IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 8, 2022, accepted February 14, 2022, date of publication February 16, 2022, date of current version March 1, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3152224

Cross Validation Voting for Improving CNN Classification in Grocery Products

JAIME DUQUE DOMINGO, ROBERTO MEDINA APARICIO, AND LUIS MIGUEL GONZÁLEZ RODRIGO

CARTIF Foundation, División de Sistemas Industriales y Digitales, Parque Tecnológico de Boecillo, 47151 Valladolid, Spain

Corresponding author: Jaime Duque Domingo (jaiduq@cartif.es)

This work was supported in part by "The Centre for the Development of Industrial Technology (CDTI)", under the "Centros Tecnológicos de Excelencia Cervera" Program, through the Project "5R-Cervera Network in Robotic Technologies for Smart Manufacturing", under Contract CER-20211007; and in part by the "Fondo Europeo de Desarrollo Regional (FEDER)" of the European Union and the "Junta de Castilla y León" through the "Instituto para la Competitividad Empresarial de Castilla y León (ICE)" through the line "2020 Proyectos I+D Orientados a la Excelencia y Mejora Competitiva de los Centros Tecnológicos", under Project CCTT3/20/VA/0003.

ABSTRACT The development of deep neural networks that has been carried out in recent years allows solving highly complex computer vision classification problems. Often, although the results obtained with these classifiers are high, there are certain sectors that seek greater accuracy from these systems. Increasing the accuracy of neural networks can be achieved through ensemble learning, which combines different classifiers with the aim of selecting a winner based on different criteria about them. These techniques have traditionally shown good results although they involve training models of different nature and can even produce an overfitting with respect to the training data, so datasets must be chosen to correctly evaluate the result. In this paper, a Cross-Validation-Voting (CVV) technique for grocery product classification is presented. This technique improves several single state-of-the-art classifiers without combining different ones and avoids the problems of overfitting with respect to the training set. The single classifiers are trained multiple times against distributed sets to show how the results obtained to date from the classification of a well-known dataset are improved. In this dataset, an extensive test set was previously selected by the authors to show comparable results with other papers in the literature. The technique is valid not only for vision nets and can be used to solve numerous problems with different kinds of neural networks and classifiers.

INDEX TERMS Cross-validation-voting, CVV, voting, boosting, ResNeXt, EfficientNet, Wide ResNet, CNN, grocery image classification.

I. INTRODUCTION

In recent years, there has been a significant development in the field of deep learning. Advances in hardware have made it possible to train complex models, such as neural networks with hundreds of hidden layers. These models are capable of learning to solve numerous classification problems but may have overfitting problems when the training data set is not sufficiently well structured and sized. The models are so sophisticated that they can achieve very high accuracy against training data but may not respond well to new test data.

In many classical methods, such as Support Vector Machines (SVMs) [1], models are trained against a dataset although no validation data are used to perform an early stopping in a simple way. These methods are usually evaluated against a test set or by cross-validation. A grid search is usually used to find the optimal model parameters, selecting the ones that give the best results in the cross-validation.

Neural networks require the division of data into three sets. A training set allows the model weights to be configured over different epochs. A validation set allows the model to be evaluated every few epochs, detecting the best performer on the validation data and stopping the training. In this way, although the model is trained with training data, we can keep the one that performs best with data not used directly in the adjustment of weights. This allows the model to generalize better to new cases. Finally, the model is evaluated with a test set. This step is necessary because when we stop the training depending on the validation evaluation, we can make the model tend to behave favorably to validation. For

The associate editor coordinating the review of this manuscript and approving it for publication was Felix Albu.

an overall evaluation of the model, some authors often use cross-validation to test how it behaves on different choices of dataset elements.

The early stopping approach followed to train neural networks makes the model favorable to the validation data, something known as validation overfitting. Although the evaluation of the model with respect to the test set is realistic, being totally independent unknown data, it may not be the best generalizing model. In this article, we propose to use a technique called Cross-Validation-Voting (CVV) that allows better generalization to unknown cases. To do so, we use the models trained by CVV in an ensemble voting technique. This technique allows each of the models participating in the voting to be trained with a different training and validation set. It allows that, although each selected model is favorable to a different validation set, as these sets are different, the result is a model that generalizes better to test.

To test the CVV method, we have used a well-known dataset of supermarket products. In the Grocery Store Dataset [2], the authors established a prior separation of training and test data, which allows realistic comparisons between different authors. Our technique has shown that, with a single classifier, it is possible to outperform the best results obtained to date. When we ensemble different models using CVV, the results improve even more. In addition to the tests using only neural networks, we also present results using other techniques, such as boosting with trees or SVM classifiers. Another important aspect to note about our technique is that individual classifiers in the CVV model can be trained for fewer epochs than a single classifier, so the training time is not substantially altered.

Although cross-validation and ensemble model techniques are well known and have been widely used, to our knowledge, no other paper has combined them to improve model generalization. More specifically, no similar work has been carried out in the agrifood sector [9]. There are authors who have used both techniques, such as in [3] or [4]. However, in both cases, they have created an ensemble model that has been evaluated using cross-validation. In these works, classifiers of different nature have been used. The main difference with our approach is that we integrate the two techniques to train the model, and the evaluation of the model is carried out with a fully independent test set. In addition, our method works with a single type of classifier. These previously mentioned methods together with other related works have been evaluated and compared with our method, showing how it achieves a remarkable improvement in the results.

The present paper is structured as follows: Section II explores the state-of-art of the technologies considered in this paper. Section III describes the procedure that has been carried out. In Section IV, the different experiments and results obtained with the proposed method are reported. An overall discussion on the obtained results is set out. Finally, Section V notes the advantages and limitations of the presented system and suggests future developments.

II. OVERVIEW OF RELATED WORK

One of the common problems in the neural network training is the lack of generalization capability [5]. This may be due to overfitting with respect to the training data but also to overfitting with respect to the validation data. Neural networks are usually trained in epochs with respect to a set of training data, periodically evaluating the behavior of the network on a validation set and detecting a worsening of certain metrics on the validation. This mechanism, called early-stopping, makes it possible to choose the model that best fits the validation data. The model is then evaluated against a test set. However, as the model chosen is the one that performed best with the validation data, these data indirectly influence the model, so that we may be faced with an overfitting respect to validation. Although the model may obtain good results with test, it is probably not optimal due to such a tendency to a specific validation.

The cross-validation mechanism [6] adds an additional step to the training mechanism. Traditionally used with SVM-type classifiers [1], it divides the training set into k validation slots and trains k models using the training data that do not belong to their respective slots. Then, the models are evaluated with respect to their particular validation slot and the average accuracy of the k models is calculated. In a grid search of the model hyperparameters, e.g. C, gamma or the kernel in SVM, a cross-validation is performed for each combination of the parameters in the grid. Finally, the parameters that best respond to the cross-validation are selected and a new model is trained with all data from the training set.

In the case of neural networks, cross-validation is mainly used as a method of model evaluation. It has been used for the evaluation of very different problems, such as medical applications [7] or the classification of grocery products [8]. It is important to mention that most of the methods used in the grocery product classification problem only use one training and validation set but not three sets (training, validation and test) [9]. The cross-validation evaluation allows selecting certain hyperparameters to improve the model. Most of the literature applying cross-validation is limited to using this method to provide an evaluation of the proposed architectures. However, in a real scenario, the selected model must be usable. When the hyperparameters have been chosen, there are different possibilities to use the model: we can select one of the k models, usually the one that offers the best results; we can also re-train the model with certain validation data randomly selected from the original data; or we can directly train the model with all data and stop at a certain number of epochs estimated during the training of the different k models. This last case is valid considering that an early-stopping mechanism has not been used. Early stopping is not commonly used for cross validation except for the training of the final model. In our article, the cross-validation mechanism allows the model to be trained using all data from the training set but maintaining the generalization by using different validations to choose the optimal parameters.

This avoids choosing parameters that overfit the model with respect to a particular validation and respond to different types of data.

On the other hand, the techniques known as *ensemble learning* [10]–[12] allow training different classifiers with the same dataset and combine them. Thus, for example, we could combine SVM, decision trees or neural networks. There are four types of ensemble methods: voting, bagging, boosting and stacking. In most situations, these techniques have been shown to improve the performance of individual classifiers.

Voting [13] can be carried out in two ways in classification problems. In *hard* voting, each model produces a vote for a class. As a final prediction, the class voted by the majority of the models is chosen. On the other hand, in *soft* voting, probabilities are used. If a model is not totally sure of a class but that class is a winner, for example with a probability of 0.6, instead of a vote for that class, its probability is taken into account. This allows the model to value more highly those outcomes of which it is truly certain. This is recommended for an ensemble of well-calibrated classifiers. The voting technique has been widely used in conjunction with Convolutional Neural Networks (CNN) for a variety of problems: signal modulation [14], coronavirus diagnosis [15], human action classification [16], multimodal emotion recognition [17] and even detection of the camera used in forensic imaging [18]. Regarding groceries, a systematic investigation on end-to-end deep recognition of grocery products using voting was presented in [19]. The authors presented the integration of different CNN classifiers using voting, showing the effectiveness of the method.

In bagging [20], several models, similar or different, are trained with a subset of the original data usually chosen randomly. This data may be repeated among the different trained models. Once all the models have been trained, they are all combined using soft or hard voting techniques. Random forest is a classifier based on this method.

Boosting techniques [21] aim at repeatedly training a model by correcting the errors of the previously trained models. To do this, a model is trained with some samples. The model is then re-trained with the same samples but assigned a weight depending on whether it was correct in the previous step or not. At the end of training, the models are combined by weighting them in a certain way. One of the most widely used classical methods has been AdaBoost [22], method also used in grocery product recognition in conjunction with CNN networks [23]. However, over the last few years, new boosting techniques have been developed and applied in this line, such as XGBoost [24], CatBoost [25] or LightGBM [26].

Finally, the stacking problem [27] consists of stacking the output of one or more models on others, which in turn can be stacked on other models. An example of this technique was used in [28] for the grocery classification problem. In [29], the authors conducted a multistage training procedure, in which they first trained with a large class-level dataset with a single view image per category, followed by an auxiliary dataset of multiple views, which allowed the model to be robust to viewpoint changes. Finally, they trained with the objects they wanted to recognize from a single image.

Grocery product recognition offers many applications, such as the control of eating habits [30]. Over the past few years, several datasets have been created for grocery store products, such as the Grocery Store Dataset [2], the MVTec D2S dataset [31], the Retail Product Checkout dataset (RPC) [32] or the Freiburg groceries dataset [33]. Among them, the MVTec D2S, RPC and Freiburg datasets focus on the problem of object detection rather than classification. The Grocery Store Dataset [2] contains image data of grocery items categorized into fine and coarse classes. It consists of 5,125 images of 81 different types of fruits, vegetables, and carton items (e.g., juice, milk or yogurt). All images were taken with a smartphone in different grocery stores. There are 81 fine classes, grouped into 43 coarse categories. As an example, fine classes *Royal Gala* and *Granny Smith* belong to the same coarse class *Apple*. The authors separated the test set so that it is possible to correctly compare different models and architectures. A classification baseline was also provided, where the authors connected the CNN-feature vector before the classification layer to an SVM classifier. Testing different CNNs: AlexNet [34], VGG16 [35] or DenseNet [36], the authors obtained 72.5% test accuracy, directly using a model trained with ImageNet [37], and 85% test accuracy performing fine-tuning. In these winning cases, the authors used a DenseNet-169. They also evaluated a DenseNet-169 without SVM, obtaining 84% test accuracy. In addition to the data, this dataset provides iconic images which represent the product taken in controlled lighting conditions and without the supermarket background. The authors of [19] have obtained the best classification results for this dataset to date, using voting techniques on different CNNs. In the following, we will show how the CVV technique can improve their results with a single classifier.

III. ANALYSIS OF THE SYSTEM

In our training method, we wanted to improve the current training results of a known dataset of grocery products. Traditional bagging methods select random groups of data to train different models. Each selected set of items includes different samples. Bagging methods usually do not take into account validation sets, something that has gained special interest in neural networks, where early-stopping mechanisms allow stopping training before overfitting occurs. In the CVV method, the training dataset is distributed in different training and validation slots so that we use all the data for training the ensemble model. Each individual model uses a part of the general training dataset as training and another part as validation. In this way, we do not lose samples during the training process. Although the method is valid for different types of classifiers, it has been evaluated with different current neural networks, SVM and boosting methods.

Figure 1 shows the approach taken to solve this problem. In the CVV approach, the data are previously randomized and

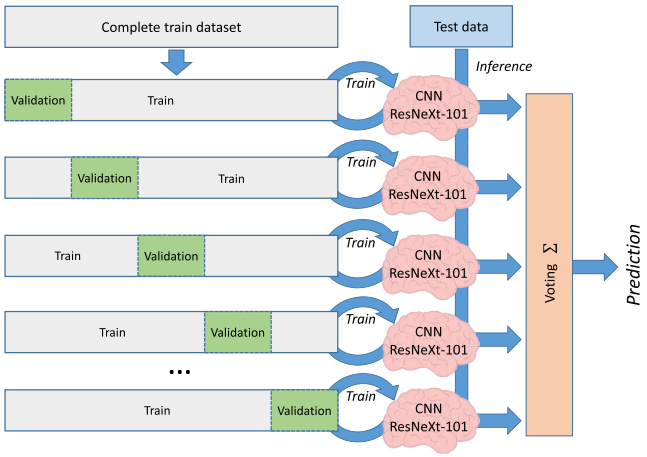


FIGURE 1. Scheme of CVV with ResNeXt-101.

k different validation slots are selected. The rest of the data from each slot are used for training.

Let τ be the set that includes all the samples of the complete training dataset. Be T_i and V_i the training and validation sets corresponding to the slot i. These sets must verify the equations 1 to 6.

$$\tau = \bigcup_{i=1}^{k} T_i \tag{1}$$

$$\bigcap_{i=1}^{k} T_i = \varnothing \tag{2}$$

$$\tau = \bigcup_{i=1}^{k} V_i \tag{3}$$

$$\bigcap_{i=1}^{k} V_i = \varnothing \tag{4}$$

$$[\tau = T_i \cup V_i] \forall i \in [1, k] \tag{5}$$

$$[T_i \cap V_i = \varnothing] \forall i \in [1, k] \tag{6}$$

Therefore, the elements of a training slot i are those used by all the other slots in validation, as shown in 7.

$$\left[T_i = \bigcup_{j=1}^{k} V_{j:j \neq i} \right] \forall i \in [1, k] \tag{7}$$

In the same way, the elements of a validation slot i are the intersection of all other training sets, as shown in 8.

$$\left[V_i = \bigcap_{j=1}^{k} T_{j:j \neq i} \right] \forall i \in [1, k] \tag{8}$$

Then, k models of the same nature are trained with each of the training and validation slots. Finally, prediction is carried out using voting techniques on the selected models. Our idea was to distribute the data into different validation slots, as cross-validation does. This allowed us to train the model with different training sets but, more importantly, to allow the model to be able to generalize to different situations. Once the models were trained with each set of validation and training slots, all methods were combined using classical voting techniques.

Let k be the number of classifiers, each one associated with its respective validation slot. For an input sample, x, $\overrightarrow{p_i(x)}$ is the vector of output probabilities given by classifier i. This vector is composed of the probabilities, $p_{ij}(x)$, which represent that a sample x belongs to class j according to classifier i. $\overrightarrow{c}$ represents the vector of labels, one for each possible class: $\{c_1, c_2, \ldots, c_N\}$. In (9), soft voting is obtained by accumulating the output probabilities of each class j. w_i is a weight associated with each classifier i, in our case $\frac{1}{k}$. The *argmax* function returns the position of the class with the highest cumulative probability.

$$S(x) = c_{arg_j max \sum_{i=1}^{k} w_i \cdot p_{ij}(x)} \tag{9}$$

Hard voting requires prior binarization of the probability, as shown in (10). For this purpose, we set only the class with the highest probability to 1 and the rest to 0.

$$b_{ij}(x) = \begin{cases} 1 \text{ if } p_{ij}(x) = max(\overrightarrow{p_i(x)}) \\ 0 \text{ otherwise} \end{cases} \tag{10}$$

In (11), the output is obtained by accumulating the binary values of each class j. As in soft voting, w_i is $\frac{1}{k}$ in our case.

$$H(x) = c_{arg_j max \sum_{i=1}^{k} w_i \cdot b_{ij}(x)} \tag{11}$$

The CVV approach can also be used with classifiers of different nature. Figure 2 shows how three different classifiers can be integrated: ResNeXt-101, EfficientNet B7 and Wide ResNet-101. In this case, the data partitioning into k slots is carried out in the same way as in the case of a single classifier. Each classifier of a different nature is trained with the similar slots used with the others, taking advantage of the goodness that each model offers against the same data partitioning. Finally, all classifiers are combined using voting techniques.

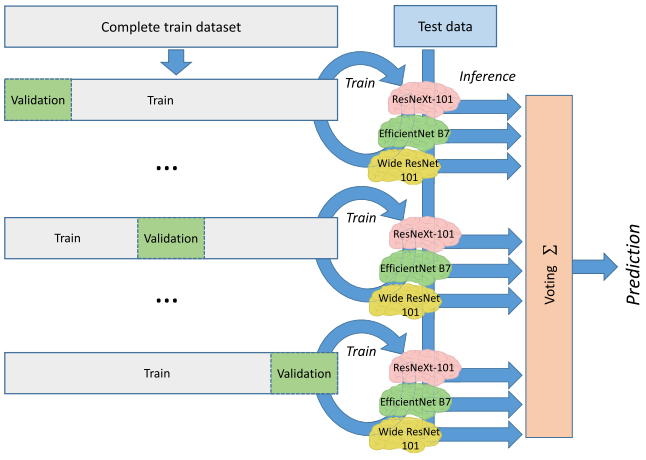


FIGURE 2. Scheme of CVV with ResNeXt-101, Efficient-B7 and WideResNet-101.

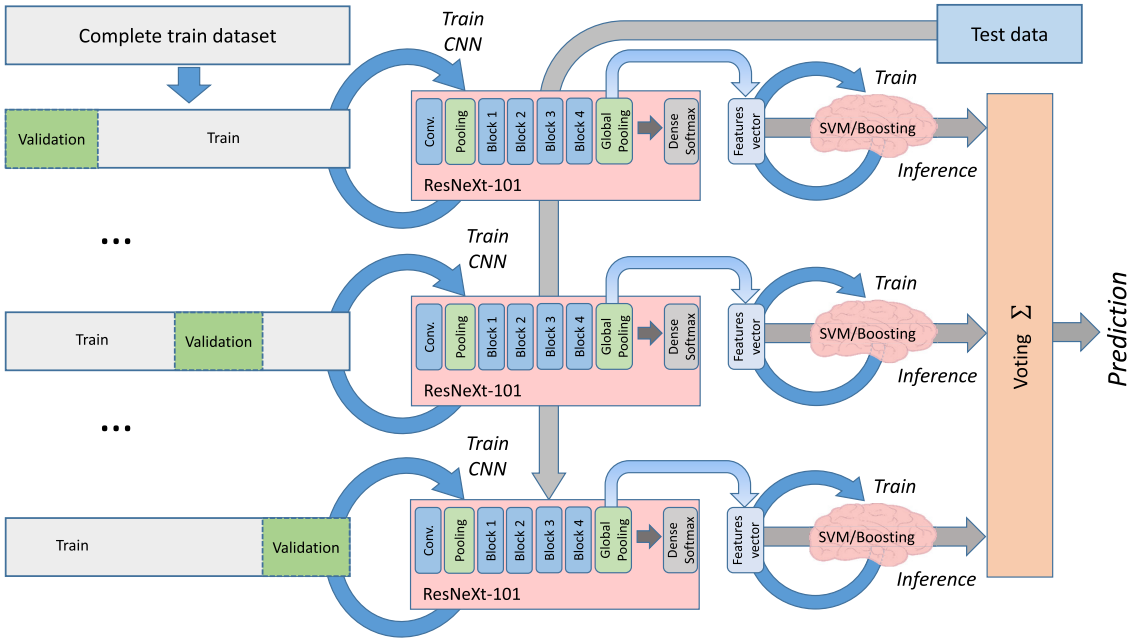


FIGURE 3. Scheme of CVV combining CNN and SVM or boosting type classifiers.

In terms of modern neural network-based classification techniques, deep learning has brought considerable progress to image classification problems using computer vision. Deep neural networks use many successive convolutional layers that attempt to capture the salient elements of images, from the most general to the most specific. Until a few years ago, most CNN networks used for food classification, such as AlexNet [34] or VGG [35], had problems with gradient fading. The accuracy started to saturate at a certain point and eventually decreased. In addition, the model did not converge because the gradients disappeared. These problems were partially solved by using residual blocks, which connect the input of a block to the output of that block through aggregation. Networks such as Inception [38] or ResNet [39] began to be used, models that are still widely used today. Another problem with deep convolutional networks was the rapidly increasing number of parameters as the number of layers increased. The ResNet architecture included residual bottleneck blocks. This model was a variant of the residual block that used 1×1 convolutions to create a bottleneck. These bottleneck blocks reduce the number of parameters and matrix multiplications without noticeably changing the result. The idea was to make the residual blocks as thin as possible to increase depth and have fewer parameters.

Among the networks used in our experiments we can find ResNeXt-101 [40], EfficientNet-B7 [41] or Wide Residual Networks (WRNs) [42]. The ResNeXt-101 [40] model is based on a ResNet model, but replaces the 3×3 convolutions within the ResNet model with clustered 3×3 convolutions. The ResNeXt bottleneck block divides a single convolution into multiple smaller parallel convolutions. A notable difference from ResNet is that ResNeXt uses aggregation instead of concatenation in the original Inception-ResNet block. This state-of-the-art network, as well as the two presented below, has been considered in our experiments as it generally gives better results than previous CNNs.

EfficientNet-B0 [41] is a novel network that seeks a balance between the number of parameters and accuracy. For this purpose, they defined this model by leveraging a multi-objective neural architecture search that optimized both accuracy and FLOPS, similarly to MNAS-Net [43]. EfficientNet-B7 [41], scaled from EfficientNet-B0, is based on a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient.

Wide Residual Networks (WRNs) [42] consider the problem that each fraction of a percent of improved accuracy costs nearly doubling the number of layers. In addition, the training of very deep residual networks is very slow because they have a problem of diminishing feature reuse. To solve this problem, the authors proposed a novel architecture where they decreased depth and increased width of residual networks.

Another approach we have analyzed consists of using the CVV model together with the features obtained by means of a CNN and different types of classifiers, such as SVM or boosting-based classifiers. For this method, we start from the same previously split dataset and train each of the CNN estimators, such as ResNeXt-101. Then, the features of the model are extracted for each estimator and for each dataset, and another SVM or boosting-type classifier is trained. Figure 3 shows the process. The blue and gray arrows show training and inference, respectively. During inference, the test data is passed through the CNN to obtain its features and subsequently passed through the SVM or boosting model to obtain

TABLE 1. Improvement of ResNeXt-101 with CVV (% except for loss).

Model \ Metric	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy	Balanced test accuracy	Precision	Recall	F1-Score
Model with a single ResNeXt-101	0.9929	0.0560	0.9848	0.0657	0.9040	0.9112	0.9210	0.9110	0.9160
Hard CVV with ensemble of 5 ResNeXt-101 classifiers (ours)					0.9352	0.9446	0.9490	0.9450	0.9470
Soft CVV with ensemble of 5 ResNeXt-101 classifiers (ours)					0.9368	0.9472	0.9510	0.9470	0.9490

TABLE 2. Improvement of EfficientNet-B7 with CVV (% except for loss).

Model \ Metric	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy	Balanced test accuracy	Precision	Recall	F1-Score
Model with a single EfficientNet-B7	0.9962	0.0377	0.9981	0.0233	0.9070	0.9243	0.9210	0.9240	0.9230
Hard CVV with ensemble of 5 EfficientNet B7 classifiers (ours)					0.9252	0.9383	0.9360	0.9380	0.9370
Soft CVV with ensemble of 5 EfficientNet B7 classifiers (ours)					0.9320	0.9447	0.9430	0.9450	0.9440

the prediction of the k estimator. Finally, voting is applied as in the previous cases to take advantage of all classifiers. This technique seeks to see if methods such as SVM or boosting allow to better refining the classification results.

Deep neural networks often have generalization problems if they have not been properly trained. Although ResNeXt-101 is one of the models with the best generalization capacity, we wanted to use a technique that could improve the classification results on a known dataset of grocery products [2]. In addition, we wanted a technique that used a single type of classifier. The authors of [19] had managed to significantly outperform the baseline results established by [2], using different classifiers independently trained on the same training data. We set out to find some technique that would allow us to improve their results without using different classifiers. Based on ResNeXt-101, we analyzed cascade classification in a previous paper [28]. That paper showed that the stacking technique worked well to improve the classifier results. However, we were not able to surpass the results of [19] with that technique. The novel CVV technique generalizes better, achieving an improvement in the results obtained by all the previous methods. It requires a single type of classifier and fewer epochs than other methods.

IV. EXPERIMENTS AND RESULTS DISCUSSION

In a first experiment, we have evaluated how the CVV training technique improves three different current models: ResNeXt-101 [40], EfficientNet B7 [41] and Wide ResNet-101 [42]. In all the defined models, apart from the convolution and pooling layers, a 0.2 dropout layer has been added between the features and the output classes. These models work with a single full connected layer connected between the feature vector, with the dropout, and the output classification vector. The network has been adapted to work with 600×600 input images. After several tests, we found that this size offered slightly better results than with other sizes, both larger and smaller. The interior architecture of the models has not been modified from the original, with the exception of the last connection layer. The cross entropy error function and Adam optimizer, with an initial learning rate of 5×10^{-5}, have been used during training. As we have used transfer learning of the models previously trained with ImageNet [37], we have used image normalization using the mean and standard deviation calculated per channel on it. This normalization method is usually applied with transfer learning and images with a histogram distribution similar to ImageNet. Beyond the normalization and use of transfer learning on all trained models, we did not use any additional image preprocessing technique.

Tables 1, 2 and 3 show the results of how the models improve by applying CVV training with 5 classifiers ResNeXt-101, EfficientNet B7 and Wide ResNet-101, respectively. Some values such as loss are not shown in this table as these values cannot be calculated in the ensemble model.

The results clearly show that the ensemble model using soft voting obtains the best results. In soft voting, the probabilities of each class are accumulated among the different models and the one with the highest probability is the winner. The number of estimators also influences the result. Table 4 shows the result of soft voting for different number of estimators applied on ResNeXt-101. Above 6 estimators, the results of the joint model start to decrease.

Table 5 shows the comparison of the models trained with our technique with other current models working on the same dataset. The authors of the Grocery Store Dataset [2] trained a

TABLE 3. Improvement of Wide ResNet-101 with CVV (% except for loss).

Model \ Metric	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy	Balanced test accuracy	Precision	Recall	F1-Score
Model with a single Wide ResNet-101	0.9962	0.0452	0.9924	0.0401	0.9000	0.9205	0.9210	0.9200	0.9210
Hard CVV with ensemble of 5 Wide ResNet-101 classifiers (ours)					0.9243	0.9358	0.9400	0.9360	0.9380
Soft CVV with ensemble of 5 Wide ResNet-101 classifiers (ours)					0.9296	0.9414	0.9450	0.9410	0.9430

TABLE 4. Ensemble of ResNeXt-101 CVV with different number of classifiers.

Model \ Metric	Test accuracy	Balanced test accuracy	Precision	Recall	F1-Score
Model with a single ResNeXt-101	0.9040	0.9112	0.9210	0.9110	0.9160
Soft CVV with ensemble of 2 ResNeXt-101 classifiers	0.9220	0.9297	0.9360	0.9300	0.9330
Soft CVV with ensemble of 3 ResNeXt-101 classifiers	0.9300	0.9379	0.9430	0.9380	0.9400
Soft CVV with ensemble of 4 ResNeXt-101 classifiers	0.9330	0.9416	0.9448	0.9420	0.9450
Soft CVV with ensemble of 5 ResNeXt-101 classifiers	0.9368	0.9472	0.9510	0.9470	0.9490
Soft CVV with ensemble of 6 ResNeXt-101 classifiers	0.9333	0.9433	0.9490	0.9430	0.9460

DenseNet-169 that combined the feature vector with an SVM classifier. They achieved 85% test accuracy with a model they fine-tuned. Similarly, they evaluated a DenseNet-169 classifier network without SVM. In that case, they achieved 84% accuracy.

Regarding the ensemble models, in [3], the authors presented an application of transfer and ensemble learning techniques (ETL) for cervical histopathology image classification. They used ResNet-50, Inception v3 [44], VGG16 and XCeption [45]. We have evaluated this technique and have seen that, although it improves the base classification results (87.8% accuracy), it obtains worse results than other existing methods and the one presented in our article. In [4], the authors presented a fuzzy rank-based ensemble of CNN models for classification of cervical cytology. They used DenseNet-169, Inception v3 and XCeption. This method achieves an accuracy of 88.45%, although also below than ours and other existing methods.

In [28], a ResNeXt-101 [40], a ResNet-152 [39] and a stacking model of two ResNeXt-101 were evaluated. For the ResNeXt-101, the test accuracy was 90.80%, with a precision of 92.50%, recall of 92.10%, balanced accuracy of 92.09%, and F1-Score of 92.30%. The same experiment was carried out with a ResNet-152. The test accuracy was 89.90%, with a precision of 91.60%, recall of 91.10%, balanced accuracy of 91.09%, and F1-Score of 91.30%. During the experiments, oranges and satsumas got very confused. To overcome that problem, a cascade classifier was proposed. The models were trained with early-stopping and 140 epochs. The validation data were split using 30% of the data, using a balanced split. In the stacking model, the result of the first classifier had 81 classes while the second classifier only 2 outputs (oranges and satsumas). That stacking model improved the test accuracy up to 92.0%.

It deserves special attention that in the experiments we have developed with CVV, we are training the models only 10 epochs and also using early-stopping. This is important since a training of 5 estimators takes even less time than a single training of a model for 140 epochs, provided that the early-stopping stops after 50 epochs.

In [19], the authors used a hard voting ensemble approach. They evaluated different cases, with the "C" and "D" ensemble models producing the best results. The ensemble "C" consisted of the models: ResNet-50, ResNet-101, ResNet-152, EfficientNet-B1, DenseNet-121, DenseNet-169 and DenseNet-201. The ensemble "D" consisted of the models: ResNet-101, ResNet-152, DenseNet-121, DenseNet-169 and DenseNet-201. These authors achieved the best classification results for this dataset to date.

The results show how of a single type of classifier trained with CVV, ResNeXt-101, in its soft voting mode, is able to outperform the results obtained by the best ensemble "C" of [19] (93.68% vs 93.48% test accuracy and 94.90% vs 94.46% F1-score).

TABLE 5. Comparison of different classification models (% except for loss).

Model \ Metric	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy	Balanced test accuracy	Precision	Recall	F1-Score
Model with a single ResNeXt-101 (10 epochs)	0.9929	0.0560	0.9848	0.0657	0.9040	0.9112	0.9210	0.9110	0.9160
Model with ResNeXt-101 [28] (140 epochs)	0.9978	0.015	0.9976	0.014	0.9080	0.9209	0.9250	0.9210	0.9230
Model with ResNet-152 [28] (140 epochs)	1.00	0.006	0.9963	0.017	0.8990	0.9109	0.9160	0.9110	0.9130
Cascade Model with ResNeXt-101 [28]	1.00	0.005	0.9988	0.009	0.9200	0.9306	0.9350	0.9310	0.9330
DenseNet-169 with SVM [2] (baseline)					0.8500				
ETL model with ResNet-50, Inception v3, VGG16 and XCeption [3]					0.8780	0.8864	0.9000	0.8860	0.8930
Fuzzy rank-based ensemble of CNN models: DenseNet-169, Inception v3 and XCeption [4]					0.8845	0.8960	0.9100	0.8960	0.9030
Ensemble "C" of different classifiers [19]					0.9348		0.9498	0.9452	0.9446
Ensemble "D" of different classifiers [19]					0.9314		0.9488	0.9442	0.9441
Soft CVV with ensemble of 5 ResNeXt-101 classifiers (ours)					0.9368	0.9472	0.9510	0.9470	0.9490
Soft CVV with ensemble of 5 EfficientNet B7 classifiers (ours)					0.9320	0.9447	0.9430	0.9450	0.9440
Soft CVV with ensemble of 5 Wide ResNet-101 classifiers (ours)					0.9296	0.9414	0.9450	0.9410	0.9430
Hard CVV with ensemble of 5 ResNeXt-101 and 5 EfficientNet B7 (ours)					0.9380	0.9478	0.9520	0.9480	0.9500
Soft CVV with ensemble of 5 ResNeXt-101 and 5 EfficientNet B7 (ours)					0.9408	0.9520	0.9550	0.9520	0.9540
Hard CVV with ensemble of 5 ResNeXt-101, 5 EfficientNet B7 and 5 Wide ResNet-101 (ours)					0.9440	0.9542	0.9570	0.9540	0.9560
Soft CVV with ensemble of 5 ResNeXt-101, 5 EfficientNet B7 and 5 Wide ResNet-101 (ours)					0.9441	0.9555	0.9580	0.9560	0.9570

Unlike other models, in our case we have trained each individual classifier for 10 epochs, which significantly reduces the training time. At the end of the table, more complex models that make a CVV of different classifiers are also shown. So for example, in the model using 5 ResNeXt-101, 5 EfficientNet B7 and 5 Wide ResNet-101, we train each of the individual classifiers using the CVV technique (spreading over 5 training/validation sets), and all the models are integrated using soft voting. The results clearly show an improvement over all other methods evaluated with this dataset, obtaining 94.41% test accuracy, 95.55% balanced test accuracy and 95.70% F1-score.

The training sessions have been carried out on an i9-10900K server with 128GB RAM and 2 GPU RTX-3090 with 24GB GDDR6X. As a guideline, this server required 60 minutes to train each of the ResNeXt-101 classifiers.

Another set of experiments have been performed to see how the model behaved using a different classification algorithm. Instead of connecting the output features of the convolutional part of the neural network to a classification layer, the features have been extracted to evaluate other models. Using the features provided by ResNeXt-101, the CVV technique has been applied for 5 estimators. Then, the ResNeXt-101 models have been trained and the image features have been extracted for each estimator. Next, several SVM classifiers have been trained, one per estimator. In the case of SVM, hard voting has been performed. Each of the SVM estimators has been trained by means of a grid-search, and each estimator could have different kernel, or C and gamma values (e.g., for the first estimator: [C: 0.1, gamma: 1, kernel: *linear*], for the second: [C: 1, gamma: 0.001, kernel: *rbf*]). The basis of this experiment has been to analyze whether this algorithm could improve the results obtained by classifying directly with the neural network or not, as they did in [2]. However, this experiment has shown that, in our case, the classification has worked better using the whole neural model with 5 estimators.

TABLE 6. CVV with SVM and Boosting over ResNeXt-101 features.

Model \ Metric	Test accuracy	Balanced test accuracy	Precision	Recall	F1-Score
Model with a single ResNeXt-101	0.9040	0.9112	0.9210	0.9110	0.9160
Hard CVV with ensemble of 5 ResNeXt-101 classifiers	0.9352	0.9446	0.9490	0.9450	0.9470
Model with a single SVM (ResNeXt-101 features)	0.8930	0.8984	0.9130	0.8980	0.9060
Hard CVV with ensemble of 5 SVM (ResNeXt-101 features)	0.9220	0.9315	0.9390	0.9320	0.9350
Model with a single AdaBoost (Decision Trees) (ResNeXt-101 features)	0.8730	0.8787	0.8990	0.8790	0.8890
Hard CVV with ensemble of 5 AdaBoost (Decision Trees) (ResNeXt-101 features)	0.9090	0.9173	0.9300	0.9170	0.9240
Model with a single XGBoost (ResNeXt-101 features)	0.8980	0.9110	0.914	0.9110	0.9120
Hard CVV with ensemble of 5 XGBoost (ResNeXt-101 features)	0.9300	0.9406	0.9430	0.9410	0.9420
Model with a single LightGBM (ResNeXt-101 features)	0.8890	0.8998	0.9070	0.9000	0.9040
Hard CVV with ensemble of 5 LightGBM (ResNeXt-101 features)	0.9230	0.9345	0.9400	0.9350	0.9370
Model with a single CatBoost (ResNeXt-101 features)	0.9030	0.9107	0.9220	0.9110	0.9160
Hard CVV with ensemble of 5 CatBoost (ResNeXt-101 features)	0.9300	0.9382	0.9500	0.9380	0.9440

Following the same procedure, different boosting algorithms have been evaluated on the basis of the features. The training of these algorithms is very computationally expensive if the whole convolutional part is trained in each estimator, so the approach using only the features is more convenient. Among the boosting algorithms, both individually and using CVV, we have evaluated:

- AdaBoost [22], the classic boosting method that consists of creating several simple predictors in sequence, so that the second one adjusts the errors of the first classifier, the third one the errors of the second one and so on. Finally, all classifiers are merged into a strong classifier. The classifier used has been a decision tree and the search for parameters has been carried out using a grid search, obtaining the best results with 1, 000 estimators, a depth per tree of 40 and a learning rate of 1.5.
- XGBoost [24] is an extreme gradient boosting method inspired in [46], a method that built an additive model in a forward stage-wise fashion, optimizing arbitrary differentiable loss functions. XGBoost adds different features such as clever penalization of trees, Newton boosting, proportional shrinking of leaf nodes, extra randomization parameter, distributed computing and automatic feature selection. After a grid search, the best results have been obtained using trees based on the faster histogram optimized approximate greedy algorithm, with a learning rate of 0.01, a number of estimators equal to 4, 000 and a maximum number of discrete bins to bucket continuous features equals 2. For this method, early-stopping has been used during training.

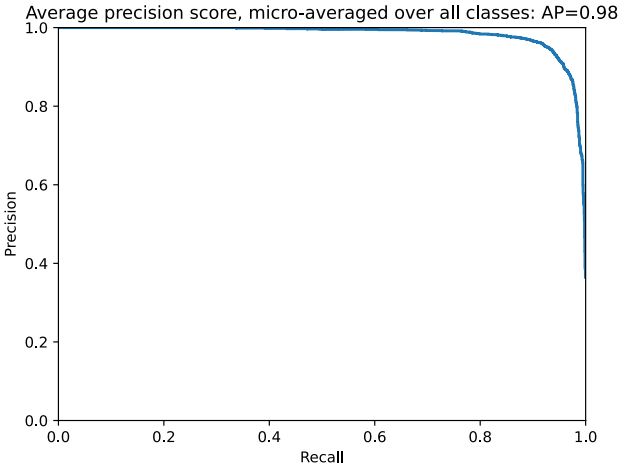


FIGURE 4. PR curve for the Soft CVV with 5 estimators of each type, with the average precision score, micro-averaged over all classes.

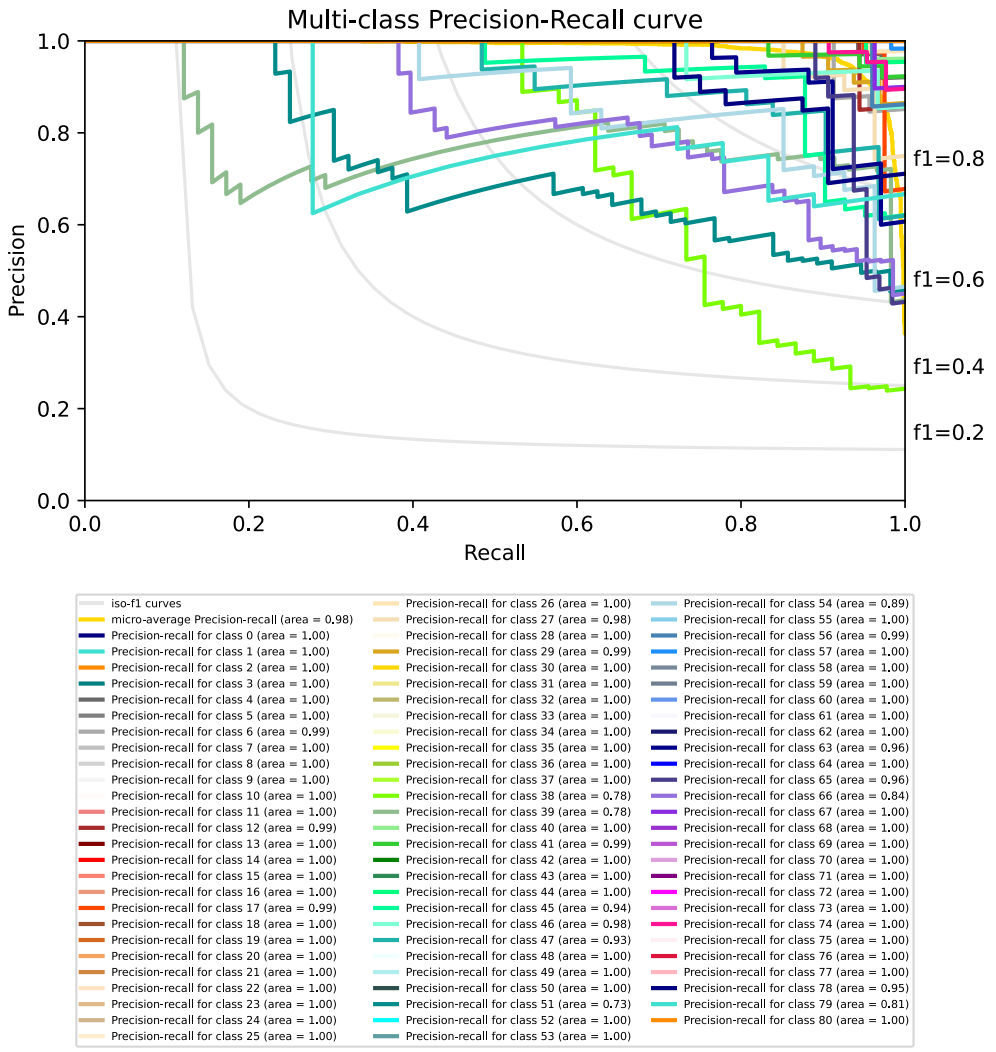


FIGURE 5. Multi-Class PR curve for the Soft CVV with 5 estimators of each type.

- LightGBM [26] is a method that implements the growing of the tree using leaf-wise, a technique that chooses the leaf it believes will yield the largest decrease in loss. LightGBM implements a highly optimized histogram-based decision tree learning algorithm, improving efficiency and reducing memory consumption. In addition, it uses Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) that allow the algorithm to keep the accuracy while running faster. After a grid search, the best results have been obtained using DART trees [47], 200 estimators, a learning rate of 0.09 and a maximum number of discrete bins equals 3.
- CatBoost [25], a gradient boosting method that handles categorical features, uses ordered boosting to overcome overfitting and oblivious or symmetric trees for faster execution. After a grid search, the best results have been obtained with 2,000 iterations, a maximum depth per tree of 5 and a learning rate of 0.1. For this method, early-stopping has also been used during training.

These methods are some of the most up-to-date methods using boosting techniques. Again, this experiment has shown that, in our case, the classification has worked better using the whole neural model with 5 estimators. Table 6 shows the results of this experiment.

Both CatBoost and XGBoost have given the best results combined with the CVV technique using the ResNeXt-101 features. SVM has obtained close but slightly inferior results. However, none of these techniques has been able to overcome the final connection layer of classification of the neural network itself. This allows us to affirm that each individual model is trained in an optimal way directly with the neural network. Although other authors, such as in [2], had obtained good results by connecting SVM to the features, it is clear from our experimentation that it is not necessary to extract the features and we simply have to let the model be trained directly using the classification layer itself.

As seen previously, the best model has been obtained by Soft CVV with 5 estimators of each type: ResNeXt-101, EfficientNet B7 and Wide ResNet-101. A Precision-Recall

TABLE 7. Confusion Matrix of the Soft CVV with 5 estimators of each type (ResNeXt-101, EfficientNet B7 and Wide ResNet-101).

Predicted

Actual	Apple	Asparagus	Aubergine	Avocado	Banana	Brown-Cap-Mushroom	Cabbage	Carrots	Cucumber	Garlic	Ginger	Juice	Kiwi	Leek	Lemon	Lime	Mango	Melon	Milk	Nectarine	Oat-Milk	Oatghurt	Onion	Orange	Papaya	Passion-Fruit	Peach	Pear	Pepper	Pineapple	Plum	Pomegranate	Potato	Red-Beet	Red-Grapefruit	Satsumas	Sour-Cream	Sour-Milk	Soy-Milk	Soyghurt	Tomato	Yoghurt	Zucchini
Apple	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Asparagus	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Aubergine	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Avocado	0.0	0.0	0.0	0.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Banana	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Brown-Cap-Mushroom	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Cabbage	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Carrots	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Cucumber	0.0	0.04	0.0	0.0	0.0	0.0	0.0	0.0	0.81	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.0	0.11
Garlic	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Ginger	0.0	0.07	0.0	0.0	0.0	0.0	0.0	0.0	0.07	0.87	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Juice	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Kiwi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.93	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.04	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Leek	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Lemon	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.76	0.07	0.0	0.05	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.0	0.02	0.05	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Lime	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.97	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Mango	0.19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.77	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Melon	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Milk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Nectarine	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Oat-Milk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Oatghurt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Onion	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Orange	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.79	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.14	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Papaya	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Passion-Fruit	0.0	0.0	0.07	0.04	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.89	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Peach	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pear	0.03	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.96	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pepper	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pineapple	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.96	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Plum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pomegranate	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Potato	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Red-Beet	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Red-Grapefruit	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Satsumas	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.65	0.0	0.0	0.0	0.0	0.0	0.0
Sour-Cream	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
Sour-Milk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Soy-Milk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
Soyghurt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
Tomato	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
Yoghurt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
Zucchini	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

curve with the average precision score, micro-averaged over all classes, is shown in Figure 4. In addition, a Multi-Class Precision-Recall curve, with all classes, is shown in Figure 5. The evaluation of the model shows a good performance, leading to 98% AUC (Area Under the Curve). The confusion matrix of this model is shown in Table 7. To show the results clearly, instead of showing the 81 fine classes, we present the 43 coarse categories into which the results have been grouped.

Among the limitations of the presented method are the need to train multiple models and the need for parallel inference in order to operate in real time. However, we have shown how individual models can be trained in a smaller number of epochs than in a normal training. Furthermore, at the real-time execution level it would be sufficient to size the GPU memory to be able to load all models simultaneously. Since the size that the models have during inference is usually quite smaller than during training (approximately 2.2GB for a ResNeXt-101 during inference), it is possible to load several models and infer in parallel without significantly reducing the total runtime with respect to when we use a single classifier. So this method, despite requiring the training of several models, can be used for real-time applications of various kinds, such as industrial applications. Beyond its use in the problem of grocery product classification, in industry and other sectors high accuracy is sought, so this technique allows to overcome the results that individual classifiers offer.

V. CONCLUSION

In recent years, there has been significant progress in deep learning methods to solve classification problems of considerable complexity, such as the recognition of grocery products by means of computer vision. The training of these models is usually performed by dividing the data into training, validation and test. One of the problems with this technique is that the model is trained with the training data and periodically evaluated with validation until the accuracy of this starts to decrease. Using early-stopping, we can stop training when this happens, ensuring that the model is optimal on data not used directly in training. Subsequently, the model is evaluated against the test data. The problem with this approach is that by using validation to block training we are making the model optimal against that set of validation, but it may not be optimal against other different validation samples. There are cross-validation techniques that allow the model to be evaluated with different validation sets, but they do not allow us to obtain a model that takes into account all of these sets. All these problems lead to a lack of generalization that we find in many models used in various fields, such as industry.

Traditionally used in classical algorithms, bagging, stacking, boosting or voting techniques, allow to integrate multiple models to obtain a more powerful classifier. However, these techniques are mainly designed to benefit from classifiers of different nature to obtain an overall result that improves all of them. Among all these techniques, bagging selects subsets of elements to train the same or different classifiers. Originally, when there were practically no validation sets in use, this method made it possible to make a model that generalized better in the face of unknown situations.

Our paper applies a method, named Cross Validation Voting (CVV), to the grocery product classification problem.

It allows benefiting from the advantages of cross validation to be integrated into the training scheme itself. The data are partitioned as in the k-fold validation technique, but each pair of slots is used to train a classifier of the same type. In this way, each classifier stops at the point where it performs best against its corresponding validation. As all validations are different, each model is optimal for its respective validation. Subsequently, by grouping all the models using voting techniques, we manage to improve the results obtained by the model at the individual level in all the current neural networks evaluated. The resulting model significantly improves the results and the generalization capacity of the individual models.

Numerous experiments have been carried out and we have shown how the best results obtained to date on a well-known dataset of grocery products have been improved. In addition, we have shown that, when classifiers of different nature are integrated into the model, the results improve even more.

We are currently applying this method to the improvement of classification systems used in other fields, such as industry, activity recognition and robotics. In industry, for example, the applications require very high precision. Beyond being used for classification models using computer vision, this technique can also be used for any other type of neural network or algorithm that allows a stop to be made on the basis of validation data. It is a method that offers very promising results and makes it possible to advance in the classification methods for products in the agrifood and supermarket sector.

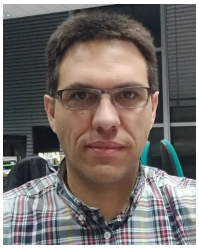
REFERENCES

[1] L. Wang, *Support Vector Machines: Theory and Applications*, vol. 177. Cham, Switzerland: Springer, 2005.
[2] M. Klasson, C. Zhang, and H. Kjellstrom, "A hierarchical grocery store image dataset with visual and semantic labels," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 491–500.
[3] D. Xue, X. Zhou, C. Li, Y. Yao, M. M. Rahaman, J. Zhang, H. Chen, J. Zhang, S. Qi, and H. Sun, "An application of transfer learning and ensemble learning techniques for cervical histopathology image classification," *IEEE Access*, vol. 8, pp. 104603–104618, 2020.
[4] A. Manna, R. Kundu, D. Kaplun, A. Sinitca, and R. Sarkar, "A fuzzy rank-based ensemble of CNN models for classification of cervical cytology," *Sci. Rep.*, vol. 11, no. 1, pp. 1–18, Dec. 2021.
[5] M. S. Advani, A. M. Saxe, and H. Sompolinsky, "High-dimensional dynamics of generalization error in neural networks," *Neural Netw.*, vol. 132, pp. 428–446, Dec. 2020.
[6] Y. Xu and R. Goodacre, "On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning," *J. Anal. Test.*, vol. 2, no. 3, pp. 249–262, Jul. 2018.
[7] M. M. Badža and M. Č. Barjaktarović, "Classification of brain tumors from MRI images using a convolutional neural network," *Appl. Sci.*, vol. 10, no. 6, p. 1999, Mar. 2020.
[8] M. Filax, T. Gonschorek, and F. Ortmeier, "Grocery recognition in the wild: A new mining strategy for metric learning," in *Proc. 16th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2021, pp. 498–505.
[9] J. Naranjo-Torres, M. Mora, R. Hernández-García, R. J. Barrientos, C. Fredes, and A. Valenzuela, "A review of convolutional neural network applied to fruit image processing," *Appl. Sci.*, vol. 10, no. 10, p. 3443, May 2020.
[10] Z.-H. Zhou, "Ensemble learning," in *Machine learning*. Cham, Switzerland: Springer, 2021, pp. 181–210.
[11] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining Knowl. Discovery*, vol. 8, no. 4, Jul. 2018, Art. no. e1249.
[12] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers Comput. Sci.*, vol. 14, no. 2, pp. 241–258, 2020.
[13] D. Burka, C. Puppe, L. Szepesváry, and A. Tasnádi, "Voting: A machine learning approach," *Eur. J. Oper. Res.*, vol. 299, no. 3, pp. 1003–1017, Jun. 2022.
[14] S. Zheng, P. Qi, S. Chen, and X. Yang, "Fusion methods for CNN-based automatic modulation classification," *IEEE Access*, vol. 7, pp. 66496–66504, 2019.
[15] E.-S. M. El-kenawy, A. Ibrahim, S. Mirjalili, M. M. Eid, and S. E. Hussein, "Novel feature selection and voting classifier algorithms for COVID-19 classification in CT images," *IEEE Access*, vol. 8, pp. 179317–179335, 2020.
[16] R. Zhu, Z. Xiao, Y. Li, M. Yang, Y. Tan, L. Zhou, S. Lin, and H. Wen, "Efficient human activity recognition solving the confusing activities via deep ensemble learning," *IEEE Access*, vol. 7, pp. 75490–75499, 2019.
[17] H. Huang, Z. Hu, W. Wang, and M. Wu, "Multimodal emotion recognition based on ensemble convolutional neural network," *IEEE Access*, vol. 8, pp. 3265–3271, 2020.
[18] H. Yao, T. Qiao, M. Xu, and N. Zheng, "Robust multi-classifier for camera model identification based on convolution neural network," *IEEE Access*, vol. 6, pp. 24973–24982, 2018.
[19] M. Leo, P. Carcagni, and C. Distante, "A systematic investigation on end-to-end deep recognition of grocery products in the wild," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 7234–7241.
[20] L. G. Kabari and U. C. Onwuka, "Comparison of bagging and voting ensemble machine learning algorithm as a classifier," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 9, no. 3, pp. 19–23, 2019.
[21] M. Praveena and V. Jaiganesh, "A literature review on supervised machine learning algorithms and boosting process," *Int. J. Comput. Appl.*, vol. 169, no. 8, pp. 32–35, 2017.
[22] R. E. Schapire, "Explaining adaboost," in *Empirical Inference*. Berlin, Germany: Springer, 2013, pp. 37–52.
[23] K. Hameed, D. Chai, and A. Rassau, "A sample weight and AdaBoost CNN-based coarse to fine classification of fruit and vegetables at a supermarket self-checkout," *Appl. Sci.*, vol. 10, no. 23, p. 8667, Dec. 2020.
[24] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, and H. Cho, "XGBoost: Extreme gradient boosting," *R Package Version* vol. 1, no. 4, pp. 1–4, Aug. 2015.
[25] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. A. Gulin, "CatBoost: Unbiased boosting with categorical features," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 6639–6649.
[26] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 3146–3154.
[27] B. Pavlyshenko, "Using stacking approaches for machine learning models," in *Proc. IEEE 2nd Int. Conf. Data Stream Mining Process. (DSMP)*, Aug. 2018, pp. 255–258.
[28] J. D. Domingo, R. M. Aparicio, and L. M. González Rodrigo, "Improvement of one-shot-learning by integrating a convolutional neural network and an image descriptor into a Siamese neural network," *Appl. Sci.*, vol. 11, no. 17, p. 7839, Aug. 2021.
[29] D. Held, S. Thrun, and S. Savarese, "Deep learning for single-view instance recognition," 2015, *arXiv:1507.08286*.
[30] B. Sainz-De-Abajo, J. M. Garcia-Alonso, J. J. Berrocal-Olmeda, S. Laso-Mangas, and I. D. L. Torre-Diez, "FoodScan: Food monitoring app by scanning the groceries receipts," *IEEE Access*, vol. 8, pp. 227915–227924, 2020.
[31] P. Follmann, T. Bottger, P. Hartinger, R. Konig, and M. Ulrich, "MVTec D2S: Densely segmented supermarket dataset," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 569–585.
[32] X.-S. Wei, Q. Cui, L. Yang, P. Wang, and L. Liu, "RPC: A large-scale retail product checkout dataset," 2019, *arXiv:1901.07249*.
[33] P. Jund, N. Abdo, A. Eitel, and W. Burgard, "The freiburg groceries dataset," 2016, *arXiv:1611.05799*.
[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25. Stateline, NV, USA, Dec. 2012, pp. 1097–1105.
[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[36] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "DenseNet: Implementing efficient ConvNet descriptor pyramids," 2014, *arXiv:1404.1869*.

[37] 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.
[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[39] 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.
[40] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1492–1500.
[41] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[42] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.
[43] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2820–2828.
[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[45] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[46] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.
[47] R. K. Vinayak and R. Gilad-Bachrach, "Dart: Dropouts meet multiple additive regression trees," in *Proc. Artif. Intell. Statist.*, 2015, pp. 489–497.

JAIME DUQUE DOMINGO received the Technical Engineering and B.S. degrees in computer engineering from the University of Valladolid, in 2000 and 2011, respectively, the M.S. degree in software and systems engineering and the Ph.D. degree *(cum laude)* in systems and control engineering from the UNED, in 2014 and 2018, respectively, and the M.Ed. degree in mathematics from the University Isabel I, in 2018. For 18 years, he worked in the development of complex IT projects for the private sector, both in Spain and abroad. He currently works as a Researcher at the CARTIF Technology Center and as a Professor at the University of Valladolid and at the European University Miguel de Cervantes (UEMC). He has carried out projects in the insurance/banking field (Seguros Caja Duero), in governmental applications (French Social Security through ATOS IT), European Commission (IT Department of the Commission in Brussels), in the automotive sector (Renault) or eCommerce projects (MachinePoint). During the last years, he has focused on the academic world. He has published ten articles in journals indexed in SCI-JCR, in the first and second quartile, and nine articles in international conferences. His field of activity is mainly focused on computer vision and robotics, specializing in positioning systems as well as deep neural networks. He has received four awards in his research: INFAIMON 2015 and 2018 (Best Computer Vision Paper), IARIA-ICWMC 2017 (Best Paper at ICWMC, Nice, France, 2017), and Extraordinary Ph.D. Prize of the UNED in 2020.

ROBERTO MEDINA APARICIO received the Industrial Engineering and Ph.D. degrees from the University of Valladolid. He has been working as a Researcher at CARTIF, since 2005, where he has combined research work and industrial development. He began his professional career with the Robotics and Machine Vision Division and is currently part of the Industrial and Digital Systems Division. He is also the Project Manager at CARTIF of CERVERA network in robotic technologies for smart manufacturing (5R). He has extensive experience in research projects involving machine vision, sensorization, 3D reconstruction, and distributed computing. He is also working on research projects related to computer vision based on deep neural networks, such as I-visart ("New Artificial Vision Methodologies for the Visual Inspection of Highly Reflective and Textured Surfaces") and Agrovis ("Artificial Vision for Products/Processes in the Agrifood Sector"). He has published eight scientific articles in scientific journals of impact and in various conferences, as well as has participated in one international patent. He worked on many research projects, both national and European.



LUIS MIGUEL GONZÁLEZ RODRIGO received the degree in automatic control and systems engineering and the M.Eng. degree in automatic control and systems program from the University of Valladolid, Spain, in 2003 and 2007, respectively. He has been a Project Manager and a Research and Development Engineer with the Robotics and Computer Vision Division, CARTIF Technological Centre, for a period of over 17 years. He has extensive experience in the design, development and management of industrial projects associated with quality control and process automation through the use of artificial vision techniques, mainly in the automotive sector. He worked on many research projects, both national and European, such as Trex ("Extended Range Robot Enabling Technologies for the Flexible Factory") and Vapex ("Novel Vapor Analyzer for the Detection of Explosives in Airports Passengers Checkpoints"), where CARTIF will define and develop an automatic robotic manipulator to manage and transport the analyzer filters. He is currently working on research projects related to computer vision based on deep neural networks, such as I-visart ("New Artificial Vision Methodologies for the Visual Inspection of Highly Reflective and Textured Surfaces") and Agrovis ("Artificial Vision for Products/Processes in the Agrifood Sector").