UNIVERSITÄT KLAGENFURT

MASTER THESIS

„Feedback Error Compensation of Inkjet-Printed Electronics Using Incremental Learning and Knowledge Distillation Methods“

submitted by

Abdalla Saber Ali Shahin

submitted in fulfilment of the requirements for the degree of

Diplom-Ingenieur

Programme: Master’s programme Information and Communications Engineering
Branch of study: Autonomous Systems and Robotics

Alpen-Adria-Universität Klagenfurt

Evaluator:
Univ.-Prof. Dr.Andrea M. Tonello
Alpen-Adria-Universität Klagenfurt
Institut für Vernetzte und Eingebettete Systeme

Klagenfurt, August 2023

Affidavit

I hereby declare in lieu of an oath that

- the submitted academic paper is entirely my own work and that no auxiliary materials have been used other than those indicated,
- I have fully disclosed all assistance received from third parties during the process of writing the thesis, including any significant advice from supervisors,
- any contents taken from the works of third parties or my own works that have been included either literally or in spirit have been appropriately marked and the respective source of the information has been clearly identified with precise bibliographical references (e.g. in footnotes),
- I have fully and truthfully declared the use of generative models (Artificial Intelligence, e.g. ChatGPT, Grammarly Go, Midjourney) including the product version,
- to date, I have not submitted this paper to an examining authority either in Austria or abroad and that
- when passing on copies of the academic thesis (e.g. in printed or digital form), I will ensure that each copy is fully consistent with the submitted digital version.

I am aware that a declaration contrary to the facts will have legal consequences.

Abdalla Shahin m.p. Klagenfurt, August 2023

Acknowledgements

I would like to deeply thank Prof. Andrea Tonello, professor at university of Klagenfurt, for being continuously reachable as well as for his technical support and his efforts in refining the thesis to be the way it is now. I am especially grateful to Dr. Christian Eitzinger, machine vision team leader at PROFACTOR GmbH, who patiently guided me throughout the whole journey, and to Sabastian Zambal and Klaus Schlachter, former employees at PROFACTOR, for their work and documentations which helped me to blend quickly in the project. Special thanks to all my fellow employees at PROFACTOR, especially Gerald Stubauer and Boris Buchroithner who helped me understand the process of inkjet printing and provided me with data samples. I would like to express my appreciation to all those I have had the opportunity to work with in the consortium of TINKER, especially to Benedikt Pressl (BESI GmbH) who provided me with the process data on which this work completely depends. Most importantly, I am greatly indebted to my family for their relentless support and for pushing me forward, whom without I would not have achieved any successes in life. I wish also to extend my acknowledgement to thank my friends for their unending inspirations.

The work of this thesis was carried out in the premises of PROFACTOR GmbH where I had the opportunity to access all needed resources and facilities. This project has received funding from the European Union's Horizon 2020 research and innovation program under the Grant Agreement n°958472, project TINKER.

Abstract

The thesis targets the development of a closed error compensation feedback step inline a fabrication process of inkjet-printed electronics. In the fabrication process, the bonding of a microchip in a Printed Circuit Board (PCB) cavity takes place in which a repair step is needed. These processes are carried out as part of the H2020 TINKER project, funded by the European Union. An essential requirement in TINKER is to develop a fast and robust algorithm that can detect whether a certain defect is present and if so segment it and determine the corrective actions needed to initiate the error compensation process. The main problem is that data generated from the post-process inspection system are not huge and they are generated in batches over time based on materials availability and inspection tools development which makes each batch of data unique. Moreover, generating a huge dataset showing process variations, so that it can be used to train the algorithm, is not applicable in the scope of TINKER. Therefore, an incremental training methodology is needed so that the algorithm has the ability to increment knowledge whenever new data is available. The thesis proposes an efficient Deep Learning (DL) based approach to act as the inline error compensation agent. Incremental Learning (IL) is targeted so that the DL structure is pre-trained using generic amount of data, and additionally, is able to fit a new chunk of data while retaining the knowledge gained from the old chunks of data without accessing them again. With the proposed approach, the DL structure will be able learn inline the process and update its trained parameters accordingly without the need of exhaustive training using all datasets.

Contents

List of Tables

List of Figures

1. Introduction

1.1. Introduction

This thesis is the outcome of the work carried out as part of the TINKER EU project. The TINKER project targets the development of a new reliable, accurate, functional, affordable, and resource efficient pathway for fabrication of sensor packages used in autonomous driving and self-driving cars, 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 is the main driver of ongoing efforts in the automotive sector to be able to integrate such devices into the car body like in the bumps and head lamps instead of attaching them (e.g. on top of the car in case of LIDAR device). With that in mind, the aim within TINKER is to enable error compensation and defect repair inline the respective fabrication step using machine learning algorithms trained via data generated by the inline inspection. Deep Neural Networks (DNNs) are gaining lots of attention currently in the processes of feedback control and defect detection, and many research-based institutes are becoming eager to utilize them in their applications [1, 2, 3]. By learning the relationship between measurements and process parameters, DL models will be able to close the feedback loop and provide reliable predictions. The developed DL model should be flexible, able to be accustomed to changes that may occur in the process, and able to learn online with the limited data available to optimize the production time based on the concept of IL [4]. IL is a learning method in which data are provided incrementally to the model that updates its parameters accordingly. A key feature in IL is avoiding catastrophic forgetting of old knowledge which usually happens to batch learning based models when retrained using a new set of data. In order to assure significant speed of the training process, the model should not re-use or re-iterate over past data samples[5]. In the course of production of the RADAR sensor package in TINKER, fabrication steps take place consequently and therefore, the quality of each step should be ensured before initializing the following step. The thesis is targeting one particular fabrication step which is the integration of RADAR micro chips, so called bare-dies, in a PCB. At first, a cavity is milled inside the PCB and filled with non-conductive adhesive material. After dispensing the adhesive and while it is still soft, the bare-die is placed on top of the adhesive inside the cavity so that it is planar and in the level of the PCB. Then the adhesive is cured with high temperature to be hardened. The process is shown in Figure 1.1. Due to the nonuniform spreading of the adhesive, there exist some places in the gap between the bare-die and the cavity edges where adhesive is missing and needs to be compensated for to reach uniform height level. Therefore, a repair step is required to fill the gap with sufficient amounts of adhesive

using inkjet printing technology. For the experimental results of this thesis, data were acquired from the bonding of components fabricated by different companies. The PCBs are manufactured by ROBERT BOSCH GmbH whereas the bare dies are fabricated by Infineon Technologies AG. The components are then sent to BESI Austria GmbH where the die-in-cavity bonding process as well as pre and post-curing inspection take place.

1.2. Problem Statement

In the course of this thesis, some challenges are addressed so as to provide an almost optimum approach that aims at closing the feedback loop and repairing the current fabrication step. The bare die is rectangular and has lateral dimensions of $5.65 \times 5.17 mm^2$ and the dimensions of the gap are in micrometers; therefore, achieving high prediction accuracy is of major importance. Similar to accuracy is the prediction robustness which means that by utilizing IL and using live streams of data, the model converges to the same optimal or at least sub-optimal solution as achieved by the batch learning algorithms, as specified in the TINKER project proposal. In a production environment, producing relevant and sufficient data showing process variation to do offline batch training is not possible due to the time constraints in these environments. Therefore, the best approach is to have a pre-trained network that requires a generic dataset and fine tune the network online based on IL concept whenever new data are provided. In this thesis, two models are developed: the gap detection model and the depth estimation model. The gap detection model is trying to separate the gap from the PCB image using image semantic segmentation, while the depth estimation model aims at producing a depth map of the segmented gap image. As the width and the depth of the gap falls in the range of tenths of a millimetre, achieving precise predictions is crucial and quite challenging. Furthermore, for the task of depth estimation, different approaches have been investigated, which will be shown in detail in the methodology section, in order to reach an optimized approach achieving the goal of that task.

1.3. Thesis Outline

The thesis is structured as follows. In the following section, previous related work will be discussed and it is divided into three parts. The first part will give an overview of the latest semantic segmentation methods both binary and multi-class segmentation, which paved the way for the implementation of semantic segmentation in this thesis. The second part will tackle the various approaches of estimating the depth using only a monocular image to dive deep into these approaches and try to find a suitable one for the depth estimation problem addressed in this thesis. The third part will discuss the latest IL methods and the state of the art method. The methods discussed in the literature review reflect back to the methodology of this thesis which will be discussed in Chapter 3. The methodology also includes in logical order a detailed description of all the steps and attempts toward achieving the thesis goal. In Chapter 4, the results of those attempts are presented, discussed, and evaluated in terms of the common evaluation metrics presented

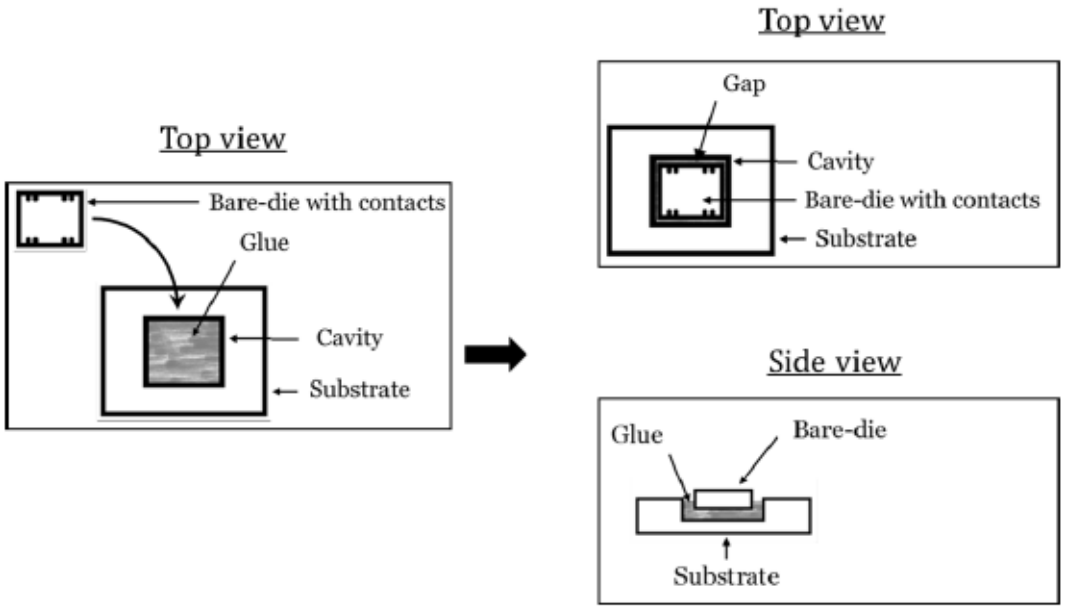


Figure 1.1.: Placement process of bare die in cavity. [6]

in similar research papers as well as in terms of the Key Performance Indicators (KPIs) of the TINKER project. Finally, conclusions, limitations and possible future improvements are presented in Chapter 5.

2. Literature Review

2.1. Overview of Semantic Segmentation Methods

2.1.1. Definition Overview

Semantic segmentation is defined as a pixel-wise labeling in which each pixel is assigned to a certain class; unlike classification, for instance, which assigns the whole image to one class [7]. Hence, semantic segmentation aims at separating the image into several meaningful parts for subsequent processing and visual understanding [8]. Currently, wide range of applications greatly relies on semantic segmentation such as self-driving vehicles, segmenting objects such as humans, cars, and road; and medical diagnosis, extracting tumor boundaries and measuring tissue volumes [7, 8]. The segmentation problem is approached with several techniques such as random forests and conditional random fields (CRF) [8]; However, the focus of this review will be on the deep learning-based methods. Moreover, the common deep learning models and backbone networks used as well as the common evaluation metrics will be all discussed in the following sections.

2.1.2. Review on Deep Learning Models

In this section, we focus on the CNN-based deep learning models since there exists a wide range of deep learning models used in segmentation which makes it very hard to cover them all. The main reasons behind targeting the CNN-based models are that they have proven to achieve outstanding performances on 2D image segmentation [9], and due to the rich literature covering this topic. The method proposed by Long et al. was one of the first CNN-based deep learning models in image segmentation [10]. They proposed a fully convolutional network (FCN) replacing the fully connected layer with up-sampling Conv. layers so that the dimension of the network output is the same as the input image. Depending on skip connections which provide a link between the final layer and lower layers, the FCN model could refine predictions by combining coarse, semantic and local appearance information. Despite being popular and effective, the FCN model has some limitations; for instance, it is not fast enough in case of real-time inference, and it does not efficiently cover global context information [7].

Another popular deep learning model is based on Conv. encoder-decoder network. Noh et al proposed a model known as DeConvNet, see Figure 2.1, which is composed of an encoder consisting of Conv. layers adopted from VGG-16 network, and a decoder consisting of deconvolutional (transposed convolutions) layers [11]. The decoder network takes as input the feature vector resulted from the encoder network, and do pixel-wise labelling to predict the segmentation mask. Another promising model known as SegNet

[12] was proposed by Badrinarayanan et al. which is very similar to the DeConvNet. The novelty in the SegNet resides first in its significantly small number of trainable parameters. Moreover, the decoder network of the SegNet uses memorized max-pooling indices from the corresponding encoder feature maps to perform feature maps up-sampling. These up-sampled feature maps are then convolved with trainable filters to yield dense feature maps. Ronneberger et al. also build upon the FCN model and propose the so called U-net, see Figure 2.2, which architecture from its name has a U shape [13]. The U-net was proposed for segmenting biological microscopy images relying on the use of data augmentation due to the small amount of annotated images. The U-net consists of a contracting path and an expansive path. The contracting part acts as a feature extractor by reducing the dimension of the feature maps while increasing their number, whereas the symmetric expansive path does the opposite by using deconvolutions. In the expansive path, feature maps from the contracting path are concatenated with the corresponding deconvolution layers to preserve and propagate the spatial information of the input image to the final predicted mask. Due to the limited availability of data, high resolution images were augmented to increase data variability, and divided into multiple sub-images to avoid Graphics Processing Unit (GPU) limitations. One last category of CNN-based deep

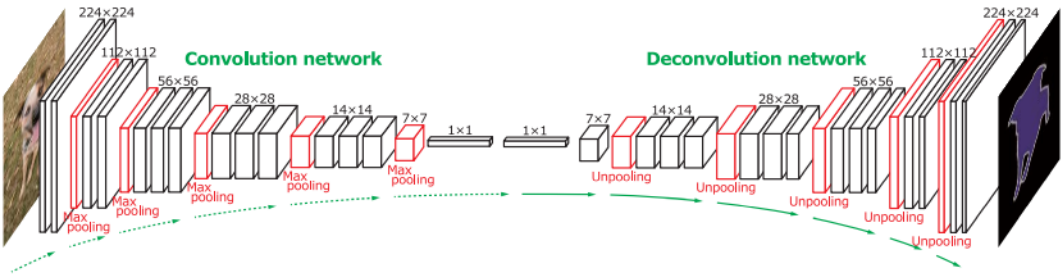


Figure 2.1.: DeConvNet architecture. [11]

segmentation models is the dilated Conv. models [7]. Dilated convolutions introduce a new parameter which is the dilation rate of the Conv. layer which defines the spacing between the weights of the layer kernel. For instance, a 3x3 kernel with a dilation rate of 2 will have the same size receptive field as 7x7 kernel with only 9 parameters, thus keeping the same computational cost. The benefit of using dilated convolutions is the ability of exponentially expanding the receptive fields without losing resolution. Yu et al. have utilized dilated convolutions to do dense image segmentation and produce high resolution predictions [14]. They have also shown that adding dilated convolutions to an existing semantic segmentation model increases the predictions accuracy of that model.

2.1.3. Review on Backbone Networks

When designing semantic segmentation models, researchers often use popular network architectures as the backbone of the encoder network, in case of encoder-decoder segment-

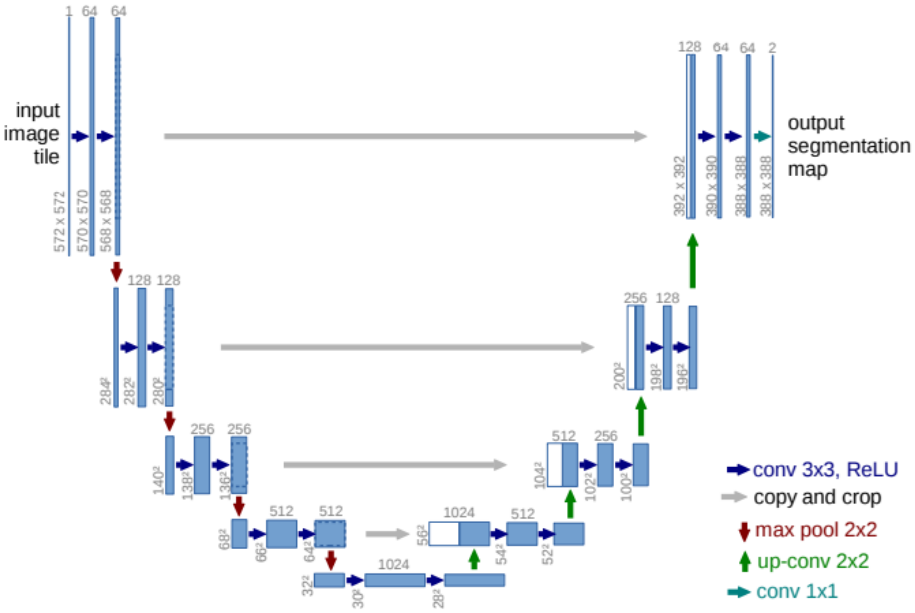


Figure 2.2.: U-net architecture. [13]

ation model, to improve model accuracy. The most commonly used backbone networks are VGG-16, ResNet-34, ResNet-101, and Xception. These backbones are used so that the model can benefit from their pre-trained weights and inherit their powerful feature extraction methods. Alokasi et al. conducted a study to compare the results of applying different backbone networks to the U-net model [15]. When being tested on a specific dataset, Inception V3 backbone achieved the highest Mean Intersection over Union (MIoU), followed by VGG-16 and ResNet-34 respectively, while the original U-net achieved the lowest MIoU. The IoU and MIoU are common powerful evaluation metrics in image segmentation, since they compute the coincidence degree between predicted and actual boundaries and thus reflect the performance of the model directly [16]. Moreover, Zhang et al. performed a comparison between different backbone networks applied in U-net and AD-LinkNet [17] to find the best combination achieving highest segmentation performance [18]. Testing the models on segmenting roads in rural, urban, and coastal scenes, they concluded their performance in Table 2.1.

Network Structure	IoU Score (%)
`Unet-VGG-16`	62.94
`AD-LinkNet-ResNet101`	63.37
`AD-LinkNet-ResNet34`	64.73
`AD-LinkNet-Xception`	64.81

Table 2.1.: Performance Comparison with Different Backbones.

2.2. Depth Estimation Based on Deep Learning

2.2.1. Definition Overview

Estimating the depth information from image datasets is an important task for several applications such as localization, navigation, and object detection [19]. Prior to deep learning, depth estimation was achieved using geometry-based methods or sensor-based methods [19]. The working principle of the geometry-based methods is the recovery of 3D depth information based on geometric constraints in a series of 2D image sequences. One example of these methods is the stereo vision matching which, analogous to the human eye, recovers 3D structure of a scene by observing two images' viewpoints [19, 20]. Moreover, from the name, the sensor-based methods estimate depth information utilizing depth sensors such as LIDAR and RGB-D cameras. On the other hand, deep learning methods based on a single image received recently more attention due to the low cost and wide applications of the monocular RGB cameras [19, 20]. Using various deep learning networks such as CNNs, VAEs, and GANs, monocular depth estimation is investigated and have shown to produce outstanding and accurate predictions [20].

2.2.2. Overview on the Learning Methods

Monocular Depth Estimation (MDE) has been targeted in several applications, and while each application has its own task, it has a unique deep learning methodology. However, these methodologies can be summarized into three main learning methods: supervised, semi-supervised and unsupervised learning [20]. With supervised MDE, depth information is estimated based on a single 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 depth value minimizing the error between predicted and GT depth map. According to [21], the depth estimation of a single image was solved using CNN composed of two components: one that predicts a coarse depth map and another refines that prediction. The output of the coarse-scale network is a blurry depth map containing only a global view of the scene which is then fed to the refining network to yield a refined scene capturing the local details such as objects and edges. The network predictions were evaluated based on the common evaluation criteria, which will be mentioned in a following section, achieving a new state-of-the-art in the time of the paper.

Moving on to the unsupervised learning methods, also sometimes called self-supervised, they usually use the constraints between frame sequences [22, 23] or stereo left and right images [24, 25] as the supervisory signal instead of a direct GT depth map used in supervised methods. Zhou et al. proposed a method which uses a depth network to estimate the depth map using a single image, and a pose network that regresses the transformation between frames for visual odometry [22].

Since unsupervised learning methods lacks the need of GT labels during training, their performance is still far behind the supervised methods. Therefore, semi-supervised learning methods have been proposed to increase the estimation accuracy and enhance the scale consistency while still reducing dependency on GT labels [19, 20]. Semi-supervised

methods are usually used with other information such as synthetic data, surface normals, and LIDAR as relatively cheap unlabeled data[26].

2.2.3. Review on Supervised MDE

In this section, different supervised MDE approaches are discussed as well as the current state-of-the-art approach. Based on what have already been discussed in the previous section, supervised MDE is able to produce an optimum and inexpensive solution for the MDE problem. The approaches that are mentioned in this section can be divided into two categories: single-task and multi-task which is mainly featuring both depth estimation and semantic segmentation as one joint task.

According to Li et al. [27], they developed an estimator that consists of two stages. First, the image pixels are grouped into super-pixels and image patches of different known sizes around the super-pixel center are extracted. Then a deep regression CNN is modeled to learn the relationship between the image patches and corresponding GT depths. Second, the estimated depth is refined from super-pixel level to pixel level using hierarchical CRF [20, 27]. The goal of the CRF method is to map the estimated depth values from the super-pixel level to the pixel level by assuming first that the super-pixel has the same depth value as the center pixel. Furthermore, the estimated depth values of the rest of the pixels are generated taking into consideration common boundary super-pixels, coherence between super-pixels, and local correlation structure. In most of the literature, CRFs are used as a post processing tool for refining the predicted depth maps [28, 29, 30]; however, it is computationally expensive. One significant improvement to the MDE single-task category focuses on the use of CNNs in the shape of encoder-decoder network which have shown an outstanding performance in extracting the spatial features of images [20, 16, 31, 32]. According to Laina et al. [33], they solve the MDE task using only RGB single images and without applying any further refining steps such as CRF. Their network consists of ResNet-50 [34] as an encoder connected to a novel up-sampling blocks acting as decoder. The integration of ResNet-50 network, having the advantage of skip connections, as well as using the novel up-sampling blocks reduced the overall training time around 15%, while the predictions evaluation was shown to surpass the other state-of-the-art method then. Another use of encoder-decoder network was presented in [35], where the authors combined the input image with a sparse depth map to provide the model with the scale information. In that case, it is required to have priory known sparse depth points, which is not always feasible to obtain, so that the developed network can predict the complete depth map. Finally, 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 [36]. This approach produced quite outstanding results and is considered as the current state-of-the-art.

Another set of papers [37, 38] approached the MDE by integrating it with a semantic segmentation task as one joint-task since segmentation and depth estimation have some common features such as detecting object boundaries and recognizing background classes. In this approach [37], the two tasks are done recursively starting with segmentation and then applying depth estimation to improve the depth estimation performance since

instances within each segmented object are more probable to have similar depth values. The architecture used was a bit similar to the one developed in [33], where it consists of residual blocks structured in the encoder and the up-sampling blocks. In [38], the proposed method also does the segmentation task first on two levels: category level where only the background classes are segmented; and instance level where the objects instances are detected. After that, the depth is predicted for each level and then composed together into globally coherent depth map.

2.2.4. Review on Loss functions and Evaluation Criteria

In this subsection, we focus on discussing the commonly used training loss functions which greatly affect the learning process. In addition, we present the most common evaluation criteria which are used in almost all the research papers covering the topic of Depth estimation. Treating the MDE task as a regression problem, the standard pixel-wise L2 loss was used [27, 35] as given in equation (2.1). Eigen et al. and Lee et al. [21, 36]used the L2 loss function as well but with minor modification to make the function scale-invariant as shown in equation (2.2). In order to take into consideration the L1 loss function, Laina et al. and Zhang et al. [33, 37] used the reverse Huber loss function which is a piece-wise function consisting of both L1 and L2 loss functions as shown in equation (3.6). Combining both functions takes the advantage of the L2 loss, which puts high weight towards pixels with high residuals, as well as the advantage of the L1 loss, which has greater impact when the residuals are low. Last, Wang et al. used L1 loss function combined with segmentation loss function since they were solving the MDE task as joint-task with segmentation [38].

$$L2 = \frac{1}{2N}\sum_{i=1}^{N} \|y_i - \hat{y}_i\|_2^2 \tag{2.1}$$

$$\begin{gathered} L2_{SI} = \frac{1}{N}\sum_{i=1}^{N} d_i^2 - \frac{\lambda}{N^2}\left(\sum_{i=1}^{N} d_i\right)^2, \\ d_i = log(y_i) - log(\hat{y}_i) \end{gathered} \tag{2.2}$$

The most popular evaluation metrics used among researchers in the MDE task are: Absolute Relative Difference (AbsRel), Root Mean Square Error (RMSE), Log10 error, Square Relative Error (SqRel) and accuracy with threshold. The mathematical formula of each of these metrics is shown in equation (2.3).

$$\begin{gathered} AbsRel = \frac{1}{N}\sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{y_i} \\ RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|^2} \\ Log10 = \frac{1}{N}\sum_{i=1}^{N} |log(y_i) - log(\hat{y}_i)| \\ SqRel = \frac{1}{N}\sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|^2}{y_i} \\ \text{Accuracy with threshold } (\delta < thr) : \% \text{ of } d_i \text{ s.t. } max(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}) < thr, \\ \text{where } thr = 1.25, 1.25^2, 1.25^3 \end{gathered} \tag{2.3}$$

Where y_i is the GT depth and $\hat{y}_i$ is the predicted depth value for a total of N data samples, in all equations (2.1), (2.2), and (2.3).

2.3. Overview of Incremental Learning Methods

2.3.1. Definition Overview

Incremental Learning refers to a learning process in which machine learning models learn continuously new information while keeping almost all the knowledge they have learnt before [39]. Trying to resemble human learning and thinking, IL allows for memorization and elimination of the need to do exhaustive retraining when receiving new data, and thus increasing memory usage efficiency. IL is mainly trying to fill the gap of catastrophic forgetting which most of the traditionally trained models suffer from when retrained with new sets of data[4, 39]. When retraining a previously trained model, the weights are adjusted by a back-propagation process based on the loss computed on available data, which will reduce the performance of the model on the past data [4]. Since real-life environments are usually dynamic and evolving, an intelligent agent should be able to keep up with these changes. With new streams of data coming out of such environments, retraining the agent from scratch will lead to continuously resetting its knowledge to the current state of the environment. That is why dynamic environments can no longer be modelled using classical learning methods that require big, static, identically distributed and well labeled data, which are very hard to satisfy especially in production environments. As a result, IL has received lots of attention in the recent years, yet it remains a long term challenge [4, 40]. There exist three main types of IL while each type has different implementation methodologies; a brief review of these types and methods are discussed in the following sections.

2.3.2. Overview on IL types

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 [41]. Task-IL refers to the case where the agent is trying to incrementally learn a number of tasks given that they are distinct and clearly distinguishable. This can be achieved, for instance, by specifying an output layer for each task or even developing a separate network for each task. However, the challenge here is to find computationally non-complex methods to use information learnt in one task to improve performance on other tasks.

Second is the Domain-IL which can be described as incrementally learning the same kind of problem but with different contexts and conditions. Regardless of the task, the agent is producing the same possible outputs adapting on possible domain shifts, such as recognizing objects under different lighting conditions or estimating depth in both indoor and outdoor scenes. The challenge of this type is to prevent catastrophic forgetting and improve generalization.

Finally, Class-IL can be referred to as the ability to distinguish between a growing number of classes or objects. It can also be identified as one or more Task-IL episodes, where each episode has a number of classes and the agent should be able to discriminate between classes within an episode and classes from different episodes. For example, the class-IL agent might learn about cars and ships and later about trucks and planes, while in task-IL the agent is not expected to distinguish between vehicles in different episodes (i.e. cars and trucks).

2.3.3. Review on IL Methods

IL addresses the problem of catastrophic forgetting, and promotes learning about new tasks while retaining the knowledge about old ones; thus, all the methods to be mentioned aim for that. Andrade et al. implemented a task-IL method using a Multi layer Perceptron (MLP) based on support vector machine (SVM) for weights adjustment and correction [42]. Their methodology is based on the transfer of weights between a source network and a target network which has one extra node in the output layer representing the addition of a new class. The target network is then retrained for half the number of epochs used while training the source network, and using a training sample which contains data of the new class and data from the original training sample. Lopez-Paz et al. proposed another task-IL model called Gradient Episodic Memory (GEM) that aims at minimizing forgetting and allowing transfer of knowledge forward and backward between tasks [43]. Moreover, Tercan et al. trained an artificial neural network (ANN) using memory-aware synapses to improve quality prediction of different injection molding tasks [5]. Their approach depends on formulating a special training loss function consisting of different parameters to direct the IL process based on the so called importance values. A similar approach is also discussed in [44, 45], where the main novelty resides in the loss function and the way of choosing its hyper-parameters to incrementally learn to segment more objects in an image.

As an example for domain-IL, Verma et al. proposed a methodology called Efficient Feature Transform (EFT) which can be generalized to a variety of network [46]. The main feature of EFT is that the network is partitioned into a global network, which can be any pre-trained architecture that is kept frozen during the training process, and task specific transformations which weights are trained and customized to the specific task. They also utilize a loss function that is composed of two parts; one for learning the new task and the other for not forgetting about old tasks. The method was tested on CIFAR-100 and ImageNet datasets. Mirza et al. continue the discussion on domain-IL with an approach for autonomous driving in all weather conditions [47]. Their method is based on the first and second order of domain statistics as the supervisory signal; which are recorded for each task. So whenever a new task arrives, the statistical vectors for that task is plugged into the model which correspondingly starts to perform well on that task.

For class-IL, Kolesnikov et al. proposed a classical approach which can learn classifiers and feature representations at the same time [48]. More efforts are concluded in [49, 50] aiming at improving the class-IL efficiency. The thesis is targeting the domain-IL, since the segmentation and depth estimation problems are not changing; however, the conditions

and contexts of data used for both models are changing. Based on the methods of literature, a feasible methodology targeting the project KPIs is chosen for this thesis. Furthermore, it is worth mentioning that the term online learning is sometimes used in literature interchangeably with IL. However, in the scope of this thesis, online learning is regarded as a training method in which minimum amount of data (i.e. one data sample) is provided to the model in each incremental step, so that the model would undergo as many incremental steps as the number of provided data samples.

3. Methodology

3.1. Data Pre-processing

3.1.1. Data Description

Before diving deep in the methodology of the thesis, the used data should be well presented and described as well as the excessive data processing that has been implemented. The models developed in this work use real data coming out of the die in-cavity bonding process as well as pre and post curing inspection processes carried out in the premises of BESI Austria GmbH. The image data of interest were recorded after the bonding of eight RADAR PCBs, each has six bare-dies, summing to a total of 48 units, which will be referred to as substrates in the context of the thesis. For all the substrates, monocular images are recorded after curing the glue; while for only 30 substrates, before-curing images were recorded. The images were provided at different intervals of time; therefore, they have different quality features and illumination conditions due to the continuing development of the inspection system. Moreover, only for 24 substrates, 3D depth images were recorded using microscopic camera. Table 3.1 summarizes the total amount of data and their specifications. The first 18 images of the after-curing substrates were recorded as part of an early attempt; then using a higher quality camera, images of 30 substrates were recorded, before and after curing each with two different illumination conditions. After that, more images of only 24 of the 30 latest substrates were recorded as well as their 3D depth data using a microscope camera at PROFACTOR GmbH. The monocular images are of high resolution and their sizes range from 768x768 (microscope images) to 2219x2219 (HQ monocular images), whereas the 3D depth images have resolution of 768x768.

3.1.2. Data Preparation and Processing

In order to prepare the image data for training the gap segmentation model, GT masks should be generated. First of all, using GIMP software tool, the cavity mask and the chip mask of one sample image were manually extracted. Then, using both masks, template matching was applied to the rest of the images to extract the chip and cavity masks in each image. Subtracting both masks yield the mask of only the gap in between which is the target GT image. Template matching is an OpenCV library which, given an image and a template, tries to find the location of the template inside that image. However, the process needs lots of arguments' tuning and requires that the template be almost the same among all images. For instance, if there is a slight difference in illumination between two substrate images, using one template for both images would not work. Figure 3.1 is a

Data Type	Data Amount	Recording Conditions
Monocular Images	18 substrate images (after curing)	Monocular camera
	30 substrate images (before curing)	High quality monocular camera applying two different illumination conditions
	30 substrate images (after curing)	High quality monocular camera applying two different illumination conditions
	24 substrate images (after curing)	Microscope camera
3D Depth Images	24 images (after curing)	Interferometry Microscope camera

Table 3.1.: Data breakdown.

diagram showing the labelling process for generating gap masks. All the images and labels are then cropped to the nearest size divisible by 256 and then divided into non-overlapping patches of size 256x256 to avoid GPU memory limitations. Now that the image data for gap segmentation are prepared and GT masks are generated, the depth data should be prepared as well for training the depth estimation model. As already mentioned, the final goal within the TINKER project is to repair each substrate by printing the missing amount of adhesive inside the gap. To do this, there should be a standard reference to which all images relate to make the printing process accurate. Moreover, in each substrate image, the bare-die is slightly rotated as a result of the larger cavity and the soft glue underneath. To mitigate all these problems, each image is counter rotated to compensate for the bare-die rotation and then cropped from the two corner circles' centers. In order to detect the rotation of each bare-die, the corners of the chip should be detected either using template matching or Harris corner detection method. Both methods have advantages and drawbacks; however, template matching was preferred because it can be generalized to more images and has less computations. To crop the images, the corner circles were detected using circle Hough transform which tries to find all the candidate circles in an image based on specified radii range and distance between centers. The same processing is also applied to the corresponding 3D depth images. Finally, the HQ monocular images were resized to the size of the depth images. Figure 3.2 shows one data sample before and after applying the mentioned processing.

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 number of incremental steps. Therefore it takes around 2.5 hours based on the number of epochs and the total number of training samples in each incremental step. All the DL models were coded using python programming language and trained using the TensorFlow framework.

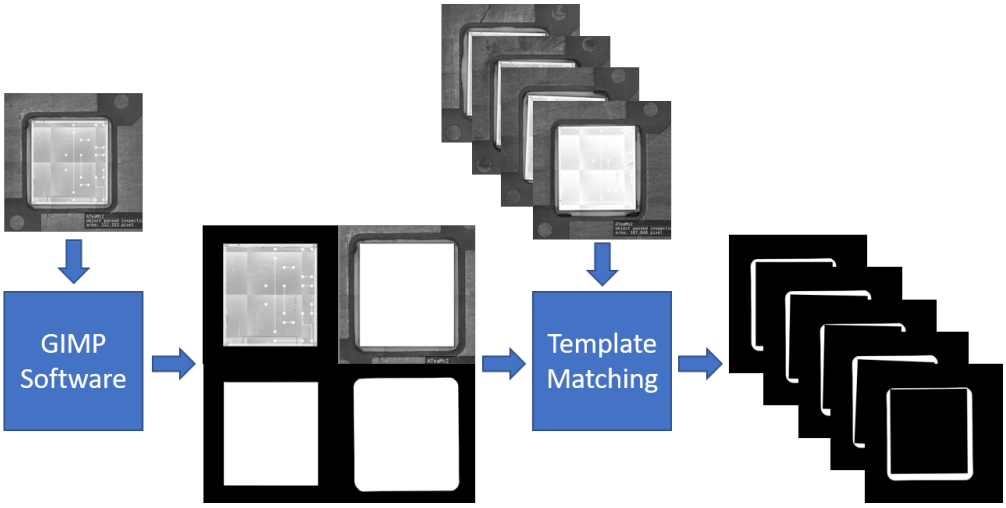


Figure 3.1.: Process of generating GT gap masks.

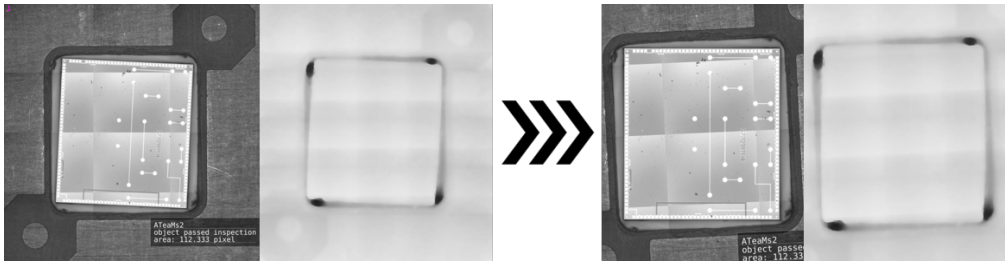


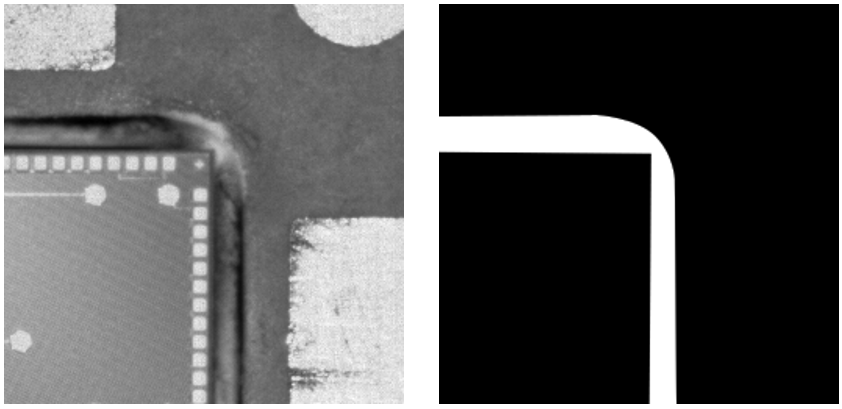
Figure 3.2.: Data sample of a 2D image and the corresponding depth map before and after processing.

3.2. Gap segmentation

3.2.1. Model Selection and Training

Based on the literature review, we learnt that there are several networks used for semantic segmentation, yet the U-net was the most famous one especially when the training data are few. Different backbone networks have been tested to see which would achieve better performance with the U-net. Those were VGG-16, ResNet-34, and inception-v2 which had the least number of training parameters when compared to other networks mentioned in the literature such as ResNet-101 and Xception. The U-net was imported from keras built-in models, assembled with each backbone network, and trained using the pre-processed image data. The first model was the base model and was trained with only the first 18 substrate images, since this was the initial dataset, and the rest of the images are supplied to the model incrementally. Implementation of IL will be discussed in details in the fine tuning section. Patching the 18 images produced a total of 450 images of equal size (256x256), then they were split into training, validation, and testing sets:

70%; 15%; 15% respectively. Figure 3.3 shows a sample training tuble after patching the images. Data augmentation was applied on the training and validation sets to enhance generalization and avoid overfitting. 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 [51] presented in Equation (3.5) and a constant learning rate of 1×10^{-4}. Because we are trying to segment only the gap, the segmentation task in this work is binary which is labelling pixels as white if inside the gap, or else black. The training loss function used in the gap segmentation model consists of a binary cross entropy (BCE) loss and Jaccard loss, which is commonly referred to as IoU loss function [52]. 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 as shown in Equation (3.1).



(a) Gap image (b) Gap mask

Figure 3.3.: A sample training tuble for the gap segmentation model.

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i) \tag{3.1}$$

Where y_i is the true value either 1 or 0, and $\hat{y}_i$ is the predicted probability of the pixel belonging to the gap for all N data samples. The IoU function measures the similarity between finite sample sets A,B as the intersection over union as shown in Equation (3.2). If the two entities are disjoint, the IoU equals zero, while it equals 1 if they are identical. For minimization purposes, the jaccard loss is defined as in Equation (3.3). The IoU score in (3.2) is used as the evaluation metric of the model.

$$IoU = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{3.2}$$

$$L_{jaccard} = 1 - \frac{|A \bigcap B|}{|A \bigcup B|} \tag{3.3}$$

3.2.2. Model Predictions Refinement

The output predictions of the gap segmentation model need some sort of post-processing to ensure there are no black pixels inside the gap, which should be all in white, and to eliminate any outlier pixels. A perfect processing candidate for those tasks is the morphological operation using OpenCV library. They consist of several functions: erosion, dilation, opening, and closing. The idea of erosion is to erode the boundaries of a foreground object, usually in white; or in other words, thinning the object and removing very small scattered noises. Dilation operation does the opposite, and is very useful in filling small voids (whitening black pixels) in the object. Opening operation is just erosion followed by dilation, while closing is dilation followed by erosion. For the purpose of refining the predicted gap images, opening and closing were applied to remove noise and outlier pixels first, and then fill any black voids inside the boundaries of the gap, while preserving the size of the gap. However, testing the segmentation model's predictions was done without applying these operations. Results of morphological operations can be seen in Appendix A.

3.3. Depth Estimation

3.3.1. Model Selection and Training

For depth estimation, several approaches were investigated as to find the best suitable one for the task. We will discuss each approach in terms of data preparation, model selection, and hyper-parameters. All approaches were evaluated using the same popular evaluation metrics that were mentioned in the literature review in section 2.2.4. Since the gap mask has been extracted from the gap segmentation model, all substrate images and 3D depth maps are multiplied by the mask to get only the gap contour as shown in Figure 3.4.

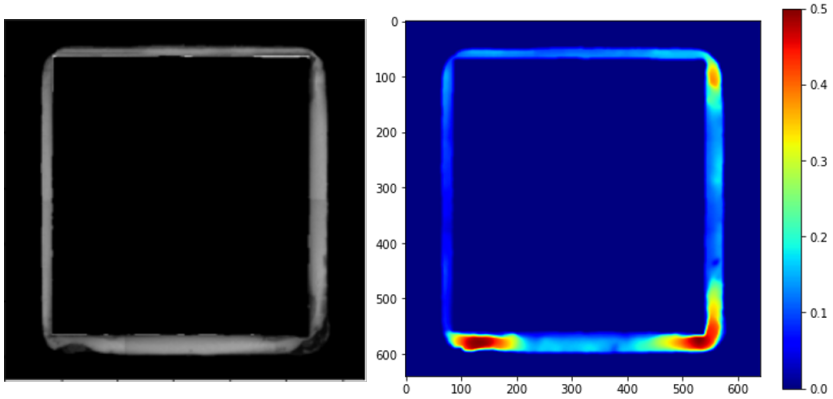


Figure 3.4.: A sample of the input-output tuble for MDE.

First, a pixel-wise regression model utilizing a CNN architecture was used. This model tries to predict the depth value for each pixel, and so the input images were divided into

small patches of size 65x65, whereas the target labels were the depth values of the center pixels of these patches. Using this approach, one could generate a huge dataset, equal to the number of pixels of the all images. However, the size of the input image should be equal to the size of the corresponding depth image, which requires resizing the high resolution input images to the small size of the depth maps. The model architecture is shown in Figrue 3.5; it consists of three double Conv. layers, each of which is followed by an average-pooling layer. Then the output is flattened and connected to three fully connected layers, each followed by a dropout layer. The hyper-parameters of the model were manipulated to get the best out of this model including the number of Conv. layers, number of filters, and number of fully connected layers. The model used the Mean Squared Error (MSE) as the training loss function (3.4), and Adam optimizer with a learning rate set initially at 1×10^{-4} and decayed exponentially with a rate of 0.9 every 25 epochs. At the end of the 100 training epochs, the learning rate had been decayed to the value of 729×10^{-7}. Equation (3.5) shows the weights optimization based on Adam optimizer and the decaying learning rate.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.4}$$

$$\begin{gathered}\Theta_t = \Theta_{t-1} - \eta.\Delta\Theta, \\ {}^{a}\eta = lr_0 \times decay_{rate}^{(step/DecaySteps)}\end{gathered} \tag{3.5}$$

Where Θ_{t-1} and Θ_t are the trainable weights at time $t-1$ and t, respectively. $\Delta\Theta$ refers to the Adam optimization function based on the gradient of the loss function with respect to the trainable parameters. The model was trained for a total of 100 epochs. The model produced good and accurate pixel-wise depth prediction; however, it showed poor generalization on the unseen data.

In the second attempt, an encoder-decoder network was implemented; more specifically, a U-net with residual blocks employed in both encoder and decoder networks similar to what was done in [33]. Both 2D monocular images and corresponding depth maps were divided into smaller patches of size 384x384. Moreover, the images were augmented with a random flip to improve data balance since higher depth values usually appear in the gap bottom corners. Unlike the previous model, this model directly predicts the depth map of a given monocular image while transferring the spatial information of the image to the depth map. The structure of the encoder is composed of five down-sampling blocks; each block consists of 2 Conv. layers with kernel sizes 3x3 added with a residual Conv. skip connection with kernel size 1x1. After addition, an activation layer is applied and then a max-pooling layer for halving the dimension. The decoder network consists of four similar up-sampling blocks; the only difference is replacing the max-pooling layer with an up-sampling layer for up-scaling. Each block in the encoder, apart from the last block, is concatenated with the corresponding decoder block for the transfer of image spatial information. The Res-unet architecture is shown in Figure 3.6. The reverse Huber loss function (berHu), in equation (3.6), was used in this attempt for computing the gradients as it combines both L1 and L2 loss functions and benefits from the advantages of both

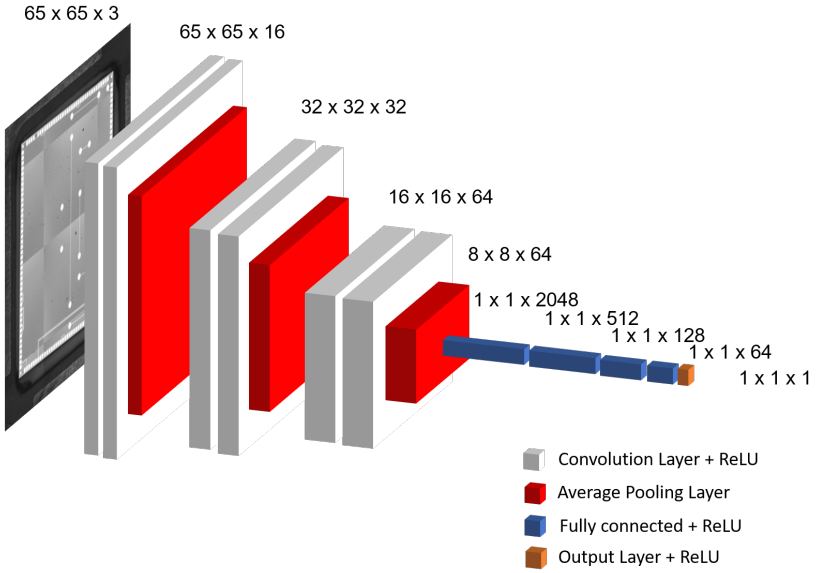


Figure 3.5.: The CNN architecture of the first MDE attempt.

functions.

$$berHu = \begin{cases} |\Delta y| & |\Delta y| \leq c \\ \frac{\Delta y^2 + c^2}{2c} & |\Delta y| > c \end{cases}, \qquad c = \frac{1}{5} max_i(|y_i - \hat{y}_i|) \tag{3.6}$$

Where y_i is the GT depth, $\hat{y}_i$ is the predicted depth value, and Δy is the computed error between them. Adam optimizer was used for optimizing the gradients with a constant learning rate equal to 1×10^{-4} throughout the 100 training epochs. The Res-unet model produced better results and showed very good generalization on unseen images as well; however, it showed non-negligible error in total depth volume between predictions and GTs.

The third attempt builds upon the second and does not have much changes. Inspired from [37, 38], the segmentation model used for detecting the gap was used while freezing the encoder layers and retraining the decoder layers. This way, the depth estimation model can directly utilize the encoder, which was trained to extract the features and the spatial information of the images, and with much less training time. The model uses a new loss function which is a combination of three loss functions similar to what was done in [54]. The total loss function is presented in 3.7.

$$Combined = \alpha_1 L_{SSIM}(z, \hat{z}) + \alpha_2 L_{MAE}(z, \hat{z}) + \alpha_3 L_{reg}(z, \hat{z}) \tag{3.7}$$

where $\hat{z}_i$ is the predicted depth map and z_i is the true depth maps. L_{SSIM} is the Structural Similarity Index which computes similarity between images on the basis of luminance, contrast, and structure 3.8; and $L_{MAE}(z, \hat{z})$ is the mean absolute error for pixel-wise comparison 3.9. Moreover, L_{reg} is the Depth Smoothness loss which compares the two

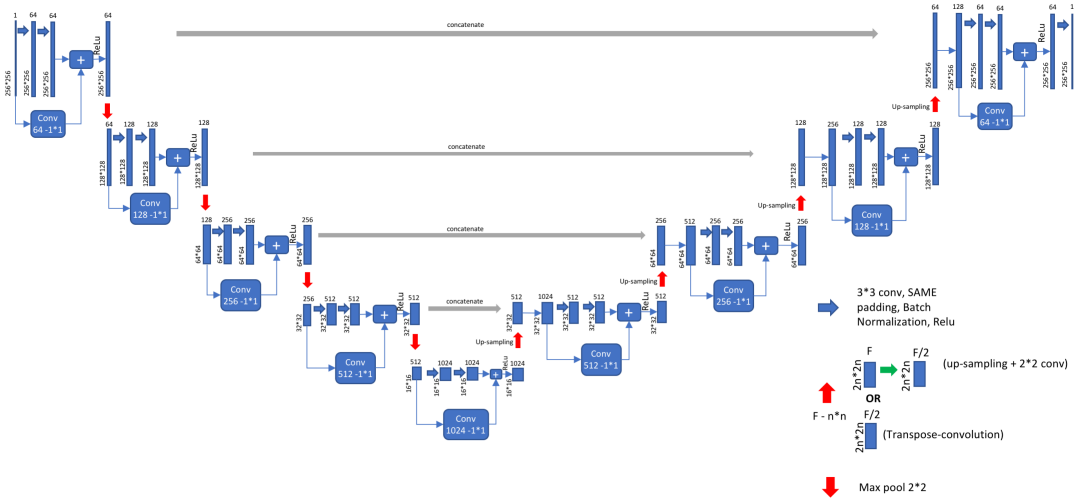


Figure 3.6.: The res-unet architecture. [53]

depth maps based on their gradients in x and y directions as in 3.10. Inspired from [55], the total L_{reg} loss is the mean of the gradients' sum over N pixels as shown in 3.11. α_1, α_2, and α_3 are weighting factor set empirically to 0.85, 0.1 and 0.9, respectively. After being trained for 100 epochs, the final model produced the best performance especially in terms of the error of the total depth volume as will be shown in results section.

$$L_{SSIM}(z,\hat{z}) = \frac{1}{2}(1 - \frac{(2\mu_{\hat{z}}\mu_z + c_1)(2\sigma_{\hat{z}z} + c_2)}{(\mu_{\hat{z}}^2 + \mu_z^2 + c_1)(\sigma_{\hat{z}}^2 + \sigma_z^2 + c_2)}) \tag{3.8}$$

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

$$L_{MAE}(z,\hat{z}) = \frac{1}{N}\sum_{i}^{N}|z_i - \hat{z}_i| \tag{3.9}$$

$$\begin{aligned} S_x &= \bigtriangledown_x \hat{z}_i \times e^{\frac{1}{N}\sum_i^N |\bigtriangledown_x z_i|} \\ S_y &= \bigtriangledown_y \hat{z}_i \times e^{\frac{1}{N}\sum_i^N |\bigtriangledown_y z_i|} \end{aligned} \tag{3.10}$$

$$L_{reg}(z,\hat{z}) = \frac{1}{N}\sum_{i}^{N}|S_x| + |S_y| \tag{3.11}$$

3.4. Fine Tuning Approaches

Given that the gap segmentation model and the MDE model have been established and trained with initial datasets, this section discusses how to deal with new coming data that

should be fed to the model. Moreover, it will discuss two main techniques of supplying the new data to the trained models in order to fine tune their parameters accordingly.

3.4.1. Training Samples Selection

As already mentioned in section 3.1.1, there are four sets of substrate images; each with one or more different settings affecting the quality and appearance of these images. Moreover, one substrate might have more than one image; however, each has different image features. Figure 3.7 shows a sample image from the four different datasets available, where the second and forth images represent the same substrate in different recording settings. Therefore, the gap segmentation model has to be trained on all the four datasets to be able to have good performance on them all. Moreover, because the GT depth images of the second and forth sets are available, the MDE model should be able to perform well on both sets as well.

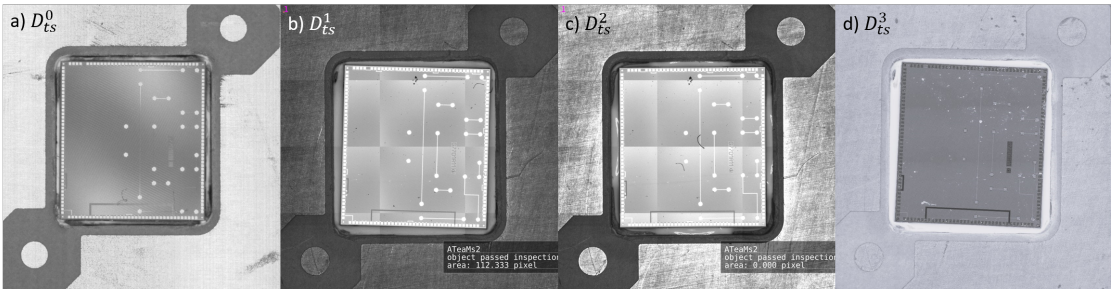


Figure 3.7.: A sample image from each dataset in the order of usage.

For retraining the gap segmentation model with a new dataset, sample images are selected that almost present all the features in that dataset and have as minimum noises and outliers as possible. A sample selection algorithm was implemented for that task, which aims at selecting few images, five in particular, from a dataset based on three functions. The first function finds the difference between the image histogram before and after histogram equalization and sorts the images accordingly to choose images with well distributed pixel values. The second function sorts the images in terms of the volume of the glue material, roughly estimated after applying a threshold function. This attribute aims at eliminating the images in which the glue material was printed by mistake outside the cavity boundaries, which provides false gap information to the model. The third function finds the rotation angle of the bare-die in the image in order to select images with various chip tilt angles. Each function yields some candidate images, then the most common five images among those candidates are selected. For the third dataset, the last function was not applied since there was almost no bare-die rotation, which occurred during the curing of the glue. On the other hand, since there is only two sets of images whose depth information is available, the data were fully supplied to the MDE model without any sample selection processing. The reason is that depth data are already few and estimating depth from monocular images is already a complex task.

3.4.2. Transfer Learning

Transfer learning is a very commonly used approach when we need to transfer the learnt knowledge of one model to another. To achieve this, the previously trained weights of a base model are transferred to another model which is trying to do a similar task to the one the base model was trained on. For example, the trained weights of a model classifying between cats and dogs can be used to train a model classifying between horses and cows. In fact, transfer learning is implemented in the gap segmentation and the MDE models as they were already trained with pre-trained weights instead of randomly initialized weights. Using pre-trained weights, the new model is able to converge to a good fit much faster than if it was trained without those weights. That being said, transfer learning might provide a solution for updating both the segmentation model and the regression model in the presence of new data, and that is why it was investigated.

First, a new segmentation model, exactly the same as the base segmentation model, was created and initialized with the same weights of the base model. After applying the same data pre-processing mentioned before, the model was retrained using selected images from the second dataset for only 30 epochs. The model achieved good performance on unseen images from the current dataset, yet the performance on the base dataset has declined noticeably. The same step was repeated with the rest of the datasets. After each step, the performance of the model on old datasets was declining drastically, which is the definition of catastrophic forgetting. The performance of the models utilizing transfer learning is presented in the results section.

3.4.3. Incremental Learning

In the scope of this thesis, the IL task belongs to the Domain-IL type in which the model tries to learn the same problem and doing the same task but in different domains and conditions. To be more specific, the gap segmentation model and the MDE model should achieve optimum performance on the different available datasets which were recorded in different settings. The models are trained once using one dataset, and then the remaining datasets are supplied incrementally to the model representing streams of new data that resembles a change in a production environment. The methodology of this thesis is adopted from [44, 45] in which the authors apply incremental learning to a segmentation task. In this thesis, we try to apply this method to both segmentation and regression models, since it is applicable to any deep network architecture and it has shown outstanding results. The training scheme of the method is shown in figure3.8, where we have a base pre-trained model that 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 total loss function is shown in Equation (3.12).

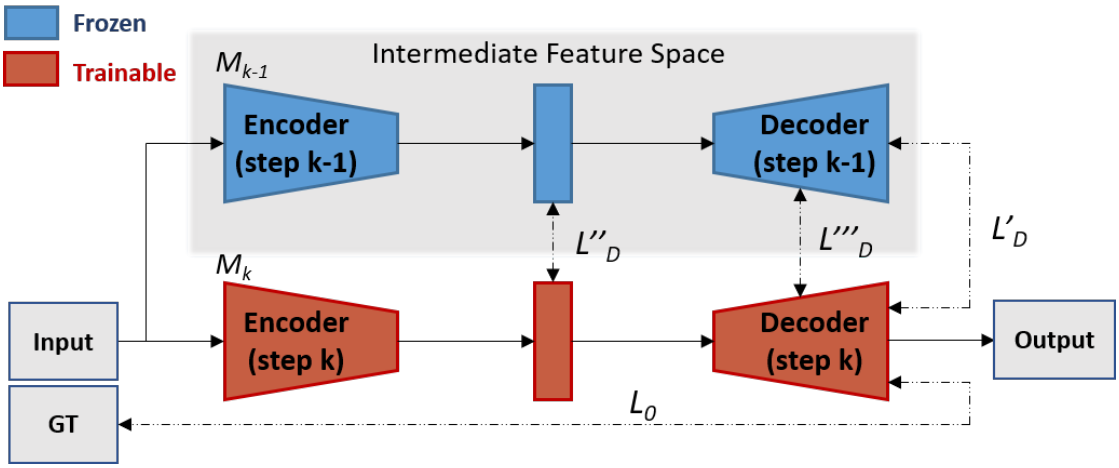


Figure 3.8.: Knowledge distillation scheme of the IL method at k-th incremental step. [44]

$$L_{total} = L_0 + \lambda_D L_D \tag{3.12}$$

Where L_0 is the initial loss function used for learning the new information, 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 old information. For gap segmentation task, BCE combined with jaccard loss funtion was used for learning new datasets' information. Equation (3.13) shows the loss function.

$$L_{seg_0} = -\frac{1}{N}\sum_{i=1}^{N} Y_i log(M_k(x_i)) + (1 - Y_i) log(1 - M_k(x_i)) - 1 + \frac{|Y_i \bigcap M_k(x_i)|}{|Y_i \bigcup M_k(x_i)|} \tag{3.13}$$

Where Y_i is the GT, x_i is the data input, and $M_k(x_i)$ is the prediction of the k-th incremental model, for N data samples. The first distillation loss function L'_{seg_D}, shown in Equation (3.14), is applied to the output predictions of the models, M_k and M_{k-1} in order to compare both of them and apply required penalty to keep the $k-th$ model from being biased to the new dataset's information. Because L'_{seg_D} is applied to a classification layer, BCE was used combined with jaccard loss as well. This combination was preferred based on empirical results.

$$\begin{aligned} L'_{seg_D} = -\tfrac{1}{N}\textstyle\sum_{i=1}^{N} M_{k-1}(x_i) log(M_k(x_i)) + (1 - M_{k-1}(x_i)) log(1 - M_k(x_i)) \\ -1 + \tfrac{|M_{k-1}(x_i) \bigcap M_k(x_i)|}{|M_{k-1}(x_i) \bigcup M_k(x_i)|} \end{aligned} \tag{3.14}$$

Where $M_{k-1}(x_i)$ is the base model prediction. 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, the same distillation loss function as L'_{seg_D} cannot be applied. Another loss function is needed that should keep the

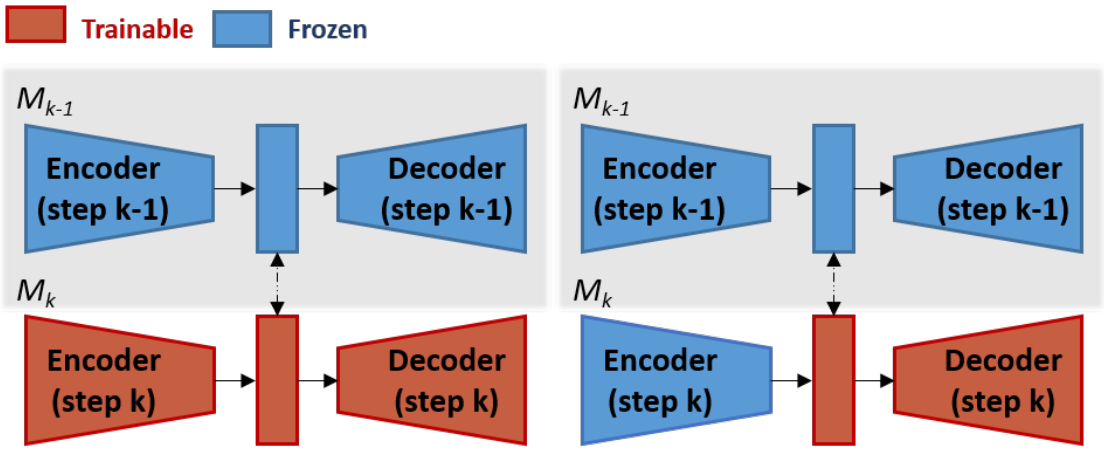
two feature spaces as close as possible such as the standard L2 loss function. Denoting the encoder of the model as E, the second distillation function can be written as in Equation (3.15).

$$L''_{seg_D} = \frac{1}{N}\sum_{i=1}^{N} \|E_{k-1}(x_i) - E_k(x_i)\|_2^2 \tag{3.15}$$

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 second distillation function was used since the output of the decoder layers of each model should be kept close as well. The final distillation loss function can be rewritten as in Equation (3.16).

$$L'''_{seg_D} = \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{3} \frac{\left\|d^l_{k-1}(x_i) - d^l_k(x_i)\right\|_2^2}{3} \tag{3.16}$$

In addition, another distillation feature can be added which is keeping the encoder of the $k-th$ model frozen (E_F) while learning the new information through the decoder only as illustrated in Figure 3.9. This aims at preserving the feature extraction capabilities of the encoder. That being said, more than one distillation function can be used in the same incremental step if needed. For example, freezing the encoder and applying the distillation loss function on the output layers of the two models were used in the first attempt. The second distillation loss function was applied alone once and then combined with the first distillation loss function without freezing the encoder, since the loss is applied on the encoders' outputs. The third distillation loss function was only implemented on its own due to high computational costs. Choosing the suitable distillation criterion depends on the complexity of each incremental step.



(a) Encoder trainable (b) Encoder frozen E_F

Figure 3.9.: The two states of the encoder at the k-th incremental step. [44]

Aside from the distillation criterion, optimizing the hyper-parameters is essential to optimize the model's performance and reduce the computational cost. First of all, the effect of the distillation factor λ_D needs to be modelled and carefully chosen to prevent the model bias towards any of the datasets. Moreover, its value is proportional to the number of incremental steps and to how similar one dataset is to another in terms of spatial features. The learning rate is crucial as well, since its value defines the size of the learning steps that the model takes to reach the global minima of the learning curve. Going through the incremental steps, the learning curve is becoming more and more complex; and the more features and new information is added, the less the value of the learning rate. The number of training epochs in each incremental step reflects the computational cost, that is why it should be kept as minimum as possible. The optimum number of epochs reached empirically was 30 epochs in each incremental step.

The same incremental procedure can be applied to the MDE model because it uses the same network architecture; however, the loss functions are different since this is a regression problem. For both the initial loss function L_{MDE_0} and the first distillation loss function $L^{'}_{MDE_D}$, the combined loss in 3.7 was used. For the second and third distillation loss functions, $L^{''}_{MDE_D}$ and $L^{'''}_{MDE_D}$, berHu (3.6) loss was used given that both the encoders and the last decoders' layers produce regression outputs which values should be kept similar. The developed MDE model went through one incremental training step since there are depth labels for only two image datasets. The two image datasets are describing the same substrates but are different in terms of the recording setup, and consequently they have different resolution and different features. Therefore, the depth measurements of both datasets are actually the same. The incremental training was concluded in 45 epochs which is about half the number of epochs needed for training the base MDE model.

4. Experimental Results

4.1. TINKER KPIs

In TINKER, the targeted KPIs 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. For both gap segmentation and depth estimation, the targeted KPIs focus on the deviation of the model 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 KPIs 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 presented in 2.3, which is used as an evaluation metric for the MDE models. Therefore, the target absolute relative error value of the MDE models is 0.1.

KPIs	Value	Description
Gap segmentation	$MIoU > 90\%$ of true label	The percentage of true predictions represented by the IoU metric
Depth estimation	Deviation $< 10\%$ of true depth values	The deviation of the predicted depth values relative to the true vales represented by the AbsRel metric

Table 4.1.: Summary of identified KPIs with respect to models' predictions.

4.2. Gap Detection

4.2.1. Base Model Selection and Training

In this section we focus on the results of the gap detection base model, including comparative evaluations of various attempts and models. In addition, the results of the best reached model are presented including more comparative evaluations of various hyper-parameters in order to conclude their effects. As mentioned in the methodology section, the U-net was used for the gap segmentation task with three candidate backbones: VGG-16, ResNet-34, inception-v2. Comparison between the backbones is conducted in terms of training parameters, training time in minutes of 75 epochs, precision (4.1), recall (4.2), and mean IoU score. The evaluation metrics are computed on unseen testing images.

The precision and recall metrics reflect the performance of the model in an unbiased fashion evaluating the performance of the model on pixels belonging to the gap. Since the class of gap pixels is small within the image, computing the accuracy, which reports the percent of pixels that are correctly classified, provides misleading model evaluation. On the other hand, precision reports how many of the predicted gap pixels actually belongs to the gap; while recall reports how many of actual gap pixels the model was able to correctly predict. Table 4.2 reports the quantitative comparison between the three models. Moreover, Figure 4.1 shows the training and validation learning curves of the models. Qualitative comparison is reported as well in Figure 4.2. From these comparisons, it can be noticed that the all models achieved similar evaluation scores, yet they differ in terms of the training time. That is why U-net with ResNet-34 was preferred due to the less training time.

In order to test the effect of the data processing functions and hyper-parameters tuning, an ablation study was conducted utilizing the U-net with ResNet-34 backbone. Table 4.3 reports the effects of such functions and parameters on the performance of the model. The variables in the table are manipulated one at a time while keeping the other variables the same as in the original model, whose performance is reported in table 4.2. Removing data augmentation functions, which is the first variable, led to a slight drop in the performance of the model. Since the dimension of one patch image is less than the dimension of the bare-die, negative patches (NP) that do not contain any gap information were present in the dataset. Removing those patches serves to reduce the class imbalance since the number of non-gap pixels is much larger than that of the gap pixels. Freezing the encoder of the model and learning through the decoder parameters only showed a drop in the model performance as well. Finally, a learning rate scheduler was applied to reduce the initial value of the learning rate (1×10^{-4}) exponentially with a decay rate of 0.5 every 10 epochs. Decaying the learning rate resulted in a slow convergence and therefore was not preferred.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

Where True Positive (TP) is the number of pixels actually belonging to the gap and were correctly predicted, False Positive (FP) is the number of pixels not belonging to the gap and were falsely predicted as gap pixels, and False Negative (FN) is the number of gap pixels that were falsely predicted as non-gap pixels. Note that the non-gap pixels which were correctly predicted is not considered in any of the two metrics.

4.2.2. IL Implementation

After training the base model on the initial dataset, the other datasets are fed incrementally to the model. The four datasets will be refered to as the following D^n where $n \in \{0, 3\}$. Each dataset is divided into parts: training D^n_{tr} and testing D^n_{ts} with percentages of 80% and 20%, respectively; whereas one third of the training data are used for cross validation.

	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%
Inception-V2	30M	13.63 mins	97.91%	98.66%	96.63%

Table 4.2.: Performance comparison between VGG-16, ResNet-34, and Inception-V2 backbones.

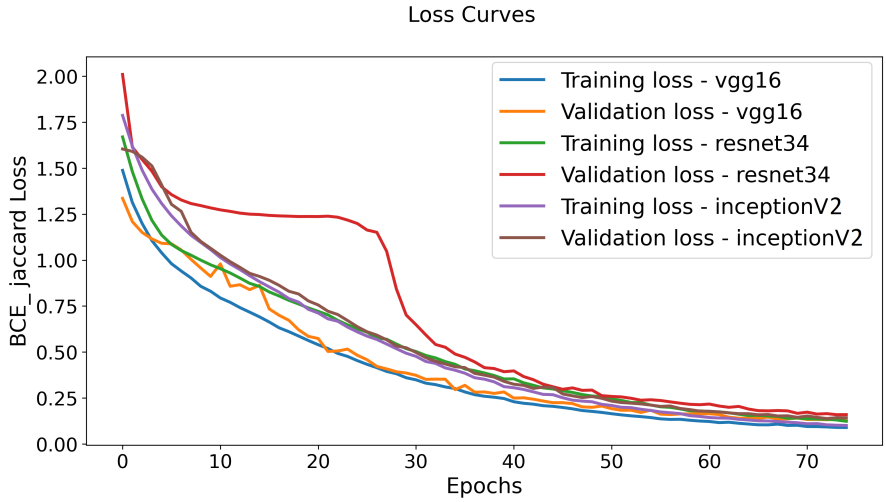


Figure 4.1.: The loss curves of the U-net with VGG-16, ResNet-34, and Inception-V2.

In this section, the results of the incremental learning steps are presented and discussed. Moreover, they are compared with results obtained using transfer learning as well as results obtained from a standard batch learning model trained with samples from the current and the previous datasets. In the first incremental step, the model structure inherits that of the base model which was trained using the old dataset (D^0_{tr}). The model is retrained on (D^1_{tr}) utilizing different distillation functions. Moreover, three values of the distillation factor λ are tested which are [0.5, 1.0, 1.5], and constant learning rate values of [1e-4 and 1e-5] are used. Each incremental step was given 30 epochs of training. In table 4.4, the resulted scores of the highest-scoring models are reported. Using the first distillation loss function while freezing the encoder achieved a fine performance especially in maintaining the old knowledge presented in dataset (D^0_{ts}). On the other hand, the second and third loss function showed a worse performance in maintaining the old knowledge; even though, they produced better scores on the new dataset (D^1_{ts}). Combining both $L^{'}_{D}$ and $L^{''}_{D}$ and increasing the value of λ to 1.0, the model was able to perform well on both datasets achieving almost the same scores as $M_0(D^{\{0,1\}}_{tr})$. This is because increasing the value of λ leads to higher distillation loss value and thus gives the

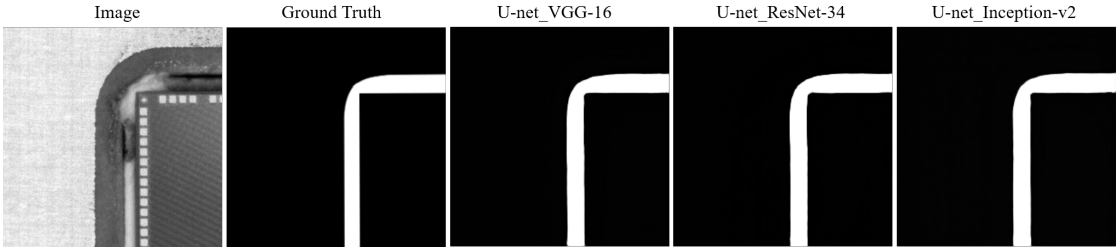


Figure 4.2.: Qualitative comparison between the prediction performance of the U-net with different backbones.

	precision	recall	MIoU
Removing Aug	95.83%	95.74%	82.56%
Keeping NP	92.05%	99.72%	27.91%
E_F	86.45%	99.35%	79.69%
Decaying LR	85.66%	98.79%	41.92%

Table 4.3.: Data processing functions and hyper-parameters tuning give a clear insight about the performance of the U-net-Inception-v2 model.

model a stronger bias towards maintaining performance on old data.

Furthermore, model retraining utilizing the concept of transfer learning was performed. The first base model, $M_0(D^0_{tr})$ trained only on D^0_{tr}, was retrained with D^1_{tr} while freezing the encoder. In table 4.4, the last row shows the evaluation of that model achieving low scores on the old data, which is known as catastrophic forgetting. Since the concept of transfer learning has proven to suffer from catastrophic forgetting already from the first incremental training step, there was no need to do further training with the remaining datasets.

	base dataset(D^0_{ts})			1st new dataset(D^1_{ts})		
M_1	precision	recall	MIoU	precision	recall	MIoU
$L^{'}_D, E_F$	79.10%	**98.70%**	85.76%	93.70%	93.60%	86.86%
$L^{''}_D$	84.65%	89.32%	77.04%	94.54%	**97.26%**	**92.09%**
$L^{'''}_D$	66.03%	93.35%	64.40%	**94.96%**	94.81%	90.28%
$L^{'}_D, L^{''}_D$	**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%
TL, E_F	65.55%	86.30%	59.48%	92.25%	96.28%	89.00%

Table 4.4.: Performance measures of the segmentation model on previous datasets after the first incremental step with different distillation criteria.

In the second incremental step, the model $M1(L^{'}_D, L^{''}_D)$ was retrained with dataset (D^2_{ts}) which shows the substrate before curing the glue; moreover, it has different lighting

conditions compared to (D^1_{ts}). According to table 4.5, it is shown that using both distillation loss functions $L^{'}_{D}$ and $L^{''}_{D}$ produced better performance than the model in which the encoder is frozen while utilizing only $L^{'}_{D}$ distillation loss function. Similarly in the last incremental step, the best model from the previous step is retrained with dataset (D^2_{ts}) which resolution is less than all the previous datasets. In table 4.6, scores of the trained models are shown where the model with distillation loss functions $L^{'}_{D}, L^{''}_{D}$ struggles in maintaining the knowledge of the previous datasets (D^1_{ts}) and (D^2_{ts}). On the other hand, freezing the encoder and applying only the distillation loss function $L^{'}_{D}$ aided the model to overcome catastrophic forgetting; even though its performance on the new dataset (D^3_{ts}) is not the best. It is worth mentioning that using $L^{'''}_{D}$ was mostly avoided due to high computational cost; in addition, the effect of that loss function on the performance was very slight in most of the incremental steps. Moreover, the value of the learning rate is decreased as more incremental steps are added which proved to improve the models' performance. This is because the more incremental steps the more complex is the learning curve and consequently the learning rate value needs to be less to not miss the global minima.

Following each incremental step, a model M_0 was trained with samples from the current and previous datasets to compare between incremental learning and normal batch learning. The scores of the batch learning models M_0 are shown in tables 4.4, 4.5, and 4.6 where in the last incremental step for instance, the M_0 model was trained with samples from all four datasets ($D^{\{0,3\}}_{tr}$). The incremental models achieved slightly lower evaluation scores than the models M_0, which was expected. However, from the qualitative results in figures 4.3 and 4.4 the deviations of incremental models predictions and M_0 models predictions from the GT are minor. Moreover, the batch learning based models have higher conputational cost and training time since they use more data samples for training. Furthermore, the fact that the incremental models achieve such performances without acessing the previous datasets gives them more credibility, especially in the case where accessing old datasets is not possible. The predicted gap masks generated by the incremental models can be enhanced by applying morphological operations as presented in Appendix A.

According to the TINKER KPIs defined before, the base segmentation model meets the target KPIs achieving a MIoU of 96.02%. As for the incremental models (M_1,M_2 and M_3), their scores were very close to the targeted KPIs; they achieved a MIoU of 87.92%, 90.32% and 88.36%, respectively over previous datasets.

	base dataset(D^0_{ts})			1st new dataset(D^1_{ts})			2nd new dataset(D^2_{ts})		
M_2	precision	recall	MIoU	precision	recall	MIoU	precision	recall	MIoU
$L^{'}_{D}, E_F$	84.90%	92.52%	79.79%	92.20%	96.07%	88.87%	94.84%	91.40%	87.11%
$L^{'}_{D}, L^{''}_{D}$	**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%

Table 4.5.: Performance measures of the segmentation model on previous datasets after the second incremental step with different distillation criteria.

M_3	base dataset(D_{ts}^0)			1st new dataset(D_{ts}^1)		
	precision	recall	MIoU	precision	recall	MIoU
$L_D^{'}, E_F$	90.01%	**96.45%**	87.20%	**92.47%**	**94.66%**	**87.89%**
$L_D^{'}, L_D^{''}$	**91.97%**	96.19%	**88.86%**	86.66%	84.66%	74.91%
$M_0(D_{tr}^{\{0,3\}})$	96.02%	92.50%	89.11%	97.19%	96.60%	93.68%
	2nd new dataset(D_{ts}^2)			3nd new dataset(D_{ts}^3)		
$L_D^{'}, E_F$	**94.77%**	**95.49%**	**90.72%**	92.87%	**87.26%**	**81.99%**
$L_D^{'}, L_D^{''}$	86.21%	87.66%	76.84%	**94.46%**	83.94%	80.20%
$M_0(D_{tr}^{\{0,3\}})$	98.63%	94.33%	93.06%	96.34%	92.87%	89.84%

Table 4.6.: Performance measures of the segmentation model on previous datasets after the final incremental step with different distillation criteria.

4.3. Depth Estimation Model

4.3.1. Base Model Selection and Training

Now that the gap mask is successfully extracted, it is used in the construction of the gap image that is fed to the MDE model, since the gap location is the only part where there is depth variations while the rest of the image is almost flat. As discussed in the methodology section, there were several attempts and several architectures implemented. The first implemented model was the CNN model shown in 3.5. Training the model took very long due to the huge number of input data samples which was inevitable so that the model reaches an acceptable performance. The model achieved the worst performance scores as shown in table 4.7. The second implemented approach for solving the MDE task was based on a U-net model with residual blocks applied to the encoder part of the model as in 3.6. Using the residual blocks allowed the model to train faster and achieve better performance; however, the convergence was not optimum and accuracy measure $\delta < 1.25$ was still low as shown in 4.7. In the third attempt, the U-net model used in the segmentation task was used together with its pre-trained weights. The use of pre-trained weights from the segmentation task allowed the model to converge faster building upon the feature extraction knowledge it has learnt. As a result, the model produced better prediction scores on unseen data samples than the previous models; yet the $\delta < 1.25$ was not good as well. Both U-net models were trained with the berHu training loss function presented in 3.6. Figure 4.5 shows a qualitative measure of the performance of the developed MDE models.

In the last attempt, the same U-net model was used but with different training loss function shown in 3.7. Using this loss function led to great improvement in the model performance which is reflected by the higher accuracy measures and lower loss measures. This is because the combined loss function takes advantage of its loss components; for example, the SSIM index which takes in consideration many factors such as image textures and edge information. The final MDE model which showed the best performance was

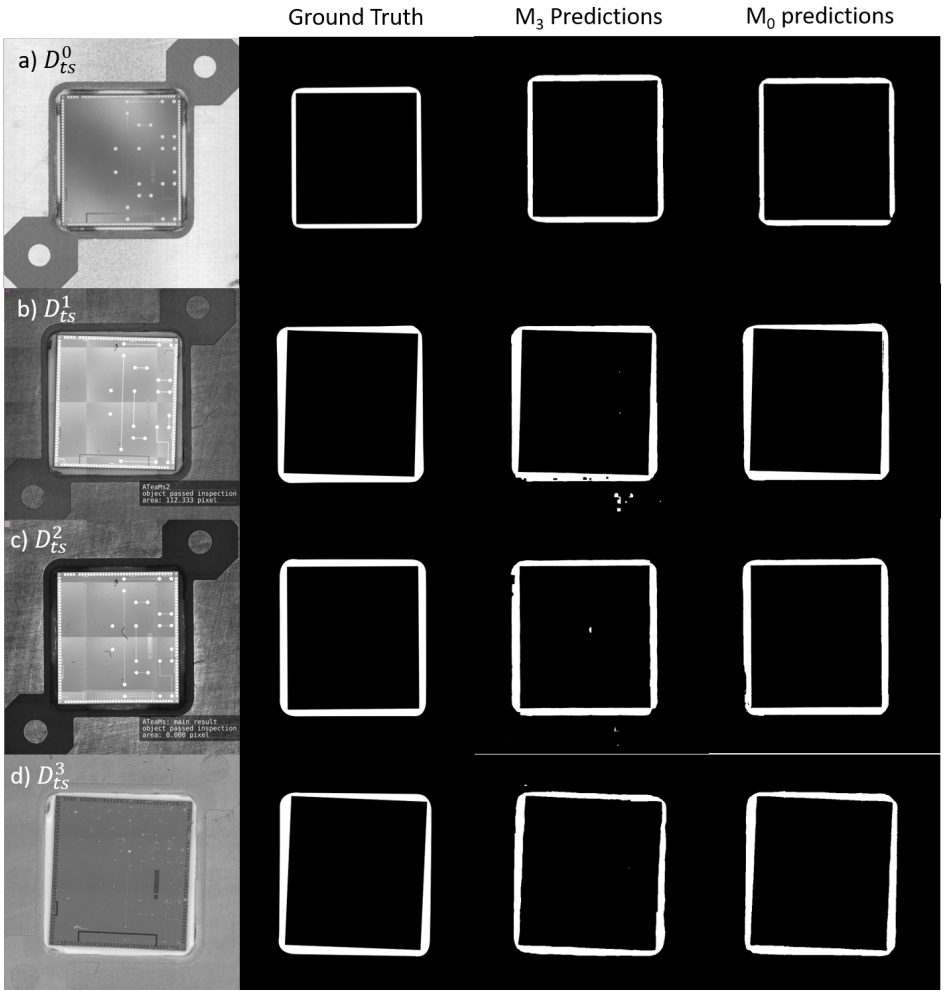


Figure 4.3.: Qualitative results of the final incremental segmentation model M_3 on testing samples from all the four datasets D_{ts}^0 (a), D_{ts}^1 (b), D_{ts}^2 (c), and D_{ts}^3 (d).

trained with a much larger dataset that was provided later in the Tinker project. The new depth measurements were much more accurate than the ones before; however, they had less resolution. To fix this issue, the depth maps were up-scaled to double their dimensions so as to match the dimensions of the old depth dataset. The model was trained with up-scaled depth dataset, and then for model testing, they were reverted to the original size. The last row of table 4.7 shows the scores of the model on unseen partition of the new dataset. Figure 4.6 shows samples from the best model predictions on the $D_t^1 s$ dataset. Note that the figure shows the full monocular images; however, for training the models, the gap contour was extracted using the gap mask as shown in 3.4. 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

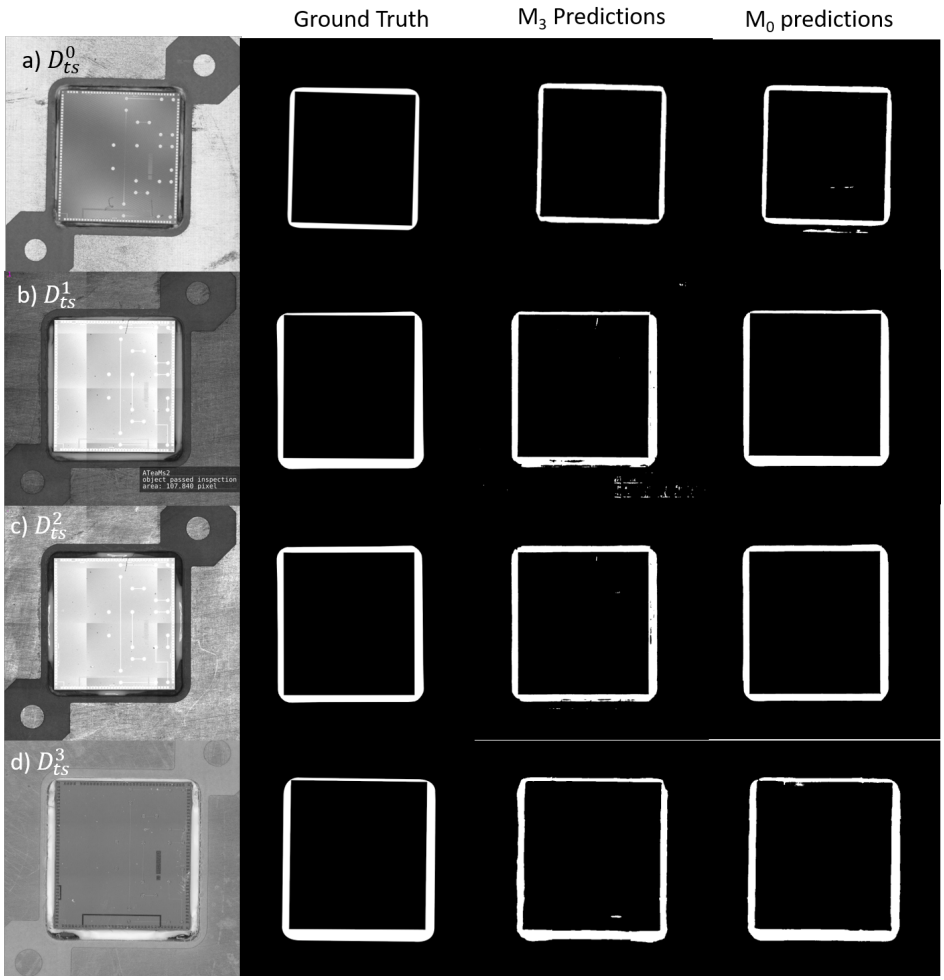


Figure 4.4.: Qualitative results of the final incremental segmentation model M_3 on testing samples from all the four datasets D_{ts}^0 (a), D_{ts}^1 (b), D_{ts}^2 (c), and D_{ts}^3 (d).

was applied. However, the model performs less in the case of high rate of change of depth values; for instance, the gap corners in the first row image of 4.6. If the rate of change is more gradual and occurs over a longer segment of the gap, the model performs much better; for instance, the gap bottom corners in the second row image of 4.6. Furthermore, Figure 4.7 shows samples of the model predictions on the new large dataset. With the larger dataset, the model was able to capture more information due to the big amount of training sample and the diversity of depth distributions along the gap. In addition, it can be noticed that the rate of change of depth values along the gap is not high, which is the most probable in the cases of fluid distribution inside a cavity.

Method	Lower is better			Higher is better		
	Abs Rel	RMSE	log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
CNN(MSE)	0.555	0.069	0.165	41.6%	69.6%	83.2%
ResUnet(berHu)	0.338	0.061	0.129	47.0%	77.1%	91.6%
Unet(berHu)	0.225	0.044	0.111	58.7%	84.1%	93.3%
Unet(compinedLoss)	**0.192**	**0.040**	**0.082**	**70.0%**	**92.0%**	**97.5%**
Unet(compinedLoss) new large dataset	0.183	0.016	0.084	85.6%	92.3%	95.5%

Table 4.7.: Performance comparison between the different MDE models with different loss functions.

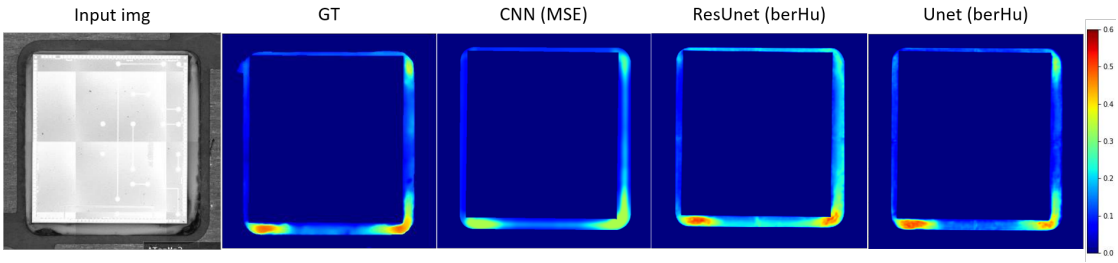


Figure 4.5.: Comparison between the different MDE models developed throughout this work before reaching the best model. Qualitative results show the progressive improvement in prediction performance.

4.3.2. IL Implementation

Following the IL procedure discussed in section 3.4.3, the developed MDE model went through one incremental training step. The 2D images of both datasets, D^1 and D^3, are describing the same substrates but are different in terms of the recording setup, and consequently they have different resolution and different features. Therefore, the GT depth measurements in both datasets are actually the same. The dataset used for training the base MDE model was D^1_{tr}; Afterwards, D^3_{tr} dataset, was used for the incremental training of the base MDE model. The training was concluded in 45 epochs with a learning rate of $5 \times 10^{-}6$ which decays linearly with a factor of 0.5 every 15 epochs. At the first attempt, the encoder was frozen and the first distillation loss function $L^{'}_{MDE_D}$ represented by equation (3.7) was used. In the second attempt, the loss functions, $L^{''}_{MDE_D}$ represented by equation (3.6) and $L^{'}_{MDE_D}$, were combined while retraining the whole model. In table 4.8, the performance after each attempt is reported where the scores are almost the same. However, in the first attempt, the training was much faster since the encoder parameters were not trainable. That is why the model utilizing $L^{'}_{D}$ and E_F was preferred. Figure 4.8 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

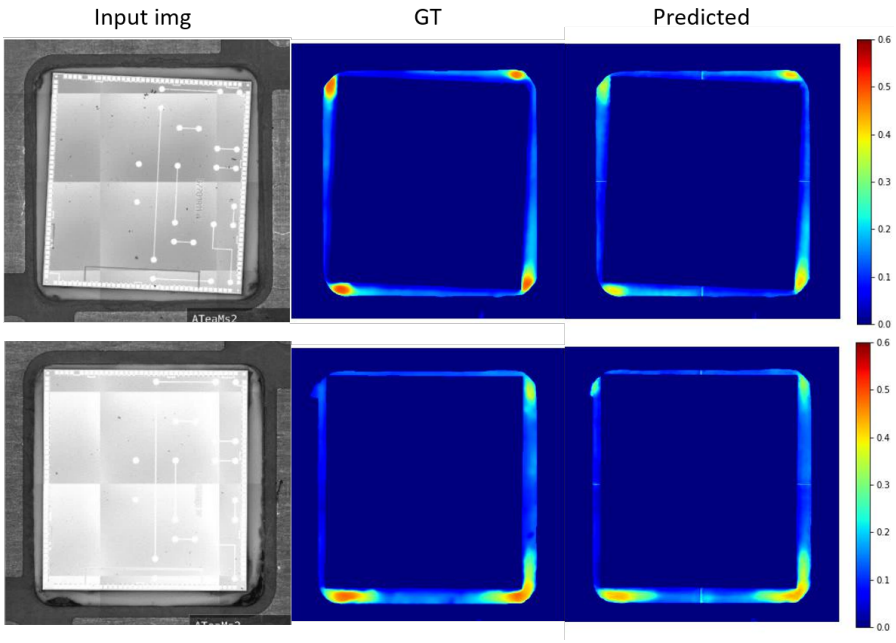


Figure 4.6.: Qualitative results of the best MDE model on samples from the $D_t^1 s$ dataset.

measure $\delta < 1.25$. From table 4.7, it is observed that the value of the accuracy measure $\delta < 1.25$ of the first four models 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.

	Lower is better			Higher is better		
Method	AbsRel	RMSE	log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
$L_D^{'}, E_F\ (D_t^1 s)$	0.222	0.045	0.097	61.4%	88.9%	96.5%
$L_D^{'}, E_F\ (D_t^3 s)$	0.234	0.056	0.102	58.7%	86.2%	96.1%
$L_D^{'}, L_D^{''}\ (D_t^1 s)$	0.212	0.043	0.094	63.7%	89.6%	96.6%
$L_D^{'}, L_D^{''}\ (D_t^3 s)$	0.236	0.055	0.103	57.6%	86.2%	96.0%

Table 4.8.: Performance measures of the MDE model on the two datasets after applying an incremental step with two different distillation criteria.

The performance of the base MDE model trained on the larger dataset shows so much improvement and the AbsRel score comes very close to the TINKER KPIs for depth estimation. For the incremental MDE model, the mean AbsRel score is a bit high relative to the KPIs; and thus there is still room for improvement, given the availability of a bit larger datasets.

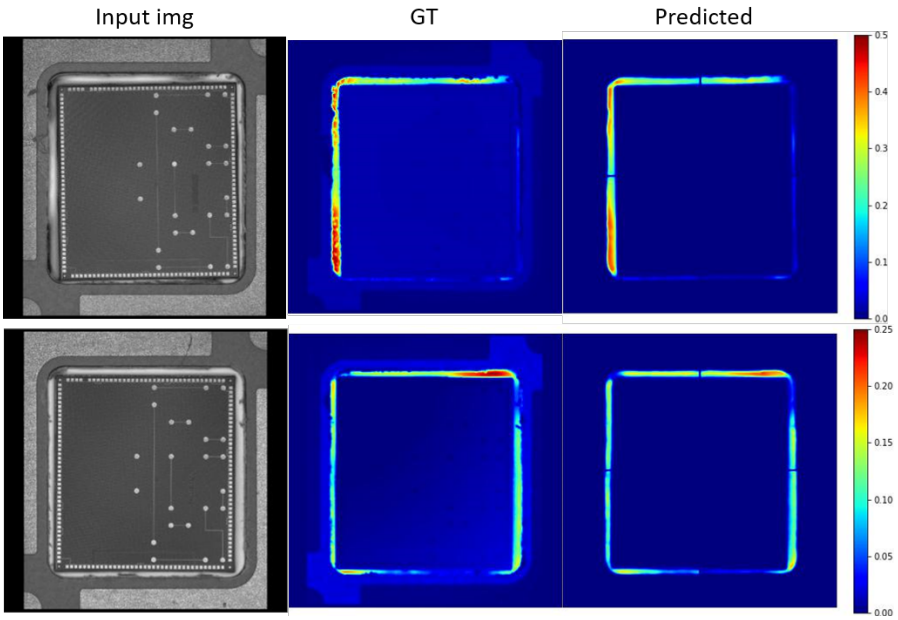


Figure 4.7.: Qualitative results of the best MDE model on samples from the new large dataset.

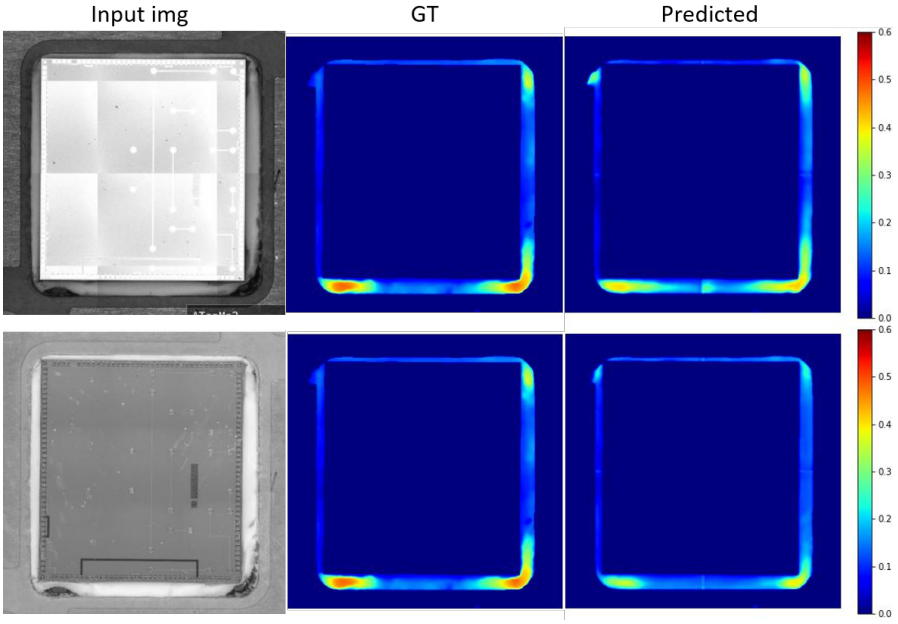


Figure 4.8.: Qualitative results of the incremental MDE model on samples from $D_t^1 s$ and $D_t^3 s$ datasets.

5. Conclusions

To sum up, the work in this thesis was done as part of TINKER, an EU funded H2020 project. One of the goals in TINKER is to push forward the inline defect repair feedback loops depending on robust error compensation Machine Learning (ML) based algorithms. To be able to compute corrective actions, these ML algorithms are trained to model the relation between the input process parameters and the corresponding process measurements. The thesis targets one fabrication process which is the placement of a microchip inside a PCB cavity filled with non-conductive adhesive. The developed system in this thesis uses the recorded images of the microchips in cavities, i.e. substrates as well as the substrates' depth maps for training, and provides an estimation of the missing glue in the gap between the chip and the boundaries of the cavity. Maintaining uniform height of the glue inside this gap is targeted in order to have a planar surface at the end of the current process to be able to initiate further processes. For estimating the depth inside the gap, two models were developed; first, a segmentation model to detect the gap shape and location; second, a MDE regression model. Using the estimated depth map, the missing glue is printed in the gap by an inkjet printer.

Moreover, The thesis utilizes an incremental learning scheme to overcome the need for generating a huge amount of data whenever changes in the fabrication process or developments of the inspection system occur. Since there are many partner companies participating in TINKER, where each inspection system has different recording conditions, the developed algorithms should be able to fit different datasets. Moreover, since the pilot line of the fabrication process is delocalized, the process data had to be provided in batches based on materials availability. With those points in mind, the incremental learning scheme was implemented to give a model the ability to incrementally learn new information without catastrophically forgetting those previously learnt. Using incremental learning, an initial DL model is trained with a small amount of data, and then the rest of the datasets are provided to the model incrementally. The main key features of applying incremental learning are the following: learning with forgetting, never accessing the older datasets, and avoiding model expansion. The incremental learning scheme was applied to both gap segmentation and MDE models and achieved very promising results showing the ability to adapt to process changes which are reflected in changes of images' features such as brightness and contrast levels.

The performance of the incrementally trained MDE model could have been better if the datasets were a bit larger. That is why it can be said that the small amount of 3D depth data is one of the limitations of this work. One of the observed constraints of applying incremental learning on the MDE task was that the range of depth values in the depth maps should be similar across different datasets. If that is not the case, modelling the relation between 2D images, ranging always from $\{0 - 255\}$, and corresponding

depth maps with fluctuating ranges becomes more difficult. Therefore, applying a second incremental training step using the latest large dataset was not successful. For future improvements, the same methodology could be tested with larger datasets, especially for the MDE model. A more robust depth metrology providing more than just depth maps; for instance scale information, would be much helpful as well. Further incremental learning criteria could be also added to improve the ability of the models to retain the old knowledge. Overcoming such limitations and applying some of these future improvements was not simple in the time of the thesis due to time and inspection metrology limitations.

Bibliography

[1] Perry Gibson and José Cano. Productive reproducible workflows for dnns: A case study for industrial defect detection, 2022.

[2] Han Jian, Lu Chenghui, Cao Zhimin, and Mu Haiwei. Integration of deep neural networks and ensemble learning machines for missing well logs estimation. *Flow Measurement and Instrumentation*, 73:101748, 2020. ISSN 0955-5986. doi: https://doi.org/10.1016/j.flowmeasinst.2020.101748. URL `https://www.sciencedirect.com/science/article/pii/S0955598620300960`.

[3] Qiyang Li, Jingxing Qian, Zining Zhu, Xuchan Bao, Mohamed Helwa, and Angela Schoellig. Deep neural networks for improved, impromptu trajectory tracking of quadrotors. 10 2016.

[4] Yong Luo, Liancheng Yin, Wenchao Bai, and Keming Mao. An appraisal of incremental learning methods. *Entropy*, 22(11):1190, 2020. doi: 10.3390/e22111190.

[5] Hasan Tercan, Philipp Deibert, and Tobias Meisen. Continual learning of neural networks for quality prediction in production using memory aware synapses and weight transfer. *Journal of Intelligent Manufacturing*, 33(1):283–292, 2021. doi: 10.1007/s10845-021-01793-0.

[6] Flaig Minnette and Zambal Sebastian. Deep learning for zero-defect inkjet-printing of electronics. In *2021 IEEE International Workshop on Metrology for Industry 4.0 IoT (MetroInd4.0IoT)*, pages 458–463, 2021. doi: 10.1109/MetroInd4.0IoT51437.2021.9488493.

[7] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020. URL `https://arxiv.org/abs/2001.05566`.

[8] Yujian Mo, Yan Wu, Xinneng Yang, Feilin Liu, and Yujun Liao. Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing*, 493:626–646, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.01.005. URL `https://www.sciencedirect.com/science/article/pii/S0925231222000054`.

[9] Irem Ulku and Erdem Akagündüz. A survey on deep learning-based architectures for semantic segmentation on 2d images. *Applied Artificial Intelligence*, 36(1):2032924, 2022. doi: 10.1080/08839514.2022.2032924. URL `https://doi.org/10.1080/08839514.2022.2032924`.

[10] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2014. URL https://arxiv.org/abs/1411.4038.

[11] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation, 2015. URL https://arxiv.org/abs/1505.04366.

[12] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2015. URL https://arxiv.org/abs/1511.00561.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

[14] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2015. URL https://arxiv.org/abs/1511.07122.

[15] Haneen Alokasi and Muhammad Bilal Ahmad. The accuracy performance of semantic segmentation network with different backbones. In *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*, pages 49–54, 2022. doi: 10.1109/CDMA54072.2022.00013.

[16] Guijuan Zhang, Yang Liu, and Xiaoning Jin. A survey of autoencoder-based recommender systems. *Frontiers of Computer Science*, 14(2):430–450, 2019. doi: 10.1007/s11704-018-8052-6.

[17] Ming Wu, Chuang Zhang, Jiaming Liu, Lichen Zhou, and Xiaoqi Li. Towards accurate high resolution satellite image semantic segmentation. *IEEE Access*, 7: 55609–55619, 2019. doi: 10.1109/access.2019.2913442.

[18] Rongyu Zhang, Lixuan Du, Qi Xiao, and Jiaming Liu. Comparison of backbones for semantic segmentation network. *Journal of Physics: Conference Series*, 1544(1): 012196, 2020. doi: 10.1088/1742-6596/1544/1/012196.

[19] ChaoQiang Zhao, QiYu Sun, ChongZhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, 63(9):1612–1627, 2020. doi: 10.1007/s11431-020-1582-8.

[20] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.12.089. URL https://www.sciencedirect.com/science/article/pii/S0925231220320014.

[21] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *CoRR*, abs/1406.2283, 2014. URL http://arxiv.org/abs/1406.2283.

[22] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. *CoRR*, abs/1704.07813, 2017. URL `http://arxiv.org/abs/1704.07813`.

[23] Xinchen Ye, Xiang Ji, Baoli Sun, Shenglun Chen, Zhihui Wang, and Haojie Li. Drm-slam: Towards dense reconstruction of monocular slam with scene depth fusion. *Neurocomputing*, 396:76–91, 2020. doi: 10.1016/j.neucom.2020.02.044.

[24] Ravi Garg, Vijay Kumar B.G., Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. *Computer Vision – ECCV 2016*, page 740–756, 2016. doi: 10.1007/978-3-319-46484-8_45.

[25] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[26] Nishant Kashyap and Anjana Mishra. Chapter 1 - a discourse on metaheuristics techniques for solving clustering and semisupervised learning models. In *Cognitive Big Data Intelligence with a Metaheuristic Approach*, Cognitive Data Science in Sustainable Computing, pages 1–19. Academic Press, 2022. ISBN 978-0-323-85117-6. doi: https://doi.org/10.1016/B978-0-323-85117-6.00012-1. URL `https://www.sciencedirect.com/science/article/pii/B9780323851176000121`.

[27] Bo Li, Chunhua Shen, Yuchao Dai, Anton van den Hengel, and Mingyi 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)*, pages 1119–1127, 2015. doi: 10.1109/CVPR.2015.7298715.

[28] Azeez Oluwafemi, Yang Zou, and B. V. K. Vijaya Kumar. Deep classification network for monocular depth estimation, 2019.

[29] Yuanzhouhan Cao, Zifeng Wu, and Chunhua Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks, 2017.

[30] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials, 2012.

[31] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation, 2017.

[32] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning, 2019.

[33] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks, 2016. URL `https://arxiv.org/abs/1606.00373`.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[35] Zhao Chen, Vijay Badrinarayanan, Gilad Drozdov, and Andrew Rabinovich. Estimating depth from RGB and sparse sensing. In *Computer Vision – ECCV 2018*, pages 176–192. Springer International Publishing, 2018. doi: 10.1007/978-3-030-01225-0_11. URL https://doi.org/10.1007%2F978-3-030-01225-0_11.

[36] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation, 2019. URL https://arxiv.org/abs/1907.10326.

[37] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Zequn Jie, Xiang Li, and Jian Yang. *Joint Task-Recursive Learning for Semantic Segmentation and Depth Estimation: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X*, pages 238–255. 09 2018. ISBN 978-3-030-01248-9. doi: 10.1007/978-3-030-01249-6_15.

[38] Lijun Wang, Jianming Zhang, Oliver Wang, Zhe Lin, and Huchuan Lu. Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[39] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification, 2020. URL https://arxiv.org/abs/2010.15277.

[40] Qing Yang, Yudi Gu, and Dongsheng Wu. Survey of incremental learning. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 399–404, 2019. doi: 10.1109/CCDC.2019.8832774.

[41] Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. doi: 10.1038/s42256-022-00568-3.

[42] Mariela Andrade, Eduardo Gasca, and Eréndira Rendón Lara. Implementation of incremental learning in artificial neural networks. In *GCAI*, 2017.

[43] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continuum learning. *CoRR*, abs/1706.08840, 2017. URL http://arxiv.org/abs/1706.08840.

[44] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation. *CoRR*, abs/1907.13372, 2019. URL http://arxiv.org/abs/1907.13372.

[45] Umberto Michieli and Pietro Zanuttigh. Knowledge distillation for incremental learning in semantic segmentation. *CoRR*, abs/1911.03462, 2019. URL `http://arxiv.org/abs/1911.03462`.

[46] Vinay Kumar Verma, Kevin J. Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient feature transformations for discriminative and generative continual learning. *CoRR*, abs/2103.13558, 2021. URL `https://arxiv.org/abs/2103.13558`.

[47] M. Jehanzeb Mirza, Marc Masana, Horst Possegger, and Horst Bischof. An efficient domain-incremental learning approach to drive in all weather conditions, 2022. URL `https://arxiv.org/abs/2204.08817`.

[48] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *CoRR*, abs/1611.07725, 2016. URL `http://arxiv.org/abs/1611.07725`.

[49] Justin Leo and Jugal Kalita. Incremental deep neural network learning using classification confidence thresholding. *CoRR*, abs/2106.11437, 2021. URL `https://arxiv.org/abs/2106.11437`.

[50] Da-Wei Zhou, Yang Yang, and De-Chuan Zhan. Learning to classify with incremental new class. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6): 2429–2443, 2022. doi: 10.1109/TNNLS.2021.3104882.

[51] Aatila Mustapha, Lachgar Mohamed, and Kartit Ali. Comparative study of optimization techniques in deep learning: Application in the ophthalmology field. *Journal of Physics: Conference Series*, 1743(1):012002, jan 2021. doi: 10.1088/1742-6596/1743/1/012002. URL `https://dx.doi.org/10.1088/1742-6596/1743/1/012002`.

[52] David Duque, Santiago Velasco-Forero, Jean-Emmanuel Deschaud, François Goulette, Andrés Serna, Etienne Decencière, and B. Marcotegui. On power jaccard losses for semantic segmentation. pages 561–568, 01 2021. doi: 10.5220/0010304005610568.

[53] Ashish Patel. Satellight image segmentation with unet and its variants. URL `https://github.com/ashishpatel26/satellite-Image-Semantic-Segmentation-Unet-Tensorflow-keras`.

[54] Saddam Abdulwahab, Hatem A. Rashwan, Najwa Sharaf, Saif Khalid, and Domenec Puig. Deep monocular depth estimation based on content and contextual features. *Sensors*, 23(6), 2023. ISSN 1424-8220. doi: 10.3390/s23062919. URL `https://www.mdpi.com/1424-8220/23/6/2919`.

[55] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2020.

Acronyms

CNN Convolutional Neural Network. xi, 5, 6, 8, 9, 19, 21

Conv. Convolutional. 5, 6, 20

DL Deep Learning. v, 1, 16, 41

DNNs Deep Neural Networks. 1

GANs Generative Adversarial Networks. 8

GT Ground Truth. xi, 8, 9, 11, 15–17, 21, 23, 25, 33, 37

IL Incremental Learning. v, viii, xi, 1, 2, 17, 24, 25, 30, 37

KPIs Key Performance Indicators. viii, ix, 3, 13, 29, 33, 38

LIDAR Laser Imaging, Detection, and Ranging. 1, 8, 9

MDE Monocular Depth Estimation. vii, ix, xi, 8–10, 16, 19, 21–24, 27, 29, 34, 35, 37–39, 41, 42

ML Machine Learning. 41

PCB Printed Circuit Board. v, 1, 2, 15

RADAR Radio Detection and Ranging. 1, 15

RGB Red Green Blue. 8, 9

VAEs Variational AutoEncoders. 8

A. Appendix

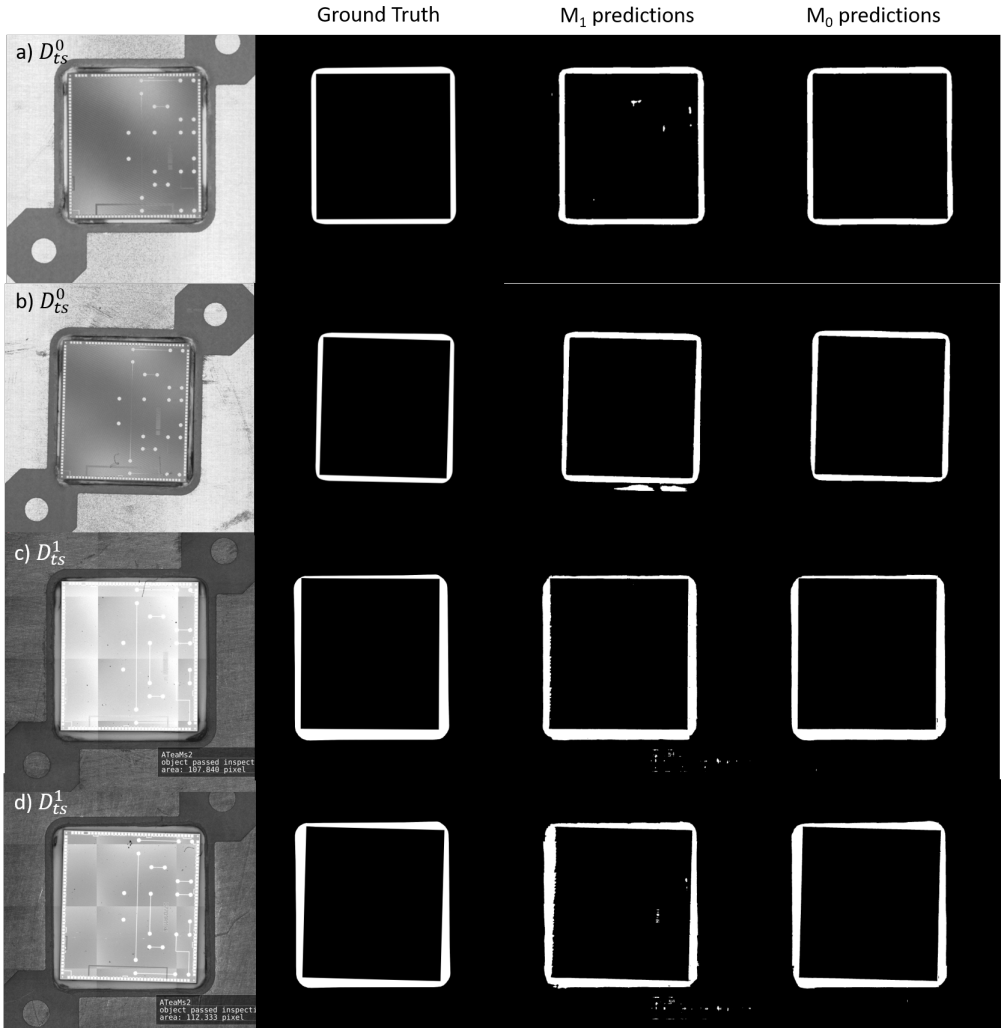


Figure A.1.: Qualitative results of the segmentation model predictions after the first incremental step M_1 against predictions of the batch learning based model M_0. Both were tested on samples from datasets D_{ts}^0 (a,b) and D_{ts}^1 (c,d).

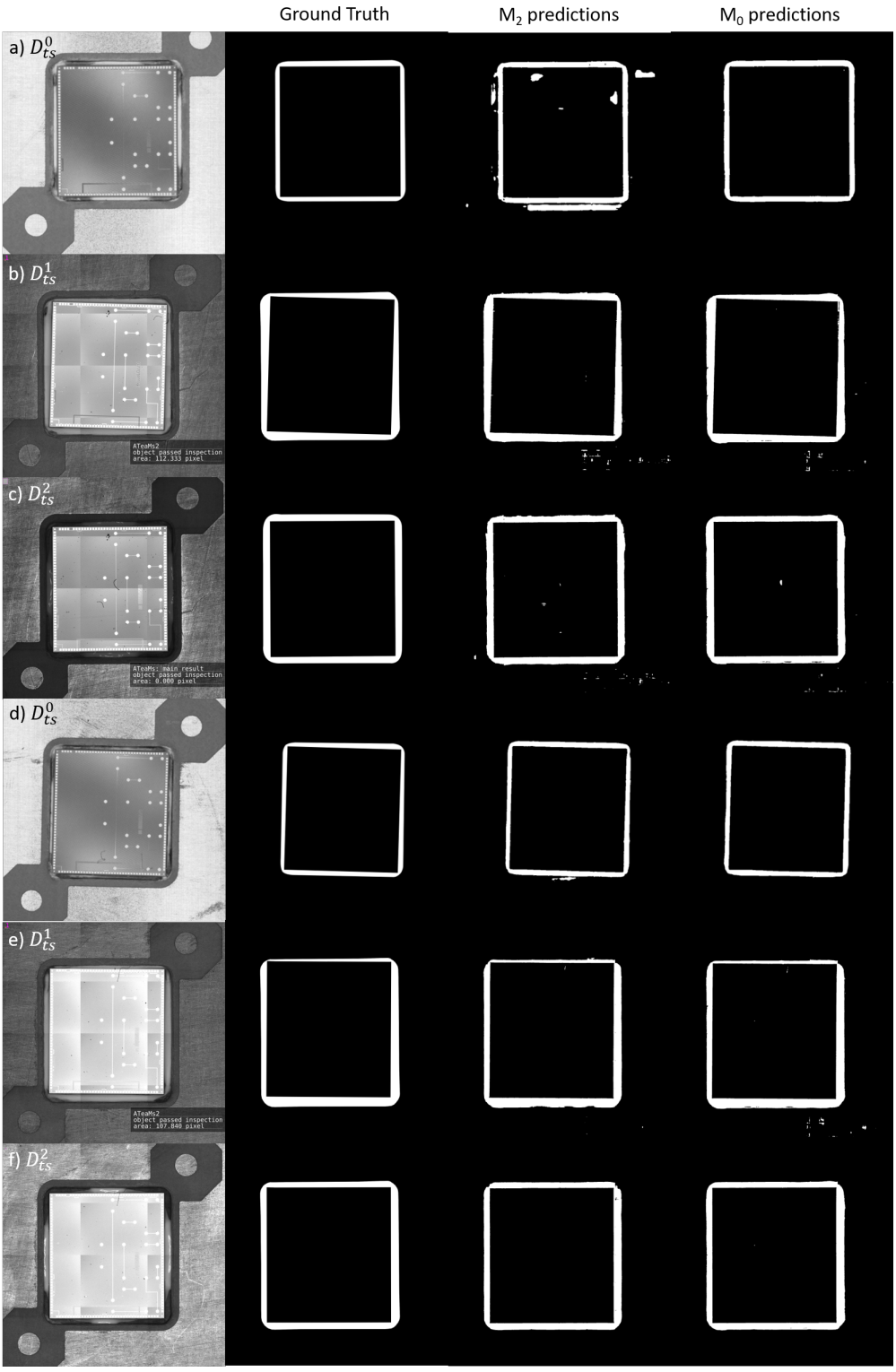


Figure A.2.: Qualitative results of the segmentation model predictions after the second incremental step M_2 against predictions of the batch learning based model M_0. Both were tested on samples from datasets D_{ts}^0 (a,d), D_{ts}^1 (b,e) and D_{ts}^2 (c,f).

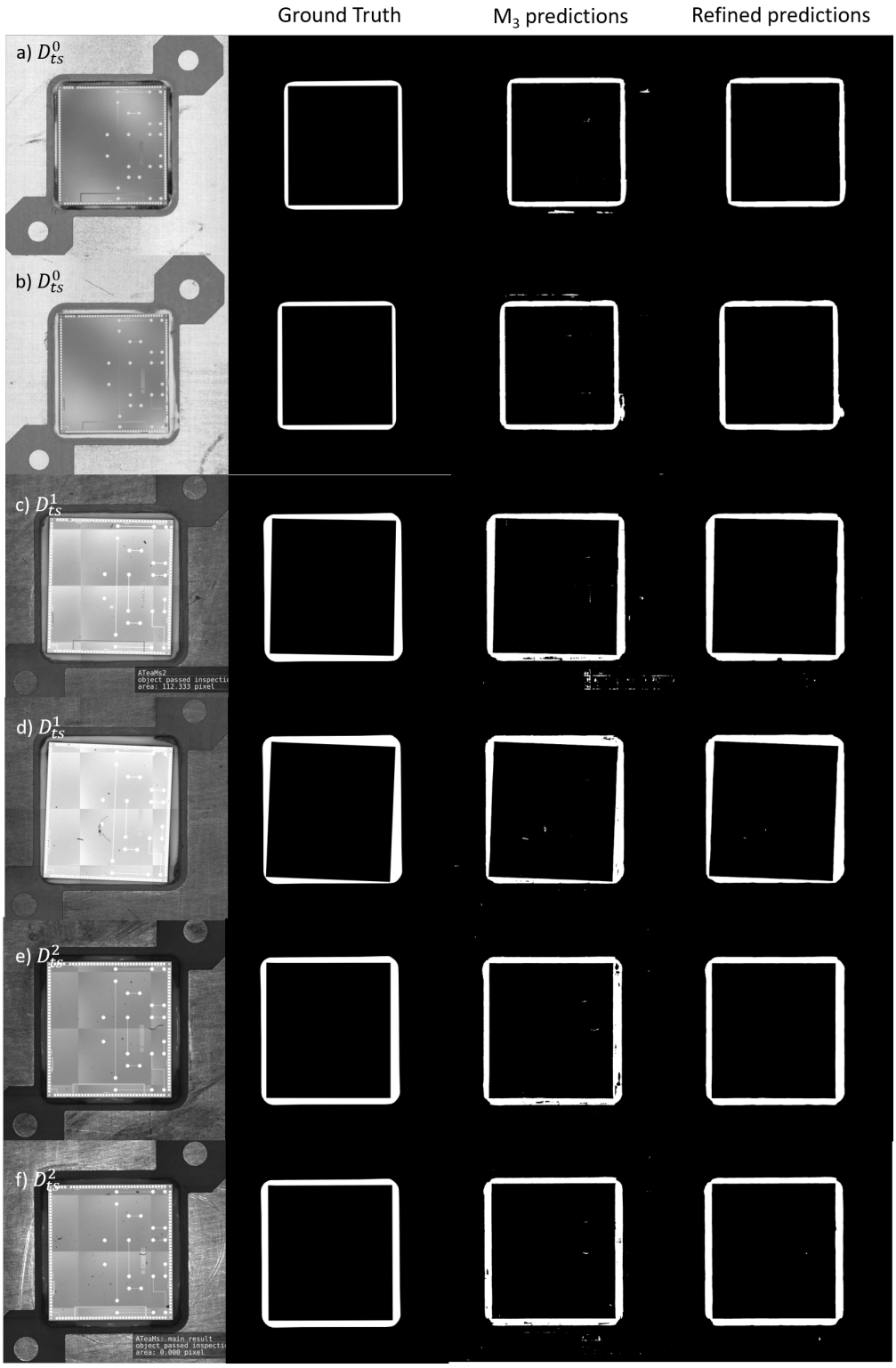


Figure A.3.: Qualitative results of applying open and close morphological operations on predicted masks.