. She is currently pursuing the Ph.D. degree in computer science with the Skolkovo Institute of Science and Technology, Moscow. Her research interests include computer vision, deep neural networks, and remote sensing.

TIMUR AKHTYAMOV received the B.S. degree from Bauman Moscow State Technical University, in 2019, and the master's degree from the Skolkovo Institute of Science and Technology (Skoltech), in 2021, where he is currently pursuing the Ph.D. degree. His current research interest includes the application of reinforcement learning in mobile robotics. He is also involved in projects related to the Internet of Things, computer vision, and underwater robotics.

DMITRII SHADRIN received the M.S. degree in applied physics and mathematics from the Moscow Institute of Physics and Technology (MIPT), in 2016, and the Ph.D. degree in data science from the Skolkovo Institute of Science and Technology (Skoltech), Russia, in 2020. He is currently a Research Scientist with Skoltech. His research interests include data processing, modeling of physical and bioprocesses in closed artificial growing systems, machine learning, and computer vision. He is actively participating in the creation of the computer vision group for solving problems in digital agriculture and remote sensing.

ANDREY SOMOV received the Graduate degree from MATI–Russian State Technological University, Russia, in 2004, the Diploma degree in electronics engineering from MATI–Russian State Technological University, Russia, in 2006, and the Ph.D. degree from the University of Trento, Italy, in 2009, for work in the field of power management in wireless sensor networks (WSN). He is currently an Assistant Professor with the Skolkovo Institute of Science and Technology (Skoltech), Russia. Before joining Skoltech, in 2017, he was worked as a Senior Researcher with the FBK CREATE-NET Research Center, Italy, from 2010 to 2015, and a Research Fellow with the University of Exeter, U.K., from 2016 to 2017. He has published more than 80 papers in peer-reviewed international journals and conference proceedings. His current research interests include intelligent sensing, machine learning, cognitive IoT, and associated proof-of-concept implementation. He holds some awards in the fields of WSN and the IoT, including the Google IoT Technology Research Award, in 2016, and the Best Paper Award at IEEE IoP Conference, in 2019.

MARIIA PUKALCHIK received the M.Sc. degree in analytical chemistry from Vyatka State University, Russia, in 2009, and the Ph.D. degree in ecology from Lomonosov Moscow State University (MSU), Russia, in 2013. In 2017, she joined Skoltech, where she is currently an Assistant Professor. She has authored 20 refereed articles and two books.

IVAN OSELEDETS received the Graduate degree from the Moscow Institute of Physics and Technology, in 2006, and the Candidate of Sciences and D.Sc. degrees from the Marchuk Institute of Numerical Mathematics of Russian Academy of Sciences, in 2007 and 2012, respectively. He is currently the Director of the Center for Artificial Intelligence Technology, Skoltech, where he has been working, since 2013. His research covers a broad range of topics. He proposed a new decomposition of high-dimensional arrays (tensors)—tensor-train decomposition and developed many efficient algorithms for solving high-dimensional problems. His current research interests include the development of new algorithms in machine learning and artificial intelligence, such as the construction of adversarial examples, theory of generative adversarial networks, and compression of neural networks. It resulted in publications in top computer science conferences, such as ICML, NIPS, ICLR, CVPR, RecSys, ACL, and ICDM. He is also an Associate Editor of *SIAM Journal on Mathematics in Data Science*, *SIAM Journal on Scientific Computing*, and *Journal of Advances in Computational Mathematics* (Springer).