Smart Structures and Systems, *Vol. 31*, *No. 4* (*2023*) *335-349*
https://doi.org/10.12989/sss.2023.31.4.335

Ensemble-based deep learning for autonomous bridge component and damage segmentation leveraging Nested Reg-UNet

Abhishek Subedi [1], Wen Tang [1], Tarutal Ghosh Mondal [2], Rih-Teng Wu*[3] and Mohammad R. Jahanshahi [1,4]

[1] *Lyles School of Civil Engineering*, *Purdue University*, *West Lafayette*, *IN*, *USA*
[2] *Department of Civil*, *Architectural and Environmental Engineering*, *Missouri University of Science and Technology*, *Rolla*, *MO*, *USA*
[3] *Department of Civil Engineering*, *National Taiwan University*, *Taipei*, *Taiwan*
[4] *Elmore Family School of Electrical and Computer Engineering*, *Purdue University*, *West Lafayette*, *IN*, *USA*

(*Received October 26, 2022*, *Revised January 6, 2023*, *Accepted February 2, 2023*)

Abstract. Bridges constantly undergo deterioration and damage, the most common ones being concrete damage and exposed rebar. Periodic inspection of bridges to identify damages can aid in their quick remediation. Likewise, identifying components can provide context for damage assessment and help gauge a bridge's state of interaction with its surroundings. Current inspection techniques rely on manual site visits, which can be time-consuming and costly. More recently, robotic inspection assisted by autonomous data analytics based on Computer Vision (CV) and Artificial Intelligence (AI) has been viewed as a suitable alternative to manual inspection because of its efficiency and accuracy. To aid research in this avenue, this study performs a comparative assessment of different architectures, loss functions, and ensembling strategies for the autonomous segmentation of bridge components and damages. The experiments lead to several interesting discoveries. Nested Reg-UNet architecture is found to outperform five other state-of-the-art architectures in both damage and component segmentation tasks. The architecture is built by combining a Nested UNet style dense configuration with a pretrained RegNet encoder. In terms of the mean Intersection over Union (mIoU) metric, the Nested Reg-UNet architecture provides an improvement of 2.86% on the damage segmentation task and 1.66% on the component segmentation task compared to the state-of-the-art UNet architecture. Furthermore, it is demonstrated that incorporating the Lovasz-Softmax loss function to counter class imbalance can boost performance by 3.44% in the component segmentation task over the most employed alternative, weighted Cross Entropy (wCE). Finally, weighted softmax ensembling is found to be quite effective when used synchronously with the Nested Reg-UNet architecture by providing mIoU improvement of 0.74% in the component segmentation task and 1.14% in the damage segmentation task over a single-architecture baseline. Overall, the best mIoU of 92.50% for the component segmentation task and 84.19% for the damage segmentation task validate the feasibility of these techniques for autonomous bridge component and damage segmentation using RGB images.

Keywords: automated bridge inspection; component segmentation; damage segmentation; smart cities; structural health monitoring

1. Introduction

1.1 Motivation

Bridges are essential infrastructures that constantly undergo wear and damage from environmental factors, old age, and natural disasters. Quick inspection and assessment of damages can help in their effective restoration at an early stage, potentially saving financial resources and, in some cases, human lives. On the other hand, quick identification of components can aid in identifying damages and provides context for damage assessment. However, current manual inspection techniques are time-consuming, risky, and taxing on the human capital involved. Recent developments in AI and CV offer a promising alternative to this manual process. Instead of manually conducting inspections, an Unmanned Aerial Vehicle (UAV) can be deployed to capture photographs of bridges at regular intervals, and then these images can be fed into an AI model to detect the damages and components autonomously. Afterward, an assessment of whether rehabilitation efforts are warranted can be performed as a downstream task. This approach could save both time and money.

1.2 Related works

Mask R-CNN proposed by He *et al*. (2017) was utilized by Hou *et al*. (2020) for the segmentation of surface defects on stay cables. Zhou and Song (2021) relied on an encoder-decoder-based FCN to demonstrate that multi-modal deep learning based on heterogeneous fusion of RGB and range data can increase the accuracy of steel surface defects segmentation.

Pan and Zhang (2022) segmented concrete roadway cracks by invoking an attention based DeepLabv3+ network

*Corresponding author, Ph.D., Assistant Professor,
E-mail: rihtengwu@ntu.edu.tw

ISSN: 1738-1584 (Print), 1738-1991 (Online)

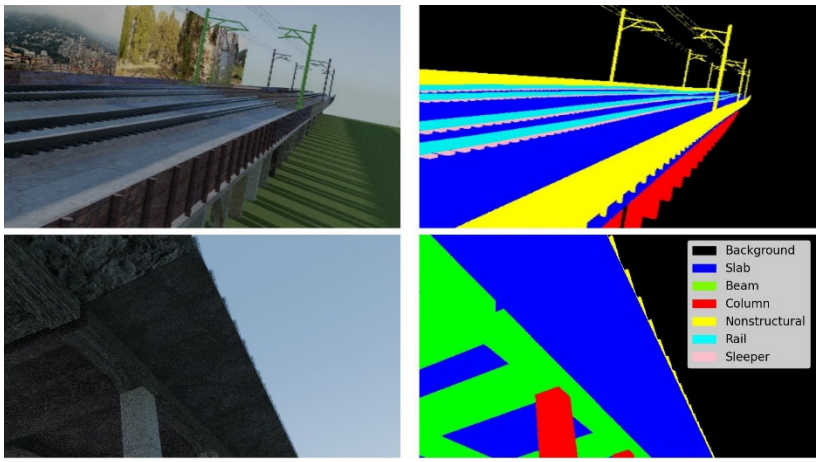

Fig. 1 Sample images with annotated components. Background class is not taken into consideration in this study

and demonstrated that the attention mechanism can help identify the most representative and meaningful features leading to a more efficient way of combining multi-scale features. Hoskere *et al*. (2020) introduced a multi-task learning semantic segmentation model that could identify the material types, such as concrete or steel, and structural damage, such as cracks or spalling, at the same time. Their hypothesis was that the material type could inform the prediction of the damage type and vice versa. They also found that multi-task learning implicitly increases the number of data samples and can significantly improve performance. Furthermore, Tang *et al*. (2022) proposed a cascaded design to detect cracks in steel girders, where the existence of cracks was first determined using a convolutional neural network, then the cracks themselves were segmented using UNet on overlapping image patches. Finally, Naive Bayes data fusion was used to merge the results in overlapping regions. Pozzer *et al*. (2021) studied deep neural networks in the context of detecting concrete damages by training the models on images from a buttress dam and testing them on images in bridges. Studies such as these suggest that deep learning methods are very effective at damage detection and segmentation. Deep learning typically requires large datasets. Recognizing this, Hoskere *et al*. (2021) introduced an open-source platform for quick labeling of images for semantic segmentation in structures. They conducted experiments by assigning human labelers to annotate using MATLAB and two variants of their process to prove the superiority of their method that made use of image processing techniques.

Rubio *et al*. (2019) created a dataset by manually annotating three labels per image for structural damages in bridge decks and used semantic segmentation to automatically estimate the damage regions. They primarily experimented with FCN and were able to achieve very high IoU values. Wang *et al*. (2020) put forward a two-stage damage segmentation method for segmenting damage in roof tiles of historic buildings. They first used an object detector to locate it and then performed a pixel-wise prediction. Although novel, their approach uses both a detector and a segmentor, which can make it expensive. Xia *et al*. (2022) proposed a method for segmenting bridge components from point cloud data. Their method works well when the number of samples in the dataset is small. Czerniawski and Leite (2020) introduced a pipeline for segmenting components in structures using a small dataset by making use of transfer learning, balancing classes, and managing overfitting. Mondal and Jahanshahi (2020) introduced a method for constructing a time evolution of damages in order to predict future damages, while Ghosh Mondal and Jahanshahi (2020) leveraged Faster R-CNN to detect spalling, surface cracks, exposed rebars, and severely buckled rebars in buildings impacted by earthquakes. Likewise, in Chen and Jahanshahi (2017), NB-CNN was proposed to automatically detect cracks in nuclear power plants, where the authors applied a novel Naive-Bayes data fusion scheme to significantly reduce the number of false positives.

While examples of CV-based assessment of components and damages for other structures are abundant in the literature, their systematic study in the context of bridges is scarce. It is important to mention here that components and damage in bridges often present with different forms and surroundings than those in buildings, dams, or nuclear power plants. Furthermore, deep learning is a highly specific endeavor, where interplays between the nature of the dataset and the nature of the model can make or break a technique. A technique that works well for buildings and pavements might not work well for bridges. As such, a proper study of AI-based tools for the unique needs of bridge inspection is paramount to propose unique solutions for the same. This study experiments with various state-of-the-art AI tools, including using different architectural configurations, loss functions, and ensembling techniques to improve the performance of damage and component segmentation in bridges. The work was carried out as part of the International Competition of Structural Health Monitoring, 2021. In this paper, the best approaches for the two tasks are presented and discussed in detail.

1.3 Contributions

This study carries out detailed assessments of some of the most recent advances in the field of semantic segmen-

tation to determine their applicability in the context of segmenting components and damages in bridges. The first assessment is with regard to the architecture. Six different architectures are compared, and it is discovered that a modified version of Nested UNet style decoder with RegNet as the encoder achieves significantly better performance than UNet, DeepLabV3+, PSPNet, LinkNet, and RefineNet. Transfer learning is employed on all the encoders to further consolidate the performance gains. The second assessment is with regard to the loss function when the pertinent class distributions are imbalanced. Weighted cross entropy and focal losses are commonly used to counter class imbalance. In this study, it is observed that Lovasz-Softmax loss function significantly outperforms both these losses. Finally, weighted-softmax ensembling of multiple architectures has proven to be an effective tool for improving the performance of machine learning systems. This study finds that thoughtful selection of architectures that form an ensemble is crucial to fully reap its benefits. This is demonstrated by observing that single-architecture Nested Reg-UNet outperforms an ensemble of five state-of-the-art architectures, while an ensemble formed by the Nested Reg-UNet architecture outperforms its single-architecture counterpart by wide margins. In a nutshell, this paper aims to fill some crucial gaps prevailing in this important area of research. Its key contributions are as follows:

- Autonomous component and damage detection have previously been explored for various structures, but the field's progress in the context of bridges lags behind. This study performs an extensive study of damage and component segmentation in bridges.
- A thorough comparison between six state-of-the-art architectures is conducted and a Nested UNet architectural configuration with a RegNet style encoder pre-trained on ImageNet is found to perform the best.
- The problem of bridge component and damage segmentation often suffers from class imbalance. In such context, loss functions such as weighted cross entropy and focal loss are more commonly used. This study demonstrates that the Lovasz-Softmax loss function is a better choice.
- An ensemble scheme that uses a weighted average of softmax scores is proven to drastically improve performance, typically when used simultaneously with the Nested Reg-UNet architecture.
- Finally, by combining these techniques, mIoU of 92.50% is achieved for the component segmentation task and 84.19% for the damage segmentation task, which are significant steps forward in autonomous damage and component segmentation in bridges.

1.4 Scope

The rest of the paper is organized as follows. Section 2 details the nature of the datasets, their splits, and their distributions. Section 3 explains the theory behind the approaches used in this paper: the Nested Reg-UNet architecture, the Lovasz-Softmax loss function, and weighted softmax ensembling. Section 4 lists out the training methodology, key hyperparameters, and the metrics of concern. Section 5 contains the results and their elaborate discussions. Finally, Section 6 summarizes the paper by highlighting the key findings.

2. Dataset

The datasets used in this study were provided as part of the International Competition of Structural Health Monitoring (ICSHM), 2021 and were created by the organizers (Narazaki *et al*. 2021) using synthetic environments based on 2000 real bridge designs. According to Spencer *et al*. (2019), synthetic data generation such as in Narazaki *et al*. (2021) and Hoskere *et al*. (2022) can solve some of the challenges associated with deep learning based structural health monitoring, such as limited data availability, manual data labeling, wide variety of environmental conditions, and generalization on unseen datasets. The full synthetic data generation process for the datasets used in this study is outlined in Narazaki *et al*. (2021). As described in the study, meshes were created to

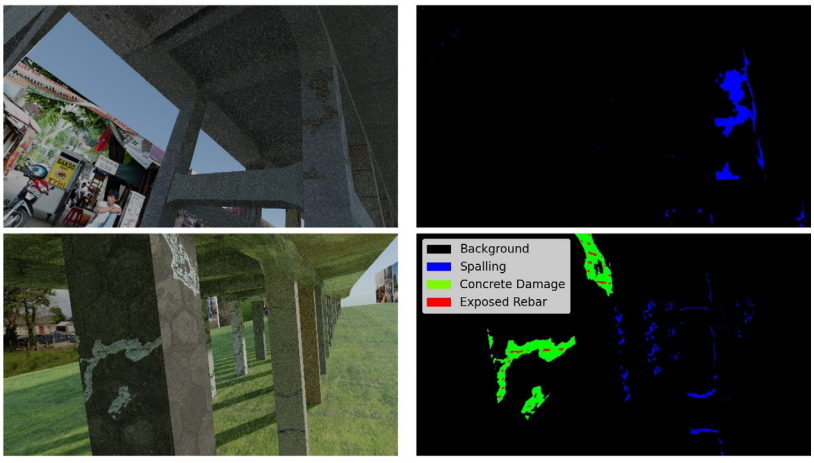

Fig. 2 Sample images with annotated damages. 'Background' and 'Spalling' classes are merged into one 'No Damage' class in this study

realize the geometry of the structures, after which texture for different regions of the mesh were extracted. Images were then mapped to the mesh surface to create a synthetic model. Finally, synthetic cameras were used to render the datasets. ICSHM, 2021 comprised two tasks, namely damage segmentation and component segmentation. A number of sample images are shown in Figs. 1 and 2. The problem was formulated as multi-class semantic segmentation, where individual pixels were classified into different classes. The background class was removed from consideration in the component segmentation task, whereas it was merged with 'Spalling' to form the 'No Damage' class in damage segmentation. A collection of 7,999 images of bridges containing ground truth component annotations were used to train and evaluate the component segmentation models. A small set of 500 images were randomly selected from the original set of 7,999 images and set aside to serve as the local test dataset. Likewise, the dataset used for damage segmentation consisted of 7,079 images of bridges. 500 of these images were randomly selected and set aside to serve as the local test dataset. Because the datasets were small, they were split only into the training and test sets, and validation set was not used in this work, which is consistent with small-dataset studies such as Hoskere *et al.* (2022), Yasuno *et al.* (2020), and Gorka and Armstrong (2021). However, with additional data, it might be a scope of future work to determine if the models generalize to the same performance levels in the presence of an additional validation set. Both datasets were highly imbalanced. The class-wise distributions of different components and damages in the respective datasets are shown in Fig. 3. Moreover, feeding the entire image into the segmentation network is ideal for preserving global information. However, as the original image size of 1920 × 1080 was too large to fit into the GPU's memory, the images were resized to 960 × 544 for the damage segmentation task and 640 × 480 for the component segmentation task.

3. Theoretical background

3.1 Nested Reg-UNet

Semantic segmentation is a well-researched problem, and the research community has widely accepted an encoder-decoder type arrangement as most suited for semantic segmentation tasks. In almost all the architectures in literature, there is an encoder, which extracts (or encodes) features at varying scales. The scale usually decreases as the encoder grows deeper, while the number of features increases. The information extracted by the encoder is then processed (or decoded) by the decoder, whose scale increases as one goes deeper but the number of features decreases. The main difference among the different state-of-the-art architectures is the layout of the encoders and/or the decoders.

Preliminary networks like UNet (Ronneberger *et al.* 2015) transferred information from the encoder to the decoder at various intermediate scales so that information from the encoder could be used during the decoding phase. However, this did not consider the gradual decrease in resolution and the consequent loss of information between different scales. To remedy it, RefineNet (Lin *et al.* 2017) used chained residual pooling and multi-resolution fusion operations to capture all the information present in the original image scale even when the scales were lowered at subsequent steps. DeepLabV3+ (Chen *et al.* 2018) introduced the Altrous Spatial Pyramid Pooling module for integrating information at various scales in order to capture both global and local details. DeepLabV3+ is used as one of the baselines in this study to compare against the proposed Nested Reg-UNet architecture since it has proven to be a formidable architecture in its own right and gave excellent initial results in background experiments. This made it a suitable candidate for comparison. In this work, a Nested UNet architectural style with a RegNet encoder is found to significantly outperform all the state-of-the-art semantic segmentation architectures that were experimented with in this study. Therefore, this combination is recommended to be used in segmenting bridge components and damages. The details of RegNet and Nested UNet are as follows.

When designing a typical network, one has to manually pick the different parameters, such as width, depth, size of convolutional filters, strides, and pooling size. The architecture is then either evaluated directly on a dataset or evaluated on a much bigger dataset, and the weights are transferred to a smaller dataset later. This is the general design strategy for almost all the network architectures in the literature but is incredibly tedious given the large number of possible combinations of the network parameters. RegNet takes this manual design to the level of a population of network architectures, thus increasing the

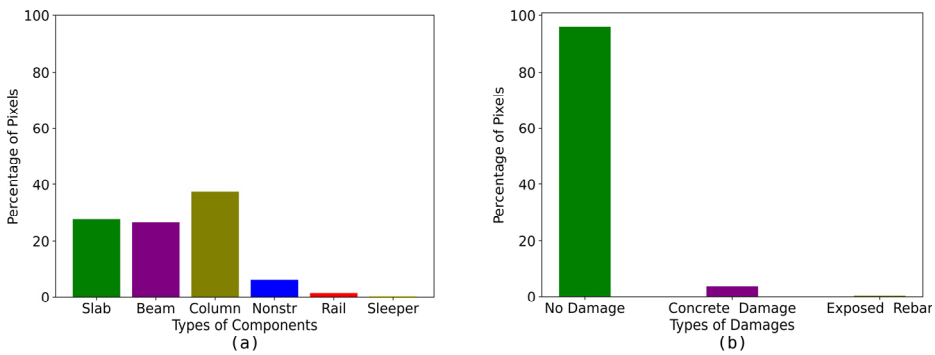

Fig. 3 Class distributions of (a) component; and (b) damage datasets

efficiency and precision of network selection. It was proposed by Radosavovic *et al*. (2020) and works on the principle of network design spaces. The process is analogous to Neural Architecture Search (NAS) but offers advantages over it such as better generalizability (Radosavovic *et al*. 2020). Unlike in NAS, the goal of the strategy is not to find an architecture that performs well for a very specific application under a very specific setting, but to find the best architecture from a population of architectures that works across a wide variety of settings, the theory being that a population of architectures contains some individual architectures that may characterize the population as a whole and thus work under different circumstances (Radosavovic *et al*. 2020). This is done by studying the characteristic patterns of different populations of architectures, which are composed of several individual networks. That is, these individual networks are sampled from their respective populations, and the distributions of the errors they produce (Eq. (1) (Radosavovic *et al*. 2020)) are analyzed to inform the understanding of the population.

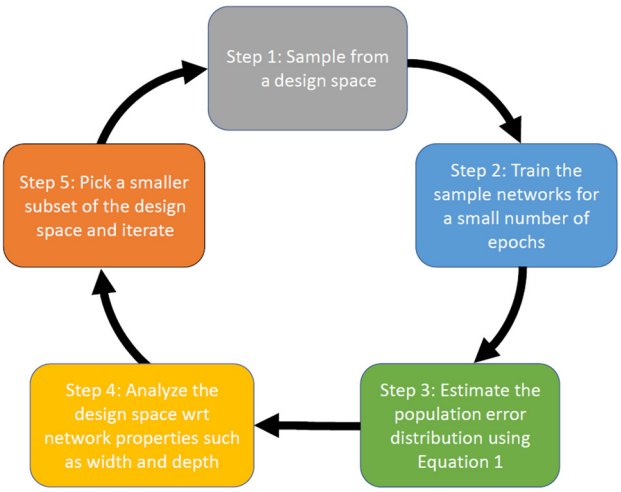

Fig. 4 Steps involved in the design of the RegNet architecture

$$F(e) = \frac{1}{n}\sum_{i=1}^{n} 1[e_i < e] \quad (1)$$

where n is the number of samples, e_i is the error in sample i, e is the thresholded error, and F (e) is the proportion of samples with an error below this threshold. The authors argue that studying the distribution of errors at the level of the population is more robust than studying one error in one individual network, which is the method used in traditional machine learning. The population is refined at each step of the training to yield a smaller and smaller version of it until convergence. The authors started with a design space called AnyNet (Radosavovic *et al*. 2020), which allowed for any possible network configuration. They then, through numerous iterations, refined this to a single architecture called RegNet. The general steps involved are outlined in Fig. 4. This work uses a pretrained RegNet architecture, the high-level details of which are presented in Fig. 5 (for the filter sizes and strides of the different components, the readers are encouraged to peruse Radosavovic *et al*. (2020)). The illustrated architecture's weights were transfered and fine-tuned from the source task (ImageNet classification) of Radosavovic *et al*. (2020) to the domain tasks of damage and component segmentation, and this forms the encoder part of the network proposed in this study.

Furthermore, as mentioned previously, a semantic segmentation architecture consists of an encoder-decoder arrangement. While RegNet was picked as the encoder, the general arrangement in which RegNet should be embedded is yet to be decided upon. Through background experiments,

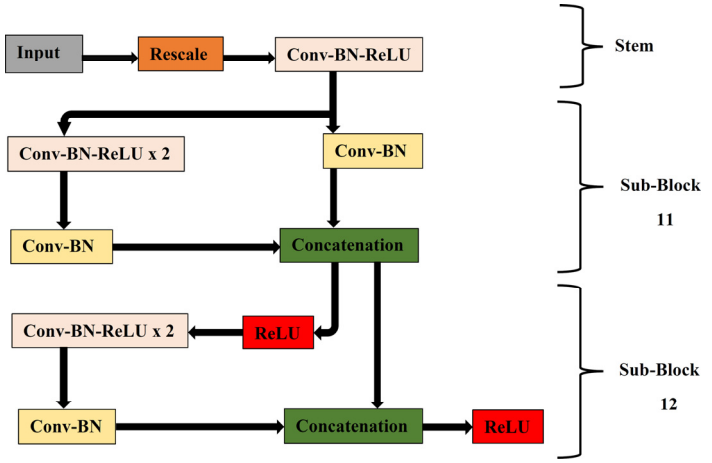

Fig. 5 Detailed diagram of the Stem and the first Block in the RegNet architecture. 'Input' represents the input image, 'Rescale' represents rescaling to the desired size, and 'Conv-BN-ReLU' represents a sequence of Convolution, Batch Normalization, and ReLU operations. Likewise, Conv-BN represents a sequence of Convolution and Batch Normalization operations. The top layer is the Stem. The Stem is connected to a Block, which contains Sub-Blocks as shown in the figure for Block 1. There are five such Blocks in the RegNet architecture. Blocks 1, 2, 3, 4, and 5 consist of 2, 6, 13, 1, and 1 sub-blocks respectively, starting with the input from the preceding Block. For instance, the input to Block 2 is the output of the final ReLU layer in the above figure

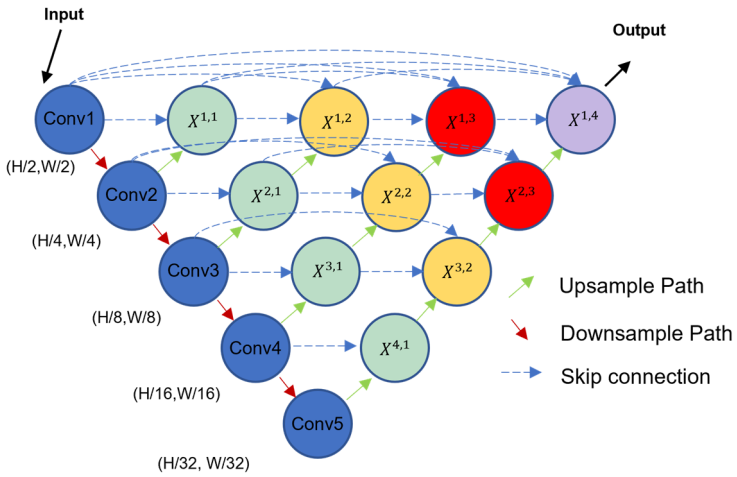

Fig. 6 A typical Nested UNet configuration. In the figure, the different colors of the nodes represent the different stages of nesting, with the same-colored nodes belonging to the same nesting stage. Likewise, 'Conv' represents convolution in the encoders, 'X' represents concatenation followed by convolution in the decoders, while 'H' and 'W' respectively represent the height and width of the input image

Nested UNet was identified as the best candidate, though it can be included in any other architectural configuration if necessary. A Nested UNet configuration is a modified version of the UNet configuration. It is illustrated in Fig. 6. Nested UNet was initially proposed by Zhou *et al*. (2018) for medical image segmentation. It is to UNet what the DenseNet architecture was to ResNet. Instead of a single path connecting the encoder to the decoder at each step, multiple paths connect to multiple nodes in the decoder. The decoder itself is composed of parallel blocks that are deeply supervised. The authors' rationale for this is to reduce the semantic gap between the encoder and decoder blocks to preserve information at each step in both phases. This resulted in improvements over the traditional UNet architecture in several medical imagery datasets (Zhou *et al*. 2018).

The Nested UNet configuration employs an optimization technique called deep supervision. This amounts to extracting intermediate feature maps and calculating loss values at several stages in the architecture, rather than using just one loss value following the output layer. The losses at intermediate stages are weighed lower than the loss from the output layer. Nevertheless, this has been proven to result in an improved performance since the loss landscapes of the intermediate features are supervised directly rather than relying on their correlation with the loss in the output layer. The proposed Nested Reg-UNet technique uses the RegNet architecture in a Nested UNet type dense configuration with deep supervision, which yields significant improvements over traditional architectures.

3.2 Lovasz-Softmax loss function

Loss function plays a critical role in the performance of deep learning models, especially when prediction classes are imbalanced. Cross entropy is one of the most popular loss functions, but its performance when the classes are imbalanced tends to be poor because it does not take class imbalance into account. Several alternatives have been proposed, such as weighted cross entropy, focal, tversky, jaccard, and dice loss functions that take class imbalance into account but tend to be difficult to optimize. Recently, Lovasz-Softmax loss function was proposed by Berman *et al*. (2018), which does not have this constraint. It optimizes over probabilistic Intersection over Union values, just like the jaccard loss, but it is much easier to optimize than all the aforementioned loss functions. The Lovasz-Softmax loss uses Lovasz extension of submodular set functions to optimize the IoU in a continuous fashion, which makes the optimization smoother. In this study, weighted cross entropy, focal, and Lovasz-Softmax loss functions were compared, and it was found that Lovasz-Softmax significantly outperforms weighted cross entropy and focal losses. Weighted cross entropy and focal losses were selected because they remain among the most widely used loss functions. The definition of each loss function is provided below.

Weighted cross entropy loss is defined as

$$L(y, y') = -\sum_{c}\sum_{p} \alpha y_c \log(y'_c) \tag{2}$$

Focal loss is defined as

$$L(y, y') = -\sum_{c}\sum_{p} \alpha y_c (1 - y'_c)^{\gamma} \log(y'_c) \tag{3}$$

Finally, Lovasz-Softmax loss function is defined as

$$L(y, y') = \sum_{c}\sum_{p} \delta_{J_c}(m(c)) \tag{4}$$

where

$$\delta_{J_c} = \frac{m(c)}{\{y = c\} \cup m(c)} \tag{5}$$

$$m(c) = \begin{cases} 1 - y_c' & if \quad y = c \\ y_c' & otherwise \end{cases} \tag{6}$$

The summations in each loss function are performed over all classes (represented as c) and pixels (represented as p). y is the ground-truth class and y' is the probabilistic output of the model corresponding to class c. Moreover, α and γ are weights that account for class imbalance by weighing tougher predictions at a higher value. In the Lovasz-Softmax loss function, instead of directly optimizing 1 - IoU, which is a common practice in the literature, a piecewise linear version of it is optimized based on the Lovasz extension. According to Berman *et al*. (2018), Eq. (4) is piecewise linear in $L(y, y')$ (readers are encouraged to refer the work for a detailed mathematical explanation of how Eqs. (5) and (6) ensure this). This is what makes the loss function so easy to optimize and results in improved performance over traditional loss functions, as demonstrated later in Section 5.1.

3.3 Weighted softmax ensembling

Ensembling refers to the practice of combining the prediction of more than one model so that the prediction of the whole can be improved. It is based on the premise that a combination of decisions reduces errors, which tend to be noisier than correct predictions. One ensemble strategy is to average the softmax scores from different models. Deep learning models output not just the classes but also assign probabilities to their predictions. A model's prediction contains information about how strongly it believes the pixel belongs to a certain class. It then assigns the pixel to the class which has the highest probabilistic value. For each pixel, the following formula holds:

$$\sum_c p_c = 1 \tag{7}$$

where c refers to the class and p refers to probabilistic decisions of the model. This means that the probabilities of all the classes should sum to 1 for any pixel in each model. Instead of taking just one model's outcome, softmax scores can be averaged across different models for each pixel location and class. In softmax-ensemble, the decision about which class the pixel belongs to is taken based on the class that has the highest average. Furthermore, to ensure that multiple poor performing models do not sway the result toward the wrong class, a weighted scheme can be adopted. That is, the probabilities can be weighted such that the better performing models have a higher weight compared to models that do not perform as well. In this study, a linearly weighted softmax ensemble strategy was used, where the weight constituted a linear function of mIoU. The formula for weighted softmax ensembling is given below:

$$C = c: \max\left(\frac{1}{m}\sum_m w_m p_{c,m}\right) \tag{8}$$

where $p_{c,m}$ refers to the softmax probability given out by the model m for class c, w_m refers to the weight given to model m, and C refers to the resulting class as predicted by the ensemble.

4. Methodology

4.1 Implementation details

The experiments were conducted on a remote server, which consisted of two Intel Xeon E5-2620 CPUs, 256 GB DDR4 RAM, and one NVIDIA RTX Quadro 8000 GPU with 48 GB memory. Transfer learning was used on all networks, where the backbones were pretrained on the ImageNet dataset (Deng *et al*. 2009). Data augmentations including flipping, random brightness, and contrast change were used to increase the diversity of the data. Likewise, Adam optimizer (Kingma and Ba 2014) was employed with a cyclical learning rate policy for the damage segmentation experiments, where the learning rate was varied between the initial value and one-tenth of the initial value. For more details about the cyclical learning rate, the readers are encouraged to refer to Smith (2017). Moreover, an exponentially varying learning rate policy was employed for the component segmentation experiments as shown in Eq. (9).

$$LR_{initial} * \left(1 - \frac{Epoch_{current}}{Epoch_{\max}}\right)^{0.9} \tag{9}$$

where $LR_{initial}$ stands for the initial learning rate, $Epoch_{current}$ represents the current epoch, and $Epoch_{max}$ denotes the maximum number of epochs. Unless otherwise stated, the experiments were performed *ceteris paribus* with an initial learning rate of 1e-4. The component segmentation architectures were trained for a maximum of 100 epochs, whereas the damage segmentation architectures were trained for a maximum of 200 epochs, and an early stopping strategy was enforced with a patience of 20 epochs. The batch size was 8 for Nested UNet, 16 for DeepLabV3+, 32 for PSPNet, and 24 for all other architectures.

4.1.1 Hyperparameter selection

The hyperparameter selection procedure was carefully controlled to ensure that each architecture achieved the best possible mIoU. Some of the hyperparameters that were deemed to be less correlated to changes in the architecture, such as data augmentation, were chosen using an ad-hoc approach for computational reasons and fixed for all architectures. During background experiments, the most important finding was that the architectures were very sensitive to changes in the ratio of initial learning rate to batch size, rather than just one or the other. Inspired by this, the same initial learning rate and learning rate schedule were chosen for all architectures, but the batch size was varied, since this also resulted in efficient training. The fact that the ratio of learning rate to batch size is more important than one or the other is a somewhat nascent realization in the field of machine learning, lately supported by papers such as He *et al*. (2019) and Goyal *et al*. (2017). Furthermore, the idea of changing the batch size but not the learning rate was adopted from the work in Smith *et al*.

Table 1 Best mIoU for different architectures during the first 20 epochs in the component segmentation task. ”-” means that the corresponding batch size does not fit in memory. The batch size decrement is done in intervals of 8. The next higher batch size of 40 doesn’t fit in a 48 GB GPU for any architecture

Batch size	Nested Reg-UNet (%)	UNet (%)	RefineNet (%)	PSPNet (%)	DeepLabV3+ (%)	LinkNet (%)
8	81.44	76.52	73.07	63.29	77.15	74.71
16	76.28	79.35	73.93	67.00	84.60	72.23
24	-	79.87	80.73	72.49	83.47	76.47
32	-	-	79.49	72.60	82.71	-

(2018). However, it was rigorously tested in background experiments by training the networks for a small number of epochs using a subset of the dataset with several initial learning-rate to batch-size ratios. The same phenomenon as the above studies was observed during this process, so the attention was completely turned to performing hyperparameter tuning of only the batch size on the full dataset, while also ensuring efficient utilization of resources. The value of batch size was varied to optimize performance, training time, and memory utilization. They were first selected in such a way that the data and the model architecture fit exactly in the GPU’s 48 GB memory. In the second step, they were gradually reduced, and the performance was noted by training the architectures for a small number of epochs. After this, the batch size that resulted in the highest performance (while also satisfying GPU memory constraint) was chosen. An interesting observation was that the optimum batch size did not depend on the dataset, but only on the architecture, since the optimum batch sizes for both the damage and component segmentation tasks were found to be the same for all architectures. Perhaps more importantly however, this method of selecting the batch size ensured faster training and efficient utilization of hardware resources, while also making sure that the best possible performance was reached for each architecture. Sample observations for the component segmentation task are presented in Table 1, considering a batch size decrement of 8 and for the first 20 epochs.

4.2 Evaluation metrics

The primary evaluation criteria for the competition was the mean Intersection over Union (mIoU) (Eq. (10)), and the same is used in this study. However, the approaches are also compared using other metrics, such as precision (Eq. (11)), recall (Eq. (12)), and f1-score (Eq. (13)). Accuracy is not used because of class imbalance. Some minor adjustments to the way the metrics were computed in the competition were made. First, in the damage segmentation task, the goal of the competition was only to segment exposed rebar and concrete damages. There were two possible ways to deal with the remaining classes: merge them or ignore them. In this study, ‘Spalling’ and ‘Background’ classes have been merged to form a third class called ‘No Damage’ because our background experiments hinted that this would result in improved performance for the main classes. Second, instead of averaging over the different images first and then classes second, all the images in the test set are concatenated (side-by-side), and the metrics are averaged per class at once. This is done because of the distribution of the dataset. Many the test images do not contain some of the minority classes, and averaging across the images rather than taking the entire test set at once would result in a faulty estimate (lower than the true value) since the standard practice for images that do not contain a specific class is to assign 0 as the metric value. This would not have been an issue of a small number of images did not contain the minority classes, which was not the case in the dataset. The formulae for the different metrics are given below:

$$mIoU = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FP_i + FN_i} \tag{10}$$

$$Precision = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FP_i} \tag{11}$$

$$Recall = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FN_i} \tag{12}$$

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{13}$$

where, C is the number of classes, while TP_i, FP_i, and FN_i stand for the number of true positives, false positives, and false negatives, respectively, for the i^{th} class.

5. Results and discussions

5.1 Lovasz-Softmax loss function

The choice of a loss function is often one of the preliminary steps in the design of a machine learning pipeline. Three loss functions were evaluated in this study, each characterized by its own strengths and weaknesses. The definitions of the loss functions are provided in Section 3.2. A detailed comparative assessment of the loss functions that required an exhaustive analysis treating all other variables as constants was beyond the scope of this study due to limited computation budget. As a less expensive

alternative, this study selected the component segmentation task and Nested Reg-UNet architecture as an experimental testbed to identify the most accurate loss function. The result of this experiment was, however, extended to all the subsequent analyses presented later in the paper. The performance of various loss functions is presented in Table 2 in terms of precision, recall, f1-score and mIoU.

As can be seen, the weighted cross-entropy, focal, and Lovasz-Softmax loss functions yielded mIoUs of 88.32%, 83.53%, 91.76%, respectively. This indicates a 3.44% improvement over weighted cross entropy, the most used loss function in the literature, and an astounding 8.23% improvement over the focal loss. A similar phenomenon is observed for precision, recall, and the f1-score, where Lovasz-Softmax loss outperforms the weighted cross entropy by 3.21%, 0.92%, and 2.06%, respectively. The corresponding improvements over focal loss are observed to be 5.15%, 4.73%, and 4.93%, respectively. Overall, the Lovasz-Softmax loss function can lead to significantly better performance than traditional loss functions and is therefore adopted for all subsequent analyses in this study.

5.2 Nested Reg-UNet

5.2.1 Damage segmentation

A set of six architectures were considered in this study for damage segmentation, namely DeepLabV3+, RefineNet, LinkNet, PSPNet, UNet, and the proposed Nested Reg-UNet. The performance of different architectures is presented in Table 3. As evident from the table, the proposed Nested Reg-UNet significantly outperforms all other architectures in terms of all the evaluation metrics considered in this study. For example, its precision, recall, f1-score, and mIoU are 1.15%, recall, f1-score, and mIoU are 1.15%, 2.26%, 1.91%, and 2.86% higher than the next-best architecture in the respective metric categories. This implies a considerable improvement over state-of-the-art and constitutes a significant contribution of this work. Processing speed is another important parameter in robotic inspections. Therefore, processing speed is measured in this study in terms of inference time, which is the time required to process a single image. The inference times for different architectures are summarized in Table 3. It is observed that the inference time for the proposed Nested Reg-UNet is comparable to all other architectures except PSPNet. The observed inference time of 31 milliseconds for an image of 544 × 960 resolution implies that the proposed network can process an inspection video in real time at a frame rate of 30 FPS. However, real-time inspection of a higher resolution video at a higher frame rate could be challenging, which is the scope for future work.

Class-wise comparison of performance in terms of IoU is presented as a heatmap in Fig. 7. All the architectures invariably exhibited high accuracy in segmenting the nondamaged areas. This can be attributed to the well-defined nature of this class and its ubiquitous presence in the dataset. However, the architectures showed a wide variation in accuracy when it came to the segmentation of concrete damage and exposed rebars. The proposed Nested Reg- UNet achieved an IoU of 79.86% and 70.10% for concrete damage and exposed rebars, respectively, which indicate a 3.2% and 4.28% improvement over the next-best architecture in the respective damage categories. Class-wise performance has an important ramification since detecting damage is more critical than detecting nondamaged areas. Thus, in terms of the mIoU computed by disregarding the 'no damage' class, the proposed Nested Reg-UNet outperforms its closest competitor (LinkNet) by a significant margin of 4.29%. The performance of the different architectures per epoch is illustrated in Fig. 8 for

Table 2 Comparison of loss functions for the component segmentation task when using Nested Reg-UNet. The best result for each metric has been boldfaced

Loss	Precision (%)	Recall (%)	F1-score (%)	mIoU (%)
wCE	92.76	94.38	93.57	88.32
Focal	90.82	90.57	90.70	83.53
Lovasz-Softmax*	**95.97**	**95.30**	**95.63**	**91.76**

*denotes the proposed loss function.
wCE = weighted Cross Entropy

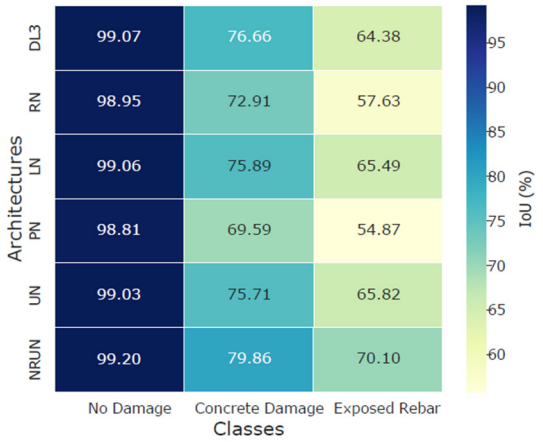

Fig. 7 Per-class comparison of the proposed architecture with other state-of-the-art architectures for the damage segmentation task; DL3: DeepLabV3+, RN: RefineNet, LN: LinkNet, PN: PSPNet, UN: UNet, and NRUN: Nested Reg-UNet

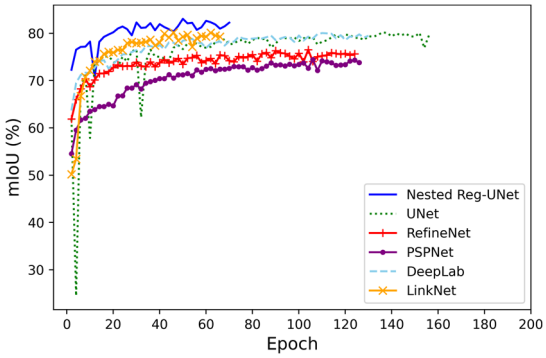

Fig. 8 Damage segmentation mIoU vs epoch curve for different architectures on the test set

Table 3 Comparison of different architectures for the damage segmentation task. The best result for each metric has been boldfaced

Architecture	Precision (%)	Recall (%)	F1-score (%)	mIoU (%)	Inference time (ms)
DeepLabV3+	89.24	87.27	88.24	80.03	32
RefineNet	85.64	85.65	85.64	76.50	31
LinkNet	88.80	87.86	88.33	80.15	31
PSPNet	85.42	82.87	84.13	74.42	**12**
UNet	88.83	87.90	88.36	80.19	32
Nested Reg-UNet*	**90.39**	**90.16**	**90.27**	**83.05**	31

the test set. From the figure, it can be observed that the performances rapidly increase during the first few epochs, then become noisy, and finally gradually become stable until they no longer increase. All the architectures converge at different number of epochs, and no architecture reaches the maximum allowable 200 epochs, since early stopping with a patience of 20 is employed. UNet takes the highest number of epochs to converge, whereas LinkNet takes the smallest.

Several sample segmentation masks predicted by different CNN architectures are presented and compared with the ground truths in Fig. 9. Likewise, the mIoUs for individual architectures are presented under each subfigure. In Figs. 9(a) and 9(b), although there is negligible qualitative difference between different architectures' predictions, some architectures have a significantly lower mIoU than Nested Reg-UNet. Upon a closer inspection, an interesting phenomenon was observed in Fig. 9(a) - while the total area (in terms of number of pixels) of damage was almost the same for most architectures, the location of damage was more precise for LinkNet, UNet, and Nested Reg-UNet, resulting in their higher mIoUs. Likewise, for Fig. 9(b), the predicted area of the damage was significantly lower than the ground truth for almost all the architectures, including Nested Reg-UNet, despite the fact that there does not seem to be much qualitative error. To be specific, the concrete damage borders were found to be ever-so-slightly thicker in the ground truth than in the predictions, but this is not visible to the naked eye. On the other hand, in the examples presented in Figs. 9(c) and 9(d), Nested Reg-

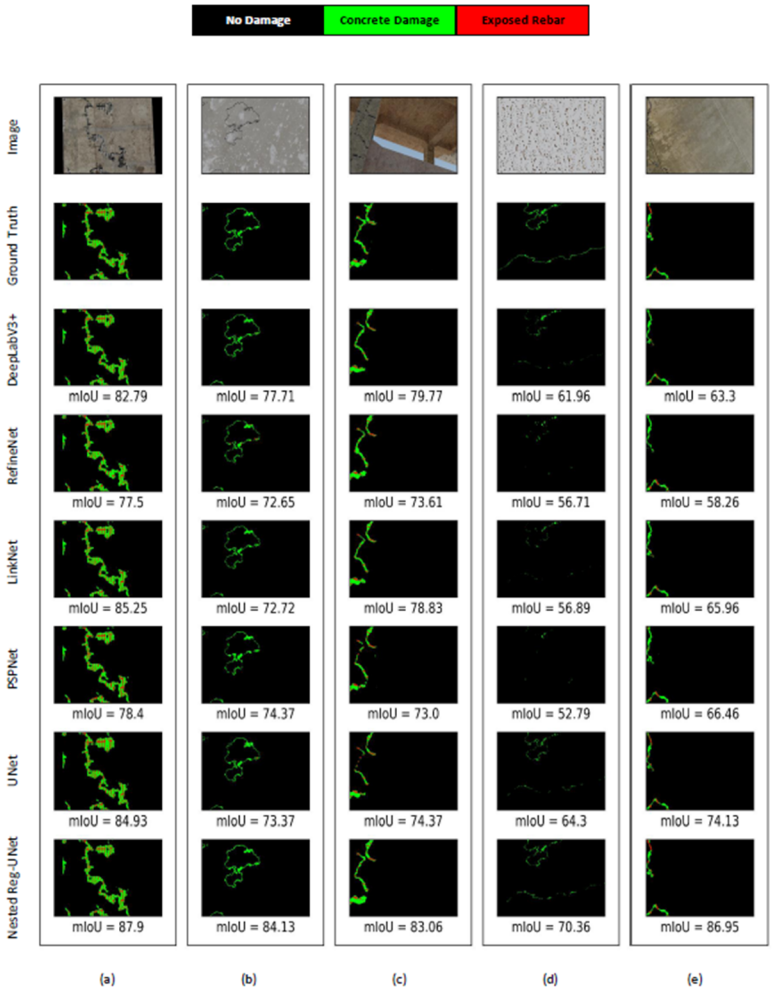

Fig. 9 Sample predictions for the damage segmentation task, where each column corresponds to an image and each row either corresponds to the ground truth or the prediction for different architectures as labeled on the left edge. The color map is shown at the top of the figure, while per-image mIoUs are shown under each subfigure for different cases

UNet performs better than its counterparts in segmenting concrete damage. However, it does not show any appreciable advantage over other architectures with regard to segmenting exposed rebars. In the case presented in Fig. 9(e), Nested Reg-UNet segments the exposed rebar more accurately than other architectures (for instance, the exposed rebar on the top left part). This yields an extremely large mIoU compared to other architectures, since the classes are given equal weightage in mIoU computation. Overall, if Nested Reg-UNet's mIoU is compared to the next best mIoU, the difference is 2.65%, 6.42%, 3.29%, 6.06%, and 12.82% respectively for Figs. 9(a)-(e). In particular, the performance difference is wider for Figs. 9(b), 9(d), and 9(e) than for Figs. 9(a) and 9(c). The former images have more challenging background color and texture, which bear a close resemblance to the corresponding damages. In such complex scenarios, Nested Reg-UNet was observed to perform much better than its counterparts. A careful analysis of the results indicates a clear trend. Nested Reg-UNet performs better than alternative architectures in a variety of cases, ranging from well-defined damages such as in Figs. 9(a), 9(b), and 9(c) to somewhat less developed damages in Figs. 9(d), and 9(e). The fact that Nested Reg-UNet performs better than other architectures in the latter category of images is an especially significant addition to the existing knowledge base, since it implies that Nested Reg-UNet can be leveraged for detecting early-stage damages, which tend to be less well-formed than mature late-stage damages. This will enable quick identification and early repair of bridge defects, which is more cost-effective than delayed overdue remediations.

5.2.2 Component segmentation

This study employed the same six architectures described in the previous section for component segmentation: DeepLabV3+, RefineNet, LinkNet, PSPNet, UNet, and the proposed Nested Reg-UNet. The results are summarized in Table 4. As before, the proposed Nested Reg-UNet outperformed all the baseline models in terms of all four evaluation metrics. The precision (95.97%), recall (95.30%), f1-score (95.63%), and mIoU (91.76%) values produced by it are 0.38%, 0.78%, 0.94%, and 1.61% higher than the best result produced by the baseline models for respective metric categories. This superiority of the proposed Nested Reg-UNet architecture over the state-of-the-art baseline models is also reflected in the class-wise accuracies depicted in Fig. 10. The Nested Reg-UNet outperformed all the baseline architectures for all component classes. However, the advantage of the Nested Reg-UNet is more vivid for the rail and sleeper classes, where a 1.66% and 3.82% performance boost were observed compared to the best baseline model in the respective component classes. For slab, beam, column, and nonstructural components, the improvements over the best baseline models were 0.52%, 0.73%, 0.40%, and 0.36%, respectively. This implies that the relative superiority of the proposed Nested Reg-UNet vis-`a-vis the baseline models cuts across different tasks pertaining to the autonomous inspection of reinforced concrete bridges. The performance of the different architectures per epoch is illustrated in Fig.

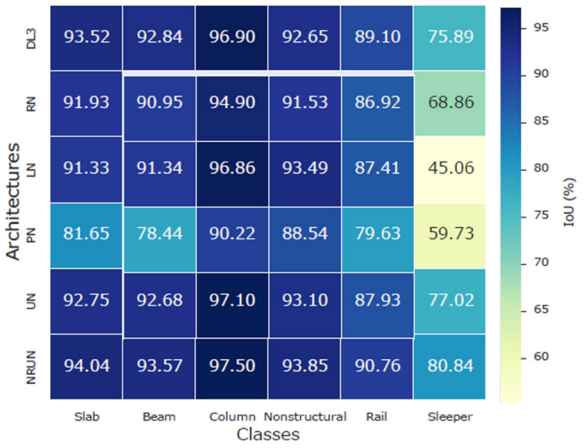

Fig. 10 Per-class comparison of the proposed architecture with other state-of-the-art architectures for the component segmentation task; DL3: DeepLabV3+, RN: RefineNet, LN: LinkNet, PN: PSPNet, UN: UNet, and NRUN: Nested Reg-UNet

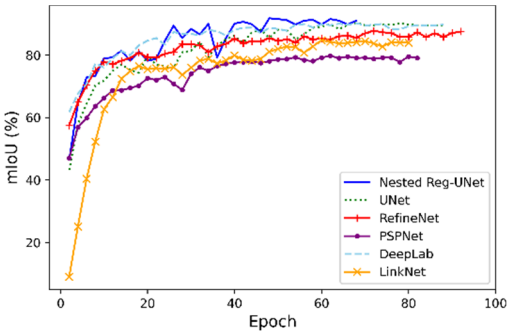

Fig. 11 Component segmentation mIoU vs epoch curve for different architectures on the test set

Table 4 Comparison of different architectures for the component segmentation task. The best result for each metric has been boldfaced

Architecture	Precision (%)	Recall (%)	F1-score (%)	mIoU (%)
DeepLabV3+	95.44	93.95	94.69	90.15
RefineNet	93.86	92.36	93.10	87.52
LinkNet	88.67	94.52	91.50	84.25
PSPNet	89.19	87.60	88.39	79.70
UNet	95.59	93.81	94.69	90.10
Nested Reg-UNet*	**95.97**	**95.30**	**95.63**	**91.76**

*denotes the proposed architecture

11 for the test set. Like in the damage segmentation task, early stopping with a patience of 20 is employed. Compared to Fig. 8, the curves appear to be more uneven, which indicates that the training is less stable for the component segmentation task than for the damage segmentation task. All the architectures converge at different epochs, with the maximum epochs taken by RefineNet and the minimum

Table 5 The results of weighted softmax ensembling for the component segmentation task. The best result for each metric has been boldfaced

Method	Precision (%)	Recall (%)	F1-score (%)	mIoU (%)
Baseline ensemble	94.76	95.54	95.15	90.94
NRUN	**95.97**	95.30	95.63	91.76
NRUN ensemble*	95.77	**96.33**	**96.05**	**92.50**

*denotes the proposed ensemble strategy.
NRUN = Nested Reg-UNet

Table 6 The results of weighted softmax ensembling for the damage segmentation task. The best result for each metric has been boldfaced

Method	Precision (%)	Recall (%)	F1-score (%)	mIoU (%)
Baseline ensemble	88.07	89.97	89.01	81.12
NRUN	90.39	90.16	90.27	83.05
NRUN ensemble*	**90.98**	**91.04**	**91.01**	**84.19**

*denotes the proposed ensemble strategy.
NRUN = Nested Reg-UNet

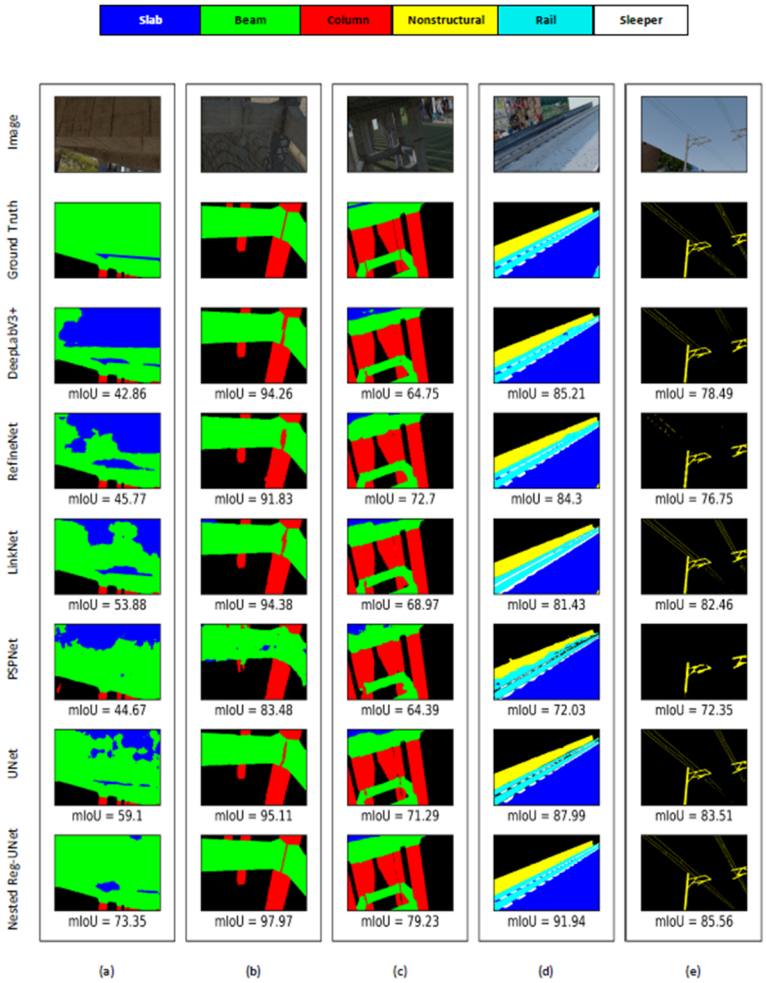

Fig. 12 Sample predictions for the component segmentation task, where each column corresponds to an image and each row either corresponds to the ground truth or the prediction for different architectures as labeled on the left edge. The color map is shown at the top of the figure, while per-image mIoUs are shown under each subfigure for different cases

taken by Nested Reg-UNet.

Several sample predictions produced by different architectures are presented in Figs. 12(a)-(e). Fig. 12(a) depicts a test case showing that the proposed Nested Reg-UNet outperforms the baseline architectures when the input image contains a region that resembles a slab but is, in fact, a beam. The mIoU in this case is 73.35% for Nested Reg-UNet, which is 14.25% higher than the second-best performing architecture, UNet. Moreover, certain patterns are apparent in other predictions — for instance, the joints between different wings of the beams depicted in Figs. 12(b) and (c) are perfectly separated by the Nested Reg-UNet, unlike the baseline architectures. This results in minor improvements in mIoU values. Apart from that, Fig. 12(d) presents an example where some baseline architectures did not preserve the checkerboard pattern between the slab and the sleeper. In contrast, the Nested Reg-UNet perfectly differentiated between the two classes, which can be

quantitatively observed in a performance gain of 3.95% over the second-best performing architecture, UNet. Last but not least, Fig. 12(e) corresponds to an example where the segmentation of the cables is more accurate in the case of the Nested Reg-UNet compared to its counterparts. Here, the improvement is 2.05%. Except Fig. 12(a), where the performance improvement is due to Nested Reg-UNet's ability to properly characterize a beam, a common thread that connects the other examples is that the proposed Nested Reg-UNet is more sensitive to fine details compared to the baseline architectures, which results in its overall superiority in the component segmentation task.

5.3 Weighted softmax ensembling

5.3.1 Component segmentation

As described in Section 3.3, a weighted softmax ensembling strategy is proposed in this study to improve performance. Since ensembling works best when the number of models is large, the best-performing architecture, Nested Reg-UNet, was ensembled by splitting the dataset into multiple cross-validation splits. The dataset was split into four such sets, where 3/4 of the data was set aside for training during each training run. Then, all four runs were tested on a common test dataset. Furthermore, the best-performing result from Table 4 was appended as well. This constituted a total of a five-split ensemble, where four of the splits were chosen using four-fold cross-validation, as stated previously, and one was chosen using a Monte Carlo validation split. The idea of ensembling only five training runs was adopted from the works in Braun *et al*. (2020), Li *et al*. (2021), and Satria *et al*. (2021). However, increasing the number of runs and determining if the same conclusions hold might be the scope of a future study.

The results of the experiment are in Table 5. As can be seen, adopting this ensemble strategy increased the mIoU of the Nested Reg-UNet from 91.76% to 92.50%, which indicates a 0.74% improvement. Furthermore, the role that the types of architectures constituting an ensemble play in the final performance was also studied. In the previous paragraph, the ensemble was constituted entirely of Nested Reg-UNet architectures trained and validated on different data splits. Next, another ensemble was created by combining DeepLabV3+, RefineNet, LinkNet, PSPNet, and UNet from Table 4. This is represented as baseline ensemble in Table 5. As can be seen from the table, an ensemble of these five architectures yielded an mIoU of 90.94%, which indicates a 0.79% improvement over the best result from Table 4 exhibited by DeepLabV3+. This leads to the conclusion that a weighted softmax ensembling can considerably improve network performance. Furthermore, it was observed that the performance of the ensembled baseline is 1.56% lower than the ensembled Nested Reg-UNet (Table 5) and even 0.82% lower than the single-architecture Nested Reg-UNet. Finally, although recall, F1-score, and mIoU of ensembled Nested Reg-UNet are better compared to a single-architecture Nested Reg-UNet, the precision is slightly worse. As we will soon see, this is in stark contrast to the damage segmentation task, indicating that this phenomenon is data specific. This implies that when it comes to segmenting components, the different parts that form the ensemble help identify components present in different regions in some of the images, decreasing the number of false negatives and increasing the recall. But at the same time, this also increases the number of false positives and decreases the precision. Likewise, using a single-architecture Nested Reg-UNet requires less time to train. Therefore, in applications where a high precision is the most important requirement, using single-architecture Nested Reg-UNet is recommended. However, if high recall, F1-score, mIoU are desired, and training time is not a factor, the results will be significantly better if an ensemble is created. Either way, the results bring to the fore the clear superiority of the proposed Nested Reg-UNet not only with regards to the individual baseline architectures but also to an ensemble of them, which is a major contribution of this work.

5.3.2 Damage segmentation

The data splits for the damage segmentation task were done in the same manner as in Section 5.3.1, and the experiments were set up in the same way. That is, a four way cross-validation split was merged with a one-way monte-carlo validation split to compose a total of five data splits for Nested Reg-UNet. Likewise, an ensemble of the remaining models was created as a baseline. The results are presented in Table 6. As can be seen, adopting this ensemble strategy increased the mIoU of the Nested Reg-UNet from 83.05% to 84.19%, which indicates a 1.14% improvement. Furthermore, the ensemble of baseline architectures yielded an mIoU of 81.12%, which is 0.93% higher than the best mIoU among the baseline architectures from Table 3 (UNet). Moreover, the mIoU for ensembled baseline was 1.93% lower than a single architecture Nested Reg-UNet and 3.07% lower than its ensembled version. The superiority of the ensembled Nested Reg-UNet and single architecture Nested Reg-UNet extend to precision, recall, and F1-scores as well. Compared to the baseline ensemble, the single architecture Nested Reg-UNet's precision, recall, and F1-score are 2.32%, 0.19%, and 1.26% higher. Similarly, these numbers are even better for ensembled Nested Reg-UNet: 2.91%, 1.07%, and 2.00% respectively. Just like in the component segmentation experiment, the results here clearly demonstrate that the proposed Nested Reg-UNet improves performance not only with respect to individual baseline architectures, but also with respect to an ensemble of them.

6. Conclusions

In this study, the applicability of some of the most advanced deep learning based semantic segmentation tools are analyzed in the context of segmenting bridge components and damages. To this end, thorough comparative assessments of several combinations of architectures, loss functions, and ensembles are performed. Through a series of experiments, it is demonstrated that the Nested UNet architectural configuration with a RegNet encoder performs significantly better than other state-of-the-art architectures without requiring higher computational resources. With respect to mIoU, for instance, the proposed

Nested Reg-UNet architecture provides an improvement of 2.86% on the damage segmentation task and 1.66% on the component segmentation task when compared to the UNet architecture. Moreover, it is discovered that the incorporation of Lovasz-Softmax loss function outperforms weighted Cross entropy loss by 3.44% on the component segmentation task. Finally, an ensembling strategy called weighted softmax ensembling is found to improve the mIoU by 0.74% in the component segmentation task and 1.14% in the damage segmentation task, when applied simultaneously with the Nested Reg-UNet architecture. Overall, the best results are mIoU of 84.19% for the damage segmentation task and 92.50% for the component segmentation task, which will hopefully help the research community move a step forward when it comes to fully autonomous damage and component segmentation in bridges.

References

Berman, M., Triki, A.R. and Blaschko, M.B. (2018), "The lov´asz-softmax loss: A tractable surrogate for the opti- mization of the intersection-over-union measure in neural networks", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4413-4421.

Braun, T., Spiliopoulos, S., Veltman, C., Hergesell, V., Passow, A., Tenderich, G., Borggrefe, M. and Koerner, M.M. (2020), "Detection of myocardial ischemia due to clinically asymptomatic coronary artery stenosis at rest using supervised artificial intelligence-enabled vectorcardiography–a five-fold cross validation of accuracy", *J. Electrocardiol.*, **59**, 100-105. https://doi.org/10.1016/j.jelectrocard.2019.12.018

Chen, F.-C. and Jahanshahi, M.R. (2017), "NB-CNN: Deep learning-based crack detection using convolutional neural network and naıve bayes data fusion", *IEEE Transact. Industr. Electron.*, **65**(5), 4392-4400. https://doi.org/10.1109/TIE.2017.2764844

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H. (2018), "Encoder-decoder with atrous separable convolution for semantic image segmentation", *Proceedings of the European Conference on Computer Vision* (*ECCV*), pp. 801-818.

Czerniawski, T. and Leite, F. (2020), "Automated segmentation of RGB-D images into a comprehensive set of building components using deep learning", *Adv. Eng. Inform.*, **45**, 101131. https://doi.org/10.1016/j.aei.2020.101131

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009), "ImageNet: A large-scale hierarchical image database." *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248-255.

Ghosh Mondal, T., Jahanshahi, M.R., Wu, R.-T. and Wu, Z.Y. (2020), "Deep learning-based multi-class damage detection for autonomous post-disaster reconnaissance", *Struct. Control Health Monitor.*, **27**(4), e2507. https://doi.org/10.1002/stc.2507

Gorka, J.G. and Armstrong, D.E. (2021), "Application of machine learning to estimate fireball characteristics and their uncertainty from infrared spectral data", *Proceedings of Algorithms, Technologies, and Applications for Multispectral and Hyperspectral Imaging XXVII*, Vol. 11727, pp. 155-166.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y. and He, K. (2017), "Accurate, large minibatch SGD: training imagenet in 1 hour", *CoRR, abs/1706.02677*. http://arxiv.org/abs/1706.02677

He, K., Gkioxari, G., Dollár, P. and Girshick, R. (2017), "Mask R-CNN", *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961-2969.

He, F., Liu, T. and Tao, D. (2019), "Control batch size and learning rate to generalize well: Theoretical and empirical evidence", In: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alch´e-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/dc6a70712a252123c40d2adba6a11d84-Paper.pdf

Hoskere, V., Narazaki, Y., Hoang, T.A. and Spencer Jr, B. (2020), "Madnet: Multi-task semantic segmentation of multiple types of structural materials and damage in images of civil infrastructure", *J. Civil Struct. Health Monitor.*, **10**(5), 757-773. https://doi.org/10.1007/s13349-020-00409-0

Hoskere, V., Amer, F., Friedel, D., Yang, W., Tang, Y., Narazaki, Y., Smith, M.D., Golparvar-Fard, M. and Spencer Jr, B.F. (2021), "Instadam: Open-source platform for rapid semantic segmentation of structural damage", *Appl. Sci.*, **11**(2), 520. https://doi.org/10.3390/app11020520

Hoskere, V., Narazaki, Y. and Spencer Jr, B.F. (2022), "Physics-based graphics models in 3D synthetic environments as autonomous vision-based inspection testbeds", *Sensors*, **22**(2), 532. https://doi.org/10.3390/s22020532

Hou, S., Dong, B., Wang, H. and Wu, G. (2020), "Inspection of surface defects on stay cables using a robot and transfer learning", *Automat. Constr.*, **119**, 103382. https://doi.org/10.1016/j.autcon.2020.103382

Kingma, D.P. and Ba, J. (2014), "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*.

Li, X., Xia, Y., Long, X., Li, Z. and Li, S. (2021), "Exploring text-transformers in AAAI 2021 shared task: Covid-19 fake news detection in English", In: *International Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situation*, pp. 106-115.

Lin, G., Milan, A., Shen, C. and Reid, I. (2017), "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1925-1934.

Mondal, T.G. and Jahanshahi, M.R. (2020), "Autonomous vision-based damage chronology for spatiotemporal con- dition assessment of civil infrastructure using unmanned aerial vehicle", *Smart Struct. Syst., Int. J.*, **25**(6), 733-749. https://doi.org/10.12989/sss.2020.25.6.733

Narazaki, Y., Hoskere, V., Yoshida, K., Spencer, B.F. and Fujino, Y. (2021), "Synthetic environments for vision-based structural condition assessment of japanese high-speed railway viaducts", *Mech. Syst. Signal Process.*, **160**, 107850. https://doi.org/10.1016/j.ymssp.2021.107850

Pan, Y. and Zhang, L. (2022), "Dual attention deep learning network for automatic steel surface defect segmentation", *Comput.-Aided Civil Infrastr. Eng.*, **37**(11), 1468-1487. https://doi.org/10.1111/mice.12792

Pozzer, S., Rezazadeh Azar, E., Dalla Rosa, F. and Chamberlain Pravia, Z.M. (2021), "Semantic segmentation of defects in infrared thermographic images of highly damaged concrete structures", *J. Perform. Constr. Facil.*, **35**(1), 04020131. https://doi.org/10.1061/(ASCE)CF.1943-5509.0001541

Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K. and Dollár, P. (2020), "Designing network design spaces." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428-10436.

Ronneberger, O., Fischer, P. and Brox, T. (2015), "U-net: Convolutional networks for biomedical image segmentation", In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234-241. https://doi.org/10.1007/978-3-319-24574-4_28

Rubio, J.J., Kashiwa, T., Laiteerapong, T., Deng, W., Nagai, K., Escalera, S., Nakayama, K., Matsuo, Y. and Prendinger, H. (2019), "Multi-class structural damage segmentation using fully convolutional networks", *Comput*. *Indust*., **112**, 103121. https://doi.org/10.1016/j.compind.2019.08.002

Satria, A., Sitompul, O.S. and Mawengkang, H. (2021), "5-fold cross validation on supporting k-nearest neighbour accuration of making consimilar symptoms disease classification", *Proceedings of 2021 International Conference on Computer Science and Engineering* (*IC2SE*), Padang, Indonesia, November, pp. 1-5.

Smith, L.N. (2017), "Cyclical learning rates for training neural networks", *Proceedings of 2017 IEEE Winter Conference on Applications of Computer Vision* (*WACV*), pp. 464-472.

Smith, S.L., Kindermans, P. and Le, Q.V. (2018), "Don't decay the learning rate, increase the batch size", In: *International Conference on Learning Representations* (*ICLR*), *abs*/*1711*.*00489*. http://arxiv.org/abs/1711.00489

Spencer Jr, B.F., Hoskere, V. and Narazaki, Y. (2019), "Advances in computer vision-based civil infrastructure inspection and monitoring", *Engineering*, **5**(2), 199-222. https://doi.org/10.1016/j.eng.2018.11.030

Tang, W., Wu, R.-T. and Jahanshahi, M.R. (2022), "Crack segmentation in high-resolution images using cascaded deep convolutional neural networks and bayesian data fusion", *Smart Struct*. *Syst*., *Int*. *J*., **29**(1), 221-235. https://doi.org/10.12989/sss.2022.29.1.221

Wang, N., Zhao, X., Zou, Z., Zhao, P. and Qi, F. (2020), "Autonomous damage segmentation and measurement of glazed tiles in historic buildings via deep learning", *Comput*.*-Aided Civil Infrastr*. *Eng*., **35**(3), 277-291. https://doi.org/10.1111/mice.12488

Xia, T., Yang, J. and Chen, L. (2022), "Automated semantic segmentation of bridge point cloud based on local descriptor and machine learning", *Automat*. *Constr*., **133**, 103992. https://doi.org/10.1016/j.autcon.2021.103992

Yasuno, T., Michihiro, N. and Kazuhiro, N. (2020), "Per-pixel classification rebar exposures in bridge eye inspection", *arXiv preprint arXiv*:*2004*.*12805*.

Zhou, S. and Song, W. (2021), "Crack segmentation through deep convolutional neural networks and heterogeneous image fusion", *Automat*. *Constr*., **125**, 103605. https://doi.org/10.1016/j.autcon.2021.103605

Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N. and Liang, J. (2018), "Unet++: A nested u-net architecture for medical image segmentation", In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pp. 3-11. https://doi.org/10.1007/978-3-030-00889-5_1