Journal of Scientific & Industrial Research
Vol. 84, October 2025, pp. 1095-1106
DOI: 10.56042/jsir.v84i10.18406

Addressing Catastrophic Forgetting in Neural Networks: A Review and Comparative Analysis of Algorithms for Crack Detection

Ivana Halfpap[1]* & Željko Đurović[1]

[1]University of Belgrade, School of Electrical Engineering, Bulevar kralja Aleksandra 73, 11000 Belgrade, Serbia

Received 01 April 2025; revised 04 September 2025; accepted 17 October 2025

Deep neural networks have demonstrated impressive results on a huge area of interest, but they suffer from catastrophic forgetting – when learning new tasks, they tend to forget what has been previously learned. Catastrophic forgetting in neural networks has garnered significant attention in recent years. The problem is too complex to generate general conclusions independently of the types of analyzed networks and specific issues. This paper addresses catastrophic forgetting in neural networks applied to crack detection. Given the limited research in this area, the existing research gap is highlighted in this study by reviewing recent works on catastrophic forgetting that explore various learning methods to mitigate forgetting. A comprehensive survey of techniques is conducted. Three representative algorithms used in crack detection datasets are evaluated - one from each major approach. Finally, a comparative analysis establishes a foundation for a novel algorithm that integrates the strengths of these different approaches.

Keywords: Catastrophic forgetting, Continuous learning, Image classification, ResNet18, Surface cracks

Introduction

Deep neural networks have outstanding performance with dealing the problems of object detection, object recognition[1,2], image classification or image segmentation.[3,4] However, despite their success, they face specific challenges. One major issue is that they fail to keep the previously learned knowledge when training on a new set of data, leading to the forgetting of information. This characteristic of the system is called catastrophic forgetting. Catastrophic forgetting is defined as a characteristic of a machine learning system where the exposure of the learning system to new information results in a severe loss of previously learned information.[5–7]

Catastrophic forgetting can occur in one system under different scenarios. In this work, the focus is on the catastrophic forgetting in continuous learning, also known as lifelong learning. Continuous learning presents a challenge because not all data is available initially. New data or knowledge is acquired over time, particularly when new classes are introduced gradually. New classes will be introduced over time and the model should retain its ability to accurately predict previously encountered tasks. Specifically, when the model is trained on the i-th task it should still maintain accurate predictions for all previously encountered tasks from the dataset j, where dataset j was introduced before dataset i.

This paper focuses on continuous learning in deep convolutional neural network for a crack detection task. The objective of this study is to sequentially train a shared model across multiple tasks while preventing catastrophic forgetting. As a trivial work around, one could store all incoming examples and retrain a deep neural network from scratch when needed, but this is impracticable in terms of required resources. As a result, numerous algorithms and techniques have been proposed in the literature to mitigate catastrophic forgetting in continual learning.

Various classifications exist for these algorithms. For the purpose of this work, the algorithms are classified as follows:

- **Replay**, which uses additional memory resources to enhance the training data with stored samples from previous tasks
- **Modularization**, which involves replicating, modifying, or adding new structures to the neural network
- **Regularization**, which focuses on preserving learned knowledge by adjusting the neural network's weights

*Author for Correspondence
E-mail: vi165019p@student.etf.bg.ac.rs

- **Combination,** which integrates elements from the above-mentioned categories to create hybrid solutions.

While recent works offer some reviews[8,9], they often lack coverage of the latest techniques and a comprehensive analysis of the algorithms. Moreover, although many studies explore catastrophic forgetting, not all are directly relevant to the crack detection problem. Previous investigations highlight that successful mitigation of catastrophic forgetting requires careful adaptation of algorithms to the characteristics of the given task.[10]

In this work, the most promising methods suitable for crack detection are evaluated, with one representative algorithm from each of the categories mentioned. To the best of Authors' knowledge, no prior study has conducted a comparative analysis of the algorithms for avoiding catastrophic forgetting focused on the crack detection problem.

The contributions of this work are as follows:

1. a comprehensive review of the existing algorithms from the aforementioned classification, specifically applied to the crack detection problem
2. extensive experiments to evaluate the effectiveness of several chosen algorithms
3. a comparative analysis of existing algorithms for mitigating catastrophic forgetting, including their advantages and disadvantages
4. suggesting future research directions by examining the benefits of various algorithms and proposing concepts for a new algorithm

Throughout this work, the following key questions are addressed:

- How do different approaches mitigate catastrophic forgetting in continual learning, specifically when introducing different categories of cracks over time?
- What are the specific benefits of each approach?
- How and when can the performance of these approaches be effectively measured? Is there a possibility of comparing these different models?
- Which method is most effective?
- Are the existing solutions sufficient, or is there potential for further improvement?

The rest of the paper is structured into five sections. First, a description of the neural network employed in the experiment is presented. Next, the databases used for this experiment are introduced, followed by an overview of existing methods and a detailed explanation of the specific approaches applied in this study. The subsequent section presents the measured performance and results, along with a comparative analysis that highlights the advantages and limitations of each method. Finally, the paper concludes with a summary that proposes ideas for a novel algorithm integrating the strengths of these approaches, as well as recommendations for future research.

Materials and Methods

Used Neural Network

In this experiment, the ResNet neural network is used.[11] ResNet18 was chosen because it has been shown to achieve high efficiency in crack detection compared to several other pretrained networks, such as GoogleNet and SqueezeNet.[12]

ResNet is composed of residual blocks. Each residual block consists of two or more layers. The key idea is that the information on the input can skip several convolutional layers and can be added back to the output. This skip connection makes the network effective even when it is very deep. Training works by updating the convolutional weights so that the residual outputs match the target outputs more closely.

The ResNet architecture consists of multiple versions with different complexity levels, from ResNet18 to ResNet152. Increasing the number of filters enhances the capacity and accuracy of the residual network. In this study, the smallest model ResNet18 is selected, an 18-layer deep network with over 11.5 million parameters. The ResNet18 architecture is as follows: a 7×7 convolutional layer with 64 filters, a 3×3 max-pooling layer, four blocks, each containing two residual units, a global average pooling layer and a fully connected layer. The structure of ResNet18 is illustrated in Fig. 1.

This research is conducted using the MATLAB® R2024 a programming language along with the Deep Learning Toolbox and Image Processing Toolbox and a processor an 11th Gen Intel® Core™ i7-11370H CPU at 3.30GHz.

In MATLAB, the pretrained version of the ResNet18 model, trained on the ImageNet database. ImageNet consists of 1.2 million images across 1,000 categories–features an output layer with 1,000 neurons, each corresponding to an ImageNet category.[13] This pretrained network is designed to classify images into a diverse range of object categories.

For this experiment, several modifications were made in MATLAB to adapt the neural network for

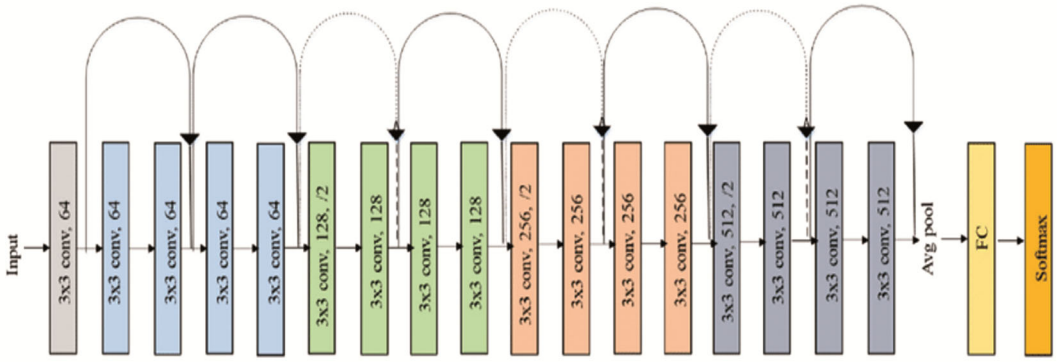

Fig. 1 — Structure of the ResNet18 neural network

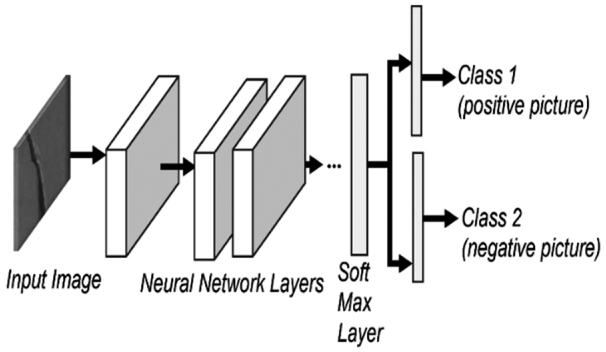

Fig. 2 — Simplified illustration of the neural network architecture

our specific task: the fully connected layer with a binary layer is replaced and the network is fine-tuned using crack training data.

The simplified version of the neural network is illustrated in Fig. 2.

Database

Neural network and algorithms are evaluated on two publicly available databases from Kaggle[14,15], which contain images illustrating different surface cracks. The cracks analyzed in this research include transverse, longitudinal, pothole, and alligator cracks. These databases were utilized in one of Authors' previous research papers.[16]

The crack detection problem is formulated as a one-class classification task, where the data has only two possible states: the surface either contains cracks or remains intact. In this study, the images are categorized as positive and negative images, images with cracks and without cracks, respectively.

The first database contains 40,000 images, equally divided into 20,000 positive and 20,000 negative samples, all with a resolution of 227 × 227 pixels, totaling 245.46 MB. The positive images in this research are categorized into two groups, focusing solely on longitudinal and transverse cracks.

The second database consists of 6,000 images, with 4,000 showing cracks and 2,000 without cracks, occupying 71.46 MB. All images in this database have a resolution of 256 × 256 pixels. These images were categorized into two groups, considering only potholes and transverse cracks for this study. To expand the dataset, these images were rotated by 90°, 180°, and 270°. The dataset for each category is split into 80% for training, 10% for testing, and 10% for validation. Further details on the number of images in each class are provided in Table 1, while examples of image classes from these two databases are shown in Fig. 3.

Methodology

In continuous learning, several events occur within a neural networks system:

- Learning: the neural network can process new data, allowing the system to recognize it
- Memorizing: the neural network is capable of retaining the learned data
- Forgetting: the neural network fails to recall previously learned data

It is essential that each new task is learned and retained while the performance for recognizing that specific task improves over time, a concept known as forward transfer. Simultaneously, previously learned data should not be forgotten, the performance of earlier tasks should improve or remains stable, a process referred to as backward transfer.[17]

In continuous learning three different scenarios are defined[18]: task-incremental, where new tasks emerge over time; domain-incremental, where the same problem must be solved across different domains; and class-incremental learning scenario, where the system gradually learns new classes over time. The main focus is on task-incremental learning, where new tasks or categories appear over time, but the system continues to classify within the same predefined classes, in this use case within positive and negative classes.

Table 1 — Summary of used datasets

	Database 1:Surface cracks		Database 2:Road cracks	
Total No. of pictures	40,000		6,000	
Used classes	2		2	
Total No. of used pictures	19112		5670	
Total No. of derived pictures	19112		17134	
No. of training pictures	15290		13708	
No. of test pictures	1911		1713	
No. of validation pictures	1911		1713	
Classification task	Category 1: positive pictures (transversal cracks) and negative pictures	Category 2: positive pictures (longitudinal cracks) and negative pictures	Category 1: positive pictures (alligator cracks) and negative pictures	Category 2: positive pictures (pothole cracks) and negative pictures

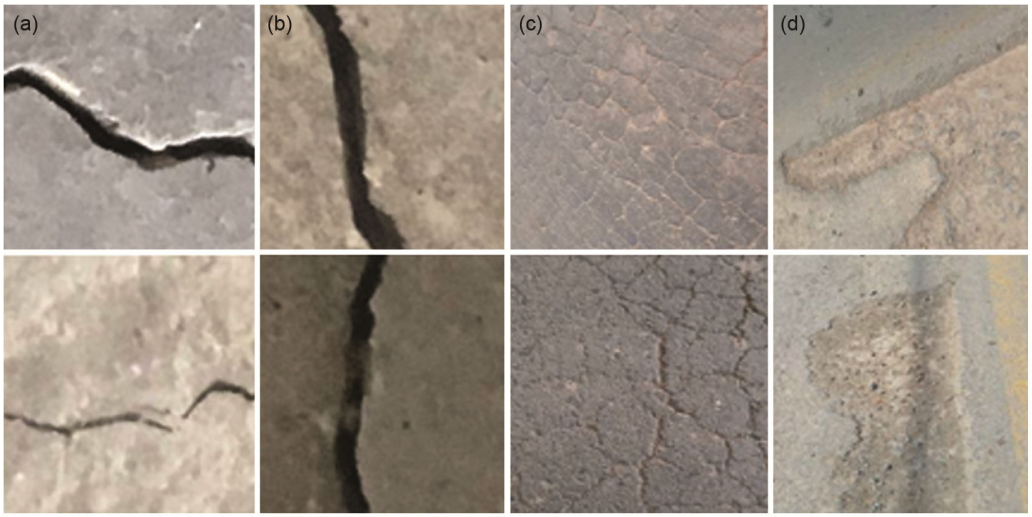

Fig. 3 — Selected examples from (a) first database - category 1 (transversal cracks), (b) first database - category 2 (longitudinal cracks), (c) second database - category 1 (alligator cracks) and (d) second database - category 2 (pothole cracks)

The continuous learning simulation follows the following approach: The first category of cracks is introduced at time t, followed by the second category at time t+1. Initially, at time t, the neural network is trained using data from the first category and evaluated. At time t+1, the neural network is retrained with new data, data from the second category. At this stage, the neural network is validated using data from the first category, and its performance is assessed to determine the extent to which catastrophic forgetting has been mitigated. Without proper retraining algorithms, previously learned knowledge rom time t may be partially rewritten and forgotten. The continuous learning approach is illustrated in Fig. 4.

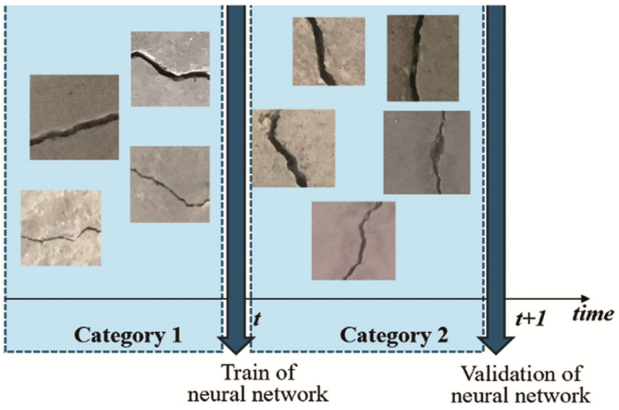

Fig. 4 — Overview of continual learning: task samples from a new category of the same class are added sequentially

Methods based on the Repetition of the Data/Memory

Replay-based methods save a selected sample of previous training data and integrate it with current data. These algorithms were initially developed as a solution to prevent catastrophic forgetting by retaining the original data in its raw form. One of the first developed algorithms in this category was

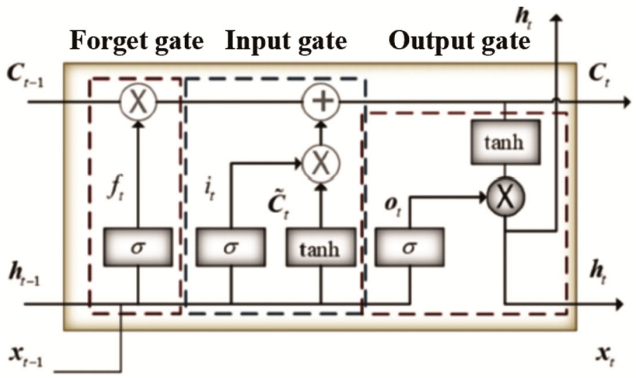

Fig. 5 — Basic structure of the Long Short-Term Memory (LSTM) unit

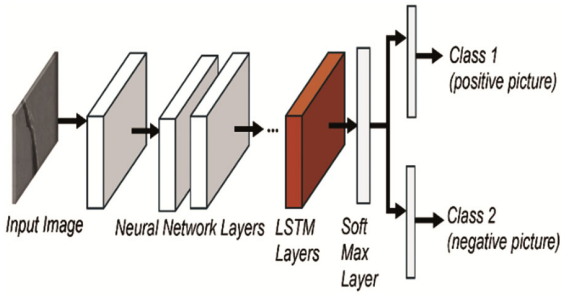

Fig. 6 — Architecture of the neural network model utilizing the LSTM layer in a cascade

replaying old data in a raw form during retraining[6], but this approach presents several challenges:

- privacy concerns related with storing raw data,
- the time required to retrain the neural network with both old and new data,
- the data may be too large and difficult to manage,
- memory limitations when storing both past and new data,
- previously seen data may not be available during retraining.

These constraints have made it impractical to maintain the algorithm as originally designed.

Over time, various improvements have been proposed to address these issues. For instance, one notable approach is the Replay Using Memory Index algorithm (REMIND), which operates in two steps: memory indexing and storage, followed by the reconstruction of old data and its integration with new data.[19] The algorithm proposed from Yang *et al.*[20] has two additional memory mechanisms: a short-term FIFO memory, which temporarily stores data and a long-term memory. From short-term memory only selected data is transferred to long-term memory and used for retraining. The FIFO buffer is refreshing continuously with incoming experiences based on five criteria: one of which is the "surprise effect", ensuring that previously unseen data is memorized. A similar approach, the Neural Data Filter (NDF), is designed to explore automatically important data instances from a sequential stream of data.[21] Furthermore, Zhang *et al.*[22] proposed a method termed memory recall. Memory recall utilizes additional memory modules to keep the feature statistics for each learned task. The data replay is achieved through Gaussian distribution-based feature regeneration. Another approach, introduces an additional latent replay layer within the neural network.[23] This layer is positioned in the middle of the neural network, between the layers responsible for low-level generic features and class-specific feature.

Our experiment is based on the approach of Gheisari *et al.*[24], which involves incorporating Long Short-Term Memory (LSTM) into the deep neural network. Long short-term memory is a type of recurrent neural network designed to model sequential data.[25] It is composed of memory cells, and gates – forget gate, input and output gate. Each memory cell has an internal state and gates. Memory cells are maintaining the information from the past. In the forget gate the input is combined with previous output and it is decided which previous information should be deleted, input gate decides how valuable the current input it is and what should be preserved, where the output gate calculates the output of the LSTM model. Model of the LSTM[26] is shown in Fig. 5.

In our approach, ResNet18 and LSTM are implemented in a cascade, or more accurately, these networks are concatenated. The ResNet18 neural network is handling feature extraction, while the LSTM layers is capturing dependencies between different image classes. The architecture is structured as follows: The input image is first processed through the ResNet18 neural network model to generate a feature map. This feature map is then fed into the LSTM layer, which processes the features over time. Finally, last layers of network are added for the classification and for the final output. LSTM layers are integrated between the ResNet layers, along with the final layers required for classification, such as the Fully Connected, SoftMax, and Classification layers. This hybrid architecture combines the ResNet18 neural network's strength in feature extraction and the LSTM's capacity for managing continuous learning and mitigating catastrophic forgetting. The architecture of the neural network is illustrated in Fig. 6.

Methods based on the Modules

These algorithms suggest that addressing catastrophic forgetting involves re-utilizing the modules or layers of a neural network. During literature research, two main approaches were identified: one involves creating a copy of the neural network with certain extensions, while the other preserves some layers or paths of the network to retain past information - primarily by freezing layers to maintain the learned weights.

The first approach encompasses methods in which a separate neural network is assigned to the source model and each target model – more precisely, to each task – with predefined connections to facilitate knowledge transfer.[27,28] In this setup, the new network learns only the new task, while the weights in the source model remain fixed. However, this means that with each new task, or target model, a significant amount of resources is required and the time needed for knowledge transfer increases substantially.

For accelerating the experimentation process by simultaneously transferring the knowledge from a previous network to deeper or wider network the Net2Net technique can be applied.[29] Another example occupying less memory resources is universal residual adapter modules, introducing only one single, designed to address different tasks.[30]

The second approach includes methods that preserve the structure of the neural network. Kazal *et al.*[31], proposed increasing the depth of the neural network by adding new layers on top of the existing ones. Additionally, a new method is introduced to identify the most similar tasks, helping determine where to insert new layers. The final architecture of the neural network takes the form of a tree, where both the width and depth of the neural network are expanded. A similar approach presented Teherkov *et al.*[32], where for every new task some specific block of neurons is added. These blocks of neurons are added throughout the network, increasing only its width but not its depth.

In this experiment, the approach of maintain the original dimensions of the neural network without expanding it will be adopted. Instead, certain modules will be modified using a simple and effective learning algorithm designed to freeze specific modules of neural network. Several approaches are discussed in the literature –some freeze the initial layers of the neural network, while others keep those layers flexible and instead freeze the final layers. Alternatively, some works do not freeze layers, but

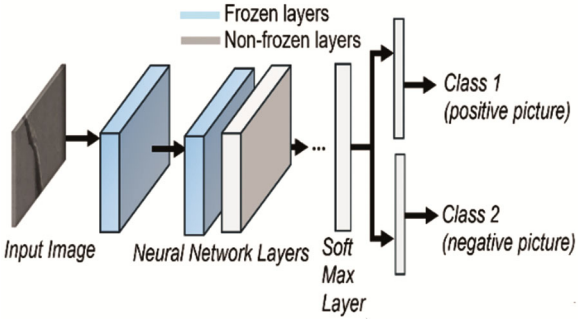

Fig.7 — Architecture of the neural network showing frozen layers

rather paths, for example in PathNet.[33] In that work, neither individual layers nor block of layers are frozen; instead certain parameters along a path learned on one task are preserved, while during time, new parameters for different tasks are introduced. In this study, the strategy of freezing layers of ResNet18. Since higher layers are more task-specific, the initial layers will be frozen, allowing the remaining layers to be adjusted and re-trained during the experiment. This model is illustrated in Fig. 7.

Methods based on the Regularization

Algorithms in this category use weights, loss functions, and other parameters of the neural network layers to mitigate catastrophic forgetting. Some methods involve simultaneously adjusting all parameters of the network like Elastic Weight Consolidation (EWC) algorithm.[34] In EWC, parameter updates are guided by a loss function based on the Fisher information matrix. Fisher information matrix identifies directions in feature space important for old, already learned data. Following the EWC algorithm, several modifications have been proposed. One such approach involves rotating parameter space of the neural network.[35] The goal of this method is to make Fisher information matrix, which is computed from gradients during the backward pass, approximately diagonal. Batra *et al.*[36] introduces an alternative approach, a hybrid algorithm that combines EWC's parameter strategy with variational posterior approximation – Elastic Variational Continual Learning with Weight Consolidation (EVCL). Another notable method based on the regularization is the Incremental Moment Matching (IMM).[37] This technique works by aligning the moments of the posterior distribution of a neural network that has been trained sequentially on two tasks. All of these methods can be effective but they are not always the optimal solution and come with certain limitations.

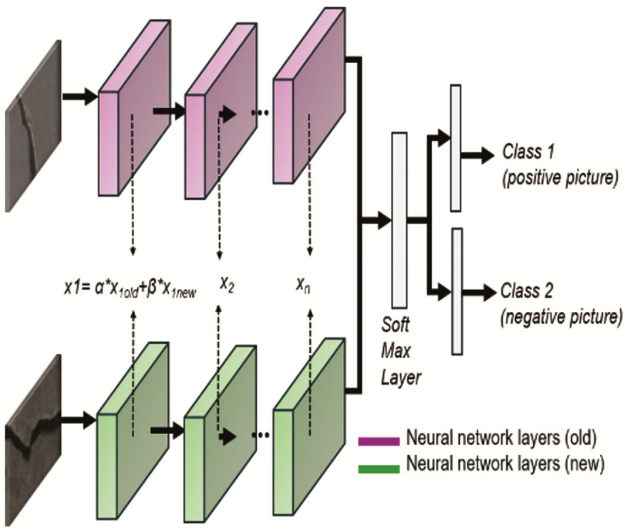

Fig. 8 — Architecture of the neural network model utilizing regularization method – two neural networks in parallel with the custom weights in between

Other approaches that are more complex, adjust only specific weights based on input data while keeping other weights unchanged. For instance, Aljundi *et al.*[38] describes the "supermask" algorithm, where weight modifications occur on demand across different layers of the neural network simultaneously. Similarly, Mallya *et al.*[39] presents Piggyback method, where binary mask during training phase is defined serving as a backbone for a network with fixed weight. During testing, the binary mask integrates with the existing network, selectively activating individual weights for specific tasks. Build upon this concept, Masana *et al.*[40] introduces the ternary mask, which expands the state options beyond simply "used" or "unused" to include "used" "learnable" and "unused" providing an optimized solution for both forward and backward transfer.

Our approach builds on *Liu et al.*[41], which presents two types of residual blocks at each residual level a: a stable block and a plastic block. The aggregation weights are introduced to adapt the balance between two blocks to the next level. In our research, this idea is extended throughout the entire neural network. New weights are calculated through Eq. 1:

$$x_i = \alpha_i x_i^{old} + \beta_i x_i^{new} \quad \dots (1)$$

For $i \in [1, N]$, where N is total number of layers in neural network. X_i^{old} represents the weight of layer i in the old neural network, x_i^{new} represents weight of layer i in the new neural network and α_i and β_i are experimentally determined parameters. This procedure is applied iteratively across all convolutional and fully-connected layers to update the network weights.

Finally, the last block's output is sent to the final layers of the neural network. An overview of this architecture is presented in Fig. 8.

Table 2 — Catastrophic forgetting in the crack detection

Time	Accuracy of the first database	Accuracy of the second database
t	100%	98.99%
t+1	91.7%	85.61%

Table 3 — Time required for train and retrain of ResNet18

Type of database	Time needed to train ResNet18	Time needed to retrain ResNet18
Database 1	174 min	152 min
Database 2	141 min	137 min

Result and Discussion

Evaluation of the algorithms is performed using the following metrics:

- **Retraining time**: The time required to retrain the neural network on the second category

Accuracy of the first category in *t+1*: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ … (2)

Recall of the first category in *t+1*: $Recall = \frac{TP}{TP+FN}$ … (3)

Precision of the first category in *t+1*: $Precision = \frac{TP}{TP+FP}$ … (4)

F1 Score of the first category in *t+1*: $F\,1\ Score = 2\ \frac{Precision * Recall}{Precision+Recall}$ … (5)

Confusion matrix

Used memory

where, *TP* represents the number of correctly predicted images from the positive class, *TN* represents the number of correctly predicted images from the negative class, while *FP* and *FN* denote the number of incorrectly sorted images from the negative and positive classes, respectively.

The existence of catastrophic forgetting will not be demonstrated, as it has already been established in one of the previous experiments.[16] These results are presented on Tables 2 & 3. These results were obtained in a scenario where no measures were taken to address the issue of catastrophic forgetting. The findings indicate that while the model was effectively trained on the new task, its performance of the old tasks through time

Table 4 — Parameters of models used in this study

Parameters	Parameters value
Learning rate	0.01
Number of epochs	8
Batch size	128
Validation frequency	10
Optimizer	SGDM

Table 5 — Performances of the LTSM algorithm

Performances	First database	Second database
Accuracy	96.65%	95.27%
Recall	97.48%	91.92%
Precision	95.88%	94.09%
F1	96.67%	92.99%
Time for retrain	222 min	234 min

Table 6 — Confusion matrix – LSTM algorithm – Database 1 & 2

Predicted Values		Actual Value	
		Positive	Negative
Database 1	Positive	465	12
	Negative	20	458
Database 2	Positive	239	21
	Negative	15	486

significantly dropped over time –dropping by more than 8% for the first database and over 13% for the second database.

After a certain number of epochs, the neural network tends to converge. Experimental results have shown that the best outcome was achieved with the following parameters shown in Table 4. The same parameters were used in all models run in this study.

Memory based Method / LSTM Memory Method

In this approach, selecting the number of hidden units in the LSTM layer was crucial. After testing various configurations, 125 units were adopted. The results are presented in Tables 5 & 6.

Results show that the outcomes are consistent, regardless of whether the data in the second category is similar to or entirely different from the first category. The algorithm produces comparable results in both cases.

Modularity Method / Layer-freezing Method

This experiment was partially covered in one of the previous research papers[16], where the number of layers to be frozen in the crack detection problem was investigated in order to make a balance between accuracy of the new task and minimizing catastrophic forgetting on the old task. The optimal results for crack detection problem were achieved by freezing initial layers up to layer 68, meaning layers 1–68 remained unchanged while the remaining layers were modified.

Table 7 — Performances of the Layer-freezing algorithm

Performances	First database	Second database
Accuracy	98.95%	92.77%
Recall	98.32%	82.31%
Precision	99.58%	95.96%
F1	98.95%	88.61%
Time for retrain	63 min	57 min

Table 8 — Confusion matrix – Layer-freezing algorithm – Database 1 & 2

Predicted values		Actual value	
		Positive	Negative
Database 1	Positive	469	8
	Negative	2	476
Database 2	Positive	214	46
	Negative	9	492

Table 9 — Performances of the aggregation weight algorithm

Performances	First database	Second database
Accuracy	97.80%	94.94%
Recall	98.67%	93.85%
Precision	96.91%	91.39%
F1	97.76%	92.54%
Time for retrain	155 min + 100 min	115 min +127 min

Table 10 — Confusion matrix – Aggregation weight algorithm – Database 1 & 2

Predicted Values		Actual value	
		Positive	Negative
Database 1	Positive	471	6
	Negative	15	463
Database 2	Positive	244	16
	Negative	23	488

The results are presented in Tables 7 & 8. Notably, the algorithm performs very effectively when the categories are similar, like in the first database. However, its effectiveness decreases in the second database, where the categories are more distinct.

Method based on the Regularization – Aggregation of Weights

The results of the algorithm are presented in Tables 9 & 10. Adjusting the aggregation weights is crucial in this approach. As previously noted, higher layers are more task-specific, so it is important to carefully determine how much data from the first category should be retained and to what extent.

In this experiment, both categories are treated as equally important. While this method is highly effective, it does have certain limitations: it relies on two neural networks, resulting in a more complex structure and significantly increased retraining time.

Comparison of the Methods

As observed, all of these methods were able to mitigate catastrophic forgetting to varying degrees. The method using LSTM memory has the drawback of requiring several additional layers to be incorporated into the neural network – such as a flatten layer, an LSTM layer, and a dropout layer, making the architecture of the neural network more complex. Except that, the time needed to retrain neural network is long, primarily because of this complex architecture. Additionally, several parameters, like the number of hidden units in the LSTM layer and the dropout probability, need to be experimentally defined for different use cases. On the other side, this method shows more stable performance compared to others, particularly when applied to different categories of images.

The layer-freezing method is simple, effective, and delivers higher accuracy. The time required to retrain the neural network is minimal, making this method a favorite in the experiment. However, it is worth noting that when dealing with entirely different categories over time, the layer freezing method struggles to remember both categories effectively. In this context, keeping all layers unchanged is not optimal. Nevertheless, some concepts - such as keeping certain parts of the modules unchanged –can be valuable for future research.

The regularization method is efficient but more complex than the modularity method, while still being simpler to implement than the memory-based approach. It requires managing two neural networks and carefully handling the dependencies between them. These dependencies play a critical role in determining how the previous category should be remembered and its relative importance. In this experiment, both categories were assumed to be equally important and variations in weight factors were therefore not explored extensively. Although retraining takes longer compared to other methods, this approach proves to be effective even when applied to entirely different categories within the datasets.

In addition to retraining time and effectiveness, it is important to compare the memory usage and number of Floating point operations per second (FLOPs) required for these mentioned algorithms. ResNet18 occupies approximately 42.62 MB in MATLAB® and 1.8 GFLOPs (Giga FLOPs). Among the evaluated methods, memory usage increases in the following order: layer freezing, LSTM, and regularization, with the regularization approach requiring more than double the memory. This method has a significant drawback in terms of memory, as it requires storing an additional copy of the neural network along with extra variables for each weight. On the other hand, the LSTM-based network requires approximately 45.05 MB, representing an increase of roughly 3 MB, along with an increase of 3.81×10^{-4} GFLOPs. The most memory-efficient approach is the layer freezing method that only alters the last layers. This method maintains overall memory unchanged, with only 1.91 KB being overwritten.

Furthermore, it is also crucial to examine whether there are any patterns within the cracks, specific types of cracks or common features that neural networks tend to forget. Additionally, understanding which specific cracks posed challenges for the algorithms and which shapes were overlooked by the neural network can provide valuable insights. Such data could potentially be reintroduced in a structured manner during the reteaching process, effectively "feeding" the neural network these images again when learning a second category.

For the first dataset, the neural networks performed exceptionally well. Due to the high success rate, both with and without catastrophic forgetting, identifying common failure patterns was very challenging. The neural network using LTSM and regularisation more frequently misclassified cracks as images without cracks as shown in the confusion matrices in Tables 6 & 10. While some misclassified images were common across both approaches, no clear pattern could be established among them. The layer freezing algorithm primarily struggled with images that had no cracks, particularly those affected by variations in lighting and shadows, as evidenced in the confusion matrices in Table 8. An example of a failure case using the layer freezing method is illustrated in Fig. 9. Although the first dataset did not reveal any definitive characteristics of the cracks, it was instrumental in demonstrating that overcoming catastrophic forgetting is achievable.

Fig. 9 — An example where the layer freezing algorithm failed

Fig. 10 — An example where all algorithms failed, leading to catastrophic forgetting in continuous learning: (a) all algorithms misclassified these images as negative in the second stage, (b) all algorithms misclassified these images as positive

A more intriguing aspect, however, is the second dataset, where the cracks exhibit greater diversity, leading to more pronounced catastrophic forgetting. Several images were consistently misclassified by all algorithms after retraining, as visualized in Fig. 10.

Based on this analysis, the comparison between the first and second datasets and the transition from successful to unsuccessful classifications over time, it can be concluded that to determine which images are more likely to be forgotten is only possible through a probability-based assessment. In the case of images that have cracks, those with small or background-blending cracks are more susceptible to forgetting. In contrast, for negative, crack-free images, those with significant shadow and lighting variations pose the greatest challenge. While no strict rules determine which images will be remembered, patterns of likelihood can be observed.

The findings from this experiment highlight areas that could be further refined. These insights lay the foundation for future research aimed at developing a novel algorithm –an algorithm that leverages the strengths of the three discussed methods, enhancing performance of the neural network and retraining time. In our future work, the custom algorithm should continue to focus specifically on the crack detection problem, as in this study.

As demonstrated in this experiment, the memory-based method offers stability, the modularity method provides simplicity, and the regularization method ensures effectiveness. Taking these factors into account, our goal for the future research is to implement an innovative algorithm that has a hybrid approach and integrates these advantages.

Some ideas include:

- introducing modular components that can be activated as needed based on the specific data categories
- preserving old, existing modules whenever possible,
- maintaining or gradually expanding memory over time,
- optimizing retraining efficiency by reducing the number of neural network parameters involved in retraining, thereby reducing retraining time.

Conclusions

Over the past decade, research in continual learning has advanced significantly, with numerous algorithms developed to mitigate catastrophic forgetting by preserving important parameters and retaining knowledge of previous data. However, a lack of comprehensive overview of existing algorithms and detailed comparative analyses of methods designed to prevent catastrophic forgetting remains, particularly in the context of the crack, highlighting a clear research gap. To address this, three representative algorithms were evaluated, each representing a major approach to mitigate forgetting: memory-based, module modification and regularization methods. Findings indicate that all approaches mitigate catastrophic forgetting to some extent, though the most suitable method depends on task-specific requirements. If minimizing retraining time or memory usage is crucial, algorithms avoiding additional architectural components, such as layer freezing, are most effective. However, if newly introduced cracks differ significantly, fixed modules are non-optimal. In such cases, incorporating

additional memory and implementing a hybrid approach is necessary.

Key insights from this analysis can contribute the development of a novel algorithm integrating the strengths of existing methods. Future research focuses on implementing this algorithm, evaluating its performance and reporting the results.

Conflict of Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Acknowledgement

This work was financially supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia under contract number: 451-03-65/2024-03/200103.

References

1 Zhao Z, Zheng P, Xu S T & Wu X, Object detection with deep learning: A Review, *IEEE Trans Neural Netw Learn Syst*, **30** (2019) 1–21, doi:10.1109/TNNLS.2018.2876865.

2 Wu X, Sahoo D & Hoi S, Recent advances in deep learning for object detection, *Neurocomputing*, **396** (2020) 39–64, doi:10.1016/j.neucom.2020.01.085.

3 Rawat W & Wang Z, Deep convolutional neural networks for image classification: A comprehensive review, *Neural Comput*, **29** (2017) 2352–2449, doi:10.1162/NECO_a_00990.

4 Voulodimos A, oulamis N, Doulamis A & Protopapadakis E, Deep learning for computer vision: A brief review, *Comput Intell Neurosci*, **2018** (2018) 1–13, doi:10.1155/2018/7068349.

5 McCloskey M & Cohen N, Catastrophic interference in connectionist networks: The sequential learning problem, *Psychol Learn Motiv*, **24** (1989) 109–165, doi:10.1016/S0079-7421(08)60536-8.

6 Ratcliff R, Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions, *Psychol Rev*, **97** (1990) 285–308, doi:10.1037/0033-295x.97.2.285.

7 Coop R & Arel I, Mitigation of catastrophic forgetting in recurrent neural networks using a fixed expansion layer, In *Int J Conf Neural Netw* (Dallas, USA) 2013, doi:10.1109/IJCNN.2013.6707047.

8 Parisi G, Kember R, Part J, Kanan C & Wermter S, Continual lifelong learning with neural networks: A review, *Neural Netw*, **113** (2019) 54–71, doi:10.1016/j.neunet.2019.01.012.

9 Maltoni D & Lomonaco V, Continuous earning in single-incremental-task scenarios, *Neural Netw*, **116** (2019) 56–73, doi:10.1016/j.neunet.2019.03.010.

10 Serra J, Suris D, Miron M & Karatzoglou A, Overcoming catastrophic forgetting with hard attention to the task, in *Int Conf Mach Learn* (Stockholm, Sweden), 2018, doi:10.48550/arXiv.1801.01423.

11 He K, Zhang X, Ren S & Sun J, Deep residual learning for image recognition, in *IEEE Conf Comput Vis Pattern Recognit*, (Las Vegas, USA) 2016, doi:10.48550/arXiv.1512.03385.

12 Halfpap I & Đurović Ž, Comparative analysis of pretrained deep neural networks for anomalies detection, in *Int Conf Electr, Electron Comput Eng*, (Niš, Serbia) 2023, doi:10.1109/IcETRAN59631.2023.10192245.

13 Deng J, Dongg W, Socher R, Li L, Li K & Fei L, ImageNet: A large-scale hierarchical image database, in *Conf Comput Vis Pattern Recognit* (Miami, USA) 20–26 June 2009, doi:10.1109/CVPR.2009.5206848; https://www.image-net.org/

14 https://www.kaggle.com/datasets/arunrk7/surface-crack-detection (Accessed: 01 April 2025)

15 https://www.kaggle.com/datasets/selahattinbareleb/road-crack-condition-images-pakistanv2 (Accessed: 01 April 2025)

16 Halfpap I & Đurović Ž, Overcoming catastrophic forgetting in neural network during continuous learning: A selective layer freezing approach for crack detection, in *Int Conf Electr*, *Electr Comput Eng*, (Niš, Serbia), 2024, doi:10.1109/IcETRAN62308.2024.10645158.

17 Hadsell R, Rao D, Rusu A & Pascanu R, Embracing change: Continual learning in deep neural networks, *Mach Behav*, **24** (2020) 1028–1040, doi:10.1016/j.tics.2020.09.004.

18 Ven G, Tuytelaars T & Tolias A, Three types of incremental learning, *Nat Mach Intell*, **4** (2022) 1185–1197, doi:10.1038/s42256-022-00568-3.

19 Hayes T L, Kafle K, Shrestha R, Acharya M & Kanan C, REMIND your neural network to prevent catastrophic forgetting, in *Comput Vis (ECCV)* (Glasgow, UK) 2020, doi:10.1007/978-3-030-58598-3_28.

20 Yang F, Tian F, Qin T&Bian J, Learning what data to learn, arXiv preprint, 2017, doi:10.48550/arXiv.1702.08635.

21 Isele D & Cosgun A, Slective experience replay for lifelong learning, in *AAI Conf ArtifIntell*, (New Orleans, USA) 2018.doi:10.48550/arXiv.1802.10269.

22 Zhang B, Guo Y, Li Y, He Y & Wang H, Memory recall: A simple neural network training framework against catastrophic forgetting, *IEEE Trans Neural Netw Learn Syst*, **33** (2022), doi:10.1109/TNNLS.2021.3099700.

23 Pellegrini L, Graffieti G, Lomonaco V & Maltoni D, Latent replay for real-time continual learning, in *IEEE Int Conf Intell Robots Sys(IROS)*, (Las Vegas, USA), 2020, doi:https://doi.org/10.48550/arXiv.1912.01100.

24 Gheisari S, Shariflou S, Kennedy P J, Agar A, Kalloniatis M & Golzan S M, A combined convolutional and recurrent neural network for enhanced glaucoma detection, *Sci Rep*, **11** (2021) 1–11, doi:10.1038/s41598-021-81554-4.

25 Hochreiter S & Schmidhuber J, Long short-term memory, *Neural Comput*, **9** (1997) 1735–1780, doi:10.1162/neco.1997.9.8.1735.

26 Jiang C, Chen Y, Chen S, Bo Y, Li W, Tian W & Guo J, A mixed deep recurrent neural network for MEMS gyroscope noise suppressing, *Electronics*, **8** (2019) 181–185, doi:10.3390/electronics8020181.

27 Rusu A A, Rabinowitz N, Desjardins G, Soyer H, Kirkpatrick J, Kavukcuoglu K, Pascanu R & Hadsell R, Progressive neural networks, arXiv preprint, 2016, doi:10.48550/arXiv.1606.04671.

28 Pathak R K & Yadav V, Improvised progressive neural network (iPNN) for handling catastrophic forgetting, in *Int Conf Electr Sustain Commun Syst* (Coimbatore, India) 2020, doi:10.1109/ICESC48915.2020.9156028.

29 Chen T, Goodfellow I & Shlens J, Net2Net: Accelerating learning via knowledge transfer, in *Int Conf Learn Represent* (San Juan, Puerto Rico) 2016, doi:10.48550/arXiv.1511.05641.

30 Rebuffi S A, Bilen H & Vedaldi A, Learning multiple visual domains with residual adapters, *Adv Neural Inf Process Syst* (Long Beach, USA) 2017, doi:10.48550/arXiv.1705.08045.

31 Kozal J & Wozniak M, Increasing depth of neural networks for life-long learning, *Inf Fusion*, **98** (2023), doi:10.1016/j.inffus.2023.101829.

32 Terekhov A V, Montone G & O'Regan J, Knowledge transfer in deep block-modular neural Networks, in *Living Mach* (Barcelona, Spain) 2015, doi:10.48550/arXiv.1908.08017.

33 Fernando C, Banarse D, Blundell C, Zwols Y, Ha D, Rusu A A, Pritzel A & Wierstrall D, PathNet: Evolution channels gradient descent in super neural networks, arXiv preprint, 2017, doi:10.48550/arXiv.1701.08734.

34 Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu A A, Milan K, Quan J, Ramalho T, Grabska-Barwinska A, Hassabis D, Clopath C, Kumaran D & Hadsell R, Overcoming catastrophic forgetting in neural networks, *Proc Natl Acad Sci*, (2017) 3251–3256, doi:10.48550/arXiv.1612.00796.

35 Liu X, Masana M, Herranz L, Weijer J, Lopez A & Bagdanov A, Rotate your networks: Better weight consolidation and less catastrophic forgetting, in *Int Conf Pattern Recognit* (Beijing) 2018, doi:10.1109/ICPR.2018.8545895.

36 Batra H & Clark R, EVCL: Elastic variational continual Learning with Weight Consolidation, in *ICML Workshop Struct Probab Inference Generative Model* (Vienna, Austria) 2024, doi:10.48550/arXiv.2406.15972.

37 Lee S W, Kim J H, Jun J, Ha J W & Zhang B T, Overcoming catastrophic forgetting by incremental moment matching, in *Proc Int Conf Neural Inf Process Sys* (Long Beach, USA) 2018, doi:10.48550/arXiv.1703.08475.

38 Aljundi R, Babiloni F, Elhoseiny M, Rohrbach M & Tuytelaars T, Memory aware synapses: Learning what (not) to forget, In*European Conf Comput Vis (ECCV)* (Munich, Germany) 2018, doi:10.1007/978-3-030-01219-9.

39 Mallya A, Davis D & Lazebnik S, Piggyback: Adapating a single network to multiple tasks by learning to mask weights, in *European Conf Comput Vis (ECCV)* (Munich, Germany) 2018, doi:10.48550/arXiv.1801.06519.

40 Masana M, Tuytelaars T & Weijer J, Ternary feature masks: Zero-forgetting for task-incremental learning, in *IEEE/CVF Conf Comput Vis Pattern Recognit Workshops* (Nashville, USA) 2021, doi:10.1109/CVPRW53098.2021.00396.

41 Liu Y, Schiele B & Sun Q, Adaptive aggregation networks for class-incremental learning, in *Conf Comput Vis Pattern Recogniti (CVPR)* (Nashville, USA) 2021, doi:10.1109/CVPR46437.2021.00257.