Journal of Ambient Intelligence and Humanized Computing (2020) 11:4491–4500
https://doi.org/10.1007/s12652-019-01640-4

ORIGINAL RESEARCH

Machine learning for quality control system

Gonçalo San-Payo[1] · João Carlos Ferreira[2] · Pedro Santos[1] · Ana Lúcia Martins[3]

Received: 30 November 2018 / Accepted: 9 December 2019 / Published online: 16 December 2019

Abstract
In this work, we propose and develop a classification model to be used in a quality control system for clothing manufacturing using machine learning algorithms. The system consists of using pictures taken through mobile devices to detect defects on production objects. In this work, a defect can be a missing component or a wrong component in a production object. Therefore, the function of the system is to classify the components that compose a production object through the use of a classification model. As a manufacturing business progresses, new objects are created, thus, the classification model must be able to learn the new classes without losing previous knowledge. However, most classification algorithms do not support an increase of classes, these need to be trained from scratch with all . Thus. In this work, we make use of an incremental learning algorithm to tackle this problem. This algorithm classifies features extracted from pictures of the production objects using a convolutional neural network (CNN), which have proven to be very successful in image classification problems. We apply the current developed approach to a process in clothing manufacturing. Therefore, the production objects correspond to clothing items

Keywords Quality control · Incremental learning · Image classification · Defect detection system

1 Introduction

Computer vision problems can be applied to quality control tasks, more precisely in defect detection and classification. There are many quality control systems of manufacturing processes that can be improved with the right use of machine learning algorithms, such as mobile phone cover glass production in Li et al. (2014), fabric production in Chan and Pang (2000), etc. Many machine learning algorithms can be used for image classification problems, but most of them have a fixed number of classes, and the algorithms cannot learn new classes incrementally. This can be a problem for applications and processes where new data and classes are created because it would require training the algorithm again from scratch with the old and new data together. The present work addresses this issue as it plays a major part in the proposed system. Quality control is a key factor in all major manufacturing businesses, as costumers and investors are increasingly demanding for higher quality. It is vital for a company to ensure that the number of defective products is kept to a minimum. Otherwise, it can have a big impact on the company's sales and business. Most of the quality control processes are still made by humans, and although these processes have improved over the years, human-based processes can lead to a few disadvantages. For example, a human usually works approximately 8 hours a day, and in some of those hours, the levels of concentration are not always the same. These levels of concentration may vary due to fatigue, lack of motivation and other factors that can lead to unnoticed defects and, therefore, hurt the business. A computer, however, can keep the same levels of "concentration" throughout the day. In the textile industry, where

✉ João Carlos Ferreira
jcafa@iscte-iul.pt

Gonçalo San-Payo
goncalo.san-payo@inov.pt

Pedro Santos
pedro.santos@inov.pt

Ana Lúcia Martins
almartins@iscte-iul.pt

1 INOV Inesc Inovação-Instituto de novas tecnologias, Lisbon, Portugal

2 Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR- IUL, Lisbon, Portugal

3 Instituto Universitário de Lisboa (ISCTE-IUL), Bussiness Research Unit (BRU-IUL), Lisbon, Portugal

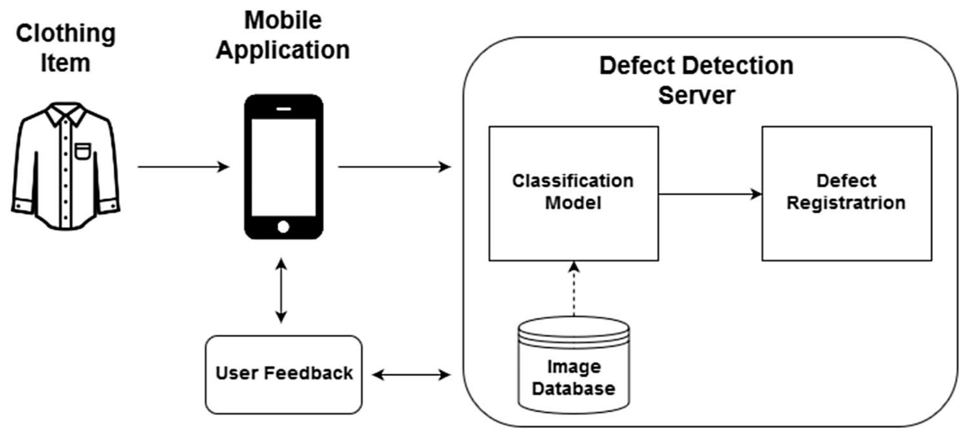


Fig. 1 QCSCM general architecture

humans are responsible for the quality control processes, only 70.

2 State of art

Quality control using machine learning techniques has been a hot research topic for a few years. Many techniques were used in this research topic, such as: Fourier analysis (Chan and Pang 2000), Gabor filters (Kumar and Pang 2002), neural networks (Celik et al. 2014). Additional work regarding the topic of quality control and defect detection can be found in Kumar (2008). Our objective is to develop a quality control system that detects defects in clothes. This system classifies the components of a clothing item and checks if they are correct, therefore our problem can be considered as an image classification problem. In more recent years, deep learning techniques have achieved state-of-the-art results in image classification problems with the development of a handful of neural network architectures in Krizhevsky et al. (2012), in Simonyan and Zisserman (2014), in He et al. (2016), in Howard et al. (2017), in Pratt et al. (1991). Most of the CNNs take a long time to train even on last-generation GPUs. However, there is a way to use the knowledge of a CNN gained when trained in a large dataset, like the ImageNet, and adapt it to a similar classification problem. This is called transfer learning, which consists of using a CNN with the parameters, weights and biases obtained when trained in a large dataset, use the first layers for feature extraction and replace the last layers (fully-connected layers) use for classification with new layers adapted to the desire classification task. This way there is only a need to train the new layers, which will save time and resources in Utgoff (1989). Incremental learning is the ability of an algorithm to gain knowledge with new unseen data. Many common classification algorithms have been adapted to this kind of learning, such as: decision trees in Lakshminarayanan et al. (2014), random forests (Polikar et al. 2001) and neural networks (Krizhevsky and Hinton 2009). Nalbach et al. (2018) also use a similar approach to quality assurance. Bray and Carpenter (2018) use also machine learning approach for image and a similar approach in Heleno et al. (2002) in a Machine Vision Quality Control System for Industrial and Ryu et al. (2010) in automatic quality control based on image. Zhu et al. (2011) also applied machine learning in agriculture for product quality and Kim (2019) in classification using a neural network approach.

3 Methodology

The purpose of the QCSCM is to detect defects on clothing items. This is achieved by using a classification model supporting incremental learning. This classification cation model can, however, be easily adapted to other contexts. The requirements of the system are as follows: (1) A system capable of detecting defects on clothing items using pictures. The system outputs a binary classification, defect or no defect, based on the classification of the clothing items components. (2) A mobile application to take pictures of the clothing items to be used by the quality control officers to perform their quality control tasks. The system is fed by the quality control officer using the mobile application. (3) Increase the speed of the quality control processes and the percentage of detected defects. For the system to be useful, it should improve the performance of the quality control processes. (4) The ability of the system to learn from new data as new components of clothing items are created. The system must learn new classes maintaining its previous knowledge. The quality control officer creates new data using the mobile application and feed the system in a collaborative way. A clothing item is made up by a set of components, such as buttons, pockets, stamps, etc. Therefore, a defect can be a wrong component or a missing component. Considering the requirements and the types of defect, the QCSCM architecture was defined in Fig. 1. Using a client-server

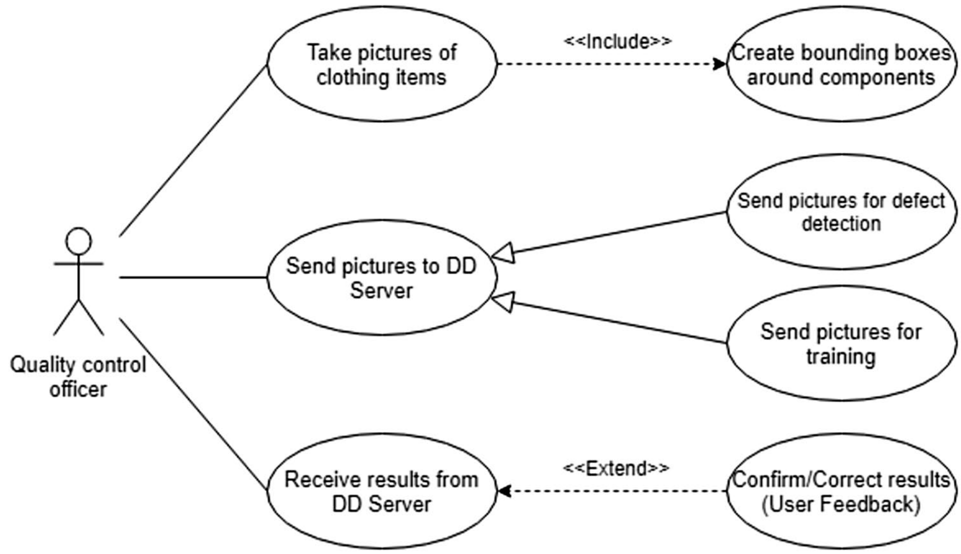


Fig. 2 Use case diagram of the quality control officer actions using the mobile application

model approach, the QCSCM consists of a mobile application and a server. We called Defect Detection Server (DD Server). The mobile application is used to take pictures of the clothing items, and the DD Server is responsible for detecting the defects making use of the classification model and finally, register the defects. To improve the QCSCM performance a user feedback approach was also defined. The responsibility of the quality control in the factory lies with a group of factory workers called quality control officers. The function of the quality control officers is to detect defects on the clothing items, register them and decide whether to send the clothing item back to the manufacturing process, remove the clothing item from production, or continue to the next production step. A clothing item is sent back to the manufacturing process if a repairable defect is detected and is removed from production if an unrepairable defect is detected. To execute their function the quality control officers use the mobile application to take pictures of the clothing items and create bounding boxes around the components that compose a clothing item. This information is sent to the DD Server that crops the content of the bounding boxes to create the images of the components. These images of the components are classified by the classification model, and the results are compared with the product data-sheet to see if there is a defect or not. Finally, the classifications of components are sent back to the mobile application be- ing used by the quality control officer. A product data sheet is an information associated with each model produced by the clothing factory. The product data sheets are defined by the clothing factory every time a new clothing item model is created. The information present in the product data-sheet information consists of a list of specifications and components that compose a clothing item The responsibility of creating images of the components to train the classification model also relies on the quality control officers. The quality control officer can also create more images by confirming or correcting the classifications it received from the DD Server, this is the user feedback feature. In Fig. 2 we defined a use case diagram that explains the actions the quality control officer performs using the mobile application.

A product data sheet is information associated to each model produced by the clothing factory. The product data sheets are defined by the clothing factory every time a new clothing item model is created. The information present in the product data sheet information consists of a list of specifications and components that compose a clothing item

The responsibility of creating images of the components to train the classification model also relies on the quality control officers. The quality control officer can also create more images by confirming or correcting the classifications it received from the DD Server, this is the user feedback feature. In Fig. 2 we defined a use case diagram that explains the actions the quality control officer performs using the mobile application.

3.1 Defect detection server

The first main component of the QCSCM is the DD Server responsible for feeding the classification model with images of the clothing items components. The DD Server must perform the following tasks:

1. Pre-process the images of the components it receives from the mobile application used by the quality control officers. This task of preprocessing the images consists of cropping the bounding boxes of the pictures taken by the quality control officers creating images of the components. These images of the components are then

resized and, in case of training, new images are created using data augmentation techniques. The pre-processing task is necessary so that the classification model can perform its tasks.
2. Predict the classes of the components. In this second task, the classification model present in the DD server predicts the classes of the components it received from the quality control officers.
3. Compare the results with the product data sheet and save the results. After the classifications are made the DD server performs the third task of comparing the results with the product data sheet. If the identified components match with the ones present on the product data sheet it means no defect was detected and nothing needs to be registered. If they do not match, it means a defect has been detected and the DD Server performs the defect registration.
4. Store pictures of the components and train the classification model with new data. This fourth and final task is only performed if a quality control officer selects the option of using the pictures to train the classification model. The DD Server after cropping the bounding boxes of the pictures taken by the quality control officers, saves the content of the bounding boxes (images of the components) along with the corresponding labels in a database. If enough images of the components are stored in the database, the training of the classification model is performed.

3.1.1 Image database

When a quality control officer sends pictures of clothing items with bounding boxes around the components and selects the option, in the mobile application, of using the pictures to train the classification model, the images of the components of the clothing items need to be stored. In this section we describe the image database represented in Fig. 1 as a module of the DD Server.

After the pictures of the clothing items and processed and the images of the components are created, the DD Server saves the images according to their classes. Each class has an associated directory where all images corresponding to that class are stored. The names of the directories serve as labels for the images when the classification model is trained.

This image database allows the creation of the dataset that is used to train the classification model. Every time the classification model needs to be trained, the DD Server loads the images and labels from the image database and feeds them to the classification model.

The image database also contains a list of the classes and the number of new images available from each class. This list is used to check if there are enough images to train the classification model and it is also sent to the quality control officers when they want to label the components of the clothing items using the mobile application.

3.1.2 Defect registration

The defect registration is represented in Fig. 1 as a module of the DD Server. It is performed after the classification model classifies the components that are sent to the DD Server and after the results of the classification are compared with the product data sheet to check if there are defects. In case of a positive defect detection, the type of the defect, missing component or wrong component, also needs to be registered. For example, let's assume we have a clothing item that is supposed to have three black buttons and one silver zipper, but the classification model returns two black buttons and one silver zipper. In this case the DD Server would register the defect as missing component along with the components that are missing, in this case a black button.

Another example using the same clothing item, the classification model returns three black buttons and a golden zipper. In this case the DD Server would register the defect as wrong component and register the mis- placed component, in this case a golden zipper instead of a silver zipper.

Apart from the image database and the defect registration the other main module of the DD Server is the classification model. However, due to its important we decided to describe the classification model in a separated section.

3.2 Mobile application

The reason of using a mobile application to take pictures instead of a fixed camera is because this way allows the quality control officers to walk around the factory and take pictures of the clothing items in different production steps.

During the creation of new data to train the classification model, after drawing bounding boxes around the relevant components in the picture, the quality control officers must label each component with the corresponding classes. The classes can be chosen from a list of existing classes or, if the object consists of class not present in the classification model, the quality control officers can create a new class that will be added to the list of existing classes.

During the defect detection process, after receiving the pictures taken by the quality control officers, the DD server sends back the results of the classification model—classified components—so that the quality control officers can give feedback on the classifications made. This interaction between the DD Server and the mobile application—user feedback—allows the quality control officer to correct wrong classifications made by the classification model of the DD Server and con- firm the correct ones.

After the corrections are made, the quality control officer sends the information again to the DD Server and new images are created to train the classification model.

3.3 Classification model

The proposed classification model is bundled inside the DD Server and is divided in a feature extraction model and a classifier with incremental learning abilities. Although in this work we used the classification model to classify components of clothing items, it can be adapted to other quality control environments.

The feature extraction model consists of a pre-trained InceptionResNET (a type of CNN model) that extracts important features from the content of the images. After the extraction, the features are classified by the classifier. We used a modified version of the Mondrian forest algorithm that supports incremental learning (Lakshminarayanan et al. 2014). We chose this architecture for the classification model, because by using the principles of transfer learning, we can combine the benefits of using a CNN to extract relevant information from an image with the ability of Mondrian forest to learn incrementally.

The idea of using a feature extraction model in the classification model was to make sure that the classifier only needs to process and classify relevant information and to reshape the input of the classification model from a three-dimensional array (an image) to a one-dimensional array that can be fed to the classifier. We chose to use a CNN as the feature extraction model because of the recent state-of-the-art results of this type of neural networks when it comes to image classification problems.

The function of the classifier is to classify the features extracted from the feature extraction model. As any other classification algorithm, the classifier present in the classification model needs to be trained with data relative to the classes it wants to classify. However, our classifier must be able to learn incrementally new classes and gain knowledge from unseen data.

A Mondrian forest is a type of random forest that can learn incrementally (Lakshminarayanan et al. 2014). The input of the Mondrian forest is a one-dimensional array, therefore, it is able to train with the feature arrays extracted using the feature extraction model. In the next chapter we detail how we developed the classification model and how our classifier (Mondrian forest) behaves when classifying the feature arrays extracted using different CNN architectures.

Table 1 Comparison of CNNs features classified with Mondrian forest

Number of classes	Inception	Resnet	InceptionResnet	MobileNet	VGG16
5	0.85	0.86	0.91	0.79	0.77
6	0.80	0.81	0.87	0.71	0.69
7	0.77	0.79	0.85	0.68	0.67
8	0.75	0.77	0.84	0.67	0.64
9	0.74	0.76	0.83	0.65	0.62
10	0.72	0.76	0.83	0.63	0.60

4 Experience

To choose which CNN to use in the final version of the classification model, we performed some experiments on some of the architectures provided by the Keras library. The chosen architectures were: VGG16,[1] MobileNet-V1,[2] Inception-V3,[3] ResNet50[4] and InceptionResnet-V2[5].

In order to set some baseline results and due to the lack of real images of components of clothing items, we used the Cifar-10[6] dataset (A. Krizhevsky and Hinton 2009) to perform some experiments and check if the classification model can perform well in an image classification problem. The Cifar-10 dataset consists of 60,000 images in 10 classes, with 6000 per class. Of these images, 50,000 are used for training and 10,000 are used for test. Each image consists in a 32×32 color image. The 10 classes are the following: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

We created a python script using several libraries such as: Google TensorFlow, Keras, Numpy and OpenCV, to train the classifier on features extracted from the Cifar-10 dataset using each of the selected CNN architectures in an incremental fashion, first we trained it with 5 classes and the we added classes progressively until the classifier was trained for all 10 classes of the dataset and measured the accuracy. The number of Mondrian trees of Mondrian forest

[1] CNN model architecture created by VGG (Visual Geometry Group, University of Oxford) for the ILSVRC-2014 contest.

[2] MobileNetV1 from Google is a CNN model particularly useful for mobile and embedded vision applications.

[3] CNN model that is the first runner up for image classification in ILSVRC-15.

[4] CNN model that won the first place in the ILSVRC-15 classification competition with top-5 error rate of 3.57%

[5] State of the art CNN model architecture combining ResNet and Inception features.

[6] The CIFAR-10 dataset (Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms.

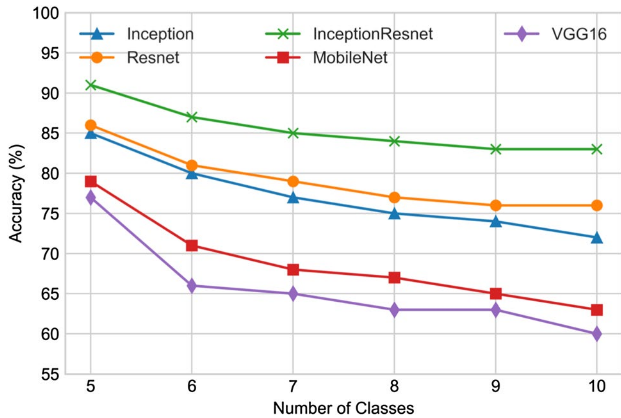


Fig. 3 Comparison of CNNs features classified with Mondrian forest (applied to Test dataset)—graph

was set to 100. We used this number of trees because it is a common value used in decision forests (Lakshminarayanan et al. 2014).

As the Table 1 and Fig. 3 show, for all CNN architectures, the accuracy decreases when new classes are added. The InceptionResnet shows the best results, followed by the Resnet and the Inception. Furthermore, the classifier trains faster on the InceptionResnet features than on the Resnet or Inception features, this is because the InceptionResnet returns a feature array of size 1536, which is smaller than the 2048 size array of both the Resnet and Inception. Although the training of the classifier with the features of the VGG16 and MobileNet was significantly faster than the training with the InceptionResNet features, the accuracies were much worst. Taking these results into account we chose to use the InceptionResnet CNN as our feature extraction model in the following experiments.

In the original implementation of the Mondrian forest when initialising the model, a series of data related parameters must be defined, such as, the number of classes of the data, the training and test data and its corresponding labels. In the implementation developed in the present work, these parameters are also defined, but after each training session, the number of classes used in that session is saved in the model so that the model can accommodate the new classes. To see how the classifier performed after the modifications we made to the original implementation of the Mondrian forest, we experimented training the classifier incrementally with new classes and training the classifier with new classes from scratch. After the experiments

Table 3 Comparison of classification model accuracies trained from scratch and trained incrementally

Number of classes	Total training	Incremental training
5	0.91	0.91
6	0.88	0.87
7	0.86	0.85
8	0.86	0.84
9	0.86	0.83
10	0.85	0.83

Table 2 Confusion matrix—Class legend: 1. zipper-white; 2. zipper-silver; 3. zipper-black; 4. button-grey; 5. button-black; 6. button-bronze; 7. button-white; 8. button-yellow; 9. button-blue, 10. button-red; 11. belt buckle-gold; 12. belt buckle-silver; 13. belt buckle-black; 14. pocket-yellow; 15. pocket-red; 16. stamp1; 17. stamp2; 18. stamp3

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	19	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	2	18	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	14	0	0	5	1	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	20	0	0	0	1	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1	21	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22

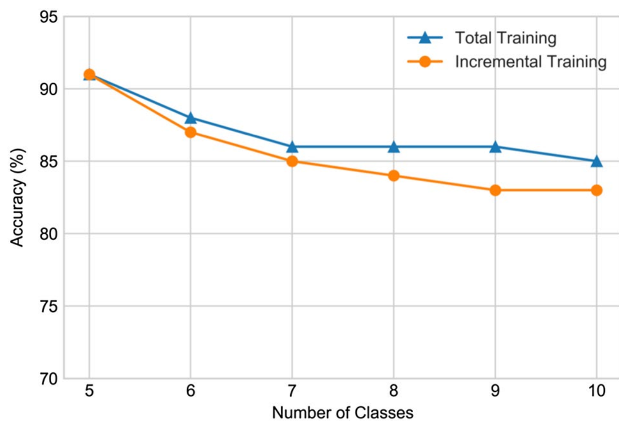


Fig. 4 Total training vs incremental training—graph

we compared the accuracies of both training methods in Tables 2 and 3.

As we can see in Fig. 4, the difference between the two training methods is not big, with just a small drop, of around 1–2%, in accuracy when trained incrementally compared to training with all classes from scratch. These results show that the classifier can be trained incrementally in a satisfactory way, which is important for the QCSCM.

In this section we evaluate our proposed classification model used in the QCSCM, ensuring that it fulfills the propose of the present work.

In following evaluation and experiments we put ourselves in the position of the quality control officers and used the developed system, more precisely the mobile application to take pictures of clothing items and create bounding boxes of the components. The pictures were sent to the DD Server that stored the images of the components along with the labels in the image database creating a custom dataset. This dataset consists of around 2100 images divided in 18 classes.

The classification model must be an efficient tool in order to be a valid option for the QCSCM and for the quality control officers in their quality control processes. To measure how efficient the classification model is, we calculated some classification metrics using the custom dataset.

Previously, we used accuracy as the metric to evaluate the incremental learning abilities of the classification model. The results were promising, but the use of this metric can be misleading sometimes. In this section, we evaluate the performance of the classification model using more metrics. The classification model was trained with all 18 classes of the dataset we created.

Using the training set of the dataset and all 18 classes we created a python script to train the classification model and then evaluated the model using the test set. In Table 2 we can see a confusion matrix describing the performance of the classification model on the test set.

Table 4 Precision, recall and F1-score

Class	Precision	Recall	F1-score
1	0.90	0.90	0.90
2	0.90	0.86	0.88
3	0.96	1.00	0.98
4	0.93	0.67	0.78
5	1.00	0.95	0.98
6	1.00	1.00	1.00
7	0.81	1.00	0.90
8	0.95	0.95	0.95
9	0.92	1.00	0.96
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	0.95	1.00	0.98
13	1.00	0.95	0.98
14	1.00	1.00	1.00
15	1.00	1.00	1.00
16	1.00	1.00	1.00
17	1.00	1.00	1.00
18	1.00	1.00	1.00

Table 5 Evaluation metrics

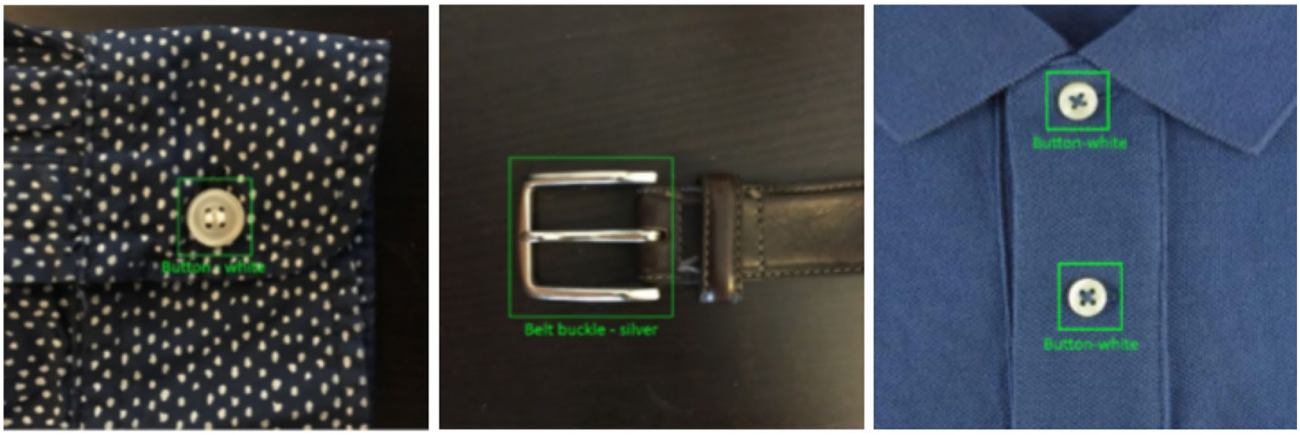
Accuracy	Precision	Recall	F1-score
96.09%	96.29%	96.04%	95.96%

With the help of the confusion matrix it is possible to calculate the precision, the recall and the F1-score. These metrics allow a better interpretation of the classification model performance. To calculate these metrics, we use the scikit-learn library and the information shown in the confusion matrix. The results of these calculations are present in Table 4. As shown in this table, the metrics are high across all classes except for class number four, which has a lower recall, and class number seven, which has a lower precision.

In the case of class number four, which is button-grey, the high precision and low recall implies that the classification model does not classifies many things as button-grey, missing a lot of them. However, when it classifies an object as button-grey it is very precise.

As for the case of class number seven, button-white, the high recall but lower precision implies that the classification model correctly classifies a significant proportion or even all the white buttons as button-white. However, it also incorrectly classifies other classes as button-white.

These results show that the classification model found it more difficult to distinguish the classes with similar characteristics. Since the number images per class is quite balance, we can average the results of each class and get the overall precision, recall and F1-score. The overall metrics,



(a) Button-White (b) Belt Buckle-Silver (c) Button-White

Fig. 5 Examples of correct classification

converted to percentages, along with the accuracy of the classification model is presented in Table 5.

4.1 QCSCM simulation

To further evaluate the classification model and to test the QCSCM, we experimented the QCSCM by taking some pictures of clothing items. Some of these pictures are presented here, where we can see how the QCSCM performed on them.

To take these pictures, we installed the developed mobile application in three mobile devices and created a simulated environment over a period of 1 week. The three installed mobile applications allowed us to put ourselves in the role of quality control officers.

By installing the mobile applications in multiple devices in the simulated environment we created, we were capable creating more images to be used by the QC- SCM in a collaborative way. All of the installed mobile applications were capable of connecting to the DD Server allowing a faster creation of images and subsequently a better training of the classification model.

In Fig. 5 it is possible to see some examples of correct classifications. On the left, a picture of a shirt sleeve with a bounding box around a component correctly labeled as button-white. On the middle, a picture of part of a belt with its buckle surrounded with a bounding box correctly classified as silver belt buckle. On the right, a picture of a polo shirt with two bounding boxes correctly classified as white buttons.

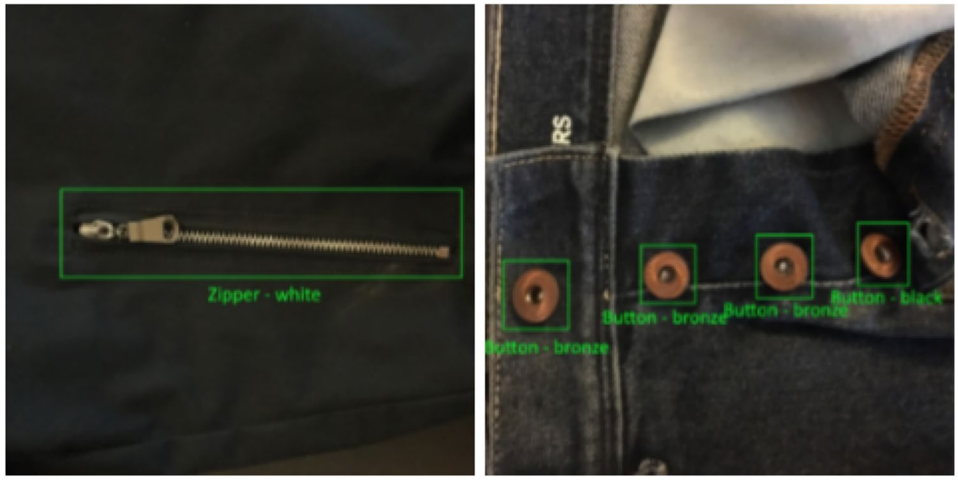
As the Fig. 5 also shows, the QCSCM can use the classification model to classify more than one component at a time. The picture on the right has two bounding boxes correctly classified.

In the real quality control environment, the quality control officers when receiving results such as the ones present in the figures above, could confirm the results and create new images for training with them. As for the DD Server, it would register a defect in case of one being detected.

As seen in previously the classification model is not 100% accurate, sometimes it makes wrong classifications of clothing items components. Figure 6 shows some of these cases. On the left, we can see a silver zipper mistakenly classified as a white zipper. On the right, it is possible to see four bronze buttons, three of them correctly classified but one incorrectly classified as a black button.

Some important information can be retrieved from these examples of incorrect classifications. In these examples the classification incorrectly classified the components, however the main characteristic of the components was correctly classified. In the case of the silver zipper, the component was correctly classified as a zipper, but the color was incorrect. The same for the buttons example, all of them were classified as buttons, but in one of them the color was incorrect. This suggests that some class hierarchy and multi-label classification could improve the performance of the classification model, since the are many components that shared some characteristics.

As said before, when the quality control officer receives incorrect results, he should make use of the user feedback feature of the QCSCM and correct wrong predictions made by the classification model. This will help the classification model improve its accuracy.



(a) Button-White (b) Belt Buckle-Silver

Fig. 6 Examples of incorrect classifications

5 Conclusion

The goal of the present work was to develop a system, that makes use of an image classification model capable of learning new classes incrementally and increase its knowledge, to help the quality control officers of a clothing factory in their quality control processes.

Using a mobile application combined with a server for central processing, the proposed QCSCM system is deployed containing a classification model created using a set of machine learning algorithms. This system can classify objects that are part of clothing items, checking if the identified objects corresponds to the reference used on a certain clothing item and also, it allows the use of machine learning algorithms applications by multiple factory workers through the use of a mobile application. At the moment, the system is applied to the clothing manufacturing but others cases and other type of productions lines can also be used.

This work also addresses transfer learning, but with a little twist. Instead of replacing the last layers of a CNN with new layers adapted to the new classes, it uses an independent and autonomous machine learning algorithm to classify the features extracted from the CNN to learn new classes incrementally.

In the current architecture of the classification model, each different component of a clothing item corresponds to a different class. The same is applied to other produced objects. If the number of classes increases exponentially this can lead to some drops in accuracy. Also, some classes of objects can be more difficult to classify than others. Taking this into account, the focus will be to create a class hierarchy and multi-label classification to create a newer version of the system. For example, the current classification model classifies a black button and a blue button as two different classes. In the future we will develop a classification model that first classifies the more generic class, such as button, zipper, pockets, etc., and then classifies its characteristics, for example, color, size, etc. in order to reach the final classification for the object.

Funding This research has been supported by Portuguese National funds through FITEC - Programa Interface, with reference CIT "INOV - INESC Inovação - Financiamento Base".

References

Bray MA, Carpenter AE (2018) Quality control for high-throughput imaging experiments using machine learning in cellprofiler. In: Johnston P, Trask O (eds) High content screening. Methods in molecular biology, vol 1683. Humana Press, New York

Celik H, Dulger L, Topalbekiroglu M (2014) Development of a machine vision system: real-time fabric defect detection and classification with neural networks. J Text Inst 105(6):575–585

Chan C, Pang GK (2000) Fabric defect detection by fourier analysis. IEEE Trans Ind Appl 36(5):1267–1276

Heleno P, Davies R, Correia (2002) EURASIP. J Adv Signal Process 2002:298501. https://doi.org/10.1155/S1110865702204114

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861

Kim H (2019) Multiple vehicle tracking and classification system with a convolutional neural network. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-019-01429-5

Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images, University of Toronto

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 1097–1105

Kumar A (2003) Neural network based detection of local textile defects. Pattern Recogn 36(7):1645–1659

Kumar A (2008) Computer-vision-based fabric defect detection: a survey. IEEE Trans Industr Electron 55(1):348–363

Kumar A, Pang GK (2002) Defect detection in textured materials using gabor filters. IEEE Trans Ind Appl 38(2):425–440

Lakshminarayanan B, Roy DM, YW (2014) The efficient online random forests. Advances in neural information processing systems, Mondrian forests, pp 3140–3148

Li D, Liang LQ, Zhang WJ (2014) Effect inspection and extraction of the mobile phone cover glass based on the principal components analysis. Int J Adv Manuf Technol 73:9–12

Nalbach O, Linn C, Derouet M, Werth D, (2018) Predictive quality: towards a new understanding of quality assurance using machine learning tools. In: Abramowicz W., Paschke A (eds) Business information systems. BIS, (2018) Lecture Notes in Business Information Processing, vol 320. Springer, Cham

Polikar R, Upda L, Upda SS, Honavar V (2001) Learn++: an incremental learning algorithm for supervised neural networks. IEEE Trans Syst Man Cybern part C (applications and reviews) 31(4):497–508

Pratt LY, Mostow J, Kamm CA, Kamm AA (1991) Direct transfer of learned information among neural networks. AAAI 91:584–589

Ryu JY, Kim SW, Kim SU, Ha DH (2010) Automatic image quality control system. In: Kim T, Vasilakos T, Sakurai K, Xiao Y, Zhao G, Slezak D (eds) Communication and networking. FGCN 2010. Communications in computer and information science, vol 120. Springer, Berlin

Simonyan AK, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition, arXiv: 1409.1556

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9

Utgoff PE (1989) Incremental induction of decision trees. Mach Learn 4(2):161–186

Zhu Y, Li S, Liu S, Yue E (2011) Design of agent-based agricultural product quality control system. In: Li D, Liu Y, Chen Y (eds) Computer and computing technologies in agriculture IV. CCTA (2010) IFIP advances in information and communication technology, vol 344. Springer, Berlin, Heidelberg