Bridge defect pixel classification using UNet and SegNet

I. Introduction

Bridge destruction and bridge-related accidents are common occurrences in the United States. In May 2021, the six-lane I-40 bridge linking Arkansas and Tennessee was closed due to a significant fracture essential to the bridge's integrity. This defect was not fixed because it went unnoticed by the inspections conducted in both 2019 and 2020. The drawbacks of the current inspection method include risk to workers, high time requirements, and human errors. Using an autonomous inspection system will eradicate most of these problems. However, that means that the system has to be very accurate in recognizing defects. Since deep learning has been widely utilized to classify anomalies, in this project, we will be utilizing deep learning to identify the exact location of defects on bridges. The deep learning technique we will use, UNet and SegNet, labels each pixel of the image as defect or non-defect. We will be comparing the performance of both networks through multiple performance measurements such as accuracy, TP, TN, memory usage, duration, etc. In addition, acquiring enough data for learning is another challenge we will also tackle. To do so, we will use data augmentation techniques to omit the need for more training data.

II. Literature Review

Bridge defect classification has evolved over the years. The early techniques were edge detection algorithms such as Laplace of Gaussian and Canny edge detector [2][3] which are efficient at detecting crack defects. While these techniques are not computationally expensive, they are not as accurate when it comes to real-world image noises. Another attempt was image classification using Support Vector machines [5] and Multi-Layer Perceptrons [4] which involves modeling a set of training data to discern between two or more classes of image. However, they

are also prone to noise and require a balance between classes or their accuracy will drop and while it indicates whether there are defects or not, it does not give the location of the defects.

Finally, in an attempt to improve detection accuracy, deep Convolutional Neural Networks (CNN) have been utilized in recent research. There are various CNN structures used for the classification of defects such as AlexNet [7] and VGGNet [6]. In addition, various segmentation methods with encoder and decoder such as SegNet [8] and UNet [1] were also created to detect defects in the pixel level with semantic segmentation. While modern CNN structures improve detection accuracy, it is computationally expensive and requires an extensive amount of training data. Since modern technology has improved, training CNN structures do not take as much time with the use of GPU. Furthermore, since the collection and processing of large data-set are time-consuming, we will be using data augmentation to reduce the need for it.

III. Methodology

UNet

Unet is an architecture for semantic segmentation consisting of four encoders and four decoders with 1 bottleneck in the middle. Each encoder and the single bottleneck are composed of a double 3x3 convolution layer followed by a rectified linear unit (ReLU), and a 2x2 max-pooling layer with stride 2 to downsample the size of the feature map. Each decoder is composed of an upsampling of the feature map followed by a 2x2 up-sampling that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. Originally, the convolutions of each encoder had the filter size of 64, 128, 256, then 512 with the bottleneck at 1024 and the reverse of the encoders' filter for the decoders respectively. However, to reduce the time, we reduced the filter size to just half at 32, 64, 128, and 256 with the bottleneck at 512 as

seen in Figure 1. In the end, the final layer is a 1x1 convolution with a filter of 1. A sigmoid layer is added since we only have two classes: defect or non-defect. We also calculated the output dimensions, parameters, memory, and operations of each layer in Table 1.

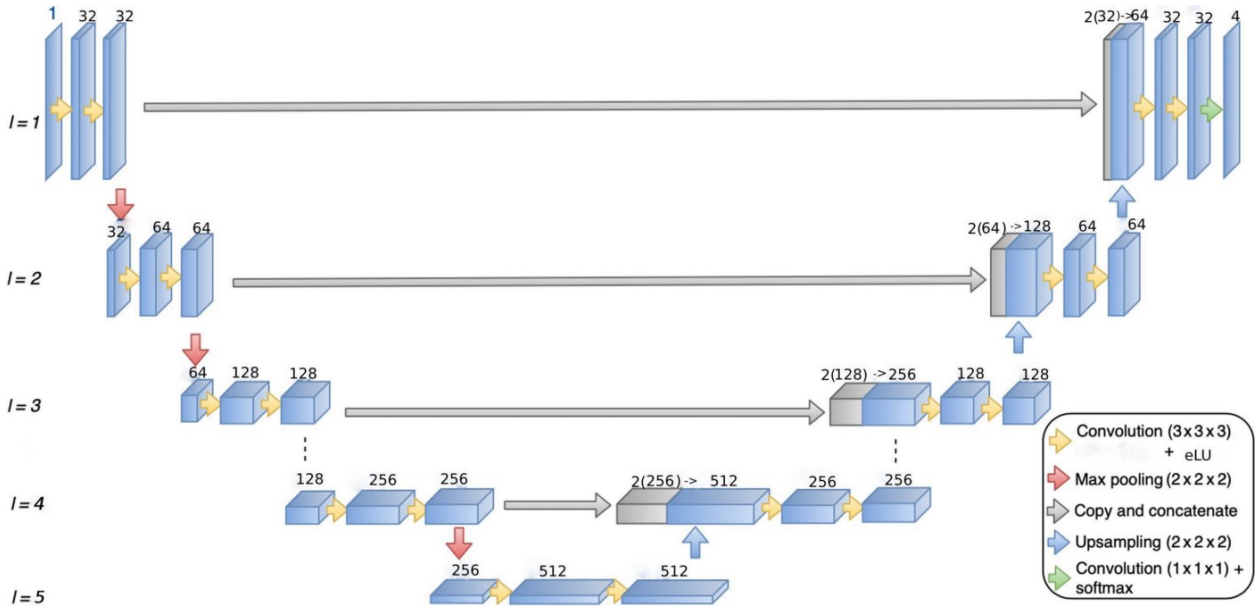

Figure 1. UNet Architecture

Layer	Filter	Size	Output	Param	Memory	Op.
Input			512 x 512 x 3		512 x 512 x 3	
Conv	32	3	512 x 512 x 32	896	512 x 512 x 32	512^2 x 32 x 27
Conv	32	3	512 x 512 x 32	9K	512 x 512 x 32	512^2 x 32^2 x 9
Max		2	256 x 256 x 32		256 x 256 x 32	
Conv	64	3	256 x 256 x 64	18K	256 x 256 x 64	256^2 x 64 x 32 x 9
Conv	64	3	256 x 256 x 64	36K	256 x 256 x 64	256^2 x 64^2 x 9
Max		2	128 x128 x 64		128 x128 x 64	
Conv	128	3	128 x128 x 128	73K	128 x128 x 128	128^3 x 64 x 9
Conv	128	3	128 x128 x 128	147K	128 x128 x 128	128^3 x 128 x 9
Max		2	64 x 64 x 128		64 x 64 x 128	

Conv	256	3	64 x 64 x 256	295K	64 x 64 x 256	64^2 x 256 x 128 x 9
Conv	256	3	64 x 64 x 256	590K	64 x 64 x 256	64^2 x 256^2 x 9
Max		2	32 x 32 x 256		32 x 32 x 256	
Conv	512	3	32 x 32 x 512	1.1M	32 x 32 x 512	32^2 x 512 x 256 x 9
Conv	512	3	32 x 32 x 512	2.3M	32 x 32 x 512	32^2 x 512^2 x 9
Up		2	64 x 64 x 256		64 x 64 x 256	
Concat			64 x 64 x 512		64 x 64 x 512	
Conv	256	3	64 x 64 x 256	1.1M	64 x 64 x 256	64^2 x 256 x 512 x 9
Conv	256	3	64 x 64 x 256	590K	64 x 64 x 256	64^2 x 256^2 x 9
Up		2	128 x 128 x 128		128 x 128 x 128	
Concat			128 x 128 x 256		128 x 128 x 256	
Conv	128	3	128 x 128 x 128	295K	128 x 128 x 128	128^3 x 256 x 9
Conv	128	3	128 x 128 x 128	147K	128 x 128 x 128	128^4 x 9
Up		2	256 x 256 x 64		256 x 256 x 64	
Concat			256 x 256 x 128		256 x 256 x 128	
Conv	64	3	256 x 256 x 64	73K	256 x 256 x 64	256^2 x 64 x 128 x 9
Conv	64	3	256 x 256 x 64	36K	256 x 256 x 64	256^2 x 64^2 x 9
Up		2	512 x 512 x 32		512 x 512 x 32	
Concat			512 x 512 x 64		512 x 512 x 64	
Conv	32	3	512 x 512 x 32	18K	512 x 512 x 32	512^2 x 32 x 64 x 9
Conv	32	3	512 x 512 x 32	9K	512 x 512 x 32	512^2 x 32^2 x 9
Conv	1	1	512 x 512 x 1	32	512 x 512 x 1	512^2 x 32

Table 1. UNet output dimensions, parameters, memory, and operations.

Total Params: 7.02M Total Memory: 116M * 4 ~= 464 MB/image

Total Operations: 46.1B

SegNet

SegNet is another architecture for semantic segmentation consisting of five encoders and five decoders. Each encoder is compose of convolution layers followed by a ReLU, and a 2x2 max-pooling layer with stride 2 to downsample the size of the feature map. As seen in Figure 2, the first 2 encoders and last 2 decoders has double convolution layers while the rest have 3 convolution layers. Each decoder is composed of an upsampling of the feature map followed by a 2x2 up-sampling that halves the number of feature channels, convolution layers, each followed by a ReLU. Similar to our UNet, while the original SegNet encoder has filter number 64, 128, 256, 512 then 512 and the reverse for decoder, we reduce the number of filter to 32, 64, 128, 256, 512 respectively. Since SegNet does not have fully connected layer at the end of the network, our last convolution layer has a filter of 1 and inputted into a sigmoid layer since we only have two classes: defect or non-defect. We also calculated the output dimensions, parameters, memory, and operations of each layer in Table 1.

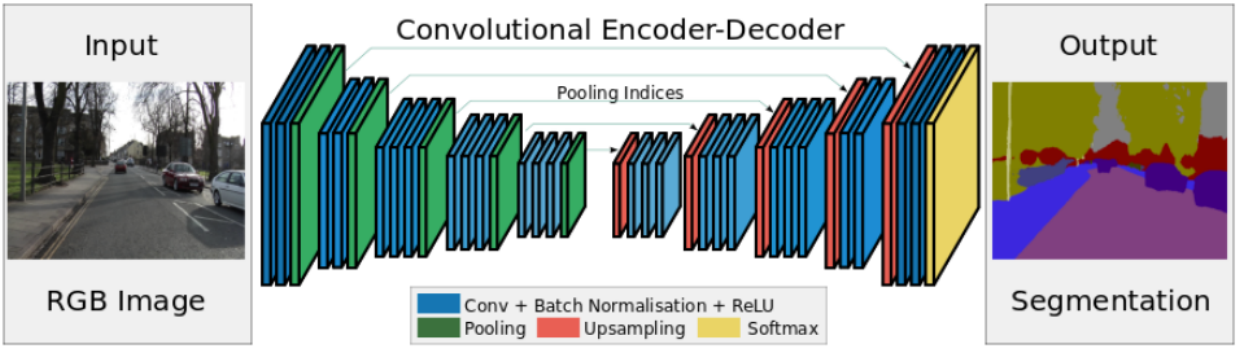

Figure 2. UNet Architecture

Layer	Filter	Size	Output	Param	Memory	Op.
Input			512 x 512 x 3		512 x 512 x 3	
Conv	32	3	512 x 512 x 32	896	512 x 512 x 32	512^2 x 32 x27

Conv	32	3	512 x 512 x 32	9K	512 x 512 x 32	512^2 x 32^2 x 9
Max		2	256 x 256 x 32		256 x 256 x 32	
Conv	64	3	256 x 256 x 64	18K	256 x 256 x 64	256^2 x 64 x 32 x 9
Conv	64	3	256 x 256 x 64	36K	256 x 256 x 64	256^2 x 64^2 x 9
Max		2	128 x128 x 64		128 x128 x 64	
Conv	128	3	128 x128 x 128	73K	128 x128 x 128	128^3 x 64 x 9
Conv	128	3	128 x128 x 128	147K	128 x128 x 128	128^4 x 9
Conv	128	3	128 x128 x 128	147K	128 x128 x 128	128^4 x 9
Max		2	64 x 64 x 128		64 x 64 x 128	
Conv	256	3	64 x 64 x 256	295K	64 x 64 x 256	64^2 x 256 x 128 x 9
Conv	256	3	64 x 64 x 256	590K	64 x 64 x 256	64^2 x 256^2 x 9
Conv	256	3	64 x 64 x 256	590K	64 x 64 x 256	64^2 x 256^2 x 9
Max		2	32 x 32 x 256		32 x 32 x 256	
Conv	512	3	32 x 32 x 512	1.1M	32 x 32 x 512	32^2 x 512 x 256 x 9
Conv	512	3	32 x 32 x 512	2.3M	32 x 32 x 512	32^2 x 512^2 x 9
Conv	512	3	32 x 32 x 512	2.3M	32 x 32 x 512	32^2 x 512^2 x 9
Max		2	16 x 16 x 512		16 x 16 x 512	
Up		2	32 x 32 x 512		32 x 32 x 512	
Conv	512	3	32 x 32 x 512	2.3M	32 x 32 x 512	32^2 x 512^2 x 9
Conv	512	3	32 x 32 x 512	2.3M	32 x 32 x 512	32^2 x 512^2 x 9
Conv	512	3	32 x 32 x 256	2.3M	32 x 32 x 256	32^2 x 256 x 512 x 9
Up		2	64 x 64 x 256		64 x 64 x 256	
Conv	256	3	64 x 64 x 256	590K	64 x 64 x 256	64^2 x 256^2 x 9
Conv	256	3	64 x 64 x 256	590K	64 x 64 x 256	64^2 x 256^2 x 9
Conv	256	3	64 x 64 x 128	295K	64 x 64 x 128	64^2 x128 x 256 x 9

Up		2	128 x 128 x 128		128 x 128 x 128	
Conv	128	3	128 x 128 x 128	147K	128 x 128 x 128	128^4 x 9
Conv	128	3	128 x 128 x 128	147K	128 x 128 x 128	128^4 x 9
Conv	128	3	128 x 128 x 64	73K	128 x 128 x 64	128^3 x 64 x 9
Up		2	256 x 256 x 64		256 x 256 x 64	
Conv	64	3	256 x 256 x 64	36K	256 x 256 x 64	256^2 x64^2 x 9
Conv	64	3	256 x 256 x 32	18K	256 x 256 x 32	256^2 x 32 x 64 x 9
Up		2	512 x 512 x 32		512 x 512 x 32	
Conv	32	3	512 x 512 x 32	9.2K	512 x 512 x 32	512^2 x 32^2 x 9
Conv	1	3	512 x 512 x 1	288	512 x 512 x 1	512^2 x 32

Table 1. SegNet output dimensions, parameters, memory, and operations.

Total Parameters: 15.6M Total Memory: 81.3M * 4 ~= 33.2MB/image

Total Operations: 48B

Data Augmentation

The two network architectures requires an extensive amount of training data with enough instances of each class, data augmentation creates more training images. This abates the need for a large number of data instances.

Our data augmentation method generated 50 sub-sample for every image sample and its pixel map. For each instance of sub-sample, we made sure to either rotate it randomly, flip it horizontally or vertically, or do a combination of each randomly. We also normalize all sub-sample to help the network to learn faster and better but doesn't extend my dataset.

IV. Experiment Results

Dataset

We obtain our dataset from a previous experiment involving a bicycle steel inspection robot climbing and collecting data on a custom-made bridge built by the Advanced Robotics and Automation (ARA) Lab. It was annotated with red on the defective pixels as seen in Figure 7. However, since we use sigmoid instead of softmax, we modified the annotation so that it is black and white. The defective pixels in the original image are assigned the color white in the pixel map. The non-defect pixels are assigned to the color black. Unfortunately, the red part of the annotation is not just (255,0,0) in RGB so we set up a range where it could have been red. That allowed us to get almost all the pixel classification correct although a small amount is incorrect labeling. In addition, since the images were originally 1920 x 1080, we had to resize them to 512 x 512, the size that we inputted into UNet and SegNet before utilizing data augmentation.

Parameters and Measurements

We optimized both our network architecture with Adam optimizer with a learning rate of 0.0001. For the loss function, we used BCEWithLogitsLoss since we are using a sigmoid layer that identifies the pixel as defect or non-defect. It was trained through 100 epochs with a batch size of 16. While I found out in previous experiments that the higher the epoch, the more accurate it became, it became evident that these models reach very high measurements and did not get any significant higher as seen in Figures 3-6. Hence, that's why I did not set the epoch to 400.

After testing several times, I realized that I can't judge each epoch base on just 1 measurement from the validation set like the last homework and save the model whose accuracy increased. Instead, I saved all epoch models and tested them with the test dataset once the

training is finished. I based the best performance on the model with the highest sum of all three measurements. Coincidentally, the model highest sum of the validation also performs the highest sum of the testing data for both UNet and SegNet.

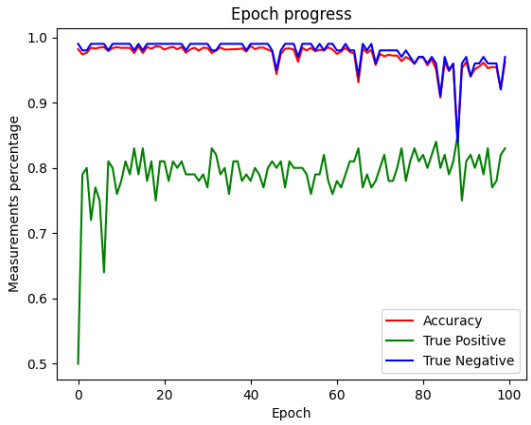

Figure 3. UNet Validation epoch progress

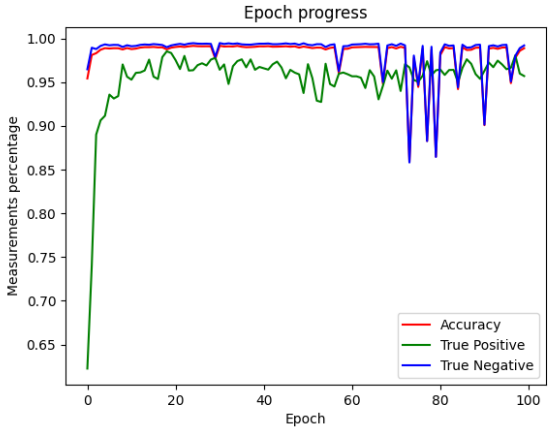

Figure 4. UNet Training epoch progress

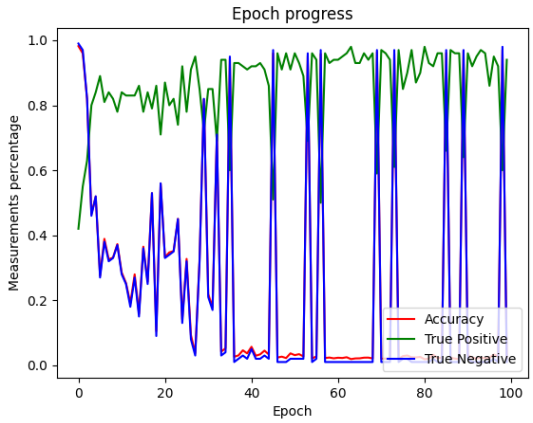

Figure 5. SegNet Validation epoch progress

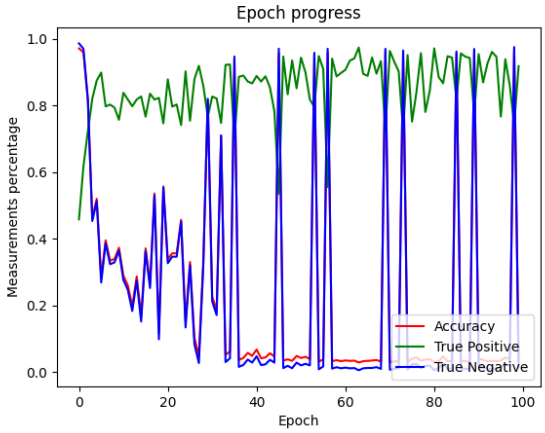

Figure 6. SegNet Training epoch progress

Network	Acc	TP	TN	Inference Time per second per epoch (it/s)	Time per epoch (min)	GPU memory usage (MiB)
UNet	0.97	0.81	0.99	2.48	2	12.9 K
SegNet	0.92	0.68	0.93	2.83	1:45	9.7 K

Table 3. SegNet and UNet best testing measurements.

Comparison

In this section, we will evaluate and compare our two network architectures with different state-of-the-art statistical measurements such as accuracy, true positive, and true negative. We defined defects as positive and non-defect as negative. Our true positive (TP) is defined as the number of correctly defect pixels detected out of the total number of defect pixels. Our true negative (TN) is defined as the number of correctly non-defect pixels detected out of the total number of non-defect pixels. Moreover, we added more information such as GPU memory usage, durations, and inference time per second of each epoch. A summary of the evaluation results performed for UNet and SegNet can be seen in Table 3.

It seems that while SegNet can run more data points and takes less memory and time to train than UNet, it vastly underperforms when it comes to the three statistic measurements we use to measure. This shows that UNet used its parameters better despite only having half the amount as SegNet. It also shows that even though the concatenation process before every decoder resulted in large memory consumption, it is an efficient way to increase state-of-the-art statistical measurements. In addition, the measurements seem to be more stable while training through epochs as seen in Figures 2 and 3 compared to SegNet's accuracy and TN jumping up and down as seen in Figures 4 and 5. Due to unstable measurement jumping up and down, we can see that in Figure 7, SegNet's testing output many randomly scattered pixels as positive defects while UNet, though its output does have sprinkles, it has few and far.

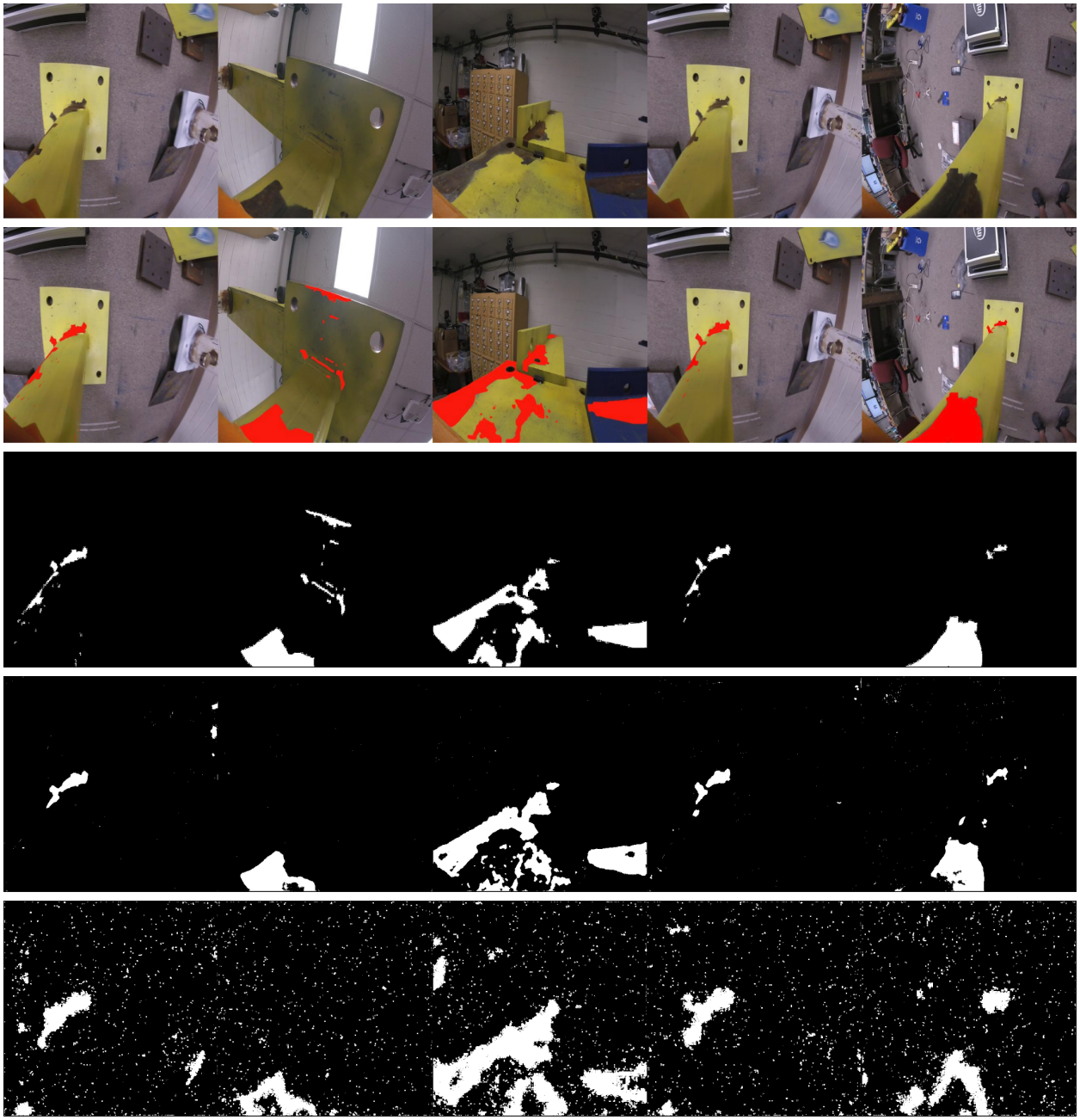

Figure 7. Model output where the order is the original image, previous annotation, our experiment annotation, UNet output, and SegNet output.

V. Conclusion and Future Work

In conclusion, we presented a comparison between two deep convolutional network architectures: UNet and SegNet. Our comparison showed that SegNet sacrifices accuracy, TP, and TN in favor of less memory usage, training time, and running more data points while UNet

did the complete opposite. It shows the concatenation implementation of UNet is very efficient for training parameters despite the large memory usage since UNet performs better in all of our statistical measurements with total parameters being half as much as SegNet. Moreover, we implemented data augmentation for network performance. In the future, we aim to perform better data augmentation techniques since we only used flip, rotations, and normalizations. In addition, we aim to optimize UNet to lower the training time and memory usage while improving SegNet statistical measurements and unstable training performance.

VI. Reference

[1] Olaf Ronneberger, Philipp Fischer, Thomas Brox. Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS, Vol.9351: 234--241, 2015

[2] R. S. Lim, H. M. La, and W. Sheng, "A robotic crack inspection and mapping system for bridge deck maintenance," IEEE Transactions on Automation Science and Engineering, vol. 11, no. 2, pp. 367–378, April 2014.

[3] R. S. Lim, H. M. La, Z. Shan, and W. Sheng, "Developing a crack inspection robot for bridge maintenance," in 2011 IEEE International Conference on Robotics and Automation, May 2011, pp. 6288–6293.

[4] ray, J., Verma, B., Li, X., He, W.: A neural network-based technique for automatic classification of road cracks. In: 2006 International Joint Conference on Neural Networks, IJCNN 2006, pp. 907–912. IEEE (2006)

[5] Prasanna, P., Dana, K., Gucunski, N., Basily, B.: Computer-vision based crack detection and analysis. In: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2012, vol. 8345, p. 834542. International Society for Optics and Photonics (2012)

[6] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, International Conference on Learning Representations, 2015

[7] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems 25 (NIPS 2012)

[8] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2481-2495, 1 Dec. 2017.