Fully-Convolutional Networks: Ship Detection via Semantic Segmentation

Ross Greer
Electrical And Computer Engineering
University of California, San Diego
regreer@eng.ucsd.edu

Vivian Meng
Electrical And Computer Engineering
University of California, San Diego
vimeng@eng.ucsd.edu

Winson Quan
Electrical And Computer Engineering
University of California, San Diego
w1quan@eng.ucsd.edu

Yangting Sun
Electrical And Computer Engineering
University of California, San Diego
yas109@eng.ucsd.edu

Yifan Xu
Electrical And Computer Engineering
University of California, San Diego
yix247@eng.ucsd.edu

Abstract

Object detection is a familiar problem in current machine learning and computer vision research. One challenge with object detection, and learning algorithms in general, is the trade-off between generalized approaches and optimized performance on specific tasks, cited as Wolpert and Macready's "no free lunch" theorem. In this project, we investigate performance of common convolutional neural network architectures applied to the problem of detecting ships in water under a variety of image conditions. In particular, we implement Shelhamer, Long, and Darrell's "Fully Convolutional Networks for Semantic Segmentation", using both AlexNet and VGGNet as internal network architectures. While using VGGnet produced reasonable results, using AlexNet failed to learn during the training process.

1 Introduction

This project is derived from the Kaggle “Airbus Ship Detection Challenge,” which motivates the problem as follows:

> *Shipping traffic is growing fast. More ships increase the chances of infractions at sea like environmentally devastating ship accidents, piracy, illegal fishing, drug trafficking, and illegal cargo movement. This has compelled many organizations, from environmental protection agencies to insurance companies and national government authorities, to have a closer watch over the open seas.*

In our project, we worked to create a model that detects all ships in satellite images as quickly as possible.

1.1 Motivation

This project can help to support the maritime industry to increase knowledge, anticipate threats, trigger alerts, and improve efficiency at sea.

2 Description of Method

2.1 Algorithm

Fully-Convolutional Networks We implement a variant of the algorithm by Long, Shelhamer, and Darrell for semantic segmentation using fully-convolutional networks (FCN). Input images are passed through a series of convolutional and pooling layers, following the architectures of AlexNet and VGGNet. At different checkpoints along these architectures, the resulting tensor has a fractional size of the original image. Long et al define three such checkpoints; one at 1/8 of the original size, one at 1/16, and one at 1/32. For each of these tensors, we can immediately apply upsampling to create a representation of the image features at decreasing resolution; that is, upsampling the smallest image to restore it to a full 256x256 image will give a blurry feature representation, while the 1/8 size image can be restored to a finer (albeit still granular) resolution. This algorithm makes use of the tradeoff between resolution and parameter specificity. The image with the lowest resolution has the greatest number of extracted features, while the image with the greatest resolution has less extracted features. These three extraction sets are fused as follows: the tensor with 1/32 size is upsampled to match the image at 1/16 size, then their weighted score representations are summed. Next, this combined tensor is upsampled to match 1/8 the original image size, scored, and summed with the score 1/8 size image. Finally, the tensor is upsampled to its original size of 256x256, and a final convolutional layer is applied to classify each pixel into its expected binary class.

2.2 Architecture

2.2.1 Convolutional Neural Network

Convolutional Neural Networks (CNN) are a special kind of multi-layer neural network, designed to recognize visual patterns directly from pixel images with minimal preprocessing. We chose to use established classification architectures AlexNet and VGGNet in order to compare their performance within the larger algorithm.

AlexNet In 2012, AlexNet significantly outperformed all the prior competitors and won the ImageNet challenge by reducing the top-5 error from 26 percent to 15.3 percent. It makes use of convolutions with filter sizes 11x11, 5x5, 3x3, max pooling, dropout, and ReLU activation layers, data augmentation, and SGD with momentum. ReLU activations are present after every convolutional and fully-connected layer.

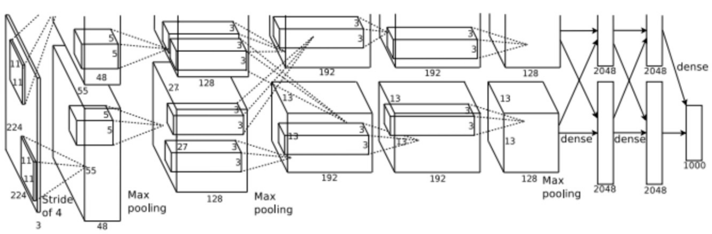

Figure 1: AlexNet Architecture.

VGGNet VGGNet was developed by Simonyan and Zisserman, achieving second place performance at the ILSVRC 2014 competition. VGGNet consists of 16 convolutional layers and is appealing because of its uniform architecture. The architecture is similar to AlexNet, however, convolutional layers use only 3x3 convolutions with a large bank of filters.

2.2.2 Fully Convolutional Networks

Fully Convolutional Networks (FCNs) are built only from locally connected layers, such as convolution, pooling, and upsampling. The network outputs a segmentation map. The architecture proposed by Long et al. reduces the number of parameters and computation time. Also, given that all

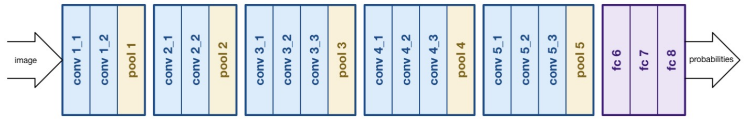

Figure 2: VGGNet Architecture.

connections are local, the network can work regardless of the original image size, without requiring any fixed number of units at any stage.

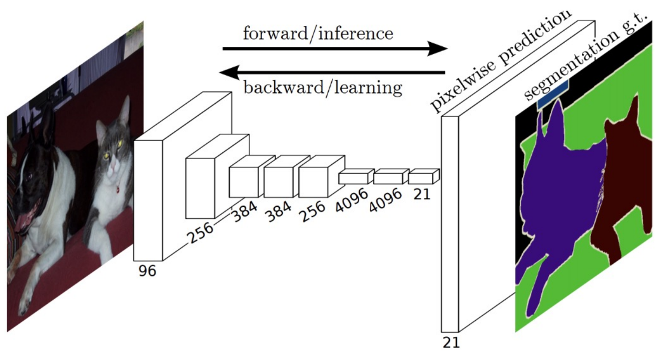

Figure 3: FCN Architecture.

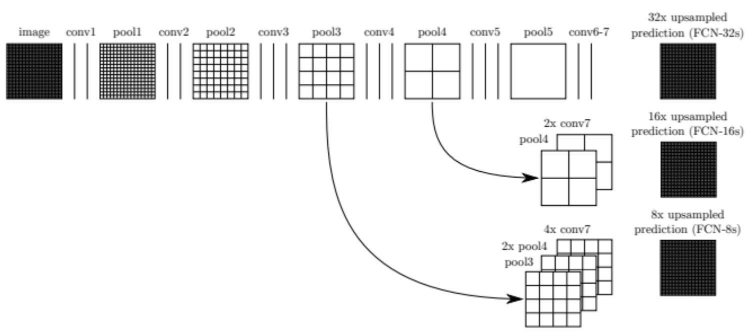

Figure 4: The above figure demonstrates the method Long et al implemented to fuse output from layers at different depths of the network. The two columns on the right represent the layer's extracted features, followed by the upsampled version which is summed to provide the final output.

2.3 Equations

Our metric for performance on the test set will be Intersection over Union (IoU) for known segments of ships on the labeled test data. Our test results on a per-image basis will be in a form of labeled pixels, with an active label corresponding to a prediction of 'ship' class, and an inactive label corresponding to a prediction of 'background'.

Intersection over Union (IoU) The equation for Intersection over Union is given in equation 1. The intersection is comprised of pixels found in both the ground truth and prediction masks. The union is comprised of all pixels found in either the ground truth or prediction masks.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} = \frac{ground\ truth \cap prediction}{ground\ truth \cup prediction} \tag{1}$$

The IoU score is calculated for each image in the testing set, then averaged for a mean IoU score for the segmentation predictions.

Weighted Cross-Entropy Loss We use a weighted cross-entropy loss function in our model to rectify the effects of having many more samples of oceanic background than images with ships.

$$L_n(W) = -w_{c_n} \log y_{c_n}(x_n, W) \tag{2}$$

In equation 2, w_{c_n} represents the weight of class c_n, and $y_{c_n}(x_n, W)$ represents the output of the network on input x_n using weights W. In our task, we assign a greater weight to class 1 (ship present), since this class is underrepresented in the dataset.

3 Implementation Details

Many implementations of fully convolution networks for semantic segmentation use publicly available pre-trained weights trained on well known architectures and ImageNet. Investigation with AlexNet using pre-trained weights showed that such parameters have difficulty recognizing the ships in our dataset, possibly because of the small size of the ships in relation to the background. In addition, networks trained on ImageNet have 1000 classes whereas we only need 2.

To decrease the time required to training, images were cropped with the ship centered in the image. This also increases the ratio of ship pixels to background pixels. For images with multiple ships, multiple cropped images were produced with each centered on a different ship. For images without ships, a single image was cropped from the center.

The AlexNet classifier was trained with cross-validation and had the training and validation accuracy evaluated every epoch (as in section 4). VGGNet trains much slower so these steps were omitted.

After the classifiers were trained, the fully convolutional layers, including deconvolution, were attached and the networks were trained with respect to a mask indicating the pixels of the ship.

In defining layers and the forward method, it is worth noting that layers which require particular learning rates for the fully-connected portion of the architecture must be separated from their preceding sequential batch in order to later define their learning rate and momentum in the model's parameters, since Pytorch handles this for an entire set of sequential-layers. In this network, we add a function "remove_classifier", which specifies if training should be performed for only the VGGNet or AlexNet architecture, or combined with the fusion, scoring, deconvolution, and convolution layers used for semantic segmentation. This abstraction allows us to take advantage of pre-trained weights for the classification networks, and to save our model and make use of transfer learning to attach learned weights to our greater context.

Following the classifier architecture, we define class FCN8s, which contains the layers of fusion in the architecture designed by Long et al. This is the portion of the code which contains upsampling via deconvolution, increasing the tensors to gradually larger and larger size, fusing them with tensors from earlier layers, until a resulting tensor of original image size is generated and its pixels are classified.

With the layers and network defined, training parameters, weighted cross-entropy loss function, and optimization method are established next. Both AlexNet and VGGNet were trained with Xavier initializations and using standard SGD with a momentum of 0.9. The fully convolutional network of VGG used the Adam optimizer. Classifiers used cross-entropy loss and the segmentation networks used weighted cross-entropy loss. The AlexNet fully convolutional network used scheduler to reduce the learning rate at a plateau, but this was unnecessary for the VGGNet.

The dataset was divided up to allow for different training, validation, and test sets. However, due to the lack of time for tuning parameters, the validation set was used as the test set and validation was not performed.

The implementation of FCN with AlexNet closely follows a implementation in CAFFE availible at (`github.com/shelhamer/fcn.berkeleyvision.org`).

4 Experimental Setting

4.1 Dataset

The dataset for this project is distributed by Kaggle, and sourced from the company Airbus. The dataset is available for download at

`https://www.kaggle.com/c/airbus-ship-detection/data`

The dataset is comprised of 192,556 satellite images, each 768x768 pixels, along with the pixel locations that comprise of the ships in the image. There are an additional 15,606 unlabeled images, provided by Kaggle for online testing.

In the images, there may be no ships, a single ship, or multiple ships. Backgrounds vary from open water to a crowded marina. Additionally, the images may have clouds, haze, or other imperfections.

The first 40,000 images were used as test data and the images in the range of 40,000 to 160,000 were used as training data.

4.2 Training Parameters

For AlexNet, the classifer used a minibatch size of 8, a learning rate of 0.005. Training was stopped early at 10 epochs as that produced the lowest traning and validation error. The VGGNet classifier was trained for 13 epochs.

4.3 Hardware Used

We trained our network on UCSD's Data Science and Machine Learning Platform (DSMLP). The network runs on the full allotment of 8 CPU cores, 32 GB of memory, and one GPU.

5 Results

5.1 Early Experiments

The robustness of the AlexNet classifier was shown with four-way cross validation, as shown in figures 5 and 6.

In figure 7, the error does not differ much between a minibatch size of 8 and 32 with the same number of epochs. However, the training is less stable with smaller minibatch sizes.

The training error of AlexNet and VGGNet are similar given the same number of epoch, but the validation error of AlexNet is much lower that that of VGGNet, as seen in figure 8. This is not unexpected because VGGNet is much deeper and hence more prone to over-fitting.

5.2 Success Cases

Our implementation of semantic segmentation using VGGNet in a fully-convolutional network shows success on cases involving ships on an open ocean background, as in Figure 5.1, and ships which blend in to the oceanic background, as in Figure 9. Our results show that we can successfully detect multiple ships in a scene. While the resolution is not as precise as the ground truth, the approximate centroids of each detected ship closely align with the ground truth centroids, and variance is contained by the ships width in any direction.

A separate success of our model is its ability to not throw false positives over hazy blocking conditions or general oceanic background, as in Figure 11. Though there is a small pattern of noise present, in general, the segmentation does not identify ambient background as ship presence.

5.3 Performance

The average Intersection over Union achieved is 0.22, averaged over images with a ship in both the ground truth and predicted segmentation mask. With respect to image-wise detection, there were

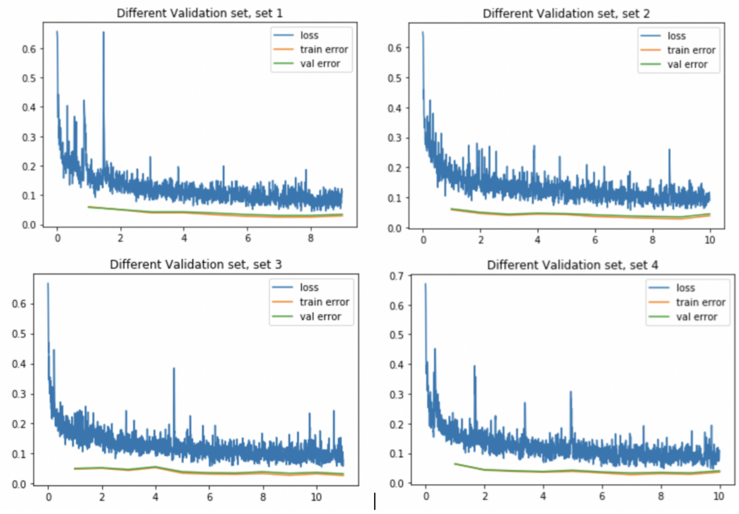

Figure 5: Validation set Error Comparisons

	training error	validation error
0	0.024482	0.030230
1	0.028355	0.035046
2	0.027317	0.032791
3	0.027177	0.033037

Figure 6: Validation set Error Comparison form

16630 true positives, 17108 true negatives, 14124 false positives, and 71 false negatives. IoU is only scored for image-wise true positive cases. The distribution of these scores is in figure 13.

5.4 Failures

The main failure of our model is in the presence of external classes of objects, such as a harbor. Our segmentation model is strong when the background is composed of ambient natural conditions, but in its current training, other manmade and geographic structures tend to be classified as ships. This is likely due to sparsity of data; while we have many training examples of open water, fog, and clouds, examples of ships are much more rare, and ships in the presence of land are even rarer. For this reason, differentiation between ships, land, and manmade structures is less finely tuned.

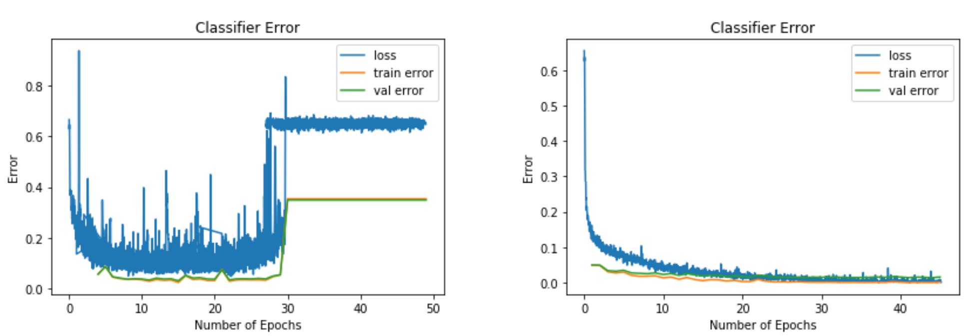

Figure 7: Classifier Error for Minibatch sizes B = 8 and B = 32

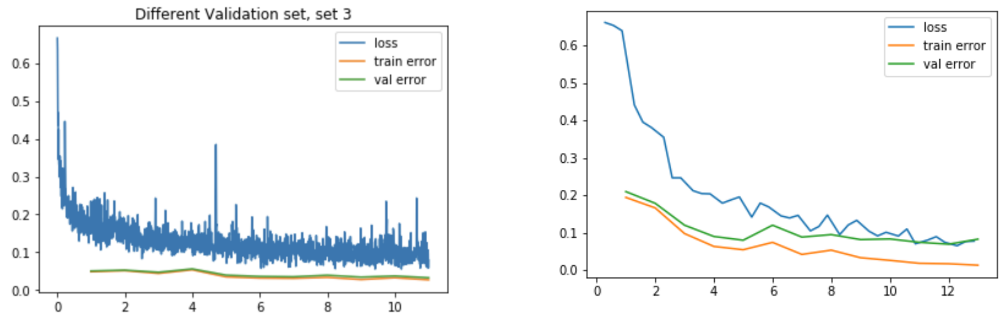

Figure 8: AlexNet and VGGNet Error

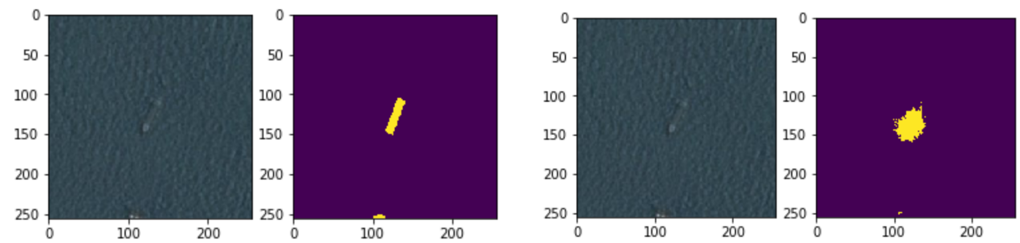

Figure 9: Ship camouflaged in water. Original images on left. Ground truth center left. Segmentation result right.

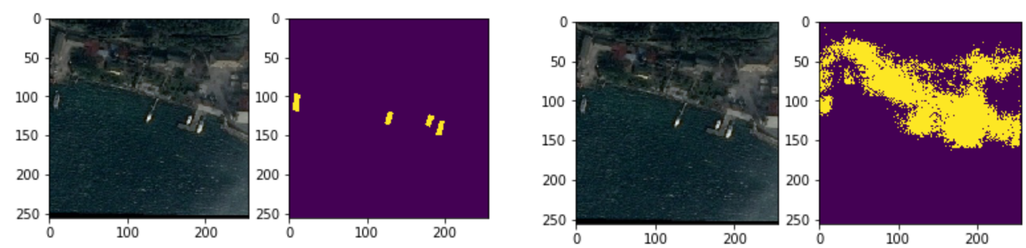

Figure 10: Ships Docked in Harbor. Original images on left. Ground truth center left. Segmentation result right.

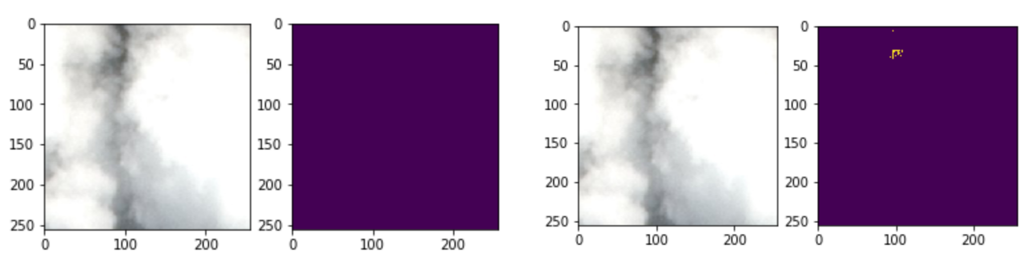

Figure 11: Scene blocked by clouds. Ground truth center left, Segmentation result right.

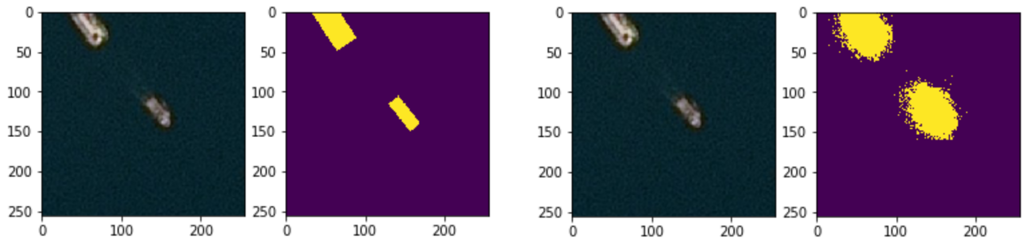

Figure 12: Two ships on open water. Ground truth center left, Segmentation result right.

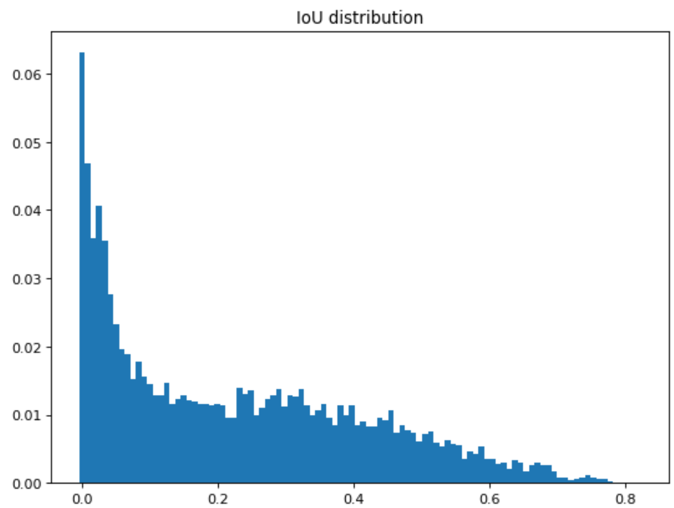

Figure 13: Distribution of intersection over union on validation set.

The implementation of AlexNet in a fully convolutional network for segmentation failed to learn during training despite having good classifier performance. This may possibly be due to shallowness of AlexNet. Unlike, VGGNet, AlexNet does not have many pooling layers from which to produce different resolution feature maps that can be upsampled and fused together.

6 Discussion

6.1 Conclusion

Fully convolutional networks presents a viable solution for semantic segmentation. However, it suffers from having to train a classifier separately in order to learn the proper feature maps, which complicates training. Unlike a classifier, the network is not completely sequential, making the implementation more complicated.

6.2 Learning

Over the course of this project, our group became familiar with two conventional CNN architectures, AlexNet and VGGNet. We were able to observe differences in both network performance and network training; for example, we noted that VGGNet seemed to train 10 times slower than AlexNet, perhaps due to the relative simplicity of the AlexNet model and parameter set.

Our group was also able to understand the problem in the context of transfer learning. In our architecture, we first trained the network as a binary classifier, leaving off upsampling and final convolution layers. Then, we attached the segmentation layers, tuning the later end of the network on the segmentation problem while transferring the learned classifier to the front.

A third principle learned over the course of this project was fusion of outputs at different layers of a network. This forced us to carefully consider the effects of stride and upsampling on tensor shape, and through this we were able to evaluate positions that we should sample from the network layers to later fuse and create a mixed-resolution output.

6.3 Challenges

Bad data There are some images in the dataset which do not correspond with the ground truth.

Really small object The ships in some of the images are really small. Sometimes we can only see bow waves which make the classification and segmentation tasks significantly harder.

Clouds or haze There are images that are completely covered by clouds, which confuses the classifier. Other photos have fog or haze camouflaging the ships with the water. Land A small amount of the images in the dataset includes only land and no water. Training time Doing two stages of training for each model takes more time to train and is more interactive. This is especially challenging when the trying to save and keep track of two sets of parameters so that they can be loaded separately.

6.4 Future Work

One approach for future experimentation would be applying progressively stronger weights to training samples containing ships in our loss function, due to the relative volume of background training samples, and creating a third class of images for non-oceanic backgrounds. By applying a greater learning weight to the ship and non-oceanic examples, this would be similar to effectively running the sample through backpropagation multiple times, compounding its learning. This is important to give balance to the spare class presence in the given training data. While we tried using weighting in our loss function to give greater value to ship training samples, we could work to further refine this weight to have optimal effect for our sparse data.

Another modification we could make to our model would be image pre-processing and data augmentation. Providing rotations of our dataset as input would create four times the current number of training points, and this would be useful as our current training appears to show difficulty in segmenting ships in the proper orientation. The ships are segmented as elliptical figures, and perhaps if trained on ships covering a span of orientations, the network could be better tuned to a ships actual location versus a particle-like range of potential positions.

A more compelling result may be possible if there was enough time or compute power to run through the whole dataset without cropping and centering the ship in the images.

7 Appendix

7.1 References

[1] Garcia-Peraza-Herrera, Luis & Li, Wenqi & Gruijthuijsen, Caspar & Devreker, Alain & Attilakos, George & Deprest, Jan & Vander Poorten, Emmanuel & Stoyanov, Danail & Vercauteren, Tom & Ourselin, Sébastien. (2016). Real-Time Segmentation of Non-Rigid Surgical Tools based on Deep Learning and Tracking. Lecture Notes in Computer Science. 10170. 10.1007/978-3-319-54057-3-8.

[2] Alex Krizhevsky, Ilya Sutskever, & Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12), F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), Vol. 1. Curran Associates Inc., USA, 1097-1105.

[3] Evan Shelhamer, Jonathan Long, & Trevor Darrell. 2017. Fully Convolutional Networks for Semantic Segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 39, 4 (April 2017), 640-651. DOI: https://doi.org/10.1109/TPAMI.2016.2572683

[4] Simonyan, Karen, & Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

7.2 Github Repository Link

The project is hosted on GitLab (`https://gitlab.com/rocketwings/ece285f`).

7.3 Github Repository Readme

```
# Fully-Convolutional Networks: Ship Detection via Semantic Segmentation

## Description
Implementation of fully convolutional networks for semantic segmentation for the Kaggle Airbus

## Requirements
Install package 'imageio' as follows:
$ pip install --user imageio

## Code organization
demo.ipynb -- Run a demo of our code (reproduce IoU scores of our report)
vggnet/VggNet-classifier.ipynb -- Train the VGGNet classifier
vggnet/FCN_Training.ipynb -- Run the fine-tuning training for the fully convolutional network
vggnet/fcn_training.pth -- Parameters for VGGNet FCN
alex/Train_Alex_FCN.ipynb -- Failed AlexNet implementation
vgg2/ -- Failed VGGNet implementation
util/ASDC_loader.py -- module for interfacing with the dataset
util/ModelUtils.py -- misc utilities
```