IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 December 2023, accepted 24 December 2023, date of publication 26 January 2024, date of current version 6 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3359169

RESEARCH ARTICLE

Rapid Flow Behavior Modeling of Thermal Interface Materials Using Deep Neural Networks

SIMON BAEUERLE[1,2], MARIUS GEBHARDT[2], JONAS BARTH[2], RALF MIKUT*[1], AND ANDREAS STEIMERT*[3]
[1]Institute for Automation and Applied Informatics (IAI), Karlsruhe Institute of Technology (KIT), 76344 Eggenstein-Leopoldshafen, Germany
[2]Robert Bosch GmbH, 72762 Reutlingen, Germany
[3]Bosch Center for Artificial Intelligence (BCAI), Robert Bosch GmbH, 71272 Renningen, Germany

Corresponding author: Ralf Mikut (ralf.mikut@kit.edu)

This work was supported by the Karlsruhe Institute of Technology (KIT)-Publication Fund of KIT. The work of Ralf Mikut was supported by the Helmholtz Associations Initiative and Networking Fund through the Helmholtz Artificial Intelligence Cooperation Unit (Helmholtz AI).

*Ralf Mikut and Andreas Steimer contributed equally to this work.

ABSTRACT Thermal Interface Materials (TIMs) are widely used in electronic packaging. Increasing power density and limited assembly space pose high demands on thermal management. Large cooling surfaces need to be covered efficiently. When joining the heatsink, previously dispensed TIM spreads over the cooling surface. Recommendations on the dispense pattern exist only for simple surface geometries such as rectangles. For more complex geometries, Computational Fluid Dynamics (CFD) simulations are used in combination with manual experiments. While CFD simulations offer a high accuracy, they involve simulation experts and are rather expensive to set up. We propose a lightweight heuristic to model the spreading behavior of TIM. We further speed up the calculation by training an Artificial Neural Network (ANN) on data from this model. This offers rapid computation times and further supplies gradient information. This ANN can not only be used to aid manual pattern design of TIM, but also enables an automated pattern optimization. We compare this approach against the state-of-the-art and use real product samples for validation.

INDEX TERMS Deep learning, electronics packaging, flow behavior, thermal interface materials, thermal management.

I. INTRODUCTION

Automotive industry is putting an increasing effort into electric and autonomous vehicles. Demand for efficient and reliable electronic components is rising accordingly. This is valid for small Electronic Control Units (ECUs) used to control e.g. the engine, but also for power electronics components such as inverters or chargers. Time-to-market tends to be shorter and is a crucial factor for global competitiveness. While power ratings increase, tight restrictions are imposed on assembly space as well. Thus, the thermal performance is a crucial factor in electronic packaging.

Thermal Interface Materials (TIMs) are widely utilized to lower the thermal resistance between individual components and thus enable an efficient heat transfer. Typically, they are applied onto a cooling surface with a dispensing machine. An example of dispensed TIM is shown on the left-hand side of Fig. 1. During the joining process of the heatsink, TIM is compressed and spreads over the surface. The state after compression is shown on the right-hand side of Fig. 1. Design engineers determine the pattern, along which TIM is dispensed. They need to consider multiple aspects. The most evident is a *high area coverage ratio* of the cooling surface with TIM after the joining process. Along with the high heat conductivity of TIM, this results in the aforementioned low thermal resistance. However, applying too large amounts of TIM, with excess material flowing beyond the cooling surface, leads to preventable *material cost*. This is especially relevant in high-volume series production, where even a small amount per part adds up to significant costs. Sensitive

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

electronic components or product features such as screw holes may be placed close to the cooling area. They are regarded as *taboo zones* during the design process and may not be covered by material.

A low coverage may be caused by simply applying too little TIM. Another cause for low coverage are air entrapments, which develop during the joining process. Air within closed contours such as circular shapes cannot escape while TIM is being compressed. The formation of such *voids* depends on the individual pattern shape and may not be easily recognized in all cases. Furthermore, the design of a dispense pattern is directly linked to the *cycle time* of the respective dispense process.

Apart from the pattern, the design process itself needs to be optimized as well. The design process is relevant for the time-to-market and thus needs to be short. The cost of human experts is also a relevant factor. Recommendations regarding optimal patterns can help engineers during their work. Specific guidelines exist for simple cooling area geometries such as rectangles [1]. However especially for larger and more complex surfaces, the dispense pattern needs to be adjusted for each individual product.

To evaluate a given pattern, the respective compressed state after joining needs to be known. It can be acquired by simulating the flow behavior of TIM. Computational Fluid Dynamics (CFD) simulations, carried out by highly specialized experts, are widely used. Modeling and evaluation take time, but yield very accurate results. Besides simulations, mechanical experiments are carried out with real product samples. In an iterative fashion of trial-and-error, the dispense patterns are optimized by development engineers. After several trials, a dispense pattern is defined. However, mechanical tolerances are prevalent in real products. When joining parts together, tolerances from multiple parts add up. The high accuracy of CFD simulations, which is achieved with high efforts, needs to be weighed against the variations of real products. A light-weight model with a lower accuracy but faster setup and computation times can fulfill the demands of dispense pattern design better.

CFD simulations of this kind are computationally expensive. Artificial Neural Networks (ANNs) can be trained on data from simulation models. They can typically be executed much faster. Since a rather high number of training samples is needed, an automated simulation setup is typically necessary.

Furthermore, ANNs or other Machine Learning models are well suited to build Digital Twins, since they are generally fit to be fine-tuned by using real-world data. They can support many design and production processes, e.g. for pose estimation from image data [2] or for quality monitoring in resistance welding [3]. Digital Twins are considered to be a *significant enabler for Industry 4.0 initiatives* [4].

In this work, we present two flow behavior models for TIM. They can be used to support development engineers both during manual dispense pattern design and by enabling an automated dispense pattern optimization. The first flow behavior model is a light-weight heuristic. The second one is an ANN. The heuristic can be used to analyze a large range of different dispense patterns automatically. Thus, training data can be generated for the ANN, which offers an even higher computational speed.

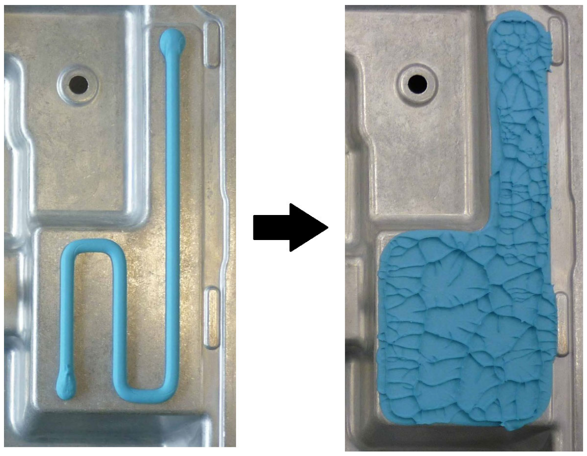


FIGURE 1. **Material flow of Thermal Interface Material (TIM) during joining the heatsink of an Electronic Control Unit (ECU). Left: state before joining, right: state after joining.**

The remainder of this paper is organized as follows. Section II provides an overview over the state-of-the-art methods, which are relevant for the dispense pattern design process. In Section III, we give a detailed insight into our light-weight heuristic. It further includes the extension with an ANN and specific details on spatial resolution and the training setup. The experiments, which we carry out to validate both of our models, are described in Section IV. Results in Section V include a study of the achieved computation speed and the accuracy both on samples from the laboratory and on a real product. Advantages and limitations of the heuristic itself as well as the combination with an ANN are discussed in Section VI.

II. RELATED WORK

Several works have highlighted that a high surface coverage with TIM enhances thermal performance of an electronic package. Ekpu et al. [5] set up a numeric simulation model including a chip, a heatsink and a TIM layer between both. They analyze the influence of TIM area coverage on thermal resistance. They report a lower thermal resistance with higher coverage percentages and recommend a coverage ratio of at least 75 %. They anticipate that their results will aid design engineers. Kesarkar and Sardana [6] also set up a numeric simulation model. Their model reproduces the thermal management problem as found in an ECU, with a TIM layer below a heat sink. They analyze different TIM coverage percentages in various configurations and report a better thermal performance for a higher TIM area coverage. Gowda et al. [7] state that *the negative effect of [...] voids on the thermal resistance of a TIM layer can be devastating*.

CFD simulation is a powerful tool to support design engineers during the design of dispense patterns. They have been used in the past both to model thermal performance and to model the flow of fluid materials. For example, Lee [8] analyze both the heat conductance within a thermal package and the heat transfer to ambient air and compare different techniques to enhance heat dissipation. Comminal et al. [9] use CFD simulations to model the flow of extruded material in additive manufacturing. They study the extrusion and deposition of highly viscous material with different settings of parameters such as nozzle velocity or extrusion velocity. This demonstrates the feasibility of using CFD simulations to model the flow behavior of TIM materials.

CFD simulations require a definition of the material behavior, which may be complex. Thermal paste is typically made up of two components, e.g. a silicone grease filled with aluminum oxide particles. In such a case, the viscosity may change both with shear stress and filler ratio [10]. The rheology of TIM has further been studied (see e.g. [11], [12]). CFD simulations typically aim to model both complex material behavior and geometries accurately.

Gu et al. [13] create training data from a Finite Element (FE) model. They modify the distribution of two materials within a composite material structure and solve for mechanical properties such as toughness and strength. They train both a linear model and a Convolutional Neural Network (CNN) on this data and speed up computation times by a factor of 250, while maintaining sufficient accuracy. Koeppe et al. [14] also train an ANN on data from an FE model. They calculate the mechanical stress for a lattice structure at given load conditions. The computation time for a single FE simulation is approximately 5-10 hours, whereas the ANN takes less than one second. They use 85 training examples to train an ANN, which has 16 output features.

A major limitation of all proposed approaches is the efficient generation of a training dataset that is sufficient for the design of complex ANNs. Those prerequisites are difficult to fulfill with experiments or state-of-the-art numeric simulation models. Therefore, we use our proposed heuristic flow behavior model to create a sufficiently high number of training samples.

III. SIMULATION METHODS

In this section, we give an overview of our approach as depicted in Fig. 2. First, we describe the in- and output data representation and the 2D discretization, which is used as a pre-processing step. We then take a closer look at each of our proposed flow behavior models: a heuristic and an ANN trained on data from the heuristic. Specific setup details regarding e.g. dataset generation and training procedure are outlined in Section IV.

A. DATA REPRESENTATION AND 2D DISCRETIZATION

We define the dispense pattern, which is the path along which TIM is applied, using a polygonal chain. In the simplest case, this equals a single line with five parameters: both endpoints of this line have continuous x- and y-coordinates. The fifth parameter is the feed rate of TIM along the line segment. For longer patterns, we iteratively append another point and a respective feed rate. The input parameters are shown on the left-hand side of Fig. 2. The state after dispensing is represented as a two-dimensional grid. The number on each grid cell represents the amount of TIM in each cell. The previously introduced input parameterization is transferred onto this two-dimensional representation. This process is visualized as *2D discretization* in Fig. 2. We apply *Unweighted Area Sampling* [15], which is a technique in the field of computer graphics to draw anti-aliased lines. It works as follows in our case: each segment of the pattern is assigned a width of one, i.e. all lines become rectangles. The intersection of each grid cell with each rectangle is calculated. The amount, which is specified via the feed rate for each line segment, is assigned to each grid cell proportional to this intersection. Fig. 3 contains a visual depiction of how the amount for each grid cell is calculated. This discretized state of the spatial TIM distribution after dispensing serves as input to a flow behavior model, which outputs the state after compression. This output is again a spatial distribution of TIM and is discretized in the same way. An example is shown on the right-hand side of Fig. 2. The flow behavior model can be either the heuristic or an ANN. Both take the dispensed state as input and output the compressed state.

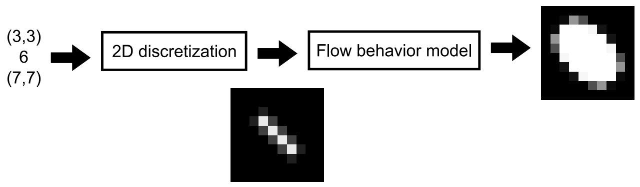


FIGURE 2. Overview of our approach for a single line of TIM. Inputs are the start point coordinate, feed rate and end point coordinate. The TIM distribution is spatially discretized before and after the compression step.

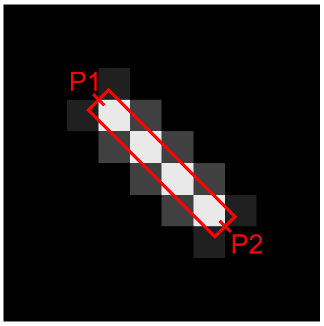


FIGURE 3. Visual representation of the 2D discretization for a line segment.

B. HEURISTIC

We now look into the details of our proposed heuristic. Algorithm 1 contains the respective pseudocode. Fig. 4 visualizes how the material spreads to neighboring cells

Algorithm 1 Pseudocode of Our Heuristic

```
1:  COMPRESS(initial)
2:     artificial_height = maximum(initial)
3:     compressed = initial
4:     while(artificial_height > 1)
5:      reduce_artificial_height()
6:      while(max(compressed) > artificial_height)
7:       temp = zeros(max(x_coords), max(y_coords))
8:       for x in x_coords
9:        for y in y_coords
10:        diff = compressed[x,y] - artificial_height
11:        if(diff > 0)
12:         compressed[x,y] -= diff
13:         temp[next_neighbors(x,y)] += diff / 4
14:      compressed += temp
15:    return compressed
```

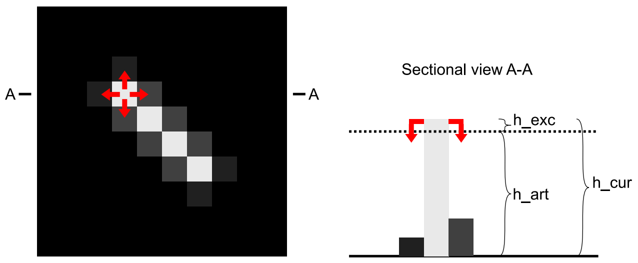


FIGURE 4. Visual representation of an exemplary iteration of the heuristic. Left: top view; right: sectional view A-A.

during a single iteration of our algorithm. First, we define an artificial height value h_{art} corresponding to the maximum TIM amount. We then enter a loop that is executed until we reach a final h_{art} equal to one. During this loop, we iteratively reduce h_{art}. While the TIM amount in any grid cell exceeds the current h_{art}, we loop over all grid cells. For each cell, we check its current TIM amount h_{cur} against h_{art}. If $h_{cur} > h_{art}$, we divide the excess amount $h_{exc} = h_{cur} - h_{art}$ by four and add it to each of the next neighboring cells in a temporary array. We subtract h_{exc} from the current cell of the compressed state. After we have looped over every cell, we update the compressed state with the temporary array. This avoids that the order of the cells within the loop has an influence on the result. Increasing the dispensed amount on the input side has the same effect as compressing down to a lower gap height.

C. ARTIFICIAL NEURAL NETWORK

The ANN is trained on data generated by the heuristic flow behavior model. As such, an advantage with regard to accuracy can not be expected. However, it can map the complex relationship between in- and output more efficiently. Programming libraries such as *Keras* [16] conveniently implement ANNs ready to be executed in parallel on a Graphical Processing Unit (GPU).

ANNs can be made up of various types of layers. Well-known architectures such as *VGG* [17], *ResNet* [18] or *Inception* [19] rely on the use of convolutional layers followed by dense layers. They have proven to work very well with image data. Since our data can be interpreted as gray scale images, we opt to work with a similar architecture. Details regarding the architecture definition and the training process are presented in Section IV.

IV. EXPERIMENTAL SETUP

This section contains information on how we set up our models and experiments. This includes the performance benchmarking for both flow behavior models. For the ANN, we describe the generation of the training data, the architecture design and the training process. For the experimental data, we describe the laboratory setup.

A. TRAINING THE ANN

The training dataset consists of 200 000 automatically generated random dispense patterns. The patterns are similar to the ones used during benchmarking as depicted in Fig. 7. We obtain the architecture of the ANN from an automated hyperparameter optimization. A template for the architecture is visualized in Fig. 5. The layers indicated in blue are always used. Yellow layers are optionally activated by the optimizer. We vary the number of convolutional layers from two to six and the number of dense layers from zero to two. The convolutional layers have either 8, 32, 128 or 512 filters with a kernel size of either three or five. If present, each dense layer has 2500 neurons. The batch size may be 8, 32 or 128. The optimizer to train the ANN is *Adam* [20] with a learning rate between 10^{-5} and 10^{-2}. We use the activation function *ReLu* for all layers except for the last, where we apply the *Sigmoid* function. The loss function to be optimized is *mean squared error*. The weights of the ANN are initialized randomly. Therefore, training an ANN multiple times on the same dataset yields fluctuating results. Preliminary manual trials of architecture tuning have shown convergence issues during the training of some hyperparameter configurations. To account for fluctuating performance and convergence issues, we train 10 ANNs for each configuration during the hyperparameter optimization. We return the lowest loss over the 10 respective runs value back to the optimizer. We use the hyperparameter optimization framework *Optuna* [21]. The hyperparameter optimization runs for 1 000 iterations. To make a high number of iterations possible, we train within each iteration for one epoch on 16 000 patterns and validate on 4 000 patterns. After finishing the hyperparameter optimization, we fine-tune the ANN by training on 160 000 patterns and validate using 40 000 patterns. During fine-tuning, the training process runs for a maximum of 100 epochs. If the validation loss does not improve for 5 consecutive epochs, the training is stopped and the model is saved. When using a different resolution, the ANN needs to be retrained as just described.

The ANN is trained for a constant output gap height. However, a change in the gap height has the same effect on the

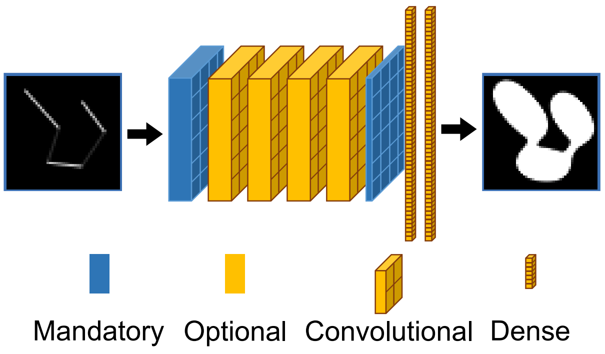


FIGURE 5. Architecture of the Artificial Neural Network (ANN). Hyperparameters such as the number of optional layers (yellow) are optimized. Mandatory layers (blue) are always included.

result as changing the input amounts. When using the ANN, different gap heights can thus be accounted for by scaling the input amounts respectively.

Since the ANN has a fixed input format, the ANN input dimensionality is set according to the maximum pattern length. To process shorter patterns, we simply append pattern segments with zero amounts up to the maximum length.

B. PHYSICAL EXPERIMENTS

To validate our model, we carry out physical experiments. We dispense TIM in various different patterns and compress it as when joining a heatsink.

The machine used for dispensing is an automated Computerized Numerical Control (CNC) machine. Its type is almost identical to the ones that are used in automotive series production. We transfer the patterns into G-code, which is a format that is readable on this kind of machine. The TIM is dispensed onto glass plates with a dimension of 70 mm $\times$ 70 mm. Thin metal plates with a carefully machined height are put on the edges of the glass plate. This ensures a uniform final gap height when putting a second glass plate on top and pushing it downwards. The dispensed and compressed states of TIM are shown for an exemplary pattern in Fig. 6. An image of the compressed state is recorded. An automatic segmentation of the blue color hue is applied and yields a representation in the same discretized format as introduced previously. Thus, each pixel is either entirely full or empty. Since the final gap height is low, the error at the area boundary made by this assumption is sufficiently small. After segmentation, the resolution is scaled down to the same resolution as in the heuristic model. During downscaling, we apply a linear interpolation between neighboring cells. The zoom level is adjusted uniformly for all experiments. This is done as a post-processing step and has the same effect as adjusting the vertical camera position. Since the experiments are carried out manually, some samples are shifted slightly. Those translational errors are corrected by re-centering each pattern during post-processing. The post-processing, of course, does not involve a modification of the overall pattern

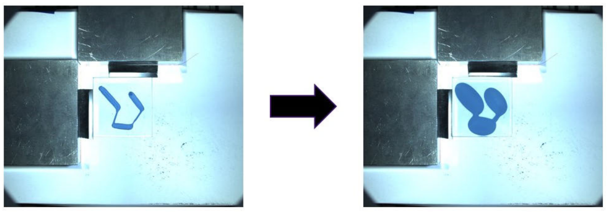


FIGURE 6. Laboratory experiments using transparent glass plates to compress TIM. Left: state before compression; right: state after compression.

shape, since this would distort the error evaluation of the model.

We further evaluate the TIM flow behavior in a physical experiment using a real product sample. This product involves a Printed Circuit Board (PCB) with mounted electronic components to be cooled. The housing is pressed onto the PCB and TIM spreads over both joining partners to form a thermal connection. In contrast to our laboratory experiments as just described, we cannot control the actual gap height that precisely in this case: multiple parts are joined together, with each part having individual mechanical tolerances. Furthermore, the PCB itself bends during the joining process. We thus use this experiment for a qualitative rather than a quantitative assessment. For comparison with our model we choose the total TIM amount to fit the actually observed amount. Instead of evaluating the general fit of our model, we evaluate the fit with respect to only the shape of the predicted coverage area outline.

C. BENCHMARKING

During this study, we use a resolution of 50 cells $\times$ 50 cells. This resolution was selected through preliminary trials with different resolutions and represents a compromise between sufficient accuracy and computational effort.

One part of our benchmarking covers the error of our simulation models. This involves the comparison of the entire simulation pipeline consisting of the discretization and either flow behavior model to the physical experiments. It further involves the error between the outputs of the heuristic and the ANN. In all cases, we calculate the absolute error of the respective compressed states:

$$e_{comp} = \sum_{i=1}^{50} \sum_{j=1}^{50} |m_{a,comp,ij} - m_{b,comp,ij}|, \tag{1}$$

with $m_{a,comp,ij}$ and $m_{b,comp,ij}$ being the TIM amounts per grid cell (i, j) in the compressed states. The indices a and b refer to either the experiment and a flow behavior model or the heuristic and the ANN. We then divide the absolute error by the sum of the covered cells

$$e_{rel} = \frac{e_{comp}}{\sum_{i=1}^{50} \sum_{j=1}^{50} m_{a,comp,ij}} \tag{2}$$

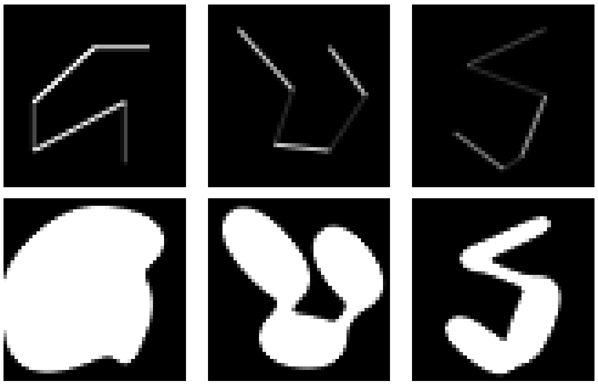


FIGURE 7. **Three exemplary patterns used in our computation time and error benchmarking. Top row: before compression, bottom row: after compression.**

and calculate its mean

$$\bar{e}_{rel} = \frac{1}{N_{pat}} \sum_{k=1}^{N_{pat}} e_{rel,k} \tag{3}$$

across $N_{pat} = 50$ dispense patterns. This relative error measure yields a better intuition of the model accuracy across the different dispense patterns.

Besides the error, we also evaluate the computation speed for our model. Both flow behavior models are called from a Python script. The library *timeit* [22] measures the computation time of a code snippet. Setup code, such as code for loading data and models, is executed separately and not included into the measurement. Background processes may interfere with the program being measured and spuriously lengthen the computation time. For this reason, it is specifically not recommended to report mean and average computation times for multiple runs of the same code [22]. Thus, we execute $N_{runs} = 10$ runs per measurement and store the minimum value

$$t_{min} = \min t_l,\ l \in \{1, \dots, N_{runs}\} \tag{4}$$

for further evaluation. Since measurement time varies for different patterns, we measure the computation time for the compression of N_{pat} individual patterns. Examples are shown in Fig. 7.

We report the mean value

$$\bar{t} = \frac{1}{N_{pat}} \sum_{n=1}^{N_{pat}} t_{min,n} \tag{5}$$

for the computation time $t_{min,n}$ across $N_{pat} = 50$ paths and the respective standard deviation

$$s = \sqrt{\frac{1}{N_{pat} - 1} \sum_{n=1}^{N_{pat}} (t_{min,n} - \bar{t})^2}. \tag{6}$$

The computation time $t_{min,n}$ for an individual pattern is, as just described, the minimum time across 10 runs per individual pattern. All computations are executed on a workstation with an INTEL E5-2680 processor and four GPUs of type NVIDIA RTX 2080Ti.

V. RESULTS

This section contains the results regarding the heuristic, the ANN and the physical experiments. We record the setup and computation time for all three approaches and calculate the relative absolute error of the compressed states as described previously. Results are listed in Table I.

First, we give a deeper insight into the process of setting up the ANN. We determine the hyperparameters by carrying out a hyperparameter optimization as described in Section IV. We obtain the following architecture: the first layers are five convolutional layers with 8, 128, 512, 512 and 8 filters respectively and a filter size of 5 × 5. They are followed by the mandatory convolutional layer with one filter and a filter size of 3 × 3. Two dense layers with 2500 neurons are appended. The best remaining hyperparameters are a batch size of 8 and a learning rate of 0.0002. The entire hyperparameter optimization process with 1 000 iterations takes one week. The fine-tuning of the final architecture takes about 12 hours. It takes about one week to create the training dataset for the ANN, which involves the simulation of 200 000 patterns. Those steps need to be carried out only once for a specific input resolution. They run fully automatic and do not need any human intervention.

We compare the trained ANN with the original heuristic approach. The error according to (3) is **3 %**. Fig. 8 shows the output of the trained ANN as compared to the heuristic. While some errors are prevalent in the outermost cells, the ANN manages to fit the data well.

We now look closer into the laboratory experiments, which are carried out as described in Section IV-B and form an independent test dataset with unseen patterns. An exemplary sample of those dispense patterns is shown in Fig. 7. Each experiment takes 30 minutes. This time includes sample preparation, dispensing, compression and post-processing of the results. The experiments serve as ground truth and therefore are listed with an error equal to zero. We are aware that they are still subject to error sources such as mechanical tolerances or measurement noise.

Our heuristic flow behavior model assumes parallel joining surfaces. A slight tilt due to mechanical tolerances is present in real experiments. This tilting varies during production for each individual workpiece and can also not be considered well with e.g. CFD simulations. It is present in our laboratory experiments and thus included into our reported error estimation.

For both of our flow behavior models, we calculate the mean relative error with respect to the experiments according to (3). The heuristic and the ANN are both able to predict the compressed shape well, with an error of **11 %** and **12 %** respectively across the 50 evaluated patterns. A visual comparison of the ANN with the experiments is presented in Fig. 9. Further samples are shown in the appendix. The left column shows the compressed state as output from the ANN for three different dispense patterns. The middle column shows the compressed state acquired from the experiments.

TABLE I. Comparison of simulation methods during deployment in manual pattern design.

Method	Mean relative error $\bar{\mathrm{e}}_{\mathrm{rel}}$	Setup time	Computation time $\bar{\mathrm{t}}$
Experiment	0	30 min	-
CFD	-	10 - 60 min	60 - 120 min
Heuristic	11 %	1 min	3.41 ± 2.75 s
Neural network	12 %	1 min	0.11 ± 0.001 s

The right column shows the difference between ANN output and experimental data along with the error score according to (2). It is shown that the overall shape is almost identical in most cases. Errors occur mainly in outer cells of the covered area.

The example of a real ECU depicted in Fig. 10 shows that the model fits the TIM behavior not only in a laboratory environment, but also in the real product. In this case, the cooling surface area is bounded and excess TIM flowing beyond the boundaries will not be compressed any further. The experiments we conducted in the laboratory ensure a complete compression of the entire shape. While the real product is certainly an important benchmark, the laboratory experiments give a deeper insight into the model accuracy.

Patterns with high errors are often characterized by the entrapment of air. Once the TIM pattern forms such an enclosed void area and is then compressed further, the entrapped air is put under pressure as well. This pressure counteracts the material flow into the void area. This effect is not taken into account by our model. It can be seen in Fig. 11 that our model predicts a quite small void, while the experimental data suggests that the TIM flows rather towards the outer areas than towards the center. The shown example exhibits a relative error of 23.0 % according to (2). The predicted shape is generally still reasonable even in those cases.

The initial dispense pattern can be calculated rather straightforward from the pattern parameters as described previously. The setup time for simulating a certain dispense pattern is therefore relatively low and takes up to one minute. This procedure is equal for both flow behavior models. The computation time amounts to **3.41 s** on average for the heuristic and displays a rather large variance across different patterns. The ANN can be executed consistently in **0.11 s**.

To enable other researchers to reproduce our results with the ANN easier, we have uploaded our test dataset and our training dataset to *Kaggle* [23]. Our datasets are available at https://www.kaggle.com/datasets/simonbaeuerle123/timflow/.

VI. DISCUSSION

The CFD simulations can be considered state-of-the-art in this field of simulation. We do not aim to analyze them deeper within this work, since they have been used extensively

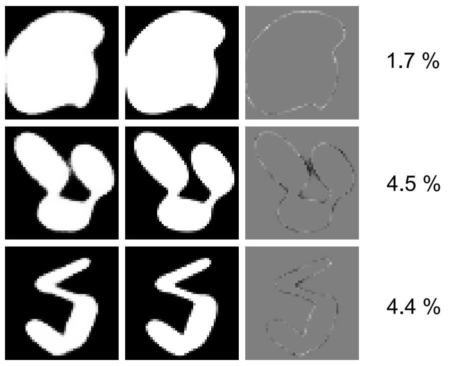


FIGURE 8. Output of the ANN as compared to data from the heuristic model. Left column: output of ANN for different dispense patterns, center column: output from heuristic, right column: difference between outputs of ANN and heuristic. Furthermore, we list the error score according to (2).

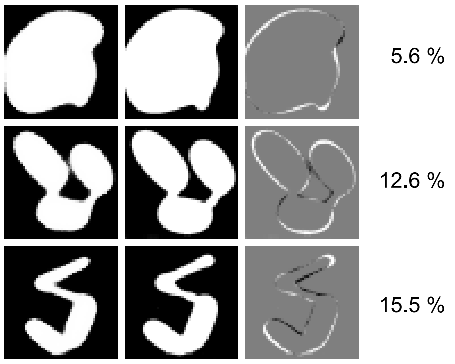


FIGURE 9. Validation on experimental data. Left column: output of ANN for different dispense patterns, center column: experimental data, right column: difference between ANN output and experimental data. Furthermore, we list the error score according to (2).

for over 20 years in a wide range of different applications. We have shown our samples to two experienced simulation experts and asked them for their professional opinion. They are regularly occupied with simulations of similar dispense patterns. They estimate the setup time for such dispense patterns to be in the range of 10 min up to 60 min. 10 min would include a very basic setup without much detail. 60 min would include a more elaborate setup, e.g. a fine modeling of 3D roundings of the dispense pattern. The computation time is estimated by consulting the simulation logging files for similar patterns and is in the range of 60 - 120 min. Due to the high effort of CFD simulations, we omit the exact error calculation on our 50 samples. We do not claim that our new simulation approach offers an advantage over the CFD simulation with regard to the error. However, they are clearly outperformed by both of our proposed surrogate flow behavior models with regard to computational speed.

The accuracy of our heuristic model is high enough to support manual development work. Its application can thus save a significant effort during manual dispense pattern design for electronics packages. It enables the use of gradient-free optimizers, which could partially automate

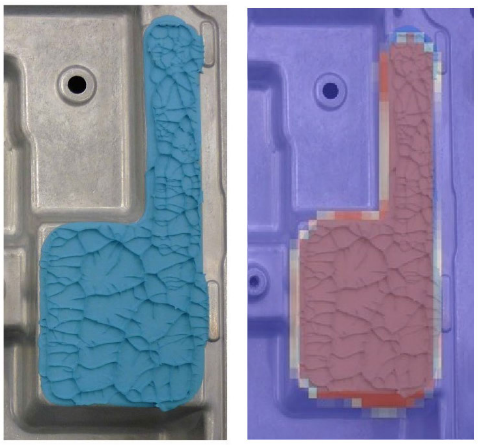


FIGURE 10. **Overlay with the compressed TIM on a real ECU. Left: image of compressed TIM in a physical experiment; right: additional overlay of the compressed state as obtained from the heuristic.**

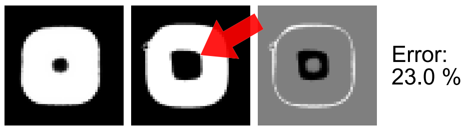


FIGURE 11. **Dispense pattern involving a void area due to air entrapment. Left: output of ANN, center: experimental data with marked air entrapment, right: difference between ANN output and experimental data. Furthermore, we list the error score according to (2).**

the dispense pattern design process. CFD simulations and physical experiments will still be necessary, but only for fine-tuning during the last design cycles. This is specifically the case in tests involving the design limits, e.g. experiments, which cover the highest expected mechanical tolerances. We do not claim to replace CFD simulations or experiments fully, but rather to reduce the number of trials.

During manual dispense pattern design, using the ANN has the following advantages. The computation time improves by a factor of about 30, whereas the mean relative error increases only by one tenth (see Table I). In contrast to the initial setup duration of the ANN, the duration of the design process is relevant for the time-to-market and needs to be carried out for each individual product. The time-to-market is valuable and product departments may require a quick feedback from design engineers. During manual dispense pattern design, a design engineer is actively working with the model. His time is more valuable than the training time. When embedding the flow behavior model into a user interface, the results from the ANN are available almost in real-time. This is an advantage in terms of user experience. Thus, it is better to invest time into the automated setup of the ANN and in turn gaining more valuable time during the dispense pattern design.

During automated dispense pattern design, using the ANN has the following advantages. While the heuristic could be paired with a gradient-free optimizer to perform a zero-order optimization, this would involve many calls to the model in order to calculate the objective function during the optimization progress. The speed-up of computation time by the ANN can thus leveraged higher as compared to the manual dispense pattern design. Gradient-based optimizers for first-order optimization could only be paired with the heuristic when utilizing the numeric gradient. This is very time consuming and would involve a high number of calls to the model during each individual optimization step. For a trained ANN, the gradient of its output with respect to its input can be retrieved easily. *Tensorflow* [24] includes an Application Programming Interface (API) called *Gradient-Tape*. During the training procedure, it is used to update the weights of the network. During an optimization involving the trained ANN, the *GradientTape.gradient()* function could be used to compute the gradient. The ANN enables the use of gradient-based optimizers and would significantly speed up the optimization when using gradient-free optimizers. In both cases, the time investments into the setup of the ANN would be compensated by the time savings during the optimization quickly. Automated pattern design with state-of-the-art optimizers could explore a much larger range of the design space and thus lead to better solutions than those found via manual trial-and-error iterations. Several solution candidates can be generated by executing the optimization with slightly varied settings. The design engineer can then choose the most promising patterns.

During training of the ANN, we only used open source software libraries. This allows the integration of our model into a custom user interface, i.e. independently from proprietary simulation software. Especially a web-based implementation could provide easy access to design engineers without any license costs as seen with e.g. CFD supported tools.

The hyperparameter optimization supported the training process of the ANN. Compared to preliminary manual hyperparameter tuning, the automated hyperparameter optimization yielded better results. This is valid not only with regard to the model performance, but also with regard to the convergence behavior of the training process. We thus recommend using an automated hyperparameter optimization when training on data of this kind. During preliminary experiments, we have also tested the hyperparameter optimization library *KerasTuner* [25] as an alternative to *Optuna*. We have found *Optuna* easier to work with. According to the number of citations, it seems to be a popular choice among other researchers as well. In its backend we utilize the optimization algorithm *tree-structured Parzen estimator* [26]. The successful hyperparameter optimization that we report validates a) our choice of this library and b) the effectiveness of the embedded optimization algorithm. The research field Neural Architecture Search (NAS) [27] has received increasing attention in recent years. We would like to refer anyone interested in more elaborate studies regarding NAS to the cited review and the works referenced within.

Our model assumes an infinitely wide planar surface and thus might model the compression behavior even beyond the physical area boundaries. This overflow is generally not

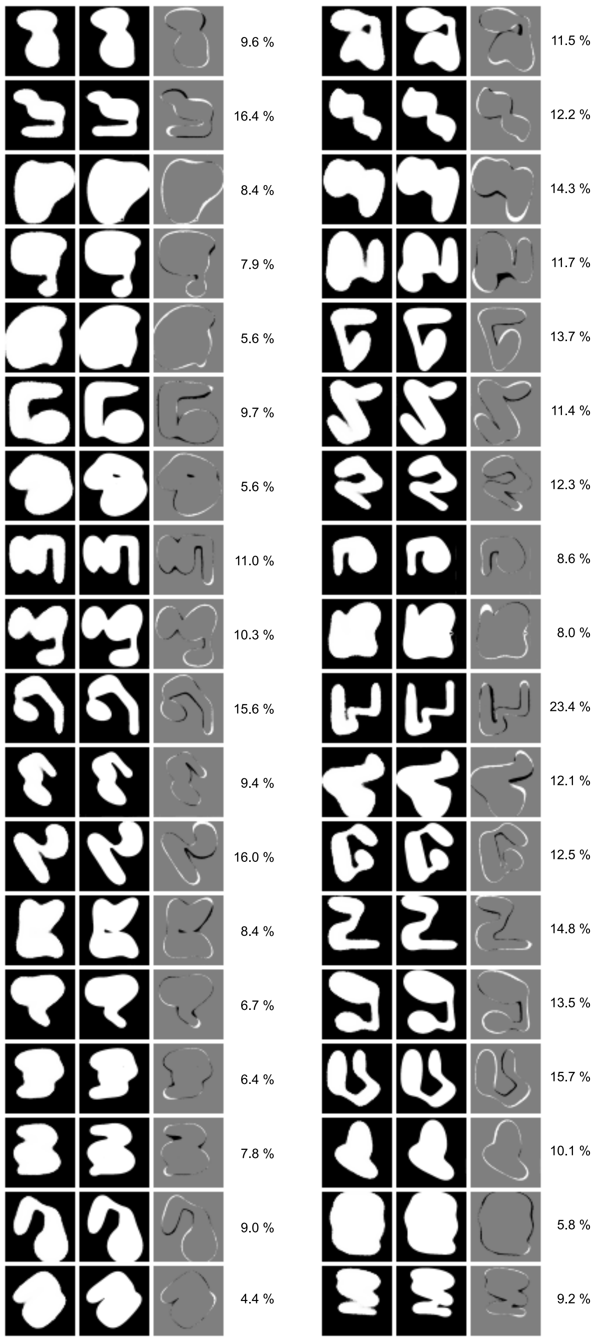


FIGURE 12. Validation on experimental data. Left column: output of ANN for different dispense patterns, center column: experimental data, right column: difference between ANN output and experimental data. Furthermore, we list the error score according to (2).

desired in practical application. Especially in the case of taboo zones such as screw holes being positioned next to the cooling surface, this needs to be avoided. Similarly, air entrapments are not desired in practical application, since the air significantly impairs thermal performance [7]. The failure to model those effects accurately is an evident limitation of our model. However, it is sufficient to predict the presence of a violation of a taboo zone or the formation of a void. A highly precise but time-consuming CFD simulation to estimate the respective quantitative extent offers only little additional benefit. We argue that in those cases, a quick feedback is more valuable than an exact feedback. This is especially the case for the first design iterations. With the last design iterations, it is obviously different: manual experiments are carried out to validate the final design choice.

The dispense pattern shown in Fig. 10 was designed manually by an engineer without the support of our model. The effect of compressed material extending beyond the cooling area can be observed for example at the bottom part of Fig. 10. This indicates that the dispense pattern, which is the result of a conventional pattern design process, could have been improved. When the dispense pattern would fit the cooling area perfectly, no overflowing material would be visible. This has an effect on both material usage and cycle time and motivates the use of our model for future product variants.

As of now, the ANN is trained solely on data from the heuristic. It inherits the previously discussed limitations from the heuristic model: the ANN will make similar errors to the heuristic, e.g. when encountering air entrapments or non-planar surfaces. However, the ANN has the ability adapt to new experimental data. With data from series production, the ANN could further be fine-tuned and improved to reflect the actual process even better. The real-world process behavior can be integrated into our model and it can be used as a Digital Twin for the design of future product generations.

VII. CONCLUSION

We present two flow behavior models, which can quickly predict the flow behavior of TIM when joining the heatsink. Our proposed heuristic aids design engineers during the definition of the initial dispense pattern by providing a quick and easy method to estimate the compressed state. This reduces the need for elaborate CFD simulations and manual experiments with product samples. The time-to-market can thus be shortened for a variety of ECUs and power electronics components. Training an ANN on data from our heuristic reduces accuracy only slightly, but yields a significant speed-up of computation time. Using an ANN thus makes the manual design process even more convenient. It further allows the efficient usage of optimizers for an automated dispense pattern optimization. We show that the predicted compressed state fits experimental results well. This is true not only in the laboratory, but also for a real ECU. Future work includes the development of a method for automated dispense pattern optimization on the basis of this model.

APPENDIX.

In Fig. 12, we present further examples from our experimental dataset.

ACKNOWLEDGMENT

The authors would like to thank Ralph Nyilas and Andras Horvath for the helpful discussions regarding the flow behavior of TIM. They also thank Balazs Solymossy and Istvan Horvath from the Simulation Department for their professional opinion regarding CFD models. They further thank Hack-Min Kim, Vivien Reuscher, and Roderich Zeiser, for their support in carrying out the laboratory experiments.

AUTHOR STATEMENT

Author Contributions: CRediT [28]: *Writing-Original Draft*: Simon Baeuerle; *Writing-Review & Editing*: Jonas Barth, Andreas Steimer, and Ralf Mikut; *Conceptualization*: Simon Baeuerle, Jonas Barth, Andreas Steimer, and Ralf Mikut; *Investigation*: Simon Baeuerle and Marius Gebhardt; *Methodology*: Simon Baeuerle and Andreas Steimer; *Software*: Simon Baeuerle and Marius Gebhardt; *Supervision*: Jonas Barth, Andreas Steimer, and Ralf Mikut; *Project Administration*: Jonas Barth and Ralf Mikut; and *Funding Acquisition*: Jonas Barth and Ralf Mikut.

REFERENCES

[1] J. J. Licari and D. W. Swanson, *Adhesives Technology for Electronic Applications: Materials, Processing, Reliability*, 2nd ed. Oxford, U.K.: William Andrew, 2011.

[2] S. Baeuerle, M. Böhland, J. Barth, M. Reischl, A. Steimer, and R. Mikut, "CAD-to-real: Enabling deep neural networks for 3D pose estimation of electronic control units: A transferable and automated approach for industrial use cases," *at-Automatisierungstechnik*, vol. 69, no. 11, pp. 880–891, 2021.

[3] B. Zhou, T. Pychynski, M. Reischl, E. Kharlamov, and R. Mikut, "Machine learning with domain knowledge for predictive quality monitoring in resistance spot welding," *J. Intell. Manuf.*, vol. 33, no. 4, pp. 1139–1163, Apr. 2022.

[4] L. F. C. S. Durão, S. Haag, R. Anderl, K. Schützer, and E. Zancul, "Digital twin requirements in the context of industry 4.0," in *Product Lifecycle Management to Support Industry 4.0* (IFIP Advances in Information and Communication Technology), P. Chiabert, A. Bouras, F. Noël, and J. Rios, Eds. Cham, Switzerland: Springer, 2018, vol. 540, pp. 204–214.

[5] M. Ekpu, R. Bhatti, N. Ekere, S. Mallik, and K. Otiaba, "Effects of thermal interface materials (solders) on thermal performance of a microelectronic package," in *Proc. Symp. Design, Test, Integr. Packag. MEMS/MOEMS*, Cannes, France, Apr. 2012, pp. 154–159.

[6] T. M. Kesarkar and N. Kumar Sardana, "How TIM impacts thermal performance of electronics: : A thermal point of view study to understand impact of thermal interface material (TIM)," in *Proc. Int. Conf. Electron. Packag. (ICEP)*, Niigata, Japan, Apr. 2019, pp. 200–206.

[7] A. Gowda, D. Esler, S. Tonapi, K. Nagarkar, and K. Srihari, "Voids in thermal interface material layers and their effect on thermal performance," in *Proc. 6th Electron. Packag. Technol. Conf. (EPTC)*, Singapore, Dec. 2004, pp. 41–46.

[8] T.-Y. Lee, "An investigation of thermal enhancement on flip chip plastic BGA packages using CFD tool," *IEEE Trans. Compon. Packag. Technol.*, vol. 23, no. 3, pp. 481–489, Sep. 2000.

[9] R. Comminal, M. P. Serdeczny, D. B. Pedersen, and J. Spangenberg, "Numerical modeling of the strand deposition flow in extrusion-based additive manufacturing," *Additive Manuf.*, vol. 20, pp. 68–76, Mar. 2018.

[10] R. Prasher, "Thermal interface materials: historical perspective, status, and future directions," *Proc. IEEE*, vol. 94, no. 8, pp. 1571–1586, Aug. 2006.

[11] C. Lin and D. D. L. Chung, "Rheological behavior of thermal interface pastes," *J. Electron. Mater.*, vol. 38, no. 10, pp. 2069–2084, Oct. 2009.

[12] L. H. Sinh, J.-M. Hong, B. T. Son, N. N. Trung, and J.-Y. Bae, "Thermal, dielectric, and rheological properties of aluminum nitride/liquid crystalline copoly (ester amide) composite for the application of thermal interface materials," *Polym. Compos.*, vol. 33, no. 12, pp. 2140–2146, Dec. 2012.

[13] G. X. Gu, C.-T. Chen, and M. J. Buehler, "De novo composite design based on machine learning algorithm," *Extreme Mech. Lett.*, vol. 18, pp. 19–28, Jan. 2018.

[14] A. Koeppe, C. A. Hernandez Padilla, M. Voshage, J. H. Schleifenbaum, and B. Markert, "Efficient numerical modeling of 3D-printed lattice-cell structures using neural networks," *Manuf. Lett.*, vol. 15, pp. 147–150, Jan. 2018.

[15] J. F. Hughes, *Comput. graphics: Princ. Pract.*, 3rd ed. Upper Saddle River, NJ, USA: Addison-Wesley, 2014.

[16] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2818–2826.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[21] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019, *arXiv:1907.10902*.

[22] Python Software Foundation. (Jan. 2022). *Timeit—Measure Execution Time of Small Code Snippets*. [Online]. Available: https://docs.python.org/3/library/timeit.html

[23] Kaggle. (2024). *Kaggle: Your Home for Data Science*. [Online]. Available: https://www.kaggle.com/

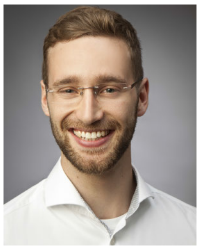
[24] M. Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/

[25] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, and L. Invernizzi. (2019). *Keras Tuner*. [Online]. Available: https://github.com/keras-team/keras-tuner

[26] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, vol. 24. Granada, Spain: Curran Associates, Dec. 2011.

[27] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter, "Neural architecture search: Insights from 1000 papers," 2023, *arXiv:2301.08727*.

[28] A. Brand, L. Allen, M. Altman, M. Hlava, and J. Scott, "Beyond authorship: Attribution, contribution, collaboration, and credit," *Learned Publishing*, vol. 28, no. 2, pp. 151–155, Apr. 2015, doi: 10.1087/20150211.

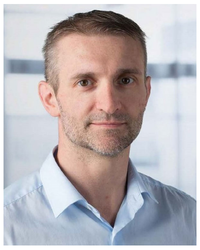


SIMON BAEUERLE received the bachelor's and master's degrees in mechanical engineering from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree with a joint research project of the Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, and Robert Bosch GmbH, Reutlingen, Germany. His research interests include machine learning, image processing, and industrial AI.

MARIUS GEBHARDT wrote his master's thesis within the scope of this research project in cooperation with Reutlingen University, Reutlingen, Germany, and Robert Bosch GmbH, Reutlingen. His research interests include machine learning, data science, and industrial AI.

JONAS BARTH is currently with Robert Bosch GmbH, Reutlingen, Germany. His research interests include machine learning, data science, and industrial AI.

RALF MIKUT received the Diploma degree in automatic control from the Dresden University of Technology, Dresden, Germany, in 1994, and the Ph.D. and Habilitation degrees in mechanical engineering from the University of Karlsruhe, Karlsruhe, Germany, in 1999 and 2007, respectively. Since 2011, he has been an Adjunct Professor with the Faculty of Mechanical Engineering and the Head of the Research Group Automated Image and Data Analysis, Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology (KIT), Germany. His current research interests include machine learning, image processing, life science applications, and smart grids.



ANDREAS STEIMER received the bachelor's degree in physics and the master's degree in microsystems technology from the Albert Ludwigs University of Freiburg, Freiburg im Breisgau, Germany, and the Ph.D. degree from the Institute of Neuroinformatics, ETH Zurich, in 2012. He held a postdoctoral position with the Institute of Neuroinformatics, ETH Zurich, until 2013. As a Researcher with the University Hospital of Bern (Inselspital), Bern, Switzerland, he was involved in the application of machine learning methods to EEG data from epilepsy patients, until 2017. He is currently associated with Robert Bosch GmbH, where his research is focused on industrial AI and centered around a diverse range of fields, such as traceability, generative modeling of images and time series, classification of failure parts, and model monitoring.