Neural Network Compression for Visual Quality Control in Industrial Edge Computing

Eliton Katsuhiro Nouchi
University of São Paulo
São Carlos, SP, Brazil
nouchieliton@gmail.com

Fernando Pereira dos Santos
University of São Paulo
Venturus - Innovation & Technology
Campinas, SP, Brazil
fernando.santos@venturus.org.br

***Abstract*—Ensuring that customers receive defect-free products is a big challenge for industries. Visual inspection is a method that can retain defective products before they leave the production line. Deep Learning has enabled us to automate this process using very complex neural networks. However, edge devices are resource-constrained, and the deployment of very deep models may not be feasible. To address this problem, our methodology investigated the application of Knowledge Distillation and Network Quantization to reduce the requirement for resources while trying not to lose performance, enabling deployment on edge computing devices. Knowledge Distillation consists of training a small model (student) under the guidance of a complex one (teacher). Network Quantization reduces the numerical precision of the weights to further decrease the storage size of the model. Our methodology was evaluated in the MVTecAD, DML, and VAD datasets. The results were compared between the teacher and student models. To measure the benefits brought by compressed techniques, we also fine-tuned the student model without the teacher's guidance and compared it to our approach. The results show that our methodology reduced the model size to 5.5% of the teacher's size to the best-case scenario, being 36 times faster, while preserving similar performance to the complex model for most objects.**

I. Introduction

One of the greatest challenges for industries is quality control on the production line, ensuring that all customers receive defect-free products. **Strict quality control** helps avoid returns, exchanges, or post-sale maintenance, which can lead to customer dissatisfaction and increased costs for the industry. Visual inspection is a manufacturing step that is applied to analyze the surface of the product in an attempt to identify defects. Detecting whether a product is defective or not is closely related to identifying instances that behave differently or have characteristics that differ significantly from most products, which characterizes an anomaly detection task [1]. However, when performed by humans for long working hours, this task can lead to misclassifications due to fatigue and cause health problems [2]. The visual process can be automated by computer vision pipelines that analyze the apparent aspects of the objects produced and return as a response whether the product is within or outside the expected standard.

Deep Learning has enabled the solution of **real-world quality control problems** by learning very complex representations, mainly from unstructured data such as images. These advances have made it possible to automate the detection of visual anomalies in industry by applying deep convolutional neural networks to analyze product images and classify them as normal or defective. Improvements in the performance of Deep Learning applications are frequently associated with an increase in the number of parameters and the complexity of the architecture [3], [4], which demands more computational and storage resources, thus limiting their application on edge devices. In cases where a product needs to be visually inspected on the production line with near-real-time processing, the images should be processed close to the data source, i.e. at the network edge, reducing latency compared to cloud-based computing [5]. However, **edge devices are more resource-constrained**, and the deployment of very complex deep learning models may not be feasible. This could lead to the use of simpler models with lower performance or require significantly higher investments in hardware. **Model compression techniques** have been studied to reduce the resource consumption of complex models with minimal loss of accuracy, enabling their deployment in embedded systems, Internet of Things (IoT), and edge devices. There are several ways in which these techniques attempt to simplify models, such as reducing the numerical precision of parameters, removing less important parameters, and training a smaller model to mimic the output of a more complex one [6].

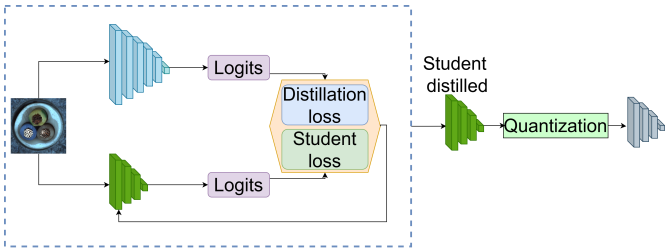

Fig. 1. Neural Network Compression for Visual Quality Control in Industrial Edge Computing. We first trained a complex neural network for classification and used it as a teacher to guide the learning of a simplified network, student. Once the student network had learned, we quantized its weights to reduce the size of the structure and make it deployable on edge devices.

This paper addresses the problem of visual anomaly detection using two convolutional neural networks (CNNs) classifiers trained on images of normal and defective products. Data augmentation techniques were employed to increase the diversity of training data and mitigate the effects of dataset

imbalance. One CNN is a large and complex model, referred to here as the teacher, which guides the training of the other CNN, a smaller and simpler model called the student. This training approach is called Knowledge Distillation and aims to enable the use of a smaller model with performance similar to that of the teacher. To further reduce the storage size of the model, the numerical precision of the parameters was reduced by Network Quantization to the distilled model. This training pipeline is illustrated in Figure 1. We evaluated our methodology on three datasets, comparing the performance and size of the models at each stage of compression. The results show that our methodology can compress a complex model to a fraction of its size while preserving similar performance, enabling deployment on edge devices.

In summary, **our contributions are two-fold**: (i) generalized analysis of the same supervised architectures for different anomaly detection datasets; and (ii) analysis of model compression by Knowledge Distillation and Network Quantization that allows the adoption of Deep Learning on edge devices with minimal loss of performance. Our goal is to show that model compression techniques allow the deployment of high-performance neural networks on devices with computational constraints and to establish a pipeline that can be applied to several visual inspection scenarios in industry.

II. Related Work

A. Deep Learning for Industrial Anomalies

Deep Learning has been able to surpass the performance of traditional image processing methods for anomaly detection, as well as reduce the need for manual feature engineering [7], [8]. The nature of this task presents several challenges. One of them is determining what truly is an anomaly, since an anomaly is anything that deviates from normal case, making class separation difficult [1]. Another challenge that can be mentioned is the rarity of abnormal data. Production lines are designed to be fault-free, and defects are exceptions. Hence, only a few samples with real defects are available, leading to highly imbalanced datasets [2]. Several approaches have been proposed to avoid this problem, including a method that cuts an image patch, applies a transformation, and pastes it at a random location of an image to artificially generate the anomaly [9]. In general, **Deep Learning addresses the anomaly detection problem** by learning feature representations or anomaly scores from data using complex multilayered neural networks [10]–[12]. The feature representations can be used to train a traditional machine learning model that separates abnormal instances from normal ones [13]. In this sense, [14] is an example of using an anomaly score in a student–teacher structure to identify defective products. In this approach, the student is trained to regress the teacher's output using only normal data. During inference, the anomaly score is computed as the distance between the teacher's and student's output, indicating whether a product is defective. **Our approach** goes beyond using just a single model compression technique [14]–[16], aggregating two steps to minimize the complexity of the neural network without significantly impacting the detection performance of abnormal cases and enabling use on edge devices.

B. Edge Neural Networks

The cloud has made high-performance hardware affordable and accessible to individuals and companies. Cloud resources can be scaled on demand and charged on a pay-as-you-go basis, making it a flexible service. However, to take advantage of these benefits, data must be transferred from the source to the cloud via the Internet. This can be an issue for applications that require real-time inference, and network bandwidth can become a bottleneck in the infrastructure with a large number of devices [17]. Edge computing, on the other hand, avoids these problems by processing data close to the source. Edge computing can be defined as a set of technologies that allow computation between the data source and the cloud [5]. Examples of edge devices include smartphones, local servers, and surveillance cameras. The general idea is that **edge computing** should process data near where it is generated. The combination of neural networks and edge computing appears to be an attractive option for a wide range of applications. However, deep neural networks have become increasingly complex in order to improve accuracy, which requires more computational resources. To make deployment feasible, some optimizations must be applied to deep neural networks, considering the trade-off between performance and resource consumption [18]. The optimization of deep learning models can be achieved by compressing existing models using techniques such as model pruning [19]. Another approach focuses on model design, aiming to reduce the number of parameters by modifying the architecture and operations of the model. An example of this is MobileNets [20], which uses depthwise separable convolutions to create a lightweight model. Hence, these methods have been able to build edge neural networks with performance comparable to that of deeper models. **Our experiments** show that less complex neural networks for edge devices perform better when trained from more complex networks than independently, even when subject to high compression rates after training.

III. Methodology and Implementation

Our methodology involves **training a complex architecture** for image classification to detect defective objects, as described in subsection III-A, and transfer this learning to guide the training of a smaller neural network through **Knowledge Distillation**, detailed in subsection III-B, aiming to lose as little as possible in terms of performance. Subsequently, in subsection III-C, numerical precision is reduced to further decrease the size of the smaller architecture with **Network Quantization**. Our process involves reducing model complexity while ensuring the lowest possible performance loss so that industrial inspection on edge devices can be guaranteed.

A. Deep Neural Networks

Convolutional Neural Networks (CNNs) have formed the foundation for Image Processing in Deep Learning for classification tasks lately. Their primary goal is to learn meaningful

feature representations from input data through a series of operations organized into layers [21]. The fundamental components of a CNN are the convolutional and pooling layers. In convolutional layers, a 2D input image $X \in \mathbb{R}^{h \times w}$ is convolved with a set of kernels $W_k \in \mathbb{R}^{d \times d}$ and biases $b_k \in \mathbb{R}$ in a hierarchical architecture. This structure generates feature maps at different levels of representation [22], where previous layers enable the learning of low-level features, such as colors and shapes, and later layers better represent high-level features, such as semantic context [23]. Internally, this composition of layers allows the acquired learning to be transferred to other contexts, adapting only the final layers to the target domain [4] through techniques such as fine-tuning or Knowledge Distillation [6]. A nonlinear activation function σ is applied after each convolution layer to modularize the response signal [24], as denoted by equation 1. Pooling layers perform downsampling to reduce redundancy by aggregating neighboring pixel values using a max or average operation [25].

$$X_k^l = \sigma(W_k^l \circledast X^{l-1} + b_k^l) \tag{1}$$

Our methodology involves learning and fine-tuning two CNNs of different complexities. The **more complex architecture** is ResNet50v2 [26]. The main idea of this architecture is the residual connections that allow the network to have hundreds or even thousands of layers without being significantly affected by the problem of the vanishing gradient [27]. The **less complex architecture** is the MobileNetv3Small [28], which was designed focusing on computational efficiency. This efficiency comes mostly from the depthwise separable convolutions that reduce the number of parameters compared to a traditional convolution. Theoretically, a more complex neural network tends to perform better than a less complex network, as long as it has the quantity and quality of data [4]. However, complex architectures require greater computational capacity, which is not feasible in edge devices [6]. The concept of Knowledge Distillation, discussed in subsection III-B, finds this scenario, in which the learning of the more complex architecture improves the performance of the simplified architecture. In our approach, both CNNs were initialized with pre-trained weights from previous training on the ImageNet dataset [29], in order to reduce the requirement for a large number of training images.

B. Knowledge Distillation

Knowledge Distillation is based on a **student–teacher architecture**, in which the teacher is a pre-trained, larger, and more complex network that guides the training process of the student. The goal is for the student to imitate the teacher's outputs. The Knowledge Distillation strategy adopted in our methodology is a response-based approach. This strategy uses the teacher response of the last fully connected layer as prior knowledge during student training [30]. A widely used approach within response-based knowledge distillation is the concept of soft targets [31]. This method employs a temperature hyperparameter T, which is applied to the logits produced by the teacher and the student. A softmax function is then applied to these logits, and when the temperature is greater than one, it produces a softer probability distribution. Equation 2 represents this operation, where k is the number of classes, q_i is the probability of the i-th class, $exp(.)$ is the exponential function, z is the vector of logits where $z = (z_1, \ldots, z_k) \in \mathbb{R}^k$ and z_i is the logits of the i-th class, and T is the temperature. In the denominator we have the sum of the exponentials of the logits from all classes.

$$q_i = \frac{exp(z_i/T)}{\sum_{j=1}^{k} exp(z_j/T)} \tag{2}$$

The probability distributions obtained by the soft targets are then used to calculate the loss. The loss is composed of two parts: the first part, $\mathcal{L}_s$, is responsible for measuring the loss of the student's predictions with respect to the ground truth, and the second part, $\mathcal{L}_d$, is the distillation loss, which measures the difference between the probability distributions of the teacher t and the student s. A hyperparameter α is used to control the proportion of each component in the final loss. This custom loss is defined as in the equation 3, where p is the function that returns the probability distribution given the logits and T is the temperature. Minimizing this loss can result in a student model capable of reproducing the teacher's logits. Our knowledge distillation process was carried out using the **Kullback-Leibler Divergence** [32] as the distillation loss and categorical cross-entropy as the student loss.

$$\mathcal{L}_f = \alpha \cdot \mathcal{L}_s(y_{true}, z_s) + (1 - \alpha) \cdot \mathcal{L}_d(p(z_s, T), p(z_t, T)) \tag{3}$$

C. Network Quantization

Deep neural networks are generally trained using weights in 32-bit float format, but this format often has a numerical precision beyond what is necessary to perform the task [33]. With this in mind, Network Quantization seeks to convert the weights and activations of a neural network to a format with lower numerical precision, with as little loss of performance as possible [3]. The reduction in numerical precision provides benefits by reducing the storage, energy consumption, and shorter response time of the model [6]. **Post-training quantization** can be applied without the necessity of fine-tuning. Hence, there is no need for additional training data to quantize the weights. The least aggressive form of quantization is to convert the weights to 16-bit float, reducing the numerical precision while keeping the weights in floating-point format. Another form of quantization is to map the relationship between values in the 32-bit float to the 8-bit integer, which can be given by the following equations [3]:

$$S = \frac{R_{max} - R_{min}}{Q_{max} - Q_{min}} \tag{4}$$

$$Z = Q_{max} - \frac{R_{max}}{S} \tag{5}$$

where R represents the original values in 32-bit float and Q represents the range of 8-bit integer values, max and min

refer to the maximum and minimum values of each set of values, where an unsigned 8-bit integer has values in the range between 0 and 255 and signed 8-bit integer has values between -128 and 127. S is the factor that represents the scale between the two formats and Z is the 8-bit integer value referring to the zero in the 32-bit float [3]. With these quantization parameters, it is possible to quantize the original values by applying the following equation:

$$Q = \frac{R}{S} + Z \tag{6}$$

Considering the trade-off between accuracy and model complexity, and the lack of training images, we focus on post-training quantization to 16-bit float format, which causes minimal loss in accuracy and reduces the model size by approximately half.

IV. Experiments Setup

This section is dedicated to presenting the datasets and measures, subsections IV-A and IV-B, respectively, to evaluate our methodology, as well as the neural network training setup, see subsection IV-C, for teacher and student architectures.

A. Datasets

Our methodology is validated through three distinct datasets that have different objects and different concepts of anomalies, addressing many scenarios that occur in quality inspection in industries. For datasets that are already split for a classification task, we kept the original split and only separated a validation set from the training set. Hence, the testing set remains unchanged. Also, the images were processed at their original resolutions, requiring only the grayscale images to be transformed to the three color channels.

MVTecAD [34], presented in Figure 2, is an industrial image dataset that focuses on anomaly detection. The dataset contains 15 different objects for a total of 4096 normal and 1258 defective images. Each object has a different degree of imbalance, with the most imbalanced being the "transistor", which has 12.8% defective images, and the least imbalanced being the "pill", with 32.5% defective examples. The images have resolutions ranging from 700×700 to 1024×1024 pixels, and only the "grid", "zipper", and "screw" classes are grayscale images, the others ("bottle", "cable", "capsule", "carpet", "hazelnut", "leather", "metal nut", "pill", "tile", "toothbrush", "transistor", and "wood") are colored. Originally, the dataset contained only normal images in the training set. To employ the dataset in a supervised learning task, the training and testing sets were merged and then split into 56% for training, 14% for validation, and 30% for testing, so that there are normal and defective examples for all subsets.

Our methodology was also applied to the **DML** dataset [35], which includes 145 normal and 55 defective images. All images are colored with a resolution of 800-squared pixels. There are only two objects in this dataset: "hazelnuts", and "coffee beans". In the "hazelnut" class, 25% of the images are defective, and in the "coffee beans" class, 30%. The original split of this dataset was already built for a supervised learning task, and the only modification was to separate 20% of the training set for validation, which can be seen in Figure 3.

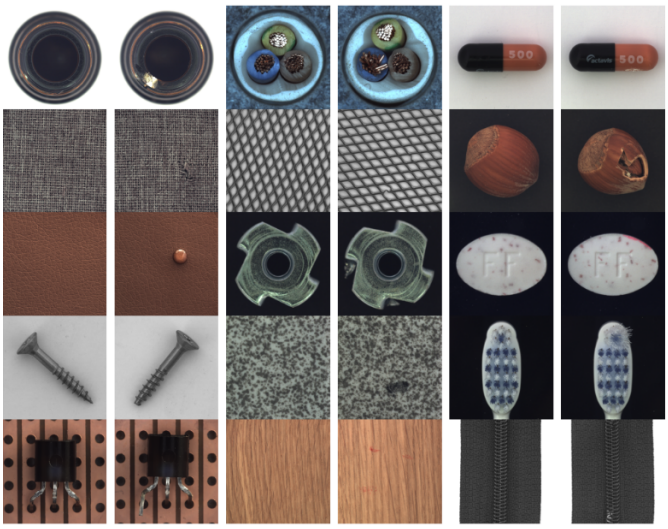

Fig. 2. MVTecAD. For each object, we have the normal and anomaly samples. Anomalies vary in each class, and can range from textures outside the acceptable standard ("grid" and "leather", for example) to defective parts, such as the "hazelnut" shell and solders on the "transistor".

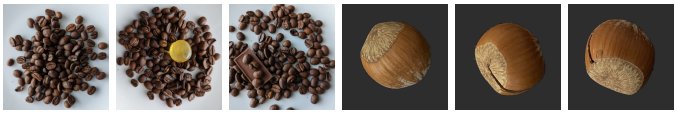

Fig. 3. DML. For each object, we have the normal case and two anomalies. Anomalies are objects other than coffee beans or cracks in the shell.

Fig. 4. VAD. We have the normal and two anomalies which illustrate small errors in solder, splashes, or incorrect positioning of wires. In this dataset, some anomalies are very subtle and difficult to notice.

The Valeo Anomaly Detection (**VAD**) [36], Figure 4, is another dataset used during evaluation. It contains images of an industrial component captured on a real production line, including the defective images, unlike most datasets in which defects are simulated. The dataset split contains 1000 defective and 2000 normal images in the training set, and 1000 defective and 1000 normal images in the testing set, all of them have a resolution of 512×512 pixels. In this dataset, 20% of the training set was considered for the validation set.

B. Measures

To evaluate the performance of our methodology, we used balanced accuracy, precision, recall, and F1-score [37]. **Balanced accuracy** was chosen because it considers the average of the true positive and the true negative rates, making it more suitable for imbalanced datasets than standard accuracy. The **precision** indicates how often the model is correct when it

predicts the positive class. **Recall** is the proportion of what the model correctly predicts as the positive class over all the actual positive class. The recall shows whether the model is capable of identifying defective objects. The **F1-score** is the harmonic mean of precision and recall, and it measures the performance of the two metrics together.

C. Neural Network Training Setup

The training process was carried out in two steps. First, the teacher model (ResNet50v2 [26] pre-trained with ImageNet [29]) was **fine-tuned** adapting to the new task. The teacher's knowledge was then **distilled** to the student model (MobileNetV3Small [28]). Student fine-tuning was also included as a parallel step to establish a baseline to measure the improvements brought about by knowledge distillation. To training all models, Adam optimizer [38] was applied with a learning rate of $1e-4$ and categorical cross-entropy loss. The training data was divided into batches of 32 images [39], and the last 13 layers were unfrozen to enable training. The **distillation hyperparameters** α and temperature T were set to 0.8 and 1.7, respectively, for the MVTecAD and DML datasets. For the VAD dataset, the only difference is that the temperature was set to 2.0. These are the optimal values found in multiple experiments, and here we report only the best result from each dataset. To add more variability to the datasets, the training images were artificially augmented by brightness, contrast, rotation, and Gaussian noise transformations.

TABLE I
PERFORMANCES OF MVTECAD WITH RESNET50V2.

Classes	Bal. Acc.	Precision	Recall	F1-Score
Bottle	1.0	1.0	1.0	1.0
Cable	0.982	1.0	0.964	0.981
Capsule	0.947	0.967	0.909	0.937
Carpet	1.0	1.0	1.0	1.0
Grid	1.0	1.0	1.0	1.0
Hazelnut	0.976	1.0	0.952	0.975
Leather	0.964	1.0	0.928	0.962
Metal Nut	1.0	1.0	1.0	1.0
Pill	0.941	1.0	0.883	0.938
Screw	0.986	1.0	0.972	0.985
Tile	1.0	1.0	1.0	1.0
Toothbrush	0.722	1.0	0.444	0.615
Transistor	0.958	1.0	0.916	0.956
Wood	1.0	1.0	1.0	1.0
Zipper	0.986	1.0	0.972	0.985

TABLE II
PERFORMANCES OF DML WITH RESNET50V2.

Classes	Bal. Acc.	Precision	Recall	F1-Score
Coffee Beans	0.8	1.0	0.6	0.749
Hazelnut	0.5	0.0	0.0	0.0

TABLE III
PERFORMANCES OF VAD WITH RESNET50V2.

Classes	Bal. Acc.	Precision	Recall	F1-Score
VAD	0.805	0.913	0.674	0.775

V. RESULTS AND DISCUSSION

Our results are divided into two subsections: (V-A) Deep Neural Network Analysis, in which we detail the performances only with the teacher network; and (V-B) Deep Neural Network Compression Analysis, in which we validate the behavior of model compression techniques in terms of performance, time response, and complexity for edge computing devices.

A. Deep Neural Network Analysis

This subsection presents the results from the teacher models, which will be used for comparisons with the compressed models later. An individual model was built for each object in each dataset, simulating plausible scenarios on industrial production lines. Hence, each performance is independent.

The first dataset to be evaluated was **MVTecAD**, in which Table I presents the performance of each teacher model. The high precision values of all models indicate that they are very realiable when they predict an image as normal (the worst precision achieved was 0.967 for the "capsule"). Sometimes, the high precision of a model is due to classifying as positive class only the instances in which it has high confidence, which can lead to some false negatives. In our case, the recall shows that most models were able to identify the most defective products, except the "toothbrush" with 0.444. However, we can observe from the anomaly sample presented in Figure 2 that this object is captured from a frontal perspective, with the main anomalies being constituted in a lateral perspective of the bristles. Therefore, a better approach would be to reposition the camera or rotate the object. The F1-score and balanced accuracy reveal that the complex models achieved good overall performance, with the exception of the "toothbrush" due to false negatives. Analyzing the respective confusion matrices for each object, Figure 5, we can see that errors occurred in the classification of defective images, mainly for "pill" and "toothbrush". However, the metrics are more affected for the "toothbrush" due to the low number of anomalous examples in the testing set. Taking into account misclassifications of normal cases, all have zero error, except for the "capsule" that misses the 1 example from 73 in total.

The performances for the **DML** teacher models are shown in Table II. The recall of the "coffee beans" class shows that it had some difficulty classifying defective images, resulting in lower balanced accuracy and F1-score. The measures of the "hazelnut" class indicate that the model was not able to learn with this configuration. Observing the description of the dataset, we notice that it consists of only 200 images in total addressing both classes, and the training set is insufficient to provide learning for very complex mathematical formulations, such as ResNet50v2. In this scenario, the concepts of Occam's Razor [40] and Bias-Variance Trade-off [41] are extremely pertinent, in which simplified models must be validated before complex models and the number of images required for learning more complex models is greater. The confusion matrices of this dataset, Figure 6, express the detailed performances, in which there is no error in identifying the normal class.

The model for the **VAD** dataset had measures similar to the

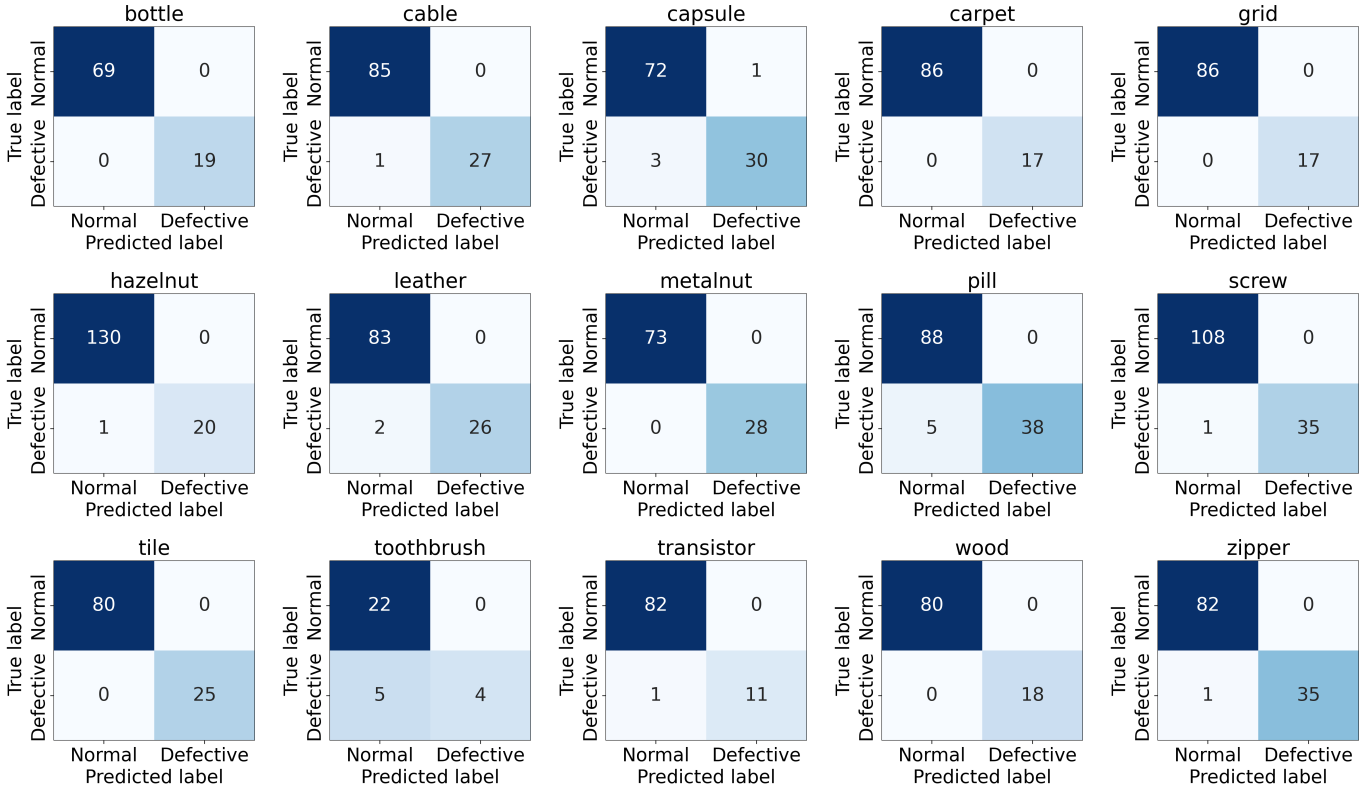

Fig. 5. Confusion Matrix of MVTecAD.

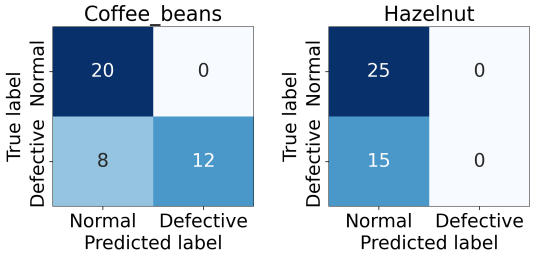

Fig. 6. Confusion Matrix of DML.

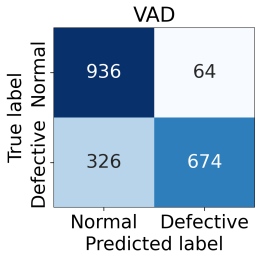

Fig. 7. Confusion Matrix of VAD.

performance of the "coffee beans" class. It is good at avoiding false positives but misses some defective products, resulting in an intermediate performance of balanced accuracy and F1-score, as shown in Table III. The testing set of this dataset is equally balanced and confirms previous analyses that normal behavior examples are "more easily" identified than anomalous examples (see Figure 7). In this dataset, the anomalies are more subtle than in the other classes performed, as exemplified in Figure 4, making it difficult to classify correctly.

B. Deep Neural Network Compression Analysis

In the first stage of compression, we applied Knowledge Distillation to guide the learning of the smaller network (MobileNetv3Small) by the refinement achieved by the larger network (ResNet50v2). Next, the second stage includes Network Quantization. Figure 8 shows the **size of each model resulting in MegaBytes** (MB) at each compression stage. We can see that the distilled model has only 10.8% of the teacher's size (9.7 MB of 89.7 MB), and after applying the quantization, the model reaches 5.5% of the teacher's size (4.9 MB of 89.7 MB). Quantizing from 32-bit float to 16-bit float yields a compression ratio of 50%. This high compression rate directly impacts the **response time** of visual inspection. We ran the three models on the same hardware[1], with the teacher model processing 0.39 frames per second (FPS); while the compressed models reached a rate of 14 FPS, 36x faster.

In order to measure the real gain from compression techniques in classification, a student model was fine-tuned without the guidance of the teacher. This baseline allows us to compare the performance of the student in different approaches and

[1]CPU Intel Xeon 2.2 GHz, 32 GB, Linux 6.6.56+, Python 3.11.11

TABLE IV

PERFORMANCES OF COMPRESSED NEURAL NETWORK (MOBILENETV3SMALL). IN BOLD ARE THE PERFORMANCES THAT ARE HIGHER OR EQUAL THAN THE STUDENT FINE-TUNING VERSION. THE "F1 DIFF" COLUMN INDICATES THE DIFFERENCE FOR THE F1-SCORE ABOUT THE COMPRESSED MODEL (FOR THE EDGE DEVICE) IN RELATION TO THE PERFORMANCE ACHIEVED WITH THE TEACHER MODEL (FOR A SERVER WITH MORE COMPUTATIONAL POWER).

Classes	Student Fine-tuning				Knowledge Distillation					Knowledge Distillation + Quantization				
	B.Acc.	Prec.	Recall	F1	B.Acc.	Prec.	Recall	F1	F1 Diff.	B.Acc.	Prec.	Recall	F1	F1 Diff.
Bottle	0.5	0.0	0.0	0.0	**0.973**	**1.0**	**0.947**	**0.972**	-0.028	**0.973**	**1.0**	**0.947**	**0.972**	-0.028
Cable	0.517	1.0	0.035	0.068	**0.964**	**1.0**	**0.928**	**0.962**	-0.019	**0.964**	**1.0**	**0.928**	**0.962**	-0.019
Capsule	0.5	0.0	0.0	0.0	**0.902**	**0.964**	**0.818**	**0.885**	-0.052	**0.902**	**0.964**	**0.818**	**0.885**	-0.052
Carpet	0.666	1.0	0.333	0.5	**0.925**	**1.0**	**0.851**	**0.92**	-0.08	**0.925**	**1.0**	**0.851**	**0.92**	-0.08
Grid	0.629	0.428	0.352	0.387	0.5	0.0	0.0	0.0	-1.0	0.529	**1.0**	0.058	0.111	-0.889
Hazelnut	0.928	1.0	0.857	0.923	**1.0**	**1.0**	**1.0**	**1.0**	+0.025	**1.0**	**1.0**	**1.0**	**1.0**	+0.025
Leather	0.75	1.0	0.5	0.666	**0.785**	**1.0**	**0.571**	**0.727**	-0.235	**0.767**	**1.0**	**0.535**	**0.697**	-0.265
Metal Nut	0.607	1.0	0.214	0.352	**0.928**	**1.0**	**0.857**	**0.923**	-0.077	**0.928**	**1.0**	**0.857**	**0.923**	-0.077
Pill	0.534	1.0	0.069	0.13	**0.854**	0.968	**0.72**	**0.826**	-0.112	**0.854**	0.968	**0.72**	**0.826**	-0.112
Screw	0.5	0.0	0.0	0.0	**0.912**	**0.885**	**0.861**	**0.873**	-0.112	**0.912**	**0.885**	**0.861**	**0.873**	-0.112
Tile	0.5	0.0	0.0	0.0	**1.0**	**1.0**	**1.0**	**1.0**	0.0	**1.0**	**1.0**	**1.0**	**1.0**	0.0
Toothbrush	0.699	0.8	0.444	0.571	**0.888**	**1.0**	**0.777**	**0.875**	+0.26	**0.888**	**1.0**	**0.777**	**0.875**	+0.26
Transistor	0.791	1.0	0.583	0.736	**0.921**	0.647	**0.916**	**0.758**	-0.198	**0.921**	0.647	**0.916**	**0.758**	-0.198
Wood	0.833	1.0	0.666	0.8	**0.972**	**1.0**	**0.944**	**0.971**	-0.029	**0.972**	**1.0**	**0.944**	**0.971**	-0.029
Zipper	0.513	1.0	0.027	0.05	**0.861**	**1.0**	**0.722**	**0.838**	-0.147	**0.861**	**1.0**	**0.722**	**0.838**	-0.147
Coffee Beans	0.5	0.0	0.0	0.0	**0.8**	**1.0**	**0.6**	**0.749**	0.0	**0.8**	**1.0**	**0.6**	**0.749**	0.0
Hazelnut	0.766	1.0	0.533	0.695	**0.933**	**1.0**	**0.866**	**0.928**	+0.928	**0.933**	**1.0**	**0.866**	**0.928**	+0.928
VAD	0.768	0.876	0.624	0.728	**0.789**	0.855	**0.696**	**0.767**	-0.008	**0.789**	0.854	**0.697**	**0.767**	-0.008

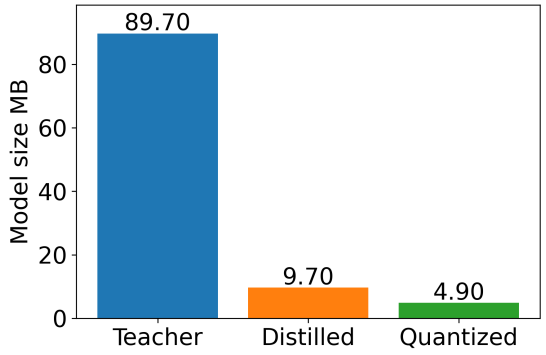

Fig. 8. Model size for teacher, student distilled, and distilled and quantized.

to see whether the compression techniques have improved the performances. Table IV shows the performances of the baseline (student fine-tuning), only Knowledge Distillation, and Knowledge Distillation followed by Network Quantization. We can notice that the **student fine-tuning baseline** for "bottle", "cable", "capsule", "carpet", "grid", "metal nut", "pill", "screw", "tile", "zipper", and "coffee beans" were not able to learn without the teacher's guidance, achieving a F1-score below of 0.5. Only "hazelnut" from the MVTecAD dataset achieved an F1-score higher than 0.9. These results express how much learning itself is impaired, especially in the identification of anomalies due to extremely low recall.

Taking into account the **distilled scenario**, with the exception of the "grid" class, the performances are superior to the baseline. The "grid" class has particularities in its pattern because it is totally related to the texture instead of more structural deformities of the objects. Also, we noticed that only the precision of the "transistor" drops from 1.0 to 0.647; however, the F1-score improves from 0.736 to 0.758 due to the increase in recall (0.444 to 0.777). In this scenario, Knowledge Distillation favors quality control by better identifying the anomalous behavior instead of the normal standard, requiring a second validation in the production line to validate the rejected normal items. These results attest to the efficiency of Knowledge Distillation in anomaly detection. Comparing the F1-score in relation to the respective teacher model (column "F1 Diff."), we notice a large loss for the "grid" and "leather" classes, once again being an anomaly due to changes in the texture of the objects. Other classes that suffer a little more are "pill", "screw", "transistor", and "zipper", ranging from 10% to 20%. Despite this, classes such as "hazelnut", "toothbrush", and "coffee beans" improved with the simplified model. An interesting fact is that "hazelnut" from the DML dataset can learn to distinguish normality patterns, while the respective teacher cannot. Applying **quantization after distillation**, the results practically remain the same, stating that neural networks can be quantized to reduce their computational storage without affecting their classification performance.

VI. CONCLUSIONS

In this study, we investigated neural network compression techniques combination to reduce computational cost and enable their deployment on edge computing devices in quality control for visual inspection for industries. Making computer vision models available at the edge devices ensures lower latency in time response and improved agility in the manufacturing process. Our results show that Knowledge Distillation, which reduces complexity by almost 90% for less complex models, ensures consistent classification performances that are equivalent to more complex models, especially for classes in which the anomaly is structural, such as "bottle", "cable", and "hazelnut". Texture anomalies suffer a little more, requiring more complex pipelines than just a neural network. By combining distillation with quantization, performances are

virtually lossless and computational complexity is reduced by almost 95% and 36x faster in time response. Hence, our experiments, on different objects and contexts, express that the combination of model compression techniques is a good approach for visual inspection control systems, requiring low computational cost and faster response to production line operators. For future investigations, we mention the improvement for texture anomalies, addressing the generation of synthetic defects, attention-based mechanisms, and post-processing to validate the outputs specific to each scenario.

REFERENCES

[1] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[2] J. Liu, G. Xie, J. Wang, S. Li, C. Wang, F. Zheng, and Y. Jin, "Deep industrial image anomaly detection: A survey," *Machine Intelligence Research*, vol. 21, no. 1, pp. 104–135, 2024.

[3] Z. Li, H. Li, and L. Meng, "Model compression for deep neural networks: A survey," *Computers*, vol. 12, no. 3, p. 60, 2023.

[4] M. A. Ponti, F. P. dos Santos, L. S. Ribeiro, and G. B. Cavallari, "Training deep networks from zero to hero: avoiding pitfalls and going beyond," in *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2021, pp. 9–16.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] A. Lopes, F. P. dos Santos, D. de Oliveira, M. Schiezaro, and H. Pedrini, "Computer vision model compression techniques for embedded systems: A survey," *Computers & Graphics*, vol. 123, p. 104015, 2024.

[7] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.

[8] Z. Liu, Y. Zhou, Y. Xu, and Z. Wang, "Simplenet: A simple network for image anomaly detection and localization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 20 402–20 411.

[9] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "Cutpaste: Self-supervised learning for anomaly detection and localization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 9664–9674.

[10] P. Mishra, R. Verk, D. Fornasier, C. Piciarelli, and G. L. Foresti, "Vt-adl: A vision transformer network for image anomaly detection and localization," in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*. IEEE, 2021, pp. 01–06.

[11] A. Aboah, "A vision-based system for traffic anomaly detection using deep learning and decision trees," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4207–4212.

[12] A. S. Khan, Z. Ahmad, J. Abdullah, and F. Ahmad, "A spectrogram image-based network anomaly detection system using deep convolutional neural network," *IEEE access*, vol. 9, pp. 87 079–87 093, 2021.

[13] H. S. Mputu, A. Abdel-Mawgood, A. Shimada, and M. S. Sayed, "Tomato quality classification based on transfer learning feature extraction and machine learning algorithm classifiers," *IEEE Access*, 2024.

[14] M. Rudolph, T. Wehrbein, B. Rosenhahn, and B. Wandt, "Asymmetric student-teacher networks for industrial anomaly detection," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2023, pp. 2592–2602.

[15] W. Zhang, C. Xie, and L. Xu, "Knowledge distillation-based multi-scale feature matching for industrial anomaly detection in intelligent manufacturing," in *2024 IEEE International Conference on e-Business Engineering (ICEBE)*, 2024, pp. 46–52.

[16] X. Chen, L. Cao, S. Zhang, X. Zheng, and Y. Zhang, "Breaking the bias: Recalibrating the attention of industrial anomaly detection," *arXiv preprint arXiv:2412.08189*, 2024.

[17] J. Chen and X. Ran, "Deep learning with edge computing: A review," pp. 1655–1674, 8 2019.

[18] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," pp. 869–904, 4 2020.

[19] J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, and W. Lin, "Thinet: Pruning cnn filters for a thinner net," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2525–2538, 2019.

[20] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[21] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320317304120

[22] F. P. dos Santos and M. A. Ponti, "Alignment of local and global features from multiple layers of convolutional neural network for image classification," in *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2019, pp. 241–248.

[23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[24] L. Jiao and J. Zhao, "A survey on the new generation of deep learning in image processing," *IEEE Access*, vol. 7, pp. 172 231–172 263, 2019.

[25] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, 2016, pp. 630–645.

[27] S. O. Haroon-Sulyman, S. S. Kamaruddin, F. K. Ahmad, S. Abdul-Rahman, and N. I. Aziz, "Techniques for mitigating the vanishing gradient problem in deep neural networks," in *Knowledge Management International Conference*. Springer, 2025, pp. 428–438.

[28] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[30] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, 6 2021.

[31] G. Hinton, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[32] T. Kim, J. Oh, N. Kim, S. Cho, and S.-Y. Yun, "Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation," *arXiv preprint arXiv:2105.08919*, 2021.

[33] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[34] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9592–9600.

[35] Karakurai Inc., "Real-world dataset for anomaly detection," 2025, acessado em: 23 de maio de 2025. [Online]. Available: https://adfi.jp/

[36] A. Baitieva, D. Hurych, V. Besnier, and O. Bernard, "Supervised anomaly detection for complex industrial images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 754–17 762.

[37] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009.

[38] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[39] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.

[40] D. Walsh, "Occam's razor: A principle of intellectual elegance," *American Philosophical Quarterly*, vol. 16, no. 3, pp. 241–244, 1979.

[41] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.