Proc. of International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA 2024)
1-2 February 2024, Victoria-Seychelles

Error Compensation of Inkjet-printed Electronics using Incremental Learning and Knowledge Distillation

1st Abdalla Shahin
Machine Vision
Profactor GMBH
Steyr, Austria
abdalla.shahin@profactor.at

***Abstract*—The paper targets the development of an error compensation step inline a fabrication process of inkjet-printed electronics. In the process, a microchip is placed in a circuit board cavity pre-filled with adhesive material which is cured afterwards. This pick and place process is very precise, yet difficult due to the miniaturization of the chip and the board. The paper targets the development of a robust algorithm that can detect a typical defect that results from the process and calculate the corrective data. The main challenge is that inspection data are generated in batches over time based on resources availability and process development which make each batch of data unique. Therefore, an incremental learning methodology is sought so that the algorithm can be trained on streams of data during process development and provide feedback for defect repair.**

***Index Terms*—Defect detection, Error compensation, Incremental learning, Inkjet printing, Inline inspection.**

I. Introduction

The TINKER project targets the development of a new reliable, functional, and resource efficient pathway for fabrication of sensor packages used in autonomous driving, most importantly radio detection and ranging (RADAR) and laser imaging, detection, and ranging (LIDAR) sensors. The public awareness and the industrial need for further miniaturization of such sensor packages are the main driver of ongoing efforts in the automotive sector to integrate such devices into the car body instead of attaching them (e.g. on top of the car in case of LIDAR device) [1], [2]. The paper is targeting the integration of the RADAR micro chip, so called bare-die, inside a printed circuit board (PCB) cavity filled initially with nonconductive adhesive. Due to the nonuniform spreading of the adhesive, there exist some places in the gap between the bare die and the cavity boundaries where adhesive is missing and needs to be compensated for to reach a flat surface. With that in mind, the aim within TINKER is to enable error compensation inline the respective fabrication step using machine learning algorithms trained via data generated by the inline inspection system [3]. The main target of such machine learning algorithms is high accuracy since the fabrication is done in a micro level. Additionally, robustness is required, which means that by utilizing incremental learning (IL), the algorithms converge to the same optimal or at least sub-optimal solution as achieved by offline trained algorithms. In this paper, two deep learning models are developed: semantic segmentation model for detecting the gap, and monocular depth estimation (MDE) model to estimate the missing adhesive volume. The paper is structured as follows: in Section II, previous related work will be discussed. The methods of the previous work reflect back to the methodology of this paper which will be discussed in Section III. In Section IV, the models' training and testing results are presented and discussed. Finally, conclusions and possible future improvements are presented in Section V.

This paper is supported partially by funding from the European Union's Horizon 2020 research and innovation program under the Grant Agreement n°958472, project TINKER. It is also partially supported by the government of Upper Austria under grant Zer0P.

II. Related Work

A. Semantic Segmentation Overview

Semantic segmentation is defined as the pixel-wise labelling in which each pixel is assigned to a certain class; unlike classification, for instance, which assigns the whole image to one class [4]. The segmentation models based on convolutional neural network (CNN) have proven to achieve outstanding performances [5], and therefore the paper investigates them in particular. One of the very popular architectures are the fully convolutional network (FCN) in [6], the DeConvNet in [7] and the SegNet in [8]. Moreover, the U-net received wide popularity as it manages to preserve and propagate the spatial information of the input image to the final predicted mask using direct connections between the encoder and decoder blocks [9]. The U-net structure is shown in Fig. 1.

B. Monocular Depth Estimation Overview

In supervised MDE, depth information is estimated based on a single monocular image and the corresponding ground truth (GT) depth map. The problem is regarded as a regression task in which the estimator learns to predict the pixel-wise

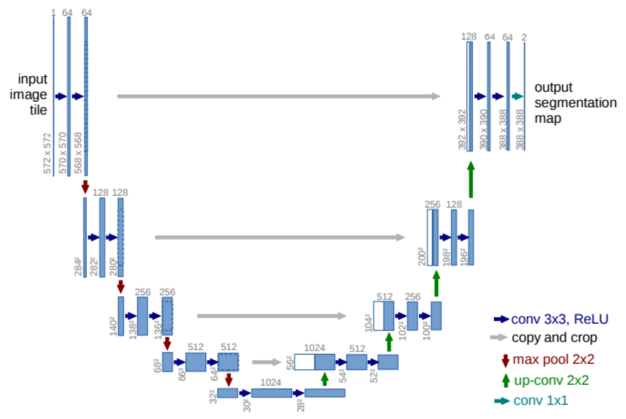

Fig. 1. U-net architecture. [9]

depth value minimizing the error between predicted and GT value utilizing CNN-based models [10], [11]. A significant improvement to the MDE task was based on an encoder-decoder structure which has shown a high performance in extracting the spatial features of images [12]–[14]. Some scholars use blocks from well-known architectures as encoder backbone to improve image feature extraction such as ResNet-50 [15]. Lee et al. proposed an encoder-decoder approach which uses novel local planar guidance layers which explicitly map the image features to the desired depth predictions [16]. This approach produced quite outstanding results and is considered as the current state-of-the-art.

C. *Incremental Learning Overview*

Many approaches have discussed IL in the recent years, but comparing these approaches is difficult due to the variety of frameworks. However, they can be concluded in three main types: task-IL, domain-IL, and class-IL [17]. The Domain-IL can be described as incrementally learning the same kind of problem but with different contexts and conditions, such as recognizing objects under different lighting conditions or estimating depth in both indoor and outdoor scenes. Some IL implementations depend on the transfer of knowledge or weights between different tasks as done by [18], [19]. Other approaches focus more on formulating a special training loss function consisting of different parameters to direct the training process based on these parameters [20]–[22]. The paper is targeting the domain-IL, since the segmentation and depth estimation outputs are always the same; however, the recording conditions and contexts of data used for both models are changing.

III. Methodology

A. *Data & Resources Description*

Throughout the TINKER project, different datasets have been collected in a sequential manner due to process development and limited material supply. There is a total of five datasets as shown in Fig. 2, two of which are of the same PCBs but with different inspection setup. Each dataset includes

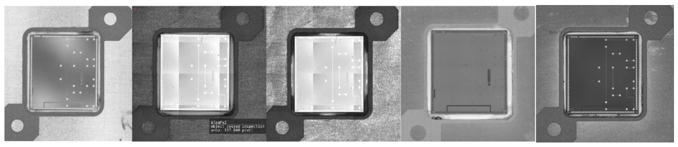

Fig. 2. A sample image from each dataset in the order of usage

high resolution monochrome images of the bare-die in cavity; whereas 3D surface scans are available for only three datasets. GT annotations for gap segmentation are generated using GIMP software. All images and GT annotations are cropped to the nearest size divisible by 256 and then divided into nonoverlapping patches of size 256x256 to have more training data. In the end, the number of image pairs in each dataset was 450, 1920, 1920, 72, and 16128 following the same order in Fig. 2. Since the monochrome images have higher resolution than the 3D scans, they are downsized to the size of the 3D scans. The total number of samples used for the MDE model was 288 image pairs. Moreover, data augmentation functions, especially horizontal and vertical flipping, are used to improve generalization and avoid overfitting. The overall training time of the models was around 4 hours using a NIVIDIA 1050 Ti GPU with 12GB memory. The training of the gap segmentation is a bit more complex than that of the MDE model due to the higher number of incremental steps. Therefore, it takes around 2.5 hours based on 75 epochs and 300 training samples for base model training, and 30 epochs and around 100 training samples for each incremental step. All the DL models were coded using python programming language and trained using the TensorFlow framework.

B. *Gap Detection*

The U-net architecture is chosen for this implementation due to its high performance especially when the training data are few. An ablation study was made to test different backbone networks in terms of number of training parameters, convergence time, and fitting accuracy. The tested networks were VGG-16, ResNet-34, and Ineception-V2 as shown in table I. ResNet-34 is chosen as backbone network due to good and fast convergence as well as less training time due to relatively small amount of training parameters. The U-net, already assembled with the ResNet-34 backbone network, is imported from keras built-in models, and trained with the processed and augmented data. Fig. 3 shows a training sample after processing. The imagenet pre-trained weights were used for initializing the encoder weights to improve the learning process and speed up convergence. The model was trained for 75 epochs utilizing Adam optimizer [23] and a constant

TABLE I
Performance comparison between VGG-16, ResNet-34, and Inception-V2 backbone networks.

Network	Training parameters	Training time	Precision	Recall	Mean IoU
VGG-16	24M	16.23 mins	97.60%	98.03%	95.73%
ResNet-34	24M	10.06 mins	97.15%	98.79%	96.02%
Ineception-V2	30M	13.63 mins	97.91%	98.66%	96.63%

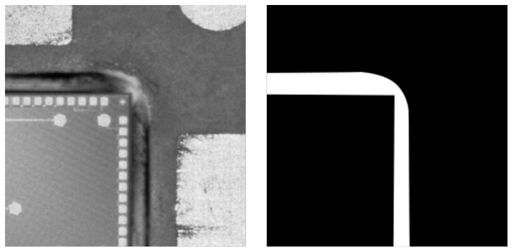
(a) Gap image (b) Gap mask

Fig. 3. A sample training tuble for the gap segmentation model

learning rate of 1×10^{-4}. Since our segmentation task is binary, the training loss function consists of a binary cross entropy (BCE) loss and Jaccard loss, which is commonly referred to as IoU loss function [24]. The BCE loss is used because it employs the log probability of the predicted values and hence penalizes those probabilities based on the distance from the true values. The combined training loss function is shown in (1).

$$L_0 = -\tfrac{1}{N}\sum_{i=1}^{N} Y_i \times log(M(x_i)) + (1-Y_i)\times log(1-M(x_i)) \\ -1 + \tfrac{|Y_i \bigcap M(x_i)|}{|Y_i \bigcup M(x_i)|}, \tag{1}$$

where Y_i is the GT, x_i is the data input, and $M(x_i)$ is the model prediction, for N data samples.

C. Depth Estimation

Since the gap mask has been extracted from the gap detection model, all substrate images and 3D scans are masked to get only the gap contour as shown in Fig. 4. For estimating the missing adhesive in the gap, the same gap detection U-net model is used while freezing the encoder layers and re-training the decoder layers. This way, the MDE model can directly utilize the model's encoder, which was trained to extract the features and the spatial information of monochrome images. The training loss function is a combination of three functions, as was done in [25]. Due to the complexity of the MDE task of this paper since a monochrome image is the only input, a loss function that takes in account all details and features of an image such as edges, pixel gradients, and luminance is needed. Moreover, the combined loss function was preferred

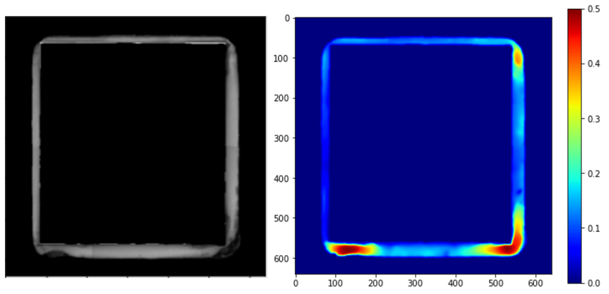

Fig. 4. A data sample used for training the MDE model where the gap is masked in both input and output data

based on its empirical results. The combined loss function is presented in (2).

$$Combined = \alpha_1 L_{SSIM}(z,\hat{z}) + \alpha_2 L_{MAE}(z,\hat{z}) + \alpha_3 L_{reg}(z,\hat{z}), \tag{2}$$

where $\hat{z}_i$ is the predicted value and z_i is the GT value. L_{SSIM} is the structural similarity index which computes similarity between images on the basis of luminance, contrast, and structure (3); and $L_{MAE}(z,\hat{z})$ is the mean absolute error for pixel-wise comparison (4). Moreover, L_{reg} is the Depth Smoothness loss which compares the two depth maps based on their gradients in x and y directions as in (5). Inspired from [26], the total L_{reg} loss is the mean of the gradients' sum over N pixels as shown in (6). α_1, α_2, and α_3 are weighting factor set empirically to 0.85, 0.1 and 0.9, respectively. After training for 100 epochs and using Adam optimizer with a constant learning rate equal to 1×10^{-4}, the model produced very good performance as will be shown in Section IV.

$$L_{SSIM}(z,\hat{z}) = \frac{1}{2}(1-\frac{(2\mu_{\hat{z}}\mu_z + c_1)(2\sigma_{\hat{z}z}+c_2)}{(\mu_{\hat{z}}^2+\mu_z^2+c_1)(\sigma_{\hat{z}}^2+\sigma_z^2+c_2)}), \tag{3}$$

where $\mu_{\hat{z}}$ is the mean of $\hat{z}$, $\sigma_{\hat{z}}$ is the standard deviation of $\hat{z}$, μ_z is the mean of z, σ_z is the standard deviation of z, $\sigma_{\hat{z}z}$ is the covariance of $\hat{z}$, and $c1$=0.012 and $c2$=0.032.

$$L_{MAE}(z,\hat{z}) = \frac{1}{N}\sum_{i}^{N}|z_i - \hat{z}_i| \tag{4}$$

$$\begin{aligned} S_x &= \bigtriangledown_x \hat{z}_i \times e^{\frac{1}{N}\sum_i^N |\bigtriangledown_x z_i|} \\ S_y &= \bigtriangledown_y \hat{z}_i \times e^{\frac{1}{N}\sum_i^N |\bigtriangledown_y z_i|} \end{aligned} \tag{5}$$

$$L_{reg}(z,\hat{z}) = \frac{1}{N}\sum_{i}^{N}|S_x| + |S_y| \tag{6}$$

D. Incremental Learning

The gap detection model has to be trained on all datasets to be able to fit them all. Moreover, because 3D scans of three datasets are available, IL can be applied to the MDE model as well. Both models are trained once using one initial dataset, and then the remaining datasets are fed incrementally to them. The methodology of this paper is inspired from [21], [22] in which the authors apply incremental learning to a multi-class segmentation task. In this paper, the method is applied to both segmentation and MDE models, since it is applicable to any deep network architecture and it has shown promising results. The training scheme of the method is shown in Fig. 5, where a base pre-trained model undergoes k incremental steps corresponding to k new datasets. In each step, a new model, with the same structure and weights as the base model, is created and complemented with a knowledge distillation feature which prevents catastrophic forgetting. Knowledge distillation is achieved through the proper choosing of the training loss function which highly affects the learning process of the model by penalizing the model weights and directing the model to learn without forgetting. The main loss function is shown in (7).

$$L_{total} = L_0 + \lambda_D L_D, \tag{7}$$

where L_0 is the initial loss function used for fitting the new data, λ_D represents the distillation factor which is a hyper-parameter, and $L_D \in \left\{L_D^{'}, L_D^{''}, L_D^{'''}\right\}$ is a distillation loss for retaining the past information. For gap segmentation task, L_{seg_0} is the same as in (1). The first distillation loss function $L_{seg_D}^{'}$ is applied to the outputs of the two models in order to compare both of them and apply required penalty to keep the new model from being biased to the new dataset. Therefore, the loss function in (1) is used as well; however, the GT values are replaced by the old model's predictions. $L_{seg_D}^{''}$ is applied to the intermediate feature space, which is the output of both encoders. Since the last layer of the encoder is not producing a classification output but rather a feature space, another loss function is needed that should keep the two feature spaces as close as possible such as the standard L2 loss function. Denoting the encoder of the model as E, the second distillation loss function can be written as in (8).

$$L_{seg_D}^{''} = \frac{1}{N}\sum_{i=1}^{N} \|E_{k-1}(x_i) - E_k(x_i)\|_2^2 \tag{8}$$

Finally, $L_{seg_D}^{'''}$ is applied to the output of the last three decoder layers; the number of layers is also a hyper-parameter. The third distillation loss function is used since the output of the decoder layers of each model should be kept close as well. The L2 loss function is used and can be re-written as in (9).

$$L_{seg_D}^{'''} = \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{3} \frac{\left\|d_{k-1}^{l}(x_i) - d_k^{l}(x_i)\right\|_2^2}{3} \tag{9}$$

In addition, another distillation feature can be added which is keeping the encoder of the new model frozen (E_F) while fitting the new data through the decoder weights only. This aims at preserving the feature extraction capabilities of the encoder. That being said, more than one distillation function can be combined in the same incremental step if needed. The same incremental procedure is applied to the MDE model because it uses the same network architecture; however, the loss functions are different since this is a regression task. For both the initial loss function L_{MDE_0} and the first distillation loss function $L_{MDE_D}^{'}$, the combined loss in (2) is used. For the second and third distillation loss functions, $L_{MDE_D}^{''}$ and

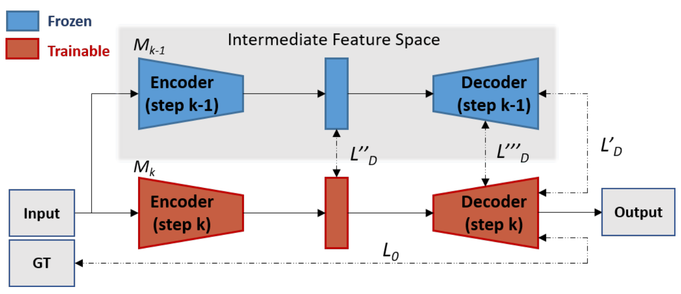

Fig. 5. Knowledge distillation scheme of the IL method at k-th incremental step. [21]

$L_{MDE_D}^{'''}$, berHu is used (10) since it combines both L1 and L2 loss functions and has the advantages of both functions.

$$berHu = \begin{cases} |\Delta y| & |\Delta y| \le c \\ \frac{\Delta y^2 + c^2}{2c} & |\Delta y| > c \end{cases}, \\ c = \frac{1}{5} max_i(|M_{k-1}(x_i) - M_k(x_i)|), \tag{10}$$

where x_i are the input images, $M_{k-1}(x_i)$ are the model's predictions at step $k-1$, $M_k(x_i)$ are the model's predictions at step k, and Δy is the computed error between them.

IV. Experimental Results

A. Reference KPIs

In TINKER, there are targeted KPIs which are defined so as to properly assess the quality and robustness of the developed models. Moreover, the targeted KPIs aim for beyond state of the art [27], [28]. For both gap segmentation and depth estimation, the targeted KPIs focus on the deviation of the models' predictions with respect to the true labels. The targeted deviation was set to a maximum of 10% of the true label. For gap segmentation, the IoU metric is used which already computes the percentage of true predictions with respect to the union of true and predicted entities. Simply the deviation is interpreted as 1 - IoU, and so the KPI for gap segmentation is achieving a minimum mean IoU of 90%. For depth estimation, the deviation of the predicted depth values was set to 10% of the true depth values as well. This describes the definition of the absolute relative error, which is a traditional evaluation metric used for MDE models. Therefore, the target absolute relative error value of the MDE models is 0.1.

B. Gap Detection with Incremental Learning

After training the base model on the initial dataset, the other datasets are fed incrementally to that model. The five datasets will be referred to as the following D^n where $n \in \{0, 4\}$. Each dataset is divided into two parts: training D_{tr}^n and testing D_{ts}^n with percentages of 80% and 20%, respectively; whereas one third of the training part are used for cross validation. In this section, the results of the incremental training steps are presented and discussed. Moreover, they are compared with results obtained from standard offline-trained models with samples from current and previous datasets at once.

In the first incremental step, the model structure inherits that of the base model which was trained using the old dataset (D_{tr}^0). The model is retrained on (D_{tr}^1) utilizing different distillation functions. With combining both $L_D^{'}$ and $L_D^{''}$ and setting the value of λ_D to 1.0, the model performs well on the two datasets achieving almost the same scores as the offline-trained model $M_0(D_{tr}^{\{0,1\}})$. This is because increasing the value of λ_D leads to higher distillation loss value and thus, gives the model a stronger bias towards maintaining performance on past data. In the second incremental step, the incremental model M_1 is retrained with dataset (D_{tr}^2) which shows the substrate before curing the adhesive; moreover, it has different lighting conditions compared to (D_{tr}^1). Using both distillation loss functions $L_D^{'}$ and $L_D^{''}$ is considered to

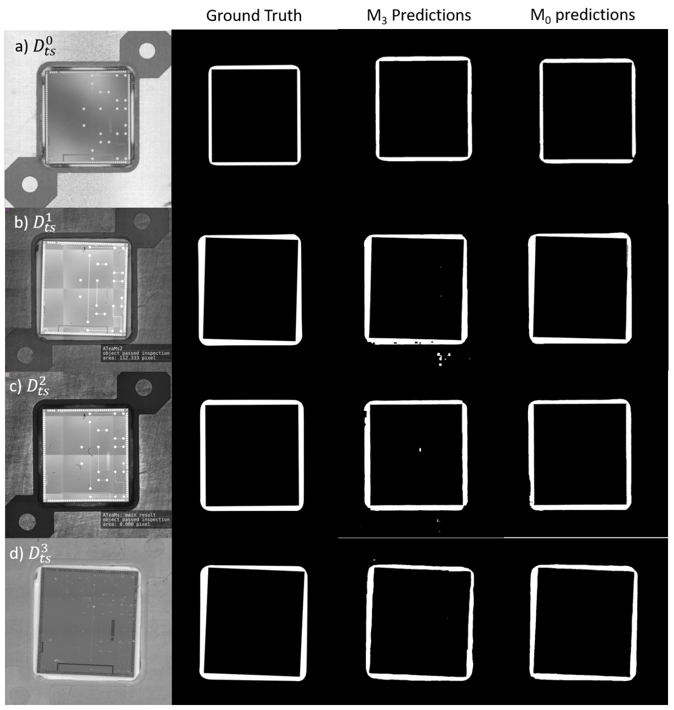

Fig. 6. Qualitative results of the final incremental segmentation model M_3 on testing samples from all the four datasets D^0_{ts} (a), D^1_{ts} (b), D^2_{ts} (c), and D^3_{ts} (d).

achieve the best performance. Similarly, in the last incremental step, the model is trained with dataset (D^3_{ts}). Freezing the encoder and applying only the distillation loss function $L^{'}_D$ allows the model to overcome catastrophic forgetting. In table II, scores of the best incremental models are shown. The incremental models achieve slightly lower evaluation scores than the offline-trained models, which was expected. However, from the qualitative results in Fig. 6, the deviations between incremental models' predictions and offline-trained models' predictions are minor. Moreover, the offline-trained models have higher computational cost and training time since they need more data samples and more epochs to fit all data. Furthermore, the fact that incremental models achieve such performances without accessing the previous datasets gives them more credibility. The

C. Monocular Depth Estimation with Incremental Learning

In this section, the performance of the MDE model is presented and evaluated in terms of the popular evaluation criteria used in literature. Table III shows the scores of the base MDE model on testing data from datasets D^1 and D^4 in the first two rows. Fig. 7 shows samples from the MDE model's predictions on the D^1_{ts} data and D^4_{ts} data. From the figure, it is clear that the trained MDE model can perform well regardless of the location of higher depth values due to the random flip augmentation function that was applied. With the large dataset D^4, the model is able to capture more information due to the big amount of training samples and the diversity of depth distributions along the gap.

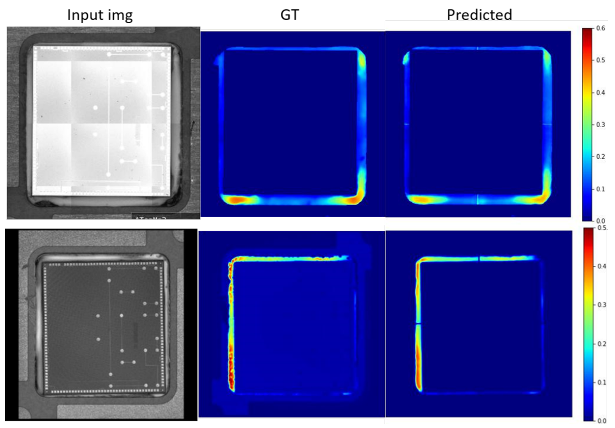

Fig. 7. Qualitative results of the best MDE model on samples from the D^1_{ts} dataset and the large dataset D^4_{ts}.

The MDE model is trained incrementally one time with the D^3_{tr} dataset. The 2D images of both datasets, D^1 and D^3, are describing the same PCB substrates but are different in terms of the recording setup, and consequently they have different resolution and different features. However, the GT depth measurements of both datasets are actually the same. The training was concluded in 45 epochs with a learning rate of 5×10^-6 which decays linearly with a factor of 0.5 every 15 epochs. In a first attempt, the loss functions, $L^{'}_{MDE_D}$ in (2) and $L^{''}_{MDE_D}$ in (10), are combined while training the whole model. In a second attempt, the encoder of the new model is frozen and only the first distillation loss function $L^{'}_{MDE_D}$ is used. Consequently, the training was much faster since the encoder parameters are not trainable. That is why the model utilizing $L^{'}_D$ and E_F is preferred since both attempts produce quite similar performances. The last four rows of table III show the performance of the MDE model after incremental training. Fig. 8 shows the qualitative performance of the incremental MDE model. Compared to the performance of the base model, the performance of the incremental model is very promising since it did not degrade much; the only noticeable degradation is in the accuracy measure $\delta < 1.25$. From table III, it is observed that the value of the accuracy measure $\delta < 1.25$ of the first model in the first row is relatively low when compared to the other accuracy measures. This is due to the low amount of data samples. On the other hand, when a larger dataset was provided, the model achieved relatively higher accuracy values as reported in the second row.

V. Conclusions

To sum up, the work in this paper was done as part of the EU project TINKER. The paper targets one fabrication process of the project which is the placement of a microchip inside a PCB cavity filled with nonconductive adhesive. The developed system in this paper uses images of the bare-dies in cavities as well as their 3D surface scans for training, and provides an estimation of the missing adhesive in the gap between the chip and the boundaries of the cavity. For

TABLE II
PERFORMANCE MEASURES OF THE SEGMENTATION MODEL ON PREVIOUS DATASETS AFTER EACH OF THE THREE INCREMENTAL STEPS WITH DIFFERENT DISTILLATION CRITERIA.

	base dataset(D^0_{ts})			1st new dataset(D^1_{ts})			2nd new dataset(D^2_{ts})			3nd new dataset(D^3_{ts})		
Model	precision	recall	MIoU	precision	recall	MIoU	precision	recall	MIoU	precision	recall	MIoU
$M_0(D^{\{0\}}_{tr})$	97.15%	98.79%	96.02%	-	-	-	-	-	-	-	-	-
$M_1(L^{'}_D, L^{''}_D)$	90.26%	96.68%	87.64%	92.74%	94.74%	88.20%	-	-	-	-	-	-
$M_0(D^{\{0,1\}}_{tr})$	96.77%	94.52%	91.68%	96.93%	96.39%	93.54%	-	-	-	-	-	-
$M_2(L^{'}_D, L^{''}_D)$	93.48%	94.87%	89.07%	94.62%	97.11%	91.80%	96.48%	92.27%	90.10%	-	-	-
$M_0(D^{\{0,2\}}_{tr})$	89.87%	96.81%	87.47%	95.55%	96.79%	92.25%	97.70%	94.51%	93.34%	-	-	-
$M_3(L^{'}_D, E_F)$	90.01%	96.45%	87.20%	92.47%	94.66%	87.89%	94.77%	95.49%	90.72%	92.87%	87.26%	81.99%
$M_0(D^{\{0,3\}}_{tr})$	96.02%	92.50%	89.11%	97.19%	96.60%	93.68%	98.63%	94.33%	93.06%	96.34%	92.87%	89.84%

TABLE III
PERFORMANCE QUALITY OF THE BASE MDE MODEL AND INCREMENTAL MDE MODEL.

	Lower is better			Higher is better		
Method	Abs Rel	RMSE	log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Unet(D^1_{ts})	0.192	0.040	0.082	70.0%	92.0%	97.5%
Unet(D^4_{ts})	0.183	0.016	0.084	85.6%	92.3%	95.5%
$L^{'}_D, L^{''}_D (D^1_{ts})$	0.212	0.043	0.094	63.7%	89.6%	96.6%
$L^{'}_D, L^{''}_D (D^3_{ts})$	0.236	0.055	0.103	57.6%	86.2%	96.0%
$L^{'}_D, E_F (D^1_{ts})$	0.222	0.045	0.097	61.4%	88.9%	96.5%
$L^{'}_D, E_F (D^3_{ts})$	0.234	0.056	0.102	58.7%	86.2%	96.1%

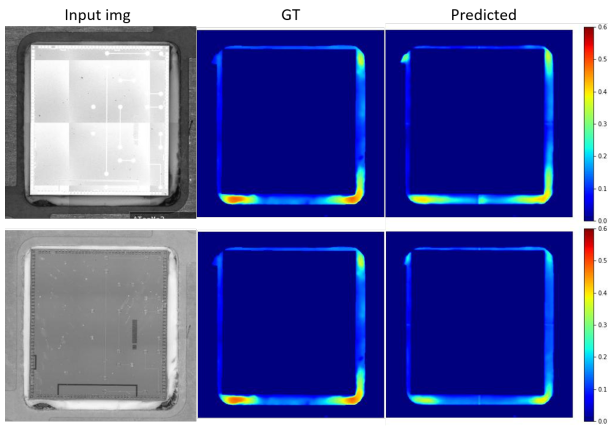

Fig. 8. Qualitative results of the incremental MDE model on samples from D^1_{ts} and D^3_{ts} datasets.

missing adhesive estimation, two models were developed; first, a segmentation model to detect the gap shape and location; second, a MDE model. Moreover, The paper investigates an incremental learning scheme to overcome the need for generating huge amounts of data at once and allow the model to fit newly generated data without catastrophically forgetting past data. The key features of applying incremental learning are the following: learning with forgetting, never accessing the past datasets, and avoiding model expansion. The incrementally trained gap segmentation model achieved great performances over all datasets. Moreover, the performance of the incrementally trained MDE model was promising as well but can be improved with more data. That is why it can be said that the small amount of 3D scans is one of the limitations of this work. Applying a second incremental training step using the dataset D^4 was not convenient due to the huge data unbalance. For future improvements, a more robust depth metrology providing scale information, for instance, would be much helpful. Further incremental learning criteria could be also added to improve the ability of the models to retain the past knowledge.

ACKNOWLEDGMENT

I would like to thank Prof. Andrea Tonello, professor at university of Klagenfurt and Dr. Christian Eitzinger, machine vision team leader at PROFACTOR GmbH for their technical support throughout this work. The datasets used in this paper are generated by the metrology systems after pick and place process at BESI GmbH, to whom I would like to extend my gratitude. Moreover, all appreciation goes to Infineon Technologies AG, the bare-dies manufacturer, and to ROBERT BOSCH GmbH, the PCBs provider. The work reported in this paper was carried out in the premises of Profactor GmbH where I had the opportunity to access all needed resources.

REFERENCES

[1] J. K. Baruah, A. Kumar, R. Bera, and S. Dhar, “Autonomous vehicle—a miniaturized prototype development,” in *Advances in Communication, Devices and Networking*, pp. 317–324, Springer Singapore, 2019.
[2] P. Wang, “Research on comparison of lidar and camera in autonomous driving,” *Journal of Physics: Conference Series*, vol. 2093, p. 012032, nov 2021.
[3] D. Gauder, J. Gölz, N. Jung, and G. Lanza, “Development of an adaptive quality control loop in micro-production using machine learning, analytical gear simulation, and inline focus variation metrology for zero defect manufacturing,” *Computers in Industry*, vol. 144, p. 103799, 2023.
[4] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, “Image segmentation using deep learning: A survey,” 2020.
[5] I. Ulku and E. Akagündüz, “A survey on deep learning-based architectures for semantic segmentation on 2d images,” *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2032924, 2022.
[6] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” 2014.
[7] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” 2015.
[8] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” 2015.
[9] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.
[10] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” *CoRR*, vol. abs/1406.2283, 2014.

[11] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He, “Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1119–1127, 2015.

[12] Y. Ming, X. Meng, C. Fan, and H. Yu, “Deep learning for monocular depth estimation: A review,” *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[13] G. Zhang, Y. Liu, and X. Jin, “A survey of autoencoder-based recommender systems,” *Frontiers of Computer Science*, vol. 14, no. 2, p. 430–450, 2019.

[14] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich, “Estimating depth from RGB and sparse sensing,” in *Computer Vision – ECCV 2018*, pp. 176–192, Springer International Publishing, 2018.

[15] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper depth prediction with fully convolutional residual networks,” 2016.

[16] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, “From big to small: Multi-scale local planar guidance for monocular depth estimation,” 2019.

[17] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias, “Three types of incremental learning,” *Nature Machine Intelligence*, vol. 4, no. 12, p. 1185–1197, 2022.

[18] M. Andrade, E. Gasca, and E. R. Lara, “Implementation of incremental learning in artificial neural networks,” in *GCAI*, 2017.

[19] D. Lopez-Paz and M. Ranzato, “Gradient episodic memory for continuum learning,” *CoRR*, vol. abs/1706.08840, 2017.

[20] H. Tercan, P. Deibert, and T. Meisen, “Continual learning of neural networks for quality prediction in production using memory aware synapses and weight transfer,” *Journal of Intelligent Manufacturing*, vol. 33, no. 1, p. 283–292, 2021.

[21] U. Michieli and P. Zanuttigh, “Incremental learning techniques for semantic segmentation,” *CoRR*, vol. abs/1907.13372, 2019.

[22] U. Michieli and P. Zanuttigh, “Knowledge distillation for incremental learning in semantic segmentation,” *CoRR*, vol. abs/1911.03462, 2019.

[23] A. Mustapha, L. Mohamed, and K. Ali, “Comparative study of optimization techniques in deep learning: Application in the ophthalmology field,” *Journal of Physics: Conference Series*, vol. 1743, p. 012002, jan 2021.

[24] D. Duque, S. Velasco-Forero, J.-E. Deschaud, F. Goulette, A. Serna, E. Decencière, and B. Marcotegui, “On power jaccard losses for semantic segmentation,” pp. 561–568, 01 2021.

[25] S. Abdulwahab, H. A. Rashwan, N. Sharaf, S. Khalid, and D. Puig, “Deep monocular depth estimation based on content and contextual features,” *Sensors*, vol. 23, no. 6, 2023.

[26] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer,” 2020.

[27] J. Spencer *et al.*, “The monocular depth estimation challenge,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, pp. 623–632, January 2023.

[28] A. Masoumian, H. A. Rashwan, S. Abdulwahab, J. Cristiano, M. S. Asif, and D. Puig, “Gcndepth: Self-supervised monocular depth estimation based on graph convolutional network,” *Neurocomputing*, vol. 517, pp. 81–92, 2023.