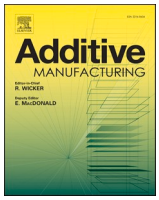
Additive Manufacturing 41 (2021) 101965

Contents lists available at ScienceDirect

Additive Manufacturing

journal homepage: www.elsevier.com/locate/addma



Research Paper

A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural networks

Erik Westphal [a,*], Hermann Seitz [a,b]

[a] Chair of Microfluidics, University of Rostock, 18059 Rostock, Germany
[b] Dept. Life, Light & Matter, University of Rostock, 18059 Rostock, Germany

ARTICLE INFO

Keywords:
Additive manufacturing
Selective laser sintering
Process monitoring
Machine learning
Convolutional neural network

ABSTRACT

Part defects and irregularities that influence the part quality is an especially large problem in additive manufacturing (AM) processes such as selective laser sintering (SLS). Destructive and non-destructive testing procedures are currently mostly used for quality control and defect detection of AM parts after production. In this context, machine learning (ML) algorithms are increasingly being used to enable computer-aided defect detection through automatic classification of manufacturing data. Convolutional neural networks (CNN) based on ML methods are widely used for this task. In this paper, complex transfer learning (TL) methods are presented, which enable the automatic classification of powder bed defects in the SLS process using very small datasets. The proposed methods use the VGG16 and the Xception CNN model with pretrained weights from the ImageNet dataset as initialization and an adapted classifier to classify good and defective image data recorded during part manufacturing. Known performance metrics were determined to evaluate and compare the performance of the models. The VGG16 model architecture achieved the best results for Accuracy (0.958), Precision (0.939), Recall (0.980), F1-Score (0.959) and AUC value (0.982). These results show the effectiveness of defect detection based on CNN and can offer an alternative method for non-destructive quality assurance and manufacturing documentation for additively manufactured parts.

1. Introduction

As one of the most popular additive manufacturing (AM) processes, selective laser sintering (SLS) is well suited for the production of individual, complex and topology-optimized parts for various industrial sectors. With the SLS process, powder particles are locally fused using a heat source (e. g. a laser). The laser sintering of a defined contour and the layer wise repetition of the sintering process then create a three-dimensional (3D) part.

The SLS process usually processes polyamides (PA) such as PA 11 and PA 12, polystyrene (PS), thermoplastic elastomers (TPE), polypropylene (PP) and certain polycarbonates (PC) or variations thereof [1]. The connection of the individual powder particles to one another takes place through thermal influence, fusion and subsequent solidification of the material. This process leads to high-quality part properties of the laser sintered structures that meet the requirements for functional components. According to Schmid [2], there are increased requirements inter alia for:

- a reproducible quality,
- process security and
- the automation of production processes.

In order to establish itself as a serious production process, quality controls and quality management must be developed, implemented and optimized for the entire SLS process chain in order to be able to compare the manufacturing process with other production techniques in terms of quality standards [2]. The quality of the parts manufactured by means of SLS is not only determined by the fusion of the powder particles of successive layers, but also by the integrity of the powder bed and the stability of the powder application [3]. A uniformly distributed powder bed, without irregularities, is desirable for a good part quality, see for example Fig. 1(a) and (c). However, various irregularities such as foreign bodies, part edges, powder accumulations and powder trenches, collectively referred to as powder bed defects, can occur in the powder bed, as shown in Fig. 1(b) and (d). These powder bed defects can lead to deficiencies in the part quality and the part properties up to quality-related rejects of the parts, which in turn results in considerable

* Corresponding author.
E-mail addresses: erik.westphal@uni-rostock.de (E. Westphal), hermann.seitz@uni-rostock.de (H. Seitz).

https://doi.org/10.1016/j.addma.2021.101965
Received 30 November 2020; Received in revised form 19 February 2021; Accepted 17 March 2021
Available online 22 March 2021

additional costs, material waste and ties up machine capacity [3]. This affects according to Xiao et al. [3] the widespread application of SLS as a profitable production technique in which constant quality, reproducibility and cost as well as waste reduction are critical. By detecting powder bed defects as early as possible during the production process, corrective measures can be initiated immediately, thereby ensuring quality as well as the reduction of costs, waste and capacity utilization [3].

The approach of the work presented here is to monitor the SLS powder bed for signs of defects using machine learning (ML) methods. For these complex methods, the recording of large amounts of image data is necessary, since a ML algorithm is trained using this data instead of being explicitly programmed [4]. ML methods are then a possibility to evaluate this amount of data almost in real time and to identify complex non-linear relationships in the datasets [5]. Conventional ML techniques are, according to Baumgartl et al. [5], divided into the three categories of supervised, semi-supervised and unsupervised learning.

These conventional ML techniques are only able to process unstructured data in its raw form (for example images, natural language etc.) to a limited extent [6]. Using modern state-of-the-art ML algorithms based on deep learning (DL), such raw data can also be processed and features extracted from it automatically [6,7]. For this process, convolutional neural networks (CNN) are almost always used in DL, because they are very effective in discovering complex structures in large amounts of data and can process unstructured data such as color images [4,6]. However, the key aspect of DL is, according to LeCun et al. [6] that a functional extractor is not designed by human developers, but is learned from data using a general learning process.

In this work complex ML algorithms based on deep learning and CNN were implemented. Process images were recorded as raw data during selective laser sintering using an inexpensive camera setup and preprocessed in a defined manner. The image data were then used to train two different CNN architectures for the classification of powder bed defects and thus to develop an automatic method for process assessment and documentation. Established evaluation metrics such as accuracy, precision, sensitivity and F1-Score were used to assess the performance of the CNN models. In addition, metrics such as the receiver operator characteristic (ROC) curve as well as the area under the curve (AUC) and heat maps were used to evaluate and visualize the results of the ML models.

2. Related work

ML methods have already been used in various AM processes and different AM procedures for error detection. Xiao et al. [3] have developed a CNN for the detection of three different types of powder bed defects during selective laser sintering. For this purpose, images of the powder bed were recorded with a digital camera and analyzed using a two-stage CNN. In the first module of the CNN, the location of the defect was identified and in the second module, error masks were generated for each specific location of the error. Compared to other methods, the accuracy of the error detection has been significantly improved by using the two-stage CNN model. However, the datasets have also been especially preconfigured to a considerable extent in order to simulate special error cases. Detailed statements on the accuracy of the two-stage CNN with real manufacturing datasets are not given.

Scime and Beuth [8] used grayscale areas from images of an integrated camera of a laser powder bed fusion (LPBF) machine to differentiate between metal-based powder bed irregularity classes. The classes were subsequently used to develop an ML algorithm for in-situ process monitoring and to analyze powder bed images. The algorithm described works in principle for the LPBF process, but has to be improved with regard to the classification accuracy.

Gobert et al. [9] used a high resolution digital single-lens reflex camera for layer-by-layer imaging in order to record images for a supervised learning of defects during a metal powder bed fusion (PBF) process. CT scans were then used to assess the results of ML detection. The resulting accuracies of the error detection algorithms during the manufacturing process reached values of up to 85%, but refer exclusively to the metal PBF process. In addition, a linear support vector machine (SVM) algorithm was used for error detection instead of a CNN.

Baumgartl et al. [5] used a combination of thermographic and off-axis in-situ imaging in a LPBF system. The images served as a data source for a DL-based CNN to identify printing errors. The model is well suited for thermographic in-situ defect detection in LPBF processes and achieves accuracies of over 96%. However, this CNN model can only detect spatter and delamination defects in metal LPBF. Other types of defects such as cracks, pores and unmelted powder, as well as additional processes such as SLS were not investigated.

In the review by Yadav et al. [10], further methods for automated in-situ process error monitoring and detection were listed and examined with regard to current progress in the field of process data analysis. Furthermore, the working principles of the most common in-situ sensor systems and commercially available in-situ monitoring solutions for metal LPBF systems were considered. The review article shows that the in-situ process recording is still at an early stage. Much of the research carried out actually focuses on the metal LPBF and the development of in-situ sensor systems to better understand this process. Detailed investigations into other PBF processes such as selective laser sintering with plastics as well as investigations for error detection in real time for both the SLS and the LPBF process still have to be investigated.

3. Materials and methods

In this section, the experimental setup used in this research is first described in detail. After that the dataset used in this paper is discussed. Thereby the generation, extraction and preprocessing of the data is described as well as some performance indicators used for the

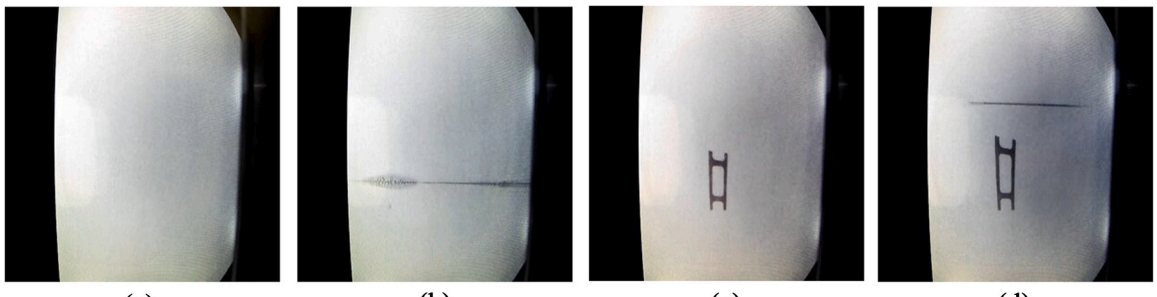


Fig. 1. Image examples from the dataset of a SLS print job with: (a) a powder bed without sintered elements and irregularities; (b) a powder bed without sintered elements with irregularities; (c) a powder bed with a sintered element without irregularities and (d) a powder bed with a sintered element and irregularities. Original images are available in a public repository (https://doi.org/10.17632/2yzjmp52fw.1). (The DOI of the dataset is reserved, but not active).

classification task plus evaluation and visualization techniques. Furthermore, CNN architectures are presented and the approaches and architectures used for this work are considered.

3.1. Experimental setup

The SLS process was monitored in real time with a high-resolution camera and recorded using a simple setup consisting of a high definition (HD) Universal Series Bus (USB) webcam and a single-board computer in order to obtain image data of the powder bed. An S2 laser sintering system (Sintratec AG, Brugg, Switzerland) was used for selective laser sintering. The laser sintering system has a 10 W diode laser with a laser wavelength of 1064 nm and a round powder bed shape with an effective printing area of 130 mm in circumference and a maximum build height of 360 mm. The build chamber can be heated up to 180 °C and does not require any operating gas during the entire printing process. The material used for printing is PA 12, which was printed with a layer thickness of 100 μm. The laser spot size was 145 μm and the scan speed 3 m/s.

The setup for recording the image data consisted of a simple Plusonic HD USB webcam (Allnet GmbH, Gemering, Germany) with a resolution of 3 megapixels and a 3.6 mm focal lens as well as a Raspberry Pi 3B+ (OKdo Technology Limited, London, UK). A 128 GB 3.0 Ultra USB stick (SanDisk Corp., Milpitas, US) was used to store the data and a 5.1 V with 2.5 A power adapter unit (TT Electronics IoT Solutions LTd., Woking, UK) was used for the power supply. In addition, a 7-inch Raspberry Pi touchscreen (OKdo Technology Limited, London, UK) was used for operational control and a standard camera tripod for positioning of the webcam. The camera was then mounted on the tripod and positioned as well as focused in front of the machine display of the laser sintering system. The high-resolution camera built into the S2 system was used to monitor the powder bed directly from above in real time and a live stream was shown on the system display, which in turn was recorded via the webcam. The recordings were made with a resolution of 640 × 480 pixels (px), whereby the recorded build area had physical dimensions of approx. 250 × 150 mm. The recordings were processed with self-written python programs. The open source programming language python, version 3.7.9 (python software foundation, Fredericksburg, US) was used for programming.

3.2. SLS powder bed dataset

The dataset used in this work was extracted from video recordings of the powder bed surface, which were recorded on an SLS printing system with a frame rate of two frames per second during manufacturing. In order to obtain individual images of the powder bed surface, the recorded Matroska (MKV) video files were broken down into individual frames without video compression and with a resolution of 640 × 480 px. Every 20th frame has been saved as a JPEG image and compressed or cropped to a size of 300 × 300 px to remove unnecessary information and image areas such as time stamps and image resolution specifications.

The dataset generated in this way comprised 9426 powder bed images. Extraneous images were then initially removed from the dataset (e.g. images with recording anomalies, images where the powder distribution unit of the system was also recorded, images with unfavorable light reflections). The remaining images were then manually divided into two different classes, OK and DEF, using process knowledge and additive expertise. Both classes together result in the adapted dataset with 8514 images. All images that showed a uniform powder bed surface without defects were classified into the OK class. This class contained 7808 images. All images on which an uneven powder bed surface with defects (cracks, ditches, foreign bodies etc.) could be seen were classified into the DEF class. As a result, 706 images were divided into the DEF class.

The dataset can be downloaded from the following address under a creative commons attribution 4.0 international license: https://doi.org/10.17632/2yzjmp52fw.1.

3.2.1. Class imbalance problem

In a binary classification problem with data from two classes, a class imbalance occurs when one class contains significantly fewer samples (minority class) than the other class (majority class) [11,12]. In the dataset used here, the two classes OK and DEF also have a different number of images. The imbalance ratio (IR) of the dataset can be calculated as follows:

$$IR = \frac{|majority\ class|}{|minority\ class|} \tag{1}$$

Liu et al. [13] examined various datasets related to the IR. The IR ranges from 1.7 to 24.3 and can, according to Wu et al. [14] also achieve values such as 10^6. The IR of this SLS powder bed dataset is:

$$IR = \frac{|OK|}{|DEF|} = \frac{7808}{706} = 11.06 \tag{2}$$

Compared to the considered datasets examined by Liu et al. [13], an IR of 11.06 is not uncommon, but it is already one of the more imbalanced datasets.

This data imbalance must always be considered in intelligent classification algorithms. When evaluating the classification results, standard metrics such as accuracy and error rate are used most often, but these are unsuitable for class imbalances since the result is dominated by the majority class [11,12]. In this way, a data distribution of 1% positive examples to 99% negative examples, the accuracy of a classification can be 99%, in that the classifier simply evaluates all examples as negative. Because of this, special metrics are required to evaluate imbalanced classification tasks. In addition, there are also various techniques that can be used to deal with imbalanced problems. These are described in detail in Section 3.2.2 below.

3.2.2. Techniques for class imbalanced data

The imbalance between two classes can be reduced by changing the data imbalance or by changing the underlying learning and decision-making process of the model to increase sensitivity to the minority class [11]. According to Johnson and Khoshgoftaar [11], the methods for dealing with class imbalances can accordingly be divided into techniques at the data level, methods at the algorithm level and hybrid approaches.

Data level techniques include oversampling, undersampling and modifying the data distributions to compensate for the level of imbalance [11]. The simplest forms of these methods are random undersampling (RUS), in which data is randomly removed from the majority class and random oversampling (ROS), in which data from the minority class is randomly duplicated [15]. In previous experiments it was found that RUS led to good results overall and often exceeded ROS [11]. Methods at the algorithm level do not change the data distribution. Instead, according to Johnson and Khoshgoftaar [11], either the learning process is adapted so that the importance of the minority class increases, or the decision threshold is shifted so that the tendency towards the majority class is reduced. Hybrid approaches combine data and algorithm methods in different ways. For example, one approach involves first data sampling to reduce the imbalance and then applying a shift in the decision threshold to reduce the influence of the majority class [11].

In this work a combination of RUS and ROS method was used. RUS was used to randomly remove data from the OK class and ROS to randomly duplicate data from the DEF class. As a result, 5808 images were removed from the OK class and 1294 images were added to the DEF class by duplicating the existing image data, thus balancing the OK and DEF classes. The dataset ultimately used for this work thus contains a total of 4000 images.

3.2.3. Image preprocessing and data structure

The images of the SLS powder bed dataset were created using a simple preprocessing procedure according to Pasa et al. [16] processed further, in order to obtain only the especially interesting image area with as much relevant information as possible. The preprocessing steps performed here are as follows:

- Remove any black stripes from the edges of the images.
- Resize the image so that the smaller edge (in this case) is 180 px long.
- Extraction of a centered square image area of 180 × 180 px.

These steps were carried out automatically by a simple, self-programmed python script. An example of the preprocessing procedure and a preprocessed image is shown in Fig. 2. With this procedure, an attempt was made to use only relevant information in the image and to only supply the CNN with pixels of interest (preferably no black borders, since no useful information for the intended classification task can be extracted from it and the calculation times take longer). It should be noted here that the square printing area (or the usable powder bed) of the laser sintering system is displayed vertically distorted as a result of the image recording (Fig. 2(a), red dashed border) and therefore could not be completely covered by a square image area (Fig. 2(a), blue border) without including undesired black border areas in the image recording. It is important to understand that square images are more beneficial for the ML model architectures. Fig. 2(b) thus shows an optimized square image of the powder bed printing area of the laser sintering system.

The dataset balanced with the RUS and ROS methods consists of 4000 images, which are automatically divided into 2000 different OK and 2000 partially different and partially duplicated DEF images using a simple, self-programmed python script. The data structure of this dataset was further subdivided according to [17] by creating three subgroups, each with separate directories for both classes: a training dataset with 1000 automatically selected images in each class, a validation dataset with 500 images in each class and a test dataset with also 500 images in each class. The 500 OK images each for the validation and test dataset were again randomly and automatically selected by a self-programmed python script from the 2000 OK images and moved to the appropriate directory. A slightly different method was used for the 500 DEF images of the validation and test dataset, as is must be ensured that no image appears twice in the individual subgroups and subdirectories, otherwise the results of the classification could be distorted. For this reason, the original 706 individual DEF images were randomly and automatically divided by a python script and accordingly 306 images were assigned to the DEF directory of the training dataset and 200 DEF images each to the DEF directories of the validation and test dataset. Another python script randomly selected DEF images in the respective DEF directories were then automatically copied until the directories contained the defined size of 1000 DEF training images and 500 DEF validation and test images each. Fig. 3 shows a general flow chart for creating the data structure described.

The preprocessed image data from the individual subgroups and directories of the dataset were then checked again manually for correct assignment. Incorrectly assigned images (which were overlooked during the initial assignment or for which a classification was unclear) were removed manually and replaced with other images that were randomly selected by an additional python script according to the respective directory.

3.2.4. Data augmentation and hyperparameter tuning

Data augmentation was carried out as a regulatory mechanism in order to avoid an immediate overfitting of the model to the training data. This would have a negative impact on the model performance on newly viewed data. With data augmentation, various operations are performed on the training dataset in real time. These operations are:

- Rescaling factor of 1/255 for normalizing the image data
- Horizontal flip of the images
- Zoom range of image area 0.15
- Width shift 0.20
- Height shift 0.20
- Shear rate 0.15
- Fill mode “nearest”
- Random image rotation of ± 20 degrees

The principles and operations of data augmentation are described in detail by Shorten and Khoshgoftaar [18] and also in the Keras library [19]. In addition to the preprocessing of the image data and data augmentation, the settings of the hyperparameters were an essential part of training the CNN models. The best model for classification of the powder bed images was only achieved after various iterations and configurations of the hyperparameters. The hyperparameters of the CNN models used in this work are listed in Table 1 and are explained in detail by Hutter et al. [20].

3.3. Convolutional neural network

Classic ML approaches to image recognition consist of two separate

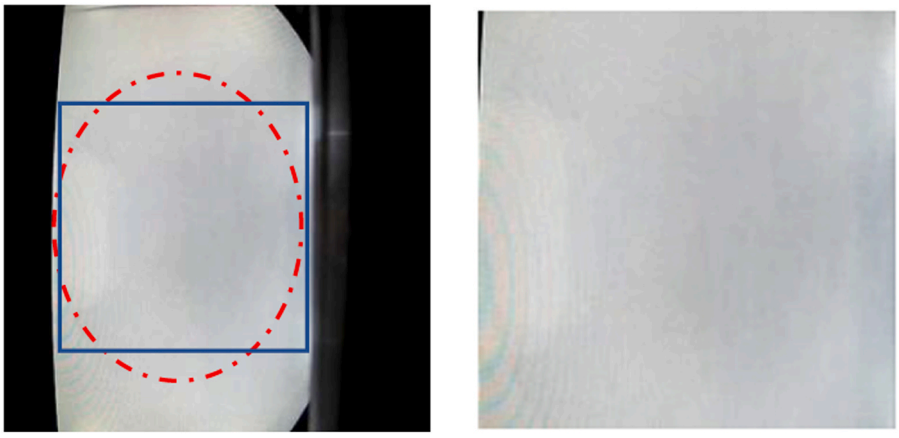


Fig. 2. Preprocessing steps that were applied to all images of the powder bed dataset. (**a**) The previously cropped powder bed image of the size 300 × 300 px with schematically shown printing area (red) and a maximum relevant square image area (blue); (**b**) the final image with the maximum square dimensions of 180 × 180 px. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

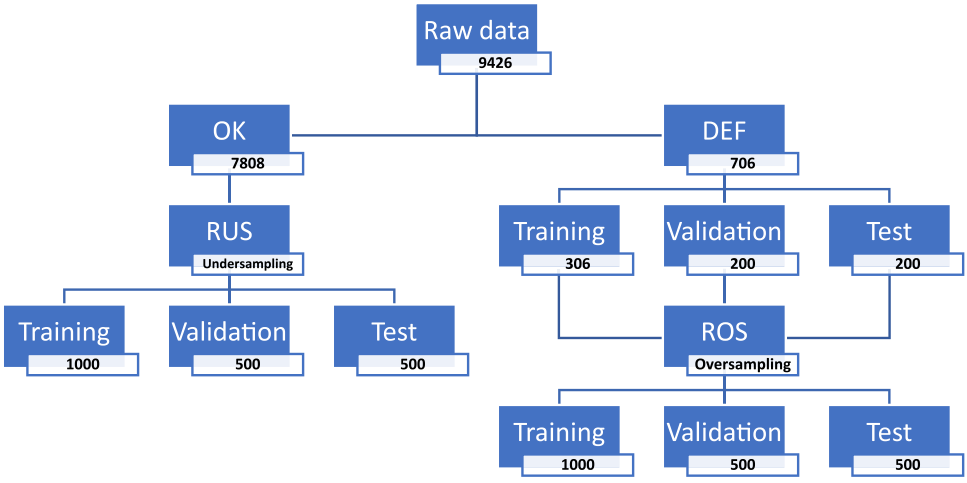


Fig. 3. General procedure for creating the data structure described from the powder bed dataset.

Table 1
CNN hyperparameters.

Cost function	Learning rate (Lr)	Optimizer	No. Epochs	Batch size	Lr decay	Early stopping
Binary cross entropy	1×10^{-3}	Adam ß1 = 0.9 ß2 = 0.999	30	64	patience = 5	patience = 20

steps [4,6]. In the first step, in what is known as feature engineering, an attempt is made to extract relevant data structures from the raw image data using various algorithms. In the second step, what is known as classification, an ML algorithm then attempts to learn a pattern that can map the data structures and a target variable. However, these patterns must have previously been extracted during feature engineering for learning. This approach often leads to unsatisfactory classification results [6].

For this reason, CNN have been widely used in recent years to solve various complex computer vision problems [16,21,22]. These modern ML architectures are also used specifically to monitor and classify certain additive manufacturing processes [3,5,8,9]. CNN are based on the complex structures of real human brain structures of the visual cortex and are one of the latest methods of DL in the field of image recognition [5,6,23]. According to Baumgartl et al. [5], the fundamental difference between a CNN and a classic ML approach lies in the combination of feature engineering and classification.

3.3.1. *Transfer learning*

The training of a CNN model from scratch requires a lot of data to get a good predictive model [21]. However, often insufficient data is available, which requires complex techniques that can produce acceptable prediction results with less data. Transfer learning (TF) offers such a technique. In the case of TF, the features of an already trained CNN model are used as initialization for training the CNN used for the actual classification. According to Tsiakmaki et al. [24], it is an effective ML method to use a CNN model that was previously trained on a very large dataset and the features generated in this way as an initialization for a new CNN model on a much smaller dataset and reuse it with a different purpose. This method is often informative, even if the new classification task is significantly different from the one for which the original model was trained.

There are two methods of using TL [4]. On the one hand, you can use a previously trained CNN model as a feature extractor for a completely different classifier. On the other hand, you can partially use the model again and carry out what is commonly known as a fine-tuning (FT). For this purpose, the top layers of the previously trained CNN model are trained again and optimized in order to better adapt the features generated there to the new dataset.

3.3.2. *CNN architectures used in this work*

As previously mentioned, a common and very effective approach for DL with small image datasets is to use a pretrained network. In this work two different CNN models are used for feature extraction and the results are compared. For this purpose, both networks are instantiated with pretrained weights and implemented with a new classifier. Before training the CNN models with the powder bed data, the individual layers of the models are frozen or defined as non-trainable so that the pretrained weights are not updated again during the new training cycle and thus destroy the previously learned features. With this setting, only the weights from the layers of the new classifier that were added to the CNN models are trained. After the training run, the networks are fine-tuned. For this purpose, individual layers of the respective network are again defined as trainable and used for feature extraction. These layers are then trained again together with the classifier. This re-uses the previously pretrained features of the CNN models to make them more relevant to the new problem. Thus, in such an approach, this investigation enables a better classification of SLS powder bed images. The CNN model architectures presented below enable a production-integrated method for the continuous detection of defects in recorded powder bed images during selective laser sintering and for classification into good and bad production images. The information generated in this way can then contribute to the assessment of the manufactured part quality. Both CNN models first analyse manually preselected image data using special computational operations and learn layer by layer possibly interesting image features. These learned features are then used to evaluate unknown images and thus enable automatic, intelligent image classification.

A CNN model that is used here for feature extraction is the VGG16 architecture, which was developed by Simonyan and Zisserman [25] in 2014. It is a simple and widespread CNN architecture, but which, in principle, no longer corresponds to the current state of the art. A more modern, state-of-the-art CNN architecture, which was also used here as a comparison, is the Xception model designed by Chollet in 2016 [25]. Some important details of the pretrained CNN models that are used with the additional classifier inserted over both architectures are explained below.

The VGG16 architecture is a pretrained CNN from the Visual Geometry Group (VGG) at Oxford University. The model was developed to

significantly increase the depth of the existing CNN architectures and uses 16 network layers for object recognition [25,26]. According to Simonyan and Zisserman [25], 224 × 224 px RGB images are provided as input, which are then passed through a block of convolution layers with a very small filter size (matrix) of 3 × 3 and a convolution step of 1 px. After the respective convolution layers, five Max-Pooling layers with a 2 × 2 px window and a convolution step of 2 px are embedded in order to compress the spatial representation of the input data [25]. The convolutional layer blocks are followed by three fully connected layers, of which the first two each have 4096 channels and the third is used for classification and has 1000 channels for 1000 different classes. A softmax layer followed as the last layer. Fig. 4 shows the structure of the VGG16 network architecture.

Based on the VGG16 architecture, the Xception CNN architecture was developed, which is schematically similar in some areas. Xception is based entirely on the approach of the depthwise separable convolution (DWSC) layers [23]. This is a more up-to-date approach than the previously frequently used separable convolutions, which were described by Mamalet and Garcia [27] in 2012. Sifre [28] developed depthwise separable convolutions in 2013 at Google Inc. and listed detailed experimental results in his doctoral thesis. The DWSC therefore not only deals with the spatial dimensions in neural networks, but also with the depth or the number of color channels of an input [28].

The Xception architecture consists of 36 convolution layers that enable feature extraction. These layers are structured in 14 modules (one module is repeated eight times), all of which, except for the first and the last module, have linear direct connections to one another. The DWSC is a spatial convolution that is executed independently of one another in parallel via each input channel and is followed by a point-by-point 1×1 convolution that projects the output of the channel onto a new channel [23]. According to Chollet [23], this results in deeper models that are extremely efficient. Another advantage of this network architecture is that fewer operations are performed, which means that the computational costs of the model is lower. The Xception model has many more parameters compared to the VGG16 network, but is usually more efficient and faster [23]. The general structure of the Xception CNN architecture is shown in Fig. 5.

Both the VGG16 and the Xception architectures are first instantiated with pretrained weights from the ImageNet dataset. This dataset is a very large benchmark dataset for the detection of objects [29,30]. The respective classification layers of the two models are not loaded in order to enable an efficient feature extraction. A new classifier is implemented for this purpose. Furthermore, the layers of the CNN models are frozen or made untrainable to prevent the pretrained weights from being updated. The new classifier is then trained with the powder bed image data. After this first training run, defined layers of the CNN models are made trainable again and used together with the classifier for fine-tuning and retraining. The classifier is identical for all investigations in this work and consists of a fully connected layer with 1000 channels, a dropout layer with a retention rate of 0.25 and a batch normalization operation. Finally, another fully connected layer with only one channel and a sigmoid activation function was provided to predict the probability of the classes or to perform the binary classification between the OK and DEF classes. Fig. 6 shows a flow chart of the entire TL process.

All calculations were carried out on a local workstation computer, which provides a Windows environment with 32 GB RAM and a GPU Nvidia GeForce RTX 2080 Ti with 11 GDDR6 VRAM.

3.4. Experimental test execution

After video files of the SLS process were recorded, broken into individual frames and cropped, the resulting dataset was cleaned up and the imbalance problem between the two classes was fixed using different techniques. The processed images were then divided into a defined data structure and checked again with regard to their correct classification.

After these steps, the VGG16 and the Xception CNN model were implemented using the Python programming language in the deep learning framework TensorFlow [31], which provides an interface for programming ML algorithms and an implementation for executing them. TensorFlow also includes a data preprocessing tool that is applied to the training and validation groups. After preprocessing the data, the respective network and an additional classifier were trained on the data and then optimized by fine-tuning. These optimized model variants were in turn used to classify the test data and compared with suitable evaluation metrics and visualization methods. Lastly, heat maps were generated to locate the defects in the powder bed images using the Grad CAM process. The experimental procedure and the proposed methodology can be summarized as follows:

1. Recording and preprocessing of SLS image data.
2. Implementation of the CNN models in TensorFlow.
3. Train the respective architecture with the training and validation data.
4. Fine-tuning of the top network layers to improve performance.
5. Enter test data into the trained network to obtain classification results.
6. Evaluation and comparison of the classification results of the networks.
7. Use Grad-CAM to create a heatmap to point to possible defect locations in the test image data.

3.5. Performance measures

At a classification task, the results can be presented and summarized in the form of a special matrix, the confusion matrix (CM) [11]. This CM is shown in Table 2 as an example. In the case of a binary classification, the CM contains the following information:

- Number of examples that are predicted to be recognized as true positive (TP)
- Number of examples that are predicted to be recognized as true negative (TN)

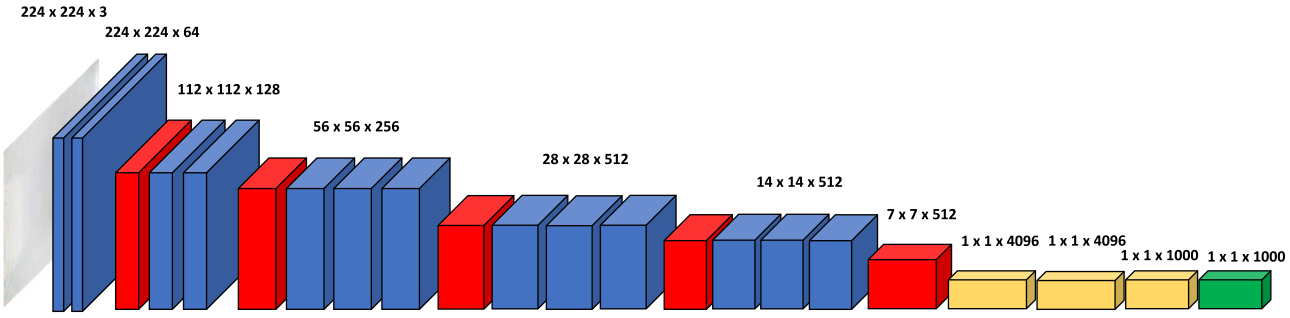


Fig. 4. VGG16 CNN model for the detection and classification of powder bed defects at the SLS.

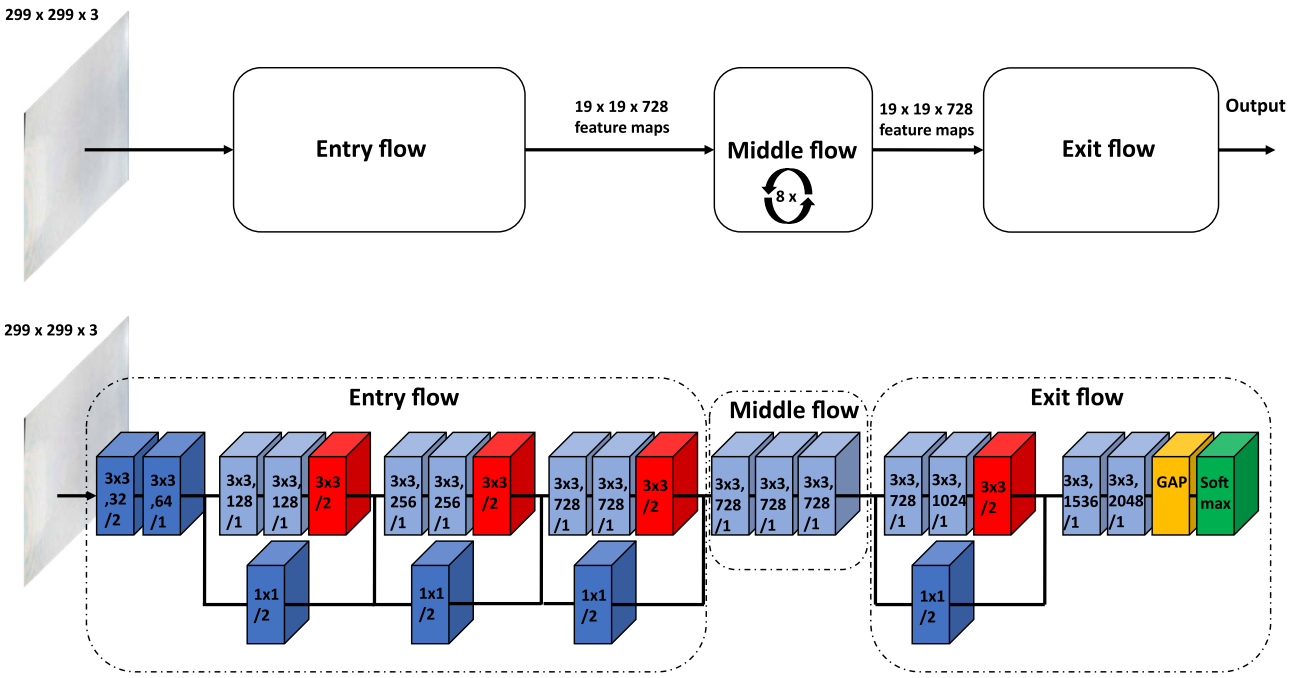


Fig. 5. Xception CNN architecture for the detection and classification of powder bed defects at the SLS. (a) Shows the general structure of the Xception model; (b) shows the detailed architecture of the original Xception model. A batch normalization operation is performed after each convolution and each spatial convolution block. Each block is numbered, the first being the kernel size, the second being the number of filters in the particular block, and the last being the size of the convolution step.

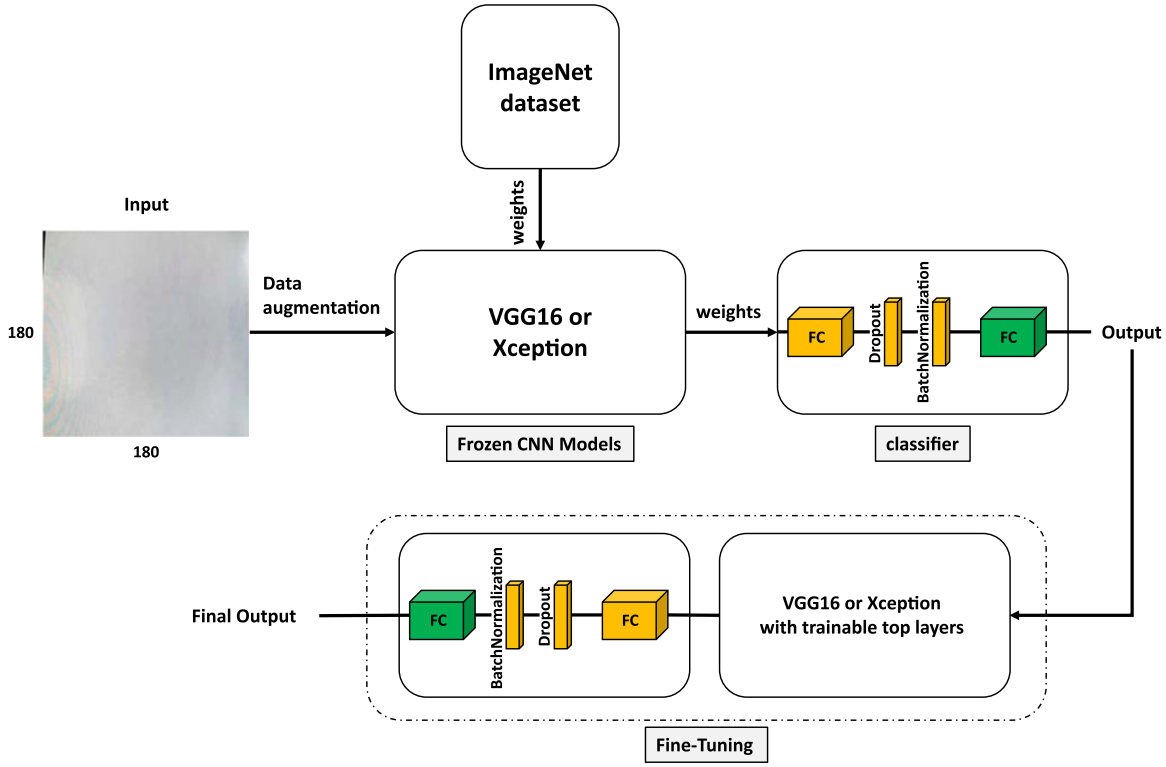


Fig. 6. Flow chart of the TF with the powder bed data.

- Number of examples that are predicted to be recognized as false positives (FP)
- Number of examples that are predicted to be recognized as false negative (FN)

Further metrics for evaluating the performance of a CNN model can then be derived from this CM. The accuracy is the most frequently used metric for binary classification and evaluates the overall effectiveness of a classifier or indicates the proportion of correct predictions. The accuracy is further explained and defined by Johnson and Khoshgoftaar

Table 2
Confusion matrix according to Johnson and Khoshgoftaar [11].

	Actual positive	Actual negative
Predicted positive	True positive (TP)	False positive (FP)
Predicted negative	False negative (FN)	True negative (TN)

[11] as well as by Branco et al. [32]. Especially when working with an imbalanced dataset (but also for balanced datasets), other metrics are often used to evaluate performance. The most frequently used are then precision, recall (formally also referred to as sensitivity) and the F1-Score [11,32].

A graph such as the ROC curve and the associated AUC measure are other useful metrics for evaluating and visualizing the performance of a CNN model in the case of imbalanced data [32,33]. The ROC curve is a graphic representation of the true positive rate (TPR) against the false positive rate (FPR) for all possible prediction thresholds. In other words, it visualizes the trade-off between correctly classified positive examples and misclassified negative examples. The terms used for the ROC curve and the AUC value are defined and further explained by Johnson and Khoshgoftaar and Branco et al. [11,32]. ROC curves, in and of themselves, do not offer a comparable performance indicator [32]. A corresponding value is determined by the AUC. The AUC value is a summarizing performance measure for all possible prediction threshold values and is, according to Branco et al. [32], often used to compare performance between different models.

In most cases, accuracy and loss or respectively cost functions are also recorded to evaluate the performance of the DL [19]. The accuracy graph shows the performance of a classification as a percentage. The loss graph considers the uncertainties of a forecast based on how much it deviates from the actual value. There are several loss functions, not expressed as a percentage, that represent the sum of the errors made for each data item in training or validation sets. This value should always be minimized during training. Both diagrams thus characterize the training process and provide initial information about the effectiveness of the selected hyperparameters and how they should be changed for more efficient training. A commonly used loss function is cross-entropy loss, and especially for binary classification tasks (such as the classification of OK and DEF SLS powder bed images), binary cross-entropy loss [34,35].

In this work, the performance of the CNN models was additionally assessed using the gradient-weighted class activation mapping (Grad-CAM) method according to Selvaraju et al. [36], which offers a heat map for localizing possible powder bed irregularities. Grad-CAM is a technique for the visual description of CNN models, that creates a rough localization map, which highlights the areas of interest in the image for the prediction.

4. Results

4.1. Validation results

Two experiments (the first with the data augmentation described under 3.2.4, the second without) were carried out using the proposed method with the pretrained VGG16 and Xception networks and the additional classifier. For this purpose, the resulting CNN architectures were first trained with the training data and then validated with the validation data. The experiments were also set up in two stages. In the first step, the pretrained network including the classifier was trained with the data and, in a second step, it was optimized through fine-tuning and renewed training.

A third experiment was performed with VGG16 and Xception networks as well as the additional classifier, which were not trained in advance (by ImageNet). For this purpose, the untrained networks including the classifier were trained with the balanced data and the proposed data augmentation and then optimized through fine-tuning and retraining to compare the results with the other experiments.

In this context, Fig. 7 shows the course of accuracy and loss of all three experiments for both the training and the validation data during fine-tuning.

The smoothed curves were presented in the graphs using the exponentially weighted moving average (EWMA). The EWMA course is a weighted representation of the data points of a time series. There the weights decrease exponentially, so that newer data points are weighted more heavily than those that are further back in time [37,38]. The weighting factor on which the EWMA representations are based is 0.75. The two CNN architectures were each trained over 30 epochs and then trained again over 30 epochs in the course of fine tuning.

For the first (1st) experiment with data augmentation, the training accuracy of the VGG16 model was then about 91%, that of the Xception model approx. 94% (see Fig. 7(a), upper graph). The training loss was around 0.24 for the VGG16 architecture and around 0.14 for the Xception architecture (Fig. 7(a), lower graph). In terms of validation accuracy, the VGG16 network architecture achieved with approx. 90% better results than the Xception architecture with approx. 80% (Fig. 7(b), upper graph). With the validation loss, the values of the VGG16 model with approx. 0.25 are ultimately also lower than with the Xception model with approx. 0.5 (see Fig. 7(b), lower graph).

For the second (2nd) experiment without data augmentation, the training accuracy of the VGG16 model was 100%, that of the Xception model was also 100%, but was stopped after 21 epochs due to the defined early stopping hyperparameter (see Fig. 7(a), upper graph). The training loss for the VGG16 model was almost 0.0 and for the Xception model architecture also around 0.0 with a stop in the calculation after 21 epochs (Fig. 7(a), lower graph). In the validation accuracy, the VGG16 network architecture without data augmentation achieved with approx. 98% much better results than the Xception architecture, which was stopped after 21 epochs with exactly 50% validation accuracy (Fig. 7(b), upper graph). At validation loss, the values of the VGG16 model with approx. 0.1 are very much lower than with the Xception model with approx. 464 after an early stop at 21 epochs (see Fig. 7(b), lower graph).

For the third (3rd) experiment with networks that were not previously trained, the training accuracy of the VGG16 model was then about 51%, that of the Xception model was about 77% (see Fig. 7(a), upper graph). The training loss was 0.7 for the VGG16 and about 0.5 for the Xception architecture (Fig. 7(a), lower graph). The validation accuracy of the VGG16 network architecture was slightly better with 51% than with the Xception architecture with approx. 50% (Fig. 7(b), upper graph). In the validation loss, the values of the VGG16 model with approx. 0.7 are significantly lower than with the Xception model with approx. 9.0 (see Fig. 7(b), lower graph).

4.2. Test results

As described above, the powder bed dataset contains preselected test data consisting of a predefined number of 500 OK and 500 DEF images, each selected at random from the total dataset. DEF images were also copied randomly during oversampling to achieve the predefined number of images. After the training, optimization and validation of the two CNN model variants with the training and validation data, the test image data were examined with the models. The same data was used for both models and all experiments. The results are displayed in Table 3 in the form of Confusion Matrix, Precision, Recall (TPR), FPR, F1 Score and AUC values. The best results for each parameter are printed in bold.

To enable a more precise comparison of the two CNN models, the respective ROC curves are, according to Johnson et al. [17], a mandatory part of ML and are also shown in Fig. 8.

First, the individual experiments are considered. It can be stated that the VGG16 model from the 2nd experiment, without data augmentation, initially delivers the best results. The accuracy (0.971) and the ROC-AUC value (0.993) are better than the accuracy (0.958) and the ROC-AUC value (0.982) for the second-best result, which was achieved in the 1st experiment using the VGG16 model architecture with data

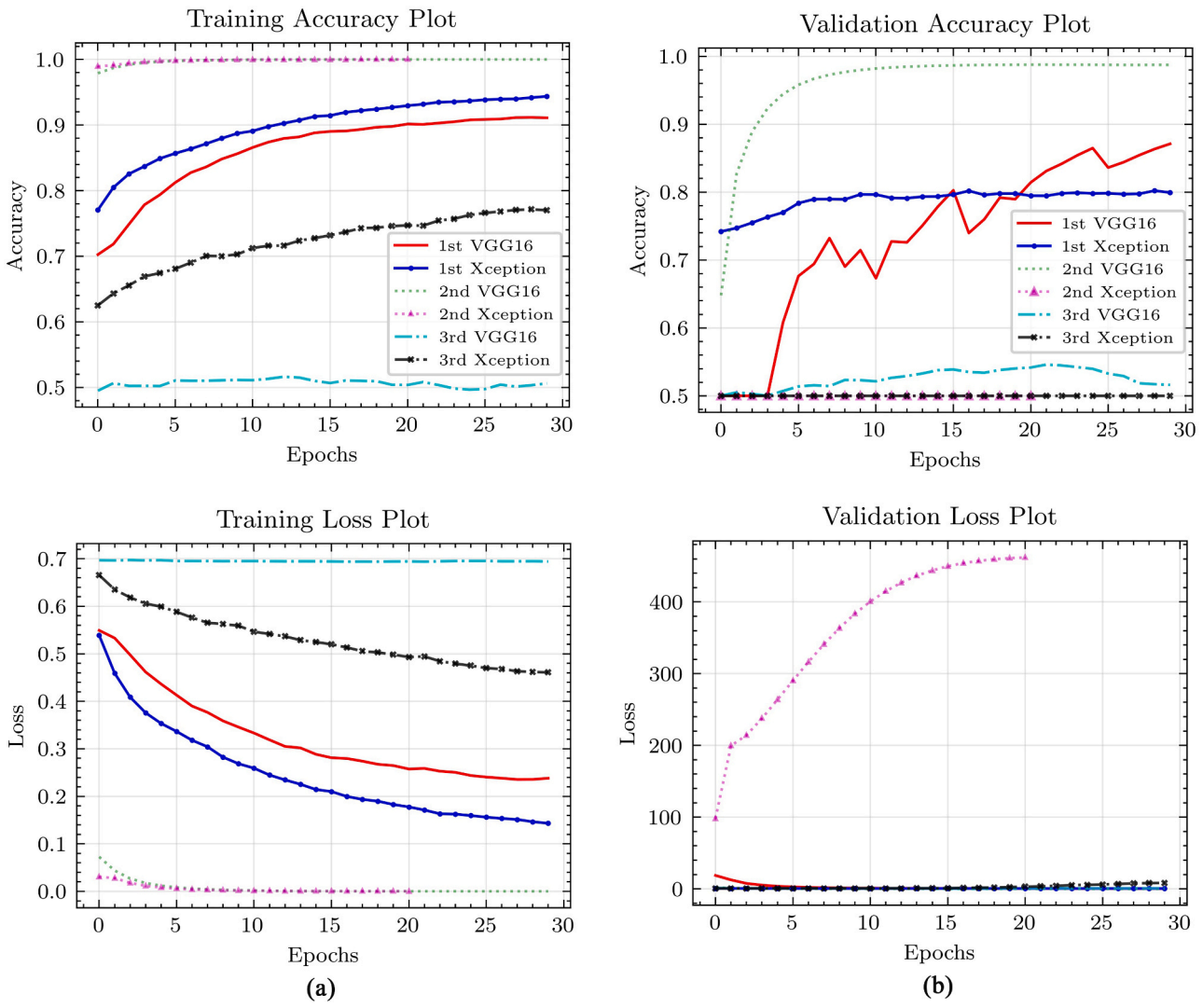


Fig. 7. Accuracy and loss of the used and optimized CNN model architectures and experiments in EWMA representations. (a) shows the course of accuracy (above) and loss (below) in the training data. (b) shows the course of accuracy (above) and loss (below) for the validation data.

Table 3
Confusion matrices and performance parameters for the examined CNN architectures for the classification of powder bed defects at the SLS process for all experiments carried out.

Experiment	Model	Confusion matrix		Accuracy	Precision	Recall (TPR)	FPR	F1-Score	ROC-AUC
1st	VGG16	490	10	0.958	0.939	0.980	0.064	0.959	0.982
		32	468						
	Xception	459	41	0.894	0.876	0.918	0.130	0.897	0.934
		65	435						
2nd	**VGG16**	**496**	**19**	**0.971**	**0.963**	**0.980**	**0.038**	**0.972**	**0.993**
		10	**481**						
	Xception	500	0	0.500	1.000	0.500	0.500	0.667	0.514
		500	0						
3rd	VGG16	180	320	0.515	0.360	0.522	0.489	0.426	0.525
		165	335						
	Xception	500	0	0.500	1.000	0.500	0.500	0.667	0.526
		500	0						

The best results for each parameter are printed in bold.

augmentation. The results of the 3rd experiment achieve accuracies of approx. 0.5 and ROC-AUC values of about 0.52 for both models considered. These results are therefore well below the values of the other two experiments. In the following, the results of the 1st experiment are discussed in more detail, as these values were obtained using the previously proposed method.

The test results of the 1st experiment show that the values of the CM and the performance metrics derived from it for the VGG16 model are higher than those of the Xception model and thus enable a better classification of the test data. Essentially, the accuracy allows a first direct comparison of the performance of the models, whereby the VGG16 accuracy of 0.958 is well above the Xception accuracy of 0.894. The

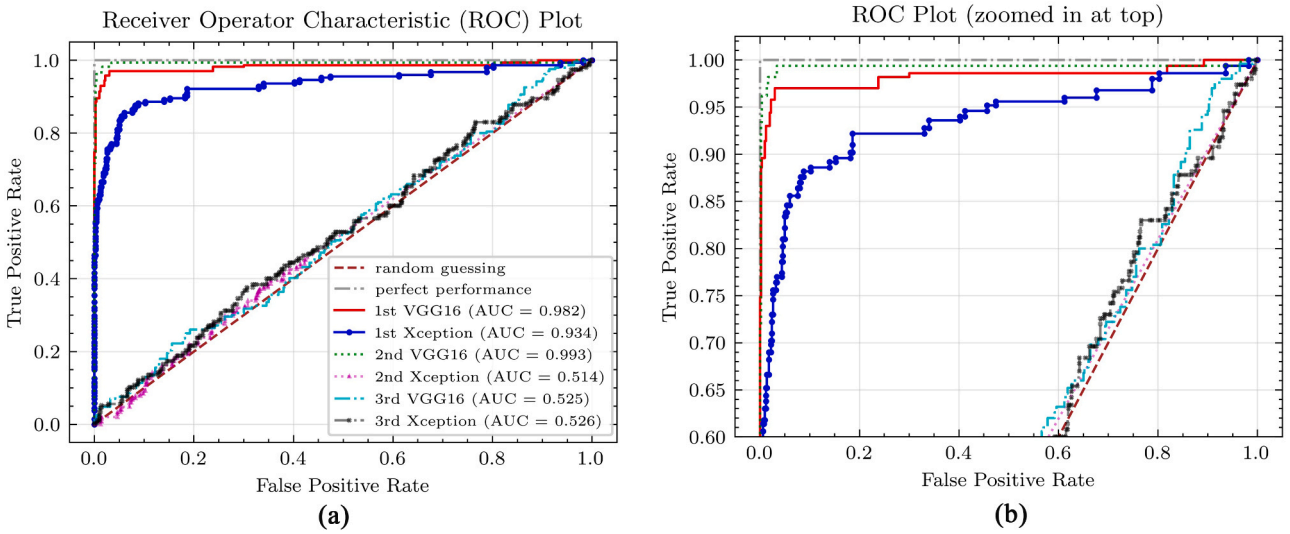


Fig. 8. ROC curves and AUC metrics of the implemented models for every three experiments. The linear dashed lines represent the ROC curve of a completely random classifier and that of a perfect classifier. (a) shows the plot of the ROC curves of the implemented models; (b) shows a zoomed in version of the top part plot.

precision shows the correspondence of the correct class with the correctly classified predictions of the model and shows a significantly better value for the VGG16 model with 0.939 than for the Xception model with 0.876. The recall indicates the efficiency of the model for the classification of the relevant class, here the DEF class and with the VGG16 model with 0.980 it is above the value of the Xception model with 0.918. In this examination with the powder bed image data, the most important thing is to correctly identify all images where defects appear (high recall values). In this way, good component quality can always be guaranteed. But, exclusive consideration of the recall without considering the precision is not recommended. For example, high precision with a low recall results in a very precise but incomplete classification. For this reason, the F1-Score was also calculated. It specifies a harmonic mean between precision and recall and defines how precisely and robustly the models perform on the test data. In principle, a higher F1-Score means a more powerful model. Accordingly, the VGG16 model with an F1-Score of 0.959 allows for a much better and more effective classification of powder bed images than the Xception model with a F1-Score of 0.897.

The ROC curves of the model architectures shown in Fig. 8 were created after fine-tuning the models of each experiment. For this purpose, the TPR was plotted over the FPR and an AUC value was determined. The maximum AUC value (0.982) was achieved by the VGG16 CNN architecture. The Xception architecture achieved a lower AUC value (0.934), which is reflected in a flatter ROC curve.

For better visualization and explanation of the test results, a gradient-weighted Class Activation Mapping was created for selected test images. The Grad-CAM technology is used to create “visual explanations” of the CNN models [36]. According to Selvaraju et al. [36] convolutional layers retain natural spatial information that is subsequently lost in fully connected layers. In this way, semantic, class-specific image information can be searched for in the convolution layers (e.g. for object components such as eyes, ears, cracks etc.). Grad-CAM then uses the gradient information flowing into the final convolution layer of the CNN to generate importance values for a particular property of interest.

In this work, the activation maps of the VGG16 and the Xception model were presented and highlighted using the gradients of the last convolution layer. This made it possible to locate areas of the image that are of most interest for the CNN networks to make decisions. The areas of the image that are most interesting to the CNN are highlighted in red and the less interesting image areas are highlighted in blue. In the case of a DEF powder bed image, the visible defects in the powder bed were recognized by both model architectures and a correct prediction was then made (see Fig. 9).

In the OK powder bed images, the CNN models partially recognized different, invisible image anomalies, which then partially influenced the prediction accuracy of the models (see Fig. 10).

5. Discussion

As shown in the 1st experiment in Table 3, the method described here for detecting and classifying powder bed defects works very well with selective laser sintering and produces excellent results. From these results it can be deduced that the VGG16 model architecture provides the best results with an AUC value of 0.982, an F1-Score of 0.959 and a test accuracy of 0.958. As a result, the developed VGG16 CNN architecture was best able to make predictions about the quality of unseen powder bed images. Unfortunately, there are no comparative values in the current literature to relate the results obtained to further CNN analyzes of powder bed images during selective laser sintering. However, Gobert et al. [9] analyzed CT scans of powder bed images during the SLM process for powder bed defects using an SVM algorithm, with a maximum accuracy of 0.85 being achieved.

The 2nd experiment in Table 3 shows that with the method proposed here and the VGG16 model without data augmentation with an AUC value of 0.993, somewhat better results can be achieved at first glance. This is basically comprehensible, since the data augmentation extends the relatively small dataset through special operations (image rotation, image mirroring etc.) in order to obtain a larger training base. For that, the complexity of the dataset increases, the individual images are no longer as similar as before and classification is more difficult for the model. According to Chollet [4], this is a desired effect in order to enable a better generalization of the models and to avoid overfitting. Especially for the 2nd experiment in Table 3, it can be seen from the curves in Fig. 7 (a) for the VGG16 model that the maximum values for accuracy and loss were already reached after a few training periods. This means that the model has learned patterns that are specific to this training data and can classify them almost perfectly. Since the validation and test data are very similar, the model was also able to achieve very good values there. In contrast to this, in the first experiment with data augmentation, a larger and more complex database was used for training, which resulted in somewhat poorer results, but the models generalize better and are better suited for larger datasets with less similarity among the individual images. For the Xception model from the 2nd experiment, it is immediately apparent that no learning effect has taken place and that the model

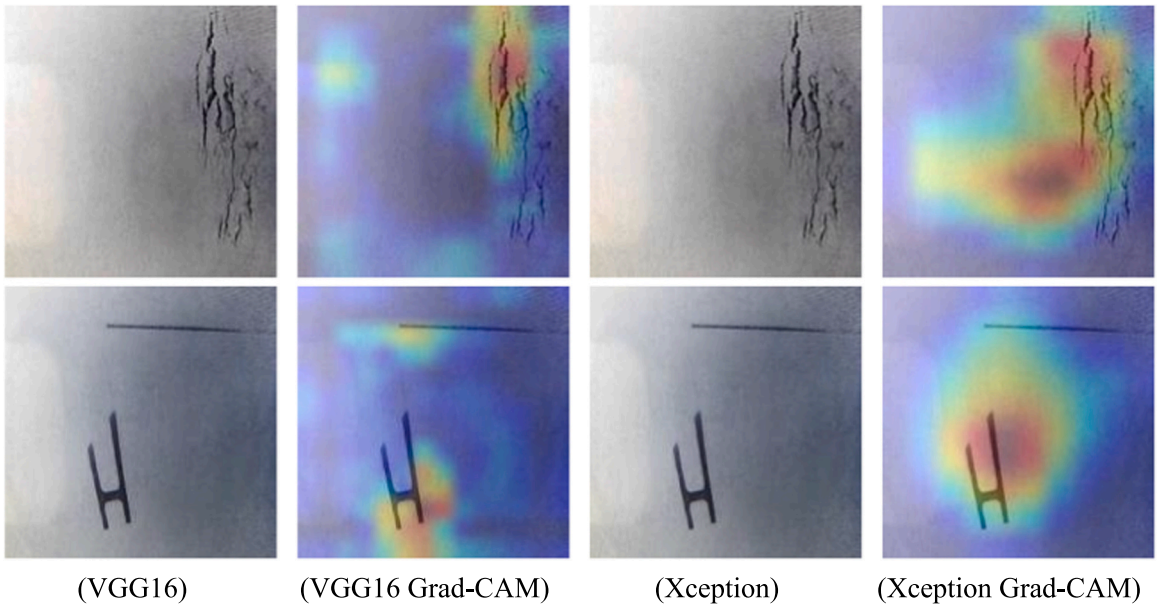


Fig. 9. Activation maps for powder bed recordings during the SLS process with visible powder bed defects. Defects were detected and localized by the CNN architectures. With the VGG16 model, a more precise localization of the effects could be achieved than with the Xception model. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

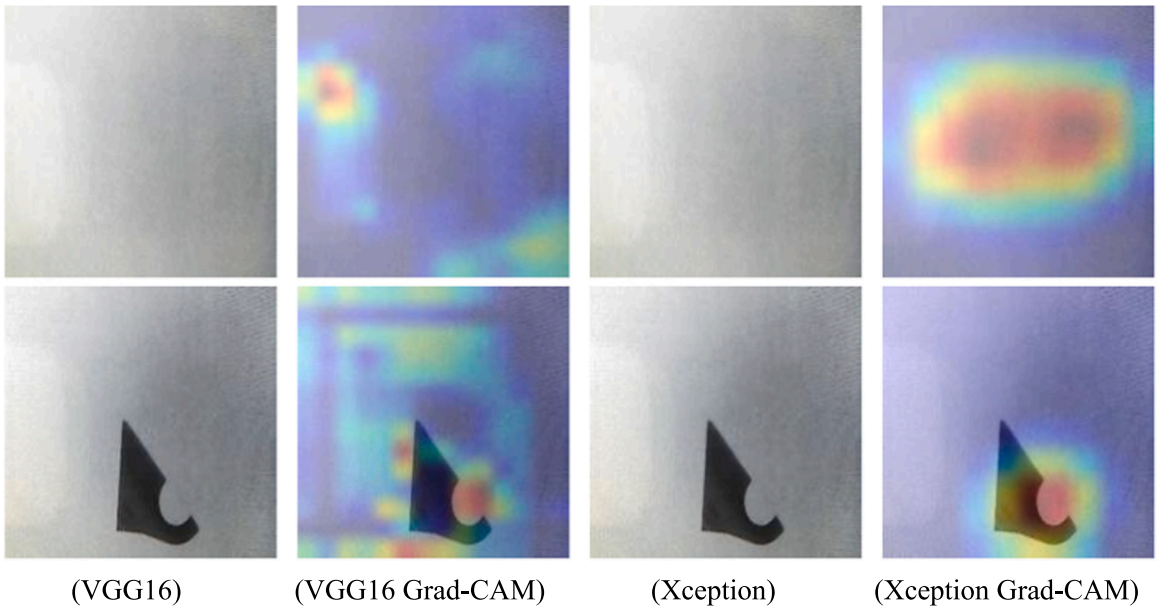


Fig. 10. Activation maps for powder bed recordings during the SLS process without visible powder bed defects. Image anomalies were detected and localized by the CNN architectures. Various anomalies were identified in the VGG16 model and localized relatively precisely. For the Xception model, larger image areas were identified as anomalies and the actual irregularities in the powder bed could therefore often be localized less precisely.

cannot classify the data without data augmentation. This means that the model behaves completely differently than the VGG16 model, but this is due to an overfitting after considering the curves from Fig. 7. The Xception model may have learned misleading or irrelevant information for classification during training. A similar behaviour can be observed for the Xception model from the 3rd experiment. Data augmentation was carried out there, but the models were not trained in advance with the ImageNet dataset. As a result, no learning successes could be determined in this experiment in the 30 observed epochs either. The same applies here to the VGG16 model. However, based on the curves of the validation accuracy and the validation loss in Fig. 7, an overfitting can also be concluded, which has a negative effect on the classification.

In summary, from the investigation of all three experiments it can be determined that the presented method with data augmentation and with pretraining with ImageNet is advantageous in order to achieve better results and to implement more robust model architectures. The data augmentation is particularly important for the generalization of the examined models and also avoids overfitting. A preliminary training with the weights from the ImageNet dataset saves a considerable amount of computational effort, since this has already been carried out there and the proposed models can be based on it. In addition, this also reduces the risk of overfitting, since a much larger database was considered when calculating the model weights.

This work also showed that both investigated CNN model

architectures could learn interesting features from the image data in order to be able to then automatically assess the quality of powder bed images. In the future, this can support the quality assurance of additively manufactured components, e.g. as a supplement to the downstream, non-destructive assessment of the part quality, but also for in-situ monitoring of the additive manufacturing process. Compared to the studies by Xiao et al. [3], real production datasets with real powder bed defects were used for this, while no powder bed irregularities were artificially introduced.

The activation maps generated in this work can specifically identify and localize powder bed defects. This was also shown in the work by Baumgartl et al. [5] for thermographic images during the SLM process, where possible delamination defects in the part layers were localized and detected through different temperature ranges. In this work, the VGG16 model architecture identified the defects more precisely than the more modern Xception architecture and is therefore better suited for image analysis for quality assurance in selective laser sintering. One reason for the better results of the VGG16 model may be that in this model a large number of model parameters (approx. 138 million) are distributed over relatively few model layers (23) and thus enable a more detailed analysis of the individual image data. In the Xception model, for example, there are far fewer parameters (about 23 million) that are distributed over a large number of layers (126). Another reason, however, is the relatively small amount of data, which can result in the Xception model not being able to learn sufficiently due to its complexity.

The lack of available data is a major problem in the basic classification of powder bed defects in selective laser sintering. Normally, DL models are trained over several thousand image data. Training CNN with only a small amount of data can easily lead to an inaccurate classification and can impair the generalization ability of the models. The adjusted powder bed dataset contained more than 8500 images, which were very unevenly distributed. It contained only 706 images with visible defects for the model to learn from. The lack of DEF data is therefore the main problem of the approach presented here. The performance of the models presented could increase as more DEF image data become available. This can first be evaluated with a so-called ablation analysis in a way that conserves resources [39]. According to Fawcett and Hoos [39], algorithms with many influencing parameters are examined with ablation studies to determine which parameters contribute most to changes in performance between two configurations of the algorithm and which changes in the standard configuration of the algorithm actually lead to better performance. For example, if an ablation study on a model architecture with less data achieves a performance comparable to that of a larger dataset (with the same parameter configuration), no significant increases in performance are expected from an even larger database.

Furthermore, the problem of image preprocessing must be considered. By cropping the images to a size of 180×180 pixels, it is not possible to capture the entire cylindrical build area of the SLS system without capturing undesired black border areas. This should be optimized through more suitable camera positioning and better camera focus.

Another difficulty in this investigation was the interpretation of the visualization results of the CNN models. A deeper understanding of the visual characteristics of a digital image and the individual convolutional operations within the neural networks is required in order to be able to explain the predictions and visualization results in detail. The resulting activation maps were recorded and fundamentally analyzed in this context, but further explanations are required in order to understand why the model particularly highlights specific areas of the image. These interpretation problems may be resolved in the future through larger datasets and more detailed research.

6. Conclusions

In this paper, we introduce different machine learning architectures that can be used to automatically differentiate between good and bad powder bed images during selective laser sintering. There, good images without visible defects in the powder bed are marked as OK and images with visible defects and irregularities as DEF. The investigated methods used techniques of transfer learning with pretrained weights of the ImageNet dataset, which served as initialization for the VGG16 CNN model as well as for the Xception CNN model. Then a new classifier was provided consisting of a fully connected layer with 1000 channels, a dropout layer with a retention rate of 0.25, a batch normalization operation and another fully connected layer with only one channel and a sigmoid activation function. The images used were preprocessed in a defined manner, divided into classes and reproduced using established methods in order to generate a balanced dataset. With transfer learning it was then possible to work effectively with the small dataset. The VGG16 CNN architecture achieved the best results and clearly outperformed the results of the Xception architecture. With the VGG16 approach and a special data augmentation, a test accuracy of 0.958 was achieved, as well as a precision of 0.939, a recall of 0.980, an F1-Score of 0.959 and an AUC value of 0.982 for the VGG16 ROC curve. The results were visualized with the Grad-CAM method and compared for both methods. Both neural network architectures were able to recognize and localize powder bed irregularities.

As regards future work, so-called ablation studies should first be carried out in order to evaluate the CNN architectures presented and to examine the performance of the models with even smaller datasets. If performance results similar to those in this study are achieved, no significant increases in performance can be expected for these model architectures, even with larger amounts of data. Otherwise, investigations should be carried out on a larger dataset in future work. In particular, more DEF image data needs to be included in the examinations and the impact on the analyzed results considered. In addition, the effects of various data preprocessing steps and methods should be investigated and various hyper parameter configurations should be tested to further improve the performance of the models. Moreover, further CNN model variants and classifiers should be investigated with the powder bed data in order to generate even more powerful and faster transfer learning variants. Finally, according to the ImageNet dataset, a special dataset should be created with various classes and image examples on process irregularities and part defects in additive manufacturing, with which the respective model architectures can be trained in advance. This should significantly increase the effectiveness of CNN models in terms of error detection in various additive manufacturing processes.

Funding

This research was funded by the European Union, which was made available through the European Regional Development Fund (ERDF) and the Ministry for Economics, Employment and Health of Mecklenburg-Vorpommern, Germany, grant number TBI-V-1-345-VBW-118.

CRediT authorship contribution statement

Erik Westphal: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing - original draft. **Hermann Seitz:** Funding acquisition, Investigation, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge the project partner ESD Elektro-Systemtechnik GmbH Dargun for their technical support and the provision of the SLS printing system and SLS printing powder for the development of this work.

References

[1] A. Mazzoli, Selective laser sintering in biomedical engineering, Med. Biol. Eng. Comput. 51 (2013) 245–256, https://doi.org/10.1007/s11517-012-1001-x.
[2] M. Schmid, Laser Sintering with Plastics: Technology, Processes, and Materials, 2018, Hanser; Munich, GER, https://doi.org/10.3139/9781569906842.
[3] L. Xiao, M. Lu, H. Huang, Detection of powder bed defects in selective laser sintering using convolutional neural network, Int. J. Adv. Manuf. Technol. 107 (2020) 2485–2496, https://doi.org/10.1007/s00170-020-05205-0.
[4] F. Chollet. Deep Learning with Python, Manning, Shelter Island, NY, 2018.
[5] H. Baumgartl, J. Tomas, R. Buettner, M. Merkel, A deep learning-based model for defect detection in laser-powder bed fusion using in-situ thermographic monitoring, Prog. Addit. Manuf. 5 (2020) 277–285, https://doi.org/10.1007/s40964-019-00108-3.
[6] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444, https://doi.org/10.1038/nature14539.
[7] H. Shen, W. Du, W. Sun, Y. Xu, J. Fu, Visual detection of surface defects based on self-feature comparison in robot 3-D printing, Appl. Sci. 10 (2020) 235, https://doi.org/10.3390/app10010235.
[8] L. Scime, J. Beuth, Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm, Addit. Manuf. 19 (2018) 114–126, https://doi.org/10.1016/j.addma.2017.11.009.
[9] C. Gobert, E.W. Reutzel, J. Petrich, A.R. Nassar, S. Phoha, Application of supervised machine learning for defect detection during metallic powder bed fusion additive manufacturing using high resolution imaging, Addit. Manuf. 21 (2018) 517–528, https://doi.org/10.1016/j.addma.2018.04.005.
[10] P. Yadav, O. Rigo, C. Arvieu, E. Le Guen, E. Lacoste, In situ monitoring systems of the SLM process: on the need to develop machine learning models for data processing, Crystals 10 (2020) 524, https://doi.org/10.3390/cryst10060524.
[11] J.M. Johnson, T.M. Khoshgoftaar, Survey on deep learning with class imbalance, J. Big Data 6 (2019) 27, https://doi.org/10.1186/s40537-019-0192-5.
[12] M. Buda, A. Maki, M.A. Mazurowski, A systematic study of the class imbalance problem in convolutional neural networks, Neural Netw. 106 (2018) 249–259, https://doi.org/10.1016/j.neunet.2018.07.011.
[13] X.-Y. Liu, J. Wu, Z.-H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Trans. Syst. Man Cybern. B Cybern. 39 (2009) 539–550, https://doi.org/10.1109/TSMCB.2008.2007853.
[14] J. Wu, S.C. Brubaker, M.D. Mullin, J.M. Rehg, Fast asymmetric learning for cascade face detection, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2008) 369–382, https://doi.org/10.1109/TPAMI.2007.1181.
[15] J. van Hulse, T.M. Khoshgoftaar, A. Napolitano, Experimental perspectives on learning from imbalanced data, in: ICML '07 & ILP '07: The 24th Annual International Conference on Machine Learning held in conjunction with the 2007 International Conference on Inductive Logic Programming, Corvalis Oregon USA, Association for Computing Machinery New York NY United States, (2007), pp. 935–942.
[16] F. Pasa, V. Golkov, F. Pfeiffer, D. Cremers, D. Pfeiffer, Efficient deep network architectures for fast chest X-ray tuberculosis screening and visualization, Sci. Rep. 9 (2019) 6268, https://doi.org/10.1038/s41598-019-42557-4.
[17] N.S. Johnson, P.S. Vulimiri, A.C. To, X. Zhang, C.A. Brice, B.B. Kappes, A. P. Stebner, Invited review: machine learning for materials developments in metals additive manufacturing, Addit. Manuf. 36 (2020), 101641, https://doi.org/10.1016/j.addma.2020.101641.
[18] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning, J. Big Data 6 (2019) 60, https://doi.org/10.1186/s40537-019-0197-0.
[19] Chollet, F. (2015). Keras, https://keras.io.
[20] F. Hutter, L. Kotthoff, J. Vanschoren, Automated Machine Learning, Springer International Publishing, Cham, 2019.
[21] J. Luján-García, C. Yáñez-Márquez, Y. Villuendas-Rey, O. Camacho-Nieto, A transfer learning method for pneumonia classification and visualization, Appl. Sci. 10 (2020) 2908, https://doi.org/10.3390/app10082908.
[22] M.F. Hashmi, S. Katiyar, A.G. Keskar, N.D. Bokde, Z.W. Geem, Efficient pneumonia detection in chest Xray images using deep transfer learning, Diagnostics 10 (2020), https://doi.org/10.3390/diagnostics10060417.
[23] F. Chollet, Xception: deep learning with depthwise separable convolutions, in: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, IEEE, 21.07.2017 - 26.07.2017, pp. 1800–1807.
[24] M. Tsiakmaki, G. Kostopoulos, S. Kotsiantis, O. Ragos, Transfer learning from deep neural networks for predicting student performance, Appl. Sci. 10 (2020) 2145, https://doi.org/10.3390/app10062145.
[25] S. Karen, and A. Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
[26] H. Shin et al., "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," in IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1285-1298, May 2016, doi: 10.1109/TMI.2016.2528162.
[27] F. Mamalet, C. Garcia, Simplifying ConvNets for fast learning. Artificial Neural Networks and Machine Learning – ICANN 2012, Springer-Verlag Berlin Heidelberg, Lausanne, Switzerland, 2012.
[28] L. Sifre, Rigid-Motion Scattering For Image Classification (Ph.D. thesis), 2014, CiteSeerX: Paris, FR.
[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2015) 211–252, https://doi.org/10.1007/s11263-015-0816-y.
[30] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (2017) 84–90, https://doi.org/10.1145/3065386.
[31] Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).
[32] B. Paula, L. Torgo, and R. Ribeiro. "A survey of predictive modelling under imbalanced distributions." arXiv preprint arXiv:1505.01658 (2015).
[33] T. Fawcett, An introduction to ROC analysis, Pattern Recognit. Lett. 27 (2006) 861–874, https://doi.org/10.1016/j.patrec.2005.10.010.
[34] Yaoshiang Ho and Samuel Wookey, The real-world-weight cross-entropyloss function: Modeling the costs of mislabeling. IEEE Access, 8:4806–4813, 2019.
[35] U. Ruby, Binary cross entropy with deep learning technique for Image classification, IJATCSE 9 (2020) 5393–5397, https://doi.org/10.30534/ijatcse/2020/175942020.
[36] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: visual explanations from deep networks via gradient-based localization, Int. J. Comput. Vis. 128 (2020) 336–359, https://doi.org/10.1007/s11263-019-01228-7.
[37] T. Fehlmann, E. Kranich, Exponentially Weighted Moving Average (EWMA) prediction in the software development process, in: Proceedings of the 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement, Rotterdam, Netherlands, IEEE, 06.10.2014 - 08.10.2014, pp. 263–270.
[38] D. Leung, J.A. Romagnoli, Fault diagnosis methodologies for process operation, in: Software Architectures and Tools for Computer Aided Process Engineering, Elsevier, 2002, pp. 535–556.
[39] C. Fawcett, H.H. Hoos, Analysing differences between algorithm configurations through ablation, J. Heuristics 22 (2016) 431–458, https://doi.org/10.1007/s10732-014-9275-9.