applied sciences

MDPI

Article

Soldering Data Classification with a Deep Clustering Approach: Case Study of an Academic-Industrial Cooperation

Kinga Bettina Faragó [1,*], Joul Skaf [1,*], Szabolcs Forgács [2], Bence Hevesi [2] and András Lőrincz [1]

[1] Department of Artificial Intelligence, Faculty of Informatics, ELTE Eötvös Loránd University, Pázmány Péter Sétány 1/A, 1117 Budapest, Hungary; lorincz@inf.elte.hu
[2] Robert Bosch Ltd., Gyömrői út 104, 1103 Budapest, Hungary; szabolcs.forgacs@hu.bosch.com (S.F.); bence.hevesi@hu.bosch.com (B.H.)
* Correspondence: faragokinga@inf.elte.hu (K.B.F.); slnj33@inf.elte.hu (J.S.)

Abstract: Modern industries still commonly use traditional methods to visually inspect products, even though automation has many advantages over the skills of human labour. The automation of redundant tasks is one of the greatest successes of Artificial Intelligence (AI). It employs human annotation and finds possible relationships between features within a particular dataset. However, until recently, this has always been the responsibility of AI specialists with a specific type of knowledge that is not available to the industrial domain experts. We documented the joint research of AI and domain experts as a case study on processing a soldering-related industrial dataset. Our image classification approach relies on the latent space representations of neural networks already trained on other databases. We perform dimensionality reduction of the representations of the new data and cluster the outputs in the lower dimension. This method requires little to no knowledge of the underlying architecture of neural networks by the domain experts, meaning it is easily manageable by them, supporting generalization to other use cases that can be investigated in future work. We also suggest a misclassification detecting method. We were able to achieve near-perfect test accuracy with minimal annotation work.

Keywords: visual inspection; industrial data; pre-trained models; image classification; deep clustering

Citation: Faragó, K.B.; Skaf, J.; Forgács, S.; Hevesi, B.; Lőrincz, A. Soldering Data Classification with a Deep Clustering Approach: Case Study of an Academic-Industrial Cooperation. *Appl. Sci.* **2022**, *12*, 6927. https://doi.org/10.3390/app12146927

Academic Editors: Shivam Kalra and Bing Li

Received: 20 May 2022
Accepted: 5 July 2022
Published: 8 July 2022

1. Introduction

Data availability has left a drastically increasing competition between companies across all domains. In order for businesses to make sense of their data, there comes the need to employ machine learning specialists or data scientists with the proper background knowledge. However, many phases of most machine learning problems can be efficiently automated. The automation of such repetitive tasks will eventually lead to the reduction of human effort which is one of the main goals of machine learning in general.

In the past few years, new methods that enable domain experts to work with AI-supported technologies have emerged, these techniques usually require no background knowledge by using software tools to connect the experts with their work directly. This is similar to the work being conducted in automated machine learning (AutoML). Recent studies investigated the efficiency of the AutoML techniques [1] applied by non-experts from various fields. Despite the easy-to-follow workflow, users expressed a need for a deeper understanding of the processing steps and results. Visualization is a straightforward way to overcome this issue, and it is also supported by studies [2,3].

Our objective is to introduce a user-friendly framework for domain experts that visually supports the understanding of the processing steps and leads toward proper evaluation of image classification and clustering tasks without requiring extensive knowledge of deep learning.

In this research, the close inspection and careful handling of the soldering-related industrial data were both prerequisites.

Visual inspection at each stage of the production line or in the quality control of the actual product is essential for the manufacturing efficiency of industries. The human workload is critical in this process since many industries still use conventional visual inspection methods and human labor instead of automation. When conventional approaches reach their limits, the commonly used automation concept is the contactless automatic optical inspection (AOI), which can have different hardware and software configurations, but most simply requires an image sensor (e.g., a camera), an illumination device, a computer for processing, a conveyor belt for transferring the products and an inspection algorithm [4].

The use of commercially available AOI systems in solder joint inspection processes is a commonly used standard method according to the work of Metzner et al. [5]. The authors revealed the limitations of such systems, which are rule-based implementation, general lack of flexibility to adapt to new cases, need for a high-level of domain knowledge for setup, and technological support during runtime. The research presented a deep learning supported solution, namely a convolutional neural network based classifier for classifying normal solder images and solder defects. After a detailed comparison and evaluation, the superior performance of the new method was concluded. As an outlook, the authors emphasized that obtaining the right amount and proportion of labeled training data is still one of the critical issues in the practical application of deep learning.

Deep learning has already been used to support automated industrial inspection in 2016, as shown in the publication by Weimer et al. [6]. Instead of the commonly used hand-designed image processing solutions, they introduced a deep convolutional neural network (CNN) for the automatic generation of powerful features. The work emphasized that general optical quality control methods are adapted to controlled environments and fail on complex textures or noisy data, their performance also relies on human capabilities. Since there is a large variation in the visual appearance of defects, a modern automated visual inspection system must be able to handle this. Since deep neural networks learn from examples, a significant amount of training data is required, but only minimal prior knowledge of the problem domain, meaning that no domain expert knowledge is needed to develop a CNN-based visual inspection system.

Another real-world example of the practical application of deep learning image classification in visual inspection is the detection of cracks in concrete surfaces [7]. The work presented a CNN-based approach with high accuracy classification performance and also showed that the domain knowledge of artificial intelligence specialists is essential to fine-tune the model appropriately.

The study by Dai et al. [8] proposed a novel solder joint defect classification framework that incorporates deep learning and is potentially applicable to real-world AOI scenarios of solder joint inspection tasks. The authors' motivation is that traditional inspection methods are still commonly used in modern factories, but their limitations make AOI techniques increasingly relevant. The paper describes in detail the three phases of AOI, which are image acquisition, solder joint localization, and classification.

The research of Wang et al. [9] has drawn attention to the growing importance of visual quality control of products in all kinds of manufacturing processes. The study also highlighted that many factories still use traditional methods instead of automation to detect defects in products, these methods are expensive and labor-intensive, but due to the natural limitations of workers, efficiency is highly dependent on fatigue. This research has made it clear that there is an urgent need for novel automated defect detection systems in manufacturing processes. The authors implemented a CNN for image classification and successfully validated the approach on a simple benchmark dataset of defect-free and defective images. However, the paper did not consider a real-world example the proposed solution was also a CNN model, similar to the work of Metzner et al. [5].

As previous works have shown, automating visual inspection is an essential step towards efficient production to ensure fast and reliable product quality control. The use of deep learning solutions leads to excellent performance, but the bottleneck to the

uptake of the technology is a large amount of training data or annotation and the lack of comprehensive knowledge of AI.

In our proposed pipeline, beyond minimizing the labelling efforts, the domain experts may also identify misclassifications by trying more than one evaluating network that we put forth here. Thus, the proposed pipeline supports domain experts to independently evaluate their data without the in-depth involvement of AI specialists. We developed a prototypical pipeline jointly with a software tool called NIPGBoard used to conduct visual inception. The tool builds on the data projection benefits of the well-known TensorBoard visualization toolkit of TensorFlow [10].

In contrast to the aforementioned works, one of our main goals was efficient clustering of soldering images for easier annotation, rather than filtering defective images. Due to their sheer number (≈75k) of samples in our database, it is difficult for the domain experts to label them by hand as it is a tedious and time-consuming process, meaning that an automated method is required. For this reason, we consider deep learning-based clustering techniques since they have become an influential trend in clustering methods and still have great potential. Another reason for considering these techniques would be that state-of-the-art deep learning architectures are available online with pre-trained weights on the ImageNet dataset [11]. Therefore, by utilizing these pre-trained models, it is possible to extract latent space representations (embeddings) from the network and train an independent algorithm to create clusters based on the features captured by these representations for any required dataset, indicating that the clustering representations of the data maybe be efficient independently from the database. The only condition is that the images have to be resized to fit the pre-trained model input shape. Training a model to fit these representations can be done with or without any labels, meaning it is possible to label a subset of the overall dataset, making this problem somehow like a semi-supervised problem instead of a fully unsupervised task. Both methods have their advantages and disadvantages, but in general, they both may work well. Nevertheless, for high accuracy, a ratio of the data needs to be manually annotated by the experts. This ratio depends on the required precision, dataset complexity, number of unique categories, variations within each class, and the dataset distribution.

Deep clustering approaches and representation learning-based clustering techniques have gained popularity over traditional clustering methods in the last few years and have shown remarkable performance in unsupervised representation learning [12].

Clustering is a fundamental unsupervised learning approach used in knowledge discovery and data science. Its principal purpose is to group data points based on their main attributes [13] without access to the ground truth labels or prior knowledge of the data distributions, categories, or the nature of the clusters. It is also defined as the unsupervised classification of patterns (observations, data items, or feature vectors) [14]. This means that the established groups could be much more than the required categories, as they can capture even the simplest variances within each class. On the other hand, supervised learning aims to find the correlation between the input and the output label by constructing a predictive model learning from a vast number of training examples, where each training example has a label indicating its ground truth output [15].

Many researchers have published state-of-the-art deep learning neural networks with trained weights [16–21] on benchmarks datasets such as the ImageNet dataset, which contains a thousands of classes.

We intend to use both techniques to join their advantages in a kind of semi-supervised learning [22]. Semi-supervised learning uses large amounts of unlabeled data and a small portion of labeled data to build better classifiers, meaning that it requires less human effort and gives higher accuracy [23] than unsupervised learning approaches. Therefore, we can attempt to find the minimal amount of labeled data that still produces high accuracy.

In our implementation, we do not attempt to train any network, as it would require large computational efforts and special knowledge that the domain experts do not possess. Therefore, we will try the second approach by directly employing the feature space and

using it in cluster construction to take advantage of the available pre-trained image classification convolutional neural networks. We train a Uniform Manifold Approximation and Projection (UMAP) model [24] to reduce the dimensionality of the problem, effectively establishing the clusters between these images based on their feature space. New implementations tend to treat the clustering and the dimensionality reduction tasks jointly to learn what is known as cluster-friendly representations. These representations have shown that optimizing the two methods together improves the performance of both [25].

Deep clustering techniques are powerful, but they usually suffer from several challenges. For instance, setting the optimal hyper-parameters for these techniques is not a trivial task [26] as the recommended parameters are mainly benchmark-based on datasets such as MNIST [27], and CIFAR-10 [28] that do not reflect the complexity of real-world tasks. Whereas in our adaptation, the network is already trained and fine-tuned by a professional group of deep learning specialists. Nevertheless, we still have some parameters to optimize concerning the dimensionality reduction algorithm, such as the number of neighbors and iterations used in UMAP. Fortunately, these parameters are reasonably limited and easy to understand. In addition, running the algorithm requires a modest amount of resources only. Reducing the size of the data to be processed is a simple way to overcome this limitation. A small subset may be sufficient for fitting the model to obtain a reasonable performance, such as in our case, but careful selection is required to balance the data.

The other major issue is the lack of theoretical background and interpretability [26], as explaining why and how these algorithms produce good results. Selecting the right pre-trained network for each dataset is also a challenge. It is often challenging to predict their behavior and performance for out-of-sample cases.

Our goal is to derive a method capable of effectively reducing the domain experts' workload by minimizing the number of samples that require manual annotation. The proposed pipeline also helps individuals with little or no experience with AI to work and achieve the desired results. The main idea relies on utilizing deep clustering representation learning using pre-trained networks to achieve high-quality clusters without the need for deep learning specialists' involvement. Also, by applying more than one network one may identify misclassifications and return them to experts for review and correction as we show. The introduced framework is based on a case study of a collaboration between AI specialists and domain experts on an image classification task on an industrial dataset.

2. Materials and Methods

In this section, we introduce the pipeline for domain experts to accomplish a general image classification task. The practical scenario was carried out on an industrial dataset as part of academic-industrial cooperation. We propose a solution that automates the image clustering process entirely, prepares the data for easier annotation, provides visual explanation, and gives domain experts the power to find data clusters independently, with high precision.

First, the domain experts select random data points from the overall dataset. In our case, approximately 8% of the dataset was selected in the process. The data needs to be examined and processed appropriately to ensure that all the images are unique and have an appropriate format. Following the first step, the pre-processed dataset is to be divided into a training and a test subset. Ordinarily, it is 80–20% as one of the most common splits suggested by the literature. However, the ratio often depends on the problem at hand, the complexity of the data distribution, and other factors. For our case this split was sufficient given that it is a binary classification problem with a relatively balanced dataset and limited within-class variations for the two categories. Other more elaborate methods, such as k-fold cross-validation, can be used to estimate the error rate more accurately. Nonetheless, the purpose of the research is to ease the works of the experts, and avoid adding more complicated evaluation criteria.

The second component is the selection of a pre-trained deep learning architecture to work with. Recently there has been an arsenal of free, online available, and easily integrated

pre-trained networks. These models include VGG16/19 [16], DenseNet [17], NASNet [18], Inception [19], InceptionResNet [19], Xception [20], EfficientNet [21], ResNet [29] and their variations.

Extraction of the representations of the train and test sets is done by feeding the sets to the pre-trained model to retrieve their corresponding feature space which we will also refer to as embedding.

Training and testing proceeds as follows. The latent representations of the train set are used to fit a UMAP model. This step can be accomplished with or without using the ground truth information. Using labels will improve the performance of the model by increasing the accuracy of the clusters. Nonetheless, labeling data is time and energy consuming, therefore a trade-off is required to achieve the highest possible accuracy with the minimal required manual annotation.

After obtaining the 2D or 3D embeddings produced by the UMAP model, the next step is the separation of the clusters from each other. In this processing phase, visual inspection can be performed using the NIPGBoard visualization tool.

The separation can be done by considering that all the data points form a fully-connected graph. The graph connections need to be minimized as the number of the edges is equal to $n(n-1)/2$. After minimizing the graph, we need to establish the boundaries between the clusters. This trained UMAP model is used to transform the latent representations of the test set into the UMAP space and the isolated clusters need to be checked independently by the domain experts by means of a few examples in each set.

Finally, one can use more than one evaluating network, compare the results and point to potential misclassifications that we detail later. Figure 1 shows the pipeline steps as a flowchart.

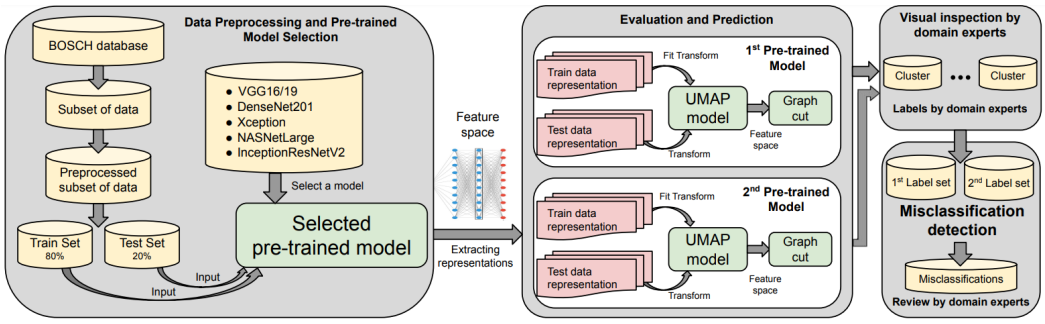


Figure 1. Pipeline for domain experts to evaluate their own dataset. On the left, the first stage can be seen as the Data Preprocessing and the Pre-trained Model Selection for further evaluation. The train and test subsets are fed into the selected pre-trained model for feature extraction. The second stage is the Evaluation and Prediction, where the UMAP method is applied on the train and test sets' representations independently, then a graph cut technique is used to define the most accurate clustering result. The next stage is the Visual Inspection by the domain experts, who supervise the results and reinforce the given labels. The final stage is Misclassification detection, where the two best performing pre-trained models classify the images and only those images that they disagree on are sent back to the domain experts for a final revision.

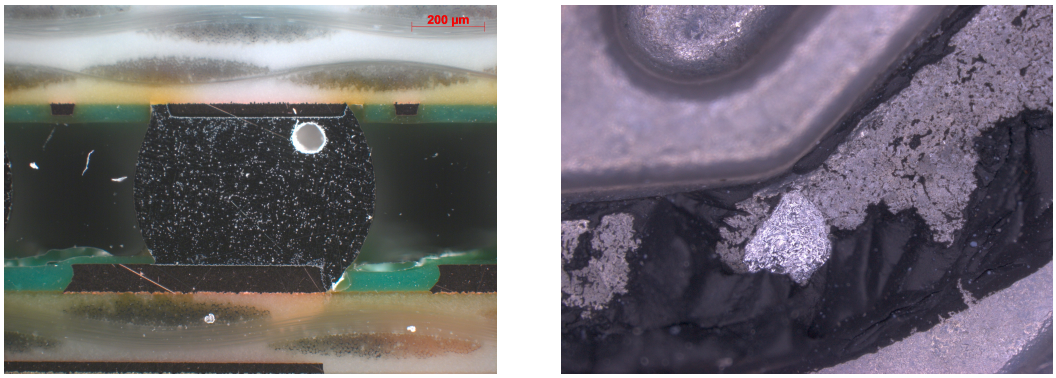
2.1. Industrial Dataset

The joint research focused on deep learning supported evaluation of a large sized, soldering-related industrial dataset, provided by Robert Bosch, Ltd. (Bosch). As a concrete industrial example, in electronics manufacturing the soldering process refers to joining two or more metallic items together by melting and putting a filler metal into the joint. This filler metal is called solder and has a lower melting point than the adjoining metals. Soldering is utilized in many household devices and in-vehicle equipment. The reliability of solder joints is generally evaluated after long environmental testing, which is part of the product development process. Several methods are used for this process, including

destructive techniques such as creating polished cross-sections and evaluating the internal structure of the solder joint.

This database represents cross-sections and other visual inspection images collected from different laboratories: pictures are taken under different lighting conditions, different camera parameters, and under different angles, indicating a difference in data distribution. The database does not have predefined rules about the capturing methodology. In addition, images are either captured using microscopes or ordinary photo cameras, showing electronic control units in whole or partially. These images happened to be collected over many years, are managed in a complex directory structure, and lack the proper labels.

Figure 2 shows example images from the Bosch dataset representing the two main image categories: *cross-section* samples and the so called other recordings. These two image group names refer to the target categories of the classification task.



(**a**) (**b**)

Figure 2. Example images from the Bosch industrial dataset (**a**) shows *cross-section* image example, (**b**) shows sample image with *other* label.

The data used in this task was randomly sampled from a set of a more comprehensive dataset (size of ≈75k). The domain experts had to randomly select images from the complete collection, which is stored in a complex directory structure. The selected data were manually renamed and labeled. This involved assigning each image one of the two given labels: either *cross-section* or *other*.

Even though there are only two distinct categories, each has multiple subclasses. The images in the other category are records of essential details and devices that are relevant in the Bosch laboratories. Such as parts and the boxes that contain the type of equipment. The cross-section images appear to have two main types: bright field and dark field. The aforementioned attributes are only general observations and these are not covering the overall characteristic of the dataset. The subset selected for the research was approximately 8% of the overall dataset.

The experts have labeled 6235 images. Out of this sample set 4159 belong to the cross-section category, and 2076 to the other visual inspection-related images. This random subset was not balanced, as it contained about twice as much cross-section data as the number of images in the other category. Although, this distribution represents the state of the full dataset, where the number of cross-section recordings is roughly twice the size of the other category. In general, the other category is more diverse, which might be problematic if the selected subset does not contain elements of each subclass. One can search for such examples by devising a method that searches for the most dissimilar (or least similar) images within the subset compared to the overall database.

2.2. Data Examination and Preprocessing

Before we proceed with the training process, potential issues related to the data collection characteristics need proper management. Since the dataset was provided by Bosch where the image recordings are stored in a complex structure, the data should

be examined with simple scripts to ensure the proper quality and distribution for the forthcoming tasks. These preprocessing steps cover the removal of duplicated images and the examination and comparison of the similarities of the images.

2.2.1. Removing Duplicates

First, we need to ensure that the dataset does not contain redundant images. In this step, we examine whether an image appears multiple times or not. With the application of a simple, fast, and effective method, we checked the hash values of each image and compared them to each other. Thus we can investigate if two or more files having identical content under a different name are present in the dataset. As a result of the filtering, we removed the redundant data from the subset to be processed.

2.2.2. Examining Similarity

The second step is to examine the similarity distribution between all the images within the dataset to learn about data correlations. After investigating the dataset, we found that several pictures had similar latent space representations. This similarity factor is measured by taking the distance between the latent space representations of the images after normalizing or standardizing the images; in this case, images were divided by the maximum possible value of the pixels (255) as the standard image normalization method.

Comparison of the latent space similarities was carried out in feature space of the VGG19 model. A different pre-trained network could have been used, but since we are comparing the most similar images based on the extracted features of the model, then no notable gain would have been obtained from picking a different pre-trained model. We used the Euclidean distance between the representations of the images and concluded the following:

- By setting a small distance threshold of 0.3 and less: 10 similar images were detected, meaning that these images have at least another significantly related sample in the dataset. For instance, one of the examples is shifted a few pixels from the other one. It is indistinguishable to the human eye, especially since the image had a large resolution, making these tiny details and differences hard to pick up by the experts.
- For a threshold of 0.6 and less: 27 similar images were identified, 10 of them were the same as in the previous case, and the others had some minor differences. For instance, in some cases, one of the two images had measurements written on them.
- By increasing the threshold to 1.0, we found 86 images. Some of the cases were the same as the previous ones. However, new instances emerged with significant differences. For example, two of the pictures were of the exact same device, one of them was with bolts, and the other one was not having bolts.

Based on these observations, the decision was to keep all the images as they will have no significant effect on the model performance or the measured accuracy. Additionally, given that this is how the original dataset is stored, we did not want to artificially influence or manipulate the data distribution. Nonetheless, such functionality might prove useful in other cases, this remains up to the industrial domain experts to decide when to use it.

2.3. Train and Test Sets

Almost any machine learning problem will need to have a separate set to ensure that model can generalize what it has previously learned into unseen data. Since we are utilizing clustering techniques to classify images into two general categories, we will need such a set as well.

Considering that we are using pre-trained models, the test set will only make a difference when applying the dimensionality reduction algorithms to the previously learned representations. Usually, keeping around 20% for the test set is an appropriate percentage, but it depends on the problem, the complexity of the issue, the size of the dataset, and many more factors that could influence the set size. Binary classification problems (as in

this case) could require an even smaller test set, but in our case, there is a lot of diversity within each class.

Thus taking less than 20% may cause some of these variations to occur in the training set only. The train set of *cross-section* category contains 3322 images and the corresponding test portion has 837 examples. The *other* category holds 1666 samples for training and 410 images for testing purposes.

2.4. Selection of Deep Neural Network Models

Over the years, numerous architectures have been published with their weights, trained on massive datasets such as the ImageNet database. A common practice is to fine-tune these pre-trained weights on the required dataset instead of taking some randomly initialized weights, as the pre-trained networks have already proven their capabilities and constructed a common feature space. However, as we seek to connect the domain experts with their work directly, such an option is only feasible for a deep learning specialist but not suitable for the domain experts. Thus we needed to take a different approach to use these pre-trained models. The method that we will be using consists of extracting the pre-trained latent representations and treating these embeddings as features. These features will then serve as an input to one of the dimensionality reduction algorithms such as principal component analysis (PCA) and UMAP.

It is difficult to determine which architecture will yield the best result on a particular dataset. Even for AI professionals picking the best structure could sometimes be reduced to trial and error. Therefore, the domain experts will try a few architectures before finding the right one. Trying such models does not take much effort as they already have been trained.

In theory, with the help of NIPGBoard visualization interface, one only has to click a few buttons to examine these networks and evaluate their results.

One of the approaches to decide on the architecture is to start with the state-of-the-art networks that achieve the highest accuracy on the ImageNet dataset. Nevertheless, this method does not warrant that it will have the best result on the current dataset; it has a better chance to obtain the most favorable outcomes.

2.5. Extracting the Representations

After splitting the dataset and choosing a pre-trained architecture, the two sets of images (train and test portions) are inputted into that particular network. The output is one of the deep layers of the model. It is usually preferred to take the layer just before the softmax/output layer as it has a representation that was optimized for achieving high classification accuracies on the ImageNet dataset. The resulting representation is a two-dimensional matrix where each row is a vector representing the captured feature space of the corresponding image. We assume that the large number of ImageNet classes have many diverse features on this last layer suitable for discriminating novel classes and thus we expect high quality clusters.

After the extraction process, we used the representations of the training and test datasets for training a UMAP model with the corresponding labels that the domain experts provided and then testing it.

2.6. The UMAP Model

Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) is a manifold learning technique typically used for dimension reduction that results in an effective and scalable algorithm that can be effectively applied to virtually any real-world dataset. It is also competitive with other dimensionality reduction algorithms such as t-SNE [30] for visualization quality and supposedly preserves more of the overall layout with excellent run time performance. Furthermore, UMAP has no computational constraints on embedding dimension, making it useful as a general-purpose dimension reduction technique for several machine learning problems. The algorithm makes three assumptions about the data. (i) A uniform distribution of the data on the Riemannian manifold, (ii) the

Riemannian metric is locally constant (or can be approximated as such), (iii) and finally, the manifold is locally connected.

2.7. UMAP Model Training

Now that we have obtained the representations for both training and test sets they will be used to train a UMAP model with the corresponding labels that the domain experts provided. It is possible to train the model without labels as well, but omitting them will cause a noticeable decrease in accuracy. In other words, we could have only labeled the test set or even not have used any labels at all. However, then measuring the accuracy score would have been difficult. We would have needed to use other measurements such as the Silhouette index [31] to get some indicator of the quality of the clusters.

Training is as follows: UMAP has several parameters that need to be fine-tuned, so we picked the ones that seemed of high relevance. Then used a grid search to get the best model within that limits of the search space. Often the results were not so much different from each other or the default parameters because the accuracy was already high.

2.8. UMAP Evaluation

There are several methods for evaluating the accuracy of the resulting clusters. Some of these techniques work with labeled or unlabeled datasets as well. But first, we need to transform the obtained train and test set representations into the UMAP space. After that, the evaluation process starts, and it is applied for both the train and test set independently, as we would like to know the training accuracy as well as the test accuracy. Additionally, this resembles the prediction part of the pipeline, where the unlabeled data need to be classified or clustered in a graph-based seperation.

After obtaining the 2D or 3D embeddings produced by the UMAP model, one can start to separate the clusters from each other. It is possible to achieve this step by different means. One of them is to build a graph, where the nodes are the data points, and the edges are the connections between these nodes. We consider the similarity to be in the inverse of the normalized adjacency distance matrix. Another method is to use the nearest neighbor approach, where each new data point is classified based on its K nearest neighbors.

The graph connections need minimization as the number of the edges are equal to $n(n-1)/2$, where n is the number of images in the set. It is done by one of two methods. The first method uses the minimal spanning tree, where only a single path will remain for connecting all of the nodes together. The other approach would keep the nearest neighbors of each node according to some pre-defined threshold. In the first approach, there is no discontinuity in the graph. Consequently, even points away from the cluster centers will be considered in the process. However, in the second approach, points that do not satisfy the distance threshold will be cut right from the start and are marked as outliers. Both methods have some advantages and disadvantages. We studied both methods and found minor differences due to the high quality of the clusters. We decided to use the minimal spanning tree due to its higher speed in the steps detailed below.

After minimizing the graph, we need to establish the boundaries between each cluster. We used community detection algorithms such as Girvan–Newman [32], and Louvain [33] that extract common elements from large-scale networks. Community structure, a general feature of most networks tries to find network nodes clustered more tightly into groups.

Between these groups, there are only loose connections [32]. Girvan–Newman revolves around finding the community boundaries by using centrality indices. Instead of constructing a measure that tells which edges are most central to communities such as the hierarchical clustering method, Girvan–Newman focuses on the least central edges that are mostly in between communities. On the other hand, the Louvain algorithm is a simple computationally efficient heuristic method based on modularity optimization.

Multiple clusters or common regions form the output of the community detection and each cluster will have some general features common between the nodes. For simplicity and computational efficiency, we used the Louvain algorithm throughout this study.

The found groups need to be assigned labels for the train and test datasets independently. It is done by counting the number of labels for each category in all of the clusters individually, then assigning each cluster the dominant type in that group where dominance is determined solely based on the number of label occurrences. However, during prediction, the system provides the label of the closest cluster to the new sample. Depending on the parameter settings, there could be between 4–32 clusters for each model.

2.9. Misclassification Detection

In the current study, the aforementioned steps are all it takes to achieve high accuracy for this particular dataset. Nevertheless, our goal was to get a near-perfect accuracy without involving deep learning specialists too much or at all. Consequently, we needed a simple approach to detect misclassifications in the model. We have considered multiple strategies such as identifying out-of-distribution data or anomalies. However, training an anomaly detector is not trivial, there is no guarantee that anomalies will get misclassified or that a misclassification is an anomaly.

We note that the UMAP model is dealing with the representations of the pre-trained model, so the features captured by the latent space will highly depend on the model architecture and the data used for training that model, such as the ImageNet dataset. With this idea in mind, we hypothesize that different pre-trained networks may result in distinctive features. Training a dimensionality reduction algorithm such as UMAP on these features might result in a different set of misclassifications for different pre-trained networks. Therefore we used more than one network to detect possible misclassifications in the data. We can say that a data point is dubious when two models classify it differently, meaning they disagree on which class the data point belongs to. For the most accurate classification, we selected the two best performing pre-trained models. Table 1 shows the common metrics of the tried algorithms on the test dataset which means in our case DenseNet201 (with 0.9992 accuracy) and Xception (with 0.9976 accuracy) were chosen. The accuracy value refers to the number of correctly classified images in relation to the total number of data instances. It has its best value at 1. More than one network is an ensemble-like approach that can be useful as they address certain issues of learning algorithms that output only a single hypothesis. For example, different networks may be stuck in different local minima of the problem space and a voting method can be advantageous as it could reduce the risk of not generalizing well to unseen data. An example is a recent Kaggle competition: the winner produced state-of-the-art results with 18 different networks combined into an ensemble [34] and larger ensembles were also competitive. For a review on ensemble methods, see, e.g., [35] and the references therein.

Table 1. Pre-trained Models' Results on the Test Set.

Model	Accuracy	Precision	Specificity	Recall	F1
VGG16	0.9960	0.9988	0.9975	0.9952	0.9970
VGG19	0.9912	0.9988	0.9975	**0.9888** [a]	0.9935
NASNetLarge	0.9968	0.9976	0.9951	0.9976	0.9976
Xception	0.9976	0.9976	0.9951	**0.9988** [a]	0.9982
InceptionResNetV2	0.9952	0.9988	0.9975	0.9941	0.9964
DenseNet201 [a]	**0.9992**	**1.0000**	**1.0000**	**0.9988**	**0.9994**

[a] The best results in each metric and the best performing model overall are bold.

In our case, we can turn to the domain experts if models disagree and, due to the high quality of the clusters, a small number of networks (two in our case) is sufficient.

Our approach for using more than one network drastically reduces misclassifications by detecting potential errors in the models' prediction. However, the more models introduced, the more clusters need to be labeled by the experts, and each model results in approximately 10–25 groups. More models will give rise to more misclassification candidates and this increases the experts' work in the process. In turn, if the number of such candidates is too high then more elaborate ensemble learning based voting system using

more networks can overcome the problem. There is trade-off between the required accuracy, the number of networks, and the amount of manual work that should be done.

2.10. Visual Inspection with NIPGBoard

To connect domain experts, technology and, deep learning professionals, we used the NIPGBoard software which was developed by the Neural Information Processing Group (NIPG) at Eötvös Loránd University in collaboration with Argus Cognitive company. NIPGBoard similarly to Tensorboard is a visualization interface with interactive functions. It uses the core features of the Tensorboard projector plugin, with additional algorithms and the ability to use pre-trained networks at the touch of a button, guaranteeing ease of use for domain experts.

The projection panel displays thumbnail versions of the data, called sprite images, after automatically applying one of the built-in dimension reduction techniques. The 3D position of the displayed data corresponds to the low-dimensional embedding of the hidden representations of the images, which are extracted using one of the deep neural networks. The user of the NIPGBoard tool can combine the available learning algorithms with the dimensionality reduction techniques. Displaying the results and switching between embeddings is an easy-to-use interface feature, so domain experts can gain comprehensive knowledge about the data and the selected methods. Labels can be displayed for each piece of data, and label-based colour overlays can be added to the images to provide further information on quality of the clusters. Label-based colour overlays also show the effectiveness of clustering by visual inspection of data in the embedding space without additional AI expert guidance.

In the course of our work, we have incorporated additional functions into the software to make the work of the expert easier. Figure 3 shows the interface of the NIPGBoard visualization tool with the loaded Bosch data.

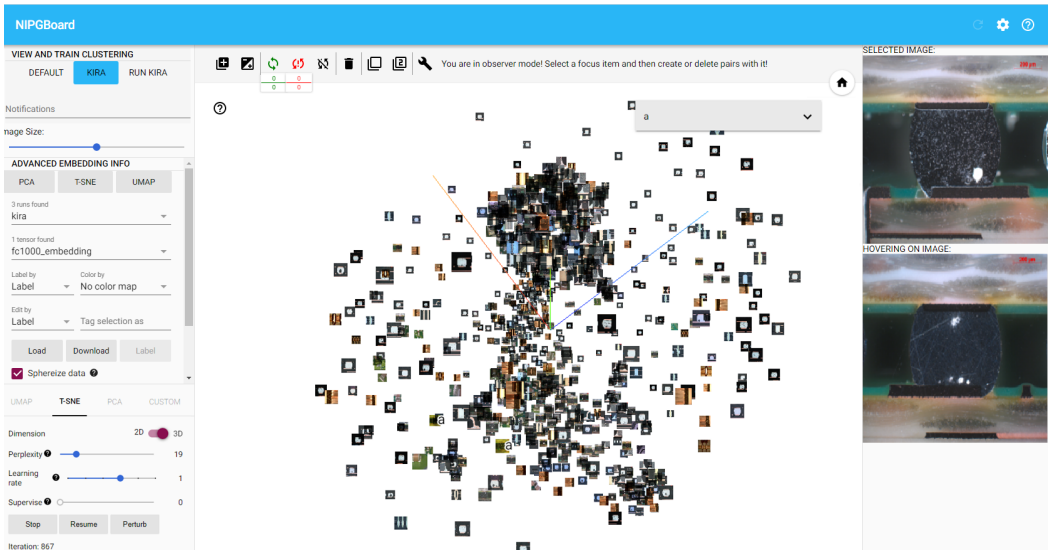


Figure 3. NIPGBoard, a modified TensorBoard interface. (**Left**) methods and parameters of the embedding and dimensionality reduction parameters. (**Middle**) projection panel. (**Right**) higher resolution samples made visible by hovering over the sprites.

3. Results

A typical domain expert has special domain knowledge, but may have limited knowledge in deep-learning, statistics, and AI to determine which pre-trained model will work well on which dataset. NIPGBoard may help them to gain experience by trying the models themselves and observing the outcomes.

The evaluation started by using VGG16, VGG19, NASNetLarge, Xception, InceptionResNetV2, and DenseNet201 architectures. The feature space is gathered by extracting the

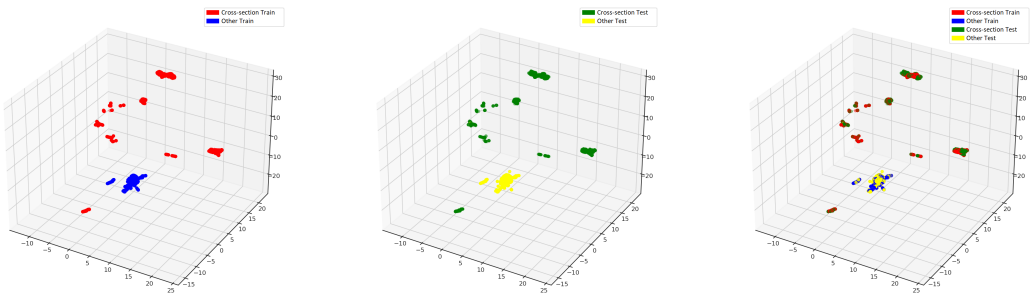
latent representations from the layer right before the output. For the VGG networks this layer is named fc2, which refers to the second fully connected layer with 4096 neurons, indicating that each image's embedding would be a vector of 4096 features. The dimensionality reduction algorithm will then make use of these features to construct the clusters.

UMAP Fine-Tuning

UMAP has several parameters that can be fine-tuned. Some of the most useful ones are the number of neighbors and the size of the local neighborhood used for manifold approximation. Larger values preserve the global view of the manifold, while smaller values affect the local view. In other words, smaller values will result in more but smaller clusters, and larger values will result in fewer but bigger clusters. Other valuable optional parameters are the learning rate used for the embedding optimization, and local connectivity indicating the number of nearest neighbors connected on the local level. The higher this value is, the more connected the manifold becomes locally.

The negative samples rate refers to the number of negative samples to select per positive sample in the optimization process. Increasing this value will result in greater repulsive force being applied, and higher optimization cost, but slightly more accuracy. We used fixed numbers of the dimensions, those were two and three, as it is easier to visualize and was sufficient for the task. If selected carefully, the number of epochs or iterations should not affect the learning process too much. However, if we consider extreme high or low values, it might severely deteriorate the generalization.

The three-dimensional display of the data after applying UMAP visually shows the clustering results. We show example visualizations for two of the best performing models' embeddings with UMAP dimension reduction after applying a grid-search on the aforementioned parameters. Figure 4 shows UMAP embedding space of DenseNet201 on the training and test set. Figure 5 shows UMAP embedding space of Xception on the training and test set.

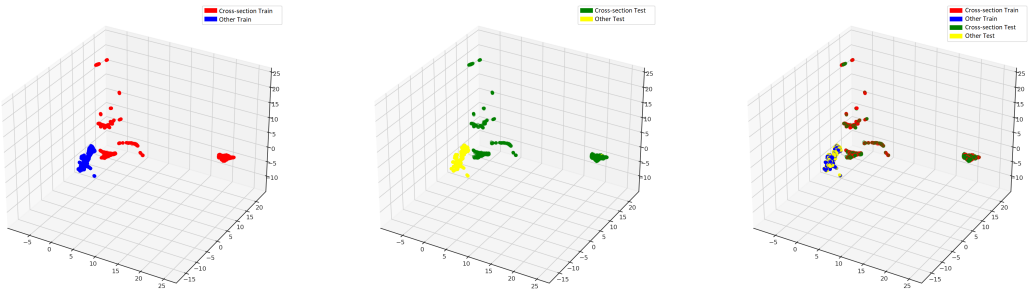


(**a**) (**b**) (**c**)

Figure 4. UMAP embedding space of DenseNet201 on training and test set. (**a**) shows *cross-section* and *other* labeled data from the training set, (**b**) shows *cross-section* and *other* labeled data from test set, (**c**) illustrates both labeled train and test data points in the same representation space.

By careful examination of Figures 4 and 5, it can be seen that the results shown in the subplots (a) and (b) in each of the individual figures are quite similar, almost indistinguishable, which is supported by the hypothesis that the test subset represents the overall dataset distribution. Therefore it will give rise to a relatively accurate estimate of the overall performance. Whereas in the corresponding subplots (a), (b), and (c) from each of the two figures, it is noticeable that the clusters are different between the plots, supporting the idea that different pre-trained models will give rise to a different set of features, leading to a different set of misclassifications between the two pre-trained models. These misclassifications are then detected by employing the ensemble-like methodology.

Clusters were separated from each other by means of the graph clustering methods detailed in Section 2.8.



(**a**) (**b**) (**c**)

Figure 5. UMAP embedding space of Xception on training and test set. (**a**) shows *cross-section* and *other* labeled data from the training set, (**b**) shows *cross-section* and *other* labeled data from test set, (**c**) illustrates both labeled train and test data points in the same representation space.

DenseNet201 and Xception were selected as follows. We repeated the evaluation with the following architectures: NASNetLarge, Xception, InceptionResNetV2 and DenseNet201. They all had promising results as Tables 1 and 2 show the numerical outcomes. We displayed commonly used evaluation metrics for the classification results. Accuracy summarizes the performance of a classification model, its value shows the amount of correct predictions in the total number of predictions. Precision is the fraction of correct predictions among all the predictions that the model made. Recall (true positive rate) refers to the probability of a positive test, which shows the ratio of correct predictions in contrast to all the possible correct predictions. Specificity (true negative rate) relates to the model's ability to not make incorrect predictions. F1 score is a measurement that is calculated as a weighted average of both precision and recall values. F1 score has its best value at 1 when the model is perfect and worst value at 0 when none of the predictions are correct.

On the train set VGG16, NASNetLarge, Xception and DenseNet201 equally achieved highest accuracy scores, as it can be seen on Table 2. On the test set DenseNet201 achieved the highest accuracy, followed by Xception with a small margin, see Table 1.

Table 2. Pre-trained Models' Results on the Train Set.

Model	Accuracy	Precision	Specificity	Recall	F1
VGG16	1.000	1.000	1.000	1.000	1.000
VGG19	0.9998	0.9994	0.9997	1.000	0.9997
NASNetLarge	1.000	1.000	1.000	1.000	1.000
Xception	1.000	1.000	1.000	1.000	1.000
InceptionResNetV2	0.9998	0.9994	0.9997	1.000	0.9997
DenseNet201	1.000	1.000	1.000	1.000	1.000

In order to make the presented pipeline suitable for other relevant tasks, we propose to further improve it by combining any of the two models already trained, using a kind of ensemble learning methodology: instead of the classical approach of combining the outputs of the models and include a voting system, we needed to find out the cases that the models assigned different labels to a given sample. In the final phase we utilized the ones that returned the best accuracy according to Table 1, i.e., Xception and DenseNet201. Their joint outcome gave rise to only four misclassification candidates in total. In our case, after the domain experts review these four images, the accuracy will increase to 1.0 for the test set.

4. Discussion

For experts with no experience in training deep networks, observing the evaluation of training and test data can provide useful knowledge. Table 2 summarizes the pre-trained model performances on the training set. It is common for train data to have higher values

than test data. As Table 1 shows DenseNet201 has the best results for the test set, even though it is not the best performing model on the ImageNet among the ones that we have tested.

In addition to the numerical assessment, visualizations also give an indication of a properly constructed pipeline. Figures 4 and 5 show that the data split was appropriate and highlights minor differences between the pre-trained models.

As it is shown in Tables 1 and 2, the difference between the worst performing model VGG19, and the best performing model DenseNet201 is less than 0.8%. This phenomenon does not imply that DenseNet201 will always perform better than the other models for every case. We conduct that DenseNet201 has a good chance of performing well on similar datasets. Consequently, this also suggests that VGG19 will not perform well on a similar dataset. This experience can be gained by the domain expert themselves without any prior knowledge of the underlying architecture of the models.

Simple methods can support the work of domain experts to further improve data management tasks beyond labelling. For example, one can search for images or groups of images similar to a given image by measuring the distances between latent representations of images that can be incorporated into our pipeline. Another approach would be to search for anomalies by finding the least similar images in the dataset, or images representing a rare subcategory.

Generative Adversarial Networks (GANs) [36] and Variational Autoencoders (VAEs) [37] are becoming increasingly popular in anomaly detection tasks. The use of these algorithms is a promising opportunity for further improvements to expand our pipeline. Yet, the integration of such methods need care as they may involve training and need artificial intelligence specialists.

Although the case we studied brought about high-quality results, there are some possible limitations to it. For instance, it heavily relies on pre-trained models for feature extraction, meaning those features are extracted from the ImageNet dataset tuned for the classes of the dataset. In turn, the selection of the final layer could not be the optimal strategy for all datasets. However, in the case of this particular cooperation, this technique achieved the excellent results. It is hard to predict how the method will perform on other datasets. Another aspect that could be considered a downside is the need for human intervention in various stages of the pipeline. The current methodology design allows the experts to directly examine and interact with the data to decide the best course of action based on their experience. However, if the extreme high level of accuracy is not required, such careful examination could be deemed unnecessary.

5. Conclusions

We developed a pipeline for the domain experts assuming that they may not have comprehensive knowledge about deep learning AI methods. We used the pipeline to solve a soldering data classification task. In this pipeline, we took advantage of pre-trained models, clustering of the feature maps of deep networks, and misclassification detection using more than one network.

Our goal was overcome the bottlenecks in academic-industrial collaborations by separating the tasks of the AI specialists and the domain experts and simplifying the tasks for the latter by automatically labelling data while achieving high accuracy with as little manual work as possible. This separation is motivated by studies pointing out that solving a problem using deep learning is traditionally still a task for AI specialists. Bottlenecks include the struggle with communication misunderstandings and knowledge gap between the expertise of the two fields. We showed that direction of separating the tasks using our pipeline is promising.

For domain experts, one novel way is to use the so-called AutoML system, but as the referred studies emphasized, confidence and positive user experience depend heavily on understanding such systems. We also used embedding into 2D or 3D, clustering of

the embeddings and the belonging visualisations for helping in the steps towards the final results.

The documented evaluation pipeline provides compact and generic guidance and visual representations for domain experts on how to manage data properly and take advantage of deep learning methods. We had to request some annotation work from the domain experts followed by simple preprocessing steps, easily accessible pre-trained models, visual and numerical comparison, and explanation of why model selection may be appropriate.

We also tried to point out potential misclassifications by means of more than one pre-trained network: disagreements between their classification results can serve as indications. Dubious samples were sent back to the experts for review. We noted that a larger set of pre-trained networks could lead to a voting system called ensemble learning. Our approach was able to achieve near-perfect accuracy on the test set with only 8% data labeled by domain experts.

Author Contributions: Conceptualization, A.L., J.S. and K.B.F.; methodology, A.L. and J.S.; software, J.S.; validation, B.H. and K.B.F.; formal analysis, J.S. and A.L.; investigation, S.F., B.H. and K.B.F.; resources, S.F., B.H. and A.L.; data curation, S.F. and B.H.; writing—original draft preparation, J.S.; writing—review and editing, K.B.F.; visualization, J.S. and K.B.F.; supervision, A.L.; project administration, A.L., K.B.F. and B.H.; funding acquisition, A.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by (a) the "Application Domain Specific Highly Reliable IT Solutions" project implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme No. 2020-4.1.1.-TKP2020 (National Challenges Subprogramme) funding scheme, and (b) the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program. The authors thank to the above projects and to Robert Bosch, Ltd. Budapest, Hungary for their generous support to the Department of Artificial Intelligence.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the software development efforts of Ervin Téglás and the work of Kadri Loubna for the partial implementation of the pipeline's pilot version. Special thanks are due to Ellák Somfai for his careful reading of the manuscript and to András Lőrincz for his help and guidance during the project.

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

References

1. Xin, D.; Wu, E.Y.; Lee, D.J.L.; Salehi, N.; Parameswaran, A. Whither AutoML? understanding the role of automation in machine learning workflows. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; pp. 1–16.
2. Weidele, D.K.I.; Weisz, J.D.; Oduor, E.; Muller, M.; Andres, J.; Gray, A.; Wang, D. AutoAIViz: Opening the blackbox of automated artificial intelligence with conditional parallel coordinates. In Proceedings of the 25th International Conference on Intelligent User Interfaces, Cagliari, Italy, 17–20 March 2020; pp. 308–312.
3. Drozdal, J.; Weisz, J.; Wang, D.; Dass, G.; Yao, B.; Zhao, C.; Muller, M.; Ju, L.; Su, H. Trust in AutoML: Exploring information needs for establishing trust in automated machine learning systems. In Proceedings of the 25th International Conference on Intelligent User Interfaces, Cagliari, Italy, 17–20 March 2020; pp. 297–307.
4. Abd Al Rahman, M.; Mousavi, A. A review and analysis of automatic optical inspection and quality monitoring methods in electronics industry. *IEEE Access* **2020**, *8*, 183192–183271.
5. Metzner, M.; Fiebag, D.; Mayr, A.; Franke, J. Automated optical inspection of soldering connections in power electronics production using convolutional neural networks. In Proceedings of the 2019 9th International Electric Drives Production Conference (EDPC), Esslingen, Germany, 3–4 December 2019; pp. 1–6.
6. Weimer, D.; Scholz-Reiter, B.; Shpitalni, M. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Ann.* **2016**, *65*, 417–420. [CrossRef]

7. Silva, W.R.L.D.; Lucena, D.S.D. Concrete cracks detection based on deep learning image classification. *Proceedings* **2018**, *2*, 489. [CrossRef]
8. Dai, W.; Mujeeb, A.; Erdt, M.; Sourin, A. Soldering defect detection in automatic optical inspection. *Adv. Eng. Inform.* **2020**, *43*, 101004. [CrossRef]
9. Wang, T.; Chen, Y.; Qiao, M.; Snoussi, H. A fast and robust convolutional neural network-based defect detection model in product quality control. *Int. J. Adv. Manuf. Technol.* **2018**, *94*, 3465–3471. [CrossRef]
10. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv* **2016**, arXiv:1603.04467.
11. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
12. Zhan, X.; Xie, J.; Liu, Z.; Ong, Y.S.; Loy, C.C. Online deep clustering for unsupervised representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 6688–6697.
13. Hartigan, J.A.; Wong, M.A. Algorithm AS 136: A k-means clustering algorithm. *J. R. Stat. Soc. Ser. (Appl. Stat.)* **1979**, *28*, 100–108. [CrossRef]
14. Jain, A.K.; Murty, M.N.; Flynn, P.J. Data clustering: A review. *ACM Comput. Surv. (CSUR)* **1999**, *31*, 264–323. [CrossRef]
15. Caruana, R.; Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 161–168.
16. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
17. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
18. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8697–8710.
19. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
20. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258.
21. 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, 10–15 June 2019; pp. 6105–6114.
22. Zhu, X.J. *Semi-Supervised Learning Literature Survey*; University of Wisconsin: Madison, WI, USA, 2005.
23. Bo, D.; Wang, X.; Shi, C.; Zhu, M.; Lu, E.; Cui, P. Structural deep clustering network. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 1400–1410.
24. McInnes, L.; Healy, J.; Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv* **2018**, arXiv:1802.03426.
25. Yang, B.; Fu, X.; Sidiropoulos, N.D.; Hong, M. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; pp. 3861–3870.
26. Min, E.; Guo, X.; Liu, Q.; Zhang, G.; Cui, J.; Long, J. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access* **2018**, *6*, 39501–39514. [CrossRef]
27. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
28. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images, Citeseer. 2009. Available online: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 7 July 2022).
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
30. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
31. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]
32. Girvan, M.; Newman, M.E. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826. [CrossRef] [PubMed]
33. Blondel, V.D.; Guillaume, J.L.; Lambiotte, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, *2008*, P10008. [CrossRef]
34. Ha, Q.; Liu, B.; Liu, F. Identifying melanoma images using efficientnet ensemble: Winning solution to the siim-isic melanoma classification challenge. *arXiv* **2020**, arXiv:2010.05351.
35. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
36. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 8–13 December 2014; Volume 27.
37. Kingma, D.P.; Welling, M. Auto-encoding variational Bayes. *arXiv* **2013**, arXiv:1312.6114.