Twin Models for High-Resolution Visual Inspections

Seyedomid Sajedi[1*], Kareem Eltouny[1+]

[1]*Ph.D. Candidate, Buffalo, New York, United States*

ABSTRACT

Visual structural inspections are an inseparable part of post-earthquake damage assessments. Camera-equipped unmanned aerial vehicles (UAVs) are becoming a new standard in the industry, assisting human inspectors by collecting massive amounts of visual data. There has been significant research on deep learning computer vision applications for autonomous inspections in recent years. One of the critical tasks in this domain is semantic segmentation. UAVs can capture multi-view, high-resolution images of affected structures to evaluate their integrity and localize damage through a post-processing stage. However, the segmentation of high-resolution images using deep learning methods is significantly challenging due to high memory computational demands. The standard solution is to uniformly downsample the raw images at the price of losing fine local details, such as fine thin cracks. On the other hand, cropping smaller parts of the images can cause a loss of global contextual information. We propose **T**rainable **R**esizing for high-resolution **S**egmentation **Net**work (TRS-Net) and DmgFormer as twin models approaching the global and local semantics from different perspectives. TRS-Net is a compound, high-resolution deep learning architecture equipped with learnable downsampler and upsampler modules in its outer level to minimize information loss for optimal performance and efficiency. It also includes an attention-based internal segmentation model which handles low-resolution data efficiently. DmgFormer utilizes a transformer backbone and a convolutional decoder head with skip connections on a grid of crops aiming for high precision learning without downsizing. An augmented inference technique is used to boost the performance further and reduce the possible loss of context due to grid cropping. Comprehensive experiments have been performed on the 3D physics-based graphics models (PBGMs) synthetic environments in the QuakeCity dataset [1]. The proposed frameworks are evaluated using several metrics on three segmentation tasks: I) component type, II) component damage state and III) global damage (crack, rebar, spalling) as part of IC-SHM 2021. TRS-Net (Task I & II) and DmgFormer (Task III) are proposed as candidate models for University at Buffalo's Structural Health Monitoring Team (UB-SHM). Code will be released at: github.com/OmidSaj/UB-Twin-Vision.

INTRODUCTION

There is no doubt that recent breakthroughs in deep learning, accompanied by advances in computing and sensing technology, have significantly impacted Structural Health Monitoring (SHM) research, especially in autonomous visual inspections [1, 2]. Camera-equipped Unmanned Aerial Vehicles (UAVs) are becoming more available in the inspection industry and assist in collecting high-resolution data from areas that are difficult or unsafe to access by human inspectors. Furthermore, deep

[*]Corresponding Author

Email: ssajedi@buffalo.edu; phone: 716-444-3997; fax: 716-645-3733

[+]Both authors worked in close collaboration and contributed equally to this project

learning technology allows to post-process massive amounts of visual data obtained from drone footage in the presence of reliable models. A common approach in dealing with the SHM vision problem is to adopt off-the-shelf deep learning algorithms and use them with techniques such as transfer learning or training from scratch. While this approach can be a working solution in many tasks, there are delicacies in structural inspections that need to be considered. Autonomous visual inspections need to have high precision, accuracy, and computational efficiency.

Limited resources often dictate downsizing images drastically for different tasks such as semantic segmentation. However, critical information might be lost, especially in the presence of structural damage. The three tasks in project 2 of IC-SHM 2021 include distant images, simulating drone footage from buildings experiencing different types of façade damage. The QuakeCity dataset [1] is a great example of both computational and performance challenges that stem from the need for high-resolution images. Our team at the University at Buffalo (UB-SHM) aimed to tackle these challenges from two perspectives, given the different nature of the three tasks in the competition. TRS-Net is developed for component detection and damage severity segmentation, which highly rely on global context information. DmgFormer is dedicated to crack, rebar, and spalling segmentation, tasks that are highly sensitive to the loss of resolution. The proposed twin models are customized based on the latest advances in deep learning computer vision research on super-resolution, image segmentation, and transformer architectures.

The remainder of this report is organized as follows. The following two sections provide insights on TRS-Net and DmgFormer architecture and network design optimization. After discussing the theory and notions behind each model, a section is dedicated to the implementation details of both models. A results section is subsequently provided to document the testing evaluation metrics. The last part of this report concludes the benefits and limitations of each model for different vision tasks.

COMPONENTS DETECTION AND DAMAGE STATE SEGMENTATIONS (TASKS 1 & 2)

The majority of the deep learning semantic segmentation models are optimized for low to medium resolution images dataset [2]. When dealing with high-resolution images, it is common to uniformly downsample high-resolution images, as a preprocessing step, to be fed into segmentation models for training and inference [3]. The predicted masks can be either used in their downsized versions or upsampled to the original size. The down/upsampling operations are typically performed uniformly or by using an interpolation method (e.g., nearest neighbor). However, these operations result in poor segmentation performance near the object edges as a result of giving equal importance to all pixels in the image. This problem is even more severe for damage segmentation in task 3. Moreover, if the segmentation networks (e.g., U-Net) are to operate directly on the high-resolution images, the model is met by GPU-memory constraints and could become computationally inefficient.

In the first two tasks of this project, we aim to provide high fidelity segmentation masks with precise edges for buildings components and their damage states. At the same time, we also aim to maintain the efficiency of low-resolution segmentation models and satisfy GPU-memory constraints. To achieve this, we propose trainable resizing for high-resolution image segmentation. It is a compound segmentation model with two-layered encoder-decoder networks (Fig. 1). The outer encoder-decoder network includes efficient trainable downsampler and upsampler modules. Their primary goals are to learn to focus on the important regions and pixels during the downsizing process and also reconstruct the high-resolution masks. TRS-Net also has an internal encoder-decoder which is a U-Net-style semantic segmentation model trained to predict pixel-wise classification for low-resolution images and masks. The internal segmentation model has a backbone based on the ResNeSt [4] and a U-Net++ decoder [5]. Both the outer and internal encoder-decoder stages are directly connected and are trained as a single model end-to-end. As an example of TRS-Net memory efficiency, we found that a simple U-

Net segmentation model with a ResNet50 backbone needs 8.4 GB of GPU memory for a single forward/backward pass and batch size of one image. On the other hand, our TRS-Net, with its more advanced internal segmentation model, requires 1.6 GB of GPU memory.

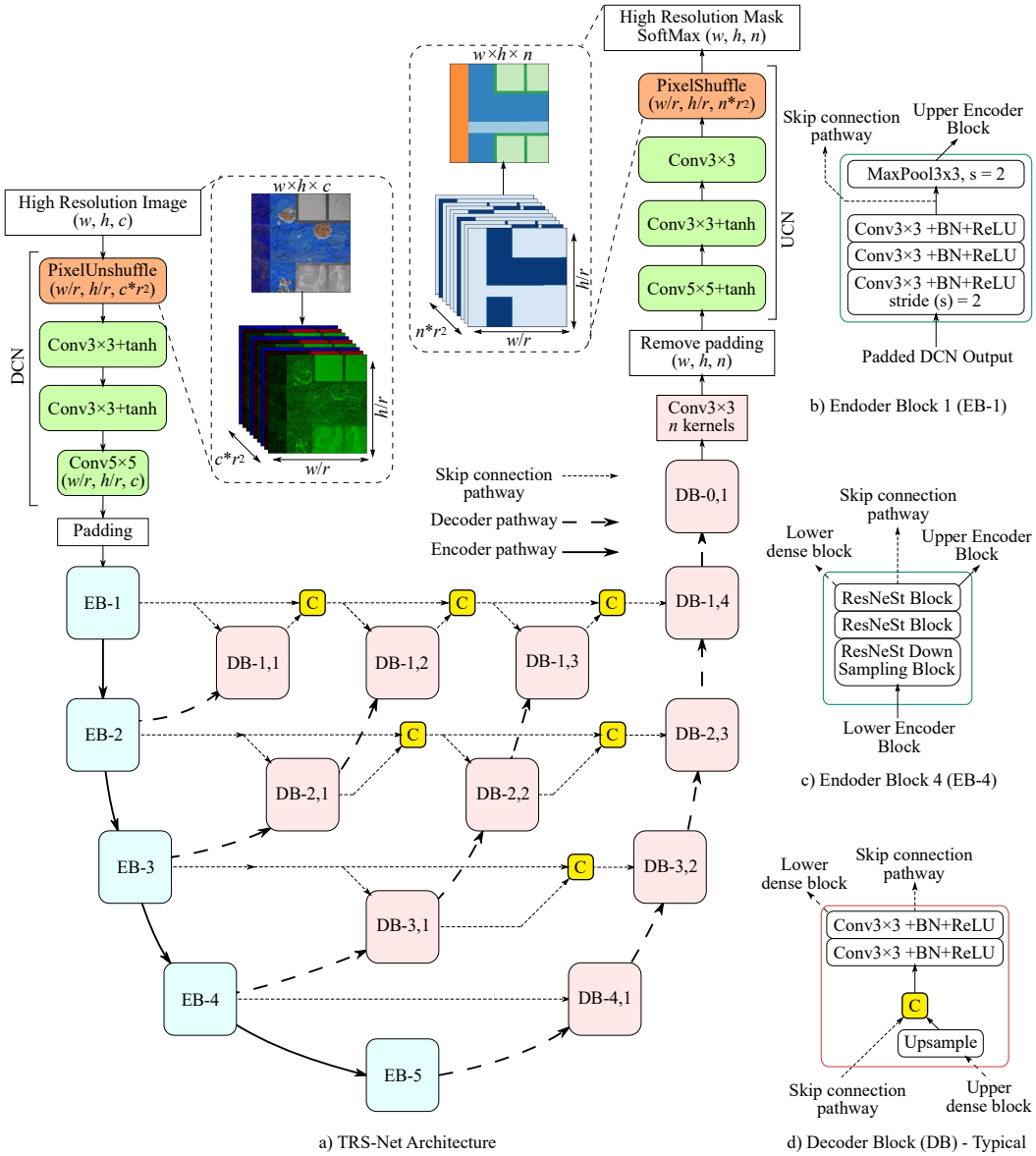
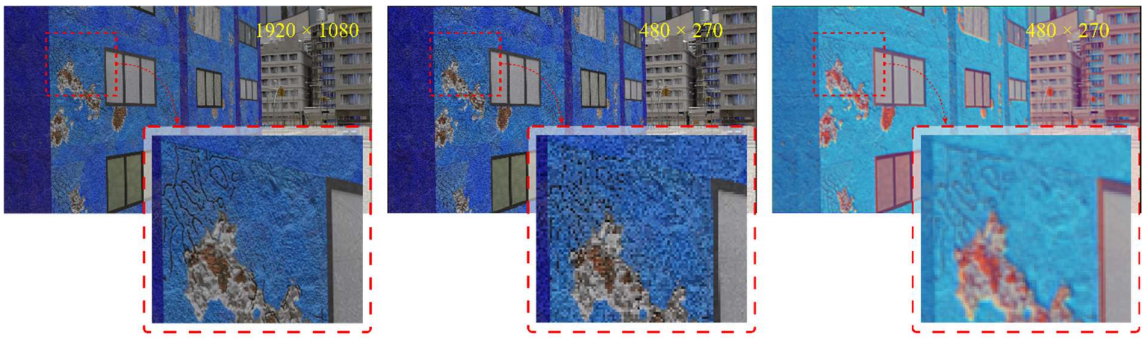


Fig. 1. TRS-Net architecture. (Conv: 2D convolution, BN: batch normalization, ReLU: rectified linear unit, EB-*i* encoder block, DB-*i*,*j*: decoder block, *c*: RGB channels, *n*: classes, *r*: scaling factor)

Learning to resize images and masks

The outer part of the model is inspired by the advances made in image super-resolution networks, in particular, the efficient sub-pixel convolution network [6]. The upsampler, called thereafter upsampling convolutional network (UCN), is composed of three singe-stride convolution layers followed by a sub-pixel convolution layer. The details of each layer are shown in Table 1. The sub-pixel convolution layer, also known as pixel shuffle, aggregates the low-resolution (480, 270, $n\times4^2$) feature maps to form four-times upscaled masks (480×4, 270×4, n) corresponding to the number of classes (n). A final activation, such as SoftMax, is attached after the pixel shuffle layer depending on the

classification task. We also propose a downsampling convolutional network (DCN) that is similar to an inverted version of UCN. It starts with pixel unshuffled, a reverse operation to pixel shuffle, ensuring that no pixels are lost in the downsizing process but are rather arranged in the channels dimension. This layer is again followed by three convolutions, with the last one providing 3 feature maps for the internal segmentation model.



a) Original high resolution image b) Uniformly downsampled image c) Our downsampler module

Fig. 2. An example comparison between different downsampling techniques. Our trainable downsampler (c) can retain most of the fine details, such as cracks, as well as edges that are usually distorted or lost when using a uniform downsampling operation.

Table 1. DCN and UCN specification

	DCN (Downsampler)			UCN (Upsampler)		
Layer #	Operator	Kernel	# Channels	Operator	Kernel	# Channels
1	PixelUnshuffle	4×4	48	Conv.	5×5	64
2	Conv.	3×3	32	Conv.	3×3	32
3	Conv.	3×3	64	Conv.	3×3	$n \times 4^2$
4	Conv.	5×5	3	PixelShuffle	4×4	n

*n is the number of classes/masks

To demonstrate the benefits of using the trainable resizing modules, we show a comparison between the commonly used non-trainable uniform downsampling process and our proposed DCN on an example image (Fig. 2). Compared to the original high-resolution image in Fig. 2.a, there is a significant loss of valuable information when performing a quarter-scale uniform downsampling. The cracks are harder to detect in the zoomed-in region in Fig 2-b, and the window edges are significantly distorted. This makes it very difficult for the internal segmentation model to provide high-quality components detection and damage-state predictions. Using DCN (Fig 2.c), the downsampler can learn the importance of these pixels and retain them in a meaningful way in low-resolution. In addition, it can also emphasize some of the important edges in the downsized image, which are even harder to detect in the original image. A good example would be the upper beam edge in the focused region in Fig. 2. This is also seen in the cracks as they appear thicker in our DCN module than in the original image. This characteristic makes DCN equivalent to a non-uniform sampler that has a warped sampling grid with dense sampling near relevant pixels compared to others.

Split Attention Blocks (ResNeSt)

One of the main building blocks of the network is the ResNeSt (Fig. 3). ResNeSt is a family of state-of-the-art image classifier networks with ResNet-like architectures [7]. ResNeSt was designed with efficiency in mind and had a similar inference speed as EfficientNet [8], with slightly improved classification performance. Based on the concept of feature map groups (also known as cardinality),

ResNeSt splits the feature maps of the first two convolutional layers into $s \times k$ groups where s and k represent the number of splits (radix) and cardinals (cardinality), respectively. They are organized such that every k cardinals are grouped into a single split. The feature maps of all cardinals in a split are concatenated, and the s splits feature maps are aggregated through summation. Based on the Squeeze and Excitation (SE) layer from SE-Net [9], the feature maps pass through a split-attention layer which calibrates the splits feature maps. The primary purpose of the split-attention module is to capture the global context of the input by learning the channels interdependencies. More details on the split-attention module can be found in Zhang et al. [4]. The split feature map groups are multiplied by the channel-wise split-attention factors and then aggregated through summation. Finally, and after the 1×1 convolution, a skip connection from the input combines the input with the output by addition.

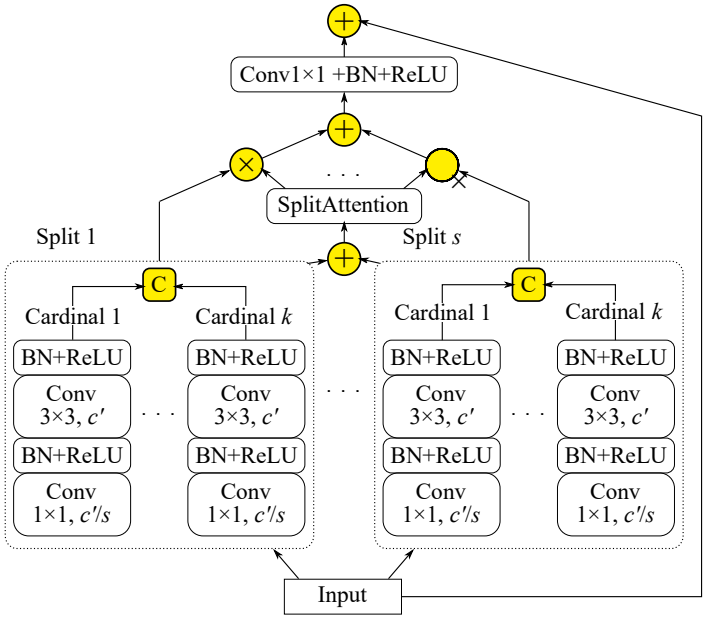


Fig. 3. Split Attention (ResNeSt) Block – reproduced from [4] (Conv: 2D convolution, BN: batch normalization, ReLU: rectified linear unit, *c'*: cardinal width, *k*: cardinality, *s*: radix).

Internal Segmentation Model

The internal encoder-decoder is a low-resolution segmentation model based on a U-Net++ decoder, implemented in [10], with a variant of ResNeSt50d encoder. The network includes multiple pathways that can be categorized into encoder, decoder, and skip-connection pathways. It is also organized into five encoder blocks (EB-*i*) and 11 decoder blocks (DB-*i*,*j*), where *i* indicates the encoder stage and *j* indexes the decoder's dense block layer. Details for each block are in Table 2. The main difference between the used U-Net++ architecture and the original U-Net is the complex dense convolutional blocks added to the skip-connections from the encoder and decoder layers. Furthermore, additional decoder pathways are added, connecting the dense blocks with the upper encoder/decoder stage. The added convolutional layers and dense connections along the skip connections pathway can bridge the semantic gap between the encoder and decoder and improve gradient flow [5]. The encoder pathway passes through five stages of encoder blocks EB-1 to EB-5. Because the input images need to be downsized by half at each stage, the image output of DCN is not compatible. For this reason, we zero-pad the DCN output with 18 rows which are also removed from the predicted mask before the UCN upsampling procedure.

In this network, we adopt the ResNeSt50d_4s2x40d variant, implemented in [11], with a cardinality of two, four splits and a cardinality base width of 40 channels. This variant offers improvement compared to the original ResNeSt50d with a comparable number of parameters. The general architecture

of the encoder stages follows the ResNetD architecture proposed in [12]. Aside from EB-0, each encoder is built from several consecutive ResNeSt blocks. Starting from EB-2, a ResNeSt downsampling block replaces the regular block as a first layer. In this block, global average pooling layers are added before the final convolution layer as well as in the input skip connection reducing the spatial dimensions while increasing the channel count. The decoder blocks receive their input from the skip-connection pathway (except for DB-0,1) and the decoder pathway. In each block, the inputs are concatenated after upsampling the decoder pathway input (using nearest neighbor) and are then passed to two convolutional layers. The decoder head includes a single convolutional layer that provides a set of feature maps corresponding to the number of classes. All convolutional layers in the internal segmentation model are followed by a batch normalization layer and a Rectified Linear Unit (ReLU) activation.

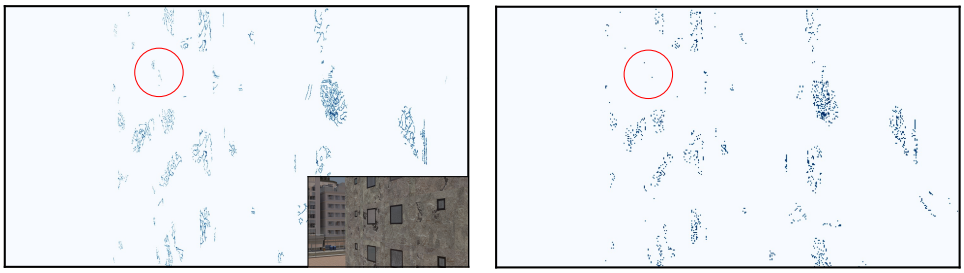
Table 2. TRS-Net Internal encoder-decoder specifications

	Encoder				Decoder # Kernels		
Stage	Operator	# Channels	Cardinals width	# Layers	Last DB	Skip connection DB	# Spatial dim.
Head	-	-	-	1	*n* - 1 layer	-	480×288
0		-	-	1	16	-	480×288
1	Conv 3×3	64	-	3	32	64	240×144
2	ResNeSt	256	40	3	64	256	120×72
3	ResNeSt	512	160	4	128	512	60×36
4	ResNeSt	1024	320	6	256	-	30×18
5	ResNeSt	2048	640	3	-	-	15×9

**n* is the number of classes/masks

DAMAGE SEGMENTATION (TASK 3)

The main challenge in task 3 is that the loss of information is often significant after the downsizing of images. It is possible that after recovering the original resolution, pixel information on damage patterns such as thin cracks or exposed rebars is lost. This issue is more critical when the camera location is distant from the building facade. An example of this phenomenon is illustrated in Fig. 4.



a) Original size crack mask and image b) Resampled crack mask

Fig. 4. Loss of damage information after downsizing the image to 25% of its dimensions and subsequently upsampling to the original resolution.

While the downsampler and upsampler networks presented earlier might alleviate this problem, we also investigated a partially loss-less approach using the full resolution images. To this end, each image is exploded into a grid of smaller crops. Cropping has both pros and cons for the segmentation tasks. For example, in component detection (task-1), some context information can be lost if the whole

building, including beams, walls, etc., is not present in the model input. Our experiments showed that for damage segmentations in task 3, this loss of information due to cropping is insignificant. Nevertheless, cropping results in a) substantial computational gain that made it possible for us to train more advanced and deeper neural network models in task 3, and b) having high-resolution damage patterns, especially for cracks.

The 1920×1080 images are initially zero-padded (2016×1120) and subsequently cropped into a 5×9 grid of 224×224 RGB images. The deep learning model will receive a batch of these crops for segmentation. As an augmentation technique, we will add random noise translation to the centroid of each crop during training. Perimeter crops that fall outside the vertical or horizontal borders of the padded image are shifted to the border with respect to the direction of trespassing. Fig. 5 demonstrates this process.

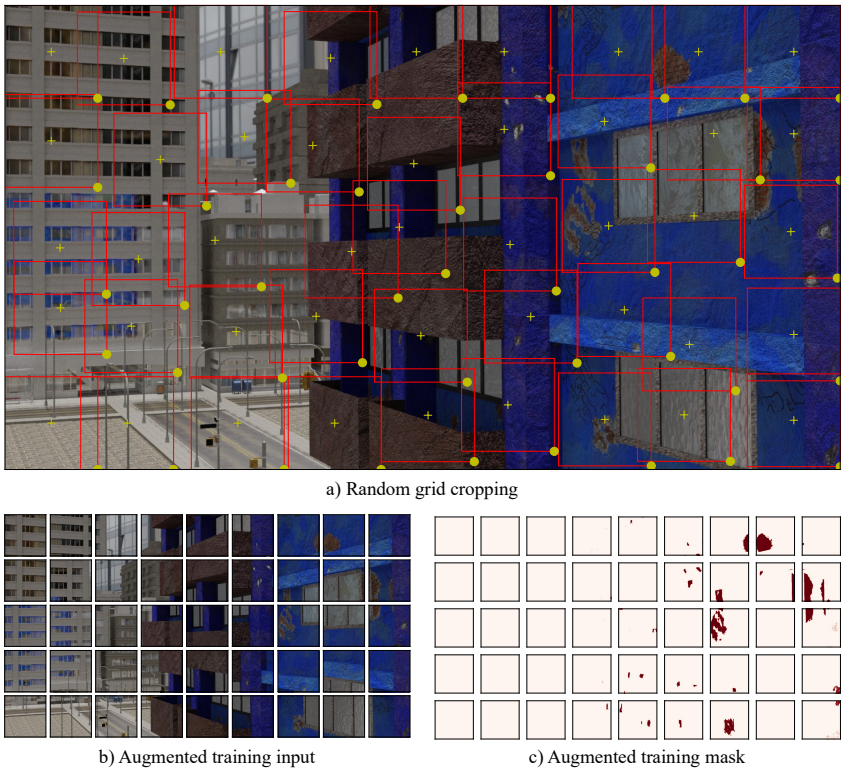
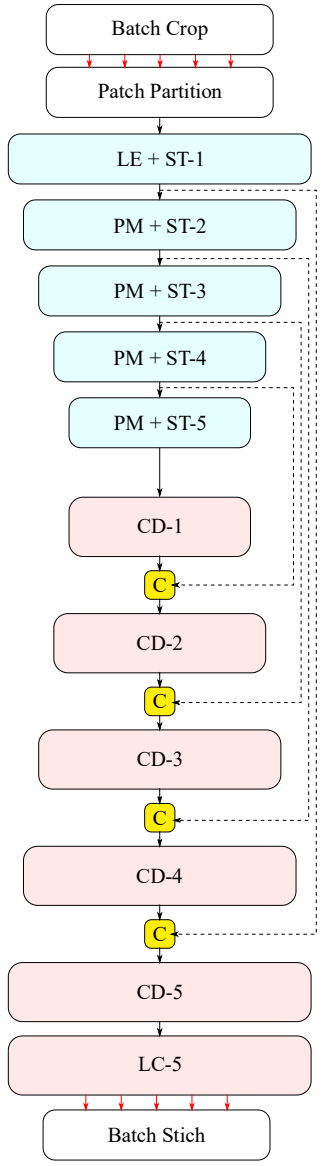


Fig. 5. Demonstration of Random grid cropping to augment the training dataset. Each crop's centroid and bottom right corner are depicted with + and o markers.

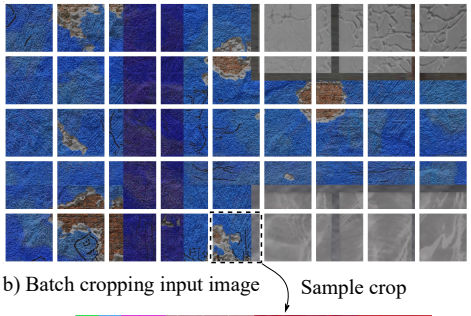
Transformers

The deep learning model for damage segmentation benefits from a transformer backbone architecture. Transformers were initially introduced in natural language processing [13] and are currently state of the art in speech recognition [14]. The core module in a transformer is the multi-head attention that enables effective learning from a sequence. Vision Transformer (ViT) is a more recent type of deep learning architecture that is currently competing with the well-established convolutional neural networks (CNN) [15, 16]. In fact, the family of ViTs has dominated the top-10 in many benchmark datasets such as ImageNet and ADE20K [17]. We introduce DmgFormer for task 3 of project 2, a customized transformer model for damage segmentation (Fig. 6).

Most common and successful examples of segmentation networks include but are not limited to Encoder-decoder CNNs such as U-Net, FCN, and DeepLabv3. The presence of several connections between the encoder and decoder modules are critical reasons for better performance in such designs, especially to recover fine segmentation details [18, 19]. Despite the powerful learning capacity of ViTs as a backbone for classification, building such connections is challenging in typical transformer architectures.

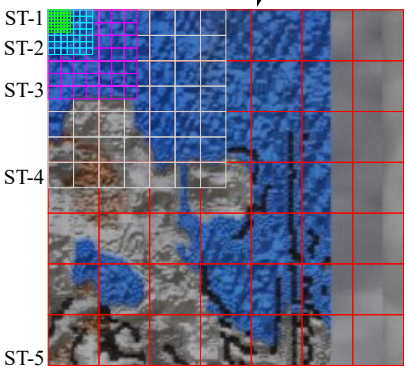


a) HDS-Transformer architecture

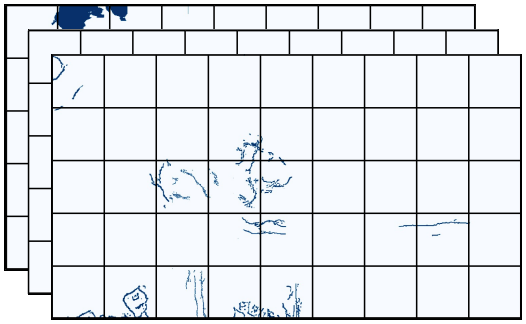


b) Batch cropping input image

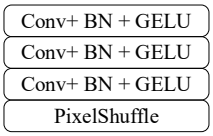
Sample crop



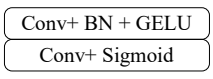
c) Demonstration of spatial output size after each transformer blocks



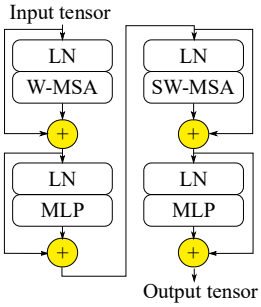
d) 3-channel damage segmentation output (crack, rebar, spall)



e) Convolution Decoder (CD) block



f) Last Convolution (LC) block



g) Two successive Swin Transformer (ST) blocks (Liu et. al 2021)

Fig. 6. DmgFormer Architecture. (Conv: 2D convolution, BN: batch normalization, GELU: Gaussian error linear unit, LN: Layer normalization, MLP: multi-layer perceptron, SW/W-MSA: regular and shifted windowed multi-head self-attention, LE: linear embedding, PM: patch merging [20]).

Swin Transformer [20] is an elegant network design that builds hierarchical feature maps by Patch Merging (PM) as the network gets deeper (Fig. 6). It has linear computational complexity compared with the original ViTs since the self-attention mechanism is computed locally. As demonstrated in Fig. 6.a, we have modified the transformer backbone by increasing the patch size to 2 instead of 4 and having 5 Swin Transformer (ST) blocks. The intuition behind this modification is that for tasks such as crack segmentation, higher precision might be necessary due to the thinner presence of "damage" pixels in the RGB images. Additionally, 5 decoder CNN blocks are designed to be consistent with the feature map tensors extracted after each ST block.

Small (S) and Large (L) variants of DmgFormer are considered depending on the initial embedding dimensionality of the attention blocks. Details of the two architectures are presented in Table 3.

Table 3. DmgFormer (S/L) architecture specification

Stage	DS rate (output size)	Attn. MLP dim.	# STL[(a)]	# Attn. head	Decoder feature dim.
1	2× (112×112)	48/64	2/2	3/4	24/32
2	4× (56×56)	96/128	2/2	6/4	96/124
3	8× (28×28)	192/256	2/2	8/8	180/240
4	16× (14×14)	384/512	6/18	12/16	336/448
5	32× (7×7)	768/1024	2/2	24/32	576/768

[(a)] # STL: Number of Swin Transformer layers

Augmented Inference

It was mentioned earlier that some context information might be lost despite the computational gains by exploding the image into smaller crops. We propose augmented inference as a technique to alleviate this loss of context. The arrangement of pixels in the 5×9 crop grid depends on the padding direction. Three types of zero padding, including right/top, Left/bottom, middle, were considered for each direction, resulting in a maximum of 8 different sets of grid crops. The augmented inference is conducted by feeding the same 1080p image to the model with different paddings and taking the average of model probabilities. This process is shown in Fig. 7.

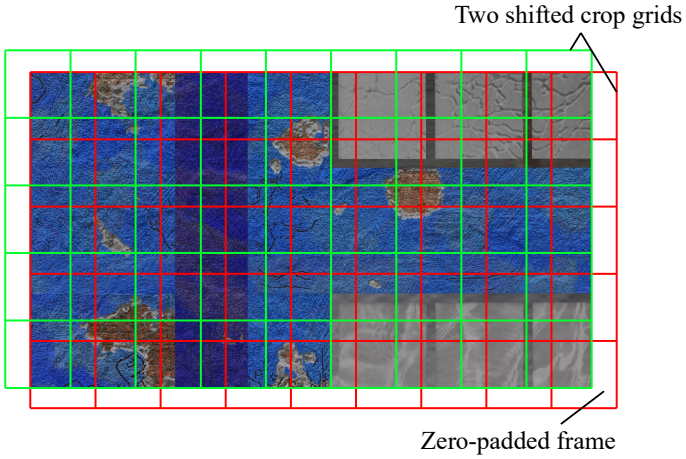


Fig. 7: Augmented inference in DmgFormer

IMPLEMENTATION

The IC-SHM dataset comprises a primary training and testing split for each task. Since the labels are not provided for the main test set, the original training portion is randomly split into 0.8, 0.1, and 0.1 subsets to further validate and test the neural network models. From this point forward, the testing set refers to the 0.1 splits mentioned earlier unless specified otherwise.

All models are created using PyTorch [21], considering its flexibility for custom architecture designs. The initial experiments for TRS-Net were conducted on a personal computer equipped with an NVIDIA GTX 1080 with 8 GB memory, an Intel i7-8700K, and 32 GB of RAM. DmgFormer experiments were also conducted on a workstation equipped with an NVIDIA Titan V GPU with 12 GB memory, an Intel Core i9 10980-XE, and 64 GB of RAM. Some hyperparameter tunings were conducted using the University at Buffalo's Center for Computational Research (UB CCR) NVIDIA TESLA GPUs with 16/32 GB of memory.

For TRS-Net, we implement data augmentation techniques during training in all our experiments which is composed of multiple image transforms such as random horizontal flip, zoom, and perspective [22]. It also includes random color manipulations such as brightness, gamma, and saturation. We found that data augmentation techniques can help reduce overfitting and help boost the quality of predictions, especially for unseen structures in the main unlabeled test split. TRS-Net model is trained using an Adam optimizer [23] with an initial learning rate of 0.001 and exponential decay rates for the 1^{st} and 2^{nd}-moment estimates of 0.9 and 0.999, respectively. We also use a scheduler that further reduces the learning rate if no improvements are observed to a minimum of 0.00001. For the loss function, we use focal loss [24] with $\alpha = 0.25$ and $\gamma = 2.0$. The model parameters corresponding to the best validation loss are used for inference. During experiments, we found that allowing the internal segmentation model to train on downsized images and mask initially and fine-tune the whole TRS-Net to the high-resolution images and masks improves training. Thus, we use this strategy to train TRS-Net for tasks 1&2. Additionally and for the sake of comparison, we have used the internal segmentation model of TRS-Net to train on each subtask of task 3 independently (binary segmentation). To reduce the effect of data imbalance, especially for exposed rebars segmentation, we added a final layer of augmentation during training that crops a random area that has the subtask mask, if it exists in the image. Otherwise, it results in a random crop. These crops have a size of 480×270, thus compatible with the internal segmentation model.

DmgFormers were trained using Adam optimizer with 5 epochs of learning rate warm up to a maximum learning rate of 0.0002 and further decreasing with a cosine function until 300 epochs. Each batch loaded two images and randomly shuffled the grid crops into smaller sub batches during training to fit the GPU memory. Focal loss with $\alpha = 0.6$ and $\gamma = 2$ was selected based on several experiments.

RESULTS

Tasks 1 and 2

The following tests are conducted. For Tasks 1 & 2, we have three tests. First, the internal network of TRS-Net is trained and tested on interpolated downsized images and masks. Second, we attach a uniform, non-trainable upsampler and downsampler to the internal network's stem and head and test on high-resolution images. Finally, we train and test our proposed TRS-Net on high-resolution images and compare the results with previous iterations.

The testing results for all six iterations are shown in Table 4. It can be seen that using a uniform sampler (such as nearest-neighbor) for resizing images and masks deteriorates the prediction quality compared to the low-resolution version. On the other hand, TRS-Nets, with their trainable upsampler and downsampler, do not only match the low-resolution models but also improve the masks prediction qualities as demonstrated by all performance metrics in the two tasks. This is due to DCN's ability to preserve important features of the original image, such as edges and fine cracks, which can be helpful in the internal segmentation model. Class-wise performance metrics for TRS-Net are shown in Tables 5 and 6. Overall, the performance metrics are relatively close except the "no-damage" state in task 2, possibly due to the limited frequency of this label in the dataset. To this end, we use TRS-Net architecture as the model for tasks 1 and 2 as our experiments showed the superiority of TRS-Net's model in components and damage-state segmentation.

Table 4. Mean testing performance metrics for components and damage state segmentation

	Components (task 1)				Damage state (task 2)			
	Precision	Recall	F1-score	IoU	Precision	Recall	F1-score	IoU
Internal TRS-Net (low resolution)	99.40	99.53	99.47	98.94	97.42	97.31	97.36	94.93
Internal TRS-Net + resizing	97.08	96.72	96.89	94.10	96.47	95.50	95.97	92.39
TRS-Net	99.56	99.56	99.56	99.13	98.60	97.83	98.21	96.52

Table 5. Class-wise testing performance metrics for components segmentation (TRS-Net)

	Wall	Beam	Column	Window frame	Window pane	Balcony	Slab	Ignore
Precision	99.81	99.52	99.75	98.95	99.8	99.84	98.86	99.98
Recall	99.82	99.59	99.75	98.86	99.78	99.83	98.85	99.96
F1-score	99.81	99.56	99.75	98.91	99.79	99.83	98.85	99.97
IoU	99.63	99.12	99.5	97.84	99.59	99.67	97.74	99.94

Table 6. Class-wise testing performance metrics for damage-state segmentation (TRS-Net)

	No damage	Light damage	Moderate damage	Severe damage	Ignore
Precision	96.98	98.5	99.23	98.42	99.85
Recall	94.07	98.92	99.22	97.12	99.81
F1-score	95.50	98.71	99.22	97.77	99.83
IoU	91.40	97.45	98.46	95.63	99.66

Task 3

The testing results of damage segmentation are given in Table 7. The testing performance of the internal portion of TRS-Net is also included for comparison. It can be observed that preserving the original resolution by cropping yields higher fidelity segmentation masks. Furthermore, by using 4 and 8 iterations of augmented inference (AI-4 & 8), performance is boosted at the cost of increased inference times. Our experiments show that training a single model that simultaneously predicts rebar, crack and spalling has superior performance over three single models. One hypothesis about this phenomenon is that a model that learns to predict all classes simultaneously learns its features maps to differentiate between cases where damage classes could have the same appearances, especially for cracks and

spalling. We also found that less than 1% of the dataset has rebar pixels, and its frequency is approximately 30 times less than cracks and spalling. To this end, the internal portion of TRS-Net was used for rebar segmentation which is equipped with augmentation as a regularization technique.

Table 7. Testing performance metrics for damage segmentation (task 3)

	Precision			F1-score		
	Crack	Rebar	Spall	Crack	Rebar	Spall
DmgFormer-S	85.46	85.56	97.75	86.06	85.17	98.09
DmgFormer-S (AI-4)	85.96	87.20	97.87	86.21	85.93	98.16
DmgFormer-S (AI-8)	86.09	87.07	97.89	86.26	85.94	98.17
DmgFormer-L	85.82	88.76	97.94	86.82	87.35	98.27
DmgFormer-L (AI-4)	86.25	89.84	98.06	86.98	87.92	98.34
DmgFormer-L (AI-8)	86.41	89.97	98.10	87.05	88.08	98.37
TRS-Net (internal network)	82.69	88.01	96.72	84.87	86.48	97.03
	Recall			IoU		
	Crack	Rebar	Spall	Crack	Rebar	Spall
DmgFormer-S	86.67	84.8	98.44	75.53	74.18	96.25
DmgFormer-S (AI-4)	86.47	84.7	98.45	75.77	75.33	96.38
DmgFormer-S (AI-8)	86.43	84.84	98.45	75.84	75.35	96.4
DmgFormer-L	87.83	85.99	98.60	76.70	77.54	96.60
DmgFormer-L (AI-4)	87.72	86.08	98.62	76.96	78.44	96.74
DmgFormer-L (AI-8)	87.70	86.27	98.63	77.07	78.70	96.78
TRS-Net (internal network)	87.17	85.01	97.35	73.72	76.18	94.24

CONCLUSION

Visual structural inspections are entering a new era with the latest advances in sensing and computing technology accompanied by the giant progress in AI research. UAVs can equip the building owners and inspectors with thousands of high-resolution frames from a building façade. Effectively processing this information for autonomous inspections requires high performance and computationally cost-effective models that ultimately contribute to the UAV navigation and damage identification. State-of-the-art deep learning vision models are moving toward the larger models with massive parameters and are not optimized for SHM vision tasks. This poses a great challenge for autonomous inspections tasks. Real-time inference in detecting structural components and their severities is critical for optimal UAV guidance. Furthermore, high-resolution segmentation is a must in tasks such as identifying thin cracks. Project 2 of the IC-SHM 2021 highlights these challenges by containing several inherently different tasks handling full HD images.

In this work, we argue that component type (Task 1) and damage severity (Task 2) segmentation rely on global context. In contrast, damage segmentation (Task 3) is highly sensitive to preserving the input image resolution. We approached this project by proposing two custom deep learning frameworks namely: TRS-Net and DmgFormer. TRS-Net utilizes an internal segmentation model based on ResNeSt backbone and U-Net++ decoder in addition to an outer encoder-decoder of trainable down and upsampling networks. It was found that the trainable resizing modules significantly improve the model's efficiency while minimizing the loss of information during resizing in Tasks 1 and 2. Damage segmentations in task 3 are conducted with DmgFormer which is equipped with a Swin transformer

backbone and a convolutional decoder. It is shown that using crops of the original image resolution with the proposed augmented inference technique can result in better segmentation metrics.

ACKNOWLEDGMENTS

The authors worked on the competition for several months while being employed by the University at Buffalo as research assistants. Our efforts were conducted outside of the designated Ph.D. programs and in compliance with the IC-SHM competition rules. We want to express sincere gratitude towards our supervisor, Dr. Xiao Liang, for his years of mentorship and support before this competition.

REFERENCES

[1] V. Hoskere, Y. Narazaki, and B. F. Spencer, "Physics-Based Graphics Models in 3D Synthetic Environments as Autonomous Vision-Based Inspection Testbeds," *Sensors,* vol. 22, no. 2, p. 532, 2022.
[2] M. H. Hesamian, W. Jia, X. He, and P. Kennedy, "Deep learning techniques for medical image segmentation: achievements and challenges," *Journal of digital imaging,* vol. 32, no. 4, pp. 582-596, 2019.
[3] A. Katharopoulos and F. Fleuret, "Processing megapixel images with deep attention-sampling models," in *International Conference on Machine Learning*, 2019: PMLR, pp. 3282-3291.
[4] H. Zhang *et al.*, "Resnest: Split-attention networks," *arXiv preprint arXiv:2004.08955,* 2020.
[5] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep learning in medical image analysis and multimodal learning for clinical decision support*: Springer, 2018, pp. 3-11.
[6] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874-1883.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.
[8] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, 2019: PMLR, pp. 6105-6114.
[9] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132-7141.
[10] P. Yakubovskiy. "Segmentation Models Pytorch." GitHub. https://github.com/qubvel/segmentation_models.pytorch (accessed.
[11] R. Wightman. "PyTorch Image Models." GitHub. https://github.com/rwightman/pytorch-image-models (accessed.
[12] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 558-567.
[13] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998-6008.
[14] Y. Zhang *et al.*, "Pushing the limits of semi-supervised learning for automatic speech recognition," *arXiv preprint arXiv:2010.10504,* 2020.
[15] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy.
[16] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," *arXiv preprint arXiv:2201.03545,* 2022.
[17] PapersWithCode. "Semantic Segmentation " https://paperswithcode.com/task/semantic-segmentation (accessed.
[18] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 11-19.
[19] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image

segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence,* vol. 40, no. 4, pp. 834-848, 2017.
[20] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030,* 2021.
[21] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems,* vol. 32, pp. 8026-8037, 2019.
[22] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *Information,* vol. 11, no. 2, p. 125, 2020.
[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.
[24] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980-2988.