DOMAIN ENRICHED MACHINE LEARNING
FOR SYNTHETIC APERTURE SONAR

A Dissertation in
Electrical Engineering
by
Isaac D. Gerg

© 2022 Isaac D. Gerg

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2022
The dissertation of Isaac D. Gerg was reviewed and approved by the following:

Vishal Monga  
Professor of Electrical Engineering  
Dissertation Advisor  
Chair of Committee

John F. Doherty  
Professor of Electrical Engineering

Jing Yang  
Associate Professor of Electrical Engineering

Alan R. Wagner  
Associate Professor of Aerospace Engineering

Thomas F. La Porta  
Evan Pugh Professor and William E. Leonhard Chair Professor of Computer Science and Electrical Engineering  
Head of the Electrical Engineering Department
Abstract

Synthetic aperture sonar (SAS) is a coherent imaging modality that generates high-resolution and constant-resolution imagery of the seafloor. Despite its ability to yield high-resolution images, collecting and curating the imagery is expensive, making little SAS imagery available for training machine learning (ML) algorithms. Furthermore, misestimation of array positions, misestimation of environmental parameters, or reverberant environments (i.e., multi-path) can result in the collected imagery exhibiting degraded image quality. Contemporary ML algorithms such as deep learning perform well, but only when training data is available in significant quantities and high-quality with no missing information. This work proposes the incorporation of domain knowledge into deep learning algorithms for SAS to overcome the lack of abundant, high-quality training data. We accomplish this task by applying domain knowledge to the network architecture or loss function of the training optimization problem. This work demonstrates that such knowledge leads to improvements in ML performance without the need to collect or label more data and, thus, make better use of existing training datasets. Specifically, we demonstrate this principle in three facets of automated SAS imagery analysis: classification, semantic segmentation, and autofocus. For classification, two parcels of domain knowledge are introduced, which take the form of custom network structure and custom loss functions resulting in improved classification performance which outperforms state-of-the-art methods on a real-world dataset. For semantic segmentation, domain knowledge of multi-look sequence processing (MLSP) is used to derive acoustic angle-of-arrival features from the complex SAS image resulting in state-of-the-art performance on another real-world SAS dataset with very limited labeled data. And finally, for autofocus, domain knowledge of $k$-space structure is used to coherently improve SAS image quality from a small database of images which we demonstrate in both self-supervised and supervised formulations. Results show that the proposed method exhibits computation and image quality gains over existing iterative-based autofocus methods. In total, several ways of integrating domain knowledge into deep networks are introduced covering various parts of the SAS image processing chain; the results demonstrate good performance is achievable for these tasks even when abundantly labeled training data is unavailable.
# Table of Contents

List of Figures vii  
List of Tables xviii  
Acknowledgments xx  

## Chapter 1  
### Introduction  
1.1 Brief Overview of Synthetic Aperture Sonar and an Account of It’s Development 1  
1.2 Open Challenges in Machine Learning for SAS 6  
1.3 Improving Deep Learning Robustness with Domain Enriched Deep Learning 8  
1.4 Overview of Contributions 11  
1.4.1 Contribution I: Prior Guided Object Recognition for Synthetic Aperture Sonar 11  
1.4.2 Contribution II: Deep Multi-Look Sequence Processing for Synthetic Aperture Sonar Image Segmentation 14  
1.4.3 Contribution III: Prior Guided Image Enhancement for Synthetic Aperture Sonar 17  
1.5 Organization of Dissertation 19  

## Chapter 2  
### Contribution I: Structural Prior Driven Regularized Deep Learning for Sonar Image Classification  
2.1 Introduction 20  
2.1.1 Open Challenges in SAS ATR 20  
2.1.2 Overview of the Proposal 22  
2.2 Previous Work 25  
2.3 Proposed Classification Method: SPDRDL 27  
2.3.1 Motivation of Approach 27  
2.3.2 Feature Extraction Network 29  
2.3.3 Structural Similarity Prior Via Data-Adaptive Image Enhancement Network 29  
2.3.4 Structural Scene Context Prior Via Target Localization Network 30
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.2</td>
<td>Overview of Our Proposal</td>
<td>82</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Contributions</td>
<td>84</td>
</tr>
<tr>
<td>4.2</td>
<td>Previous Work</td>
<td>85</td>
</tr>
<tr>
<td>4.3</td>
<td>Proposed Autofocus Method: Deep Autofocus</td>
<td>86</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Background</td>
<td>86</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Motivation of Approach</td>
<td>88</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Design of Deep Autofocus Network</td>
<td>90</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Network Loss Function</td>
<td>92</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Training Procedure and Data Augmentation Scheme</td>
<td>94</td>
</tr>
<tr>
<td>4.4</td>
<td>Experiments</td>
<td>96</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Dataset Description</td>
<td>96</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Evaluation Against Comparison Methods</td>
<td>97</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Autofocus Results for Each Method</td>
<td>99</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Phase Correction Error Compared to Ground Truth</td>
<td>100</td>
</tr>
<tr>
<td>4.5</td>
<td>Learning Rate Cross-Validation Results for Metric-Based Comparison Methods</td>
<td>104</td>
</tr>
<tr>
<td>4.6</td>
<td>Example Autofocus Results on Test Set</td>
<td>105</td>
</tr>
<tr>
<td>4.7</td>
<td>Sample image corrupted with increasing levels of QPE</td>
<td>106</td>
</tr>
</tbody>
</table>

Chapter 5
Conclusions and Future Directions 108
5.1 SAS Object Recognition 108
5.2 SAS Semantic Image Segmentation 109
5.3 SAS Autofocus 110

Bibliography 113
List of Figures

1.1 Comparison of a RAS, (a), and SAS, (b), images over the same seafloor area. The sonar collected the images from the starboard side with the along-track direction traversing vertically; therefore, the nadir is the left vertical edge of each image. We see the SAS image has a constant resolution in both range and along track, whereas the RAS image only has a constant resolution in range. Furthermore, uniform spatial coverage is present in the SAS image since the reconstruction algorithm compensates for vehicle position to form the synthetic aperture; this is not true for the RAS image. (d)-(g) compare corresponding seafloor areas between the RAS and SAS images of (a) and (b). We see the SAS image captures detail of seafloor textures and point scatterers much better than the RAS. 2

1.2 Typical sensing geometry of a SAS system mounted on an unmanned underwater vehicle (UUV) performing a survey in the stripmap imaging configuration. 3

1.3 Depiction of the forward model described by Eq. 1.1. A time domain back-projection algorithm for image reconstruction inverts this forward model to solve each pixel for \( \sigma(\mathbf{x}) \). Since each pixel in the integration can be processed independently, this formulation lends itself to being straightforward to parallelize on a graphics processing unit (GPU). 4

1.4 Number of search hits on Google Scholar by year of the term “synthetic aperture sonar” (quotes included in query). 4

1.5 Wavelet coefficient distribution [38–41] for horizontal (a) and vertical (b) bands of natural images (COCO) versus a SAS image database [36]. The wavelet coefficients of the SAS imagery are characterized by: (1) an asymmetry between horizontal and vertical bands resulting in lighter and heavier tails respectively when compared to the natural imagery, and (2) a broader peak compared to the natural imagery which diminishes when despeckled. “SAS + Despeckle’ created using 3 \( \times \) 3 median filter. The results are similar for different wavelet selections (‘db2’ used here). 7
1.6 Geometric diagram motivating the use of domain enrichment in deep learning. Here we show training, test, and out-of-distribution (OOD) samples (e.g., samples from a different but similar sensor). We also show three possible training boundaries that separate the training data. The red boundary minimizes the training loss but performs badly on the test data (i.e., overfitting to training set). The blue boundary does a reasonably good job separating the test data but not the OOD data (i.e., overfit to the test set). However, we can see that only the green curve captures the structure associated with all the data (i.e., robust). The green curve represents a solution based on domain knowledge of the problem at hand. It thereby captures the greater structure of the problem not explicitly represented as X,Y training data pairs. The difference between the green and blue boundaries can fool a practitioner into thinking their ML model is robust when it is not.

1.7 Example SAS processing chain showing places where this dissertation makes research contributions. Our proposed algorithm for Seafloor Segmentation traces a dotted line to the SLC as an input for our proposed algorithm, a novelty separating it from existing SAS segmentation algorithms that rely on the DRC image.

1.8 Selected list of popular SAS classification papers over the last twenty-five years. The purpose of this timeline is to highlight popular research areas and show the evolution of the field to its state today.

1.9 Selected references from the SAS seafloor segmentation literature. The vast majority of these methods are unsupervised partly due to the difficulty of obtaining ground-truth labeled SAS imagery.

1.10 Timeline of research themes used in SAS autofocus.

2.1 SAS is capable of producing high-quality, high-resolution imagery of seafloor and objects. The images here are examples collected from the Centre for Maritime Research and Experimentation MUSCLE system. In this work, we provide an ATR algorithm to classify MUSCLE images into four classes. Example objects from each class are shown: (a) background, (b) cylinder, (c) truncated cone, and (d) wedge. Difficulties in classification result because there are often objects which appear target-like (e,f), but are not targets (false alarms); and some targets are difficult to discern (g,h) because of orientation, burial depth, or background topography causing them to be ignored (missed detections).
2.2 The relationship of priors to the problem domain and the training data. Our priors encapsulate domain knowledge which is not explicitly represented in the data but projected into the data. The priors used in this work, structural similarity and structural scene context, are employed through regularization losses which are augmented to the primary task’s objective function which is classification error. We jointly train all losses so the network finds a minimum consistent with both the data and the domain priors.

2.3 The SPDRDL network architecture; the network input is a SAS image and the output is a classification and target center position when a target is present. SPDRDL is composed of four modules: image enhancement network, feature extraction network, target localization network, and a classification network. SPDRDL leverages two domain priors to improve classification: (1) image enhancement algorithms like despeckling improve image interpretability and (2), the detector produces SAS images with well-centers targets. For the former, an enhancement network leverages the human visual system priors incorporated into the structural similarity prior to enhance the image for classification. For the latter, input images are translated as part of the data augmentation procedure during training and this translation is estimated in addition to predicting image class. Image classification, enhancement, and target localization are simultaneously trained.

2.4 The detector returns well-centered target images. We can use this prior during our data augmentation procedure. During data augmentation, we translate the image via random crops and record the effective translation shift induced as $p_{\text{shift}}$. To incorporate this prior into the network, we add a target localization task which encourages the network to estimate this translational shift in addition to outputting classification. As an example of this procedure, the detector returns a well-centered target image (left) and random crops of this image are fed to the network as data augmentation (middle, right). The target localization network estimates the difference between the true center of the target (green) and the shifted center (red). The translation shift estimate is denoted as $\hat{p}_{\text{shift}}$.

2.5 The dataset we use for our experiment is split into two groups based on the collection year. Dataset A is used for the training set and Dataset B is used for the validation and tests sets.
2.6 The validation and test sets are derived from eight-way neighbor translations of 59cm. The original tile is from Dataset B and cropped. All nine cropings are used for the test set with a random crop used for the validation set shown by the boxed image. We show the test and validation data generation scheme for example images of the (a) background class and (b) target class.

2.7 Confusion matrices for SPDRDL and comparison methods. Larger numbers along the diagonal indicate better performance. We see the benefits of the added domain priors through improved performance, especially of the wedge class.

2.8 Precision-recall curves for target type: (a) cylinder, (b) wedge, (c) truncated cone. AUCPR in parentheses; larger values are better.

2.9 PR curves for all methods obtained using a one-versus-all method. AUCPR in parentheses; larger values are better.

2.10 Deep learning methods work well when the training data is plentiful. Sometimes such a situation is not possible as collection of training data can be expensive. To demonstrate the necessity of the domain priors our method introduces, we train each method using all the training data available and then train using a random 10% subset. We can see from the low training data scenario that SPDRDL still produces quite competitive results indicating the increased performance granted by the domain priors.

2.11 To understand the sensitivity of selection bias in a low training data scenario, we trained the top three methods each ten times with a random sample of 10% of the training data and plot the results. Due to the large run-time associated with evaluating all the test set imagery, we report results on the validation set only. Larger AUCPR indicates better performance. As shown, we can clearly see a benefit in performance with the addition of the domain priors over the next best method, Densetnet121.

2.12 (a) and (b) Target scores (SPDRDL/Densenet121 predicted classification probability) as a function of diagonal image translation of the top two performing algorithms for a sample input image. We can see for the Densenet121 method that a small translation of the input image results in an unpredictable classification whereas SPDRDL does not exhibit this. (c) Image under test for (a) and (b).
2.13 Top panel, AUCPR as a function of range from the sonar for all methods. Larger values indicated better performance. Bottom panel is a zoom of the top panel highlighting the differences of the top three methods: SPDRDL, DenseNet121, and Emigh, et al.

2.14 We analyze the frequency spectrum of the original images input to the network (a) and of our learned image enhancement (b). In panel (b), we can see selective attenuation across spatial frequencies and orientations due to the use of the data-adaptive image enhancement prior.

2.15 AUCPR as a function of weight pruning proportion. Weights are sorted by magnitude and removed starting with the lowest magnitude weights first. Significant pruning is accomplished while still maintaining good classification performance.

3.1 Example SAS image (left) with its weakly-labeled segmentation truth map (right). In the right image, the blue and brown areas indicate human-provided labels of two seafloor classes while the black area represents unlabeled pixels. The border of the labeled regions is shown in the left image in white. We see the human only labeled some of the "easy" portions of the image containing mostly unambiguous seafloor texture leaving the remaining pixels unlabeled.

3.2 Seafloor classes considered in this work using data from [125]. For each image pair (i.e. column), the image on the top is the original SAS image and the image on the bottom highlights one of the corresponding seafloor classes present in the top image deemed "easy" to label by a human. In viewing (a) and (b) we see classes like dark sand and shadow look similar in magnitude imagery and thus are easily confused. However, we resolve this ambiguity by examining angle-of-arrival information through multi-look processing employed by our proposed network architecture, MLSP-Net.
3.3 Example MLSP sequence showing benefits of $k$-space filtering a SAS image. (Top) Image squint sequence derived from SLC image (right-most column). (Bottom) Corresponding $k$-space filter with vertical axis approximately proportional to look-angle, $\theta$, and horizontal axis approximately proportional to frequency. In the top right quadrant of the images of the top row, there is an arrow above each image pointing to a “fuzz” moving vertically in the image sequence. This is likely biologics in the water column moving during the collection of the synthetic aperture. The $k$-space filtering produces a pseudo-motion effect across frames of the image sequence capturing this movement. This phenomenon is not apparent when $k$-space filtering is disabled (see right-most column, “No Filtering” for this image) resulting in seafloor occlusion or false seafloor texture.

3.4 Our proposed network called multi-look sequence processing network (MLSP-Net) for SAS segmentation is composed of two primary paths: Multi-Look Sequence and Static Image. The Multi-Look Sequence path is composed of eight Filter-Modules operating in $k$-space and are initialized using domain knowledge so an initial useful multi-look sequence emerges (see Figure 3.3). The eight looks are sent through a shared U-Net encoder, re-weighted by squeeze and excitation modules, and then fed to a bidirectional 2D convolutional LSTM to extract meaningful features from the image sequence. The Static Image path simply ingests the input image and processes it through a U-Net. Features from both paths are then concatenated in the feature dimension and processed by a segmentation network for Final Image Segmentation. We show the utility of the MLSP by doing an ablation study whereby we initialize and fix the Filter-Modules to unity thereby removing their effect. Now, all eight paths have the same non-filtered input image resulting in the same feature map for each step of the LSTM. Results in Table 3.5 show the benefit of the MLSP than without it. Circular taps in the figure show representative results; diamond taps show data type (real $\mathbb{R}$ or complex $\mathbb{C}$) and tensor shape.

3.5 Model used for the parametric filter estimate. The yellow boxes represent the receiver array. $\theta_{BW}$ is squinted beam angle bandwidth and $\theta_0$ is squinted beam angle offset. In addition to filter attenuation, these parameters are learned by MLSP-Net as part of the training procedure.
3.6 A depiction of the differences between true AoA with that of our approximation made in $k$-space for the SAS system of [139] (see 3.1) at 0.015m resolution. The picture overlays colored iso-lines of true constant AoA (horizontal lines) and frequency (vertical lines) onto $k$-space. The grey-colored grid attached to the axes $k_x$ and $k_y$ show the AoA and frequency approximation used in MLSP-Net (Equation 3.6). Recall, we use this approximation because we are not given the sonar center frequency, $f_0$, for the data; not an uncommon scenario. We see no more than a degree of AoA error across the frequency band using our approximation.

3.7 Maximum angle-of-arrival error as a function of fractional bandwidth. As the fractional bandwidth decreases, the angle-of-arrival approximation error decreases. Several high-frequency (HF) SAS systems (as used here) and their fractional bandwidths are in Table 3.1.

3.8 (Top) Convergence to the rectangle function by the limit of a rational function in Equation 3.8. The figure shows a growing sequence of $n$ ultimately converging to the rectangle function as $n \to \infty$. (Bottom) Magnitude Fourier transform (i.e. magnitude impulse response) of top showing convergence to a sinc function as $n \to \infty$. The parameter $n$ in Equation 3.8 allows us to trade off filter fidelity with ringing artifacts of the impulse response (i.e. Gibbs phenomenon) as well as dampening the coefficients away from the origin to effectively reduce the support of the filter. We use $n = 10$ for each Filter-Module in MLSP-Net.

3.9 A sample output of softfilter$_{10}(\theta, \theta_{BW} = \frac{1}{3}, \theta_0 = -0.125, \alpha = 0.5)$ using $n = 10$ in Equation 3.9. The softfilter$_{10}$ function is used in each Filter-Module of Figure 3.4 to filter the input SLC in the $k$-space domain to filter by specific look-angles of acoustic energy. See Figure 3.3 for an example of these filters applied to a SAS image and Figure 3.6 to see $k$-space coordinates of an example high-frequency SAS system which the softfilter$_{10}$ function is applied across all frequencies to filter by look-angle.
3.10 Sample results from the top three methods of Table 3.5: Rahnemoonfar, et al. [148]; our proposed method with ablation MLSP-Net-No-Filter (which is effectively a U-Net ensemble, see Section 3.4.4); and our full proposal MLSP-Net. For rows top to bottom: input image; ground truth labels (recall these are weak/partial labels); Rahnemoonfar, et al. [148] results; classification errors between ground truth and Rahnemoonfar, et al. [148]; MLSP-Net-No-Filter results; classification errors between ground truth and MLSP-Net-No-Filter, MLSP-Net results; and classification errors between ground truth and MLSP-Net. As shown, MLSP-Net makes fewer classification errors overall compared to the other methods. Furthermore, MLSP-Net classifies the shadow area on the left side of the image (iii) better than our proposed method with ablation (MLSP-Net-No-Filter) thus demonstrating the improvement given by the AoA filtering. See Table 3.2 for class-color mappings.

3.11 Distribution of each class (including unlabeled pixels) for the training and test sets. The dataset was weakly labeled hence a large number of unlabeled pixels. Moreover, the dataset is highly class imbalanced.

3.12 Confusion matrices for the two best methods of Table 3.5, our proposed MLSP-Net method and baseline MLSP-Net-No-Filter. Class abbreviations defined in Table 3.2. These results are composed of ten training runs. We see MLSP-Net yields better MPA in four of the seven classes and does especially better at differentiating shadow and dark sand classes. Note, the MLSP-Net-No-Filter configuration is essentially two U-Nets put together in a small ensemble so we forgo explicitly evaluating the U-Net as a comparison since it would be redundant.

3.13 Box and whisker plot of MPA over ten runs of the two best methods of Table 3.5. Orange lines of the boxplot indicate median values (0.921 versus 0.902) and green triangles indicate average (0.920 versus 0.906). We see the MLSP-Net yields better performance and less sensitivity to training sample selection for training than MLSP-Net-No-Filter which does no k-space filtering. Thus, this test demonstrates not only the utility of this domain knowledge in improving network performance but also its robustness to training sample selection which is especially important when abundant data is not available as is the case here.

3.14 The eight Filter-Module parameters as a function of training epoch for one run of MLSP-Net. We see the most significant changes occur in filter bandwidth and position where the filter near boresight is allocated the largest bandwidth.
4.1 A well-focused SAS image, (a), corrupted by three types of systemic phase corruptions, (b). The resulting imaging artifacts sometimes fool operators into thinking the imagery is well-focused while simultaneously modifying the geometric structure of interesting objects thereby causing an operator to incorrectly overlook them. (a) is defocused using three types of systemic phase error: (c) quadratic phase error (QPE), (d) sinusoidal phase error, and (e) sawtooth/yaw phase error corresponding to panel (b). QPE manifests as a blur in the along-track direction while sinusoidal and sawtooth/yaw error manifests as image copies in the along-track direction [67]. This article proposes an autofocus method for images with phase errors like in panels (c),(d), and (e). Images (f), (g), and (h) are autofocused versions of (c), (d), and (e), respectively, using our proposed method.

4.2 A block diagram describing the application of a phase correction to a SAS image. The input is the defocused SAS image, $g$, along with the proposed phase correction, $\hat{\phi}$, and the output is the autofocused image, $\hat{g}$. $\mathcal{F}$ is the Fourier transform.

4.3 Network diagram of our proposal, Deep Autofocus. Deep Autofocus takes as input a defocused single-look-complex (SLC) image and outputs a well-focused dynamic range compressed (DRC) image. During inference, only the path in red is executed thus shifting the bulk computational burden of the algorithm to the training phase. Vector sizes and types (i.e. real $\mathbb{R}$ or complex $\mathbb{C}$) at relevant intermediate stages are listed.

4.4 A sample of ten phase error draws, $\psi = \Psi$, used for test set creation. The phase error draws are composed of polynomial functions, sawtooth functions, and combinations of thereof.

4.5 RMSE phase error between ground truth and estimate phase error by network. (top) Distribution of overall phase error. (bottom) Distribution of phase errors as a function of aperture position. Lower numbers are better for all.

4.6 Examples of point-spread-function improvement using Deep Autofocus (supervised scheme). These two images are not from the test set and from a real-world SAS collection just after image reconstruction but with residual phase error. From left to right for each row: (a) Original image with zoom inset of point-of-interest, (b) autofocused image, (c) along-track slice of point, (d) estimate phase error by Deep Autofocus.
4.7 Image quality of the test set compared to ground truth. For all plots, higher numbers indicate better performance. Box extends from first quartile to third quartile, median is given as orange line, green triangle is mean. Shown are evaluations against three IQA metrics: (a) PSNR, (b) SSIM, and (c) MS-SSIM. As shown, DeepAF does significantly better than the comparison methods. For reference, we provide results for the original unfocused imagery as “No Autofocus.”

4.8 Image quality improvement for each image of the test set compared to ground truth. For all plots, higher numbers indicate better performance. Box extends from first quartile to third quartile, median is given as orange line, green triangle is mean. Shown are evaluations against three IQA metrics: (a) PSNR, (b) SSIM, and (c) MS-SSIM. As shown, DeepAF does significantly better than the comparison methods.

4.9 Results of the grid search used to determine the best learning rate for each iterative autofocus metric and associated image quality metric. The goal is to find a learning rate which yields the best possible result for each configuration.

4.10 From left to right: original defocused image, Deep Autofocus (supervised) result, ground truth well focused image, difference between Deep Autofocus and ground truth images, estimated (gray) and ground truth (black) phase error.

4.11 Sample image corrupted with varying levels of QPE to visually illustrate how QPE strength relates to image degradation. Top row are the images and bottom row is the amount of QPE.

4.12 Image quality metrics as a function of quadratic phase error (QPE) strength for test set. The image quality metrics evaluated are (a) SSIM, (b) MS-SSIM, and (c) PSNR. Image are not despeckled prior to computing the metric. These plots, along with Figure 4.11, are provided for convenience to aid in interpreting the results of Figure 4.7 and Figure 4.8. Box extends from first quartile to third quartile and median is given as orange line.
5.1 Four example images (DRC images in top row) from the dataset used in Chapter 2 along with their $k$-space power spectral densities (bottom row). Images (a) and (b) exhibit a horizontal streaking effect in the DRC image whose cause is unknown and unable to be speculated. Images (a), (b), and (d) have a vertical spectral null on the right side of the spectrum where image (c) has this null on the left side of the spectrum. We posit that this artifact may be due to the signal processing or transmit waveform used to construct the image but we do not have enough information to discern this with certainty (nor to correct it).
List of Tables

2.1 Brief summary of the loss terms in our objective function admitted from our domain priors. We list the relationships among the incorporated domain knowledge, the employed prior, and the associated regularization loss in our formulation. We also list the primary task, classification, and its loss function for completeness. .................................................... 25

2.2 Description of SPDRDL architecture network architecture. The network input is a $256 \times 256$ pixel grayscale SAS image normalized to $[0, 1]$. The network has two outputs, a classification output and a target position output. AA indicates anti-aliasing was applied to the layer. ....................... 50

2.3 We evaluate performance of each domain prior in our loss function, Eq (2.7), to demonstrate their utility. AUCPR is reported on the test set for the high (100% of training data available) and low (10% of training data available) training data scenarios. Note, the enhancement network is still present in the CL and CL+SSCP scenarios but the associated regularization loss is removed from the objective function. ...................... 51

2.4 We evaluate the performance of our SSP domain prior against several off-the-shelf despeckling methods to show SSP’s utility. We do this by retraining the network but removing SSP’s associated loss term in Eq (2.7) and the Enhancement Network in Figure 2.3 (SSCP is still included). We feed to network three types of despeckled imagery and report AUCPR using 100% of the available training data. We see that despeckling does give some performance gains over the CL configuration of Table 2.3, but not as much as when the SSP is active (CL+SSP and CL+SSP+SSCP configurations of Table 2.3). Recall, our method (CL+SSCP+SSP) yields an AUCPR of 0.9538 as shown in Table 2.3. ......................... 51

2.5 Number of trainable parameters for the deep learning methods. ....... 51
2.6 Classifier scores were compiled for all nine shifts of each image in the test set. Next, we compute the standard deviation of scores for each image. Finally, the table shows the mean of those scores over the entire test set; lower numbers indicate more shift invariance (i.e., better performance). We see that SPDRDL has the best translation invariance of all the methods. We hypothesize that Densetnet121 has the second best performance because it contains only a single instance of MaxPooling and has many levels of feature averaging.

3.1 Fractional bandwidths for several SAS systems mentioned in the literature. We see for high-frequency SAS systems the maximum AoA error is less than five degrees.

3.2 Classes making up our dataset along with the corresponding color in plots.

3.3 Description of the Segmentation Net block of Figure 3.4.

3.4 Description of the U-Net model based around a pre-trained Resnet50 encoder [97]. For conciseness, we list the decoder side of the network. The U-Net multiplies the input by 128 to be consistent with the scaling used during the pre-training. Upsampling uses the nearest neighbor method.

3.5 Pixel accuracy by class and mean pixel accuracy (MPA) for each method. Larger numbers indicate better performance. Classes are listed in Table 3.2. Best for each class in bold. We see the MLSP-Net has the best MPA of all the methods and yields best results for four of the seven classes. Moreover, we show improved performance over the next best method for the shadow class which is easily confused with dark sand (MPA of 0.632 for MLSP-Net-No-Filter versus 0.714 for MLSP-Net) demonstrating the utility of the Filter-Modules and 2D modeling of the MLSP features.

4.1 Network architecture of the Regression Network shown in Figure 4.3. GELU is Gaussian error linear unit [186]. The regression network takes input from the Feature Extraction Network, a pretrained Densenet121 backbone. The network computes the phase error using a semi-parametric model composed of two parametric models, polynomial and sinusoid, and a non-parametric model, vector with shape 256.

4.2 List of image sharpness metrics used in this work for comparison to Deep Autofocus. The input single look complex (SLC) is \( g \in \mathbb{C}^{M \times M} \) and \( \text{stddev}(x) \) is the standard deviation over the elements of \( x \). We make \( b = \epsilon \) in OSF as it was shown when \( b \) is large the metric is equivalent to SSI.
Acknowledgments

In some ways, this section is the essential part of this work, demonstrating scientific progress is possible only through interaction with other humans and in many ways has little or nothing to do with actual science itself but with sharing the “ups and downs” of life. This is probably the most read section too.

Above all, I express my utmost gratitude to my wife, as I could not have and would not have accomplished this endeavor without her. Only through her patience, diligence, time, compassion, and selflessness could I have set forth to conduct any of this work. This dissertation is for you, Ryanne.

Somewhere along my life journey, I have found that I do my best when I surround myself with people who are more intelligent than I. The fact that I decided to do a Ph.D. in my mid-thirty’s which I am finishing at forty while my lab-mates in iPal will likely start and complete theirs in their 20s shows how much brighter they are than I am. Thank you for all your help and support over the last four years and for treating me like a peer and not like an “old man.” Thank you, Xuelu, Tiantong, Khaled, Venkat, Yung-Chen, Ethan, Amir, Kareem, Mallory, Trung, Haichuan, and Junho. Thank you to Hein Hundal, who helped me prep for my qualifying exam, and Carl Cotner for the countless conversations oriented around mathematics and life. Thank you to my co-authors, Yung-Chen Sun and David Williams.

Here is a list of people who convinced me to go back in my mid thirties for a Ph.D. Shame on you, Dan Brown, Steve Wagner, Brian Reinhardt, Joonho Park, Tony Lyons, Shawn Johnson, Derek Olson, Tom Montgomery, John McKay, and Dan Cook. (Seriously though, thank you for your encouragement and wisdom). Finally, here is the list of folks who helped me through gallbladder disease when I was visiting the emergency room weekly during the fall of 2014 as, without them, I surely would not be here: Dr. Stephen Meltzer, Dr. Sameer Dhalla, and Dr. Ellen Stein.

I was fortunate to have good instruction, stability, and love while growing up. Thank you to my mother, Mary, father Douglas, and brother Ian (who incidentally started his Ph.D. before the age of thirty and, therefore, also more intelligent than I am). You guys inspired and supported me through thick-and-thin; I can’t say thank you enough.

A special thank you to my advisor, Vishal Monga, who provided much guidance and wisdom during this journey. He did an excellent job mentoring and encouraging me in this atypical graduate school situation as I worked full time to complete this degree and started much later in life than most. I sincerely appreciate the patience and time he has given to me over the last four years (and hopefully more to come). Also, thank you to
my committee members, John Doherty, Jing Yang, and Alan Wagner, for your support, time, and encouragement during this journey.

Countless other humans have inspired, guided, and corrected me allowing for arrival at this destination; some names are listed here in no particular order: Victor W. Brunsden (Penn State Math), Gary J. Weisel (Penn State Physics), William Locke (Bettis Atomic Power Lab), Ken Courtney (Bettis Atomic Power Lab), Richard L. Tutwiler (Penn State ARL), Timothy J. Kane (Penn State EE), and Ashley Holleman (PSU Cru).

Thank you to the NATO Centre for Maritime Research & Experimentation (CMRE) for providing the data used in Chapter 2; the NATO Allied Command Transformation funded the collection of this data. Thank you to the Naval Surface Warfare Center - Panama City Division (NSWC-PCD) for providing the data used in Chapters 3 and 4.

The tools used to make this work included: FFTW, HDF5 (h5py), Inkscape, JAX [1], joblib, LaTeX, Matplotlib [2], Notepad++, numba [3], numpy [4]/scipy [5], OpenCV [6], Overleaf, pillow (PIL fork), Python, scikit-image [7], scikit-learn [8], Stack Exchange, Tensorflow [9], TQDM, and Wing Python IDE. This work contains several plots whose colormaps derive from [10] and [11] (I think it’s unfortunate that colormaps are often an afterthought).

The US Navy Office of Naval Research supported this work under grants N00014-19-1-2563, N00014-19-1-2638, and N00014-20-1-2406. However, the findings and conclusions in this work do not necessarily reflect the views of the US Navy. Additionally, the Penn State Applied Research Laboratory partially funded the writing of this work. Finally, this work was made possible by morale contributions from Freeze Thaw Cycles, Wood Cliff Cabin, Walter’s Bakery, and Bun Bun.
Chapter 1
Introduction

1.1 Brief Overview of Synthetic Aperture Sonar and an Account of It’s Development

Synthetic aperture sonar (SAS) is an imaging technique for generating constant- and high-resolution sonar imagery of the seafloor. The image SAS produces is an improvement over side-scan-sonar (SSS), sometimes called real-aperture sonar (RAS), where resolution in the along-track dimension worsens with range. SAS is able to achieve these benefits by coherently combining consecutive sonar pings to synthetically (and dynamically) lengthen the aperture making resolution constant everywhere throughout the resulting image [12]. Such properties allow us to develop almost photograph-like scenes of the seafloor. Figure 1.1 shows an image collected by a SAS sonar processed as RAS and SAS to show the image quality gains SAS realizes in image resolution and uniform spatial sampling.

SAS works by transmitting sound in water in the direction of the seafloor using rapid, periodic pulses (often called ping)(ing) and then coherently combining the acoustic returns from the seafloor over several pings to produce an image. This process differs from RAS, where an image is created by building up raster scanlines from individual transmit-receive pulse pairs, a process where along-track resolution worsens with range.

The along-track resolution of a sonar array improves proportionally to the length of the array/aperture. In SAS, the aperture is synthetically lengthened (hence its namesake) by coherently integrating over several pings to obtain constant resolution throughout the image. The SAS community often refers to this property as constant resolution with range. In RAS, the resolution in the along-track direction degrades as a function of range due to the spherical spreading of the transmitted waveform. In both SAS and RAS, the range resolution is constant and chiefly determined by the bandwidth of the transmitted pulse. Figure 1.2 shows a common geometry used to collect SAS imagery from an unmanned
underwater vehicle (UUV) known as *stripmap* imaging. This dissertation assumes this geometry throughout.

![Comparison of a RAS, (a), and SAS, (b), images over the same seafloor area. The sonar collected the images from the starboard side with the along-track direction traversing vertically; therefore, the nadir is the left vertical edge of each image. We see the SAS image has a constant resolution in both range and along track, whereas the RAS image only has a constant resolution in range. Furthermore, uniform spatial coverage is present in the SAS image since the reconstruction algorithm compensates for vehicle position to form the synthetic aperture; this is not true for the RAS image. (d)-(g) compare corresponding seafloor areas between the RAS and SAS images of (a) and (b). We see the SAS image captures detail of seafloor textures and point scatterers much better than the RAS.](image)
Forming a SAS image is often called image reconstruction or beamforming. Greatly simplified, image reconstruction consists of two steps. The first step is estimating the sonar array positions as a function of time, and the second step is coherently combining pings to reconstruct the image. Researchers have proposed many methods for the second step, but practitioners regularly use one of two classes of algorithm: frequency-domain or time-domain. We will briefly discuss the time-domain approach since its arguably the more intuitive to understand of the two.

The time-domain image reconstruction algorithm, commonly referred to as time-domain backprojection, works by inverting the forward scattering model of the acoustic wave relative to the seafloor. The forward model is described by,

\[
e(t, x_{RX}(t)) \approx \int \frac{\sigma(x)}{\|x_{TX} - x\| \|x_{RX}(t) - x\|} \cdot q \left( t - \frac{\|x_{TX} - x\| + \|x_{RX}(t) - x\|}{c} \right) dx
\]

where \( e(t, x_{RX}(t)) \) represents the time-series of the array, \( t \) is time, \( x_{TX} \) is the transmitter location (we assume it’s stationary during transmit), \( x_{RX}(t) \) is the receiver location, \( x \) is the scattering location, \( \sigma \) is the acoustic scattering cross-section function [13], \( c \) is the speed of sound, and \( q \) is the transmitted signal waveform. Figure 1.3 gives a depiction of this process for one pixel of the integration. An important detail is that if \( e \) is basebanded (which it almost always is), the resulting \( \sigma \) will be complex. This fact means the SAS image comprises complex-valued pixels, each with a real and imaginary component. This raster is called the single-look complex (SLC) image. Because of the complex-valued pixels of the SLC, a computer cannot readily display the image in a manner for human consumption. We resolve this problem by applying dynamic range
Figure 1.3: Depiction of the forward model described by Eq. 1.1. A time domain back-projection algorithm for image reconstruction inverts this forward model to solve each pixel for $\sigma(x)$. Since each pixel in the integration can be processed independently, this formulation lends itself to being straightforward to parallelize on a graphics processing unit (GPU).

Figure 1.4: Number of search hits on Google Scholar by year of the term “synthetic aperture sonar” (quotes included in query).

compression (DRC) to the magnitude portion of the SLC. It is not uncommon for SAS magnitude imagery to have dynamic exceeding 100 dB making its display problematic. DRC is employed further to improve human interpretability amid this obstacle. We specifically mention the relationship of the SLC image to the DRC image because it plays a part in the contributions made by this dissertation. The DRC algorithm used throughout this work is based on the rational mapping function of [14] but with an improvement to automatically adjust overall image brightness from [15].

The earliest works in SAS were theoretical, describing its potential. However, the
technology to practically deploy such systems was lacking during this development period. As technology matured in energy storage, compute efficiency, data storage density, and analog-to-digital conversion, the creation and testing of towed SAS systems commenced. In towed systems, a surface vessel tows a sonar-equipped vehicle, and the captured sonar data is usually sent back to the vessel over a cable in real time. The collection of SAS imagery from early towed systems resulted in several popular papers from the University of Canterbury in New Zealand. Their group’s work laid out the fundamentals of SAS on many fronts leading to the maturation of the area (e.g. [12,13,16–19]).

Eventually, practitioners integrated SAS with modern unmanned underwater vehicles (UUVs), and they predominantly use them this way today. Today, systems can produce imagery on the UUV during a survey [15,20]. Such capability allows for the automated analysis of this imagery in situ where UUV autonomy engines modify UUV behavior in real-time according to the SAS image they generate onboard [21–25]. Furthermore, UUVs have allowed deviations from standard stripmap geometry to obtain more angular looks of the ocean floor. Geometries such as circles and corkscrews are now possible, allowing sophisticated signal processing to generate stunning, high-resolution imagery products containing multi-faceted acoustic scattering phenomenology [26,27]. Finally, SAS has even been applied in downward-looking geometries for subsurface imaging for unexploded ordnance (UXO) detection [28].

Collecting SAS images is time-consuming and logistically challenging [29]. Surveys usually require divers to obtain seafloor ground-truth and chase boats for UUV deployment and recovery. These factors motivated the need for SAS simulation, and such capability has existed since at least the mid-1990s [30]. The need for simulation persists as the capability of SAS increases. Today, models such as the point-based sonar scattering model (PoSSM) [28,31] simulate sophisticated seafloor environments using procedure textures and bioturbation models. These methods provide raw sonar time series (in addition to SAS imagery) at a fidelity capable of image reconstruction through existing sensor-driven processing chains without needing special attention.

Although, underwater sonar was historically pursued for military purposes including mine countermeasure applications [32], the field has matured and the commercialization of SAS systems has become feasible. SAS is now a capability accessible in the civilian space with several companies offering systems [33] [34]. Today, SAS is a popular undersea remote sensing tool with a rich research community as Figure 1.4 shows the growth of research interest in SAS since its inception.
1.2 Open Challenges in Machine Learning for SAS

Deep learning consumes the majority of practice for contemporary machine learning (ML). Despite its successes, it is not a panacea. Following, we outline three key deficiencies of current deep learning methods concerning the automated processing of SAS imagery:

1. **Deep learning works well when abundant labeled data is available, but the collection of labeled SAS imagery is non-trivial, thereby making large labeled datasets difficult to obtain.** SAS systems are usually mounted on a UUV where surveys require: deploy/recover teams for the UUV, chase boats to ensure safe UUV operation, divers to obtain seafloor ground truth, and human analysts to make sense of the collected imagery. Such hurdles prevent the formation of large SAS datasets for which deep learning methods necessitate to obtain good performance. For example, the largest dataset available for this dissertation consists of roughly 55,000 images with a few object categories, whereas ImageNet [35], a dataset often used to train deep networks, contains 1.2 million images and 1,000 categories. SAS does not have the luxury of relying on high quality, abundantly labeled datasets for training deep networks as enjoyed by the consumer imaging space.

2. **Deep networks are largely designed around optical image datasets, but SAS imagery exhibits different collection geometry and statistics than optical imagery driven by the SAS phenomenology.** There is often inherent structure in SAS imagery which is not explicitly represented as X-Y training pairs and therefore neglected as part of the learning process. Such information often contains discriminating features which are robust across environments. Furthermore, stripmap SAS imagery is collected perpendicular to the synthetic aperture resulting in the scene being observed acoustically by low grazing angles, a geometry rarely used in optics and consumer imaging. Also, the synthetic aperture process necessitates the use of coherent processing resulting in image speckle, which is inherent to the produced images, a phenomenon absent in optical imagery. In addition, the pixels are a fixed size everywhere in the scene, a scenario common to aerial imagery but rarely present in optical/consumer imaging. Figure 1.5 shows wavelet coefficient statistics for a SAS database used in this work and compares them [36] to a consumer imaging dataset, Common Objects in Context (COCO) [37]. Consequently, SAS imagery occupies a different manifold than optical images when presented in a
high dimensional space and deserves special care when applying deep network architectures developed around optical consumer grade imaging to obtain best performance.

3. **Almost all SAS machine learning methods use dynamic range compressed magnitude SAS imagery as input discarding the useful phase portion of the complex image.** DRC’ing the imagery is necessary for display but completely discards the phase portion of the image, which contains acoustic angle-of-arrival (AoA) and frequency information, phenomena useful for scene interpretation. Although convenient for display, the DRC process has largely carried over into SAS ML methods as the preferred data input for most algorithms. As a result, these methods do not exploit any acoustic AoA features during learning. The result is that the vast majority of ML methods proposed for SAS imagery operate only on magnitude, real-valued imagery, ignoring the rich acoustic information available in the native complex image resulting from image reconstruction.
1.3 Improving Deep Learning Robustness with Domain Enriched Deep Learning

Deep learning (DL) methods perform well when training data is available in significant quantities, of high quality, and no missing labels. However, obtaining such data in undersea remote sensing like SAS is difficult [29]. This work proposes the incorporation of domain knowledge into deep learning networks for SAS to overcome the lack of abundant, high-quality training data. We define domain knowledge as parcels of information encapsulated in the data but not explicitly realized as image-label pairs (i.e., X-Y training data pairs). We refer to deep learning methods incorporating domain knowledge as domain enriched deep learning.

Two means of applying domain knowledge to neural networks are: (1) physics-informed network architectures and (2) domain inspired regularization terms added to the training loss function. We will address both of these flavors in this work.

The first method of injecting domain knowledge into a neural network is by modifying its architecture so that features known to be physics-based and discriminatory are encouraged to be used by the network when making its decision. The physical parameters used to model the input phenomena are good candidates for such features. Often this model is differentiable, making integrating it into a deep learning pipeline feasible as gradient descent based methods used to train deep networks require a differentiable objective function. The physics-based parameters are learned as part of the training process and done so in a way that best fits the underlying physical model of the data. Moreover, these parameters can be used in a data-adaptive manner, allowing the network to handle input from various environments. Finally, the resulting parameters can be extracted after model training or during inference to explain the network’s decision since these parameters have physical meaning.

The second method of injecting domain knowledge into a neural network is by introducing additional terms in the training loss function. The additional terms serve as regularization and capture information known to be true about network outputs or intermediate results a priori. During training, these loss terms guide and influence the network to learn features consistent with the prior knowledge known to be discriminative based on the structure of the problem-at-hand.

Using a geometric framework, we motivate the benefits of using domain knowledge in improving deep learning when training data is sparse. For a given labeled dataset, we know that many decision boundaries are capable of minimizing the loss of the ML
task-at-hand. However, the addition of physics-based architectures and domain-specific loss terms (i.e., domain knowledge) encourages the network to learn decision boundaries forming structure that we know to be meaningful from the physical underpinnings of the data.

Figure 1.6 shows an example of how different decision boundaries can “solve” the learning optimization task. However, all may not generalize well to out-of-distribution data because of non-sensical assumptions made amid model ignorance. The figure shows two data classes, X’s and O’s, distinguished in three ways: (1) training samples: black solid line, (2) test samples: black dotted line, and (3) out-of-distribution (OOD) / distribution shift samples: purple dotted line. These OOD samples could be, for example, from a different system or context (time of year, environment, new system, system upgrade, etc.).

The OOD samples differ from the test and training set in that they are not available for use during training but usually arrive as the sensor is forward deployed. Examples of OOD samples are from the sensor placed in a novel environment or data from a similar modality but a different sensor. In the former, the ML fails to generalize if it learns features specific to the environments given by in training set. In the latter, the ML fails to generalize if sensor-specific idiosyncrasies are learned, thereby making it unable to perform well on similar input but from different sensor manufacturers. A recent study highlighted similar effects in ML training for X-ray pneumonia detection across hospitals [42]. In this work, the application of an ML model to hospitals outside of the training set catastrophically failed. Upon analysis, researchers determined that the ML algorithm used the source hospital as part of its decision despite obtaining good performance on their “test set.” The application of domain enrichment to deep learning helps mitigate such problems.

Figure 1.6 shows three decision boundaries for a ML problem: one separating only the training data (red), one separating only the training and test boundaries (blue), and one separating the training, test, and OOD data (green). We could imagine that the red boundary derives from a data-agnostic neural network, the blue derives from a kernel support-vector-machine, and the green derives from a domain enriched network. Our goal with this thesis is to capture the information needed to encourage the network to learn the green boundary, which captures the problem’s greater structure despite not being present in the training data (this structure arises from domain enrichment). Figure 1.6 shows where the ML practitioner may fool themselves into thinking they have a robust model. Because of the excellent performance obtained with the blue boundary on
Figure 1.6: Geometric diagram motivating the use of domain enrichment in deep learning. Here we show training, test, and out-of-distribution (OOD) samples (e.g., samples from a different but similar sensor). We also show three possible training boundaries that separate the training data. The red boundary minimizes the training loss but performs badly on the test data (i.e., overfitting to training set). The blue boundary does a reasonably good job separating the test data but not the OOD data (i.e., overfit to the test set). However, we can see that only the green curve captures the structure associated with all the data (i.e., robust). The green curve represents a solution based on domain knowledge of the problem at hand. It thereby captures the greater structure of the problem not explicitly represented as X,Y training data pairs. The difference between the green and blue boundaries can fool a practitioner into thinking their ML model is robust when it is not.

the training and test data, a practitioner may expect migration of the model to future sensors/environments to perform similarly. However, in reality, it fails, as shown by the OOD samples. Finally, the ML practitioner may observe similar test set performance from both the green and blue boundaries making it uncertain as to which model to deploy. Clearly, the model using the green decision boundary resulting from domain enrichment is the best choice.

Although this work exclusively deals with SAS imagery, the principles described will likely have good success on datasets of other phenomenologies. Likely, domain enriched deep learning may help solve some the distribution-shift problem researchers face in regime of optical image datasets using pure “black-box” deep networks. Recently, several researchers point out a a widespread problem in using optical datasets like ImageNet:
overfitting to the validation and test data. This was shown to be true even for large datasets like ImageNet [43]. Moreover, [44] demonstrated that neural networks are brittle to small perturbations like image translation despite good performance on a test set, once again indicating data overfitting. Finally, [45] showed that common image corruptions to optical imagery cause networks to catastrophically fail even when humans have no trouble classifying these images.

1.4 Overview of Contributions

Figure 1.7 shows an example SAS processing chain and notes this dissertation’s contributions. The chain begins with data acquisition by the sonar, usually aboard a UUV. The data from a transmitted acoustic pulse is received and digitized along with sonar array attitude data for the precise positioning needed to construct a SAS image. Next, an Image Reconstruction algorithm forms the data into a single-look-complex (SLC) image. At this stage, the SAS image is formed and is composed of complex-valued pixels. Next, an autofocus routine is applied to improve image focus, usually due to imprecise sound speed measurements or motion estimation produced by the previous two stages. Once complete, the algorithm discards the phase image, and the remaining magnitude image is dynamic range compressed to make it suitable for human consumption and display. Next, Image Enhancement ingests the image and despeckles it to improve image quality further. Finally, the image is passed to Object Recognition and/or Seafloor Segmentation for automated processing to aid a human analyst in decision making.

This dissertation makes three contributions to the processing chain of Figure 1.7 specifically in the areas of Autofocus, Image Enhancement / Object Recognition, and Seafloor Segmentation. The dotted line in the figure shows our proposed seafloor segmentation algorithm bypassing several processing stages by operating directly on the SLC, a novelty of our proposed method.

1.4.1 Contribution I: Prior Guided Object Recognition for Synthetic Aperture Sonar

Figure 1.8 depicts a timeline of popular SAS research themes in object recognition over the last few decades. An examination of the historical literature on ML in SAS revealed that many sources of information present in the data went unutilized. Moreover, human analysts use several cues in making sense of the imagery, such as the target-shadow
Figure 1.7: Example SAS processing chain showing places where this dissertation makes research contributions. Our proposed algorithm for Seafloor Segmentation traces a dotted line to the SLC as an input for our proposed algorithm, a novelty separating it from existing SAS segmentation algorithms that rely on the DRC image.

Figure 1.8: Selected list of popular SAS classification papers over the last twenty-five years. The purpose of this timeline is to highlight popular research areas and show the evolution of the field to its state today.

pairing of objects. Beyond this, however, many of these cues were rarely modeled in the existing literature.

My first contribution is described in Chapter 2 and motivated by two critical parcels of domain knowledge. First, image despeckling is often performed on SAS imagery before human consumption to improve human interpretation. Second, before processing by automatic target recognition (ATR), a lightweight detector algorithm is usually run over all the SAS images to prune benign areas of the seafloor allowing only interesting image chips to pass to the ATR. Noteworthy, the detector usually produces candidate detections with objects well-centered in the image chip.
For the first parcel of domain knowledge mentioned, I introduce a data-adaptive enhancement task in front of a typical classification network to enhance the image in such a way so that classification is improved. The formulation does not require ground-truth noised-denoised image pairs as such image pairs do not exist since the speckle phenomenon is inherent to the coherent imaging modality of SAS; anything provided by a despeckler is purely a subjective improvement. Note, image quality assessment (IQA) is a well-developed area for optical images with the paradigm of measuring the image quality of an image-under-test with respect to a reference image. I apply the same concept here but in reverse: I am given an image-under-test and need to determine the unknown reference image likely to improve classification performance due to its reduced noise levels. I augment an existing classification network with an image enhancement network, knowing that the observed image-under-test and a “clean” image will be structurally related. I enforce this constraint by using the differentiable (and thus amenable to gradient descent based optimization) multi-scale similarity metric (MS-SSIM). I call this parcel of domain knowledge the structural similarity prior (SSP).

For the second parcel of domain knowledge, I add a semi-supervised task to the existing classification task. This new task determines the target center position in an image when a target is present. A detector is often used as a pre-screener for object recognition (to reduce compute resources) and this detector usually centers on the object. As part of our data augmentation scheme, I input cropped translations of the input image when training the network. I leverage the known shifts as a semi-supervised task and make the network predict it in addition to its classification task. I call this parcel of domain knowledge the structural scene context prior (SSCP).

Both priors used to embed the domain knowledge are semi-supervised tasks, allowing them to be applied to most existing SAS datasets easily. Furthermore, this forgoes the need to create more hand-labeled training data, an time consuming and potentially expensive process.

I evaluate my proposed method on a medium-sized SAS dataset consisting of roughly 55,000 images. I compare my results against a variety of existing algorithms. I show that my proposed method yields performance gains above all existing methods especially in low training data regimes. Additionally, I show the necessity of each prior in obtain best results through an ablation study.

Associated References


### 1.4.2 Contribution II: Deep Multi-Look Sequence Processing for Synthetic Aperture Sonar Image Segmentation

SAS image segmentation provides the ability to characterize imagery spatially. It has an important role in: habitat and environmental monitoring [46–51], providing context for other automated algorithms like target recognition [52–56], and dynamically modifying vehicle behavior in order to maximize survey objectives [21–25]. Since SAS is an active sensing modality and operates at low grazing angles to the seafloor, the presentation of the seafloor in a SAS image is collection geometry dependent. For example, sand ripples running parallel to the collection track appear ripple-like in the image, whereas sand ripples running perpendicular to the collection track appear to lose their periodic nature. Practitioners observe the acoustic angle-of-arrival effects for a single SAS image when reprocessing the complex-valued SAS image (often called a single-look-complex (SLC)) only to show acoustic returns of a subset of receive angles. With such processing, non-isotropic seafloor scattering and objects moving during the scene collection become emphasized. Examination of the SAS segmentation literature did not return any existing methods exploiting this feature. Figure 1.9 depicts a timeline of popular SAS research themes in the area of seafloor segmentation over the last two decades.

My second contribution is described in Chapter 3 and utilizes the fact SAS imagery filtered intelligently in the $k$-space domain yields images with different “squints” or “looks” of the seafloor by steering the receive beam of the aperture. Sweeping over a consecutive set of look-angles creates a sequence of images akin to a movie calling attention to aspect-dependent scattering effects and motion of objects in the receive beam. Organizing such a process results from domain knowledge of the SAS imaging modality and is sometimes called multi-look sequence processing (MLSP).

I introduce a novel domain enriched network architecture utilizing magnitude and **phase** information contained in the complex-valued input SAS image to derive image
acoustic angle-of-arrival features. I will show that these features enable superior performance when labels and data are sparse, as is often the case in SAS datasets. Abundant labels in SAS data are challenging to obtain because of the difficulty in determining accurate pixel-level labels (discussed in Section 2.1.1). The proposed network architecture takes as input a single-look-complex SAS image atypical for current methods [50,52,57–66] which discard phase information and exclusively analyze the magnitude image. Our proposal seeks to utilize this often discarded data because of the rich information we know it contains from our domain knowledge of the problem. I show the network outperforms many recent methods and especially yields improvement in discriminating between acoustic shadow and dark sand classes where the former is an artifact of the imaging geometry and the latter is a proper seafloor type. Through an ablation study (Section 2.4.4) I show the improvements in classification performance by incorporating the phase in a domain enriched manner.

I introduce a set of “filter modules” each ingesting the input complex-valued SAS image and using $k$-space filtering to generate a sequence of images based on acoustic
angle-of-arrival (AoA), thus allowing our network to discover AoA-dependent features. Furthermore, I explicitly model this sequence temporally, which human analysts have demonstrated to be helpful in SAS image analysis. Specifically, I model the multi-look sequence using a recurrent neural network (RNN) (i.e., Bidirectional 2D long-short-term-memory (LSTM)), enabling us to capture temporal correlations in the image sequence. The multi-look sequence filters are initialized in an intelligent manner guided by domain knowledge of the SAS imaging platform, thereby creating an image sequence emphasizing the aspect-dependent acoustic reflections of the seafloor and scene movement in the acoustic path during the survey. For example, fish schools appear as fuzzy clouds in a static SAS image obscuring the seafloor resulting in segmentation difficulty. However, through $k$-space filtering, such occlusion may be mitigated (see Figure 3.3 for an example). Likewise, classes such as shadow are easily confused with dark sand but become discernible through multi-look sequence processing.

I introduce a filter-kernel formulation for the “filter module” which is fully differentiable and therefore learnable as part of the training process. The filter-kernel’s structure takes a band-pass form inspired by the $k$-space representation, succinctly capturing acoustic angle-of-arrival information. Further, we devise a differentiable form of the filter so filter bandwidth, offset, and attenuation are all trainable parameters during the learning process. Our filter form allows the network training in an end-to-end fashion without the need to explicitly filter the images (using a fixed filter) as a pre-processing step before the learning process.

I evaluate my proposed method on a weakly-labeled SAS dataset consisting of roughly 100 images. I compare my results against a variety of existing algorithms used in SAS and RAS image segmentation. Further, I perform an ablation study to show specifically that my contributions are what lead to best results among all the algorithms evaluated.

Associated References


1.4.3 Contribution III: Prior Guided Image Enhancement for Synthetic Aperture Sonar

Most ML research applied to SAS has focused on automated analysis of the DRC’ed human-consumable imagery resulting from the image reconstruction process. As previously mentioned in an earlier section, the resulting image from this process is called the SLC which contains complex-valued pixels. Also, recall that imagery produced for human consumption has the phase and large dynamic range discarded. In this work, I revisit the SLC and apply ML directly to it to enhance focus quality.

My third contribution is described in Chapter 4 and is motivated by the problem of automatically correcting defocused SAS imagery, a process called *autofocus*. Recall from Equation 1.3, only when the time-of-flight of each ping is well known will a well-focused image be constructed since this information is necessary to form the coherent synthetic aperture. Unfortunately, it is not uncommon to misestimate the time-of-flight when high vehicle motion, bathymetry, or heterogeneous speed of sound is present (this can occur, for example, when fresh water mixes with salt water in an estuary). Misestimating these quantities degrades coherent phase integration causing the image to defocus and the
resulting point spread function (PSF) to smear. Figure 1.10 depicts a timeline of popular SAS research themes in the area of SAS autofocus over the last three decades.

The inspiration for this work is [67], where the authors show that common systematic phase errors in the SAS image reconstruction process result in easily interpretable features by human analysts. The authors produce a figure akin to an “eye-chart” for practitioners to pattern match their imagery against to determine the type and source of these phase errors. My work ponders if computer vision techniques can automate such a procedure since humans can easily identify the phase error class; I answer in the affirmative and call the proposed method Deep Autofocus. Interestingly, many common classes of phase error result in symmetrical PSFs, which require the SLC phase to disambiguate. I also confirmed this during the work and demonstrate that the addition of phase resolves this ambiguity quite well. This feat is remarkable since the phase of a SAS image, although shown to have some structure in it [68], is mainly random by nature. Despite the randomness of the phase map, I will show that a convolution neural network (CNN) can discover valuable features capable of accurately estimating the phase error upon the aperture.

I evaluate my proposed method on a small SAS dataset consisting of roughly 400 images. I compare my results against a variety of existing iterative algorithms including phase gradient autofocus (PGA) which is a popular method in use today. I show that my proposed method yields gains in robustness, compute burden, and image quality over the comparison methods.

**Associated References**


1.5 Organization of Dissertation

Chapter 2 describes a method for SAS object recognition (referred to as automatic target recognition (ATR) in its context) using structural priors related to SAS image enhancement and target placement in detector output. Next, Chapter 3 describes a method for SAS image segmentation using the phase information contained in the SLC to form a multi-look-sequence having temporal and spatial features, which is exploited by a recurrent neural network. Next, Chapter 4 describes a method for SAS autofocus (also using the SLC like the proposed segmentation method) using deep learning with the ability to be trained in either an unsupervised or fully-supervised manner. Finally, Chapter 5 summarizes the methods, their results, and proposes future research areas.
Chapter 2  
Contribution I: Structural Prior Driven Regularized Deep Learning for Sonar Image Classification

2.1 Introduction

Contemporary SAS systems are capable of producing image quality that lends itself to tasks such as automatic target recognition (ATR). Despite the improvements, the problem of detecting and classifying objects in imagery remains challenging because of distractors in the environment and the complex configurations possible by targets. Figure 2.1 shows examples of difficult cases along with prototypes of the object classes used in this work.

2.1.1 Open Challenges in SAS ATR

SAS ATR algorithms were originally established from ATR algorithms used in side scan sonar, which we call real aperture sonar (RAS). RAS systems have similar collection geometry to SAS, but cannot produce the constant resolution with range achieved by SAS. We refer the reader to Chapter Two of [12] and Chapter Three of [69] for the differences between RAS and SAS imaging. Despite these differences, initial SAS imagery looked similar enough to RAS for researchers to reasonably justify the use of RAS ATR algorithms on SAS.

Some of the more popular early work in sonar ATR involved the use of kernel filters [70, 71]. When large amounts of SAS imagery began to be produced, these were some of the first techniques applied to it. Over time, these methods began to utilize multiple target looks to aid in classification by exploiting the overlapped coverage
Figure 2.1: SAS is capable of producing high-quality, high-resolution imagery of seafloor and objects. The images here are examples collected from the Centre for Maritime Research and Experimentation MUSCLE system. In this work, we provide an ATR algorithm to classify MUSCLE images into four classes. Example objects from each class are shown: (a) background, (b) cylinder, (c) truncated cone, and (d) wedge. Difficulties in classification result because there are often objects which appear target-like (e,f), but are not targets (false alarms); and some targets are difficult to discern (g,h) because of orientation, burial depth, or background topography causing them to be ignored (missed detections).

of the seafloor most SAS surveys exhibit [72]. Other techniques focused on model-based approaches [73] and uncertainty modeling. Eventually, the popular classification algorithms of the early 2000’s, before the boon of deep learning, were applied to imagery including decision trees [74] and Markov random fields [75].

Coincidentally, this paper is about the use of neural networks (NN) to address the classification problem. The use of such techniques is not new and one of the more popular early works employed them [76].

With all the recent success of SAS, there remains persistent challenges with respect to ATR. One of the biggest challenges for obtaining good results is collecting and labeling large amounts of imagery which is needed for contemporary machine learning (ML) algorithms using deep learning. SAS collection from unmanned underwater vehicles (UUV’s) requires an inordinate amount of support infrastructure including: support vessels, ship crew, and divers making the endeavor financially expensive. Furthermore, the objects often sought upon during surveys are scarce.

It is also difficult to create an environment-independent ML algorithm when little
training data is available. Practitioners quickly discover that deploying classification in unseen environments often results in high false alarm rates, even for state-of-the-art SAS ATR algorithms. Despite this, the detection performance of these methods is often quite good as the literature shows. However, all the false alarms returned by the algorithm quickly overwhelm human operators. Furthermore, objects simple to rule out by humans are often called by the ATR, preventing trust between the operators and the algorithms – this renders the ATR useless. When combined, these factors result in manual human inspection as a preferred means to cull the imagery; a costly process.

2.1.2 Overview of the Proposal

We present a deep learning classifier exhibiting significantly reduced false alarm rates compared to contemporary SAS ATR algorithms while maintaining high detection accuracy. Our approach integrates high-level, domain knowledge unique to the SAS domain in order to achieve these good results. We do this by integrating parcels of domain knowledge, which we call priors, into the training objective.

For a given problem, there exist attributes which are directly represented by the given training data. In the case of image classification, this would be the images/label pairs used for training; this information is explicitly provided to the training algorithm. This is the common scheme for the vast majority of image classification problems. However, there exists domain-specific information which is projected into the training data but may not be explicitly represented by it. For example, for the dataset used in this work, we have some knowledge of how it was pre-processed before given to us for use. Specifically, we know that the detection algorithm used to produce the image chips is generally good at centering targets within the chip. However, we do not have any kind of bounds or statistic on how well the targets are centered. Furthermore, the true target centers are not explicitly encoded for each image. Thus, the fact that the detector is reasonably good at centering the targets is domain knowledge derived from a subject matter expert (SME) (we will see this forms the scene context prior which will we discuss in future sections). We refer to these parcels of domain knowledge as priors. In a deep learning framework, each prior is employed through a regularization loss which is augmented to the primary task’s objective function. Figure 2.2 is a Venn diagram illustrating the concept.

In this work, we define two priors which, when used individually each improve classification performance, but when used together, act synergistically to improve performance beyond the use of each exclusively. The first prior we use addresses an often overlooked part of the SAS image reconstruction pipeline: image enhancement. This prior originates
Figure 2.2: The relationship of priors to the problem domain and the training data. Our priors encapsulate domain knowledge which is not explicitly represented in the data but projected into the data. The priors used in this work, structural similarity and structural scene context, are employed through regularization losses which are augmented to the primary task’s objective function which is classification error. We jointly train all losses so the network finds a minimum consistent with both the data and the domain priors.

from the domain knowledge that image enhancement algorithms applied to SAS imagery aid in improving human interpretation. An example of such an algorithm is despeckling [77]. We name this prior the structural similarity prior because it encapsulates the function of any image enhancement algorithm: improve scene content in a way which improves downstream task performance (in this case classification) while simultaneously preserving scene structure and semantics. Quantitatively, this prior is captured by the regularization term in Eq (2.3) which is described in detail in Section 2.3.3.

The second prior we use leverages a common quality exhibited by detection algorithms: the ability to localize targets. The majority of SAS classification algorithms are preceded by a detection algorithm whose purpose is to quickly find target-like objects in the queue of mostly-benign seafloor images. This prior originates from the domain knowledge that the detector algorithm is usually able to localize the target in the image which humans also do when parsing a scene. We name this prior the structural scene context prior because it encapsulates the role of ground truth target position knowledge: well learned features for image interpretation should encode target location. The images output by the detector usually center the target and we can translate the target through image crops. We then encourage prediction of the new ground truth target location but using the same features used for classification. In this manner, we improve the quality of the
features which consequently improves classification performance. Quantitatively, this prior is captured by Eq (2.4) which is described in detail in Section 2.3.4.

Recall that our two priors, structural similarity prior and structural scene context prior, exist for the purpose of improving our primary task: classification. The domain knowledge captured by these two priors is employed through the use of regularization losses, Eq (2.3) and Eq (2.4), augmented to the primary task objective function, Eq (2.6). Together this forms the final loss we jointly-optimize during training, Eq (2.7).

Using the aforementioned priors above, this research makes the following technical contributions:

1. Image enhancement through despeckling is often used to improve image interpretability for humans. We ask the question, *Is there an image enhancement function which improves classification?* to which we will answer in the affirmative (results in Table 2.3). To this end, we incorporate a data adaptive image enhancement network with a self-supervised, domain-specific loss to an existing classification network for purposes of improving classification performance. Our image enhancement function is learned from the data removing the onerous task of selecting a fixed despeckling algorithm. Furthermore, ground truth noise/denoised image pairs as required by previous methods [78] are not needed.

2. Most SAS ATRs only determine the presence of a target object but are aloof to where in the image it appears. To this end, we incorporate a target localization network in addition to a classification network for the purpose of also improving classification performance. Like the image enhancement network, this is also trained using a self-supervised, domain-specific loss. Now, our classifier not only learns target class, but also target position thus acquiring scene context. Our target localization network is trained using the domain knowledge that objects are centered when passed to the classifier. We use the common data augmentation technique of image translation through cropping to induce new target positions when training, and have the target localization network estimate the induced target position in addition to the primary task of classification. This encourages the model to learn “where” of the scene in addition to the “what”.

3. We train the two aforementioned networks and the classification network
Table 2.1: Brief summary of the loss terms in our objective function admitted from our domain priors. We list the relationships among the incorporated domain knowledge, the employed prior, and the associated regularization loss in our formulation. We also list the primary task, classification, and its loss function for completeness.

<table>
<thead>
<tr>
<th>Prior Name</th>
<th>Employed Knowledge</th>
<th>Domain</th>
<th>Loss Type</th>
<th>Loss Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural similarity</td>
<td>Image enhancement improves human interpretability</td>
<td>Domain-specific</td>
<td>Equation 2.3</td>
<td></td>
</tr>
<tr>
<td>Structural scene context</td>
<td>Targets output from detector are image centered</td>
<td>Domain-Specific</td>
<td>Equation 2.4</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>Primary Task, Classification</td>
<td>Equation 2.6</td>
<td></td>
</tr>
</tbody>
</table>

simultaneously through the addition of regularization terms to the primary classification loss objective function. This makes our formulation self-supervised and thus requires no extra data or labels making it suitable as a drop-in replacement for training against existing datasets. Table 2.3 shows through ablation that each domain-specific loss improves classification performance and that when combined, the best classification performance is achieved.

Both of our priors incorporate structural domain knowledge so we call our method Structural Prior Driven Regularized Deep Learning (SPDRDL). Each prior mentioned has never used in previous SAS classification works. Table 2.1 shows the relationship among losses used in our final objective function.

2.2 Previous Work

Recent SAS ATR schemes have focused on improving feature representations through various means. For many years, representations were hand crafted and much of the research was in attempting to discover useful features through subject matter expert input. Techniques employed bag-of-words models [79] using handcrafted features as complex vocabularies describing SAS images. Eventually, techniques emerged which removed the need for this explicit feature engineering task. Dictionary learning methods were some of the first to forgo the explicit feature engineering path [80,81] and automatically learn features as part of the classification process. Today, deep learning techniques are employed in the same vein [82].
Recently, investigations into alternate representations to improve classification have shown to be a fruitful endeavor. [68,83,84] have examined representations derived from the k-space and have found they contain useful information for classification. Traditionally, the human consumable image, which arrives after extensive post-processing of the raw SAS data, has been used for input to the classifier. The human consumable image is the result of a lengthy signal processing pipeline which discards information related to the frequency and direction of the received acoustic wavefronts. A coarse explanation of a typical image reconstruction pipeline is as follows: (1) raw sonar echos are collected from the sonar array over multiple transmissions, (2) signal processing is applied to these echoes to correct them for imperfections, (3) the data is matched filtered to obtain resolution in the range dimension, (4) an optional motion compensation step is performed to interpolate the data to a regular grid (e.g. preparation for $\omega$-k beamforming), (5) the data is beamformed to generate a single look complex (SLC) image, and (6) a human consumable image is formed by taking the absolute value of the SLC and applying dynamic range compression (DRC). Consequently, the absolute value operation removes the phase portion of the SLC potentially discarding useful information.

Deep learning has been applied to sonar ATR resulting in a substantial improvement in classification performance. An initial work in the area is [82] whereby convolutional neural networks (CNNs) were used to automatically learn features for classification. In [85], the authors demonstrated the canonical transfer learning approach commonly used in training data-limited networks works well for SAS; a pre-trained CNN trained on the Imagenet dataset [35] was fine-tuned on SAS imagery yielding good results. This work was generalized in [86], where the authors integrate the feature learning of both SAS images and selected photographs simultaneously, yielding good ATR performance in the midst of limited training data. Finally, transfer learning among SAS sensors was demonstrated in [87] where a CNN initially trained on one SAS sensor was used to quickly train with another.

In several of the works described thus far, class imbalance has been mentioned as a noteworthy issue. Many SAS datasets have far more imagery of the benign seafloor than of objects of interest. To combat this issue, general adversarial network (GANs) have recently been applied to SAS for the purpose of generating more training data to balance the classes. In [88], a hybrid simulation and GAN based approach is used to generate a simulated, optical version of the desired scene and then a learned transform is applied to the simulated scene to give the appearance of a real SAS image. Their hybrid approach gives fine control over the generated scene content so the data balancing
procedure can be accomplished with precision; particular objects, their orientations, and their range from the sonar can be specifically generated. Model based GAN approaches have not been limited to SAS, but also have been used for real-aperture sonar (RAS) systems as described in [89] where GANs are applied to RAS imagery to augment data for underwater person detection.

Today, SAS systems are multi-band and operate over several frequency ranges. This ability has not been overlooked in the context of ATR. An early work utilizing multi-band sonars for classification is [90]. In this work, the authors demonstrate good detection performance when using a low-resolution broadband sonar in addition to a high frequency SAS. Even more recently, deep learning has been applied to multi-band SAS imagery with good success and without the need of using a pre-trained network [91,92].

2.3 Proposed Classification Method: SPDRDL

2.3.1 Motivation of Approach

Recent ATR schemes using deep learning demonstrate great performance but at the cost of requiring large amounts of training data. As previously discussed, SAS data collection is costly resulting in small datasets which are almost always class imbalanced. Consequently, it is crucial to use all the available information from a SAS image during classifier training. To this end, we propose a new scheme which incorporates prior knowledge of SAS images in a novel way as a mechanism to extract more information from each image. This additional information is used to positively influence classifier training.

One mechanism by which we inject prior knowledge into the classification pipeline is by addressing the inherent speckle phenomenon present within every SAS image. The speckle is often seen as noise and a nuisance for human interpretation. Much work has been done in the development of despeckling algorithms [77] with the purpose of enhancing image interpretability. A natural outcome of this work is to ask if such types of enhancement are beneficial for improving classification performance and if so, which methods provide the most benefit. Furthermore, can we forgo the onerous choice of selecting an enhancement algorithm and have the network learn the image enhancement transform in an unsupervised fashion?

Another prior which thus far has been overlooked in SAS ATR, are the assumptions given by the detector, sometimes called a pre-screener. As background, traditional SAS
ATR methods use a detector-classifier approach. In this approach, a simple detection algorithm is first passed over the scene. The detector produces candidate images of interest, sometimes called *chips*, which are then passed to a classification algorithm for further inspection. Usually the detector is computationally efficient and can quickly prune areas of the image which appear to be benign (e.g. a flat sandy sea-floor). Such a process reduces the amount of imagery the classifier has to process. It is believed such an approach was adopted initially for compute reasons – early SAS classification systems were not capable of processing every possible sub-tile of an image in a timely manner due to limited compute power. However, current compute capabilities, specifically in the form of graphics processing units (GPUs), provide ample compute power enabling a classifier to examine the whole scene quickly, removing the need for the explicit detection step. Notwithstanding, for this approach to work, the classifier must be translation equivariant.

Good detectors can localize the target well and output SAS images with the target well-centered in the image. The Mondrian detector [93] is a good example of such a detector. It uses prior knowledge of the target and sonar geometries to model expected relationships among local pixel neighborhoods; its quite capable for returning well-centered targets to a classifier. However, current classifiers for SAS do not use this information. They assume a target is present in the image, but do not explicitly estimate or assume its position. On the other hand, our proposed method jointly estimates target class and target position.

Because our proposed method estimates target position (in addition to object class), it is desirable to have a feature space embedding which is translation *equivariant*. By equivariant, we mean that as the target translates smoothly across an image, its associated embedding also translates smoothly. Despite the convolutional nature of CNNs, they do not inherently provide translation equivariance. Recent works such as [44, 94, 95] have pointed out this common misnomer and have made progress towards improvement. We utilize these techniques in our proposed method making it very robust to scene translation.

Having the classifier robust to translations has an added benefit in that we can forgo the traditional detection step and run the classifier on across the entire scene. This has an immediate benefit: the detection rate of the ATR is no longer bounded by the detector performance. For example, if a detector exhibits an eighty-percent detection rate, the overall ATR can do no better than an eighty-percent detection rate. Hence, even with an oracle classifier, the best detection rate that can be achieved is eighty-percent.
2.3.2 Feature Extraction Network

Traditional image classification pipelines using deep learning are composed of a feature extraction network followed by a classification network. Much recent work has been spent on designing an optimal feature extraction network as illustrated by the vast number of off-the-shelf (OTS) options available. DensetNet [96], Resnet [97], Inception [98], MobileNet [99], VGGNet [100], and AlexNet [101] are popular examples of such OTS networks. We leverage the good results of these OTS network architectures and begin the construction of SPDRDL around a popular one: DenseNet-121. SPDRDL is composed of Densetnet-121 as a feature extraction network (pink box) followed by a standard classification network (yellow box) shown in Figure 2.3.

2.3.3 Structural Similarity Prior Via Data-Adaptive Image Enhancement Network

Building upon the feature extraction and classifier networks, we introduce a data-adaptive image enhancement network which is added to the front of the feature extraction network. This enhancement network is shown in the blue box in Figure 2.3. The purpose of this network is to learn an image transformation which improves classification performance while still maintaining the original image semantics by obtaining an enhanced image (i.e. enhanced for classification purposes not necessarily human consumption) that is structurally similar to the original. We implement this network as a U-Net architecture [102] with the original image as input and the enhanced image as output.

To encourage image enhancement for classification, we utilize a novel loss function between the desired enhanced image and the original, the multiscale structural similarity measure (MS-SSIM) [103, 104] which is a scale-aware version of the SSIM measure,

$$SSIM(x, y) = \left( \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \right)^\alpha \cdot \left( \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \right)^\beta \cdot \left( \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \right)^\gamma$$ (2.1)

where $x$ and $y$ are images being compared, $\mu$ and $\sigma$ are the patch-wise mean and standard deviation respectively of the corresponding image, $\sigma_{xy}$ is the covariance between image $x$ and $y$, $\{\alpha, \beta, \gamma\}$ are shaping constants, and $C_{1,2,3}$ are calibration constants. $SSIM(x, y) \in [0, 1]$ where higher values indicate higher perpetual similarity between the images. The SSIM is differentiable and tractable for incorporation into a deep learning...
MS-SSIM introduces scale dependence by computing structural and contrast factors of SSIM over several staged, low-pass-filtered versions of the input image and then combining their results. It is given by,

\[
\text{MS-SSIM}(x, y) = [l_M(x, y)]^{\alpha M} \prod_{j=1}^{M} [c_j(x, y)]^{\beta j} [s_j(x, y)]^{\gamma j}
\]

(2.2)

where functions \( l, c \) and \( s \) represent the corresponding luminance, contrast, and structural components of Eq (2.1) respectively, and \( M \) is the total number of scales to evaluate. For all constants, we use the same values as specified in [104].

By seeking to maximize the MS-SSIM between the original and the enhanced image in the enhancement net, we leverage human visual system priors designed into the MS-SSIM perceptual loss function [106]. This utilizes the desired domain knowledge which we seek to embed in our formulation: there exists an enhanced image which is structurally similar to the original image but is able to yield improved classification performance. Finally, we define the structural similarity prior (SSP) regularization loss as,

\[
L_{\text{SSP}}(x, x_{\text{enhanced}}) = 1 - \text{MS-SSIM}(x, x_{\text{enhanced}})
\]

(2.3)

where \( x \) is the input image and \( x_{\text{enhanced}} \) is the improved image output by the enhancement network. Without this loss term, the network has no notion of a “noise model” and simply seeks to minimize the weights of the function with no understanding of the hand-crafted network structure we designed to exploit the domain prior.

### 2.3.4 Structural Scene Context Prior Via Target Localization Network

As previously mentioned, the detector returns targets centered in the image. During the data augmentation process, these targets are translated by a random amount. This augmentation procedure is commonly used in other SAS ATR methods. However, our method is different in that we do not discard the translation parameters but encourage the network to estimate them while simultaneously performing classification. In this manner, we embed the target position domain knowledge into the feature extraction network by encouraging it to learn a spatially-aware context of the scene in addition to features for classification. With this prior, the likelihood of the network learning features
Figure 2.3: The SPDRDL network architecture; the network input is a SAS image and the output is a classification and target center position when a target is present. SPDRDL is composed of four modules: image enhancement network, feature extraction network, target localization network, and a classification network. SPDRDL leverages two domain priors to improve classification: (1) image enhancement algorithms like despeckling improve image interpretability and (2), the detector produces SAS images with well-centered targets. For the former, an enhancement network leverages the human visual system priors incorporated into the structural similarity prior to enhance the image for classification. For the latter, input images are translated as part of the data augmentation procedure during training and this translation is estimated in addition to predicting image class. Image classification, enhancement, and target localization are simultaneously trained.

which are not target-centric is reduced, and the creation of features derived from biases within the dataset, like seafloor texture, is reduced.

We encourage the network to learn target localization by augmenting our feature extractor with a target localization network whose task is to estimate the target position from the feature embedding. Recall that the ground truth for this estimate is determined through the data augmentation procedure. The target localization network is represented by the orange box in Figure 2.3. It is composed of a set of $1 \times 1$ convolutions to reduce the dimensionality of the embedding. This reduction serves as a bottleneck which then feeds two dense layers which both have no post-activation function simply returning the position estimates. Formally, we define the structural scene context prior (SSCP) regularization loss as,

$$
L_{SSCP}(p_{\text{shift}}, \hat{p}_{\text{shift}}) = \frac{1}{2} \| p_{\text{shift}} - \hat{p}_{\text{shift}} \|^2 \tag{2.4}
$$

where $L_{SSCP}$ represents the mean-squared error between the shift (i.e. translation) applied during data augmentation, $p_{\text{shift}}$, and the shift estimated by the network, $\hat{p}_{\text{shift}}$.

Figure 2.4 shows an example of how the positional shifts are created. First, the Mondrian detector returns an image with the target centered. Next, a random crop is applied to the image during data augmentation. This induces a translation of the target in the image. We denote this translation as $p_{\text{shift}}$ and add this information to the network.

31
Figure 2.4: The detector returns well-centered target images. We can use this prior during our data augmentation procedure. During data augmentation, we translate the image via random crops and record the effective translation shift induced as \( p_{\text{shift}} \). To incorporate this prior into the network, we add a target localization task which encourages the network to estimate this translational shift in addition to outputting classification. As an example of this procedure, the detector returns a well-centered target image (left) and random crops of this image are fed to the network as data augmentation (middle, right). The target localization network estimates the difference between the true center of the target (green) and the shifted center (red). The translation shift estimate is denoted as \( \hat{p}_{\text{shift}} \).

via the backpropagation through the loss of Eq (2.4).

Thus far in this sub-section, we have developed a novel method to encourage the network to learn target position within the scene, and we have done so in a self-supervised fashion. One assumption we have made which we has not yet addressed is to assume that the feature space is translation equivariant. Despite the use of convolutional layers in our network, non-unitary strides associated with convolution and pooling operations prevent translation equivariance as we will show. Additionally, we have not provided local pixel positioning information to the network likely resulting in position information being determined by specific neurons in the dense layers which is undesirable for generalization. In the next two subsections, we will address each assumption.

### 2.3.4.1 Addition of Anti-Aliasing Filtering Before Pooling Layers

Most CNNs are not inherently shift invariant [44] when combined with pooling layers. This is caused by the lack of proper filtering done during image subsampling in pooling and convolutional layers when the stride is greater than one. Strided layers perform two operations: (1) a filtering procedure which is run over the entire image (in the case of max pooling, this is an order-statistic filter), and (2), image subsampling to reduce the image dimensions most commonly done by striding to reduce the compute burden. The striding operation is subsampling the image for the purposes of decimation. During this procedure, the energy of the discarded frequencies is folded into the desired lower
frequency band reducing the signal-to-noise ratio (SNR) of the resulting embedding. This results in a feature space which is not translation equivariant: translations in the input image do not correspond to translations in the feature embedding.

We can overcome the faults of the traditional pooling layer by introducing an anti-aliasing (AA) filter before all strided operations [107]. In our setup, this means placing an AA filter before all strided convolutions and pooling operations. The AA filters prevents out-of-band frequencies from aliasing back into the remaining spectrum post-subsampling. This results in increasing the signal-to-noise ratio (SNR) of the embedding and to encourage translation equivariance.

2.3.4.2 Feature Position Encoding

As previously mentioned, CNNs do not naturally provide translation invariance when used in tandem with strided convolution and/or pooling layers. In addition, [108] demonstrated that CNN’s have difficulty with position oriented tasks because they do not encode feature positions. Indeed, this at first seems surprising given the translational nature of the convolution operator. However, the convolution operator takes as input a 2D map and also outputs a 2D map; a feature’s position through this process is a function of the map domain but not explicitly coded in the representation. Hence, when a 2D map is flattened and used as input to a dense layer, a feature’s position is lost.

The interesting problem of CNNs not recording positional information was not only noted by [108]. In an early and popular work, [109] noted that CNNs are good at providing “what” but not the “where.” They specifically design their CNN architecture to compensate for this fault. Furthermore, the supplementary material of [110] also notes this as they cite the addition of positional information improved image in-painting tasks for their Deep Image Prior technique.

We augment SPDRDL with target position information by using the CoordConv solution of [108]. In particular, we augment the output of the image enhancement network, $x_{\text{enhanced}}$, with two additional channels, each one describing a positional dimension of the input map as described by Eq (2.5),

$$
V, H \in \mathbb{R}^{h \times w}
$$

$$
V[k,l] = l/w - [l/2]
$$

$$
H[k,l] = k/h - [h/2]
$$

(2.5)

where $V$ and $H$ are the generated 2D maps augmented to the channel dimension of the
input, \( h, w \) are the height and width of the image respectively in pixels, and \( k, l \) are pixel locations.

### 2.3.5 Classification Loss

Categorical cross-entropy is a commonly used loss function for penalizing classification error in neural networks. It accounts for errors probabilistically by measuring the amount of surprise between the predicted and true labels. The measure works well in the presence of balanced class and accurate labels. However, we know for SAS that the number of negative examples far outweighs the positive examples.

To mitigate the shortcomings of categorical cross-entropy in the presence of class imbalance, we use a specified weighted version of the measure called the focal loss \([111]\). The focal loss is given by Eq (2.6),

\[
L_{FL}(y, \hat{y}) = \sum_{c=1}^{N} -\alpha(1 - \hat{y}_c)^\gamma y_c \log(\hat{y}_c)
\]  

(2.6)

where \( N \) is the number of classes, \( y_c \) is the true probability of class \( c \), \( \hat{y}_c \) is the estimated probability of class \( c \), and we use the strength coefficients given by the paper of \( \alpha = 0.25 \) and \( \gamma = 2 \). Focal loss is a weighted version of the cross-entropy loss whereby correct classification is de-weighted. Consequently, the effect of the focal loss is to place more emphasis on grossly mis-classified samples compared to virtually correct classified samples. Through the use of the focal loss, error gradients of correct classifications are greatly diminished during training time while grossly incorrect classification maintain their error magnitude. In this way, the focal loss focuses the training on the misclassification samples and largely leaves the easy, correct classifications untouched.

There are several ways to place emphasis on negative samples during training of which a common one is to assign label weights. However, we chose the focal loss because of several positive properties it offers for our setup. Following, we describe each.

The first benefit realized by focal loss is that it can be viewed through the lens of importance sampling \([112]\) but without the explicit overhead associated with such techniques. Recently, \([83]\) showed importance sampling works well to improve the performance of SAS ATR. In importance sampling, misclassified samples are shown more often during the training procedure than correctly classified samples. In a similar manner, using the focal loss can instill a similar training policy without the overhead of maintaining a list of the misclassified samples. Using the focal loss, a mini-batch of
images is fed to the training algorithm and the misclassified samples are dynamically weighted proportional to their error. For each batch, correctly classified samples induce little error gradient and effectively are removed from the batch.

The second benefit realized by focal loss is that the effective batch size is reduced over time. Reducing batch sizes has been associated with better generalization error [113]. Assuming the distribution of easy- and hard-to-classify samples is uniform throughout the minibatch, at the beginning of training all samples in the minibatch are considered hard-to-classify. As training progresses, some samples become easier-to-classify and their error gradients vanish, effectively removing them from the minibatch reducing the effective minibatch size.

2.3.6 SPDRDL: Jointly Learned Image Enhancement and Object Location Estimation

In the previous sections, we examined sources of structural information in SAS images currently not utilized by contemporary ATR methods. Our proposed approach builds upon an existing CNN backbone network commonly used for feature extraction by utilizing this overlooked structural information. In this section, we bring together the aforementioned sections and fully present our proposed method, SPDRDL.

Incorporating the losses discussed in the previous section, we arrive at the final loss function for SPDRDL, Eq (2.7),

\[
L(\Theta_{\text{enhance}}, \Theta_{\text{reg}}, \Theta_{c}, \Theta_{\text{fe}}) = L_{\text{FL}}(y, \hat{y}, \Theta_{c}) + \lambda_1 L_{\text{SSP}}(x, x_{\text{enhanced}}, \Theta_{\text{enhance}}) + w\lambda_2 L_{\text{SSCP}}(p_{\text{shift}}, \hat{p}_{\text{shift}}, \Theta_{\text{reg}})
\]

where \(x\) is the input image, \(x_{\text{enhanced}}\) is the data adaptive enhanced image, \(y\) is the true target class, \(\hat{y}\) is the predicted target class, \(p_{\text{shift}}\) is the true target translation, \(\hat{p}_{\text{shift}}\) is the estimated target translation, \(\Theta_{\text{reg}}\) are the target localization network parameters, \(\Theta_{\text{enhance}}\) are the image enhancement network parameters, \(\Theta_{c}\) is the classification network parameters, \(\Theta_{\text{fe}}\) is the feature extraction network parameters, and \(\lambda_1, \lambda_2\) are regularization weights. Finally, \(w\) is a class-dependent weight for the localization task given by Eq (2.8),

\[
w = \begin{cases} 
0 & \text{background class} \\
1 & \text{any target class}
\end{cases}
\]

SPDRDL’s network description is in Table 2.2. Convolutional layers are followed by
ReLU activation and use initialization of [114]. Anywhere subsampling was used (which includes pooling layers and strided convolutions), anti-aliasing filtering was applied before subsampling using a $3 \times 3$ kernel of \[
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1
\end{bmatrix}
\].

### 2.4 Experiments

In this section, we describe how we measure the performance of SPDRDL and demonstrates its efficacy against contemporary methods. First, we will describe how we setup the experiments. Next, we describe the comparison methods. Finally, we show results by comparing all the methods.

#### 2.4.1 Setup

The ultimate goal of our experiments is to show the superiority of SPDRDL over existing methods. Equally important are two regimes to characterize. The first regime is classification performance of each object class. We show results in this regime by using confusion matrices whose purpose is to provide an overview of the classifier accuracy as a function of class. The second regime is classification performance in a one-versus-all scenario, whereby the target classes are collated into a single group. This collation converts our four class problem to a two class problem consisting of a target class and background class. Additionally, we will use a variation of this regime by showing performance of a particular target class versus all others.

As mentioned, we present results in a one-versus-all regime through conversion of a multi-class problem into a binary one. For binary class problems, many metrics exist by which to measure efficiency. A popular method is to measure area under the receiver operating characteristic curve (AUCROC). AUCROC reports the statistics of any chosen pair of samples being classified correctly. However, the method has been shown to be sensitive to class imbalance [115] which is pervasive here. Therefore, we choose area under the precision-recall (AUCPR) as our performance metric based on the analysis of [115] which determined that AUCPR is superior over AUCROC when the number of negative class samples greatly outnumbers the positive class samples, which is true here. Furthermore, [115] demonstrates that the stability of AUCPR over AUCROC meaning a performance curve dominating in precision-recall (PR) space also dominates in ROC space but not vice-versa.
As previously mentioned, we will use confusion matrices to measure per class accuracy. For the one-versus-all cases mentioned, we will use AUCPR. This metric has a benefit over a confusion matrix because it does not force us to specify a threshold as all thresholds are evaluated. Usually, a threshold is set to optimize for a specific performance metric which is context dependent. In lieu of having to select a particular context, we simply report AUCPR on a one-versus-all basis.

Sonar image fidelity is often a function of range. For example, spreading and absorption losses in the medium attenuate the reflected signals as a function of range. Therefore, the SNR of sonar echoes is reduced at long ranges. To measure the contribution of such effects on our classifier, we evaluate the classifier performance as a function of observation range.

Recalling that CNNs with strides or pooling are not translation invariant, we also evaluate translation performance. Ideally, translation invariance would be evaluated at every possible translation of the target but this becomes prohibitively expensive to compute. So, we evaluate translation invariance at eight extreme shifts of 59cm as shown in Figure 2.6. Good translation invariance will yield the same classification regardless of shift so we compute performance by measuring the standard deviation (stdev) of the output score as a function of these nine shifts (eight extreme shifts plus the center crop).

2.4.2 Dataset Description

We train and evaluate SPDRDL on the dataset of images which are output from the Mondrian detector of SAS imagery collected by the CMRE MUSCLE SAS sensor [20]. It is the same dataset used in [68] but with three modifications: (1) The original dataset contains detections on image boundaries and these images are extrapolated by mirroring resulting in target shapes which are not seen in the real environment. (2) Some of the images contained quadratic phase error (QPE) based on visual inspection [67]. We removed this error by applying a brute force autofocus in the k-space domain. (3) The images were dynamic range compressed using an algorithm based on the rational mapping function [14].

Overall, the dataset is composed of two partitions (dataset A and dataset B) based on collection year which Figure 2.5 depicts. Dataset A is composed of 27,748 images containing 1,385 targets collected from 2008 through 2013. Dataset B from a set of 21,181 images composed of 639 targets collected from 2013 through 2018. Each image in the dataset has resolution of 1.5cm and of size 335 × 335 pixels. However, translation is induced through cropping the images down to 256 × 256 pixels at training/inference.
Figure 2.5: The dataset we use for our experiment is split into two groups based on the collection year. Dataset A is used for the training set and Dataset B is used for the validation and tests sets.

Figure 2.6: The validation and test sets are derived from eight-way neighbor translations of 59cm. The original tile is from Dataset B and cropped. All nine croppings are used for the test set with a random crop used for the validation set shown by the boxed imaged. We show the test and validation data generation scheme for example images of the (a) background class and (b) target class.

The data is partitioned into three sections for evaluation purposes: (1) The training set is composed of Dataset A and augmentations of it. (2) The validation set is composed of a single translation of each image of Dataset B. (3) The test set is composed of nine translations of each image of Dataset B.

The translations applied to the validation and test set are from the set of shifts in the horizontal and vertical dimensions of the set \([-59\text{cm}, 0\text{cm}, 59\text{cm}].\) This configuration yields a total of nine possible shifts (including the center crop which is not shifted at all) for each image of the test set. The validation set is composed of one random crop from the set of nine available for each image. Figure 2.6 shows two examples of how the proposed cropping scheme contributes to the test and validation sets. In the examples, a
given image from Dataset B is eight-way shifted 59cm yielding a total of nine images (the mosaic) which are assigned to the test set and of one image (bounded by the solid line) which is randomly selected to be in the validation set. Overall, the training set is composed of 27,748 images, the validation set is composed of 21,181 images, and the test set is composed of 190,629 images.

2.4.3 SPDRDL Training Procedure

We train SPDRDL in a similar fashion to most other CNNs. We use a mini-batch size of sixteen of which we assign half the batch an image from the background class and half the batch an image from one of the three target classes. This 50:50 background-to-target-class split is based on the analysis of [116]. Recall, images from the dataset are 335 × 335 pixels. For training, a random crop of 256 × 256 pixels is selected for the mini-batch. The associated translational shift induced by cropping is recorded for the images containing a target. For each image in the mini-batch, the network estimates the class and translational shift (when the image is of a target class) and the errors are backpropagated appropriately. One epoch of training consists of the number of mini-batches required to see each image of the training set once on average.

CNNs perform best when lots of training data is available. In many situations though, large amounts of training data are not available. We call these instances low training data scenarios and in them, application of a domain prior becomes particularly important as its presence can significantly boost classification performance. To study this effect, we trained each of the methods on a random (but consistent across methods) 10% subset of the training. Figure 2.10 shows these results, and for convenience, the results when the full training data is available; recall these results are the AUCPRs of Figure 2.9. We show that the application of domain priors results in improved performance over all of the comparison methods when operating in low training data scenarios. These results demonstrate how application of our domain priors, image enhancement and target localization, improve performance on both abundant and low training data scenarios. The priors we introduce utilize information implicit during human interpretation and provide useful contextual information for our image classification method.

Deep networks often require a hyper-parameter search for optimal performance; SPDRDL is no different. SPDRDL uses the RMSProp optimization scheme [117] with a fixed learning rate of $10^{-4}$ which was determined through cross-validation. Furthermore, the weights of the domain priors in Eq (2.7) were also found through cross-validation giving the best results when $\lambda_1 = 10^{-6}$ and $\lambda_2 = 10^{-1}$.  

39
2.4.4 Ablation Study: Impact of Domain Priors

Table 2.3 shows the performance of SPDRDL with no additional multi-task losses and the incremental addition of each domain prior. For the low and high training scenarios, each additional loss provides improved classification performance with both priors giving better performance than each individual prior.

Next, we compare SPDRDL against three common despeckling techniques to show the benefit of using the learned enhancement network with the SSP. Table 2.4 shows the results for the high training scenario when we retrain the network by setting $\lambda_1 = 0$ in Eq (2.7) and supplanting the enhancement network with one of the following despeckling filters: Gaussian filter, median filter, and total variation [118,119]. We can see that the best AUCPR performance of the pre-processed despeckled images is 0.9451 which is not as good as when the SSP is present, 0.9538.

2.4.5 Comparison Against State of the Art

We demonstrate the efficacy of SPDRDL by comparing against three state of the art deep learning methods for sonar ATR and two recent shallow-learning methods. The deep learning methods were trained using Tensorflow 1.13.1 [9]. The number of trainable parameters for each network is given in Table 2.5. Development of the deep learning algorithms and their specific parameters follows.

Emigh, et al. (IOA SAS/SAR 2018) [91]. This network is based on the Resnet-18 architecture, does not rely on Imagenet pre-training, ingests dual-band SAS images, and is a binary classifier with output of a single scalar indicating target/non-target score. We modify the network to input a single-band SAS image and output four classes by using a softmax function after the last dense layer. We use the categorical cross-entropy loss when training the network as binary cross-entropy loss was originally specified by the authors and we adapted it to this multi-class scenario. We trained using the Adam optimizer [120] with a learning rate of $10^{-3}$. The paper mentions decreasing the learning rate when a loss plateau occurs but does not give details on its parameters. In lieu of this, we forgo the learning rate schedule and invoke the same early stopping rule used during SPDRDL training.

Galusha, et al. (SPIE 2019) [92]. This network has only a few layers and is an Alexnet-like architecture. Like Emigh, et al., this network originally consumed dual-band SAS imagery and output a binary classification score representing target/non-target. We modify the network in a similar fashion as Emigh, et al. by using only a single-band
SAS image as input and modify the output to support classification of four classes using a softmax activation after the last dense layer. As in Emigh, et al., we use the categorical cross-entropy loss when training the network as the binary cross-entropy loss was originally specified by the authors. We train the network in the same fashion the authors used in their work: stochastic gradient descent for 2,000 epochs at a learning rate of $10^{-3}$.

**DenseNet121 (CVPR 2017)** [96]. This is a common state-of-the-art off-the-shelf DenseNet architecture with 121 layers pre-trained on the Imagnet dataset. We choose this network for comparison because it is the feature extraction network used in SPDRDL, and serves as a baseline to provide evidence demonstrating our proposed priors improve classification performance. As with SPDRDL, we use the focal loss instead of the cross-entropy loss in order to demonstrate that our performance gains are not simply from this different classification loss. DenseNet121 ingests three-channel color imagery; we simply replicate the SAS input image over two additional channels to arrive at a three-channel image. Finally, we use the global average pooling option after the feature extraction layer and apply a four class output with softmax. We train using the same procedure as SPDRDL.

**BoW-HOG (CVPR 2015).** This method is a bag-of-words using a histogram of oriented gradients features inspired by [79]. A comparison to a similar algorithm was also made by [80]. For this approach, each image is divided into $16 \times 16$ pixel tiles of which the HOG features are computed. These features are then clustered using mini-batch k-means clustering [121] into what are known as words in this setup. The clusters of words from the vocabulary for the bag-of-words model.

The size of the vocabulary, $k$, and the regularization parameters for the SVM, $C$, are chosen using a random search [8, 122] of fifty iterations. The hyper-parameters for the search are chosen from $C \sim 10^{U[-5,10]}$ and $k \sim [U[10,500]]$ where $U$ represents the uniform distribution over the specified interval. The hyper-parameter search returned best results for $c = 4.36, k = 402$. A radial basis function kernel was used. BoW-HOG method is costly to compute so we use no data augmentation during training and measure performance solely on the center crops of Dataset B.

**DSRC (IEEE TGRS 2017).** This method is a dictionary sparse reconstruction (DSRC) algorithm inspired by [80]. We use mini-batch dictionary learning [123] with coordinate descent to learn the dictionary atoms.

At test time, the learned dictionary for each class is used to reconstruct the test images one class at a time. The inverse of these errors from each reconstruction is
transformed by the softmax function to class probabilities.

The sparsity of the $L_1$ reconstruction loss, $\alpha$, and the number of dictionary atoms per class, $k$, are chosen using a random search of fifty iterations. The hyper-parameters for the search are chosen from $\alpha \sim 10^{U[-5.6]}$ and $k \sim [U[10, 250]]$ and the best results were $\alpha = 10^{-2.02}$, $k = 147$. A mini-batch size of ten was used for dictionary learning.

Similar to the BoW-HOG method, the extensive compute resources necessitated by this algorithm lead to using only center crops of the images of the training and validation sets. As with the BoW-HOG method, performance is only reported on the validation set.

### 2.4.6 Results and Analysis

In this section, we present results comparing SPDRDL against the several contemporary methods, demonstrate the necessity of each prior in our loss function formulation, analyze properties of the learned image enhancement function, and demonstrate the ability to reduce the network size considerably for use in low power embedded systems while maintaining good classification performance.

#### 2.4.6.1 Classification Task

We show confusion matrices in Figure 2.7, precision recall curves for each target class versus background in Figure 2.8, and precision recall curves for target versus background in Figure 2.9. We demonstrate the learning efficiency of our method by showing results from the ablation study. From these figures, we can see the benefits our domain priors afford us. In almost all metrics, SPDRDL outperforms existing methods.

Indeed, the results of Figure 2.10 show SPDRDL performance very similar to Densenet121 in the low training data scenario with SPDRDL exhibiting a slight performance gain. However, the gap between SPDRDL and Densenet121 may be much larger in reality. In this case, it is quite likely we are seeing the effects of selection bias of the training data subset. To examine issues of sample selection bias, we train the top three methods ten times each using a random subset of 10% of the training data; the same subsets were used for each algorithm evaluation. Evaluating the test set on all of these trained models would be computationally prohibitive due to the large test set size. Therefore, we report on the validation set performance for the best epoch of each. As shown in Figure 2.11, the performance gap between SPDRDL and the next best method, Densenet121, becomes much more distinct when viewing the results in the context of selection bias.
Figure 2.7: Confusion matrices for SPDRDL and comparison methods. Larger numbers along the diagonal indicate better performance. We see the benefits of the added domain priors through improved performance, especially of the wedge class.

Figure 2.8: Precision-recall curves for target type: (a) cylinder, (b) wedge, (c) truncated cone. AUCPR in parentheses; larger values are better.

We also examined the performance gain of each regularization term of Eq (2.7) to demonstrate its necessity. This was done by setting each loss term to zero and then retraining and reevaluating the network. Consequently, we can see both priors of Eq (2.7) improve classification performance even in the low training data scenario demonstrating the benefits of incorporating domain knowledge.

We can see from the results that the SPDRDL method outperforms the deep learning
methods and the shallow learning methods by a significant amount demonstrating the usefulness of the domain priors. For our ablation analysis, we focus on two aspects: (1) Generalization efficacy using in a low training data scenario where only 10% of the training data is used. (2) Necessity of the additional loss terms from our domain priors used in Eq (2.7).

Figure 2.10 shows AUCPR for all methods using 10% and 100% of the training data; SPDRDL outperforms all the comparison methods showing the efficacy of the additional domain priors.

Finally, we examine the two-class performance as a function of object range from sensor. Figure 2.13 shows SPDRDL performing well over an extensive range from the sonar due to the addition of the domain priors. Especially at the nearest and farthest ranges from the SAS, SPDRDL outperforms the comparison methods.
2.4.6.2 Image Enhancement Task

Because of speckle noise, image despeckling algorithms are often employed to enhance SAS images for the purpose of improving human interpretability. We posit that there exists an image enhancement function which improves classification performance and we call this the structural similarity prior; any enhancement algorithm must preserve the structural similarity between the input and output images. We employ this prior in SPDRDL through the use of a learned image enhancement transform which is constrained by the loss of Eq (2.3).

We analyze the output of the enhancement network by examining its frequency response and compare it to the frequency response of the original images. That is, we compute the frequency response of all the images in target classes and present their averaged spectra; all images are windowed with a 2D hamming window prior to frequency domain conversion. Figure 2.14 depicts these results. We can see selective suppression in
Figure 2.11: To understand the sensitivity of selection bias in a low training data scenario, we trained the top three methods each ten times with a random sample of 10% of the training data and plot the results. Due to the large run-time associated with evaluating all the test set imagery, we report results on the validation set only. Larger AUCPR indicates better performance. As shown, we can clearly see a benefit in performance with the addition of the domain priors over the next best method, Densetnet121.

both spatial frequency and orientation from the integration of our image enhancement network. This behavior is in contrast to what one would get with a simple 2D Gaussian filter which would give an isotropic frequency response.

2.4.6.3 Target Localization Task and Translation Equivariance

CNNs lose their translation equivariance through the addition of non-unitary strided pooling and convolution layers. In this section, we quantify this phenomenon over the deep learning methods to demonstrate the necessity of adding anti-aliasing filters before subsampling in the network. To first illustrate the problem, we show classification scores of an image over a large set of translations, Figure 2.12. As shown, classification scores can vary drastically even at small pixel shifts. For example, suppose a well-centered target is presented to the classifier and is correctly classified. By translating the image by one pixel, the classifier will now misclassify it!

We analyze the translation invariance performance of each deep learning algorithm to assess its ability in providing translation invariance. We only examined the target/background scenario by converting the class prediction estimates to a single scalar from \([0, 1]\) indicating target score. For a single image, we compute its scores for a center crop and eight extreme crops. We then measure the standard deviation of these scores and call this metric, \(\Psi\), the image’s shift invariance score. More formally, we define the shift invariance score as

\[
\Psi(f, x) = \text{stdev}( \{ f(x_{-h,-w}), f(x_{-h,0}), f(x_{-h,w}), f(x_{0,-w}), f(x_{0,0}), f(x_{0,w}), f(x_{h,-w}), f(x_{h,0}), f(x_{h,w}) \} ) \tag{2.9}
\]
(c) Image under test for (a) and (b) which contains a target-class object.

Figure 2.12: (a) and (b) Target scores (SPDRDL/Densenet121 predicted classification probability) as a function of diagonal image translation of the top two performing algorithms for a sample input image. We can see for the Densenet121 method that a small translation of the input image results in an unpredictable classification whereas SPDRDL does not exhibit this.

(c) Image under test for (a) and (b).

where function $f$ is the inference model, $x_{a,b}$ is the input image translated by $a$ and $b$, and $h = w = 59$cm. As a result of this formulation, lower shift invariance scores indicate better translation robustness. We report the results in Table 2.6. We can see our proposed method has the greatest translation invariance due to the addition of the anti-aliasing filters layers employing non-unitary stride.

2.4.6.4 Network Compute Burden and Reduction

Deep networks consisting of a large number of parameters can be challenging to deploy on embedded hardware because of the large memory and computational footprint required. However, we can reduce the number of parameters used during inference by utilizing
Figure 2.13: Top panel, AUCPR as a function of range from the sonar for all methods. Larger values indicated better performance. Bottom panel is a zoom of the top panel highlighting the differences of the top three methods: SPDRDL, DenseNet121, and Emigh, et al.

a network pruning algorithm. Although such algorithms remove weights, classifier performance is often maintained and in some cases, even improved.

We reduce the number of free parameters of the network using the pruning method of [124]. In this method, we sort the absolute value of the network weights and set the lowest proportion of weights to zero. Figure 2.15 illustrates the results of performing this operation on SPDRDL. We can see that even when the number of parameters is reduced by half, the network still results in competitive classification performance.
Figure 2.14: We analyze the frequency spectrum of the original images input to the network (a) and of our learned image enhancement (b). In panel (b), we can see selective attenuation across spatial frequencies and orientations due to the use of the data-adaptive image enhancement prior.

Figure 2.15: AUCPR as a function of weight pruning proportion. Weights are sorted by magnitude and removed starting with the lowest magnitude weights first. Significant pruning is accomplished while still maintaining good classification performance.
Table 2.2: Description of SPDRDL architecture network architecture. The network input is a 256 × 256 pixel grayscale SAS image normalized to [0, 1]. The network has two outputs, a classification output and a target position output. AA indicates anti-aliasing was applied to the layer.

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Layer Function</th>
<th>Dimensions</th>
<th># Filters</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>input1</td>
<td>Input</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>conv1a</td>
<td>Convolution</td>
<td>3x3</td>
<td>16</td>
<td>input1</td>
</tr>
<tr>
<td>conv1b</td>
<td>Convolution</td>
<td>3x3</td>
<td>16</td>
<td>conv1a</td>
</tr>
<tr>
<td>pool1</td>
<td>AA Max Pooling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv1b</td>
</tr>
<tr>
<td>conv2a</td>
<td>Convolution</td>
<td>3x3</td>
<td>32</td>
<td>pool1</td>
</tr>
<tr>
<td>conv2b</td>
<td>Convolution</td>
<td>3x3</td>
<td>32</td>
<td>conv2a</td>
</tr>
<tr>
<td>pool2</td>
<td>AA Max Pooling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv2b</td>
</tr>
<tr>
<td>conv3a</td>
<td>Convolution</td>
<td>3x3</td>
<td>64</td>
<td>conv2b</td>
</tr>
<tr>
<td>conv3b</td>
<td>Convolution</td>
<td>3x3</td>
<td>64</td>
<td>conv3a</td>
</tr>
<tr>
<td>pool3</td>
<td>AA Max Pooling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv3b</td>
</tr>
<tr>
<td>conv4a</td>
<td>Convolution</td>
<td>3x3</td>
<td>128</td>
<td>pool3</td>
</tr>
<tr>
<td>conv4b</td>
<td>Convolution</td>
<td>3x3</td>
<td>128</td>
<td>conv4a</td>
</tr>
<tr>
<td>pool4</td>
<td>AA Max Pooling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv4b</td>
</tr>
<tr>
<td>conv5a</td>
<td>Convolution</td>
<td>3x3</td>
<td>256</td>
<td>pool4</td>
</tr>
<tr>
<td>conv5b</td>
<td>Convolution</td>
<td>3x3</td>
<td>256</td>
<td>conv5a</td>
</tr>
<tr>
<td>up1</td>
<td>Upsampling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv5b</td>
</tr>
<tr>
<td>conv6a</td>
<td>Convolution</td>
<td>3x3</td>
<td>128</td>
<td>up1</td>
</tr>
<tr>
<td>merge1</td>
<td>Concatenate</td>
<td>N/A</td>
<td>N/A</td>
<td>conv6a, conv4b</td>
</tr>
<tr>
<td>conv6b</td>
<td>Convolution</td>
<td>3x3</td>
<td>128</td>
<td>merge1</td>
</tr>
<tr>
<td>conv6c</td>
<td>Convolution</td>
<td>3x3</td>
<td>128</td>
<td>conv6b</td>
</tr>
<tr>
<td>up2</td>
<td>Upsampling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv6c</td>
</tr>
<tr>
<td>conv7a</td>
<td>Convolution</td>
<td>3x3</td>
<td>64</td>
<td>up2</td>
</tr>
<tr>
<td>merge2</td>
<td>Concatenate</td>
<td>N/A</td>
<td>N/A</td>
<td>conv7a, conv3b</td>
</tr>
<tr>
<td>conv7b</td>
<td>Convolution</td>
<td>3x3</td>
<td>64</td>
<td>merge2</td>
</tr>
<tr>
<td>conv7c</td>
<td>Convolution</td>
<td>3x3</td>
<td>64</td>
<td>conv7b</td>
</tr>
<tr>
<td>up3</td>
<td>Upsampling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv7c</td>
</tr>
<tr>
<td>conv8a</td>
<td>Convolution</td>
<td>3x3</td>
<td>32</td>
<td>up3</td>
</tr>
<tr>
<td>merge3</td>
<td>Concatenate</td>
<td>N/A</td>
<td>N/A</td>
<td>conv8a, conv2b</td>
</tr>
<tr>
<td>conv8b</td>
<td>Convolution</td>
<td>3x3</td>
<td>32</td>
<td>merge3</td>
</tr>
<tr>
<td>conv8c</td>
<td>Convolution</td>
<td>3x3</td>
<td>32</td>
<td>conv8b</td>
</tr>
<tr>
<td>up4</td>
<td>Upsampling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv8c</td>
</tr>
<tr>
<td>conv9a</td>
<td>Convolution</td>
<td>3x3</td>
<td>16</td>
<td>up4</td>
</tr>
<tr>
<td>merge4</td>
<td>Concatenate</td>
<td>N/A</td>
<td>N/A</td>
<td>conv9a, conv1b</td>
</tr>
<tr>
<td>conv9b</td>
<td>Convolution</td>
<td>3x3</td>
<td>16</td>
<td>merge4</td>
</tr>
<tr>
<td>conv9c</td>
<td>Convolution</td>
<td>3x3</td>
<td>16</td>
<td>conv9b</td>
</tr>
<tr>
<td>conv9d</td>
<td>Convolution</td>
<td>3x3</td>
<td>2</td>
<td>conv9c</td>
</tr>
<tr>
<td>conv9e</td>
<td>Convolution</td>
<td>1x1</td>
<td>1</td>
<td>conv9d</td>
</tr>
<tr>
<td>lambda1</td>
<td>$\frac{\text{in}<em>{\text{min}}(\text{in}) - \text{in}</em>{\text{min}}(\text{out})}{\text{max}(\text{out}) - \text{min}(\text{out})}$</td>
<td>N/A</td>
<td>N/A</td>
<td>conv9e</td>
</tr>
<tr>
<td>densenet1</td>
<td>AA DenseNet121</td>
<td>N/A</td>
<td>N/A</td>
<td>lambda1</td>
</tr>
<tr>
<td>gap1</td>
<td>Global Average Pooling</td>
<td>N/A</td>
<td>N/A</td>
<td>densenet1</td>
</tr>
<tr>
<td>classification</td>
<td>Dense with softmax</td>
<td>4</td>
<td>N/A</td>
<td>gap1</td>
</tr>
<tr>
<td>conv10</td>
<td>Convolution</td>
<td>1x1</td>
<td>256</td>
<td>densenet1</td>
</tr>
<tr>
<td>conv11</td>
<td>Convolution</td>
<td>1x1</td>
<td>128</td>
<td>conv10</td>
</tr>
<tr>
<td>conv12</td>
<td>Convolution</td>
<td>1x1</td>
<td>64</td>
<td>conv11</td>
</tr>
<tr>
<td>flatten1</td>
<td>Flatten layer</td>
<td>N/A</td>
<td>N/A</td>
<td>conv12</td>
</tr>
<tr>
<td>xPosEstimate</td>
<td>Dense</td>
<td>1</td>
<td>N/A</td>
<td>flatten1</td>
</tr>
<tr>
<td>yPosEstimate</td>
<td>Dense</td>
<td>1</td>
<td>N/A</td>
<td>flatten1</td>
</tr>
</tbody>
</table>
Table 2.3: We evaluate performance of each domain prior in our loss function, Eq (2.7), to demonstrate their utility. AUCPR is reported on the test set for the high (100% of training data available) and low (10% of training data available) training data scenarios. Note, the enhancement network is still present in the CL and CL+SSCP scenarios but the associated regularization loss is removed from the objective function.

<table>
<thead>
<tr>
<th>Domain Priors</th>
<th>10%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>None, Only Classification Loss (CL)</td>
<td>0.8742</td>
<td>0.9281</td>
</tr>
<tr>
<td>CL + Structural Scene Context Prior (SSCP)</td>
<td>0.8919</td>
<td>0.9503</td>
</tr>
<tr>
<td>CL + Structural Similarity Prior (SSP)</td>
<td>0.8969</td>
<td>0.9456</td>
</tr>
<tr>
<td>Both, CL + SSCP + SSP</td>
<td><strong>0.9079</strong></td>
<td><strong>0.9538</strong></td>
</tr>
</tbody>
</table>

Table 2.4: We evaluate the performance of our SSP domain prior against several off-the-shelf despeckling methods to show SSP's utility. We do this by retraining the network but removing SSP's associated loss term in Eq (2.7) and the Enhancement Network in Figure 2.3 (SSCP is still included). We feed to network three types of despeckled imagery and report AUCPR using 100% of the available training data. We see that despeckling does give some performance gains over the CL configuration of Table 2.3, but not as much as when the SSP is active (CL+SSP and CL+SSP+SSCP configurations of Table 2.3). Recall, our method (CL+SSCP+SSP) yields an AUCPR of 0.9538 as shown in Table 2.3.

<table>
<thead>
<tr>
<th>Despeckling Algorithm</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL + SSCP + Gaussian Filter</td>
<td>0.9450</td>
</tr>
<tr>
<td>CL + SSCP + Median Filter</td>
<td>0.9409</td>
</tr>
<tr>
<td>CL + SSCP + Total Variation</td>
<td>0.9451</td>
</tr>
</tbody>
</table>

Table 2.5: Number of trainable parameters for the deep learning methods.

<table>
<thead>
<tr>
<th>Network</th>
<th>Number of Trainable Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPDRDL</td>
<td>$\approx 9.16 \times 10^6$</td>
</tr>
<tr>
<td>Densenet121</td>
<td>$\approx 6.95 \times 10^6$</td>
</tr>
<tr>
<td>Emigh, et al.</td>
<td>$\approx 11.2 \times 10^6$</td>
</tr>
<tr>
<td>Galusha, et al.</td>
<td>$\approx 243 \times 10^6$</td>
</tr>
</tbody>
</table>

Table 2.6: Classifier scores were compiled for all nine shifts of each image in the test set. Next, we compute the standard deviation of scores for each image. Finally, the table shows the mean of those scores over the entire test set; lower number indicate more shift invariance (i.e. better performance). We see that SPDRDL has the best translation invariance of all the methods. We hypothesize that Densenet121 has the second best performance because it contains only a single instance of MaxPooling and has many levels of feature averaging.

<table>
<thead>
<tr>
<th>Network</th>
<th>Shift Invariance Score (Lower is Better)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPDRDL</td>
<td>$5.41 \times 10^{-4}$</td>
</tr>
<tr>
<td>Densenet121</td>
<td>$5.82 \times 10^{-4}$</td>
</tr>
<tr>
<td>Emigh, et al.</td>
<td>$2.17 \times 10^{-3}$</td>
</tr>
<tr>
<td>Galusha, et al.</td>
<td>$3.94 \times 10^{-2}$</td>
</tr>
</tbody>
</table>
Chapter 3  
Contribution II: Deep Multi-Look Sequence Processing for Synthetic Aperture Sonar Image Segmentation

3.1 Introduction

Underwater sensing modalities like SAS and side-scan-sonar (SSS) are important for seafloor situational awareness. Furthermore, SAS image segmentation provides the ability to spatially characterize imagery having an important role in habitat and environmental monitoring [46–51], providing context for other automated algorithms like target recognition [52–56], and dynamically modifying vehicle behavior in order to maximize survey objectives [21–25].

3.1.1 Open Challenges in SAS Image Segmentation

Obtaining labeled training data has been a consistent hurdle for SAS [29], especially for the problem of SAS image segmentation. Many SAS image segmentation methods operate in an unsupervised fashion in order to make research progress despite this hurdle; we review the literature in Section 3.2. However, current methods easily confuse classes resulting in mediocre performance on real-world datasets or presenting good efficacy on only one or two classes.

Obtaining accurately labeled imagery for SAS image segmentation necessitates the need for divers and oceanographers to accurately characterize the imagery post-survey. This task is quite burdensome as coordinating divers to manually survey an area is time-consuming and logistically challenging. To mitigate this, researchers generally create labeled data post-survey using human experts who are accustomed to analyzing
Figure 3.1: Example SAS image (left) with its weakly-labeled segmentation truth map (right). In the right image, the blue and brown areas indicate human-provided labels of two seafloor classes while the black area represents unlabeled pixels. The border of the labeled regions is shown in the left image in white. We see the human only labeled some of the “easy” portions of the image containing mostly unambiguous seafloor texture leaving the remaining pixels unlabeled.

the imagery. However, this method presents its own challenges. Disambiguation of some seafloor textures (e.g. ripples [52]) requires imagery from multiple seafloor-to-sonar viewpoints since the same area of seafloor may appear different in a SAS image depending on the collection geometry.

Despite the aforementioned issues with obtaining labeled SAS imagery, we are given a weakly-labeled dataset where the data are labeled post-survey by a human, but only some of the “easy” portions of the image are labeled leaving the majority of the pixels unlabeled. Figure 3.1 shows an example weakly-labeled image from our dataset. Consequently, abundant SAS imagery may be collected during a survey, but due to the difficulty of the labeling process, labeled imagery remains scarce presenting a challenge to deep learning methods which usually require abundant labeled training data for success.

3.1.2 Overview of the Proposal

In this work, we utilize the fact that SAS imagery filtered intelligently in the $k$-space domain yields images with different “squints” or “looks” of the seafloor by steering the receive beam of the aperture. Sweeping over a consecutive set of look-angles creates a sequence of images akin to a movie calling attention to aspect-dependent scattering
Figure 3.2: Seafloor classes considered in this work using data from [125]. For each image pair (i.e. column), the image on the top is the original SAS image and the image on the bottom highlights one of the corresponding seafloor classes present in the top image deemed “easy” to label by a human. In viewing (a) and (b) we see classes like dark sand and shadow look similar in magnitude imagery and thus are easily confused. However, we resolve this ambiguity by examining angle-of-arrival information through multi-look processing employed by our proposed network architecture, MLSP-Net.

Figure 3.3: Example MLSP sequence showing benefits of $k$-space filtering a SAS image. (Top) Image squint sequence derived from SLC image (right-most column). (Bottom) Corresponding $k$-space filter with vertical axis approximately proportional to look-angle, $\theta$, and horizontal axis approximately proportional to frequency. In the top right quadrant of the images of the top row, there is an arrow above each image pointing to a “fuzz” moving vertically in the image sequence. This is likely biologics in the water column moving during the collection of the synthetic aperture. The $k$-space filtering produces a pseudo-motion effect across frames of the image sequence capturing this movement. This phenomenon is not apparent when $k$-space filtering is disabled (see right-most column, “No Filtering” for this image) resulting in seafloor occlusion or false seafloor texture.

effects and motion of objects in the receive beam. Organizing such a process is the result of domain knowledge of the SAS imaging modality and is sometimes called multi-look sequence processing (MLSP) from where our proposed network derives its name.
3.2 Previous Works

Seafloor segmentation for SAS has been studied over the last decade as more SAS systems come online and the need to characterize wide swaths of ocean floor increased. Generally speaking, SAS segmentation schemes are grouped into two high-level categories: (1) seafloor type segmentation, and (2) target/background segmentation. Regarding the latter, only a few works exist so we will quickly cover it first.

In the target/background segmentation scheme, one seeks to identify specific spatial locations of targets by precisely identifying the pixels belonging to the target, its shadow, and the surrounding seafloor. In one prior work [126], the authors use elastic shape models to outline the target from the background by using known target shape priors. The result of their algorithm is a precise outline of the target object in the scene. In another previous work [127], the authors use the min-cut/max-flow algorithm to distinguish areas of target highlight, target shadow, and background. In [128], the authors map neighborhood pixel statistics to the mean and standard deviation plane and then use this representation to localize target echoes from the background in a SAS image; their work is elaborated upon in [129].

The remaining algorithms we cover in this review propose solutions to segment the seabed into categories corresponding to environmentally driven textures such as rock, large/small sand ripples, and seagrass. The methods presented in the literature are largely unsupervised. In [57], the authors extract local spatial features from an image using a filter bank, use these features to form and merge superpixels, and then cluster the superpixels assuming a Dirichlet process. They show good results on a hand-labeled real-world SAS dataset. In [58], the authors propose a method extracting features from an image and cluster these features using $k$-means. The features extracted are derived from pixel-driven autocorrelation functions and neighborhood $k$-shape parameter. The authors demonstrate good results against a four-class hand-labeled real-world SAS dataset. This work was extended in [59] and compared against two popular filter banks, Haralick and Wavelet features. Their results show improved performance against the two comparison methods. In [60], the authors use a variety of features specific to SSS imagery with special focus on pruning features (in this case textons) to create a universal feature set used to spatially characterize all the images. Superpixel quantization is then applied to the texton features and Dirichlet process clustering is performed to arrive at the final pixel-level class assignments. They show good performance on a seven-class hand-labeled real-world SAS dataset. In [61], the author extracts features using
coefficients from a wavelet decomposition of local neighborhoods. Spectral clustering is then applied to the feature vectors and class labels are assigned. Limited results on real-world SAS dataset are presented. In [62], the author uses pixel-level neighborhoods of mean-normalized variance (which as been linked to the sonar-specific physical property of k-shape parameter) to demonstrate this feature’s seafloor characterization power. Further, the author demonstrates the metric’s ability to predict object recognition performance. In [52], the authors use a model of sand ripples in order to forgo the need for any training data for their segmentation. They compose a filter bank modeling shadow-highlight features of ripples and this is done over different lengths and scales. Next, they propose a “ripplicity” metric which is computed from the convolved response of the filter bank upon the SAS images. The maximum of the responses is used as a ripple indicator for classification. Finally, the result is thresholded and an area filter is run on the result to remove small or spurious labelings. The authors demonstrate good results on a real-world SAS dataset. In [63], the authors observe class membership for SAS images is often fuzzy and develop a novel method accounting for this. Additionally, they acknowledge images may contain noise or artifacts from the imaging process which are viewed as outliers and their proposed technique accounts for these also. The authors propose a soft segmentation technique to address the aforementioned concerns. Specifically, they perform soft segmentation using the fuzzy local information C-Means (FLICM) objective with an additional objective augmented to the optimization procedure to mitigate outliers which is derived from the possibilistic and possibilistic-fuzzy clustering algorithms (hence their proposed algorithm name, PFLICM). The authors use features from Sobel edge histogram descriptors, histogram of oriented gradients (HoG), and local binary patterns (LBP). The authors demonstrate their method on real-world SAS imagery consisting of a variety of seafloor environments. In [64], this work is extended by incorporating a feature selection algorithm to only keep features that result in optimal performance. This is accomplished by ranking features using the Xie-Beni (XB) cluster validity measure. Features are ranked by this measure and then pruned accordingly using a forward feature selection process. A wide variety of features are considered including: Sobel, HoG, LBP, mean, variance, shape, Haralick, Gabor, Gaussian, lacunarity, and Laplacian of Gaussian (LoG). Qualitative clustering results on a real-world SAS dataset are presented. In [65], the authors use hand-crafted features to train a support vector machine (SVM) and compare their results to a small CNN. In [66], the authors use a Bayesian framework with a Gaussian process classification algorithm giving seafloor class and uncertainty.

Recently, [130,131] demonstrate a SAS segmentation technique combining the SLIC
Figure 3.4: Our proposed network called multi-look sequence processing network (MLSP-Net) for SAS segmentation is composed of two primary paths: Multi-Look Sequence and Static Image. The Multi-Look Sequence path is composed of eight Filter-Modules operating in k-space and are initialized using domain knowledge so an initial useful multi-look sequence emerges (see Figure 3.3). The eight looks are sent through a shared U-Net encoder, re-weighted by squeeze and excitation modules, and then fed to a bidirectional 2D convolutional LSTM to extract meaningful features from the image sequence. The Static Image path simply ingests the input image and processes it through a U-Net. Features from both paths are then concatenated in the feature dimension and processed by a segmentation network for Final Image Segmentation. We show the utility of the MLSP by doing an ablation study whereby we initialize and fix the Filter-Modules to unity thereby removing their effect. Now, all eight paths have the same non-filtered input image resulting in the same feature map for each step of the LSTM. Results in Table 3.5 show the benefit of the MLSP than without it. Circular taps in the figure show representative results; diamond taps show data type (real R or complex C) and tensor shape.

[132] superpixel method with deep learning in an iterative framework. The method works solely on the magnitude SAS image and is conceptually different than the method we propose here. Our method exploits the complex-valued SAS image using a SAS-specific insight, MLSP.

Our proposed method uses beam steering as it’s a key component of MLSP. We acknowledge the idea of enhancing SAS image interpretation through beam steering and MLSP is not new but, as far as the authors are aware, has never been applied explicitly to the problem of SAS segmentation. The beam-steering technique for SAS has been used for: input to target recognition algorithms [133,134], coherent change detection [135], speckle reduction [136], and image visualization [137]. However, our approach is novel in that: (1) we apply beam steering to generate multi-look sequences for the purpose of SAS segmentation, (2) our approach uses several multi-look frames whereas existing multi-look methods for SAS utilize only a few, and (3) we specifically model the 2D temporal structure of the image sequence generated by the MLSP.
3.3 Multi-Look Sequence Processing Network for SAS Image Segmentation (MLSP-Net)

3.3.1 Background

The output of SAS image reconstruction is an image containing complex-valued pixels called a single-look complex (SLC) image which encodes angle-of-arrival (AoA) of the scattered pulse in addition to its magnitude and phase. The 2D Fourier transform of the SLC yields a \( k \)-space representation with axes \( k_x \) and \( k_y \) for spatial dimensions of \( x \) and \( y \) for range and along-track respectively [69]. Once in \( k \)-space, interpolation is done to arrive at \( f, \theta \) space where \( f \) is frequency and \( \theta \) is look angle or AoA. For sufficiently low fractional bandwidth systems (i.e. \( f_s \), the sampling rate of the system, is much smaller than \( f_0 \), the operating center frequency of the system), the angular dependence on waveform arrival becomes independent of frequency. One can think of this in terms of the Fourier slice theorem [12, 138] when the ratio \( f_0/f_s \) is large enough, the arc formed by a small continuous section of look angles is locally linear (i.e. a small arc of a circle becomes linear as the circle diameter goes to infinity) resulting in lines of constant angle to be nearly horizontal (see Figure 3.6). Using this assumption, \( k_y \) is now approximately proportional to \( \theta \), the look angle. With this representation, band-pass filtering in \( \theta \) across all frequencies and then performing an inverse 2D Fourier transform results in an SLC whereby the array is “squinted” and directed to a subset of look angles in \( \theta \); the beam geometry is shown in Figure 3.5. Squinting the receiver beam in this manner results in reduced azimuthal spatial resolution but at the benefit of disambiguating the AoA of the acoustic returns.

When we filter with a fixed beamwidth over consecutive angle offsets (Figure 3.3, bottom), we arrive at a sequence of images useful for image interpretation (Figure 3.3, top). Despite the reduced azimuthal resolution, we now discern aspect-dependent features such as those arising from acoustic shadows. In addition, we view the beam steering effect as a temporal filter whereby visible changes image-to-image in the sequence are the result of motion from objects (like fish or seagrass) during the formation of the synthetic aperture.
3.3.2 Justification of $k_y$ as Angle-of-Arrival Approximation

To properly filter the AoA’s in $k$-space we must know the sonar center frequency. It is not uncommon to be given SAS imagery with the sonar center frequency withheld by the manufacturer for proprietary reasons; this is the case for the data used in this work. Without the center frequency, we cannot exactly know the shape of the appropriate filter in $\theta$ (see Figure 3.6) but for systems with a sufficiently small fractional bandwidth (commonly referred to as high-frequency (HF) SAS systems), we can approximate the shape of the AoA filters which is what we do here. This section provides the mathematical model for this justification.

We begin by defining the $k$-space of an SLC as,

$$K = \mathcal{F}^2\{x\}$$

(3.1)

where $\mathcal{F}^2$ is the 2D Fourier transform of the input SLC, $x$. $K$ has axes of $k_x, k_y$. The $k$-space transform for a stripmap image (which is the imaging geometry of our data) is interpreted as roughly organizing incoming acoustic waves by AoA in the $k_y$ dimension. Figure 3.6 depicts this intuition for the sample SAS given by [139] (recall we are not given the sonar center frequency hence our approximation). We see in the figure lines of constant AoA run approximately horizontally in $k$-space. If the sonar center frequency is known, we can perform a simple non-linear transform to go from $k$-space to $\theta - f$ space. The figure shows $k$-space with a colored overlay of isolines representing constant AoA.
with those of our approximation shown as the grey horizontal lines.

Using the geometry of the $k$-space representation, we approximate AoA as a scalar multiple $\alpha$ and $k_y$. Let’s begin with the definition of $\theta$ from [69],

$$\theta = \arctan \left( \frac{k_y}{2k_0 + k_x} \right) \quad (3.2)$$

where $k_0$ is the carrier wavenumber (i.e. $\frac{2\pi f_0}{c}$, $f_0$ is the carrier frequency of the system, and $c$ is the speed of sound in the medium). We seek to arrive at the form $\theta \approx \alpha k_y$ giving

$$\frac{\arctan \left( \frac{k_y}{2k_0 + k_x} \right)}{k_y} \approx \alpha \quad (3.3)$$

and when $k_0$ is sufficiently large, the argument of the arctan is small giving us via $\arctan(x) \approx x$ since $x$ is small,

$$\frac{k_y}{2k_0 + k_x} \approx \alpha \quad (3.4)$$

$$\frac{1}{2k_0 + k_x} \approx \alpha \quad (3.5)$$

giving us our linear approximation,

$$\theta \approx \left( \frac{1}{2k_0 + k_x} \right) k_y \quad (3.6)$$

to what the fractional bandwidth of the system is small enough.

Furthermore, we quantify the error of our approximation as a function of fractional bandwidth. If we are given a bin in $k_y$, we define the AoA error, $\xi$, as the difference between AoA at $f_{\text{min}}$ and $f_{\text{max}}$. For a given image resolution ($\Delta_x, \Delta_y$), $f_0$, and speed of sound ($c$), this error is defined as,

$$\xi = \theta_{\text{max}} - \theta_{\text{min}} =$$

$$\arctan \left( \frac{\pi}{2k_0 - \frac{\pi}{\Delta_x}} \right) - \arctan \left( \frac{\pi}{2k_0 + \frac{\pi}{\Delta_x}} \right) \quad (3.7)$$

where $\xi$ is the worst case AoA error for the given system parameters, and $k_0$ is the reference in $k$-space and defined as $k_0 = 2\pi \frac{f_0}{c}$. We see from Equation 3.7, as $k_0$ goes to infinity, the argument of the arctan goes to zeros thus making Equation 3.7 go to zero. Figure 3.7 shows $\xi$ over a variety of fractional bandwidths, and in conjunction
Figure 3.6: A depiction of the differences between true AoA with that of our approximation made in $k$-space for the SAS system of [139] (see 3.1) at 0.015m resolution. The picture overlays colored iso-lines of true constant AoA (horizontal lines) and frequency (vertical lines) onto $k$-space. The grey-colored grid attached to the axes $k_x$ and $k_y$ show the AoA and frequency approximation used in MLSP-Net (Equation 3.6). Recall, we use this approximation because we are not given the sonar center frequency, $f_0$, for the data; not an uncommon scenario. We see no more than a degree of AoA error across the frequency band using our approximation.

with Table 3.1, most high-frequency SAS systems commonly have $B_F \leq 0.4$ giving a maximum AoA error less than five degrees using the approximation of using Equation 3.6, an acceptable error for the purposes of MLSP-Net. Finally, we see from Equation 3.7 lines of constant AoA become constant in $k_y$ as the center frequency increases and the error of $\theta \approx \alpha k_y$ vanishes.

### 3.3.3 MLSP-Net Network Design & Data Flow

We first describe the overall flow of MLSP-Net and then describe each component in detail: the Multi-Look Sequence Path, the Static Image Path, and Final Image Segmentation. Figure 3.4 shows an architectural diagram of the method. Overall, the input to MLSP-Net
Figure 3.7: Maximum angle-of-arrival error as a function of fractional bandwidth. As the fractional bandwidth decreases, the angle-of-arrival approximation error decreases. Several high-frequency (HF) SAS systems (as used here) and their fractional bandwidths are in Table 3.1.

Table 3.1: Fractional bandwidths for several SAS systems mentioned in the literature. We see for high-frequency SAS systems the maximum AoA error is less than five degrees.

<table>
<thead>
<tr>
<th>Source</th>
<th>Sonar Type</th>
<th>Fractional Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>[140]</td>
<td>High-frequency</td>
<td>0.41</td>
</tr>
<tr>
<td>[141]</td>
<td>High-frequency</td>
<td>0.17</td>
</tr>
<tr>
<td>[139]</td>
<td>High-frequency</td>
<td>0.2</td>
</tr>
<tr>
<td>[141]</td>
<td>Low-frequency</td>
<td>0.5</td>
</tr>
</tbody>
</table>

is a complex-valued SLC image and the output is a vector of class probabilities for each pixel. Upon input, the SLC is in real-imaginary form with each pixel representing a complex value. The SLC is then fed to two paths, the Multi-Look Sequence Path (where temporal features are extracted) and the Static Image Path (where static features are extracted). The output of these paths is then concatenated and fed to a Final Image
Segmentation network whose output is a vector class probabilities for each pixel.

For all convolutions in MLSP-Net, we use the “valid” padding scheme but prepend the convolutions with “reflect” padding along the height and width dimensions so the resultant convolution has the same height and width dimensions as the input. Unless otherwise specified, the network weights are initialized using a normal distribution scaled by the scheme in [114]. Tensorflow 2.5.0 [9] is used to train our networks. All network parameters are updated every mini-batch via backpropagation since our network is fully differentiable from input to output. This includes all U-Nets, Filter-Modules, Squeeze and Excitation Modules, 2D Convolutional LSTM, and the Segmentation Net. MLSP-Net contains 5,103,159 trainable parameters.

3.3.3.1 Multi-Look Sequence Path

The Multi-Look Sequence Path begins with the input SLC and transforms it to $k$-space via a complex 2D Fourier transform (a differentiable operation). Next, the $k$-space map is copied eight times with each copy filtered by its respective Filter-Module numbered one through eight; this operation narrows the receive beam of each image to a subset of its full azimuthal beam pattern (see Figure 3.5 for a geometric diagram) and then steers it to the desired look-angle. The eight filtered images are then transformed back to the spatial domain via an inverse complex 2D Fourier transform (also a differentiable operation). Next, the magnitude of the resultant complex-valued images is taken and each image is dynamic range compressed (DRC’ed) using the differentiable method of [15]. Next, each image is downsampled by a factor of two in both dimensions using integration; this is done for memory and compute considerations. Next, each image is fed to a shared U-Net (see Table 3.4) whose output is fed to individual squeeze and excitation (S&E) modules [142] using a reduction ratio of $r = 8$. The final sequence of features maps are then fed to a bi-directional 2D convolutional long-term-short-term-memory (LSTM) model [143,144]. 2D Convolutional LSTMs provide a mechanism to model spatial correlations over sequentially meaningful tensors by creating and updating a context vector after each tensor of the sequence is presented to the module. The bi-directional aspect of the module considers both forward and reverse instantiations of the sequence which we know from domain knowledge are equally relevant. The 2D convolutional LSTM consists of sixteen kernels each of size $5 \times 5$. The resultant output represents the end of the Multi-Look Sequence Path and has shape $256 \times 256 \times 32$.

Each Filter-Module contains a differentiable filter function specified in the $k$-space domain which is capable of learning and fine-tuning the filter parameters during the
training procedure. We accomplish this by constructing the filter to be of a band-pass
form defined by a rational approximation of the rectangle function shown in Equation
3.8 as,

\[
\text{rect}(t) = \lim_{n \to \infty, n \in \mathbb{Z}} \frac{1}{(2t)^{2n} + 1}
\]

where \( t \) is the domain. Figure 3.8 shows the formation of the rectangle shape of the filter
as \( n \) increases. Consequently, defining the rectangle function as a soft approximation
using a fixed \( n \ll \infty \) yields a smooth function alleviating Gibbs artifacts normally present
when using a hard rectangle filter. Furthermore, we specify the relevant filter parameters
of bandwidth, offset, and attenuation all in a differentiable manner as given in Equations
3.9 and 3.10,

\[
\text{softrect}_n(t) = \frac{1}{(2t)^{2n} + 1}
\]

\[
\text{softfilter}(\theta, \theta_0, \theta_{BW}, \alpha) = \text{softrect}_n \left( \frac{\theta - \theta_0}{\theta_{BW}} \right) \cdot \alpha + (1 - \alpha)
\]

where \( n \) controls the filter fidelity (i.e. the “sharpness” of the filter), \( \theta_0 \in [-0.5, 0.5] \) is
the filter offset, \( \theta_{BW} \in [\epsilon, 1] \) is normalized look angle bandwidth, and \( \alpha \in [\epsilon, 1] \) is the
attenuation. Figure 3.9 illustrates the how the filter is defined.

We now address the initialization of the filter parameters for the Filter-Modules. Recall Figure 3.3 whereby the consecutive sequence of images filtered by incremental AoA
advances yields a richer semantic interpretation of the images than the “No Filtered” image
alone. Moreover, the sequence loses some of its interpretability if it is randomly shuffled.
Therefore, we initialize the \( k \)-space filters in a domain enriched fashion as to initially
produce a useful ordered temporal sequence based on author experience and [145]. The
resulting initialization yields a sequence that is productively used by the 2D convolutional
LSTM. Conversely, choosing the \( k \)-space filter parameters (e.g. \( \theta_{BW}, \theta_0, \alpha \)) randomly may
not induce this meaningful image sequence we wish our network to exploit. We initialize
the \( k \)-space filters in the Filter-Modules as follows: (1) the positions (e.g. \( \theta_0 \)) are spaced
uniformly across look angle with a bandwidth of \( \theta_{BW,i} = 0.6 \), and (2), the attenuation
is set to \( \alpha_i = 0.5 \). Finally, we use a filter fidelity of \( n = 10 \) (reference Equation 3.9).
Consequently, for a given set of input parameters \( \{\theta_{0,i}, \theta_{BW,i}, \alpha_i\} \), the Filter-Module, \( i \),
the 2D \( k \)-space filter is constructed as

\[
\text{Filter-Module}_i = \text{softfilter}(\theta, \theta_{0,i}, \theta_{BW,i}, \alpha_i) \otimes 1^T
\]

where \( \theta = \{r : r = \frac{n}{255} - 0.5, n \in \{0, 1, ..., 255\}\} \) and \( \otimes \) is the Kronecker product (used as
3.3.3.2 Static Image Path

The Static Image path processes the input SLC similar to existing SAS segmentation methods by examining the magnitude SLC image.

This path ingests the input SLC and downsamples it by a factor of two in each dimension using averaging. Next, we convert the SLC to a magnitude image and then standardize it using

\[
\text{Standardize}(x) = \frac{|x| - \text{mean}(|x|)}{\text{stddev}(|x|)}
\]

where \( x \) is the input SLC and stddev is the standard deviation. Through cross-validation, we found the best results by standardizing the image as opposed to applying DRC as done in the Multi-Look Sequence Path. This image is then input to a U-Net (see Table 3.4) which is a separate U-Net from the Multi-Look Sequence Path. The resultant output represents the end of the Static Image Path and has shape 256 × 256 × 16.

3.3.3.3 Final Image Segmentation

The two resultant feature maps from each path (Multi-Look Sequence Path and Static Image Path) are concatenated in the feature dimension and then passed to a “Segmentation Net” (see Table 3.3).

Segmentation Net contains a “Global Channel Weight (GCW)” layer, 2D spatial dropout [146] with dropout proportion set to 0.8, and finally a 1x1 “2D” Convolution [147]. GCW functions similarly to a squeeze and excitation (S&E) network [142] but we remove the dependency on the input data so the same weighting is given to each input sample. Through cross-validation, we found this re-weighting scheme gives us better results than a canonical S&E scheme. Our re-weighting scheme is input-independent, unlike S&E, which we think benefits our limited training data scenario in preventing overfitting with input-dependent weighting. In this way, we get the good benefits of channel re-weighting as given by S&E networks but forgo making the re-weighting input-dependent. We initialize all the channel weights to unity at the start of training. The output of Segmentation Net is a set of logits for each pixel.

The output of Segmentation Net is upsampled by a factor of two using nearest-neighbor interpolation. Finally, a softmax operation is performed along the logits dimension to arrive at \( \hat{y} \), the predicted class probabilities for each pixel. The resultant output represents
Table 3.2: Classes making up our dataset along with the corresponding color in plots.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Abbreviation</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shadow</td>
<td>SW</td>
<td>Red</td>
</tr>
<tr>
<td>Sand (dark)</td>
<td>SD</td>
<td>Blue</td>
</tr>
<tr>
<td>Sand (light)</td>
<td>SL</td>
<td>Green</td>
</tr>
<tr>
<td>Seagrass</td>
<td>SG</td>
<td>Purple</td>
</tr>
<tr>
<td>Cobble / Rock</td>
<td>RK</td>
<td>Yellow</td>
</tr>
<tr>
<td>Ripple (Small)</td>
<td>RS</td>
<td>Brown</td>
</tr>
<tr>
<td>Ripple (Large)</td>
<td>RL</td>
<td>Pink</td>
</tr>
</tbody>
</table>

the end of Final Image Segmentation and the output of MLSP-Net. The final output shape is $512 \times 512 \times 7$.

3.4 Experiments

3.4.1 Dataset Description

The imagery used to evaluate our experiment is from a high-frequency SAS aboard an unmanned underwater vehicle [125]. Each image starts as size $1001 \times 1001$ pixels and is downsampled using integration to $512 \times 512$ pixels for training/evaluation due to memory constraints. The dataset totals 113 SLC images composed of seven labeled classes (Table 3.2 and Figure 3.2) and one “unlabeled” class. No location information or other metadata was provided for the images including the sonar center frequency, $f_0$, (see Section 3.3.2 for the implications of this for our method). The sizes of the training and test sets are 80 and 33 images respectively. Of the training data, we use 64 images for training and 16 images for validation. The distribution of labels is class imbalanced and shown in Figure 3.11.

3.4.2 Experimental Setup

We trained MLSP-Net for 500 epochs using Adam [120] with a learning rate of $10^{-3}$ and a mini-batch size of 64. Training a single model of MLSP-Net takes approximately 6.5 wall clock hours on an NVIDIA Titan X Pascal GPU. The training iteration giving the best validation result is used to evaluate the test set. The loss function used to train the
Table 3.3: Description of the Segmentation Net block of Figure 3.4.

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Layer Function</th>
<th>Dim.</th>
<th># Filters</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcw</td>
<td>Global Channel Weighting</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>s2dd</td>
<td>Spatial 2D Dropout (0.8)</td>
<td>N/A</td>
<td>N/A</td>
<td>gcw</td>
</tr>
<tr>
<td>output</td>
<td>2D Conv</td>
<td>1 x 1</td>
<td>7</td>
<td>s2dd</td>
</tr>
</tbody>
</table>

Table 3.4: Description of the U-Net model based around a pre-trained Resnet50 encoder [97]. For conciseness, we list the decoder side of the network. The U-Net multiplies the input by 128 to be consistent with the scaling used during the pre-training. Upsampling uses the nearest neighbor method.

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Layer Function</th>
<th>Dim.</th>
<th># Filters</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1a</td>
<td>Pad+Conv+BN+GELU</td>
<td>3x3</td>
<td>128</td>
<td>conv3_block4_out</td>
</tr>
<tr>
<td>conv1b</td>
<td>Pad+Conv+BN+GELU</td>
<td>3x3</td>
<td>128</td>
<td>conv1a</td>
</tr>
<tr>
<td>up2</td>
<td>Upsampling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv1b</td>
</tr>
<tr>
<td>conv2a</td>
<td>Pad+Conv+BN+GELU</td>
<td>3x3</td>
<td>64</td>
<td>up2, conv2_block3_out</td>
</tr>
<tr>
<td>up3</td>
<td>Upsampling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv2a</td>
</tr>
<tr>
<td>conv3a</td>
<td>Pad+Conv+BN+GELU</td>
<td>3x3</td>
<td>32</td>
<td>up3, conv1_relu</td>
</tr>
<tr>
<td>conv3b</td>
<td>Pad+Conv+BN+GELU</td>
<td>3x3</td>
<td>32</td>
<td>conv3a</td>
</tr>
<tr>
<td>up4</td>
<td>Upsampling</td>
<td>2x2</td>
<td>N/A</td>
<td>conv3b</td>
</tr>
<tr>
<td>conv4a</td>
<td>Pad+Conv+BN+GELU</td>
<td>3x3</td>
<td>16</td>
<td>up4</td>
</tr>
<tr>
<td>output</td>
<td>Pad+Conv+BN+GELU</td>
<td>3x3</td>
<td>16</td>
<td>conv4a</td>
</tr>
</tbody>
</table>

The network is the weighted categorical focal loss [111] given by Equation 3.13,

\[ \mathcal{L} = \sum_{k=1}^{h} \sum_{kk=1}^{w} w_y(k, kk) \cdot \mathcal{L}_{\text{focal}}(y(k, kk), \hat{y}(k, kk)) \]  

(3.13)

where \( h, w \) are the image height and width respectively, \( w_y \in \mathbb{R}^{h \times w} \) is an indicator map of \( \{0, 1\} \) noting if the pixel is labeled (i.e. see Figure 3.1 right; we do not accrue loss for unlabeled pixels), \( \mathcal{L}_{\text{focal}} \) is the categorical focal loss [111], \( y \in \mathbb{R}^{h \times w \times 7} \) are the ground truth labels for each pixel, and \( \hat{y} \in \mathbb{R}^{h \times w \times 7} \) is the estimated labels for each pixel.

We evaluate and compare methods using mean pixel accuracy (MPA) because of its robustness to class imbalance which is present in our dataset. We define MPA as,

\[ \text{MPA} = \frac{1}{7} \sum_{c=1}^{7} \text{Acc}_c \]  

(3.14)

where \( \text{Acc}_c \) is the accuracy of class \( c \) defined in a one-versus-all manner as,

\[ \text{Acc}_c = \frac{\text{TP}_c + \text{TN}_c}{\text{total number of labeled pixels}} \]  

(3.15)
where $\text{TP}_c$ represents number of true positives in a one-vs-all scenario and is defined as the total number of labeled pixels where $\arg\max y(x, y) = c$ and $\arg\max \hat{y}(x, y) = c$, and $\text{TN}_c$ represents the number of true negatives in a one-vs-all scenario and is defined as the total number of labeled pixels where $\arg\max y(x, y) \neq c$ and $\arg\max \hat{y}(x, y) \neq c$.

Note, the set of $\text{Acc}_c, c \in \{1, 2, \ldots, 6, 7\}$ are the diagonal elements of the confusion matrix (see Figure 3.12) and MPA is the mean of the diagonal elements. Note, the denominator of Equation 3.15 only considers the number of ground-truth labeled pixels across all images since this is a weakly-labeled dataset with less than 45% of pixels labeled in the test set (see Figure 3.11).

### 3.4.3 Comparison with State-of-the-Art Algorithms

We compare our results with three state-of-the-art methods used specifically in sonar/SAS seabed environment image segmentation: Lianantonakis, et al. (2007), Williams (2009) [61], and Zare, et al. (2017) [149]. We also compare against a SOTA deep learning segmentation algorithm, the U-Net [102]. In this section, we give the implementation details we use in generating the comparison methods as no source code is publicly available to evaluate. We make a best effort attempt to reproduce the methods as given in their respective sources. Our implementation is based on Python 3.7 code running on an Intel (R) Core (TM) i9-7960X 2.80GHz CPU with Linux operating system.

**Lianantonakis, et al. (2007) [150].** This method uses Haralick features [151] derived from the gray-level co-occurrence matrix (GLCM) and couples this with active contours to arrive at a binary class mapping. We extend this work to multiple classes by simply using the same feature descriptors as the original work but apply $k$-means++ [152] to cluster; a similar replication approach is used in [59]. We ran $k$-means++ with one-hundred random initializations and selected the run producing the minimum within cluster sum of squares error in a manner consistent with [61].

**Williams (2009) [61].** This method uses wavelet features along with spectral clustering to compute the segmentation map. We found spectral clustering results in similar performance as using $k$-means++ so we opt to use for simplicity as we did in Lianantonakis, et al. mentioned above; a similar replication approach is used in [59].

**Zare, et al. (2017) [149].** In this work, the feature sets are produced by Sobel edge

---

\[1\] There are many well-known deep learning segmentation algorithms, but they are not designed to incorporate the SAS-specific insight of MLSP. MLSP-Net can use any SOTA segmentation module. However, MLSP-Net provides SAS-specific enhancements over black-box methods such as a U-Net.
descriptors (Sobel) [153], histograms of oriented gradients (HOG) [154], and local binary pattern (LBP) features [155]. For each feature descriptor, we use the same sliding window strategy of Lianantonakis, et al. [150] to derive a feature vector for each pixel.

Rahnemoonfar, et al. (2019) [148]. In this work, a deep network composed of dilated convolutions, dense modules, and inception modules is used to perform semantic segmentation for the automatic extraction of potholes in SSS imagery. We compare against this method despite being applied to SSS imagery because it was recently developed, supervised (few supervised algorithms exist for SAS segmentation), and obtains SOTA results on a real SSS dataset. We train this algorithm using the Adam optimizer and a learning-rate of $10^{-3}$ as specified by the paper. We select the epoch for test set evaluation using the same scheme as MLSP-Net which is to use early stopping based on the MPA of the validation set.

Sun, et al. (2021) [130]. This work combines a super-pixel method with deep learning using an algorithm similar to Deep Cluster [156] to cluster the data and assign class labels. Each image is parsed by a deep network and then super-pixels are formed from the pixel embedding and updated periodically during training. The network is trained by iteratively learning the pseudo-labels generated by the super-pixels and the super-pixel assignment is updated every 200 iterations.

We report confusion matrices and mean pixel accuracy (MPA) to assess performance. MPA is more robust to class imbalance so we use it as a metric rather than traditional pixel accuracy. Notably, we cannot apply the commonly used segmentation metric of intersection over union (IoU) because only some portion of each class is labeled and the true class probability for a pixel may be a mixture of classes (i.e. mixture of seafloor textures). Consequently, these properties make the intersection and union operations of IoU not applicable to this setup.

Table 3.5 shows the pixel average per class and the mean pixel average (MPA) for several comparison methods. We see in the Table our MLSP-Net method has the highest MPA of all the methods and provides the best classification for classes light sand (SL), rock (RK), small ripple (RS), and large ripple (RL). Additionally, we see superior performance over MLSP-Net-No-Filter, essentially a U-Net and defined in Section 3.4.4, for the shadow class which is especially easy to confuse with the dark sand class demonstrating the benefit of activating the learnable Filter-Module. Figure 3.12 shows the confusion matrices for the top two methods Table 3.5 of ten runs of MLSP-Net and MLSP-Net-No-Filter. The Figure shows MLSP-Net has better performance on average for the majority of classes but also shows reduced variance demonstrating its robustness.
to training sample selection.

### 3.4.4 Ablation Study

To demonstrate the necessity of the AoA filters in MLSP, we train MLSP-Net without this processing step and report results. This is accomplished by nulling the filtering operation in each Filter-Module in Figure 3.4 so no filtering occurs and all images fed to the convolutional 2D LSTM module are exactly the same thus removing any notion of a temporal sequence. Mathematically, this is accomplished by making the parameters of the Filter-Modules fixed and setting $\theta_{0,i} = 0, \theta_{BW,i} = 1.0$, and $\alpha_i = 0$ in Equation 3.10. We refer to this configuration as “MLSP-Net-No-Filter” in our results. This configuration essentially results in two U-Nets in a small ensemble with magnitude-only input imagery so we forgo explicitly evaluating a U-net for comparison since it is already represented by this configuration.

Figure 3.13 shows a distribution of MPA over ten runs between our proposed method and MLSP-Net-No-Filter described in Section 3.4.4. The figure shows our proposed method has a higher median (0.921 versus 0.902) and mean MPA (0.920 versus 0.906) than the next best method, MLSP-Net-No-Filter. Finally, Figure 3.10 shows example segmentations of our proposed method and the next two best methods, MLSP-Net-No-Filter and Rahnemoonfar, et al. [148].

### 3.4.5 Filter Parameters During Training

We show each Filter-Module’s bandwidth, attenuation, and position as a function of training epoch for one run of MLSP-Net in Figure 3.14. We see from the figure the most significant changes occur with filter position and bandwidth and the least change occurs with filter attenuation. Figure 3.14.b shows a thirteen percent increase in filter bandwidth of a near-boresight filter denoted in red for each subfigure indicating the network emphasizing boresight AoA features.
Table 3.5: Pixel accuracy by class and mean pixel accuracy (MPA) for each method. Larger numbers indicate better performance. Classes are listed in Table 3.2. Best for each class in bold. We see the MLSP-Net has the best MPA of all the methods and yields best results for four of the seven classes. Moreover, we show improved performance over the next best method for the shadow class which is easily confused with dark sand (MPA of 0.632 for MLSP-Net-No-Filter versus 0.714 for MLSP-Net) demonstrating the utility of the Filter-Modules and 2D modeling of the MLSP features.

<table>
<thead>
<tr>
<th>Method</th>
<th>SW</th>
<th>SD</th>
<th>SL</th>
<th>SG</th>
<th>RK</th>
<th>RS</th>
<th>RL</th>
<th>MPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lianantonakis, et al. [150]</td>
<td>0.00</td>
<td>0.64</td>
<td>0.46</td>
<td>0.17</td>
<td>0.67</td>
<td>0.69</td>
<td>0.79</td>
<td>0.49</td>
</tr>
<tr>
<td>Williams [61]</td>
<td>0.72</td>
<td>0.04</td>
<td>0.84</td>
<td>0.23</td>
<td>0.85</td>
<td>0.25</td>
<td>0.97</td>
<td>0.56</td>
</tr>
<tr>
<td>Zare, et al. [149]</td>
<td>1.00</td>
<td>0.00</td>
<td>0.21</td>
<td>0.54</td>
<td>0.62</td>
<td>0.00</td>
<td>0.01</td>
<td>0.34</td>
</tr>
<tr>
<td>Rahnemoonfar, et al. [148]</td>
<td>0.477</td>
<td>0.971</td>
<td>0.971</td>
<td>0.811</td>
<td>0.972</td>
<td>0.719</td>
<td>0.997</td>
<td>0.845</td>
</tr>
<tr>
<td>Sun, et al. [130]</td>
<td>0.81</td>
<td>0.87</td>
<td>0.87</td>
<td>0.69</td>
<td>0.96</td>
<td>0.45</td>
<td>0.98</td>
<td>0.80</td>
</tr>
<tr>
<td>MLSP-Net-No-Filter</td>
<td>0.632</td>
<td>0.986</td>
<td>0.969</td>
<td>0.831</td>
<td>0.951</td>
<td>0.974</td>
<td>0.997</td>
<td>0.906</td>
</tr>
<tr>
<td>MLSP-Net</td>
<td>0.714</td>
<td>0.977</td>
<td>0.978</td>
<td>0.816</td>
<td>0.972</td>
<td>0.986</td>
<td>0.998</td>
<td>0.920</td>
</tr>
</tbody>
</table>
Figure 3.8: (Top) Convergence to the rectangle function by the limit of a rational function in Equation 3.8. The figure shows a growing sequence of $n$ ultimately converging to the rectangle function as $n \to \infty$. (Bottom) Magnitude Fourier transform (i.e. magnitude impulse response) of top showing convergence to a sinc function as $n \to \infty$. The parameter $n$ in Equation 3.8 allows us to trade off filter fidelity with ringing artifacts of the impulse response (i.e. Gibbs phenomenon) as well as dampening the coefficients away from the origin to effectively reduce the support of the filter. We use $n = 10$ for each Filter-Module in MLSP-Net.
Figure 3.9: A sample output of softfilter_{10}(\theta, \theta_{BW} = \frac{1}{3}, \theta_0 = -0.125, \alpha = 0.5) using n = 10 in Equation 3.9. The softfilter_{10} function is used in each Filter-Module of Figure 3.4 to filter the input SLC in the k-space domain to filter by specific look-angles of acoustic energy. See Figure 3.3 for an example of these filters applied to a SAS image and Figure 3.6 to see k-space coordinates of an example high-frequency SAS system which the softfilter_{10} function is applied across all frequencies to filter by look-angle.
Figure 3.10: Sample results from the top three methods of Table 3.5: Rahnemoonfar, et al. [148]; our proposed method with ablation MLSP-Net-No-Filter (which is effectively a U-Net ensemble, see Section 3.4.4); and our full proposal MLSP-Net. For rows top to bottom: input image; ground truth labels (recall these are weak/partial labels); Rahnemoonfar, et al. [148] results; classification errors between ground truth and Rahnemoonfar, et al. [148]; MLSP-Net-No-Filter results; classification errors between ground truth and MLSP-Net-No-Filter, MLSP-Net results; and classification errors between ground truth and MLSP-Net. As shown, MLSP-Net makes fewer classification errors overall compared to the other methods. Furthermore, MLSP-Net classifies the shadow area on the left side of the image (iii) better than our proposed method with ablation (MLSP-Net-No-Filter) thus demonstrating the improvement given by the AoA filtering. See Table 3.2 for class-color mappings.
Figure 3.11: Distribution of each class (including unlabeled pixels) for the training and test sets. The dataset was weakly labeled hence a large number of unlabeled pixels. Moreover, the dataset is highly class imbalanced.
Figure 3.12: Confusion matrices for the two best methods of Table 3.5, our proposed MLSP-Net method and baseline MLSP-Net-No-Filter. Class abbreviations defined in Table 3.2. These results are composed of ten training runs. We see MLSP-Net yields better MPA in four of the seven classes and does especially better at differentiating shadow and dark sand classes. Note, the MLSP-Net-No-Filter configuration is essentially two U-Nets put together in a small ensemble so we forgo explicitly evaluating the U-Net as a comparison since it would be redundant.
Figure 3.13: Box and whisker plot of MPA over ten runs of the two best methods of Table 3.5. Orange lines of the boxplot indicate median values (0.921 versus 0.902) and green triangles indicate average (0.920 versus 0.906). We see the MLSP-Net yields better performance and less sensitivity to training sample selection for training than MLSP-Net-No-Filter which does no $k$-space filtering. Thus, this test demonstrates not only the utility of this domain knowledge in improving network performance but also its robustness to training sample selection which is especially important when abundant data is not available as is the case here.
Figure 3.14: The eight Filter-Module parameters as a function of training epoch for one run of MLSP-Net. We see the most significant changes occur in filter bandwidth and position where the filter near boresight is allocated the largest bandwidth.
Chapter 4  
Contribution III: Domain Enriched Deep Learning for Coherent Autofocus of Synthetic Aperture Sonar Imagery

4.1 Introduction

Synthetic aperture sonar (SAS) is a modality that coherently combines acoustic sonar pings to synthesize a synthetic array to deliver high- and constant-resolution imagery of the seafloor [12]. These attributes make it advantageous over side-scan-sonar (SSS), where resolution is a function of range and artifacts from platform motion are common in the resultant imagery. SAS systems are often deployed on an unmanned underwater vehicle (UUV) where imagery is collected for oceanographic and military applications. It is not uncommon for the collected pings to be processed into single-look-complex (SLC) images onboard the UUV in real-time [15] for support of topside operators and to maximize survey objectives; tasks crucially dependent on the formed imagery being well-focused.

SAS imagery becomes defocused in the cross-track direction when the time-of-flight (ToF) of the transmitted ping to the seafloor and back to the receiver is incorrectly determined. Accurate ToF measurements are necessary to nominally align and coherently integrate the acoustic time series into a well-focused SAS image. The source of ToF error has many sources [67], but in practice is commonly from the misestimation of the speed-of-sound (SoS) in the water, incorrect knowledge of the platform position resulting in the incorrect distance from the transmit/receive array to the seafloor (e.g., unknown bathymetry), or systematic errors due to shortcomings of the vehicle motion estimation.
Figure 4.1: A well-focused SAS image, (a), corrupted by three types of systemic phase corruptions, (b). The resulting imaging artifacts sometimes fool operators into thinking the imagery is well-focused while simultaneously modifying the geometric structure of interesting objects thereby causing an operator to incorrectly overlook them. (a) is defocused using three types of systemic phase error: (c) quadratic phase error (QPE), (d) sinusoidal phase error, and (e) sawtooth/yaw phase error corresponding to panel (b). QPE manifests as a blur in the along-track direction while sinusoidal and sawtooth/yaw error manifests as image copies in the along-track direction [67]. This article proposes an autofocus method for images with phase errors like in panels (c), (d), and (e). Images (f), (g), and (h) are autofocused versions of (c), (d), and (e), respectively, using our proposed method.

Systematic errors present a unique problem for SAS image defocusing compared to random phase errors. Random phase error results in reduced signal-to-noise ratio in the resulting image while systemic phase error often result in spatially translated “copies” of the resultant seafloor superimposed on each other; see Figure 4.1.e as an example. Moreover, the point spread function of systematic errors often has structure which manifests itself in organized ways in the image. This kind of artifact is not readily
detected by humans in all seafloor types making the operator believe the image is actually well focused when it is not. Consequently, this kind of artifact also creates false image structure which may fool a human operator into incorrectly overlooking an object of interest.

To mitigate the aforementioned issues, often a broad-scoped class of algorithms referred to as autofocus, is used on the SLC image, to detect, characterize, and correct any phase errors resulting from inaccurate time-of-flight measurements. Often, these methods operate on independent small image patches by seeking to solve an optimization problem serving as a proxy for the properties of a well-focused image. However, these algorithms are often brittle and computationally slow, preventing real-time operation. Hence, their use for autofocusing SAS imagery in-situ aboard a UUV is tentative, especially if the imagery is used in downstream processing such as target recognition and segmentation [157] to guide autonomy. Often, a sophisticated optimization process or a series of heuristic image quality checks are employed for topside operation to ensure autofocus success. The former relies on having ample compute capability and the later relies on human-designed heuristics continually evolving (e.g., the numerous sharpness/contrast metrics as will be later covered) as new error cases are discovered and patched in existing methods.

4.1.1 Open Challenges in SAS Autofocus

Many SAS autofocus algorithms borrow principles or techniques from the synthetic aperture radar (SAR) community. For certain SAS collection configurations, many off-the-shelf (OTS) SAR autofocus algorithms directly apply to SAS imagery. However, sometimes careful assessment of the underlying assumptions of a SAR autofocus algorithm warrant further attention before applying to SAS [158–160]. Common to SAS, just as in SAR, many autofocus algorithms often exhibit one of the following pitfalls when trying to deploy in a real-world system:

1. The algorithms are iterative and take several dozen iterations and to converge making them unsuitable for embedded real-time deployment.

2. The algorithms require sophisticated optimization methods often necessitating hand-tuning to encourage convergence. These optimization “knobs” are usually tuned and then fixed for deployment allowing no in-situ adaptation.

3. The algorithms may fail catastrophically when unfavorable scene content is present.
causing well-focused imagery to become defocused after performing autofocus.

4. Many autofocus routines described in the literature report results on a small number of image samples making it difficult to assess their robustness when deployed on real-world datasets.

4.1.2 Overview of Our Proposal

Many deep learning methods exist for image de-blurring/autofocus [161–163] but most function as a “black-box” ignoring much of the physics of the underlying scattering phenomenology. Our proposal is novel in that it is SAS specific in that we use domain knowledge unique to synthetic aperture imaging in the design and optimization of our proposal. We do this in three ways:

1. SAS images are composed of complex-valued pixels (as opposed to real valued pixels of optical imagery) and our proposal necessitates the use the magnitude and phase from this representation in order to achieve success.

2. We use a SAS specific image degradation model that necessitates the use the both spatial and frequency domains (the latter commonly referred to as k-space domain in SAS) and is also a significant departure from traditional deblurring models often relying on real-valued deblurring kernels.

3. Due to the use of phase and k-space unique to SAS, our model is inherently coherent in that the speckle phenomena inherent to synthetic aperture imaging is preserved, something black-box deblurring methods have difficulty preserving because it is often interpreted as noise and not inherent to the phenomenology.

Common sources of phase error yield distinct point-spread-function (PSF) structure on the resultant image easily recognizable by humans [67]. For example, SoS error results from quadratic phase error (QPE) and broadens the PSF presenting in the image as blurring in the along track direction further broadening with range. Likewise, sinusoidal and sawtooth error, which result from vehicle position misestimation, result in a PSF with multiple distant sidelobes in along track direction presenting in an image as overlayed image copies of itself shifted in the along-track direction. Figure 4.1 shows examples of QPE, sinusoidal, and sawtooth error in a real-world SAS image.

CNNs have been shown to well model parts of the human visual system [164] so we find them as a suitable starting model for this objective. Inspired by the human ability
to recognize certain types of phase errors in SAS imagery [67], we propose to utilize this phenomenon by modeling SAS images using a convolutional neural network (CNN) and attempt to recover the underlying phase error. CNNs are capable of modeling both low-level and high-level semantic structure in images [165] which is crucial for assessment of SAS focus quality since errors may occur on both local (e.g., QPE) and global spatial scales (e.g., sawtooth error).

Recently, neural networks have shown promise in improving the performance of several classes of classically iterative image enhancement methods but without the need for explicit iteration during inference [166] making them attractive for real-time operation. We utilize this quality by proposing a CNN architecture which forgoes the need for iteration as common to many autofocus algorithms thus providing significant speed-up for real-time operation. We will demonstrate that the bulk of processing time necessitated by our proposal is shifted to the training phase of the network, and during the inference phase (such as when deployed on a UUV) only a single iteration or forward-pass of our network is necessary to compute results and thus significantly reduces processing load and algorithmic complexity.

A variety of autofocus algorithms exist which are based on an iterative scheme whereby the SLC is modified so a metric quantifying image sharpness (or contrast) is optimized [19, 167–171]. These algorithms are often differentiable and exploit this characteristic to arrive at a closed-form solutions of the gradient (e.g., [167]) in order to use gradient-based optimization algorithms. We also leverage the differentiability of this setup and bring to bear current deep learning techniques for the autofocus problem since the Fourier transform is differentiable and therefore amenable to optimization via stochastic gradient decent.

To this end, we present a SAS autofocus algorithm, which we call Deep Autofocus, that is computationally efficient and forgoes the need to provide hand-tuned parameters or special heuristics in order to provide well-focused imagery. Our approach leverages the knowledge that:

1. humans are good at assessing SAS image quality and readily identify the specific source(s) of phase error present in a SAS image by means of their effects on the image point-spread-function (PSF),

2. CNN’s provide a good model of the human-visual-system, and

3. CNN’s are capable solve classically iterative methods using only a single iteration.
Deep Autofocus is a deep neural network which is trained on pairs of focused-defocused SLC image pairs. Our network ingests a SLC image which is converted to a DRC version and an associated phase map. These are then fed to a pretrained off-the-shelf (OTS) CNN whereby features are extracted and used to estimate the results phase-error through a regression network. Next, the network applies the phase correction in the $k$-space domain of the input SLC and then Fourier transforms the result back to the spatial domain. This image is DRC’ed and compared to ground truth (also DRC’ed) where any errors are backpropigated during the training procedure. Although developed specifically for SAS, we see no reason why our proposal could not be applied to other synthetic aperture domains.

4.1.3 Contributions

In our initial work [172], we demonstrate the ability to autofocus low-frequency polynomial phase errors by examining the single-look-complex (SLC) image to determine a phase correction and then apply it in the $k$-space all in a single iteration. Here, we again use a domain-specific deep neural network (DNN) architecture automatically estimating phase error from a defocused image and correct it in real time, but we extend this work in the following ways:

1. We extend the phase error model expanding it from a polynomial based parametric model to a semi-parametric model which includes parametric models for polynomial and sinusoidal phase error.

2. We demonstrate that our semi-parametric model is capable of characterizing and correcting high-frequency sawtooth-error resulting from uncompensated yaw. We also describe shortcomings of our previous polynomial model and propose a more challenging one.

3. We improve our training regime and extend our previously self-supervised formation to a purely supervised one.

4. We augment our experimental section to further validate our results. We now include comparison against a common windowing technique used to improve convergence of traditional iterative sharpness/contrast metrics, and we compare our results to another popular method autofocus method, phase gradient autofocus (PGA).
4.2 Previous Work

There are several autofocus methods developed specifically around stripmap SAS collecting geometries. [173] proposes a generalization of the phase gradient autofocus (PGA) algorithm in order to handle the stripmap collection geometry of SAS. They identify several limitations of their method including: sensitivity to window size & placement, discarding potentially useful information in clutter areas, and a restriction that no or little platform yaw is present. [174] examine PGA for SAS and summarizes its strength and weakness along with citing differences in application to SAS from SAR. Finally, [175] improve upon the PGA method of [69] which proposes running PGA on subtiles of SAS image. [19] is a metric-based method which optimizes for image contrast and while applying a penalty to solutions which suggest uncommon vehicle motion dynamic. Notably, they show plots of the cost surface which is quite rough illustrating the challenge of the problem from an optimization viewpoint. [176] proposes a semi-parametric technique capable of autofocusing spatially varying blur over large areas and requires no identification of target scatters. [177] proposes an incoherent autofocus method whereby they exploit the fact that adjacent pings have correlated envelopes; a technique used by a class of sheer-averaging autofocus algorithms. As opposed to typical shear-averaging algorithms, they perform magnitude-only correlation. [178] proposes and improvement over an existing method by providing a means to mitigate bias from extended prominent scatters resulting in “overfocusing.” Additionally, they have a multi-band SAS system and integrate the use of both frequency bands to aid in the phase unwrapping procedure of their proposed method.

Phase gradient autofocus (PGA) methods are also common in SAR [173,179,180]. Generally, these methods take advantage of the fact that the point-spread-function is generally constant throughout the scene and therefore affects all scatterers similarly. Like the metric based methods, PGA too operates in an iterative fashion and uses either a parametric or non-parametric phase model.

There are dozens of autofocus algorithms in the synthetic aperture radar (SAR) literature of which many contain concepts applicable to SAS autofocus. It is beyond the scope of this work to address all of them. However, it would be thoughtless if we made no mention of metric-based SAR autofocus algorithms of which our proposal is based, has been commonly used in practice for SAS, and is a basis for many SAR autofocus methods. [167,171] evaluate the use of sharpness metrics commonly used in optics and derive closed form versions of their derivatives for use in gradient-based optimization.
schemes. Furthermore, they illuminate the need for “designer metrics" applied to scene specific content as no one metric seems uniformly best. [181] specifically addresses the problem of avoiding local minima in entropy-based SAR autofocus by proposing two strategies. The first strategy uses a wavelet-decomposition to reduce the resolution and thus reduce local minima resulting from clutter. An entropy based metric is then used to autofocus the image. The image resolution is then increased and the process repeats using the previous phase correction to bootstrap the optimization procedure. The process repeats until the native resolution of the image is reached. The second strategy using the general global optimization scheme of simulated annealing of which the authors note is computational burdensome. [169] derives a sharpness metric based on a statistical model of idealized SAR data. The derived metric is a limiting case of the popular intensity-squared metric. [170,182] evaluate the use of the minimum-entropy metric for SAR autofocus but require the selection of dominate scatters as a pre-condition. [183] uses auto-regressive backprojection while optimising the popular intensity-squared metric for incremental autofocus during the image reconstruction process; [184] proposes a similar method but operates on a per-pulse basis.

4.3 Proposed Autofocus Method: Deep Autofocus

4.3.1 Background

We begin by describing common metric-based iterative autofocus methods [167,171] as this serves as the basis from which our method will extend to a DNN framework. We are given a square, well-focused complex-valued SAS image, an SLC, which we denote as \( g \in \mathbb{C}^{M \times M} \) where the first dimension is along-track, the second dimension is range, and the sonar transmission arrives on the left side of the image (i.e., \( g \) represents a starboard-side collected SLC). We model the defocused image by a spatially uniform phase error throughout the scene represented by

\[
G_e = (e^{i\phi} \otimes \mathbf{1}^T) \odot G
\]

(4.1)

where \( G \) is the 1-D Fourier transform of \( g \) in the along-track dimension (over the image columns) and we denote this as \( G = \mathcal{F}\{g\} \). The phase error over the aperture is \( \phi \in \mathbb{R}^{M \times 1} \) and \( \mathbf{1} \) is an \( M \)-element column vector of all ones. \( \otimes \) is the Kronecker product (used as a broadcasting operator here) and \( \odot \) is the Hadamard product (i.e., pointwise multiplication). The estimated phase error responsible for the image defocusing is \( \hat{\phi} \).
Figure 4.2: A block diagram describing the application of a phase correction to a SAS image. The input is the defocused SAS image, \( g_e \), along with the proposed phase correction, \( \hat{\phi} \), and the output is the autofocused image, \( \hat{g} \). \( \mathcal{F} \) is the Fourier transform.

and is determined by solving the minimization problem (N.B. maximizing sharpness is minimizing negative sharpness)

\[
\hat{\phi} = \arg \min_{\phi} -\mathcal{M}(\mathcal{F}^{-1}\{(e^{-i\phi} \otimes 1^T) \odot G_e\}) \tag{4.2}
\]

where \( \mathcal{M} \) is one of the sharpness metrics in Table 4.2. The autofocused image \( \hat{g} \) is then given by

\[
\hat{g} = \mathcal{F}^{-1}\{(e^{-i\phi} \otimes 1^T) \odot G_e\} \tag{4.3}
\]

which is illustrated as a block diagram in Figure 4.2.

Often, a weighting function, \( w \in \mathbb{R}_+^{M \times M} \), applied to the argument of \( \mathcal{M} \) to remove the influence of unfavorable areas of the image [167]. Accounting for this, the minimization problem becomes

\[
\hat{\phi} = \arg \min_{\phi} -\mathcal{M}(w(|g_e|) \odot |\mathcal{F}^{-1}\{(e^{-i\phi} \otimes 1^T) \odot G_e|\}) \tag{4.4}
\]

Equation 4.4 is solved for each image \( g_e \) independently using an iterative method such as gradient descent (GD) or simulated annealing [181]. The resultant \( \hat{\phi} \) is then applied to \( g_e \) using Equation 4.3. Selection of \( w \) is determined through a hand-crafted function of the image-under-test; [167] gives an example of a common weighting function.
4.3.2 Motivation of Approach

As in our initial work [172], we note that humans have little trouble identifying defocused imagery. Furthermore, it was shown in [67] that common systematic phase errors manifest themselves as predictable visual artifacts due to their influence on the underlying point spread function (PSF) and that these artifacts are easily recognized by humans. We extend our previous work into the supervised learning regime thereby making use of much more information about each image during training, and we extend our phase modeling by making it semi-parametric and show that it is capable of estimating a type of high-frequency phase error resulting from uncompensated yaw.

All of the algorithms in Section 4.2 suffer from at least one of the following problems:

1. Performance results are reported on limited amounts of real-world data such as only a few images.

2. Iterative computation requiring several dozens of iterations to converge or necessitate sophisticated optimization algorithms to encourage convergence. Such methods are computational expensive or brittle for use in-situ on a UUV.

3. Necessitate the manual tuning of several parameters to encourage convergence.

4. Require identification of point-scatters as a pre-condition.

5. Only operate on a single image at a time discarding information learned from autofocusing other images.
We are motivated by the recent success of deep learning with the hope of improving upon these existing issues. Specifically, the work of [67] demonstrated how adept humans are at recognizing systematic phase error in SAS images and describe their impact on the resulting PSF in the imagery. Motivated by this, we sought to understand if convolutional neural networks could sufficiently model the resulting PSF effects and back them out in an automated and robust fashion.

Upon initial inspection of this problem we discovered ambiguities in phase error result in the same PSF. From [67], we see the PSF of a phase corrupted signal is represented by,

\[ f(y) = F_{k_y}^{-1} \{ [S(k_y) *_{k_y} G(k_y)] \tilde{S}(k_y) \} \]  

(4.5)

where \( G(k_y) \) is the Fourier transform of the phase error, \( *_{k_y} \) is the convolution operator in the \( k_y \) direction, \( S(k_y) \) is the frequency domain representation of an ideal signal. An example of this is the PSF for QPE which is plotted as Figure 1(a) in [67]. We see from [67], that the Fourier transform in the azimuthal direction of a signal with QPE gives the following form (with constants removed for clarity),

\[ G(k_y) = \sqrt{\frac{j}{\nu}} \text{rect} \left( \frac{k_y \nu}{\nu} \right) \exp \left\{ -j \frac{k_y^2}{\nu} \right\} \]  

(4.6)

where \( k_y \) is the azimuthal Fourier domain direction and \( \nu \) is the QPE strength. Taking the inverse Fourier transform of Equation 4.6, we get the PSF for QPE as,

\[ g(y) = j \sqrt{2\pi} \delta(y) \underbrace{\frac{\nu}{i}}_{1} \underbrace{\frac{|\nu|\text{sinc}\left(\frac{\nu}{2}\right)}{\sqrt{2\pi}}} \underbrace{e^{\frac{1}{2}j\nu^2}}_{3} \]  

(4.7)

and for clarity we distinguish the expression into three terms. For terms one and three, negating the sign of \( \nu \) results in the conjugate and does not change the magnitude of those terms. For the second term, we see the sinc function width is unaffected by a sign change of \( \nu \) (i.e., \( \text{sinc}(x) = \text{sinc}(-x) \)) since sinc is even symmetric) as well as the \( |\nu| \) term (also even symmetric). Thus, the overall expression does not change magnitude when the sign of \( \nu \) changes thus making its PSF ambiguous to sign. Evidently, to resolve this ambiguity, we must resort to information in the SLC phase map as our proposal will do. Thus, the overall expression does not change magnitude when the sign of \( \nu \) changes thus making its PSF ambiguous to sign. Evidently, to resolve this ambiguity, we must resort to information in the SLC phase map as our proposal will do.
We noticed among the previous methods that each image was autofocused independently. At most, optimization parameters from a previous image were used to bootstrap a follow-on image under the assumption that the phase error varied slowly spatially from image to image. Also, we pondered if we the information about the cost surface traversal during optimization of one image could be used in some intelligent way for speed up convergence for another.

Notably, we originally thought a convolutional neural network may not function for this task since the phase map is necessary to autofocus the image and its this map is quite random in nature. However, [185] demonstrated the ability of CNNs in conjunction with stochastic gradient descent (SGD) to solve a problem with an erratic cost surface: XOR decryption from noisy data. In this problem, a database of imagery is “encrypted” using a single key and the XOR function. [185] demonstrated that given “encrypted” / “decrypted” image pairs as input and output to a U-Net [102], that the net was able to successfully undo the XOR operation on imagery it had not seen during training. Moreso, the network was able to produce reasonable image reconstructions even when noise was added to the original “encrypted” images; quite a challenging task to solve.

Finally, we noted from [67] that the aberrations recognized by humans were in the spatial magnitude domain but the associated phase error is the $k$-space domain. Thus, it would be desirable if any proposed method could easily operate between these domains. We see that the Fourier transform is used to alternate between the spatial and $k$-space domains and is differentiable and thus easily integrated into deep learning training with SGD.

### 4.3.3 Design of Deep Autofocus Network

Deep Autofocus extends the metric-based optimization scheme of Equation 4.4 in two ways. First, we extend the form of $w$ so it is implicitly learned from a set of training images, specifically from DRC images and phase maps of the SLC. Second, we reformulate the optimization of Equation 4.4 so during deployment, an iterative method to solve for each image is not needed. Instead, a fast, single function is applied to all images during deployment.

The optimization of Equation 4.9 requires specification of the function family $h$. We use a CNN, DenseNet121 (defined in [96]), followed by a multi-layer perceptron (MLP) [187] (network shown in Figure 4.3). DenseNet121 is composed of 121 layers and serves as a feature extractor generating an output vector in $\mathbb{R}^{8 \times 8 \times 1024}$ which is then dimensionality reduced using global average pooling (GAP) [147] to $\mathbb{R}^{1024}$. This vector
Table 4.1: Network architecture of the Regression Network shown in Figure 4.3. GELU is Gaussian error linear unit [186]. The regression network takes input from the Feature Extraction Network, a pretrained Densenet121 backbone. The network computes the phase error using a semi-parametric model composed of two parametric models, polynomial and sinusoid, and a non-parametric model, vector with shape 256.

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Layer Function</th>
<th>Output Dim.</th>
<th># Filters</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>input1</td>
<td>Regression Network Input</td>
<td>8x8x1024</td>
<td>N/A</td>
<td>Features Extraction Network</td>
</tr>
<tr>
<td>bn1</td>
<td>Batch Normalization</td>
<td>8x8x1024</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>feature_vec</td>
<td>Global average pooling</td>
<td>1024</td>
<td>N/A</td>
<td>bn1</td>
</tr>
<tr>
<td>f1</td>
<td>Fully connected</td>
<td>N/A</td>
<td>512</td>
<td>feature_vec</td>
</tr>
<tr>
<td>f2</td>
<td>Fully connected + GELU</td>
<td>256</td>
<td>256</td>
<td>f1</td>
</tr>
<tr>
<td>f3</td>
<td>Fully connected + GELU</td>
<td>128</td>
<td>128</td>
<td>f2</td>
</tr>
<tr>
<td>f4</td>
<td>Fully connected + GELU</td>
<td>64</td>
<td>64</td>
<td>f3</td>
</tr>
<tr>
<td>f5</td>
<td>Fully connected + GELU</td>
<td>32</td>
<td>32</td>
<td>f4</td>
</tr>
<tr>
<td>f6</td>
<td>Fully connected + GELU</td>
<td>16</td>
<td>16</td>
<td>f5</td>
</tr>
<tr>
<td>poly_coefs</td>
<td>Fully connected + GELU</td>
<td>10</td>
<td>10</td>
<td>f6</td>
</tr>
<tr>
<td>poly</td>
<td>Evaluate polynomial</td>
<td>256</td>
<td>N/A</td>
<td>poly_coefs</td>
</tr>
<tr>
<td>s_freq</td>
<td>Fully connected</td>
<td>1</td>
<td>1</td>
<td>f6</td>
</tr>
<tr>
<td>s_phase</td>
<td>Fully connected</td>
<td>1</td>
<td>1</td>
<td>f6</td>
</tr>
<tr>
<td>s_amp</td>
<td>Fully connected</td>
<td>1</td>
<td>1</td>
<td>f6</td>
</tr>
<tr>
<td>sinusoid</td>
<td>Evaluate sinusoid</td>
<td>256</td>
<td>N/A</td>
<td>s_freq, s_phase, s_amp</td>
</tr>
<tr>
<td>f7</td>
<td>Fully connected + GELU</td>
<td>256</td>
<td>256</td>
<td>feature_vec</td>
</tr>
<tr>
<td>f8</td>
<td>Fully connected + GELU</td>
<td>64</td>
<td>64</td>
<td>f8</td>
</tr>
<tr>
<td>bottleneck</td>
<td>Fully connected + GELU</td>
<td>16</td>
<td>16</td>
<td>f9</td>
</tr>
<tr>
<td>f10</td>
<td>Fully connected + GELU</td>
<td>256</td>
<td>256</td>
<td>bottleneck</td>
</tr>
<tr>
<td>f11</td>
<td>Fully connected + GELU</td>
<td>256</td>
<td>256</td>
<td>f10</td>
</tr>
<tr>
<td>f12</td>
<td>Fully connected + GELU</td>
<td>256</td>
<td>256</td>
<td>f11</td>
</tr>
<tr>
<td>non-parametric</td>
<td>Fully connected</td>
<td>256</td>
<td>256</td>
<td>gap1</td>
</tr>
<tr>
<td>output</td>
<td>sum, remove mean, detrend</td>
<td>256</td>
<td>N/A</td>
<td>poly, sinusoid, non-parametric</td>
</tr>
</tbody>
</table>

is then fed down two paths. The first path is an MLP used to estimate the coefficients of a ten degree polynomial which then evaluated on the normalized aperture position \( t \in [-1, 1] \) and the parameters (amplitude, phase, and frequency) of a sinusoid evaluated over the interval \([0, 1]\). The second path is an MLP used to estimate a non-parametric phase error.

Table 4.1 shows the architecture of the regression network including the polynomial and non-parametric sub-networks. The table is divided into four section. Starting from the top of the table, the first section extracts and formats features from our DenseNet121 backbone. The second section describes a MLP used to generate the parameters for two parametric regression functions: a ten-degree polynomial and sinusoid. Both functions derive parameters from a common MLP head. The third section describes an MLP producing parameters for the non-parametric portion of our phase estimate. This section produces a 256-element-long vector. In the fourth and final section, the evaluated polynomial and sinusoid along with the non-parametric 256-element-long vector are added together producing the final phase error vector, the mean is then removed, and finally, the phase error vector is linear detrended. The mean is removed because its
presence has no effect on the resultant image. The phase error vector is detrended because it has no contribution on image quality but simply translates the image in the along-track dimension.

Since Equation 4.9 (including the Fourier transform and dynamic range compression) is differentiable, SGD is employed for optimization to learn \( \Theta \) using a small database of training images with data augmentation. Once training completes, we arrive at the non-iterative function \( f \) with fixed, but learned, weights \( \Theta \) which estimates the ground truth image \( g \) from a potentially defocused image \( g_e \).

We implicitly learn the weighting function \( w \) through \( h \). Function \( h \) takes as input an image and produces features suitable for phase error estimation which is similar to the purpose of \( w \). However, \( h \) extends \( w \) as \( w \) is only capable of weighting the image so “bad” areas of the image are suppressed while \( h \) is able to do this and selectively enhance or create new features from the image.

The point spread function is symmetric for many types of common phase errors (e.g., quadratic phase error) implying the sign of the phase error is not discernible from the DRC image. Thus, phase information is necessary to properly estimate \( \phi \). We verified this by training a network with the phase map input set always to zero and observed suboptimal results. Additionally, we substituted the DRC and phase map input with a different representation of the SLC, real and imaginary maps, and also observed suboptimal results.

4.3.4 Network Loss Function

The goal of Deep Autofocus is to find parameters \( \Theta \) for a function \( f \) so

\[
\hat{g} = f(g_e, \Theta)
\]  

holds for an image \( g_e \) selected from a typical population of SAS images. \( \Theta \) is a vector of learned but fixed parameters associated with \( f \). We solve for \( \Theta \) by minimization of

\[
\arg\min_{\Theta} \mathcal{L}(g, f(g_e, \Theta))
\]

where

\[
f(g_e, \Theta) = |\mathcal{F}^{-1}\{(i \cdot \exp(h(f_{DRC}(g_e), \arg(g_e), \Theta) \otimes 1^T) \otimes G_e)\}|
\]  

\( h \) is our CNN, \( \mathcal{L} \) is the loss function, \( f_{DRC} \) is the DRC function mapping the SLC to a low dynamic range, human consumable image. We use two flavors of loss function to
solve this optimization problem: self-supervised and supervised.

**Self-Supervised.** In this scheme, we use the self supervised formulation of [172]. This loss function seeks to find the phase error which improves the contrast between the input and resultant output image. The measure of contrast is given by,

\[
\mathcal{M}_{MNS} = \frac{\text{stddev}(|g|)}{\text{mean}(|g|)}
\]

(4.11)

and the resulting loss function is,

\[
\mathcal{L}(g, \hat{g}) = -\frac{\mathcal{M}_{MNS}(\hat{g}) - \mathcal{M}_{MNS}(g_e)}{\mathcal{M}_{MNS}(g_e)}
\]

(4.12)

This formulation does not require ground-truth defocused-focused image pairs.

**Supervised.** In this scheme, we directly try to minimize the perceptually difference between the resulting output image and its ground truth. We utilize two common perceptually oriented metrics for training and evaluation: structural similar metric (SSIM) [103] and multi-scale structural similarity metric (MS-SSIM) [104]. SSIM is defined as

\[
\text{SSIM}(x, y) = \left( \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \right) \cdot \left( \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \right) \cdot \left( \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \right)
\]

(4.13)

and MS-SSIM which is defined as

\[
\text{MS-SSIM}(x, y) = [l_M(x, y)]^{\alpha_M} \prod_{j=1}^{M} [c_j(x, y)]^{\beta_j} [s_j(x, y)]^{\gamma_j}
\]

(4.14)

where \( M = 5 \), \( \alpha_M = 0.1333 \), and \( \beta_j = \gamma_j \) and is equal to the associated scale power factor from the set \{0.0448, 0.2856, 0.3001, 0.2363, 0.1333\}.

We use the MS-SSIM as our network loss as to capture the local and global defocus effects present in each image. Our final network loss is,

\[
\mathcal{L}(g, g, \Theta) = \mathcal{L}_{\text{MS-SSIM}}(f_{\text{NN}}(g_e, \Theta), f_{\text{DRC}}(g))
\]

(4.15)
where \( f_{\text{NN}}(g_e, \Theta) \) is our DNN function with input image \( g_e \) and parameters \( \Theta \), \( \mathcal{L}_{\text{MS-SSIM}} = 1 - \text{MS-SSIM} \) \cite{188}, and \( f_{\text{DRC}} \) is the dynamic range compression function which is the rational tone mapping operator of \cite{14}.

\[
f_{\text{DRC}}(g) = \frac{q \cdot |g|}{(q - 1) \cdot |g| + 1} \quad (4.16)
\]

\[
q = \frac{0.3 - 0.3 \cdot \text{median}(|g|)}{\text{median}(|g|) - 0.3 \cdot \text{median}(|g|)} \quad (4.17)
\]

### 4.3.5 Training Procedure and Data Augmentation Scheme

To train our network, we use mini-batch size of thirty-two, stochastic gradient descent for optimization, and a learning rate schedule. The learning rate schedule uses an initial value of \( 10^{-1} \) for 250,000 epochs. Next, we train using a learning rate of \( 10^{-2} \) for 10,000 epochs. Finally, we use a learning rate of \( 10^{-3} \) for 10,000 epochs. We use the resulting model for evaluating the test set. Our training and validation datasets are each composed of 120 images. The initial weights, \( \Theta \), for the feature extraction network portion of \( h \), are from an ImageNet pre-trained Densenet121 model from \cite{9}. For the regression network portion of \( h \), the layers are initialized using \cite{189}. The model was trained using Tensorflow 2.4.1 \cite{9} on a graphics processing unit (GPU).

We employ data augmentation on the images during training. To mimic realistic low and high-frequency phase error seen in practice \cite{190}, we corrupt each image (see Equation 4.1) of the training and validation sets with phase error from a combination of polynomial, sinusoidal, and sawtooth forms using Equation 4.18,

\[
\Psi = \Lambda_{\text{poly}} \cdot S_{\text{poly}} \cdot f_{\text{poly}}(N)
+ \Lambda_{\text{sinusoid}} \cdot S_{\text{sinusoid}} \cdot f_{\text{sinusoid}}(\Omega, \Phi)
+ \Lambda_{\text{sawtooth}} \cdot S_{\text{sawtooth}} \cdot f_{\text{sawtooth}}(P) \quad (4.18)
\]

where \( \Lambda_{\text{poly}}, \Lambda_{\text{sinusoid}}, \Lambda_{\text{sawtooth}} \) are Bernoulli random variables with probability \( \frac{1}{2} \), \( S_{\text{poly}} \sim \mathcal{U}_{[-20, 20]} \), \( S_{\text{sinusoid}} \sim \mathcal{U}_{[-0, 10]} \), \( S_{\text{sawtooth}} \sim \mathcal{U}_{[-1.74, 1.74]} \) (i.e., ±100 degrees) are all scaling random variables, \( f_{\text{poly}}(n = N) \) returns a normalized polynomial of degree \( n \) defined as Algorithm 1, \( N \sim \mathcal{U}(2, 6) \), \( f_{\text{sinusoid}}(\omega = \Omega, \psi = \Psi) \) returns a sinusoid with frequency \( \omega \) [Hz] and phase \( \psi \) defined as Algorithm 2, \( \Omega \sim \mathcal{U}(0, 4) \), \( \Psi \sim \mathcal{U}_{[-\pi, \pi]} \), and \( f_{\text{sawtooth}}(p = P) \) returns a normalized sawtooth waveform with phase \( p \) and period six defined as Algorithm 3 where \( P \sim \mathcal{U}(0, 6) \). We alternate data augmentation between random perturbation
with random strength (Equation 4.18) and random perturbation with maximum strength (Equation 4.20). The final data augmentation scheme is given by,

\[ \Gamma = \alpha \Psi + (1 - \alpha)\Psi_{\text{max}} \]  

(4.19)

where \( \alpha \) is a Bernoulli random variable with probability \( \frac{1}{2} \).

Note that in Algorithm 3, the period is set to six. For a given SAS system, the period of yaw error is established by the number of active imaging channels in the array. Based on practical experience, we think six is a reasonable setting to use. However, this number may be modified accordingly in order to train with data from another system with a different number of channels.

---

**Algorithm 1: Definition of \( f_{\text{poly}} \)**

**Input:** \( n \in \{2, 3, 4, 5, 6\} \)

**Output:** \( p \in [-1, 1] \), normalized polynomial of degree \( n \) evaluated over \( t \)

Let \( t = \{ r : r = 2 \cdot \frac{n}{255} - 1, n \in \{0, 1, ..., 255\}\} \)

Select \( n \) random roots from \([-1, 1]\)

if \( n > 4 \) then

- Overwrite first root new value from \( U_{[-1,1]} \)
- Overwrite second root new value from \( U_{[1,1]} \)

end

Convert roots to polynomial coefficients, \( c_0, c_1, ..., c_6 \)

\( c_0 \leftarrow 0, c_1 \leftarrow 0. \)

\( v(t) \leftarrow \sum_{d=0}^{6} c_d \cdot t^d \)

\( p(t) \leftarrow \frac{v(t)}{\max|v| + \epsilon} \)

---

**Algorithm 2: Definition of \( f_{\text{sinusoid}} \)**

**Input:** \( f \in [0, 4], \phi \in [0, \pi] \)

**Output:** \( s \in [-1, 1] \), sinusoid function with phase \( \phi \) and frequency \( f \)

Let \( t = \{0, 1, ..., 255\} \)

\( s \leftarrow \sin(2\pi f \frac{1}{255} + \phi) \)

---

**Algorithm 3: Definition of \( f_{\text{sawtooth}} \)**

**Input:** \( \rho \in \{0, 1, 2, 3, 4, 5\} \)

**Output:** \( s \in [-0.5, 0.5] \), sawtooth function with phase \( \rho \) and period 6.

Let \( t = \{0, 1, ..., 255\} \)

\( s \leftarrow \frac{(t+\rho) \mod 6}{5} - 0.5 \)
Figure 4.4: A sample of ten phase error draws, $\psi = \Psi$, used for test set creation. The phase error draws are composed of polynomial functions, sawtooth functions, and combinations of thereof.

4.4 Experiments

4.4.1 Dataset Description

We use a real-world dataset from an HF SAS mounted on a UUV [125]. The dataset consists of 504 SLC images each $256 \times 256$ pixels in size and were constructed using an $\omega$-k beamformer. The dataset contains seven classes of seafloor: rock, packed sand, mud, small ripple, large ripple, sea grass, and shadow. Of the 504 images, a subset of 264 images are used as test images for algorithm evaluation. We use these original images as ground truth. The remaining 240 images are used to train our deep network with half of the images being used for training and half of the images being used for validation.

The test and validation images are corrupted once and used for all comparisons using a more difficult scheme than specified by Equation 4.18. We corrupt the test set severely by removing the stochastic weightings and use large fixed weightings as to evaluate the algorithm in difficult cases. This process is described as Equation 4.20,

$$\Psi_{\text{max}} = (2R - 1) \cdot c_{\text{maxQPE}} \cdot f_{\text{poly}}(N) + c_{\text{maxSin}} \cdot f_{\text{sinusoid}}(\Omega, \Phi) + (2R - 1) \cdot c_{\text{maxST}} \cdot f_{\text{sawtooth}}(P) \quad (4.20)$$

where $c_{\text{maxQPE}} = 18$, $c_{\text{maxSin}} = 8$, and $c_{\text{maxST}} = 1.71$. Figure 4.4 shows ten sample draws from $\Psi$ applied test set.
In the previous work [172], we form the polynomial by choosing a random degree and random coefficients. However, it was shown in [191] that the real roots of a polynomial with random coefficients tend towards the unit circle as the degree increases. This would place most of the real roots then at 1 or -1. Since we define normalized aperture coordinates over the same interval, the resulting polynomials produce uninteresting functions in this interval. We would like the roots to be evenly distribution across the aperture giving more compelling phase error structure. We accomplish this by selecting random real roots for our polynomial instead of random coefficients. Empirically, we noticed for polynomials greater than degree four that function asymptotes would form near the ends of the aperture causing the normalizing step to minimize fluctuations near the center. We diminish this effect by moving two of the random roots to just outside the unit circle for polynomials greater than degree four.

### 4.4.2 Evaluation Against Comparison Methods

To demonstrate the efficacy of our approach, we compare our results against several existing techniques including four metric-based methods and the standard phase-gradient-autofocus (PGA) method.

*Metric-based Autofocus* [19], [170], [169], [167]: These methods seek to optimize an image sharpness/contrast metric by proposing phase corrections which hope to improve overall image quality after several iterations. To improve convergence, the input image is often windowed to remove areas of the image deemed unfavorable for the optimization procedure. We evaluate four of these metrics (see Table 4.2) with and without the
windowing method of Equation 7 in [167] given by,

$$w(x) = \frac{1}{\left[ \sum y' |g_e(x,y')|^2 \right]^2}$$  \hspace{1cm} (4.21)

where $x$ is the along-track dimension and $y'$ is the range dimension. This weighting function result in each range bin containing the same amount of energy. Since we do not “patchify” our input images, we use the same weight everywhere (i.e., $w_{ko} = 1$ in [167], Equation 7).

*Phase Gradient Autofocus (PGA) [179,180]:* This method utilizes the fact that all parts of the image are corrupted by the same phase error. First, it finds the maximum magnitude value in each range bin and circularly shifting it to the center of the image which is then windowed based on non-coherent range averaged energy. Next, it windows the max-aligned image representation data using a threshold of -25 dB. After that, the windowed result is then Fourier transformed where the phase gradient is estimated, detrended, and phase unwrapped. Finally, the correction is computed and then applied in the $k$-space domain.

To measure autofocus efficacy, we use three common image quality assessment (IQA) metrics: peak-signal-to-noise ratio (PSNR) [103], structural similar (SSIM) [103], and multi-scale structural similarity (MS-SSIM) [104]. PSNR is a traditional metric historically used for image comparison and given by Equation 4.22,

$$\text{PSNR}(x,y) = 10 \cdot \log_{10} \left( \frac{\max(x)}{\text{MSE}(x,y)} \right)$$  \hspace{1cm} (4.22)

where

$$\text{MSE}(x,y) = \frac{1}{h \cdot w} \sum_{m=0}^{h-1} \sum_{n=0}^{w-1} (x(m,n) - y(m,n))^2$$  \hspace{1cm} (4.23)

and $h, w$ are the image height and width respectively. SSIM and MS-SSIM are contemporary methods that correlate well with human-based assessments of image quality. For each IQA metric, we compare the the original image (the ground truth before corruption with phase error) and the autofocused version we obtain by processing the defocused/corrupted image. The images are not despeckled prior to computing the image metric.

It is not uncommon for SAR autofocusing algorithms to evaluate efficacy using integrated sidelobe ratio (ISLR) or peak sidelobe ratio (PSLR). These metrics require the identification of point scatters which are have to be identified by hand or cooperative placed on the ground during collection. The latter is infeasible since the dataset provided
did not have any cooperative reflectors placed during collection. The former is cumbersome for large amounts of data and may be impossible to accurately identify depending on the environment. For this reason, we forgo the use of these metrics for evaluation / training and use visually-oriented metrics (i.e., SSIM and MS-SSIM) which mimic the perceived distortion as assessed by a human operator.

For run-time performance, we measure the time it takes to autofocus all images in the test set. To garner a useful comparison, we allow the sharpness metrics to optimize for ten iterations, likely conservative for deployment in UUV SAS operations. Recall, Deep Autofocus is designed to run using just a single iteration. Each sharpness metric models phase error as a ten-degree polynomial and is minimized using gradient descent (GD). To garner accurate run-time results, we implemented the sharpness metrics on the same GPU used to run Deep Autofocus. We did this by implementing the sharpness metrics and the GD procedure on a GPU using Tensorflow. All methods were run on an NVIDIA Titan X. The GD procedure of the sharpness metrics requires a tuning parameter, the learning rate used for GD. To give the best possible results, we used cross-validation to obtain the optimal learning rate for each metric from the set of learning rates \{10^{-7}, 10^{-6}, ..., 10^6\} (results shown in Section 4.5). For each sharpness metric, we selected the learning rate giving the best mean result over the test set.

The comparison methods may improve image quality but induce a slight along-track translation in the form of dilation or contraction of spatial features. This occurs even if the linear trend in phase correction is removed prior to its application. Although these images may not align with the reference image (which is required for our quality metrics), they may still be of good focus quality. To mitigate this issue and garner a fair comparison, we evaluate each metric at a range of along-track translations from -10 to 10 pixels and select the translation giving us the best score.

4.4.3 Autofocus Results for Each Method

Figure 4.7 shows the distributions of image quality scores for three metrics. All three subfigures show our proposed technique gives best results for the vast majority of the images in the test set compared to the comparison methods. Figure 4.8 shows the distributions of absolute improvement for each image for each metric. From the figure, we show that our proposed method gives the most improvement per image with no images resulting in worse image quality compared to the original defocused image. Finally, we compare the two formulations of our method: self-supervised and supervised. We show that the supervised formulation of our method gives slightly better image quality than
Figure 4.6: Examples of point-spread-function improvement using Deep Autofocus (supervised scheme). These two images are not from the test set and from a real-world SAS collection just after image reconstruction but with residual phase error. From left to right for each row: (a) Original image with zoom inset of point-of-interest, (b) autofocused image, (c) along-track slice of point, (d) estimate phase error by Deep Autofocus.

Figure 4.6 shows two example images where the image reconstruction process erroneously returned images with defocus. We show the autofocus focusing of a selected point-scatterer from each image along with a cut through its along-track magnitude which demonstrates how our proposed methods minimizes the width of the PSF thus sharpening the image. Additionally, we show the estimate phase error. We do not show ground truth imagery since it is unavailable in this scenario.

Section 4.6 Figure 4.10 shows an additional set of test results. Additionally, we provide Section 4.7 Figure 4.11 and Figure 4.12 as calibration for the reader showing the distribution of each image quality metric as a function of QPE strength, a common SAS phase error.

4.4.4 Phase Correction Error Compared to Ground Truth

Figure 4.5 shows the error between the estimated phase correction and the ground truth phase correction using RMSE. Additionally, we examine the RMSE phase error as a function of the aperture position. We see that the bulk of the phase error occurs at the ends of the aperture, which provides the least amount of impact on the imagery focus.
Figure 4.7: Image quality of the test set compared to ground truth. For all plots, higher numbers indicate better performance. Box extends from first quartile to third quartile, median is given as orange line, green triangle is mean. Shown are evaluations against three IQA metrics: (a) PSNR, (b) SSIM, and (c) MS-SSIM. As shown, DeepAF does significantly better than the comparison methods. For reference, we provide results for the original unfocused imagery as "No Autofocus."
Figure 4.8: Image quality improvement for each image of the test set compared to ground truth. For all plots, higher numbers indicate better performance. Box extends from first quartile to third quartile, median is given as orange line, green triangle is mean. Shown are evaluations against three IQA metrics: (a) PSNR, (b) SSIM, and (c) MS-SSIM. As shown, DeepAF does significantly better than the comparison methods.
The input single look complex (SLC) is $g \in \mathbb{C}^{M \times M}$ and $\text{stddev}(x)$ is the standard deviation over the elements of $x$. We make $b = \epsilon$ in OSF as it was shown when $b$ is large the metric is equivalent to SSI.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Metric Name</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[19]</td>
<td>Mean Normalized Stddev (MNS)</td>
<td>$\mathcal{M}_{\text{MNS}} = \frac{\text{stddev}(|g|)}{\text{mean}(|g|)}$</td>
</tr>
<tr>
<td>[170]</td>
<td>Minimum Entropy (ME)</td>
<td>$\mathcal{M}_{\text{ME}} = \sum_x \sum_y (|g|^2 \ln(|g|^2))$</td>
</tr>
<tr>
<td>[169]</td>
<td>Optml. Sharpness Function (OSF)</td>
<td>$\mathcal{M}_{\text{OSF}} = \sum_x \sum_y \ln(|g|^2 + b)$</td>
</tr>
<tr>
<td>[167]</td>
<td>Sum of Squared Intensity (SSI)</td>
<td>$\mathcal{M}_{\text{SSI}} = \sum_x \sum_y</td>
</tr>
</tbody>
</table>
4.5 Learning Rate Cross-Validation Results for Metric-Based Comparison Methods

Figure 4.9: Results of the grid search used to determine the best learning rate for each iterative autofocus metric and associated image quality metric. The goal is to find a learning rate which yields the best possible result for each configuration.

We conducted a grid search to find the best learning rate for the traditional metric-based approaches in order to garner a fair comparison against our proposed method. A search was performed for each sharpness-metric (e.g., MNS, ME, OSF, SSI) over each image quality metric, PSNR (a,d), SSIM (d,e), and MS-SSIM (c,f). We evaluated the sharpness metrics with and without the weighting function proposed by [167], Equation 4.21, and explained in Section 4.4.2. The learning rate with the best performance was used to evaluate the test set for a given sharpness-metric and image quality metric. Figure 4.9 contains the results of the grid search.
4.6 Example Autofocus Results on Test Set

Figure 4.10: From left to right: original defocused image, Deep Autofocus (supervised) result, ground truth well focused image, difference between Deep Autofocus and ground truth images, estimated (gray) and ground truth (black) phase error.
4.7 Sample image corrupted with increasing levels of QPE

Our QPE model is defined as,

\[ s \cdot t^2 \]  \hspace{1cm} (4.24)

where \( t = \{ r : r = 2 \cdot \frac{n}{255} - 1, n \in \{0, 1, \ldots, 255\} \} \) and \( s \) represents QPE strength.

Figure 4.11: Sample image corrupted with varying levels of QPE to visually illustrate how QPE strength relates to image degradation. Top row are the images and bottom row is the amount of QPE.
Figure 4.12: Image quality metrics as a function of quadratic phase error (QPE) strength for test set. The image quality metrics evaluated are (a) SSIM, (b) MS-SSIM, and (c) PSNR. Image are not despeckled prior to computing the metric. These plots, along with Figure 4.11, are provided for convenience to aid in interpreting the results of Figure 4.7 and Figure 4.8. Box extends from first quartile to third quartile and median is given as orange line.
Chapter 5  
Conclusions and Future Directions

Contemporary ML algorithms such as deep learning perform well, but only when training data is available in significant quantities and high-quality with no missing information. This work proposed incorporating domain knowledge into deep learning algorithms for SAS to overcome the lack of abundant, high-quality training data. This task was accomplished by applying domain knowledge to the network architecture or the optimization problem’s loss function used for network training. Furthermore, this work demonstrated that inclusion of such knowledge leads to improvements in ML performance without the need to collect or label more data and, thus, make better use of existing training datasets.

In Chapter 1, I provided a brief overview of the SAS imaging technique and its history, along with an introduction to domain enriched deep learning. Together, these topics motivate the use of integrating domain knowledge into the SAS machine learning pipeline: (1) to exploit problem structure that is not represented explicitly in X-Y labeled training pairs, and (2) yield competitive machine learning performance when labeled training data is sparse which is common for remote sensing applications like SAS. Regarding (2), the data used in Chapter 2 is about 50,000 images in size, the data used in Chapter 3 is about 100 images in size, and the data used in Chapter 4 is about 500 images in size; for reference, the ImageNet dataset is about 1.2 million images in size. Finally, although this dissertation focuses on SAS, the ideas presented here likely apply to other coherent imaging modalities such as side-scan-sonar, synthetic aperture radar, ultrasound, and radio astronomy.

5.1 SAS Object Recognition

In Chapter 2, a SAS ATR algorithm was developed that exhibited improved performance over state-of-the-art methods by integrating domain knowledge of SAS images previously overlooked by existing methods. The formulation jointly learns image enhancement and
target localization to improve the downstream image classification task. The proposed method was compared to several state-of-the-art techniques, including two recent deep-learning methods. The proposed method outperforms existing methods especially when low amounts of labeled training data are available. Furthermore, in images with low signal-to-noise ratio such as in close to the sonar (i.e., not in the main beam) or at far range (i.e., spreading loss) the proposed method outperforms existing methods by a significant margin. The ability to outperform existing methods in low labeled training data regimes is especially important for SAS as obtaining abundant labeled training data may be politically or financially infeasible to accomplish in a short time period (note, the dataset used for this work was collected over ten years and contained about 50,000 images, a far cry from the 1.2 million images available in ImageNet). Such an attribute allows one to collect a little data for a new area of interest and quickly integrate it into the classifier in a manner yielding good expected performance. Concluding, a recently proposed pruning technique was used to show that the number of free parameters in the propose network can be reduced in half and still achieve competitive performance, thus demonstrating the feasibility of real-time deployment.

5.2 SAS Semantic Image Segmentation

In Chapter 3, a domain enriched deep neural network is proposed for SAS image segmentation utilizing the unique $k$-space properties afforded by a complex-valued SAS SLC image. By examining a sequence of images $k$-space filtered over a set of consecutive look angles, a temporal sequence, in addition to spatial features, was able to be extracted from the seafloor. Together, the sequence and features were found useful in discriminating certain types of seafloor texture. The temporal sequence was modeled using a 2D convolution LSTM model. Moreover, the problem was posed as a $k$-space filtering procedure in a differentiable form, so the filters are learned or fine-tuned during the training procedure, and the entire network is trainable in a simple, end-to-end fashion.

SOTA results of the proposed network were shown on a real-world, weakly-labeled, and highly class-imbalanced SAS dataset and compared the results with several recent SAS-specific segmentation methods. The results showed the best overall MPA, and the proposed model especially outperformed the comparison methods in discriminating challenging classes like shadow and dark sand. These good results were attributed to the incorporation of complex-valued pixel information – discarded in current SAS segmentation schemes – through a domain enriched learnable “filter module” utilizing
-space filtering to generate a multi-look-sequence producing temporal and aspect-dependent features which are productively modeled using a 2D convolutional LSTM. The domain-specific utility of MLSP-Net was demonstrated through an ablation study showing the best performance when MLSP-Net was allowed to filter in $k$-space as part of the learning procedure.

I hope this work motivates researchers to consider the complex-valued SAS image (i.e. SLC) as input to deep learning algorithms in the future. Traditionally, phase information has been discarded as most SAS machine learning algorithms focus their efforts exclusively on magnitude imagery. The results here demonstrate: (1) improvements in image segmentation performance by exploiting the complex-valued nature (i.e., phase) of the SAS SLC, (2) a network design by which phase information is handled using traditional signal processing techniques in a differentiable manner, and (3) reduced algorithm complexity by enabling end-to-end training thus forgoing the need to pre-process the SLC into magnitude imagery before network input.

Future research in this area includes generalizing the MLSP to make the number of filters in the sequence a learnable parameter and to extend the filter specification to handle low-frequency SAS systems with larger fractional bandwidth.

### 5.3 SAS Autofocus

Systematic phase errors degrade SAS image quality. However, compared with random phase errors, systematic phase errors may also introduce artificial structure into the resulting SAS imaging making it difficult for human operators to accurately interpret. In Chapter 4, I present a domain enriched deep network that I call Deep Autofocus to mitigate the aforementioned problem. The method accomplishes the task of coherently autofocusing common systemic phase errors in SAS images. The method is novel in that I don’t require hand-crafted weighting functions, designer sharpness metrics, complex optimization schemes, or dozens of iterations for convergence like many traditional autofocus methods. I also show the method works with both supervised and self-supervised objective functions. Further, I demonstrate the ability to mitigate high frequency sawtooth error (often from platform yaw misestimation) as well as sinusoidal error in addition to polynomial error. Finally, the experimental section includes comparisons against a common windowing technique used to improve convergence of traditional iterative sharpness/contrast metrics, and I additionally compare the results to the popular phase gradient autofocus (PGA) method.
Several future directions of the work should be considered:

1. A natural question following this work is to ask if it is possible to beamform the images from the raw complex time-series and navigation data (or perhaps without navigation data!) using on a single forward pass of a DNN.

2. The ability to accurately estimate systemic phase error has the potential to aid in assessing image quality, especially where current methods break down [192,193].

3. The proposed autofocus network could be integrated into a larger automatic target recognition or segmentation network whereby autofocus is automatically done in addition to image classification. In this way, the images are autofocus implicitly by the need for a downstream task to have well-focused imagery.

4. The training setup could be used to autofocus a large batch of imagery in an offline fashion using the self-supervised loss. Such a task could be useful to autofocus a database of images for characterizing residual focus error or debugging motion compensation algorithms.

5. Sometimes, a channel in the sonar becomes inactive or corrupt due to manufacturing defects or wear-and-tear. This results in the reconstructed image having distinct visual artifacts. Could a derivative work be used to replace the bad channel and remove the artifacts coherently?

Item three could not be considered for inclusion into this work because:

1. The dataset used for object recognition in Chapter 2 contains unknown imaging artifacts and unknown spectral shifts in k-space which may violate the mathematical setup described in Chapter 4. Figure 5.1 shows examples of this phenomena. Furthermore, the imagery was constructed using a code which is not available for analysis to resolve these ambiguities.

2. The dataset used for segmentation in Chapter 3 is limited in size containing about 100 images and is generally well focused. A pilot studying integrating Deep Autofocus and MLSP-Net was performed but did not yield any benefits in performance (of focus quality) likely due to both the limited number of labeled training samples available and the already good focus quality of the dataset. However, this topic likely warrants further study and should be investigated in the future as more or defocused data becomes available.
Figure 5.1: Four example images (DRC images in top row) from the dataset used in Chapter 2 along with their $k$-space power spectral densities (bottom row). Images (a) and (b) exhibit a horizontal streaking effect in the DRC image whose cause is unknown and unable to be speculated. Images (a), (b), and (d) have a vertical spectral null on the right side of the spectrum where image (c) has this null on the left side of the spectrum. We posit that this artifact may be due to the signal processing or transmit waveform used to construct the image but we do not have enough information to discern this with certainty (nor to correct it).
Bibliography


URL https://doi.org/10.7717/peerj.453


URL https://www.tensorflow.org/


Vita
Isaac D. Gerg

Education

Professional Experience
2004 - 2009  Senior Systems Engineering I. Raytheon - Intelligence and Information Systems, State College, PA.
2018  Visiting Scientist. NATO Centre for Maritime Research & Experimentation (CMRE). La Spezia, Italy.

Research Website https://www.researchgate.net/profile/Isaac-Gerg

Publications Related to Ph.D. Research