Continual-learning-based framework for structural damage recognition

Jiawei Zhang[1], Jiangpeng Shu[1], Reachsak Ly[1], Yiran Ji[1]
[1]College of Civil Engineering and Architecture, Zhejiang University, Hangzhou, China
email: author1@fe.up.pt, author2@fe.up.pt, lyreachsak@zju.edu.cn

ABSTRACT: Multi-damage is common in reinforced concrete structures and leads to the requirement of large number of neural networks, parameters and data storage, if convolutional neural network (CNN) is used for damage recognition. In addition, conventional CNN experiences catastrophic forgetting and training inefficiency as the number of tasks increases during continual learning, leading to large accuracy decrease of previous learned tasks. To address these problems, this study proposes a continual-learning-based damage recognition model (CLDRM) which integrates the "learning without forgetting" continual learning method into the ResNet-34 architecture for the recognition of damages in RC structures as well as relevant structural components. Three experiments for four recognition tasks were designed to validate the feasibility and effectiveness of the CLDRM framework. In this way, it reduces both the prediction time and data storage by about 75% in four tasks of continuous learning. Three experiments for four recognition tasks were designed to validate the feasibility and effectiveness of the CLDRM framework. By gradual feature fusion, CLDRM outperformed other methods by managed to achieve high accuracy in the damage recognition and classification. As the number of recognition tasks increased, CLDRM also experienced smaller decrease of the previous learned tasks. Results indicate that the CLDRM framework successfully performs damage recognition and classification with reasonable accuracy and effectiveness.

KEY WORDS: Damage recognition and classification, Concrete Structures, Continual-learning-based damage recognition model (CLDRM), ResNet, Learning without forgetting.

1 INTRODUCTION

1.1 Background

Reinforced concrete (RC) civil structures gradually approach their design life expectancy, posing a risk to the safety structures and people [1]. Therefore, it is necessary to effectively inspect damages of RC structures. Conventionally, different manual inspections methods were used. However, these methods have the disadvantages of unreliable inspection results and considerable time consumption [2]. Owing to the rapid development of machine learning, which has shown excellent performance in image processing and computer vision in recent years, researchers and engineers are increasingly applying it for damage detection and structural assessment [3][4]. One of the well-known applications of deep learning in civil engineering is crack detection and recognition. The types of damage of RC structures are various. Thus, the demand for the recognition and classification of damage types is also on the rise. Moreover, it is becoming easier to retrieve more structural-damage-related information in the form of image data based on the technical development of deep learning and data acquisition. For this reason, numerous solutions have been explored. A region-based CNN (Faster- RCNN) [5] was adopted to detect different damage types such as concrete cracks and steel corrosion with two levels (medium and high), including bolt corrosion and steel delamination [6]. In addition, it was utilized to detect spalling and severe damage for exposed and buckled rebars[7]. YOLO [8], which is another region-based CNN, was used to detect multiple concrete bridge damage types such as pop-out or exposed rebars [9]. Based on ImageNet [10] Gao and Mosalam (2018) proposed structural ImageNet with four baseline recognition functions, i.e., component type determination, spalling condition check, damage level evaluation, and damage type determination[11]. They employed transfer learning (TL) to prevent overfitting, along with two strategies, namely, feature extraction [12] and fine-tuning [13]. Gao and Mosalam [14] proposed a large-scale multi-attribute benchmark dataset of structural Images, namely, the Pacific Earthquake Engineering Research (PEER) Hub ImageNet (ϕ-Net).Based on this dataset , Gao and Mosalam [15] introduced a Generative Adversarial Network model, namely, Deep Convolutional Generative Adversarial Network (DCGAN) and propose a Leaf-Bootstrapping (LB) method to improve the performance of this DCGAN.

1.2 Multitask Damage Recognition

It is worth noting that with the continuous development of deep learning, the number of recognition tasks in damage detection is increasing. One of the common problems among the abovementioned applications is the limitation of the number of recognition tasks for training. These training methods are time/resource consuming and do not utilize the feature correlation between similar tasks such as crack detection and spalling detection. To address this issue, one neural network can be trained simultaneously for multiple recognition tasks. The trained model can identify similar features and characteristics with improved accuracy [16].

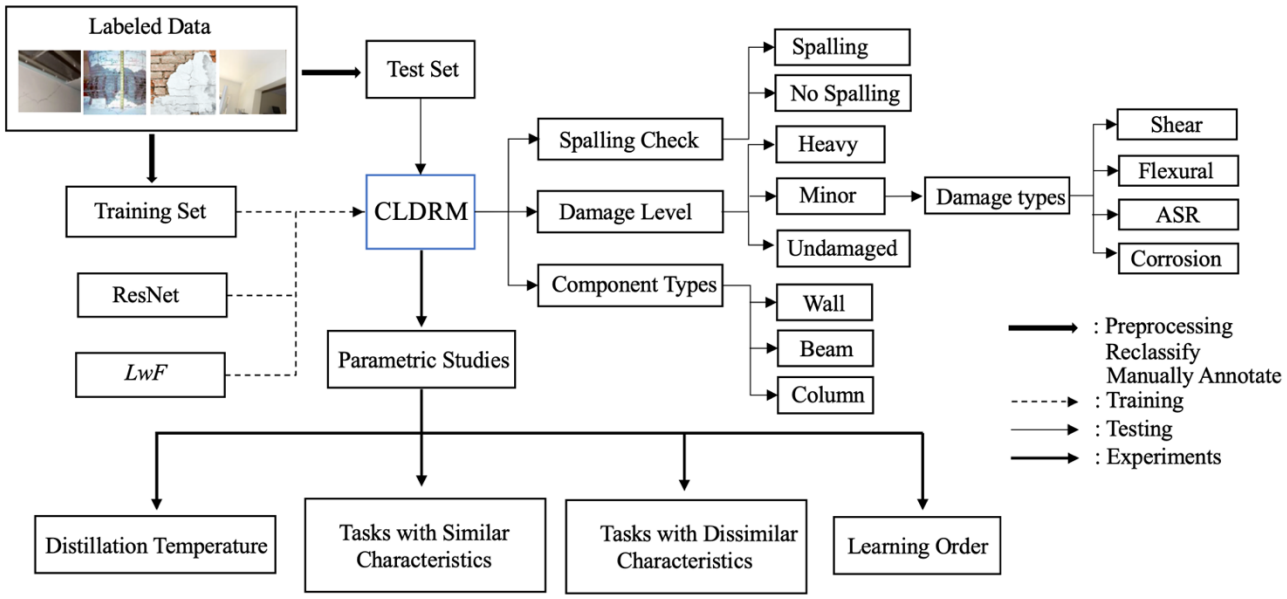


Figure 1. Flowchart of continual-learning-based damage recognition model (CLDRM) framework.

1.3 Continual Learning

Continual learning has been proposed as a revolutionary method [17] for two purposes: (1) to perform multitask recognition using one CNN; (2) to maintain an optimal recognition accuracy for previously trained tasks with a low training cost. Continual learning is a generic method, which can be adopted in any CNN framework. Continual learning utilizes the features and knowledge learned from the recognition of previous tasks, facilitates training for next tasks, and maintains the features learned from previous tasks. This is achieved by limiting changes in parameters, i.e., adding a regularization term.

To overcome the aforementioned limitations of previous methods, this study proposes a novel continual-learning-based damage recognition model (CLDRM), which can recognize multiple damage types of RC structures with the continual learning ability. The detailed objectives of this study are as follows: (1) To establish a database of multitask recognition, including damage level evaluation, spalling condition check, component type determination, and damage type determination, with approximately twenty thousand images. (2) To build a new deep CNN training framework, namely, the CLDRM, which utilizes the L*w*F method with the residual network (ResNet)[18] architecture for the detection and classification of crack damage in RC structures. (3) To study the effect of different parameters, such as correlation of features as well as learning orders, on the results of recognition based on proposed CLDRM.

2 OVERVIEW OF THE PROPOSED METHOD

Figure 1 shows the flowchart of the proposed framework, training steps, and testing steps. A comprehensive dataset that includes typical damage categories is prepared for training. A labeled training dataset is the input. A separate test dataset is prepared to verify the model and to investigate its performance on four different recognition tasks, which are (1) three-class classification for damage level evaluation, (2) binary classification for spalling condition check, (3) three-class classification for component type determination, and (4) four-class classification for damage type determination. Each task is defined in detail in Section 3. This study utilizes a collected large dataset of damaged RC image including PEER Hub ImageNet dataset[11]. Overall, 16,840 colorful images with a resolution of 224×224 are prepared, preprocessed, reclassified, and manually annotated for the aforementioned recognition tasks.

The CLDRM adopts the continual-learning-based "learning without forgetting (L*w*F)" method[16], which is a form of knowledge distillation[19]. Instead of using a previous image dataset for retraining, the knowledge learned from previous tasks is retained by learning the soft targets provided by the previous model, which is stored after training. In other words, the previous model can be saved and stored for subsequent training without any image datasets of a previous task. This significantly reduces memory requirement and training complexity. A new deep CNN training framework, namely, the CLDRM, which utilizes the L*w*F method with the residual network (ResNet)[18] architecture for the detection and classification of crack damage in RC structures, is developed.

This research's content is described as follows. Section 2 presents the synopsis of the proposed method. Section 3 provides the details of the four recognition tasks and their corresponding datasets. Section 4 explains the overall architecture of the networks and the proposed continual learning-based model training methodologies as well as the hyperparameters, dataset and settings used to train the model. Section 5 demonstrates how the framework evaluates test images from the proposed experiments and includes discussions regarding the method's performance and potential. Section 6 concludes this article and overviews some future possible studies to be conducted in this research area.

3 RECOGNITION TASKS AND DATASET

3.1 Damage level evaluation

To achieve a clearer boundary between the levels of damage, this study categorizes crack damage into three main categories, i.e., "undamaged," "minor damage," and "heavy damage," which are defined as follows: Minor damage does not pose a risk and does not need to be repaired urgently. Heavy damage can pose a risk and must be repaired urgently.

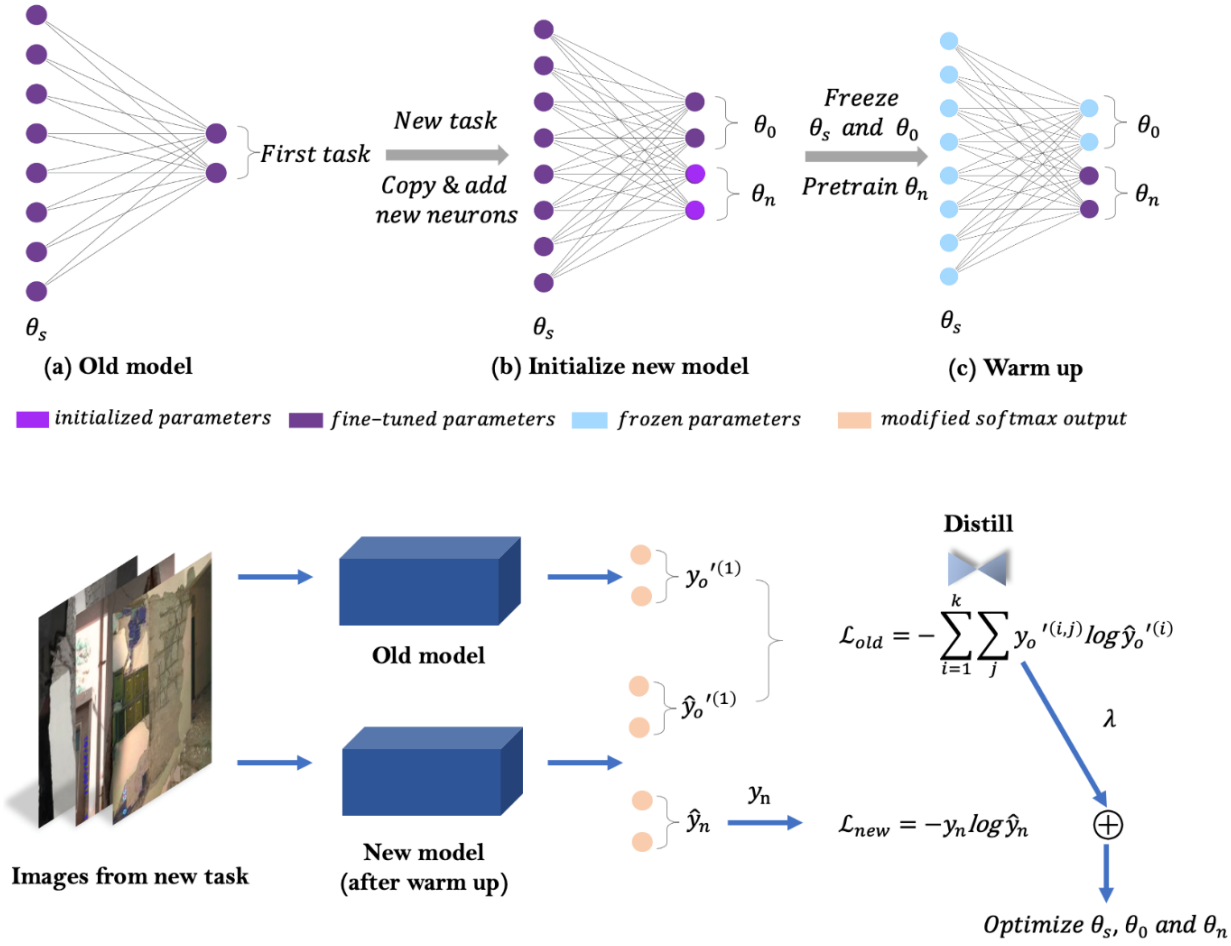


Figure 2. CLDRM training framework.

3.2 *Spalling condition check*

Spalling condition is a binary classification task with two classes: spalling and no spalling. The spalling condition pertains to the loss of cover of the component's surface.

3.3 *Component type determination*

Component type determination is a three-class classification task for "columns," "walls (or slabs)," and "beams" .

3.4 *Damage type determination*

Four typical crack damage types are selected based on the causes of cracks. All of them have evident characteristics, which are suitable for identification. The proportion of images for each crack type is shown in Figure 2.

1) Shear cracks: An extremely large shear stress causes cracks close to supports such as walls or columns. These cracks are typically inclined at 45°, and they have X or V shapes in a few cases.
2) Flexural cracks: Flexural damage is caused by overloaded bending stress. It mostly occurs in the horizontal or vertical direction or at the end of a component with a horizontal or vertical edge.
3) Cracks caused by Alkali–Silica reaction (ASR): These cracks [20] occur over time in concrete between the highly alkaline cement paste and non-crystalline silicon dioxide found in common aggregates.
4) Cracks caused by corrosion of reinforcing steel: Corrosion generally refers to stress corrosion cracking, which is induced by the combined influence of tensile stress and corrosive environments.

4 NETWORK ARCHITECTURE AND MODEL

4.1 *Network Architecture of ResNet*

ResNet [18] is well known for its remarkable object detection and image classification capabilities. It has been demonstrated that residual mapping and shortcut connections facilitate the training process and lead to better results compared to very deep plain networks. Network depth is crucial for improving the network performance of CNNs; however, deeper networks are more difficult to train. He et al. (2016) [18] reported that an increase in network depth leads to degradation problems, which cause accuracy to become saturated. To address this problem, they used [batch normalization (BN), rectified linear unit (ReLU), and a convolution layer] × 2 as a basic mapping backbone and subsequently added an extra skip connection, i.e., simple identity mapping, to form a complete residual block. This residual block learns the necessary residual mapping for a given task and improves the capability of training considerably deeper networks. This method reduces training time and improves convergence speed, resulting in more accurate prediction. The ResNet architecture has shown impressive performance not only on image classification benchmarks, such as CIFAR and ImageNet, but also on object detection benchmarks such as MS COCO [21]. The CNN

architecture used in this study is the original ResNet-3architecture pretrained on the 1000-class ImageNet dataset to accelerate the convergence of training and maintain a suitable number of model parameters while ensuring a certain feature extraction effect.

4.2 Learning Without Forgetting

The L*w*F method was first developed by Li and Hoiem (2018) [16], and has been adopted in the proposed CLDRM. There are three important notations in L*w*F, i.e., θ_s, which denotes the parameters in the CNN shared across all tasks, θ_o, which denotes the specific parameters learned from previous tasks, and θ_n, which denotes the specific parameters that are learned in a new task.

In this study, ResNet-34 is utilized to implement multitask learning and θ_s (the parameters of layers before the tail of the fully connected layer) is initialized with the pretrained parameters of the 1000-class ImageNet dataset to accelerate the convergence of training. The features of images extracted from the first few layers of a CNN tend to be more generalized [22], while the features extracted from the latter layers are more relevant to a specific dataset. Therefore, researchers tend to freeze the parameters of the first few layers after transferring the parameters trained on the dataset with richer features; only the parameters of the next few layers are fine-tuned[13]. However, fine-tuning all parameters provides better performance in multitask learning. For this reason, θ_s is focused in the training process. In the tail part of the fully connected layer, fc-1000 in the original ResNet is replaced by a new linear layer with 512 input features and m output features, where m is the number of classes for the first task. Then, after the training process of the first task, n, which denotes the number of classes for the next new task, is appended to this fully connected layer. All newly appended to this fully connected layer. All newly appended parameters are initialized using Kaiming initialization [18].

Then, the size of the fully connected layer becomes $512 \times (m+n)$. θ_o and $\theta_n$correspond to parameters $512 \times m$ and $512 \times n$, respectively.

The main strategy of this study is to add θ_n parameters for a new task within the same model and learn the parameters with θ_s and $\theta_o$to improve the recognition accuracy for old and new tasks without using the data from old tasks (see Figure 3).

The loss ($\mathcal{L}_{new}$) during the learning of a new task can be defined as the cross entropy between y_n and $\hat{y}_n$, as expressed in Equation (1). $\hat{y}_n$ denotes the output corresponding to the new tasks of the new model, where y_n is the ground truth.

$$\mathcal{L}_{new} = -y_n \log \hat{y}_n \tag{1}$$

y_o denotes the output (after softmax is applied separately for each old task) of the old model. $\hat{y}_o$ denotes the output corresponding to the old tasks of the new model. It is worth noting that the loss during the training for old tasks cannot be retrieved by directly adding the cross entropy calculated between y_o and $\hat{y}_o$ for each old task; this results in poor prediction performance. Thus, hyperparameter T (Equation (2)), which represents temperature (Hinton, Vinyals, and Dean, 2015), is introduced to amplify the information carried by the small probabilities. The modified probabilities of y_o and $\hat{y}_o$ are given by Equation (2).

$$y_o'^{(i,j)} = \frac{(y_o^{(i,j)})^{1/T}}{\sum_j (y_o^{(i,j)})^{1/T}}, \quad \hat{y}_o'^{(i,j)} = \frac{(\hat{y}_o^{(i,j)})^{1/T}}{\sum_j (\hat{y}_o^{(i,j)})^{1/T}} \tag{2}$$

Superscript i represents the i^{th} old task in the previous training, and it varies between $\mathbf{1}$ to k, which represents the number of the old tasks that have been previously trained. Superscript j represents the j^{th} class in the i^{th} old task. As T is set to be 2, the loss ($\mathcal{L}_{old}$) for an old task can be expressed Superscript i represents the i^{th} old task in the previous training, and it varies between 1 to k, which represents the number of the old tasks that have been previously trained. Superscript j represents the j^{th} class in the i^{th} old task. As T is set to be 2, the loss ($\mathcal{L}_{old}$) for an old task can be expressed as Equation (3).

$$\mathcal{L}_{old} = -\sum_{i=1}^{k} H(y_o'^{(i)}, \hat{y}_o'^{(i)}) = -\sum_{i=1}^{k}\sum_{j} y_o'^{(i,j)} \log \hat{y}_o'^{(i)} \tag{3}$$

Overall, the training process can be concluded as follows:

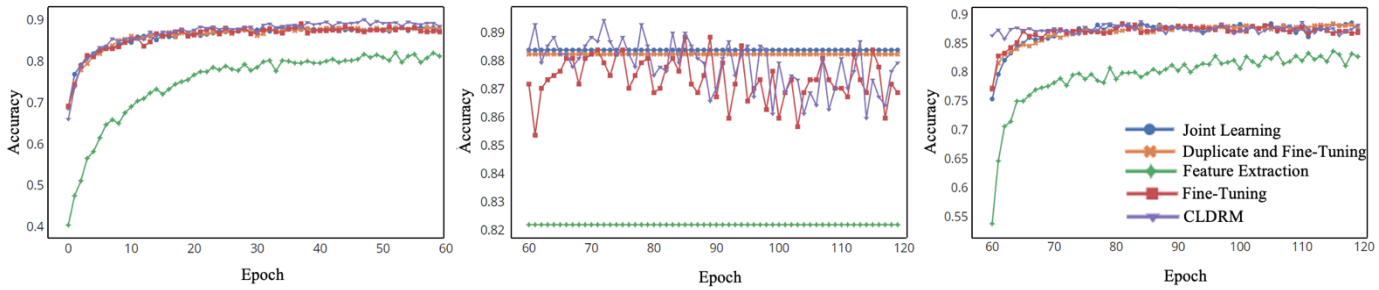
- Step 1: The first task is trained using the typical training process in TL.
- Step 2: When a new task is added, a new model will be created and it inherits the θ_s and θ_o from the old model. The newly appended parameters θ_n will be initialized.
- Step 3: θ_s and θ_o are frozen, and θ_n is pretrained in the dataset of new task to convergence (warm up).
- Step 4: θ_s and $\theta_o$are unfrozen, $\hat{y}_n$ and $\hat{y}_o'^{(i,j)}$ are output from the new model, and $y_o'^{(i,j)}$ is input from the old model using the same batch images of new task.
- Step 5: The total loss $\mathcal{L}_{total} = \mathcal{L}_{new} + \lambda \mathcal{L}_{old}$ is calculated with the ground truth y_n and the outcome of the Step 3, namely, $\hat{y}_n$, $\hat{y}_o'^{(i,j)}$, $y_o'^{(i,j)}$. λ is a hyperparameter that balances the learning tendency between the new task and the old tasks. In this study, λ is set to 1.
- Step 6: θ_s, θ_o and θ_n are simultaneously updated to minimize the total loss.

4.3 Comparative Evaluation

The performance of the proposed CLDRM framework is evaluated through experiments and compared to a conventional CNN model with four existing training methods, which are described below.

4.3.1 Feature Extraction

Feature extraction[12] is used to filter out the most critical and distinguishable information from redundant data. Extracted features are typically represented by feature vectors. In deep learning, the output of a certain layer (the last convolutional layer) can be extracted as the feature representation of input images and then used to classify image in the testing process.



a) Damage level evaluation (0 – 59) b) Damage level evaluation (60 – 119) c) Spalling condition check

Figure 3. Variation in recognition accuracy of the model trained with different methods for similar tasks.

Thus, in the multitask learning investigated in this study, θ_s and θ_o are defined as constants. Additionally, the features extracted from the last layer are used to determine θ_n through training so that training can be accelerated by neglecting backpropagation for θ_s. However, the features extracted from the network are shared by all tasks. Such features are independent of any specific task; this slightly reduces classification accuracy.

4.3.2 Fine-Tuning

To overcome the abovementioned problem of feature extraction, θ_o is fixed and a low learning rate is applied for training θ_s and θ_n (Yosinski et al., 2014). In this manner, the network learns the features related to a new task. However, it may forget the features learned in the first task.

4.3.3 Duplicate and Fine-Tuning

During the training for a new task, the relatively important parameters for the identification of the first task are partially modified. This causes the network to forget the features learned in the previous task. For this reason, the duplicate and fine-tuning method is applied to the network for each new task.

4.3.3 Joint training

Joint training simultaneously trains all parameters (θ_s, θ_o and θ_n) using all data from old and new tasks[23].The network can extract and integrate the characteristics of each task. This may improve the accuracy for a few recognition tasks compared to the individual training of each task. However, storing all data is an issue and extremely expensive in terms of time and computation resources. In order words, whenever a new task is added, it must be trained again with the old tasks that have been previously trained on the network.

4.4 Model Training

This section describes the CLDRM training process, including the training and test sets, optimization methods, and hardware configuration. All tasks are performed on a workstation with the Intel(R) Xeon(R) E5-2678 v3 2.50 GHz CPU, 64.0 GB RAM, and the NVIDIA RTX2080TI-11G GPU.

4.4.1 Train and test set

The number of training and test sets used in the experiments for different recognition tasks are listed in

Table 1.In the data preprocessing part, data augmentation (horizontal flip, vertical flip, rotation, color jitter, etc.) is implemented with a batch size of 64. However, a small proportion of the datasets in the structure component part contain useful environmental background information. Therefore, in most cases, at a certain angle of rotation of images, inclined beams may look like vertical columns and vice versa. Therefore, to prevent this, images are not rotated in the component type determination task.

Table 1. Number of images for the training and test sets for each task.

Task	***Damage Level***	***Spalling***	***Component***	***Damage Type***
Training	3776	4864	3968	1728
Test	663	820	693	328

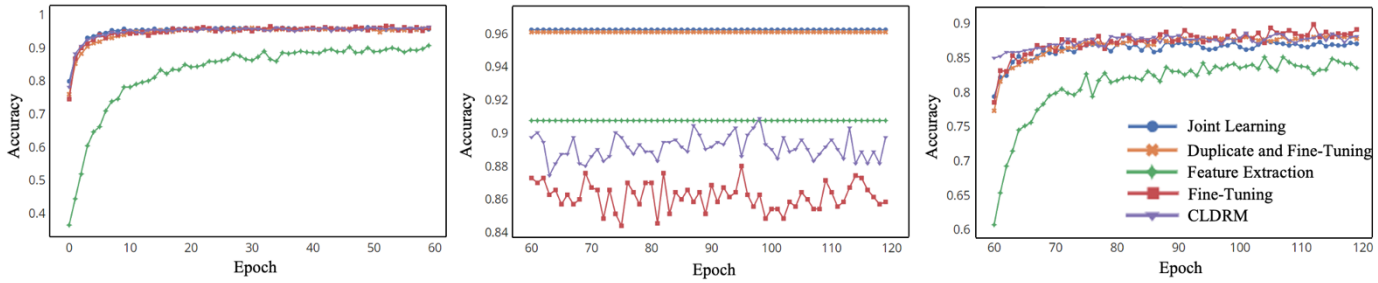
4.4.2 Optimization

The results of the initial experiments for the training of continual learning tasks show that the Adam optimizer [24] is relatively unstable and ineffective, which results in poor model performance. The recognition accuracy for an old task decreases significantly during the training of a new task. Thus, the Adam optimizer is not selected as the training method for the CLDRM. SGD with momentum provides considerably better performance, and it is employed as the optimizer for the CLDRM. The learning rate of the SGD optimizer is 1×10^{-3}, the momentum is 0.9, and the weight decay rate is 4×10^{-5}. These parameter settings are adopted to pretrain θ_n, and the model is trained for 40 epochs. When θ_s, θ_o, and θ_n are simultaneously trained, the learning rate is reduced to 1×10^{-4} and the number of training epochs is increased to 60.

5 EXPERIMENTS AND RESULTS

The feasibility of the CLDRM is experimentally verified for four different recognition tasks, i.e., damage level evaluation, spalling condition check, component type determination, and damage type determination. The objective is to achieve accurate multitask damage recognition through a single neural network. However, typically, the recognition accuracy for a previous task decreases when new tasks are input to the neural network. The target is to minimize the decrease in the recognition accuracy for previous tasks so that the total accuracy is high. The influence of the following parameters on the model is examined: a) influence of characteristic

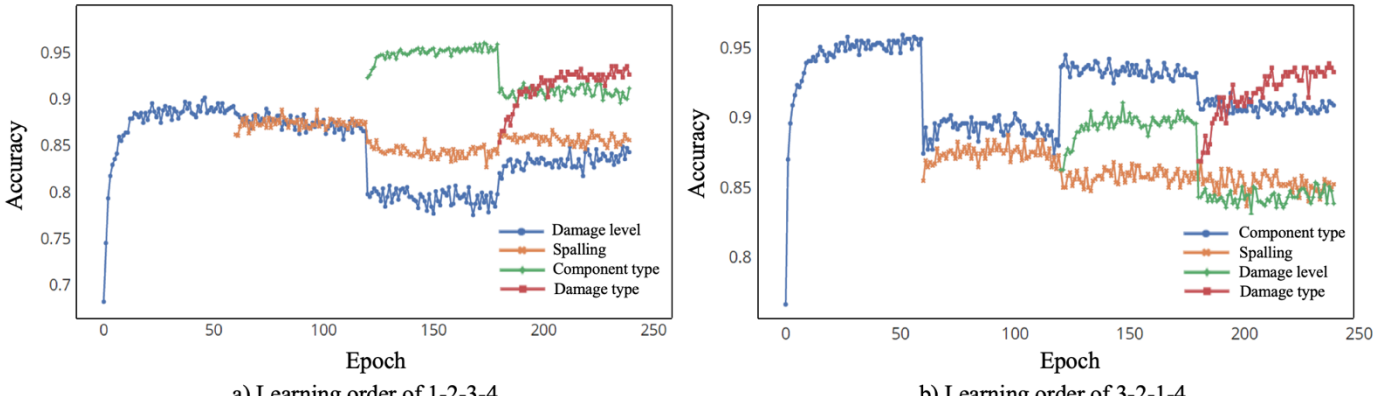
similarity on continual learning; b) influence of learning order



a) Component type determination (0 – 59) b) Component type determination (60 – 119) c) Spalling condition check

Figure 4. Variations in recognition accuracy of the model trained with different methods on dissimilar tasks

on mixed similar and dissimilar tasks. Moreover, based on the



a) Learning order of 1-2-3-4 b) Learning order of 3-2-1-4

Figure 5. Variation in recognition accuracy for different tasks and different learning orders.

results of the experiments, the appropriate distillation temperature improves the effect of knowledge distillation from old tasks. **T** is set as 5 in the subsequent experiments to obtain the best accuracy.

5.1 Tasks with correlated Characteristics

This section describes the impact of the feature correlation between learning tasks on the performance of continual learning. When different tasks share numerous similar or same feature extractors, fine-tuning tends to perform better than joint training. The CLDRM effectively retains the knowledge of a previous task. Moreover, owing to the recurrence of related features, the training of the second task may even improve the recognition accuracy for old tasks in a few cases.

First, tasks with similar characteristics are investigated. The damage level evaluation and spalling condition check tasks are used as the first and second learning tasks, respectively, owing to their characteristic similarities. Figure 4 a) and c) illustrates the variation in the recognition accuracy for the first (0–59 epochs) and second (60–120 epochs) tasks during training, respectively. Figure 4 b) shows the variation in the recognition accuracy for the old task during the training of the new task (60–120 epochs). Besides, the training performed using the joint training method and duplicate and fine-tuning method is only for 60 epochs each. During the training of the new task, θ_s and θ_o are constant for the feature extraction method (Figure 4 a) and b)) The accuracies of the three methods are shown in Figure 4 b) with their largest values in the first 60 epochs. The results show that the overall performance of the CLDRM is better than the fine-tuning method. In addition, the CLDRM significantly outperforms the feature extraction method.

To investigate tasks with dissimilar characteristics, the component type determination and spalling condition check tasks are used as the first and second tasks, respectively, owing to the difference between their characteristics. Figure 5 a) shows the variation in the accuracy for the first task in the first 60 epochs, and Figure 5 c) illustrates the variation in the accuracy for the second task in the last 60 epochs of training. Figure 5 b) demonstrates the effects of the training of the second task on the accuracy for the first task. When the characteristics of different tasks are not similar, the performance of the joint training method is better than the CLDRM, as predicted in Section 5.1. This observation provides useful guidance for the subsequent adjustments of the learning order of the CLDRM; this is discussed in detail in the next section. Moreover, as seen from Figure 5 b) , the CLDRM can retain more knowledge and information of old tasks compared to the fine-tuning method. Moreover, the model with the CLDRM not only retains the important features of old tasks but also increases the recognition accuracy for these tasks during the training of new tasks, which is related to the learning order.

5.2 Learning Order

This section describes the effect of the learning order of different tasks on the recognition accuracy of the CLDRM by mixing the feature-related and feature-unrelated tasks. The purpose of this experiment is to find whether the learning order of these four tasks has a significant impact on the final accuracy for a particular task. The damage type determination task is selected as the last task to be trained, and the learning

order of the other three tasks is arbitrarily changed. The damage level evaluation, spalling condition check, component type determination, and damage type determination tasks are denoted by numbers 1, 2, 3, and 4, respectively. For instance, the learning order of the damage level evaluation, spalling condition check, component type determination, and damage type determination tasks is denoted as *1-2-3-4.*

Table 3. Recognition accuracy of CLDRM for different learning orders.

Task	Damage level		Spalling Condition		Component type		Damage type	Average of
	Initial (%)	Final (%)	Initial (%)	Final (%)	Initial (%)	Final (%)	Final (%)	Final (%)
1-2-3-4	90.20	84.92	88.90	86.71	96.10	92.06	93.60	89.32
1-3-2-4	89.74	84.01	88.78	87.32	96.54	91.34	93.29	88.99
2-1-3-4	90.65	85.67	88.05	85.00	96.25	91.63	93.90	89.05
2-3-1-4	91.10	84.92	88.41	85.37	96.83	91.77	93.29	88.83
3-1-2-4	89.89	84.62	88.17	88.29	96.10	91.20	93.90	89.50
3-2-1-4	91.10	85.37	88.78	86.83	95.96	91.77	93.90	89.46

Table 4. Recognition accuracy of fine-tuning method for different learning orders.

Task	Damage level		Spalling Condition		Component type		Damage type	Average of
	Initial (%)	Final (%)	Initial (%)	Final (%)	Initial (%)	Final (%)	Final (%)	Final (%)
1-2-3-4	90.05	83.41	88.41	84.15	95.67	92.21	92.07	87.96
1-3-2-4	89.59	81.45	88.90	83.22	95.82	91.20	90.55	86.61
2-1-3-4	90.65	85.67	88.05	84.00	96.25	87.63	91.90	87.30
2-3-1-4	91.10	84.92	88.41	83.37	96.83	88.77	91.29	87.08
3-1-2-4	89.89	84.62	88.17	83.29	96.10	86.20	91.90	86.50
3-2-1-4	91.10	84.92	88.17	84.02	95.96	85.28	92.99	86.80

Table 3 shows the recognition accuracy of the CLDRM for all tasks for different learning orders. The CLDRM shows excellent performance in maintaining the knowledge of old tasks. The decrease in the recognition accuracy for old tasks during the training of new tasks is less than 6%. Additionally, the learning order of *3-1-2-4* shows the highest performance among these six learning orders, with an average accuracy of 89.50% for the four tasks.

Figure 6 compares the recognition accuracy for the four tasks between two different learning orders, i.e., *1-2-3-4* and *3-2-1-4*. During the training for the final task (damage type determination), the accuracies for the first task in *1-2-3-4* (damage level evaluation) and *3-2-1-4* (component type determination) increase by almost 4% and 6%, respectively. This occurs under the condition that the same characteristics are adopted for training previous and new tasks. However, in the training for a new task, the accuracy for an old task first decreases slightly and then increases. For instance, in *3-2-1-4*, the accuracy for the first task significantly decreases by 8% when new tasks (spalling condition check) are added.

Table 4 shows the recognition accuracy of the fine-tuning method for different learning orders. The overall recognition accuracy of the fine-tuning method is approximately 1–2% lower than the CLDRM. Regardless of the learning order, the accuracy for the first task learned by the fine-tuning method decreases significantly by an average of 8% after learning the fourth task, as compared to only 5% in the CLDRM.

5.3 Model Evaluation

The feature extraction method requires the least training time but provides mediocre performance for new tasks. The duplicate and fine-tuning method provide good performance on old and new tasks; however; the prediction time of the corresponding model is quite high. The joint training method provides superior performance for old and new tasks but requires a large amount of data storage while training. On the contrary, the CLDRM not only provides high recognition accuracy and speed for old and new tasks but also requires less data storage and parameters during training.

6 CONCLUSION AND FUTURE WORKS

In order to meet the requirement of multi-damage recognition in engineering practice, this study proposed a new deep CNN framework for the damage detection of RC structures, namely, the CLDRM, by combining the state-of-the-art L*w*F with the ResNet 34 architecture. Three different experiments for four recognition tasks, including damage level, spalling check, component type and damage type determination were designed based on this training method to explore the optimal model parameters and applicable scenarios in damage recognition. The conclusions are as follows:

- Compared to conventional neural network models, the CLDRM can continuously train a model for multiple recognition tasks, without losing the prediction accuracy of old tasks. Additionally, the CLDRM provides robust performance with higher recognition accuracy and faster prediction for old and new tasks. It is achieved by optimized the model with less parameters and data storage.
- The CLDRM is more suitable for feature-related multitask learning. The recognition accuracy for old tasks decreases negligibly when learning new tasks.
- The learning order influences the CLDRM. It is important to determine the appropriate learning order of tasks and select when to end the learning process for achieving

better performance. To improve the recognition accuracy of the CLDRM for a particular task after fine-tuning, this task must be trained as the last task. This process promotes feature fusion to help improve the recognition accuracy for the final task.

Unlike TL, continual learning is a relatively new training technique in deep learning, and it has not yet been widely applied in civil engineering applications. Several extensions of continual learning can be explored in future, as follows:

- A GAN (Generative Adversarial Network) model [25] can be simultaneously trained with a particular task to maintain the memory of the dataset used in this training. Thus, when new tasks are added in the future, the past data collected by the GAN model can be directly used for training. This is equivalent to improving the training efficiency of joint training.
- Combining long-term memory and short-term memory, such as LSTM (Long Short Term Memory) [26], for model training can be further explored to alleviate the adverse effects of the gradual forgetting problem of continual learning [27]. In this manner, the model can preserve the recognition accuracy for a previously learned task and simultaneously improve the accuracy for a new recognition task.
- The CLDRM has the potential to be used for portable devices such as drone/robot owing to its small number of parameters and ability to rapidly and accurately learn new tasks.

ACKNOWLEDGMENTS

The authors would like to gratefully acknowledge the support from the National Key R&D Program of China (2018YFE0125400) and the National Natural Science Foundation of China (U1709216), which made the research possible. The project was also funded by Centre for Balance Architecture, Zhejiang University. In addition, the authors also need to acknowledge that the first author and the second author have equivalent contribution to the paper.

REFERENCES

[1] Z. Liu, Y. Cao, Y. Wang, and W. Wang, "Computer vision-based concrete crack detection using U-net fully convolutional networks," *Autom. Constr.*, vol. 104, pp. 129–139, 2019.

[2] Y. J. Cha, W. Choi, and O. Büyüköztürk, "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks," *Comput. Civ. Infrastruct. Eng.*, vol. 32, no. 5, pp. 361–378, 2017.

[3] B. F. Spencer, V. Hoskere, and Y. Narazaki, "Advances in Computer Vision-Based Civil Infrastructure Inspection and Monitoring," *Engineering*, vol. 5, no. 2. pp. 199–222, 2019.

[4] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[6] Y. J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk, "Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types," *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 9, pp. 731–747, 2018.

[7] T. Ghosh Mondal, M. R. Jahanshahi, R. T. Wu, and Z. Y. Wu, "Deep learning-based multi-class damage detection for autonomous post-disaster reconnaissance," *Struct. Control Heal. Monit.*, vol. 27, no. 4, 2020.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 779–788.

[9] C. Zhang, C. chen Chang, and M. Jamshidi, "Concrete bridge surface damage detection using a single-stage detector," *Comput. Civ. Infrastruct. Eng.*, vol. 35, no. 4, pp. 389–409, 2020.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[11] Y. Gao and K. M. Mosalam, "Deep Transfer Learning for Image-Based Structural Damage Recognition," *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 9, pp. 748–768, 2018.

[12] H. C. Shin *et al.*, "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems*, 2014, vol. 4, no. January, pp. 3320–3328.

[14] Y. Gao and K. M. Mosalam, "PEER Hub ImageNet: A Large-Scale Multiattribute Benchmark Data Set of Structural Images," *J. Struct. Eng.*, vol. 146, no. 10, p. 04020198, 2020.

[15] Y. Gao, B. Kong, and K. M. Mosalam, "Deep leaf-bootstrapping generative adversarial network for structural image data augmentation," *Comput. Civ. Infrastruct. Eng.*, vol. 34, no. 9, pp. 755–773, 2019.

[16] Z. Li and D. Hoiem, "Learning without Forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, 2018.

[17] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113. pp. 54–71, 2019.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 770–778.

[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," pp. 1–9, 2015.

[20] A. Grinys, V. Bocullo, and A. Gumuliauskas, "Research of Alkali Silica Reaction in Concrete With Active Mineral Additives," *J. Sustain. Archit. Civ. Eng.*, vol. 6, no. 1, 2014.

[21] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8693 LNCS, no. PART 5, pp. 740–755.

[22] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8689 LNCS, no. PART 1, pp. 818–833.

[23] R. Caruana, "Multitask Learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[24] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[25] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, vol. 3, no. January, pp. 2672–2680.

[26] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] A. Gepperth and C. Karaoguz, "A Bio-Inspired Incremental Learning Architecture for Applied Perceptual Problems," *Cognit. Comput.*, vol. 8, no. 5, pp. 924–934, 2016.