Event classification on subsea pipeline inspection data using an ensemble of deep learning classifiers.

DANG, T., NGUYEN, T.T., LIEW, A.W.-C. and ELYAN, E.

2025

Cognitive Computation (2025) 17:10
https://doi.org/10.1007/s12559-024-10377-y

RESEARCH

Event Classification on Subsea Pipeline Inspection Data Using an Ensemble of Deep Learning Classifiers

Truong Dang[1] · Tien Thanh Nguyen[1] · Alan Wee-Chung Liew[2] · Eyad Elyan[1]

Received: 8 May 2024 / Accepted: 10 September 2024

Abstract
Subsea pipelines are the backbone of the modern oil and gas industry, transporting a total of 28% of global oil production. Due to several factors, such as corrosion or deformations, the pipelines might degrade over time, which might lead to serious economic and environmental damages if not addressed promptly. Therefore, it is crucial to detect any serious damage to subsea pipelines before they cause dangerous catastrophes. Inspections of subsea pipelines are usually made using a Remote Operating Vehicle and the inspection data is usually processed manually, which is subject to human errors, and requires experienced Remote Operating Vehicle operators. It is thus necessary to automate the inspection process to enable more efficiency as well as reduce costs. Besides, it is recognised that specific challenges of noisy and low-quality inspection data arising from the underwater environment prevent the industry from taking full advantage of the recent development in the Artificial Intelligence field to the problem of subsea pipeline inspection. In this paper, we developed an ensemble of deep learning classifiers to further improve the performance of single deep learning models in classifying anomalous events on the subsea pipeline inspection data. The output of the proposed ensemble was combined based on a weighted combining method. The weights of base classifiers were found by minimising the difference between the weighted combining result and the given associated ground truth annotation information. Three inspection datasets, gathered from different oil and gas companies in the United Kingdom, were analysed. These datasets were recorded under varying conditions and include a range of anomalies. The results showed that the proposed ensemble achieves around 78% accuracy on two datasets and more than 99% accuracy on one dataset, which is better compared to base classifiers and two popular ensembles.

Keywords Deep learning · Inspection · Ensemble of classifiers · Ensemble learning · Ensemble of deep learning · Subsea pipeline

Introduction

Subsea pipeline plays an important role in modern oil and gas infrastructure systems by transporting oil and gas extracted from oceans to the mainland for consumption. In 2018, offshore production accounted for 28% of total global oil production [1]. With near-shore reservoirs becoming depleted, it is expected that the oil and gas industry will increasingly venture into deeper waters and harsher environments [2]. However, it is recognised that harsh environmental conditions such as corrosion, deformation, fracture, or typhoons mean that the subsea pipelines are always in a state of continuous degradation [3]. Over time, this can lead to damage in subsea pipelines, resulting in serious economic and environmental consequences. Although the average failure rates for subsea pipelines are relatively low, the potential consequences of failure can be extremely severe

Highlights
- We propose an ensemble of deep learning-based classifiers to classify extracted images from inspection videos into several classes.
- We propose to combine the deep learning-based classifiers by using a weighted combining method.
- Experimental results showed that our proposed ensemble performs better than benchmark algorithms on the experimental datasets.

✉ Tien Thanh Nguyen
t.nguyen11@rgu.ac.uk

1 School of Computing, Engineering and Technology, Robert Gordon University, Aberdeen, UK

2 School of Information and Communication Technology, Griffith University, Gold Coast, Australia

[4]. Therefore, it is important to promptly detect any pipeline damage to prevent catastrophic consequences.

Regular visual inspections of subsea pipelines and platforms are typically conducted to assess the conditions of these assets. During such inspections, a Remote Operating Vehicle (ROV) is deployed to collect survey data from various sensors and instruments, which usually consist of 1) video footage captured from three camera angles (left, center, and right) 2) An Inertial Measurement Unit (IMU) to record the orientation of the ROV, 3) multi-beam echo data for mapping the seabed surface, and 4) a magnetic pipe-tracker to track the pipe location. During the inspection, a data coordinator onboard the surface vessels will provide commentary on the survey data and produce initial annotations, subjecting to Quality Control (QC) [5]. However, during manual inspection routines, the data annotation and QC processes are subjected to human error. Additionally, the speed of the ROV is limited by the rate at which human operators can process the data. Moreover, manual inspection requires ROV operators to have extensive experience and years of practice. Given the expansion of the oil and gas industry, a shortage of qualified ROV operators is anticipated in the future [6]. Automating the survey process would enable more efficient inspections, reduce costs, and mitigate the risks associated with having human operators offshore.

Machine learning (ML) is an active area of research in designing algorithms which can perform various tasks with human-level accuracy. In recent years, ML has been extensively applied to automating subsea pipeline inspection [7]. There are three stages in applying ML to automate pipeline inspection: data collection, data transfer, and condition assessment. During the first stage, pipeline data which consists of routine operation data (e.g., the flow rate and pressure), NDT signals (ultrasonic data), and computer vision data (e.g., images and videos) are collected. In the second stage, the collected data is transferred from the inspection site to a data analysis center. Finally, in the third stage, several preprocessing steps are made before a ML model is applied to yield the final prediction. In [8], the authors used Artificial Neural Networks (ANNs) to predict pipeline conditions based on the physical, external, and operational data collected from existing offshore oil and gas pipelines in Qatar. In [8], Rashid et al. designed a smart wireless sensor network (WSN) for leak detection and size estimation in long-range pipelines using Support Vector Machine (SVM), k-Nearest Neighbours (kNN), and Gaussian Mixture Model (GMM). In [9], the authors proposed a pipeline leak detection and localisation model by training SVM to detect Negative Pressure Wave (NPW) in a pressure curve.

Recently, deep learning approaches have made important strides in subsea pipeline inspection [10]. Li et al. [11] used a Long Short-Term Memory (LSTM) model to learn the relationship between pipeline corrosion depth and its influencing factors. In [12], the authors proposed a novel visual deep transfer learning method which not only predicts the defect size but also estimates the defect cross-sectional profile of oil and gas pipelines as well. Experiments on laboratory data demonstrate the effectiveness of this method. In [13], the authors combined percussion, variational mode decomposition (VMD), and capsule neural network to classify six levels of pipeline elbow erosions. Zhang et al. [14] combined the Mel-frequency cepstral coefficient and LSTM using 1152 operating conditions, including flow pattern, leak size, direction, and location for leak detection and leak size identification in gas–liquid two-phase flow pipelines. In [15], the authors proposed an injurious (crack, metal losses, etc.) or non-injurious (noise events, manufacturing irregularities, etc.) defect identification method from magnetic flux leakage (MFL) images based on CNN. In [16], the authors proposed a novel deep offline-to-online deep learning framework for pipeline leakage detection. In the offline stage, a deep learning model is trained using additional regularisation terms in order to extract domain-invariant features and early fault features from pipeline samples under different scenarios. During the online stage, the trained model is employed to monitor the operating condition of the pipeline in real-time.

However, it should be noted that the deep learning model generally requires more data and takes more time to train compared to traditional models. Even if data is available, the performance of the deep learning model might not be satisfactory. A potential solution to this problem is to combine the predictions of different deep learning models to improve the final predictions, which is known as ensemble learning [17]. In recent years, there have also been many works on designing ensembles of deep learning models [18]. In [19], the authors proposed ResNet, a deep residual learning-based neural network which can extend deeply into hundreds of layers. An ensemble of these networks won the ILSVRC 2015 competition. Calisto et al. proposed AdaEn-Net [18], which is an ensemble of deep 2D-3D fully convolutional networks for medical image segmentation. In [20], the authors proposed a bagging ensemble of stacked denoising autoencoders for classification with binarisation of multi-class problems, achieving good results on the MNIST handwritten digit recognition dataset. In [21], the authors investigated the use of an ensemble of deep learning models for uncertainty estimation and demonstrated the effectiveness of their method on a large image classification dataset. In [22], the authors used a CNN-based model for feature extraction, which is then learned by three deep learning models. Majority voting is used to combine the results of these models and output the final predictions. In [23], the authors introduced a deep learning-based ensemble for accurate fog-related low-visibility events forecasting. Seven deep learning models are trained with different subsets of the data using bagging to enhance the diversity of the models, and three different weight-based ensemble methods are investigated in the paper.

In this paper, we designed an ensemble of deep learning-based classifiers to classify images of subsea pipelines into several classes (exposure, free span, field joint, anode, and debris). The ensemble includes a number of deep learning-based classifiers, trained on datasets obtained from survey videos of three different companies. The images extracted from the videos were associated with the provided annotation information to create the experimental datasets. The deep learning classifiers were trained on the training dataset using a T-fold cross-validation procedure to create metadata which is the predictions of the classifiers for images in the training datasets. Based on the recognition that a classifier may perform differently on different datasets, we propose to weigh the classifiers and use the weights for ensemble aggregation. If a classifier performs well on the training dataset, it may get a higher weight than a poorer peer. In this study, the weights were found by minimising the difference between the weighted combining results and the associated ground truth annotation information. The performance of the proposed ensemble was compared to those of base deep learning classifiers and two ensemble methods namely Sum Rule and Snapshot Ensemble [24]. The results indicate that our proposed ensemble achieved better results compared to those benchmark algorithms.

The contributions of our paper are as follows:

- We create three datasets concerning subsea pipeline inspection activities. These datasets will be published for non-commercial use.
- We propose an ensemble of deep learning-based classifiers to classify extracted images from inspection videos into several classes. We trained 9 deep learning methods on each dataset and combined their outputs to obtain the final prediction.
- We propose to combine deep learning-based classifiers by using a weighted combining method. We minimise the distance between the combining results and the associated ground truths to obtain the combining weights.
- Experimental results showed that our proposed ensemble obtained between 1 and 2% better accuracy compared to the deep learning-based classifiers and benchmark algorithms except for Dataset C.

In Sect. 2, we briefly introduce the datasets and nine deep ensemble methods that are used to generate the ensemble. In Sect. 3, we give a detailed description of the proposed method including how to train the base classifiers and the proposed ensemble. Experimental studies and comparisons are provided in Sect. 4, followed by conclusions in Sect. 5.

Materials and Methods

Dataset Information

We conducted the experiments on inspection data (including survey videos and the annotations of interesting events), provided by three different oil and gas companies in the UK. The events' annotation information was created by trained data coordinators through a manual inspection process, i.e., watching survey videos and recording the time when events are detected. The annotated data was checked for correctness three times, with the first check from the data coordinator on the vessel, subsequently by the QC personnel at the data center, and finally, on the extracted video frames by another data coordinator for confirmation. For some types of anomalies, the annotators not only relied solely on visual footage but also used information from sonar echo which maps the seabed terrain, making annotation for these events consistent. If some anomalous events were missed in the first stage, they will be checked and verified by the QC personnel in the office before generating the report. The following types of anomalies are annotated:

- Anode: Bracelet anodes are usually designed to prevent corrosion in subsea pipelines. During a manual inspection, the data coordinator identifies the anodes visually by the orthogonal banding around the pipeline, and these anodes do not have any surface vegetation growth [10].
- Debris: This class is for instances when there are objects, such as scaffolding (first row, second column) or boulders (third row, second column of Fig. 1), that touch the subsea pipeline
- Exposure: This class is for cases where the pipeline is broken and the internal area is exposed to the environment. An example can be seen on the third row, the third column of Fig. 1.
- Freespan: This class denotes cases in which the pipeline segment is elevated and not supported by the sea-

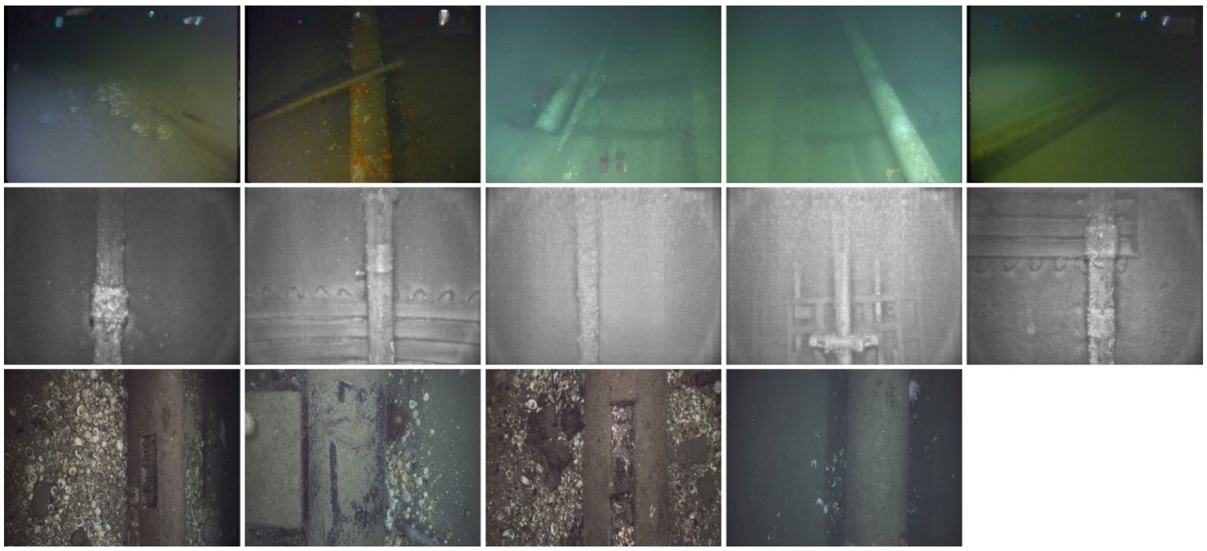


Fig. 1 Examples of events in subsea pipeline images. Each row represents different datasets (A, B, and C). First row (dataset A), left to right: anode, debris, exposure, freespan, normal. Second row (dataset B), left to right: Anode, fieldjoint, normal, freespan, span correction. Third row (dataset C), left to right: concrete damage, debris, exposure, normal

bed, either due to seabed erosion or uneven seabed during installation. It should be noted that freespan is more apparent on the starboard and port video feeds compared to the center.

- Normal: This class denotes all cases where the image is normal.
- Fieldjoint: This class denotes cases where two pipe sections meet and are welded together and is recognised during manual inspection based on the depression on the pipeline surface.
- Concrete damage: This class denotes cases where the outer concrete layer of the pipeline has been damaged.
- Span correction: This class denotes cases where the pipeline span is corrected by placing some instrument such as sand or grout bag underneath the pipeline [25].

It is noted that each company is interested in a different subset of these event types. The sources of the data were anonymised as Dataset A, B, and C. Figure 1 shows some examples of extracted frames of each dataset. It is recognised that different datasets have different characteristics because of different data acquisition conditions, e.g., different video recording devices and different environmental conditions. For example, Dataset B contains grayscale images while Datasets A and C contain colour images. Overall, the images from Dataset A and B are of medium quality, while the quality of images in Dataset C is better. The videos from which Dataset A, Dataset B, and Dataset C were extracted have a frame rate of around 25 frames per second. Each image in Dataset A has a height of 576 and a width of 768, while for Dataset B the height is 890 and the width is 1336, and for Dataset C the height is 576 and the width is 704. The average video length from which the datasets were extracted ranged from 12 to 15 min. The information details of the three datasets are summarised in Table 1.

Table 1 General information about the three datasets

Information	Dataset A	Dataset B	Dataset C
Image type	Color	Grayscale	Color
Quality	Medium	Medium	High
Average video length (in minutes)	13.27	14.46	12.35
Number of classes	5	5	4
Image size (width×height)	768×576	1336×890	704×576

Dataset Preprocessing

In this section, we discuss the preprocessing steps involved in the creation and annotation of the datasets. We extracted the video frames from the survey videos and then matched them with the associated annotations to create three datasets. Since consecutive frames are highly correlated to each other, one frame per second is extracted from the survey videos. The ground truth labels were annotated by experts in the participating companies. When the frames were extracted from the survey videos, we also performed a further manual inspection to ensure no inconsistencies in the data labels. We performed data augmentation to increase the quantity and diversity of extracted images.

The augmentation processes include horizontal and vertical flips, random rotation by 15 degrees, random width shift and height shift, random shearing, and random zoom. After preprocessing, dataset A has 5 classes, containing a total of 819 images, among which there are 260 normal images, 139 anode images, 102 debris images, 120 exposure images, and 198 freespan images. In Dataset B there are a total of 5212 images from 5 classes, among which there are 3702 normal images, 263 anode images, 691 fieldjoint images, 273 freespan images, and 283 span correction images. For Dataset C, there are 15 concrete damage images, 23 debris images, 908 exposure images, and 945 normal images, making a total of 1891 images from 4 classes. The detailed information about the class distribution in each dataset is summarised in Table 2. During the training of the deep learning-based classifiers, the dataset is divided into training, validation, and testing sets. The validation and testing sets were each chosen from 10% of the entire dataset.

Deep Neural Networks

The architectures used in our proposed method are based on the Convolutional Neural Network (CNN) and the Transformer architectures. A CNN consists of three main types of layers: convolutional, pooling, and fully connected. In the convolutional layer, a set of filters is individually convolved with the input image to generate a series of feature maps as outputs [26]. In the pooling layer, the input is downsampled via a max or averaging operation. Finally, in the fully connected layer, the input vector is multiplied by the layer weights. The Transformer architecture is a sequence-to-sequence model consisting of an encoder and a decoder, each of which is a stack of identical blocks. Each encoder block is composed of a multi-head self-attention module followed by a position-wise feedforward neural network, residual connection, and layer normalisation. The decoder block is similar but has additional cross-attention modules which connect with the encoder. Moreover, the self-attention modules in the decoder are modified to prevent each position from attending to subsequent positions [27]. In this study, six base classifiers based on CNN were used: VGG16 [26], ResNet50 [19], InceptionV3 [28], InceptionResNetV2 [29], DenseNet121 [30], XCeption [31], and three base classifiers based on the Transformer architecture were used: VisionTransformer [32], MaxViT [33] and Swin-Transformer [34], making a total of nine base classifiers.

Table 2 Number of instances of each class in experimental datasets

Class	Dataset A	Dataset B	Dataset C
Anode	139	263	-
Debris	102	-	23
Exposure	120	-	908
Freespan	198	273	-
Normal	260	3702	945
Fieldjoint	-	691	-
Concrete damage	-	-	15
Span correction	-	283	-

- VGG16 is a CNN with very small (3×3) convolutional filters, which allows the network to be deeper compared to previously introduced CNNs. VGG16 consists of five convolutional blocks, followed by three fully connected layers. Each convolutional block is composed of a number of convolutional layers followed by one pooling layer. The first two convolutional blocks consist of two convolutional layers and one pooling layer, while the remaining three blocks have three convolutional layers and one pooling layer each. This architecture was the winning solution to the ImageNet Challenge 2014 competition.
- ResNet50 is a CNN architecture which solved the problem of extending the depth of DNNs. Previous architectures, such as VGG16, could not extend the network beyond a certain depth. He et al. [19] introduced the ResNet architecture, in which a layer is connected with the next layer as well as the layer after that via a skip connection, which allows the network depth to be extended significantly. The ResNet50 architecture consists of one 7×7 convolutional layer and a pooling layer, followed by four residual blocks. In each residual block, there are three convolutional layers and a skip connection. This architecture won first place on the ILSVRC 2015 classification task.
- InceptionV3 is a CNN architecture that consists of a number of blocks called Inception blocks in which instead of using several convolutional layers sequentially, a number of convolutional layers, such as 3×3, 5×5, and 1×1 are used in parallel. Their outputs are then concatenated as input to the next block, which allows for dimensionality reduction. To further reduce the model size, the convolutional layers are factorised using asymmetric convolution, in which a NxN convolution is replaced by a 1xN and Nx1 convolution. An auxiliary classifier is also used to act as a regulariser to reduce the vanishing gradient problem.
- InceptionResNetV2 is a CNN architecture that combines the ideas of ResNet and InceptionV3. In this architecture, each Inception block is followed by a filter expansion layer (1×1 convolution without activation) to scale up the filter bank dimension before addition to match the input depth. Batch normalisation is applied only to the traditional layers but not the summation. The authors also

note that if the number of filters exceeds 1000, then the network starts to exhibit instabilities, and thus proposed to scale down the residual by a small multiplier before adding them to the previous layer activation to stabilise training.

- DensetNet121 is a CNN architecture in which each layer is connected to all previous layers. Counterintuitively, this requires fewer parameters compared to traditional CNNs since there is no need to learn redundant feature maps. Another advantage of this architecture is that each layer has direct access to the gradients from the loss function and the original input signal, which improves the information and gradient flow, making them easy to train. DenseNet121 is divided into a number of dense blocks in which each layer obtains additional inputs from all preceding layers and passes on its feature maps to all subsequent layers, which facilitates down-sampling in the architecture.
- XCeption is a CNN architecture that improves upon the Inception architecture by using depthwise separable convolution, which consists of NxN spatial convolution performed independently over each channel, followed by 1×1 convolution. This is the reverse of Inception, which applies 1×1 convolution before applying each of the filters on each of the depth spaces. Another difference is that in the Inception architecture, there is a non-linearity operation after each operation, while the XCeption architecture does not introduce any non-linearity after these operations. This architecture slightly outperforms Inception on the ImageNet dataset but achieves much better results on a larger dataset comprising 350 million images and 17,000 classes.
- VisionTransformer (ViT) is an application of the popular Transformer architecture [27] to computer vision. The ViT model represents an image as a series of 16×16 image patches, similar to the series of word embeddings when transformers are used for the text data type. Patch embedding with positional encoding is used to embed the image patches, and the resulting vector is fed to a standard Transformer encoder. When pretrained on a large dataset and transferred to small benchmarks, ViT achieves excellent results compared to CNN-based models while requiring much less time to train.
- Multi-axis Vision Transformer (MaxViT), a variant of ViT, is based on the observation that the lack of adoption of transformer-based vision backbones is due to the lack of scalability of the self-attention mechanism concerning the image size. Based on this observation, the authors introduced multi-axis attention, which consists of two aspects: blocked local and dilated global attention. This enables global–local spatial interactions for input resolutions of any size with only linear complexity. The authors also combined the attention model with convolution, allowing the network to access global information even in the earlier, high-resolution stages. For image classification, MaxViT achieves state-of-the-art performance under various settings, while for downstream tasks such as object detection and visual aesthetic assessment, the model also delivers favourable performance.
- SwinTransformer is a transformer model that works as a general-purpose backbone in computer vision. The authors of this model note that the challenges in adapting the transformer architecture to computer vision are due to differences between the two domains, such as large variations in the scale of visual entities and the high pixel resolution in images compared to text. Based on this observation, they proposed the Swin transformer, which is a hierarchical transformer model based on shifting windows. The shifted windows increase efficiency since they limit self-attention computation to non-overlapping windows while still allowing cross-window connections. Another advantage of this architecture is that it can model at various scales and has linear computational complexity concerning image size, making it compatible with a broad range of vision tasks, including image classification, object detection, and semantic segmentation.

Ensemble Model

In this section, we introduce our proposed ensemble model. Figure 2 shows the high-level overview of our proposed ensemble. Let $\mathbf{D} = \{\mathbf{I}_n, \mathbf{Y}_n\}_{n=1}^{N}$ be the training set where N is the number of images, $\mathbf{I}_n$ is an input image of size (H, W, C) in which H is the image height, W is the image width, and C is the number of channels in the images. The ground truth $\mathbf{Y}_n$ denotes the class to which $\mathbf{I}_n$ belongs, i.e., $\mathbf{Y}_n \in \mathcal{Y}$ where $\mathcal{Y} = \left\{y_m\right\}, m = 1, \dots M$ is the set of M classes. We aim to learn a hypothesis $\mathbf{h} : \mathbf{I}_n \rightarrow \mathbf{Y}_n$ (i.e., classifier) to approximate the unknown relationship between images and their corresponding ground truths. We also denote $\{\mathcal{L}_k\}_{k=1}^{K}$ by the set of K classification algorithms (methods). Each classification algorithm $\mathcal{L}_k$ learns on the dataset $\mathbf{D}$ to obtain a trained classifier $\mathbf{h}_k$. To clarify, a classification algorithm $\mathcal{L}_k$ can be considered as a ML or DL model whose parameters (such as weights, connections, etc.) are initialised randomly, while the trained classifier $\mathbf{h}_k$ is the same model but with the parameters found after training on the given data. In ensemble learning, K classifiers $\{\mathbf{h}_k\}_{k=1}^{K}$ are combined by a combining model $\mathbf{C}$ to obtain the final classification result. The proposed ensemble model includes two phases:

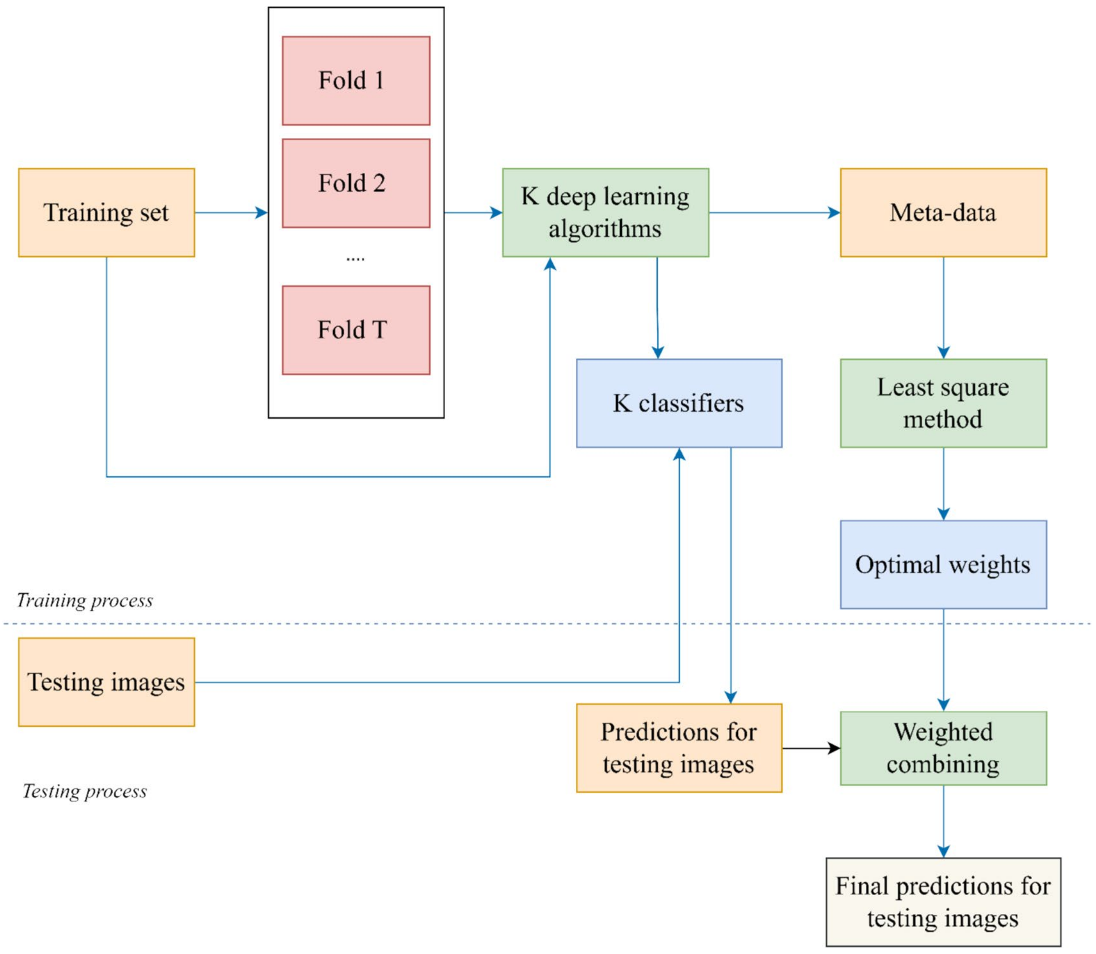


Fig. 2 The proposed ensemble of deep learning-based classifiers

-Training K classification algorithms on the training set $\mathbf{D}$ to obtain an ensemble of classifiers $\{\mathbf{h}_k\}_{k=1}^K$

-Training weights of $\{\mathbf{h}_k\}_{k=1}^K$: We propose to combine $\{\mathbf{h}_k\}_{k=1}^K$ based on a weighted combining method. Each classifier is associated with a weight which is found based on the relation between the training images and their associated ground truths.

In the first phase, K deep learning methods are used to train the classifiers $\{\mathbf{h}_k\}_{k=1}^K$. It is noted that training deep learning models with random weight initialisation requires a large dataset causing a challenge when applying to small datasets. In this study, we adopt a common approach known as transfer learning [35], in which a model already trained on a large dataset will be used for training on another smaller dataset. This approach works because the initial layers of a DNN extract generic features, such as color blob detectors or edge detectors, which suggests that the weights of these layers can be reused in different scenarios. In contrast, the subsequent layers are responsible for more finetuned features and therefore they should be retrained for each dataset. In this study, deep learning models were pre-trained on ImageNet [36] and their obtained weights were then continued to train on our datasets during the transfer learning process.

An optimisation algorithm also needs to be chosen to perform training with deep learning models. Two optimisation algorithms are popular among the deep learning community namely stochastic gradient descent (SGD) and Adam [37]. The SGD algorithm works by calculating the gradient concerning a loss function on each batch of the dataset, and a momentum term is usually added to prevent the algorithm from getting stuck in a local optimum. SGD was central to many successes in deep learning [36], however, this algorithm is known to be sensitive to hyperparameter choice. The Adam (adaptive moment estimation) algorithm meanwhile solves this problem by calculating the exponential moving average of the gradient, the squared gradient, and two hyperparameters β_1 and β_2 control the decay rates of these moving averages. The idea behind this algorithm is that these moving averages are approximations of the first and second raw moments of the gradient. The authors in [36] noted that the exponential decay rate β_1 should be chosen such that the exponential moving average assigns small weights to gradients far in the past. They also noted that the large value of β_2 are required so that the algorithm could be robust with respect to sparse gradients. They experimentally found that $\beta_1 = 0.9$ and $\beta_2 = 0.999$ provides the best results, which we use in this paper. In [36], a bias correction method is also introduced to ensure that these moving averages are initialised correctly. They found that $E\left[v_t\right]$, which is the expected value of the exponential moving average at timestep t, relates to the true second moment $E\left[g_t^2\right]$ via the following equation: $E\left[v_t\right] = E\left[g_t^2\right]\left(1 - \beta_2^t\right) + \zeta$, in which ζ is an error term which can be kept small with a proper value of β_1 such that the exponential moving average assigns small weights to gradients far in the past, and β_2^t is β_2 raised to the power of t. Based on this observation, they divided the first and second moments with $\left(1 - \beta_1^t\right)$ and $\left(1 - \beta_2^t\right)$ respectively to correct for the bias.

The optimisation algorithm was used to minimise a loss function. The most common loss function used in the classification problem is the cross-entropy loss, which is defined as follows:

$$Loss = -\sum_{n=1}^{N}\sum_{c=1}^{M} y_{n,m}\log\left(p_{n,m}\right) \qquad (1)$$

where N is the number of observations, M is the number of classes, $y_{n,m}$ is a binary variable indicating whether the n^{th} observation is in the m^{th} class, and $p_{n,m}$ is the probability the n^{th} observation is predicted to be in the m^{th} class. This loss function is used by all deep learning models in the paper, except for InceptionV3. In InceptionV3, a modified loss function called Inception loss is used in which an auxiliary classifier is used in the intermediate stage of the network to improve the gradient signal. During the training process, the final loss is calculated as a weighted combination of the real and auxiliary classifier (which is weighted by 0.3), while in the testing process, the auxiliary classifier is discarded.

The images in our datasets were resized to 224×224 before inputting into the deep learning models. Six deep learning classification models, namely VGG16, Resnet50, InceptionV3, InceptionResNetV2, DenseNet121, and XCeption were used from the Keras deep learning framework, while the three transformer-based models, Vision transformer, MaxViT, and Swin transformer were used from the Pytorch framework. Since the pre-trained models were trained on the ImageNet dataset, which has a different number of classes compared to our datasets, the last layer of each model was replaced by a fully-connected layer with the number of outputs being the number of classes for prediction. Finally, two additional hyperparameters need to be determined: the number of epochs and the batch size. The number of epochs denotes the number of passes the deep learning model will go through all the data, while the batch size denotes the number of instances the deep learning model will work through before updating its internal parameters in each epoch. In this paper, the number of epochs was set to 100 and the batch size was set to 16.

In the second phase, we obtain the combining weights by exploiting the relation between training images and their associated ground truths. We first generate the meta-data which are the predictions of deep learning-based classifiers for training images [17]. In this study, we divide the training set $\mathbf{D}$ into T disjointed parts $\{\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_T\}$ in which the number of training images in each $\mathbf{D}_t$ $(t = 1, \ldots, T)$ are nearly equal. The procedure loops through each $\mathbf{D}_t$ so that deep learning-based classification methods $\{\mathcal{L}_k\}_{k=1}^{K}$ are trained on $\mathbf{D}\backslash\mathbf{D}_t$ to obtain classifiers $\mathbf{h}_{k,t}$ $(t = 1, \ldots, T; k = 1, \ldots, K)$. The training images in $\mathbf{D}_t$ are then classified by using $\mathbf{h}_{k,t}$. For each image $\mathbf{I} \in \mathbf{D}_t$, let $P_k(y_m|\mathbf{I})$ be the probability prediction that $\mathbf{h}_{k,t}$ assigns to image $\mathbf{I}$ to be in class y_m, $0 \le P_k(y_m|\mathbf{I}) \le 1$ [17]. The predictions of K classifiers $\{\mathbf{h}_{k,t}\}$ showing the probabilities that image $\mathbf{I}$ belongs to all M classes are given by the following vector of size $M * K$:

$$\mathbf{L}(\mathbf{I}) = \left[P_1\left(y_1|\mathbf{I}\right) \ \ldots \ P_1\left(y_M|\mathbf{I}\right) \ \ldots \ P_K\left(y_1|\mathbf{I}\right) \ \ldots \ P_K\left(y_M|\mathbf{I}\right)\right] \qquad (2)$$

Using the representation in (2) for N training images in $\mathbf{D}$, the predictions of $\mathbf{D}$ (called meta-data) is a matrix of size $(N, M * K)$ given by the following representation [17]:

$$\mathbf{L} = \begin{bmatrix} P_1(y_1|\mathbf{I}_1) & \cdots & P_1(y_M|\mathbf{I}_1) & \ldots & P_K(y_1|\mathbf{I}_1) & \ldots & P_K(y_M|\mathbf{I}_1) \\ P_1(y_1|\mathbf{I}_2) & \cdots & P_1(y_M|\mathbf{I}_2) & \ldots & P_K(y_1|\mathbf{I}_2) & \ldots & P_K(y_M|\mathbf{I}_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_1(y_1|\mathbf{I}_N) & \cdots & P_1(y_M|\mathbf{I}_N) & \ldots & P_K(y_1|\mathbf{I}_N) & \ldots & P_K(y_M|\mathbf{I}_N) \end{bmatrix} \qquad (3)$$

Next, we propose an approach to obtain weights for the classifiers $\{\mathbf{h}_k\}_{k=1}^{K}$ obtained in phase 1. Let $\mathbb{W}$ be a weight matrix

$$\mathbb{W} = \begin{bmatrix} w_{11} & \cdots & w_{1M} \\ w_{21} & \ldots & w_{2M} \\ \vdots & \ddots & \vdots \\ w_{K1} & \cdots & w_{KM} \end{bmatrix} \qquad (4)$$

in which $w_{k,m}$ is the weight associated with the classifier $\mathbf{h}_k$ concerning the class $y_m (k = 1, \ldots, K; m = 1, .., M)$. Given an image $\mathbf{I}_n$, the ensemble prediction will be a weighted combination of the predictions of the classifiers:

$$pred_m(\mathbf{I}_n) = \sum_{k=1}^{K} w_{k,m} P_k(y_m|\mathbf{I}_n) \tag{5}$$

The probability made by the ensemble for each class can be found by normalizing the predictions in the above equation. In this paper, we formulate an optimisation problem by minimising the distance between the weighted combining results on the predictions in **L** and the ground truth of training images [17] [38] [39]. This idea is based on the concept that the derived weights will ensure the predictions of the base classifiers, when combined with the weights for each observation in the training set (with known labels), align most closely with the true class label of that observation, compared to other class labels.

Based on the meta-data **L**, we extract the probabilities associated with class y_m:

$$\mathbf{L}_\mathrm{m} = \begin{bmatrix} P_1(y_m|\mathbf{I}_1) & \cdots & P_K(y_m|\mathbf{I}_1) \\ P_1(y_m|\mathbf{I}_2) & \dots & P_K(y_m|\mathbf{I}_2) \\ \vdots & \ddots & \vdots \\ P_1(y_m|\mathbf{I}_N) & \cdots & P_K(y_m|\mathbf{I}_N) \end{bmatrix} \tag{6}$$

We also define a crisp label vector (i.e., belonging to $\{0,1\}$) of size $(N, 1)$ associated with class y_m as follows:

$$\mathbb{Y}_m = \begin{bmatrix} \mathbb{I}[\mathbf{Y}_1 = y_m] \\ \dots \\ \mathbb{I}[\mathbf{Y}_N = y_m] \end{bmatrix} \tag{7}$$

where $\mathbb{I}[.]$ is the indicator function. For example, on a dataset with 3 classes (i.e. $M = 3$) and 4 training instances (i.e. $N = 4$), suppose the 1st instance is from the 2nd class, the 2nd instance is from the 1st class, the 3rd instance is from the 2nd class and the 4th instance is from the 3rd class. Then the crisp label vector for each class would be as follows:

$$\mathbb{Y}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \mathbb{Y}_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \mathbb{Y}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{8}$$

By defining the crips label vector and the probabilities associated with class y_m, for each class y_m, the following optimisation problem will be solved to obtain the weight vector $\mathbb{W}_m$ which is the m^{th} column of $\mathbb{W}$:

$$\min_{\mathbb{W}_m} ||\mathbf{L}_\mathrm{m}\mathbb{W}_m - \mathbb{Y}_m||_2 \\ s.t 0 < w_{k,m} < 1 (k \in \{1, \dots, K\}) \tag{9}$$

The optimisation problem in (9) aims to find the weights so that combining result $\mathbf{L}_\mathrm{m}\mathbb{W}_m$ align most closely to 1 for class y_m while align most closely to 0 for the other classes. Different constraints can be made on the weight matrix, such as Bounded-Variables Least Square or Bounded Variable with Constant Sum. For simplicity, in this paper, we choose to bound the weights between 0 and 1 like in [40]. The optimisation problem in (9) can be solved by the Least Square method with the bounded constraint. The algorithm first finds the unconstrained least-square solution using the Moore–Penrose inverse. If the solution lies within the boundary, then it is returned, otherwise, the Trust Region Reflective algorithm [41] is used to solve the problem.

Given a test image $\mathbf{I}_{test}$, the ensemble of classifiers $\{\mathbf{h}_k\}_{k=1}^{K}$ will predict probabilities $P_k(y_m|\mathbf{I}_{test})$ that image belongs to each class. Then the optimal weights $\widehat{w}_{k,m}$ $(k = 1, \dots, K; m = 1, \dots, M)$ obtained by solving (8) are used to combine the probabilities $P_k(y_m|\mathbf{I}_{test})$ for the final prediction:

$$\mathbf{I}_{test} \in y_{\widehat{m}} \text{where} \widehat{m} = argmax_{m=1,\dots,M} \sum_{k=1}^{K} \widehat{w}_{k,m} P_k(y_m|\mathbf{I}_{test}) \tag{10}$$

The procedure for training the proposed ensemble is described in Algorithm 1. The algorithm inputs the training set and the deep learning methods and will output the ensemble of classifiers and the optimal weights. Lines 1–11 describe the meta-data generation process. In lines 1–2, the training set is divided into T folds, and **L** is initialised as the empty set. From line 3 to line 11, for each fold $\mathbf{D}_t$, the deep learning-based classifiers first are trained on $\mathbf{D}\backslash\mathbf{D}_t$ (line 5). The classifiers $\mathbf{h}_{k,t}$ will classify each image in $\mathbf{D}_t$ and output the probability for each class (line 7) which will then be added to **L** (line 8). After this T-fold Crossover procedure, we obtained **L** in the form of (3).

Lines 12–14 describe the generation process for base classifiers, in which each deep learning method is trained on the training set (line 13). Lines 15–19 describe the weight vector optimisation process. For each class, we first extract $\mathbf{L}_m$ and $\mathbb{Y}_m$ using Eq. (6) and (7) respectively (lines 16–17). Then, the optimal weights are found by solving the Eq. (9) (line 18). The ensemble of classifiers and the optimal weights are returned as the outputs of the training process (line 20).

Algorithm 2 describes the testing process of the proposed ensemble. The algorithm inputs the test image $\mathbf{I}_{test}$, ensemble of classifiers $\{\mathbf{h}_k\}_{k=1}^{K}$ and the combining weight matrix $\widehat{\mathbb{W}}$. The classifiers are used to predict the probability that image belongs to each class (lines 1–3). Then Eq. (10) is used to obtain the final prediction (line 4) which will be the output of the testing process (line 5).

Algorithm 1 Training the ensemble model

```
Input: Training set $\mathbf{D} = \{\mathbf{I}_n, \mathbf{Y}_n\}_{n=1}^{N}$, deep learning methods for classification $\{\mathcal{L}_k\}_{k=1}^{K}$
Output: Ensemble of classifiers $\{\mathbf{h}_k\}_{k=1}^{K}$ and optimal weights $\widehat{\mathbb{W}}$
(Meta-data generation)
1.  $\{\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_T\} = T - partition(\mathbf{D})$
2.  $\mathbf{L} = \emptyset$
3.  For $t \leftarrow 1$ to $T$ do
4.    For $k \leftarrow 1$ to $K$ do
5.      $\mathbf{h}_{k,t}$ = Train$(\mathcal{L}_k, \mathbf{D} \backslash \mathbf{D}_t)$
6.      For I in $\mathbf{D}_t$ do
7.        $\{\mathrm{P}_k(y_m|\mathbf{I})\}_{m=1}^{M}$ = Classify$(\mathbf{h}_{k,t}, \mathbf{I})$
8.        Add $\{\mathrm{P}_k(y_m|\mathbf{I})\}_{m=1}^{M}$ to $\mathbf{L}$ based on Equation (3)
9.      End for
10.   End for
11. End for
(Base classifiers generation)
12. For $k \leftarrow 1$ to $K$ do
13.   $\mathbf{h}_k$ = Train$(\mathcal{L}_k, \mathbf{D})$
14. End for
(Weighted vector optimization)
15. For $\boldsymbol{m} \leftarrow \mathbf{1}$ to $\boldsymbol{M}$ do
16.   Get $\mathbf{L}_m$ from $\mathbf{L}$ by Equation (6)
17.   Get $\mathbb{Y}_m$ from $\mathbf{Y}_n$ by Equation (7)
18.   Obtain optimal weights $\widehat{\mathbb{W}}_m = \{w_{k,m}\}, k = 1, \ldots, K$ by solving Equation (9)
19. End for
20. Return $\{\mathbf{h}_k\}_{k=1}^{K}$ and $\widehat{\mathbb{W}} = \{\widehat{\mathbb{W}}_m\}_{m=1}^{M}$
```

Algorithm 2 Testing the ensemble model

```
Input: Test image $\mathbf{I}_{test}$, ensemble of classifiers $\{\mathbf{h}_k\}_{k=1}^{K}$ and the weight matrix $\mathbb{W}$
Output: Prediction for $\mathbf{I}_{test}$
1.  For $k \leftarrow 1$ to $K$ do
2.      $\{P_k(y_m|\mathbf{I}_{test})\}_{m=1}^{M} = \text{Classify}(\mathbf{h}_k, \mathbf{I}_{test})$
3.  End for
4.  Use Equation (10) to get the final prediction $y_{\hat{m}}$
5.  Return $y_{\hat{m}}$
```

Experimental Results and Discussion

Comparison with the Base Classifiers

In this section, we compare and discuss the experimental results of the base classifiers, the proposed ensemble, and the benchmark ensemble method. We run each model one time following the experiments in [42] [43]. The number of folds for cross-validation needs to be chosen carefully since a low number of folds would mean that the algorithm would not generalise well to unseen data, while a large number of folds might require too much time to run. In this study, we use fivefold cross-validation in this paper by referring to the experiments in [44] [45] [17].

Table 3 shows the accuracy of the 9 base classifiers and the proposed ensemble on Dataset A, B, and C. It can be seen that the proposed ensemble achieved the best results while VGG16 obtained the worst results among all methods on the three experimental datasets. On Dataset A, the proposed ensemble achieved an accuracy of 0.7753, followed by MaxViT at 0.7640 and SwinTransformer at 0.7303, while the accuracy of the other base classifiers ranged from around 0.59 to around 0.69 only. For Dataset B, the proposed ensemble obtained a score of 0.7837, which is higher than the second-best (InceptionV3) by 1.24%. VGG16 (0.7474), Resnet50 (0.7522), and XCeption (0.7455) on the other hand, are the three worst performance methods on Dataset B. For Dataset C, both the proposed ensemble and MaxViT gained the best accuracy at 0.9974, followed by DenseNet121 (0.9948). Unlike the other two datasets, on Dataset C, the accuracy of all models is very high, from around 0.98 to 0.99.

Table 4 shows the F1-score of the 9 base classifiers and the proposed ensemble on Dataset A, B, and C. The proposed ensemble also ranks first among all methods on the three datasets. On Dataset A, the proposed ensemble

Table 3 The accuracies of the base classifiers and the proposed ensemble

Method	Dataset A	Dataset B	Dataset C
VGG16	0.5955	0.7474	0.9870
Resnet50	0.6517	0.7522	0.9922
InceptionV3	0.6742	0.7713	0.9922
InceptionResNetV2	0.6966	0.7598	0.9870
DenseNet121	0.6854	0.7675	0.9948
XCeption	0.6404	0.7455	0.9896
VisionTransformer	0.5955	0.7598	0.9792
MaxViT	0.7640	0.7579	**0.9974**
SwinTransformer	0.7303	0.7675	0.9870
Proposed ensemble	**0.7753**	**0.7837**	**0.9974**

Table 4 The F1 scores of the base classifiers and the proposed ensemble

Method	Dataset A	Dataset B	Dataset C
VGG16	0.5939	0.4150	0.8710
Resnet50	0.6177	0.4297	0.8480
InceptionV3	0.6790	0.5142	0.8869
InceptionResNetV2	0.6931	0.4643	0.8293
DenseNet121	0.6656	0.4992	0.9623
XCeption	0.6180	0.4554	0.8246
VisionTransformer	0.5926	0.4891	0.6933
MaxViT	0.7518	0.5524	**0.9636**
SwinTransformer	0.7109	0.4865	0.8293
Proposed ensemble	**0.7765**	**0.5680**	**0.9636**

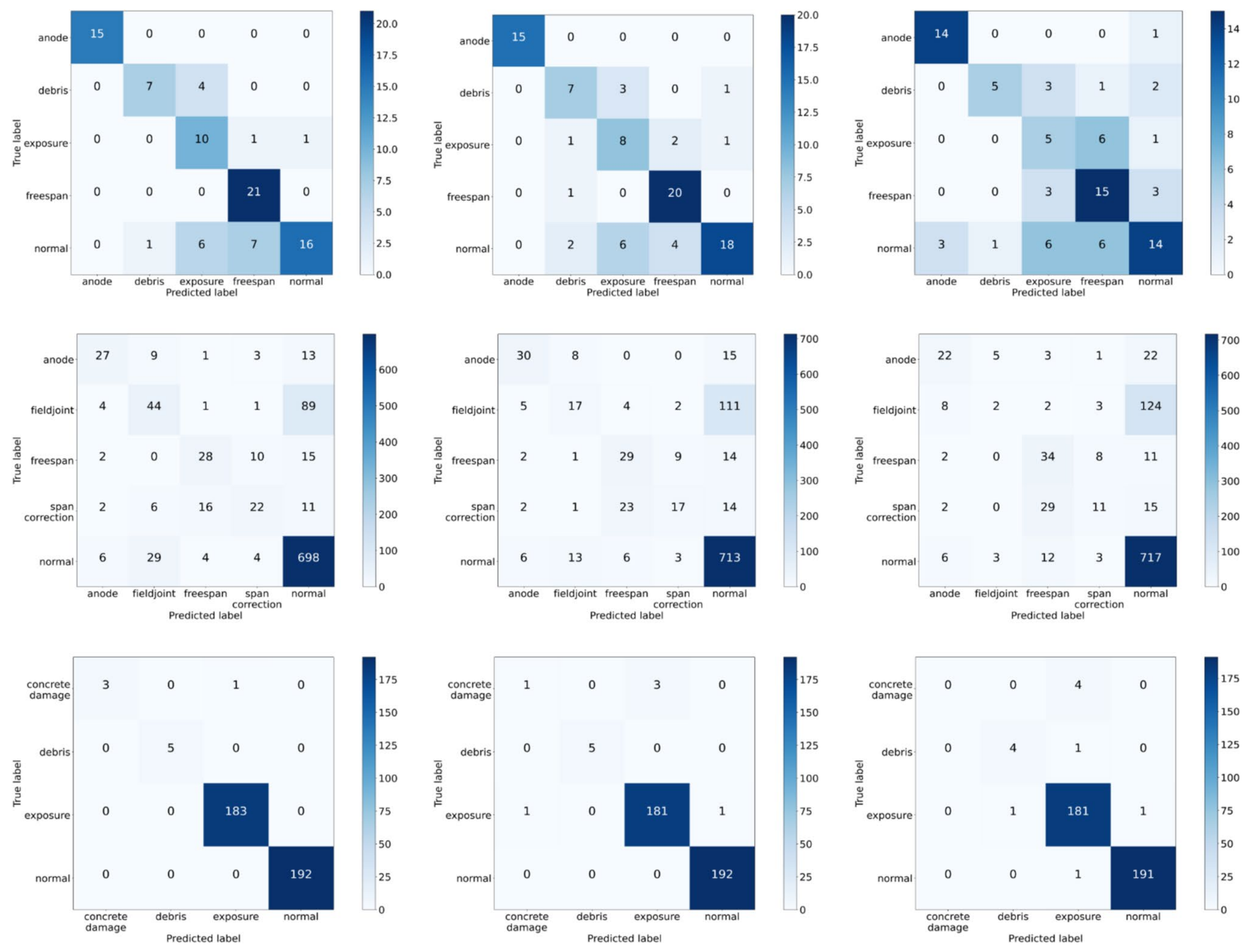


Fig. 3 Confusion matrices (from left to right) of the proposed ensemble, MaxViT and VGG16 on Dataset A, the proposed ensemble, InceptionV3 and Resnet50 on Dataset B, the proposed ensemble, SwinTrasnformer and VisionTransformer on Dataset C

achieved a F1 score of 0.7765, which is higher than the second-best (MaxViT) by 2.47%. The poorest performing methods on this dataset are VGG16 and VisionTransformer at 0.5939 and 0.5926, while the F1 scores of the other base classifiers are in the range of 0.61 to 0.71. For Dataset B, the proposed ensemble obtained the highest score of 0.5680, followed by MaxViT (0.5524) and InceptionV3 (0.5142) while the F1 scores of the other base classifiers are from 0.4150 of VGG16 to 0.4992 of DenseNet121. On Dataset C, both the proposed ensemble and MaxViT achieved the highest score at 0.9636, followed by DenseNet121 (0.9623) and InceptionV3 (0.8869). The other base classifiers obtained scores from 0.8246 to 0.8710 except VisionTransformer (0.6933 of F1 score only).

We analysed the confusion matrices of the top 2 classifiers and a poorer classifier on each dataset to show the outstanding performance of the proposed ensemble. The first row of Fig. 3 (from left to right) shows the confusion matrix on Dataset A of the proposed ensemble (which has an accuracy of 0.7753), MaxViT (0.7640), and VGG16 (0.5955). The confusion matrices of the other classifiers can be seen in the Online Resources. Compared to the proposed ensemble, MaxViT had more wrong misclassifications, such as predicting two images of class exposure and freespan respectively to be in class debris or misclassifying a debris image to be normal. MaxViT had the same number of correct predictions for anode and debris compared to the proposed ensemble (at 15 and 7 respectively), however, while the proposed ensemble had 10 correct predictions for the exposure class, and 21 for the freespan class, MaxViT only had 8, and 20 predictions respectively. On the other hand, while the proposed ensemble only had 16 correct normal predictions, MaxViT had 18 correct predictions for the normal class. MaxViT also had fewer misclassifications of normal images into freespan

Table 5 Accuracy of sum rule, snapshot ensemble models, and the proposed ensemble

Method	Dataset A	Dataset B	Dataset C
Sum Rule	**0.7753**	0.7799	0.9922
SE-5-VGG16	0.3933	0.7091	0.7422
SE-5-Resnet50	0.6517	0.7837	0.9792
SE-5-InceptionV3	0.4382	0.7282	0.9844
SE-5-InceptionResNetV2	0.6067	0.7388	0.9948
SE-5-DenseNet121	0.6854	0.7828	0.9896
SE-5-XCeption	0.6405	0.7809	0.9740
SE-5-VisionTransformer	0.5955	0.7569	0.9818
SE-5-MaxViT	0.7416	0.7876	0.9948
SE-5-SwinTransformer	0.7303	0.7617	0.9870
SE-10-VGG16	0.3933	0.7091	0.7500
SE-10-Resnet50	0.6629	0.7876	0.9844
SE-10-InceptionV3	0.4270	0.7292	**0.9974**
SE-10-InceptionResNetV2	0.6742	0.7340	**0.9974**
SE-10-DenseNet121	0.6517	0.7885	**0.9974**
SE-10-XCeption	0.6517	0.7761	0.9948
SE-10-VisionTransformer	0.6405	0.7598	0.9844
SE-10-MaxViT	0.7303	**0.7895**	**0.9974**
SE-10-SwinTransformer	0.7303	0.7569	0.9922
Proposed ensemble	**0.7753**	0.7837	**0.9974**

Table 6 F1 score of sum rule, snapshot ensemble models, and the proposed ensemble

Method	Dataset A	Dataset B	Dataset C
Sum Rule	0.7658	0.5208	0.8480
SE-5-VGG16	0.2636	0.1660	0.3702
SE-5-Resnet50	0.6712	0.4838	0.6197
SE-5-InceptionV3	0.3672	0.2733	0.7388
SE-5-InceptionResNetV2	0.6266	0.2984	0.9402
SE-5-DenseNet121	0.6849	0.5205	0.8466
SE-5-XCeption	0.6268	0.4497	0.8456
SE-5-VisionTransformer	0.5963	0.4758	0.6946
SE-5-MaxViT	0.7478	0.5595	0.9153
SE-5-SwinTransformer	0.7207	0.4535	0.8293
SE-10-VGG16	0.2296	0.1660	0.3748
SE-10-Resnet50	0.6585	0.4930	0.7626
SE-10-InceptionV3	0.3804	0.2652	**0.9636**
SE-10-InceptionResNetV2	0.6834	0.2758	**0.9636**
SE-10-DenseNet121	0.6468	0.4904	**0.9636**
SE-10-XCeption	0.6507	0.4343	0.9153
SE-10-VisionTransformer	0.6403	0.4802	0.6959
SE-10-MaxViT	0.7308	0.5459	**0.9636**
SE-10-SwinTransformer	0.7285	0.4378	0.8908
Proposed ensemble	**0.7765**	**0.5680**	**0.9636**

(4 images) compared to the proposed ensemble (7 images). Since VGG16 is a low-performing method on this dataset, it can be seen that it made many more errors compared to the proposed ensemble. For example, VGG16 misclassified one anode image as normal, three freespan images as normal, and three normal images as anode (as compared to 0 by both the proposed ensemble and MaxViT). Six exposure images were misclassified by VGG16 as freespan, as compared to 1 and 2 images by the proposed ensemble and MaxViT, respectively.

The second row of Fig. 3 (from left to right) shows the confusion matrix on Dataset B of the proposed ensemble, InceptionV3 and Resnet50 (which have accuracies of 0.7837, 0.7713, and 0.7522 respectively). It can be seen that the proposed ensemble had much fewer misclassified samples compared to InceptionV3 and Resnet50. For example, while the proposed ensemble wrongly predicted 89 fieldjoint images as normal, InceptionV3 and Resnet50 misclassified 111 and 124 images, respectively. The proposed ensemble also correctly classified 44 fieldjoint images as opposed to just 17 images by InceptionV3 and 2 images by Resnet50. For the span correction class, the proposed ensemble only misclassified 16 images as freespan, while InceptionV3 and Resnet50 wrongly classified 23 images and 29 span correction images as freespan. The number of normal images misclassified as freespan by Resnet50 is 12, which is two times higher than InceptionV3 and three times higher than the proposed ensemble. Concerning the anode class, ResNet50 and InceptionV3 were slightly better than the proposed ensemble in which the number of anode images classified as fieldjoint, freespan, and span correction by the proposed ensemble were 9, 1, and 3 while the corresponding numbers of Resnet50 and InceptionV3 were (5, 3 and 1) and (8, 0, 0) respectively.

The third row of Fig. 3 (from left to right) illustrates the confusion matrix on Dataset C of the proposed ensemble (which has an accuracy score of 0.9974), SwinTransformer (0.9870) and VisionTransformer (0.9792). It is noted that the confusion matrix of the proposed ensemble and some methods like MaxViT, DenseNet121, ResNet50, and InceptionV3 are nearly similar and we only showed that of the proposed ensemble. Unlike the other two datasets, for Dataset C the accuracies and F1 scores of the experimental methods are also much higher compared to those of Dataset A and Dataset B. It is noted that this dataset also has only four classes, while the other two datasets both have five classes. The proposed ensemble had one misclassified sample from concrete damage to exposure, while for SwinTransformer the corresponding number was 3. SwinTransformer also wrongly classified one exposure image as concrete damage and another as normal, while for these classes the proposed ensemble did not make any errors. Compared to SwinTransformer, VisionTransformer had more misclassified images

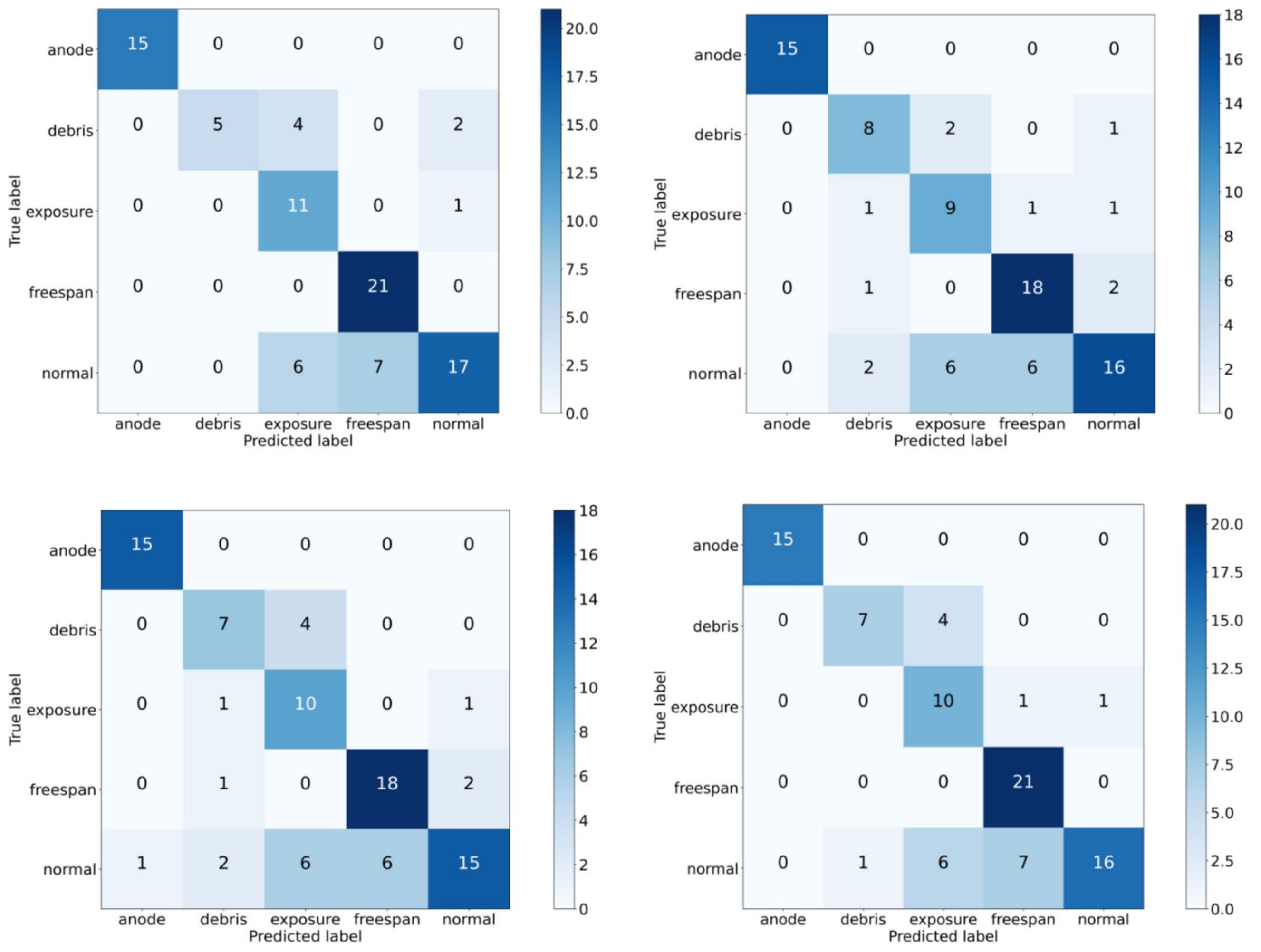


Fig. 4 Confusion matrix of sum rule, SE-5-MaxViT, SE-10-MaxViT and the proposed ensemble on Dataset A

in which this model did not manage to correctly predict any concrete damage images. Finally, while both the proposed ensemble and SwinTransformer did not misclassify any debris image as exposure, VisionTransformer made one mistake.

Comparison with Other Ensemble Methods

In this section, we compared the results of the proposed ensemble with two popular ensemble methods namely sum rule and snapshot ensemble [24]. The sum rule works by averaging the probability predictions of base classifiers. In the experiment, we used the same base classifiers as in the proposed method to construct an ensemble with the sum rule. Meanwhile, in the snapshot ensemble, a single deep learning model is trained, and different snapshots of this model are taken periodically to generate an ensemble of classifiers. The average of the predictions of all the snapshots (classifiers) is used as the final prediction. Two hyperparameters are required when using a snapshot ensemble: the number of snapshots and the number of epochs per snapshot. In this paper, we experimented on two different cases where 5 and 10 snapshots were used, with 40 epochs per snapshots following the experiments in [24]. Table 5 shows the accuracy of the proposed method, sum rule, and the snapshot ensemble methods, where SE-5-DenseNet121 for example denotes that the method is a snapshot ensemble of DenseNet121 with 5 snapshots. It can be seen that the proposed ensemble achieved the best result among all methods and the performance of the snapshot ensemble heavily depends on its base classifier. On Dataset A, both the proposed ensemble and Sum rule achieved the best result at 0.7753, with the second best being SE-5-MaxViT at 0.7416 followed by SE-5-SwinTransformer, SE-10-MaxViT, and SE-10-SwinTransformer which are all at 0.7303. Among the snapshot ensemble methods, the ones based on VGG16, Resnet50,

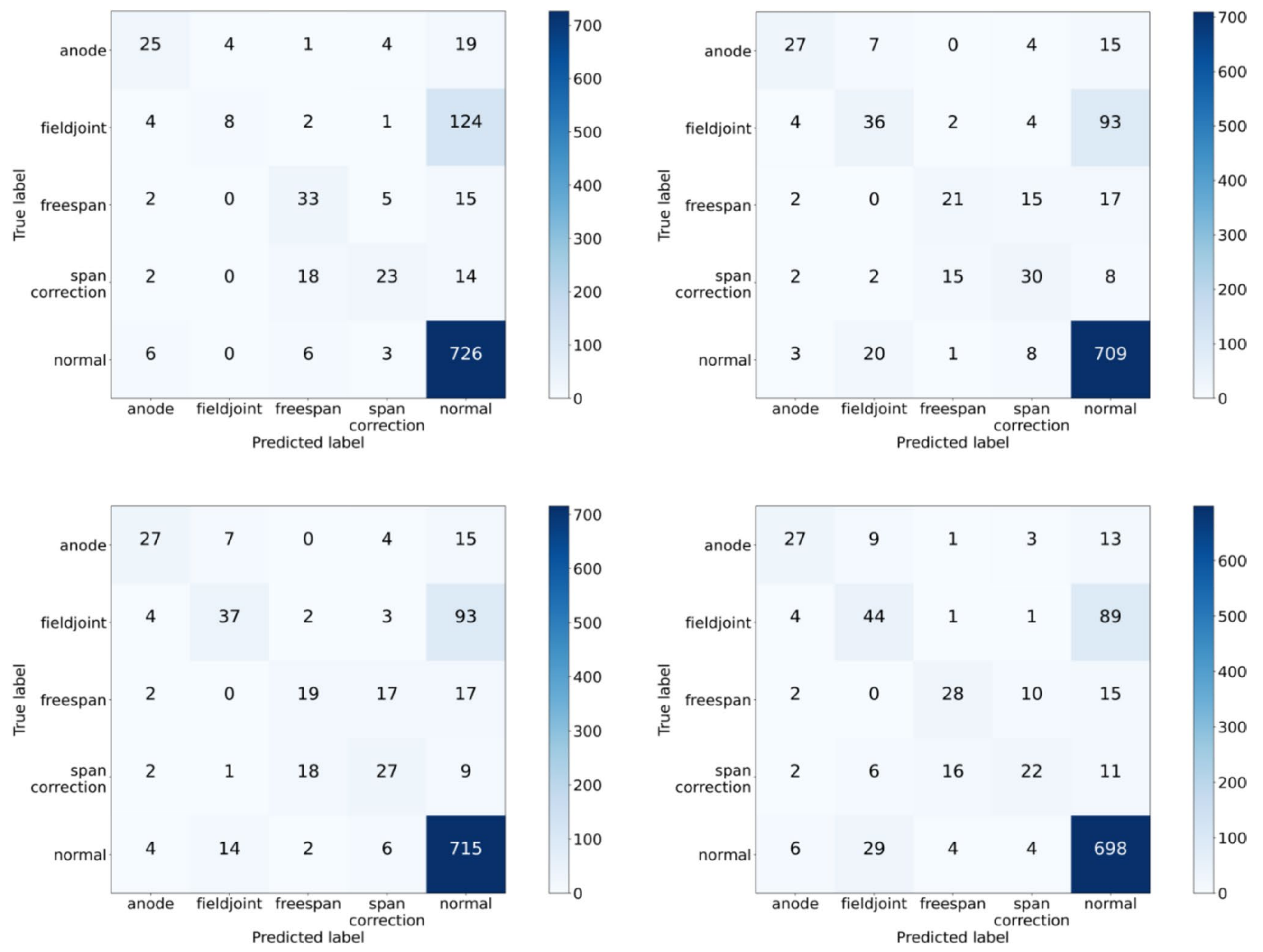


Fig. 5 Confusion matrix of sum rule, SE-5-MaxViT, SE-10-MaxViT and the proposed ensemble on Dataset B

and InceptionV3 obtained poor results. Moreover, using 10 snapshots generally provided better results compared to using 5 snapshots, with 5 exceptions of VGG16 (0.3933 in both cases), InceptionV3 (0.4382 vs. 0.4270), DenseNet121 (0.6854 vs. 0.6517), MaxViT (0.7416 vs. 0.7303) and SwinTransformer (0.7303 in both cases). For Dataset B, SE-10-MaxViT obtained the best accuracy at 0.7895 followed by SE-10-DenseNet121 at 0.7885, SE-5-MaxViT and SE-10-Resnet50 at 0.7876, and the proposed ensemble and SE-5-Resnet50 at 0.7837. Among the other methods, SE-5-VGG16 and SE-10-VGG16 obtained the lowest value at 0.7091 while the other models obtained accuracies from around 0.71 to 0.78. On Dataset C, the best position is shared between the proposed ensemble, SE-10-MaxViT, SE-10-DenseNet121, SE-10-InceptionResNetV2, and SE-10-InceptionV3 at 0.9974. On the other hand, VGG16 obtained the worst result for both cases at 0.7422 and 0.7500, and the accuracies of the other methods mostly range from 0.97 to 0.99.

Table 6 shows the F1 score of the proposed method, the sum rule, and the Snapshot ensemble methods. Overall, the proposed ensemble achieves the best results on all three datasets. On Dataset A, the proposed ensemble and Sum rule achieved the best and second-best accuracy values at 0.7765 and 0.7658 respectively. Concerning the Snapshot ensemble methods, SE-5-VGG16, SE-5-InceptionV3, SE-10-VGG16, and SE-10-InceptionV3 obtained very low scores, from around 0.22 to around 0.38, while the F1 scores of the other methods are around 0.6 to 0.75. The best snapshot ensemble methods are those based on MaxViT and SwinTransformer in both cases, which have F1 scores from around 0.72 to slightly over 0.74. For Dataset B, although

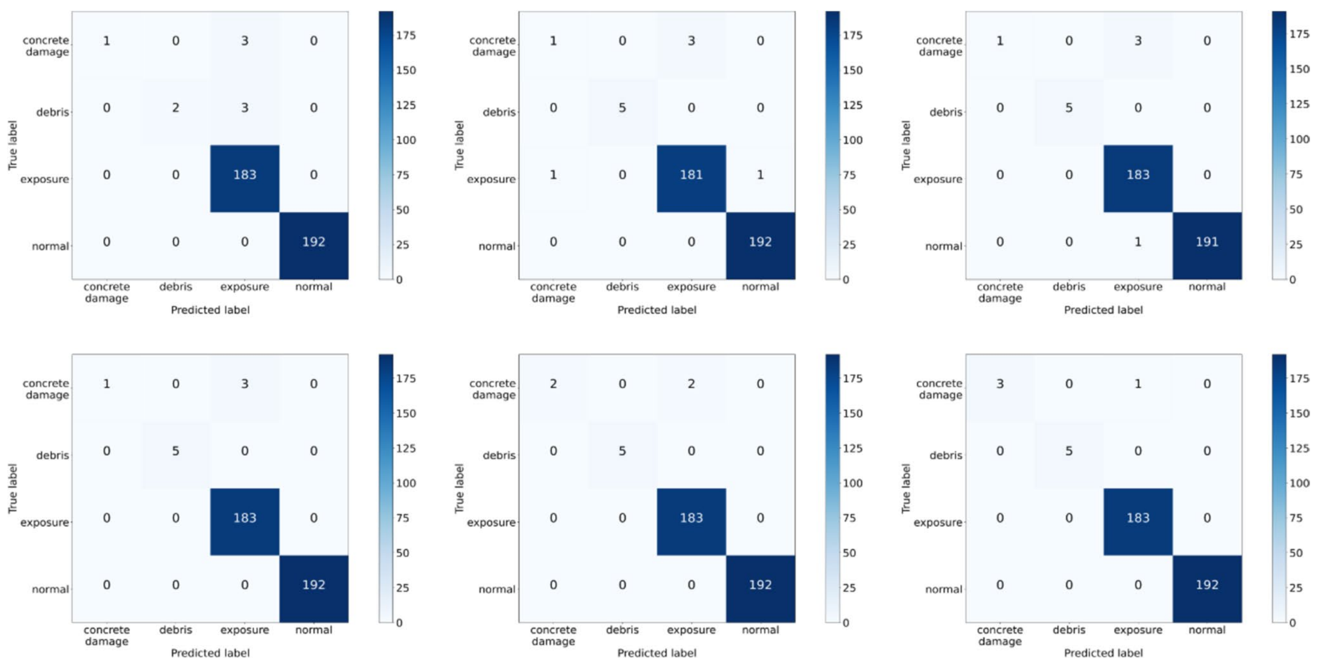


Fig. 6 Confusion matrix of SE-5-InceptionV3, SE-5-SwinTransformer, SE-5-DenseNet121, sum rule, SE-10-XCeption and the proposed ensemble on Dataset C

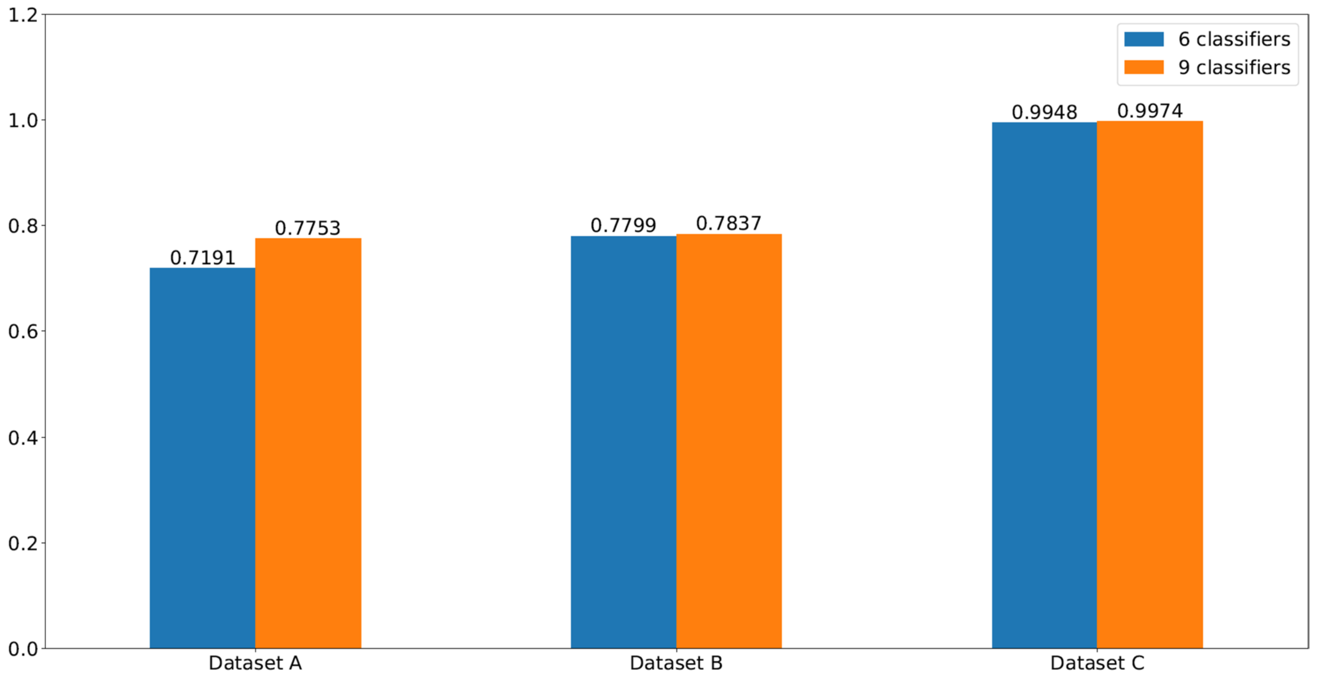


Fig. 7 Accuracy of the proposed ensemble when 6 and 9 classifiers were used

the accuracy of the proposed ensemble was slightly lower than SE-10-MaxViT, SE-10-DenseNet121, SE-5-MaxViT, and SE-10-Resnet50 (as shown in Table 5), its F1 score is better than those of these methods (0.5680 vs. 0.5459, 0.4904, 0.5595 and 0.4930). We note that compared to Dataset A, Dataset B, and Dataset C are much more imbalanced. Specifically, while the number of instances for each class in Dataset A ranges from 100 to 260, in Dataset B the normal

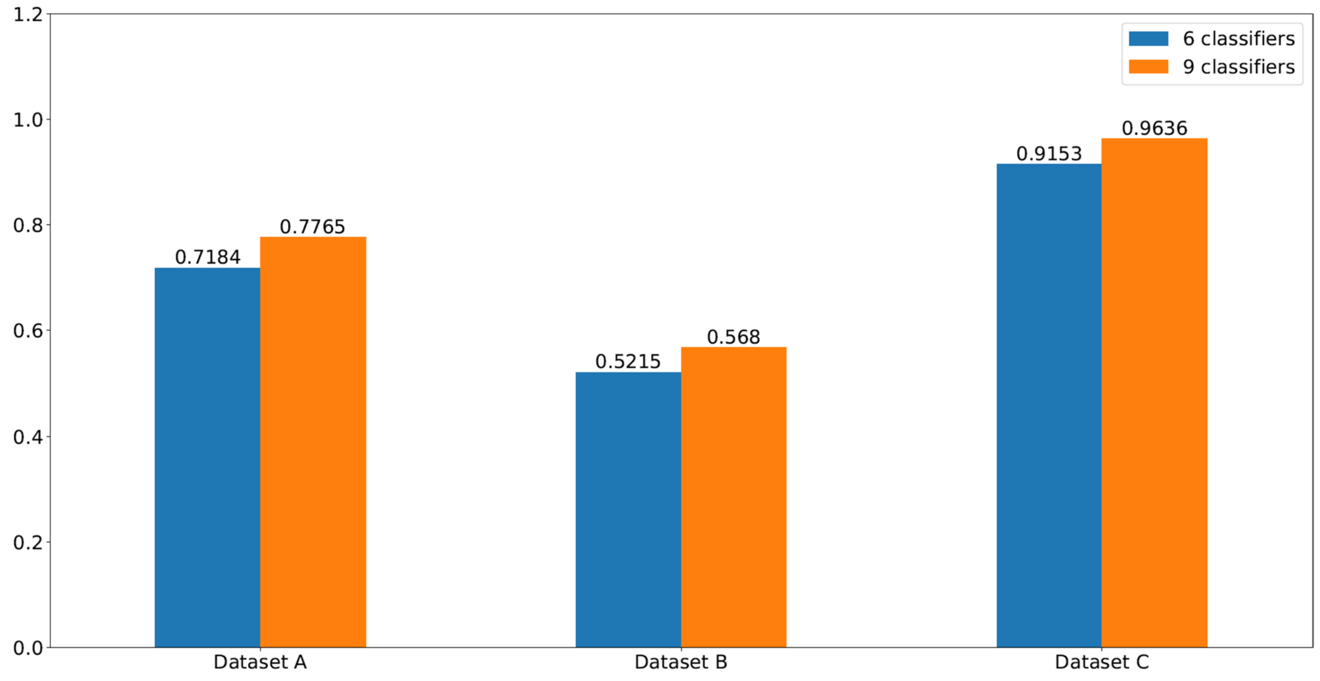


Fig. 8 F1 score of the proposed ensemble when 6 and 9 classifiers were used

class has 3702 instances while each other class has only around 250 to 700 instances. The same can be said for Dataset C, in which there are more than 900 images in the normal and exposure classes, but only around 15–25 for the remaining classes. It is known that the F1 score is a better metric compared to accuracy on imbalanced datasets, therefore this shows that our proposed ensemble is better compared to the benchmark ensemble methods. The F1 score of the proposed ensemble is also 4% better than that of the sum rule (0.5680 vs. 0.5208). Concerning the other methods, VGG16 performed the poorest for both 5 and 10 snapshots, while for other models such as XCeption, the F1 score decreases when using 10 snapshots instead of 5 (reducing from 0.4497 to 0.4343). Among the non-transformer models, VGG16, InceptionV3, and InceptionResNetV2 achieved F1-score lower than 0.4 for both cases when 5 and 10 snapshots were used. On Dataset C, the proposed ensemble, SE-10-MaxViT, SE-10-DenseNet121, SE-10-InceptionResNetV2, and SE-10-InceptionV3 obtain the best F1 score at 0.9636, while the sum rule only achieves a score of 0.848. VGG16 once again had the lowest scores among all methods.

Figure 4 shows the confusion matrix of the sum rule, SE-5-MaxViT, SE-10-MaxViT, and the proposed ensemble on Dataset A (from top to bottom, left to right). On this dataset, the proposed ensemble and sum rule are among the best methods, followed by SE-5-MaxViT and SE-10-MaxViT. All four models did not make any mistakes on anode images, and only SE-10-MaxViT mistakenly classified one normal image to be of the anode class. For the freespan class, both the proposed ensemble and sum rule correctly predicted 21 images, while the number for both SE-5-MaxViT and SE-10-MaxViT was 18.

Table 7 The weights of 6 classifiers on Dataset B in the proposed ensemble

Class Classifier	Anode	Fieldjoint	Freespan	Span correction	Normal
VGG16	0.0895	0.0584	0.1003	0.1721	0.0297
Resnet50	0.2812	0.1488	0.2033	0.0711	0.1367
InceptionV3	0.1612	0.3778	0.0337	0.2212	0.2514
InceptionResNetV2	0.3155	0.1808	0.1677	0.2432	0.2618
XCeption	0.0181	0.2498	0.0000	0.0570	0.0642
SwinTransformer	0.0575	0.0633	0.5417	0.2968	0.2725

Table 8 The weights of 9 classifiers on Dataset B in the proposed ensemble

Class Classifier	Anode	Fieldjoint	Freespan	Span correction	Normal
VGG16	0.0963	0.0233	0.0914	0.1869	0.0000
Resnet50	0.1528	0.0800	0.2022	0.0398	0.0701
InceptionV3	0.1240	0.2338	0.0153	0.1198	0.1624
InceptionResNetV2	0.1764	0.0687	0.1190	0.1495	0.1117
DenseNet121	0.0503	0.1712	0.0600	0.0444	0.0847
XCeption	0.0000	0.0146	0.0000	0.0000	0.0000
VisionTransformer	0.0000	0.0000	0.0000	0.0310	0.0000
MaxViT	0.3130	0.5429	0.1045	0.2725	0.4574
SwinTransformer	0.0000	0.0000	0.4274	0.2120	0.1115

Concerning the debris class, while the proposed ensemble only misclassified 4 debris images as exposure, the sum rule made the same mistakes while wrongly misclassified two additional images as normal. On the other hand, while SE-10-MaxViT has the same errors as the proposed ensemble on this class, SE-5-MaxViT has one less error. The total number of correct predictions for all classes by the proposed ensemble is 69, which is the same as the sum rule, and this number is higher than those of SE-5-MaxViT (66) and SE-10-MaxViT (57). Both the proposed ensemble and sum rule did not misclassify any exposure and freespan images as debris, while both SE-5-MaxViT and SE-10-MaxViT made one error in each category. All four methods misclassified at least 6 normal images as exposure and 6 as freespan, although the sum rule and the proposed ensemble misclassified another normal image as freespan.

Figure 5 shows the confusion matrix of the sum rule, SE-5-MaxViT, SE-10-MaxViT, and the proposed ensemble on Dataset B (from top to bottom, left to right). On this dataset, even though SE-10-MaxViT obtains the highest accuracy, the proposed ensemble still manages to achieve the best F1 score. Compared to the sum rule, which is wrongly predicted as the normal class for 124 fieldjoint images, the proposed ensemble only misclassifies 89 images of this category. The proposed ensemble also misclassified a total of 128 abnormal images as normal, which is less than the sum rule by 44 images. However, while the sum rule correctly classifies 726 normal images, the proposed ensemble only has correct predictions for 698 normal images and misclassifies 29 normal images as fieldjoint. Both SE-5-MaxViT and SE-10-MaxViT correctly classified around 36 fieldjoint images, which is higher than that of the sum rule by more than four times. However, it is still smaller than that of the proposed ensemble (44 images). SE-5-MaxViT misclassifies 15 freespan images as span correction, and 17 as normal, which is higher than those of the proposed ensemble by 5 and 2 images respectively. On the other hand, the number of freespan images being misclassified as anode and fieldjoint is the same for SE-5-MaxViT, SE-10-MaxViT, and the proposed ensemble (at 2 and 0 images respectively). The number of images correctly classified by the proposed ensemble for the fieldjoint and freespan class is 44 and 28 respectively, which is higher than those of SE-5-MaxViT (36 and 21) and SE-10-MaxViT (37 and 19). However, 43 normal images are wrongly classified by the proposed ensemble as abnormal, while for SE-5-MaxViT and SE-10-MaxViT the numbers for this type of error are just 32 and 26 respectively.

Figure 6 shows the confusion matrix of SE-5-InceptionV3, SE-5-SwinTransformer, SE-5-DenseNet121, sum rule, SE-10-XCeption, and the proposed ensemble on Dataset C (from top to bottom, left to right). Methods like the proposed method, SE-10-MaxViT, SE-10-InceptionV3, SE-10-InceptionResNetV2, and SE-10-DenseNet121 performed similarly on this dataset and their confusion matrices are similar and thus we only show the confusion matrix of the proposed method. It can be seen that the proposed ensemble made the fewest number of misclassifications among all four models. SE-5-InceptionV3 made errors with 3 concrete damage images and 3 debris images being misclassified as exposure, while the proposed ensemble only misclassified one image, which is of the concrete damage class, to be exposure and did not make wrong predictions on any debris image at all. Concerning SE-5-SwinTransformer, this model made mistakes on 3 images of concrete damage as exposure and misclassified one exposure image as concrete damage and another one as normal. Instead of misclassifying exposure images, SE-5-DenseNet121 wrongly classified one normal image as exposure. Sum rule, SE-10-XCeption and the proposed ensemble all misclassified 3, 2, and 1 concrete damage images as exposure respectively.

With respect to training and testing time, on Dataset A, each base classification model takes roughly 15 min to train, and the meta-data generation process took approximately 13 h. On Dataset B, each base classification model takes roughly 130 min to train, and the meta-data generation process took approximately 120 h. On Dataset C, each base classification model takes roughly 83.58 min to train, and the meta-data generation process took approximately 75 h. The least square calculation on each dataset took approximately 0.5, 1.5, and 1.2 min. On Dataset A, Dataset B, and Dataset C the snapshot ensemble using 10 snapshots for each classifier took approximately 55 min, 670 min, and 420 min respectively. For the testing time, the proposed ensemble took 10 s on Dataset A, 51 s on Dataset B, and 23 s to run on Dataset C. All experiments were run on a laptop with a GeForce RTX 370 GPU with 8 GB of VRAM.

Influence of Using Different Numbers of Base Classifiers

In this section, we investigated the influence of the number of base classifiers on the performance of the proposed ensemble. Figure 7 and Fig. 8 show the accuracy and F1 score when 6 and 9 classifiers were used (The 6 classifiers are: VGG16, Resnet50, InceptionV3, InceptionResNetV2, XCeption, and SwinTransformer). It can be seen that concerning accuracy, using 9 classifiers achieved significantly better results on Dataset A and achieved slightly better results on Dataset B and Dataset C. For Dataset A, the proposed ensemble (9) obtained an accuracy of 0.7753, which is higher than the proposed ensemble (6) by 5.62%. For Dataset B and C, the proposed ensemble (9) achieved slightly higher accuracy compared to the proposed ensemble (6) (0.7837 vs. 0.7799 on Dataset B, 0.9974 vs. 0.9948 on Dataset C). Concerning the F1 score, it can be seen that on all three datasets, using 9 classifiers improved the results compared to using just 6 classifiers. On Dataset A, the proposed ensemble achieved an F1 score of 0.7765, which is higher than the case when 6 classifiers were used by 5.81%. For Dataset B, the F1 score of the proposed ensemble (9) is 0.568 while the proposed ensemble (6) only obtained a score of 0.5215. Similarly, the proposed ensemble (9) achieved an F1 score of 0.9636 which is higher than the proposed ensemble (6) by a margin of 4.83% on Dataset C. The outstanding performance of the proposed ensemble (9) compared to the proposed ensemble (6) indicates the importance of selecting base classifiers in the ensemble.

Combining Weights of Base Classifiers

Table 7 and Table 8 show the combining weights of base classifiers in the ensemble of 6 and 9 classifiers on Dataset B (the weights of classifiers on Dataset A and C can be found in the Online Resources). Each column represents the weights concerning a class, while each row represents the weights for each classifier (the ordering of the classes and the classifiers have been explained in the previous sections).

Concerning the proposed ensemble (6), the highest weight among those of the 6 classifiers associated with each class is quite low, usually below 0.35. SwinTransformer has the highest weights on Freespan (0.5417), Span correction (0.2968), and Normal class (0.2725). While InceptionResNetV2 has the highest weight on the Anode class (0.3155) and InceptionV3 has the highest weight on the fieldjoint class (0.3778). These three classifiers were the best-performing classifiers on Dataset B and thus their weights are higher than those of the other classifiers. In contrast, the weights of the low-performing classifiers, such as VGG16, are very low, with many of the weights having values below 0.1. For example, VGG16 had three weights, XCeption had four weights, and Resnet50 and InceptionV3 each had one weight below 0.1. It is noted that even for well-performing classifiers such as SwinTransformer, there are also low weights, such as SwinTransformer's weight for the fieldjoint class (0.0633).

Concerning the proposed ensemble (9), it can be seen that there are many weights whose value is zero compared to the proposed ensemble (6) (11 vs. 1). Most of the zero weights are concentrated in the 6th and 7th rows, which are weights of XCeption and VisionTransformer, two of the less well-performing methods on Dataset B. SwinTransformer in this case meanwhile also has two zero weights. Overall, the weights of the low-performing classifiers are much lower compared to the case when 6 classifiers are used. For example, InceptionV3's average weight is 0.2091 when 6 classifiers are used and only 0.1311 when 9 classifiers are used, while InceptionResNetV2's average weight is 0.1251, which is only half of the average weight value when 6 classifiers are used in the ensemble. The classifiers with the highest weights are MaxViT on 4 classes (0.3130 on Anode, 0.5429 on Fieldjoint, 0.2725 on Span correction, and 0.4574 on Normal class). SwinTransformer meanwhile has the highest weight on the freespan class (0.4274), although the average weight of SwinTransformer is smaller compared to the average weight when six classifiers are used. The highest weight among all classifiers and all classes is 0.5429, which is from MaxViT for the fieldjoint class, which is nearly similar to the highest weight in the case where 6 classifiers are used.

The results of the proposed ensemble on Dataset C are more than 99% accuracy and 96% F1 score, which are competitive compared with human-level accuracies for the subsea inspection task. Automated inspection by using the proposed ensemble also takes a small amount of time to detect anomalous events in the data, e.g., it took about 23 s to classify 384 images on the test dataset of Dataset C. Manual inspection meanwhile undergoes several time-consuming steps including spending hours to watch inspection videos and deciding events in video frames which may subject to human errors. Therefore, by applying the proposed ensemble, the entire inspection process can be automated, resulting in much greater efficiency while maintaining a high level of accuracy.

On the other hand, the results on Dataset A and Dataset B are only around 78% of accuracies, which is much lower than that of Dataset C. A main reason for this discrepancy is possible because the images of Dataset C are of very high quality while Dataset A and B have poorer-quality images (e.g., images in Dataset B are grayscale and quite blurry). It is noted that the originated inspection videos of Dataset A and B also were recorded from a higher distance compared to those of Dataset C, making them blurrier. Moreover, there are only 4 classes in Dataset C while there are 5 classes in Dataset A and Dataset B which may make the classification problem on Dataset A and B more challenging. The results of these datasets highlight a significant challenge in the design of efficient automated subsea pipeline inspection systems in which data quality is an important determining factor of the accuracy of these systems. In future works, we plan to systematically investigate the effects of data quality on the performance of automated subsea inspection systems, which will provide important information for industrial deployments of these systems. Besides, we noted that Dataset A, B, and C in this paper are experimented independently because of their different characteristics and set of events. Data fusion techniques in the future will be applied to integrate multiple data sources into a single one with more consistent and useful information than those provided by any individuals to obtain better classification results.

Conclusion

It is recognised that subsea inspection normally is conducted on noisy and low-quality data arising from low-brightness collection conditions and significant backscatter because of the presence of suspended particles. That causes challenges in designing automated solutions to the problem of subsea pipeline inspection. In this paper, an ensemble of deep learning classifiers was proposed to further improve the performance of deep learning-based subsea pipeline inspection systems. Three datasets, provided by three oil and gas companies in the UK, were used in the experiments. These datasets including a number of images, which are called Dataset A, B, and C for anonymity reasons, are recorded under different conditions and contain several types of anomalies. We used nine popular deep learning models namely VGG16, Resnet50, InceptionV3, Inception-ResNetV2, DenseNet121, XCeption, VisionTransformer, MaxViT, and SwinTransformer to train base classifiers for the proposed ensemble. We proposed to combine the outputs of the base classifiers by using a weighted combining method in which each classifier is associated with a weight vector of M weights for a $M-$ class classification problem. To find the combining weights, we first divided the training dataset into T disjoint parts. For each part, we trained classifiers on its complementary and predicted images in that part to obtain the predictions in the form of probabilities. By running through all the divided parts, we obtained the probability predictions for all images in the training set. Based on these probabilities and the given ground truth annotation information, the optimal weights are found by solving a number of optimisation problems. The performance of the proposed ensemble was compared with the base classifiers, snapshot ensemble, and sum rule. We also investigated the influence of using different numbers of classifiers on the ensemble predictions. Experimental results indicate that the proposed ensemble achieves better results in general compared to the baseline models in different settings. Some future works were suggested concerning the effect of image quality on the performance of the classification methods and the applications of data fusion techniques to the problem of subsea pipeline inspection. We also plan to extend the proposed ensemble to other types of subsea inspection activities, such as inspection of the base area of offshore wind turbines, and inspection of any damages resulting from human activity to the local seabed.

Appendix

Table 9 Notation table

Notation	Explanation
$\mathbf{D}$	The training set
$\mathbf{I}_n$	The n^{th} image in the training set
$\mathbf{Y}_n$	The ground truth for the n^{th} image in the training set
$\mathcal{Y} = \{y_m\}, m = 1, \dots M$	The list of M classes
$\mathbf{h} : \mathbf{I}_n \rightarrow \mathbf{Y}_n$	A hypothesis that approximates the unknown relationship between images and their corresponding ground truths
$\{\mathcal{L}_k\}_{k=1}^{K}$	The set of K classification algorithm
$\{\mathbf{h}_k\}_{k=1}^{K}$	The set of K trained classifiers
$\mathbf{C}$	The combining model
β_1	The decay rate of the exponential moving average of the gradient in the Adam optimization algorithm
β_2	The decay rate of the exponential moving average of the squared gradient in the Adam optimization algorithm
m_t	The first-moment estimate at timestep t
v_t	The second-moment estimate at timestep t
g_t	The gradient at timestep t
$E[g_t^2]$	The true value of the second moment at timestep t
ζ	The error term
$y_{n,m}$	The binary variable indicates whether the n^{th} observation is in the m^{th} class
$p_{n,m}$	The probability the n^{th} observation is predicted to be in the m^{th} class
$Loss$	The loss function
T	The number of cross-validation folds
$\mathbf{D}_t$	The t th fold of the training set
$\mathbf{h}_{k,t}$	The k th classifier trained on the remainder of the t th fold of the training set
$\mathbf{I}$	An image
$P_k(y_m \vert \mathbf{I})$	The probability prediction that $\mathbf{h}_{k,t}$ assigns to image $\mathbf{I}$ to be in class y_m
$\mathbf{L}(\mathbf{I})$	The predictions of K classifiers $\{\mathbf{h}_{k,t}\}$ showing the probabilities that image $\mathbf{I}$ belongs to all M classes
$\mathbf{L}$	The meta-data of $\mathbf{D}$
$\mathbb{W}$	The weight matrix
$w_{k,m}$	The weight associated with the classifier $\mathbf{h}_k$ concerning the class $y_m(k = 1, \dots, K; m = 1, .., M)$
$pred_m(\mathbf{I}_n)$	The ensemble prediction with respect to the m th class for $\mathbf{I}_n$
$\mathbf{L}_{\text{m}}$	The probabilities associated with class y_m
$\mathbb{I}[.]$	The indicator function
$\mathbb{Y}_m$	The crisp label vector (i.e., belonging to $\{0,1\}$) of size $(N, 1)$ associated with class y_m
$\mathbb{W}_m$	The weight vector associated with the class y_m
$\mathbf{I}_{test}$	The test image
$P_k(y_m \vert \mathbf{I}_{test})$	The probability prediction that $\mathbf{h}_{k,t}$ assigns to test image $\mathbf{I}_{test}$ to be in class y_m
$\hat{w}_{k,m}$	The optimal weight associated with the classifier $\mathbf{h}_k$ concerning the class $y_m(k = 1, \dots, K; m = 1, .., M)$
$\widehat{\mathbb{W}}$	The optimal weight matrix
$\widehat{\mathbb{W}}_m$	The optimal weight vector associated with the class y_m
$\hat{m}$	The index of the predicted class by the ensemble

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12559-024-10377-y.

Acknowledgements This work was supported by the Scottish Government through The RSE Scotland Asia Partnerships Higher Education Research (SAPHIRE) Fund [grant number 2970].

Author Contribution Truong Dang: Methodology, Writing- Original draft preparation, Experimental Study. Tien Thanh Nguyen: Conceptualization, Methodology, Writing- Original draft preparation, Supervision. Alan Wee-Chung Liew: Methodology, Supervision, Writing-Reviewing and Editing. Eyad Elyan: Conceptualization, Validation, Writing- Reviewing and Editing.

Funding This study was funded by the Scottish Government through The RSE Scotland Asia Partnerships Higher Education Research (SAPHIRE) Fund [grant number 2970].

Tien Thanh Nguyen reports financial support was provided by the Royal Society of Edinburgh.

Data Availability Data cannot be shared openly but are available on request from authors The data that support the findings of this study are available from NetZero Technology Centre, but restrictions apply to the availability of these data, which were used under licence for the current study and so are not publicly available. The data are, however, available from the authors upon reasonable request and with the permission of NetZero Technology Centre.

Declarations

Ethical Approval This article does not contain any studies with human participants performed by any of the authors.

Conflict of Interest The authors declare no competing interests.

References

1. 'Global crude oil onshore and offshore production distribution 2025', Statista. [Online]. Available: https://www.statista.com/statistics/624138/distribution-of-crude-oil-production-worldwide-onshore-and-offshore/
2. Eastvedt D, Naterer G, Duan X. Detection of faults in subsea pipelines by flow monitoring with regression supervised machine learning. Process Saf Environ Prot. 2022;161:409–20.
3. Cai B, et al. Remaining useful life estimation of structure systems under the influence of multiple causes: subsea pipelines as a case study. IEEE Trans Industr Electron. 2020;67(7):5737–47.
4. Davis P, Brockhurst J. Subsea pipeline infrastructure monitoring: A framework for technology review and selection. Ocean Eng. 2015;104:540–8.
5. Zingaretti P, Zanoli SM. Robust real-time detection of an underwater pipeline. Eng Appl Artif Intell. 1998;11(2):257–68.
6. Xia P, You H, Du J. Visual-haptic feedback for ROV subsea navigation control. Autom Constr. 2023;154:104987.
7. Liu Y, Bao Y. Review on automated condition assessment of pipelines with machine learning. Adv Eng Inform. 2022;53:101687.
8. El-Abbasy MS, Senouci A, Zayed T, Mirahadi F, Parvizsedghy L. Artificial neural network models for predicting condition of offshore oil and gas pipelines. Autom Constr. 2014;45:50–65.
9. Chen H, Ye H, LV C, Su H. 'Application of support vector machine learning to leak detection and location in pipelines.' In Proceedings of the 21st IEEE Instrumentation and Measurement Technology Conference, 2004. pp. 2273–2277.
10 Stamoulakatos A, et al. Automatic annotation of subsea pipelines using deep learning. Sensors. 2020;20(3):674.
11. Li X, Guo M, Zhang R, Chen G. A data-driven prediction model for maximum pitting corrosion depth of subsea oil pipelines using SSA-LSTM approach. Ocean Eng. 2022;261:112062.
12. Zhang M, Guo Y, Xie Q, Zhang Y, Wang D, Chen J. Estimation of defect size and cross-sectional profile for the oil and gas pipeline using visual deep transfer learning neural network. IEEE Trans Instrum Meas. 2023;72:1–13.
13. Chen J, Cao L, Song G. Detection of the pipeline elbow erosion by percussion and deep learning. Mech Syst Signal Process. 2023;200:110546.
14. Zhang Z, Xu C, Xie J, Zhang Y, Liu P, Liu Z. MFCC-LSTM framework for leak detection and leak size identification in gas-liquid two-phase flow pipelines based on acoustic emission. Measurement. 2023;219:113238.
15. Feng J, Li F, Lu S, Liu J, Ma D. Injurious or noninjurious defect identification from mfl images in pipeline inspection using convolutional neural network. IEEE Trans Instrum Meas. 2017;66(7):1883–92.
16. Wang C, Wang Z, Liu W, Shen Y, Dong H. A novel deep offline-to-online transfer learning framework for pipeline leakage detection with small samples. IEEE Trans Instrum Meas. 2023;72:1–13.
17. Dang T, Nguyen T, Mccall J, Elyan E, Moreno-García C. Two-layer ensemble of deep learning models for medical image segmentation. Cogn Comput. 2024;16:1–20.
18 Baldeon Calisto M, Lai-Yuen SK. AdaEn-Net: An ensemble of adaptive 2D–3D fully convolutional networks for medical image segmentation. Neural Netw. 2020;126:76–94.
19. He K, Zhang X, Ren S, Sun J. 'Deep Residual Learning for Image Recognition.'in IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016;770–778.
20. Alvear-Sandoval RF, Figueiras-Vidal AR. On building ensembles of stacked denoising auto-encoding classifiers and their further improvement. Information Fusion. 2018;39:41–52.
21. Lakshminarayanan B, Pritzel A, Blundell C. 'Simple and scalable predictive uncertainty estimation using deep ensembles.' In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS). 2017;6405–6416.
22. Jain DK, Zhao X, Garcia S, Neelakandan S. Robust multi-modal pedestrian detection using deep convolutional neural network with ensemble learning model. Expert Syst Appl. 2024;249:123527.
23. Peláez-Rodríguez C, et al. Deep learning ensembles for accurate fog-related low-visibility events forecasting. Neurocomputing. 2023;549:126435.
24. Huang G, Li Y, Pleiss G, Liu Z, Hopcroft JE, Weinberger KQ. 'Snapshot Ensembles: Train 1, get M for free.' arXiv:1704.00109 [cs]. 2017.
25. Bai Q, Bai Y. '14 - Pipeline Spans and VIV Fatigue.' in Subsea Pipeline Design, Analysis, and Installation. 2014;337–363.
26. Simonyan K, Zisserman A. 'Very Deep Convolutional Networks for Large-Scale Image Recognition.' arXiv 1409.1556 [cs]. 2014.

27. Vaswani A, et al. 'Attention is All you Need.' In Advances in Neural Information Processing Systems. 2017;6000–6010.
28. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. 'rethinking the inception architecture for computer vision.' In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016;2818–2826.
29. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. Proc Thirty-First AAAI Conf Artif Intell. 2017;31(1):4278–84.
30. Huang G, Liu Z, Maaten LVD, Weinberger KQ. 'densely connected convolutional networks.' In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017 Jul;2261–2269.
31. Chollet F. 'Xception: Deep learning with depthwise separable convolutions.' in IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017;1800–1807.
32. Dosovitskiy A, et al. 'An Image is Worth 16x16 Words: Transformers for image recognition at scale.' In International Conference on Learning Representations (ICLR). 2021.
33. Tu Z, et al. 'MaxViT: Multi-axis Vision Transformer.' In European Conference on Computer Vision (ECCV); 2022. 459–479.
34. Liu Z, et al. 'Swin transformer: hierarchical vision transformer using shifted windows.' In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2021.
35. Zhuang F, et al. 'A comprehensive survey on transfer learning.' arXiv: 1911.02685 [cs]; 2019.
36. Krizhevsky A, Sutskever I, Hinton GE. 'ImageNet classification with deep convolutional neural networks.' In Advances in Neural Information Processing Systems; 2012.
37. Kingma D, Ba J. 'Adam: a method for stochastic optimization.' International Conference on Learning Representations (ICLR); 2014.
38. Dang T, Nguyen TT, McCall J, Liew AW-C. 'Ensemble learning based on classifier prediction confidence and comprehensive learning particle swarm optimisation for medical image segmentation.' in 2022 IEEE Symposium Series on Computational Intelligence (SSCI); 2022. pp. 269–276.
39. Nguyen TT, Pham NV, Dang T, Luong AV, McCall J, Liew AW-C. 'Multi-layer heterogeneous ensemble with classifier and feature selection.' In Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO); 2020. pp. 725–733.
40. Nguyen TT, Dang MT, Liew AW, Bezdek JC. A weighted multiple classifier framework based on random projection. Inf Sci. 2019;490:36–58.
41. Branch MA, Coleman TF, Li Y. A Subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. SIAM J Sci Comput. 1999;21(1):1–23.
42. Chen T, Kornblith S, Norouzi M, Hinton G. 'A simple framework for contrastive learning of visual representations.' In Proceedings of the 37th International Conference on Machine Learning, in ICML'20. JMLR.org, 2020.
43. Tan M, Le Q. 'EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.' In: Proceedings of the 36th International Conference on Machine Learning, Chaudhuri K, Salakhutdinov R, editors. In Proceedings of Machine Learning Research. PMLR; 2019 Jun. vol. 97, pp. 6105–6114.
44. Han K, Pham T, Vu TH, Dang T, McCall J, Nguyen TT. 'VEGAS: a variable length-based genetic algorithm for ensemble selection in deep ensemble learning.' Springer; 2021. pp. 168–180.
45. Kuhn M, Johnson K. Applied Predictive Modeling. Springer, 2013.

Supplementary materials

Event classification on subsea pipeline inspection data using an ensemble of deep learning-based classifiers

Truong Dang[1], Tien Thanh Nguyen[1] *, Alan Wee-Chung Liew[2], Eyad Elyan[1]
[1]School of Computing, Robert Gordon University, Aberdeen, UK
[2]School of Information and Communication Technology, Griffith University, Gold Coast, Australia

* Corresponding author: Tien Thanh Nguyen
Email: t.nguyen11@rgu.ac.uk

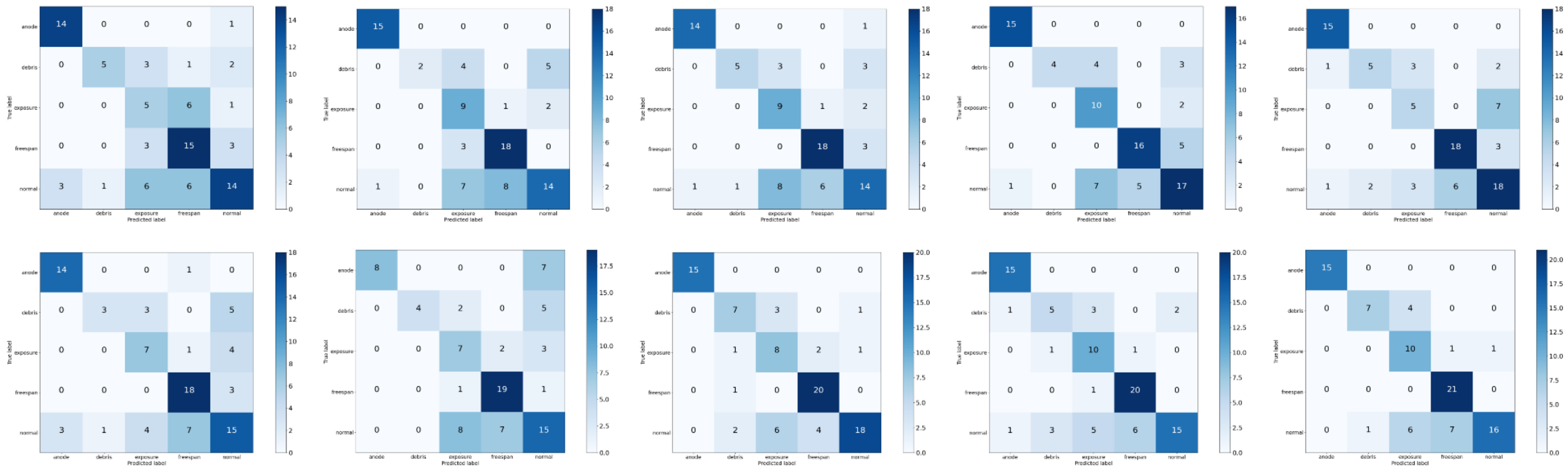
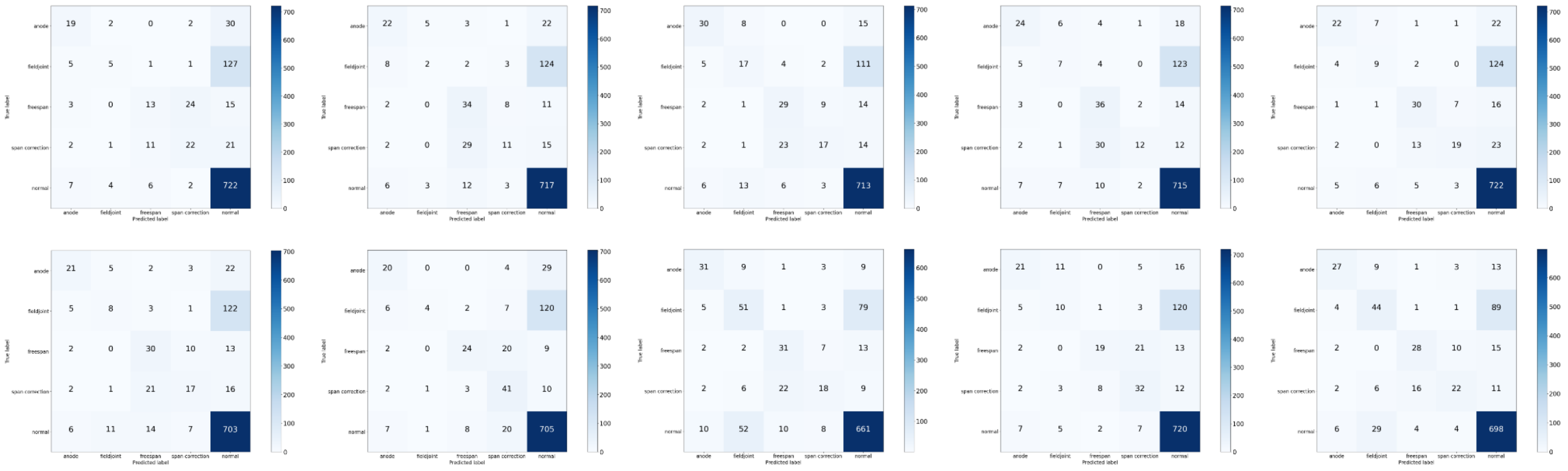


Figure S1 - Confusion matrix of the base classifiers and the proposed ensemble on Dataset A



	anode	fieldjoint	freespan	span correction	normal
anode	19	2	0	2	30
fieldjoint	5	5	1	1	127
freespan	3	0	13	24	15
span correction	2	1	11	22	21
normal	7	4	6	2	722

	anode	fieldjoint	freespan	span correction	normal
anode	22	5	3	1	22
fieldjoint	8	2	2	3	124
freespan	2	0	34	8	11
span correction	2	0	29	11	15
normal	6	3	12	3	717

	anode	fieldjoint	freespan	span correction	normal
anode	30	8	0	0	15
fieldjoint	5	17	4	2	111
freespan	2	1	29	9	14
span correction	2	1	23	17	14
normal	6	13	6	3	713

	anode	fieldjoint	freespan	span correction	normal
anode	24	6	4	1	18
fieldjoint	5	7	4	0	123
freespan	3	0	36	2	14
span correction	2	1	30	12	12
normal	7	7	10	2	715

	anode	fieldjoint	freespan	span correction	normal
anode	22	7	1	1	22
fieldjoint	4	9	2	0	124
freespan	1	1	30	7	16
span correction	2	0	13	19	23
normal	5	6	5	3	722

	anode	fieldjoint	freespan	span correction	normal
anode	21	5	2	3	22
fieldjoint	5	8	3	1	122
freespan	2	0	30	10	13
span correction	2	1	21	17	16
normal	6	11	14	7	703

	anode	fieldjoint	freespan	span correction	normal
anode	20	0	0	4	29
fieldjoint	6	4	2	7	120
freespan	2	0	24	20	9
span correction	2	1	3	41	10
normal	7	1	8	20	705

	anode	fieldjoint	freespan	span correction	normal
anode	31	9	1	3	9
fieldjoint	5	51	1	3	79
freespan	2	2	31	7	13
span correction	2	6	22	18	9
normal	10	52	10	8	661

	anode	fieldjoint	freespan	span correction	normal
anode	21	11	0	5	16
fieldjoint	5	10	1	3	120
freespan	2	0	19	21	13
span correction	2	3	8	32	12
normal	7	5	2	7	720

	anode	fieldjoint	freespan	span correction	normal
anode	27	9	1	3	13
fieldjoint	4	44	1	1	89
freespan	2	0	28	10	15
span correction	2	6	16	22	11
normal	6	29	4	4	698

Figure S2 - Confusion matrix of the base classifiers and the proposed ensemble on Dataset B

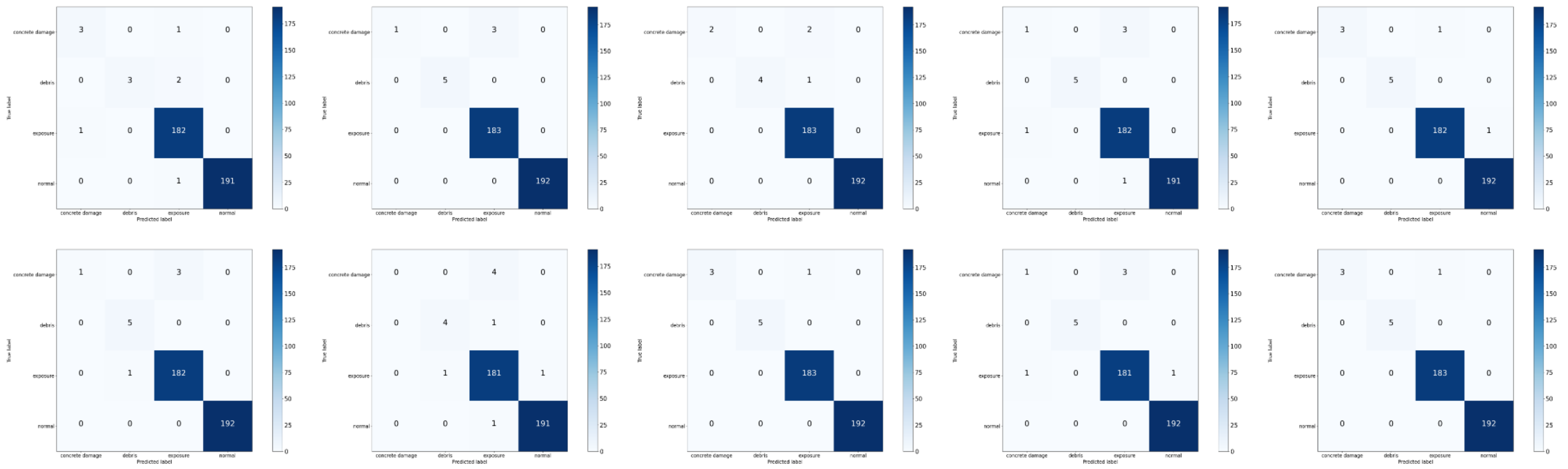


Figure S3 - Confusion matrix of the base classifiers and the proposed ensemble on Dataset C

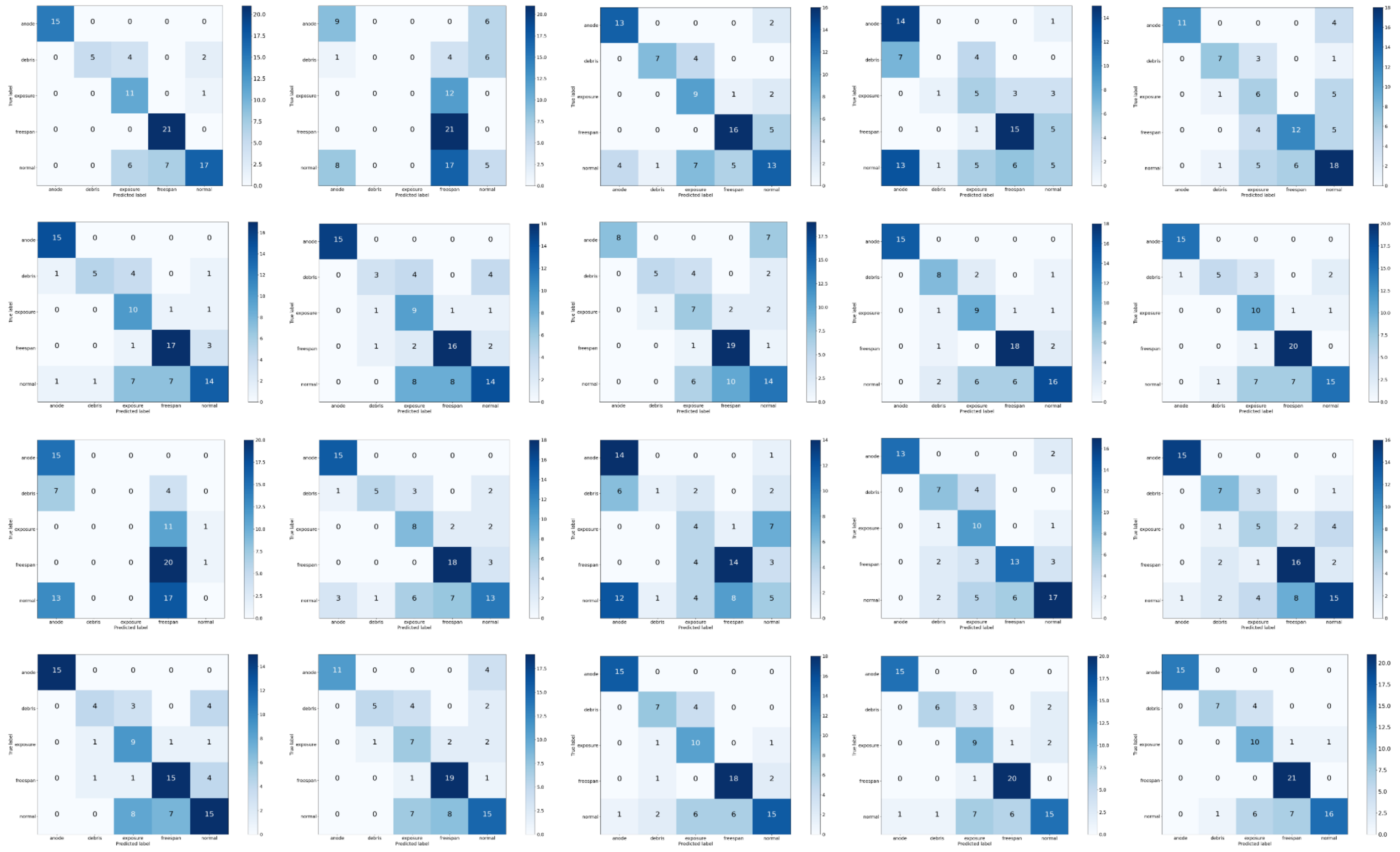


Figure S4 - Confusion matrix of sum rule, the snapshot ensemble models and the proposed ensemble on Dataset A

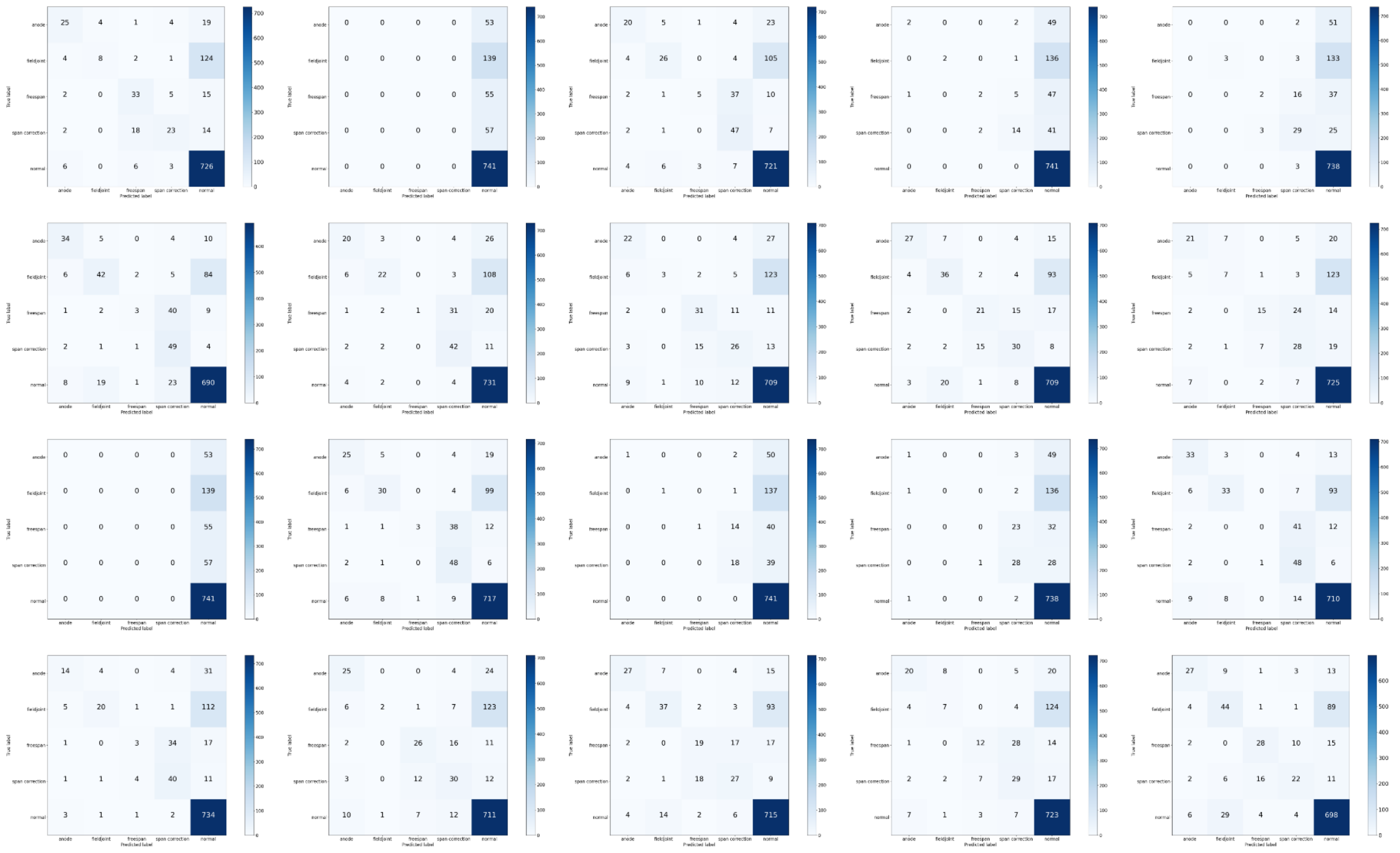


Figure S5 - Confusion matrix of sum rule, the snapshot ensemble models and the proposed ensemble on Dataset B

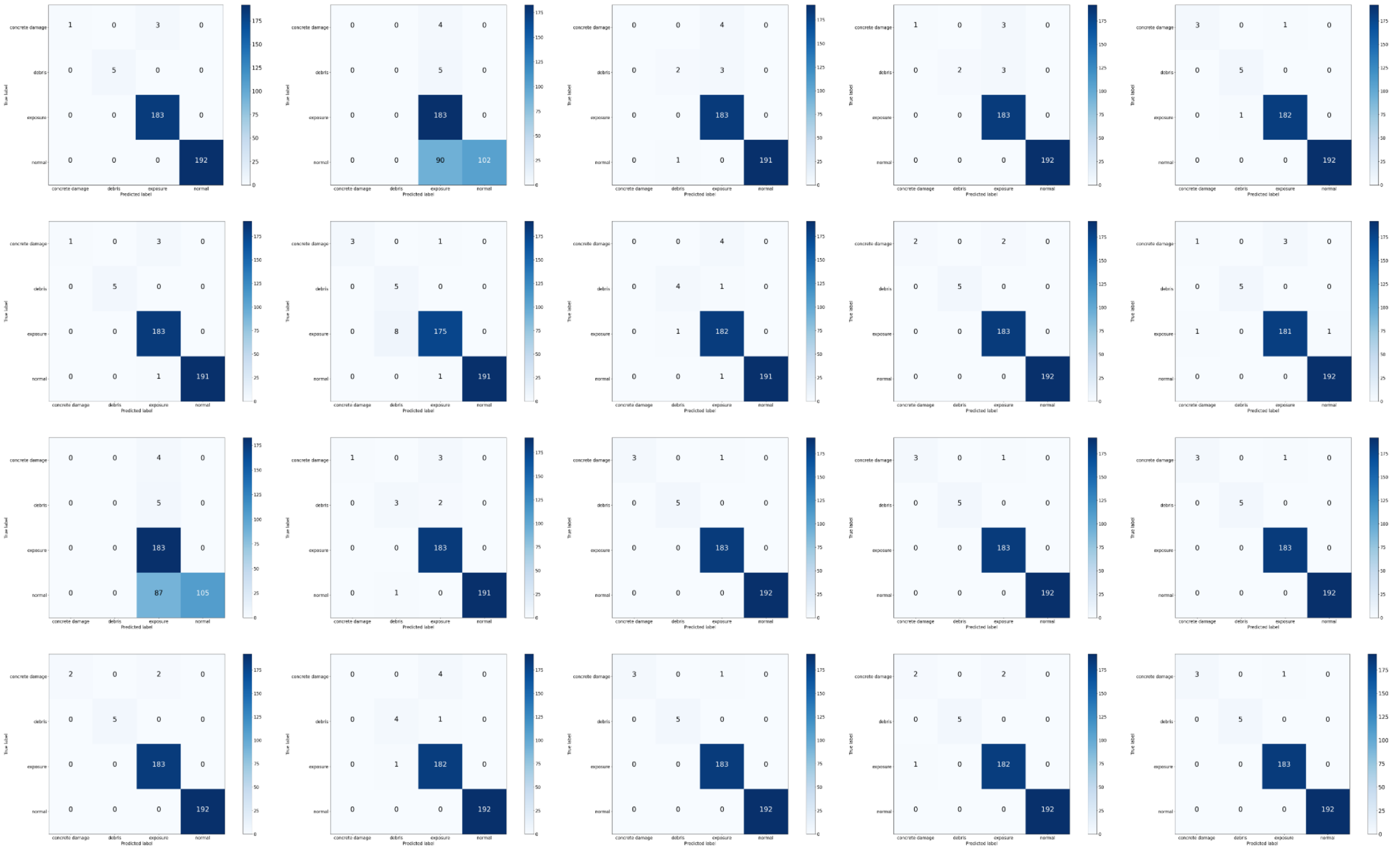


Figure S6 - Confusion matrix of sum rule, the snapshot ensemble models and the proposed ensemble on Dataset C

Table S1 -The weights of 6 classifiers on Dataset A in the proposed ensemble

	Anode	Debris	Exposure	Freespan	Normal
VGG16	0.0577	0.1617	0.0000	0.2283	0.1092
Resnet50	0.1919	0.1607	0.2820	0.3636	0.3027
InceptionV3	0.2570	0.0145	0.0621	0.0752	0.2358
InceptionResNetV2	0.1840	0.3125	0.0000	0.0000	0.0299
XCeption	0.3242	0.2440	0.1747	0.1474	0.1240
SwinTransformer	0.0623	0.2921	0.4924	0.1631	0.2450

Table S2 - The weights of 9 classifiers on Dataset A in the proposed ensemble

	Anode	Debris	Exposure	Freespan	Normal
VGG16	0.0000	0.0925	0.0000	0.1287	0.0474
Resnet50	0.0491	0.0000	0.0798	0.2573	0.1698
InceptionV3	0.1197	0.0000	0.0000	0.0173	0.1084
InceptionResNetV2	0.0696	0.2049	0.0000	0.0000	0.0000
DenseNet121	0.2208	0.2322	0.2067	0.0502	0.3912
XCeption	0.1703	0.0666	0.0609	0.0713	0.0136
VisionTransformer	0.0000	0.0000	0.1460	0.1281	0.0384
MaxViT	0.4207	0.3851	0.3215	0.2793	0.2577
SwinTransformer	0.0000	0.1508	0.2282	0.0000	0.0000

Table S3 - The weights of 6 classifiers on Dataset C in the proposed ensemble

	Concrete damage	Debris	Exposure	Normal
VGG16	0.1705	0.0000	0.0000	0.0111
Resnet50	0.8179	0.5146	0.5715	0.1994
InceptionV3	0.0882	0.0548	0.0204	0.5758
InceptionResNetV2	0.3140	0.2804	0.2039	0.0172
XCeption	0.0000	0.0801	0.0000	0.0643
SwinTransformer	0.1700	0.3166	0.2027	0.1332

Table S4 - The weights of 6 classifiers on Dataset C in the proposed ensemble

	Concrete damage	Debris	Exposure	Normal
VGG16	0.0000	0.0000	0.0000	0.0091
Resnet50	0.0565	0.1191	0.1077	0.0027
InceptionV3	0.0000	0.1419	0.0000	0.1696
InceptionResNetV2	0.0000	0.0000	0.0000	0.0115
DenseNet121	0.2492	0.0000	0.0547	0.6021
XCeption	0.1001	0.0000	0.0000	0.0092
VisionTransformer	0.0000	0.0321	0.0000	0.0000
MaxViT	0.8562	0.8735	0.8344	0.1670
SwinTransformer	0.0000	0.0000	0.0000	0.0296