The International Journal of Advanced Manufacturing Technology (2023) 125:5605–5618
https://doi.org/10.1007/s00170-023-10973-6

ORIGINAL ARTICLE

CAD-based data augmentation and transfer learning empowers part classification in manufacturing

Patrick Ruediger-Flore[1] · Moritz Glatt[1] · Marco Hussong[1] · Jan C. Aurich[1]

Received: 8 June 2022 / Accepted: 20 January 2023 / Published online: 18 February 2023

Abstract

Especially in manufacturing systems with small batches or customized products, as well as in remanufacturing and recycling facilities, there is a wide variety of part types that may be previously unseen. It is crucial to accurately identify these parts based on their type for traceability or sorting purposes. One approach that has shown promising results for this task is deep learning–based image classification, which can classify a part based on its visual appearance in camera images. However, this approach relies on large labeled datasets of real-world images, which can be challenging to obtain, especially for parts manufactured for the first time or whose appearance is unknown. To overcome this challenge, we propose generating highly realistic synthetic images based on photo-realistically rendered computer-aided design (CAD) data. Using this commonly available source, we aim to reduce the manual effort required for data generation and preparation and improve the classification performance of deep learning models using transfer learning. In this approach, we demonstrate the creation of a parametric rendering pipeline and show how it can be used to train models for a 30-class classification problem with typical engineering parts in an industrial use case. We also demonstrate how our method's entropy gain improves the classification performance in various deep image classification models.

Keywords Part classification · Object recognition · Machine learning · Data augmentation and synthesis · CAD-based rendering · Transfer learning

1 Introduction

Manufacturing systems show a vast demand for methods to detect distinctive parts based on appearance. Within the growing field of remanufacturing, a labor-intensive task is to sort incoming parts (e.g., automotive components [25]) for further part type-specific processing. Furthermore, especially manufacturing systems that incorporate small batches or customized products have highly varying material flows. In these scenarios, it is crucial to track the location of distinct parts and products. Using hardware-based identification technologies (e.g., RFID) requires costly infrastructure and causes handling efforts, such as placing tags on workpiece carriers. In both cases, advances in deep learning–based image classification approaches offer broad potentials [35]. Within the first use case of remanufacturing, models that were trained with image data of relevant part classes can be used to determine a class assignment based on the visual appearance of the part. In customized part manufacturing, the appearance of a part can be seen as a distinct characteristic, which can be used to detect the parts' presence at a specific station. These examples show that classifying physical parts offers broad economic improvement potentials due to reduced manual activities and small required investments.

ML techniques provide a robust foundation for the described task, as they can learn classification features independently from real-world conditions like varied lighting, part orientation, or pollution. However, deep neural networks are the most potent class of ML-based image classification methods, which require large amounts of training data [6]. While other industries (e.g., automotive) tend to have high numbers of training images, this is often a challenge in manufacturing. Especially in small-batch manufacturing or in scenarios with previously unseen

Moritz Glatt, Marco Hussong, and Jan C. Aurich contributed equally to this work.

✉ Patrick Ruediger-Flore
patrick.ruediger@mv.uni-kl.de

[1] Institute for Manufacturing Technology and Production Systems (FBK), RPTU Kaiserslautern, P.O. Box 3049, 67653, Kaiserslautern, Germany

objects, obtaining sufficient image data per object is often challenging. Even in scenarios with larger batches where it is feasible to generate sufficient image data or in already trained classification models, methods to facilitate the generation of training data help to adapt the models to new tasks quickly. Synthetic data and data augmentation techniques are favorable for ensuring the generalization of a model, as the collected data most likely contains an unbalanced amount of "good" examples. However, most of these techniques use probabilistic approaches. In this context, the generation of synthetic data offers the possibility to lower the required amount of real-world image data. At the same time, manipulating the original training data results in difficult-to-explain modifications and thus leading to less explainable models. To overcome the overfitting effects while using image augmentation techniques on small image datasets, we propose a method to derive synthetic data from photorealistic renderings of CAD data (more specifically, the geometry export in the form of an STL or OBJ file), benefitting from the condition that this data is broadly available in manufacturing settings.

In comparison, the direct usage of images from the CAD data for training only generalizes well if the latter model application scenario takes the images in a highly controlled environment. Consequently, the generalization capability of models trained this way is expectantly lower. Thus, our approach provides an image data synthesis pipeline for manufacturing parts from CAD data with increased realism. This allows for higher detection rates, lower training effort and more over it enables the usage of deep image classification in low volume manufacturing scenarios. In order to further ensure a suitable generalization of our classification models, the most significant effect is achieved through combining synthetic data with transfer learning. Here, the ability of the latest generation of deep neural network architectures and their pre-trained models is used to implicitly interpret rudimentary properties of an image, such as corners, edges, and simple patterns, as features. This generalization capability enables training new classes with only a minor amount of sample data. The major challenges for applying deep image classification in manufacturing are the need for large training datasets, a lack of spatial invariance, distinguishing between objects of interest and background, and addressing inconsistencies in lighting. Additionally, implementing those deep neural network classifiers in the industry would require infrastructure upgrades and cloud computing services, which come with a financial cost [18, 24]. Hence, we investigate the effect of increased realism through data synthesis from CAD on the classification performance and the usage of transfer learning and its effect on the training data size for current state-of-the-art image classification models. For our study, we can define *Transfer Learning* as an approach for exploiting an existing model in a related task (e.g., image classification) for a new task through the transfer of knowledge that has already been learned. Consequently, we introduce the benefits of transfer learning on deep image classification for part classification. We can now summarize the contributions of this paper as follows:

- We demonstrate a photorealistic rendering pipeline based on CAD data to simplify the generation of explainable synthetic training data, which is both highly customizable and automatable.
- Furthermore, we show how the achieved realism increase for the training images, with our method, moderates the classification performance significantly.

To ensure the generalization of the postulated effect for deep image classification, we evaluate the classification differences for different state-of-the-art deep image classification base models and their applicability for few-shot learning. Finally, an industrial use case shows the feasibility of the approach for supporting order-picking tasks in a small-batch scenario. We demonstrate the applicability of our approach in comparison to standard augmentation techniques and the usage of various image classification model architectures based on an exemplary and openly available dataset containing typical manufacturing parts.

2 Image classification in manufacturing applications

In terms of classifying entire geometric objects, the predominant use of deep learning is the detection of workpieces: The approach by Kruger et al. [17] applies several CNN architectures (e.g., ResNet) for workpiece classification. Implementing these common architectures in easy-to-access libraries (e.g., TensorFlow for Python) has fostered their application for object detection in manufacturing. One big field is the detection of surface defects for quality assurance, e.g., [33]. Part sorting is also the focus of the deep learning application in the work of [32]. Subakti et al. [28] have developed an AR system combining deep learning to monitor the machine status in manufacturing systems. The work of Blondheim et al. [1] focuses on an explainability approach for ML models in defect classification applications. The use of image classification in manufacturing has the potential to greatly improve product cycle times, cost and reliability [24]. There are several advantages to using convolutional neural networks (CNNs) in machine learning tasks, such as image recognition. These include the ability to capture features that traditional neural networks cannot, reduced complexity and improved computational efficiency, improved model performance, the ability to extract features

from a trained network, and the ability to be used for 1D and 3D image classification. However, there are also some limitations to consider when using CNNs, such as the need for large training datasets, a lack of spatial invariance in the input data, the challenge of tuning hyperparameters, and the need for input normalization for datasets with diverse parameters [18]. Regarding the application in manufacturing environments, the challenges that need to be addressed in order to effectively use image classification are developing filters that can distinguish between objects of interest and background features and addressing inconsistencies in lighting. Additionally, implementing those deep neural network classifiers in industry would require infrastructure upgrades and the use of cloud computing services, which come with a financial cost [24].

2.1 Few-shot learning

The successful application of deep learning in many different areas fostered its use in manufacturing in the past years. However, deep techniques require a large amount of labeled data that involves high effort and cost, especially in manufacturing. For instance, Kruger et al. [17] state that collecting and preparing sufficient image data requires several months of full-time work. Accordingly, this significantly hampers the application of deep image classification in industrial settings. As a result, recent works have dealt with applying one-shot and few-shot techniques that require less labeled data. The work of Deshpande et al. [8] used a Siamese CNN for one-shot recognition of defects in steel surfaces. A few-shot learning model for the prediction of bunching defects in industrial hoses based on simulated data is outlined by Khajezade et al. [16]. A review of one-shot learning techniques for custom identification tasks is done by O' Mahony et al. [23]. In the work of Wu et al. [34], a few-shot transfer learning approach is used for fault diagnosis on bearings and gears during manufacturing. Similarly, in the work of Zhang et al. [37], a few-shot learning method is proposed based on a Siamese neural network for fault diagnosis of rolling bearings. In Zhou et al. [39], a Siamese neural network–based few-shot learning is described to detect anomalies in industrial cyber-physical systems.

2.2 Image data augmentation

2.2.1 General data augmentation on images

General techniques in data augmentation can be divided into basic image manipulations and deep learning–based techniques. Basic image manipulations include geometric transformations, color space augmentations, kernel filters, mixing of images, and random erasing, whereas deep learning–based techniques include feature space augmentation, generative adversarial networks, and neural style transfer [26]

Geometric transformations change the shape of the image, for example, cropping, rotation, translation, flipping, or noise injection. Color space augmentations involve altering the color distribution of an image by changing the RGB values of pixels [26]. Kernel filters are used to sharpen or blur a picture by sliding a filter in the shape of a matrix over the pixel values of an image [15]. The mixing of images is done by taking several images and merging them with different techniques such as cropping or averaging pixel values [26]. Random erasing works by randomly defining a patch of an image and changing the pixel values of the patch, typically either to 0 (black) or 255 (white), and therefore deleting parts of an image, forcing the model to learn other descriptive features of the image [38].

Feature space augmentation is a deep learning–based technique of data augmentation manipulating the intermediate representations of an artificial neural network [26]. Generative adversarial networks can create new data points by training two separate networks: the generator and the discriminator. The generator learns to create an image. The discriminator then judges this image. The discriminator has, therefore, to decide if the input image is from the training set or created by the generator [11]. The neural style transfer algorithm manipulates the sequential representations across a neural network by transferring the style of the underlying image to another while sustaining the original content of the underlying image [10].

2.2.2 Data augmentation within manufacturing applications

Applying machine learning algorithms in manufacturing has led to data augmentation techniques to solve the issue of a low quantity of data. Data augmentation was not purely used for image data but included augmentation techniques on time series. For example, Lyu et al. [21] proposed a data augmentation technique consisting of the weighted combination of Gaussian noise and signal stretching on time series data for fault diagnosis of motors. Li et al. [20] use five data augmentation techniques, namely Gaussian noise, masking noise, signal translation, amplitude shifting, and time stretching, on time series data to predict rotating machinery fault diagnosis. For image augmentation, Meister et al. [22] propose a synthetic data augmentation for the fiber layup inspection process in the aerospace industry

by combining geometrical transformations and generative adversarial networks. Chiu and Chen [3] used rotations and a copy-and-paste approach to classify and locate defect patterns on wafer maps. The copy-and-paste approach involves copying pixels and labels from one image and pasting them onto another to address class-imbalance issues and create more images with defect patterns. Another approach that addresses the class-imbalance issue is proposed by Dasari et al. [5] in which a cluster-based adaptive data augmentation algorithm is used to predict defects in additive manufacturing. Another study considering additive manufacturing from Li et al. [19] used a combination of supervised and unsupervised learning techniques for data augmentation for the quality analysis of metal surfaces. Another analysis of metal surface defects is outlined by Jain et al. [14] testing three different generative adversarial network architectures. Furthermore, Yun et al. [36] use general adversarial networks to increase the amount of data for classifying defects on metal surfaces.

It can be concluded that there is a high demand for deep-learning image classification approaches in numerous application scenarios in manufacturing. However, one-shot and few-shot learning for part classification are still poorly investigated. The availability of training data in the right amount and quality and its impact on the explainability of the model's results remain the major challenge. Therefore, this work focuses on an approach to generating highly realistic synthetic images based on photorealistic rendered CAD models (see Fig. 1) to enhance the amount of data provided for training. Accordingly, this approach highly reduces the manual effort in data generation and preparation for part classification in manufacturing while maintaining a higher explainability than classical image augmentation approaches.

(a) Real Part (b) Rendered Part

Fig. 1 Exemplary depiction of a real image (a) and its photo-realistically rendered counterpart (b)

3 Parametric renderings from CAD

The work of Deorr et al. [9] already pointed out the benefit of using multiple views for classification tasks in manufacturing. Increasing the views provided for the deep neural network architectures results in a limited model accuracy increase. Most data augmentation techniques, as outlined in Section 2, significantly impact the generalization and accuracy of the models. However, the causing effect of this increase is hardly explainable and only empirically verified. Consequently, the models need more than just geometry information if we want to increase their generalization capabilities.

In the presented approach, we used photorealistic renderings based on CAD model geometry output to enhance the database for image classification tasks. This way, we achieve a more explainable data augmentation, as the pipeline is in full user control. Therefore, we designed a parameterized rendering pipeline, which was then implemented and executed using Blender. The general concept of our parameterized rendering pipeline is depicted in Fig. 2.

First, the CAD model is converted from the STEP format, generally applied in manufacturing, to the OBJ format, which is suitable for computer graphic applications. This way, we preserve all geometric properties including the definition of different materials, which are needed for photorealistic rendering while omitting all overhead data regarding the manufacturing process, such as tolerances our surface roughness. Afterward, we set up several realistic environments by using high dynamic range images (HDRI) from different locations of a factory (see Fig. 2). These images are then used as background projections for a 360° scene. The HDRI images are created by stitching together multiple high-resolution images in a 360° view using Google Earth's Streetview API. This way, we achieve a realistic background for not only our rendered images, which are visible in reflections on the rendered parts but also an environmental lighting setting. We expect that this step further enhances the learning process in contrast to scenes without background or environmental information. It is a step towards avoiding overfitting in the training process, as it often occurs when using synthetic data. Adding multiple background settings further enables the models to learn the separation of fore- and background. Using the intregrated gradients methods [29], we can get a first impression of how well this separation is learned (see Fig. 3). The idea behind the method is to calculate the contribution of pixel x to the prediction of the model's output class by adding up the local gradients at that pixel and assigning it a score based on its importance. Now, we can visualize the feature importance of each pixel during the prediction process.

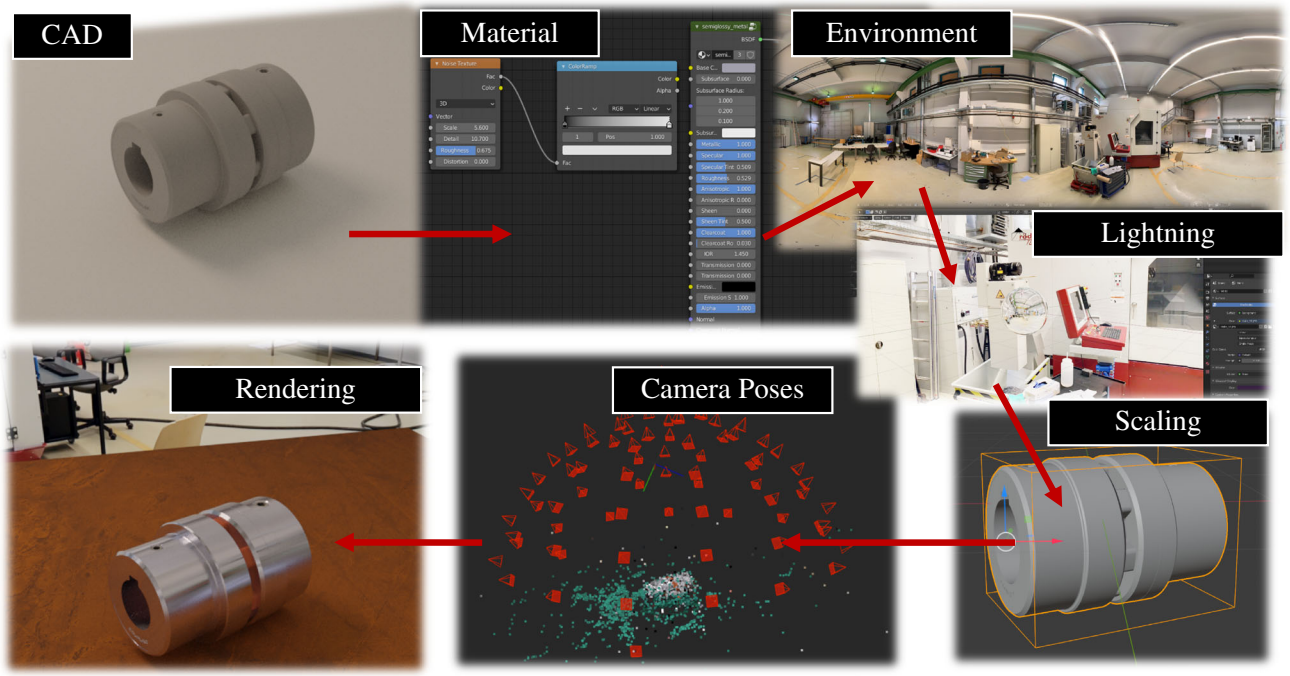


Fig. 2 Conversion Pipeline. Starting with the CAD files for an object, we can synthesize photorealistic renderings of the part in a generic way, such that we can augment the training data significantly and also allow for zero training data in terms of real images from the manufactured part

In the next step, we modeled a standard ground plate with a good contrast color and a slightly used texture to avoid aliasing effects. Every object gets placed onto this plate and scaled into a normalized bounding box. While this is mainly a design choice, in realistic environments, we assume that the inspected parts are relatively in the exact size dimensions, and if not, the camera is likely to be moved. This allows us to keep the camera positions fixed for every object in a trade-off for the actual relative object size information but enables a more controlled experimental setup. For general-purpose use of our proposed method, the camera sphere radius should be adapted instead of scaling the objects if the respective target part sizes differ significantly. We assume this helps separate the region of interest in the image from the background noise, which should not be learned. For the dynamic lighting, we choose a matrix grid of light sources parallel to the ground plate for lighting. Consequently, the object lies between the light source and the ground plate; thus, shadows always point downward on the plate. This results in controlled and minimal shadows in the image, which further helps the learning process. Next, we must model the material

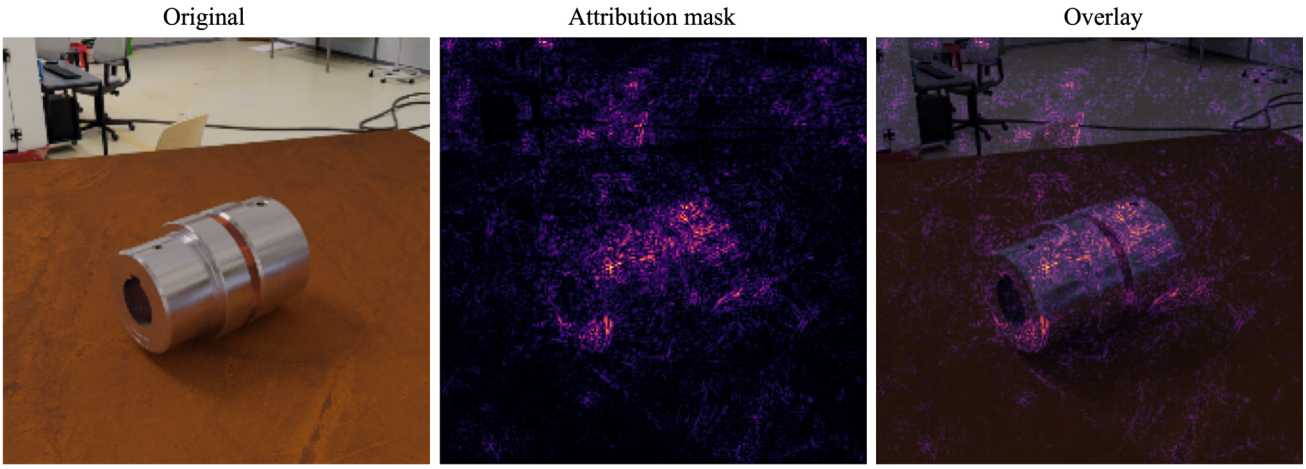


Fig. 3 Integrated gradient of a bearing image with and without background. The method highlights the pixels based on feature importance during the prediction process. Here, we use it to evaluate how well the model learned the separation of fore- and background

properties by editing the rendering graph. We found this part the most time-consuming, but fortunately, a wide range of commercial material databases can be used to counter this. Two typical materials were considered in the exemplary application: shiny metal and dull black rubber. Finally, we set up an ico-sphere (Fig. 2) for our camera positions where the images are rendered afterward. With the camera object, we could specify to always point to the center of our bounding box and set the dimensions to our liking. In consequence, the object is completely visible in all images taken. With this setting, we keep the near and far planes constant; thus, the number of voxels passed by each ray in the rendering step is constant. This further allows us to create equally sampled depth images of all objects, which can be used in other applications. The proposed pipeline is part of a software demonstrator, which is openly available (see the Code availability section for more information and a demo application).

4 Experimental design

We evaluate our proposed augmentation technique in our experimental design compared to no augmentation and state-of-the-art image data augmentation approaches. Furthermore, we focus on applying transfer learning to allow for training with minor data and higher generalization capability. We opt out of an extensive model hyperparameter tuning and instead perform a grid search on a subset of the models using only the plain rendered CAD images. Based on the results from this preliminary study, we set the hyperparameters for our study as depicted in Table 1. For testing, we chose 30 parts from the DMU-Net dataset [7] with varying complexity but with a primary focus on mechanical parts. The publicly available database covers a wide range of typical manufacturing parts. The most commonly applied ML-model architecture types, which we will also investigate in our study include *ResNet, VGG, EfficientNet, DenseNet, Inception, Xception,* and *NASNet*. This already covers a wide range of architectures and allows for a good comparison of the applicability of those for mechanical part classification. VGG has the classical CNN architecture consisting of convolution, max pooling, and fully connected layers and was created based on the analysis of how to increase the depth of the first CNNs [27]. In contrast, Inception's architecture is wider than the first CNNs, performing convolution with different sizes of filters and concatenating them afterward [30]. Increasing a CNN's depth and width are two possible ways to scale the network size. Another one is the increase in image resolution. Balancing these three ways of scaling a CNN led to the creation of EfficientNet, whose architecture is defined by a ratio between depth, width, and image resolution [31]. The topology of ResNet introduces residual learning by adding skipped connections or shortcuts to bypass inputs from earlier convolutions to later convolutions [12]. Combining skipped connections from ResNet with depthwise separable convolution, which perform channel-wise spatial convolutions, led to the creation of the Xception architecture [4]. DenseNet architecture increases the number of connections between the different layers of CNNs. Whereas traditional CNN has connections between a layer and its subsequent layer, DenseNet architecture connects each layer to every other layer in a feed-forward way [13]. The beforehand mentioned nets require significant architecture engineering. In contrast, the architecture of NASNet is not predefined but instead determined by a Neural Architecture Search (NAS), in which a reinforcement learning search method is used to optimize the architecture configurations of the CNN. Different forms exist for some approaches, which mainly differ in the number of layers and, thus, the number of trainable parameters. These model architectures come with pre-trained weights from the ImageNet dataset. In general, the ImageNet dataset mainly consists of everyday objects, and therefore, we expect a low bias in the pre-trained weights for the chosen classes. The head layer of those architectures maps the trained features to the respective classes of the ImageNet dataset. In our experiment, we only want to exploit the already learned features and therefore cut off the last layer and replace it with a small transfer model as depicted in Fig. 4. We start with an average 2D pooling with a window size of $(7, 7)$. This way, all the features learned until the last convolutional layer are equally sized. A flattening transformation follows the pooling to reduce the feature dimension from 2D to 1D. Thus, we can add a fully connected dense layer with $width \times height$ number of neurons. The idea is to allow a trainable parameter from every pixel in our training image to the previously learned features. Afterward, we only keep the weights

Table 1 Hyperparameters used throughout the study. A full-factorial hyperparameter study is performed for each train dataset, resulting in 105 settings per model used

Basemodel (variations)	Optimizer	Learning rate	Random seeds
VGG (16,19)	Adam	256e-1	42
ResNet (50,101,152)	SGD	45e-2	38
InceptionV3	RMSProp	1e-2	16
InceptionResNet		5e-3	72
Xception		1e-3	3
DenseNet (169,201)		5e-4	
NASNet		1e-4	
EfficientNet (B0-B4)			

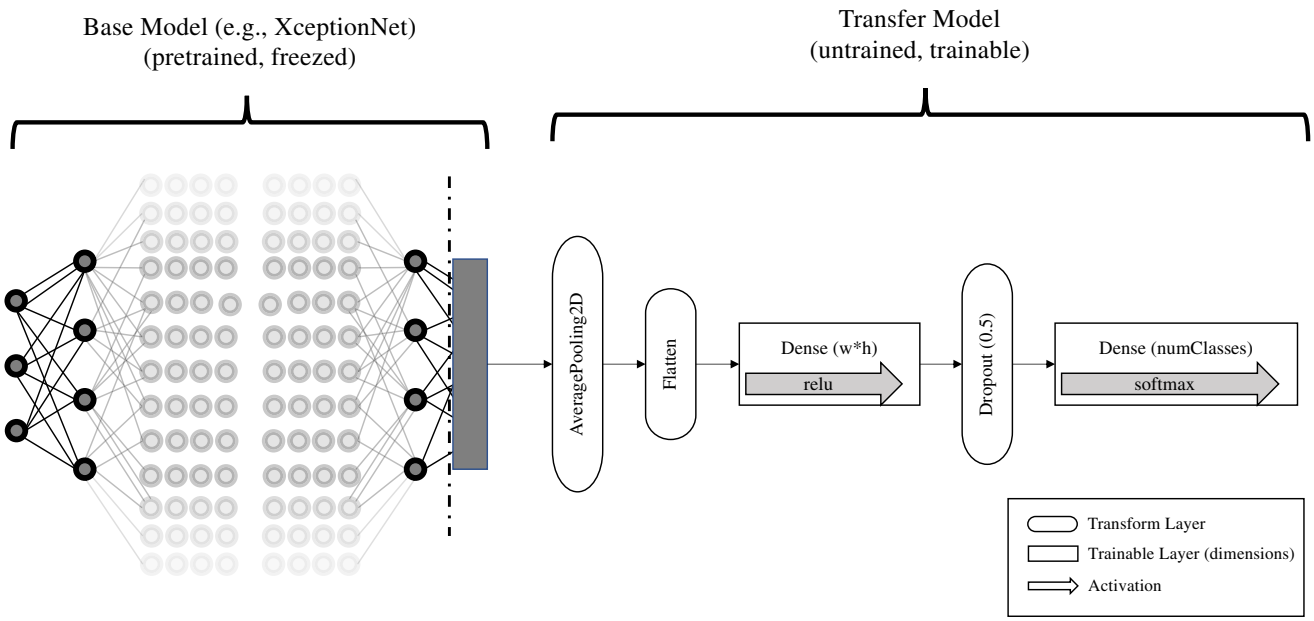


Fig. 4 Schematic depiction of the transfer learning model. From the pre-trained base models, the dense head layers are replaced with the transfer model. During training, the weights in the base model are fixed, and only the transfer model is trained with the new data

higher than 0.5 and drop the rest before we finally add the layer which maps to our newly trained target classes. For all architectures we use the standard resolutions of the respective models (between (255, 255) and (600, 600)). In addition to the variation of model architectures, we define five data settings with varying input image realism and

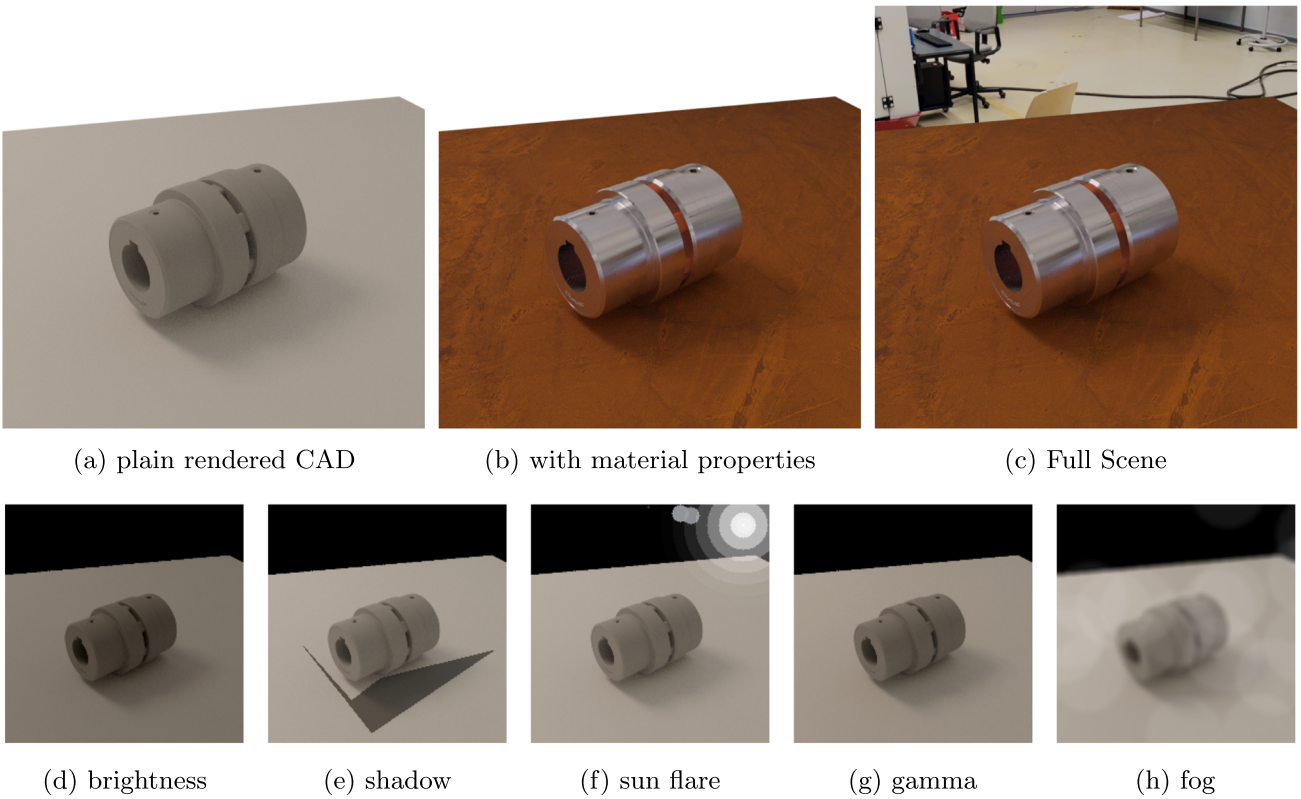


Fig. 5 Exemplary depiction of the different augmentation stages. Top row: our augmentation settings based on different rendering settings. Bottom row: standard image augmentation techniques using the albumentations framework [2]

Table 2 Experimental Setup. Besides the eight different model architectures with varying numbers of trainable parameters, four augmentation settings were investigated. In *"plain rendered CAD"*, the objects are simply rendered with rudimentary shadows and no background. In *"with material properties"*, a realistic light source and material properties are added. In *"full scene"*, the object is rendered regarding a complete environmental setting

Augmentation settings					# training images per class	
Experiment block	Plain rendered CAD	With material properties	Full scene	Albumentation augmentation	Full	Scaling input size
(1)	x				92	[9,18,36,55]
(1*)	x			x	184	[18,36,72,110]
(2)		x			92	[9,18,36,55]
(3)			x		92	[9,18,36,55]
(1)+(2)	x	x			276	[27,54,108,165]
(1)+(2)+(3)	x	x	x		184	[18,36,72,110]

further vary the number of input images by combining different sources. Figure 5 shows an example of the different data settings used. We form a baseline for comparing data augmentation techniques using 74 images/views from the pool of "plain rendered CAD" per class, which only contains the geometry of the underlying part. Based on these 74 images we use the *Albumentation API* [2] as the current standard tool for data augmentation. We decided upon five augmentation methods with random parameters, which are applied to each image with a probability of $p = 0.2$. Brightness and contrast are typical image parameters that often vary in the image collection process, and thus a model should generalize well for these modifications. Additionally, we applied random shadows, sun flares (bright illumination spots), gamma (similar to brightness), and fog to simulate a dirty lens (see Fig. 5 for an example of those modifications). For all settings (depicted in Table 2), we perform a k-fold cross-validation, which means that we randomly generate $k = 5$ sets of the training images with a $(0.8, 0.2)$ train-test split. We also performed the training with $r = 5$ random seeds to avoid lucky initial weight sets. Finally, we conduct a scaling experiment in which each class is only trained with 9 to 55 images, respectively (see Table 2).

The performance of the classifiers is measured with respect to precision, recall, and f1-score with a fixed validation dataset. The validation dataset consists of newly generated high-resolution renderings in a different environment (lightning and background) and from different camera positions than the training data.

$$precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{1}$$

$$recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2}$$

$$f1\text{-score} = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

Depending on the application, one or the other metric is more important, but both always have to be considered. Improving precision avoids overestimation, while recall monitors the model's underestimation. The f1-score is, therefore, a good overview measure to combine the two metrics. However, it shows weaknesses when monitoring improvement trends, while performing model optimization. In our analysis, we did not spot any anomalies when comparing the precision and recall; thus, we only report the f1-score in the results section.

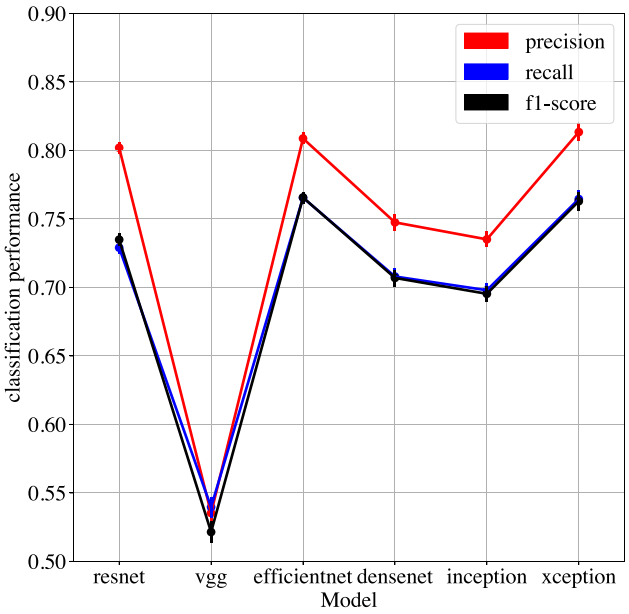
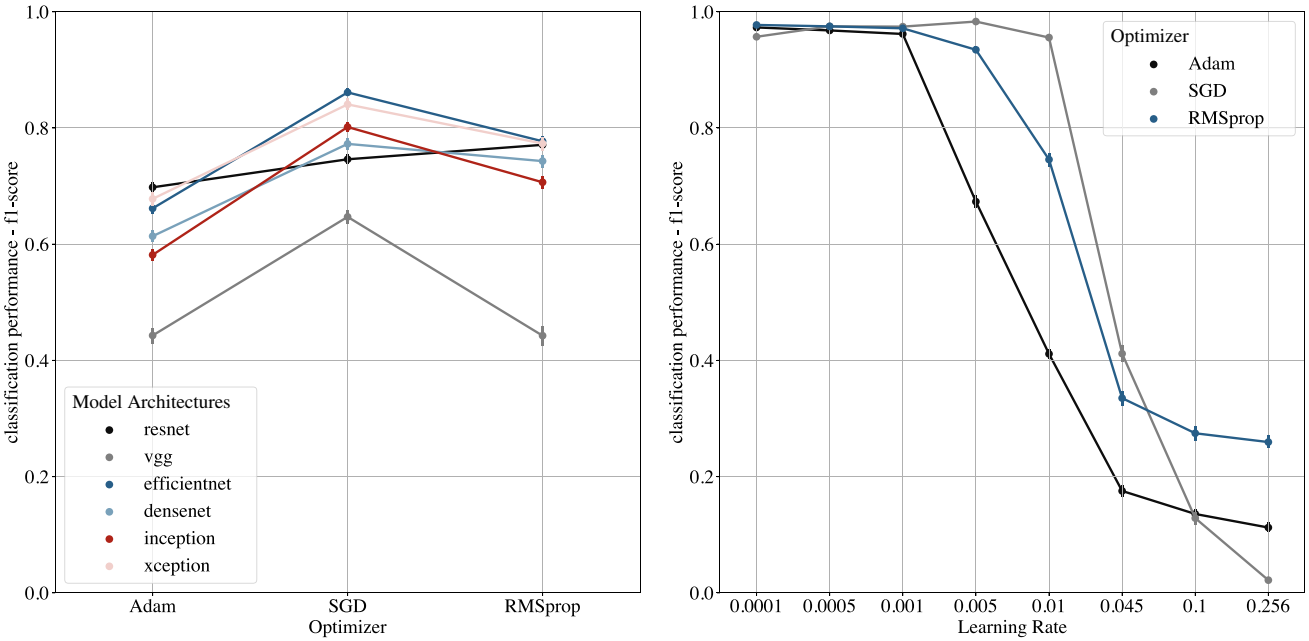


Fig. 6 Summarized classification scores for 30 classes grouped by model architecture for the complete hyperparameter set

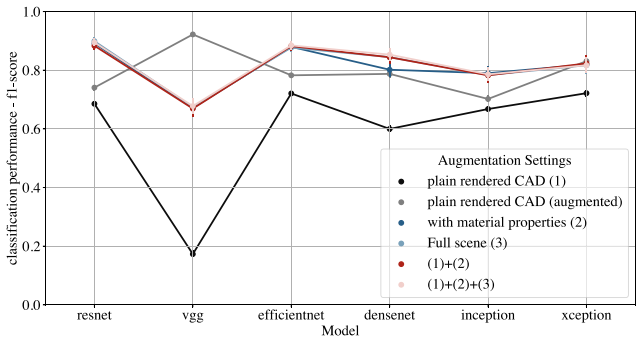


(a) Optimizer test (b) Learning rate range test

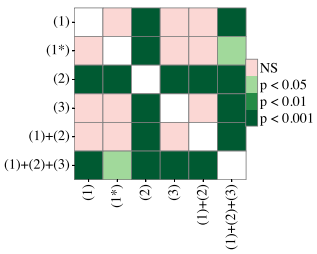
Fig. 7 Results from the hyperparameter study as part of the preliminary study. The NASNet model architecture is excluded due to poor results. For the further augmentation setting comparison, we chose the SGD optimizer with a learning rate of $lr = 0.01$

5 Results

The computation demand for training highly depends on the used hardware. We used an NVIDIA Tesla V100 GPU for the training in our experiment. The average training time per epoch varies between 4.56 and 68.11 s. We found out that the loss for all models is converging after 25 training epochs. Furthermore, our models tend to underestimate as the recall is significantly lower than the precision (Fig. 6). As this difference remains constant throughout all model architectures, we focus on reporting just the f1-score in our further analysis. Our hyperparameter study outlined in Fig. 7 reveals the best results throughout all model architectures while using the stochastic gradient descent (SGD) optimizer and a learning rate $lr < 0.01$. The fastest model architectures are the *inception-v3* and *VGG19* with under 6 s per epoch. The slowest model by far was the *DenseNet201* with 68.11 s on average. The number of trainable parameters lies between 600,000 and 900,000, while the VGG models have significantly less (229,000) and the *NASNet* significantly more (1,800,000). We could not spot a direct correlation between trainable parameters and training duration. Figure 8 and Table 3 shows the classification performance based on the f1-score summarized for the different model architectures. All models using the plain rendered CAD perform lower than the augmented versions, while the number of images remains constant. There is only a slight performance increase between the augmentation setting "with material properties" and "full Scene". Combining the augmented images can achieve an f1-score of up to 0.89 (precision 0.86, recall 0.91) with the *resnet* model architecture. The



(a) Performance scores based on augmentation setting



(b) Results from Conover's posthoc test.

Fig. 8 Results from the augmentation setting study. We observe a clear benefit for any of the used augmentation settings. In most cases, our proposed augmentation approaches (2,3) led to higher performance than the classical augmentation approach (1 augmented). The most significant increase is already achieved through the added material properties

Table 3 Statistical results summary for the classification study with different model architectures and augmentation settings. We can observe an increase in the mean classification performance based on the f1-score for our proposed augmentation technique (2,3) and a lower standard deviation leading to more robust models

Model group	Augmentation setting	Mean	Std	Q25%	Q50%	Q75%
densenet	plain rendered CAD (1)	0.5998	0.4615	0.0000	0.9474	1.0000
	plain rendered CAD (augmented)	0.7874	0.3861	0.9143	1.0000	1.0000
	with material properties (2)	0.8016	0.3654	0.8873	1.0000	1.0000
	full scene (3)	0.8446	0.3323	0.9474	1.0000	1.0000
	(1) + (2)	0.8441	0.3320	0.9474	1.0000	1.0000
	(1) + (2) + (3)	0.8532	0.3221	0.9474	1.0000	1.0000
efficientnet	plain rendered CAD (1)	0.7208	0.4093	0.3478	0.9730	1.0000
	plain rendered CAD (augmented)	0.7826	0.3766	0.8485	1.0000	1.0000
	with material properties (2)	0.8796	0.2774	0.9444	1.0000	1.0000
	full scene (3)	0.8771	0.2837	0.9444	1.0000	1.0000
	(1) + (2)	0.8821	0.2748	0.9444	1.0000	1.0000
	(1) + (2) + (3)	0.8846	0.2696	0.9444	1.0000	1.0000
inception	plain rendered CAD (1)	0.6677	0.4347	0.0000	0.9500	1.0000
	plain rendered CAD (augmented)	0.7019	0.4247	0.1000	0.9730	1.0000
	with material properties (2)	0.7898	0.3681	0.8824	0.9744	1.0000
	full scene (3)	0.7831	0.3712	0.8485	0.9730	1.0000
	(1) + (2)	0.7825	0.3746	0.8571	0.9737	1.0000
	(1) + (2) + (3)	0.7859	0.3722	0.8824	0.9744	1.0000
resnet	plain rendered CAD (1)	0.6854	0.4156	0.1905	0.9474	1.0000
	plain rendered CAD (augmented)	0.7407	0.3941	0.5926	0.9744	1.0000
	with material properties (2)	0.8914	0.2285	0.9231	0.9744	1.0000
	full scene (3)	0.8990	0.2145	0.9189	0.9744	1.0000
	(1) + (2)	0.8833	0.2396	0.9189	0.9744	1.0000
	(1) + (2) + (3)	0.8946	0.2235	0.9143	0.9744	1.0000
vgg	plain rendered CAD (1)	0.1731	0.3544	0.0000	0.0000	0.0000
	plain rendered CAD (augmented)	0.9220	0.1834	0.9268	1.0000	1.0000
	with material properties (2)	0.6712	0.4226	0.0642	0.9268	1.0000
	full scene (3)	0.6742	0.4246	0.0000	0.9444	1.0000
	(1) + (2)	0.6691	0.4240	0.0000	0.9268	1.0000
	(1) + (2) + (3)	0.6777	0.4196	0.0691	0.9444	1.0000
xception	plain rendered CAD (1)	0.7216	0.3928	0.4800	0.9474	1.0000
	plain rendered CAD (augmented)	0.8309	0.3163	0.8837	0.9730	1.0000
	with material properties (2)	0.8195	0.3131	0.8485	0.9730	1.0000
	full scene (3)	0.8170	0.3148	0.8426	0.9730	1.0000
	(1) + (2)	0.8232	0.3159	0.8636	0.9730	1.0000
	(1) + (2) + (3)	0.8169	0.3164	0.8439	0.9730	1.0000

EfficientNet and *resnet* architectures are performing best, while the *VGG* is performing the worst but still reaching an average f1-score of 0.67 if all data sources are used for training. We opted out of the *NASnet* model architectures as they only achieved a score of 0.27 and are not well suited for transfer learning in this application scenario. In contrast to all other investigated model architectures, the *VGG* model profits the most from applying traditional augmentation techniques. The Conover's posthoc analysis reveals a significant improvement for the augmentation setting *with material properties(2)* and when combining all data sources (1)+(2)+(3). In setting (2), the number of training images is substantially lower. The combined setting also triples the number of training images, and thus we expect a significant increase in classification performance. In general, we can observe three major trends:

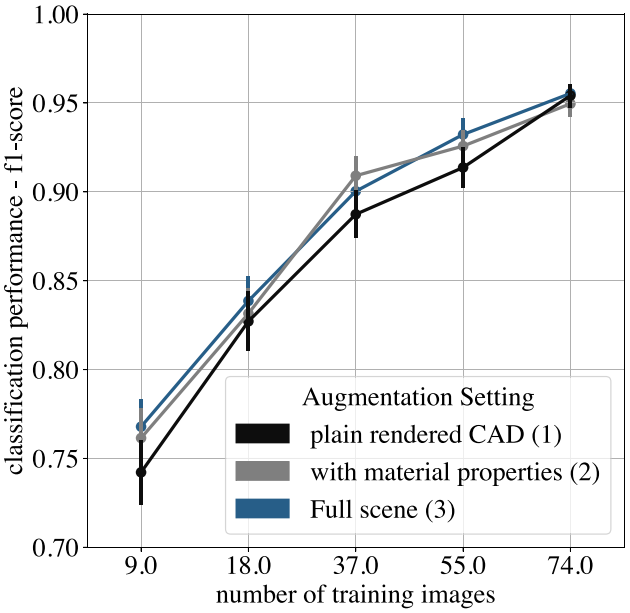


Fig. 9 Summarized f-1 classification scores for 30 classes grouped by model architecture based on the number of training images. With an increasing number of images used for training, the augmentation effect is decreasing

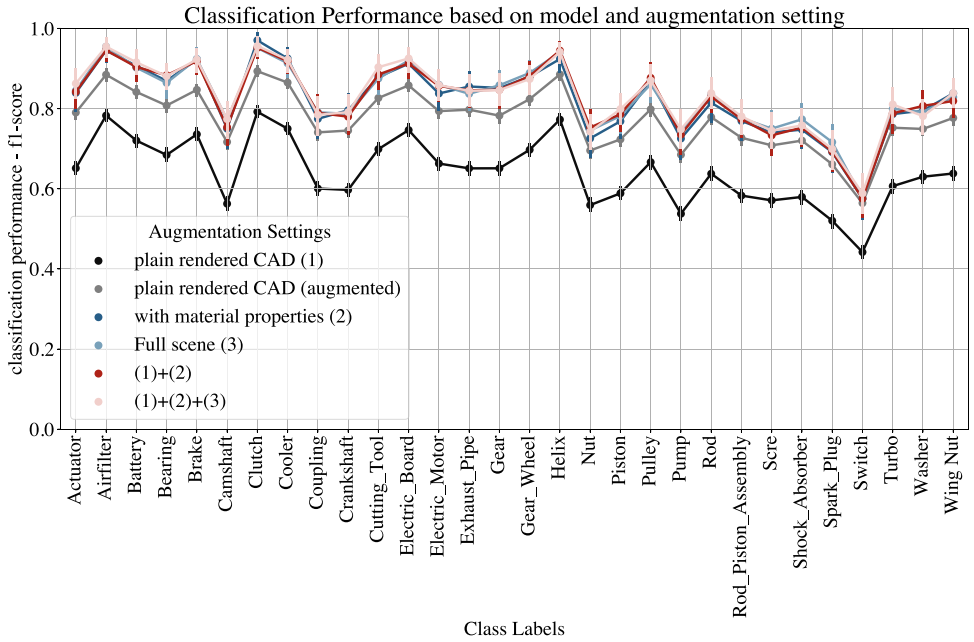
1. Choosing a proper learning rate is crucial before comparing different model architectures, but it highly influences the training time
2. Increasing the scene's realism enhances the classification performance, with the biggest increase through adding proper material properties.
3. Combining different realism settings and increasing the overall training data improves the classification performance.
4. The pre-trained models on the ImageNet data are also suitable for part classification throughout all model architectures.

In addition to our main experiment, we tested the classification performance in scaled training size setting, with only 9 to 55 images (Fig. 9) and a fixed learning rate ($lr = 0.01$) and optimizer (SGD). Here, we observe a clear performance increase with more training images. The effects from our proposed augmentation setting are higher when using a smaller amount of images per class, making it even more valuable in settings with strict memory or time restrictions (e.g., when implemented on EDGE devices). Interestingly with only nine images per class, we already achieve a classification performance of over 0.74, which is a great result compared to previous results [18].

6 Discussion

Based on the results, we can conclude that simple scene enhancement, such as valuable light sources for shading and basic material properties, significantly improves the training dataset's entropy (Fig. 8b) and classification performance. Still, the initial effort for setting up the rendering pipeline is manageable. Most time-consuming tasks like capturing the environment and modeling the material properties are one time-tasks and can be reused in multiple settings. The rendering time of those images can also be seen as negligible with the availability of modern graphics hardware. Furthermore, we could observe that combining different stages and sets of rendered objects could increase the classification performance. Note that the combined

Fig. 10 Summarized f-1 classification scores for each of the 30 classes using the full set of training images. With our proposed pipeline, we could significantly increase the classification performance of difficult parts, e.g., "nut" and "switch" compared to the plain rendered CAD setting



settings have two to three times more training data available, but this only seems beneficial when the training data size exceeds a certain threshold. We expect that the scalability of this augmentation technique is limited and should be investigated further to evaluate the practical relevance for high-quality demands with acceptable error rates below 1%. Taking a deeper look at the classification performance for the different classes (Fig. 10), we could observe that delicate parts or parts with a significantly higher dimension than the others, e.g., switches, are poorer classified (f1-score between 0.45 and 0.6). It seems that the 2D-convolution filters used in every of the investigated model architectures struggle to generate feature maps that capture such objects. The pre-trained weights may provide another explanation as they were primarily generated from everyday objects and landscapes, so poorly classified shapes may be underrepresented in the initial training data. In Fig. 10, we evaluate the classification performance for the respective classes (sample parts). We can observe that the general tendency between the different data settings is preserved in the detailed view. For some of the more difficult classes, e.g., "switch," we achieved an increase in the f1-score of 67% or for the "nut" of 82%. It seems that not only the shape and geometry of the respective parts play a crucial role, but the distinction from the rest, which we achieve with increased realism.

7 Conclusion

Image classification based on deep learning shows applicability in numerous manufacturing areas but is hampered by the often challenging collection of image data. The results in this paper show that with modern graphics hardware and easy access to computer graphics modeling tools, the effort for generating photorealistic images of parts from CAD data is manageable and dominated mainly by one-time tasks. Our method can train an image-based part classifier without prior data collection. Adding more realism to our part images could increase the classification performance independently from the used model architecture. Additionally, the effect is even more dominant for smaller training sample sizes. Existing augmentation approaches use image transformations with random parameters or highly specialized generator networks. Both cases lack substantial explainability in their effect on classification performance. While our approach even led to better results in many settings, it has increased explainability due to its user-friendly and strong visual support. We already pointed out that modeling the material properties is the most challenging and less automated part of our proposed pipeline, which can be solved by incorporating data from the PLM system. Therefore, it could be worth investigating the effect of different material properties and their difference from the real ones in future experiments. So far, we strongly exploit the advantages of deep learning techniques while neutralizing their main disadvantage, the need for huge training data via transfer learning and using computer graphics to augment the training database with photorealistic image data. Using those concepts allows us to train and deploy an image-based part classification system before producing any physical part specifically suitable for small-batch manufacturing. Furthermore, our system was successfully tested in a manufacturing environment of a mid-sized company. The fast training and image generation times will enable us to quickly incorporate new parts in the classification model within a couple of hours. In summary, we could demonstrate that our augmentation approach uses less data, is more explainable, and needs fewer hyperparameter tuning than recent image augmentation approaches. At the same time, the most prominent effect is achieved with the combination of augmentation techniques.

Funding Open Access funding enabled and organized by Projekt DEAL. This research was funded by the European Union (EU) from the regional development fund (EFRE) and the Commercial Vehicle Cluster South-West (P1-SZ2-3 Project: Nutzung von Künstlicher Intelligenz in der Nutzfahrzeugproduktion).

Data availability The data used in this study is available from the DMU-NET webpage: https://www.dmu-net.org. A subset of the study data is available in the git repository. Please contact the authors for the complete image data used in training and testing.

Code availability The code is publicly available through github.
The code for the data synthesize pipeline is hosted at: https://github.com/PatRuediger/imdatasynth.git
The code for the model deployment and augmentation study is hosted at: https://github.com/PatRuediger/cad_data_aug_study.

Declarations

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication Not applicable

Conflict of interest The authors declare no competing interests.

References

1. Blondheim D (2022) Improving manufacturing applications of machine learning by understanding defect classification and the critical error threshold. Int J Met 16(2):502–520. https://doi.org/10.1007/s40962-021-00637-0
2. Buslaev A, Iglovikov VI, Khvedchenya E et al (2020) Albumentations: fast and flexible image augmentations. Information 11(2). https://doi.org/10.3390/info11020125
3. Chiu MC, Chen TM (2021) Applying data augmentation and mask r-cnn-based instance segmentation method for mixed-type wafer maps defect patterns classification. IEEE Trans Semicond Manuf 34(4):455–463
4. Chollet F (2017) Xception: deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
5. Dasari SK, Cheddad A, Palmquist J et al (2022) Clustering-based adaptive data augmentation for class-imbalance in machine learning (cada): additive manufacturing use case. Neural Comput & Applic 1–14
6. Dean J, Patterson D, Young C (2018) A new golden age in computer architecture: empowering the machine-learning revolution. IEEE Micro 38(2):21–29. https://doi.org/10.1109/MM.2018.112130030
7. Dekhtiar J, Durupt A, Bricogne M et al (2018) Deep learning for big data applications in CAD and PLM – research review, opportunities and case study, vol 100, pp 227–243. https://doi.org/10.1016/j.compind.2018.04.005
8. Deshpande AM, Minai AA, Kumar M (2020) One-shot recognition of manufacturing defects in steel surfaces. Procedia Manufacturing 48:1064–1071. https://doi.org/10.1016/j.promfg.2020.05.146
9. Doerr K, Samarabandu J, Wang X (2014) Efficient object classification using multiple views in manufacturing environments. In: 7Th international conference on information and automation for sustainability. IEEE, Colombo, pp 1–5, https://doi.org/10.1109/ICIAFS.2014.7069597
10. Gatys LA, Ecker AS, Bethge M (2015) A neural algorithm of artistic style. arXiv:150806576
11. Goodfellow I, Pouget-Abadie J, Mirza M et al (2020) Generative adversarial networks. Commun ACM 63(11):139–144
12. He K, Zhang X, Ren S et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
13. Huang G, Liu Z, van der Maaten L et al (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
14. Jain S, Seth G, Paruthi A et al (2020) Synthetic data augmentation for surface defect detection and classification using deep learning. J Intell Manuf 1–14
15. Kang G, Dong X, Zheng L et al (2017) Patchshuffle regularization. arXiv:170707103
16. Khajezade M, Ramezankhani M, Fard FH et al (2021) Toward using few-shot learning for prediction of complex in-service defects of composite products: a case study. In: 2021 IEEE Canadian conference on electrical and computer engineering (CCECE). IEEE, Canada, pp 1–7, https://doi.org/10.1109/CCECE53047.
17. Krüger J, Lehr J, Schlüter M et al (2019) Deep learning for part identification based on inherent features. CIRP Ann 68(1):9–12. https://doi.org/10.1016/j.cirp.2019.04.095
18. Kusiak A (2020) Convolutional and generative adversarial neural networks in manufacturing. Int J Prod Res 58(5):1594–1604. https://doi.org/10.1080/00207543.2019.1662133
19. Li X, Jia X, Yang Q et al (2020a) Quality analysis in metal additive manufacturing with deep learning. J Intell Manuf 31(8):2003–2017
20. Li X, Zhang W, Ding Q et al (2020b) Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation. J Intell Manuf 31(2):433–452
21. Lyu P, Zhang H, Yu W et al (2022) A novel model-independent data augmentation method for fault diagnosis in smart manufacturing. Procedia CIRP 107:949–954
22. Meister S, Möller N, Stüve J et al (2021) Synthetic image data augmentation for fibre layup inspection processes: techniques to enhance the data set. J Intell Manuf 32(6):1767–1789
23. O' Mahony N, Campbell S, Carvalho A et al (2019) One-shot learning for custom identification tasks; a review. Procedia Manufacturing 38:186–193. https://doi.org/10.1016/j.promfg.2020.01.025
24. Riordan ADO, Toal D, Newe T et al (2019) Object recognition within smart manufacturing. Procedia Manufacturing 38:408–414. https://doi.org/10.1016/j.promfg.2020.01.052
25. Schlüter M, Niebuhr C, Lehr J et al (2018) Vision-based identification service for remanufacturing sorting. Procedia Manufacturing 21:384–391
26. Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. Journal of Big Data 6(1):1–48
27. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. arXiv: 1409.1556
28. Subakti H, Jiang JR (2018) Indoor augmented reality using deep learning for industry 4.0 smart factories. In: 2018 IEEE 42nd annual computer software and applications conference (COMPSAC). IEEE, Tokyo, pp 63–68. https://doi.org/10.1109/COMPSAC.2018.10204
29. Sundararajan M, Taly A, Yan Q (2017) Axiomatic attribution for deep networks. In: International conference on machine learning, PMLR, pp 3319–3328
30. Szegedy C, Liu W, Jia Y et al (2015) Going deeper with convolutions. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, Boston, pp 1–9. https://doi.org/10.1109/CVPR.2015.7298594
31. Tan M, Le QV (2020) EfficientNet: rethinking model scaling for convolutional neural networks. arXiv:1905.11946
32. Wang T, Yao Y, Chen Y et al (2018) Auto-sorting system toward smart factory based on deep learning for image segmentation. IEEE Sensors J 18(20):8493–8501. https://doi.org/10.1109/JSEN.2018.2866943
33. Weimer D, Scholz-Reiter B, Shpitalni M (2016) Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. CIRP Ann 65(1):417–420. https://doi.org/10.1016/j.cirp.2016.04.072
34. Wu J, Zhao Z, Sun C et al (2020) Few-shot transfer learning for intelligent fault diagnosis of machine. Measurement 166:108,202. https://doi.org/10.1016/j.measurement.2020.108202
35. Wuest T, Weimer D, Irgens C et al (2016) Machine learning in manufacturing: advantages, challenges, and applications. Production & Manufacturing Research 4(1):23–45. https://doi.org/10.1080/21693277.2016.1192517
36. Yun JP, Shin WC, Koo G et al (2020) Automated defect inspection system for metal surfaces based on deep learning and data augmentation. J Manuf Syst 55:317–324

37. Zhang A, Li S, Cui Y et al (2019) Limited data rolling bearing fault diagnosis with few-shot learning. IEEE Access 7:110, 895–110,904. https://doi.org/10.1109/ACCESS.2019.2934233
38. Zhong Z, Zheng L, Kang G et al (2020) Random erasing data augmentation. In: Proceedings of the AAAI conference on artificial intelligence, pp 13,001–13,008
39. Zhou X, Liang W, Shimizu S et al (2021) Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems. IEEE Trans Industr Inf 17(8):5790–5798. https://doi.org/10.1109/TII.2020.3047675