A GENERATIVE ADVERSARIAL NETWORK-BASED CONTINUAL OVERSAMPLING MODULE FOR AUGMENTING A TRAINING DATASET

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of priority to Singapore patent application no. 10202400853V which was filed on 25 March 2024, the contents of which are hereby incorporated by reference in its entirety for all purposes.

TECHNICAL FIELD

[0002] This application relates to a module and method for configuring a generative adversarial network-based continual oversampling (GAN-COS) module to augment a training dataset. The augmented training dataset may be used to train a classifier module to detect defects in a dataset. During the configuration of the GAN-COS module, datapoints associated with tasks in a training dataset are balanced using a data balancing module before the balanced dataset is then processed by a shared generator and a private generator of the GAN-COS module. The latent vectors produced by these two types of generators then undergo latent space oversampling and are used in the training of a discriminator of the GAN-COS module. The latent vectors may then be combined and used to form the augmented training dataset.

BACKGROUND

[0003] Within the semiconductor industry, it is important for defects to be reliably and accurately detected to ensure quality control. However, a fundamental challenge in the training of defect detection classifiers arises due to the significant imbalance between the defect class samples and healthy class samples in the training datasets. The underrepresentation of defect classes leads to gradient dominance, which negatively impacts the learning process of defect detection classifiers and this in turn affects their overall performance. Further, as new defect classes are introduced, trained classifiers or models will need to be retrained to incorporate the new defects with the previously learned defects. This incremental retraining process results in computationally expensive training cycles.

[0004] To mitigate the issue of the underrepresentation of defect classes in training datasets, those skilled in the art have proposed data oversampling techniques that enhance the representation of minority defect classes. In one approach, those skilled in the art have

proposed the use of generative adversarial networks (GANs) to address class imbalance due to GAN's ability to generate synthetic samples based on real samples. GAN-based adversarial oversampling techniques have demonstrated efficacy in generating synthetic defect samples that closely resemble real minority-class data, thereby improving the performance of classifiers by improving the training datasets of such classifiers. Those skilled in the art have also suggested the use of adversarial oversampling (AO) and data-space oversampling (DO) approaches as such techniques appear to be effective in handling high-dimensional, highly imbalanced, and large-scale multi-class tabular datasets. However, existing defect detection solutions that employ these techniques predominantly focus on predictive maintenance and do not account for the continual introduction of newly defined defects into the training dataset.

[0005] Hence, those skilled in the art have proposed continual learning methodologies that enable models to learn from new data while retaining previously acquired knowledge. However, these approaches often do not account for class imbalance, making them less suitable for use in defect detection applications where class imbalance is a key issue. Furthermore, imbalanced learning techniques that incorporate adaptive feature generation methods for continual learning tend to be limited to binary classification tasks and primarily operate within the data space, rather than in the latent space.

[0006] Despite advancements in defect detection, existing solutions fail to comprehensively integrate generative oversampling in the latent space, continual learning, and imbalanced learning in the creation of training datasets for defect classification. Hence, those skilled in the art are constantly looking for a solution that enables effective training of defect detection classifiers, where this solution may address class imbalances while ensuring that the classifiers may continually adapt to newly introduced defect classes.

SUMMARY

[0007] In one aspect, the present application discloses a computing module for configuring a generative adversarial network-based continual oversampling (GAN-COS) module to augment a training dataset. The disclosed computing module comprises a processing unit, and a non-transitory media readable by the processing unit, the media storing instructions that when executed by the processing unit causes the processing unit to provide the training dataset to the GAN-COS module where the training dataset comprises a plurality of tasks wherein each task

is associated with a specific category of defect types and comprises a set of defect classes. The processing unit then configures the GAN-COS module to, for each of the plurality of tasks in the training dataset perform the following processes. The processing unit balances, using a data balancing module, distribution of datapoints across defect classes to generate a balanced dataset and then applies, a shared generator of the GAN-COS module, to the balanced dataset to generate task-invariant latent vectors common across multiple tasks. The processing unit then selects and applies, a corresponding private generator of the GAN-COS module, to the balanced dataset to generate task-specific latent vectors, wherein the corresponding private generator is selected based on a category of defect types associated with the balanced dataset before performing, using an oversampling module, latent space oversampling to generate a set of synthetic latent vectors based on the task-invariant and task-specific latent vectors. The processing unit then trains a discriminator of the GAN-COS module using the set of synthetic latent vectors, the generated task-invariant and task-specific latent vectors, wherein the discriminator is configured to distinguish between real and synthetic representations, wherein the discriminator's output is used to compute an adversarial loss which is backpropagated to update parameters of the shared generator and the private generator. The processing unit then concatenates the generated task-invariant and task-specific latent vectors, whereby the concatenated latent vectors are provided to an augmented training dataset.

[0008] In embodiments of this aspect, before the instructions to direct the processing unit to configure the GAN-COS module to balance the distribution of datapoints across defect classes within the task, the instruction comprises further instructions for directing the processing unit to configure the GAN-COS module to sub-divide the task into batches, wherein the steps of balancing the distribution of datapoints, applying the shared generator of the GAN-COS module to the balanced dataset, selecting and applying the private generator of the GAN-COS module to the balanced dataset, performing latent space oversampling, training the discriminator of the GAN-COS module and concatenating the generated task-invariant and task-specific latent vectors, are performed sequentially on each batch of the task.

[0009] In embodiments of this aspect, the computing module further comprises instructions for directing the processing unit to provide a new training task to the GAN-COS module, the new training task being associated with a specific category of defect types and comprises a set of new defect classes. The processing unit then configures the GAN-COS module to balance, using the data balancing module, distribution of datapoints across the new defect classes to

generate a new balanced dataset, and to apply, the shared generator of the GAN-COS module, to the new balanced dataset to generate new task-invariant latent vectors, wherein the generation of the new task-invariant latent vectors is based on parameters of the shared generator that have been updated through training on the tasks in the training dataset. The processing unit then selects and applies, a corresponding private generator of the GAN-COS module, to the new balanced dataset to generate new task-specific latent vectors, wherein the generation of the new task-specific latent vectors is based on parameters of the corresponding private generator that have been updated through training on the tasks in the training dataset before performing, using the oversampling module, latent space oversampling to generate a new set of synthetic latent vectors based on the new task-invariant and new task-specific latent vectors. The processing unit then trains the discriminator of the GAN-COS module using the new set of synthetic latent vectors as the synthetic representations, and the generated new task-invariant and new task-specific latent vectors as the real representations, wherein the discriminator is configured to distinguish between real and synthetic representations. The processing unit then proceeds to concatenate the generated new task-invariant and new task-specific latent vectors, whereby the concatenated new latent vectors are provided to the augmented training dataset.

[0010] In embodiments of this aspect, the computing module further comprises instructions for directing the processing unit to train a classifier module using the augmented training dataset and corresponding defect class labels such that it learns to map latent vectors from the augmented training dataset to their respective defect classes.

[0011] In embodiments of this aspect, an inference system for classifying defects in a first dataset is further disclosed. The disclosed system comprises the GAN-COS module according to embodiments of this aspect, wherein the GAN-COS module is configured to apply the shared generator of the GAN-COS module to the first dataset to generate first task-invariant latent vectors, wherein the shared generator was trained using the training dataset and to select and apply a corresponding private generator of the GAN-COS module to the first dataset to generate first task-specific latent vectors, wherein the corresponding private generator is selected based on a category of defect types associated with the first dataset, wherein the corresponding private generator was trained using the training dataset. The GAN-COS module is then configured to concatenate the generated first task-invariant and first task-specific latent vectors. The disclosed system also includes a trained classifier module configured to classify

defects in the first dataset based on the concatenated first task-invariant and first task-specific latent vectors as generated by the GAN-COS module, wherein the classifier module was trained using the augmented training dataset and corresponding defect class labels such that it learns to map latent vectors from the augmented training dataset to their respective defect classes.

[0012] In another aspect of the disclosure, the present application discloses a method for configuring a generative adversarial network-based continual oversampling (GAN-COS) module to augment a training dataset. The disclosed method comprises the step of providing, using a computing module, the training dataset to the GAN-COS module, the training dataset comprising a plurality of tasks wherein each task is associated with a specific category of defect types and comprises a set of defect classes. For each of the plurality of tasks in the training dataset, the method comprises the steps of balancing, using a data balancing module provided within the GAN-COS module, distribution of datapoints across defect classes to generate a balanced dataset; applying a shared generator of the GAN-COS module, to the balanced dataset to generate task-invariant latent vectors common across multiple tasks; selecting and applying, a corresponding private generator of the GAN-COS module, to the balanced dataset to generate task-specific latent vectors, wherein the corresponding private generator is selected based on a category of defect types associated with the balanced dataset; performing, using an oversampling module provided within the GAN-COS module, latent space oversampling to generate a set of synthetic latent vectors based on the task-invariant and task-specific latent vectors; training a discriminator of the GAN-COS module using the set of synthetic latent vectors, the generated task-invariant and task-specific latent vectors, wherein the discriminator is configured to distinguish between real and synthetic representations, wherein the discriminator's output is used to compute an adversarial loss which is backpropagated to update parameters of the shared generator and the private generator; and concatenating the generated task-invariant and task-specific latent vectors, whereby the concatenated latent vectors are provided to an augmented training dataset.

[0013] In embodiments of this another aspect, a method for classifying defects in a first dataset using an inference system including the GAN-COS module according to embodiments of this disclosure is disclosed. The disclosed method comprises the steps of applying the shared generator of the GAN-COS module to the first dataset to generate first task-invariant latent vectors, wherein the shared generator was trained using the training dataset; selecting and

applying a corresponding private generator of the GAN0COS module to the first dataset to generate first task-specific latent vectors, wherein the corresponding private generator is selected based on a category of defect types associated with the first dataset, wherein the corresponding private generator was trained using the training dataset; concatenating the generated first task-invariant and first task-specific latent vectors, and classifying defects, using a trained classifier module, in the first dataset based on the concatenated first task-invariant and first task-specific latent vectors as generated by the GAN-COS module, wherein the classifier module was trained using the augmented training dataset and corresponding defect class labels such that it learns to map latent vectors from the augmented training dataset to their respective defect classes.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] Various embodiments of the present disclosure are described below with reference to the following drawings:

Figure 1 illustrates a block diagram of components or modules that are provided within a computing module for configuring a generative adversarial network-based continual oversampling (GAN-COS) module to augment a training dataset in accordance with embodiments of the present disclosure;

Figure 2 illustrates an example of a multi-class tabular dataset in accordance with embodiments of the present disclosure;

Figure 3 illustrates a block diagram of a processing system for performing embodiments of the present disclosure;

Figure 4 illustrates a flow chart showing the process for configuring a GAN-COS module to augment a training dataset in accordance with embodiments of the present disclosure;

Figure 5 illustrates a block diagram of components or modules that are provided within a computing module for detecting defects in a dataset using a trained GAN-COS module and a trained classifier; and

Figure 6 illustrates a flow chart showing the process for detecting defects in a dataset using a trained GAN-COS module and a trained classifier.

DETAILED DESCRIPTION

[0015] The following detailed description is made with reference to the accompanying drawings, showing details and embodiments of the present disclosure for the purposes of illustration. Features that are described in the context of an embodiment may correspondingly

be applicable to the same or similar features in the other embodiments, even if not explicitly described in these other embodiments. Additions and/or combinations and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0016] In the context of various embodiments, the articles "a", "an" and "the" as used with regard to a feature or element include a reference to one or more of the features or elements.

[0017] In the context of various embodiments, the term "about" or "approximately" as applied to a numeric value encompasses the exact value and a reasonable variance as generally understood in the relevant technical field, e.g., within 10% of the specified value.

[0018] As used herein, the term "and/or" includes any and all combinations of one or more of the associated listed items.

[0019] As used herein, "comprising" means including, but not limited to, whatever follows the word "comprising". Thus, use of the term "comprising" indicates that the listed elements are required or mandatory, but that other elements are optional and may or may not be present.

[0020] As used herein, "consisting of" means including, and limited to, whatever follows the phrase "consisting of". Thus, use of the phrase "consisting of" indicates that the listed elements are required or mandatory, and that no other elements may be present.

[0021] One skilled in the art will recognize that certain functional units in this description have been labelled as modules throughout the specification. The person skilled in the art will also recognize that a module may be implemented as circuits, logic chips or any sort of discrete component. Still further, one skilled in the art will also recognize that a module may be implemented in software which may then be executed by a variety of processor architectures. In embodiments of the disclosure, a module may also comprise computer instructions or executable code that may instruct a computer processor to carry out a sequence of events based on instructions received. The choice of the implementation of the modules is left as a design choice for a person skilled in the art and does not limit the scope of the claimed subject matter in any way.

[0022] The disclosed generative adversarial network-based continual oversampling (GAN-COS) module integrates adversarial oversampling, continual learning, and imbalanced learning to improve classification accuracy in highly imbalanced training datasets. The GAN-COS module is configured to handle high-dimensional structured and unstructured data and is capable of operating in a non-iterative mini-batch mode to process defect samples in the training dataset efficiently.

[0023] The GAN-COS module further incorporates latent space oversampling, which facilitates the generation of synthetic defect samples in the feature space, thereby mitigating class imbalance at a deeper representation level. Additionally, the GAN-COS module supports multi-label defect class balancing. This ensures that samples or datapoints belonging to multiple defect classes are properly weighted thereby preventing minority-class defects from being overshadowed by dominant defect types.

[0024] As an example, semiconductor related defects may be categorized into classes such as, but are not limited to, "center", "donut", and "edge" types of defects. However, as defect detection systems improve, more complex mixed defects emerge, such as "center+donut", i.e., two-mixed defects and "center+donut+edge", i.e., three-mixed defects. This introduces additional challenges, requiring an approach that not only balances imbalanced defect classes but also continually learns and integrates new defect types without extensive retraining.

[0025] One skilled in the art will recognize that the disclosed GAN-COS module is not limited to use with only semiconductor defect training datasets and may also be employed to augment other types of datasets, including but not limited to image classification datasets (e.g., CIFAR-100, Mini-ImageNet), computer vision-based defect detection datasets (where new defect classes are continually introduced), and industrial defect detection datasets. Potential applications of trained defect classifiers include solar panel defect detection (using high-resolution electroluminescence images), vehicle defect and damage detection (for identifying manufacturing defects and accident-related damages), and fabric defect detection (to automate quality control and reduce human error).

[0026] A block diagram of components or modules that are provided within computing module 100 for configuring generative adversarial network-based continual oversampling (GAN-COS) module 101 to augment training dataset 112 in accordance with embodiments of the present disclosure is illustrated in Figure 1.

[0027] As can be seen from Figure 1, GAN-COS module 101 comprises of data balancing module 104 whose output is provided to two key generators: a shared generator 106 which is configured to generate task-invariant latent vectors, and private generators 107 which are each configured to generate task-specific latent vectors. The latent vectors from shared and private generators 106 and 107 respectively, are then provided to oversampling module 108. Combined latent vectors 109 that are generated by oversampling module 108 are then provided to discriminator 110 of GAN-COS module 101. Combined latent vectors 109 are also concurrently provided to augmented training dataset 112 and this dataset may be used to train classifier 114.

[0028] In operation, training dataset 102, which comprises an imbalanced dataset, is first provided to GAN-COS module 101. In embodiments of the disclosure, computing module 100 may initiate the processing of training dataset 102 by extracting and providing each of the tasks contained within training dataset 102 to data balancing module 104 in a sequential manner. Data balancing module 104 is configured to balance the distribution of datapoints across various defect classes of a received task to generate a balanced task to prevent bias and to ensure a constant batch size.

[0029] In embodiments of the disclosure, training dataset 102 may comprise a plurality of tasks where each task may be associated with a specific category of defect types and may include a set of defect classes. An exemplary multi-class tabular training dataset in accordance with embodiments of the present disclosure is illustrated in Figure 2.

[0030] As illustrated, it can be seen that the training dataset in Figure 2 comprises a plurality of tasks where each task comprises multiple defect classes, and each task is categorized based on the type of defects it represents. Additionally, each defect class of each task is associated with a corresponding number of datapoints. As can be seen, the training dataset is organized into three primary columns: task 202, classes per task 204, and category 206. The task column

202 assigns a numerical identifier to each task, implying that the tasks may be used in a sequential training manner. The "classes per task" column 204 lists the defect classes or labels included in each task along with their corresponding number of datapoints, and this shows the distribution of the defect samples for each task. The category column 206 indicates the category of defect that is associated with the task, e.g., single defects, two-mixed defects, or more complex multi-defect classifications.

[0031] For example, it can be seen that Task 1 comprises of single defect types, meaning that each defect sample is classified under a single defect category. In this example, Task 1 comprises of four different classes of defects: Class 1 defect with 400 datapoints, Class 2 defect with 200 datapoints, Class 3 defect with 10 datapoints and Class 4 defect with 300 datapoints. This distribution highlights an imbalance in defect sample availability in Task 1, where certain defect classes, e.g., Class 3, are significantly underrepresented compared to the other classes in this task.

[0032] One skilled in the art will recognize that the training dataset is not limited to a specific number of tasks or defect classes. Each task may include any number of defect classes, and the tasks may be associated with various categories of defects without departing from the scope of this disclosure. The training dataset illustrated in Figure 2 is meant to be a non-limiting example as to how the defect data and labels may be presented. Other methods of arranging, organizing and/or presenting the dataset may be employed based on specific application requirements, dataset characteristics, or model training objectives.

[0033] Returning to Figure 1, when a task from training dataset 102 is provided to data balancing module 104, this module then proceeds to balance the distribution of datapoints across various defect classes of the task. In embodiments of the disclosure, data balancing module 104 performs the balancing of the datapoints of each of the classes for the task by first identifying class imbalances within the defect classes of the task to identify overrepresented defect classes, and underrepresented defect classes. Data balancing module 104 then proceeds to resample datapoints of the overrepresented defect classes to reduce a number of datapoints associated with the overrepresented effect classes before it proceeds to generate synthetic datapoints for the underrepresented defect classes based on stored statistical distributions. The

combined resampled datapoints, the synthetic datapoints, and unmodified datapoints are then provided to shared generator 106 and corresponding private generator 107.

[0034] In embodiments of the disclosure, the stored statistical distributions may comprise, but are not limited to, the average and standard deviations derived from real underrepresented defect samples, where the synthetic datapoints are generated based on these statistical parameters. Additionally, the stored statistical distributions may also comprise Gaussian distributions, Poisson distributions, or empirical probability distributions constructed from real underrepresented defect samples. In such cases, the synthetic datapoints are designed to closely follow the statistical properties of the respective distributions. The resampled datapoints associated with the overrepresented defect classes, the synthetic datapoints associated with the underrepresented defect classes, and unmodified datapoints are then collected and combined by data balancing module 104 such that the total number of datapoints across the defect classes for the task remains unchanged.

[0035] For example, as illustrated in Figure 2, it can be seen that Task 2 contains a total of 2,400 datapoints, with Class 14 defects being underrepresented in comparison to other defect classes within the task, e.g., Class 56, 22, 35. After Task 2 has been processed by data balancing module 104, the distribution across all defect classes in Task 2 should be approximately equal, with each class containing around 600 datapoints. Additionally, it should be noted that the total number of datapoints in Task 2 remains unchanged at 2,400 ensuring that the overall size of the task is maintained.

[0036] In another embodiment of the disclosure, instead of generating synthetic datapoints for the underrepresented defect classes based on stored statistical distributions, data balancing module 104 may instead adopt a reservoir resampling approach where additional datapoints of the underrepresented defect classes are resampled from other batches of the same task to increase a number of datapoints associated with the underrepresented effect classes of the batch. This approach may be employed when each of the tasks are sub-divided into batches with each batch containing a smaller subset of multi-label defect classes. The embodiment whereby each of the tasks are sub-divided into smaller sized batches will be described in detail in the later sections below.

[0037] The balanced task comprising the combined resampled datapoints, the synthetic datapoints, and unmodified datapoints, is then provided to both shared generator 106 and corresponding private generator 107.

[0038] At this stage, GAN-COS module 101 may compute the adversarial continual learning loss, L_{ACL}, composed of the adversarial loss, L_{adv}, the task-specific loss, L_{task} and the difference loss, L_{diff} based on the balanced task and information stored in a memory of GAN-COS module 101 where the memory is used to retain information and data of tasks that were previously processed by GAN-COS module 101. The adversarial continual learning loss, L_{ACL} is then used to update the parameters of shared generator 106 and corresponding private generator 107.

[0039] In embodiments of the disclosure, the adversarial continual learning loss, L_{ACL}, the adversarial loss, L_{adv}, the task-specific loss, L_{task} and the difference loss, L_{diff}, parameters may all be computed based on the Adversarial Continual Learning (ACL) framework which is described in "*Adversarial Continual Learning*" by the authors Sayna Ebrahimi; Franziska Meier; Roberto Calandra; Trevor Darrell; and Marcus Rohrbach. The following is a brief summary of the workings of the ACL framework and its integration with parts of the GAN-COS module 101.

[0040] The adversarial loss L_{adv} is designed to enforce task invariance in the shared generator by training it to produce features that confuse the discriminator. The discriminator in turn attempts to correctly classify the task labels of the input samples. Mathematically, the adversarial loss L_{adv} follows a minimax optimization approach where the discriminator maximizes the probability of correctly predicting the task labels while the shared generator minimizes this probability. The task-specific loss L_{task}, on the other hand, ensures that the private generators learn task-specific representations that map input samples to task specific outputs. This loss is typically computed using a cross-entropy function over the classifier's predictions. Additionally, the difference loss L_{diff} enforces an orthogonality constraint between the task-invariant features produced by the shared generator and task-specific latent features learned by the private generators, ensuring that task-specific latent features remain distinct from the shared representations. The adversarial continual learning loss, L_{ACL} may then be defined as a weighted sum of these three loss components:

$$L_{ACL} = \lambda_1 L_{adv} + \lambda_2 L_{task} + \lambda_3 L_{diff} \quad \text{...equation (1)}$$

where λ_1, λ_2 and λ_3 are defined as regularization parameters that control the effect of each loss component.

[0041] The shared generator is responsible for extracting task-invariant features that generalize across multiple tasks. To achieve this, it is trained to minimize the adversarial loss L_{adv} making it difficult for the discriminator to distinguish between task labels. This process ensures that the shared features capture generalizable representations that support learning across multiple tasks. Conversely, each private generator is optimized to learn task-specific representations that complement the shared generator's features. The private generators are updated by minimizing the task-specific loss L_{task} to ensure accurate task predictions, while also minimizing the difference loss L_{diff} to maintain a clear separation between shared and task-specific features.

[0042] The discriminator's role in the adversarial training process is to classify task labels based on the representations extracted by the shared generator. During training, the discriminator receives task-invariant features extracted by the shared generator and is trained to maximize the probability of correctly predicting the task labels. Simultaneously, the shared generator is trained via adversarial learning to generate task-invariant features. This is achieved through a gradient reversal layer that ensures the shared generator maximizes the discriminator's classification loss, making task classification more difficult and enforcing invariance. This adversarial training process is implemented using a gradient reversal layer, which negates gradients during backpropagation.

[0043] Returning to Figure 1, upon receiving the balanced task, shared generator 106 processes the task to generate task-invariant latent vectors that capture features common across multiple tasks.

[0044] When the balanced task is provided to private generator 107, GAN-COS module 101 proceeds to select an appropriate private generator to process the task based on the category of the task. In particular, private generator 107 processes the balanced task concurrently with

shared generator 106, ensuring that both task-invariant and task-specific latent vectors for the task are produced in parallel.

[0045] To recap, each task is associated with a specific defect category, such as single defects, two-mixed defects, or four-mixed defects, and GAN-COS module 101 may maintain an indexed set of private generators where each private generator has been trained on a particular type of defect category. GAN-COS module 101 then dynamically determines whether an existing private generator can be used or if a new private generator needs to be initialized. If the new task introduces a defect category that significantly differs from existing categories, GAN-COS module 101 may create a new private generator for handling the new task. Otherwise, the private generator which matches the task label assigned to performs incremental training to retain knowledge of past tasks while processing the incoming data. The selected private generator then proceeds to generate task-specific latent vectors based on the balanced task.

[0046] As the task-invariant and task-specific latent vectors are produced for the balanced task by shared generator 106 and selected private generator 107 respectively, these latent vectors are provided to oversampling module 108.

[0047] Oversampling module 108 then proceeds to generate synthetic latent vectors based on these received latent vectors where the number of synthetic latent vectors that are generated are similar to that received, i.e., based on the distribution of the generated task-invariant and task-specific latent vectors. In embodiments of the disclosure, oversampling module 108 may generate the synthetic latent vectors using a random sampling technique or based on stored statistical distributions. The stored statistical distributions may comprise, but are not limited to, the average and standard deviations derived from existing latent vectors, where the synthetic latent vectors are generated based on these statistical parameters.

[0048] The task-invariant and task-specific latent vectors together with the synthetic latent vectors are then provided to discriminator 110. GAN-COS module 101 then uses this data to train discriminator 110 to distinguish between real and synthetic representations and to classify task labels, thereby enforcing task-invariance in shared generator 106. Discriminator 110's

output is then used to compute an adversarial loss which is subsequently backpropagated to update parameters of shared generator 106 and private generator 107.

[0049] Simultaneously, the task-invariant and task-specific latent vectors, along with the synthetic latent vectors generated for the balanced task, are provided to augmented training dataset 112. Once this process is completed, GAN-COS module 101 sequentially retrieves the next task from training dataset 102 and processes it using the same steps outlined above. This iterative procedure continues until all tasks in training dataset 102 have been processed sequentially.

[0050] In an embodiment, after all the tasks in training dataset 102 have been processed, computing module 100 may utilize augmented training dataset 112, along with the corresponding class labels, to train classifier 114. In other embodiments, computing module 100 may adopt an incremental training approach where it continuously trains classifier 114 as each new set of task-invariant and task-specific latent vectors, along with their corresponding synthetic latent vectors are added to augmented training dataset 112. In this approach, computing module 100 does not need to wait for all the tasks in training dataset 102 to be fully processed before it initiates the training of classifier 114.

[0051] In some embodiments, classifier 114 may be implemented as a task-specific multi-layer perceptron (MLP) head, designed to make final class predictions based on the learned feature representations.

[0052] In embodiments of the disclosure, as each of the tasks contained within training dataset 102 are extracted and provided to data balancing module 104 in a sequential manner by computing module 100, computing module 100 may trigger GAN-COS module 101 to sub-divide the task into multiple batches before each of these batches are sequentially processed as previously described.

[0053] In some embodiments, such a batch-wise processing approach allows GAN-COS module 101 to handle large-scale training datasets more efficiently as this approach reduces computational overhead while maintaining a consistent defect class distribution. Further, this approach ensures that defect class imbalances are addressed at a finer granularity thereby

preventing the data from being overfitted when an entire task is processed at once. Still further, batch-wise task subdivision also facilitates incremental learning, allowing GAN-COS module 101 to dynamically update augmented training dataset 112 as new batches are processed. This in turn allows for classifier 114 to be dynamically trained.

[0054] It should be noted that GAN-COS module 101 is configured to continuously learn and adapt to newly introduced defect types, enabling it to accommodate incremental changes without the need for it to be retrained from scratch. The following embodiment illustrates this advantage.

[0055] In embodiments of the disclosure, a new training task that was not contained within training dataset 102 may be introduced to GAN-COS module 101. The new task may be associated with a specific category of defect types and may include of a set of new defect classes. Upon receiving the new training task, GAN-COS module 101 may balance the distribution of datapoints across the newly introduced defect classes in the new training task using data balancing module 104 as previously discussed.

[0056] Once the new balanced dataset is generated, shared generator 106 may use the new balanced dataset to produce new task-invariant latent vectors that are generated based on parameters of the shared generator that have been updated through training on previous tasks from the original training dataset. Simultaneously, a corresponding private generator may be selected as previously described and applied to the new balanced dataset to generate new task-specific latent vectors based on parameters of the private generator that have been updated through prior training on previous tasks.

[0057] Following this, oversampling module 108 is utilized to perform latent space oversampling, generating a new set of synthetic latent vectors based on the newly created task-invariant and task-specific latent vectors. Discriminator 110 is then trained using these synthetic latent vectors as synthetic representations, while the task-invariant and task-specific latent vectors are used as real representations. During this process, discriminator 110 is configured to distinguish between real and synthetic representations, refining its classification capability through adversarial training. The parameters of the private and shared generators are also indirectly updated accordingly. Finally, the newly generated task-invariant and task-

specific latent vectors are concatenated, and the resulting concatenated latent vectors are provided to augmented training dataset 112. This updated augmented training dataset 112 may then be used to update the training of classifier 114. Through the steps set out above, this shows GAN-COS module 101's ability to continuously learn and adapt to newly introduced defect types.

[0058] An exemplary pseudocode that describes the processes performed by GAN-COS module 101 is set out in Algorithm 1 below. The process begins with data preparation, where the dataset, i.e., the training dataset, is divided into tasks, and the necessary network components are initialized (lines 1-7). This setup ensures that the module is structured for continual learning, allowing it to process multiple tasks sequentially without losing previously acquired knowledge.

Algorithm 1 GAN-COS

1: Load the imbalanced dataset $\mathcal{X}$
2: Perform data preparation according to Section 5 and get the class labels $\mathcal{Y}$
3: Divide tasks $\mathcal{T}_{1...n}$ according for continual learning ▷ For unstructured data, refer Table 1
4: Initialize shared generator $\mathcal{S}$ and $\theta_{\mathcal{S}}$
5: Initialize private generators $\mathcal{P}[1...n]$ and $\theta_{\mathcal{P}}$
6: Initialize classifier $\mathcal{C}$
7: Initialize discriminator $\mathcal{D}$ and $\theta_{\mathcal{D}}$
8: **for** task $t = 1...n$ **do**
9: **for** batches $b = 1...n$ **do**
10: $X_b, Y_b \leftarrow$ Load $\mathcal{T}_t$ for batch size bs
11: $BAL = \texttt{BalanceData}(\mathcal{X}_b, \mathcal{Y}_b)$
12: Compute $L_{ACL}, L_{adv}, L_{task}, L_{diff}$ for $\mathcal{S}$ and $\mathcal{P}$ using $\{BAL, \mathcal{Y}_b, t\}$ from $BAL \cup \mathcal{M}$
13: Update $\theta_{\mathcal{S}}, \theta_{\mathcal{P}}$ with L_{ACL}
14: $\mathbf{z}_s, \mathbf{z}_p \leftarrow \mathcal{S}(BAL, t), \mathcal{P}[t](BAL)$
15: $\mathbf{z}_f \leftarrow \texttt{GenerateFakeLatent}()$
16: Compute L_{adv} for $\mathcal{D}$ using $\mathbf{z}_s, \mathbf{z}_p$ as real data and $\mathbf{z}_f$ as fake data.
17: Update $\theta_{\mathcal{D}}$ with L_{adv}
18: $\mathcal{C} \leftarrow (\theta_{\mathcal{S}} \bigoplus \theta_{\mathcal{P}})$

19: **procedure** BalanceData($\mathrm{X_b}, \mathrm{Y_b}$)
20: balancedDataCount $= |X_b|/|\mathcal{Y}_t|$
21: $E \leftarrow$ Sample labels using balancedDataCount
22: Initialize an empty array BAL
23: Assign B_y with samples from X_b w.r.t Y_b
24: **for** sample i **do**
25: **if** $E[i] \neq \mathcal{Y}_b[i]$ **then**
26: **if** $B_{E[i]}$ is not empty **then**
27: $BAL[i] \leftarrow \texttt{pop}(B_{E[i]})$
28: **else**
29: $BAL[i] \leftarrow \texttt{RandSample}(mean_{E[i]}, std_{E}$
30: **else**
31: $BAL[i] \leftarrow X_b[i]$
32: **return** BAL

33: **procedure** GenerateFakeLatent($\mathrm{X_b}, \mathrm{Y_b}$)
34: balancedDataCount $= |X_b|/|\mathcal{Y}_t|$
35: $E \leftarrow$ Sample the class labels using balancedDataCount
36: Initialize an empty array BAL
37: **for** Sample i **do**
38: $BAL[i] \leftarrow \texttt{RandSample}(mean_{E[i]}, std_{E[i]})$
39: **return** BAL

[0059] Once initialized, the data may be loaded in mini-batch mode (line 10) and balanced using the BalanceData() function. The balancing process first calculates the balanceDataCount and generates the corresponding class labels (L) (lines 20-21). The original data samples are then separated based on their respective class labels and stored in a temporary storage variable

(B_y) (line 23). To ensure that minority class samples are sufficiently represented, the module compares the original class label of each sample (i) with E[i]. If B_y is not empty, the balanced dataset (BAL) is populated using original samples from B_y. If the minority class is still underrepresented, the module generates synthetic samples using a random sampling process based on the pre-stored mean and standard deviation of E[i] (lines 24-32).

[0060] After obtaining the balanced dataset (BAL), the module proceeds to compute multiple loss functions: L_{ACL} (adversarial continual learning loss), L_{adv} (adversarial loss), L_{task} (task-specific loss), and L_{diff} (difference loss). These losses are computed using samples from BAL and memory (M) to ensure that the model retains knowledge from previous tasks (lines 12-13). The BAL dataset is then fed into the shared and private generators, which generate task-invariant and task-specific latent vectors, respectively (line 14). The module further generates fake latent vectors (z_f) using a random sampling process within the GenerateFakeLatent() function (lines 33-39).

[0061] Following this, the adversarial loss (L_{adv}) is computed for the discriminator (D), and the discriminator's parameters are updated accordingly to improve its ability to distinguish between real and synthetic latent vectors (lines 15-17). Finally, the classifier (C) is updated using the concatenated latent vectors from the shared and private generators, ensuring that the classifier learns to correctly predict defect class labels based on a combination of task-invariant and task-specific features (line 18).

[0062] It can be seen that Algorithm 1 describes an embodiment where the tasks are sub-divided into batches. However, one skilled in the art will recognize that the other parts of Algorithm 1 may be easily adapted to accommodate the embodiment where each task is processed as a whole.

[0063] In accordance with embodiments of the present disclosure, a block diagram representative of components of processing system 300 that may be provided within computing module 100, and/or any of the modules shown in Figure 1 to carry out the computing and processing functions in accordance with embodiments of the disclosure is shown in Figure 3. One skilled in the art will recognize that the exact configuration of each processing system provided within these modules may be different and the exact configuration of processing

system 300 may vary and the arrangement illustrated in Figure 3 is provided by way of example only.

[0064] In embodiments of the disclosure, processing system 300 may comprise controller 301 and user interface 302. User interface 302 is arranged to enable manual interactions between a user and the computing module as required and for this purpose includes the input/output components required for the user to enter instructions to provide updates to each of these modules. A person skilled in the art will recognize that components of user interface 302 may vary from embodiment to embodiment but will typically include one or more of display 340, keyboard 335 and optical device 336.

[0065] Controller 301 is in data communication with user interface 302 via bus 315 and includes memory 320, processing unit or processor 305 mounted on a circuit board that processes instructions and data for performing the method of this embodiment, an operating system 306, an input/output (I/O) interface 330 for communicating with user interface 302 and a communications interface, in this embodiment in the form of a network card 350. Network card 350 may, for example, be utilized to send data from these modules via a wired or wireless network to other processing devices or to receive data via the wired or wireless network. Wireless networks that may be utilized by network card 350 include, but are not limited to, Wireless-Fidelity (Wi-Fi), Bluetooth, Near Field Communication (NFC), cellular networks, satellite networks, telecommunication networks, Wide Area Networks (WAN) etc.

[0066] Memory 320 and operating system 306 are in data communication with processor 305 via bus 310. The memory components include both volatile and non-volatile memory and more than one of each type of memory, including Random Access Memory (RAM) 323, Read Only Memory (ROM) 325 and a mass storage device 345, the last comprising one or more solid-state drives (SSDs). One skilled in the art will recognize that the memory components described above comprise non-transitory computer-readable media and shall be taken to comprise all computer-readable media except for a transitory, propagating signal. Typically, the instructions are stored as program code in the memory components but can also be hardwired. Memory 320 may include a kernel and/or programming modules such as a software application that may be stored in either volatile or non-volatile memory.

[0067] Herein the term "processor" or "processing unit" is used to refer generically to any device or component that can process such instructions and may include: a microprocessor, a processing unit, a microcontroller, a programmable logic device or other computational device. That is, processor 305 may be provided by any suitable logic circuitry for receiving inputs, processing them in accordance with instructions stored in memory and generating outputs (for example to the memory components or on display 340). In this embodiment, processor 305 may be a single core or multi-core processor with memory addressable space. In one example, processor 305 may be multi-core, comprising—for example—an 8 core CPU. In another example, it could be a cluster of CPU cores operating in parallel to accelerate computations.

[0068] **Examples of Structured and Unstructured Training Dataset**

[0069] A combination of a SECOM dataset with a Wafer dataset were utilized as the structured datasets in this example. This combination expands the number of classes available for training the GAN-COS module, allowing for a larger set of training tasks and defect classes beyond that of the individual datasets. To ensure that the dimensions of the two datasets are consistent, each Wafer dataset sample, which has a size of 152 x 1, is padded with zeros to match the dimensional format of the SECOM dataset, which has a size of 590 x 1.

[0070] After the dimensions of the two datasets have been matched, the two datasets are then partitioned for training and testing purposes. In this example, 1023 positive samples and 73 negative samples from the SECOM dataset were allocated to Task 1, while 97 positive samples and 903 negative samples from the Wafer dataset were assigned to Task 2 for training purposes. For the test set, 440 positive samples and 31 negative samples were allocated from the SECOM dataset, and 665 positive samples and 5499 negative samples were allocated from the Wafer dataset. It should be noted that the Wafer dataset split is intentionally designed to reflect real-world scenarios, where the number of testing samples typically exceeds the number of training samples.

[0071] The MixedWM38 dataset was used as the unstructured dataset in this example. As an initial step, the dataset's one-hot encoded class labels were relabeled into categorical single-label format. Although the dataset is inherently imbalanced, it is further intentionally modified to create a highly imbalanced version, which is then used for both training and evaluation. To

ensure a fair comparison across benchmark algorithms, a fixed number of data samples per class is maintained instead of randomizing the sample counts when the data is loaded. The test set is then constructed using all the remaining data samples and the fixed data sample counts of the MixedWM38 dataset is set out in Table 1 below.

Task	Classes per Task				Category
Task 1	Class 1	Class 2	Class 3	Class 4	Single Defect
	800	700	600	500	
Task 2	Class 10	Class 11	Class 12	Class 13	Two Mixed
	800	700	600	500	
Task 3	Class 23	Class 24	Class 25	Class 26	Three Mixed
	800	600	400	200	
Task 4	Class 35	Class 36	Class 37	Class 38	Four Mixed
	800	600	400	200	
Task 5	Class 5	Class 6	Class7	Class 8	Single Defect
	800	400	300	200	
Task 6	Class 14	Class 15	Class 16	Class 17	Two Mixed
	500	400	300	200	
Task 7	Class 18	Class 19	Class 20	Class 21	Two Mixed
	100	200	300	800	
Task 8	Class 27	Class 28	Class 29	Class 30	Three Mixed
	700	500	300	100	
Task 9	Class 31	Class 32	Class 33	Class 34	Three Mixed
	800	600	400	200	
Task 10	Class 9	Class 22	Class 1	Class 19	Single Defect and Two Mixed
	100	800	800	200	

TABLE 1

[0072] The structured and unstructured datasets described above were then provided to the GAN-COS module so that the GAN-COS module may generate an augmented training dataset based on these two datasets.

[0073] The performance of the GAN-COS module is then evaluated based on certain key metrics, including Average Class-Specific Accuracy (ACC), Backward Transfer (BWT), and Geometric Mean (GM) Score. The performance of two variants of the GAN-COS module was compared against the performance of the baseline Adversarial Continual Learning (ACL) framework. The first variant comprises a configuration of GAN-COS+S module, where imbalanced learning was only applied in the shared generator, and the second variant comprises a configuration of GAN-COS+SP module, which extends imbalanced learning to both the shared and private generators.

[0074] Unlike the baseline ACL algorithm, which does not explicitly address class imbalance in incoming tasks, the GAN-COS variants introduce a continual data balancing mechanism that adjusts for imbalances as new tasks are learned. This enables more stable learning while ensuring that minority-class information is preserved during training. From the results shown in Table 2 below, it can be seen that the BWT values of GAN-COS models remain comparable to those of the baseline ACL, demonstrating that the ability to retain knowledge from previously learned tasks is not compromised.

[0075] Furthermore, the ACC and GM scores show a clear advantage in favor of the GAN-COS models, indicating that they outperform the baseline ACL in handling imbalanced data classification. These findings highlight the effectiveness of oversampling strategies in both the shared and private generators, reinforcing the ability of the GAN-COS module in handling class imbalanced datasets while maintaining continual learning capabilities.

[0076] A flowchart which sets out the process for configuring a generative adversarial network-based continual oversampling (GAN-COS) module to augment a training dataset in accordance with embodiments of the present disclosure is illustrated in Figure 4. In embodiments of the disclosure, process 400 as illustrated in Figure 4 may be performed by computing module 100 or any combination of modules described in the sections above.

	ACL		
	ACC	BWT	GM
MixedWM38	81.92% ± 5.11	−4.22 ± 2.16	0.58 ± 0.02
SECOM + Wafer	64.48% ± 4.92	−4.08 ± 2.59	0.61 ± 0.01
	GAN-COS+S		
	ACC	BWT	GM
MixedWM38	93.04% ± 2.28	−4.02 ± 2.07	0.81 ± 0.02
SECOM + Wafer	81.96% ± 3.36	−4.89 ± 2.11	0.75 ± 0.01
	GAN-COS+SP		
	ACC	BWT	GM
MixedWM38	94.38% ± 1.73	−3.20 ± 2.62	0.84 ± 0.01
SECOM + Wafer	83.77% ± 2.84	−4.04 ± 2.99	0.77 ± 0.01

TABLE 2

[0077] Process 400 begins at step 402 with process 400 receiving a training dataset, wherein the training dataset comprises a plurality of tasks, each task being associated with a specific category of defect types and comprises a set of defect classes. For each of the plurality of tasks in the training dataset, process 400 then balances, using a data balancing module provided within the GAN-COS module, the distribution of datapoints across the defect classes to generate a balanced dataset. This takes place at step 404. At step 406, process 400 then applies a shared generator of the GAN-COS module to the balanced dataset to generate task-invariant latent vectors common across multiple tasks. Process 400 then selects and applies a corresponding private generator of the GAN-COS module to the balanced dataset to generate task-specific latent vectors. This occurs at step 408. At this step, the corresponding private generator is selected based on a category of defect types associated with the balanced dataset.

[0078] Process 400 then proceeds to perform at step 410, using an oversampling module provided within the GAN-COS module, latent space oversampling on the task-invariant and task-specific latent vectors to generate a set of synthetic latent vectors. The set of synthetic latent vectors are then provided to a discriminator of the GAN-COS module. At step 412, process 400 then trains the discriminator using the set of synthetic latent vectors, the generated task-invariant and task-specific latent vectors. During the training step, the discriminator is configured to distinguish between real and synthetic representations, wherein the discriminator's output is used to compute an adversarial loss which is backpropagated to update parameters of the shared generator and the private generator. At step 414, process 400 then concatenates the generated task-invariant and task-specific latent vectors before they are provided to an augmented training dataset. It should be noted that steps 404 – 414 are executed in a sequential manner for each task in the training dataset, ensuring that each task undergoes the same processing cycle. This approach continues until all tasks in the dataset have been fully processed, allowing process 400 to systematically learn from each task.

[0079] In embodiments of the disclosure, before process 400 balances the distribution of datapoints across defect classes within the task, process 400 further sub-divides the task into batches. The steps of balancing the distribution of datapoints, applying the shared generator of the GAN-COS module to the balanced dataset, selecting and applying the private generator of the GAN-COS module to the balanced dataset, performing latent space oversampling, training the discriminator of the GAN-COS module and concatenating the generated task-invariant and task-specific latent vectors, are then performed sequentially by process 400 on each batch of the task.

[0080] In embodiments of the disclosure, the step of balancing the distribution of datapoints across defect classes further comprises process 400 identifying class imbalances within the defect classes to identify overrepresented defect classes, and underrepresented defect classes, resampling datapoints of the overrepresented defect classes to reduce a number of datapoints associated with the overrepresented defect classes, generating synthetic datapoints for the underrepresented defect classes based on stored statistical distributions, and combining the resampled datapoints, the synthetic datapoints, and unmodified datapoints such that a total number of datapoints across the defect classes remains unchanged.

[0081] In embodiments of the disclosure, the step of balancing the distribution of datapoints across defect classes further comprises process 400 identifying class imbalances within the defect classes of the batch to identify overrepresented defect classes, and underrepresented defect classes, resampling datapoints of the overrepresented defect classes of the batch to reduce a number of datapoints associated with the overrepresented effect classes, resampling additional datapoints of the underrepresented defect classes from other batches of the task to increase a number of datapoints associated with the underrepresented effect classes of the batch, and combining the resampled datapoints of the overrepresented and underrepresented defect classes and unmodified datapoints such that a total number of datapoints across the defect classes remains unchanged.

[0082] In embodiments of the disclosure, the step of performing the latent space oversampling to generate the set of synthetic latent vectors by process 400 further comprises process 400 determining a number of synthetic latent vectors to be generated based on a distribution of the generated task-invariant and task-specific latent vectors and generating the set of synthetic latent vectors based on the determined number of synthetic latent vectors using stored statistical distributions.

[0083] In embodiments of the disclosure, process 400 further comprises the steps of providing a new training task to the GAN-COS module using the computing module, the new training task being associated with a specific category of defect types and comprises a set of new defect classes, balancing, using the data balancing module, distribution of datapoints across the new defect classes to generate a new balanced dataset, applying, the shared generator of the GAN-COS module, to the new balanced dataset to generate new task-invariant latent vectors, wherein the generation of the new task-invariant latent vectors is based on parameters of the shared generator that have been updated through training on the tasks in the training dataset, selecting and applying, a corresponding private generator of the GAN-COS module, to the new balanced dataset to generate new task-specific latent vectors, wherein the generation of the new task-specific latent vectors is based on parameters of the corresponding private generator that have been updated through training on the tasks in the training dataset, performing, using the oversampling module, latent space oversampling to generate a new set of synthetic latent vectors based on the new task-invariant and new task-specific latent vectors, training the discriminator of the GAN-COS module using the new set of synthetic latent vectors

as the synthetic representations, and the generated new task-invariant and new task-specific latent vectors as the real representations, wherein the discriminator is configured to distinguish between real and synthetic representations, and concatenating the generated new task-invariant and new task-specific latent vectors, whereby the concatenated new latent vectors are provided to the augmented training dataset.

[0084] In embodiments of the disclosure, in process 400, the set of synthetic latent vectors is defined as synthetic representations, and the generated task-invariant and task-specific latent vectors are defined as real representations, the synthetic and real representations being used in the training of the discriminator. Process 400 also includes the step of training a classifier module using the augmented training dataset and corresponding defect class labels such that it learns to map latent vectors from the augmented training dataset to their respective defect classes.

[0085] **Embodiment of an Inference System**

[0086] In accordance with further embodiments of the disclosure, a block diagram of components or modules that are provided within a computing module for detecting defects in a dataset using a trained GAN-COS module and a trained classifier is illustrated in Figure 5. As can be seen, computing module 100 comprises trained GAN-COS module 501 that is coupled to trained classifier 514. Trained GAN-COS module 501 in turn comprises trained shared generator 506 and trained private generator 507, both of which have been trained using a training dataset as described above. Trained classifier 514 was trained using an augmented training dataset that was generated based on the steps described in earlier sections.

[0087] The inference process begins when computing module 100 provides dataset 502 to shared generator 506. Shared generator 506 then proceeds to generate task-invariant latent vectors based on the tasks in dataset 502. These task-invariant latent vectors represent generalizable features that have been learned from the training dataset. Concurrently, trained GAN-COS module 501 selects and applies a corresponding trained private generator 507 based on the category of defect types associated with the tasks in dataset 502. The selected trained private generator 507 then extracts task-specific latent vectors that capture defect-related characteristics unique to the classification task. Once the task-invariant and task-specific latent

vectors have been generated, they are then concatenated to form a comprehensive feature representation set 509. Trained classifier 514 then processes comprehensive feature representation set 509 to classify defects in dataset 502. In embodiments of the disclosure, the tasks contained in dataset 502 may be processed as a whole or may be sub-divided into batches whereby the batches are processed sequentially as described above.

[0088] A flowchart which sets out the process for classifying defects in a dataset using an inference system that includes a GAN-COS module that has been trained as described above is illustrated in Figure 6. In embodiments of the disclosure, process 600 as illustrated in Figure 6 may be performed by computing module 100 or any combination of modules described in the sections above.

[0089] Process 600 begins at step 602 with process 600 applying the shared generator of the GAN-COS module to the dataset to generate first task-invariant latent vectors, wherein the shared generator was trained using the training dataset as described in earlier sections. At step 604, process 600 then selects and applies a corresponding private generator of the GAN-COS module to the first dataset to generate first task-specific latent vectors, wherein the corresponding private generator is selected based on a category of defect types associated with the first dataset, wherein the corresponding private generator was trained using the training dataset. At step 606, process 600 then concatenates the generated first task-invariant and first task-specific latent vectors. Process 600 then provides the concatenated latent vectors to a trained classifier of the inference system. At step 608, process 600 then classifies defects in the dataset based on the concatenated first task-invariant and first task-specific latent vectors as generated by the GAN-COS module, wherein the classifier module was trained using the augmented training dataset and corresponding defect class labels such that it learns to map latent vectors from the augmented training dataset to their respective defect classes.

[0090] Numerous other changes, substitutions, variations, and modifications may be ascertained by the skilled in the art and it is intended that the present application encompass all such changes, substitutions, variations, and modifications as falling within the scope of the appended claims.

CLAIMS:

1. A computing module for configuring a generative adversarial network-based continual oversampling (GAN-COS) module to augment a training dataset, the computing module comprising:

a processing unit; and

a non-transitory media readable by the processing unit, the media storing instructions that when executed by the processing unit causes the processing unit to:

provide the training dataset to the GAN-COS module, the training dataset comprising a plurality of tasks wherein each task is associated with a specific category of defect types and comprises a set of defect classes;

configure the GAN-COS module to:

for each of the plurality of tasks in the training dataset,

balance, using a data balancing module, distribution of datapoints across defect classes to generate a balanced dataset;

apply, a shared generator of the GAN-COS module, to the balanced dataset to generate task-invariant latent vectors common across multiple tasks;

select and apply, a corresponding private generator of the GAN-COS module, to the balanced dataset to generate task-specific latent vectors, wherein the corresponding private generator is selected based on a category of defect types associated with the balanced dataset;

perform, using an oversampling module, latent space oversampling to generate a set of synthetic latent vectors based on the task-invariant and task-specific latent vectors;

train a discriminator of the GAN-COS module using the set of synthetic latent vectors, the generated task-invariant and task-specific latent vectors, wherein the discriminator is configured to distinguish between real and synthetic representations, wherein the discriminator's output is used to compute an adversarial loss which is backpropagated to update parameters of the shared generator and the private generator; and

concatenate the generated task-invariant and task-specific latent vectors, whereby the concatenated latent vectors are provided to an augmented training dataset.

2. The computing module according to claim 1, wherein before the instructions to direct the processing unit to configure the GAN-COS module to balance the distribution of datapoints across defect classes within the task, the instruction comprises further instructions for directing the processing unit to configure the GAN-COS module to:

sub-divide the task into batches, wherein the steps of balancing the distribution of datapoints, applying the shared generator of the GAN-COS module to the balanced dataset, selecting and applying the private generator of the GAN-COS module to the balanced dataset, performing latent space oversampling, training the discriminator of the GAN-COS module and concatenating the generated task-invariant and task-specific latent vectors, are performed sequentially on each batch of the task.

3. The computing module according to claims 1 or 2, wherein the instructions to direct the processing unit to configure the GAN-COS module to balance the distribution of datapoints across defect classes comprises further instructions for directing the processing unit to configure the GAN-COS module to:

identify class imbalances within the defect classes to identify overrepresented defect classes, and underrepresented defect classes;

resample datapoints of the overrepresented defect classes to reduce a number of datapoints associated with the overrepresented effect classes;

generate synthetic datapoints for the underrepresented defect classes based on stored statistical distributions; and

combine the resampled datapoints, the synthetic datapoints, and unmodified datapoints such that a total number of datapoints across the defect classes remains unchanged.

4. The computing module according to claim 2, wherein the instructions to direct the processing unit to configure the GAN-COS module to balance the distribution of datapoints across defect classes comprises further instructions for directing the processing unit to configure the GAN-COS module to:

identify class imbalances within the defect classes of the batch to identify overrepresented defect classes, and underrepresented defect classes;

resample datapoints of the overrepresented defect classes of the batch to reduce a number of datapoints associated with the overrepresented effect classes;

resample additional datapoints of the underrepresented defect classes from other batches of the task to increase a number of datapoints associated with the underrepresented effect classes of the batch; and

combine the resampled datapoints of the overrepresented and underrepresented defect classes and unmodified datapoints such that a total number of datapoints across the defect classes remains unchanged.

5. The computing module according to any one of claims 1 to 4, wherein the instructions to direct the processing unit to configure the GAN-COS module to perform the latent space oversampling to generate the set of synthetic latent vectors comprises further instructions for directing the processing unit to configure the GAN-COS module to:

determine a number of synthetic latent vectors to be generated based on a distribution of the generated task-invariant and task-specific latent vectors; and

generate the set of synthetic latent vectors based on the determined number of synthetic latent vectors using stored statistical distributions.

6. The computing module according to any one of claims 1 to 5 further comprising instructions for directing the processing unit to:

provide a new training task to the GAN-COS module, the new training task being associated with a specific category of defect types and comprises a set of new defect classes;

configure the GAN-COS module to:

balance, using the data balancing module, distribution of datapoints across the new defect classes to generate a new balanced dataset;

apply, the shared generator of the GAN-COS module, to the new balanced dataset to generate new task-invariant latent vectors, wherein the generation of the new task-invariant latent vectors is based on parameters of the shared generator that have been updated through training on the tasks in the training dataset;

select and apply, a corresponding private generator of the GAN-COS module, to the new balanced dataset to generate new task-specific latent vectors, wherein the generation of the new task-specific latent vectors is based on parameters of the corresponding private generator that have been updated through training on the tasks in the training dataset;

perform, using the oversampling module, latent space oversampling to generate a new set of synthetic latent vectors based on the new task-invariant and new task-specific latent vectors;

train the discriminator of the GAN-COS module using the new set of synthetic latent vectors as the synthetic representations, and the generated new task-invariant and new task-specific latent vectors as the real representations, wherein the discriminator is configured to distinguish between real and synthetic representations; and

concatenate the generated new task-invariant and new task-specific latent vectors, whereby the concatenated new latent vectors are provided to the augmented training dataset.

7. The computing module according to claim 1 wherein the set of synthetic latent vectors is defined as synthetic representations, and the generated task-invariant and task-specific latent vectors are defined as real representations, the synthetic and real representations being used in the training of the discriminator.

8. The computing module according to claim 1 further comprising instructions for directing the processing unit to:

train a classifier module using the augmented training dataset and corresponding defect class labels such that it learns to map latent vectors from the augmented training dataset to their respective defect classes.

9. The computing module according to claim 3, wherein the statistical distributions comprise a mean or a standard deviation of datapoints for the corresponding defect class.

10. An inference system for classifying defects in a first dataset, the system comprising:

the GAN-COS module according to claim 1, wherein the GAN-COS module is configured to:

apply the shared generator of the GAN-COS module to the first dataset to generate first task-invariant latent vectors, wherein the shared generator was trained using the training dataset;

select and apply a corresponding private generator of the GAN-COS module to the first dataset to generate first task-specific latent vectors, wherein the corresponding private generator is selected based on a category of defect types associated with the first

dataset, wherein the corresponding private generator was trained using the training dataset;

concatenate the generated first task-invariant and first task-specific latent vectors, and

a trained classifier module configured to classify defects in the first dataset based on the concatenated first task-invariant and first task-specific latent vectors as generated by the GAN-COS module, wherein the classifier module was trained using the augmented training dataset and corresponding defect class labels such that it learns to map latent vectors from the augmented training dataset to their respective defect classes.

11. A method for configuring a generative adversarial network-based continual oversampling (GAN-COS) module to augment a training dataset, the method comprising:

providing, using a computing module, the training dataset to the GAN-COS module, the training dataset comprising a plurality of tasks wherein each task is associated with a specific category of defect types and comprises a set of defect classes;

for each of the plurality of tasks in the training dataset,

balancing, using a data balancing module provided within the GAN-COS module, distribution of datapoints across defect classes to generate a balanced dataset;

applying a shared generator of the GAN-COS module to the balanced dataset to generate task-invariant latent vectors common across multiple tasks;

selecting and applying, a corresponding private generator of the GAN-COS module, to the balanced dataset to generate task-specific latent vectors, wherein the corresponding private generator is selected based on a category of defect types associated with the balanced dataset;

performing, using an oversampling module provided within the GAN-COS module, latent space oversampling to generate a set of synthetic latent vectors based on the task-invariant and task-specific latent vectors;

training a discriminator of the GAN-COS module using the set of synthetic latent vectors, the generated task-invariant and task-specific latent vectors, wherein the discriminator is configured to distinguish between real and synthetic representations, wherein the discriminator's output is used to compute an adversarial loss which is backpropagated to update parameters of the shared generator and the private generator; and

concatenating the generated task-invariant and task-specific latent vectors, whereby the concatenated latent vectors are provided to an augmented training dataset.

12. The method according to claim 11, wherein before the step of balancing the distribution of datapoints across defect classes within the task, the method further comprises the steps of:

sub-dividing the task into batches using the computing module, wherein the steps of balancing the distribution of datapoints, applying the shared generator of the GAN-COS module to the balanced dataset, selecting and applying the private generator of the GAN-COS module to the balanced dataset, performing latent space oversampling, training the discriminator of the GAN-COS module and concatenating the generated task-invariant and task-specific latent vectors, are performed sequentially on each batch of the task.

13. The method according to claims 11 or 12, wherein the step of balancing the distribution of datapoints across defect classes comprises further the steps of:

identifying class imbalances within the defect classes to identify overrepresented defect classes, and underrepresented defect classes;

resampling datapoints of the overrepresented defect classes to reduce a number of datapoints associated with the overrepresented effect classes;

generating synthetic datapoints for the underrepresented defect classes based on stored statistical distributions; and

combining the resampled datapoints, the synthetic datapoints, and unmodified datapoints such that a total number of datapoints across the defect classes remains unchanged.

14. The method according to claim 12, wherein the step of balancing the distribution of datapoints across defect classes further comprises the steps of:

identifying class imbalances within the defect classes of the batch to identify overrepresented defect classes, and underrepresented defect classes;

resampling datapoints of the overrepresented defect classes of the batch to reduce a number of datapoints associated with the overrepresented effect classes;

resampling additional datapoints of the underrepresented defect classes from other batches of the task to increase a number of datapoints associated with the underrepresented effect classes of the batch; and

combining the resampled datapoints of the overrepresented and underrepresented defect classes and unmodified datapoints such that a total number of datapoints across the defect classes remains unchanged.

15. The method according to any one of claims 11 to 14, wherein the step of performing the latent space oversampling to generate the set of synthetic latent vectors further comprises the steps of:

determining a number of synthetic latent vectors to be generated based on a distribution of the generated task-invariant and task-specific latent vectors; and

generating the set of synthetic latent vectors based on the determined number of synthetic latent vectors using stored statistical distributions.

16. The method according to any one of claims 11 to 15 further comprises the steps of:

providing a new training task to the GAN-COS module using the computing module, the new training task being associated with a specific category of defect types and comprises a set of new defect classes;

balancing, using the data balancing module, distribution of datapoints across the new defect classes to generate a new balanced dataset;

applying, the shared generator of the GAN-COS module, to the new balanced dataset to generate new task-invariant latent vectors, wherein the generation of the new task-invariant latent vectors is based on parameters of the shared generator that have been updated through training on the tasks in the training dataset;

selecting and applying, a corresponding private generator of the GAN-COS module, to the new balanced dataset to generate new task-specific latent vectors, wherein the generation of the new task-specific latent vectors is based on parameters of the corresponding private generator that have been updated through training on the tasks in the training dataset;

performing, using the oversampling module, latent space oversampling to generate a new set of synthetic latent vectors based on the new task-invariant and new task-specific latent vectors;

training the discriminator of the GAN-COS module using the new set of synthetic latent vectors as the synthetic representations, and the generated new task-invariant and new task-specific latent vectors as the real representations, wherein the discriminator is configured to distinguish between real and synthetic representations; and

concatenating the generated new task-invariant and new task-specific latent vectors, whereby the concatenated new latent vectors are provided to the augmented training dataset.

17. The method according to claim 11 wherein the set of synthetic latent vectors is defined as synthetic representations, and the generated task-invariant and task-specific latent vectors are defined as real representations, the synthetic and real representations being used in the training of the discriminator.

18. The method according to claim 11 further comprising the steps of:

training a classifier module using the augmented training dataset and corresponding defect class labels such that it learns to map latent vectors from the augmented training dataset to their respective defect classes.

19. The method according to claim 13, wherein the statistical distributions comprise a mean or a standard deviation of datapoints for the corresponding defect class.

20. A method for classifying defects in a first dataset using an inference system including the GAN-COS module according to claim 1, the method comprising the steps of:

applying the shared generator of the GAN-COS module to the first dataset to generate first task-invariant latent vectors, wherein the shared generator was trained using the training dataset;

selecting and applying a corresponding private generator of the GAN-COS module to the first dataset to generate first task-specific latent vectors, wherein the corresponding private generator is selected based on a category of defect types associated with the first dataset, wherein the corresponding private generator was trained using the training dataset;

concatenating the generated first task-invariant and first task-specific latent vectors, and

classifying defects, using a trained classifier module of the inference system, in the first dataset based on the concatenated first task-invariant and first task-specific latent vectors as generated by the GAN-COS module, wherein the classifier module was trained using the augmented training dataset and corresponding defect class labels such that it learns to map latent vectors from the augmented training dataset to their respective defect classes.