MDPI

Article

Neural Network Pruning for Lightweight Metal Corrosion Image Segmentation Models

Firstname Lastname [1,†,‡], Firstname Lastname [2,‡] and Firstname Lastname [2,*]

[1] Affiliation 1; e-mail@e-mail.com
[2] Affiliation 2; e-mail@e-mail.com
* Correspondence: e-mail@e-mail.com; Tel.: (optional; include country code; if there are multiple corresponding authors, add author initials) +xx-xxxx-xxx-xxxx (F.L.)
† Current address: Affiliation.
‡ These authors contributed equally to this work.

Abstract: The threat of metal corrosion is a critical concern across various industries, as it can lead to structural failures, safety risks, and significant economic losses. Therefore, effective inspection is paramount in early detection metal corrosion. Recently, computer vision methods, especially deep learning (DL)-based methods, for aiding visual detection of metal corrosion have been gaining popularity. DL-based methods not only make the inspection more efficient, but also maintain high accuracy in corrosion detection. Although DL-based methods offer promising enhancement to the inspection tasks, one needs to consider its high computational requirements, especially for deploying them in remote areas where only resource-constrained devices, e.g., edge devices are affordable. Therefore, it is essential to develop lightweight DL models that can be deployed on edge devices, ensuring efficient corrosion detection even in resource-constrained environments. This study proposes to develop lightweight DL models for metal corrosion segmentation by pruning the models. The aim of pruning is to remove redundant parameters, thus reducing the size and computational load, without compromising much on performance. In this study, we evaluate five image segmentation models, i.e., U-Net, U-Net++, FPN, LinkNet and MA-Net, and three pruning algorithms, i.e., linear, AGP and movement pruning, on two metal corrosion image segmentation datasets, i.e., NEA and SSCS datasets. The conducted experimental study shows that we can train the models, e.g., FPN, and prune the model up to 90% sparsity with less than 10% IoU reduction on SSCS dataset and less than 5% IoU reduction on NEA dataset.

Keywords: keyword 1; keyword 2; keyword 3 (List three to ten pertinent keywords specific to the article; yet reasonably common within the subject discipline.)

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Mathematics* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

Metal corrosion is an ever-present problem in infrastructure maintenance around the world. If left undetected and unattended, corrosion can cause serious damage to the infrastructure resulting in premature end-of-life of the infrastructures, direct and indirect financial loss, and ultimately poses critical safety risks. It is no wonder that the global cost estimates of corrosion is $25 trillion [1]. Therefore, it is imperative to inspect infrastructures for metallic corrosion regularly. Corrosion detection and maintenance are vital to prevent metallic corrosion [2].

Commonly, corrosion detection is conducted visually by experts in the field. The inspection is also conducted on images collected by unmanned aerial vehicle (UAV), especially on hard-to-reach parts of the structure. Afterward, the collected images are analyzed to detect corrosion. Nowadays, various computer-based methods have been proposed to detect corrosion from these images. The methods range from traditional image processing techniques, e.g., color space-based detection [3] and texture analysis-based detection [4], to

machine learning-based methods, e.g., support vector machine (SVM) [5] and various convolutional neural network (CNN)-based models [6,7]. deep learning (DL)-based methods have been gaining more popular due to its faster process while achieving even pixel-level accuracy.

Various DL-based methods have been proposed to aid structural visual inspection, including metal corrosion detection. Petricca et al. [8] trained an AlexNet to classify whether an image contains rust or not, which performs better than a traditional color-based image segmentation techniques. Atha and Jahanshahi [9] trained a VGG network to classify whether patches of images contain corroded areas. Similarly, Papamarkou et al. [10] trained ResNets [11] to detect corrosion on tiled images of nuclear fuel canisters' surfaces with acceptable performance. Zhang et al. [12] trained an SE-ResNet [13] for rust image segmentation task on an original metal corrosion image dataset collected by conducting salt-spray atmospheric accelerated corrosion test. Nash et al. [14] studies the impact of dataset label quality and size for training an fully convolutional network (FCN) [15] for semantic segmentation of metal corrosion. They found that a large noisy dataset (250 images) is better than a very small expertly labelled dataset (10 images). Afterward, Nash et al. [6] incorporate Bayesian methods into FCN to quantify the uncertainty estimates of the predicted class so that the decision-makers can be more informed about the model's confidence and to improve risk management. Liu et al. [7] proposed a dual (spatial and channel) attention module, and incorporated it into ResNet and feature pyramid network (FPN) for metal corrosion segmentation task. They showed that the proposed dual attention module can improve the performance of the models on both datasets of metal corrosion images in the natural and laboratory environment.

Despite its qualities, one also needs to consider the required computational resources, including the financial cost, networking, and accessibility, in deploying DL-based solutions. This is even more important when one wants to deploy DL-based solutions in remote areas where only edge devices can be afforded. Therefore, to reduce computational resource requirements of these methods, lightweight models can be a solution for these limitations.

In this study, we propose to produce lightweight models for the multiclass image segmentation task for pixel-level metal corrosion detection. We make the models lightweight by pruning the neural network's weights. We propose to prepare the lightweight models by training the models, and pruning their weights, followed by fine-tuning the pruned models. In this study, we experimentally evaluate five neural network architectures, i.e., FPN [16], U-NET [17], U-NET++ [18], multi-scale attention network (MA-Net) [19], and LinkNet [20]. For each of these architectures, we employ 101 layers residual network (ResNet-101) [11] pre-trained on ImageNet dataset as the feature extractor. Three pruning strategies are considered in the pruning stage, namely linear pruning (LP), automated gradual pruning (AGP) algorithm [21], and movement pruning (MP) algorithm [22]. We prune the models up to 90% sparsity and evaluate their performance based on their achieved intersection over union (IoU). The methods are evaluated on two metal corrosion image datasets, i.e., the steel corrosion condition state (SSCS) dataset [23,24] and the naturally eroded wall (NEA) dataset [7].

2. Background

2.1. Architecture

U-NET [17] is an extension to FCN [15] where the architecture comprises two parts, the left contracting path and the right expanding path. The contracting path is a repeated component comprising two 3×3 unpadded convolutions, a rectified linear unit (ReLU), and a 2×2 max pooling layer with stride 2 for downsampling. The number of feature channes is double at each downsampling step. Similarly, the expanding path is also repeated components comprising feature map unsampling, 2×2 convolutions to halve the number of feature channels, a concatenation with the cropped feature map from the contracting part, and two 3×3 convolutions each followed by a ReLU. In total, the network has 23 convolutional layers.

U-Net++ [18] extends U-Net by adding a dense network of skip connections between the contracting and expanding paths. This extension is based on DenseNet [25]. Instead of directly concatenating the feature maps from the contracting path onto the corresponding layers in the exapnsive path, as is done in U-Net, U-Net++ has several skip connections between the corresponding layers. Each skip connection unit takes the features map from the all previous units at the same level and the upsampled feature map for its immediate lower unit. The addition of these skip connections minimize the loss of semantic information between the two paths.

MA-Net [19] also extends U-Net by integrating skip-connections and, more importantly, the self-attention mechanism. Two new blocks based on self-attention mechanism, i.e., position-wise attention block in the bottleneck part (in-between the contracting and expanding paths), and multi-scale fusion attention block in the expanding path, are proposed to capture spatial and channel dependencies of the feature maps. The proposed dual attention mechanism enhances the feature representation ability of the model resulting in better performance of the model for liver and tumor segmentation task compared to other state-of-the-art models, e.g., U-Net, U-Net++, and Densely FCN [26].

Similary to the previous U-Net-based architectures, FPN [16] can also basically be divided into two paths, the bottom-up path, and the top-down path. The bottom-up path computes a feature hierarchy consisting of several scaled feature maps with a scaling step of two. The path comprises several network *stages*, each stage consists of multiple layers producing output maps of the same size. Each stage equals one pyramid level in FPN. The top-down path generates higher resolution features by upsampling the feature maps from the higher pyramid levels, resulting in spatially coarser but semantically stronger feature maps compared to the higher pyramid levels. The resulted features are combined with the feature from the bottom-up path of the same pyramid level through a lateral connection. The design of FPN is flexible in the choice of module for the building blocks. However, the experimental evaluation show by Lin et al. [16] on varying the modules only shows marginal difference in the resulted model's performance. Therefore, similar to the original study, this study opts for the simple design choice.

LinkNet [20] also comprises two parts similar to an encoder-decoder structure. The encoder and decoder are further divided into several levels. The encoder employed in LinkNet is a pre-trained network, e.g., ResNet18 that efficiently extracts high-level features from the input. The decoder upsamples the feature to its original size using transposed convolutions. The output of the encoder is combined with the input to the decoder of the same level via a residual network. The use of lightweight pretrained encoder block and smaller number of parameters compared to, e.g., U-Net-based architectures, results in a more efficient model without majorly sacrificing its performance, making LinkNet suitable for real-time tasks.

2.2. Pruning algorithms

In this study, we evaluate three pruning algorithms, i.e., LP algorithm, AGP algorithm [21], and MP algorithm [22]. Before describing the three algorithms, we briefly discuss the common notation of pruning neural network. Let $\boldsymbol{W}, |\boldsymbol{W}| = d$ be the weight or parameters of a given model, and d is the number of the parameters. A binary mask $\boldsymbol{M}, |\boldsymbol{M}| = d$, is applied to $\boldsymbol{W}$ so that the output of the model becomes:

$$\boldsymbol{y} = (\boldsymbol{W} \odot \boldsymbol{M})\boldsymbol{x}, \tag{1}$$

where $\odot$ is the Hadamard or element-wise product. The zero-ed out elements in $\boldsymbol{W}$ as a result of the mask application is denoted as the pruned weights. The binary mask is commonly computed based on a score $\boldsymbol{S}$ with its element correspond to each element of $\boldsymbol{M}$. One of the method to compute $\boldsymbol{M}$ based on $\boldsymbol{S}$ is:

$$M_i = \begin{cases} 1, & \text{if } S_i \text{ in the top } \zeta\% \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

with ζ is the desired sparsity ratio of the model's parameters.

These pruning algorithms are gradual pruning algorithms, i.e., they increase the sparsity of the model weights gradually. LP is the most basic pruning algorithm where the sparsity of the neural network at hand will be increased linearly throughout the pruning phase. The target sparsity increases linearly as given by:

$$\zeta(t) = \zeta_0 + \frac{t - t_0}{n_p \Delta t} \zeta_f, \tag{3}$$

with $\zeta(t)$ the target sparsity at the t-th training iteration, n_p the total number of pruning steps, Δt the training iteration interval between two pruning steps, t_0 the starting training iteration, ζ_f the final target sparsity, and ζ_i the initial sparsity which is usually $\zeta_0 = 0$. One training iteration means one update step of the model's weights, which commonly comprises one forward and backward pass, and one update step of the optimizer.

AGP is similar to LP, but it uses cubic sparsity schedule to calculate the target sparsity, as given by:

$$\zeta(t) = \zeta_f + (\zeta_0 - \zeta_f)(1 - \frac{t - t_0}{\Delta t n_p})^3. \tag{4}$$

Calculating the target sparsity using (4) results in more pruned weights at the early training iterations when there are expected to be more redundant connections compared to the later pruning stage.

Various immediate pruning methods (as opposed to gradual) can be employed at every pruning steps of LP and AGP. In this study, we use the Taylor pruning method [27]. This method compute the importance of a parameter, say W_i, based on the difference between the prediction errors produced by the model with and without that parameter, as given by:

$$\mathcal{I}_i(W) = (E(\mathcal{D}, \boldsymbol{W}) - E(\mathcal{D}, W \mid W_i = 0))^2. \tag{5}$$

To avoid the expensive computation of evaluating $|\boldsymbol{W}|$ different versions of the model, then the difference can be approximated using the first-order Taylor expansion as:

$$\mathcal{I}_i^{(1)}(W) \triangleq (g_i W_i)^2, \tag{6}$$

with $g_i = \frac{\delta E}{\delta W_i}$ is the i-th element of the gradient $\boldsymbol{g}$. To approximate the joint importance of a set of parameters, say $\mathcal{I}_P^{(1)}(\boldsymbol{W}) \triangleq \sum_{j \in P} \mathcal{I}_j^{(1)}(W)$, with $P = \{j_1, \ldots, j_{|P|}\}, \forall j \in P, 0 \leq j \leq |W|$.

Lastly, we briefly describe MP. While most pruning algorithms retain the parameters that are far from zero, MP retains the parameters that are moving away from zero. Based on this, it can be denoted that MP derived the importance of parameters from the first-order information instead of the zeroth-order information. The score variable $\boldsymbol{S}$ now accumulates this movement of the parameters after t' training iterations as given by:

$$S_i^{(t')} = -\alpha_S \sum_{t<t'} \left(\frac{\delta L}{\delta W_i} \right)^{(t)} W_i^{(t)}, \tag{7}$$

where L is the employed loss function, and $\boldsymbol{W}^{(t)}$ is the model parameters at the t-th training iteration. The schedule of sparsity level for each training iteration in MP is the same as AGP, i.e., as given by (4).

3. Proposed Method

This section describes the proposed framework to train lightweight semantic segmentation models for rust detection, the datasets and the experimental settings. The conducted experimental study is depicted in Figure 1. As previously mentioned, in this study we consider five architectures, namely FPN [16], U-NET [17], U-NET++ [18], MA-Net [19], and

LinkNet [20]. First, we train the models on each dataset. Afterward, the trained models are pruned using the three considered pruning algorithms, i.e., LP, AGP algorithm [21], and MP algorithm [22]. Therefore, at the end of the pruning stage, we obtained 15 pruned models. Each pruned model is then further fine-tuned to try to recover the loss in performance due to the pruning process. Afterward, we evaluate the models performance on each dataset to find the best performing model and pruning algorithm on each dataset.

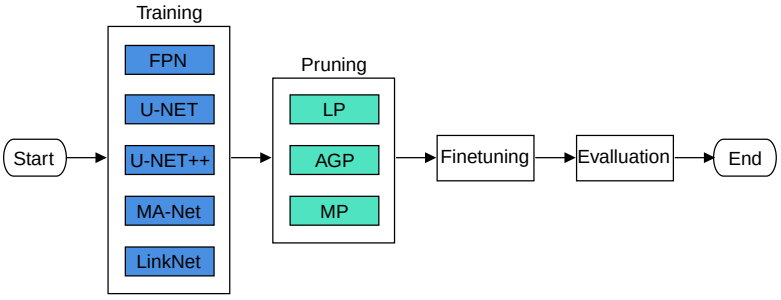


Figure 1. Experimental study flowchart

Throughout all stages, the models are trained to minimize the Tversky loss function [28] as given by:

$$L(\hat{\alpha}, \hat{\beta}) = \frac{\mathrm{TP}}{\mathrm{TP} + \hat{\alpha}\mathrm{FP} + \hat{\beta}\mathrm{FN}}, \tag{8}$$

where TP denotes the true positives, FP the false positives and FN the false negatives. The two hyperparameters of the loss function are $\hat{\alpha}$ and $\hat{\beta}$ that adjust the trade-off between false positives and false negatives. The metric used to evalute the performance of the trained models is the IoU, which is given by:

$$IoU = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}. \tag{9}$$

3.1. Datasets

In this study, the models are trained and evaluated on two metal corrosion image datasets, namely, the SSCS dataset [23,24] and the NEA dataset [7]. SSCS is a set of images from the structural inspection domain, especially from the bridge inspection report by the Virginia Department of Transportation. The dataset comprises 440 annotated steel corrosion condition state images, which are split into 396 training images and 44 testing images. The corrosion state in the images are split into four corrosion class categories, i.e., good (background), fair, poor and severe corrosion state. An example of the image and its corresponding labels from SSCS dataset is depicted in Figure 2. These images are resized into 512×512 images in this study, as is also done in the original paper [24].

NEA dataset comprises images of corroded metal wall in natural environment collected by a UAV. The dataset comprises 292 images of size 1280×720. The annotations comprise three class categories; one no corrosion class and two corrosion classes. Unfortunately ,the difference between the two corrosion classes is not clearly stated. In the original study [7], the authors do not differentiate between the two corrosion class, thus rendering the task as a binary semantic segmentation task. However, in this study, we assume that one class is for poor corrosion state, while the other is for severe corrosion state, as shown in Figure 3. Therefore, this study uses NEA dataset for a multiclass semantic segmentation task.

We can further describe the datasets by looking at their label distribution represented by the total number of pixels for each class. The label distributions for SSCS and NEA dataset are depicted in Figure 4a and Figure 4b, respectively. We can argue that a class imbalance occurs in both datasets even between only the corrosion-related labels. In SSCS

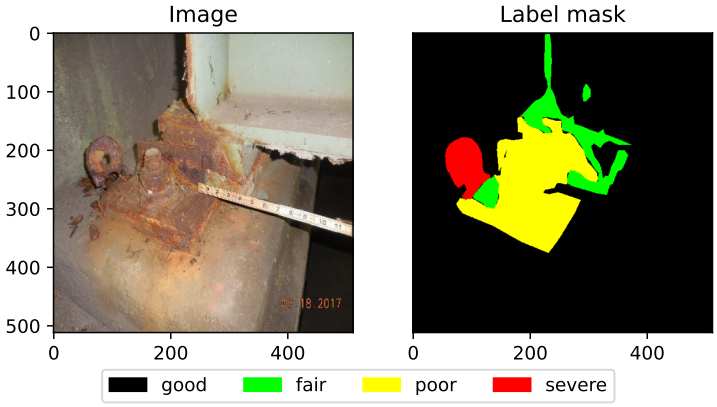


Figure 2. An example of SSCS image and its corresponding label mask

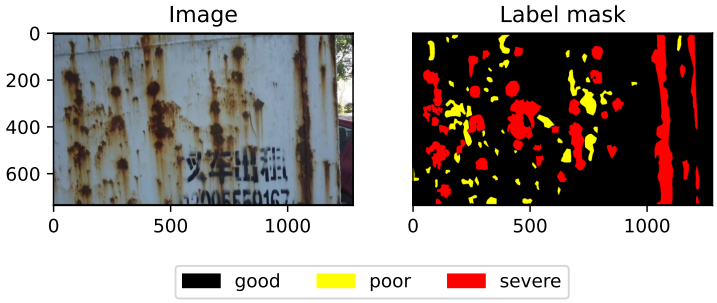
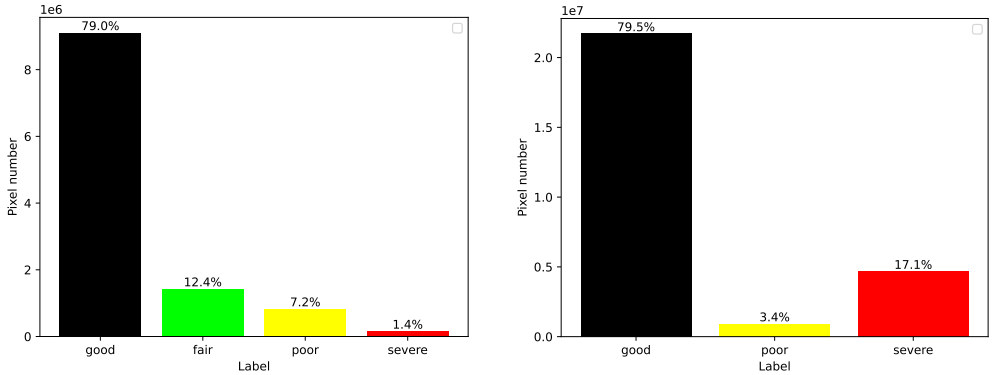


Figure 3. An example of NEA image and its corresponding label mask

dataset, only 1.4% of the total pixels are of the severe corrosion label, while the fair corrosion label occupies most pixels (12.4%), followed by poor corrosion (7.2%). In NEA dataset, the severe corrosion label occupies five times more pixels than the poor label. As a side remark, one can argue that the label "good" is a misnomer. The label "good" can not refer to a metallic surface in good condition because the background pixels are also labeled as good. Therefore, one may also try to split the label "good" into the background and actual metallic surface with good conditions and study the impact of splitting the label on the performance of the trained models. However, we left this attempt for future studies.



(a) SSCS dataset **(b)** NEA dataset

Figure 4. Label distribution

3.2. Experimental settings

All of the experiments are conducted on a single NVIDIA GeForce GTX 2080 Ti and Intel i9-7900X with 20 cores. The source code in Pytorch is available at (github link provided before publishing).

In this experimental study, we use stochastic gradient descent (SGD) with momentum as the optimizer and cyclical learning rate scheduler [29]. The momentum of the SGD optimizer is set to 0.5 and the batch size is set to 4. In addition, the hyperparameters for the cyclical learning rate scheduler is set to $lr_0^{\min} = 0.0001, lr_0^{\max} = 0.01$ and $\delta_t = 2000$. The hyperparameters for the optimizer and the learning rate scheduler are the same for the first training stage, the pruning stage and the fine-tuning stage.

In the first training stage, the models will be trained for $T = 100$ epochs. In the pruning stage, the models will be pruned for $T_{\text{pruning}} = 50$ epochs. Afterward, the pruned models will be fine-tuned for another $T_{\text{tuning}} = 100$ epochs. For MP algorithm, we set $t_i = \frac{20N}{4}$ iterations and $t_f = \frac{10N}{4}$. For the linear and AGP algorithms, the pruning is done gradually in interval of ten epochs, thus the pruning will be done five times in the span of $T_{\text{pruning}} = 50$ epochs.

Throughout all training stages, the training set will be split into two, 80% for the training and 20% for validation. The data augmentation methods applied to the training dataset are affine operations, flipping, rotation, scaling, and random cropping. The target final sparsity we evaluate in this study is $\zeta_f \in \{0.2, 0.5, 0.9\}$. We will evaluate the performance of each combination of model, pruning algorithm, and target sparsity based on the achieved IoU on the testing dataset. In total there are $5 \times 3 \times 3 = 45$ combinations evaluated in this experimental study.

4. Results and Discussion

We describe the progress of the first training stage. The validation IoU obtained by the models throughout the epochs of training is shown in Figure 5. The faded curve is the actual value, while the the bold curve is the exponential moving average values with $\alpha = 0.1$. The figure shows that starting from around the 50-th epoch, the IoU obtained by the models has plateaud. The learning progress of FPN is more stable compared to other models, as it shows smaller variance and it started with higher IoU compared to the other models. On the other hand, other models showed great improvement throughout the training progress as their starting IoU is very small, even less than 0.1 for LinkNet and U-Net on SSCS dataset.

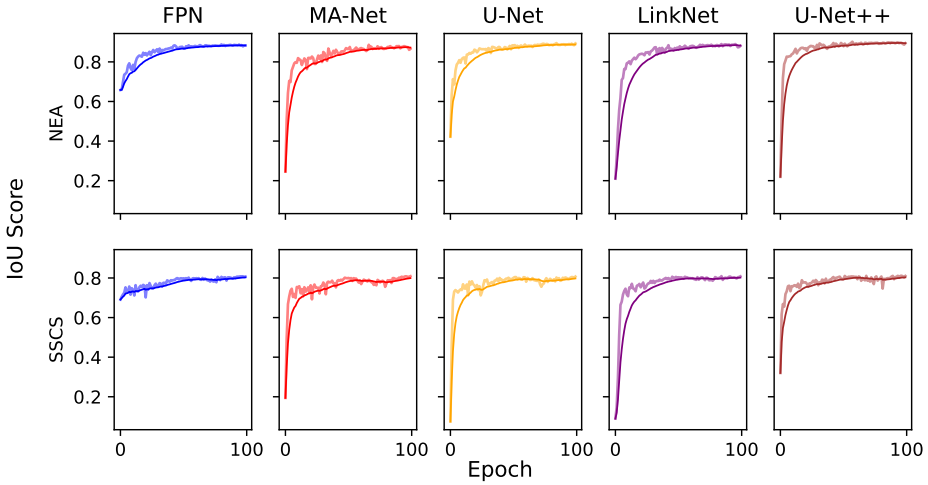
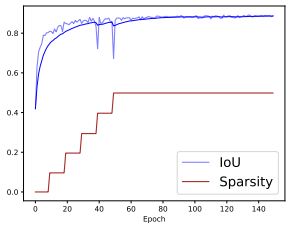


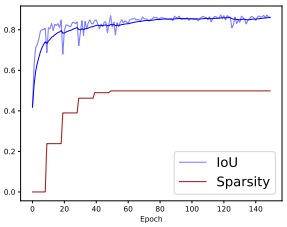
Figure 5. Validation IoU score on the first training stage

Afterward, we briefly describe the pruning progress based on the validation IoU of each combination of model, pruning algorithm and sparsity. We choose several representa-

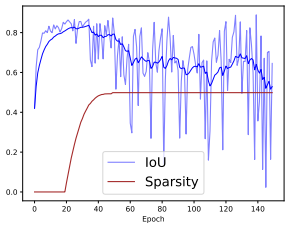
tive figure to be shown in this figure, while the rest are accessible in the Github repository. Figure 6 shows the validation IoU obtained by the models throughout the pruning stage and the fine-tuning stage. As previously described, the fine-tuning starts after 50 epochs of pruning. Similar to Figure 5, Figure 6 also shows the actual and the exponential moving average values.



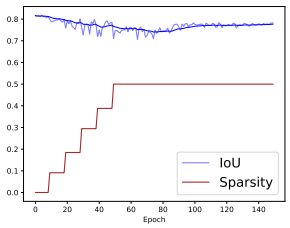
(a) U-Net, LP, and $\zeta_f = 0.5$ for NEA dataset



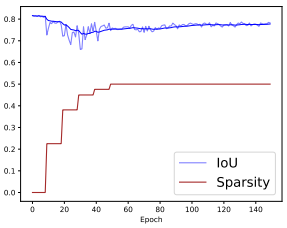
(b) U-Net, AGP, and $\zeta_f = 0.5$ for NEA dataset



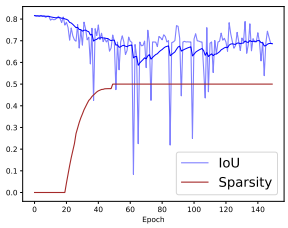
(c) U-Net, MP, and $\zeta_f = 0.5$ for NEA dataset



(d) FPN, LP, and $\zeta_f = 0.5$ for SSCS dataset



(e) FPN, AGP, and $\zeta_f = 0.5$ for SSCS dataset



(f) FPN, MP, and $\zeta_f = 0.5$ for SSCS dataset

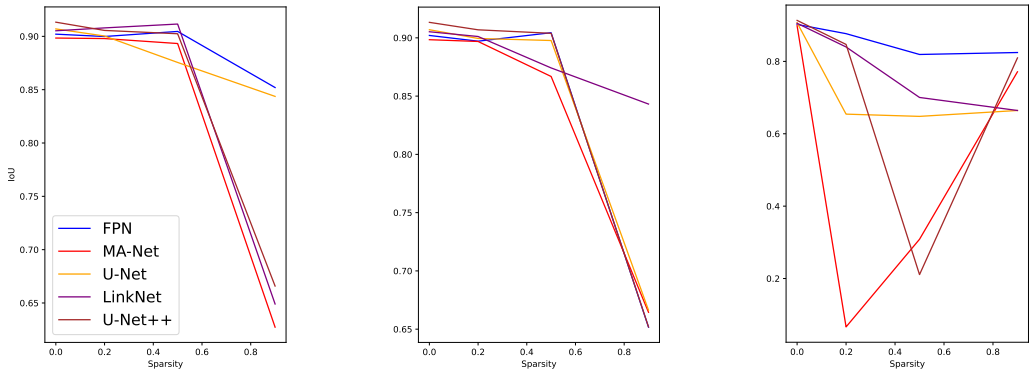
Figure 6. The validation IoU in the pruning and fine-tuning stages

Firstly, we can see that for LP and AGP, the volatility in validation IoU is muchh less compared to that of MP in both models and datasets. For LP and AGP, there is a notable reduction in validation IoU every Δt or when the sparsity increases. However, for ζ_f, the validation IoU can be retained and even improved after the fine-tuning phase. On the other hand, we can notice very high volatility in validation IoU when MP is employed. In addition, these great changes in validation IoU not only occured in the pruning phase, but even more in the fine-tuning phase where the sparsity of the model is retained. Therefore, we can denote that based on the validation IoU, LP and AGP are preferred because they can better retain the models' performance after pruning and have a more stable learning process.

Finally, we will evaluate the performance of each combination of model and pruner based on the achieved IoU on the testing dataset as shown in Figure 7 and Figure 8. Several noteworthy observations can be made on the two figures. Firstly, for AGP and LP, the achieved IoU decreases as ζ_f increases. We can also note that FPN and LinkNet best retain IoU on $\zeta_f = 0.9$ compared to the other models on SSCS dataset with AGP and LP. Meanwhile, on NEA dataset, U-Net and FPN best retain IoU on $\zeta_f = 0.9$ with AGP, while LinkNet and U-Net best retain IoU with LP.

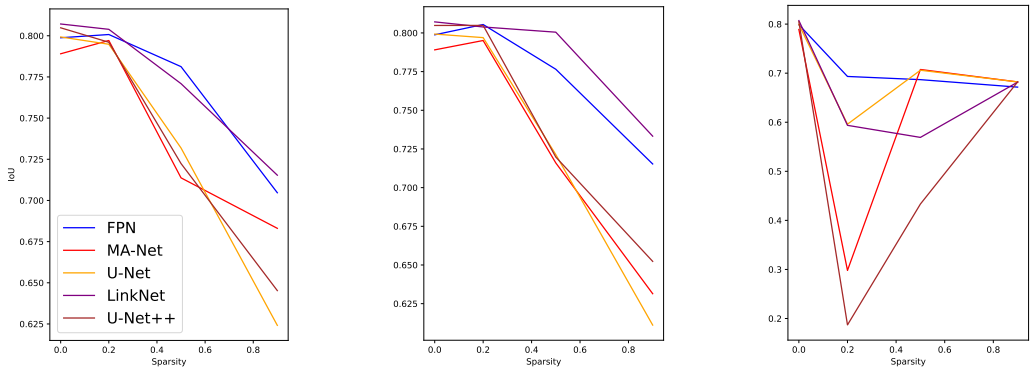
Interestingly, it can be observed on both NEA and SSCS datasets, MP retains better IoU with all models on $\zeta_f = 0.9$ compared to AGP and LP. However, contrary to AGP and LP, the IoU achieved by the models pruned by MP is the lowest on $\zeta_f = 0.2$. Therefore, a further experimental study needs to be conducted to determine the cause of extremely low achieved IoU for MP on $\zeta_f = 0.2$. This experimental study can include hyperparamater tuning for the pruning phase and fine-tuning phase, and a more granular choice of ζ_f. We leave these additional experimental evaluations for our future study.

Lastly, we will show representative examples of the predicted labels generated by the pruned models. Other visualizations are provided in the GitHub repository. Figure 9 shows the labels generated by FPN with varying sparsity with the three pruning methods on NEA dataset. The displayed figures can represent the difference in testing IoU previously



(a) AGP **(b)** LP **(c)** MP

Figure 7. The testing IoU for various pruning algorithms in increasing sparsity on NEA dataset



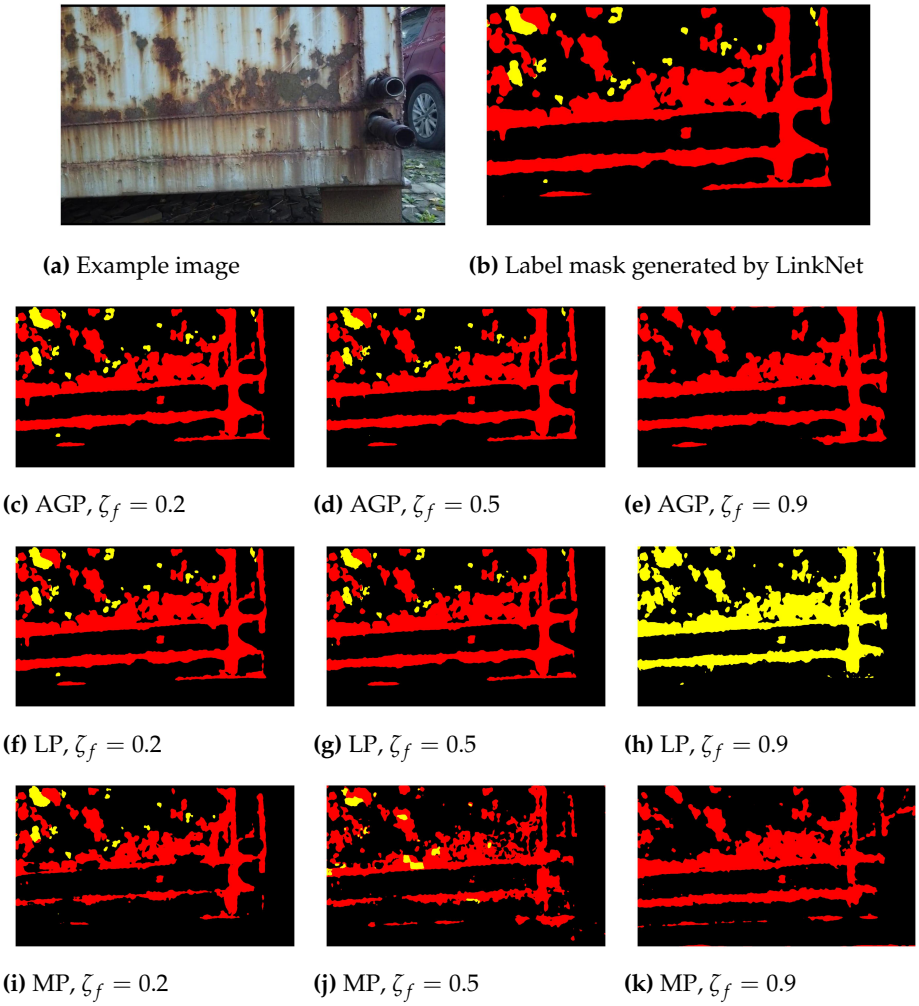
(a) AGP **(b)** LP **(c)** MP

Figure 8. The testing IoU for various pruning algorithms in increasing sparsity on SSCS dataset

described. As can be observed, using AGP or LP, the generated label mask up to $\zeta_f = 0.5$ is still very similar compared to the initial label mask, which is resembled by the testing IoU that is retained up to $\zeta_f = 0.5$. On the other hand, the model pruned with MP with $\zeta_f = 0.5$ generates significantly different label masks, where some poor labels are missing and some severe areas are misclassified as poor areas. For models pruned up to $\zeta_f = 0.9$, where the testing IoU deteriorates significantly, we can observe that all models can only classify the pixels into one class. AGP and MP classified all corroded areas into severe areas, while LP classified all into poor areas. Most corroded areas are severe areas, therefore the models pruned by AGP and MP have more pixels correctly labeled, compared to the model pruned by LP. This explains why LP has the worst testing IoU on $\zeta_f = 0.9$ compared to AGP and MP.

Figure 10 shows the label generated by LinkNet on SSCS dataset. As shown by the figures, the labels generated by LinkNet pruned by AGP and LP up to $\zeta_f = 0.5$ are still quite similar to the labels generated by the un-pruned model. Similar to Figure 9, for $\zeta_f = 0.9$, the models pruned by AGP and LP can only predict two labels. This observation can explain the change in IoU obtained by AGP and LP as ζ_f increases. However, the model pruned by MP fails miserably, even for $\zeta_f = 0.2$. Starting from $\zeta_f = 0.2$ the generated labels still show a little resemblance to the original labels, but we can notice how the background is being mislabeled more and strange Moiré pattern appears. For $\zeta_f = 0.5$, the labels are already random and all over the place. Finally, for $\zeta_f = 0.9$, the model fails to predict any corrosion label. Interestingly, this observation can explain the IoU obtained by MP. The labels are all over the place when $\zeta_f = 0.5$, therefore the IoU is lowest. On the other hand, despite failing to predict any corrosion label at $\zeta_f = 0.9$, its IoU is higher than $\zeta_f = 0.2$ and $\zeta_f = 0.5$. This is because the good label, i.e., background and metallic surface with good condition, has significantly more pixel numbers than corrosion-related labels, as shown in

Figure 4a. Therefore, simply labeling all pixels as good can result in higher IoU on average compared to wrongly mislabelling the corrosion state.



(a) Example image **(b)** Label mask generated by LinkNet

(c) AGP, $\zeta_f = 0.2$ **(d)** AGP, $\zeta_f = 0.5$ **(e)** AGP, $\zeta_f = 0.9$

(f) LP, $\zeta_f = 0.2$ **(g)** LP, $\zeta_f = 0.5$ **(h)** LP, $\zeta_f = 0.9$

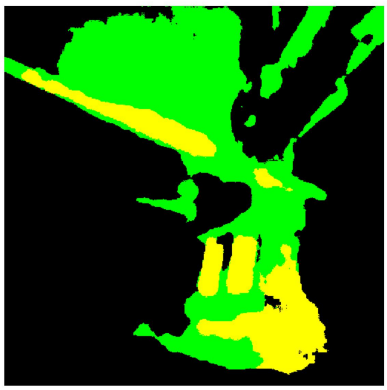
(i) MP, $\zeta_f = 0.2$ **(j)** MP, $\zeta_f = 0.5$ **(k)** MP, $\zeta_f = 0.9$

Figure 9. Example of image and the label mask generated by FPN on NEA dataset

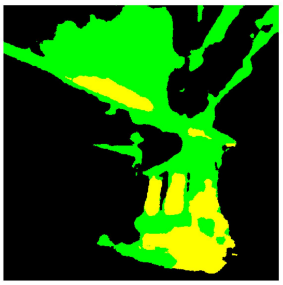
These results indicate that we can produce lightweight metal corrosion segmentation models by the training, pruning, and fine-tuning framework as is done in this study. We have shown that the models, especially FPN, and LinkNet, can be pruned with AGP and LP up to $\zeta_f = 0.5$ without significant reduction in performance. Further study is needed to improve the performance of the models if we aim to prune the models up to $\zeta_f = 0.9$. On the other hand, we have significant performance reduction by pruning the model, e.g., pruning LinkNet by MP on SSCS dataset. This indicates that not all architecture is directly suitable to be pruned by MP. Further study is needed to find the component of the architecture that is too sensitive to be pruned and the component suitable to be pruned.

In addition, the results of LinkNet pruning on SSCS also shows one of the effects of the class imbalance in the dataset. Due to the significant difference in the pixel number between good and corrosion-related labels, a model can achieve a higher IoU on average by not predicting at all rather than slight mislabelling. Therefore, further study must incorporate methods to address the class imbalance, e.g., using weighted loss functions or re-sampling techniques. One other approach to address the class imbalance is by improving the dataset quality. As discussed previously, we can separate the label for the background and the actual metallic surface with good condition, and improve the consistency in the labeling. Such an attempt to extend and improve the dataset quality is both relevant and necessary, given the scarcity of visual structural inspection datasets, including those for metal corrosion inspection [30].

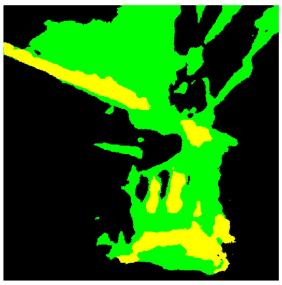
(a) Example image



(b) Label mask generated by FPN

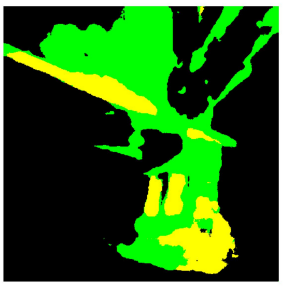


(c) AGP, $\zeta_f = 0.2$

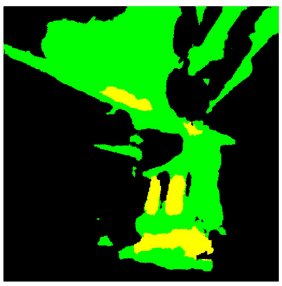


(d) AGP, $\zeta_f = 0.5$

(e) AGP, $\zeta_f = 0.9$

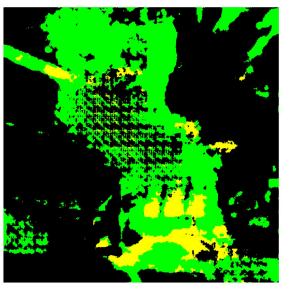


(f) LP, $\zeta_f = 0.2$

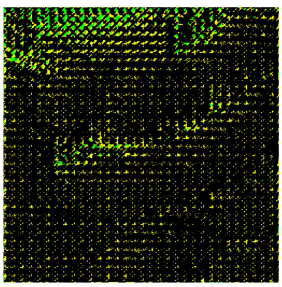


(g) LP, $\zeta_f = 0.5$

(h) LP, $\zeta_f = 0.9$



(i) MP, $\zeta_f = 0.2$



(j) MP, $\zeta_f = 0.5$

(k) MP, $\zeta_f = 0.9$

Figure 10. Example of image and the label mask generated by LinkNet on SSCS dataset

5. Conclusions

In this study, we have trained and evaluated five segmentation models, i.e., FPN, U-Net, U-Net++, LinkNet and MA-Net. The results indicate that for these two datasets, FPN and LinkNet perform better compared to the other three trained models. The models can be pruned, especially with LP and AGP, up to $\zeta_f = 0.5$ or 50% of the parameters are removed with little reduction in performance compared to the original trained model. However, the performance of the model deteriorates significantly when pruned up to $\zeta_f = 0.9$ with

all three pruning algorithms. Significant performance reduction is also observed when using MP with several architectures, e.g., LinkNet, on SSCS dataset. This suggests that further analysis and study are needed to improve the performance of the models if we aim to prune the model up to $\zeta_f = 0.9$. This future study can include hyperparameter tuning for the fine-tuning stage, a more granular choice of ζ_f to find the sweet spot of sparsity and performance, evaluating other pruning algorithms, and addressing the class imbalance in the datasets. In addition, further study can also aim to expand and improve the quality of the limited dataset on metal corrosion image segmentation.

References

1. Koch, G.; Varney, J.; Thompson, N.; Moghissi, O.; Gould, M.; Payer, J. International measures of prevention, application, and economics of corrosion technologies study. *NACE international 216* **2016**.
2. Wang, Z.; LI, Y.; XU, W.; YANG, L.; SUN, C. Analysis of Global Research Status and Development Trends in the Field of Corrosion and Protection: Based on Bibliometrics and Information Visualization Analysis. *Journal of Chinese Society for Corrosion and protection* **2019**, *39*, 201. https://doi.org/10.11902/1005.4537.2018.123.
3. Igoe, D.; Parisi, A.V. Characterization of the corrosion of iron using a smartphone camera. *Instrumentation Science & Technology* **2016**, *44*, 139–147. https://doi.org/10.1080/10739149.2015.1082484.
4. Bonnin-Pascual, F.; Ortiz, A., Corrosion Detection for Automated Visual Inspection; 2014; pp. 619–632. https://doi.org/10.5772/57209.
5. Chen, P.H.; Shen, H.K.; Lei, C.Y.; Chang, L.M. Support-vector-machine-based method for automated steel bridge rust assessment. *Automation in Construction* **2012**, *23*, 9–19. https://doi.org/https://doi.org/10.1016/j.autcon.2011.12.001.
6. Nash, W.; Zheng, L.; Birbilis, N. Deep learning corrosion detection with confidence. *npj Materials Degradation* **2022**, *6*, 26. https://doi.org/10.1038/s41529-022-00232-6.
7. Liu, X.; Luo, Y.; Lu, Y.; Jin, Y.; Vu, Q.V.; Kong, Z. A dual attention network for automatic metallic corrosion detection in natural environment. *Journal of Building Engineering* **2023**, *75*, 107014. https://doi.org/https://doi.org/10.1016/j.jobe.2023.107014.
8. Petricca, L.; Moss, T.; Figueroa, G.; Broen, S. Corrosion Detection Using A.I : A Comparison of Standard Computer Vision Techniques and Deep Learning Model. 05 2016, Vol. 6, pp. 91–99. https://doi.org/10.5121/csit.2016.60608.
9. Atha, D.J.; Jahanshahi, M.R. Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection. *Structural Health Monitoring* **2018**, *17*, 1110–1128. https://doi.org/10.1177/1475921717737051.
10. Papamarkou, T.; Guy, H.; Kroencke, B.; Miller, J.; Robinette, P.; Schultz, D.; Hinkle, J.; Pullum, L.; Schuman, C.; Renshaw, J.; et al. Automated detection of corrosion in used nuclear fuel dry storage canisters using residual neural networks. *Nuclear Engineering and Technology* **2021**, *53*, 657–665. https://doi.org/https://doi.org/10.1016/j.net.2020.07.020.
11. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.
12. Zhang, S.; Deng, X.; Lu, Y.; Hong, S.; Kong, Z.; Peng, Y.; Luo, Y. A channel attention based deep neural network for automatic metallic corrosion detection. *Journal of Building Engineering* **2021**, *42*, 103046. https://doi.org/https://doi.org/10.1016/j.jobe.2021.103046.
13. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141. https://doi.org/10.1109/CVPR.2018.00745.
14. Nash, W.; Drummond, T.; Birbilis, N. Deep learning AI for corrosion detection. In Proceedings of the Corrosion 2019. NACE International, 2019, NACE - International Corrosion Conference Series. NACE International - Corrosion 2019 ; Conference date: 24-03-2019 Through 28-03-2019.
15. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, jun 2015; pp. 3431–3440. https://doi.org/10.1109/CVPR.2015.7298965.
16. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 936–944. https://doi.org/10.1109/CVPR.2017.106.

17. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015; Navab, N.; Hornegger, J.; Wells, W.M.; Frangi, A.F., Eds., Cham, 2015; pp. 234–241.
18. Zhou, Z.; Rahman Siddiquee, M.M.; Tajbakhsh, N.; Liang, J. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. In Proceedings of the Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support; Stoyanov, D.; Taylor, Z.; Carneiro, G.; Syeda-Mahmood, T.; Martel, A.; Maier-Hein, L.; Tavares, J.M.R.; Bradley, A.; Papa, J.P.; Belagiannis, V.; et al., Eds., Cham, 2018; pp. 3–11.
19. Fan, T.; Wang, G.; Li, Y.; Wang, H. MA-Net: A Multi-Scale Attention Network for Liver and Tumor Segmentation. *IEEE Access* **2020**, *8*, 179656–179665. https://doi.org/10.1109/ACCESS.2020.3025372.
20. Chaurasia, A.; Culurciello, E. LinkNet: Exploiting encoder representations for efficient semantic segmentation. In Proceedings of the 2017 IEEE Visual Communications and Image Processing (VCIP), 2017, pp. 1–4. https://doi.org/10.1109/VCIP.2017.8305148.
21. Han, H.G.; Zhang, S.; Qiao, J.F. An adaptive growing and pruning algorithm for designing recurrent neural network. *Neurocomputing* **2017**, *242*, 51–62. https://doi.org/https://doi.org/10.1016/j.neucom.2017.02.038.
22. Sanh, V.; Wolf, T.; Rush, A. Movement Pruning: Adaptive Sparsity by Fine-Tuning. In Proceedings of the Advances in Neural Information Processing Systems; Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; Lin, H., Eds. Curran Associates, Inc., 2020, Vol. 33, pp. 20378–20389.
23. Bianchi, E.; Hebdon, M. Corrosion Condition State Semantic Segmentation Dataset **2021**. https://doi.org/10.7294/16624663.v2.
24. Bianchi, E.; Hebdon, M. Development of Extendable Open-Source Structural Inspection Datasets. *Journal of Computing in Civil Engineering* **2022**, *36*, 04022039. https://doi.org/10.1061/(ASCE)CP.1943-5487.0001045.
25. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269. https://doi.org/10.1109/CVPR.2017.243.
26. Kaluva, K.C.; Khened, M.; Kori, A.; Krishnamurthi, G. 2D-Densely Connected Convolution Neural Networks for automatic Liver and Tumor Segmentation, 2018, [arXiv:cs.CV/1802.02182].
27. Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; Kautz, J. Importance Estimation for Neural Network Pruning. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 11256–11264. https://doi.org/10.1109/CVPR.2019.01152.
28. Salehi, S.S.M.; Erdogmus, D.; Gholipour, A. Tversky Loss Function for Image Segmentation Using 3D Fully Convolutional Deep Networks. In Proceedings of the Machine Learning in Medical Imaging; Wang, Q.; Shi, Y.; Suk, H.I.; Suzuki, K., Eds., Cham, 2017; pp. 379–387.
29. Smith, L.N. Cyclical Learning Rates for Training Neural Networks. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), 2017, pp. 464–472. https://doi.org/10.1109/WACV.2017.58.
30. Bianchi, E.; Hebdon, M. Visual structural inspection datasets. *Automation in Construction* **2022**, *139*, 104299. https://doi.org/https://doi.org/10.1016/j.autcon.2022.104299.