applied sciences

MDPI

Article

Classification and Fast Few-Shot Learning of Steel Surface Defects with Randomized Network

Amr M. Nagy [1,2,*] and László Czúni [1]

[1] Faculty of Information Technology, University of Pannonia, Egyetem u. 10, 8200 Veszprém, Hungary; czuni@almos.vein.hu
[2] Faculty of Computers and Artificial Intelligence, Benha University, Banha 13511, Egypt
* Correspondence: amr.nagy@virt.uni-pannon.hu; Tel.: +36-707-854-218

Abstract: Quality inspection is inevitable in the steel industry so there are already benchmark datasets for the visual inspection of steel surface defects. In our work, we show, contrary to previous recent articles, that a generic state-of-art deep neural network is capable of almost-perfect classification of defects of two popular benchmark datasets. However, in real-life applications new types of errors can always appear, thus incremental learning, based on very few example shots, is challenging. In our article, we address the problems of the low number of available shots of new classes, the catastrophic forgetting of known information when tuning for new artifacts, and the long training time required for re-training or fine-tuning existing models. In the proposed new architecture we combine EfficientNet deep neural networks with randomized classifiers to aim for an efficient solution for these demanding problems. The classification outperforms all other known approaches, with an accuracy 100% or almost 100%, on the two datasets with the off-the-shelf network. The proposed few-shot learning approach shows considerably higher accuracy at a low number of shots than the different methods under testing, while its speed is significantly (at least 10 times) higher than its competitors. According to these results, the classification and few-shot learning of steel surface defects can be solved more efficiently than was possible before.

Keywords: steel surface defects; visual inspection; deep learning; few-shot learning; EfficientNet; randomized neural network

Citation: Nagy, A.M.; Czúni, L. Classification and Fast Incremental Learning of Steel Surface Defects with Randomized Network. *Appl. Sci.* **2022**, *12*, 3967. https://doi.org/10.3390/app12083967

Academic Editor: Dae-Ki Kang

Received: 11 March 2022
Accepted: 8 April 2022
Published: 14 April 2022

1. Introduction

There is a variety in the appearance of surface defects in industry, for example, hot-rolled steels, solar panels strips, electronic commutators, steel rails, fabrics, printed circuit boards, magnetic tiles, and more. Steel is the most important metal in terms of quantity and variety of applications in the modern world. Surface defects in hot-rolled steel can be associated with the steel production process, its casting, deformation conditions, crystallization of the ingot, etc. They have a considerable impact on the metal's technological qualities during future processing as well as on its operational features. Due to the manufacturing process and environmental conditions, steel surfaces can have a variety of defects. The non-uniform surface brightness and the variety of shapes of defects make their detection challenging. Additionally, new defects, which have never been seen before, may arise throughout the manufacturing process. In the beginning, these new defects may have few shots, so we have to go through fast incremental learning to incorporate them into the classification model as soon as possible.

Steel surface defects show various random patterns, which are good targets for the testing and comparison of concurrent classification methods. Figures 1 and 2 illustrate samples of the two popular benchmark datasets: the Xsteel surface defect dataset (X-SSD) [1] and the Northeastern University surface defect database (NEU) [2]. There are six categories in NEU and seven in X-SSD, with two types of defects present in both. Since the two common artifacts (inclusion and scratches) look somewhat different, we handle them independently.

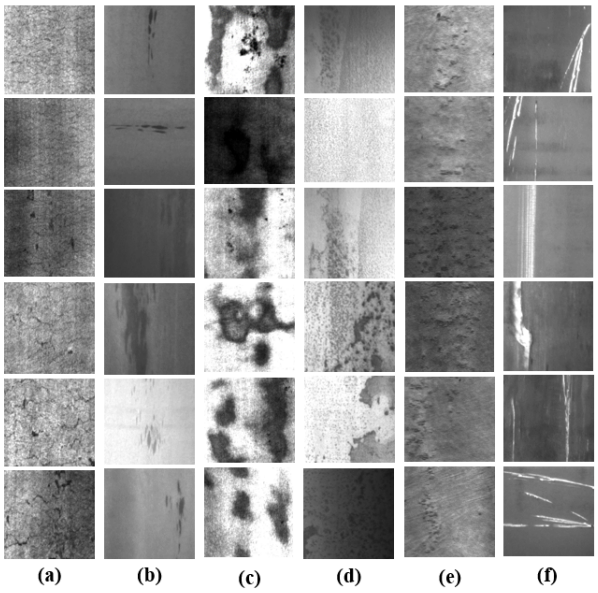

Figure 1. Examples for the six kinds of defect classes of the Northeastern University surface defect database (NEU) [2]: (**a**) crazing (Cr), (**b**) inclusion (In), (**c**) patches (Pa), (**d**) pitted surface (Ps), (**e**) rolled-in scale (Rs), (**f**) scratches (Sc).

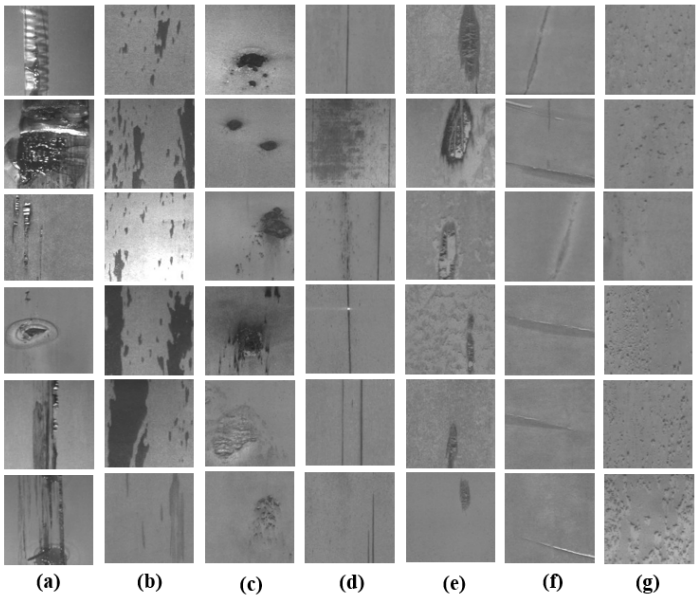

Figure 2. Examples for the seven kinds of defect classes of the Xsteel surface defect dataset (X-SSD) [1]: (**a**) inclusion (Si), (**b**) red iron sheet (Ri) , (**c**) iron sheet ash (Is), (**d**) scratches (Ss), (**e**) oxide scale-of-plate system (Op), (**f**) finishing roll printing (Fr), (**g**) oxide scale-of-temperature system (Ot).

The image processing methods dealing with steel surface defects can be classified into two main categories:

- Detection: The task is to decide whether the image patch has any errors, sometimes the error should be localized (segmented) and also classified [3–7];
- Classification: In case of classification, the image patches should be labeled according to the visible defect type. Training examples are available to learn the visual features of known classes [8–10].

Surface defect detection can be considered as a typical problem in visual quality inspection thus it is related to many areas of pattern recognition, machine learning, image

processing, and computer vision. In most applications, beside the questions of detection and classification, the following key problems are to be answered [11]:

- The real-time problem: The detection time should be fast enough to support the undergoing (production) process without significant loss in accuracy;
- Small target problem: Often the absolute or the relative (to the image) size of the defect is small. In our article we do not have to face this problem directly since we use benchmark datasets, however, in a real-life application, the size can have an indirect effect on the real-time problem;
- The small sample problem: The number of defect images, used either in manual parameter tuning or in automatic learning mechanisms, is often very limited;
- Unbalanced sample problem: This problem mainly exists in the task of supervised learning and relates to the previous one. Often the number of normal samples forms the majority, while the amount of defected samples only accounts for a small part. The few-shot learning problem means that we have a very unbalanced set;
- Domain shift [7]: When the dataset used in training and the dataset in practice are captured under different conditions, it can result in poor detection performance.

In incremental learning, new data (i.e., new shots of previously seen or unseen classes) arrive in phases over time and we have to extend our classification model to include these new classes or new samples of classes. In general, in the literature different synonyms are used for this problem, such as continual learning [12,13], lifelong learning [12,14], or incremental learning [15–20]. The three major challenges in these scenarios are:

- Catastrophic forgetting: It was already investigated in the early 1990s [21] that a neural network is going to have a lower performance of previously trained classes when re-trained or tuned with new ones in focus. The problem is most serious if we have no training data for the already trained classes, or the dataset for re-training is unbalanced due to the missing samples from old classes. There are several approaches to avoid this case by methods such as scaling the weights of trained classifiers [22], using dual memories for storing old images and their statistics [23], or progressive incremental learning [24] where sub-networks are added incrementally to previous ones as the task is growing with new classes;
- Low number of representatives of new classes: In practical applications there is a data collection period when lots of samples are collected about the possible artifacts and used for model building. In contrast, when the system is in use for long periods, the undergoing background manufacturing processes (changes in the environment, or other influences) can result in new defect classes, which might appear in very few samples initially. In general, solutions are categorized into three main sets: using prior knowledge to augment the supervised experience (in our case data augmentation of the available few shots); modifying the model, which uses prior knowledge of known classes; and algorithmic solutions to alter the search for the best hypothesis in the given hypothesis space [25]. In extreme cases, we talk about zero-shot learning: when a new type of defect appears, it is a question whether we are able to classify it as a new class or if it will fall into an existing category. We would like to avoid this latter case and thus we have to create a new category and be able to tune the existing model to consider it [10,26,27];
- Complexity of updating the classifiers: It is a practical problem during the application of deep learning models that the re-training or fine-tuning of new or updated models can be time consuming or computationally very complex. A fast adaptation to the extended set of classes and easy training procedures are needed for real-life applications.

Typically, the above three problems appear simultaneously. The advantage of our proposed solution is that it addresses all of them with one architecture, instead of attacking the problems independently with different methods. The main contributions in our article are the following:

- Instead of re-tailoring old models or defining new deep neural architectures, we show that a well-proven and efficient deep neural network (DNN) (EfficientNet-B7 [28]) can give the best known accuracy for the classification of steel surface defects on two major benchmark datasets;
- We propose a novel architecture designed for incremental learning with the following features: it avoids the catastrophic forgetting of old classes, it has good performance in case of very few shots, and it can be trained very fast for new classes. To achieve this, we apply randomized networks concatenated to the feature extraction of a pre-trained DNN. The computation of its weights can be done very efficiently by the Moore–Penrose generalized inverse.

Our paper is organized as follows. In the next section we overview related papers, then, in Section 3, we introduce the proposed architecture and training method. In Section 4, the used datasets are described while the experiments, evaluations, and discussions can be found in Sections 5 and 6. Finally, we conclude our article in the last section.

2. Related Works

Surface defect detection methods can be classified into traditional feature-based machine vision algorithms and deep learning algorithms [11]. Since all of the latest papers belong to the latter set, our review contains such approaches. We review related papers in three groups: steel surface classification methods, zero-shot learning approaches, and few-shot learning approaches (with some particularly tested on steel surface defects). For the summary see Table A1.

2.1. Detection and Classification of Steel Surface Defects

In ref. [5], the authors introduced an improvement of the YOLOv4 architecture by adding a feature pyramid network (FPN) module after sampling, on the so-called neck part of the network. Enhancing the feature information this way, the experimental results showed a better average detection accuracy (92.5%) than the original network. Tests were carried out only on three defect types of the NEU dataset (crazing, patches, scratches) and more improvements were expected by the authors if larger training sets could be involved.

In ref. [4], an improved faster R-CNN model was proposed by using deformable convolutions instead of conventional convolutions to get complex features for the detection of the various artifacts. Moreover, the RestNet backbone-based network was enhanced by multi-level feature fusion. By these techniques, the detection rate could be enhanced by appr. 0.13 MAP.

In ref. [6], a framework called CPN (classification priority network) was described for the detection and classification of defects. In CPN, the image is first classified by a multi-group CNN, training different groups of convolution kernels separately to extract the feature map groups of different types of defects. Then, according to the classification result, the feature map groups (named multiple group CNN, MG-CNN) that may contain defects are separately input into another neural network, to regress the bounding boxes of the corresponding defects. The detection rate reached 96% and the recognition rate 98.3% on their own dataset. These seem to be good results; however, the zero-shot possibilities were not mentioned.

In ref. [3] proposed an end-to-end defect detection network (DDN). First, ResNet is used to generate feature maps of the input images. Then, these feature maps are fed into the proposed multi-level feature fusion network (MFN). MFN generates one feature vector from the lower-level and higher-level features. A region-proposal network is used to generate regions of interest (ROIs) based on these hierarchical features. Finally, a detector, which consists of a classifier and a bounding box regressor, is used to produce the final detection results for each ROI. DDN results, when using ResNet50 as backbone, show that it could achieve 99.67% accuracy for defect classification and 82.3 MAP for defect detection. We note that the former value is equivalent to the plain ResNet50 according to ref. [29].

In ref. [7] attacks the classification problem when there are changes in the features of the patterns, for example, the appearance of steel surface defect changes during long production intervals. Their framework is called DA-ACNN, since it combines domain adaptation (DA) and the adaptive learning rate of the convolutional neural networks (ACNN). To enable cross-domain and cross-task recognition, they included an additional domain classifier and a constraint on label-probability distribution to account for the lack of labels in a new domain. Additionally, to increase network performance, the normal distribution and a quadratic function are utilized to optimize the loss.

In ref. [8], the authors proposed a steel surface defection classification technique fine-tuned with the help of a feature visualization network. The proposed model consists of two main components: the VGG19 network, trained, fine-tuned, and later used for the classification of surface defects, and a decoding part (DeVGG19). The feature maps of the decoding part are used to find the best settings of VGG19 using image and feature map comparisons with the structured similarity index measure (SSIM), and while it is an interesting approach, it is not obvious for the reader why the aspects of the decoded visual appearance could have a positive feedback on the parameters of the network used purely for classification. This interesting approach could reach 89.86% on the NEU dataset, which is below the state of the art (see VSD in Section 5).

In ref. [9], different versions of ResNet were investigated to classify three kinds of defects on metal surfaces (scratches, scrapes, abrasions, and normal samples). The best results were achieved with ResNet152, with a classification accuracy of 97.1% on their own collection of images (including images of NEU). This paper is a good illustration that properly trained off-the-shelf networks can reach a good performance on steel surfaces. Another example is a modified AlexNet for feature extraction, where the classification was solved with a support vector machine [30]. Performance on all classes of NEU showed 99.70% accuracy. The highest known performance is published in ref. [10], where VGG16 was extended with several layers for classification (ExtVGG16) and could reach 100% on the six classes of NEU.

2.2. Zero-Shot Learning

Zero-shot learning can be considered as a special case of few-shot learning, when there are no samples at all from some of the classes. In general, it models learned parameters for seen classes together with their class representations and rely on the representational similarity among class labels. Semantic attributes can represent the characteristics, as latent attributes, of samples. The success of zero-shot learning thus depends on the quality of semantic attributes: whether they have sufficient discrimination and expressiveness. However, semantic attribute annotations are very hard to get in industrial applications. One example of the usage of such an approach for surface defects is in ref. [27], where the so-called GloVe model [31] was used to extract word vectors as the auxiliary knowledge source (since they used a different dataset, the results are not comparable to the others involved in our article). If no such kind of semantic attribute annotations are available, then still-clustering-type approaches can work. In ref. [26], the authors show a method in which zero-shot learning may be used to detect steel surface defects with a Siamese network. The network is to learn whether two input samples belong to the same class or not. In experiments, three defect classes were trained from the NEU dataset, and the Siamese-based clustering was run on the other three classes, which were never shown to the network during training. The accuracy of this task, reaching 83.22%, could be surpassed in ref. [10], where a Siamese network was also used. This latter approach uses a slightly more complex structure and there are also differences in dropouts and regularization, which may all play an important role to reach better accuracy (85.8%) at the same testing and training conditions.

2.3. Few-Shot Learning

A learning model fed with sufficient and high-quality data is more likely to yield more-or-less accurate results. However, sometimes it is difficult to collect enough defect

samples to achieve a good training model based on traditional learning structures. This can be the case in the production of steel strips, where steel surface defects are scarce and hard to collect. To overcome this problem, special approaches appeared, which can efficiently learn from a very small number of training data.

In ref. [32] shows a simple approach for few-shot learning of steel surface defects. Three feature extraction networks (ResNet, DenseNet, and MobileNet) are compared and two kinds of feature transformations are tested (mean subtraction and L2 normalization). For one-shot learning, classification is accomplished by choosing the nearest neighbor, whereas for the multi-shot setting, the average value of the known sample feature vectors of each class is used as a class prototype to compute the nearest neighbor for the query. For the one-shot case, MobileNet reached the highest accuracy with 87.3%, while for five shots, 92.33% was achieved by DenseNet121 on the NEU dataset.

In ref. [33], a more sophisticated semi-supervised learning model is described, which learns from both labeled and unlabeled samples. They use graph convolutional networks (GCN) [34], naming their model multiple micrographs graph convolutional network (MMGCN). GCNs construct graphs where the images are represented by nodes and their relationships by edges. Feature information propagates between connected nodes, and the distance of connected nodes is closer than the distance of the disconnected ones. MMGCN performs graph convolution by constructing multiple micrographs instead of a large graph, and labels unlabeled samples by propagating label information from labeled samples to unlabeled samples in the micrographs to obtain multiple labels, while final labels are obtained by weighting the labels. The experimental results demonstrate that the proposed MMGCN can achieve a lower computation complexity and practicality, and a higher accuracy than GCN. In fully supervised mode, MMGCN could reach 99.72% accuracy, while when the ratio of known labels was only 25%, accuracy was 98.06%.

Another approach for label propagation is in ref. [35], where defects on the surface of lithium batteries are to be found. Training images were used to fine-tune a pre-trained ResNet10 model, then a k-NN graph was built to evaluate the distance between samples, including both the labeled and the unlabeled ones. To refine the scoring of samples, the label propagation method of ref. [36] was used. Unfortunately, while classification accuracy on their own dataset was good, the effect of label propagation was not analyzed in the article.

Few-shot learning can be interpreted as the problem of unbalanced training sets. Ref. [37] proposes a meta-training approach to handle the unbalanced data caused by the newly arriving error classes having only few shots. Each training epoch is composed of so-called episodes and sub-sets of training data are called support sets. At the end of the episodes, the classification performance of a query set, based on knowledge gained from the respective support set, is evaluated to fine-tune the model parameters. The tuning of DNN parameters is solved by a prototypical approach. Each class is represented by a prototype feature vector and the weights of the feature extractor DNN will be tuned to separate the sample features from other prototypes the most. The superiority of this training approach is compared to other traditional classification models by evaluations on a textile dataset.

3. Proposed Methods

What we have learnt in the previous section is that different ideas were implemented to improve the performance of DNNs for the detection of defects. Such were: feature pyramids [5], deformable convolutions [4], separated multiple classifiers for classes (MG-CNN) [6], SVMs for the classification of DNN features [30]. Few-shot learning was attacked by label propagation [33,35] and prototyping approaches [32,37]. Now we show that state-of-the-art deep neural networks are capable of an almost-perfect classification of steel surface defects on the two benchmark datasets. Then, we discuss the problem of few-shot learning and introduce our proposed architecture.

3.1. Classification of Defects with EfficientNet-B7

There is a large number of DNNs for object recognition or classification and to compare their performance different benchmark datasets are created, such as ImageNet [38], MNIST [39], CIFAR-10, and CIFAR-100 [40]. If one network has high performance on some dataset it has high-probability for good performance on others, however, the different characteristics of data from different domains does not guarantee this.

In 2019, Google Brain published open source EfficientNet [28] as a family of image classification models to achieve state-of-the-art accuracy, yet still being an order-of-magnitude smaller and faster than previous models. The members of the family are the differently scaled versions (from B0 to B7) of a base model. The largest variant (B7) achieved state-of-the-art top-one accuracy on ImageNet in 2019, and it could reach the same performance as the previous state-of-the-art model but being 8.4 times smaller and 6.1 times faster during inference. In ref. [28], a compound scaling method was introduced to scale the depth (number of layers), the width (number of kernels in a layer), and resolution (size of input image) of an existing model and a baseline network with fine-tuned layers in a balanced manner considering the computation limits. They followed the idea that a small model can be easily fine-tuned on a small problem, and that a systematic scaling, changing only the dimensions but not the main structure and operation of layers, should result in an efficient network. Compound scaling proposes the right ratio between dimensions, while the baseline network (EfficientNet-B0) was designed with the help of optimization [41], considering both accuracy and computational cost on ImageNet.

The building blocks of the base EfficientNet model, obtained through a neural architecture search, are the so-called mobile-inverted bottleneck [42] layers, originally developed to be used as mobile-size networks (networks able to operate in mobile and embedded devices). These blocks use well-proven existing techniques, such as residual connections and depth-wise separable convolutions, and on the other hand also apply the special-techniques-inverted residuals and a linear bottleneck. The interested readers can find details about these in ref. [42].

Since its introduction, there is a very wide range of applications where EfficientNet or its variants showed very good accuracy, e.g., mushroom recognition [43], skin disorder recognition [44], iris recognition [45,46], forest fire detection [47], steganalysis [48], or plant disease detection [49], just to mention a few recent articles. Our experiments, detailed in Section 5, show that EfficientNet-B7 outperforms all previously published architectures for both NEU and X-SSD datasets. Moreover, in the next subsection we use EfficientNet-B7 (pre-trained on ImageNet) as the backbone of our proposed architecture for few-shot learning.

3.2. Few-Shot Learning of Defects

As automation is spreading widely in manufacturing processes, the need for automatic anomaly detection is growing. However, if we trained our machine intelligence for a given task, there is always a non-zero probability that unseen events might happen that the system is not trained to handle yet. A part of this problem is few-shot learning, where new kinds of errors appear to be classified as soon as possible, typically with very low number of training samples. We have to carry out incremental learning, since previously trained knowledge should not be forgotten. Moreover, the re-training of the whole architecture could be resource-demanding: the large amount of time, memory, and processing power is typically not available on site or in time.

In a new approach, we answer these challenges by the combination of two very efficient neural network architectures. The solution, illustrated by Figure 3, is composed of two parts:

1. The backbone is a deep neural network with an output feature vector of relatively large dimension (1024). We have chosen EfficientNet-B7 for this task due to its design for optimal size, structure, and accuracy. The task of the backbone is to adapt from ImageNet weights to steel surface defects with high accuracy with conventional training;

2. The back-end structure, for the further processing of extracted features, is a two-layer neural network where the first layer contains random weights. Since the backbone is previously fine-tuned to a set of known classes of steel surface defects, the task of this back end is to quickly learn the new classes based on the features of the backbone.

The reasoning of using this architecture is as follows:

- EfficientNet-B7 is very efficient for the classification of steel surface defects (see Section 5). Near its output, it still has a lengthy (hopefully rich) feature vector used by our back end;
- The back end has a random layer responsible for the generalization of fine-tuned feature values suitable for learning unseen classes;
- Since the back end has only one layer to be trained, it can be explicitly computed with algebraic computations without a lengthy backpropagation method. These computations give optimal solutions in the least-squares sense.

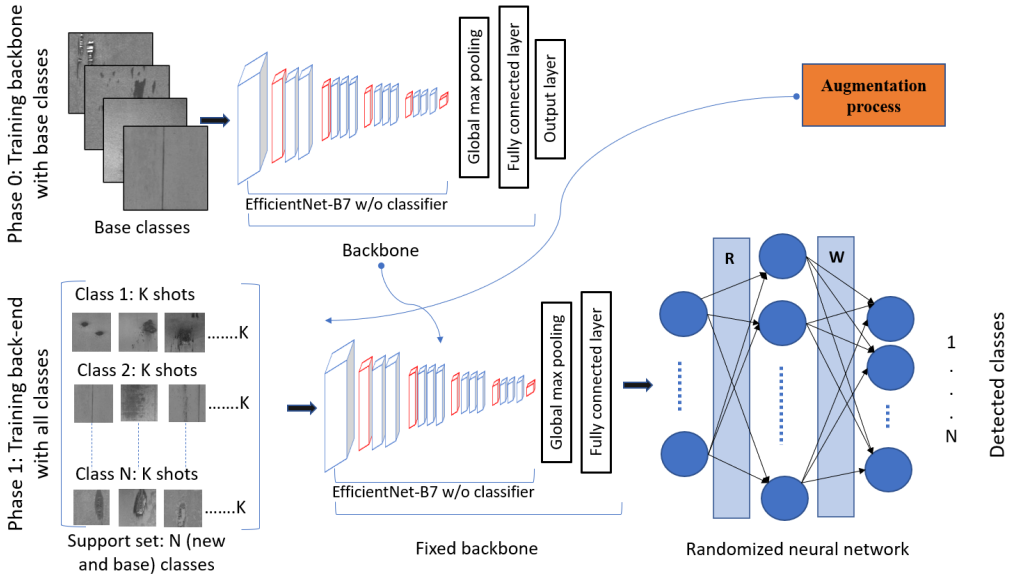

Figure 3. The proposed architecture (EffNet+RC) for few-shot learning. In phase 0, we train the backbone through a large number of samples of base classes. In phase 1 (and further phases), we use features extracted by the previously trained backbone. Here, only the weights W are computed with the help of a few samples, while weights R are random and fixed.

3.2.1. Randomized Weights for Generalization and Fast Tuning

It is well known that randomizing weights in neural networks can result in improved accuracy. In ref. [50], the input data of a single (hidden)-layer feed-forward neural network (SLFN) were weighted with random weights. Then, in the output layer, a fully connected network with bias was applied, where its weights could be calculated by solving a linear set of equations by standard numerical methods. In ref. [51], the random vector functional links network (RVFL) was proposed, where beside the input patterns, their randomized version is also generated and fed to the output using the standard weighted connections. Extreme learning machines (ELM) [52] have subtle variations to these randomized networks (no direct connection between inputs and outputs like RVFL, other usage of bias than in ref. [50]) but became more popular recently, in spite of reports that RVFL gives better performance than ELM in some cases [53]. For more information, we propose to read ref. [54], a review on neural networks with random weights.

A formal description of the applied randomized back-end network follows. Consider a set of N distinct training samples (x_i, y_i), $i = 1, \ldots, N$. Then, a SLFN with L hidden neurons has the following output equation:

$$\mathbf{t}(x_i) = \sum_{j=1}^{L} \mathbf{w}_j \phi(\mathbf{r}_j \mathbf{x}_i + b_j), \tag{1}$$

where ϕ is a sigmoid function, $\mathbf{r}_j$ are the random- and fixed-input weight vectors, b_j are the biases, $\mathbf{w}_i$ are the output weight vectors to be tuned, and $\mathbf{t}$ is the target vector (the outputs for the different classes). $\mathbf{R}$ and $\mathbf{W}$ in Figure 3 correspond to the weights here. Now, let us see how to compute $\mathbf{W}$ if $\mathbf{R}$ is set randomly. In practice, closed-form solutions are used to find $\mathbf{w}_i$ in a matrix form. Thus, we can shorten the equation:

$$\mathbf{T} = \mathbf{H}\mathbf{W}, \tag{2}$$

as the outputs of all hidden neurons are gathered into the matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} \phi(\mathbf{r}_1 \mathbf{x}_1 + b_1) & \cdots & \phi(\mathbf{r}_L \mathbf{x}_1 + b_L) \\ \vdots & \ddots & \vdots \\ \phi(\mathbf{r}_1 \mathbf{x}_N + b_1) & \cdots & \phi(\mathbf{r}_L \mathbf{x}_N + b_L) \end{bmatrix}, \tag{3}$$

given $\mathbf{W} = \left(\mathbf{w}_1^T \cdots \mathbf{w}_L^T\right)^T$, and $\mathbf{T} = \left(\mathbf{t}_1^T \cdots \mathbf{t}_N^T\right)^T$.

A unique solution for this system can be given by using the Moore–Penrose generalized inverse (pseudoinverse) [55] of the matrix H, denoted as $\mathrm{H}^{\dagger}$. From Equation (2):

$$\mathbf{W} = \mathbf{H}^{-1}\mathbf{T}. \tag{4}$$

To find the "best fit" (least squares) solution to the system of linear equations the pseudoinverse is computed [55]:

$$\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T. \tag{5}$$

Finally, we get the weights:

$$\mathbf{W} := \mathbf{H}^{\dagger}\mathbf{T}. \tag{6}$$

This explicit calculation of weights enables us to achieve an instantaneous fine-tuning of the architecture for new incoming classes in incremental learning.

3.2.2. Randomizing EfficientNet Features

Unfortunately, a neural network with only one hidden layer cannot cope with deep learning networks regarding accuracy. Besides, we would like to solve the problem of few-shot learning, where we have not enough information to train our network to the new classes. What we can do, is to transfer some information from the same domain, generalize it, and use this information for the classification of the new classes. For this transfer, we propose to use the randomized back-end network to generalize previously learnt information (indirectly the feature extractors) stored in a deep backbone network.

Basically, this can be considered as a multi-phase incremental learning mechanism. In phase 0, we fine-tune a large network (based on EfficientNet-B7) with lots of samples of the base classes. This backbone network is constructed by leaving the classification block of EfficientNet-B7, adding a global max-pooling layer, and two fully connected layers. The first has 1024 neurons, the second is the fully connected output layer with the number of classes to be trained. After training it with the available samples, this network, without the output layer, is frozen and used in the next phase as the backbone. Phase 1 is an incremental step, where we fine-tune only the back-end randomized network given the

1024 feature vectors as input. Naturally, phase 1 can be repeated many times, as new shots (or classes) appear.

More formally, suppose we have the set of classes $C = \{C^B, C^1, \ldots, C^P\}$, where C^i denotes the new classes (or equivalently class labels) appearing at step i, P is the maximum number of steps in incremental learning, and B denotes the base classes with lots of samples available ($C^B = C^0$, and preferably $||C^B|| \gg ||C^i||$, for $i \neq B$). Correspondingly, $D^B, D^1, \ldots, D^P$ represent the training datasets, where $D^i = (x_s^i, y_s^i)$ ($s = 1, \ldots, N^i$), and x_s^i is the s-th example in training phase i, and y_s^i is its corresponding class label.

In phase 0, we train the backbone model for classes C^B. In later phases, we create a support set $D^i = (x_s^i, y_s^i)$, such that $y_s^i \in \bigcup\{C^l\}, l \leq i$.

While Algorithm 1 gives the pseudo-code of this process, parameters are given in Section 6. Please note that the incremental phase (or phases) can introduce new classes, not only shots.

Algorithm 1: Incremental training with few shots with randomized EfficientNet features.

Input: Training data in phases: $D^0, D^1, D^2, \ldots, D^P$; EfficientNet-B7 network; randomized SLFN classifier

Output: Trained backbone and back-end randomized classifier models

```
i = 0
Phase i: Train EfficientNet-B7 for base classes C^i by samples in D^i
Cut top layer from EfficientNet-B7 to create backbone
while i ≤ P do
    i = i + 1
    Randomly sample D^i and make the augmentation of samples
    Use the pre-trained backbone to extract latent features for x_s^i
    Train the randomized SLFN as given in Equation (6)
```

The combination of EfficientNet-B7 backbone and the randomized classifier (RC) is named **EffNet+RC** in our article.

4. The Benchmark Datasets

There are several steel surface defect datasets available for the benchmarking of algorithms. First, we mention GC10-DET [56], which contains 3570 gray-scale photos of steel surfaces, including 10 kinds of defects. Another new dataset, published in ref. [57], consists of 21,853 RGB images showing areas with and without failures called pitting, while in ref. [57], only two classes are specified—this dataset shows the progression of failures at different time moments, which can be very useful for those who want to detect failures as soon as possible.

To test EfficientNet-B7, we used the two benchmark datasets already illustrated in Figures 1 and 2 and described in the following subsections. The reason for choosing them, beside the relatively large number of classes and large variance in appearance of these errors, is the wide availability of the concurrent methods used in the evaluations. Since we found X-SSD more challenging (see the experimental section below), we analyze our few-shot learning technique only on the X-SSD.

4.1. Northeastern University Surface Defect Database

NEU [2] consists of six different kinds of surface defects. Images, with resolutions of 200 × 200 pixels and having only one channel, were collected from hot-rolled steel strips. It contains 1800 images: each defect class has 300 samples. The classes, illustrated by Figure 1, are: crazing, inclusion, patches, pitted surface, rolled-in scale, and scratches. Unfortunately, ref. [2] does not contain information about how the defects are generated during the production of the hot-rolled steel strips. The dataset was divided randomly into 80% for training and 20% for testing.

Beside NEU, there is a dataset called NEU-DET [3] with the annotated bounding boxes of defects. This is out of focus from our perspective now.

In the tests of EfficientNet-B7, images were resized to 128 × 128 pixels (without a loss of accuracy, as will be shown later).

4.2. Xsteel Surface Defect Dataset (X-SSD)

The work [1] introduced a dataset for hot-rolled steel surface defects with 1360 images. Each image has a resolution of 128 × 128 pixels and has three color channels. This dataset, as illustrated in Figure 2, consists of seven different defect classes: 397 red iron sheet (Ri), 122 iron sheet ash (Is), 238 inclusions (Si), 134 surface scratches (Ss), 203 finishing-roll printing (Fr), 63 oxide scale-of-plate system (Op), and 203 oxide scale-of-temperature system (Ot). Paper [1] gives some information how the different defects are generated during the production of the hot-rolled steel strips:

- Red iron sheet: high silicon content in steel and high heating temperature of slab;
- Iron sheet ash: accumulated contamination (e.g., dust, oil) falls onto the surface;
- Inclusions: inclusion of slags in the steel;
- Surface scratches: hot-rolling area with projections—dead or passive rolls can cause friction on the surface;
- Finishing-roll printing: the slippage between the work roll and the support roll can result in dot and short-strip damages on the surface of the work roll;
- Oxide scale-of-plate system: if the roller table is damaged it can also damage the surface of the rolled piece where the iron oxide particles can accumulate and they can be rolled into the steel in the subsequent rolling process;
- Oxide scale-of-temperature system: it can be caused by many things, such as improper temperature settings, high carbon content, and unwanted intense oxidation.

5. Experimental Results of Classification and Its Discussion

In case of both datasets, we used the same hyper-parameters: The training was running for a maximum of 200 epochs and we used early stopping to avoid over-fitting. For the loss function we have chosen categorical cross-entropy, the learning rate was set to 0.0001 in all cases.

To enrich the dataset we applied traditional augmentation by applying the following online image transformations:

- Random rotations between 0° and 30°;
- Vertical flipping;
- Horizontal flipping;
- Zooming randomly between 0 and 20% in size.

The description of our hardware and software configuration is given in Table 1.

Table 1. Running and testing environment.

OS	CPU	GPU	Keras	Python
Ubuntu 18.04	Intel(R) Xeon(R) Gold 5115 CPU @ 2.40 GHz	NVIDIA Quadro P6000 GPU with 24 GB RAM	2.3.0	3.6.9

5.1. Classification Results on the NEU Dataset

To evaluate the performance of classification, we compare EfficientNet-B7 with several state-of-the-art algorithms. Training and testing sets were defined as given in Section 4. Previously, the best known accuracy on the NEU dataset was achieved by a variant of VGG16 [10], now denoted as ExtVGG16 (since it has several extra fully connected classification layers). Table 2 shows the comparison of all known alternatives. All results are very close to perfection: EfficientNet-B7 resulted in 100% accuracy, as well as ExtVGG16. Although, the number of parameters of ExtVGG16 is 53 Million compared to the 66 Million of EfficientNet-B7.

Table 2. Comparison of the classification accuracy of different models on the NEU dataset. If not specified then information is based on [29]. Training/testing ratio is 80/20 in general. Best values are highlighted in bold.

Model	MMGCN [1] [33]	SBF-Net	ResNet50 +MFN	Res-Net50	MVM -VGG	Res-Net34	Decaf	VSD [2] [8]	ResNet43 +MFN	AECLBP	OVERFEAT	Classic ResNet50	BYEC	ExtVGG16 [10]	EfficientNet-B7
Accuracy	99.72%	99.72%	99.67%	99.67%	99.5%	99.33%	99.27%	89.17%	99.17%	98.93%	98.7%	98.67%	96.3%	**100%**	**100%**

[1] A quantity of 40% of images used for testing, 60% for training. [2] A total of 50 images per class used for testing, remaining images for training.

5.2. Classification Results on the X-SSD Dataset

In the case of X-SSD, we followed the same steps as described in refs. [1,10] regarding the division for training and testing: 70% (952) images were used for training and the remaining 30% (408) for testing. The augmentation was identical to the above described, as well as other parameters of the training process.

The best previous results were also produced by ExtVGG16 [10] (a little below 100%), and could be now beaten slightly with EfficientNet-B7. Table 3 shows, beside accuracy, the computed macro-precision, macro-recall, and macro-F1 values for 14 different methods, including EfficientNet-B7. Macro-values refer to the averaging of class averages. We also included the confusion matrix and a misclassified example for EfficientNet-B7 in Figure 4.

Table 3. Comparison of EfficientNet-B7 with other models on X-SSD (all data are based on ref. [1] except for ExtVGG16 [10] and EfficientNet-B7). Best values are highlighted in bold.

Model	EspNet-v2	GhostNet	ShuffleNet	SqueezeNet	Xception	VGG16	ResNet50	ResNet101	ResNet152	RepVGG B1g2	RepVGG B3g4	RepVGG B3g4+SA	ExtVGG16	EfficientNet-B7
Accuracy	89.95%	88.72%	87.50%	91.42%	90.44%	92.65%	93.87%	87.01%	92.16%	88.97%	91.67%	95.10%	99.00%	**99.26%**
Macro-recall	84.19%	87.87%	85.84%	83.21%	87.39%	90.46%	89.41%	88.30%	89.41%	82.04%	85.28%	93.92%	98.00%	**98.71%**
Macro-precision	88.28%	86.93%	84.83%	90.36%	89.41%	91.70%	93.45%	88.18%	91.41%	90.79%	88.46%	95.16%	99.00%	**99.14%**
Macro-F_1 score	84.28%	87.07%	84.68%	84.15%	88.25%	90.92%	90.02%	87.05%	89.92%	81.58%	84.94%	93.25%	98.57%	**99.00%**

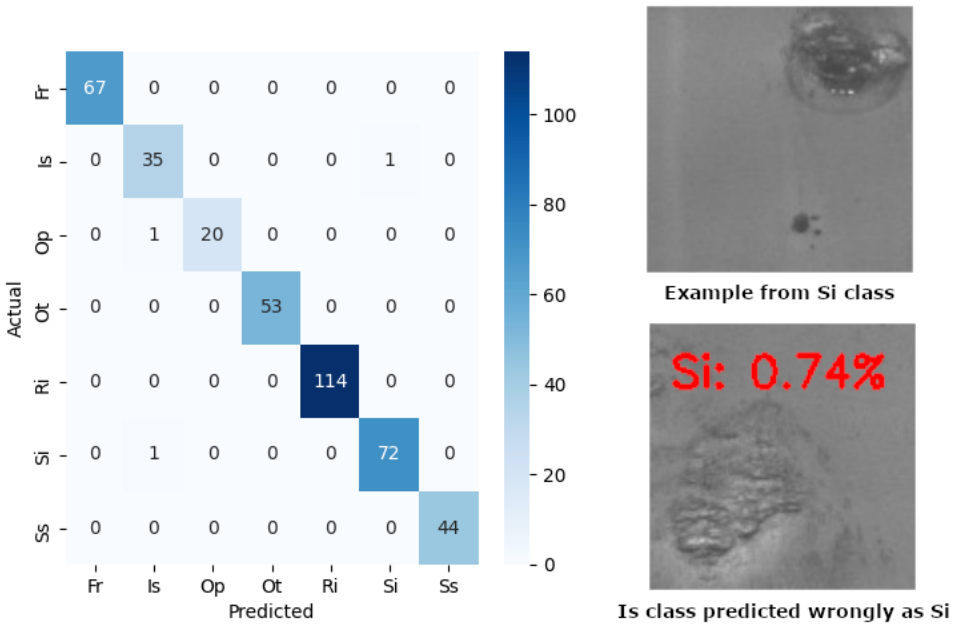

Figure 4. (**Left**) Confusion matrix of EfficientNet-B7 classification of the seven error types of the X-SSD dataset. (**Right**) An example image of the Si class and an Is defect wrongly classified as Si.

We can conclude that the NEU dataset seems to be too easy since many methods reached over 99% accuracy and both ExtVGG16 and EfficientNet-B7 could result in perfect classification. There is a larger gap between the accuracy of the different methods on X-SSD, where EfficientNet-7 showed the best accuracy.

6. Testing Few-Shot Learning with EffNet+RC

In the experiments we are implementing two phases of class increments. In phase zero, we train four classes of artifacts (finishing-roll printing, oxide scale-of-temperature system, red iron sheet, inclusion), then, in phase one, we are testing both the four base and the three new classes (iron sheet ash, scratches, oxide scale-of-plate system). In further phases we do not increase the number of classes, only the number of shots (K). When configuring the datasets for training and testing the few-shot incremental models, we had to make a different setup than for the previous classifications. Since in one-shot learning there is no sample for the validation of the learning process, so we cannot apply an early stopping mechanism, we run each process for 100 epochs, and alternatively, for 200 epochs. Table 4 summarizes how X-SSD was cut into parts for training and testing purposes for the 4 base and the 3 incremental classes.

Table 4. The distribution of original X-SSD images in the training and testing datasets for few-shot learning. K is the number of shots in the experiments. The real number of images fed to the network during training is larger due to augmentation.

Model	**All Images of X-SSD: 1360**			
	Training: 1092		**Testing: 268**	
	Base Classes: 835	**New Classes: 257**	**Bases Classes: 206**	**New Classes: 62**
EffNet backbone	All	None	None	None
EffNet+RC	K shots	K shots	All	All
EffNet+FtC	K shots	K shots	All	All
Ft EffNet	K shots	K shots	All	All
Ft EffNet Unbal	All	K shots	All	All

When training any variant of EfficientNet we applied the same random augmentation of images in each epoch as it was described before. Since **EffNet+RC** does not have epochs we applied augmentation to have 700 extra training images to build up matrix **H** in Equation (3).

To evaluate the accuracy of the proposed architecture (**EffNet+RC**), we compared it with different alternatives. Since EfficientNet-B7 showed the highest accuracy for the classification of the seven classes, it was natural to use it as a reference. The following variations are compared:

- **EffNet+RC**: Fixed EfficienNet-B7 backbone (in phase zero, see Figure 3), plus randomized classifier. To train this network we can use the explicit formula of Equation (6);
- **EffNet+FtC**: This network only differs from **EffNet+RC** in that instead of random weights, backpropagation fine-tuned weights are used in the classifier, and the backbone is still frozen see Figure 5. The purpose of this network is to learn the effect of randomization (when compared to **EffNet+RC**). We ran the training for 100 and 200 epochs;
- **Ft EffNet**: Fine-tuned EfficienNet-B7 by few shots (being augmented). To keep the fine-tuning dataset balanced the ratio of base classes is the same as of the new ones;
- **Ft EffNet Unbal**: Fine-tuned EfficientNet-B7 with unbalanced data. The same as above, but possibly all samples from the base classes were used in fine-tuning. This means unbalanced training, since new classes were sampled only by the few shots.

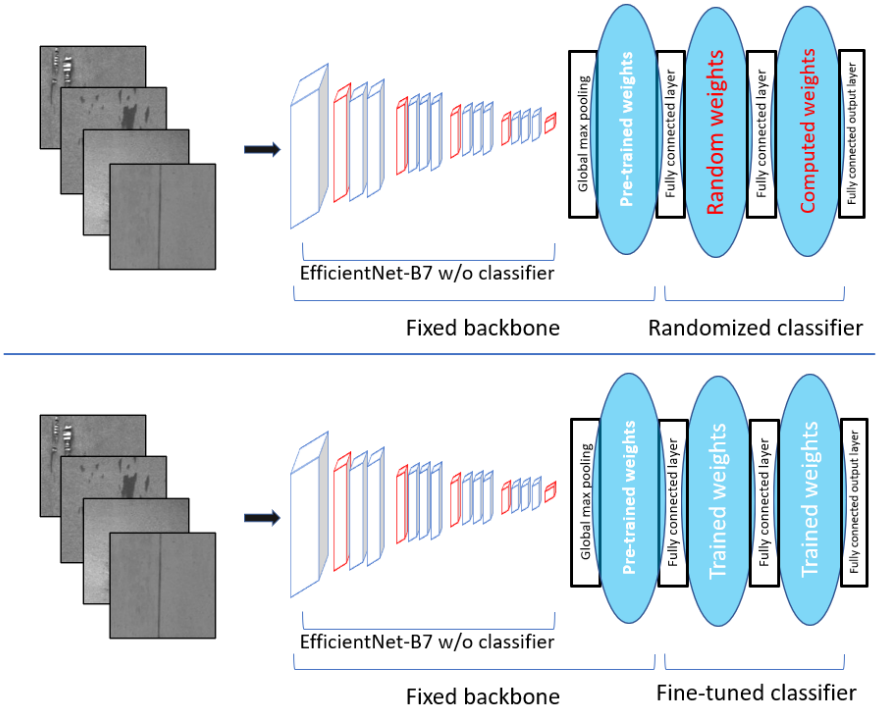

Figure 5. The illustration of the **Effnet+RC** and the **EffNet+FtC** networks. While the structures are similar, there is a big difference as the former should not be trained with backpropagation.

The accuracy, measured on all (base and new), and also separately on base and new classes, are shown in Figures 6–8, correspondingly. To investigate the effect of increasing the number of shots, K was set to 1, 3, 5, 10, 15, and 20. All experiments were repeated five times, randomly choosing the training shots, and averaging the results. All test images were used in the accuracy measurements (as indicated in Table 4).

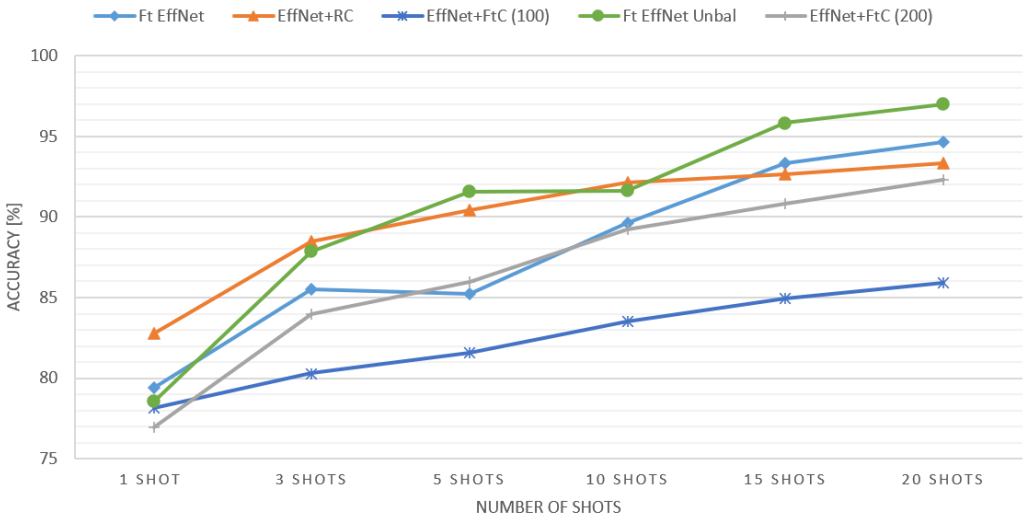

Figure 6. The accuracy of the different classification models for all classes (base and new). Ft EffNet: fine-tuned EfficienNet-B7; EffNet+RC: fixed EfficienNet-B7 backbone plus randomized classifier; EffNet+FtC: fixed EfficienNet-B7 backbone plus fine-tuned classifier; Ft EffNet Unbal: fine-tuned EfficienNet-B7 with unbalanced dataset.

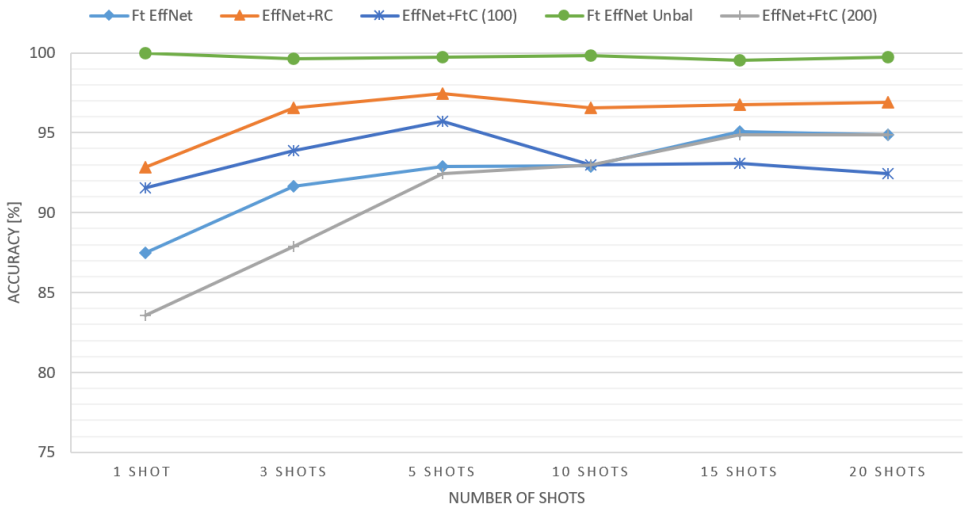

Figure 7. The accuracy of the different classification models for base classes. Ft EffNet: fine-tuned EfficienNet-B7; EffNet+RC: fixed EfficienNet-B7 backbone plus randomized classifier; EffNet+FtC: fixed EfficienNet-B7 backbone plus fine-tuned classifier; Ft EffNet Unbal: fine-tuned EfficienNet-B7 with unbalanced dataset.

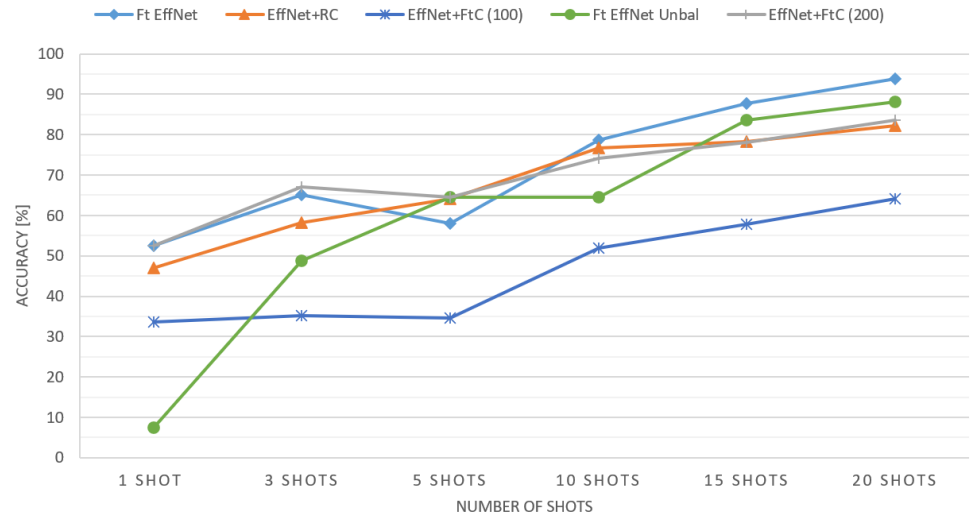

Figure 8. The accuracy of the different classification models for new classes. Ft EffNet: fine-tuned EfficienNet-B7; EffNet+RC: fixed EfficienNet-B7 backbone plus randomized classifier; EffNet+FtC: fixed EfficienNet-B7 backbone plus fine-tuned classifier; Ft EffNet Unbal: fine-tuned EfficienNet-B7 with unbalanced dataset.

When talking about few-shot incremental learning, in some scenarios, only few samples from the new and base classes are available for re-training. In our tests, except for **Ft EffNet Unbal**, the same number of shots were used from all classes for re-training. One would first expect that **Ft EffNet** could have the highest accuracy, since the whole network is re-trained, but this is not the case. While it can quickly adapt to the new classes, it gives the worst performance for the old ones (see corresponding curves in Figures 7 and 8). This is similar to catastrophic forgetting, and the drop from almost 100% to 87.5–93% is significant and the worst among all. Contrarily, if we involve more samples from the base classes to the re-training (and thus generate an unbalanced training dataset), the performance on old classes remains almost 100% but gives quite bad results for one to three shots of the new classes. It is illustrated by the green curve (**Ft EffNet Unbal**) in Figure 8.

Now, compare the proposed **EffNet+RC** with **EffNet+FtC**. These two networks are similar, except for the fact that the latter does not have a randomized fully connected layer,

but all of its weights are trained by backpropagation (see Figure 5 for illustration). As can be read out from Table 5 and Figure 6, **EffNet+RC** outperformed not only **EffNet+FtC (100)** and **EffNet+FtC (200)** but all other models at $K = 1, 3, 10$. At a higher number of shots, it still remained in the mid-range. Our proposed randomized model behaved quite well for the old classes; meanwhile, it could nicely follow the best curve of the fine-tuned EfficientNet-B7 (**Ft EffNet**) for the new ones.

Table 5. The accuracy and rank of the different classification models evaluated on all classes (base and new). Our values are highlighted in bold.

Method	1 Shot		3 Shots		5 Shots		10 Shots		15 Shots		20 Shots	
	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
EffNet+RC	**82.75%**	1	**88.50%**	1	90.39%	2	**92.14%**	1	92.62%	3	93.31%	3
EffNet+FtC (100)	78.13%	4	80.29%	5	81.56%	5	83.50%	5	84.92%	5	85.89%	5
EffNet+FtC (200)	76.94%	5	83.95%	4	85.97%	4	89.25%	4	90.82%	4	92.31%	4
Ft EffNet	79.40%	2	85.51%	3	85.24%	3	89.62%	3	93.35%	2	94.62%	2
Ft EffNet Unbal	78.58%	3	87.83%	2	**91.56%**	1	91.63%	2	**95.81%**	1	**97.01%**	1

Beside the accuracy values Table 5, we included the weighted F_1 score and rank of the different classification models evaluated on all classes (base and new) in Table 6. The position of the proposed approach remained the same as was in Table 5.

Table 6. The weighted F_1 score and rank of the different classification models evaluated on all classes (base and new). Our values are highlighted in bold.

Method	1 Shot		3 Shots		5 Shots		10 Shots		15 Shots		20 Shots	
	F_1 Score	Rank	F_1 Score	Rank	F_1 Score	Rank	F_1 Score	Rank	F_1 Score	Rank	F_1 Score	Rank
EffNet+RC	**82.4%**	1	**87.00%**	1	89.20%	2	**92.20 %**	1	92.20%	3	92.6 %	3
EffNet+FtC (100)	76.8 %	4	77.4 %	5	78.6 %	5	82.4 %	5	84.6 %	5	85.6 %	5
EffNet+FtC (200)	78.75 %	2	86.00 %	3	85.5 %	3	90.5 %	3	91.5 %	4	92.5 %	4
Ft EffNet	78.20 %	3	85.8 %	4	83.4 %	4	89.8 %	4	92.6 %	2	94.80 %	2
Ft EffNet Unbal	70.8 %	5	86.2 %	2	**90.8 %**	1	90.8%	2	**95.8 %**	1	**96.8 %**	1

Time Complexity

While the average inference time for an image of size 128×128 is around 0.06 s, the time complexity of re-configuring, re-training, or fine-tuning the classification methods can still cause problems in industrial applications. Stopping the production while waiting for adaptation to avoid new defects, or continuing the production with an increased ratio of false products, can both result in high costs. Due to these reasons, the extremely fast re-tuning of our model has a significant advantage over others. Table 7 and Figure 9 both contain the training times of the different models. **EffNet+RC** has constantly very low time requirements, independently from the number of shots, at least one order faster than **EffNet+FtC (100)**, the next fastest. The reason why **Ft EffNet Unbal** seems to be so slow is due to the large number of training images in each epoch $(835 + k(shots) \times 3(new\ classes))$, while in case of **Ft EffNet**, the training set contained far less images (since it was balanced).

Table 7. Time spent for training at different number of k-shots.

Method	1 Shot	3 Shots	5 Shots	10 Shots	15 Shots	20 Shots
EffNet+RC	2.80 s	3.01 s	3.03 s	3.04 s	3.30 s	3.50 s
EffNet+FtC (100)	29.24 s	29.49 s	45.67 s	53.9 s	67.41 s	78.09 s
EffNet+FtC (200)	39.53 s	39.83 s	69.70 s	88.29 s	114.18 s	136.35 s
Ft EffNet	151.80 s	159.80 s	193.43 s	243.85 s	293.50	347.23 s
Ft EffNet Unbal	1408.10 s	1429.50 s	1443.30 s	1473.20 s	1489.10 s	1496.60 s

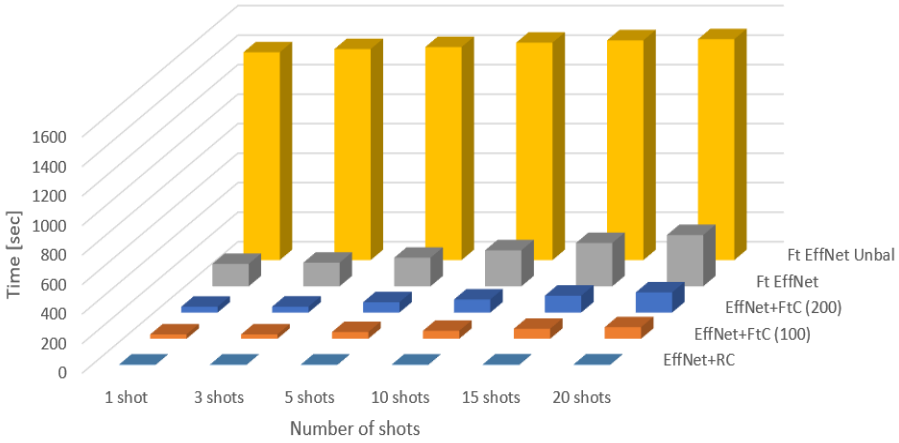

Figure 9. Elapsed training time of the different models under investigation.

7. Summary and Future Work

We can summarize the main contributions of our article in the following points:

1. We showed that state-of-the-art DNNs (namely EfficientNet-B7) can solve the classification of steel surface defects of the two often-used datasets almost perfectly, and that they were superior to other made-to-measure techniques;
2. We applied randomized networks, concatenated to the feature extraction of a pre-trained DNN, to give a solution for the few-shot problem. Since the computation of its weights can be done very efficiently by the Moore–Penrose generalized inverse, the solution has the following advantages:
    - Regarding very few shots (one to three), our model outperformed other variants when classifying both old and newly appearing classes of steel surface defects;
    - Regarding the base classes, catastrophic forgetting could be avoided;
    - The fine-tuning for new classes or shots is significantly faster than the fine-tuning of any conventional DNNs.

Our statements are validated with thousands of experiments on steel surface defects of the NEU and X-SSD datasets. In future, we plan to further improve the model by sub-network extensions, similar to ref. [24], and to investigate the model using other kinds of data.

Author Contributions: A.M.N. and L.C. contributed equally in this research. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partly implemented by the TKP2020-NKA-10 project with the support provided by the Ministry for Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the 2020 Thematic Excellence Programme funding scheme. We acknowledge the financial support of the Hungarian Scientific Research Fund grant OTKA K 135729. Amr Mohamed Nagy Abdo is funded by a scholarship under the joint Stipendium Hungaricum Programme between the Arab Republic of Egypt and Hungary.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We are grateful to the NVIDIA corporation for supporting our research with GPUs obtained by the NVIDIA GPU Grant Program.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RC	Randomized Classifier
X-SSD	Xsteel Surface Defect dataset
NEU	Northeastern University Surface Defect Database
DNN	Deep Neural Network
CNN	Convolutional Neural Network
YOLOv4	You Look Only Once Network Version 4
FPN	Feature Pyramid Network
R-CNN	Region-Based Convolutional Neural Network
RestNet	Residual Neural Network
MAP	Mean Average Precision
CPN	Classification Priority Network
MG-CNN	Multiple Group Convolutional Neural Network
DDN	Defect Detection Network
MFN	Multi-level Feature Fusion Network
ROI	Region of Interest
DA	Domain Adaptation
ACNN	Adaptive Learning Rate of the Convolutional Neural Networks
VGG19	Visual Geometry Group from Oxford 19
DeVGG19	Decode VGG19
SSIM	Structural Similarity Index Measure
GCN	Graph Convolution Networks
MMGCN	Multiple Micrographs Graph Convolutional Network
k-NN	k-Nearest Neighbor Graph
SENet	Squeeze-and-Excitation Networks
RN	Relation Network
CD-FSL	Cross-Domain Few-Shot Learning
BSR	Batch Spectrum Regularization
MNIST	Modified National Institute of Standards and Technology Database
CIFAR	Canadian Institute for Advanced Research
SLFN	Single Layer Feed-Forward Neural Network
RVFL	Random Vector Functional Links Network
ELM	Extreme Learning Machines
EffNet+RC	EfficientNet-B7 Backbone and the Randomized Classifier (RC)
VGG16	Visual Geometry Group from Oxford
ExtVGG16	Extended VGG16
EffNet	EfficientNet
EffNet+FtC	Frozen EfficientNet Backbone with Backpropagation Fine-Tuned Weights
Ft EffNet	Fine-tuned EfficienNet-B7
Ft EffNet Unbal	Fine-tuned EfficientNet-B7 with Unbalanced Data

Appendix A

Table A1. Steel surface detection methods discussed in Section 2.

Task	Title	Description	Dataset	Accuracy	MAP
Detection and classification methods					
	Detection of metal surface defects based on YOLOv4 algorithm [5].	Improvement of YOLOv4 architecture by adding a feature pyramid network (FPN) module after sampling, on the so-called neck part of the network.	NEU	92.5%	-

Table A1. *Cont.*

Task	Title	Description	Dataset	Accuracy	MAP
	A new steel defect detection algorithm based on deep learning [4].	Improved Faster R-CNN model by using deformable convolutions and multi-level feature fusion.	NEU-DET	-	75.2
	Defect detection of hot-rolled steels with a new object detection framework called classification priority network [6].	Classification priority network: two-stage classification.	Author dataset	96.00%	-
	An end-to-end steel surface defect detection approach via fusing multiple hierarchical features [3].	The processing goes through the steps: 1. feature map generation by ResNet; 2. multi-level feature fusion network; 3. region proposal network; 4. classifier and a bounding box regressor.	NEU-DET	99.67%	82.3
	Visual inspection of steel surface defects based on domain adaptation and adaptive convolutional neural network [7].	It combines domain adaptation and the adaptive learning rate of the convolutional neural networks to handle the changes during the production. Furthermore, applies an additional domain classifier and a constraint on label probability distribution to enable cross-domain and cross-task recognition, and to account for the lack of labels in a new domain.	NEU	99.00%	-
	A steel surface defect recognition algorithm based on improved deep learning network model using feature visualization and quality evaluation [8].	A steel surface defect classification technique fine-tuned with the help of a feature visualization network was proposed.	NEU	89.86%	-
	Recognition of scratches and abrasions on metal surfaces using a 534 classifier based on a convolutional neural network [9].	Different versions of ResNet were investigated to classify three kinds of defects on metal surfaces.	NEU	97.10%	-
	Classification of surface defects on steel strip images using convolution neural network and support vector machine [30].	A modified AlexNet for feature extraction, where the classification was solved with a support vector machine.	NEU	99.70%	-
	Zero-shot learning and classification of steel surface defects [10].	VGG16 was extended with several layers for classification (ExtVGG16).	NEU	100.00%	-

Table A1. *Cont.*

Task	Title	Description	Dataset	Accuracy	MAP
Zero-shot Learning					
	Zero-sample surface defect detection and classification based on semantic feedback neural network [27].	Word vectors, extracted from an auxiliary knowledge source, are used to solve the zero-shot learning problem.	Cylinder-liner defect dataset (CLSDD)	89.28%	-
	One-shot recognition of manufacturing defects in steel surfaces [26].	A Siamese neural network is used to decide whether two input samples belong to the same class or not.	NEU	83.22%	-
	Zero-shot learning and classification of steel surface defects [10].	Similar to ref. [26] but with a more efficient structure.	NEU	85.8%	-
Few-shot Learning					
	Steel Surface defect classification based on small sample learning [32].	Comparing ResNet, DenseNet, and MobileNet for feature extraction and mean subtraction and L2 normalization for feature transformation. For one-shot learning a nearest neighbor approach, whereas for the multi-shot setting the average value of known sample feature vectors are used.	NEU	92.33%	-
	A new graph-based semi-supervised method for surface defect classification [33].	A semi-supervised learning model called multiple micrographs graph convolutional network was proposed, which learns from both labeled and unlabeled samples.	NEU	99.72%	-
	Few-shot learning approach for 3D defect detection in lithium battery [35].	Training images were used to fine-tune a pre-trained ResNet10 model, then a k-NN graph is built to evaluate the distance between samples, including both the labeled and the unlabeled ones. To refine the scoring of samples a label propagation method was used.	Lithium batteries	97.17%	-
	Fabric defect classification using prototypical network of few-shot learning algorithm [37].	A meta-training prototype-based approach to handle the unbalanced data caused by the newly arriving error classes having only few shots. Training is cut into episodes, after each episode the model is fine-tuned.	Textile	99.72%	-

References

1. Feng, X.; Gao, X.; Luo, L. X-SDD: A new benchmark for hot rolled steel strip surface defects setection. *Symmetry* **2021**, *13*, 706. [CrossRef]
2. Song, K.; Yan, Y. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Appl. Surf. Sci.* **2013**, *285*, 858–864. [CrossRef]

3. He, Y.; Song, K.; Meng, Q.; Yan, Y. An end-to-end steel surface defect detection approach via fusing multiple hierarchical features. *IEEE Trans. Instrum. Meas.* **2019**, *69*, 1493–1504. [CrossRef]
4. Zhao, W.; Chen, F.; Huang, H.; Li, D.; Cheng, W. A new steel defect detection algorithm based on deep learning. *Comput. Intell. Neurosci.* **2021**, *2021*, 13. [CrossRef]
5. Zhao, H.; Yang, Z.; Li, J. Detection of metal surface defects based on YOLOv4 algorithm. *J. Phys. Conf. Ser. IOP Publ.* **2021**, *1907*, 012043. [CrossRef]
6. He, D.; Xu, K.; Zhou, P. Defect detection of hot rolled steels with a new object detection framework called classification priority network. *Comput. Ind. Eng.* **2019**, *128*, 290–297. [CrossRef]
7. Zhang, S.; Zhang, Q.; Gu, J.; Su, L.; Li, K.; Pecht, M. Visual inspection of steel surface defects based on domain adaptation and adaptive convolutional neural network. *Mech. Syst. Signal Process.* **2021**, *153*, 107541. [CrossRef]
8. Guan, S.; Lei, M.; Lu, H. A steel surface defect recognition algorithm based on improved deep learning network model using feature visualization and quality evaluation. *IEEE Access* **2020**, *8*, 49885–49895. [CrossRef]
9. Konovalenko, I.; Maruschak, P.; Brevus, V.; Prentkovskis, O. Recognition of scratches and abrasions on metal surfaces using a classifier based on a convolutional neural network. *Metals* **2021**, *11*, 549. [CrossRef]
10. Nagy, A.M.; Czúni, L. Zero-shot learning and classification of steel surface defects. In Proceedings of the Fourteenth International Conference on Machine Vision (ICMV 2021). International Society for Optics and Photonics, Rome, Italy, 8–14 November 2021; Volume 12084, pp. 386–394.
11. Chen, Y.; Ding, Y.; Zhao, F.; Zhang, E.; Wu, Z.; Shao, L. Surface defect detection methods for industrial products: A review. *Appl. Sci.* **2021**, *11*, 7657. [CrossRef]
12. Seff, A.; Beatson, A.; Suo, D.; Liu, H. Continual learning in generative adversarial nets. *arXiv* **2017**, arXiv:1705.08395.
13. Shin, H.; Lee, J.K.; Kim, J.; Kim, J. Continual learning with deep generative replay. *arXiv* **2017**, arXiv:1705.08690.
14. Parisi, G.I.; Kemker, R.; Part, J.L.; Kanan, C.; Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Netw.* **2019**, *113*, 54–71. [CrossRef] [PubMed]
15. Belouadah, E.; Popescu, A. DeeSIL: Deep-shallow incremental learning. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2018; pp. 151–157.
16. Castro, F.M.; Marín-Jiménez, M.J.; Guil, N.; Schmid, C.; Alahari, K. End-to-end incremental learning. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 233–248.
17. He, C.; Wang, R.; Shan, S.; Chen, X. Exemplar-Supported Generative Reproduction for Class Incremental Learning. In Proceedings of the 2018 British Machine Vision Conference, Newcastle, UK, 3–6 September 2018; p. 98.
18. Rebuffi, S.A.; Kolesnikov, A.; Sperl, G.; Lampert, C.H. iCarL: Incremental classifier and representation learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2001–2010.
19. Luo, Y.; Yin, L.; Bai, W.; Mao, K. An appraisal of incremental learning methods. *Entropy* **2020**, *22*, 1190. [CrossRef] [PubMed]
20. Boukli Hacene, G.; Gripon, V.; Farrugia, N.; Arzel, M.; Jezequel, M. Transfer incremental learning using data augmentation. *Appl. Sci.* **2018**, *8*, 2512. [CrossRef]
21. Ratcliff, R. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychol. Rev.* **1990**, *97*, 285. [CrossRef]
22. Belouadah, E.; Popescu, A. Scail: Classifier weights scaling for class incremental learning. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision,Snowmass, CO, USA, 1–5 March 2020; pp. 1266–1275.
23. Belouadah, E.; Popescu, A. Il2m: Class incremental learning with dual memory. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 583–592.
24. Siddiqui, Z.A.; Park, U. Progressive convolutional neural network for incremental learning. *Electronics* **2021**, *10*, 1879. [CrossRef]
25. Wang, Y.; Yao, Q.; Kwok, J.T.; Ni, L.M. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.* **2020**, *53*, 1–34. [CrossRef]
26. Deshpande, A.M.; Minai, A.A.; Kumar, M. One-shot recognition of manufacturing defects in steel surfaces. *Procedia Manuf.* **2020**, *48*, 1064–1071. [CrossRef]
27. Guo, Y.; Fan, Y.; Xiang, Z.; Wang, H.; Meng, W.; Xu, M. Zero-sample surface defect detection and classification based on semantic feedback neural network. *arXiv* **2021**, arXiv:2106.07959.
28. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 3 June 2019; pp. 6105–6114.
29. Schlagenhauf, T.; Yildirim, F.; Brückner, B.; Fleischer, J. Siamese basis function networks for defect classification. *arXiv* **2020**, arXiv:2012.01338.
30. Boudiaf, A.; Benlahmidi, S.; Harrar, K.; Zaghdoudi, R. Classification of surface defects on steel strip images using convolution neural network and support vector machine. *J. Fail. Anal. Prev.* **2022**, *V22* 1–11. [CrossRef]
31. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
32. Wu, S.; Zhao, S.; Zhang, Q.; Chen, L.; Wu, C. Steel Surface defect classification based on small sample learning. *Appl. Sci.* **2021**, *11*, 11459. [CrossRef]
33. Wang, Y.; Gao, L.; Gao, Y.; Li, X. A new graph-based semi-supervised method for surface defect classification. *Robot. Comput.-Integr. Manuf.* **2021**, *68*, 102083. [CrossRef]

34. Yang, L.; Zhan, X.; Chen, D.; Yan, J.; Loy, C.C.; Lin, D. Learning to cluster faces on an affinity graph. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 2298–2306.
35. Wu, K.; Tan, J.; Li, J.; Liu, C. Few-shot learning approach for 3D defect detection in lithium battery. *J. Phys. Conf. Ser. IOP Publ.* **2021**, *1884*, 012024. [CrossRef]
36. Zhou, D.; Bousquet, O.; Lal, T.; Weston, J.; Schölkopf, B. Learning with local and global consistency. *Adv. Neural Inform. Process. Syst.* **2003**, *16*.
37. Zhan, Z.; Zhou, J.; Xu, B. Fabric defect classification using prototypical network of few-shot learning algorithm. *Comput. Ind.* **2022**, *138*, 103628. [CrossRef]
38. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision* **2015**, *115*, 211–252. [CrossRef]
39. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
40. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Department of Computer Science, University of Toronto: Toronto, ON, Canada, 2009.
41. Kyriakides, G.; Margaritis, K. An introduction to neural architecture search for convolutional networks. *arXiv* **2020**, arXiv:2005.11074.
42. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
43. Kiss, N.; Czùni, L. Mushroom image classification with CNNs: A case-study of different learning strategies. In Proceedings of the 12th International Symposium on Image and Signal Processing and Analysis (ISPA), Zagreb, Croatia, 13–15 September 2021; pp. 165–170.
44. Hridoy, R.H.; Akter, F.; Rakshit, A. Computer vision based skin disorder recognition using EfficientNet: A transfer learning approach. In Proceedings of the 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 14–15 July 2021; pp. 482–487.
45. Ab Wahab, M.N.; Nazir, A.; Ren, A.T.Z.; Noor, M.H.M.; Akbar, M.F.; Mohamed, A.S.A. Efficientnet-lite and hybrid CNN-KNN implementation for facial expression recognition on raspberry pi. *IEEE Access* **2021**, *9*, 134065–134080. [CrossRef]
46. Garg, H.; Sharma, B.; Shekhar, S.; Agarwal, R. Spoofing detection system for e-health digital twin using EfficientNet convolution neural network. *Multimed. Tools Appl.* **2022**, 1–16. [CrossRef]
47. Xu, R.; Lin, H.; Lu, K.; Cao, L.; Liu, Y. A forest fire detection system based on ensemble learning. *Forests* **2021**, *12*, 217. [CrossRef]
48. Yousfi, Y.; Butora, J.; Fridrich, J.; Fuji Tsang, C. Improving efficientnet for JPEG steganalysis. In Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security, virtual event Belgium, 22–25 June 2021; pp. 149–157.
49. Gao, F.; Sa, J.; Wang, Z.; Zhao, Z. Cassava disease detection method based on EfficientNet. In Proceedings of the 7th International Conference on Systems and Informatics (ICSAI), Chongqing, China, 13–15 November 2021; pp. 1–6.
50. Schmidt, W.F.; Kraaijveld, M.A.; Duin, R.P. Feed forward neural networks with random weights. In Proceedings of the International Conference on Pattern Recognition, The Hague, The Netherlands, 30 August–3 September 1992; p. 1.
51. Pao, Y.H.; Park, G.H.; Sobajic, D.J. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing* **1994**, *6*, 163–180. [CrossRef]
52. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
53. Suganthan, P.N.; Katuwal, R. On the origins of randomization-based feedforward neural networks. *Appl. Soft Comput.* **2021**, *105*, 107239. [CrossRef]
54. Cao, W.; Wang, X.; Ming, Z.; Gao, J. A review on neural networks with random weights. *Neurocomputing* **2018**, *275*, 278–287. [CrossRef]
55. Moore, E.H. On the reciprocal of the general algebraic matrix. *Bull. Am. Math. Soc.* **1920**, *26*, 394–395.
56. Lv, X.; Duan, F.; Jiang, J.j.; Fu, X.; Gan, L. Deep metallic surface defect detection: The new benchmark and detection network. *Sensors* **2020**, *20*, 1562. [CrossRef]
57. Schlagenhauf, T.; Landwehr, M.; Fleischer, J. *Industrial Machine Tool Element Surface Defect Dataset*; Karlsruher Institut für Technologie: Karlsruhe, Germany, 2021.