APPLICATIONS OF A U-NET VARIANT NEURAL NETWORK: IMAGE CLASSIFICATION FOR VEGETATION COMPONENT IDENTIFICATION IN OUTDOORS IMAGES AND IMAGE TO IMAGE TRANSLATION OF ULTRASOUND IMAGES

by

Adam Honts

A Thesis Submitted in
Partial Fulfillment of the
Requirements for the Degree of

Master of Science
in Mathematics

at
The University of Wisconsin-Milwaukee
May 2021

ABSTRACT

APPLICATIONS OF A U-NET VARIANT NEURAL NETWORK: IMAGE CLASSIFICATION FOR VEGETATION COMPONENT IDENTIFICATION IN OUTDOORS IMAGES AND IMAGE TO IMAGE TRANSLATION OF ULTRASOUND IMAGES

by

Adam Honts

The University of Wisconsin-Milwaukee, 2021
Under the Supervision of Professor Istvan Lauko

Convolutional Neural Networks have been applied in many image applications, for both supervised and unsupervised learning. They have shown their ability to be used in an array of diverse use cases which include but are not limited to image classification, segmentation, and image enhancement tasks. We make use of Convolutional Neural Networks' ability to perform well in these situations and propose an architecture for a Convolutional Neural Network based on a network known as U-Net. We then apply our proposed network to two different tasks, a vegetation classification task for images of outdoors environment, and an image to image translation task for ultrasound images. For the vegetation classification task we make use of our previous work of a green vegetation filter that is used to annotate our data set and then use images that are converted to gray scale to pair with the annotations from the green vegetation filter in order to train our proposed network to classify where generic vegetation appears in an image. For the ultrasound image to image translation task, we show that our proposed network can be used as part of a system which is composed of a set of neural networks, called CycleGAN, that is used to translate ultrasound images from images acquired by a low frequency transducer to an image domain of ultrasound images acquired by a high frequency transducer.

We propose using an approach that trains our proposed network to learn local estimations of the two image domains and detail a filtering process that when applied to an ultrasound image, acquired from a low frequency transducer, gives the low frequency transducer ultrasound image the appearance that it was acquired from a high frequency transducer.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to acknowledge and thank each of the members of my committee, Dr. Istvan Lauko, Dr. Dexuan Xie, and Dr. David Spade, as well as Dr. Gabriella Pinter. I have had the pleasure of taking courses offered by each of these members of my committee and they have helped shape me into a better person because of it.

Dr. Spade has taught me many necessary skills to handle problems that I encounter on a daily basis at work. He has always taken the time to answer any questions I have had, be it about school, work, or life. I have enjoyed all of our conversations after our class and in his office hours, and his hands on problem solving approach to teaching courses.

Dr. Xie's course in Numerical Analysis was one of the most interesting courses I have taken and it allowed me to gain many insights into the algorithms that I have encountered at work. He taught me the necessary rigor in proving why algorithms converge and how they can be applied in practice. He also has helped me become a better a programmer. I sincerely appreciate all of his help and transfer of knowledge.

Dr. Lauko and his wife Dr. Pinter have been some of the most kind human beings I have ever had the pleasure of meeting. Through out my undergraduate degree, my work with them as part of the Bio Math program and graduate degree they have been the best influence on me as an applied mathematician. They have presented me with opportunities to learn and participate on amazing projects such as the Green Index and the work I am presenting in this thesis. They have given me rides home from school, helped me with my school work, answered any questions I had, taught me fascinating concepts, and helped me become a better person. I am forever grateful for everything they have done for me.

Dr. Lauko I would especially like to thank for being my advisor and working with me on these projects. These were the most fun projects I have ever been a part of, and the results

we were able to produce wouldn't have been possible with out his help. I'm incredibly proud to have worked with him.

I would like to acknowledge Jacob Beihoff for his work on creating the green vegetation filter with Dr. Lauko and me. His contributions are a major reason as to why we were able to do this project.

I would like to acknowledge Scott Rupprecht for his work on creating the MKE County Green Index website where we display our results from the green vegetation filter.

I would like to acknowledge Nathan Chicks Wojciechowski and Christine E Grams for their help in acquiring ultrasound images. They spent many hours working with Dr. Lauko to develop a process for improving ultrasound images.

I would like to thank my parents. Without their loving support through many tough times during my childhood, my undergraduate degree, my short tenure of full time work after undergrad, and my graduate degree. I would not be where I am today, and which I can say is the happiest I have ever been. I love them very much and I appreciate everything they have done for me.

Lastly, I would like to thank my significant other Lily Gierke who has taken care of me during my transition from undergraduate degree, to full time work, and back to graduate school. There has been many difficult times because of school and work, and Lily has been with me every step of the way with all of her love and support. Without her I would not be where I am and like I mentioned earlier I am the happiest I have ever been and that is also thanks to her.

1 Introduction

Deep learning has become a popular method used by machine learning researchers when working with image data, due to the availability of GPUs that enable a large number of calculations to be performed in parallel thus increasing the speed and efficiency of training deep learning models [23, 32]. Deep learning models such as convolutional neural networks (CNNs) have been used for image classification tasks and have performed these tasks with high levels of accuracy [3, 7, 15, 16, 32, 38, 41, 42, 50]. CNNs consist of layers called convolutional layers that are a set of learnable filters that are small spatially localized and are able to learn spatial and color patterns in an image set which aid in the process of classifying objects within an image [32, 41]. CNNs have also been shown to be able to perform segmentation tasks at a high level of accuracy to point out the areas within an image where an object of interest is located [38, 48]. A specific type of CNN called U-Net has been a popular network for classification and segmentation tasks and has been used in medical applications as well as vegetation detection applications [38, 46, 48]. U-Net is a residual learning network, that passes information learned early on in the network to deeper layers within the network [16, 38]. Residual learning networks have been used in generative models for tasks such as image to image translation [25, 46, 49]. A U-Net type network has also been used in a generative model setting to show that it has the capability to improve the quality of portable ultrasound images [46]. Making use of the ability that U-Net has shown to perform well in a number of different applications such as classification, image to image translation tasks, and improving ultrasound images, we propose a variant of a U-Net type architecture and apply it to a vegetation classification application and an image to image translation of ultrasound images application.

Using our prior work of a green vegetation filter that was used to classify green vegetation pixels in a large data set composed of Google Street View images that span Milwaukee County, we use the outputs of the green vegetation filter to annotate a data set that is composed of the gray scale version of the images that were passed into the green vegetation filter

[17]. This labeled data set composed of gray scale images paired with the output classification of the green vegetation filter is then used to train the proposed variant U-Net neural network to improve upon the capabilities of the green vegetation filter by classifying other colors of vegetation by forcing the network to learn the spatial characteristics of vegetation rather than using color information. We show results of the proposed network ability to locate green vegetation as well as non-green vegetation. We then proceed to use the same proposed variant U-Net neural network to perform an image to image translation task for translating ultrasound images acquired from a low frequency transducer domain to a higher frequency transducer domain. This is done by making use of a type of deep learning generative model called a generative adversarial network which is used as a part of a system called CycleGAN that has shown promising results when performing image to image translation tasks [11, 49]. We display and discuss the results of the proposed networks ability to give to an ultrasound image that was acquired from a low frequency transducer the appearance that it was acquired from a high frequency transducer, by using the outputs of the network as a local estimation filter.

1.1 Vegetation Detection

Street-scape image databases provide a novel and rich data source for identifying urban vegetation views and density as perceived by human observers at the street level. The abundant availability of such data has recently spawned interest and investigations of urban natural environments and their relations to human health, quality of life and other socioeconomic aspects [22, 43, 10]. An important component to such investigations is the ability to identify vegetation on complex images efficiently and reliably. Automated driving and automated low elevation flight technologies also face requirements to identify, model and evaluate complex physical environments with vegetation components based on image and video data.

Recently emerging machine learning technologies have proved to be well suited for such visual analysis tasks, but their use and efficacy is currently limited by the lack of availability

of large and high quality annotated image data sets for training purposes.

We rely on an efficient image processing method developed by the authors earlier [17, 6], to generate annotated data sets of street-scape images for training, and will demonstrate that relying on a large and variable annotated training set we use a variant of a U-Net architecture with very robust vegetation detection performance and excellent generalization capabilities. The network is trained to assign to each image pixel an estimated probability or confidence value (a value between 0 and 1) which represents the network's estimate of whether the pixel belongs to a vegetation image component.

A key step in achieving robust performance in vegetation identification from the trained network is to remove color information from the training image set prior to training and thus forcing the network to rely only on gray scale image features in the training and vegetation detection process. This is done to enable the network to generalize from the training data and to recognize the learned vegetation image features in color shades other than green and remarkably in images with only gray scale information. We will demonstrate that the network can consistently handle vegetation identification in general image data and in particular in images with color content that the original image processing teaching (i.e., annotating) algorithm would have failed on.

1.2 Ultrasound Image To Image Translation

Ultrasound imaging is a safe non-invasive diagnostic tool that is used to image the anatomy inside of a body [21, 20]. Ultrasound is a low risk method for physicians to capture images, as opposed to x-ray and CT scans which expose the patient to radiation [21, 20]. Recent advancements have also led to the ability to use Ultrasound technology on handheld devices, which makes using ultrasound more accessible than other common medical imaging techniques such as X-Ray, CT scans, and MRI [46]. Ultrasound Imaging utilizes a probe called a transducer that produces high-frequency sound waves and reconstructs an image by capturing the returning sound waves. Transducers typically have a given frequency of sound

wave that they can produce. The higher the frequency of the transducer, the higher the resolution of the image that can be reconstructed [40]. A caveat of using a higher frequency transducer is that the returning image is not able to extract information that exist deeper in the body as well as a lower frequency transducer able to retrieve [40].

Recent deep learning architectures, such as CycleGAN, have been able to identify image characteristics from one image class and apply it to another [49]. Some examples of this are taking images of apples and giving them the appearance of an orange, or giving a horse zebra stripes [49]. A U-Net type network placed used within the CycleGAN framework was shown to be able to improve the quality of portable Ultrasound images by translating images acquired from a portable Ultrasound device that produced low quality images, and mapped them to a domain defined by a set of higher quality ultrasound images [46]. Both of these ideas are the motivation for using neural networks for giving an Ultrasound image taken with a low frequency transducer and giving it the appearance that it was taken from a high frequency transducer. Reliably giving an image from a low frequency transducer the appearance that it was taken from a high frequency transducer would allow a physician to be able to see deeper into the body than they would be able to with a high frequency transducer, but also could result in reducing noise, that is present in a low frequency transducer image.

We propose using the same variant of a U-Net architecture that was used in the vegetation detection application to be trained in the methodology used in CycleGAN to produce images that resemble images taken from a higher frequency transducer. An unpaired image set of low frequency and high frequency transducers taken on the area of the cyphallic vein of volunteers was collected and used to train the network to produce the results presented in this work.

1.3 Short Outline of Thesis

In section 2, we describe in greater detail U-Net and the components that exist within the network architecture. We then propose our variant U-Net neural network and describe the

new components that we introduce.

In section 3, we discuss the vegetation detection application and the green vegetation filter that was used to annotate the data set we used to train our variant U-Net neural network to classify vegetation. We describe the data set that was used in the training process, and the characteristics that were conducive to being able to train our network to classify vegetation within an image. We then give a brief description as to how the green vegetation filter that was used to annotate our data set works. At the end of the section we discuss the training of the network and the network's results.

In section 4, we further describe the ultrasound image to image translation of translating ultrasound images acquired from a low frequency transducer to an image acquired from a high frequency transducer. We describe the history and theory behind the generative adversarial networks and CycleGAN to help give a better understand of the methodology used in training our proposed variant U-Net neural network. We give a description of the data set and discuss characteristics between the differences of the images between the low frequency ultrasound image domain and the high frequency ultrasound image domain. We detail the algorithm used for training the model as well as the filtering algorithm used to create the final image of the low frequency transducer ultrasound image that is translated to a high frequency transducer image domain. At the end of the section we discuss the results and shortcomings of the methodology.

In section 5, we conclude the paper by pointing out the key contributions that were given from this thesis.

In Appendix A, we give more examples of results from the network's ability to classify vegetation in Google Street View images.

In Appendix B, we give more examples of results from the network's ability to translate images from a low frequency transducer ultrasound image that is translated to a high frequency transducer image domain.

2 Description of U-Net and the proposed U-Net Variant

Our proposed network is a modification on the U-Net architecture developed by Ronneberger et al. [38]. U-Net and variations on U-Net have been used in image classification and segmentation tasks because of its ability to learn and maintain structural information of an image that have proven to work well in both medical and remote sensing applications. For example, Ronneberger et al. implemented U-Net for the segmentation of neuronal structures in electron microscopic recordings [38]. Blanc-Durand et al. used a 3-D implementation of U-Net to segment and detect brain tumors from PET scans [3]. Zhang et al. was able to apply a U-Net architecture to identifying roads from satellite imagery [48]. Li et al. showed that U-Net could be used for extracting buildings from satellite imagery [35].

The original U-Net architecture consists of a series of down sampling steps that contract the length and width dimensions of the input images and subsequent tensor blocks in the network, followed by a series of up sampling steps, and an interpolation of the output tensor block, to get back to the original length and width dimensions of the input images [38]. A major contribution that U-Net offers is that residual connections are created between the down sampling and up sampling process by concatenating tensor blocks, that were output during the down sampling process, to the output tensor blocks that are obtained by the deconvolution layers that are in the up sampling process [38]. A visualization of the original U-Net architecture can be seen in Figure 1, and the process of how U-Net works as well as the components that make up U-Net are discussed in the following sections.

2.1 Down Sampling Process in U-Net

In this section, we describe how the down sampling process of U-Net works and give details of the terminology that is commonly used when describing convolutional neural networks.

An overview of the down sampling process consists of passing tensor blocks through 2

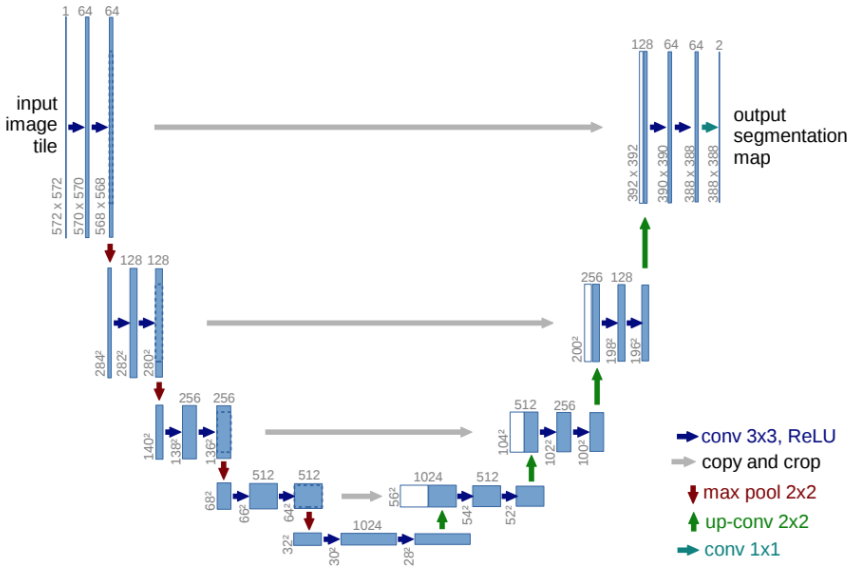


Figure 1: Original U-NET architecture [38]

sets of 3×3 2-D convolutions each followed by a rectified linear unit (ReLU) activation layer, and then a max pool layer that is used for down sampling. These series of steps are repeated 4 times, which is seen in the left half of the U shape of the network in Figure 1.

At the beginning of the network, the input image, represented as a tensor with the dimensionality of Height $\times$ Width $\times$ Channels, has a 2-D padded reflection applied to it before it is passed into U-Net. Doing this increases the size of the height and width dimensions of the input tensor. This is done because the 2-D convolutions and max pooling layers inside of U-Net will decrease the height and width dimensions of the tensors that move through the network. The number of pixels to add to the height and width dimensions of the input tensor, via the 2-D reflection padded, are carefully chosen so that the original image with a 2-D padded reflection applied to it will result in the network outputting a tensor that is the same size in the height and width dimension of the original image. The output tensor produced by the network is the output segmentation map which is applied to the original image to point out where objects of interest are located and therefore needs to have the same dimensionality in the height and width dimensions to accurately display where objects are located. In the original U-Net implementation, input images were of size 388 $\times$ 388 in the

height and width dimensions, and the 2-D padded reflection created a tensor of size 572 $\times$ 572 in the height and width dimensions that was used as the input into the network. These dimensions are labeled at the top left and top right of the network in Figure 1.

After the 2-D padded reflection is applied to the input image, the resulting tensor with increased height and width dimensions are passed through a set of 3×3 2-D convolutional kernels. The output of each of these 3×3 2-D convolutions produces a new tensor block with a height dimension of the input's height minus 2 and a width dimension of the input's width minus 2, and the number of channels is now the number of convolutional kernels that were used. A visualization of 2-D Convolution can be found in Figure 2. A 2-D convolutional layer in a neural network consists of a series of convolutional kernels where each cell of the kernel is a learnable parameter that is updated while optimizing the network. Typically in each convolutional layer there are multiple kernels so that multiple spatial and color features can be detected in an image or subsequent tensor blocks [41]. An example of learned kernels can be seen in Figure 3 where Krizhevsky et al. displayed a visualization of 96 convolution kernels of a layer in a neural network [32].

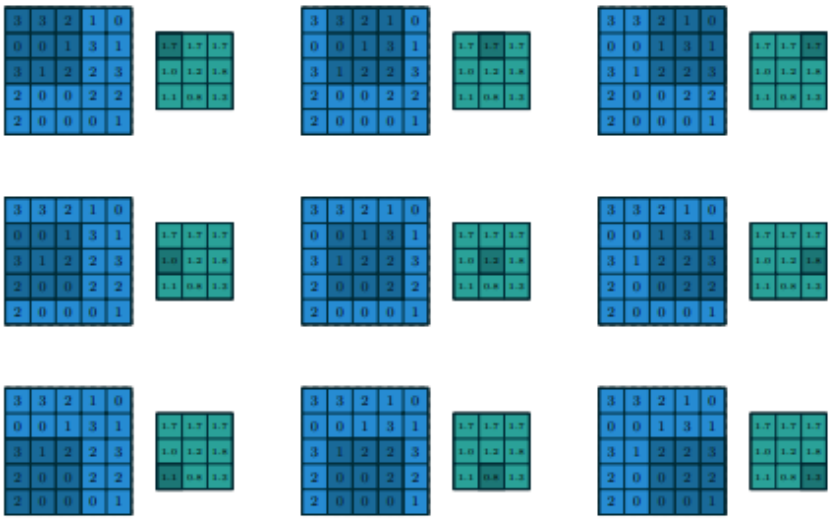


Figure 2: Convolution visualization created by Dumoulin et al., where a 3×3 2-D convolutional kernel is passed over a 5x5 matrix, producing a 3×3 matrix[9]

After a tensor block is then passed through a 3×3 2-D convolutional layer, the resulting

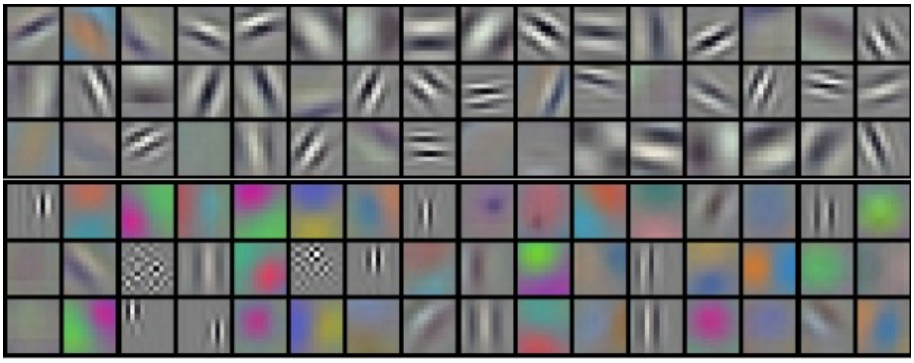


Figure 3: Visualization of characteristic trained convolutional kernel sets [32]

tensor block is then passed through the non-linear activation function called a rectified linear unit (ReLU). Non-linear activation functions are a key part of neural networks that allow the network to learn non-linearities in data during the optimization process [23]. A mathematical definition of the ReLU activation function is given in equation 1.

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

This non-linear function and its variants have become a popular activation function in neural networks because they were able to solve the problem of vanishing gradients in deep neural networks with many layers, that has plagued the extensive use of other activation functions such as sigmoid or hyperbolic tangent functions [23].

Once the tensor block is passed through the ReLU function, it is then passed to a 2×2 max pooling layer to down sample the length and width dimensions by a factor of $\frac{1}{2}$, to down sample the tensor block. The 2×2 max pooling layer can be visualized in Figure 4.

2.2 Up Sampling Process

After the down sampling process, the bottom of the U in U-Net is reached. The tensor block from the max pooling layer is passed into a series 3×3 2-D convolutions which are followed by a ReLU activation function, and then the resulting tensor blocks are passed to

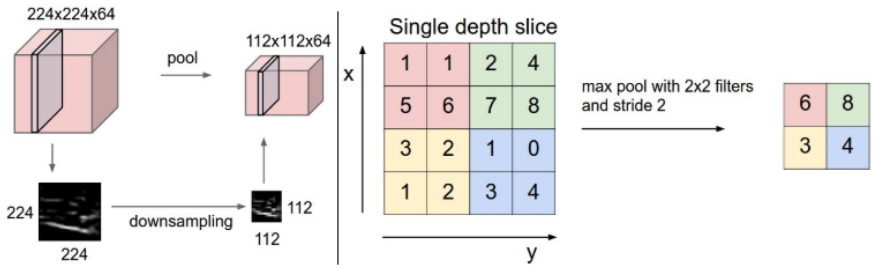


Figure 4: 2×2 max pooling layer visualization created by Stanford cs231 course creators. A 2×2 filter is passed along each channel of a tensor block with a stride of 2, and a new channel is generated by taking the maximum value that the 2×2 filter is on [41].

the up sampling process. During the up sampling process, the authors use sets of 2×2 2-D transposed convolutions that are performed to undo the down sampling from the 2×2 max pooling layers, by doubling the height and width dimensions of the tensor block.

A direct convolution applied to a tensor block can alter the height and width dimensions of the output tensor block, as we have seen with the 3×3 2-D convolution described earlier. The transpose convolution can be thought of as a process that will get you back to the dimensions that you started with before the direct convolution was performed [9]. A visualization of the transpose convolution can be seen in Figure 5.

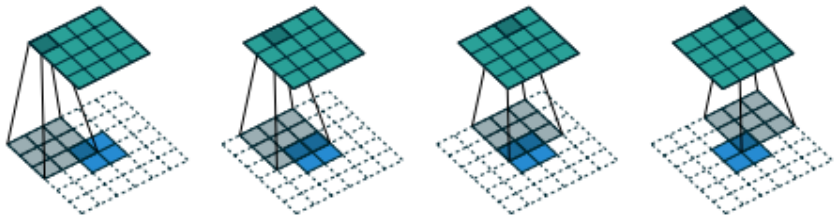


Figure 5: Transposed convolution visualization created by Dumoulin et al., where a 3×3 2-D transpose convolutional kernel is passed over a 2×2 matrix, producing a 4×4 matrix [9]

After a tensor block is passed through a transposed convolution, the tensor block that is at the same level in the U as that of the current level that exists in the down sampling process are cropped to match the length and width dimensions, and then concatenated together. This concatenated tensor block is then passed through two 3×3 2-D convolutions

that are each passed through a ReLU activation layer. After the second ReLU, the tensor block is then passed to the next 2×2 2-D transposed convolution layer. This up sampling process is repeated 4 times, the same number of times the down sampling was performed.

At the top right of the U, i.e. after the last transposed convolution is performed, 2 more 3×3 2-D convolutions each followed by a ReLU are performed, the tensor block is passed through a 1x1 convolution which keeps the height and width of the input tensor block the same and condenses all the channels into the desired number of classes that are being used for segmentation [38]. Since the 2-D padded reflection was applied to the input image, the final output of the network has the same height and width dimensions as the original image, which can be seen in Figure 1. A loss function is then applied to the output tensor and the segmentation target which is then used to optimize the network's parameters to minimize the loss function[38].

2.3 Proposed Variant U-Net Neural Network Architecture

Our proposed variant U-Net neural network does not require any padding to the input image, unlike the original U-Net architecture. Our proposed network also contains a series of 4 down sampling convolutional layers each of which reduces the dimensionality of the previous tensor block by a factor of 2, without needing to use a max pooling layer. Each layer utilizes two different sets of trainable convolution kernels during the down sampling process of U-Net, a set of 4×4 kernels and a set of 8×8 kernels. Having both of these kernels gives the network kernels that can learn finer features or details captured in the 4×4 kernels and larger details in the 8×8 kernels. These kernels produce tensor blocks of the same dimensions that are then passed through a Leaky ReLU function, an Instance Normalization layer, and then concatenated via a residual connection to tensor blocks of the same dimension in the deconvolutional process.

In the last downsampling layer only an 8×8 convolutional kernel is used. The tensor block is then passed through a Leaky ReLU and then through an Instance Normalization

layer.

A Leaky ReLU is a piece wise linear activation function which is a variation on the ReLU that allows for negative inputs to obtain nonzero activation value. When using a Leaky ReLU the user defines a percentage value, p, for the negative input values for function to become a fraction of that value, thus encoding additional gradient information on the trainable parameters into the loss function. In the proposed network, the value for p is set to 0.2 for every Leaky ReLU. The Leaky ReLU is defined in equation 2.

$$f(x) = \begin{cases} x, & x \geq 0 \\ p * x, & x < 0 \qquad (0 < p < 1). \end{cases} \tag{2}$$

Instance Normalization was developed by Ulyanov et al. [45]. The development of Instance Normalization helped improve the ability for fast stylization and was adopted by researchers improving Generative Adversarial Networks that produce images [26][49]. Instance Normalization computes the mean and standard variation across each channel of a tensor block, and then uses that information to normalize the values across the channels [45]. The equation for Instance Normalization can be found in equation (3), where y_{ijk} is the new value for the ijk-th element in the tensor block that Instance Normalization is applied to. i spans the channels, j spans the height, and k spans the width dimensions of the tensor block. x_{ijk} is the ijk-th element in the tensor block. μ_i is the mean of the i-th channel. σ^2 is the variance of the i-th channel. H is the height of the tensor block and W is the width of the tensor block.

$$y_{ijk} = \frac{x_{ijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \mu_i = \frac{1}{HW} \sum_{l=1}^{W} \sum_{m=1}^{H} x_{ilm}, \sigma_i^2 = \frac{1}{HW} \sum_{l=1}^{W} \sum_{m=1}^{H} (x_{ilm} - m\mu_i) \tag{3}$$

The tensor block produced by the last convolutional layer is passed into a deconvolutional layer that has a kernel size of 8×8, a stride of 2, and a dilation of 3 to double the height and width dimension of the tensor block. The tensor blocks obtained from the convolutional

layers in the same level of the U during the down sampling steps are then concatenated onto the current tensor block produced by the deconvolution. As opposed to the original U-Net architecture, our proposed network does not need to crop the tensors that are concatenated from the down sampling process thereby retaining all information that was produced in the down sampling process. This new tensor block is then passed through a Leaky ReLU followed by an Instance Normalization layer. This is repeated 3 more times. The last deconvolutional layer produces a tensor block with the same height and width dimensions as the gray scale input image, and is then passed through a 3×3 2D convolutional layer with stride of 1 and a dilation of 1 to preserve the height and width dimensions and has a depth dimension of 1 to match the dimensionality of the target tensor. This tensor block is then passed through a sigmoid function to produce values between 0 and 1 which represent probabilities of each pixel being considered vegetation or not.

Our proposed variant U-Net neural network architecture is used in the following two applications. A visualization of our proposed network architecture is shown in Figure 6.

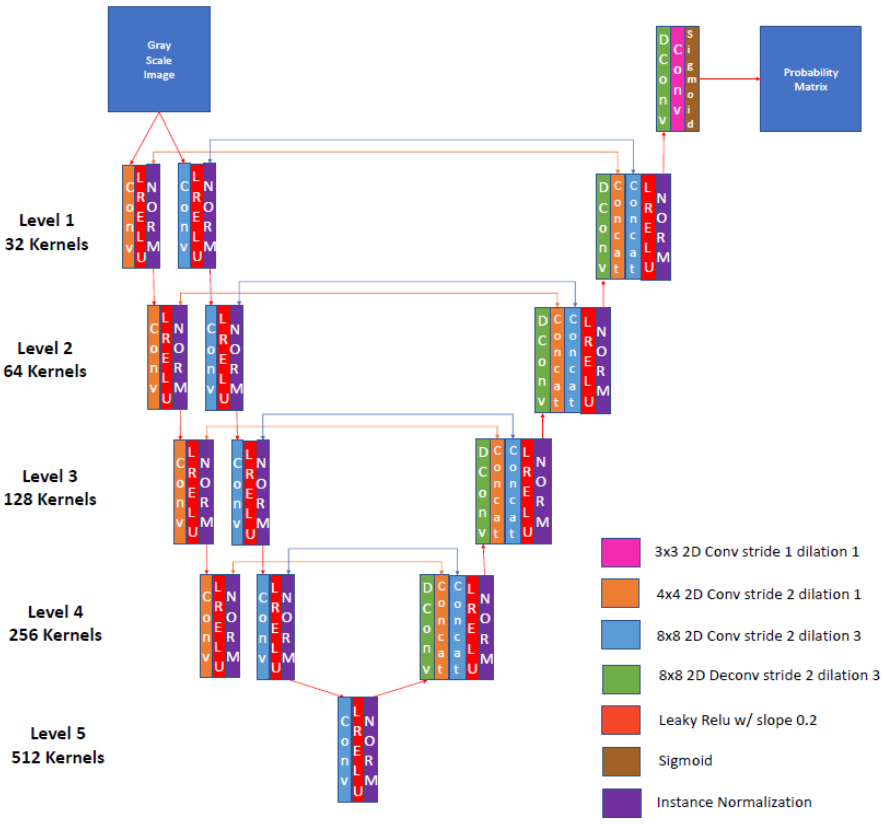


Figure 6: Proposed variant U-Net neural network with two separate convolution tracks (using kernels with 8×8 and 4×4 spatial dimensions respectively) during the down sampling process. The 8×8 and 4×4 convolutions, and the 8×8 deconvolutions have the indicated number of kernels specific to the particular level.

3 Green Index and Applications to Vegetation Classification

In previous work, we developed a green vegetation filter to calculate a green index for locations that span Milwaukee County by utilizing Google Street View's API to collect the images [17]. The green vegetation filter we developed was an attempt to improve upon a green vegetation filter proposed by the authors Li et. al. [44]. Their methodology was able to use color information from a pixel's RGB value to determine whether a pixel is green vegetation. A short coming of their filter is that it would classify artificial green objects as green vegetation, such as street signs, garbage cans, and city buses. A key contribution of

our work improved upon the green vegetation filter by creating a filter that helped eliminate the classification of these artificial objects while maintaining the ability to classify green vegetation pixels [17]. Both of these approaches were used to map a distribution of green vegetation across urban and suburban areas [18, 19].

Using only green vegetation may be a useful tool for mapping the distribution of vegetation since a majority of vegetation in an area such as Milwaukee County is green during the summer months [17]. The approach is limiting for classifying vegetation in image data that was acquired outside of a summer month or if a more complete distribution of vegetation is needed, since there exists many species of plants that are not green. We propose an extension of the work using green vegetation filters to classify vegetation by combining our green vegetation filter, the Milwaukee County Google Street View data set and the proposed U-Net variant network to classify all colors of vegetation rather than just green vegetation.

3.1 Data Collection

Google's Street View was launched in 2007 and enabled users of Google Maps to explore greater metropolitan areas at the street level. Later the service was expanded to a global scale and as of April 2020, Google continues to update and expand the geographical locations worldwide that it serves Google Street View (GSV) images for, available on its Google Maps platform [13]. Along with the ability to interact with street level images via a web interface, Google created an API that allows for developers to download GSV images giving software developers and researchers access to one of the largest outdoors image data resources compiled [14].

In 2015 researchers created image processing methods to detect green vegetation and applied them to a selection from the GSV database [44, 37]. A modification on the green vegetation filter developed by Li et al., was developed by the author of this paper and its collaborators[44, 17, 6]. The modified vegetation detection algorithm was applied to over 90 thousand 360 degree panoramic GSV image sets collected on Milwaukee County in the state

of Wisconsin, a subset of the GSV images of the area taken ranging from the year 2007 to 2017. All images in this Milwaukee County GSV image collection were taken between the months of May through September, and for each location with multiple dates available the ones with the most recent date where included. Details of Milwaukee County's vegetation density map compiled from this data set can be seen in Figure 7 [18, 6].

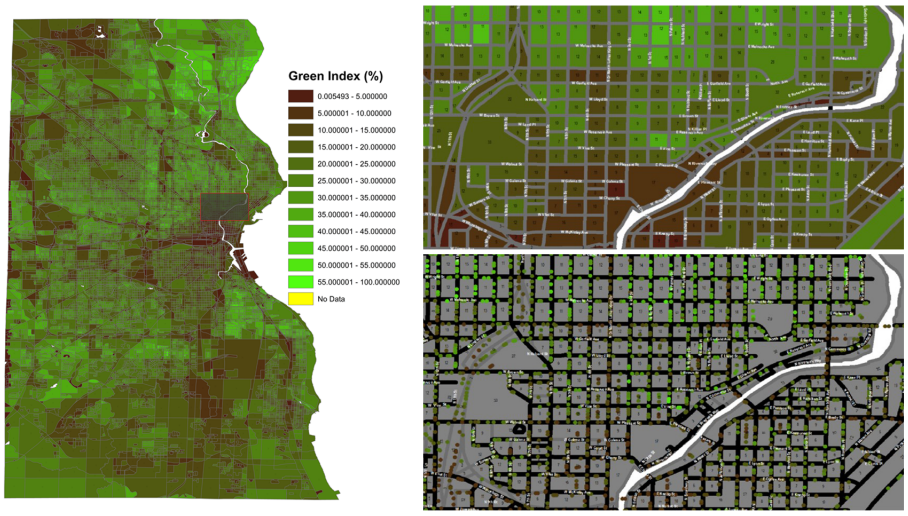


Figure 7: Visualization of locations where GSV images were collected and a map of the compiled green-view index for Milwaukee County [18, 6]. Left: View of all of Milwaukee County. Right Top: Zoomed in image of Milwaukee County. Right Bottom: Green dots indicate the location where a GSV image was taken.

The five months (May through September) were chosen to have images that most likely contained green vegetation. In the early spring and late fall months much of the vegetation is not green, and using image processing algorithms that detect green hues on outdoors images would not capture vegetation image components efficiently, as seen in Figures 8 and 9. Such algorithms also fail to detect foliage with characteristic yellow, purple and other color hues specific to some trees and other vegetation. They also have a tendency of detecting green non-vegetation image components (roofs, vehicles, walls, signs etc.) as false positives when tasked with vegetation detection.

This gives the motivation to create an algorithm that is able to identify other vegetation image components than just green-colored ones. The creation of such an algorithm would

Figure 8: The color-based image processing filter fails to detect purple-colored as well as deep dark green vegetation image components. Left: Original RGB Image. Right: Masked image that has been produced by a color based green vegetation image processing filter [17, 6]

Figure 9: The image processing filter fails to detect yellow-colored vegetation image components. Notice also false positive detection of green components of a building. Left: Original RGB Image. Right: Masked image that has been produced by a green vegetation detecting image processing filter [17, 6]

lead to the ability to expand the months that we would be able to obtain images. This would yield more reliable data and would give a better quantitative measure to how much vegetation is present in an image.

The large GSV data set of Milwaukee County collected in our previous work contains images where the overwhelming majority of vegetation is well captured by the green vegetation filter [17, 18, 6]. Thus such a data set, tagged by a high quality color-based image

processing filter, is expected to be well suited to serve as a training set to train a suitable neural network for vegetation identification in a supervised training setting.

Additionally, as biological neural networks (e.g. the human brain) are very capable of detecting vegetation on images with gray scale information only, the question arises whether artificial neural networks would be able to perform the same task, and how well. In this paper we have set out to investigate this question, by training a convolutional neural network to recognize vegetation on outdoors images with gray scale image information only.

A training data set was curated where each input image was a gray scale version of an image from the Milwaukee County GSV data set and its paired target image was a binary map obtained from the output of the green vegetation filter. In the target image, each pixel was given a value of 1 if it was recognized as vegetation by the green vegetation filter, otherwise the pixel was given the value of 0. The choice of converting all the input images to gray scale images in the training set was done so that our convolutional neural network would be compelled to rely on structural and texture image information only, rather than color information, to predict vegetation components identified in the target image. The vast majority of vegetation contained in the Milwaukee County GSV data set is green vegetation. Since the network would be fed an overwhelming majority of training images where the vegetation image component maps very well to the target image, the gray scale training is expected to lead to better recognition of "other-colored" vegetation image components.

3.2 Green Index Image Processing Filter

We utilize the observations and heuristics outlined above to formulate a multicomponent vegetation image-segmenter, that avoids chronic errors induced by the presence of artificial green colored objects and improper lighting.

In an initial step of the segmentation algorithm, performed in HSV representation, we saturate the images by scaling the S color component by a factor $\kappa > 1$. This is done to compensate for shading variation before the additional filtering steps.

After the pixel color saturation step, to segment the images we apply and combine the results of a series of three independent logical-valued image filters or masks, parametrized by the values $(\alpha, \beta, \gamma) = (1.5, 0.4174, 0.1569)$ and localization parameter $m = 5$:
(1) in the original RGB color space with the parametrized condition $2G > \alpha(R + B)$ we select a restricted subset of the generic green image pixels;
(2) with the hue-limiting condition $H < \beta$, we additionally exclude image pixels with a substantial blue component that is characteristic of many "artificial" green pixel colors; and
(3) using all three RGB channels we apply a binary filter composed from local rank filters to exclude image locations with neighborhoods of low local variations, specifically selecting pixels that satisfy the condition

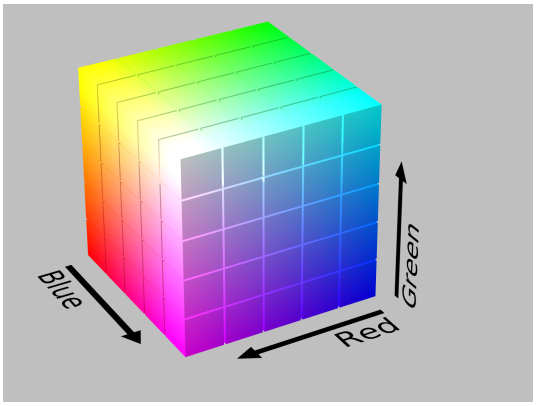
$$min(R_{max} - R_{min}, G_{max} - G_{min}, B_{max} - B_{min}) > \gamma.$$

Here X_{max} and X_{min} are the maximal and minimal rank filters in a local neighborhood of pixels size m, centered at the pixel location, with $X \in \{R, G, B\}$.

The parameter values $\kappa, \alpha, \beta, \gamma$ and the shape and size m of the local neighborhood used in our implementation were determined based on the segmentation performance on a representative test set of several hundred images that were hand-selected and evaluated to include a wide variety of image features occuring in GSV images.

3.3 Training and Results

Using the variant of the U-Net described earlier, we train the network using the annotated gray scale image data set described in the previous two sections. Training of the network was done on a machine with an NVIDIA GeForce 1080 TI GPU with 11 GB of RAM. The Milwaukee County GSV dataset was randomly split into training, validation, and test sets, with 425,353 images in the training set, 50,000 images in the validation set, and 10,000 images in the test set. The network trained for 10 epochs on the training set. Each image of the



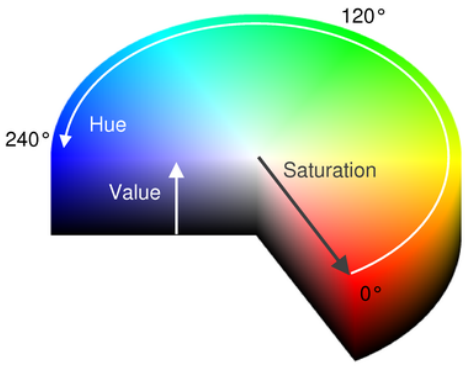


Figure 10: The RGB standard represents pixel colors with values in the unit cube of $\mathbb{R}^3$ in Cartesian coordinates (interpreted as an additive weighted mixture of the primary colors Red, Green, and Blue) or in many implementations simply as a quantized mesh of the scaled unit cube, typically with values ranging from 0 to 255 in each color channel (left). The HSV (i.e. Hue-Saturation-Value) color space (right) represents color in the closed unit cylinder in cylindrical coordinates, with Hue and Saturation being the polar coordinate components, and the cylindrical component Value is related to light intensity.

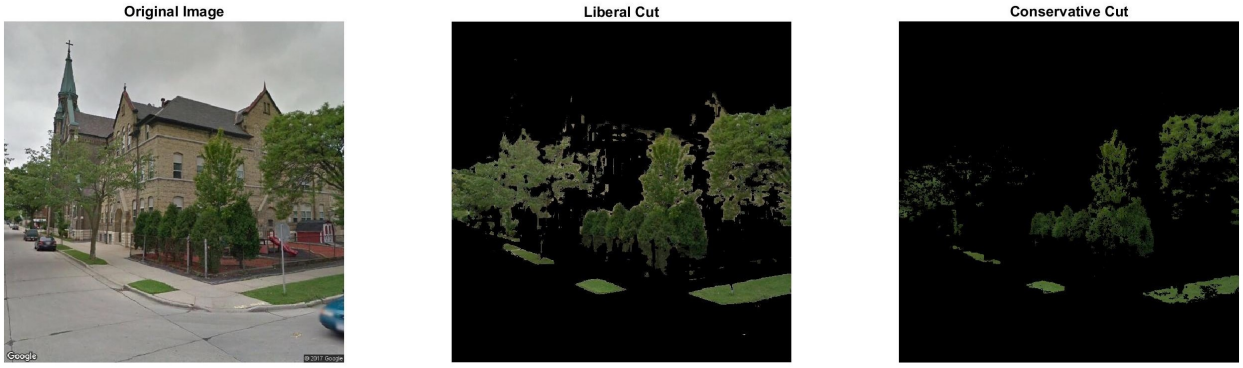


Figure 11: Examples of Errors Induced by Under-Processed and Over-Processed Vegetation Filtering Methods. In the middle image, denoted as the "Liberal Cut" method, we have artificial error induced from creating too wide of a color processing filter. Almost all vegetation within the original image is present, albeit at the inclusion of building structure pixels remaining in the background post-processing. On the right, denoted as the "Conservative Cut" method, we have an over-processed image resulting from too restrictive of a color-processing method. No artificial error is present, albeit at the loss of significant vegetation data.

training set was passed through the network and a binary cross entropy loss was computed between the binary target matrix and the estimated pixel probability matrix. The binary

cross entropy loss is given in equation (4).

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\mathbb{E}_{y \in \mathbf{y}}[ylog(\hat{y}) + (1-y)log(1-\hat{y})] \tag{4}$$

Here, $\mathbf{y}$ is the set of all target labels produced by the green index filter for a given batch of images, $\hat{\mathbf{y}}$ is the set of all predicted classifications produced by the network, y is a sampled target label and $\hat{y}$ is the networks predicted classification for the sampled image associated with the given sample target label. This loss function is what the network learned to minimize. The Adam optimizer was then used to update the parameters of the network to minimize the binary cross entropy on the current image. Adam is a first-order gradient-based optimization method for stochastic objective functions, based on adaptive estimates of lower-order moments. The algorithm created by Kingma et al. is outlined in Figure 12 [30]. In our optimization process we set $\alpha = 0.0002$, and $\beta_1 = 0.5$. We do not use ϵ since we do not use a convergence criteria to stop the optimization process.

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Figure 12: Adam Optimizer Algorithm [30]

After each pass through the entire training set, the average training loss was computed. The validation set was then passed through the network and the average validation loss was also computed. The results of the mean loss of both the training and validation sets, after

each epoch, can be seen in Figure 13.

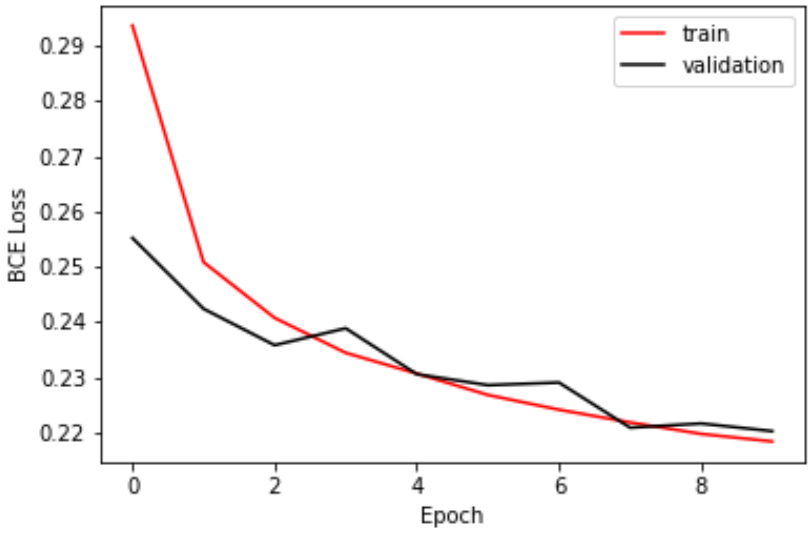


Figure 13: Training and validation loss function over a 10-epoch training period, starting with epoch 0, show balanced training and convergence

Due to the lack of gold-standard annotated data for complex vegetation images, our results for this methodology come in the form of observation and qualitative assessment of our network's output applied to non-grayscale input images. We present a series of before and after images that are created by taking the output image from the network, which is a vegetation pixel probability matrix and multiplying each channel of the original RGB images by the probability matrix. This helps indicate and view where the network estimates that vegetation is in the gray scale version of the image. The process' steps of taking an original RGB image, converting the image to gray scale, passing the gray scale image through the network to get a probability matrix, and multiplying the probability matrix by the original RGB image is visualized in Figure 14.

As can be seen in Figure 15, on the sample images presented in Figures 8 and 9, the network is able to pick up the purple and yellow vegetation as opposed to its "trainer", the color-based image processing green vegetation filter. Figure 15 displays two additional examples of the network's ability to highlight the areas where vegetation exists and suppress

the areas where artificial objects, such as roads, buildings and vehicles are located.

For the interested reader, many more images, paired with their network-produced vegetation estimation, can be found in Appendix A, highlighting the network's remarkable ability to locate vegetation and reject components of artificial objects and structures based on gray scale information only. For the majority of the image selection in Appendix A, our algorithm used for training data generation would fail partially or fail badly when performed on the original colored images, nonetheless the optimized neural network without color information produced high quality segmentation results that are comparable to human performance.

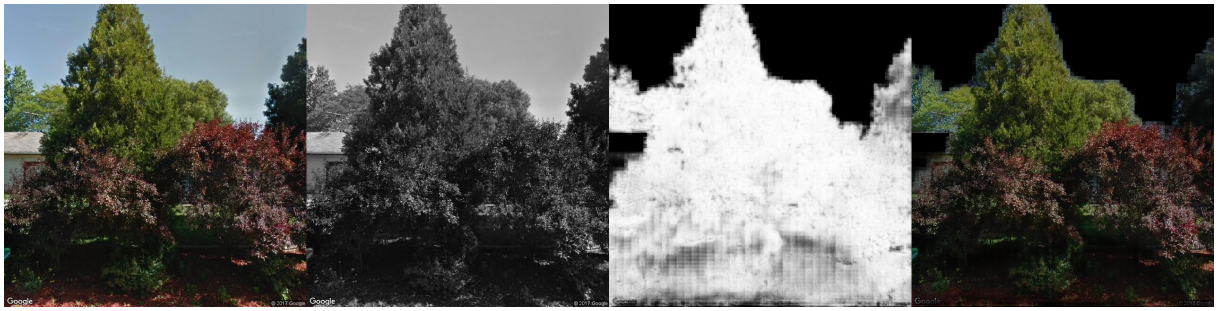


Figure 14: The first image is the original RGB image. The second image is the gray scale image. The third image is the probability matrix predicted by the network and multiplied by 255 to visualize where the network predicted vegetation to be located. The last image is the probability matrix multiplied by the RGB image.

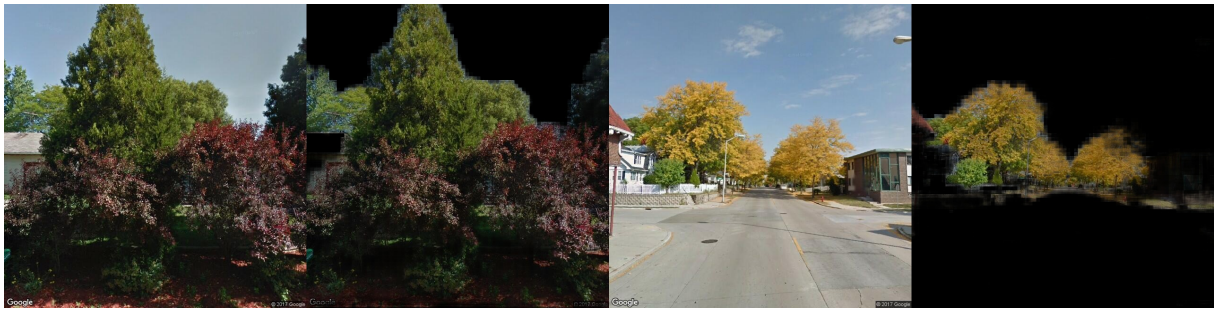


Figure 15: Vegetation estimation using the trained network. Before and after Images for image data seen in Figures 2 and 3

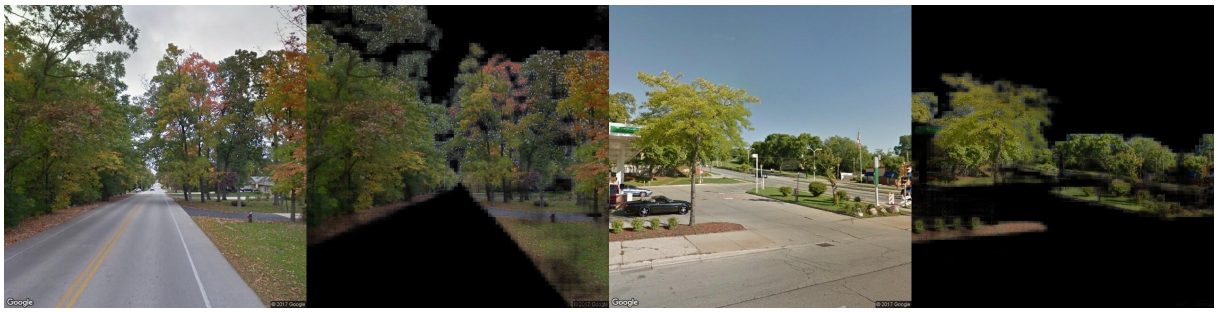


Figure 16: Viewing network performance and robustness. Before and After Images using the trained network

4 Ultra Sound Low Frequency

Ultrasound imaging has become a popular non-invasive medical imaging technique, most notably being used for fetal imaging. Ultrasound imaging is also used for other diagnostic purposes such as diagnosing patients with cancer, and with recent advancements in Doppler ultrasound it used for observing blood flow which can be used to find blockages in the vascular system [4, 5]. It is also a cost effective medical imaging modality as compared to MRI or CT scans [4].Ultrasound has recently been used in handheld devices which is not possible for the imaging modalities of MRI or CT scans [46]. Good image quality in ultrasound imaging is key to leading to better diagnostics and higher end systems, which are more expensive are able to produce high quality images. One of the components of an ultrasound system that is vital for producing high quality images is the ultrasound transducer. The ultrasound transducer is the probe that is applied to body to acquire the images. Ultrasound transducers are sold with varying frequency capabilities, and having a higher frequency transducer is typically associated with higher quality images [40]. There are trade offs between using a higher frequency and a lower frequency transducer. Although the higher frequency transducer is able to produce cleaner images with higher resolution, it is unable to penetrate further in depth into the human anatomy than a lower frequency transducer, which leads to more information at the superficial layers of the anatomy and less information that exists deeper in the anatomy [40]. On the contrary, a lower frequency transducer produces images with more

noise in comparison to a higher frequency transducer, but the lower frequency transducer is able to capture more information that exists deeper in the anatomy [40]. With the recent advancement in deep learning technology, it is possible to give the appearances of an image from one domain to that of another domain, in this case, it may be possible to give the positive aspects of the appearance of a high frequency transducer to an image that was taken from a lower frequency transducer.

We propose using the same variant U-Net neural network that was described earlier, in a new system that utilizes what is called a Generative Adversarial Network and a local estimation filter, to improve the image quality of ultrasound images that were acquired via a lower frequency transducer. This type of problem of translating an image from one domain to another domain, i.e. translating images from a low frequency transducer domain to a high frequency transducer domain, and is called image to image translation. This proposed system was developed using a technique that is common in image to image translation tasks using deep learning, which is called CycleGAN.

4.1 Generative Adversarial Networks

In recent years, Generative Adversarial Networks (GANs) have become the basis for creating generative models and is widely used within the deep learning research community when unsupervised learning is required [29, 11, 12, 26, 25, 45, 47, 28, 49]. The GAN architecture was first proposed by Goodfellow et. al. in 2014. The GAN architecture that Goodfellow et. al. proposed trains two neural networks together to estimate parameters of a generative model for a distribution of a given data set [11]. The two networks that are used in this process are commonly referred to as a generator network, which is the network used for estimating the distribution of a given data set, and a discriminator network that is used for determining if data passed into the network was a sample from the given dataset, or if it was produced by the generator network. This is done by the discriminator network outputting a value between 0 and 1, which represents its estimate of the probability that the input data

belongs to the given data set. Typically, this is implemented by specifying the output layer of the discriminator network to be a sigmoid function. The generator network takes in an arbitrary vector, in the original framework a noise vector with a defined prior $p_z(z)$ is used as an input, and then produces a vector of the same dimensions as the data that is sampled from the data set which is being estimated. An example of outputs from the GAN in the original paper can be seen in Figure 17, where a GAN was trained to produce images of handwritten digits using the MNIST handwritten dataset [34].

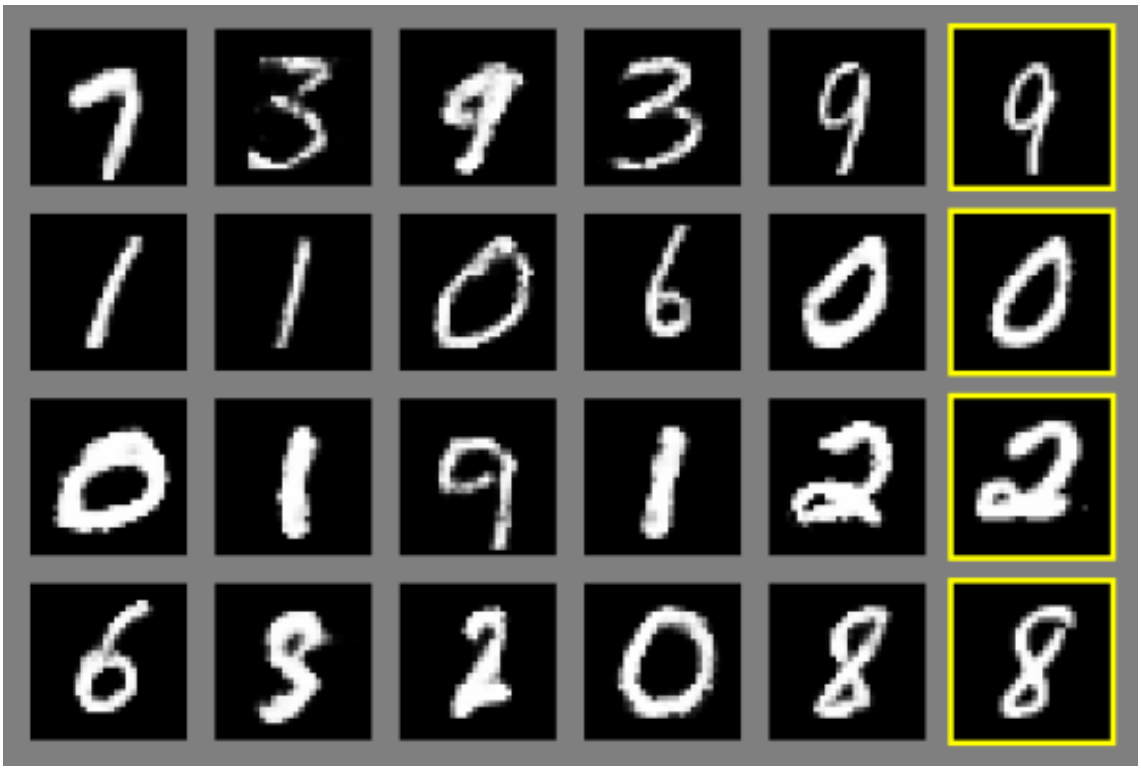


Figure 17: Output of a GAN trained on the MNIST handwritten digit data set, where examples from the MNIST data set are highlighted in yellow and the rest of the images were produced by a GAN [11]

In the original GAN paper by Goodfellow et. al. the authors proposed to train the two networks together by using a two player zero sum minimax game setup, where the setup of the training process is as follows [11]. Let X be a sample space with distribution modeling a given dataset where $X \sim p_{data}$, let $z \in Z$ be a noise vector where $z \sim p_z$, and let G denote the generator network where $G : Z \to X$ and the outputs of G have the distribution

p_g. Let $x \in X$ be a sample data point from X. Let D denote the discriminator network where D maps x to an estimated probability of whether x was sampled from X such that $D : X \to \mathbb{R}_{[0,1]}$. Next, assign ground truth probabilities to x that was sampled from X with the probability of 1 indicating that it is from the real data set, and the output $G(z)$ with the probability of 0 indicating that it was generated (i.e. outputted by G). The minimax game between the two networks is set up by training D to maximize $D(x)$ by assigning the correct ground truth probabilities to x, and training $G(z)$ to minimize the function $log(1-D(G(z)))$. The minimax game is then used with the objective function $\min_G \max_D V(G, D)$ defined to be:

$$\min_G \max_D V(G, D) = \min_G \max_D \mathbb{E}_{x \sim p_{data}}[log(D(x))] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]. \quad (5)$$

Optimizing this function can be done by using common optimization algorithms that are used when training neural networks such as a mini batch stochastic gradient descent or the ADAM optimization algorithm described previously. The Goodfellow et. al. paper detailed an algorithm using mini batch stochastic gradient descent, where a minibatch sample consists of m noise vectors $\{z^1, \ldots, z^m\}$ and m samples from the data set X, $\{x^1, \ldots, x^m\}$ [11]. These samples are then used to update the parameters of the discriminator network by ascending its stochastic gradient

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [logD(x^{(i)}) + log(1 - D(G(z^{(i)})))].$$

The parameters for generator network are updated by descending its stochastic gradient

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} log(1 - D(G(z^{(i)}))).$$

Descending the generator's stochastic gradient, the generator minimizes the log of the probability that the discriminator is correctly classifying the images produced by the generator. This formulation for updating the parameters of the networks which was derived from the minimax game defined in equation 5, was later considered by Goodfellow et. al. and shown that the defined minimax game resembles minimizing the Jensen-Shannon divergence between the distribution of the data set and the distribution of the output data from the Generator Network [11, 12]. This is convenient then for finding that there exists a global optimum such that $p_g = p_{data}$ when $D(x^{(i)}) = \frac{1}{2}$ and $D(G(z^{(i)})) = \frac{1}{2}$ producing an optimal value of for $V(G, D) = -log(4)$.

In a more recent paper by Goodfellow, Goodfellow describes that their initial proposal of the GAN as theoretically convenient for the existence of optimal solutions, but has shortcomings when implemented in practice [12]. In practice, the previously described approach is not guaranteed to converge with Deep Neural Networks since convexity is needed, and neural networks are not convex functions[12]. Since this gave a good framework but performed poorly in practice, a variation was described by Goodfellow [12].

Goodfellow notes that when training GANs, the cost functions can be separated into the cost function for the Discriminator, J^D, and the generator, J^G. The cost function for the discriminator is defined as

$$J^D(D, G) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}}[log(D(x))] - \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))]. \tag{6}$$

This formulation is the cross-entropy cost that is minimized during the training process, which is also the same cost function that was used to train our proposed Network for identifying vegetation. A similar cost function is used for the generator network, which is defined as:

$$J^G(D, G) = -\frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[log(D(G(z)))]. \tag{7}$$

This is also a cross-entropy cost function which is minimized during the training process. In the formulation for the cost function of the generator network is where we see the main contribution of the modification to the original GAN formulation. Instead of minimizing the log of the probability that the Discriminator is correct, the cost function now has the generator maximizing the log of the probability that the discriminator is incorrect [12]. The motivation for this modification is that maximizing $log(D(G(z)))$ produces stronger gradients results in faster learning at the beginning of the training process [12].

Although Goodfellow described a new strategy for training GANs which produced better results, other authors have noted that the sigmoid cross entropy cost function suffers from the vanishing gradients phenomenon[27]. The vanishing gradient problem is a common problem when training deep learning models, and several remedies to this have been proposed such as various cost functions and network activation functions [24, 11]. One of the proposed implementations to help alleviate this issue is to use a Least Squares cost function instead of a cross entropy function [27]. The formulations for the Least Squares cost functions for the Discriminator and Generator are

$$J^D(D,G) = \frac{1}{2}\,\mathbb{E}_{x\sim p_{data}}[(D(x)-1)^2] + \frac{1}{2}\,\mathbb{E}_{z\sim p_z(z)}[(D(G(z)))^2], \tag{8}$$

$$J^G(D,G) = \mathbb{E}_{z\sim p_z(z)}[(D(G(z))-1)^2]. \tag{9}$$

Many other cost function implementations have been shown to improve upon the variation proposed by Goodfellow, such as using a Wasserstein distance cost function [29], but we focus on the Least Squares cost function since it is the cost function that was used in the Networks that produced the results shown in this paper.

Papers describing variations on the original formulation have become prominent in the field of deep learning and are showing results where the generative models are able to produce seemingly indistinguishable data between the data produced by the generator and data sampled from the domain of a given data set [11, 12, 31, 47, 28]. A famous example of realistic

data produced by a GAN comes from a variation on the original GAN called StyleGAN, where the authors were able to train a GAN to produce realistic human faces [31]. An example of this can be seen in Figure 18.

Figure 18: Examples of realistic human faces generated by StyleGAN [31].

GANs have started to be used in the area of medical imaging research [28, 47]. The work proposed in this paper for improving ultrasound imaging requires an unsupervised learning technique, as there is no ground truth data from mapping the images acquired from a low frequency transducer domain to images acquired from a higher frequency image domain. The use of GANs is an ideal candidate to accomplish this task.

4.2 CycleGAN

In 2017, Zhu et al. published a paper detailing an unpaired image to image translation methodology that utilizes two sets of GANs to map an image from one domain to another domain [49]. Examples given in this work include translating paintings by Claude Monet to realistic photographs of the same landscape, translating images of horses to zebras, and

apples to oranges. In recent works in translating medical images from one domain to another, the use of the CycleGAN methodology has shown promising results. Examples include a GAN called HarmonicGAN developed by Zhang et. al. [28] and a structurally constrained CycleGAN developed by Yang et. al. [47]. Zhang et. al. built a methodology based on CycleGAN that was able to translate medical images acquired from a FLAIR MRI and were able to translate it to the T1 MRI image domain and. Yang et. al. developed a CycleGAN with an added a loss function to their system that would help the translation of images from CT to MRI maintain structural consistency during the image to image translation process [28, 47].

The ability to give the appearance of one image to another image without needing to have a ground truth image, which would be needed in all supervised learning tasks, is appealing to the task of improving medical images using two sets of images taken from different equipment, since it is difficult to register the images to recreate a perfect one to one match between images in the different sets. This framework gives researchers the ability to acquire images without needing to worry about the difficulties of harmonized precision of the location of anatomical inquiry when acquiring medical images for these types of tasks. These recent works, paired with our ability to acquire images from two different transducers in which we don't have a perfect match, motivate the use of a CycleGAN for our research problem.

The CycleGAN framework we use relies on two sets of GANs to create an image to image translation from an image in the low frequency transducer domain to the image domain of the high frequency transducer, see Figure 22 for examples of images from both domains. As described in the previous section, both sets of GANs have both a discriminator and a generator network. Let l be an image from L and h be an image from H, where L denotes the low frequency image domain, and H denotes the high frequency image domain. Let G_{LH} denote a Generator Network that maps images from L to H. Let D_H denote the discriminator network that takes in an image as an input and maps the image to a probability that the image came from H. Let G_{HL} denote the Generator Network that maps images from H to L. Let

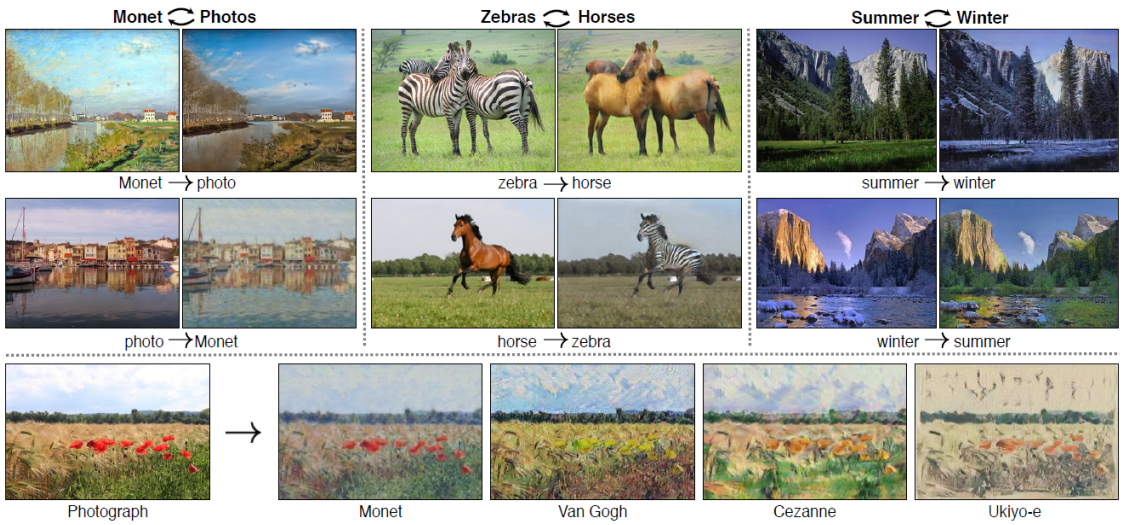


Figure 19: Image to image translation examples from CycleGAN [49].

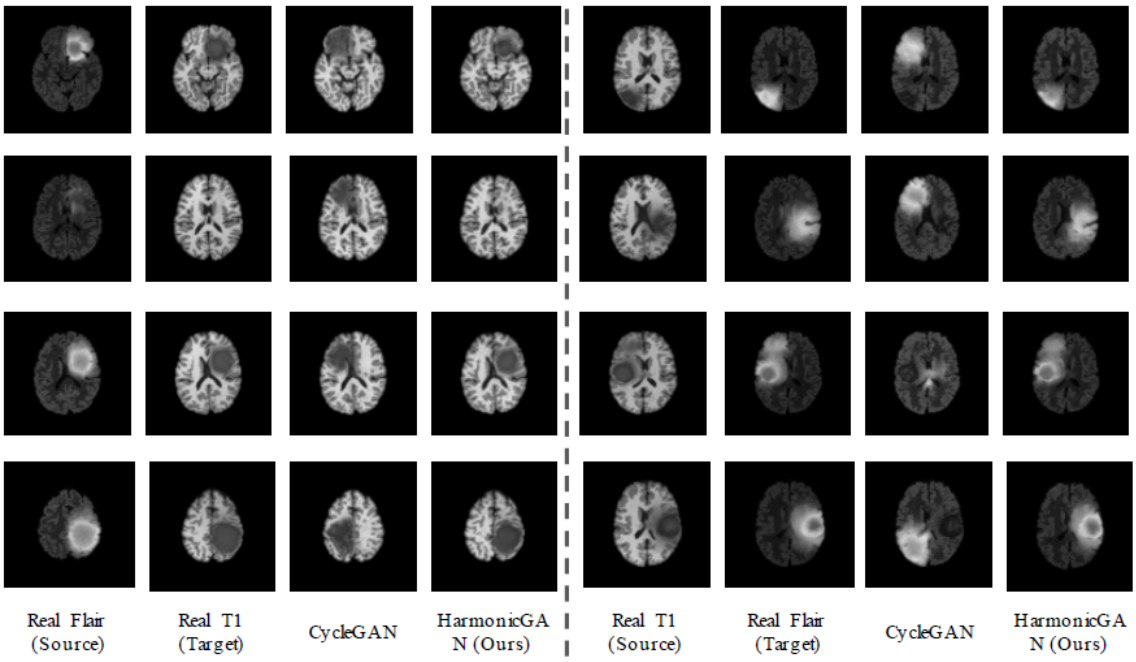


Figure 20: Qualitative comparison of image to image translation of the CycleGAN and Harmonic GAN for translating images acquired from a FLAIR MRI, labeled source, to a T1 MRI image domain, labeled target [28].

D_L denote the discriminator network that takes in an image as an input and maps the image to a probability that the image came from L. As described in the work from Zhu et al. for the networks G_{LH} and G_{HL} to learn the respective mappings of $L \rightarrow H$ and $H \rightarrow L$, three loss functions are used, adversarial loss functions which are Least Squares loss functions, the

cycle consistency loss function and identity loss function. The cycle consistency loss function is a major contribution of Zhu et al [49]. The adversarial loss functions are defined as:

$$\mathcal{L}_{GAN}(G_{HL}, D_L, L, H) = \mathbb{E}_{l \sim pdata(l)}[D_L(l)^2] + \mathbb{E}_{h \sim pdata(h)}[(D_L(G_{HL}(h)) - 1)^2], \tag{10}$$

$$\mathcal{L}_{GAN}(G_{LH}, D_H, L, H) = \mathbb{E}_{h \sim pdata(h)}[D_H(h)^2] + \mathbb{E}_{l \sim pdata(l)}[(D_H(G_{LH}(l)) - 1)^2]. \tag{11}$$

The cycle consistency loss is defined as:

$$\mathcal{L}_{cyc}(G_{HL}, D_L, L, H) = \mathbb{E}_{l \sim pdata(l)}[\|l^* - l\|_1] + \mathbb{E}_{h \sim pdata(h)}[\|h^* - h\|_1], \tag{12}$$

where $l^* = G_{HL}(G_{LH}(l))$, and $h^* = G_{LH}(G_{HL}(h))$. The core idea behind cycle consistency loss is as follows: An image $l \in L$ when passed through G_{LH} will be mapped to H. The image $G_{LH}(l)$ is then passed through G_{HL} which is mapped back into L. The resulting image l^* should give back the same image l thus helping ensure a consistency between the networks when images are cycled through both networks [49]. The cycle consistency is performed for both domains L and H and the L^1 distance between the starting image and the result of passing the image through both Generator networks which maps back into the original domain, is measured and minimized in the optimization process. When minimizing the total loss we multiply an importance coefficient λ_{cyc}, which gives more weight to the loss incurred by $\mathcal{L}_{cyc}$ if the images l and l^*, and h and h^* are not similar to each other [49]. For the training of the networks that produced the results in this paper we used $\mathcal{L}_{cyc} = 10$.

The last loss function that is used is called the Identity loss. Its purpose is to regularize the generators when the sampled images from the generators range are passed in as inputs [49]. The goal is to have the generator output the same image that was the input i.e.

$G_{LH}(h) = h$ and $G_{HL}(l) = l$. When training the networks we multiply the Identity loss by an importance factor $\lambda_{identity}$. For the training of the the networks that produced the results in this paper we used $\lambda_{identity} = 10$. The identity loss function is defined as:

$$\mathcal{L}_{identity}(G_{LH}, G_{HL}, L, H) = \mathbb{E}_{l\sim pdata(l)}[\|G_{HL}((l)) - l\|_1] + \mathbb{E}_{h\sim pdata(h)}[\|G_{LH}((h)) - h\|_1]. \quad (13)$$

Adding the loss functions described above gives the overall loss function that the CycleGAN is minimizing.

$$\begin{aligned} \mathcal{L}(G_{LH}, G_{HL}, D_L, D_H, L, H) = \mathcal{L}_{GAN}(G_{HL}, D_L, L, H), \\ + \mathcal{L}_{GAN}(G_{LH}, D_H, L, H), \\ + \lambda_{cyc}\mathcal{L}_{cyc}(G_{HL}, D_L, L, H), \\ + \lambda_{identity}\mathcal{L}_{identity}(G_{LH}, G_{HL}, L, H). \end{aligned} \quad (14)$$

4.3 Ultrasound data description

The data used to produce the results shown in this paper, was acquired from 15 volunteers. Ultrasound scans were performed on both cephalic veins of the volunteers the left and right arms. The scans framed the cephalic vein on both arms in the transverse plane and video clips were taken from moving the ultrasound transducer from superior to inferior. Two types of transducers were used to acquire the scans. A lower frequency transducer, the 9L-D probe, and a higher frequency transducer, the SP10-16-D probe. The scans were performed and recorded on a GE VOLUSON E8 ultrasound system. Each scan that was performed lasted approximately 5 seconds. Converting the videos to frames resulted in approximately 15 thousand images from both the lower frequency transducer, and the higher frequency transducer. In Figure 22, we selected 3 images from both domains that were representative of the characteristics in both domains L and H. When using the data to train our CycleGAN

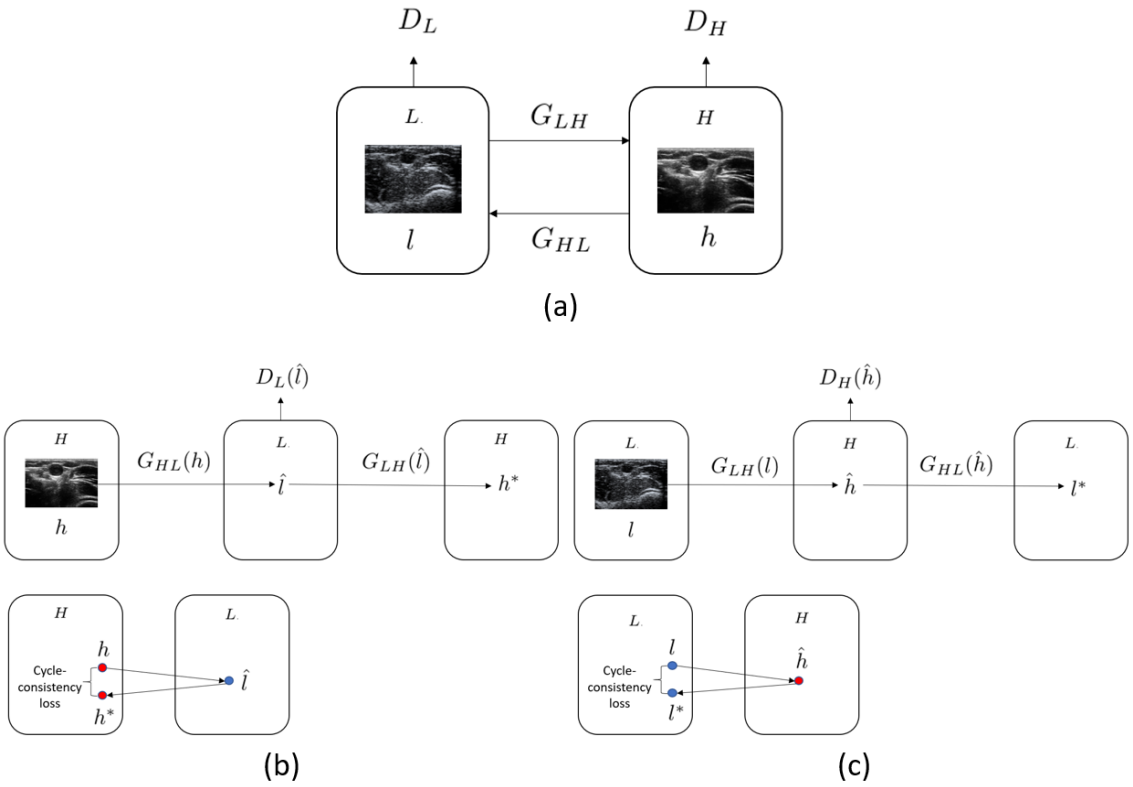


Figure 21: (a) The entire CycleGAN architecture where an image from the low frequency transducer domain, $l \in L$, is used to train D_L and is also mapped to H by G_{LH}, and an image from the high frequency transducer domain, $h \in H$, is used to train D_H and is also mapped to H by G_{HL}. (b) The cycle when starting with a sample $h \in H$. h is mapped to l by the generator network G_{HL} and the estimated image $G_{HL}(h) = \hat{l}$ is produced. $\hat{l}$ is then passed into the discriminator network D_L and the probability $D_L(\hat{l})$ is measured. $\hat{l}$ is then passed into the generator network G_{LH} and mapped back to H and the estimated image $G_{LH}(\hat{l}) = \hat{h}$ is produced and the cycle-consistency loss is measured between h and $\hat{h}$. (c) The cycle when starting with a sample $l \in L$. l is mapped to H by the generator network G_{LH} and the estimated image $G_{LH}(l) = \hat{h}$ is produced. $\hat{h}$ is then passed into the discriminator network D_H and the probability $D_H(\hat{h})$ is measured. $\hat{h}$ is then passed into the generator network G_{HL} and mapped back to L and the estimated image $G_{HL}(\hat{h}) = \hat{l}$ is produced and the cycle-consistency loss is measured between l and $\hat{l}$.

framework, we chose to select random crops of the images and created a local estimation filter approach that is described later, to map images from L to H. In training, when selecting random crops from H, we restricted the area of where we could crop to the top half of an image $h \in H$. The images from both domains are not of the same dimensionality, and resizing the images would result in one of the image domains not being able to maintain is original aspect ratio. This property of the image data sets motivated us to use crops to

create the local estimation filter approach.

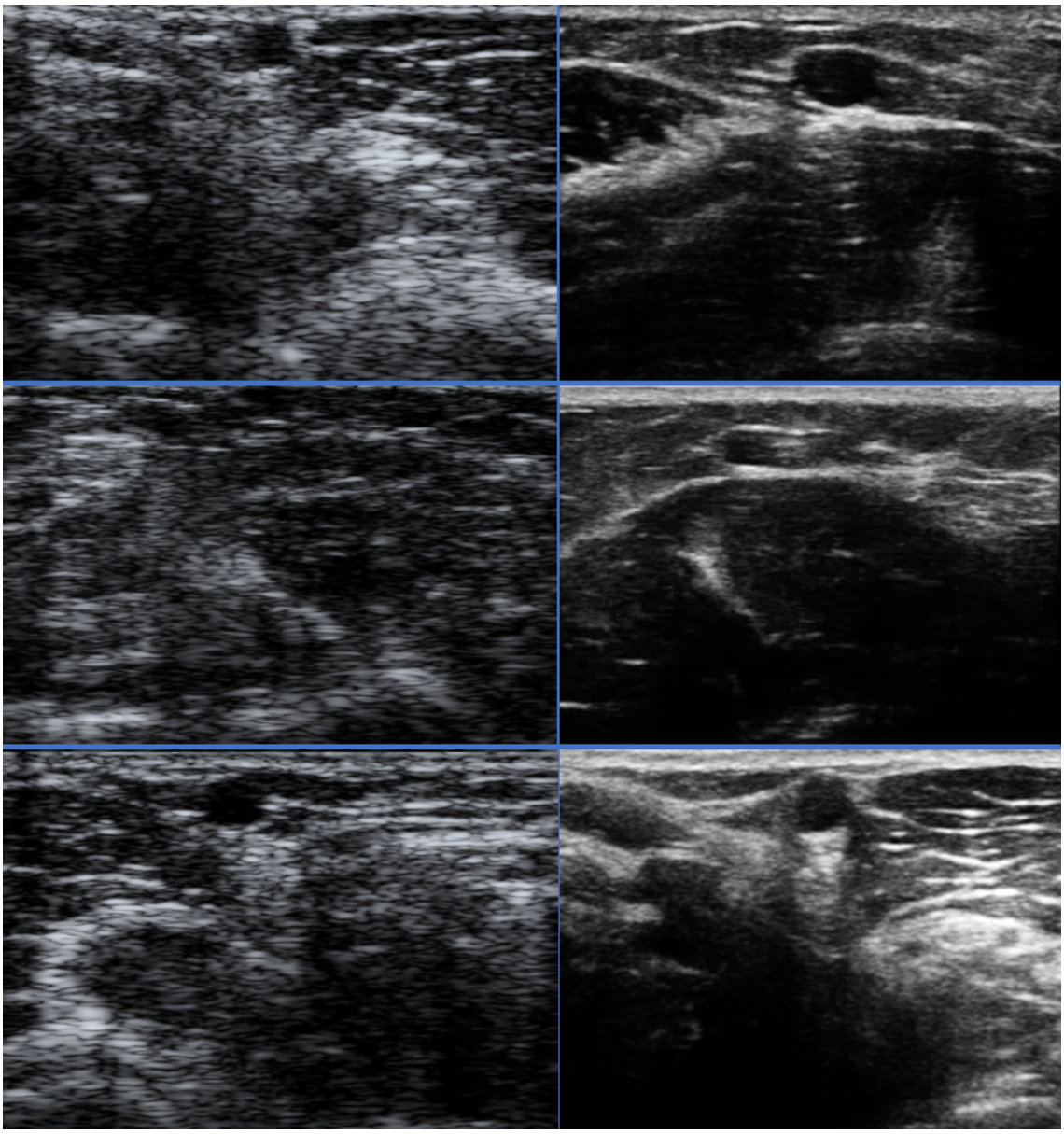


Figure 22: Comparison of 3 Images between the Low Frequency Transducer (Left) and High Frequency Transducer (Right) domains.

4.3.1 Characteristics of the Low Frequency Ultrasound Data

On the left column of Figure 22, there are 3 sample images from L. In these images we see the same type of noise pattern that exists in the 3 sample images. It is harder to distinguish the structures that are in the sample images from L, than it is to distinguish them in the

sample images from H, which is to say that there is a significant difference in the image contrast between the two domains. When looking further in depth in the sample images from L, we see that there exists more structural information that is present in the images than we do when comparing to the sample images from H.

4.3.2 Characteristics of the High Frequency Ultrasound Data

On the right column of Figure 22, there are 3 sample images from H. In these images we notice that there is less noise that exists in the images when compared to the sample images from L. There is better image contrast that exists in the sample images from H than the sample images from L. Further in depth there exist areas where there is less structural information, due to the properties of the attenuation for high frequency transducer signals. This aspect of the images from H led to our decision to take random crops from the top half of the images, when sampling from H, to increase the probability that structural information will exist in the random crop.

4.4 Training

As noted previously, the images from and L and H do not have the same dimensionality between the two domains. To mitigate this issue we implemented a local estimation filter approach where we randomly crop the sampled images, where the random crops have the same dimensionality. Since we are choosing to crop the image instead of rescaling both images, which would have to affect one of the image domains aspect ratio, we are able to preserve the structures that are present in each of the images. The random crops that are taken from the image samples l and h, are used to train the networks D_L, D_H, G_{LH}, and G_{HL}. We develop a filter approach where the final image to image translation of L to H is done after the training process by scanning the filter across the image and employ an averaging where there is overlap.

In preliminary results, it was found that when randomly cropping over the entirety of

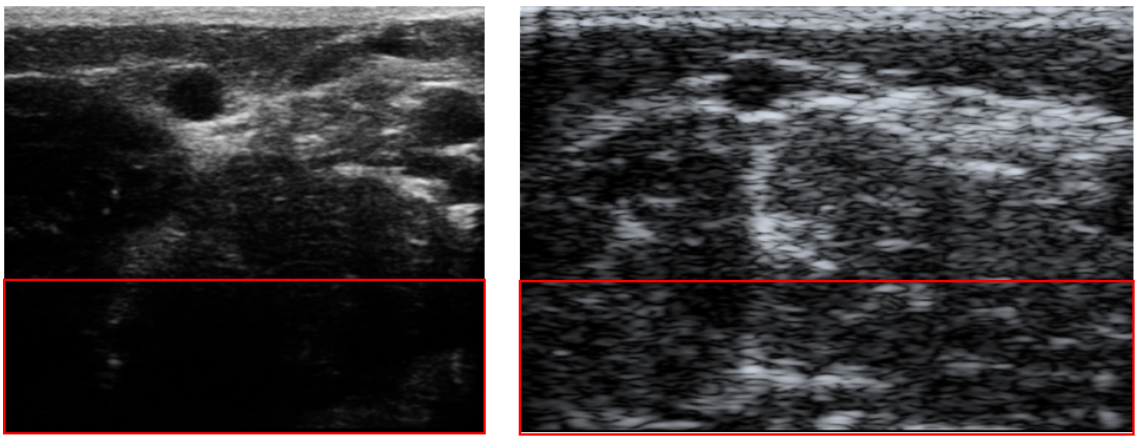


Figure 23: A comparison between image from H with the bottom half of the image almost completely black, and an image from L where information is present throughout the entire image.

images from H many areas that are further in depth have a majority of black pixels which can be seen in the right hand side of Figure 22. This caused the mapping from L to H with the network G_{LH} to produce dark images. For a better local estimation of features that exist in the entire image, specifically for images in L, we chose to only allow the croppings from H to be in the top half of the image where it is more likely that information other than black pixels would be contained. In the images in L, this problem is not as persistent because lower frequency transducers are able to gain information that is deeper in depth than images acquired from a high frequency transducer, therefore we allowed the croppings from L to be anywhere in the image. The croppings used were of dimensions of 128 pixels in depth by 256 pixels in width.

After cropping two images from both L and H, the images were passed into the cycle-GAN process which was described in the previous section. Upon each pass through the network, the networks were updated using the ADAM optimizer described earlier with the same parameters to optimize the networks. The networks were trained for 50 epochs while saving the network parameters at each epoch.

During the training process, a history of the last 50 generated images produced by G_{LH} and G_{HL} are stored inside a history buffer and are used as inputs into D_H and D_L respectively

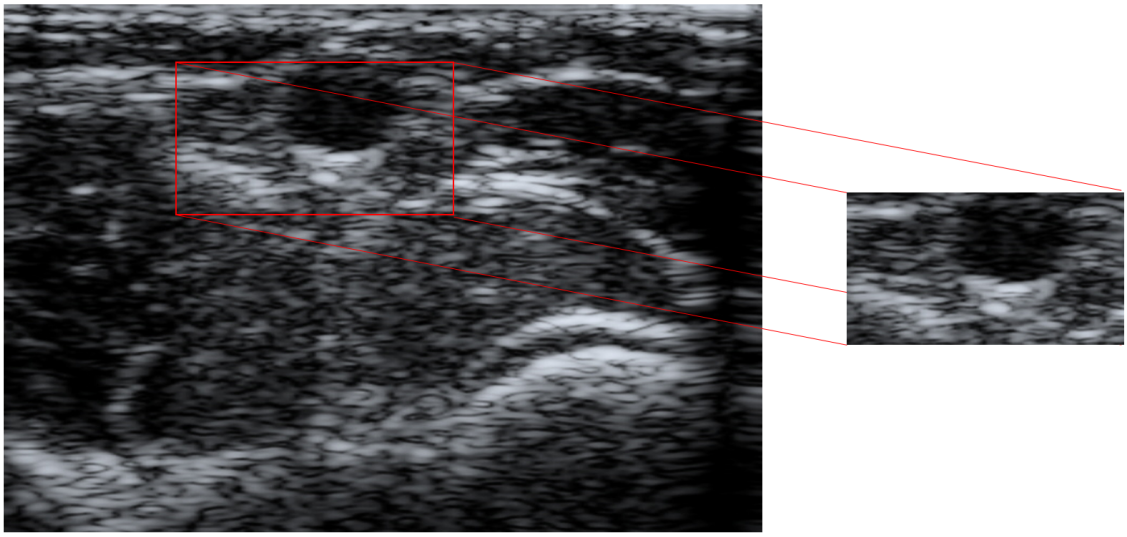


Figure 24: Example of a random crop being taken from an image l from L.

for minimizing $\mathbb{E}_{h\sim pdata(h)}[D_H(h)^2]$ when $h = G_{LH}(l)$ and $\mathbb{E}_{l\sim pdata(l)}[D_L(l)^2]$ when $l = G_{HL}(h)$ [39, 49]. A random image from the history buffer is used to feed into the loss functions during the training process. This gives the Discriminators images that are from past iterations of the generators rather than consistently giving the Discriminators images produced by the current state of the Generators. This was proposed by Shrivastava et. al. to reduce model oscillation.

In practice, the common way of training the networks in the CycleGAN architecture is not obvious with the loss functions listed above. Often the Loss functions are split into two sections when training the CycleGAN, a section for Training the generator networks, and the other section is dedicated to training the discriminator networks [11, 12]. We provide an algorithm that is a common way of training a CycleGAN, and was used to produce the results shown in this paper.

Algorithm 1: CycleGAN Training Algorithm

Result: G_{LH}, G_{HL}, D_L, D_H
initialization;
set numEpochs;
set numItersPerEpoch;
set λ_{cyc};
set $\lambda_{identity}$;
epoch = 0;
while *epoch* $<$ *numEpochs* **do**
  iter = 0;
  while *iter* $<$ *numItersPerEpoch* **do**
    l = Sample From L without replacement and perform random crop;
    h = Sample From H without replacement and perform random crop;
    $l_{identity} = G_{HL}(l)$;
    $h_{identity} = G_{LH}(h)$;
    $\hat{h} = G_{LH}(l)$;
    $\hat{l} = G_{HL}(h)$;
    Add $\hat{l}$ and $\hat{h}$ to the replay buffers Buffer(L) and Buffer(H);
    $l^* = G_{HL}(\hat{h})$;
    $h^* = G_{LH}(\hat{l})$;

    # Train the generator networks
    $\mathcal{L}_{identity} = \|l_{identity} - l\|_1 + \|h_{identity} - h\|_1$;
    $\mathcal{L}_{GAN} = (D_L(\hat{l}) - 1)^2 + (D_H(\hat{h}) - 1)^2$;
    $\mathcal{L}_{cyc} = \|l^* - l\|_1 + \|h^* - h\|_1$;
    $\mathcal{L}_{total} = \mathcal{L}_{identity} + \mathcal{L}_{GAN} + \lambda_{cyc}\mathcal{L}_{cyc}$;
    Update the parameters of G_{LH} and G_{HL} by a gradient descent method to minimize $\mathcal{L}_{total}$;

    # Train the discriminator networks
    $\hat{l}$ = sample from Buffer(L);
    $\mathcal{L}_{D_L} = \frac{(D_L(l)-1)^2 + D_L(\hat{l})^2}{2}$;
    Update the parameters of D_L by a gradient descent method to minimize $\mathcal{L}_{D_L}$;
    $\hat{h}$ = sample from Buffer(H);
    $\mathcal{L}_{D_H} = \frac{(D_H(h)-1)^2 + D_H(\hat{h})^2}{2}$;
    Update the parameters of D_H by a gradient descent method to minimize $\mathcal{L}_{D_H}$;
    iter += 1;
  end
  epoch += 1;
end

4.5 Local Estimation Filter

In this work we propose a local estimation filtering process that is applied to the entire image. This is done since the network is trained on random crops of the training images from both L and H, and the goal is to ultimately map an entire image from L to H. The proposed local estimation filtering process is to partition the entire image into overlapping crops that span the entire image $l \in L$. Each of these crops are then passed into G_{LH}, and the resulting cropped images are stored in memory as well as where each of the pixels in the cropped images are located on the original image l. When all the partitioned crops are passed through G_{LH} a reconstruction of the original image is done by averaging all computed pixel values corresponding to each location on the original image. The result of this pixel-averaging process is presented as our approximation of the translated high frequency image. A more detailed description of this algorithm is described below, and provide images describing the Filter Process algorithm which can be seen in Figures 25, 26, 27, and 28.

Let D and W be the depth and width, respectively, of the image l from L. Let c denote the current cropped image. Let c_d and c_w be the crop depth and crop width of the image from L respectively, in this work we use a crop depth of 128 pixels and a crop width of 256 pixels. Let i denote a pixel at depth i in image l and let j denote a pixel at width j in image l. Next, we create 3 separate matrices. A matrix of dimensions D and W used for counting the number times a pixel at (i,j) was contained in a crop, which we will denote as Count. Another matrix of dimensions D and W used for adding the output pixel intensity of $G_{LH}(c)$ at (i,j) for all c, which we will denote as Intensity. The third matrix is of dimensions c_d and c_w which is a matrix of all ones, which we will denote as Ones. Let row and col denote the current row and column that the top left of the c is located on l. Let m be the number of pixels to slide the next crop along the width axis of the image, and let n be the number of pixels to slide the next crop along the depth axis of the image. In algorithm 2, the convention $l(i : j, l : k)$ is used to denote where the image, in this case image l, is being cropped. i denotes the row of the top of the crop, j denotes the the row of the bottom of

the crop, l denotes the left column of the crop, and k denotes the right column of the crop on the image being cropped.

Algorithm 2: Filter Process

Result: Write here the result
row = 1;
while *row* $< D - c_d$ **do**
    # Section 1
    col = 1;
    while *col* $< W - c_w$ **do**
        c = l(row:row+c_d, col:col+c_w);
        Intensity(row:row+c_d, col:col+c_w) += $G_{LH}(c)$;
        Count(row:row+c_d, col:col+c_w) += Ones;
        col += m
    end
    # Section 2
    c = l(row:row+c_d, W - c_w:W);
    Intensity(row:row+c_d, W - c_w:W) += $G_{LH}(c)$;
    Count(row:row+c_d, W - c_w:W) += Ones;
    row += n
end
# Section 3
col = 1;
row = D - c_d;
while *col* $< W - c_w$ **do**
    c = l(row:D, col:col+c_w);
    Intensity(row:D, col:col+c_w) += $G_{LH}(c)$;
    Count(row:D, col:col+c_w) += Ones;
    col += m
end
# Section 4
c = l(row:D, W - c_w:W);
Intensity(row:D, W - c_w:W) += $G_{LH}(c)$;
Count(row:D, W - c_w:W) += Ones;
row += n;
FinalImage = Intensity / Count;

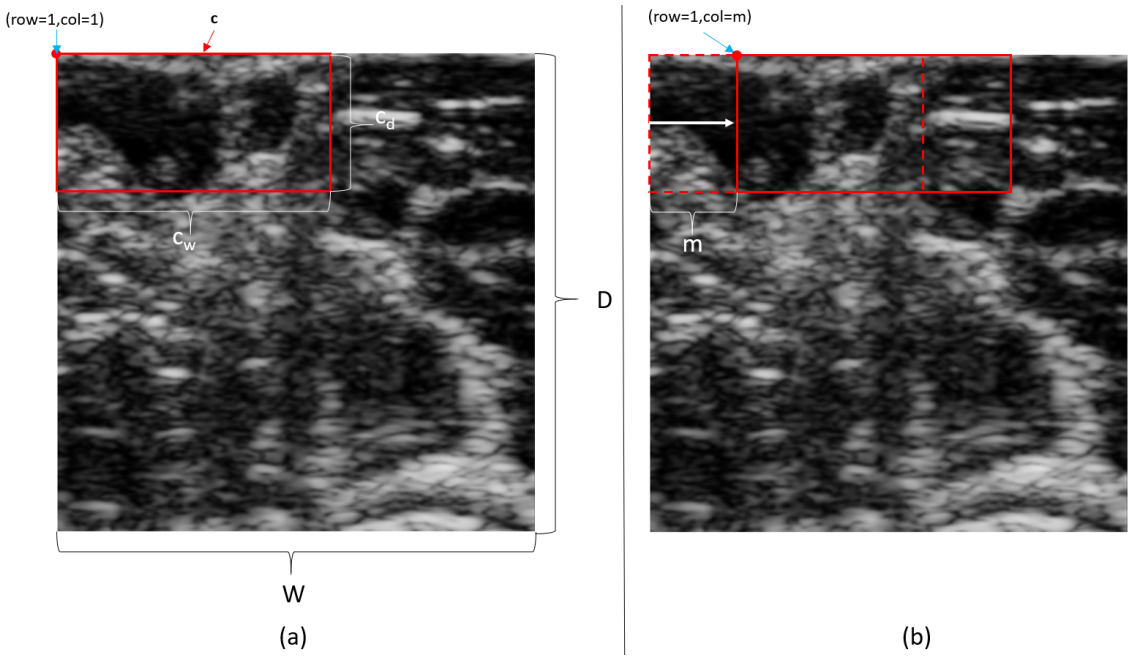


Figure 25: (a) The visualization of the initial state of the filtering process being applied to a image from L, which is the beginning of section 1. The image is of dimensions D in depth and W width. The local estimation filter, G_{LH}, is applied to the crop c of size c_d by c_w, that has its top right pixel located at row 0 and column 0. (b) The visualization of the while loop in section 1. A new crop, which is depicted by the solid red rectangle, taken m pixels to the right of the original crop, depicted as the red rectangle that has dashed lines.

4.6 Results

After the entire training process was over, we then passed a set of test images through the filter process for each of the saved network parameters. This was done since the results of this work are evaluated qualitatively rather than quantitatively. The typical approach using the network parameters that are at the minimum of the validation losses of all the epochs might not be the best network parameters to use. We show results that we considered to be the best qualitative results from all the epochs and discuss the shortcomings of the current approach. More examples of the results can be found in Appendix B where the input image l, located on the left hand side, is passed through the trained G_{LH} generator network and the output of the network that maps l to H can been seen on the right.

Overall the local estimation filter is able to smooth out noise that is present in L, and gives the appearance that the image was from H. The contrast in the resulting image is

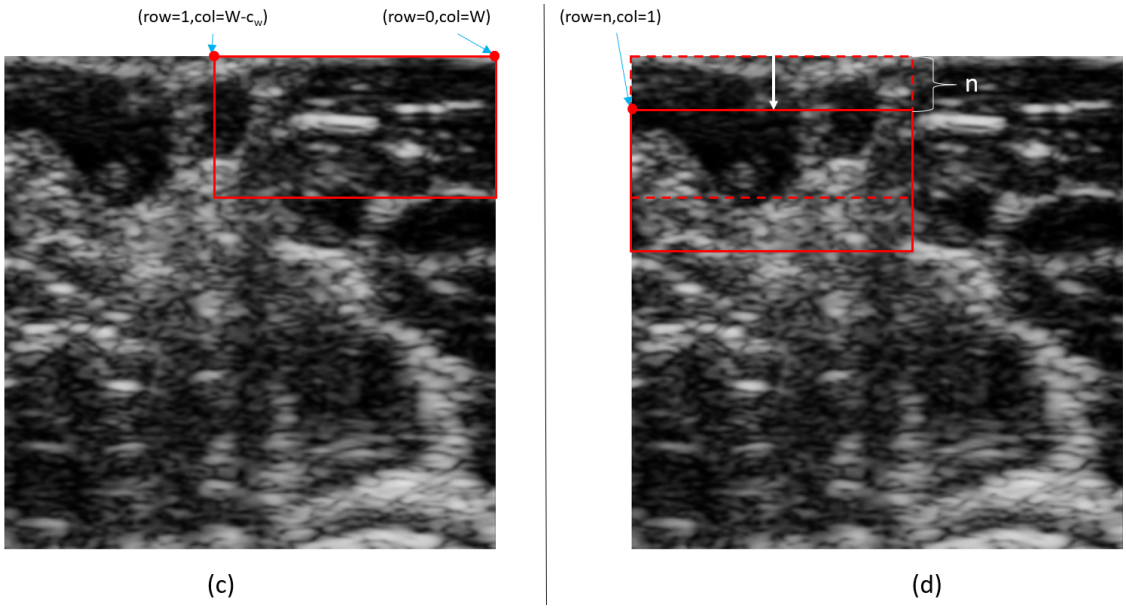


Figure 26: (c) The visualization of the start of section 2, where a crop needs to be taken on the right edge of the image. This done to ensure that the filter is applied to the pixels on the right edge of the crop since it is possible to set m, the number of pixels to move the crop along the width axis, to be an integer such that $W - c_w \bmod m \neq 0$. This case would result in some pixels towards the right edge of the image to not have the filter applied to it if section 2 were not implemented. (d) The visualization of the end of section 2 in the filtering process. A new crop, which is depicted by the solid red rectangle, taken n pixels to the below of the original crop, depicted as the red rectangle that has dashed lines. The process depicted in (a), (b), (c), and (d) is repeated until $row \geq D - c_d$.

improved which gives more definition to the structures that are in the image. There are still shortcomings to this current results that would need to be improved upon before it were to be considered acceptable for use. In the resulting images, the local estimation filter will delete information from the original image from L that the filter is being applied to. This occurs when pixels from the original image don't have a high intensity, such as in the areas where an orifice is present. There are artifacts that are produced in the resulting images after the local estimation filter is applied. One artifact that is always present are dim vertical lines on the left and right sides of the image. Bright speckle artifacts will sometimes occur in the images as well. The most concerning shortcoming is that there are times when the filter will give the appearance of an orifice present when in fact there is no orifice present but is instead tissue.

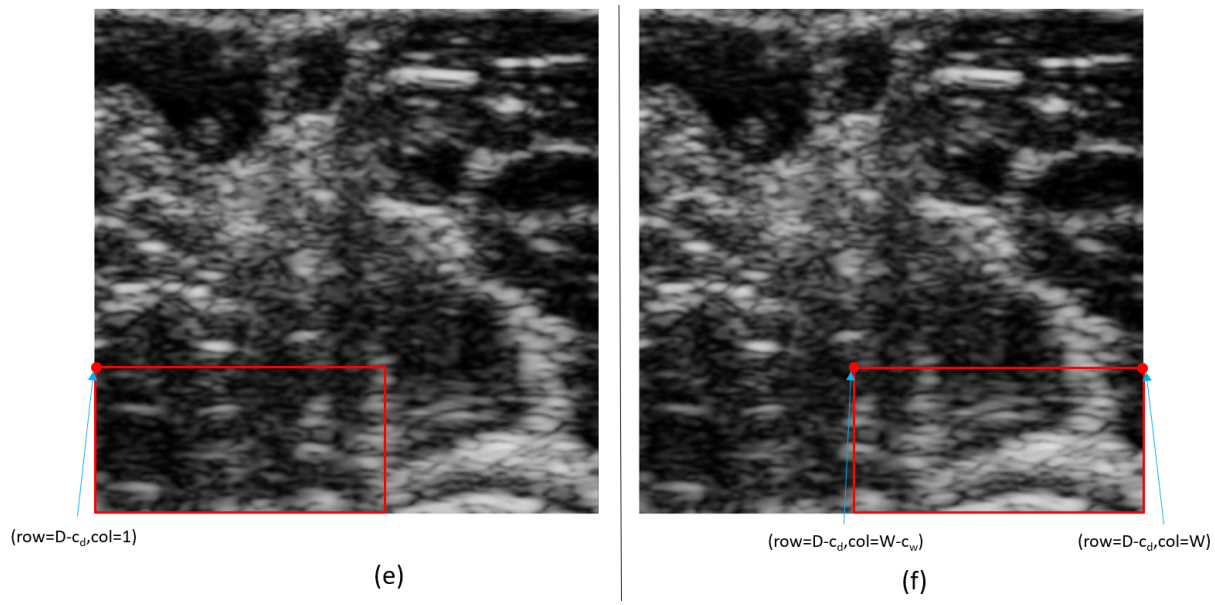


Figure 27: (e) The visualization of the start of section 3, where a crop needs to be taken on the bottom edge of the image. This done to ensure that the filter is applied to the pixels on the bottom edge of the crop since it is possible to set n, the number of pixels to move the crop along the width axis, to be an integer such that $D - c_d \bmod n \neq 0$. This case would result in some pixels towards the bottom edge of the image similar to the reasoning given in for including section 2. (f) The visualization of section 4 in the filtering process. Section 4 ensures that the last crop taken includes the pixels in the bottom right corner of the sample image l.

An implementation of this work can be found at https://github.com/hadam1993/Ultrasound/blob/master/CycleGANUltrasoundExperiment.ipynb.

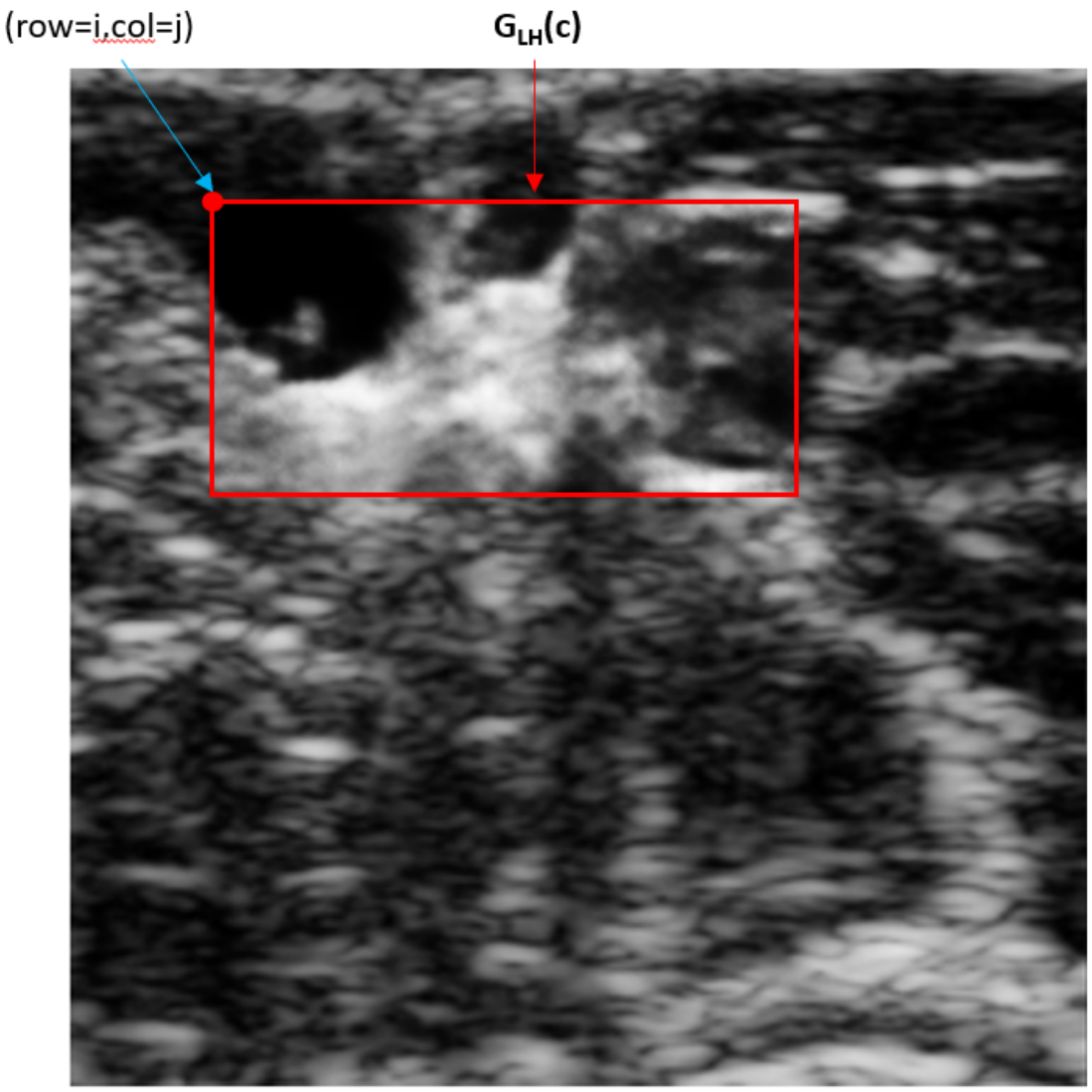


Figure 28: An example of the local estimation filter G_{LH} being applied to the sample image l at a location of (i, j) at an arbitrary location in the filtering process. The resulting crops pixel values are applied to the Intensity matrix at the correct indices that correspond to the pixels of l that were included in the crop c.

5 Conclusion

From both the applications presented and the results of the proposed variant U-Net neural network display promising results to applications that are not similar to one another, showing that this type of network is a useful tool for a wide range of tasks involving images. Another contribution from this work is that we were able to show that what would have been a time

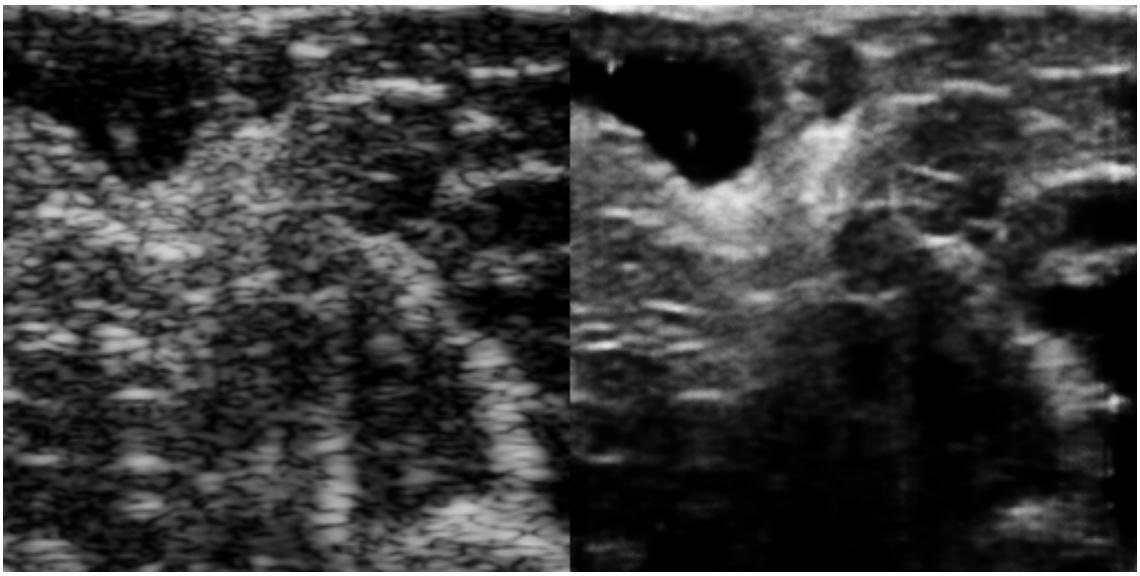


Figure 29: A comparison between the original image from L on the left, and the resulting image from the local estimation filter applied to the image on the right. In the resulting image with the local estimation filter, the filter was able to smooth out the noise that is present in the image from L and create a more clear image with better contrast than what the low frequency transducer was able to produce. The resulting image does not preserve all of the information at the bottom of the image as wanted. It also has a few speckle artifacts that exist in the top left orifice.

consuming task of annotating our large data set of Google Street View images could be automated by making use of our green vegetation filter. This automated data annotation of green vegetation of images could be used to train a neural network that extended our ability to be able to classify any color of vegetation by exploiting the fact that our large data set contained a majority green vegetation and that our green vegetation filter was able to effectively locate most green vegetation present in an image. This was done by converting our image data set to gray scale and pairing them with the labeled image of where green vegetation pixels were present, which forced the network to learn spatial characteristics of vegetation within an image rather than using color information. The last contribution was that the same variant U-Net neural network could be used as a local estimation filter to give ultrasound images acquired from a low frequency transducer the appearance that it was acquired from a high frequency transducer. We showed that using random crops from both image domains could be used to train our network to map image crops from one image domain

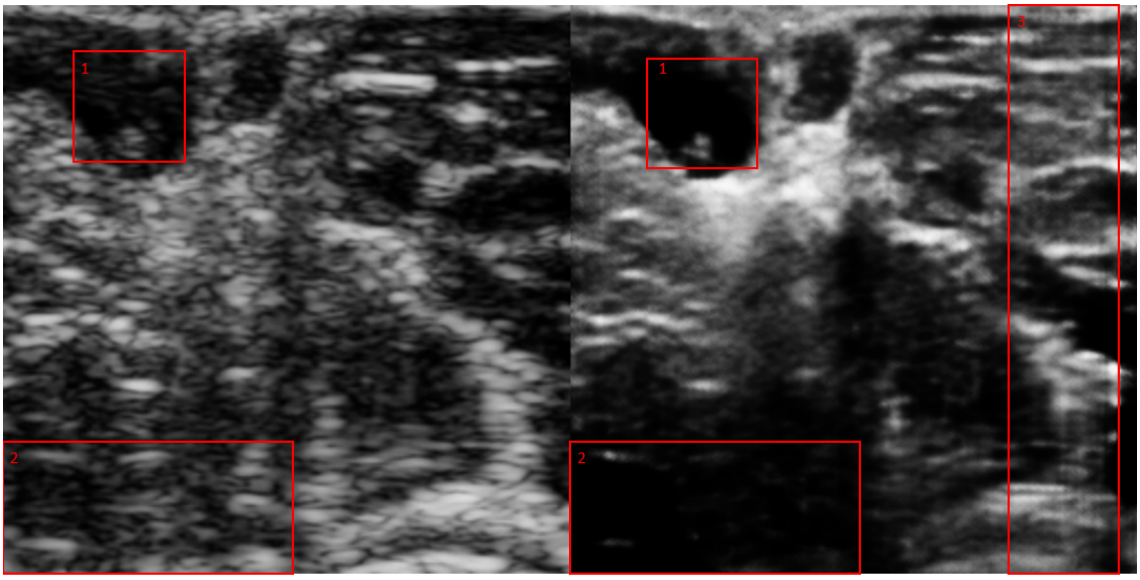


Figure 30: A comparison between the original image from L on the left, and the resulting image from the local estimation filter applied to the image on the right with areas of the image labeled that are shortcomings of the current Approach. In 1 and 2, on the original image from L we see that there exists information inside the red boxes. In the resulting image on the right, the information has been deleted by the local estimation filter. In the box on the right labeled 3, there are vertical line artifacts that are produced by the local estimation filtering process.

to the other to learn local estimations of the image domains. This trained network could then be used by taking crops that partition of an image from the low frequency transducer domain, and average the resulting pixel locations from the output of the trained network to produce an ultrasound image that has the appearance that it was acquired from a high frequency transducer.

References

[1] Blog Post, *More Data for Clearer Skies in Los Angeles*, Nov. 6, 2017, Aclima, https://blog.aclima.io/

[2] Blog Post, *Mapping the invisible: Street View cars add air pollution sensors*, [accessed: January 2018], Google Environment, https://environment.google/projects/airview

[3] P. Blanc-Durand, A. Gucht, N. Schaefer, E. Itti, J. Prior. 2018. Automatic lesion detection and segmentation of 18F-FET PET in gliomas: A full 3D U-Net convolutional neural network study. PLOS ONE. 13. e0195798. 10.1371/journal.pone.0195798.

[4] S. M. Bierig, A. Jones. 2009. Accuracy and Cost Comparison of Ultrasound Versus Alternative Imaging Modalities, Including CT, MR, PET, and Angiography. Journal of Diagnostic Medical Sonography, 25(3), 138–144. https://doi.org/10.1177/8756479309336240

[5] A. Oglat, M. Matjafri, N. Suardi, M. Oqlat, M. Abdelrahman, A. Oqlat. 2018. A Review of Medical Doppler Ultrasonography of Blood Flow in General and Especially in Common Carotid Artery. J Med Ultrasound. 2018;26(1):3-13. *doi* : $10.4103/JMU.JMU_1 1_1 7$.

[6] A. Honts, I.G. Lauko, J. Beihoff, S. Rupprecht. 2020. Local color and morphological image feature based vegetation identification and its application to human environment street view vegetation mapping, or how green is our county? Geo Spat. Inf. Sci. 2020, 23, 222–236.

[7] B.Y. Cai, X. Li, I. Seiferling, C. Ratti. 2018. Treepedia 2.0: Applying Deep Learning for Large-scale Quantification of Urban Tree Cover. arXiv:1808.04754.

[8] C. Davenport, Google's Street View cameras are getting an upgrade, Sept. 5, 2017, https://www.androidpolice.com/

[9] V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning. 2018. arXiv:1603.07285v2.

[10] L.M.G. Fonseca, G.A. Boggione, A.M.V. Monteiro, R. Santos. 2012. ANALYSIS OF THE RELATIONSHIP BETWEEN INTRA-URBAN VEGETATION CHANGE AND SOCIO-ECONOMIC DATA, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XXXIX-B2, 57–62 https://doi.org/10.5194/isprsarchives-XXXIX-B2-57-2012

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. 2014. Generative adversarial networks. In NIPS'2014

[12] I. Goodfellow. 2016. NIPS 2016 Tutorial: Generative Adversarial Networks.

[13] Website, *Google Street View*, [accessed: April 2020], Google Maps, https://maps.google.com/intl/en/streetview/understand/

[14] Website, *Google Street View Developer API*, [accessed: April 2020], Google Maps Platform, https://developers.google.com/maps/documentation/streetview/intro

[15] I. Harbas, M. Subasic. 2014. Detection of Roadside Vegetation Using Features From The Visible Spectrum, 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 26-30, DOI:10.1109/MIPRO.2014.6859751.

[16] K. He, X. Zhang, S. Ren, J. Sun. 2016. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, doi: 10.1109/CVPR.2016.90.

[17] Github, A. Honts, Green Vegatation Image Processing Filter. https://github.com/hadam1993/MKECountyGreenIndex/blob/master/ImageProcessing/green_veg_filter.py

[18] Website, A. Honts, Milwaukee County Green Index. www.mke-green-index.adamhonts.com

[19] Website, MIT Senseable City Lab, Treepedia. http://senseable.mit.edu/treepedia

[20] Website, National Institute of Biomedical Imaging and Bioengineering, Ultrasound. https://www.nibib.nih.gov/science-education/science-topics/ultrasound

[21] Website, U.S. Food and Drug Administration, Ultrasound Imaging. https://www.fda.gov/radiation-emitting-products/medical-imaging/ultrasound-imaging

[22] O. Kardan, P.Gozdyra, B. Misic, F. Moola, L. J. Palmer, T. Paus, M. G. Berman. 2015. Neighborhood greenspace and health in a large urban center. Sci Rep 5, 11610. https://doi.org/10.1038/srep11610

[23] J. Kelleher. 2019. Deep Learning. Cambridge Massachusetts. The MIT Press. The MIT Press Essential Knowledge Series.

[24] I. Goodfellow, Y. Bengio, A. Courville. 2016. Deep Learning. Cambridge Massachusetts. The MIT Press.

[25] J. Langr, V. Bok. 2019. GANs In Action. Shelter Island New York. Manning Publications Co.

[26] J. Kim, M. Kim, H. Kang, K. Lee. 2020. U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation. arXiv:1907.10830v4.

[27] X. Mao, Q. Li, H. Xie, R. Lao, Z. Wang, S. Smolly. 2017. Least Squares Generative Adversarial Networks. arXiv:1611.04076v3.

[28] R. Zhang, T. Pfister, J. Li. 2019. Harmonic Unpaired Image-to-image Translation. arXiv:1902.09727v1.

[29] M. Arjovsky, S. Chintala, L. Bottou. 2017. Wasserstein GAN. arXiv:1701.07875v3.

[30] D. P. Kingma, J. L. Ba. 2017. ADAM: A Method For Stochastic Optimization. arXiv:1412.6980v9.

[31] T. Karras, S. Laine, T. Aila. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. arXiv:1812.04948v3.

[32] A. Krizhevsky, I. Sutskever, G. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[33] A. Letchford, Open-Source Code, *Google Street View API*, [modified: July 2017], www.dradrian.com, github.com/robolyst/streetview

[34] Y. LeCun, C. Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/

[35] J. Li, M. Xu, H. Xiu. 2018. U-net Network for Building Information Extraction of Remote-Sensing Imagery. International Journal of Online Engineering (iJOE); 14(12):179-190 https://doi.org/10.3991/ijoe.v14i12.9335

[36] MIX, *Google is mapping out air pollution levels on Google Earth*, Nov. 7, 2017, TNW, https://www.thenextweb.com/

[37] D.R. Richards, P.J. Edwards. 2017. Quantifying Street Tree Regulating Ecosystem Services Using Google Street View. *Ecological Indicators*, 77, 31-40.

[38] O. RonneBerger, P. Fischer, T. Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597.

[39] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, R. Webb. 2017. Learning from Simulated and Unsupervised Images through Adversarial Training. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2242-2251, doi: 10.1109/CVPR.2017.241.

[40] K.K. Shung. 2009. High Frequency Ultrasonic Imaging. Journal of medical ultrasound, 17(1), 25–30. https://doi.org/10.1016/S0929-6441(09)60012-6

[41] Website, *CS231n Convolutional Neural Networks for Visual Recognition*https://cs231n.github.io/convolutional-networks/

[42] P. Stubbings. 2019. A Hierarchical Urban Forest Index Using Street-Level Imagery and Deep Learning. *Remote Sensing*, 11, 2019, 1395; https://doi.org/10.3390/rs11121395

[43] R.L. Thayer, B.G. Atwood. 1978. Plants, complexity, and pleasure in urban and suburban environments. J Nonverbal Behav 3, 67–76. https://doi.org/10.1007/BF011356040

[44] X. Li, C. Zhang, W. Li, R. Ricard, Q. Meng, W. Zhang. 2015. Assessing Street-Level Urban Greenery Using Google Street View and a Modified Green View Index, *Urban Forestry and Urban Greening*, 14, 2015, 675-685.

[45] D. Ulyanov, A. Vedaldi. 2017. Instance Normalization: The Missing Ingredient for Fast Stylization. arXiv:1607.08022v3.

[46] R. Wang, Z. Fang, J. Gu, Y. Guo, S. Zhou, Y. Wang, C. Chang, J. Yu. 2019. High-resolution image reconstruction for portable ultrasound imaging devices. EURASIP J. Adv. Signal Process. 2019, 56. https://doi.org/10.1186/s13634-019-0649-x

[47] H. Yang, J. Sun, A. Carass, C. Zhao, J. Lee, J.L. Prince, Z. Xu. 2020. Unsupervised MR-to-CT Synthesis Using Structure-Constrained CycleGAN. IEEE Trans Med Imaging. 2020 Dec;39(12):4249-4261. doi: 10.1109/TMI.2020.3015379. Epub 2020 Nov 30. PMID: 32780700.

[48] Z. Zhang, Q. Liu, Y. Wong. 2017. Road Extraction by Deep Residual U-Net. arXiv:1711.10684.

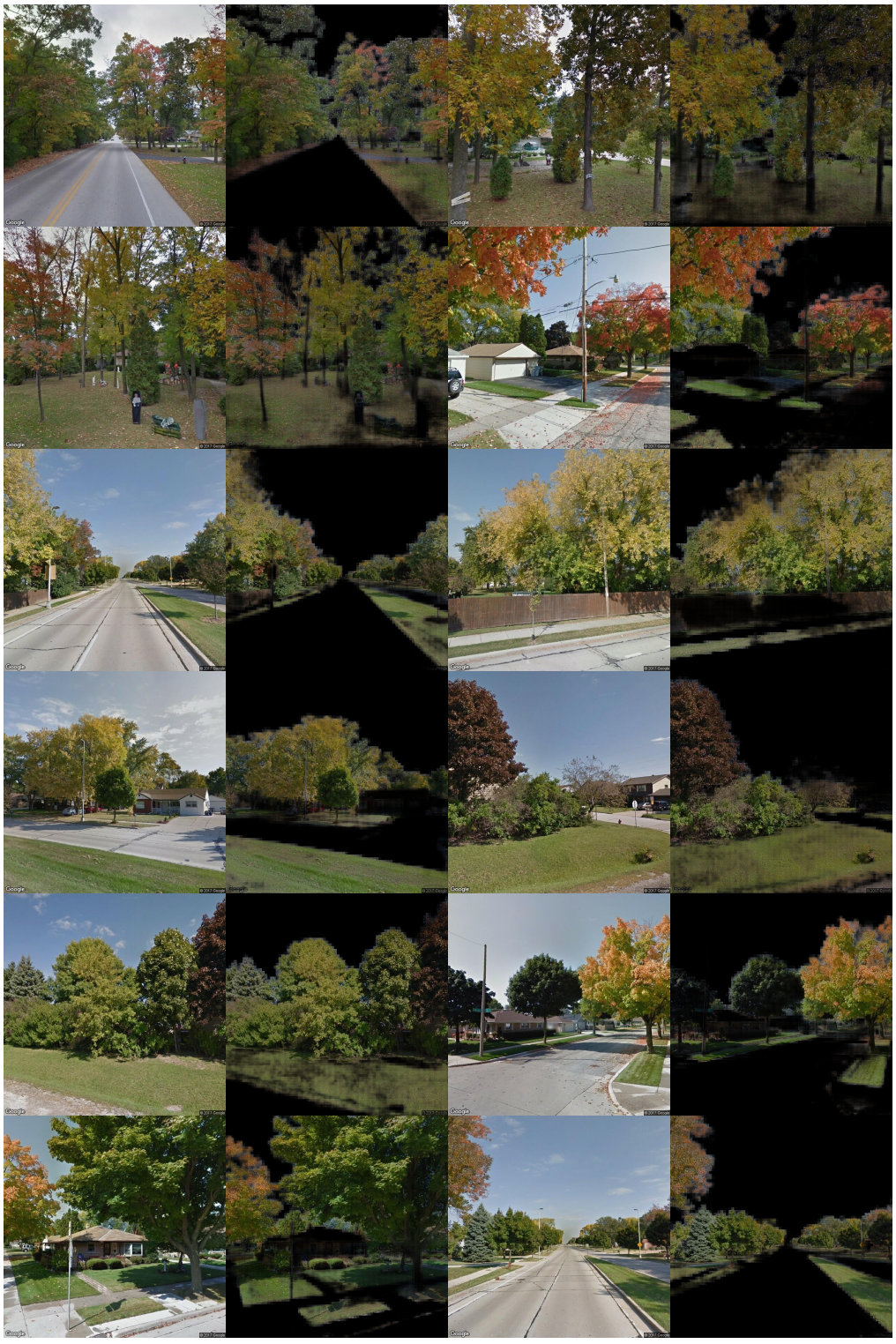
[49] J.Y. Zhu, T. Park, P. Isola, A.A. Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, in IEEE International Conference on Computer Vision (ICCV) 2017.

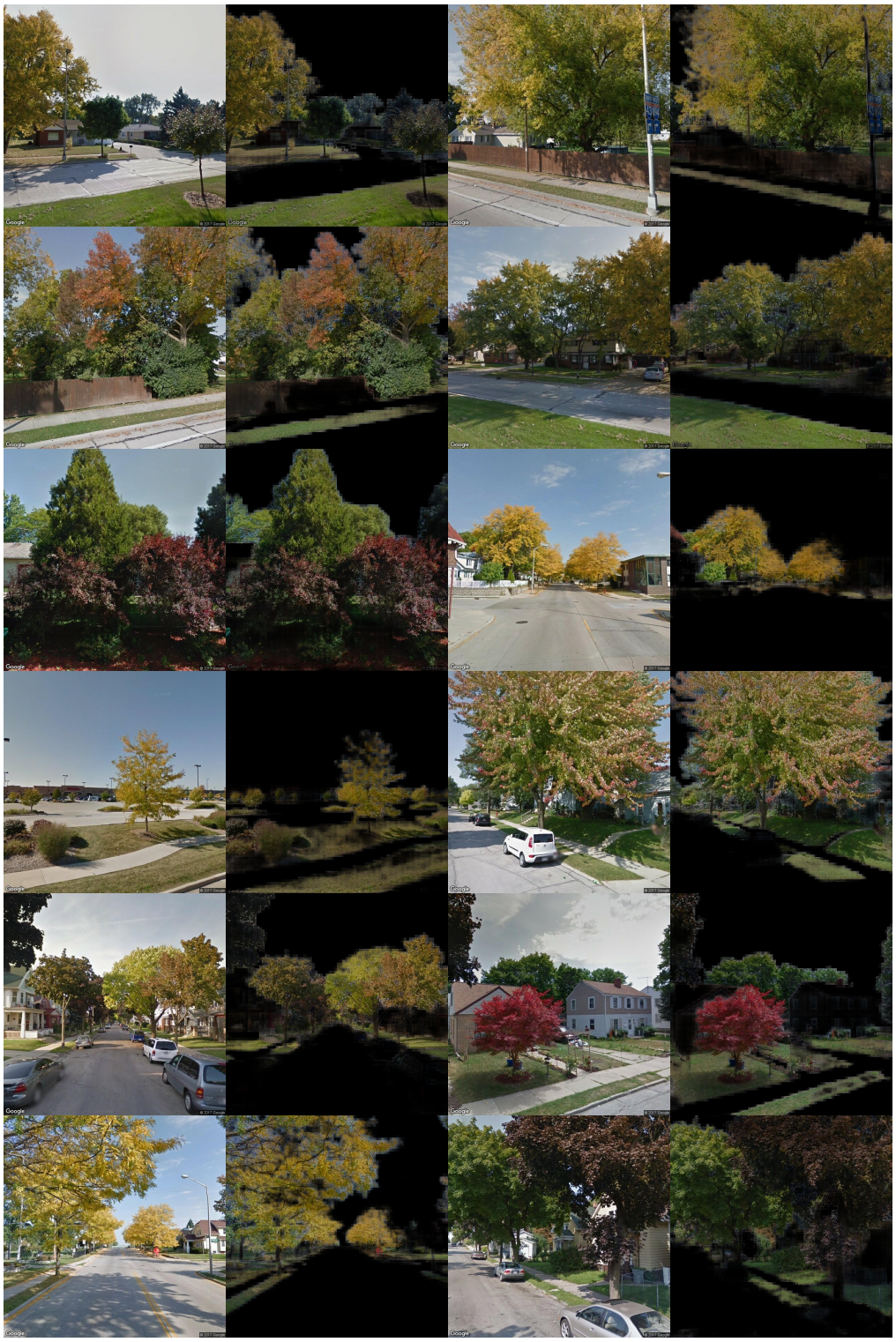
[50] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia. 2017. Pyramid scene parsing network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 21–26 July 2017, 2881–2890.

[51] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, A. Torralba. 2016 Semantic understanding of scenes through the ADE20K dataset. arXiv:1608.05442.

Appendices

A Results of Vegetation Classification by Our Proposed Network





B Results of Ultrasound Image to Image Translation by Our Proposed Network