Industrial Image Grouping Through Pre-Trained CNN Encoder-Based Feature Extraction and Sub-Clustering

Selvine G. Mathias[a], Saara Asif[b], Muhammad Uzair Akmal[c], Simon Knollmeyer[d], Leonid Koval[e] and Daniel Grossmann[f]

AImotion Bavaria, Technische Hochschule Ingolstadt, Esplanade 10, 85049 Ingolstadt, Germany
{SelvineGeorge.Mathias, Saara.Asif, MuhammadUzair.Akmal, Simon.Knollmeyer, Leonid.Koval, Daniel.Grossmann}@thi.de

Keywords: Clustering, CNN Encoder, Convergence, C-Score.

Abstract: A common challenge faced by many industries today is the classification of unlabeled image data from production processes into meaningful groups or patterns for better documentation and analysis. This paper presents a sequential approach for leveraging industrial image data to identify patterns in products or processes for plant floor operators. The dataset used is sourced from steel production, and the model architecture integrates feature reduction through convolutional neural networks (CNNs) like VGG, EfficientNet, and ResNet, followed by clustering algorithms to assign appropriate labels to the observed data. The model's selection criteria combine clustering metrics, including entropy minimization and silhouette score maximization. Once primary clusters are identified, sub-clustering is performed using near-labels, which are pre-assigned to images with initial distinctions. A novel metric, C-Score, is introduced to assess cluster convergence and grouping accuracy. Experimental results demonstrate that this method can address challenges in detecting variations across images, improving pattern recognition and classification.

1 INTRODUCTION

Raw data in many plant floors tends to be unlabeled, nasty, and unstructured due to erroneous data recording and machine or robot failures. Despite these challenges, any data arriving from the plant floors, whether from machines directly or databases recording these observational values, constitutes a readily available source of information to work upon. One of the important problems in investigating data is to know the class or category or label of the data (Xu and Tian, 2015; Xu and Wunsch, 2005). This is done by manual inspection in most cases which is time-consuming, inefficient, and downtime-inducing. Having a category for the samples gives usable knowledge for further analysis such as classifications. However, no standard methodology is available to obtain them. This paper proposes a simple yet effective scheme of grouping obtained high-dimensional samples into clusters based on feature reductions employed by transfer learning (TF) techniques (Cardoso and Ferreira, 2021; Silva and Capretz, 2019).

This research varies from the current literature in two aspects: 1) The proposed scheme presents a pipe-lining of readily available algorithms to tackle labeling issues and 2) this research focuses more on the end-user application, rather than performance for leveraging the ability to generalize this scheme across different types of image datasets. For this study, a public dataset has been used for comparisons of the working of the model. This paper has been divided into the following sections: Section 2 discusses the prevalent applications in the literature focusing on clustering, artificial neural networks (ANNs), and TF approaches. Sections 3 and 4 describe the proposed methodology and its implementation details. Section 5 presents a discussion of the performance of the models across the different algorithms along with some observations. The final sections present an end-to-end view of such an application on the production floor and conclude with some brief thoughts on future works.

[a] https://orcid.org/0000-0002-6549-0763
[b] https://orcid.org/0009-0006-1284-5635
[c] https://orcid.org/0009-0007-3961-1174
[d] https://orcid.org/0009-0002-1429-6992
[e] https://orcid.org/0000-0003-4845-6579
[f] https://orcid.org/0000-0002-7388-5757

Mathias, S. G., Asif, S., Akmal, M. U., Knollmeyer, S., Koval, L. and Grossmann, D.
Industrial Image Grouping Through Pre-Trained CNN Encoder-Based Feature Extraction and Sub-Clustering.
DOI: 10.5220/0013189000003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 2*, pages 496-506
ISBN: 978-989-758-737-5; ISSN: 2184-433X

2 RELATED LITERATURE

2.1 Clustering in Industrial Image Data

Clustering is an essential unsupervised learning method used to group data points based on inherent similarities. In industrial settings, clustering on image data has become increasingly relevant due to the rise of automated visual inspection, quality control, and fault detection systems (Xu and Wunsch, 2005; Xu and Tian, 2015; Oyelade et al., 2019). In the context of industrial applications, images from manufacturing lines, assembly processes, and finished products are used for tasks such as defect detection, process monitoring, and automated sorting. One of the most commonly applied methods is *k-means clustering*, which groups image data based on pixel values or features extracted from images. For instance, k-means has been applied in industries such as *semiconductor manufacturing* for clustering images of wafers to identify defect patterns (Saad et al., 2015). However, simple clustering methods like k-means struggle with high-dimensional and complex image data, where the underlying features are often nonlinear and challenging to separate into distinct clusters. In response to these limitations, *Gaussian Mixture Models (GMMs)* have been applied to image clustering tasks where pixel distributions may overlap, offering a probabilistic framework that allows for the modeling of complex distributions (Zhang and Chen, 2022).

2.2 Feature-Based Clustering Approaches

Given the high-dimensional nature of image data, direct clustering on raw pixel values often proves inadequate. To overcome this, feature extraction techniques are employed prior to clustering. Dimensionality reduction techniques, including *PCA*, *t-SNE*, and *UMAP*, reduce complexity for high-dimensional data clustering (van der Maaten and Hinton, 2008). Additionally, *graph-based descriptors* model pixel relationships, and biologically inspired methods like *Gabor Wavelets* mimic visual cortex processing (Daugman, 1985). Hand-crafted methods like *HOG* and *SIFT* extract edge orientations and key points robust to transformations, while *LBP* and *Gabor* Filters emphasize texture and spatial frequency features, often used in object detection and texture analysis (Dalal and Triggs, 2005; Lowe, 2004) . Transform-based approaches like *Fourier Transform* and *Wavelet Transform* provide frequency and multi-resolution details critical for pattern recognition in medical or satellite images (Strang and Nguyen, 1996). These approaches remain indispensable in clustering high-dimensional data, particularly, images.

The use of *Convolutional Neural Networks (CNNs)* for feature extraction has been a breakthrough in clustering industrial image data. CNNs automatically learn hierarchical features from images, such as edges, textures, and higher-level abstractions, which can then be fed into clustering algorithms. This approach has proven effective in industries like *automotive manufacturing*, where images of parts or components are clustered based on defects, wear, or assembly anomalies (Fan, 2024).

For instance, in automated *quality inspection systems*, CNNs have been used to extract relevant features from product images, which are subsequently clustered to identify common types of defects or categorize products based on visual similarity. These clustering results are then utilized for quality control and further analysis to improve the production process (Hartner et al., 2022).

2.3 Advanced Sub-Clustering in Industrial Image Data

In more complex industrial environments, single-level clustering often fails to capture the fine-grained distinctions between image data. This has led to the development of *sub-clustering approaches*, where clusters are further divided into sub-clusters to reveal more nuanced patterns in the data. Clustering with more clusters partitions the entire dataset into a larger number of groups, capturing global distinctions across all data points. Sub-clustering, on the other hand, refines an existing cluster into smaller groups, uncovering localized patterns or nuances within a specific subset. While the former provides broad segmentation, the latter offers detailed insights into the internal structure of predefined clusters. Sub-clustering is particularly useful in applications like *visual inspection of textured surfaces* where different defect types may not be entirely distinct but exist within overlapping regions in feature space (Li et al., 2021). By identifying sub-clusters, companies can differentiate between minor variations in defects, enabling better quality control.

Another sub-clustering approach is *deep clustering* (Xie et al., 2016; Chang et al., 2017), where deep learning models are used to both extract features and simultaneously perform clustering. This technique, often involving autoencoders or self-organizing maps, reduces the dimensionality of the image data before clustering, revealing both coarse and fine clusters of related images. This approach has been applied in *industrial robotics*, where sub-clusters are used to

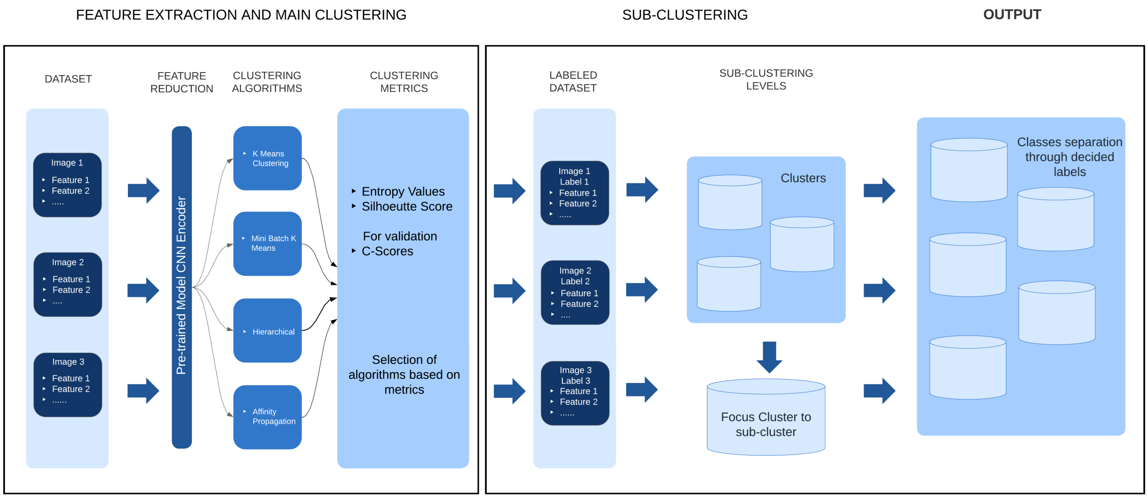


Figure 1: A Workflow Representation of the Proposed Methodology for Industrial Image Grouping.

refine the classification of object parts or tools for improved precision in automated assembly lines (Fu et al., 2024).

Similar to our work, the authors in (Biswas and Jacobs, 2014) introduced a two-pronged strategy to enhance clustering effectiveness: Sub-clustering, where the proposed algorithm focuses on clustering a subset of images which aims to generate smaller but purer clusters that provide representative examples from each class and Active Sub-clustering, where to further refine the clustering results, the authors incorporate human input through an active sub-clustering algorithm. This interaction allows for more accurate adjustments based on user feedback, leading to improved clustering outcomes. Key differences are the use of public datasets such as face image and leaf image datasets, along with involvement of additional cost functions that are implemented to improve cluster purities that is absent from our study. A major difference is the evaluation metric, which similar to our work, is based on Jaccard's co-efficient. However, in this case, the metric calculation is very different from the one proposed later in Section 3.3.1.

2.4 Research Gap and Novel Contributions

Despite the success of clustering techniques in industrial image data analysis, several challenges remain. One major gap in the literature is the limited use of *real-time clustering* in industrial settings, particularly for streaming image data collected from production lines. Industrial environments often require continuous monitoring and analysis of images, but traditional clustering approaches are batch-oriented and lack the ability to update cluster memberships dynamically as new data arrives. Research into *streaming clustering algorithms* for image data, which can handle continuous data flow while maintaining performance, is still in its infancy (Wang et al., 2023).

Additionally, current methods of clustering industrial images tend to rely heavily on pre-defined features or manual tuning of hyperparameters. For instance, the performance of k-means or GMMs can be sensitive to initialization and the selection of cluster numbers. While CNN-based feature extraction has alleviated some of these issues, the *automatic determination of optimal cluster numbers* for large-scale image datasets remains an open problem. Further research is needed into adaptive clustering techniques that can dynamically adjust cluster numbers based on data complexity, which would be particularly useful in scenarios where the characteristics of image data evolve over time.

Lastly, while sub-clustering approaches have proven useful, there is a lack of standardization in defining when and how sub-clustering should be applied. Current sub-clustering techniques are often domain-specific, and there is little research into generalizable frameworks that can be applied across different industrial image datasets. The development of new metrics, such as the *C-Score*, which aggregates similarities between true labels and predicted clusters, offers a promising direction for evaluating clustering performance, particularly when sub-clusters are involved. A focus on more robust evaluation metrics and automated sub-clustering frameworks could fill an important gap in the literature.

3 METHODOLOGY

Given the increasing volume of image data from industrial environments, a widely-used unsupervised approach that minimizes the reliance on expert knowledge for image differentiation has not been thoroughly explored. Therefore, the methodology presented here utilizes available tools from the AI domain to introduce a degree of distinction when processing acquired images (see Figure 1). However, to draw a sense of rationality in introducing sub-clustering, there must be some intermediate labeling or distinction among images before clustering. This can not only improve the separation process, but also evaluate clustering at every level so as not to pursue irrelevant or redundant approaches. We term this intermediate labeling as *near-labels* so as to provide a baseline for separation. This is broadly explained in Section 3.3.1.

3.1 Feature Reduction Through Pre-Trained CNN Encoders

Using pre-trained CNN encoders for feature extraction involves repurposing convolutional layers from models like ResNet, VGG, or EfficientNet, which were trained on large datasets such as ImageNet. These CNNs learn a hierarchical representation of features, where early layers capture basic patterns like edges and textures, and deeper layers represent more abstract patterns such as object parts. To extract features, the output from the penultimate layer of the CNN is commonly used. This layer, typically just before the final classification layer, provides a condensed high-dimensional representation of the input image. In many architectures, this corresponds to the **"global average pooling" (GAP) layer**, which pools spatial information to form a feature vector. These feature vectors are then used as inputs to other models or classifiers for further processing. The following models are used here for feature extraction:

- **EfficientNet-B7.** A highly efficient model that uses compound scaling to optimize depth, width, and resolution for improved performance on image classification tasks.
- **InceptionV3.** An advanced CNN architecture that employs inception modules to capture multi-scale features through parallel convolutional filters, enhancing classification accuracy.
- **ResNet50.** A deep residual network that introduces skip connections to mitigate the vanishing gradient problem, allowing for the training of very deep networks with improved accuracy.
- **VGG16.** A straightforward and deep architecture characterized by its use of small (3x3) convolutional filters and a uniform architecture, which significantly impacted image classification tasks.

The features extracted from these are fed to the next step in the pipeline, namely the conventional clustering algorithms. The selection of these models as backbone feature extractors is driven by their high performance on ImageNet and diverse architectural designs, leading to varied feature sets. **VGG16** provides hierarchical feature maps through sequential convolutional layers, making it a robust general-purpose extractor (Simonyan and Zisserman, 2015). **ResNet50**, with its residual connections, allows for deeper networks and improved feature discrimination, addressing the vanishing gradient problem (He et al., 2016). **EfficientNetB7** achieves superior accuracy and efficiency through compound scaling of depth, width, and resolution (Tan and Le, 2019). **InceptionV3** enhances multi-scale feature extraction via inception modules, enabling versatility in capturing details at various resolutions, suitable for sub-clustering tasks (Szegedy et al., 2016). Additionally, other models replicating or extending these architectures, such as deeper variants of ResNet or scaled versions of EfficientNet, can also be considered for similar purposes, given their alignment with the structural principles and performance metrics of the selected backbones.

3.2 Conventional Clustering Approach

The authors implemented part of the methodology as used in (Mathias et al., 2019), specifically different clustering algorithms evaluated by entropy minimization and silhouette score maximization across a range of clusters, to determine the best algorithm and number of clusters for sub-clustering.

3.2.1 Algorithms

The following common clustering techniques were used for a comparative analysis:

- K-Means
- Agglomerative (Hierarchical)
- Mini Batch K-Means
- Spectral Clustering
- Gaussian Mixture Model

The in-depth working of these algorithms is discussed in (Xu and Tian, 2015). These algorithms require the number of clusters to be fed as an input. Each algorithm is fed the reduced data that is obtained from the last layer of a CNN encoder, without any scaling or augmentation. The results are based on the metrics discussed in the next subsection.

3.2.2 Metrics for Selection of Clustering

Entropy Minimization. To address how many clusters can be expected in a group of samples, the entropy metric is computed for a cluster number (Mathias et al., 2019). It measures disorder in a vector and is calculated as follows:

$$H(X) = -\sum_{i=0}^{n-1} P(x_i) log_2(P(x_i)) \tag{1}$$

where $X = x_i$ is a label vector containing n unique labels. $P(x_i)$ is a probability measure for every x_i occurring in X, calculated by

$$P(x_i) = \frac{\texttt{no. of occurrences of } x_i}{\texttt{length of } X}$$

A higher entropy value indicates more disorder in the vector and hence, minimization is seen as a good measure to determine the number of clusters for analysis. Once an ideal range is identified, the clustering techniques are applied to the data for each number in the range.

Silhouette Score. This score evaluates the similarity of an object to its own cluster or group (Mathias et al., 2019). It is calculated as follows:

$$S(x_i) = \frac{b(x_i) - a(x_i)}{max\{a(x_i), b(x_i)\}} \tag{2}$$

where $b(x_i)$ represents the smallest mean distance of x_i to all points in any other cluster except its own and $a(x_i)$ is the mean distance between x_i and all other points in its own cluster. Silhouette Score is the average of all such $S(x_i)$ showing how tightly points are clustered. It lies between -1 and 1, 1 denoting the best case.

The performance of the algorithms is evaluated using Silhouette scores with the maximum score being the best across all the algorithms and the range of clusters.

3.3 Sub-Clustering Scenario

Once main clusters are formed in the given dataset through appropriately selected algorithms, sub-clustering is based on the cluster that needs more separation. Visually, this can be seen when the clusters are well separated corresponding to higher silhouette scores and lower entropy values. However, sub-clustering becomes necessary when the target entities are contained in a particular cluster or group. To proceed, we implement the same pipeline as in Section 3.2. In addition to that, we also calculate an additional metric that serves the purpose of near-labeling. This is explained in the following subsection.

3.3.1 C-Score

This is a novel metric used to evaluate the performance of clustering algorithms by measuring the overlap between true labels and predicted clusters. It is defined as the sum of Jaccard similarities for all pairs of true labels C_t and predicted clusters C_p:

$$\texttt{C-Score} = \sum_{i=1}^{k}\sum_{j=1}^{m} J(C_{ti}, C_{pj}) = \sum_{i=1}^{k}\sum_{j=1}^{m} \frac{|C_{ti} \cap C_{pj}|}{|C_{ti} \cup C_{pj}|},$$

where k represents the total number of unique true labels and m represents the total number of predicted clusters.

The summation calculates the amount of common points to the total number of points in the two sets and runs over all classes available, in the true classes as well as clusters formed, and does not the count the equality of the number of clusters to the actual classes. Thus, a clustering algorithm might have detected more patterns than the actual known classes if more clusters are formed during clustering. Simply speaking, this metric measures the amount of intersections among the clusters from two different clusterings or groupings. It does not necessarily represent similarity or closeness of clusters, since the computation can exceed to higher numbers depending on the intersections. Higher C-Score implies good amount of intersections, for example, a defect class contained in a cluster. The 'convergence' alludes to the possibility that as more clusters are formed, a targeted class may coincide with one of the clusters.

In the case of converged clustering, where each predicted cluster contains exactly one true label and no points from other labels, the points of each true label C_{ti} are distributed across multiple disjoint clusters $C_{pj_1}, C_{pj_2}, \ldots$, with each cluster containing only points from C_{ti}. For each pair where C_{pj} contains points from C_{ti}, we have:

$$\frac{|C_{ti} \cap C_{pj}|}{|C_{ti} \cup C_{pj}|} = \frac{|C_{ti} \cap C_{pj}|}{|C_{ti}|} \tag{3}$$

Since the clusters are disjoint and cover all points in C_{ti}, the sum of these ratios for a true label C_{ti} becomes:

$$\sum_{j} \frac{|C_{ti} \cap C_{pj}|}{|C_{ti}|} = 1 \tag{4}$$

Repeating this for all true labels C_{ti}, the total C-Score is:

$$\texttt{C-Score} = \sum_{i=1}^{k} 1 = k \tag{5}$$

Thus, under converged clustering, the C-Score converges to k, the total number of true labels. The equation for C-Score is symmetric with respect to k and m, so in effect, the C-Score converges to the minimum of the two numbers in case of converged clustering, i.e. we can say that generally,

$$0 < \texttt{C-Score} \leq \min\{k, m\} \tag{6}$$

To provide a balanced and interpretable evaluation of clustering performance, the C-Score is normalized by the product of k and m:

$$\text{Normalized C-Score} = \frac{\texttt{C-Score}}{k \times m}.$$

This normalization yields a metric that ranges from 0 to 1, where a score of 1 indicates perfect clustering alignment between the true labels and predicted clusters. By dividing the C-Score by $k \times m$, the normalized C-Score accounts for both the number of true labels and clusters, facilitating meaningful comparisons across different datasets and clustering solutions. This approach ensures that the evaluation metric is sensitive to both the richness of the true label space and the granularity of the clustering outcome. Hence, after normalization, the limits for converged clustering would be

$$0 < \text{Normalized C-Score} \leq \min\left\{\frac{1}{k}, \frac{1}{m}\right\} \tag{7}$$

Since for every execution, the number of clusters are varying, it is impractical to consider only the maximum C-Score. Hence, after obtaining the normalized C-Score, it is subtracted from its upper limit to calculate the Convergence Difference. This difference, referred to as the *C-Score Convergence Difference*, is minimized to determine the optimal algorithm and number of clusters going forward.

3.4 Post-Clustering Layer Alternatives

Incorporating a post-clustering layer into the methodology introduces an additional stage of analysis that can refine the results obtained from the initial clustering process. This layer can utilize classification algorithms, such as support vector machines (SVM), decision trees, or neural networks, to assign specific labels or categories to the clusters formed in the preceding step. Alternatively, other algorithms like anomaly detection, regression models, or rule-based systems can be applied, depending on the objective of the analysis. The purpose of this post-clustering stage is to leverage the inherent structure identified by the clustering to perform a more granular classification or to extract further insights by combining cluster properties with supervised learning methods. Such a hybrid approach enhances the interpretability of clusters by associating them with known classes or patterns, enabling a more comprehensive understanding of the data.

4 IMPLEMENTATION

4.1 Dataset Overview

The data used in this study consists of images from the Severstal Steel images (SS) dataset (Grishin et al., 2019). It consists of images of steel produced at the end of a production line obtained from the popular data science website Kaggle. Although this dataset comprises of multiple defect classes, the authors have combined all defect class labels as a single near-label to evaluate the methodology outlined in this study.

For practical application to avoid high time complexity, the authors chose to take 1000 images for training. These contain almost an equal amount of two classes, namely near-label 0 corresponding to non-defect images and near-label 1 corresponding to defect images. Each image consists of dimensions $256 \times 1600 \times 3$ grey-scaled for identifying different pixels in the images that may or may not correspond to a fault. An example image is shown in Figure 3 where the grey-scaled image shows a visible impact region.

4.2 Model Training

The framework is only dependent on CNN encoders for feature extraction, whereas the training is only undertaken during the clustering process. This leads to a reduction of the run-time of the outlined framework, thereby making it efficient for inclusion in a larger pipeline.

Post feature extraction, the clustering algorithms are run for a fixed range of clusters from 2 to 50. For each algorithm and number of clusters, the corresponding metrics from Section 3.2.2 are calculated leading to a criteria for selection. Principal Component Analysis (PCA) is applied to the input data to improve time complexity of applying multiple clustering algorithms in a sequence. However, the PCA is only kept to minimum of the size of the input, i.e. either the number of samples or features.

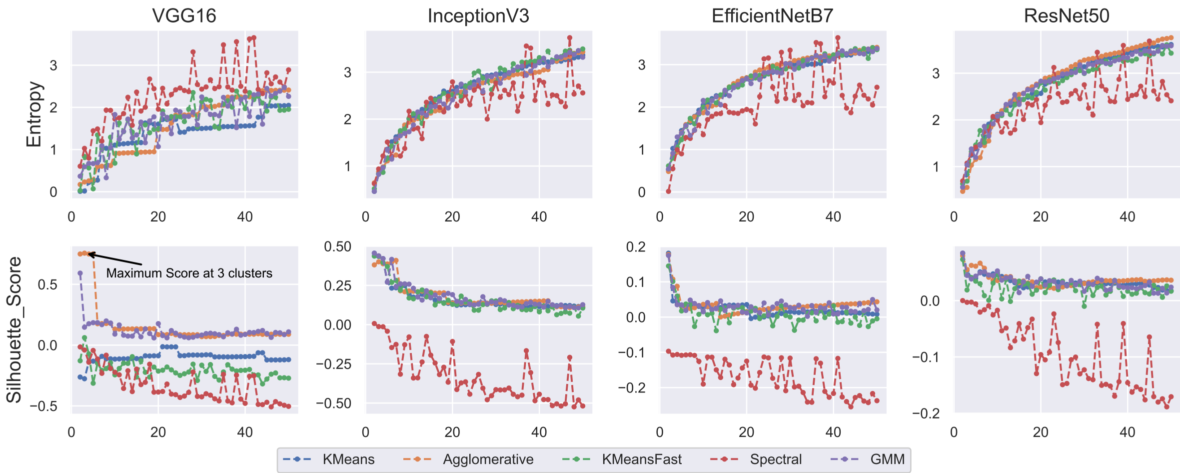


Figure 2: Entropy and Silhouette Scores for the Main Clustering on CNN Encoder-Based Data.

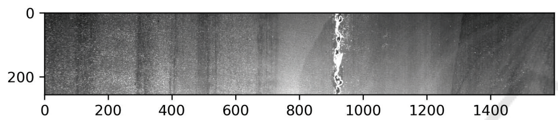
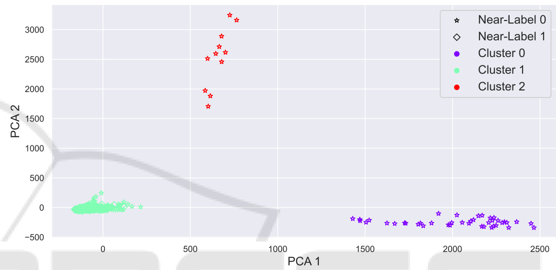


Figure 3: Grey-scaled Pre-processed Image from SS Dataset.

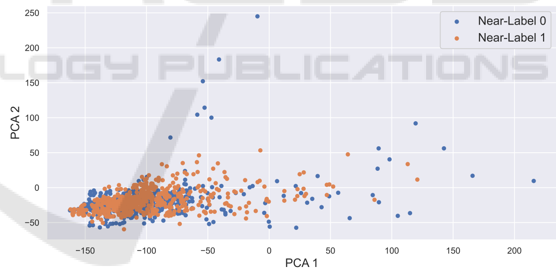
We follow the 'Min-Max' criteria established in (Mathias et al., 2019) to exclude those algorithms that exude larger entropy and minimum silhouette scores. This leads to a range of clusters that can be contended with for further analysis. This process is repeated for every level of sub-clustering, however, the use of 'Min-Max' criteria or the C-Score is subject to the user based on the observed clusters.

5 RESULTS

The extraction of features using the CNN models reasonably is the primary time-consuming step in the proposed approach. However, it eliminates the need for computationally intensive clustering on raw inputs directly. Additionally, this approach avoids the complexity and resource burden of designing custom feature extraction methods tailored to specific datasets. The trade-off lies in leveraging these optimized models, whose outputs can exceed expectations by delivering robust, transferable representations that enhance the efficiency and reliability of subsequent clustering tasks. This balance between computational efficiency and effective feature extraction highlights the practicality of using these available models.



(a) Agglomerative Clustering of Reduced Features with 3 Clusters.



(b) Cluster 1 which contains all Near-Labels 1.

Figure 4: Main Clustering Best Performance.

5.1 Main Clustering Evaluation

The performance of the CNN encoders, evaluated using Min-Max approach of Entropy and Silhouette Score is presented in Figure 2 with entropy shown in the top row and silhouette scores in the bottom row.

Entropy. For all architectures, entropy consistently increases as the number of clusters grew, indicating higher disorder in cluster assignments as more clusters were formed. This trend was observed across all models, reflecting a general increase in complexity with additional clusters.

Silhouette Score. The silhouette scores follow a different pattern, peaking early and progressively decreasing as the number of clusters increased. For VGG16, the maximum silhouette score is observed at 3 clusters, as indicated by the labeled point in the figure. This suggests that the optimal number of clusters for VGG16 is 3, offering the best separation and cohesion among clusters. Beyond 3 clusters, silhouette scores declined for all networks, and in several cases, the scores dropped to negative values, particularly for the red cluster, indicating poor clustering quality.

Across all models, the initial silhouette scores indicated strong clustering quality, but the drop in performance as the number of clusters increased suggests that over-clustering likely occurred, leading to less meaningful groupings. The results show that VGG16 was the most effective at maintaining cluster quality, with 3 clusters yielding the most optimal outcome in terms of the silhouette score.

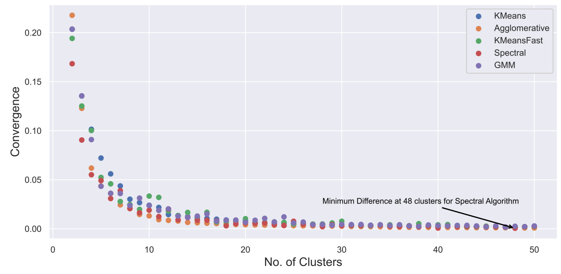
For the purpose of visualization, the extracted data from VGG16 was further reduced to two components based through PCA. Figure 4 presents the best performing clustering algorithm, i.e Agglomerative on this data. Figure 4a) shows the distinctly clustered group of points into 3 well-distanced groups while Figure 4b) shows the focus cluster 1 which contains all the near-labels 1 of the original data.

5.2 Sub-Clustering Evaluation

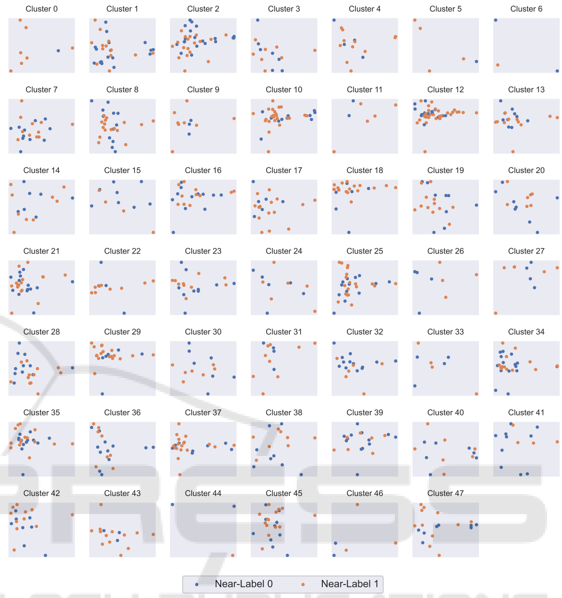
Once the main clustering is done, the necessity of sub-clustering is dependent on the user. In this case, we applied the same methodology as in the main clustering. However, the evaluation at later levels is done either using Min-Max criteria or through C-Score Convergence Difference to identify which algorithm and what number of clusters are effective in discerning the near-labels in smaller groups. For example, after main clustering, level 1 clustering yields the results shown in Figure 5.

The C-Score Convergence Difference yields the minimum value for the Spectral algorithm with 48 clusters of main cluster 1 in the previous step (see Figure 5a)). The 48 clusters are shown in Figure 5b) which shows some dominant groupings with respect to near-labels. For example, clusters 0, 5, 8, 11, 22, 29, 43, and 46 mainly consist of near-label 1. Many clusters are still visibly indistinguishable with respect to the initial labels like clusters 2, 12, 21, 25, and 42. Since elements are grouped into small sizes, it becomes easier to work with smaller clusters to make a certain distinction through near-labels. For this reason, cluster 25 was chosen for the next sub-clustering.

As more clusters are formed, the C-Score Convergence Difference will eventually lead to the minimum score of 0 as the number of clusters matches the number of samples. In level 2 clustering, since cluster 25 only had 33 points, the authors chose to only work with 2 clusters, same as the number of near-labels. The corresponding clusters for each algorithm are shown in Figure 6. At this stage, the decision to select the appropriate clusters is dependent on the user. For example, GMM clustering for two clusters shows that cluster 1 must be inspected deeply, since cluster 1 mainly contains near-labels 1. However, GMM also exihibits the best clustering since most of near-labels 1 are entirely contained in cluster 1 while cluster 0 has a majority of near-labels 0. Observably, cluster 0 can be well separated further into 2 clusters with sub-clustering imminent for cluster 1. The authors have, however, chosen to halt the sub-clustering process at this level to avoid redundancy.



(a) C-Score Convergence Difference at Level 2 Clustering.



(b) Best Performing Spectral Algorithm with 48 Clusters.

Figure 5: Level 1 Clustering of Cluster 1 from Main Clustering.

(a) KMeans Clustering.

(b) Agglomerative Clustering.

(c) KMeansFast Clustering.

(d) Spectral Clustering.

(e) GMM Clustering.

Figure 6: Level 2 Clustering of Cluster 25 from Level 1 Clustering.

5.3 Observations

We make some observations from the employed scheme of sub-clustering:

- In main clustering, it is evident that using just available CNN encoders for feature extraction is sufficient in some cases to mark a certain boundary between images, as in Figure 4. However, this does not guarantee that the extraction or clustering gives optimal results.
- The C-Score is a set-related metric, hence it can be used to decide the similarity between collections. However, in this scenario, it is only useful when some initial distinction is available to judge the quality of clustering.
- In the absence of near-labels, sub-clustering is purely visual, with only distances between samples determining cluster structures and no other information.
- For sub-clustering, since the number of samples is reduced, this scheme can also incorporate other feature reduction techniques on the same data within sub-clustering levels. This can help in improving clustering. However, this has not been implemented in this study to avoid an overload of framework execution.

6 AN END-TO-END LABELING SCHEME IN PRODUCTION ENVIRONMENTS

To augment the undertaken experiment, an end-to-end labeling scheme can be constructed using only the basic tools of machine learning, as shown in Figure 7. In any application building, the finer front-end architecture is dependent on the end-user/client. However, on the production floor, a simple working application that comprises of database, clustering algorithms and neural networks is beneficial for the operator to immediately inspect product quality without waiting for deeper quality inspection. The operator also benefits through clustering in identifying varying processes from the captured images. For example, as seen in Figure 4a), three patterns are distinctly recognized by the algorithms with one cluster completely containing the defect labels. The other groups of points can be further investigated for mechanical properties or process related parameters issues.

To complete such a predictive application, an application workflow that begins with capture of image, then flows through a storage such as database, passed through a pre-trained model prediction followed by a custom classification to obtain groups is a fairly easy task as long as the architecture remain in one location such as a local PC. The complexity however increases when the distribution of layers is across different platforms such as cloud storage for data, followed by remote location of analysing PC and output transmission to correct users/clients. This falls in the domain

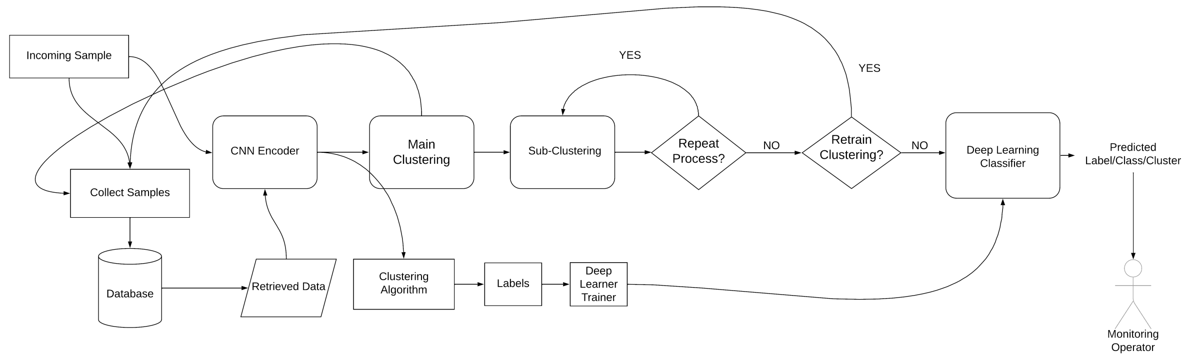


Figure 7: An End-to-End Labeling Scheme with Sub-Clustering.

of Internet-of-Things which can be investigated in future research.

7 CONCLUSION

A labeling scheme involving a transfer learning approach utilizing available CNN encoders for feature extraction of industrial images and transferring them to clustering algorithms for cluster labels is presented. The layers of the architecture can be varied according to the user requirements, for example, replacing CNN networks with custom nets which is an additional advantage. Similarly, the clustering algorithms used can also be varied depending on the problem requirement. The results that are presented show that such a scheme if developed for production floors, it can assist the monitoring operators in detecting varying patterns which are critical in avoiding defects or anomalies. Because pre-trained models are used, it is clear that analysis becomes more efficient and robust as no customized feature extraction or advanced knowledge of process or product is required to label/group the incoming samples. The applicability of this approach can also be considered to be very broad since images are fed directly into the feature extraction steps without any pre-requisite condition of their origin. An outlook on the development of a working application on the production floor is also discussed. The focus in this study is not improvement of proposed methodology, rather the practical application in real-time scenarios. Improvement of clustering algorithms can be further undertaken in a deeper studies involving multiple datasets, varied CNN models and comparisons with established baselines. Some topics related to cluster convergence through C-Score and cluster profiling of image-based data can be investigated in future works.

REFERENCES

Biswas, A. and Jacobs, D. (2014). Active subclustering. *Computer Vision and Image Understanding*, 125:72–84.

Cardoso, D. and Ferreira, L. (2021). Application of predictive maintenance concepts using artificial intelligence tools. *Applied Sciences*, 11(1).

Chang, J., Wang, L., Meng, G., Xiang, S., and Pan, C. (2017). Deep adaptive image clustering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5880–5888.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1.

Daugman, J. G. (1985). Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J Opt Soc Am A*, 2(7):1160–1169.

Fan, C.-L. (2024). Multiscale feature extraction by using convolutional neural network: Extraction of objects from multiresolution images of urban areas. *ISPRS International Journal of Geo-Information*, 13(1).

Fu, K., Dang, X., Zhang, Q., and Peng, J. (2024). Fast uois: Unseen object instance segmentation with adaptive clustering for industrial robotic grasping. *Actuators*, 13(8).

Grishin, A., V, B., iBardintsev, inversion, and Oleg (2019). Severstal: Steel defect detection.

Hartner, R., Komar, J., and Mezhuyev, V. (2022). An approach for increasing the throughput of a cnn-based industrial quality inspections system with constrained devices. In *Proceedings of the 2022 11th International Conference on Software and Computer Applications*, ICSCA '22, page 179–184, New York, NY, USA. Association for Computing Machinery.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Li, W., Hannig, J., and Mukherjee, S. (2021). Subspace clustering through sub-clusters. *Journal of Machine Learning Research*, 22(53):1–37.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Mathias, S. G., Großmann, D., and Sequeira, G. J. (2019). A comparison of clustering measures on raw signals of welding production data. In *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*, pages 55–60.

Oyelade, J., Isewon, I., Oladipupo, O., Emebo, O., Omogbadegun, Z., Aromolaran, O., Uwoghiren, E., Olaniyan, D., and Olawole, O. (2019). Data clustering: Algorithms and its applications. In *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*, pages 71–81.

Saad, N., Sabri, N. M., Hasan, A., Ali, A., and Saleh, H. M. (2015). Defect segmentation of semiconductor wafer image using k-means clustering. In *Design and Development of Sustainable Manufacturing Systems*, volume 815 of *Applied Mechanics and Materials*, pages 374–379. Trans Tech Publications Ltd.

Silva, W. and Capretz, M. (2019). Assets predictive maintenance using convolutional neural networks. In *2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 59–66.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Strang, G. and Nguyen, T. (1996). *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Philadelphia, PA.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Wang, X., Wang, Z., Wu, Z., Zhang, S., Shi, X., and Lu, L. (2023). Data stream clustering: An in-depth empirical study. *Proc. ACM Manag. Data*, 1(2).

Xie, J., Girshick, R., and Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 478–487, New York, New York, USA. PMLR.

Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678.

Zhang, Q. and Chen, J. (2022). Distributed learning of finite gaussian mixtures. *Journal of Machine Learning Research*, 23(99):1–40.