Active Deep Clustering: Exploratory Analysis to Assist in Decision-Making on Incremental Label Morphing Datasets

Connor Clarkson[1 [0000−0001−7191−0886]], Michael Edwards[1 [0000−0003−3367−969X]], and Xianghua Xie[1 [0000−0002−2701−8660]]

Computer Science Department, Swansea University, Swansea, United Kingdom
{c.a.clarkson, michael.edwards, x.xie}@swansea.ac.uk

Abstract. Many supervised-based training schemes rely on the need to have a single associated label for each data sample within a set. Where the goal is to learn different levels of granularity of the data in an implicit form, in the context of neural networks, this is accomplished with different layers to extract features at distinct levels. In this work, we explore more explicit labelling structures where each sample has multiple labels forming relationships at both an abstract and fine-grained level, producing a tree for each associated data sample. This novel type of training scheme utilises a refinement strategy based on deep clustering approaches to detect the colliding and splitting of clusters where each is assigned a label. Experts can then be queried to determine if those colliding clusters should belong to a single label, or alternatively, if they are splitting, should we create new labels, forming an active component of our training scheme. Colliding clusters form a parent label while splitting clusters form sibling labels within the tree structure. By utilizing a tree data structure to represent labels at different levels of granularity, we can invoke explicitly defined relationships and dependencies to form a more structured and interpretable representation of data. Instead of treating data as flat, homogenous sets, we allow for the exploitation of hierarchical relationships and leverage inherent structure to improve data efficiency. We present a case study of the approach applied within the steel manufacturing domain, where quality control remains an active challenge due to morphing labels as products move down the production line.

Keywords: Dataset Refinement · Training Schemes · Tree-based Modelling.

1 Introduction

Deep learning has made significant strides in many domains including manufacturing, where many processes are performed automatically [12]. In steel manufacturing, however, quality control remains an active challenge due to the complex nature of steel, how it gets produced for different targeted purposes, and the type

of data that often gets produced in these manufacturing plants. The composite nature of steel forms defects labelled into one class but contains many shared features of other classes, resulting in difficulty of optimising such functions [13]. It is common to see large defects form into different types of classes as the damage spreads spatially throughout the coil of steel. Therefore, some but not all classes are not discrete but rather a threshold between many others. Many have approached these challenges from various directions [13]. In this work, we examine the performance of generative models for the task of deep clustering within the manufacturing domain. Our focus is threefold: first, we explore the effectiveness of these models in detecting and categorising steel defects. Secondly, we present a series of exploratory analysis approaches through a graphical user interface allowing for human-in-the-loop approaches during training. Thirdly, we introduce a training scheme as a refinement strategy utilising deep clustering to build hierarchical labelling datasets. As we have active learning components throughout this work with a focus on highly skilled and technical industries like steel manufacturing, an interface is designed to allow domain experts to provide reasoning on how these models detect defects. This forms a bridge of interoperability between complex algorithms and practical applications.

2 Related Work

Like many areas in machine learning, deep networks have contributed to cluster analysis to form deep clustering. Using rich feature representations that have a hierarchical inductive bias has shown to be a beneficial component in supporting traditional algorithms. Generative methods are of particular interest due to the constraint of focusing only on the essential features of the dataset instead of relying on the targets. We focus on deep clustering approaches where learnt representations are used for the task of clustering. We do not use features of raw data or traditional dimensionality reduction in these methods as it has been demonstrated that performance is limited as the dimensionality increases, especially in computer vision [5]. Instead, we use non-linear generative methods that propose a set of latent variables used for clustering. Existing work in deep clustering is classified into 2 categories: *separated clustering* and *embedded clustering*. Separated clustering first learns a lower dimensional representation space of a given dataset, to then perform cluster analysis on those representation, in a 2 step process [10, 7]. Such schemes take advantage of deep networks to map data into a representative feature space, but the learning of this space and the clustering are two separate processes, due to the objectives not being optimised jointly. Embedding clustering takes the objective of clustering into the training scheme, forming a deep clustering task and a new framework [9, 14, 5]. Most approaches in embedding clustering form a closed-loop 2-step process where a deep network maps the original data into a representative feature space and then performs clustering on the current embeddings. The approach alternates the steps in a loop until convergence by optimising a Kullback Leibler Divergence (KL) loss to enforce a self-training target distribution. The limitation with these approaches

is that by using KL the distance between a sample A to sample B is not necessarily the same as B to A. This results in directly influencing the quality of the clusters being generated. The second approach within embedding clustering is using deep networks with clustering on the embedding space simultaneously, by optimising the reconstruction and a clustering loss in a single-step process [2, 1]. Forming cluster analysis as a semi-supervised task by providing varying degrees of supervision has the potential to enhance performance as a form of domain knowledge inference into the training scheme. This domain knowledge could be from utilising labelled data, or a form of human supervision via iterative user feedback in an active learning task [4, 3, 8, 11, 1]. Incorporating supervision of some form into the training scheme involves a great deal of time-consuming effort in labelling such data, because in many situations if clustering is the proposed method of processing then usually no labelling is given. This results in the potential direction of active learning, where we may refine data over time or simply use it to enhance the performance of the clustering. In this case, a decision needs to be made by a human, in many domains where a higher form of expertise is required then more information is needed to be given to the human in order to make an accurate decision.

3 Methodology

Our methodology is an iterative process of exploratory analysis where we first experiment with deep clustering at varying levels of supervision and apply active learning by querying expert steel engineers. Secondly, we focus on dataset refinement to form hierarchical labelled datasets where we start with flat homogenous labels. Finally, we then evaluate these new hierarchical labelled datasets with their flat counterparts.

3.1 Exploratory Approach

We introduce an exploratory approach to assist in decision-making by a human for an active deep clustering task. For ease of use, we build a graphical user interface with customisation options for different views as well as different interaction techniques. This allows the expert to not only get some insight into what the selected model is deciding based on the current embedding space but also to assist in making decisions. In this section, we introduce a manufacturing dataset that our experiments are based around, the graphical user interface and the active learning experiment.

Throughout the set of experiments, we evaluate this work on two datasets. The first is MNIST, while the second is steel data from quality control within a steel manufacturing plant, which was captured during the cold rolling process. The dataset consists of $5,000$ grayscale images, with each containing between one and three defects. As steel is produced for a wide variety of use cases, different rankings of importance for features within the defect change. One reason for this is due to the composite nature of steel and how it is targeted for a given

purpose. For this reason, we use 4 different types of steel coils, where each coil has 32 cameras capturing different angles and positions along the mill. Labelling is provided as a Region of Interest (ROI) over the defect. These labels should be used only as a suggestion due to the many errors, some defects do not have an ROI, while others have more than one defect in the ROI. The variance in types of steel and the accuracy of the labels motivate the need for a semi-supervised approach to model such data.

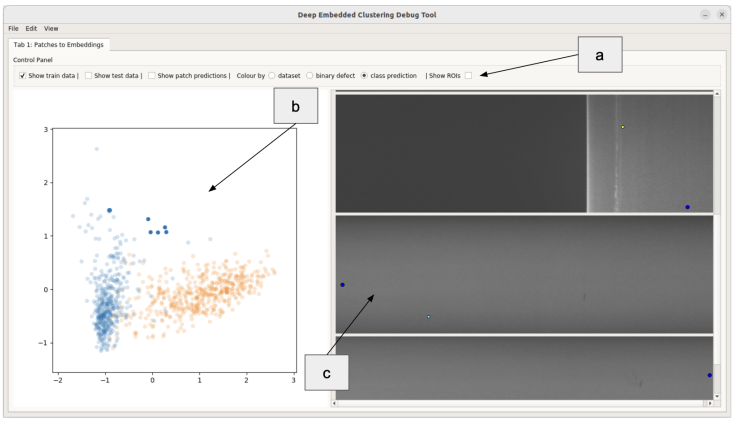

Fig. 1. Explore current embedding space via a selected model in the user interface. This embedding space changes over each session of labelling in an active learning experiment. *a*: the control panel of this view, where we customise how we show *b* and *c*. *b*: a selected dimensionality reduction technique is performed on the embeddings to view in 2-D. Users can select different points that represent patches of images and their location is then shown in *c*. *c*: segments of steel for different clusters are shown. The dots hovering on the image are the locations of the selected points from *b*.

The initial view of the graphical user interface is shown in figure 1. Experts can select different models based on the amount of supervision given as part of that experiment, some models also do not include a clustering training scheme as a form of comparison. Experts interact with the scatter plot representing an embedding space from the model. Note that the embeddings have been reduced to 2-D from a selected dimensionality reduction technique. Each point in this scatter plot is a patch from the dataset, where experts can right-click to see extra information such as which coil it comes from, the segment of steel within that coil, patch location and its current label if it has one. By selecting different points, experts can then view the location of the patch in the image via section *c* of the figure. Within the scatter plot users can use a lasso section on a group of points, individual selection of points or a multi-selection of points. Showing the global context of patch location is important in domains like steel defect detection due to the changing nature of defects as they move across the conveyor

belt. By showing its location, experts can examine if the defect morphs into a different one in a different part of the image. This view gets updated as different labelling sessions are provided. After each labelling session training of the model is performed until convergence. This of course changes the embedding space and therefore the user interface needs to respond accordingly. As successive labelling sessions are performed more densely packed clusters can be seen in this view. Sub-clusters can also be viewed in that different segments of the cluster will form smaller groups, implying a hierarchical labelling structure.

The second view of the interface is grouped into user interaction and exploring different dimensionality reduction techniques. In this view, experts can upload images to the interface or select segments from the dataset. The experts can then perform inference on patches of the selected image based on the type of user interaction.

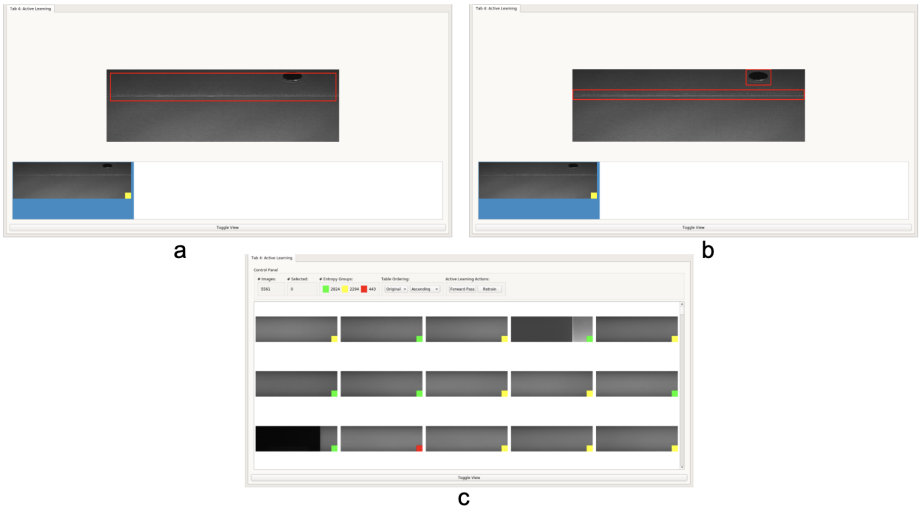

Fig. 2. We perform an active deep clustering task where supervision information is provided as part of the training. *a* shows a segment of steel before it was relabelled by a human. The coloured box in the corner represents the level of uncertainty. The red bounding box is the current label. In *b* a human has relabelled the segment. *c* is an image viewer of a selected dataset with its control panel for different ordering options, performing a forward pass, and re-training of the selected model.

The final view of the interface forms the active learning experiments, where experts can relabel different images of the dataset. This is shown in figure 2. At the start of the experiments, experts are shown the interface in section *c* of the figure, this represents the image viewer. A forward pass of the dataset is performed and a colour is given to each image. The colour represents the average entropy of the softmax predictions, it is an average because we sample each

image n times, creating n patches. Red refers to the image being uncertain to the model, green means that the model is confident in its prediction while yellow falls in the middle of these two options. Therefore, samples that are indicated as red or yellow may require further inspection by an expert. The image viewer allows us to sort images, see the total number of each entropy group, as well as perform additional forward passes or train the model until convergence. Any newly labelled samples get added to the pool of supervision samples. Experts can then fine-tune with this larger pool of samples. Once complete the interface updates to the newest version of the model. Allowing experts to explore the different views again. Sections a and b show a relabelling of a selected sample from the image viewer. Experts can remove labels, add new labels or change the current shape of the label.

3.2 Building Labelling Systems

An application of clustering is building labelling systems that can then be utilised either in an end-to-end or in a two-stage approach to supervised problems. In this section, we explore the aim of identifying similar images based on different levels within the label hierarchy. We cluster embeddings in a semi-supervised fashion and identify the colliding clusters. A shortest path is then generated between clusters, synthetic data is uniformly generated around the middle point of that path. Experts label this data, which is added to the supervised component of clustering. This process is repeated by changing the shortest path to include one of the newly labelled image samples from one of the clusters. Label hierarchies in deep learning can improve accuracy but determining the appropriate level of granularity for the hierarchy is challenging.

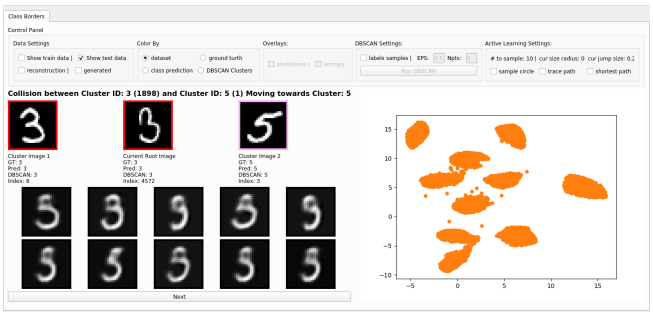

Fig. 3. The graphical user interface for labelling generated samples based on the path between two centroids. We detect colliding clusters using a semi-supervised extension to DBSCAN. Initial centroids are labelled and we perform an algorithm around these centroids. If a bridge can be made between different centroids this forms a path and we sample in the middle of that path. The user can then label those generated samples.

We utilise a semi-supervised density-based clustering algorithm which is an extension of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) called SSDBSCAN [6]. Labelled data points are treated as separate classes, and the algorithm tries to find the optimal clustering of the remaining unlabelled data points. The labelled data points serve as reference points to guide the clustering process. We loop over each labelled point performing DBSCAN to form clusters. Figure 3 displays the graphical user interface for this active learning experiment. DBSCAN builds a graph based on the density. By clustering the samples based on a labelled sample, we may hit another labelled sample that should be its cluster. This forms a bridge between the two centroids. We display this path to the user in the interface, and we then generate samples in the middle of the path. These generated samples are then shown to the user. In section 3.3 we demonstrate two different ways of generating these samples to be shown to the expert. Once the expert has labelled the generated samples to either be part of centroid one or centroid two, we then build a new path starting with the sample that is closest to centroid two. Samples that have not been labelled by the user are ignored. By following this loop over generating new paths to centroid 2 we can explore the boundary of that centroid. Once enough experts have labelled the generated samples, the major decision is then chosen to detect if the two clusters should become one and thus a new label is created.

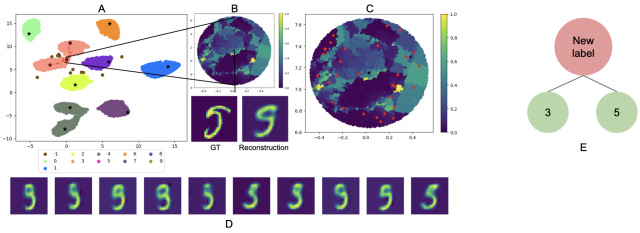

Fig. 4. Once a path has been made we generate samples. We perform a uniform sampling approach to measure uncertainty. We then select the most uncertain samples within an area to be labelled by the expert. Once enough samples have been labelled, a majority decision is made on whether the colliding clusters should form a new label.

3.3 Sampling Strategies

Sampling refers to the location where we select our generated samples to be labelled by experts. We explore two approaches: adaptive sampling and uncertainly-based sampling. In adaptive sampling, we have a ϵ which is the initial size of the sample area at the starting point after centroid 1. It then decays after each jump to a new point. We also have a minimum sample size size referred to as ϵ_{min}. As we get closer to the border of a new class the more uncertain the model will generally become, due to lacking samples within that region as well as classes

not being discreet but rather a threshold. Adjusting ϵ we can start making more fine-grained labels. Uncertainty-based sampling computes the samples within a radius defined by the point of interest. At the start of the experiments, this is the middle point of the path between both centroids. We accomplish this by generating synthetic samples via a generator model and then computing the class probabilities. We select n synthetic samples and choose m most uncertain ones. These uncertain samples are then labelled by the user.

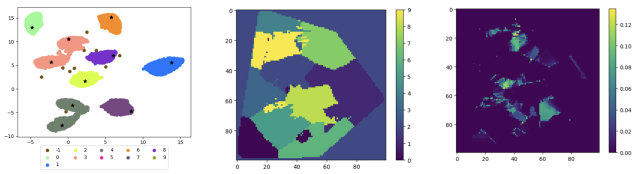

Fig. 5. We compute the uncertainty of a circle defined by the point of interest and the radius. This is computed by generating synthetic samples via a generator and then computing the class probabilities. We then select n samples and choose m most uncertain ones. These are then labelled by the user.

4 Experiments

In this section, we discuss the implementation and design structure of our experiments. We first explore our data generation pipeline which creates batches of local patches between the different steel coils. This follows a discussion on learning representations with a clustering training scheme. We also explore adding forms of supervision into the training to form better representations. This supervision can be from the labels or a human in an active learning setting.

A common challenge in datasets that have an ROI labelling system is balancing between the background and the object inside the ROI, this becomes problematic when there are parts of the object's spatial structure leaking outside of the ROI, more than one object inside the ROI, and objects which are not labelled. The spatial nature of most objects also means that there are many areas within the ROI which is background. To help in learning better spatial structure of the defect, we uniformly sample pixel locations on steel segments between defects and background. The pixels from the centre of a patch and the current label are given based on whether it is located within the ROI or not, forming a binary labelled dataset of images. As different steel coils are targeted for a specific customer purpose, the defects and background features can change and as a result, we split into training, testing and validating data over all segments of steel.

4.1 Learning Representation

The training scheme involves using the k-means algorithm incorporated as part of a loss [2]. This forces the embeddings proposed by the autoencoder to form an embedding space where the latent variables have a high signal if the samples are similar. We minimise the distance between samples and a given assigned centroid in the embedding space

$$E_2 = \lambda \times \frac{1}{2N} \sum_{i=1}^{N} || \ h^t(x_i) - c_i^* \ ||^2 \tag{1}$$

where N is the number of samples, λ is a clustering weight scaler to determine the amount of contribution the clustering has on the overall loss function, $h^t(\cdot)$ is the internal representation of a sample at t iteration, and c_n^* is the assigned cluster to the n^{th} sample. The overall loss function is then the combination of E_2 and a reconstruction loss E_1, which is the mean squared error for our experiments. At each epoch, we minimise the reconstruction while keeping the cluster centres fixed. After each epoch and we have obtained a new internal representation for each sample, we assign each one to a new closest centroid:

$$c_n^* = \arg\min_{c_m^{i-1}} || \ h^t(x_n) - c_m^{t-1} \ ||^2 \tag{2}$$

where c_m^{t-1} is the cluster centre from the previous epoch. Like k-means we then update the cluster centers using the sample assignment:

$$c_m^t = \frac{\sum_{x_n \in c_m^t - 1} h^t(x_n)}{\sum c_m^{t-1}} \tag{3}$$

where c_m^{t-1} are all samples becoming to m^{th} cluster, $\sum c_m^{t-1}$ is the number of samples that belong to m^{th} cluster. We also have the option to use some supervision in the training scheme then we add a third loss E_3 that we need to minimise, in our class we use the cross-entropy loss function:

$$E_3 = -\sum t_{i,j} log(p_{i,j}) \tag{4}$$

We use a modified version of the autoencoder architecture proposed by Alqahtani *et al.* [1]. We still target the clustering loss at the internal representation but to inject some supervision we use the learnt features from the encoder by attaching a classification head to it. The autoencoder architecture consists of three convolutional layers, followed by two dense layers which have k neurons, these are the hidden representations for each cluster during the training process. The representations are fed into clustering loss. For supervision, we have a dense layer with a softmax activation attached to the layer before the internal representation layer. For the decoder we use a dense layer so that we can reshape the representations, this then follows three deconvolutional layers. Rectified Linear Unit (ReLU) is the activation function of choice throughout the network. This forms an end-to-end training scheme where supervision knowledge either from the labels or from a human is used across the learning process, this also creates compact and discriminative clusters using the k-means algorithm.

5 Results

We evaluate our results using clustering accuracy, which measures how well the clusters match the known labels. Unlike classification accuracy, clustering algorithms produce arbitrary label assignments thus we need to find the optimal mapping between predicted clusters and the labels using the Hungarian algorithm:

$$acc = \frac{1}{n}\sum_{i=1}^{n}\delta(y_i, map(\hat{y}_i)), \tag{5}$$

where n is the number of samples, y_i is the label for sample i, $\hat{y}_i$ is the predicted cluster for sample i, $map(\cdot)$ is the optimal mapping function, and $\delta(y_i, map(\hat{y}_i))$ equals 1 if $y_i = map(\hat{y}_i)$, 0 otherwise.

Table 1 is our list of results from the deep clustering experiments. We use MNIST as a baseline, which is commonly used within deep clustering literature. We also use the patch-based steel dataset provided by a steel manufacturer. For both datasets we use three different amounts of labelled samples to form our supervision experiments, these are 20%, 50% and 100% of the training dataset. We also use the graphical user interface to form our active learning experiments where if a sample of the MNIST or a patch of the steel dataset gets labelled then this is added to the pool of labelled samples seen by the model during training. The supervision experiments also include both convergences on the reconstruction of the input as well as clustering of the internal representations. In general, all experiments perform well when using learnt feature representations to perform clustering compared to without. The features of MNIST are quite well defined even if we disregard the spatial aspect of the data and simply flatten it into a 1-D vector, we get a good accuracy of 59.98%. By using a clustering loss within the optimisation we get more compact clusters and this only improves with more supervision. When running supervision via a human to label we get a slightly worse result, this is because some of the labels made by a human are incorrect, which also demonstrates a disadvantage with active learning in that the inference from a human is very strong and if it is indeed incorrect then it can have a large effect on the optimisation. This could have been solved with a majority vote approach to labelling by humans. The steel dataset shows to be more of a challenge than MNIST, this is because we sample patches based on selecting pixels in and out of the ROI. There will be many samples with a lot of background within a patch that is labelled as a defect. The spatial geometry of the defect also affects the reconstruction in other ways because many defects are relatively unique in their features. Adding more supervision helps in general with the best being 73.23% but we get a much better score of 96.32% if we incorporate samples labelled by a human. In this case, we use all samples that have been labelled which is why the table states 100% labelled. This is not all labelled samples of the data like it is for MNIST.

Modelling hierarchical relationships within our datasets forms part of our future work but we provide a second set of experiments as a proof of concept over the single labelled version datasets. This is shown in table 2. Each experiment

	MNIST	Steel
k-means	59.98%	27.2%
Reconstruction	82.72%	36.98%
+ clustering	84.84%	58.5%
+ supervision (20%)	89.99%	54.21%
+ supervision (50%)	92.12%	63.08%
+ supervision (100%)	98.78%	73.23%
+ supervision via human	98.43%	96.32%

Table 1. Table of results for clustering quality with the clustering accuracy metric. For the supervision results, we converge with both incorporated reconstruction and clustering loss terms. The percentage refers to the amount of labelled data we use during the experiment. The only exception to this is when we perform an active learning experiment, all labelled data from a human is used during the training process.

uses a 10-fold of the dataset for both MNIST and the steel. The table is split into using single and hierarchical labels belonging to a sample. MNIST did not perform as well on the hierarchical experiments compared to single but this was to be expected. Our methodology tries to find labels that should be merged into one, or alternatively, split labels into two. MNIST is a very clean dataset with not many obvious errors. The results from the steel dataset are more realistic than what we would find in many industries, with more errors and less cleanly labelled data. Refinement of labelling is useful both from a performance point of view but also from an interpretability viewpoint as we can demonstrate to the engineers the changing of labels which forms a tree as new data is passed into the methodology.

	Single				Hierarchical			
	Accuracy	Recall	Precision	F1	Accuracy	Recall	Precision	F1
MNIST	0.92 ± 0.001	0.95 ± 0.004	0.94 ± 0.0003	0.94 ± 0.002	0.84 ± 0.01	0.75 ± 0.01	0.75 ± 0.002	0.75 ± 0.004
Steel	0.37 ± 0.03	0.39 ± 0.03	0.37 ± 0.03	0.19 ± 0.01	0.89 ± 0.002	0.80 ± 0.01	0.78 ± 0.005	0.76 ± 0.001

Table 2. Table of results on using single labels associated with a sample compared to using a hierarchical set of labels associated with a simple. Each experiment uses a 10-fold separation of the data.

6 Conclusion

In this work, we proposed iterative exploratory process which ends with a methodology on dataset refinement from single label for each sample to a multiple label per sample dataset. This allows for the exploitation of hierarchical relationships between labels of the same sample, which can be leveraged as a new type of training scheme. We started this process by first exploring the effectiveness of generative models in detecting and categorising steel defects, presented a graphical user interface allowing experts to be in the loop of the training scheme as

well as exploring analysis at a current iteration. Then we introduced a training scheme as a refinement strategy utilising deep clustering to build hierarchical labelling datasets. This work was targeted in the steel manufacturing domain, where quality control remains an active challenge. Utilising more explicit forms of machine learning such as provided expert knowledge to the model within an active learning setting allows for improved data efficiency at similar performance to methods without such active approaches.

References

1. Alqahtani, A., Xie, X., Deng, J., Jones, M.: Learning discriminatory deep clustering models. In: International Conference of Computer Analysis of Images and Patterns (2019). https://doi.org/10.1007/978-3-030-29888-3_18
2. Alqahtani, A., Xie, X., Deng, J., Jones, M.W.: A deep convolutional auto-encoder with embedded clustering. In: 25th IEEE International Conference on Image Processing. pp. 4058–4062 (2018)
3. Basu, S., Banerjee, A., Mooney, R.J.: Semi-supervised clustering by seeding. In: 19th International Conference on Machine Learning (2002)
4. Cohn, D., Caruana, R., McCallum, A.: Semi-supervised clustering with user feedback. Constrained Clustering: Advances in Algorithms, Theory, and Applications (2008)
5. Guo, X., Liu, X., Zhu, E., Yin, J.: Deep clustering with convolutional autoencoders. In: International Conference on Neural Information Processing (2017)
6. Lelis, L., Sander, J.: Semi-supervised density-based clustering. In: IEEE International Conference on Data Mining (2009). https://doi.org/10.1109/ICDM.2009.143
7. Li, F., Qiao, H., Zhang, B.: Discriminatively boosted image clustering with fully convolutional auto-encoders. In: Pattern Recognition (2018). https://doi.org/10.1016/j.patcog.2018.05.019
8. Pedrycz, W., Waletzky, J.: Fuzzy clustering with partial supervision. In: IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) (1997). https://doi.org/10.1109/3477.623232
9. Song, C., Liu, F., Huang, Y., Wang, L., Tan, T.: Auto-encoder based data clustering. In: Ruiz-Shulcloper, J., Sanniti di Baja, G. (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
10. Tian, F., Gao, B., Cui, Q., Chen, E., Liu, T.Y.: Learning deep representations for graph clustering. In: Twenty-Eighth AAAI Conference on Artificial Intelligence. AAAI Press (2014)
11. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained k-means clustering with background knowledge. In: 18th International Conference on Machine Learning (2001)
12. Wang, J., Ma, Y., Zhang, L., Gao, R.X., Wu, D.: Deep learning for smart manufacturing: Methods and applications. In: Journal of Manufacturing Systems (2018). https://doi.org/10.1016/j.jmsy.2018.01.003
13. Wen, X., Shan, J., He, Y., Song, K.: Steel surface defect recognition: A survey. In: Coatings (2023). https://doi.org/10.3390/coatings13010017
14. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, PMLR (2016)