TransferD2: Automated Defect Detection Approach in Smart Manufacturing using Transfer Learning Techniques

Atah Nuh Mih*, Hung Cao*, Joshua Pickard†, Monica Wachowicz*‡, Rickey Dubay§
* *Analytics Everywhere Lab, University of New Brunswick, Canada*
† *Eigen Innovation Inc.*
‡ *RMIT University, Australia*
§ *Department of Mechanical Engineering, University of New Brunswick, Canada*

2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS) | 979-8-3503-4647-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/COINS57856.2023.10189312

***Abstract*—Quality assurance is crucial in the smart manufacturing industry as it identifies the presence of defects in finished products before they are shipped out. Modern machine learning techniques can be leveraged to provide rapid and accurate detection of these imperfections. We therefore propose a transfer learning approach, namely TransferD2, to correctly identify defects on a dataset of source objects and extend its application to new unseen target objects. We present a data augmentation technique to generate a large dataset from the small source dataset for building a classifier. We then integrate three different pre-trained models with different network depths (Xception, ResNet101V2, and InceptionResNetV2) into the classifier network and compare their performance on source and target data. We use the classifier to detect the presence of imperfections on the unseen target data using pseudo-bounding boxes. Our results show that ResNet101V2 performs best on the source data with an accuracy of 95.72%. Xception performs best on the target data with an accuracy of 91.00% and also provides a more accurate prediction of the defects on the target images. The results also indicate that the choice of a pre-trained model is not dependent on the depth of the network. Our proposed approach can be applied in defect detection applications where insufficient data is available for training a model and can be extended to identify imperfections in new unseen data.**

***Index Terms*—Transfer Learning, Smart Manufacturing, Defect Detection, Deflectometry Data, Data Augmentation, Product Quality Assurance**

I. Introduction

The manufacturing industry is rapidly shifting to the new revolution wave (e.g., Industry 4.0 in Germany, Industrial Internet in the US) with many disruptive technologies that support effective and accurate engineering decision-making through the convergence of a vast amount of networked data and emerging technologies such as Internet of Things, Cyber-Physical System, Edge Computing, Advanced Analytics and AI for governing and operating manufacturing operations. Many companies today are optimizing their systems by adopting more automated systems in their product manufacturing processes in preference to using human resources. Product quality assurance is one of the stages that are being drastically transformed into automation in the manufacturing process. With the appearance of advanced techniques in machine vision, many product imperfections can be quickly and precisely detected without the task of judging by the human eye. In some cases, conventional machine vision systems can increase manufacturing efficiency and provide more accurate quality inspection with the support of several special sensors. Some examples of surface defect detection applications are Printed Circuit Board (PCB) defect detection [1], leather defect classification [2], and steel defect detection [3].

In this work, we aim to solve complex quality inspection problems within the manufacturing industry through the application of customized deep learning models that leverage machine vision and process data, combined with advanced image processing and data standardization pipelines, to predict the existence and location of relevant quality defects. Using the deflectometry technology developed by our industry partner, a wide range of objects with specific defects can be detected in multiple applications. Figure 1 depicts several examples of paint inspection images applied to this technology to different objects.

Fig. 1. Paint inspection images obtained with deflectometry inspection technology for three different objects.

Although the current inspection technology has been proven to work, it is inefficient when trying to scale an existing solution to new applications with new objects. A primary challenge when developing solutions for new applications, which may include new part geometries, new camera perspectives, additional defects, and other process changes, is ensuring machine learning (ML) model robustness and reliability without the need for collecting excessively large labeled datasets that deter adoption and prolong return on investments. To overcome this challenge, it is necessary to have a solution that does

not require too much labeled input data but still ensures the accuracy of the trained ML model when applied to the defect identification of new objects. This requires leveraging the results of the previous solution to transfer knowledge to a new and similar application (ie. model architecture, trained models, labeled datasets). A relevant example of this challenge is paint inspection where the inspection images have well-defined features associated with specific defects (eg. scratches, contamination, paint runs, etc.) but the wide range of part geometries, colors, material, and variation in part placement present difficulties in successfully scaling a solution to new applications with new objects.

The primary objective of this research project is to identify the relevance of leveraging transfer learning techniques and synthetic data approaches to minimize the necessity for large labeled datasets at the start of each new application on new objects. By using transfer learning, we aim to get a high accuracy level on the imperfection detection of new objects when we only use the learned knowledge and a limited amount of data from the initial objects.

The main contributions of this work are as follows:

1) We propose an end-to-end approach, namely Transfer Defect Detection (TransferD2) to leverage different transfer learning models through an architecture design for product defect detection applications in the context of smart manufacturing.
2) We devise a method to process the original small dataset and augment the data records to produce appropriately large labeled datasets that are suitable for training robust ML models.
3) We also experiment with different sets of transfer learning models representing 3 depth levels of neural network architecture (i.e., network depth level $\sim$ 100, $\sim$ 200, $\sim$ 400) and compare the performance of these models.

The remainder of this paper is as follows. Section II describes the related work of product defect detection in smart manufacturing applications and the state-of-the-art transfer learning techniques. Section III presents our proposal for product quality assurance and imperfection detection leveraging transfer learning methods. Section IV realizes our proposed approach through the implementation of a practical application on a real dataset. The evaluation and performance of our proposed approach are discussed in Section V. Finally, we conclude our paper in Section VI.

II. Related Work

This section discusses the state-of-the-art works related to the imperfection detection application in the manufacturing industry. We also discuss several related transfer learning approaches in the context of smart manufacturing.

A. Smart Manufacturing

Extensive research has been done on integrating deep learning techniques in various areas of smart manufacturing, such as product quality inspection, fault diagnosis, and defect prognosis [4]. In our paper, we explore product quality inspection and focus on surface defect detection. This research area leverages image processing techniques and computer vision to classify and localize defects on the surface of materials.

One of the major challenges faced by deep learning in surface defect detection has been the lack of large amounts of training data necessary for training deep learning models [5]. Several authors have explored various approaches to deal with this challenge. For example, Tabernik et al. [6] proposed a segmentation-based architecture for the detection and segmentation of surface defects. In their method, each pixel of the surface defect was considered a training sample, thereby increasing the amount of training data available. In another work [2], the authors had 27 sample images of leather surface defects. They proposed a method to split each image into 24 smaller ones but manually selected images such that each image only contained one defect. Another research [3] proposed generating synthetic data for steel defect detection using rendering software to re-create the images. The images were shifted to be viewed from different angles, and shader parameters were altered to vary the defect shapes and locations. Jain et al. [7] also proposed using a Generative Adversarial Network to generate synthetic data for surface defect detection. They used a generator network to produce realistic synthetic images, which were then used to train their classifier.

The nature of modern manufacturing systems provides flexible configurations for producing a wide variety of products. Using traditional machine learning methods poses a challenge as the feature representations will need to be redesigned from scratch. To overcome the challenge of domain shift, several transfer learning techniques have been proposed.

B. Transfer Learning

Transfer learning has emerged as a popular deep learning approach in which knowledge from a *source domain* is applied to a *target domain* to improve training performance. This approach has become popular in applications in which extensive training data are not available to build a model from scratch. By using transfer learning, the parameters from the pre-trained model can be adjusted to suit the new domain which provides faster convergence than would have been achieved from random weight initialization when training from scratch.

A popular source domain for transfer learning has been the ImageNet dataset. Many deep learning models developed for this dataset have become base models for transfer learning in various applications such as in garbage classification [8], fault detection in rail components [9], and fault diagnosis [10].

Specific to smart manufacturing, several frameworks have also been proposed that build upon transfer learning. Zhu et al. [11] used transfer learning to build a model that could detect bridge defects in a dataset of images not used during training. They used InceptionV3 as a feature extractor and built a classifier network on top of the base model. Gong et al. [12] proposed a deep transfer learning method to identify defects in aeronautics composite materials. They trained the labeled images of their source and target data together by feeding

image pairs consisting of both images from both datasets as image pairs, thereby providing sufficient input data for training their model.

The choice of the base model when building a classifier for a new dataset plays an important role [5]. Abu et al. [13] did a comparative study on the performance of ResNet, VGG, MobileNet, and DenseNet on the SEVERSTAL and NEU datasets. They used transfer learning to train the models on these datasets and found MobileNet to be the best-performing model. In another study with a different dataset [14], the researchers found Xception to be the most performant model on the Environmental Microorganism Image Dataset (EMDS-6) dataset. Another work [5] chose ResNet-18 as the base model of their study due to its relatively simple implementation and high performance. This proves that there is no clear metric in selecting the base model for implementation.

III. Proposed Method

A. Overview approach

The detection of surface defects can be considered a binary image classification problem [6], in which the image analyzed is either defective or non-defective. Emphasis should therefore be laid on identifying features that enable the model to correctly classify an image as defective or not. This implies irrelevant features such as the object's shape, size, and color should be given less importance in the classification of the image. As such, a model trained to correctly identify surface defects on a specific object will therefore have a significant performance when identifying defects on a new object, irrespective of shape, size, or color.

We propose an approach, as shown in Figure 2, that is divided into two phases: a classification phase and a detection phase. In the classification phase, we build a binary classifier that identifies if an image is defective or not. The detection phase then uses the classifier to run inference on new data and identify the defective segments of the image. We explain both phases in detail in the following subsections.

B. Classification Phase

The execution of an accurate classifier is the most crucial step in our proposed approach. The classification phase is further broken down into Data Pre-processing, Dataset Augmentation, and Training and Building of Learning Models stages.

1) Data Pre-processing: Data pre-processing provides the basis for building the classifier's dataset. In this stage, each image in the source dataset is split into smaller tiles, and the tiles are labeled as defective or non-defective.

The image is split into a grid of *m* columns and *n* rows. Tiles are labeled as either defective or non-defective by verifying if they contain a defect. If the coordinates of the defect on the original image overlap or lie within the coordinates of the tile, that tile is marked as defective. Otherwise, the tile is marked as non-defective. The *Label*, *Source Image ID*, and *x* and *y* values of the tile are used to save the tile as an independent image in the new dataset (where *Label = 0* for non-defective and *Label = 1* for defective), thereby providing a unique file name for each tile. Algorithm 1 presents the procedure to pre-process all images in the source dataset, *S* and generate a new augmented dataset, *E* by cropping the original images and splitting them into labelled tiles. Specifically, each original source image in the pre-processing stage is split into a grid of *m* columns and *n* rows, and each tile is saved as a new image. This implies that each original source image produces *m*n* tiles that are saved as independent images.

Algorithm 1 Dataset Pre-processing

Input: Source Dataset, S
Output: Augmented Dataset, E

Let E be an empty dataset
for $img \in S$ **do**
  $img \leftarrow crop(img)$
  for $i \leftarrow 0, m$ **do** ▷ m columns
    for $j \leftarrow 0, n$ **do** ▷ n rows
      $tile \leftarrow img[i : j]$
      if tile has defect **then**
        $label \leftarrow 1$
        $E \leftarrow tile, label$
      else
        $label \leftarrow 0$
        $E \leftarrow tile, label$
      end if
    end for
  end for
end for

Consider a source dataset of *N* images. After pre-processing all the images in the source dataset, we obtain a new dataset of size *E = N*m*n* images and their corresponding labels (0 for non-defective and 1 for defective). The detailed implementation for this procedure is explained in Section IV-B.

The classifier's performance depends on its identification of crucial features of the image, which include the defect itself and its surroundings. The classifier is trained to learn features that identify if a tile is defective or not rather than learning the general features of the object itself. Hence, splitting the image into tiles provides a basis for the generalization of the classifier on a dataset with unseen and unlabelled images.

2) Dataset Augmentation: Defects occupy a small area of the image, and defective images will, therefore, provide a significantly smaller percentage of the images in the new dataset E. Such data imbalance will result in poor model performance, as stated in [8], as training will be biased towards the class with a more significant number of samples [15]. Several methods have been proposed to solve the data imbalance problem, such as an even-odd mechanism to balance the defective and non-defective samples [6]. In [16], the authors evaluated the performance of classifiers for undersampling and oversampling and observed that oversampling had better results.

We therefore use an even-odd mechanism to select images and oversample the class of defective images randomly for better results. Algorithm 2 illustrates our approach to obtain

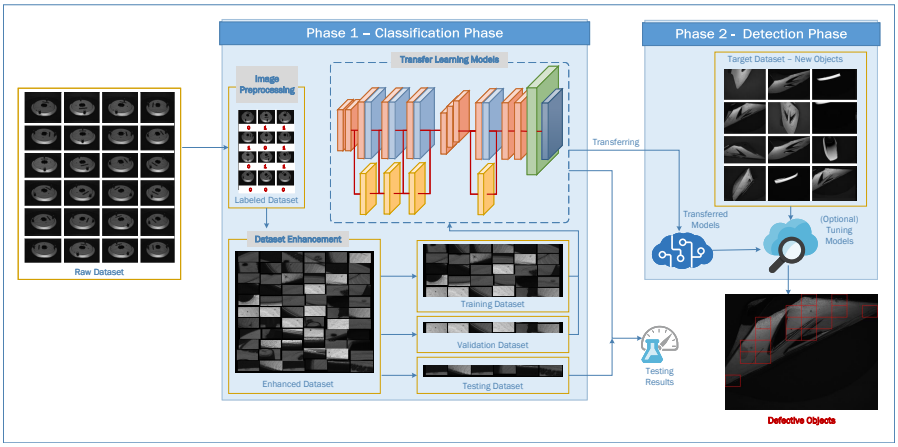


Fig. 2. The proposed methodology for product imperfection detection using Transfer Learning.

a balanced dataset D from an augmented dataset E in the previous stage. At even iterations, random defective samples are selected and randomly rotated or flipped to avoid repetitive images that could lead to overfitting. The same approach is used to select random non-defective samples at odd iterations. This process results in the oversampling of defective images and the undersampling of non-defective images. Extensive training data is required to tune the parameters of the transfer learning model; therefore, we iterate the data balancing process over the size of the imbalanced dataset to produce a balanced dataset of the same size. The resulting dataset is evenly balanced with an equal number of defective and non-defective samples. The images are assigned to their respective classes, that is, 0 for non-defective samples and 1 for defective samples, and the dataset is then split into the training and validation datasets.

Algorithm 2 Dataset Augmentation

Input: Augmentation Dataset, E
Output: Balanced Dataset, D

Let D be an empty dataset
Let class 0 be an empty set of non-defective tiles
Let class 1 be an empty set of defective tiles
for $index, tile, label \in E$ **do**
    if index is even **then**
        select tile with label = 1
        randomly rotate or flip tile
        assign tile to class 1
    else
        select tile with label = 0
        randomly rotate or flip tile
        assign tile to class 0
    end if
end for
$D \leftarrow Class0 + Class1$

3) Training and Building of Learning Models: Transfer learning has been proven to be beneficial over training a model from scratch [17] by making use of predefined weights from the pre-trained model. These predefined weights are then adjusted to learn features on a new dataset [17], thereby saving computational time required to train a model from scratch with random weight initialization.

Our proposed approach uses transfer learning to build the classifier network on top of a model whose weights are pre-trained on the ImageNet dataset. The classifier network consists of an average pooling layer, a dropout layer, and a dense output layer with two neurons (for binary classification) and a sigmoid activation function. To obtain better performance, the entire network is trained end-to-end, and binary cross entropy is used as the loss function.

C. Detection Phase

The saved transfer learning models are then loaded for inference on the target dataset containing unlabelled images of different objects. The classifier is trained to classify tiles from input images and not the entire image itself. As the classifier is built using a sigmoid function, it returns prediction values within the range of 0 to 1. The closer the prediction is to 0, the more likely it is to be a non-defective image, and the closer the image is to 1, the more likely it is to be a defective image. This implies that prediction values greater than 0.5 are considered defective and vice versa. To prevent the mislabelling of defective tiles, the prediction threshold should be kept suitably higher than 0.5.

A sliding window is used to split the image of the new object from the target dataset into a grid of tiles, and each tile is input into the classifier for prediction. The result of the prediction is then used to determine if the tile is defective or not, as described above. Defective tiles are then flagged and bounded by a rectangle on the input image, thereby providing pseudo-object detection.

IV. Implementation

A. Deflectometry and data acquisition

Deflectometry inspection systems use synchronized patterned lighting and image capture, combined with image

processing and deep learning, to inspect high gloss parts for surface and subsurface defects. Here, 16 black and white patterned images are shown on a digital display, a Flir BFS-PGE-27S5M-C monochrome camera captures the 16 corresponding images of the pattern reflections on the part surface, and deflectometry image processing generates a single processed image that clearly shows the defects. This inspection is repeated multiple times for each unique part and the resulting processed images are annotated to identify all scratch and dirt/contamination defects visible in the image.

The resulting dataset contains 227 black and white images of a source object and defect annotations that define the type of defect and their bounding boxes. As mentioned in Section III-A, we focus on a binary classification problem and therefore ignore the different classes of defects.

B. Data Pre-processing

The source dataset contains images and the corresponding defect annotations. As seen from Figure 3, the object is bounded within a black background, which is irrelevant for the classifier. Canny edge detection provides a better method of background removal over fixed cropping due to the varying shape of the objects in the image. With canny edge detection, an image box is dynamically identified based on the object's edges, and the resulting box is used to crop the image. In contrast, cropping with a fixed box fails to take into consideration the dimensions of the object and crops out parts of the object that fall outside the prescribed cropping range.

The image obtained from cropping has different dimensions from the original image, and therefore, the annotations must be mapped to the new image. The location of pixels on an image is referenced from the image's origin coordinates. Cropping does not modify the distance of the pixels from the origin, but rather re-assigns a new point of origin. Therefore, the coordinates of the annotated pixels can be mapped to the new image by calculating new coordinates based on the cropped image's new point of origin. The new image's defects are now mapped from the re-assigned annotation values.

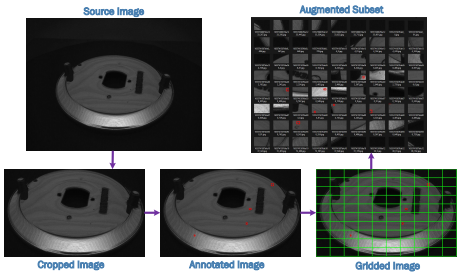


Fig. 3. The proposed method for dataset augmentation

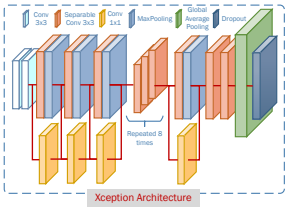
C. Dataset Augmentation

For each image in the dataset, the black background is cropped out to prevent having a dataset of redundant black images. The annotations are mapped onto the cropped image as shown in Figure 3 (Note that the boxes and grid are only for demonstration purposes). The cropped image is divided into a 10x10 grid whose tiles become the images of the new dataset, thereby generating 100 labeled images for the new dataset. Each tile is labeled as either 0 for non-defective or 1 for defective and assigned a unique file name that includes its label for easy identification.

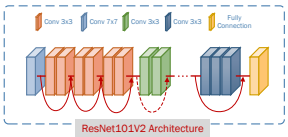
Applying the process above on all the images in the source dataset generates a dataset of 22,700 images containing both the defective and non-defective classes. However, the dataset was heavily imbalanced, with 1,609 images in the defective class and 21,091 images in the non-defective class. The even-odd approach to data balancing mentioned in Section III-B2 generated a balanced dataset with 11,350 images in each of the classes (22,700 images in total). To train and evaluate the model, the dataset is split into the training, validation, and test datasets in the ratio 8:1:1. The dataset obtained consists of a training set of 18,160 images, a validation set of 2,270 images, and a test set of 2,270 images.

D. Transfer Learning Model Design

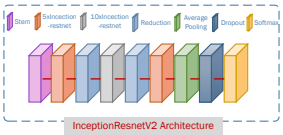
The success of transfer learning approaches lies in the choice of the base model selected [5]. The depth of the network plays an important role in the accuracy of the CNN, as deeper networks extract more complex features about the dataset. In our implementation, the selection of the base model depends on the depth of the network. We choose a pre-trained model from each of the following three classes of models: network depth level $\sim$ 100; $\sim$ 200; and $\sim$ 400 to evaluate our approach. Figure 4 illustrates the architecture of our network.



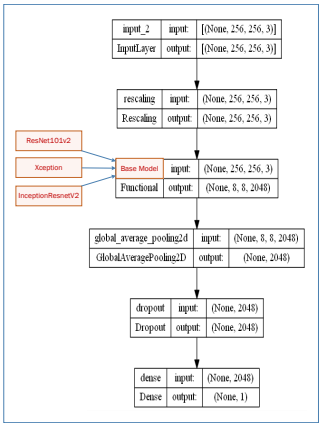
(a) The general Xception architecture



(b) The general ResNet101V2 architecture



(c) The general InceptionResNetV2 architecture



(d) The Transfer Learning architecture

Fig. 4. The proposed architecture design for training 3 different Transfer Learning models

In our first class of the base model, we choose Xception [18] with 81 convolution layers. Xception is inspired by the Inception model but is more lightweight and offers better performance (79.0% Top-1 Accuracy on the ImageNet dataset) [18]. In our second class of the base model, we choose ResNet101V2 [19], with a network depth of 205 layers. ResNet101V2 is part of the residual family of networks that are easy to train and whose accuracy increases with an increase in depth. The model achieved a Top-1 Accuracy of 77.2% on the ImageNet. In our final class of the base model, we choose InceptionResNetV2 [20], with a network depth of 449 layers. InceptionResNetV2 combines the Inception architecture and residual connections, which significantly improves the training of the Inception networks. The model achieved 80.3% Top-1 Accuracy on the ImageNet dataset.

Our network consists of an input layer, a rescaling layer, the functional base model, a global average pooling layer, a dropout layer, and an output layer. The shape of the image (fixed for all images) is specified as the parameters for the input layer. The range of the input image's pixel values is converted from [0,255] to [-1,1] in the rescaling layer. This normalization ensures that large pixel values are not given more importance over smaller pixel values during weight updates. The output of the rescaling layer is connected to the input layer of the base model. The base model then serves as a feature extractor, and its pre-trained weights provide the basis from which information can be learned about the new dataset.

During feature extraction, all features are mapped to their corresponding feature space in a convolutional process. However, the convolution does not differentiate between relevant and redundant features, resulting in a large feature space. The authors in [21] proved the effectiveness of global average pooling in reducing redundant features. Another work [22] proved that the classification performance of global average pooling was similar to that of global max pooling, but performed better on the localisation of features. This advantage provides a necessary foundation for future work involving more localised bounding boxes of the defects.

During training, the model is prone to overfitting on the training dataset as some settings of weights can perfectly predict the output but fail to predict correctly on the validation data. The authors in [23] proposed a dropout layer to overcome this deficit. By introducing this layer, some neurons are randomly omitted from the network and cannot be depended upon for weight update. The prevention of this strong co-dependency between neurons thus prevents overfitting.

The output of the network is passed through a sigmoid activation function in the output layer. The sigmoid function returns the prediction as a probability value between 0 and 1. Output values closer to zero represent predictions of the non-defective class, while predictions closer to 1 represent the defective class.

An identical network is repeated for all three base models, maintaining the overall architecture but only altering the pre-trained model. The model is then trained and fine-tuned for 50 epochs and saved for inference in the Detection Phase.

E. Defect Detection

During inference, a random image is loaded from the target dataset. The image is split into an equal number of tiles as specified during the pre-processing step. A sliding window selects individual tiles and passes them one by one into the model for prediction. A threshold value of 0.7 is assigned to determine if the prediction should be classified as defective or not. As greater importance is given to correctly classifying defective tiles, we set the threshold to be comfortably higher than 0.5 but not too high, as it could miss out on the classification of some images.

A bounding box is drawn over the defective tiles on the overall image providing a pseudo-object detection. We discuss the results of the defect detection in Section V.

V. Results and Discussions

A. Evaluation Metrics

The choice of several metrics provides a more robust, accurate comparison as one classifier could perform well on a single metric and poorly on a different metric [24]. We therefore evaluate the performance of our classifiers using binary accuracy, precision, recall, F1 Score, and AUC.

The accuracy gives the ratio of correct classifications of the model. Accuracy is obtained by comparing the predicted label against the ground truth label. A model with high accuracy correctly predicts the label of the image majority of the time. In evaluating the performance of a classifier, we also want to know the fraction of relevant instances out of the total number of predicted instances (precision) and the fraction of relevant instances that were correctly predicted out of the total relevant instances (recall). The precision gives the fraction of true positives against the number of predicted positives (i.e., the sum of true positives and false positives). Recall gives the fraction of true positives against the number of actual predicted positive instances (i.e., the sum of true positives and false negatives).

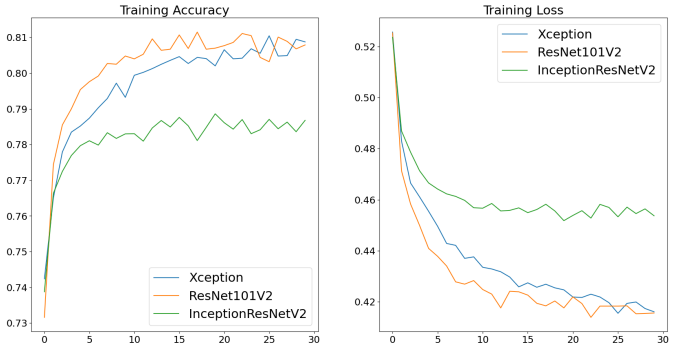
A simple measure of the precision and recall of different classifiers could be misleading as a classifier may have better precision and worse recall as compared to another classifier. We therefore use the F1 Score of the models to provide a more accurate comparison of the classifiers. The F1 Score measures the harmonic mean of precision and recall. It provides a reliable metric for evaluating classifiers on a balanced dataset [25].

The AUC (Area Under the ROC Curve) provides an alternative measure of performance in the absence of a confusion matrix [25]. It evaluates all possible classification thresholds of a model, thereby giving an accurate quality measure of the classifier.

B. Evaluation on the Source Dataset

The summary of the feature extraction accuracy and loss during the training phase is shown in Figure 5.

The results show that the accuracy of the Xception (network depth level $\sim$ 100) and ResNet101V2 (network depth level $\sim$ 200) models increased rapidly and converged faster than the



(a) Training Accuracy (b) Training Loss

Fig. 5. Results of feature extraction accuracy and loss for the transfer learning models

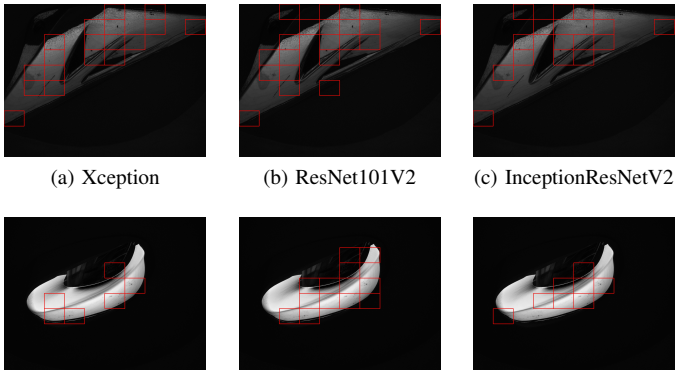
InceptionResNetV2 (network depth level $\sim$ 400) model during the training phase, while the training loss of the Xception and ResNet101V2 models drop significantly faster than the training loss of the InceptionResNetV2. This phenomenon is interesting because although the InceptionResNetV2 model had the highest network depth, it performed more poorly than the other two models. This result proves that the choice of the base model for a transfer learning architecture does not depend on the network's depth.

After fine-tuning for better performance, we evaluate the results of the models on the test dataset consisting of 2,270 images and present the results in Table I. We can observe that the ResNet101V2 model had the best performance with accuracy 95.72% and also outperformed Xception and InceptionResNetV2 in recall, F1 score, and AUC. It is only surpassed by InceptionResNetV2 in precision (95.41%). InceptionResNetV2 also has a better performance than Xception across the metrics. We observe that the accuracy of each model is similar to its F1 score. This is due to the fact that the models are trained on a balanced dataset.

1) Evaluation on the Target Dataset: For the evaluation of our approach on unseen target data, an image is split into a 10x10 grid producing 100 new images. Each resulting tile is manually inspected for the presence of defects and assigned to its corresponding class, that is, 0 for non-defective and 1 for defective. We produce an evaluation dataset of 100 images by this approach and evaluate the performance of the pre-trained models. The results are illustrated in Table II. We observe that the Xception model performs better than the two other models with an accuracy of 91.00% and also a better precision, F1 Score, and AUC. Xception is only surpassed by Resnet101V2 in recall.

C. Defect Detection

We discussed the defect detection phase in Section III-C and Section IV-E. We now evaluate the results of the defect detection on two images in our target dataset. We compare the results of the pseudo bounding boxes predicted by the transfer learning models on two new unseen target objects and present the results in Figure 6.



(a) Xception (b) ResNet101V2 (c) InceptionResNetV2

(d) Xception (e) ResNet101V2 (f) InceptionResNetV2

Fig. 6. Result of defect detection using three different models on two new unseen target objects

The consistent performance of Xception on the unseen target objects is visible in the pseudo-object detection, where it correctly identifies more defective tiles (by visual inspection) than ResNet101V2 and InceptionResNetV2. Although InceptionResNetV2 has better accuracy, precision, and F1 score than ResNet101V2 (as seen in Table II), in practice, ResNet101V2 performs better than InceptionResNetV2. The performance of InceptionResNetV2 is significantly poor on the defect detection.

We conclude that the ability of the model to better extract features is crucial for its inference on the target images, irrespective of its performance after fine tuning. The results also justify the success of our data augmentation approach as both Xception and ResNet101V2 are able to extract relevant features about the source images from the augmented dataset. Knowledge about these features is then transferred to the target dataset and used for inference on the new images.

VI. Conclusions

Machine learning techniques have an important role in the smart manufacturing industry and have been well-embraced as a solution to many of the existing challenges in the domain. The identification of product imperfections is a crucial process in the manufacturing industry. Smart manufacturing makes use of machine learning techniques to facilitate efficient and rapid identification of these defects. However, the process is limited by the unavailability of large amounts of data required for model training.

This paper proposed an architectural design that leverages transfer learning models for product defect detection applications with the goal of automating the quality assurance process in modern manufacturing. We also presented a method to augment a small dataset to produce a sufficiently large dataset for training the models. We built an augmented dataset of 22,700 labeled images from a source dataset containing 227 annotated images. We then used transfer learning models to build a robust classifier from the augmented dataset that could

TABLE I
PERFORMANCE OF TRANSFER LEARNING MODELS ON TESTING DATA

Model	Network Depth Level	Accuracy	Precision	Recall	F1 Score	AUC
Xception	~ 100	0.9502	0.9436	0.9577	0.9506	0.9746
ResNet101V2	~ 200	**0.9572**	0.9505	**0.9647**	**0.9575**	**0.9758**
InceptionResNetV2	~ 400	0.9542	**0.9541**	0.9541	0.9541	0.9712

TABLE II
COMPARISON OF MODEL PERFORMANCE ON NEW UNSEEN TARGET OBJECT

Model	Network Depth Level	Accuracy	Precision	Recall	F1 Score	AUC
Xception	~ 100	**0.9100**	**0.5385**	0.7000	**0.6087**	**0.9128**
Resnet101V2	~ 200	0.7800	0.2857	**0.8000**	0.4210	0.8883
InceptionResNetV2	~ 400	0.8700	0.3846	0.5000	0.4347	0.7833

accurately predict the presence of defects on industrial objects. We found that ResNet101V2 outperforms Xception and InceptionResNetV2 on our source data, with an accuracy of 95.72%. However, Xception produced better results (accuracy of 91.00%) than the other two models when tested on unseen target objects. From our results, we also inferred that the choice of a pre-trained model is not dependent on the depth of the network.

The success of our approach is justified by the significant performance on both the source dataset and a target dataset containing unseen and unlabelled images. This supports the conclusion that our proposed method can be implemented on defect detection applications with limited source data available, and the knowledge learned can be extended to new unseen data. Our proposed approach is currently limited to the datasets discussed in this paper. However, future work can be extended by applying the method to different defect datasets and evaluating the results.

ACKNOWLEDGMENT

This work is supported by the NBIF Pre-AI Voucher Fund (AIP2023-007) and the NBIF Talent Recruitment Fund (TRF2003-001).

REFERENCES

[1] R. Ding, L. Dai, G. Li, and H. Liu, "Tdd-net: a tiny defect detection network for printed circuit boards," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 2, pp. 110–116, 2019.

[2] S. T. Liong, D. Zheng, Y. C. Huang, and Y. S. Gan, "Leather defect classification and segmentation using deep learning architecture," *International Journal of Computer Integrated Manufacturing*, vol. 33, pp. 1105–1117, 2020.

[3] A. Boikov, V. Payor, R. Savelev, and A. Kolesnikov, "Synthetic data generation for steel defect detection and classification using deep learning," *Symmetry*, vol. 13, 7 2021.

[4] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, 2018.

[5] W. Zhu, B. Braun, L. H. Chiang, and J. A. Romagnoli, "Investigation of transfer learning for image classification and impact on training sample size," *Chemometrics and Intelligent Laboratory Systems*, vol. 211, 2021.

[6] D. Tabernik, S. Šela, J. Skvarč, and D. Skočaj, "Segmentation-based deep-learning approach for surface-defect detection," *Journal of Intelligent Manufacturing*, vol. 31, pp. 759–776, 3 2020.

[7] S. Jain, G. Seth, A. Paruthi, U. Soni, and G. Kumar, "Synthetic data augmentation for surface defect detection and classification using deep learning," *Journal of Intelligent Manufacturing*, vol. 33, pp. 1007–1020, 4 2022.

[8] Rismiyati, S. N. Endah, Khadijah, and I. N. Shiddiq, "Xception architecture transfer learning for garbage classification," Institute of Electrical and Electronics Engineers Inc., 11 2020.

[9] M. Yilmazer, M. Karakose, I. Aydin, and E. Akin, "Transfer learning based fault detection approach for rail components," Institute of Electrical and Electronics Engineers Inc., 2022.

[10] L. Wen, X. Li, X. Li, and L. Gao, "A new transfer learning based on vgg-19 network for fault diagnosis," in *2019 IEEE 23rd CSCWD*, pp. 205–209, IEEE, 2019.

[11] J. Zhu, C. Zhang, H. Qi, and Z. Lu, "Vision-based defects detection for bridges using transfer learning and convolutional neural networks," *Structure and Infrastructure Engineering*, vol. 16, pp. 1037–1049, 7 2020.

[12] Y. Gong, H. Shao, J. Luo, and Z. Li, "A deep transfer learning model for inclusion defect detection of aeronautics composite materials," *Composite structures*, vol. 252, p. 112681, 2020.

[13] M. Abu, A. Amir, Y. H. Lean, N. A. Zahri, and S. A. Azemi, "The performance analysis of transfer learning for steel defect detection by using deep learning," vol. 1755, IOP Publishing Ltd, 3 2021.

[14] P. Zhao, C. Li, M. M. Rahaman, H. Xu, H. Yang, *et al.*, "A comparative study of deep learning classification methods on a small environmental microorganism image dataset (emds-6): from convolutional neural networks to visual transformers," *Frontiers in Microbiology*, vol. 13, 2022.

[15] M. Saini and S. Susan, "Data augmentation of minority class with transfer learning for classification of imbalanced breast cancer dataset using inception-v3," vol. 11867 LNCS, pp. 409–420, Springer, 2019.

[16] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: overview study and experimental results," in *2020 11th international conference on information and communication systems (ICICS)*, pp. 243–248, IEEE, 2020.

[17] M. Hussain, J. J. Bird, and D. R. Faria, "A study on cnn transfer learning for image classification," in *Advances in Computational Intelligence Systems*, (Cham), pp. 191–202, Springer International Publishing, 2019.

[18] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[19] 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*, pp. 630–645, Springer, 2016.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] M. Lin, Q. Chen, and S. Yan, "Network in network," 12 2013.

[22] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

[23] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[24] N. Seliya, T. M. Khoshgoftaar, and J. V. Hulse, "A study on the relationships of classifier performance metrics," pp. 59–66, 2009.

[25] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, 1 2020.